



PRIVACYSHIELD: Relaying BLE Beacons to Counter Unsolicited Tracking

Florian Hofhammer
EPFL

Daniele Antonioli
EURECOM

Mathias Payer
EPFL

Abstract

Offline finding networks such as Apple’s Find My, Google’s Find My Device, or Samsung’s SmartThings Find are frequently abused to stalk unsuspecting victims. These networks allow users to attach small, cheap tags to items to locate them if they are lost. The tags announce their presence via Bluetooth Low Energy (BLE) beacons, and nearby Internet-connected devices such as smartphones report their location to the finding network. However, the low price and easy-to-hide footprint of offline finding tags makes them appealing to malicious actors, who place tags on their unwitting victims. Nearby devices or even the victim’s own device then unknowingly report the victim’s location to the stalker.

We analyze the anti-stalking measures put in place by offline finding networks with a focus on Apple’s Find My and Google’s Find My Device. We show how malicious actors can bypass those measures and propose PRIVACYSHIELD, a novel relay network protecting stalking victims. Our network takes advantage of the fact that offline finding BLE beacons are unauthenticated and can be relayed to arbitrary locations. Relayed beacons cause third-party devices to report incorrect locations to the finding network, obfuscating the victim’s location. We demonstrate PRIVACYSHIELD’s effectiveness in masking a tag’s location, and show the robustness of the system against attempts to thwart its usage. Then, we suggest practical recommendations for offline finding network providers to improve stalking protection.

1 Introduction

Since Apple’s reveal of the Find My network in 2019 with the introduction of AirTags, *offline finding* networks have surged in popularity. Google’s push for their own Find My Device network in early 2024 [29], and the shared efforts with Apple for standardization of the underlying protocols [31, 50] are further fueling the deployment of these networks. Across providers such as Apple, Google, Samsung, and Tile, these networks share the same design principles. Customers attach

so-called *tags* to their personal belongings, and these tags regularly broadcast an identifier (i.e., a beacon) via *Bluetooth Low Energy (BLE)*. The beacons are received by nearby devices such as smartphones, which upload them to a server together with their (encrypted) location. To locate a lost object via a tag, a user requests the tag’s location using the offline finding app on their smartphone, which retrieves relevant location reports from the server. Tags do neither require an active Internet connection nor precise location capabilities. Instead, they piggy-back on location capabilities of nearby devices. Consequently, offline finding tags are available at a low price point, are small in size and require little power.

These properties make such tags a popular choice for stalking [12, 24, 25, 34]. Attackers hide the small, hard-to-detect tags among their victims’ belongings, such as in a purse, or the victim’s vehicle. Then, they track their victim through the network’s location reports. The fundamental design of these offline finding networks enables this malicious abuse, creating a tension between the utility of the system and the privacy of its users. To address this threat, companies, such as Apple, implement *anti-tracking* measures. For example, a smartphone displays alerts when an unknown tag is following its user over an extended period of time [4, 5].

We first analyze and systematize the anti-tracking measures currently deployed as part of the arms race between tag manufacturers and attackers. We argue that these countermeasures are insufficient because they attempt to *patch existing systems* instead of *addressing deficiencies in the core design*. We demonstrate how an attacker can circumvent these measures with low effort using (modified) commercial or custom tags. For example, disabling the AirTag builtin speaker makes the tag harder to locate for victims [34], while a custom tag allows circumventing tracking detection algorithms, e.g., by controlling the beacon rotation [27].

Based on this analysis, we conclude that offline finding providers so far have failed to sufficiently protect victims from stalking through their networks. As a tool for users to protect themselves, we propose PRIVACYSHIELD, a novel relay network that enables victims to hide their location from

an attacker. PRIVACYSHIELD leverages the fact that the tag’s BLE beacons are *by design* unauthenticated and relays them to arbitrary physical locations to obfuscate the victim’s location from the attacker’s perspective.

Our relay design is *generic*, as it applies to any offline finding network. It is *robust* against disruption, as its use cannot be prevented by the attacker and even by network and offline finding providers. Furthermore, it is *selective and functionality-preserving*, as only tags in BLE range of a relay client have their location reporting capabilities degraded. The attacker’s tags cannot be reliably located, as they stay in close proximity to the victim. Unrelated third-party tags can be reliably located again as soon as they leave the BLE range of the protected victim, as their broadcast identifier is then not relayed anymore. Legitimate tracking of lost items is thus only temporarily degraded.

Our prototype implementation of PRIVACYSHIELD is based on ESP32 microcontrollers to relay beacons. The BLE advertisements are submitted to the relay network via a smartphone application. Consequently, deploying PRIVACYSHIELD is cheap and requires no specialized hard- or firmware from the end user, installing our smartphone application suffices. Our prototype targets Apple’s AirTags due to their ubiquity. We evaluate PRIVACYSHIELD’s effectiveness empirically, showing that it can effectively mask the location of stalked victims. We demonstrate this through experiments across varying locations and numbers of devices. This setup models a real-world deployment of PRIVACYSHIELD.

PRIVACYSHIELD is user-centric, empowering victims to defend themselves without collaborating with large companies, some of which routinely collect and use user data for tracking and advertisement purposes, including their location [22]. Yet, network operators should improve the situation, and we use our observations to propose additional measures that network operators could implement to increase the privacy of their networks’ participants.

Our contributions are:

1. An in-depth analysis of the role of BLE beacons in offline finding networks, a systematization of current tracking prevention mechanisms and their shortcomings,
2. PRIVACYSHIELD, a novel and effective anti-stalking mechanism built on a defensive relay network. PRIVACYSHIELD is complementary to existing anti-stalking techniques and does not require the collaboration of the offline finding network providers,
3. An empirical demonstration of PRIVACYSHIELD’s effectiveness. Our open-source prototype is available at <https://doi.org/10.5281/zenodo.17964520> and <https://github.com/HexHive/privacyshield>.

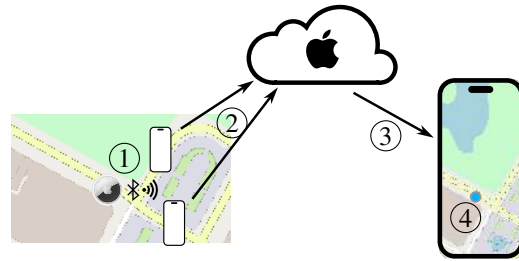


Figure 1: Apple’s Find My offline finding network workflow.

2 Offline Finding Networks

Offline finding networks rely on low-cost, battery-operated, small, and portable devices, referred to as *tags*. A tag emits Bluetooth Low Energy (BLE) beacons and is not connected to the Internet. Third-party devices, such as smartphones, detect the tags via their BLE beacons and report their location to an Internet-connected backend server.

First introduced by Tile in 2013 [46], this technology was adopted by industry giants such as Apple, Google, and Samsung [7, 21, 40]. While implementation details between Tile, Apple’s Find My, Google’s Find My Device and Samsung’s SmartThings Find differ, these systems share the same underlying principles. Figure 1 provides an overview of the functionality on the example of Apple’s Find My.

① A tag broadcasts BLE beacons (i.e., BLE advertisements) containing an identifier. These identifiers are regularly rotated to prevent long-term tracking of a user by monitoring the shared radio medium for the user’s beacon. ② Whenever the beacon is picked up by a device with localization capabilities participating in the finding network, the device (finder) sends its location together with a tag identifier to the finding network servers. ③ The tag owner can retrieve location reports for her tags and ④ determine its location.

To fulfill small physical size and longevity requirements, offline finding networks leverage *connection-less* BLE advertisements (beacons) to locate a tag. The Bluetooth Core Specification [10] also specifies a *connection-oriented* client-server architecture for BLE. However, BLE connections are not adequate for offline finding networks as they consume additional energy, provide extra latency, and have limited concurrency in comparison to broadcast advertisements. For example, the latest Android version sets an upper bound of 16 concurrent BLE links [3]. Hence, connection-based communication is only used in specific scenarios, such as tag provisioning.

The location reports submitted to a network’s servers are typically end-to-end-encrypted, i.e., encrypted by the finder and only decryptable by the tag’s owner. In particular, Apple’s and Google’s protocols leverage end-to-end encryption as they only upload encrypted location reports to their respective servers [6, 21]. In contrast, Samsung’s location reports are not end-to-end-encrypted but transmitted in the clear [13]. As a

result, the Samsung backend can see a tag’s location.

Offline finding can be implemented at the operating system level or via a mobile app. For Apple’s Find My and Google’s Find My Device networks, the location reporting is managed by dedicated iOS and Android services and daemons [6,21,23]. Similarly, Samsung integrates this functionality directly into their Android distribution, hence not requiring a dedicated application for a device to participate in the SmartThings Find network [40,49]. Instead, Tile uses a dedicated mobile application to detect beacons and communicate with its backend [37,43].

An offline finding network enables locating not only tags but also smartphones, laptops, wearables, and other compatible devices. For example, Apple’s iPhones emit offline finding advertisements when the battery level is too low to keep the main application processor running but when capacity is still high enough to power the lightweight Bluetooth chip.

The low price point and high accuracy of offline finding tags such as Apple’s AirTags quickly led to an abuse of the technology in the form of stalking [12,24,25,34]. This problem forced the industry to respond, which first caused Apple to introduce anti-stalking features in their Find My system and then prompted an industry-wide response in the form of a standardization proposal for anti-tracking measures brought forward jointly by Apple and Google [4,5,31].

3 Privacy Risks in Finding Networks

In this section, we define our threat model, detail how existing anti-stalking measures are retrofitted to offline finding networks, and describe how attackers can bypass them.

3.1 Threat Model

Attacker. An attacker is a malicious actor in an offline finding network who tries to track other people via offline finding tags. The attacker can arbitrarily place tags on other people, e.g., by dropping them into another person’s open pocket or mounting them in a hidden place on the target’s car. The attacker can also modify the tag’s hardware and firmware to reduce the risk of detection. Such modifications include, among others, disabling a tag’s builtin speaker so that the tag cannot be located via acoustic clues [34], changing the firmware of the tag, or building a custom attacker tag. The latter can be achieved for example through the OpenHaystack project [27], which implements tracking capabilities on top of Apple’s Find My network by creating “fake” AirTags.

Victim. A victim is getting tracked by an attacker via offline finding tags. The victim can leverage the tracking detection mechanisms built into mobile operating systems such as iOS and Android [50] or dedicated applications such as AirGuard [26] to detect tracking attempts. However, the victim

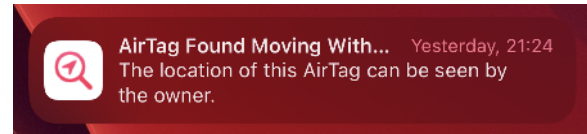


Figure 2: An iPhone tracking alert shown after an AirTag was carried with the iPhone over a distance of approximately 3 kilometers and a duration of approximately 40 minutes.

does not have physical access to the tracking device as the attacker may have disabled hardware features such as speakers required for locating the tag. The tag placed on the victim emits BLE beacons to advertise its presence. These advertisements are then picked up by either the victim’s phone or by third-party devices and the tag location is sent to the finding network backend. Consequently, the attacker can determine the victim’s *reported location* via the offline finding network. The attack succeeds if the attacker can successfully determine the *actual physical location* of the victim.

3.2 Limitations of Anti-Stalking Measures

In the years following Apple’s introduction of Find My in 2019 and with the corresponding increase in the popularity of offline finding networks, both prior research [1,26,28,35,36,44,45] as well as real-world stalking incidents [4,12,24,25,34,48] have highlighted the privacy concerns around the technology.

Under the threat model described in Section 3.1, unsolicited tracking of a victim is a real risk. For this reason, offline finding network operators either implement or propose standards for notifying users when an unknown tag is following them [4,5,31]. Notifications are displayed on a victim device if a tag is detected nearby for a prolonged time and across multiple physical locations (cf. Figure 2). Moreover, tags are equipped with speakers that play a ringing noise in regular intervals when away from their owner, which can also be manually triggered to allow locating and disabling the tag (by, e.g., removing the battery).

