SoK: Offline Finding Protocols for Lightweight Location Tracking

Akshaya Kumar
Georgia Institute of Technology
Atlanta, Georgia, USA
akshayakumar@gatech.edu

Carolina Ortega Pérez
Cornell University
New York, New York, USA
carolina@cs.cornell.edu

Joseph Jaeger
Georgia Institute of Technology
Atlanta, Georgia, USA
josephjaeger@gatech.edu

Thomas Ristenpart
University of Toronto
Toronto, Ontario, Canada
ristenpart@cs.toronto.edu

Michael A. Specter
Georgia Institute of Technology
Atlanta, Georgia, USA
specter@gatech.edu

Abstract

Offline finding (OF) protocols—such as Apple's Find My, Google's Find Hub, Samsung's SmartThingsFind, and Tile—enable hundreds of millions of users to track their belongings via Bluetooth-based tracker tags. However, their scale and tracking capabilities give rise to privacy risks for tag owners and bystanders, as well as safety risks for victims of tag-facilitated stalking. In response, academics and practitioners have suggested cryptographic and non-cryptographic mitigations to improve privacy and anti-stalking protections, working to navigate complex and subtle tensions between these goals. The result is a large landscape of privacy goals, threat models, protocol designs, implementations, and analyses.

In this work, we systematize the OF protocol landscape. We gather and analyze a corpus of 49 research papers and OF protocol technical specifications, and use it to develop a taxonomy capturing the functionality, security, and privacy goals of OF protocols. We use the taxonomy to guide a focused assessment of the four major OF deployments along with six academic constructions, comparing design choices, consolidating known attacks, and analyzing the designs' trade-offs between privacy, security, abusability, and efficiency. We provide a simple OF protocol that achieves most security goals, and which clarifies the essential cryptographic components underlying OF protocols. We also provide a survey of physical layer attacks and usability issues that undermine protections in practice. Finally, we discuss open problems and potential research directions towards secure, interoperable, and abuse-resistant OF systems.

1 Introduction

Over the past decade, Bluetooth-based offline finding (OF) systems have transformed how users track their belongings. First popularized by Tile in 2014 [**tileNews**], OF systems leverage large crowdsourced networks of participating "finder" devices to physically locate tracker tags. Tags can be as small as a coin, and use Bluetooth to advertise themselves to finder devices, which forward the advertisement together with the finder's current location to the vendor's backend. The tag's owner can then access this tracking information. OF systems now underpin a massive, always-on location-tracking infrastructure supporting hundreds of millions of trackers via networks operated by Apple [5], Google [31], Samsung [60], and Life360 (which bought Tile) [49].

The very properties that make OF systems effective for finding items also introduce a variety of security, privacy, and safety risks. In terms of privacy, tags could be used by OF network operators to track tag owners or finders [42, 41, 50, 81]. At the same time, tags are a quintessential dual-use technology: malicious tag owners can slip them into a victim's bags, clothing, or vehicle to easily track their location. Trackers are routinely implicated in stalking, theft, physical assault, murder, and other criminal activities [15, 17, 53, 9, 68, 69, 28]. In response, vendors introduced "anti-stalking" features that notify users when an unknown tracker is persistently detected nearby [6, 4, 59, 48, 67, 46, 23]. Complications arise here given tensions between threat models: naive anti-stalking features sacrifice owner privacy (and vice versa).

These challenges are compounded by the proprietary and undocumented nature of OF protocols requiring the academic community to reverse engineer them individually [38, 80, 58, 51, 10, 44]. Unsurprisingly, these papers and others [82, 52, 81, 47, 50, 16, 55, 11] found attacks that circumvent privacy and anti-abuse mitigations. Simultaneously, researchers formalized various cryptographic properties for OF systems and proposed secure protocol constructions [25, 20, 54, 22, 19, 36]. The result is a patchwork and complex landscape, with so far no comprehensive treatment of threat models and security, privacy, and safety goals, nor a clear explanation of the core mechanisms that underlying OF protocols use to achieve them.

In this work, we provide the first comprehensive systematization of security, privacy, and anti-stalking in the context of OF protocols. We review 40 academic papers, including protocol designs, attacks, measurement, and user studies, as well as nine industry specifications. To start, we use the various papers and available specifications to reconstruct and compare the protocol designs from the major tag vendors (i.e., Apple, Google, Samsung, and Life360). Based on this, we describe the general OF protocol phases and detail a relatively simple protocol that can be viewed as a "textbook" version of how OF protocols can be built.

We then define 11 security properties for OF protocols, spanning privacy guarantees for all protocol participants as well as protections against abuse. Ten of these properties are derived from descriptions in our corpus, while one is a new goal we propose after identifying unaddressed gaps.

We partition the 11 goals into ones targeting the security of tag owners and ones targeting the security of finders. We further

Proceedings on Privacy Enhancing Technologies 2026(4), 131–155

https://doi.org/10.56553/popets-2026-0113

categorize a subset of these goals as *baseline* goals, which are specifically targeted and achieved by most vendors and can be thought of as the core security and privacy goals of an OF system.

We evaluate both deployed protocols and academic proposals against our taxonomy. We find that no system achieves all baseline goals—an unsurprising result given the inherent tensions between many security objectives. We provide intuition for the design decisions and vulnerabilities that prevent protocols from achieving specific properties, and illustrate the trade-offs that arise when balancing privacy and abuse prevention.

Based on our findings, we identify promising directions for future research and policy to guide the development of usable, privacy-preserving OF systems that minimize abuse while respecting the practical constraints of large-scale deployment.

Summary. We provide the first systematization of the security, privacy, and safety landscape of OF protocols. In this work, we:

- contribute a corpus of relevant papers between 2014 and 2025, which we will submit as an artifact;
- contribute a comprehensive set of security, privacy, and anti-abusability goals and corresponding threat models to evaluate the security of OF protocols;
- describe the OF protocol implementations of Apple, Google, Samsung, and Tile, identifying differences in cryptographic design and evaluating them relative to the security goals;
- discuss open research and policy challenges in balancing efficiency, privacy, and abuse prevention.

2 Background

We begin by reviewing the origins and security analyses of OF systems to motivate the need for a systematization. We then provide background on Bluetooth Low Energy (BLE), the core communication layer underlying OF networks.

The evolution of OF networks. The idea of crowdsourcing to track objects using Bluetooth-capable tags and location-enabled smartphones was proposed in early location-tracking systems [24]. In the 2010s, this vision was commercialized by companies such as TrackR [29], Tile [49], Nut [57], and Cube [71]. However, these systems offered little in the way of privacy or anonymity and finder networks were limited to application users.

Apple's introduction of AirTags and the Find My network [40] marked a turning point, promising strong cryptographic guarantees for the security and privacy of users. Samsung and Google soon launched their own finder networks [61, 30]. Unlike earlier systems with small user bases, these major vendors could instantly bootstrap massive finder networks from their global smartphone ecosystems.

Despite their scale, these systems are remarkably opaque. Vendors keep designs proprietary and offer minimal documentation. Academic researchers have had to reverse engineer their protocols piece by piece to understand how they work [38, 80, 58, 11, 51, 10].

The complexity of analyzing these systems is exacerbated by their inherently conflicting privacy and abuse-prevention goals. On the one hand, privacy of tag owners requires that a tag's Bluetooth advertisements be frequently rotated and *unlinkable*, so that a "super observer" cannot fingerprint the tag and track its owner over time. On the other hand, abuse-prevention requires the opposite: bystander devices must be able to *link* a tag's advertisements in order to detect when a rogue tag is following them. Cryptography helps reconcile these goals, but every vendor solves the problem differently, and the details are rarely public. Academic work has shown attacks against both the privacy [82, 52, 81, 47, 50, 16] and abuse-prevention [37, 55, 58, 11, 10] implementations. In parallel, researchers have developed formal models to capture the complex security and privacy goals of OF protocols and proposed secure solutions [25, 20, 54, 22, 19].

What has resulted is a fragmented landscape: results are spread across different attack surfaces, adversary models, and mitigations, making it difficult to systematically assess which protections work, which fail, and what design principles should guide future OF systems. Without transparency, it's hard for researchers or policymakers to assess whether the solutions are secure or even functional. This motivates our systematization effort.

Bluetooth Low Energy (BLE). BLE communication occurs via advertisements or full connections. Advertisements are connectionless broadcasts used for discovery and connection initiation [79]. When a connection is established, bidirectional data exchange occurs. However, to save energy, a device can keep advertising without forming a connection even after pairing, and peers can reply with lightweight responses to indicate they are in range.

Hardware constraints. Solutions to secure OF networks are bound by the hardware constraints of tracker tags. Tags are designed to run for approximately one year on a coin cell battery, requiring them to minimize computation and transmission. Deployments therefore use BLE for its power efficiency. However, BLE advertisement payloads are capped at 31 bytes in Bluetooth 4.0 [79] constraining the size of any cryptographic material a tag can broadcast.

3 Literature search methodology

To construct our corpus of papers, we used two independent approaches, each conducted by a different author. For the first method, we used a seed set of 13 recent highly cited papers that directly addressed the security and privacy of OF systems and recursively examined backward and forward citations to identify additional relevant work until reaching saturation. For the second method, we conducted a systematic keyword-based search across major academic venues in security, privacy, and cryptography: USENIX Security, ACM CCS, IEEE S&P, NDSS, PETS, SOUPS, ACM WiSec, ACM WPES, and IACR ePrint from 2014-2025. Our final corpus, available at our repository [43], is the union of the two methods, consisting of 40 academic papers and nine industry documents. We detail our methodology in Appendix A.

Roadmap. In Section 4, we introduce the functionality goals of an OF protocol, using a canonical construction that we call BasicOF as a running example of how each goal can be instantiated, while simultaneously describing deployed protocols. In Section 5 we present a set of security goals for OF systems categorized into owner privacy and finder security. We evaluate deployed and academic protocols against these properties in Section 6. In Section 7, we highlight physical and UI-level challenges in securing OF networks. Finally, we list takeaways and directions for future work in Section 8 and conclude with the limitations of our work in Section 9.

4 Protocol phases & implementations

OF systems can be divided into four main sub-protocols: pairing, connected mode, lost mode, and unwanted tracking detection. Here we discuss each of these stages. Along the way we present a canonical construction, BasicOF, that achieves the described functionality and discuss how four major vendors—Apple, Google, Samsung, and Tile—instantiate these phases in their respective implementations.

4.1 Pairing

Pairing is the first step in the life cycle of a tracker tag, where a user pairs a *tracker tag* with an *owner device*. Tracker tags have no internet or GPS capabilities, meanwhile the owner device is typically a smartphone with internet connectivity. In this phase, the tag and its owner device (1) authenticate themselves as legitimate participants in the service provider's network and (2) establish a shared key, with some help from the service provider. Intuitively, this can be achieved by executing an Authenticated Key Exchange (AKE) protocol between the tag and the owner device where the service provider establishes the relevant public key infrastructure.

Canonical construction. In Figure 1, we present a canonical protocol for the pairing phase of an OF system. We assume that the service provider provisions unique identities, id_{tag} and id_{owner}, and signing key pairs, $(sk_{\mathsf{tag}}, vk_{\mathsf{tag}})$ and $(sk_{\mathsf{owner}}, vk_{\mathsf{owner}})$, to the tag and owner device respectively. The service provider has its own signing key pair $(sk_{\mathsf{server}}, vk_{\mathsf{server}})$, with vk_{server} known to the tag and owner.

Our protocol involves the execution of a standard signed Diffie-Hellman (DH) AKE protocol to derive a shared secret K_{shared} between the tag and its owner. The tag and the owner each generate ephemeral key pairs (x, X) and (y, Y) respectively. Each party signs their ephemeral public key using their signing key and sends their identity, their ephemeral public key, and the corresponding signature to the server, with the owner device acting as a proxy to the server for the tag. The server verifies the received signatures and ensures that the tag was not previously registered. Upon successful verification, the server maps the tag to the owner as registered pairs in its local database. Finally, the tag and the owner device execute a Diffie-Hellman key exchange to derive K_{shared} using their respective shares X and Y. This setting mirrors deployed protocols and differs from standard AKE since instead of mutually authenticating each other, the tag and the owner rely on the server for authentication.

Deployed protocols. We now give a high-level overview of the pairing phase implementations of the four major vendors. We provide a summary in Table 1 and full details in Appendix B.

All deployments use an AKE protocol to establish a shared key between the tag and the owner device during pairing. In all implementations, the owner device authenticates itself by requiring the owner to log in to their account. However, they differ in:

(1) *Who participates in the KE:* We observe two distinct design choices: either the KE is executed directly between the tag and the owner device (Apple and Google), or between the tag and the server (Samsung and Tile). In the latter case, the server sends K_{shared} to the owner device over TLS.
(2) *How tags authenticate:* Tags may authenticate themselves to the server via manufacture-provisioned secrets (Apple, Tile, Samsung) or cryptographically to the owner (Google).

These design choices impact both the security and privacy guarantees of the OF system as discussed in Section 6.

4.2 Connected mode

After registration, the tag operates in connected mode when it is within Bluetooth range of its owner device. In this phase, the tag periodically broadcasts BLE advertisements derived from the shared key, which its owner device can recognize and acknowledge.

Tags continue advertising after pairing instead of switching to an encrypted BLE connection to save energy. BLE connections between the owner device and tag are only established when performing user actions on the tag, such as instructing a tag to emit a sound.

BLE advertisements are broadcast unencrypted, making them readable by any attacker with a radio frequency (RF) detector. If a tag uses a static identifier—or a static MAC address—such an attacker can easily fingerprint it and track a user based on the advertisements broadcast by their tag. This is a known issue in Bluetooth privacy and MAC address randomization is a standard mitigation [78, 2]. However, randomizing the MAC address alone is insufficient if the payload contains a static identifier. So, tags periodically rotate their identifiers, such that they are unlinkable.

Canonical construction. We present a canonical protocol for connected-mode interactions in Figure 1. Our construction derives connected-mode identifiers id_i by applying a key-derivation function KDF to the shared key K_{shared} and an epoch counter i. The counter is incremented every Δ_{conn} minutes; this is a configurable parameter of the scheme. Connected-mode advertisements also encode their mode—as indicated by the string "conn" in the advertisement—signaling to nearby finder devices that the tag is in connected mode and that its location should not be reported.

Deployed protocols. We detail the connected-mode identifier-generation methods for the four vendors in Table 2. All implementations set some epoch duration Δ_{conn} and derive their connected-mode advertisements by applying a KDF to (a function of) K_{shared}. They differ in exactly what function of K_{shared} is used to derive the identifiers id_i and how the state of the tag is encoded in these advertisements. Though Apple and Google's connected-mode identifiers are public keys, they are never used for encryption. Apple and Google reserve a single bit in advertisements to indicate the tag's mode. Samsung tags can advertise one of six different operating states, based on how many devices the tag is connected to and how long it has been disconnected (see Table 1 of [80]). Tile does not distinguish between connected and lost modes.