Next, we detail how these anti-stalking measures integrate into the offline finding workflow. We describe the intended workflow’s *four phases* in Sections 3.2.1 to 3.2.4: provisioning, losing, encountering, and locating a tag. Based on our analysis of the system, we define *four prerequisites PR1 to PR4* for triggering tracking alerts. Violating any one of them leads to tracking alerts being suppressed in both current and proposed future unwanted tracking detection implementations. Attacks A1 to A4 describe ways an attacker can abuse this insight. This observation motivates our development of PRIVACYSHIELD as described in Section 4 to fill this gap.

3.2.1 Provisioning a Tag

At first, the device to be tracked via an offline finding network needs to be provisioned to participate in the network. Both tags and other devices, such as smartphones, can be tracked via offline finding networks. During provisioning, a BLE connection is established between a tag and the owner device to install key material on the tag. When provisioning a non-tag device, the key material is generated on the corresponding device. The key material serves as the seed for a key derivation function (KDF) generating rotating ephemeral asymmetric keys. The ephemeral public key is used as the tag identifier later on and is included in the BLE beacon.

Apple and Google leverage Elliptic Curve Cryptography (ECC) to generate the asymmetric keys, as it provides shorter keys and better performance than RSA. Short keys are crucial for an offline finding network based on BLE beacons, since the payload size of standard BLE advertisements is limited to only 37 bytes, including link-layer address and other mandatory fields that cannot readily be used for payload data. Despite using ECC to encrypt location reports on the phone, the BLE beacons emitted by the tags are not cryptographically authenticated.

Differentiating tags and non-tag devices is crucial from a privacy perspective, as previous work has shown that offline finding beacons emitted by iPhones do not trigger tracking alerts in Apple's Find My network [36]. While Apple could show tracking alerts for non-AirTag devices as well, such a feature has not (yet) been implemented. This design decision was likely taken based on the assumption that the comparatively high cost and form factor of an iPhone make it unsuitable for tracking attempts.

PR1: Dedicated tag

The tracking device needs to be a tag, like an Apple AirTag.

The BLE beacons emitted by AirTags as well as iPhones follow the same structure and differ in a single bit in the broadcast advertisement. This single-bit difference in the beacon format causes tracking alerts to be suppressed.

A1: Tracking via non-tag devices

An attacker using a custom tag can impersonate an iPhone instead of an AirTag by changing the BLE beacon payload.

We successfully tested [attack A1](#) using an attacker iPhone and a custom BLE tag implemented using an ESP32 development board [19]. For the attack, we recorded an AirTag's advertisements, changed the device type flag in the BLE beacon to represent a non-tag device, and re-emitted the advertisements. The attacker can still locate the custom tag via Apple's

Find My app, since the location reports sent to Apple's servers do not take the device type into account.

3.2.2 Losing a Tag

To announce their presence, a tag emits BLE beacons with an ephemeral public key. The public key and the tag's BLE address rotate regularly. Without rotation, a tag could be identified via its advertised payload, allowing an attacker to track a victim by eavesdropping on the tag's advertisements. Due to key rotation, however, a tag appears to an observer as two distinct devices when sampling the BLE advertisements before and after key rotation.

Offline finding tags adjust the identifier rotation frequency based on whether a tag is with its owner ("near-owner mode") or not ("separated mode"). While in near-owner mode, a tag may either not emit beacons or use a rotation frequency of around 15 minutes to prevent reliable tracking of the tag's owner through the broadcast identifier. When separated from its owner, a tag changes its identifier every 24 hours to permit detection for anti-tracking alerts. According to Apple's and Google's joint standard proposal, a tag transitions from near-owner to separated mode at least 30 minutes after the last connection to the owner's device [31].

PR2: Separated mode

A tag needs to be in separated mode, away from its owner.

The unwanted tracking standard proposal by Apple and Google does not dictate any further details of the offline finding protocol implementation apart from advertisement frequency and payload format [31]. Implementers of the standard define any payload contents. At present, neither Apple's nor Google's offline finding system respects the advertisement message format defined in their shared proposed standard for unwanted tracking protection. However, the two manufacturers are expected to adapt their advertisement payloads in the near future [6,21,31]. We describe the current implementation details for Apple's Find My and Google's Find My Device networks in [Table 1](#).

3.2.3 Encountering a Tag

If a tag in separated mode is detected over an extended period of time by a device, the device displays a tracking alert to its owner (cf. [Figure 2](#)) [6,21].

PR3: Continuous detection

A tag must be detected for an extended time frame, i.e., 30 minutes [36] to multiple hours [45].

Table 1: Implementation details for the offline finding workflow described in Sections 3.2.1 to 3.2.4 [6, 21, 30, 31].

Feature	Apple	Google
BLE advertisement types	legacy advertisements only	legacy and extended advertisements
Provisioned key material	NIST P-224 curve key pair, 32 byte symmetric key	32 byte Ephemeral Identity Key (EIK)
Elliptic curves for broadcast key material	NIST P-224	SECP160R1 (legacy advertisement), SECP256R1 (extended advertisement)
Broadcast key material	NIST P-224 public key	x coordinate of SECP160R1 or SECP256R1 public key, respectively
Broadcast key derivation function (KDF)	Based on ANSI X.963 KDF (cf. Section A.1)	Based on encryption of a counter under AES-ECB-256 (cf. Section A.2)
KDF for symmetric key material	ANSI X.963 with SHA-256	HKDF-SHA256
Authenticated encryption algorithm	AES-128-GCM	AES-256-EAX
Key rotation frequency (near-owner mode)	every 15min	every 1024s + small random backoff (approx. 17min)
BLE address rotation frequency (near-owner mode)	every 15min (derived from key)	every 1024s + small random backoff (approx. 17min)
Key rotation frequency (separated mode)	every 24h	every 1024s + small random backoff (approx. 17min)
BLE address rotation frequency (separated mode)	every 24h (derived from key)	every 24h
Uploaded location report identifier	SHA-256 hash of public key	Upper 80 or 128 bits of public key (SECP160R1/SECP256R1, respectively)
Location report aggregation	Yes, to “generate a more precise location” [6]	Yes, to prevent a single device leaking its location

Apple’s AirTags start emitting BLE beacons with their corresponding public key material as soon as they lose connection to the owner’s device. An attacker then has a time window of approximately 30 minutes to track a victim without them receiving a tracking notification (prerequisite PR3). Even this limited time frame is often enough for an attacker. For example, slipping an AirTag into a victim’s pocket at a shared workplace could already provide an attacker with information about the victim’s commute route and consequently the victim’s home address.

A2: Tracking via early separated tags

An attacker can (ab)use the time window during which the victim is not alerted of the tracking attempt.

Furthermore, the tag needs to physically follow the victim, i.e., no alert is displayed if the tag and victim remain stationary. Such a situation could occur, e.g., in an office building where a coworker’s tag might be located close to the victim over the course of the workday in a benign manner. While the Apple/Google unwanted tracking standard proposal does not detail this requirement, the document defines tracking alerts as “notifying the user of the presence of an unrecognized accessory that may be traveling with them over time” [31].

PR4: Non-stationary detection

A tag must move with the victim.

After having verified that non-tag device advertisements evade tracking detection, we verify how long and over what distance a separated tag needs to follow a victim to trigger a tracking alert. During our tests, carrying an AirTag over a distance of approximately three kilometers and over a duration of 30 minutes was required to trigger a tracking alert. The location reports for a stationary tag had outliers that differed up to 500 meters from the actual tag location. These results suggest that Apple chose a larger distance to consider a tag

non-stationary to compensate for such outliers in the reported locations.

With these prerequisites and results in mind, we describe yet another tracking attack. The attacker can program a custom tag, using tools such as OpenHaystack [27], to rotate its identifier more frequently than expected to avoid being detected. The custom tag is still required to adhere to the advertisement payload format imposed by Apple to have location reports sent to Apple’s servers. However, the custom tag would be interpreted as multiple distinct AirTags due to the short key rotation interval, thus violating prerequisites PR3 and PR4.

A3: Tracking via frequent identifier rotation

A custom tag frequently rotating the tag identifier evades detection as a singular following tag.

From a technical perspective, a device first receives a tag’s advertisement with public key material p (① in Figure 3), Apple’s and Google’s offline finding implementations then generate an ephemeral key on the corresponding elliptic curves (② in Figure 3). They leverage the ephemeral key to derive a shared secret between the owner of the tag and the finder via Elliptic Curve Diffie-Hellman (ECDH) (③ in Figure 3).

The finder device derives a symmetric key from the computed ECDH secret and uses this key with an authenticated encryption mechanism to encrypt its current location. Note that only the resulting location report is authenticated, the BLE beacon payload itself is not. The encrypted location report together with its authentication tag, the finder’s ephemeral public key from step ② and an identifier for the broadcast public key from step ① is finally uploaded to the offline finding network servers (④ in Figure 3).

3.2.4 Locating a Tag

Locating a tag can be seen from two perspectives: the (far away) tag owner’s or a tracking victim’s perspective after

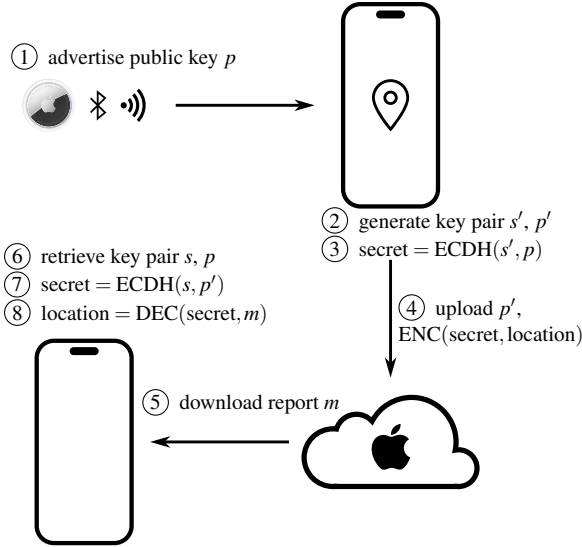


Figure 3: Overview over the cryptographic operations involved in location reporting. s and s' denote secret keys, p and p' the corresponding public keys of an elliptic curve key pair. The figure uses the example of Apple’s Find My but the operations in Google’s Find My Device network are analogous as described in Sections 3.2.3 and 3.2.4.

receiving a tracking alert.

If a victim receives a tracking alert on their mobile device as implemented in iOS and Android [50], disabling the tag and therefore cutting the attacker’s access to up-to-date location information requires physical access to the tag. Existing tags do not support remote disabling and the proposed standards do not include such a feature for future offline finding tag implementations [31].

A4: Tracking by hiding the tag

An attacker can bypass anti-tracking measures by making the tag hard to locate, e.g., by disabling built-in speakers that are used for locating a tag in off-the-shelf devices [34].

A tag owner’s device knows the provisioned key material and can follow the tag’s key rotation (cf. Section 3.2.2). It can then calculate the same public key identifiers that third-party devices encountering a tag sent to the finding network’s servers. The device requests location reports for the most recent public keys (5) in Figure 3) from the server. Based on the finder’s uploaded ephemeral public key and its own private key, the owner device derives the same ECDH secret that was used for encrypting and authenticating the location report (6) and (7) in Figure 3). Finally, after verifying the location report’s integrity and authenticity via the attached authentication tag, the owner’s device decrypts the tag’s location

(8) in Figure 3) and shows it on a map [6, 21].

Offline finding networks only take a small number of the most recent location reports into consideration for determining the associated tag’s location to be displayed on the map. Apple’s Find My shows the last reported location of a tag, whereas Google’s Find My Device implementation requires multiple location reports for a device which are aggregated before being shown to the owner [30]. This filtering and aggregation takes place on the owner’s end device, since the location reports are end-to-end-encrypted.

In light of those limitations to anti-tracking measures currently implemented by Apple and pushed into an IETF standard draft by Apple and Google [31], we deem the current measures insufficient to properly protect victims of stalking or other unsolicited tracking. We argue that these issues are inherent to the design of the currently deployed offline finding networks. In the following, we describe PRIVACYSHIELD, which reestablishes the privacy of victims of unsolicited tracking attempts by relaying BLE beacons to arbitrary locations.