4.3 Lost mode

Once a tag has been outside Bluetooth range of its owner device for a threshold duration, it transitions to lost mode. In this state, the tag periodically broadcasts BLE advertisements intended for discovery by nearby finder devices—bystander devices in the service provider's network—that have both GPS and internet capabilities and continuously scan for such advertisements. Upon receiving a lost-mode advertisement, a finder extracts a special lost-mode identifier therein, computes its current GPS location, and sends the identifier and its current location to the service provider. To locate the lost tag, the owner can subsequently query the provider with

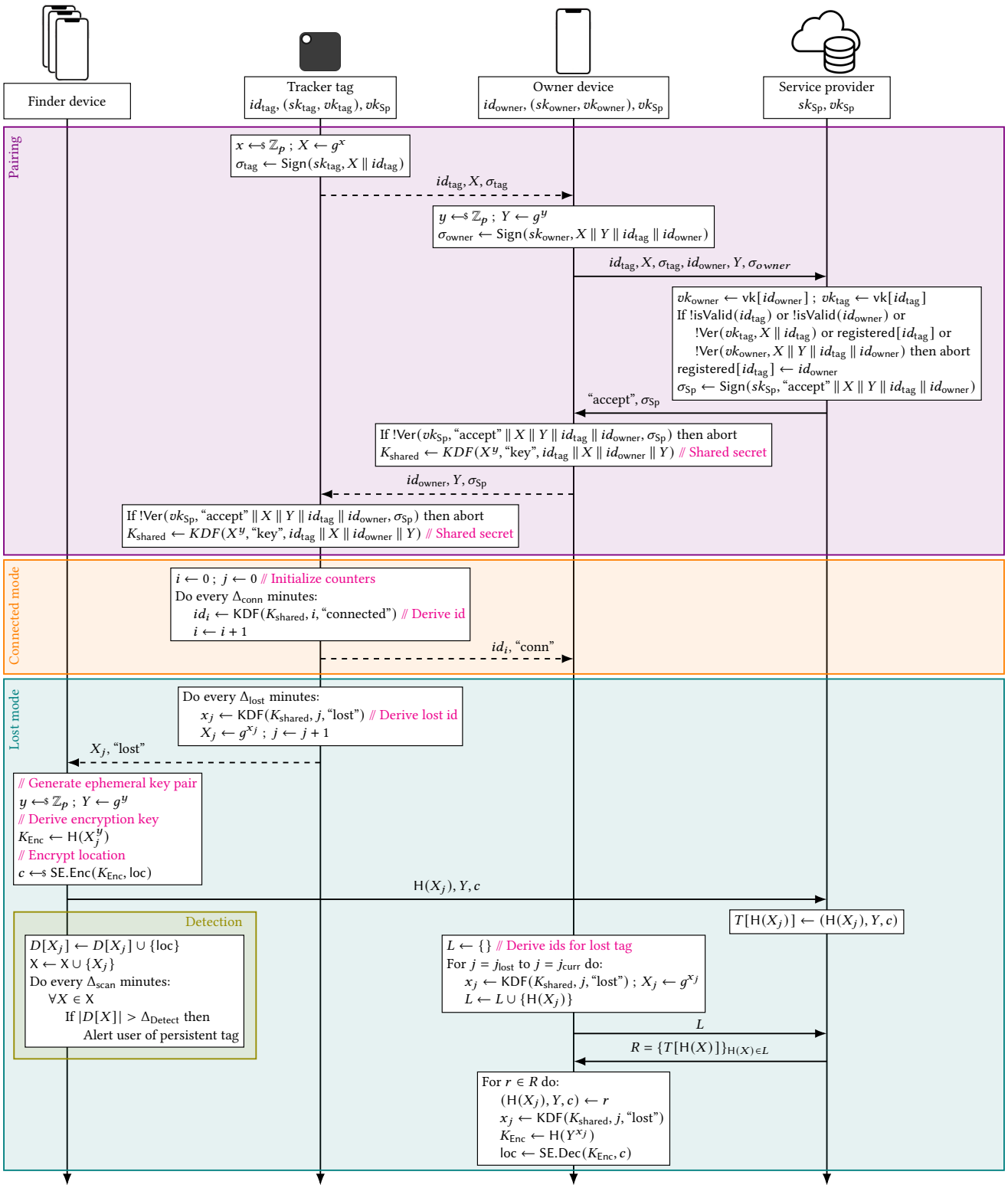

Figure 1: The canonical construction BasicOF **for an OF protocol built using an authenticated key exchange protocol** AKE**, a key derivation function** KDF**, and a symmetric encryption scheme** SE **as building blocks. Here, we instantiate** AKE **using a standard signed Diffie-Hellman key exchange protocol. Dashed arrows denote BLE advertisements; solid arrows denote TLS connections.**

	Key Exchange Participants	(Tag) authentication method	Summary of KE
Apple	Tag, owner	Tag authenticates to the server by encrypting its serial number and other secrets provisioned at manufacture time under the server's public encryption key pk_{server}. The server decrypts and checks whether the serial number matches the secrets and ensures that the tag was not previously registered.	The tag and the owner device generate ephemeral key pairs, (x, X) and (y, Y), and nonces n_{tag} and n_{owner} respectively. The tag and owner then compute the "master" public key $X_0 \leftarrow X \cdot Y$, the connected-mode secret key (SKN_0) and the separated-mode secret key (SKS_0) as $SKN_0 \parallel SKS_0 \leftarrow \mathsf{KDF}(X_0, n_{\text{tag}} \parallel n_{\text{owner}})$. The shared key K_{shared} is (X_0, SKN_0, SKS_0).
Google	Tag, owner	Tag only authenticates implicitly to the owner device after pairing by producing a MAC over a nonce n using an intermediate key K_{account} derived during key exchange. The owner device verifies the MAC against a locally generated one to authenticate tag.	The server assigns a key pair $(x_{\text{modelId}}, X_{\text{modelId}})$ to each tag of the same model modelId. The owner samples its ephemeral key pair (y, Y) and computes $K_{\text{DH}} \leftarrow \mathsf{H}(X^y_{\text{modelId}})$. Then it samples two symmetric keys K_{account} and K_{shared}. It encrypts K_{account} under K_{DH} and $K_{\text{ephemeralId}}$ under K_{account}, and sends both ciphertexts to the tag. The tag derives K_{DH} and decrypts ciphertexts to get K_{shared}.
Samsung	Tag, server	Tag authenticates to the server by sending a hash of its MAC address. Tag also authenticates to the owner device by producing an encryption of the fixed string "smartthings" under the shared key. The owner decrypts the ciphertext and checks if the message obtained equals "smartthings".	The tag sends $h \leftarrow \mathsf{H}_0(\text{macAddress})$ to the owner device which samples a random string rand and sends (h, rand) to the server. The server retrieves the tag's public key X using h, samples its ephemeral key pair (y, Y), and computes $K_{\text{master}} \leftarrow \mathsf{H}_1(X^y \parallel \text{rand})$. It sets parameters IV, poolSize, and seed and sends K_{master} and$(IV, \text{seed}, \text{poolSize}, Y)$ to the owner device, which forwards the latter to the tag. The tag computes K_{master} and sets $K_{\text{shared}} \leftarrow (K_{\text{master}}, IV, \text{seed})$.
Tile	Tag, server	Tag authenticates to the server by providing a unique device identifier and a hash of the model-specific key provisioned at manufacture time. The server verifies the hash and ensures that the tag was not previously registered.	After generating sresT, the tag computes the shared key $K_{\text{shared}} \leftarrow \mathsf{KDF}(K_{\text{interim}}, \text{sresT})$. Upon receiving the modelId, the server retrieves the corresponding K_{interim}. It derives sresT analogously to the tag and $K_{\text{shared}} \leftarrow \mathsf{KDF}(K_{\text{interim}}, \text{sresT})$. Finally, the server sends K_{shared} to the owner device.

Table 1: Comparison of pairing implementations across vendors.

	Connected-mode ID generation	State encoding
Apple	$(X_0, SKN_0, SKS_0) \leftarrow K_{\text{shared}}$ $SKN_i \leftarrow \mathsf{KDF}(SKN_{i-1}, \text{"update"})$ $u_i \parallel v_i \leftarrow \mathsf{KDF}(SKN_i, \text{"diversify"})$ $u_i \leftarrow u_i + 1\ ;\ v_i \leftarrow v_i + 1$ $id_i \leftarrow X_0^{u_i} \cdot g^{v_i}$	The second bit of the second byte of the payload is set to 1 to indicate connected mode.
Google	$r'_i \leftarrow \mathsf{KDF}(K_{\text{shared}}, \text{0xFF} \parallel \text{rotExp} \parallel i \parallel \text{0x00} \parallel \text{rotExp} \parallel i)$ $r_i \leftarrow r' \bmod n$ $id_i \leftarrow g^{r_i}$	The unwanted tracking byte (UT) encodes a tag's state and is set to 0x40 in connected mode.
Samsung	$(K_{\text{master}}, \text{seed}, IV) \leftarrow K_{\text{shared}}$ $K_{\text{privId}} \leftarrow \mathsf{KDF}_0(K_{\text{master}}, \text{"privacy"})$ $x_i \leftarrow 0 \parallel (i \wedge 255) \parallel \text{seed}$ $\quad \parallel 0 \parallel (i \wedge 255)$ $id_i \leftarrow \mathsf{KDF}_1(K_{\text{privId}}, IV, x_i)[:64]$	Tags encode their state in bits 5–7 of the first byte of the advertisement payload. A tag can be in one of six states (see Table 1 [80]).
Tile	$K_{\text{seed}} \leftarrow \mathsf{KDF}(K_{\text{shared}}, \text{tile\_uuid} \parallel \text{"identity"})$ $id_i \leftarrow \mathsf{KDF}(K_{\text{seed}}, i)[:64]$	Tags do not differentiate between modes and hence do not encode state.

Table 2: Comparison of connected-mode implementations across vendors.

its tag's identifiers and retrieve associated reports. We summarize these interactions in Figure 2.

Canonical protocol. We present a canonical protocol for lost-mode interactions in Figure 1. It uses an epoch duration Δ_{lost}, which determines how frequently a tag rotates its lost-mode identifier. As we discuss in Section 4.4, Δ_{conn} and Δ_{lost} are distinct to accommodate the differing privacy goals of the two modes. A lost tag derives ephemeral secret key x_j by applying a key-derivation function KDF to the shared key K_{shared} and the epoch counter j, and it then generates the corresponding public key as $X_j \leftarrow g^{x_j}$. Additionally, lost-mode advertisements encode this state, as indicated by the string "lost" in the advertisement. While the derivation is structurally similar to connected mode, it is context-separated.

Upon receiving a lost-mode advertisement, a finder sees the tag's ephemeral public key X_j, generates its own ephemeral key pair (y, Y), obtains its current GPS location loc, and constructs a location report $c \leftarrow \mathsf{Enc}(K, \text{loc})$ where $K \leftarrow \mathsf{H}(X_j^y)$. It then sends $\mathsf{H}(X_j)$, Y, and c to the service provider.

To locate their lost tag, the owner device determines a range $[j_{\text{lost}}, j_{\text{curr}}]$ of epoch counters covering when the tag was lost, computes X_j for $j \in [j_{\text{lost}}, j_{\text{curr}}]$ using K_{shared}, and queries the service provider for associated reports by sending each $\mathsf{H}(X_j)$. The provider responds with any matching reports, which the owner decrypts using x_j to recover the tag's location loc.

Deployed protocols. Apple's derivation of lost-mode identifiers is similar to the connected-mode derivation given in Table 2, except it uses a separated-mode key SKS_0 instead of SKN_0. Moreover, it unsets the second bit of the second byte of the payload. Google and Samsung use the same identifier generation as in the connected mode, but set the UT byte to 0x41 (Google) and the 3-bit operational state to 010 or 011 depending on how long the tag was disconnected from its owner (Samsung). Apple, Samsung, and Google all set Δ_{lost} to 24 hours in the lost mode. As mentioned in Section 4.2, Tile does not differentiate between lost and connected modes.

Apple and Google require finders to end-to-end encrypt location information for lost tags by default. In these systems the lost-mode identifiers are public encryption keys. Finders encrypt their location under the advertised public key, and submit the encryption and the hash of the public key to the service provider. The owner decrypts reports by deriving the appropriate decryption key from K_{shared}.

Samsung offers end-to-end encryption as an opt-in feature. The owner sets a 6-digit PIN, generates encryption keys (pk_e, sk_e), encrypts sk_e under the hash of the PIN and an IV, producing ciphertext c, and sends (pk_e, c, IV) to the server. Tags with E2EE enabled encode that fact in a designated field of the advertisement. Finders request pk_e from the server using the tag's identifier, encrypt the location under pk_e, and send it with the identifier to the server. The owner decrypts the reports using sk_e.

Tile does not end-to-end encrypt location reports. Finder devices send id_j and the location in plaintext (over TLS) to the service.

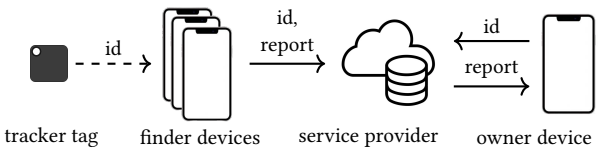

Figure 2: A summary of the lost mode of an OF protocol. Dashed arrows denote BLE advertisements and solid arrows denote TLS connections.

4.4 Unwanted tracking detection

The phases of an OF protocol we have described so far assume that users of the protocol are benign. However, tracker tags have aided several criminal activities including theft, stalking, assault, and even murder [15, 17, 53, 9, 68, 69, 28]. In this setting, a malicious user surreptitiously places their own tag on their target's person or belongings and uses the network to monitor the target's movements. To combat tracker-aided stalking, manufacturers have deployed unwanted tracking detection or "anti-stalking" mechanisms that attempt to alert a user if an unknown tag has been following them.

Intuitively, this requires a finder device to link advertisements broadcast by a tag over time. However, this goal directly conflicts with the unlinkability property we discussed in Section 4.2. A common approach to address this tension is to introduce distinct epoch durations, Δ_{conn} and Δ_{lost}, for the connected and lost modes respectively, such that $\Delta_{\text{conn}} < \Delta_{\text{lost}}$. This means that tags rotate their identifiers more frequently in connected mode than in lost mode. As a result, in lost mode, a finder device can more easily link identifiers across locations to detect tracking, while in connected mode the faster rotation protects the owner's privacy against RF adversaries. In effect, implementations trade off some privacy against RF adversaries in the lost mode to allow unwanted tracking detection.

Canonical protocol. We present a canonical protocol for unwanted tracking detection in Figure 1. Recall that in this construction, finder devices run periodic Bluetooth scans for lost-mode advertisements. In addition to reporting the location of a lost tag, finder devices run a detection algorithm. This algorithm maintains a database of observed lost-mode identifiers and the locations where they were recorded. If the same identifier is recorded at more than a threshold number of distinct locations, the finder device's user is alerted of potential unwanted tracking.

Deployed protocols. All major OF vendors provide some form of unwanted tracking detection. Apple provides *Item Safety Alerts*, Google provides *Unknown Tracker Alerts*, Samsung offers *Unknown Tag Detection*, and Tile provides its *Scan and Secure* feature. At their core, Apple, Google, and Samsung follow the same basic approach as in Fig. 1 with Δ_{lost} set to 24 hours, running detection scans periodically in the background at the OS level. However, individual implementations may enforce additional heuristics like time of day, battery level, and connection to cellular/WiFi network to optimize their algorithms. These heuristics are opaque, owing to the proprietary nature of the systems, and have not been reverse-engineered in academic work, so we omit them from our description.