4 PrivacyShield

As shown in Section 3, current anti-stalking measures in offline finding networks, notably Apple’s Find My, are insufficient to properly protect victims from unsolicited tracking. Further, previous proposals improving the privacy of users in offline finding networks through protocol changes and cryptographic measures [35, 41] have not seen any adoption in either current implementations or future standard proposals. Existing proposals require the offline finding network operator’s cooperation or significant changes to the underlying protocol.

We propose PRIVACYSHIELD as a new solution for preventing unsolicited tracking by hiding a user’s location. Unlike prior work, our solution does not rely on the network operators’ cooperation or any changes to the protocol. A victim’s location is masked by relaying a tag’s BLE advertisements to different locations, where the beacons are re-emitted. From a network participant’s perspective, this approach creates copies that are indistinguishable from the original tag. Since the metadata in BLE beacons by design does not carry location information, an observer cannot detect the relay mechanism by inspecting the relayed packets’ structure.

Wireless relay attacks have been applied to keyless entry systems for cars, payment terminals, or smart locks [17, 20, 42]. Moreover, they have been leveraged to influence the distance measurements in BLE proximity detection systems such as COVID-19 tracing applications [14, 39]. In contrast to previous offensive work, PRIVACYSHIELD employs relaying as a defensive measure instead of for attack purposes.

PRIVACYSHIELD’s design is based on our analysis of existing anti-tracking measures’ shortcomings and consultations about privacy and ethics implications with our Institutional Review Board (IRB). Our prototype aims to provide a straightforward basis for future collaborations with anti-stalking ad-

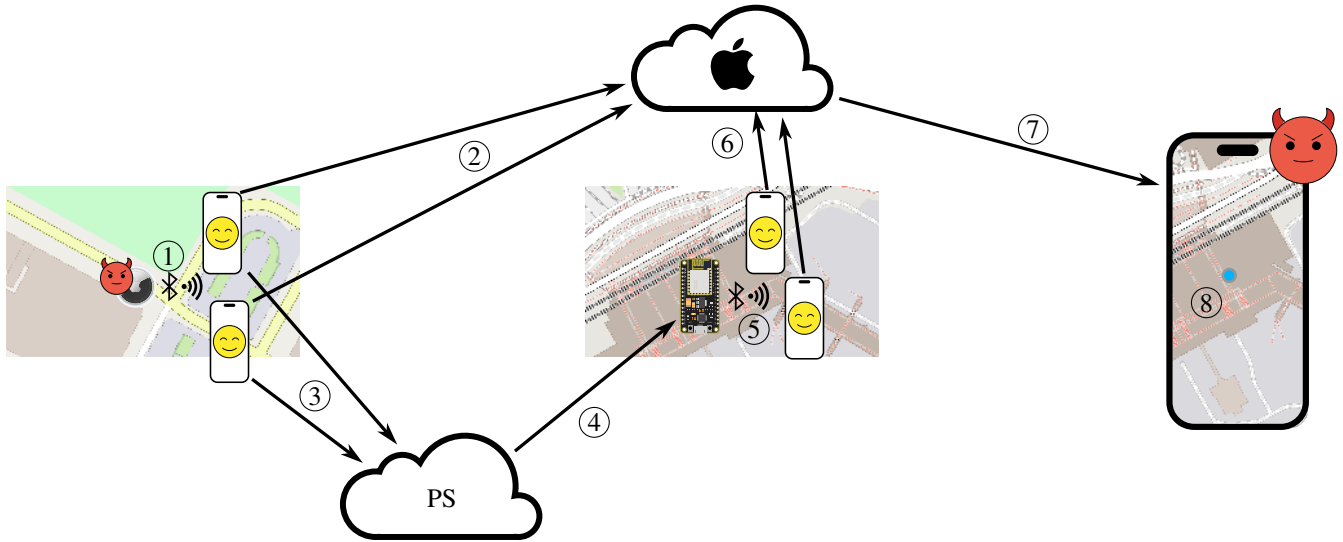


Figure 4: PRIVACYSHIELD’s functionality on the example of Apple’s Find My. An attacker’s AirTag emits its advertisements ①. Third-party devices pick up the advertisements and send their location reports to Apple’s servers ② but at the same time participate in the PRIVACYSHIELD network and transmit the encountered advertisements to a server in our network ③. A relay fetches beacons from the server ④ and re-emits them ⑤. Again, third-party devices send their location reports for these advertisements to Apple’s servers ⑥. If the relayed location reports drown the original signal, the attacker’s device downloads the masked location reports from Apple’s servers ⑦, and consequently displays the masked location to the attacker ⑧.

vocacy groups. In the following, we detail PRIVACYSHIELD’s requirements, design, and implementation.

4.1 Requirements

Based on the shortcomings of existing anti-tracking measures as described in Section 3, we identify the *five requirements R1 to R5* for PRIVACYSHIELD:

R1: Immediate protection. The system needs to take effect as soon as an attacker places a tag on a victim. Even short periods of time during which a victim is tracked (cf. *attack A2*) are not acceptable.

R2: Transparent to network operators. The system must work even if the network operator (e.g., Apple) is not cooperative.

R3: Robust to attacker interference. The system is required to be robust with regards to an attacker’s attempts to block its usage.

R4: Resilience against customized hard- and firmware. The system must be capable of operating reliably even in the presence of custom malicious tags, including a tag that rotates keys more frequently than expected to avoid being detected (cf. *attack A3*).

R5: Non-detectable by an attacker. The system must not be detectable by the attacker. Otherwise, the attacker might be prompted to take more drastic measures in both the digital and physical world.

4.2 Design

PRIVACYSHIELD’s design is based on a client-server model depicted in Figure 4. A client can be a *sniffer* and/or a *relay*.

A sniffer listens for BLE advertisements originating from offline finding tags. It then uploads the detected advertisements to a server, which stores the beacon payload (cf. ③ in Figure 4). Because sniffers by design do not adapt their behavior based on how often or for how long a tag is continuously detected nearby, they do not differentiate between legitimate and custom tags. Thus, PRIVACYSHIELD is resilient against attacks based on customized tags as demanded by *requirement R4*.

Sniffers do not make use of a “grace period” during which an offline finding advertisement is ignored, contributing to the goals of *requirement R1*. Sniffing is passive and does not require interacting with the attacker’s tag. Therefore, the attacker is neither aware of the sniffer nor can the attacker prevent a sniffer from being employed, fulfilling *requirement R3*.

A *relay* requests tag advertisements from the server. Upon reception of advertisement payloads, the relay configures its Bluetooth interface to match the beacon’s link-layer address and relays the beacon in an arbitrary location. The link-layer

BLE broadcast traffic emitted by a relay is identical to the originating tag’s advertisements.

The relayed beacons are then picked up by third-party devices which upload their location to the offline finding network operator’s servers (cf. ④, ⑤ in Figure 4). Note that the detour via third-party devices instead of generating bogus location reports and directly submitting them to the offline finding servers is necessary because the corresponding APIs are restricted to authenticated users. Consequently, submitting a suspiciously large number of location reports would likely result in dropping the received reports and potentially blocking the used accounts. Mass-creating accounts is considered infeasible due to identity verification checks put in place by Apple and Google, and we consider buying credentials for existing accounts at scale both ethically dubious and prohibitively expensive. Such an approach could therefore not reliably inject falsified location reports into the system. In contrast, the relayed advertisements are indistinguishable from the original beacons and PRIVACYSHIELD never interacts with the network operator’s servers. Hence, the relayed location reports can neither be filtered out of the data stream directly on the mobile device nor on the servers. PRIVACYSHIELD thus cannot be blocked by Apple or Google and fulfills requirement R2.

The *server* distributes advertisements to relays. Based on the push-pull-model for distributing recorded beacons, server instances can forward their received advertisement messages to other servers, increasing the relay network’s resilience and coverage. Since the number of beacons stored by a server grows depending on the number of sniffers, a server does not reply to a request from a relay station with the full list of managed beacons. Instead, the number of returned beacons is configurable to achieve *fairness* and *liveness*. Fairness ensures that beacons are relayed with uniform probability, and liveness that any submitted advertisement is relayed.

The relaying clients should be situated in locations that are highly frequented by third-party devices that can pick up the relayed locations. If a relaying client is located at a rarely frequented site, few relayed beacons are picked up by third-party devices and the victim’s location is therefore not masked. In consequence, venues such as train stations, airports, or other highly frequented public places are ideal locations for deploying relaying clients. This insight together with the fairness and liveness guarantees of a relay implementation ensures that a victim is protected from tracking attempts as required by requirement R1.

Ideally, relaying clients are geographically distributed, and relays can be selected per beacon based on geographical filters. For the functionality of location masking, a single relaying client is technically sufficient. However, providing geographically distributed and selectable relays allows to mask a victim’s location with high plausibility of the fake location.

For example, relaying beacons recorded in New York City to San Francisco immediately alerts an attacker that there

is an issue with the tag the attacker placed on the victim. On the other hand, relaying from one end of Manhattan to the other provides a victim with a safe distance between the actual location and the location reported to an attacker but also may look plausibly close and therefore not immediately alert the attacker. This behavior makes PRIVACYSHIELD’s location masking harder to detect for an attacker as intended by requirement R5.

Location masking as implemented by PRIVACYSHIELD is *probabilistic*. As described in Section 3.2.4, Apple’s Find My considers only the most recent location reports for the location of a tag displayed in the Find My application. By drowning legitimate location reports in the stream of masked location reports, PRIVACYSHIELD probabilistically hides a victim’s real location. Because a tag’s beacons can be relayed at multiple arbitrarily chosen and highly-frequented locations at the same time, we do not consider the probabilistic nature of PRIVACYSHIELD’s location masking a limitation in real-world deployments. Location aggregation as implemented by Google [30] is impacted even more by location masking through beacon relaying. In this case, it is not necessary to override *all* recent location reports to hide the real location reliably but it suffices to inject *some* masked location reports to manipulate the aggregated location.

PRIVACYSHIELD defends against the four tracking attacks mentioned in Section 3.2. It *immediately* starts relaying a beacon once it is encountered and reported by one of the sniffers. Consequently, a victim’s location is masked through relaying corresponding beacons, thwarting attack A2. Because PRIVACYSHIELD operates independently of the tag hardware, disabling a speaker as described in attack A4 or similar measures do not allow an attacker to bypass our relay methodology. PRIVACYSHIELD also counters attacks A1 and A3. It neither takes relay decisions based on when and how often an offline finding advertisement is encountered nor what type of device it originates from. It therefore protects against custom tag implementations piggy-backing on offline finding networks. By design, our system can relay offline finding beacons emitted by devices such as smartphones just as well as those originating from dedicated tags such as AirTags.

4.3 Implementation

We focus our implementation of PRIVACYSHIELD on Apple’s Find My network and the corresponding AirTags because of their ubiquity, widespread and low-cost availability, mature ecosystem, and simple identifiability through fixed byte patterns in the advertisements as depicted in Table 3 in Section B. Our implementation follows open science principles (cf. Section 11) and is available at <https://doi.org/10.5281/zenodo.17964520> and <https://github.com/HexHive/privacyshield>.

In our prototype implementation, we add our sniffing component that uploads encountered AirTag beacons to the PRI-

VACYSHIELD server as an extension to the AirGuard Android application [26]. This choice provides us with multiple advantages over an implementation from scratch. First, the AirGuard application already parses encountered beacons and stores them in a local database annotated with the corresponding device type. Consequently, we can leverage AirGuard’s higher-level interfaces to encountered beacons, easily filter for beacons originating from Apple’s AirTags, and thus quickly discard irrelevant BLE traffic. Second, Android’s BLE API provides us with information about the signal strength of any encountered beacons. We leverage this BLE advertisement package metadata combined with the advertised link-layer address to ensure the absence of interference by other devices in our experiments, since we can filter for our own devices and immediately discard BLE frames originating from devices other than ours. Through this approach, we also ensure that our implementation does not interfere with the functionality of offline finding networks for other users that may have tags in the vicinity of our experiment locations.

The relay component of our PRIVACYSHIELD prototype implementation is a custom firmware for Espressif’s ESP32 [19]. This chip combines WiFi and BLE capabilities in a small and power-efficient form factor. It is cheap and development boards cost less than 1\$. Leveraging these properties, relay stations can be installed at low cost, and with low effort in highly frequented public places in big numbers. Relays connect via WiFi to our server, retrieve beacons from the server, and emit them via their BLE interface.

Our server implementation is based on the Python Flask framework [38]. It provides a RESTful API over HTTP, with endpoints for submitting encountered advertisements and retrieving advertisements from the server. Public keys are stored in a database and returned in a round-robin manner. We consider a broadcast key valid for 24 hours after it was detected, since this is the common identifier rotation frequency in offline finding networks (cf. Section 3). This approach is an overapproximation, since a key could have already been in use for multiple hours before it is encountered. Our experimental evidence suggests that Apple’s Find My also accepts keys that have been in use for more than 24 hours.

5 Evaluation

In this section, we describe our evaluation’s ethical aspects, setup, and results.

5.1 Ethical Considerations

We conduct the experiments in an ethical and responsible manner with clearance from our Institutional Review Board (IRB). We ensure that we only store and process information from our tags and immediately discard any other potentially recorded data. Through technical measures such as filtering based on BLE link layer address, we also prevent negative

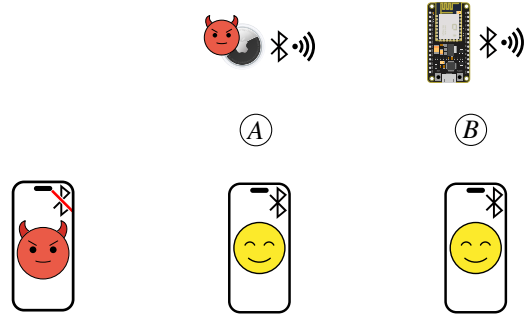


Figure 5: An overview of our experimental setup. iPhones not belonging to the attacker are co-located with the attacker’s AirTag and the relay in locations *A* and *B*, respectively. The attacker’s iPhone has Bluetooth disabled to put the AirTag in separated mode (cf. Section 3.2.2).

impact on the legitimate functionality of Apple’s Find My system during our experiments. We provide further details as requested in the Call for Papers in Section 10.

5.2 Experiment Setup

In our experiments, we focus on Apple’s Find My network. Both Apple’s pervasiveness, especially in western societies [2, 16], as well as the extensive prior research motivate this deeper analysis of anti-tracking measures in the Find My network. As highlighted in Sections 2, 3 and 4.2, our approach builds on generic offline finding network properties and the results of our experiments should therefore transfer to other networks.

We base our experiments on three variables: (i) the frequency f with which relayed beacons are emitted, (ii) the number of devices n_o that pick up the original beacons, and (iii) the number of devices n_r that pick up the relayed beacons. Intuitively, increasing f or n_r relative to n_o should increase the probability of successfully masking a tag location.

Across all our experiments, we use a single Apple AirTag to represent an attacker-controlled tag. We define two geographically distinct locations *A* and *B*, where *A* is the location of the AirTag and *B* is the location of our beacon relay. In our experiments, the distance between *A* and *B* is approximately four kilometers. We then adjust the frequency of beacon emission f at *B* as well as the number of potential beacon receivers (i.e., iPhones) at both *A* and *B*.

We use multiple iPhone 11 and 11 Pro which meet Apple’s system requirements for Find My [9]. One of the iPhones is registered with the same Apple ID as the AirTag and represents the attacker. The other iPhones are registered with a different Apple ID and represent third-party devices picking up the emitted beacons. Figure 5 depicts our setup, with locations *A* and *B* being physically separate.

We focus on three research questions:

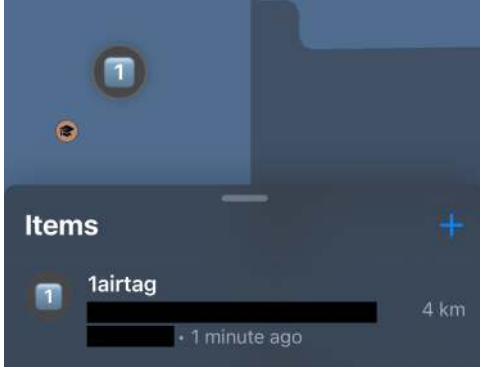


Figure 6: Tracking a relayed AirTag from an attacker’s perspective. The 4 kilometers in the screenshot are the distance to the real AirTag location. The location displayed on the map is the location of the relay.

RQ1: Feasibility. Can we successfully mask a victim’s location in Apple’s Find My network by applying PRIVACYSHIELD?

RQ2: Dependence on f . How does the frequency f at which we emit relayed advertisements relate to the success of masking a victim’s location?

RQ3: Dependence on n_o and n_r . How do variations in the numbers n_o and n_r of third-party devices at locations A and B relate to the success of masking a victim’s location?

5.3 Feasibility (RQ1)

To answer whether relaying BLE beacons to mask a victim’s location is feasible, we leverage three iPhones and an AirTag. These devices represent the attacker’s devices and third-party devices picking up the emitted beacons. Furthermore, we deploy a relay station based on our ESP32 firmware retrieving beacons from our server. The first iPhone is associated with the same Apple ID as the AirTag, and therefore constitutes our assumed attacker. The other two iPhones are logged into different Apple accounts and represent third-party devices. We deploy one of these devices at each of the locations A and B , i.e., $n_o = n_r = 1$. This setup is depicted in Figure 5.

We relay the AirTag’s BLE advertisements from location A to B and re-emit them with $f = 2\text{Hz}$, i.e., every 500ms. In this experiment, *the AirTag reliably shows up at location B when checking its location via the attacker iPhone, positively answering RQ1.* Figure 6 shows a screenshot of how the masked location is displayed to an attacker in Apple’s native Find My client.

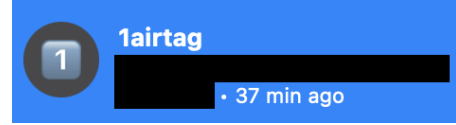


Figure 7: A location report for an AirTag in Apple’s Find My app, showing that the location does not get updated anymore.

5.4 Dependence on f (RQ2)

To determine the requirements for masking a victim’s location with PRIVACYSHIELD, we vary the frequency f at which relayed advertisements are emitted. We keep n_o and n_r equal to 1 to rule out any influence of these variables. We adjust f (relay frequency) in steps, taking into account that an AirTag typically transmits BLE advertisements every two seconds, i.e., $f = 0.5\text{Hz}$. We retrieve the location reports from Apple’s iCloud API and correlate them with the location displayed in the Find My app. If $f = 0.5\text{Hz}$, the devices at locations A and B upload the same number of reports.

According to the data retrieved from Apple’s API, an iPhone uploads a location report every five to ten minutes and attaches a confidence score to the report. While the calculation of this confidence score is not documented, we observe in our data that the score is higher if we emit beacons at a higher frequency. This behavior hints to the confidence score being adjusted based on the number of received beacons.

Apple combines location reports for an AirTag in “the owner’s app to generate a more precise location” [6]. However, we observe that relaying beacons at the same frequency as they are emitted at the original location leads to Apple not updating the location in the attacker’s Find My application (cf. Figure 7). This observation indicates that Apple does not combine contradicting location reports but discards them if none of the reported locations exhibits a higher number of associated reports, making the respective other location a clear outlier. In consequence, when PRIVACYSHIELD is employed, a victim’s current position is shown at a different location, or not shown at all, i.e., stalking is prevented in either case.

If the relay at location B emits *at least two beacons per second, i.e., $f = 2\text{Hz}$, the AirTag is reliably shown to the attacker to be at the relay location.* The observed behavior supports our earlier assumption that Find My discards contradicting location reports if one location is not reported with a stronger signal than the conflicting location. While, if we set $f = 1\text{Hz}$, i.e., one beacon per second, the location reported in the Find My app jumps back and forth between locations A and B .

5.5 Dependence on n_o, n_r (RQ3)

To determine PRIVACYSHIELD’s efficiency in a real-world scenario, we vary the numbers n_o and n_r of devices picking up the original and the relayed advertisements, respectively.

Table 2: Location masking success relative to the frequency of beacon emission f and number of nearby devices n_o and n_r . ✓: location reliably masked, −: reported location jumps back and forth between the real location and the relayed location, ✗: location is not masked, N/A: no location reported by the Find My app. All outcomes apart from ✗ are considered a success for the defender, as the real location is not reliably reported to the attacker.

Devices	$f = 5\text{Hz}$	$f = 2\text{Hz}$	$f = 1\text{Hz}$	$f = 0.5\text{Hz}$
$n_o = n_r = 1$	✓	✓	−	N/A
$n_o \simeq 70, n_r = 1$	✗	✗	✗	✗
$n_o \simeq 70, n_r = 2$	✓	✓	−	✗

In the first experiment, we bring our relay to a public location such as a crowded train station. While this makes it hard to precisely determine n_r , we estimate $n_r \simeq 70$ based on the location reports retrieved from Apple’s API. We first keep $n = 1$ and $f = 0.5\text{Hz}$, i.e., we relay beacons with the same regularity they are emitted and have only a single device register the original beacons. In this case, the AirTag’s location in Apple’s native Find My app is always to be shown at the relay location, despite a small number of location reports for its actual location being returned by the iCloud API.

In a second step, we reverse the roles and put an AirTag in a highly frequented location, i.e., $n \simeq 70$, similar to the above experiment. We relay beacons with the number of devices picking up the relayed beacons being either $n_r = 1$ or $n_r = 2$ as depicted in Table 2. We adjust the frequency from $f = 0.5\text{Hz}$ to $f = 5\text{Hz}$, i.e., transmitting BLE advertisements from one every two seconds to five per second. For each combination, we observe the reported location for of at least ten minutes.

We retrieve 325 location reports for the AirTag’s location and 18 reports for the relay’s location. Only when $n_r = 2$ and $f \geq 2\text{Hz}$ the AirTag’s reported location jumps back and forth between its actual location and the relay location. In the other cases, i.e., with $n_r = 1$ or $f < 2\text{Hz}$, the AirTag is shown at its actual position on the map. Our results indicate that *the number of devices picking up an advertisement and reporting its location is crucial for where the tag is shown on the map. But, PRIVACYSHIELD can mask the tag location even if $n_o \gg n_r$.*

Our experiments demonstrate that PRIVACYSHIELD can reliably mask a victim’s location, even if the number n_o of devices picking up the original AirTag’s advertisements is significantly larger than the number n_r of devices detecting the relayed beacons. Table 2 summarizes our observations of the behavior exhibited in Apple’s native Find My app. Notably, our capability of emitting the beacons with a higher frequency than an AirTag increases the chances of the relayed beacons getting picked up by third-party devices and therefore increases PRIVACYSHIELD’s reliability.

6 Discussion

In the following, we discuss the takeaways from our experiments and propose changes to offline finding network designs. These proposals require implementation by the network operators, and we position PRIVACYSHIELD as a stop-gap solution until a potential adoption of our proposals. Further, we describe limitations to our approach.

6.1 Takeaways

Reducing the time window for tracking alerts. As we argue in Section 3.2, 30 minutes of tracking before an alert is displayed to a user provides an attacker with sufficient time to invade a victim’s privacy. While tracking alerts could still be bypassed with custom hard- and firmware, reducing the time window until a tracking alert is raised increases the barrier for successful stalking attempts. A reduction of this time window would lead to an increased number of alerts for users, many of which could be false positives. However, offline finding providers could implement an allowlist functionality that allows a user to permanently disable alerts for a given tag. For benign tags, the alerted user and the tag’s owner could establish an out-of-band channel for communicating the tag’s rolling identifiers. Based on the identifiers, the user can then silence tracking alerts for a given tag.