Tile's approach differs fundamentally in that it requires the user to manually initiate a Bluetooth scan. The scan requires the finder device to be in motion for about 10 minutes, during which six consecutive Bluetooth scans are run. The device records all observed Tile advertisements and forwards them to the server, which then returns identifying information about the detected tags.

5 OF Security and Privacy

In this section, we provide a synthesis of the security and privacy goals discussed in the literature. This synthesis also helps us highlight goals that are missing and that we believe future work should address. We group these goals into two categories, owner security and finder security. We begin by detailing the adversaries and their capabilities, before presenting the security goals. In Section 6, we use these properties as a framework for evaluating the security of deployed OF implementations as well as academic proposals.

Adversaries and their capabilities. As we will see, a wide variety of security properties have or can be targeted by OF systems. Underlying them are a smaller number of threat models related to adversarial capabilities, which we distill down into the following:

- *RF adversary:* An attacker equipped with a radio frequency (RF) detector who can passively intercept Bluetooth transmissions within range or actively inject arbitrary messages.
- *BT MITM adversary:* An RF attacker that is additionally positioned between a tag and owner during pairing.
- *Malicious tag:* An attacker who fabricates counterfeit tracker tags that behave similarly to provider-approved tags. These tags exploit the OF network for tracking while evading detection mechanisms intended to prevent misuse such as stalking.
- *Malicious owner:* An attacker who owns a legitimate tag but deviates from the protocol and exploits the OF network to learn more information than it is privy to.
- *Temporary physical access:* An attacker who gains temporary physical access to a tag and can extract the tag's secrets.
- *Network adversary:* An attacker positioned between the owner and server (or finder and server) that can passively observe or actively modify protocol messages exchanged between them.
- *Service provider:* Since OF systems rely on centralized infrastructure, we consider the service provider itself as a potential adversary capable of passively observing or actively manipulating protocol interactions.

We allow for collusion between adversaries. We define the threat model for the security goals defined in the following section based on these adversary definitions.

The complete set of goals appears in Table 3, where for each goal, we list the corresponding adversary, and give a security definition.

Owner privacy. We begin by discussing definitions in the owner security category. These goals protect tag owners' location information from surveillance by the service provider and third party adversaries including RF and Bluetooth MITM.

Unlinkability ensures that a tag's broadcasts cannot be correlated over time by RF, Bluetooth MITM, network adversaries, or the service provider. In its absence, an attacker could track an owner based on the advertisements broadcast by its tag. For example, an attacker could place an RF receiver near the target's home and monitor recorded Bluetooth advertisements to learn whether the target

	Security Goal	Adversary	Definition
Owner security	**Unlinkability** [54, 22, 19, 20, 50, 51]	RF BT MITM Network Service provider	Any two BLE advertisements broadcast by the same tag that are at least Δ time apart are indistinguishable from advertisements broadcast by two different tags.
	Location confidentiality [54, 22, 19, 51]	Network Service provider	Any two location reports r_0 and r_1 generated by a finder device for distinct locations (ℓ_0, t) and (ℓ_1, t) are indistinguishable.
	Location report integrity [19]	Network Service provider	It is infeasible for an attacker to modify honestly generated location reports without being detected.
	Forward secrecy [19]	Network Temporary physical access	It is infeasible for an attacker to compromise the location confidentiality of reports submitted before tag compromise.
	Post-compromise security [19]	Network Temporary physical access	It is infeasible for an attacker to compromise the location confidentiality of reports submitted after tag compromise.
	Finder proximity [25]	Malicious finder	It is infeasible for an attacker to submit any report for a tag at location (ℓ, t) and have it be accepted by the corresponding owner without actually having been in the tag's proximity at time t.
Finder security	**Tag-owner binding** [54, 51, 25]	Malicious owner Malicious tag BT MITM Network	It is infeasible for an attacker that controls multiple owner accounts to register the same tag simultaneously under more than one account.
	Detectability [22]	Malicious owner Malicious tag	A finder device can efficiently determine and alert the user of suspected stalking if a sequence of lost-mode advertisements $\{\mathsf{adv}_0, \mathsf{adv}_1, \ldots, \mathsf{adv}_{\Delta_{\mathsf{detect}}}\}$ recorded at distinct locations $(\ell_0, t_0), (\ell_1, t_1), \ldots, (\ell_{\mathsf{detect}}, t_{\mathsf{detect}})$ originated from the same tag.
	Stalker identity integrity	Malicious owner Malicious tag	A finder device can efficiently retrieve the identity of the owner of a tag that was detected for potential stalking.
	Finder privacy [25]	Malicious owner Service provider	Any two location reports submitted by the same finder device are indistinguishable from reports submitted by two distinct finder devices.
	Advertisement unforgeability [54, 19]	Malicious tag	It is infeasible for an attacker to forge advertisements that are indistinguishable from ones produced by legitimate tags and retrieve location reports for them.

Table 3: Security goals for Offline Finding protocols and the corresponding threat model. Here we use "infeasible" and "indistinguishable" to mean computationally infeasible and computationally indistinguishable, respectively.

is at home. In the worst case, the attacker could set up a network of RF detectors and surveil the movements of large populations.

Location confidentiality ensures that neither the service provider nor network adversaries learn the location of the finder or the reported tag. In its absence, the service provider/network adversary effectively runs a mass-surveillance network.

Location report integrity prevents network adversaries and the service provider from modifying reports submitted by finders.

Forward secrecy ensures that even if an adversary compromises a tag's secrets, it cannot retroactively learn the tag's past location from reports submitted before tag compromise.

Post-compromise security ensures that an adversary cannot learn tag locations from reports submitted after tag compromise. Together, forward secrecy and post-compromise security prevent long-term privacy loss from a single compromise: the attacker learns at most a small window of locations, not the tag's full history.

Finder proximity ensures that only finders physically near a lost tag can submit any reports for it. Without it, malicious finders could remotely inject false reports for tags of their choice.Such an adversary could, for example, generate "evidence" that an honest user's tag was seen at a particular location, thereby framing them for being complicit in stalking. This property is especially important when considering stalker identity integrity (see below).

Finder security. We now discuss properties in the finder privacy category. These properties protect the location information of reporting finders from malicious owners with potentially malicious tags and the service provider. As we will see in Section 6, these properties are not achieved by most deployed systems and have received little academic attention, but are ones we believe are important to ensure the security of finders' locations.

Tag-owner binding ensures that a tag is paired with exactly one owner account throughout its life cycle, and is defined against the malicious owner, malicious tag, Bluetooth MITM, RF, and network adversaries. Without this property, an adversary could re-register an already registered tag, allowing them in the worst case, to track the original owner without their knowledge.

Detectability provides potential stalking victims a reliable way to link advertisements from a tag over time to recognize when an unknown tag is following them. Without detectability, a malicious owner/tag adversary could place their tag on their target's person or belongings and use the OF network to monitor their movements.

Stalker identity integrity enables a user that has been a target of unwanted tracking to—in collaboration with a judge or the

service provider—retrieve the identity of the tag owner. Then, appropriate action can be taken against the perpetrator. Without this property, victims might detect they are being tracked but have no way to identify or prosecute the perpetrator.

Finder privacy prevents the service provider and malicious owners from correlating different location reports submitted by the same finder. Without it, the malicious party could deanonymize finders and infer their movements based on the reports they submit. For instance, an attacker could place several tags along a particular geographic area and, by correlating the location reports submitted for different tags by the same finder, reconstruct the finder's trajectory. Although vendors often claim that finders remain anonymous, these claims have not been rigorously analyzed or formalized.

Advertisement unforgeability ensures that only tags approved by the provider can generate valid advertisements that owners can later use to retrieve their tag's location. In particular, it prevents malicious tag adversaries from using the OF network. Note that if a protocol achieves advertisement unforgeability, then it also trivially achieves detectability against malicious tags. Mayberry et al. [54] formalized this notion as beacon unforgeability and gave a construction based on partially blind signatures to achieve it.

Baseline properties. Of the properties defined in Table 3, we jointly refer to unlinkability, location confidentiality/integrity, tag-owner binding, detectability, and finder privacy as *baseline* goals. These are the core security and privacy goals of an OF system, and are targeted by most deployed protocols.

5.1 Unlinkability versus anti-abuse

Unlinkability is inherently in tension with detectability and stalker identity integrity. The former requires that broadcasts hide identifying information that the latter properties require to be revealed in some circumstances. As discussed in Section 4.4, some deployed protocols balance this tension by using distinct modes of operation—connected and lost—for the tag. In the connected mode, the tag changes its identifier every $\Delta_{\text{conn}} = 15$ minutes and in the lost mode every $\Delta_{\text{lost}} = 24$ hours. So in the lost mode, such protocols only achieve unlinkability for Δ_{lost} which is strictly weaker than unlinkability achieved in the connected mode. This deliberate weakening of achieved unlinkability allows users to detect stalking.

6 Security and privacy analysis

In this section, we use the definitions from Table 3 to evaluate the security of four major deployed protocols and six constructions proposed in academic papers.

We begin with the BasicOF protocol (Fig. 1) and provide intuition for why it satisfies all baseline security goals. Our intention is to show that a canonical protocol built from standard cryptographic building blocks achieves all baseline security goals. Then we evaluate deployed protocols. For each property a protocol fails to achieve, we detail the relevant attacks or design choices, citing prior work where applicable. Then we present the academic constructions. We describe the set of goals they target and provide intuition for how they achieve them. Table 4 provides a summary.

In Table 4, we evaluate unlinkability guarantees in both connected and lost modes by separately analyzing unlinkability for $\Delta = 15$ minutes and $\Delta = 24$ hours. Protocols that rely on distinct modes of operation to balance the privacy–detectability tension should therefore always achieve unlinkability for $\Delta = 24$ hours, and achieve unlinkability for $\Delta = 15$ minutes in connected mode.

Scope of analysis. Our analysis evaluates the cryptographic protocols underlying OF systems in isolation. In practice, despite achieving security properties at the cryptographic level, deployments may fail to achieve them overall due to factors outside the cryptographic protocol such as client authentication and static MAC addresses. We discuss such instances of when implementation-level factors weaken the achieved security in Section 6.1.

The BasicOF **construction**. Recall that BasicOF is constructed by canonically composing an authenticated key exchange protocol (AKE), a key derivation function (KDF), and a symmetric encryption scheme (SE). Tag-owner binding follows from the (authentication) security of AKE. Unlinkability follows from the key indistinguishability security of AKE and the PRF security of KDF. Location confidentiality and integrity follow from the IND-CPA and INT-CTXT security of SE [7]. Detectability against malicious owners follows from slower identifier rotation—trading off unlinkability—in the lost mode, allowing finder devices to check if the same lost-mode identifier (i.e., the same X_j) has been recorded at multiple locations over time. Since no identifying information about finders is included in location reports, BasicOF also achieves finder privacy.

We note that BasicOF makes blackbox use of KDF and SE but instantiates AKE with a variant of the signed Diffie-Hellman protocol. This reflects the fact that the pairing phase in OF cannot be captured using standard AKE syntax. While the latter is executed between two parties who mutually authenticate each other and derive a shared secret, the former involves three parties: the tag, the owner device, and the server, where the tag and owner authenticate to the server while establishing a shared key with each other.

Deployed protocols often implement this phase in ways that introduce vulnerabilities. The concrete instantiation of AKE in BasicOF serves to show that, in principle, the required security guarantees can still be achieved using known techniques.

6.1 Deployed protocols

All deployed protocols achieve 15-minute and 24h unlinkability, location confidentiality and integrity, forward secrecy, as well as post-compromise security against network adversaries, as protocol messages between online parties are sent over TLS. We omit the corresponding rows from Table 4.

Unlinkability (Δ = 15min). Apple achieves 15-min unlinkability against the server and RF adversaries only in connected mode, but does not achieve it against BT MITM adversaries. Recall that during pairing, the tag transmits (x, n_{tag}), where x is the tag's ephemeral secret and n_{tag} a random nonce; the phone responds with (Y, n_{owner}). Rotating keys are derived from $pk_0 = Y \cdot g^x$, n_{tag}, and n_{owner}. In theory, a passive BT MITM adversary can observe all messages exchanged between the tag and the owner during pairing, derive pk_0, and link advertisements (vulnerability V7 in [51]). However, in practice, mounting such attacks is highly non-trivial.

Google achieves unlinkability from RF adversaries and Bluetooth MITM adversaries in both connected and lost modes, but not from the server. Owners must upload precomputed advertisements so

	Security Goal	Deployed protocol						Academic proposal					
		BasicOF	Apple	Google	Samsung (default)	Samsung (E2EE)	Tile	PrivateFind [77]	SECROW [25]	Eddystone-EID [20]	BlindMy [54]	EBGHJ24 [22]	BcnNtf [19]
Owner security	**Unlinkability** [Δ = 15minutes]												
	Radio-frequency adv.	◐	◐	●	○	○	○	●	○	●	◐	●	●
	Bluetooth MITM	◐	○	●	○	○	○	●*	○	-	-	●	●
	Service provider	◐	◐	○	○	○	○	●	○	○	◐	●	○
	Unlinkability [Δ = 24hours]												
	Radio-frequency adv.	●	●	●	○	○	○	●	○	●	●	●	●
	Bluetooth MITM	●	○	●	○	○	○	●*	○	-	-	●	●
	Service provider	●	●	○	○	○	○	●	○	○	●	●	○
	Location confidentiality	●	●	●	○	●	○	●	●	○	●	●	●
	Location report integrity	●	●	●	○	●	○	●	●	○	●	●	●
	Forward secrecy	○	●	○	○	●	○	-	-	○	●	-	●
	Post-compromise security	○	●	○	○	●	○	-	-	○	●	-	●
	Finder proximity	○	○	○	○	○	○	●	●	-	-	-	-
Finder security	**Tag-owner binding**												
	Bluetooth MITM	●	●	●	●	●	●	○	●	-	●	-	-
	Malicious tag	●	●	●	●	●	●	○	●	-	●	-	-
	Malicious owner	●	●	○	○	○	●	○	●	-	●	-	-
	Detectability												
	Malicious tag	○	○	○	○	○	○	-	-	-	○	○	-
	Malicious owner	●	●	○	●	●	●*	-	-	-	●	●	-
	Stalker identity integrity												
	Malicious tag	○	○	○	○	○	○	-	-	-	○	-	-
	Malicious owner	○	●	○	○	○	●	-	-	-	●	-	-
	Finder Privacy												
	Malicious owner	●	●	●	●	●	●	●	●	-	●	-	-
	Service provider	●	●	●	●	●	●	●	●	-	●	-	-
	Advertisement unforgeability	○	○	○	○	○	●	-	-	-	●	-	-

Table 4: Summary of protocols and their achieved security properties. We use ● to denote that a protocol achieves a property, ◐ to denote that a property is satisfied only in connected mode, and ○ to indicate that a property is not achieved. We use - to indicate that a given property is not targeted by the corresponding academic protocol. When a property is achieved (or not achieved) for all adversaries in the threat model, we collapse all the adversaries into a single row.

that the server can preemptively match location reports to owners (c.f. Appendix C.2) at the cost of unlinkability.