Tracking alerts for all device classes. The existence of devices that do not trigger an alert is also worrisome and should be removed from the protocol. As shown by Mayberry et al. and confirmed in our experiments, masquerading a custom tag as an iPhone instead of an AirTag bypasses any tracking alerts raised [36]. A simple solution to this attack vector is to raise tracking alerts no matter the device class, i.e., not only for tags such as AirTags. However, this approach may lead to an increased number of false positive alerts.

Identification of a tag’s owner. The unwanted tracking prevention standard proposed by Apple and Google [31] proposes maintaining an ownership registry for offline finding tags accessible to law enforcement. This information could be retrieved when a tag is used for malicious purposes such as stalking. Again, custom hard- and firmware can bypass ownership registration altogether but we see this proposal as an additional hurdle for low-effort stalking attempts. Unfortunately, such an approach has privacy implications for benign tag owners, as they would not be anonymous from the network operator’s perspective anymore.

Limiting location downloads to genuine hard- and firmware. Malicious actors can deploy custom hard- and firmware to bypass stalking mitigations. To reduce the effectiveness of such custom deployments, a network operator can restrict the capability of downloading location reports to

genuine devices and client software. As an example, Apple could enforce a device attesting that is an actual iPhone running Apple’s original Find My application to download the corresponding location reports. Applications such as OpenHaystack would fail this attestation step, and could not retrieve location reports for custom tags from Apple’s servers.

Detection of relay attacks. If the location reports are end-to-end encrypted, an offline finding network operator cannot determine whether a location report originates from a legitimate tag or a relay. However, a client retrieving the full set of location reports for a given time frame can decrypt all reports, and detect discrepancies in the reported locations. For example, a tag jumping back and forth between locations that are kilometers apart in a matter of seconds as observed in our experiments in [Sections 5.4](#) and [5.5](#) showcases an unrealistic movement pattern. Here, the locating device should report the presence of suspicious patterns to the user, and provide the user with a means to review and cluster or discard reports. Current interfaces provide a user only with a single location shown on a map without information on how the displayed location was determined from the set of available reports.

As the discussion of our proposals shows, all changes to the current implementations of offline finding systems come with trade-offs. We argue that there is no silver-bullet solution to provide full privacy guarantees to users in current offline finding systems without impacting the functionality or usability of the system. We see a need for a more widespread discussion of the existing issues and potential improvements to protect users from malicious actors while avoiding detrimental effects on the effectiveness of the systems.

6.2 Limitations

Extensive prior research [[1,26,28,35,36,44,45](#)] and real-world stalking incidents [[4, 12, 24, 25, 34, 48](#)] highlight the urgent need for anti-stalking measures in offline finding networks and motivate the design of PRIVACYSHIELD. Our design has been informed by an extensive assessment by our Institutional Review Board (IRB), attesting to the societal need and ethical conformity of our system. PRIVACYSHIELD is an immediately deployable stop-gap solution until offline finding networks incorporate the needed anti-stalking measures. Our system comes with limitations and trade-offs from a technical and ethical perspective. We detail them in the following section and in [Section 10](#), respectively.

Ground-truth knowledge and statistical analysis. In our threat model, we assume an attacker with access to custom hard- and firmware for the employed tags. This capability does not affect PRIVACYSHIELD’s masking effectiveness by design. If the attacker in addition uses a custom client to retrieve and decrypt *all* location reports from the offline finding networks’ servers instead of the single location reported in

the corresponding native application, the attacker can conduct further statistical analysis to filter out relayed beacons.

For example, the attacker can frequently rotate the emitted keys via custom firmware and then overlay the locations of temporally adjacent beacons. The original location reports necessarily are in close proximity for two consecutive beacons, whereas the location reports for relayed beacons may differ in location. We argue that such analysis is not a threat to the functionality of PRIVACYSHIELD, as the relaying algorithm can be extended to relay consecutive beacons reported from the same client along a realistic trajectory. The capability of choosing realistic relay locations as described in [Section 4](#) is a first step in this direction.

Additionally, the attacker can exclude relayed beacons by filtering on the location reports’ timestamps, assuming that the oldest timestamps indicate the real location reports. However, the resolution of such timestamps as observed in Apple’s Find My network is one second, and PRIVACYSHIELD relays beacons in as little as a few milliseconds. The lower bound for this latency is mainly dictated by network latency, as processing and relaying latency in our design are negligible if a sufficiently large number of relay stations ensures prompt relaying. Hence, timestamps do not provide an attacker with a reliable means to filter out relayed reports.

PRIVACYSHIELD’s design explicitly does not consider existing knowledge about the victim. If the attacker already knows, e.g., where the victim lives, she can correlate the location reports with this ground truth information. Yet, PRIVACYSHIELD still provides a victim with capabilities for at least partial protection against the attacker. For instance, by carefully selecting the relay, the victim can make the attacker believe that she is at home while she is at the local sports center and vice versa. The attacker then does not learn new information about the victim’s regularly frequented locations.

Scaling PRIVACYSHIELD in a real-world deployment.

As highlighted in [Section 5](#), PRIVACYSHIELD does not require many relay stations to successfully mask a victim’s location. However, for counteracting attacks as outlined in the previous paragraph, a sufficiently large number of relay stations and clients submitting beacons to the system are required. The more relay stations are deployed, the more fine-grained and realistic locations can be faked to deter an attacker. With only a small number of relay stations, the victim’s location can still be reliably masked, but the faked location might be unrealistic which in turn might raise suspicion in the attacker. For example, relaying beacons from New York to San Francisco might effectively hide the victim’s actual location but might be deemed unrealistic by the attacker.

Another important angle is to raise awareness of the system. Only potential victims that know of PRIVACYSHIELD can protect themselves against stalking. We aim to advertise PRIVACYSHIELD at events in the privacy and security community to establish collaborations with anti-stalking ad-

vocacy organizations for local deployment and operation. In addition, our artifact repository contains tutorials to setup PRIVACYSHIELD in practice to enable (many) others to join our federated network. Consumer prices for single-digit quantities for an ESP32 board, a SIM module for network connectivity, as well as a small solar panel and battery for power are approximately \$15 on online shopping platforms. The simplicity and low cost of our system enables plug-and-play deployment of relay stations to improve technological accessibility.

Note that the number of relay stations is not linked to an impact on legitimate tracking, as the relays do not interfere with benign tag advertisements.

Preventing legitimate tracking. To successfully mask a victim’s location, PRIVACYSHIELD requires either the victim or a third party in their vicinity to run our client application that records tag beacons and submits them to the relay network. Consequently, legitimate tracking of items is degraded as long as the corresponding tag is in BLE range of a protected victim. The more clients participate in the network, the higher the likelihood that a benign tracking use case might be temporarily impacted by PRIVACYSHIELD’s location masking. However, as soon as the tag leaves the physical proximity of the victim again, the item can be tracked anew.

Further, this effect is independent of the number of deployed relay stations. PRIVACYSHIELD mainly aims to scale the number of relay stations to provide victims with realistic fake locations rather than to increase the number of clients running the client application. While our application is free and open source, we do not expect the general public to run it on their devices in large numbers, but rather expect only users who suspect stalking attempts to install it.

We acknowledge the resulting potential for abuse, e.g., by hiding the location of stolen items that have tags attached, and that the (even temporary) unreliability of legitimate tracking might undermine user trust in offline finding networks. Despite these issues, we argue that the fundamental design of those finding networks endangers victims of unsolicited tracking. Systems such as Apple’s Find My have provided stalkers with a powerful tool for their attacks, and PRIVACYSHIELD equips victims with a tool to fight back. These systems require extensive design changes to protect victims. Hence, we position PRIVACYSHIELD as a stop-gap solution on the quest to reliable tracking protection.

PRIVACYSHIELD *prioritizes protecting victims* of stalking attempts over potential impacts on the legitimate functionality of offline finding systems. This potential degradation in tracking capabilities is only temporary and outweighs the impact on a victim’s privacy and safety. We further elaborate on the ethical implications of our approach in [Section 10](#).

Other tracking technologies. PRIVACYSHIELD focuses on BLE-based offline finding networks such as Apple’s Find My. If an attacker uses other tracking technologies such as GPS

trackers, PRIVACYSHIELD cannot protect a victim. GPS trackers are available at a similar price point to BLE-based tags, and have also received attention in prior work [15]. Protecting victims from GPS-based stalking attempts is orthogonal to our work and requires jamming or removing the GPS tracker [33].

Vendor cooperation. PRIVACYSHIELD is designed to work without any cooperation from offline finding network operators. If network operators like Apple or Google were to cooperate, they could either redesign their protocols to improve detection of stalking attempts as outlined in related work (cf. [Section 7](#)) or directly provide victims with a means to opt out of their location being reported indirectly. For example, users could be provided with an option to opt out of location tracking via a system-wide setting on their end devices. Any encountered beacons from tags in their vicinity would then be reported to the network operator, and the network operator could discard any location reports for these tags.

However, this option is not realistic, as it generalizes into a built-in denial-of-service mechanism. We therefore see protocol-level changes as described in [Section 7](#) as a more promising approach to protect victims from stalking attempts. Note that these changes aim to detect stalking attempts reliably rather than preventing them altogether. Thus, they would not fully replace PRIVACYSHIELD’s functionality but mitigate some of the tracking alert bypass techniques outlined in [Section 3](#). Hence, approaches such as PRIVACYSHIELD that do not require vendor cooperation are the only viable stop-gap solutions to protect victims of unsolicited tracking.

7 Related Work

Weller et al. [47] analyze multiple Bluetooth tags that predate Apple’s AirTags. The authors focus on security flaws such as unauthenticated API endpoints or user information leakage through lack of or improper use of cryptographic mechanisms. They also propose an offline finding protocol dubbed *PrivateFind*, whose design is similar to the Find My Apple network described in [Section 2](#). Li et al. [32] analyze the efficiency of offline finding implementations with a focus on the advertisement broadcast frequency. They design *ElastiCast*, a scheme to improve the latency of finding a tag by adapting advertisement broadcast and scan windows.

Heinrich et al. [28] are the first to analyze the accuracy of Find My location reports, and identify security shortcomings such as insecure key storage on a tag owner’s device. The authors discovered that, despite end-to-end-encrypted location reports, Apple can correlate them and infer whether users have been in close proximity. The same authors proposed OpenHaystack [27], an offline finding implementation piggybacking on Apple’s Find My network. OpenHaystack uses custom firmware and off-the-shelf microcontrollers to emulate AirTags. AirGuard, also developed by Heinrich et al.,

is an Android app to detect nearby tags [26]. The app is motivated by privacy concerns in offline finding networks. AirGuard reliably and quickly detects nearby tags and allows a user to trigger the speaker on a tag without waiting for Apple’s or Google’s tracking detection heuristics.

Turk et al. [45] also argue that the anti-tracking measures deployed in the current implementations of offline finding networks are insufficient. They focus on the unreliability of Apple AirTags, Samsung SmartTags, Chipolo One, and Tile Sticker tags. The authors analyze those tags in terms of duration until a tracking alert is shown to a stalking victim and alert sound level of built-in speakers. They argue that this time span is too long, while the sound level is too low for reliably detecting a stalking attempt and locating the tag.

In concurrent work, Yu et al. [49] analyze the security and privacy of Samsung SmartTags. Similar to the argument we make in Section 3.2, the authors find that a custom firmware emulating a tag with rotating identifiers circumvents Samsung’s tracking detection mechanisms. The authors also mention relay attacks but do not detail them further or leverage this fact as the basis of a privacy protection mechanisms.

Mayberry et al. [35] propose Blind My, a cryptographic protocol extension to Apple’s Find My protocol to prevent stalking attempts. The protocol aims to prevent rotating keys more frequently than intended to evade tracking alerts as described in attack A3. Blind My achieves this by requiring Apple’s servers to hand out blind signatures on key material upon provisioning of a tag, and by verifying the signatures when a tag owner tries to retrieve location reports for a signed public key. This way, the server limits location report retrieval to correctly rotated key material. This cryptographic approach is orthogonal to PRIVACYSHIELD. Blind My requires Apple’s collaboration to implement the proposed protocol and update the tags, whereas PRIVACYSHIELD works independently of Apple’s support as a drop-in solution.

Eldridge et al. [18] propose new algorithms for stalking detection. Their scheme is based on multi-dealer secret sharing, which involves splitting a tag’s broadcast data into multiple shares. Based on the proposed algorithm, stalking detection is improved, as detecting a tag as an adversarial tag does not require constant detection over a prolonged period of time anymore but only the detection of a configured share of the tag’s broadcast identifiers. This work is orthogonal to PRIVACYSHIELD in two ways. First, it focuses on *stalking detection* instead of *stalking prevention*. Second, the proposed protocol changes require cooperation of network providers, whereas PRIVACYSHIELD operates on top of the existing protocols.

8 Conclusion

Offline finding networks may invade the privacy of users and enable attackers to track unsuspecting victims. We analyze the status quo of these networks and highlight holes in existing anti-stalking measures, exposing ways in which attackers can

bypass these measures with low effort. In particular, we find that the four prerequisites PR1 to PR4 that are required to send tracking notification are easy to circumvent with practical and low-cost attacks A1 to A4, like pretending to be an iPhone instead of a tag or varying the beacon frequency.

To address these relevant issues and combat stalking, we propose PRIVACYSHIELD, a privacy protection method that can thwart stalking attempts using a relay network. We leverage that BLE beacons can be relayed to trigger valid location reports in arbitrary places. Our relay network provides immediate protection (requirement R1), cannot be blocked by a network operator (requirement R2) or the attacker (requirement R3), is effective even if the attacker employs custom tags (requirement R4), and since it is passive, it is hard to detect by the adversary (requirement R5).

We empirically evaluate PRIVACYSHIELD’s effectiveness, and show that little effort is needed to effectively mask user locations at a low cost. Our experiments confirm that location masking via PRIVACYSHIELD is feasible in general (RQ1), even if the number n_o of devices picking up the original advertisements is larger than the number n_r of devices detecting the relayed beacons (RQ3). Moreover, by increasing the relay frequency f we can improve reliability (RQ2).

Finally, we propose operator-based measures to improve the privacy of their systems’ users. These measures include user interface changes such as displaying tracking alerts not only for tags but also other device classes as well as protocol changes protecting users from tracking in the first place and enabling identification of perpetrators. While a small risk of unsolicited tracking remains and can only be mitigated by extensive changes to the underlying protocol, our proposed measures in addition to PRIVACYSHIELD significantly raise the bar for an attacker to track unsuspecting victims.

9 Acknowledgements

We thank our reviewers and shepherd for their comments on our paper and for their feedback in the shepherding process.

We also thank Carmela Troncoso for her valuable initial feedback on our system’s design and its privacy aspects.

The authors used map material from OpenStreetMap under the Open Database License as well as emoji symbols from OpenMoji under the CC BY-SA 4.0 license for visualizations in the paper. The authors used Google’s Gemini LLM to help find synonyms to avoid word repetitions throughout the paper. The use of LLM-based aid was limited to this use case with simple prompts such as “synonyms for “to identify””.

This work was partially supported by the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation program (grant agreement No. 850868), SNSF PCEGP2_18697, and the French National Research Agency (ANR) under the France 2030 label with the REV (ANR-22-PECY-0009) and NF-HiSec (ANR-22-PEFT-0009) research grants.

10 Ethical Considerations

We worked closely with our Institutional Review Board (IRB) to ensure that our system and experiments adhere to ethical guidelines. We further consulted the IRB to identify and assess stakeholders affected by our work.

In this section, we discuss our experiments from an ethical perspective and describe the nine stakeholders involved based on our assessment and the consultations with the IRB. We further detail the ethical impact of PRIVACYSHIELD on the different stakeholders.

10.1 Experimental setup

Our experimental setup was submitted to and reviewed by our IRB. In this process, we laid out how we limit our experiments to our own devices by applying filters both in the physical realm (e.g., by limiting signal strength) and in the software used for our experiments (e.g., by filtering for known link-layer addresses). These measures aim to protect unaffiliated third parties from the adversary side effects of our experiments, e.g., tags showing up at unexpected locations when locating them.

While we cannot guarantee that no third-party device picks up our relayed beacons, we limit the transmission power of our relay stations to minimize interference with third-party devices. Even if nearby devices pick up our relayed beacons, we ensure that we only ever relay the beacons of our own tags and therefore do not affect the offline finding functionality of third-party tags in our experiments.

In consequence, our experiments did not cause any direct harm to third parties or indirectly harm them by disrupting offline tracking services for their own devices.

10.2 Stakeholder Analysis

In this section, we list the stakeholders affected by the deployment of PRIVACYSHIELD and describe the impact of the system on each stakeholder. Note that a real-world entity can assume multiple roles, e.g., a person can both be a stalking victim and a user of offline finding networks' legitimate tracking capabilities.

Stalking victims. Stalking victims are the primary beneficiaries of PRIVACYSHIELD. When employing PRIVACYSHIELD, e.g., by running our client application on their smartphone, stalking victims can hide their actual location from an attacker. Their location can therefore only be determined by an attacker using means outside of the realm of offline finding networks, such as physically following the victim.

Stalking perpetrators. Stalking perpetrators are negatively affected by PRIVACYSHIELD. By relaying beacons from tags

in the vicinity of a stalking victim, the perpetrator cannot reliably determine the victim's location via offline finding networks anymore. Such an attacker therefore has to resort to other means of tracking the victim, which might either be more technically involved (e.g., GPS trackers) or require more effort (e.g., physically following the victim).

Users of offline finding networks for legitimate tracking.

As described in [Section 6.2](#), a wide-spread deployment of PRIVACYSHIELD in the future could potentially negatively impact legitimate tracking capabilities of tags attached to personal belongings. However, PRIVACYSHIELD only affects legitimate tracking capabilities temporarily as long as the tag is in BLE reach of a potential victim. Its impact on legitimate tracking is therefore minimal. We discuss this aspect in further detail in [Section 10.3](#).

Users of PRIVACYSHIELD's client application.

Users who run our client application to submit beacons to the relay network on their smartphones may experience a slight increase in battery consumption and data usage. As their systems already scan for BLE advertisements in the background anyways and the network communication payload is minimal, we expect this impact to be negligible in practice.

PRIVACYSHIELD server operators.

Server operators running the PRIVACYSHIELD server that receives beacons from clients and dispatches them to relay stations may try to correlate reported beacons. More specifically, if two clients report the same beacon, the server operator can deduce that the two clients were in close proximity at the time of reporting. However, the beacons themselves do not encode location information, and the only information the server operator has about the client is its IP address. This is a technical necessity and not unique to PRIVACYSHIELD. The upstream offline finding networks' servers suffer from the same issue.

PRIVACYSHIELD relay station operators.

Relay station operators can create a log of all received beacons. These beacons do not encode location information or any other sensitive user data and we thus do not consider this a potential privacy issue.

Third parties picking up relayed beacons.

Third parties that pick up our relayed beacons may incur additional data usage by uploading location reports for these beacons to the upstream offline finding network operators' servers. This impact is equivalent to an increased number of legitimate tags in the vicinity of the affected user. As this behavior is part of the inherent design of offline finding networks, we do not consider this an additional negative impact of PRIVACYSHIELD.

Offline finding network operators. The increased number of location reports for a given beacon, i.e., both from the legitimate tag and PRIVACYSHIELD’s relay stations, may increase the load on the offline finding network operators’ servers. Similar to the previous paragraph, this impact is equivalent to an increased number of third parties picking up a legitimate tag’s beacons, e.g., by bringing a tag to a crowded area. We therefore consider this part of the offline finding networks’ design and not a negative impact of PRIVACYSHIELD.

Privacy advocates and anti-stalking organizations. Privacy advocates and anti-stalking organizations benefit from PRIVACYSHIELD as it provides them with an additional tool to protect victims of unsolicited tracking. These organizations can deploy, operate, and advertise PRIVACYSHIELD’s relay network to help protect victims in their area of influence.

10.3 Impact on Legitimate Tracking

As we highlight in [Section 6.2](#), a wide-spread deployment of PRIVACYSHIELD in the future could potentially negatively impact legitimate tracking capabilities of tags attached to personal belongings.

We acknowledge the ethical responsibility of keeping the underlying offline finding systems functional but argue that the ethical duty of protecting victims of unsolicited tracking weighs higher. We argue that it is the service providers’ responsibility to protect third-parties from stalking attempts by their users. As we showcase in [Section 3](#), the current state-of-the-art does not sufficiently address this responsibility. Until service providers introduce the necessary defenses, users can resort to measures such as PRIVACYSHIELD to protect themselves from unsolicited tracking. We position PRIVACYSHIELD as a stop-gap solution that is only required as long as offline finding network providers do not adapt the fundamental design of their systems to better protect victims of unsolicited tracking. Further, PRIVACYSHIELD only affects legitimate tracking capabilities temporarily as long as the tag is in BLE reach of a potential victim. Its impact on legitimate tracking is therefore minimal.

10.4 Legal Implications

We do not interact with Apple’s servers to inject bogus location reports ourselves. As such, PRIVACYSHIELD is not breaching any Terms of Service (ToS) agreements with Apple. Even in the case where we were subject to such agreements, the iCloud ToS, which cover the Find My system, do not refer to Find My’s BLE beacons [8]. Further, previous work piggybacking on Find My such as OpenHaystack [27] or Send My [11] is, to the best of our knowledge, not in violation of the ToS either.

11 Open Science

We fully support USENIX Security’s move towards open science principles in addition to the already existing artifact evaluation badges. In this spirit, we publish the code required to run PrivacyShield and collect the data used in our experiments at <https://doi.org/10.5281/zenodo.17964520> and <https://github.com/HexHive/privacyshield>.

The artifact consists of four main components used in our experiments:

1. The server application that receives tracking beacons and forwards them to relay stations,
2. the modified version of the AirGuard [26] Android application that registers nearby AirTags and reports them to the server,
3. the ESP32 firmware that retrieves beacons from the server and rebroadcasts them, and
4. scripts to retrieve location reports from Apple’s servers without relying only on the visual feedback given in Apple’s official Find My applications.

The corresponding README files in the artifacts’s sub-directories provide further instructions on how to run our code.