In Samsung and Tile, the server stores the shared master key (c.f. Appendix B), so neither achieves unlinkability against the server. In Samsung (resp. Tile) advertisements recycle through 65, 535 (resp. 8,640) values, allowing an RF adversary to, in theory, passively capture and link advertisements. In practice [51] (resp [44]) shows that it would take an attacker 3 years (resp. 3 months) to execute such an attack. Moreover, recall that for tags in Samsung's E2EE mode, a finder queries the server for the tag's public key to encrypt the location. This key is long term, so the finder can use the returned public key to identify the tag.

Implementation-level weaknesses. Google's 15-min unlinkability is weakened by the use of static MAC addresses in the lost mode (c.f. Appendix D). Tile's 15-min unlinkability is largely diminished by its use of static MAC addresses at all times, allowing RF adversaries to trivially break the property.

Unlinkability (Δ = 24h). Apple achieves 24-hour unlinkability from the server and RF adversaries, but not from Bluetooth MITM adversaries due to the attack described above. Google provides 24-hour unlinkability from RF adversaries and Bluetooth MITM adversaries, but not from the server for reasons listed above. Samsung and Tile fail for the same reasons as before.

Location confidentiality and integrity. Finders in Apple, Google, and Samsung (E2EE mode) encrypt location reports end-to-end using an authenticated encryption scheme (c.f. Section 4), thereby achieving location confidentiality and integrity against the service provider. Tile and Samsung (default mode) do not encrypt location data (c.f. Section 4), and hence do not achieve these properties.

Forward secrecy and post-compromise security. Apple and Samsung (E2EE) achieve forward secrecy and post-compromise security. In Apple's protocol, a tag learns only the master public key during pairing, not the corresponding secret key (c.f. Appendix B). As a result, if an adversary compromises a tag, they can reconstruct all past and future public keys used to encrypt location reports, but not the decryption keys. In the Samsung protocol's E2EE mode, the key used for encrypting location reports is generated and stored by the owner (c.f. Appendix D), so tag compromise does not break location confidentiality of past or future reports.

By contrast, Google's protocol establishes a shared symmetric key between the tag and its owner (c.f. Appendix B), from which both tag identifiers and encryption/decryption keys are derived. Compromising a tag's secrets would allow the adversary to decrypt all past and future location reports. Tile and Samsung (default) trivially fail to provide either property.

Finder proximity. No deployed protocol achieves finder proximity by design. If an attacker knows a tag's identifier (by observing it or via an unlinkability attack), they can submit arbitrarily many location reports for the identifier without its presence.

Tag-owner binding. Apple and Tile achieve this against all adversaries by cryptographically authenticating the tag and the owner device during pairing, then storing a map between the owner account and the tag in their databases (see Section 4.1). AirTags can only be paired to one account; resetting the AirTag does not allow its registration with a new owner. Tile allows for device re-registration after reset. However, resetting a tag removes the associated owner-tag mapping. Google and Samsung use a similar process for pairing, but their protocols are vulnerable to attacks by malicious owners.

In Google, the server cannot check if a tag is already registered; tags are registered in Find Hub against a secret key (shared by all tags of the same model) and a set of precomputed advertisements. A malicious owner can therefore record another user's tag advertisements and re-register them as a "fake" tag (location report denial-of-service attack described in Section 6.5 of [10]).

Samsung attempts authenticating tags during registration, but their implementation is vulnerable to the following attack. An attacker can reset an already registered tag (say, tag_1), forcing it into the pairing phase. Because Samsung's key exchange occurs between the server and the tag, and a tag's long-term secret key is derived deterministically from its MAC address, a malicious owner device can mount an attack (V5 of [51]), impersonating a new tag (tag_2) to the server. The server records the attacker as the owner of tag_2, but the attacker actually pairs with tag_1, which was already registered to another user, breaking tag-owner binding.

Detectability. None of the deployed protocols achieve detectability against malicious tags. In all these protocols, finders cannot distinguish honest tag broadcasts from those by malicious tags (see below). Malicious tags can then mimic honest tags and rotate identifiers frequently in the lost mode and evade detection.

Apple and Samsung achieve detectability against malicious owners through automatic background scanning, and Tile only partially satisfies it since scans must be triggered manually (c.f. Section 4).

Google's Find Hub relies on the server to control tags' transition to the lost mode: a tag only enters lost mode if the server has not received recent reports from its owner (c.f. Appendix D). A malicious owner can evade detectability by repeatedly submitting fake location reports for a tag, thereby preventing its transition to lost mode and hence its detection.

Stalker identity integrity. None of the deployed protocols achieve stalker identity integrity against malicious tags as they fail to achieve detectability against malicious tags, and detectability is a pre-requisite for stalker identity integrity.

Apple achieves stalker identity integrity against malicious owners: it provides a mechanism for a stalking victim to recover partial identity of a detected tag's owner (e.g. first four digits of phone number or characters of an email address). Apple's approach is non-cryptographic: once the user locates the detected tag, they extract the tag's unique serial number and query Apple's server, which returns partial account information for the owner. Because the server maintains a binding between serial numbers and owners, the integrity of the returned identity is preserved.

Google implements a similar process, but as mentioned above, it does not achieve detectability against malicious owners, and hence also does not achieve stalker identity integrity.

Tile also claims to support stalker identification, but unlike Apple and Google, users cannot retrieve the information themselves—victims are required to involve law enforcement, after which Tile claims to provide the owner's identity. To the best of our knowledge, Samsung does not support any such mechanism.

Finder privacy. All deployed protocols achieve finder privacy against malicious owners and the service provider, since location reports do not contain identifying information about the finder.

Implementation-level weaknesses. Finders must authenticate to the provider before submitting location reports in all deployments. This allows the service provider to, in theory, link location reports to the finder, weakening this guarantee in practice. A malicious owner can further diminish the finder privacy of Apple's protocol by using metadata appended to reports by the server as a side channel to recover finder sub-trajectories [70].

Advertisement unforgeability. None of the deployed protocols except Tile achieve this property: by using standard advertisement formats with pseudorandom identifiers, they permit a malicious tag adversary to forge advertisements indistinguishable from genuine tag broadcasts. Prior work has shown that such attacks enable unauthorized use of the OF network and evasion of unwanted-tracking detection [37, 55, 11, 8, 10, 13].

In Tile, the server verifies that a tag is registered with the requesting owner before returning location reports associated with the tag. This prevents attacks using malicious tags, since they cannot be registered in the first place. Note that this check is possible because Tile's server maintains a mapping between tag identifiers and their owners (c.f. Appendix B), which comes at the cost of unlinkability.

Takeaway #1: None of the deployed protocols achieve all baseline goals due to: (i) design decisions (e.g., Samsung and Tile do not encrypt location reports), (ii) tensions between security properties (e.g., Apple and Google trade off unlinkability for detectability), (iii) vulnerable cryptographic implementation (e.g., Apple's and Samsung's key exchange protocols), or (iv) threats outside of their models (e.g., Google does not aim for unlinkability against the server).

Takeaway #2: Apple and Google achieve most of the owner security goals, but fail at achieving several finder security goals.

6.2 Academic proposals

Here, we describe and evaluate six academic proposals using our framework. Each proposal addresses only a subset of the goals defined in Table 3. Accordingly, we focus on the specific goals each proposal aims to achieve. We also discuss efficiency constraints that hinder practical deployment.

PrivateFind [77]. PrivateFind achieves unlinkability, location confidentiality and integrity, finder privacy, and finder proximity. It does not achieve tag-owner binding as there is no registration with the server; anyone with physical access to the tag can pair to it.

The pairing phase of PrivateFind establishes K_{shared} between the tag and the owner. The paper defines two ways of pairing: local and manufacturer-verified. The former is over an insecure channel which is susceptible to BT MITM attacks whereas the latter happens over an encrypted channel using a key provisioned to the tag at manufacture time. After pairing, the tag and owner derive ephemeral identifiers as $id_{i+1} \leftarrow \mathsf{KDF}(K_{\text{shared}}, id_i)$ every 15 minutes, achieving 15-minute unlinkability. Tags become discoverable only when out of Bluetooth range of the owner. A finder detecting a lost tag sends its location to the tag, which encrypts it under K_{shared} and returns it to the finder, who forwards the ciphertext to the server. As K_{shared} is only known to the owner and tag, location confidentiality and integrity are ensured.

PrivateFind achieves finder privacy by design: finders are not required to register accounts with the provider and the server omits any metadata in its reports. Thus, neither a malicious owner nor the service provider can identify which finder submitted a given report. Finally, as a finder must interact with the tag over BLE to report a location and replay attacks are prevented using counters in the AEAD scheme, PrivateFind also achieves finder proximity.

Deployment challenge. Requiring tags to be interactive in lost mode adds significant computation and communication overhead, thereby reducing battery life.

SECROW [25]. SECROW achieves tag-owner binding, location confidentiality and integrity, finder proximity, and finder privacy. Each tag and owner device is provisioned a long-term signing key pair and a unique identifier; owner and finder devices additionally have a Trusted Execution Environment (TEE) that can generate attested ephemeral signing keys. During pairing, both the tag and the owner device authenticate themselves to the service provider using their long term private keys and establish K_{shared}. This together with the fact that the server stores a tags-to-owners mapping guarantees tag-owner binding.

The tag's static identifier is accessible to finders. During the location-reporting phase, a finder authenticates itself to the service provider using one of its ephemeral signing keys generated by the TEE, hence achieving finder privacy. The server returns a nonce, which the finder forwards to the tag along with its own location. The tag encrypts the location under K_{shared} using an authenticated encryption scheme, signs the server's nonce, and returns the ciphertext, signature, its identifier to the server via the finder. The server verifies the signature and stores the report. As locations are encrypted with K_{shared}, SECROW achieves location confidentiality and integrity. Moreover, since location reporting requires active interactions with the tag over BLE, it achieves finder proximity. Due to the use of static identifiers, SECROW does not achieve unlinkability.

Deployment challenge. The security of SECROW requires a TEE on all owner and finder devices, limiting large scale deployment.

Eddystone-EID [20]. Eddystone-EID is designed for a setting that is slightly different from that of OF systems: in addition to an identifier, tags broadcast some telemetry data (e.g., battery status, temperature). The protocol guarantees unlinkability against RF adversaries and confidentiality of the tag's telemetry data.

During pairing, the tag and the owner establish two symmetric keys: one for identifier generation and another for end-to-end encryption of telemetry data. The owner device shares the identifier generation key with the server. Since the server knows the identifier generation key, there is no unlinkability against the server. The paper does not detail the actual key exchange protocol, so we mark security against BT MITM as being out of scope. The server uses this to create an identifier-to-owner mapping by generating all identifiers for a tag. After pairing, the tag broadcasts ephemeral identifiers generated by applying a KDF to the identifier-generation key and the current epoch along with the AEAD encryption of its telemetry data under the end-to-end encryption key. Finders append their location to the tag's broadcast and forward them to the server, which forwards it to the intended owner. Since the location data is not encrypted, it does not guarantee location confidentiality or integrity. While Eddystone-EID does not introduce deployment challenges, it does not address or achieve most goals.

BlindMy [54]. Apple's unwanted-tracking alerts were shown to be easily bypassed by programming a malicious tag to emulate multiple AirTags, producing lost messages that nearby finders would accept and report [55]. An attacker could then query these reports to track a victim. BlindMy cryptographically prevents such exploits by achieving advertisement unforgeability, while inheriting other security guarantees from Apple's protocol.

BlindMy uses partially blind signatures to achieve advertisement unforgeability. It modifies Apple's pairing and location retrieval phase. During pairing, in addition to establishing K_{shared} as in Apple's protocol, the owner precomputes all identifiers that the tag will broadcast over its lifetime. Then, the owner device sends the hash of these identifiers ("blinded" so the server does not learn them) along with their corresponding epochs to the server. The server verifies that the epochs are valid and produces signatures over each (blinded identifier hash, epoch) pair and returns these signatures to the owner device. The owner device unblinds the signatures and stores all (identifier, unblinded signature) pairs, and

shares all the identifiers with the tag. The paper does not detail how identifiers are shared with the tag, so we mark unlinkability against BT MITM adversaries as out of scope.

When the owner requests reports, it sends the signed identifier hashes and epochs to the server. The server verifies these before returning matching reports, ignoring any invalid requests. The owner then decrypts the reports analogously to Apple's protocol.

Deployment challenge. The pairing and location report download phases are significantly different from those of existing protocols, making their integration non-trivial.

EBGHJ24 [22]. As pointed out in Section 4.4, unlinkability and detectability are inherently conflicting goals and typically, deployments trade off unlinkability in the lost mode for detectability. The OF protocol proposed in [22], that we refer to as EBGHJ24, improves this trade-off; it simultaneously guarantees strong 15-minute unlinkability and detectability against malicious owners. It inherits location confidentiality and integrity by following Apple's protocol.

To achieve strong unlinkability and privacy, [22] introduce a new cryptographic primitive called multi-dealer secret sharing (MDSS). Analogous to traditional secret sharing [64]—where a dealer splits a secret into shares such that a threshold number of them can recover the initial secret—MDSS sets a threshold for reconstruction. MDSS additionally guarantees that even when multiple dealers share different secrets, a receiver can recover these secrets from a set of mixed shares. Using MDSS as a building block, [22] propose an OF construction that achieves both unlinkability and detectability.

Deployment challenge. While [22] implemented their construction on hardware to demonstrate feasibility, the advertisement size in their construction requires the tag to broadcast twice as many times as in Apple's protocol, reducing its battery life by half.

BcnNtf [19]. BcnNtf achieves unlinkability against RF and BT MITM adversaries, but not against the server, for similar reasons as Google. It also achieves location confidentiality and integrity, and forward and post-compromise security. The protocol also defines an integrity notion which is similar to advertisement unforgeability but does not consider security against malicious tags.

BcnNtf operates in the same setting as Eddystone-EID, but instead of separately broadcasting its identifier and encrypted telemetry, BcnNtf encapsulates telemetry within the identifier itself. It inherits the same unlinkability properties as Eddystone-EID. The identifier also serves as a public key, which finders use to encrypt location data, thereby ensuring confidentiality and integrity.

To achieve forward and post-compromise security, [19] introduces XDHIES, an extension of the DHIES [1] scheme that provides inherent forward and backward security. Crucially, BcnNtf separates keys for identifier generation from those for decrypting location reports, and tags only learn the former. Then, tag compromise reveals public keys but not decryption keys, preserving confidentiality and integrity of reports, as in Apple's protocol.

In the paper, the authors suggest that the anti-abuse measures deployed by Apple or Google can be integrated with their protocol, but it is not detailed in the paper. Since it is unclear which of Apple's or Google's guarantees it would inherit, in Table 4 we mark the anti-abuse properties as not targeted by BcnNtf.

Deployment challenge. BcnNtf deviates significantly from existing deployments to achieve advanced properties, making it hard to achieve backwards compatible integration.

Takeaway #3: Academic proposals individually achieve many of the additional goals that deployed protocols fail to achieve. In principle, combining these techniques could yield a construction that achieves the full set of desired properties. However, the proposals either do not meet the battery and bandwidth constraints of tags deployed in practice or introduce compatibility issues with current implementations, rendering deployment infeasible.

7 Challenges beyond the protocol level

We now summarize works that identify risks and challenges extending beyond the protocol layer. Appendix E provides a full discussion.

BLE security analyses. Even when OF protocols achieve cryptographic unlinkability, physical-layer properties often leak identifying information, enabling large-scale device fingerprinting and tracking. Static identifiers, timing behavior, and hardware characteristics can all be exploited to link broadcasts. Prior work has demonstrated wide-area fingerprinting of Tile devices [82], linkage of BLE advertisements to Bluetooth Classic frames to obtain unique device IDs [52], and tracking of BLE devices via allow-list communication patterns even with randomized MACs [81]. Other attacks include UWB distance-reduction [47], the Battery Insertion Attack, which exploits timing patterns to break unlinkability [50], and large-scale exploitation of BLE address-type inconsistencies in Apple's Find My network [16].

Unwanted tracking detection in practice. Research shows that IoT devices, including tags, are abused to cause interpersonal harm [65, 66, 76, 14]. Studies focusing on tracker tags in particular examine how existing anti-stalking tools operate [36, 73, 63], revealing weaknesses ranging from fundamental flaws in notification logic to the lack of cross-vendor support. Additional work demonstrates attacks that suppress safety notifications altogether [63, 55]. In response, researchers have proposed new applications and techniques for detecting malicious trackers [36, 12, 56, 18, 21]. Beyond the logic of these tools themselves, several works performed user studies to analyze the usability of these anti-stalking features [39, 72, 26, 27], finding many usability issues like desensitization to notifications.

8 Discussion

In this section, we reflect on the lessons learned from our analysis. We highlight recurring challenges, tensions between privacy and abuse-prevention, and gaps in current designs that leave room for both abuse and unintended privacy harms. We identify five main themes that we believe should guide future work on OF systems.

Privacy versus abuse prevention. Unlinkability has emerged as one of the most difficult properties to achieve in practice. Even though most protocols explicitly aim to achieve unlinkability, both protocol-level and physical-layer attacks repeatedly break this property. Unlinkability often complicates stalker detection, sometimes creating more opportunities for abuse than it prevents.

Achieving unlinkability and detectability simultaneously at the protocol level is challenging, with many designs sacrificing one

to preserve the other. Among the systems we evaluated, only the construction of [22] achieves both properties, but at the cost of higher communication overhead on the tag, making it impractical for real-world deployment. Unlinkability is further undermined by physical-layer vulnerabilities in BLE and Bluetooth that allow efficient tag fingerprinting. These attacks are difficult to mitigate due to specification complexity, wide deployment, and legacy constraints.

Future work suggestion: We suggest that detectability is a critical and pressing property to target for improvements given the ongoing, widespread use of tags for stalking. Ideally, the community would work to develop global standards for detectability of any location-enabled device that could be used for surreptitious tracking, building off and improving nascent efforts [46, 23]. Towards informing such standards, future work could focus on new design points that target better detectability while preserving, or even modestly relaxing, current unlinkability mitigations.

Accountability. Detectability without accountability is insufficient. Despite their concerning use for harm, most OF protocols lack accountability features that would enable reporting or banning misbehaving users.

Achieving accountability requires a reporting mechanism that lets a victim notify the platform or law enforcement about a detected tag. The platform/law enforcement can then work with the victim to identify the tag's owner. This is technically challenging due to conflicts with other goals. For example, a mechanism for retrieving a stalker's identity could also potentially allow the service provider or law enforcement to deanonymize owners of arbitrary tags, compromising privacy.

Future work suggestions: We see potential opportunity in exploring cryptographic accountability mechanisms. Prior work on message franking—a privacy-preserving accountability mechanism for end-to-end encrypted messengers—demonstrates that privacy-accountability tensions can be balanced using cryptographic tools [35, 74, 45], and cryptographic proof-of-stalking extensions for OF protocols seems a plausible future direction. In addition, future work should conduct further research on non-cryptographic accountability mechanisms: reporting that uses static hardware-defined identifiers, digital stalking reports that cannot be forged if the tag's software is intact, and supporting procedures and policies.

Interoperability. A recurring challenge across all protocols we studied is the lack of interoperability by design. Before Apple and Google drafted the DULT [46] specification, each vendor's ecosystem was siloed: a user could only detect unwanted trackers if they belonged to the same vendor network. While DULT has improved cross-vendor detection between Android and Apple, vendors like Tile are still confined to only being able to detect their own tags.

Future work suggestions: For user safety, we argue that interoperability must be a design priority. This means that we need internet standards, akin to TLS, that should be adopted by all potentially privacy-damaging devices—not just tracker tags, but also other wireless beacons and emerging IoT hardware. Building towards this vision would require generic constructions, and cooperation across vendors, regulators, and standards bodies. Ideally this would end up win-win: in addition to privacy and safety improvements, interoperable finder networks would improve functionality.

Formal analyses. Several cryptographic analyses of OF protocols exist, but most focus on a subset of properties in isolation. This is understandable given the complexity of these systems, which pursue multiple, often orthogonal, goals. Isolating individual properties allows for fine-grained analysis, but leaves open whether guarantees hold when protocols are modeled and analyzed comprehensively.

In addition, some properties such as tag–owner binding do not map neatly onto existing cryptographic primitives. Classical abstractions from key-exchange security are insufficient, since OF introduces a unique paradigm that involves not only the tag and owner but also the server, which authenticates entities.

Future work suggestions: We call for the development of new cryptographic models and abstractions tailored to OF protocols, as well as analyses that explicitly address how these properties interact and compose in practice.

OF extensions for data back-hauling. A small but growing body of work explores how OF networks can be repurposed for opportunistic data backhaul. Prior works demonstrate that Apple's Find My can support arbitrary data muling [8, 62] as well as distributed sensing via crafted BLE advertisements [34]. A more recent academic proposal, Nebula [75], explicitly addresses the privacy risks of such piggybacking, introducing a decentralized architecture that prevents the service provider from tracing relay devices while enabling payments, compensation, and spam protections.

Future work suggestions: As the scope of data piggybacked onto OF-like networks expands, so too do the associated privacy threats. Future work should carefully analyze how these systems compose and how the privacy risks inherent in OF networks may propagate to higher-level protocols and applications.

9 Limitations

Our analysis is based on the academic literature and vendor specifications/statements in our corpus. Our discussion of the deployed protocols reflects their current state: (i) academic analysis of these protocols is recent (Apple: 2021-2025, Samsung: 2024-2025, Google: 2025, Tile: 2026) and (ii) most attacks identified result from design choices or hardware vulnerabilities, rather than implementation mistakes. In doing our analysis we searched for statements made by vendors about changes to their respective protocols. There is no public evidence that vendors have changed their protocol designs substantively. However, we did not experimentally reconfirm details for all deployed protocols.

Relatedly, our assertions that a protocol achieves a given property are supported by prior analyses. Academic proposals include security proofs, which we used to guide Table 4. For deployed protocols, prior work provides either provable security analysis (e.g., [54] proves tag-owner binding, location confidentiality and integrity, and unlinkability of Apple's protocol, [20] proves location confidentiality and integrity of Google's protocol) or formal verification (e.g., [51] proved the location confidentiality and integrity of Samsung's protocol when E2EE is enabled). We did not perform independent provable security or formal analysis to reconfirm results.

Acknowledgments

Generative AI-based tools were used to improve flow and correct typos, grammatical errors, and awkward phrasing.

References

[1] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. 1999. DHAES: an encryption scheme based on the diffie-hellman problem. Cryptology ePrint Archive, Paper 1999/007. (1999). https://eprint.iacr.org/1999/007.

[2] Mohammad Afaneh. 2020. Bluetooth addresses & privacy in bluetooth low energy. (2020). https://novelbits.io/bluetooth-address-privacy-ble/.

[3] Apple. 2020. Find my network accessory specification. https://www.frandroid.com/wp-content/uploads/2020/06/Find_My_network_accessory_protocol_specification.pdf. Accessed: 2025-04-13. (2020).

[4] Apple. 2025. Legal process guidelines. https://www.apple.com/legal/privacy/law-enforcement-guidelines-us.pdf. Accessed: 2025-04-13. (2025).

[5] Apple. [n. d.] One app to find it all. https://www.apple.com/icloud/find-my/. ().

[6] Apple. 2025. What to do if you get an alert that an airtag, set of airpods, find my network accessory, or compatible bluetooth location-tracking device is with you. https://support.apple.com/en-us/119874. Accessed: 2025-04-13. (2025).

[7] Mihir Bellare and Chanathip Namprempre. 2000. Authenticated encryption: relations among notions and analysis of the generic composition paradigm. Asiacrypt. (2000). https://eprint.iacr.org/2000/025.

[8] Alex Bellon, Alex Yen, and Pat Pannuto. 2023. Tagalong: a free, wide-area data-muling service built on the airtag protocol. In *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems* (SenSys '22). Association for Computing Machinery, Boston, Massachusetts, 782–783. ISBN: 9781450398862. DOI: 10.1145/3560905.3568085.

[9] Lindsey Bever. 2022. She tracked her boyfriend using an airtag — then killed him, police say. (June 2022). https://www.washingtonpost.com/nation/2022/06/11/apple-airtag-murder-boyfriend-indianapolis-morris/.

[10] Leon Böttger, Alexander Matern, Dennis Arndt, and Matthias Hollick. 2025. Okay google, where's my tracker? security, privacy, and performance evaluation of google's find my device network. *Proceedings on Privacy Enhancing Technologies*.

[11] Fabian Bräunlein. [n. d.] Send my: arbitrary data transmission via apple's find my network. (). https://positive.security/blog/send-my.

[12] Jimmy Briggs and Christine Geeng. 2022. Ble-doubt: smartphone-based detection of malicious bluetooth trackers. In *2022 IEEE Security and Privacy Workshops (SPW)*, 208–214. DOI: 10.1109/SPW54247.2022.9833870.

[13] Lukas Burg, Max Granzow, Alexander Heinrich, and Matthias Hollick. 2022. Openhaystack mobile - tracking custom find my accessories on smartphones. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (WiSec '22). Association for Computing Machinery, San Antonio, TX, USA, 277–279. ISBN: 9781450392167. DOI: 10.1145/3507657.3529655.

[14] Rose Ceccio, Sophie Stephenson, Varun Chadha, Danny Yuxing Huang, and Rahul Chatterjee. 2023. Sneaky spy devices and defective detectors: the ecosystem of intimate partner surveillance with covert devices. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, (Aug. 2023), 123–140. ISBN: 978-1-939133-37-3. https://www.usenix.org/conference/usenixsecurity23/presentation/ceccio.

[15] Hartley Charlton. 2021. Apple's airtag item trackers increasingly linked to criminal activity. (Dec. 2021). https://www.macrumors.com/2021/12/31/airtag-increasingly-linked-to-crime/.

[16] Junming Chen, Xiaoyue Ma, Lannan Luo, and Qiang Zeng. 2025. Tracking you from a thousand miles away! turning a bluetooth device into an apple airtag without root privileges. In *Proceedings of the 34th USENIX Security Symposium*. Short Presentation, Distinguished Artifact Award Winner.

[17] Samantha Cole. 2022. Police records show women are being stalked with apple airtags across the country. (Apr. 2022). https://www.vice.com/en/article/y3vj3y/apple-airtags-police-reports-stalking-harassment.

[18] Dylan Conklin, Primal Pappachan, and Roberto Yus. 2025. Bl(u)e crab: a user-centric framework for identifying suspicious bluetooth trackers. In *2025 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, 570–572. DOI: 10.1109/PerComWorkshops65533.2025.00131.

[19] Liron David, Omer Berkman, Avinatan Hassidim, David Lazarov, Yossi Matias, and Moti Yung. 2025. Extended Diffie-Hellman Encryption for Secure and Efficient Real-Time Beacon Notifications. In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, Los Alamitos, CA, USA, (May 2025), 4406–4418. DOI: 10.1109/SP61157.2025.00230.

[20] Liron David, Avinatan Hassidim, Yossi Matias, Moti Yung, and Alon Ziv. 2022. Eddystone-eid: secure and private infrastructural protocol for ble beacons. *IEEE Transactions on Information Forensics and Security*, 17, 3877–3889. DOI: 10.1109/TIFS.2022.3214074.

[21] Tess Despres, Noelle Davis, Prabal Dutta, and David Wagner. 2023. Detagtive: linking macs to protect against malicious ble trackers. In *Proceedings of the Second Workshop on Situating Network Infrastructure with People, Practices, and Beyond*.

[22] Harry Eldridge, Gabrielle Beck, Matthew Green, Nadia Heninger, and Abhishek Jain. 2024. Abuse-resistant location tracking: balancing privacy and safety in the offline finding ecosystem. In *USENIX Security Symposium*.

[23] C. Fossaceca and E. Rescorla. 2024. Finding tracking tags. IETF working group. (2024). https://www.ietf.org/archive/id/draft-ietf-dult-finding-00.html.

[24] Christian Frank, Philipp Bolliger, Christof Roduner, and Wolfgang Kellerer. 2007. Objects calling home: locating objects using mobile phones. In *Proceedings of the 5th International Conference on Pervasive Computing* (PERVASIVE'07). Springer-Verlag, Toronto, Canada, 351–368. ISBN: 9783540720362.

[25] Chinmay Garg, Aravind Machiry, Andrea Continella, Christopher Kruegel, and Giovanni Vigna. 2021. Toward a secure crowdsourced location tracking system. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (WiSec '21). Association for Computing Machinery, Abu Dhabi, United Arab Emirates, 311–322. ISBN: 9781450383493. DOI: 10.1145/3448300.3467821.

[26] Daniel Gerhardt, Matthias Fassl, Carolyn Guthoff, Adrian Dabrowski, and Katharina Krombholz. 2025. Airtag-facilitated stalking protection: evaluating unwanted tracking notifications and tracker locating features. In *34nd USENIX Security Symposium (USENIX Security 25)*. USENIX Association, Seattle, WA, (Aug. 2025). https://www.usenix.org/conference/usenixsecurity25/presentation/gerhardt.

[27] Daniel Gerhardt, Matthias Fassl, and Katharina Krombholz. 2024. Poster: will our colleagues detect the airtag? let's check (consensually).

[28] Thomas Germain. 2024. Alleged stalking victims accuse tile of advertising its devices as women trackers. Accessed: 2024-10-21. (Aug. 2024). https://gizmodo.com/tile-stalking-lawsuit-advertising-amazon-1850743154.

[29] Mark Gibbs. 2013. Phone halo's trackr devices make it harder to lose your stuff. *Network World*, (Nov. 2013). https://www.networkworld.com/article/747102/security-phone-halo-s-trackr-devices-make-it-harder-to-lose-your-stuff.html.

[30] Google. 2024. 5 ways to use android's new find my device. (2024). https://blog.google/products/android/android-find-my-device/.

[31] Google. [n. d.] Find hub. https://www.google.com/android/find/. ().

[32] Google. 2025. Find hub network accessory specification (fmdn). https://developers.google.com/nearby/fast-pair/specifications/extensions/fmdn. Google Fast Pair documentation. (2025).