References

- [1] Hosam Alamleh, Michael Gogarty, David Ruddell, and Ali Abdullah S. AlQahtani. Securing the Invisible Thread: A Comprehensive Analysis of BLE Tracker Security in Apple AirTags and Samsung SmartTags. <http://arxiv.org/abs/2401.13584>, January 2024.
- [2] Ron Amadeo. Apple hits “all-time high” smartphone market share, takes #1 spot for 2023. <https://arstechnica.com/gadgets/2024/01/apple-hits-all-time-high-smartphone-market-share-takes-1-spot-for-2023/>, January 2024.
- [3] Android Developers. Bt_target.h – Android Code Search. https://cs.android.com/android/platform/superproject/main/+main:packages/modules/Bluetooth/system/internal_include/bt_target.h;l=272, May 2024.
- [4] Apple Inc. An update on AirTag and unwanted tracking. <https://www.apple.com/newsroom/2022/02/an-update-on-airtag-and-unwanted-tracking/>, February 2022.
- [5] Apple Inc. What to do if you get an alert that an AirTag, Find My network accessory, or set of AirPods is with you. <https://support.apple.com/en-us/HT212227>, March 2022.
- [6] Apple Inc. Apple Platform Security. Technical report, Apple Inc., May 2024. https://help.apple.com/pdf/security/en_US/apple-platform-security-guide.pdf.
- [7] Apple Inc. iCloud - Find My. <https://www.apple.com/icloud/find-my/>, 2024.
- [8] Apple Inc. Legal – iCloud. <https://www.apple.com/legal/internet-services/icloud/us-en/terms.html>, September 2024.
- [9] Apple Inc. System requirements for iCloud. <https://support.apple.com/en-vn/118308>, March 2024.
- [10] Bluetooth SIG. Bluetooth Core Specification v5.3. https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=521059, 2021.
- [11] Fabian Bräunlein. Send My: Arbitrary data transmission via Apple’s Find My network. <https://positive.security/blog/send-my>, May 2021.
- [12] Thomas Brewster. People Use Find My iPhone For Long Distance Stalking – And There’s Not Much Apple Can Do. <https://www.forbes.com/sites/thomasbrewster/2023/10/03/apple-find-my-iphone-a-bused-by-stalkers-and-traffickers/>, October 2023.
- [13] Jimmy Briggs and Christine Geeng. BLE-Doubt: Smartphone-Based Detection of Malicious Bluetooth Trackers. In *2022 IEEE Security and Privacy Workshops (SPW)*, pages 208–214, San Francisco, CA, USA, May 2022. IEEE.
- [14] Marco Casagrande, Mauro Conti, and Eleonora Losiouk. Contact Tracing Made Un-relay-able. In *Proceedings of the Eleventh ACM Conference on Data and Application Security and Privacy*, pages 221–232, Virtual Event USA, April 2021. ACM.
- [15] Rose Ceccio, Sophie Stephenson, Varun Chadha, Danny Yuxing Huang, and Rahul Chatterjee. Sneaky Spy Devices and Defective Detectors: The Ecosystem of Intimate Partner Surveillance with Covert Devices. In Joseph A. Calandrino and Carmela Troncoso, editors, *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*, pages 123–140. USENIX Association, 2023. <https://www.usenix.org/conference/usenixsecurity23/presentation/ceccio>.
- [16] Counterpoint. US Smartphone Shipments Recover 8% YoY in Q4 2023 as Apple Market Share Reaches Highest Since Q4 2020. <https://www.counterpointresearch.com/insights/us-smartphone-market-q4-2023/>, February 2024.
- [17] Saar Drimer and Steven J. Murdoch. Keep Your Enemies Close: Distance Bounding Against Smartcard Relay Attacks. In Niels Provos, editor, *Proceedings of the 16th USENIX Security Symposium, Boston, MA, USA, August 6-10, 2007*. USENIX Association, 2007.
- [18] Harry Eldridge, Gabrielle Beck, Matthew Green, Nadia Heninger, and Abhishek Jain. Abuse-resistant location tracking: Balancing privacy and safety in the offline finding ecosystem. In *Proceedings of the 33rd USENIX Conference on Security Symposium, SEC ’24, USA, 2024*. USENIX Association.
- [19] Espressif Systems. ESP SoCs. <https://www.espressif.com/en/products/socs>, December 2023.
- [20] Aurélien Francillon, Boris Danev, and Srdjan Capkun. Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2011, San Diego, California, USA, 6th February - 9th February 2011*. The Internet Society, 2011.
- [21] Google LLC. Find My Device Network Accessory Specification. Technical report, Google LLC, December 2023. <https://developers.google.com/nearby/fast-pair/specifications/extensions/fmdn>.

- [22] Google LLC. Target ads to geographic locations. <https://support.google.com/google-ads/answer/1722043?hl=en>, 2024.
- [23] Matthew Green. How does Apple (privately) find your offline devices? <https://blog.cryptographyengineering.com/2019/06/05/how-does-apple-privately-find-your-offline-devices/>, June 2019.
- [24] Andy Greenberg. The Simple Way Apple and Google Let Domestic Abusers Stalk Victims. *Wired*, July 2019. <https://www.wired.com/story/common-apps-domestic-abusers-stalk-victims/>.
- [25] Zac Hall. Should AirTags get lost over safety concerns? A better solution can be found. <https://9to5mac.com/2022/01/17/airtag-stalking-prevention/>, January 2022.
- [26] Alexander Heinrich, Niklas Bittner, and Matthias Hollick. AirGuard - Protecting Android Users from Stalking Attacks by Apple Find My Devices. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 26–38, San Antonio TX USA, May 2022. ACM.
- [27] Alexander Heinrich, Milan Stute, and Matthias Hollick. 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*, pages 374–376, Abu Dhabi United Arab Emirates, June 2021. ACM.
- [28] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. Who can find my devices? Security and privacy of apple’s crowd-sourced bluetooth location tracking system. *Proc. Priv. Enhancing Technol.*, 2021(3):227–245, 2021.
- [29] Erik Kay. 5 ways to use the new Find My Device on Android. <https://blog.google/products/android/android-find-my-device/>, April 2024.
- [30] Dave Kleidermacher. 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>, April 2024.
- [31] Brent Ledvina, Zachary Eddinger, Ben Detwiler, and Siddika Parlak Polatkan. Detecting Unwanted Location Trackers. Internet-Draft draft-detecting-unwanted-location-trackers-01, Internet Engineering Task Force, December 2023. <https://datatracker.ietf.org/doc/draft-detecting-unwanted-location-trackers/01/>.
- [32] Tong Li, Jiaxin Liang, Yukuan Ding, Kai Zheng, Xu Zhang, and Ke Xu. On Design and Performance of Offline Finding Network. In *IEEE INFOCOM 2023 - IEEE Conference on Computer Communications*, pages 1–10, New York City, NY, USA, May 2023. IEEE.
- [33] Yue Li, Zhenxiong Yan, Wenqiang Jin, Zhenyu Ning, Daibo Liu, Zheng Qin, Yu Liu, Huadi Zhu, and Ming Li. GPSBuster: Busting out Hidden GPS Trackers via MSoC Electromagnetic Radiations. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 3302–3316, Salt Lake City UT USA, December 2024. ACM.
- [34] Ben Lovejoy. AirTags with deactivated speakers being sold on eBay and Etsy. <https://9to5mac.com/2022/02/03/airtags-with-deactivated-speakers-being-sold/>, February 2022.
- [35] Travis Mayberry, Erik-Oliver Blass, and Ellis Fenske. Blind My - An Improved Cryptographic Protocol to Prevent Stalking in Apple’s Find My Network. *Proceedings on Privacy Enhancing Technologies*, 2023(1):85–97, January 2023.
- [36] Travis Mayberry, Ellis Fenske, Dane Brown, Jeremy Martin, Christine Fossaceca, Erik C. Rye, Sam Teplov, and Lucas Foppe. Who Tracks the Trackers?: Circumventing Apple’s Anti-Tracking Alerts in the Find My Network. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, pages 181–186, Virtual Event Republic of Korea, November 2021. ACM.
- [37] Lauren R. Pace, LaSean A. Salmon, Christopher J. Bowen, Ibrahim Baggili, and Golden G. Richard. Every step you take, I’ll be tracking you: Forensic analysis of the tile tracker application. *Forensic Science International: Digital Investigation*, 45:301559, July 2023.
- [38] Pallets Developers. Flask. <https://flask.palletsprojects.com/en/stable/>, February 2025.
- [39] Sultan Quasim Khan. Technical Advisory – BLE Proximity Authentication Vulnerable to Relay Attacks. <https://research.nccgroup.com/2022/05/15/technical-advisory-ble-proximity-authentication-vulnerable-to-relay-attacks/>, May 2022.
- [40] Samsung Electronics Co., Ltd. SmartThings Find. <https://smarthingsfind.samsung.com>, 2024.
- [41] Narmeen Shafqat, Nicole Gerzon, Maggie Van Nortwick, Victor Sun, Alan Mislove, and Aanjan Ranganathan. Track You: A Deep Dive into Safety Alerts for Apple AirTags. *Proceedings on*

Privacy Enhancing Technologies, 2023(4):132–148, October 2023.

- [42] Paul Staat, Kai Jansen, Christian Zenger, Harald Elders-Boll, and Christof Paar. Analog Physical-Layer Relay Attacks with Application to Bluetooth and Phase-Based Ranging. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 60–72, San Antonio TX USA, May 2022. ACM.
- [43] Tile Inc. How Tile Works. <https://www.tile.com/how-it-works>, 2024.
- [44] Kieron Ivy Turk and Alice Hutchings. Stop Following Me! Evaluating the Effectiveness of Anti-Stalking Features of Personal Item Tracking Devices. <http://arxiv.org/abs/2312.07157>, December 2023.
- [45] Kieron Ivy Turk, Alice Hutchings, and Alastair R. Beresford. Can’t Keep Them Away: The Failures of Anti-stalking Protocols in Personal Item Tracking Devices. In Frank Stajano, Vashek Matyáš, Bruce Christianson, and Jonathan Anderson, editors, *Security Protocols XXVIII*, volume 14186, pages 78–88. Springer Nature Switzerland, Cham, 2023.
- [46] Kyle Vanhemert. Tile Might Be a Revolutionary Gizmo For Finding Lost Keys and Stolen Purses. <https://www.wired.com/2013/08/tile-a-better-way-to-find-your-lost-keys-and-maybe-your-stolen-bike-too/>, August 2013.
- [47] Mira Weller, Jiska Classen, Fabian Ullrich, Denis Waßmann, and Erik Tews. 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*, pages 184–194, Linz Austria, July 2020. ACM.
- [48] Rachel Younger. AirTags becoming ‘weapon of choice of stalkers and abusers’ as GPS tracker cases rocket by 317%. <https://www.itv.com/news/2024-04-02/airtags-become-weapon-of-choice-of-stalker-s-and-abusers-as-cases-rocket>, April 2024.
- [49] Tingfeng Yu, James Henderson, Alwen Tiu, and Thomas Haines. Security and Privacy Analysis of Samsung’s Crowd-Sourced Bluetooth Location Tracking System. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5449–5466, Philadelphia, PA, August 2024. USENIX Association.
- [50] Joshua Zeliger. Apple and Google deliver support for unwanted tracking alerts in iOS and Android. <https://www.apple.com/ca/newsroom/2024/05/apple-and-google-deliver-support-for-unwanted-tracking-alerts-in-ios-and-android/>, May 2024.

A Appendix: Key Derivation Algorithms

A.1 Apple

With (d_0, p_0) and SK_0 being the initial P-224 key pair and symmetric key generated at provision time, Apple generates the broadcast public key p_i as follows:

$$\begin{aligned} SK_i &= \text{KDF}(SK_{i-1}, \text{“update”}, 32) \\ (u_i, v_i) &= \text{KDF}(SK_i, \text{“diversify”}, 72) \\ d_i &= (d_0 * u_i) + v_i \\ p_i &= d_i * G \end{aligned}$$

Here, KDF denotes the ANSI X.963 key derivation function, and G is the generator for the NIST P-224 curve [6]. As these equations show, the current broadcast key is based on the previous round’s symmetric key and the key pair generated during provisioning.

A.2 Google

Google follows a slightly different approach to generate the advertised public key material than Apple. It does not rotate the key material and use a previous round’s key material for deriving the new key but uses a time-based generation algorithm. The algorithm for Google’s key derivation is as follows:

$$\begin{aligned} t &= \text{32-bit big-endian beacon time counter} \\ t' &= t \ \& \ 0\text{xFFFFFFC00} \quad // \text{Clear lowest 10 bits} \\ T &= 0\text{xFFFFFFFFFFFFFFFFFFFFFFF0A} \ || \ t' \ || \\ &\quad 0\text{x0000000000000000000000000000A} \ || \ t' \\ r' &= \text{AES-ECB-256}(SK, T) \quad // \text{Encrypt } T \text{ under } SK \\ n &= 160 \text{ for SECP160R1, } 256 \text{ for SECP256R1} \\ r &= r' \ \text{mod } n \\ R &= r * G \\ p_i &= Rx \quad // \ x \text{ coordinate of } R \end{aligned}$$

The current round’s broadcast key is derived from the encryption of a counter under the provisioned symmetric key. The hashed status flags referenced in Table 3 are the result of XORing the battery level and unwanted tracking mode flags with the least significant byte of $\text{SHA}_{256}(r)$ [21].

B Appendix: Advertisement Payload Format

Table 3 details the Bluetooth Low Energy (BLE) advertisement payloads as used by Apple and Google in their corresponding offline finding networks.

Table 3: Advertisement payloads for Apple’s and Google’s offline finding implementations as well as the shared proposal [6,21,31]. p denotes the public key. Notably, Apple encodes parts of the public key in the BLE address in order to fit the full 28 byte key into an at maximum 37 byte advertisement.