[33] Google. 2025. Google fast pair service: introduction. https://developers.google.com/nearby/fast-pair/specifications/introduction. Google Fast Pair documentation. (2025).

[34] Max Granzow, Alexander Heinrich, Matthias Hollick, and Marco Zimmerling. 2024. Poster: leveraging apple's find my network for large-scale distributed sensing. In *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services* (MOBISYS '24). Association for Computing Machinery, Minato-ku, Tokyo, Japan, 666–667. ISBN: 9798400705816. DOI: 10.1145/3643832.3661412.

[35] Paul Grubbs, Jiahui Lu, and Thomas Ristenpart. 2017. Message franking via committing authenticated encryption. In *Advances in Cryptology – CRYPTO*.

[36] Alexander Heinrich, Niklas Bittner, and Matthias Hollick. 2022. 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*.

[37] Alexander Heinrich, Milan Stute, and Matthias Hollick. 2021. Openhaystack: a framework for tracking personal bluetooth devices via apple's massive find my network. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*.

[38] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. 2021. Who can find my devices? security and privacy of apple's crowd-sourced bluetooth location tracking system. *Proceedings on Privacy Enhancing Technologies*.

[39] Alexander Heinrich, Leon Würsching, and Matthias Hollick. 2024. Please unstalk me: understanding stalking with bluetooth trackers and democratizing anti-stalking protection. *Proceedings on Privacy Enhancing Technologies*.

[40] Apple Inc. 2021. Apple introduces airtag. (2021). https://www.apple.com/newsroom/2021/04/apple-introduces-airtag/.

[41] Jon Keegan and Alfred Ng. 2021. The popular family safety app life360 is selling precise location data on its tens of millions of users. (Dec. 2021). https://themarkup.org/privacy/2021/12/06/the-popular-family-safety-app-life360-is-selling-precise-location-data-on-its-tens-of-millions-of-user.

[42] Jon Keegan and Alfred Ng. 2021. There's a multibillion-dollar market for your phone's location data. (Sept. 2021). https://themarkup.org/privacy/2021/09/30/theres-a-multibillion-dollar-market-for-your-phones-location-data.

[43] Akshaya Kumar, Carolina Ortega Pérez, Joseph Jaeger, Thomas Ristenpart, and Michael A. Specter. 2026. Pets-26-sok-artifact. https://github.com/cortegap/pets-26-sok-artifact. (2026).

[44] Akshaya Kumar, Anna Raymaker, and Michael Specter. 2025. Security and privacy analysis of tile's location tracking protocol. https://arxiv.org/abs/2510.00350. (2025). https://arxiv.org/abs/2510.00350 arXiv: 2510.00350 [cs.CR].

[45] Junzuo Lai, Gongxian Zeng, Zhengan Huang, Siu Ming Yiu, Xin Mu, and Jian Weng. 2023. Asymmetric group message franking: definitions and constructions. In *Advances in Cryptology – EUROCRYPT 2023*. Carmit Hazay and Martijn Stam, (Eds.) Springer Nature Switzerland, Cham, 67–97. ISBN: 978-3-031-30589-4.

[46] Brent Ledvina, David Lazarov, Ben Detwiler, and Siddika Parlak Polatkan. 2024. Detecting unwanted location trackers accessory protocol. IETF working group. (2024). https://www.ietf.org/archive/id/draft-ietf-dult-accessory-protocol-00.html.

[47] Patrick Leu, Giovanni Camurati, Alexander Heinrich, Marc Roeschlin, Claudio Anliker, Matthias Hollick, Srdjan Capkun, and Jiska Classen. 2022. Ghost peak: practical distance reduction attacks against HRP UWB ranging. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, (Aug. 2022), 1343–1359. ISBN: 978-1-939133-31-1. https://www.usenix.org/conference/usenixsecurity22/presentation/leu.

[48] Life360. 2022. How to handle unwanted tracking. https://support.life360.com/hc/en-us/articles/30582987898135-How-to-Handle-Unwanted-Tracking. (2022).

[49] life360. [n. d.] Tile tracker. https://www.life360.com/tile-trackers. ().

[50] David Liron, Avinatan Hassidim, Yossi Matias, and Moti Yung. 2025. The battery insertion attack: is periodic pseudo-randomization sufficient for beacon privacy? *Proceedings on Privacy Enhancing Technologies*.

[51] Xiaofeng Liu, Chaoshun Zuo, Qinsheng Hou, Pengcheng Ren, Jianliang Wu, Qingchuan Zhao, and Shanqing Guo. 2025. A thorough security analysis of ble proximity tracking protocols. In *34th USENIX Security Symposium (USENIX Security 25)*. USENIX Association, (Aug. 2025).

[52] Norbert Ludant, Tien D. Vo-Huu, Sashank Narain, and Guevara Noubir. 2021. Linking bluetooth le & classic and implications for privacy-preserving bluetooth-based protocols. In *2021 IEEE Symposium on Security and Privacy (SP)*, 1318–1331. DOI: 10.1109/SP40001.2021.00102.

[53] Ryan Mac and Kashmir Hill. 2021. Are apple airtags being used to track people and steal cars? (Dec. 2021). https://www.nytimes.com/2021/12/30/technology/apple-airtags-tracking-stalking.html.

[54] Travis Mayberry, Erik-Oliver Blass, and Ellis Fenske. 2023. Blind my–an improved cryptographic protocol to prevent stalking in apple's find my network. *Proceedings on Privacy Enhancing Technologies*.

[55] Travis Mayberry, Ellis Fenske, Dane Brown, Jeremy Martin, Christine Fossaceca, Erik C Rye, Sam Teplov, and Lucas Foppe. 2021. Who tracks the trackers? circumventing apple's anti-tracking alerts in the find my network. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*.

[56] Katharina O. E. Müller, Louis Bienz, Bruno Rodrigues, Chao Feng, and Burkhard Stiller. 2023. Homescout: anti-stalking mobile app for bluetooth low energy devices. In *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, 1–9. DOI: 10.1109/LCN58197.2023.10223406.

[57] Inc. Nut Technology. 2025. Nutfind: official online store for nut brand smart tracker series. (2025). https://www.nutfind.com/.

[58] Thomas Roth, Fabian Freyer, Matthias Hollick, and Jiska Classen. 2022. Airtag of the clones: shenanigans with liberated item finders. In *2022 IEEE Security and Privacy Workshops (SPW)*. IEEE.

[59] Samsung. 2025. How to disable smarttag 2 location tracking? https://www.samsung.com/ae/support/mobile-devices/how-to-disable-smarttag-2-location-tracking/. Accessed: 2025-08-19. (2025).

[60] Samsung. [n. d.] Smartthings find. https://www.samsung.com/levant/apps/smartthings-find/. ().

[61] Ltd. Samsung Electronics Co. 2021. Introducing the new galaxy smarttag+: the smart way to find lost items. (2021). https://news.samsung.com/global/introducing-the-new-galaxy-smarttagplus-the-smart-way-to-find-lost-items.

[62] Spanddhana Sara, Moteen Shah, Dhairya Shah, Rajashekar Reddy Chinthalapani, and Ambuj Varshney. 2024. Gatehaul: a gateway architecture using backhauling networks to address the connectivity challenges of embedded systems. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (ACM MobiCom '24). Association for Computing Machinery, Washington D.C., DC, USA, 1811–1813. ISBN: 9798400704895. DOI: 10.1145/3636534.3698868.

[63] Narmeen Shafqat, Nicole Gerzon, Maggie Van Nortwick, Victor Sun, Alan Mislove, and Aanjhan Ranganathan. 2023. Track you: a deep dive into safety alerts for apple airtags. *Proceedings on Privacy Enhancing Technologies*.

[64] Adi Shamir. 1979. How to share a secret. *Commun. ACM*, 22, 11, (Nov. 1979), 612–613. DOI: 10.1145/359168.359176.

[65] Sophie Stephenson, Majed Almansoori, Pardis Emami-Naeini, and Rahul Chatterjee. 2023. "it's the equivalent of feeling like you're in Jail": lessons from firsthand and secondhand accounts of IoT-Enabled intimate partner abuse. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, (Aug. 2023), 105–122. ISBN: 978-1-939133-37-3. https://www.usenix.org/conference/usenixsecurity23/presentation/stephenson-lessons.

[66] Sophie Stephenson, Majed Almansoori, Pardis Emami-Naeini, Danny Yuxing Huang, and Rahul Chatterjee. 2023. Abuse vectors: a framework for conceptualizing IoT-Enabled interpersonal abuse. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, Anaheim, CA, (Aug. 2023), 69–86. ISBN: 978-1-939133-37-3. https://www.usenix.org/conference/usenixsecurity23/presentation/stephenson-vectors.

[67] Google Support. [n. d.] Find unknown trackers. https://support.google.com/android/answer/13658562?hl=en. ().

[68] CBS Chicago Team. 2023. Man killed girlfriend after she removed airtag he'd secretly placed in her car, prosecutors say. (July 2023). https://www.cbsnews.com/chicago/news/armoni-henry-charged-murder-jailene-flowers-marianos-evergreen-park/.

[69] FOX 7 Austin Digital Team. 2023. Texas man uses apple airtag to track down stolen truck, then shoots, kills suspect: police. (Apr. 2023). https://www.fox7austin.com/news/san-antonio-texas-apple-airtag-stolen-truck-suspect-killed-police.

[70] Leonardo Tonetto, Andrea Carrara, Aaron Yi Ding, and Jörg Ott. 2022. Where is my tag? unveiling alternative uses of the apple findmy service. In *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 396–405. DOI: 10.1109/WoWMoM54355.2022.00059.

[71] Cube Tracker. 2025. Cube tracker: real-time gps tracking solutions. (2025). https://cubetracker.com/?srsltid=AfmBOortcXDdxy3BBR-DSTKWhLiFIkmG42Hbc8nAUGaQNxSqDPmutdv9.

[72] Kieron Ivy Turk and Alice Hutchings. 2024. Stop following me! evaluating the malicious uses of personal item tracking devices and their anti-stalking features. In *Proceedings of the 2024 European Symposium on Usable Security* (EuroUSEC '24). Association for Computing Machinery, 277–289. ISBN: 9798400717963. DOI: 10.1145/3688459.3688477.

[73] Kieron Ivy Turk, Alice Hutchings, and Alastair R Beresford. 2023. Can't keep them away: the failures of anti-stalking protocols in personal item tracking devices. In *Cambridge International Workshop on Security Protocols*. Springer, 78–88.

[74] Nirvan Tyagi, Paul Grubbs, Julia Len, Ian Miers, and Thomas Ristenpart. 2019. Asymmetric message franking: content moderation for metadata-private end-to-end encryption. In *Advances in Cryptology – CRYPTO 2019*. Alexandra Boldyreva and Daniele Micciancio, (Eds.) Springer International Publishing, Cham, 222–250. ISBN: 978-3-030-26954-8.

[75] Jean-Luc Watson, Tess Despres, Alvin Tan, Shishir G. Patil, Prabal Dutta, and Raluca Ada Popa. 2024. Nebula: a privacy-first platform for data backhaul. In *2024 IEEE Symposium on Security and Privacy (SP)*, 3184–3202. DOI: 10.1109/SP54263.2024.00092.

[76] Miranda Wei, Eric Zeng, Tadayoshi Kohno, and Franziska Roesner. 2022. Anti-Privacy and Anti-Security advice on TikTok: case studies of Technology-Enabled surveillance and control in intimate partner and Parent-Child relationships. In *Eighteenth Symposium on Usable Privacy and Security (SOUPS 2022)*. USENIX Association, Boston, MA, (Aug. 2022), 447–462. ISBN: 978-1-939133-30-4. https://www.usenix.org/conference/soups2022/presentation/wei.

[77] Mira Weller, Jiska Classen, Fabian Ullrich, Denis Waßmann, and Erik Tews. 2020. Lost and found: stopping bluetooth finders from leaking private information. In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks* (WiSec '20). ACM, (July 2020). DOI: 10.1145/3395351.3399422.

[78] Martin Woolley. 2015. Bluetooth technology protecting your privacy. (2015). https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy/.

[79] Martin Woolley. 2022. The bluetooth low energy primer. https://www.bluetooth.com/bluetooth-le-primer/. Bluetooth SIG, (2022).

[80] Tingfeng Yu, James Henderson, Alwen Tiu, and Thomas Haines. 2024. Security and privacy analysis of samsung's Crowd-Sourced bluetooth location tracking system. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, Philadelphia, PA, (Aug. 2024), 5449–5466. ISBN: 978-1-939133-44-1. https://www.usenix.org/conference/usenixsecurity24/presentation/yu-tingfeng.

[81] Yue Zhang and Zhiqiang Lin. 2022. When good becomes evil: tracking bluetooth low energy devices via allowlist-based side channel and its countermeasure. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security* (CCS '22). Association for Computing Machinery, Los Angeles, CA, USA, 3181–3194. ISBN: 9781450394505. DOI: 10.1145/3548606.3559372.

[82] Chaoshun Zuo, Haohuang Wen, Zhiqiang Lin, and Yinqian Zhang. 2019. Automatic fingerprinting of vulnerable ble iot devices with static uuids from mobile apps. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (CCS '19). Association for Computing Machinery, London, United Kingdom, 1469–1483. ISBN: 9781450367479. DOI: 10.1145/3319535.3354240.

A Methodology (Extended)

We now describe the methodology we used to construct our corpus of prior work on offline finding systems and our inclusion criteria. The corpus is included in our artifact, available at https://github.com/cortegap/pets-26-sok-artifact.

Inclusion criteria. To be included in our study, a paper must have satisfied at least one of the following: (1) directly analyzed the security, privacy, or safety of OF systems or vendor-specific devices (e.g., AirTags, Tiles, SmartTags), (2) contributed broadly to the understanding of adversarial models, security definitions, or abusability concerns in OF systems, (3) proposed secure constructions for or based on bluetooth-based proximity tracking protocols.

Search methodology. To increase coverage, we used two independent approaches, each conducted by a different author.

Method 1: Snowball sampling from seed papers. We began with a seed set of 13 highly cited and recent papers that directly addressed the security and privacy of OF systems. From each paper, we examined both backward and forward citations to identify additional relevant work. A cited paper was added to the corpus if it satisfied our inclusion criteria. We repeated this process iteratively until reaching saturation—that is, no new papers meeting the criteria were discovered in subsequent citation rounds. Our citation-based snowballing approach provided breadth of venue coverage. The corpus resulting from this method is available in our artifact, in `method1-snowball.csv`.

Method 2: Targeted keyword search across venues. To complement the snowballing approach, we conducted a systematic keyword-based search in July 2025 across major academic venues in security, privacy, and cryptography. We searched for papers using the following keywords: *Find My, AirTag, Tile tracker, Find Hub, Chipolo, Pebblebee, SmartTag, offline finding, safety alert, Bluetooth tracking, stalkerware, crowdsourced network, proximity finding, finder device, location tracking, anti-stalking, tracking tag, BLE advertisements*, and variants thereof. The full list of keywords is available in our artifact, in `keywords.csv`.

The academic venues we searched were

> **Security/Privacy Conferences:** USENIX Security, ACM CCS, IEEE S&P, NDSS, PETS, SOUPS.
> **Workshops and Specialized Venues:** ACM WiSec, ACM WPES
> **Preprint Archives:** IACR ePrint.

We used venue-specific databases for our search: Google Scholar (USENIX, NDSS, PETS, SOUPS), the ACM Digital Library (CCS, WiSec, WPES), and IEEE Xplore (IEEE S&P). This search surfaced 231 papers, which we manually reviewed and filtered down based on our inclusion criteria. The list of 231 papers is available in our artifact, in `method2-keywords.csv`.

Constructing the final corpus. After having an initial list of papers, the authors responsible for each method categorized the papers in their respective corpus. We explain the categorization below. This categorization helped further reduce both corpuses. Additionally, unpublished papers were discarded. For all papers that were found by both approaches and were included by one and excluded by the other (5 papers), the first and second authors met and discussed until reaching consensus. Additionally, all papers included though one of the methods and not found by the other were reviewed by the author responsible for the latter.

We additionally included (1) the Detecting Unwanted Location Tracking IETF draft(s), (2) official specifications and privacy policies for Apple's Find My (deprecated at time of writing) and Google's Find Hub protocols, and (3) support guides explaining what to do in case of unwanted tracking by Apple, Google, Samsung, and Tile. Our final corpus is composed of 49 documents and is included, along with the categorization, in our artifact, in `final-corpus.csv`.

Categorization. Selecting the categories was an iterative process, that included several discussions between the authors responsible for methods 1 and 2, and multiple rounds of re-categorization of papers. Here we describe the final categories. We sorted the papers in the final corpus based on topics and type. For topics, papers were sorted among one or more of the following categories: physical layer, OF core protocol, anti-stalking, OF piggybacking, OF-inspired crowdsourced systems, and privacy policy. For types, the categories were: protocol proposal, attack, formalization, measurement study, user study, reverse engineering, and legal document.

B Pairing protocol implementations

B.1 Pairing for Apple's FindMy

We summarize the pairing protocol for Apple's Find My network in Figure 3 and describe it below. For brevity, we group certain steps together. Our description is based on the reverse engineering efforts of Heinrich et al. [38] and Liu et al. [51], and Apple's now deprecated Find My specification [3]. The owner device authenticates itself to the service provider by requiring the user to log in to their iCloud account.

Manufacture-time setup. At manufacture time, each AirTag is provisioned a unique 16-byte serial number, a 1024-byte software token, and a 16-byte software UUID. Additionally, the tag is also given the server's public encryption key $pk^{\mathsf{Enc}}_{server}$ and the server's public verification key $pk^{\mathsf{Sign}}_{\mathsf{server}}$. The Apple server maintains a database that maps each tag's serial number to its corresponding software token and software UUID.

Tag authentication. As the first step of the pairing process, the owner device samples a uniformly random 32-byte nonce n and generates an 89-byte EncryptedBlob1 and sends both to the tag. [1] The tag generates its ephemeral keys $(sk^e_{\mathsf{tag}}, pk^e_{\mathsf{tag}})$, its own 32-byte nonce n_{tag}, and a 32-byte random seed seedT. Then it generates a commitment C1 by hashing sk^e_{tag} and n_{tag}. Subsequently, the tag creates a message M as $n \,\|\, \mathsf{C1} \,\|\, \mathsf{deviceInfo} \,\|\, \mathsf{EncryptedBlob1} \,\|\, \mathsf{authToken}$ where deviceInfo contains its serial number, software token, and software UUID. The tag then encrypts M under the server's public key $pk^{\mathsf{Enc}}_{\mathsf{server}}$ to produce EncryptedBlob2. Finally, the tag sends EncryptedBlob2 to the owner device.

The owner device now samples its ephemeral key pair $(sk^e_{\mathsf{owner}}, pk^e_{\mathsf{owner}})$ and nonce n_{owner}, and produces commitment $H1$ by hashing pk^e_{owner} and n_{owner}. The owner device forwards EncryptedBlob2 and $H1$ to the server. The server decrypts EncryptedBlob2 to get

[1] Neither Apple's specification [3] nor [51] specify what is encrypted within EncryptedBlob1, and we were not able to independently verify what it encrypts so we omit the corresponding details here.

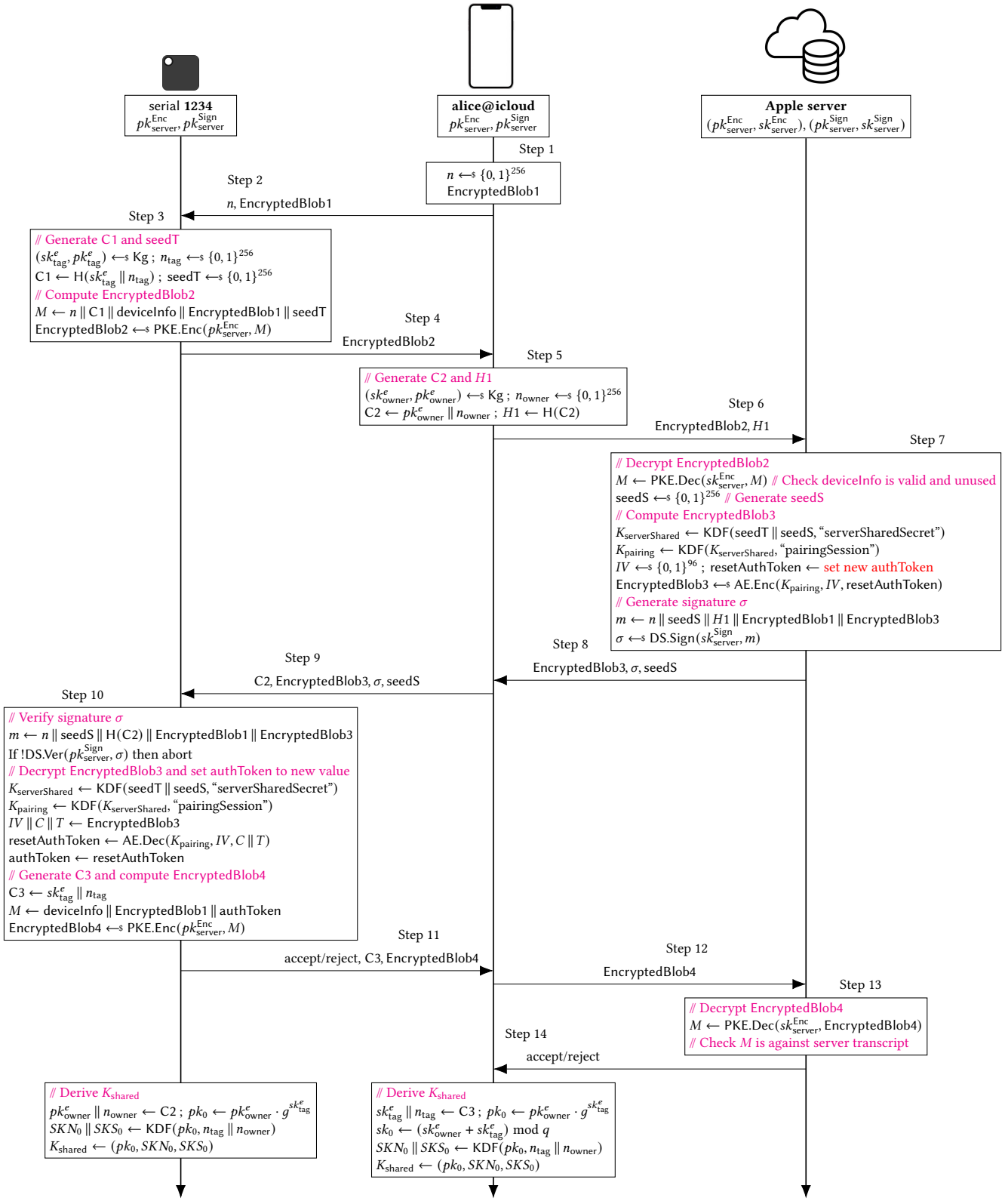

Figure 3: Pairing for Apple's Find My network.

M and checks that the serial number, software token, and software UUID are as reflected in its internal database, and the tag has not been registered before. It then parses M to get deviceInfo ‖ EncryptedBlob1 ‖ authToken, and samples its own 32-byte seed seedS. The server derives its shared secret with the tag $K_{\text{serverShared}}$ by applying a key derivation function KDF to seedT and seedS. It then derives its pairing key with the tag, K_{pairing}, by applying KDF to $K_{\text{serverShared}}$. Next, the server samples a fresh software token and encrypts it under K_{pairing} using a fresh IV to get EncryptedBlob3. Finally, it signs n ‖ seedS ‖ $H1$ ‖ EncryptedBlob1 ‖ EncryptedBlob3 to get the signature σ, and sends EncryptedBlob3, σ, seedS to the owner device, which forwards them to the tag along with pk^e_{owner} and n_{owner}.

The tag verifies the server's signature σ after reconstructing m locally. It derives $K_{\text{serverShared}}$ and K_{pairing} analogously to the server and uses K_{pairing} to decrypt EncryptedBlob3 and get the new software token. Finally, it encrypts the new software token and EncryptedBlob1 under $pk^{\text{Enc}}_{\text{server}}$ to get EncryptedBlob4, and sends EncryptedBlob4, sk^e_{tag}, and n_{tag} to the owner device. The owner device forwards EncryptedBlob4 to the server which decrypts and verifies it against its transcript. This completes tag authentication.

Shared key derivation. As the last step of the pairing protocol, the tag and the owner establish their shared key. The shared master public key pk_0 is generated as $pk^e_{\text{owner}} \cdot pk^e_{\text{tag}}$ and the secret key sk_0 as $(sk^e_{\text{owner}} + sk^e_{\text{tag}})$. Both the tag and the owner device also compute SKS_0 and SKN_0 by applying KDF to n_{owner} and n_{tag}. The shared key between the tag and the owner is (pk_0, SKS_0, SKN_0). Looking ahead, SKS will be used during the lost mode and SKN during connected mode. Note that only the owner device can derive sk_0, as it is only used to decrypt location reports on the owner device.

B.2 Pairing for Google's FindHub

We summarize the pairing protocol for Google's FindHub network in Figure 4 and describe it below. For brevity, we group certain steps together and omit those that are not directly relevant to FindHub operations or security. Our description is based on the reverse engineering efforts of Bottger et al. [10], and Google's Find Hub and FastPair specifications [32, 33].

FastPair background. The FindHub pairing protocol builds on Google's *FastPair* protocol [33]. We recall some details of the FastPair protocol that are relevant to the FindHub registration phase. To support FastPair, a tag manufacturer must first register their device model with Google. Once registered, Google assigns the tag a model identifier modelId and an *Anti-Spoofing* public key pair $(sk_{\text{modelId}}, pk_{\text{modelId}})$, which in practice is an Elliptic Curve key pair. The private key sk_{modelId} is embedded in every tag of that model, while Google's servers maintain a mapping from each modelId to its corresponding pk_{modelId}. Notably, all tags sharing the same modelId use the same anti-spoofing key pair.

FindHub registration: Owner preprocessing. The FindHub registration process begins with the server provisioning the owner with a 32-byte owner key K_{owner}.[2] This key is later used to encrypt other long-term secrets in the FindHub protocol.

[2]In practice, K_{owner} is sent to the owner encrypted under a key shared with the server and stored in the owner's Google Vault under the "finder_hw" domain.

Then the owner device queries the tag for its modelId. The owner device forwards this modelId to the server which responds with the public anti-spoofing key pk_{modelId} associated with the tag. The owner device then generates its own ephemeral key pair $(sk_{\text{owner}}, pk_{\text{owner}})$ and derives a Diffie–Hellman shared secret K_{DH} as $sk_{\text{owner}} \cdot pk_{\text{modelId}}$. From K_{DH}, the owner device generates a 128-bit key K_{Enc}, by applying the SHA-256 hash function and truncating the output to 128 bits.

Next, the owner device samples three additional keys: (1) a 256-bit AES key K_{AES}, (2) a 128-bit account key K_{account} (the "master" FastPair key, used for mutual authentication), and (3) a 256-bit ephemeral identity key $K_{\text{ephemeralId}}$ (the "master" FindHub key, used to derive future keys and secrets for FindHub operations).

The owner encrypts these keys in a nested fashion using the AES-128-ECB symmetric encryption scheme.

$$\begin{aligned} c_0 &= \text{AES-128-ECB.Enc}(K_{\text{Enc}}, K_{\text{AES}}),\\ c_1 &= \text{AES-128-ECB.Enc}(K_{\text{AES}}, K_{\text{account}}),\\ c_2 &= \text{AES-128-ECB.Enc}(K_{\text{account}}, K_{\text{ephemeralId}}). \end{aligned}$$

Additionally, it also computes the recovery key K_{recovery} by applying the SHA-256 hash function to $K_{\text{ephemeralId}} \,\|\, 0x01$ and the unwanted tracking protection key $K_{\text{uwTracProtec}}$ by applying SHA-256 to $K_{\text{ephemeralId}} \,\|\, 0x03$. The recovery key K_{recovery} is stored on the server to allow the owner to recover $K_{\text{ephemeralId}}$ if it is lost. The purpose of $K_{\text{uwTracProtec}}$ is explained in Appendix D.3. Finally, the owner sends $(c_0, c_1, c_2, pk_{\text{owner}})$ to the tag.

FindHub registration: Tag preprocessing. The tag uses its private key sk_{modelId} and the received pk_{owner} to derive the shared K_{DH} and compute K_{Enc} as above. It decrypts the ciphertexts in sequence:

$$\begin{aligned} K_{\text{AES}} &= \text{AES-128-ECB.Dec}(K_{\text{Enc}}, c_0),\\ K_{\text{account}} &= \text{AES-128-ECB.Dec}(K_{\text{AES}}, c_1),\\ K_{\text{ephemeralId}} &= \text{AES-128-ECB.Dec}(K_{\text{account}}, c_2). \end{aligned}$$

It then derives K_{recovery} and $K_{\text{uwTracProtec}}$ analogously to the owner.

To prove knowledge of K_{account}, the tag generates a random 8-byte nonce n, computes the tag h by applying HMAC-SHA-256 to $(K_{\text{account}}, 0x01 \,\|\, n \,\|\, 0x02 \,\|\, 0x08)$, and sends (h, n) to the owner.

FindHub registration: Final steps. The owner device authenticates h by comparing it to the locally generated MAC h'. If the authentication succeeds, the owner device encrypts $K_{account}$ and $K_{\text{ephemeralId}}$ using AES-GCM under K_{owner} and uniformly random 12-byte nonces n_3 and n_4 to obtain the ciphertexts c_3 and c_4. Finally, the owner device sends c_3, c_4, K_{recovery}, and $K_{\text{uwTracProtec}}$ to the server.

This completes registration for the tag. The key $K_{\text{ephemeralId}}$ is the shared secret K_{shared} established between the tag and the owner device and is used for all future FindHub operations.

B.3 Pairing for Samsung's SmartThings Find

In this section, we describe the pairing protocol used in Samsung's SmartThings Find. Our description is based on the reverse engineering efforts of Yu et al. [80] and Liu et al. [51].

Manufacture-time setup. At manufacture time, each Samsung SmartTag is provisioned a unique 6-byte MAC address macAddress.