Bytes	Apple	Google (BLE legacy advertising)	Google (BLE extended advertising)	Unwanted tracking standard
0..5	$(p[0] \ll (0b11 \ll 6)) \ll p[1..5]$	BLE address	BLE address	BLE address
6	Payload length = 0x14	Flag length = 0x02	Flag length = 0x02	Flag length = 0x2 (optional)
7	Advertisement type = 0xFF	Flag type = 0x01	Flag type = 0x01	Flag type (optional)
8	Company ID = 0x004C	Flag data = 0x06	Flag data = 0x06	Flag data (optional)
9		Service data length = 0x18 or 0x19	Service data length = 0x24 or 0x25	Service data length
10	Offline finding (OF) type = 0x12	Service data type = 0x16	Service data type = 0x16	Service data type
11	OF payload length = 0x19	Service UUID = 0xFEAA	Service UUID = 0xFEAA	0xFCB2
12	Status flags (e.g., battery level)			
13		Mode = 0x40 (near-owner mode) or 0x41 (separated mode)	Mode = 0x40 (near-owner mode) or 0x41 (separated mode)	Network ID = 0x01 (Apple) or 0x02 (Google)
14	$p[6..27]$	$p[0..19]$		Mode = 0x00 (near-owner mode) or 0x01 (separated mode)
15..33			$p[0..31]$	
34		Hashed status flags		Proprietary company payload
35	$p[0] \gg 6$			
36	Hint (0x00 for iOS reports)			
37..45				
46			Hashed status flags	



USENIX Security '26 Artifact Appendix: PRIVACYSHIELD: Relaying BLE Beacons to Counter Unsolicited Tracking

Florian Hofhammer
EPFL

Daniele Antonioli
EURECOM

Mathias Payer
EPFL

A Artifact Appendix

A.1 Abstract

PRIVACYSHIELD is a relay-based system to protect users against unsolicited tracking via offline finding trackers such as Apple AirTags. These offline finding trackers emit Bluetooth Low Energy (BLE) beacons that are then picked up by nearby devices such as smartphones. These devices then upload their current location as an approximation of the trackers location to the vendor's servers, from which the owner of the tracker can then retrieve it.

As such trackers are small, easy to hide, and available at low cost, they are frequently abused for malicious purposes such as stalking. PRIVACYSHIELD aims to prevent such misuse by masking a potential victim's location. To this end, PRIVACYSHIELD relays AirTag beacons via a network of relay stations that reemit the beacons at different locations than where they were originally. Consequently, third-party devices detect the beacons at these locations and report the wrong location to the vendor's servers. The AirTag's real location is thus hidden from the owner of the tracker, which in this malicious case is the perpetrator.

This artifact appendix describes our artifact for PRIVACYSHIELD. The artifact contains the source code and documentation for (i) the relay application submitting AirTag beacons to the PRIVACYSHIELD server, (ii) the relay server collecting and redistributing AirTag beacons, (iii) the relay firmware reemitting beacons received from the server, and (iv) scripts to collect location records from Apple's servers for evaluation purposes.

A.2 Description & Requirements

This section describes the software and hardware requirements to reproduce a deployment of PRIVACYSHIELD.

A.2.1 Security, privacy, and ethical concerns

As described in the Ethical Considerations section of our paper, PRIVACYSHIELD selectively hides the location of trackers from their owners. While this is intended to protect potential victims from stalking, it may also temporarily and inadvertently prevent legitimate object tracking use cases.

Therefore, we recommend evaluators to take precautionary measures when recording and relaying AirTag beacons, such as notifying people in the vicinity about the ongoing experiment and adjusting the minimal signal strength threshold in the AirGuard application to only capture beacons from AirTags in the direct vicinity, i.e., AirTags that are part of the experiment.

A.2.2 How to access

PRIVACYSHIELD is available at <https://doi.org/10.5281/zenodo.17964520> and <https://github.com/HexHive/privacyshield>. We use the (stable) Zenodo reference to point to the version of the artifact as it passed the Artifact Evaluation process. The (floating) GitHub reference points to the latest version of the artifact, which at the time of writing is identical to the Zenodo version and identified by the tag `sec26-artifact-evaluated`, commit `11a509cfcab20feaa12e735ef701366ba88f6066`.

A.2.3 Hardware dependencies

The minimal hardware requirements for evaluating general functionality of PRIVACYSHIELD are as follows:

- an AirTag,
- an Android device with Bluetooth Low Energy (BLE) support to run the AirGuard application for recording and uploading beacons,
- an Apple device (iPhone, iPad, or Mac) to pick up the beacons and upload the location records to Apple's servers,
- a second Apple device to retrieve the location records and show the AirTag's location in the Find My application,
- a computer (physical or virtual) to run the PRIVACYSHIELD server (must be accessible from the Android device running the application as well as the relay station), and
- an ESP32-C3-based development board to act as a relay station. Other ESP32 variants may also work but have not been tested.

If one of the Apple devices used is a Mac, the PRIVACYSHIELD server can also be run on the same device. This does not accurately reflect a real-world deployment but is sufficient for the sake of the experiments.

Note that the above hardware requirements are minimal and can only verify the general functionality of PRIVACYSHIELD. In order to retrieve location reports from Apple's servers for evaluation purposes outside of the Find My application, a Mac is required as one of the Apple devices. This requirement stems from the need to extract the AirTag's private key from the macOS keychain, which is not possible on iOS or iPadOS devices.

A.2.4 Software dependencies

All the steps for building binary artifacts and running the relay server are containerized. The only software requirement is therefore a working Podman or Docker installation.

Note that the (optional) evaluation scripts based on [Findmy.Py](#) require a working Python3 installation and extracting AirTag's private keys from a macOS keychain first. This step is a prerequisite for the scripts and is not part of our artifact. We point the evaluator to upstream tutorials on how to perform this step, e.g., <https://docs.mikealmel.ooo/FindMy.py/getstarted/02-fetching.html>.

A.2.5 Benchmarks

None.

A.3 Set-up

Make sure to have the aforementioned software dependencies installed.

Furthermore, if you aim to retrieve location reports from Apple's servers outside of the Find My application, make sure to correctly extract the AirTag's private keys from a macOS keychain first. We refer to upstream tutorials such as <https://docs.mikealmel.ooo/FindMy.py/getstarted/02-fetching.html> for this step. This is not required for general functionality tests but was used to collect statistical information for our evaluation.

A.3.1 Installation

First, download the artifact, e.g., via `git clone https://github.com/HexHive/privacyshield`.

Set up and run the PRIVACYSHIELD server, the modified AirGuard application, and the relay firmware as described in the `README.md` in the repository. This includes installing required Python packages, applying our patches to upstream projects where necessary, and building and running binary artifacts and scripts.

A.3.2 Basic Test

If all the components are installed and running correctly, you can perform some basic tests to verify the general functionality of PRIVACYSHIELD.

To test whether the relay application is able to pick up AirTag beacons and upload them to the PRIVACYSHIELD server, start the application on the Android device and run `adb logcat --pid=$(adb shell ps | grep seemoo | cut -d ' ' -f 8) | grep insertScanResult` on a device that has adb access to the Android device, e.g., the machine you built the application on. Then, bring an AirTag in the vicinity of the Android device. If everything works correctly, you should see log messages indicating that AirTag beacons were picked up.

To test whether the server can receive beacons, run it with increased logging output, e.g., via `python3 server.py -v`. To test whether the server correctly redistributes beacons to relays, you can manually submit a beacon via `curl: curl -L --json '{"data": "tyPkKWDIHv9MABIZEE2KEXyvZ+Vfq1kYAXoD35MRr1ER I44CAA==", "valid_to": "2026-02-28 00:00:00"}' "http://<server address here>/api/v1/airtag"`. This should successfully add the beacon to the server's database, which you can then retrieve via `curl -vL "http://<server address here>/api/v1/airtag?valid=true"`.

Once you configured and installed the relay firmware on the microcontroller development board, you can check its log outputs via a serial connection, e.g., using `screen /dev/ttyUSB0 115200` (assuming the serial device shows up as `/dev/ttyUSB0`, replace if necessary).

Optionally, as an end-to-end test, you can verify that relaying beacons works by bringing the AirTag close to the Android device running the application, and then removing the battery from the AirTag to stop it from emitting beacons. If everything works correctly, the AirTag should still show up as providing updates in the Find My application, as the relay station now effectively turns into a clone of the original AirTag.

A.4 Evaluation workflow

This section describes the steps required to evaluate the masking functionality of PRIVACYSHIELD based on a working system as set up in the previous section.

[Mandatory for Artifacts Functional & Results Reproduced, optional for Artifact Available] This section should include all the operational steps and experiments which must be performed to evaluate if your your artifact is functional and to validate your paper's key results and claims. For that purpose, we ask you to use the two following subsections and cross-reference the items therein as explained next.

A.4.1 Major Claims

- (C1):** PRIVACYSHIELD effectively masks the location of AirTags from their owner by relaying their beacons at a different location. We demonstrate this in the experiments described in Section 5.3 of the paper.
- (C2):** PRIVACYSHIELD can mask the location successfully even when there are significantly fewer devices picking up the relayed beacons compared to the original beacons. We demonstrate that this behavior can be achieved by adjusting the frequency at which the relays emit beacons. This is shown in the experiments described in Sections 5.4 and 5.5 of the paper.

A.4.2 Experiments

- (E1):** [Functionality Verification] [30 human-minutes]: This experiment verifies the functionality according to claim (C1).

Preparation: Make sure the Android application, the PRIVACYSHIELD server, and the relay firmware are correctly installed and running as described in [Section A.3](#).

Execution: Bring the experiment AirTag close to the Android device running the sniffing application. With the relay station powered on and being located physically separate from the AirTag, check whether the AirTag shows up in the Find My application at the relay stations position instead of its real position.

Results: The AirTag should show up in the Find My application at the relay station’s position, demonstrating that PRIVACYSHIELD successfully masked the AirTag’s real location.

- (E2):** [Verification of Resilience to Adversarial Environments] [30 human-minutes]: This experiment verifies that PRIVACYSHIELD can successfully mask the location of an AirTag even when there are significantly fewer devices picking up the relayed beacons compared to the original beacons, as described in claim (C2).

Preparation: Make sure the Android application, the PRIVACYSHIELD server, and the relay firmware are correctly installed and running as described in [Section A.3](#). Ensure that the AirTag’s beacons are picked up by multiple devices, e.g., by placing it in a busy area. Ensure that the relay station is placed in an area with significantly fewer devices picking up beacons, e.g., a controlled office environment.

Execution: Bring the experiment AirTag close to the Android device running the sniffing application. With the relay station powered on and being located physically separate from the AirTag, check whether the AirTag shows up in the Find My application at the relay stations position instead of its real position. If this is not the case, adjust the frequency at which the relay emits the relayed beacons by reducing the `BLE_ADVERTISEMENT_INTERVAL` value for the relay

firmware and repeat the experiment.

Results: The AirTag should show up in the Find My application at the relay station’s position after adjusting the advertisement frequency. The results should mirror those presented in Table 2 in the paper, if the environment is similar to the one used in our evaluation.

Note that the above experiments are sufficient to validate the major claims of the paper. To get more fine-grained results that back up the claims made in the paper, the evaluator can optionally retrieve the history of location reports from Apple’s servers using the scripts provided in the `findmy/` subdirectory of the repository. These scripts allow precomputing the AirTag’s public keys for a configurable time frame, retrieving location reports, and plotting them on a map.

Warning

Interacting with Apple’s servers with an unofficial client may get your account banned. If you choose to do those fully optional steps, make sure to use a secondary Apple ID!

A.5 Notes on Reusability

Our server implementation can easily be scaled horizontally by deploying it on multiple machines. These can then either serve disjoint sets of AirTag beacons, or share a common database by replacing the current SQLite backend with a networked database such as PostgreSQL, MySQL, or similar. Furthermore, the server can be federated by submitting beacons to other servers via the provided REST API.

The relay firmware can be adapted to other microcontroller platforms leveraging their built-in BLE capabilities and APIs. The current implementation is based on the Espressif SDK, and consequently should run successfully on any ESP32 variant that supports BLE and WiFi.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at <https://secartifacts.github.io/usenixsec2026/>.