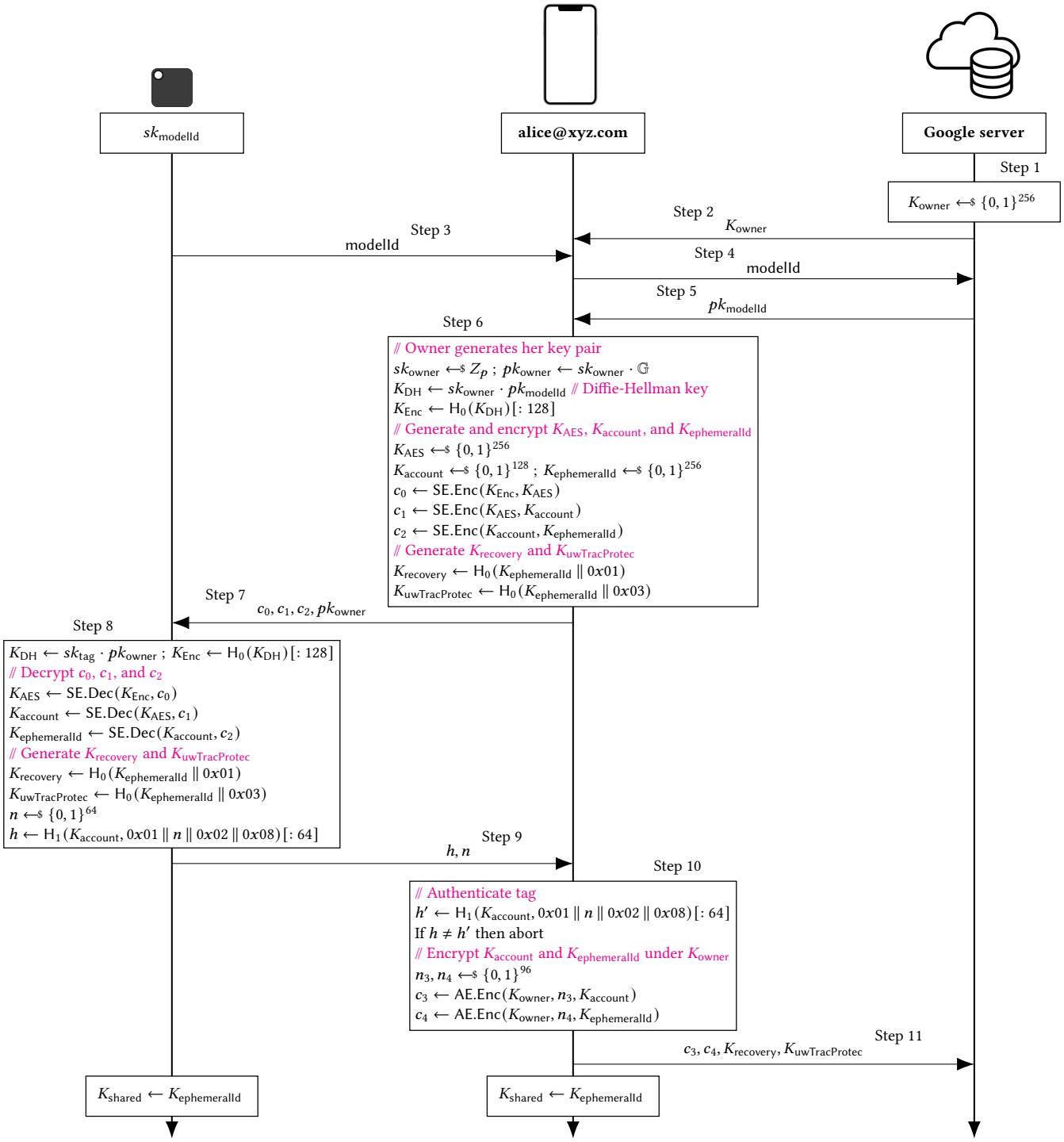

Figure 4: Registration for Google's Find Hub network. H_0 **is** SHA-256, H_1 **is** HMAC-SHA-256, **and** SE **is AES-128-ECB.**

The long-term secret key of the tag, sk_{tag} is derived deterministically from the macAddress by applying a hash function H_0. The corresponding public key pk_{tag} is $sk_{\text{tag}} \cdot \mathbb{G}$. The Samsung server retains a database T that maps $\text{H}_1(\text{macAddress})$ to pk_{tag} where H_1 is a hash function.

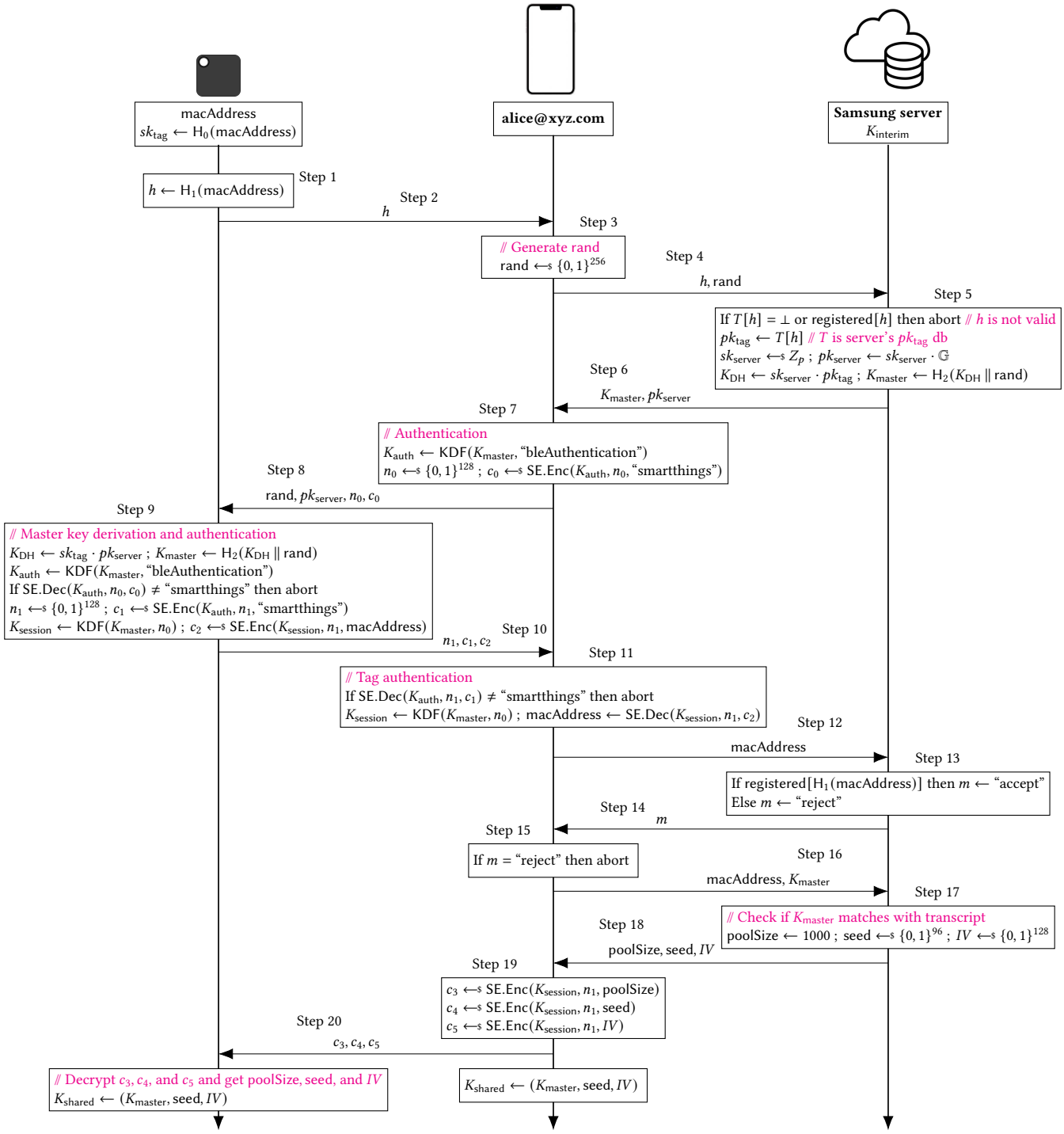

Figure 5: Registration for Samsung's SmartThings Find network.

Master key derivation and tag-owner authentication. The pairing process begins when the tag sends $h = H_1(\text{macAddress})$ to the owner device. The owner device samples a 32-byte random value rand and sends h, rand to the server. If the queried MAC address is invalid, i.e., $T[h] = \perp$, the server aborts. Otherwise, the server retrieves pk_{tag} from T, samples its ephemeral key pair $(sk_{server}, pk_{server})$, and derives the Diffie-Hellman key $K_{DH} \leftarrow sk_{server} \cdot pk_{tag}$ and the master secret key $K_{master} \leftarrow H_2(K_{DH} \| rand)$.

Then the server sends K_{master} and pk_{server} to the owner device over TLS. The owner device derives authentication key K_{auth}, which is used for mutual authentication between the tag and the owner device, by applying a key derivation function KDF to K_{master} and the string "bleAuthentication." The owner device then encrypts the string "smartthings" under K_{auth} and a uniformly random 16-byte nonce n_0 to produce the ciphertext c_0 and sends rand, pk_{server}, n_0, and c_0 to the tag.

The tag derives K_{master} from sk_{tag} and pk_{server} analogously to the server and K_{auth} analogously to the owner device. Using K_{auth} and n_0, the tag authenticates the owner device by verifying that c_0 decrypts to "smartthings." Then the tag derives the session key K_{session} by applying KDF to K_{master} and n_0. The tag subsequently computes two ciphertexts: (1) c_1 by encrypting the string "smartthings" under K_{auth} and a uniformly random string n_1, and (2) c_2 by encrypting macAddress under K_{session} and n_1. The tag sends n_1, c_1, and c_2 to the owner device. The owner device authenticates the tag by decrypting c_1 using K_{auth} and n_1, and obtains the macAddress by decrypting c_2 using K_{session} and n_1.

Pairing finalization. The owner device sends macAddress to the server, which returns "accept" if the tag wasn't already registered, and "reject" otherwise. If the server rejected the pairing, the owner device aborts. Otherwise, it sends macAddress, K_{master} to the server which verifies that K_{master} matches its internal records and sets three parameters: (1) a poolSize which is always set to 1000, (2) a uniformly random 12-byte seed, and (3) a 16-byte IV which is a fixed value for all tags. The server sends poolSize, seed, and IV to the owner device which in turn shares it with the tag encrypted under K_{session} and n_1. The shared key K_{shared} is $(K_{\text{master}}, \text{seed}, IV)$.

B.4 Pairing for Tile's protocol

We now describe Tile's pairing protocol. At manufacture time, each tag of a particular model is provisioned the same 16-byte interim key K_{interim} and a unique 8-byte tile_uuid. The server maintains a mapping between tile_uuids and K_{interim} values.

The tag initiates pairing by sending its tile_uuid to the owner device. The owner device generates a uniformly random 14-byte string randA and sends it to the tag. The tag generates its own 10-byte secret randT, and computes the authentication string sresT as $\mathsf{H}(K_{\text{interim}}, \text{randA} \,\|\, \text{randT} \,\|\, \text{tile\_uuid})$ where H is a hash function. It also derives the shared key K_{shared} as $\mathsf{H}(K_{\text{interim}}, \text{sresT})$. Then the tag sends randT and sresT to the owner device which forwards it to the server along with randA and tile_uuid. The server authenticates the tag by verifying sresT and ensuring that the tile_uuid is not already registered, and derives K_{shared} analogously to the tag. Finally, the server sends K_{shared} to the owner device, completing the pairing.

C Connected mode implementations

C.1 Apple's FindMy

Once a tag is registered with its owner device, it begins broadcasting BLE advertisements containing a 4-byte ephemeral identifier id which rotates every 15 minutes. The identifiers are generated from K_{shared} as shown in Figure 7. The tag parses K_{shared} to get (pk_0, SKN_0, SKS_0). For deriving the identifier for the i^{th} epoch id_i, the tag first computes SKN_i by applying a KDF to SKN_{i-1}, then derives $u_i \,\|\, v_i$ by applying a KDF to SKN_i. Finally, id_i is computed as $pk_0^{u_i} \cdot g^{v_i}$. When broadcasting, the tag uses the first six bytes of id_i as the advertisement address and only includes the first two bits of the first byte in the payload. Though id_i is a public key, it is never broadcast or used as such.

C.2 Google's FindHub

Once a tag is registered with its owner device, it begins broadcasting BLE advertisements containing an ephemeral identifier id. The identifier can be 20 or 32 bytes long, depending on whether the tag uses Bluetooth 4.0 or Bluetooth 5.0, respectively.

Identifier generation. The connected-mode protocol sets the epoch duration Δ_{conn} to $1024 = 2^{10}$ seconds and defines a rotation exponent rotExp $= 10$. The tag generates id_i as described in Figure 8. The time counter timeCtr is computed as the time (in seconds) since the tag was activated, with the lowest rotExp bits set to 0. The value r' is obtained by encrypting the value (0xFF ‖ rotExp ‖ timeCtr ‖ 0x00 ‖ rotExp ‖ timeCtr) using AES-ECB under $K_{\text{ephemeralId}}$. It is then mapped to the finite field of either SECP160R1 (for 20-byte id_i) or SECP256R1 (for 32-byte id_i) by reducing r' modulo the curve order n to obtain r. Finally, id_i is set to the x-coordinate of the curve point $R = r \cdot \mathbb{G}$. Note that the value of id_i changes with timeCtr which increments every Δ_{conn} seconds. [3]

Uploading identifiers. The owner device periodically uploads a batch of precomputed future identifiers to the server. When a finder later submits a location report for a lost tag, the report contains the observed identifier. The server uses this identifier to determine which tag it belongs to and associate the report with the correct user account.

The UT **byte**. Each advertisement includes a one-byte unwanted tracking mode value, UT, that encodes the tag's operational mode: connected (0x40) or lost (0x41). This value is intended to mitigate unwanted tracking by indicating when a tag is within range of its owner. Tags in connected mode rotate their randomized MAC addresses in sync with id_i. In lost mode, however, MAC addresses rotate only once every 24 hours.

Location reporting. Unlike Apple, Google owner devices periodically submit location reports for their own tags even in connected mode. This is because Google offers an additional feature through its OF protocol that allows an owner to retrieve the last location where they were connected to their tag. These reports are encrypted under SHA-256($K_{\text{ephemeralId}}$) using AES-GCM with a fresh 12-byte nonce n.

C.3 Samsung's SmartThingsFind

Once a tag is registered with its owner device, it begins broadcasting BLE advertisements containing an ephemeral 8-byte identifier.

Identifier generation. The connected-mode protocol sets the epoch duration Δ_{conn} to 15 minutes. The tag generates id_i as described in Figure 9. First, the tag computes a privacy ID key K_{privId}

[3] To avoid timing-based attacks against unlinkability, Google recommends tag manufacturers to implement small random delays, i.e., instead of incrementing timeCtr every Δ_{conn} seconds, increment it every $\Delta_{\text{conn}} + \epsilon$ seconds where $\epsilon \leftarrow\$ [1, 204]$ seconds.

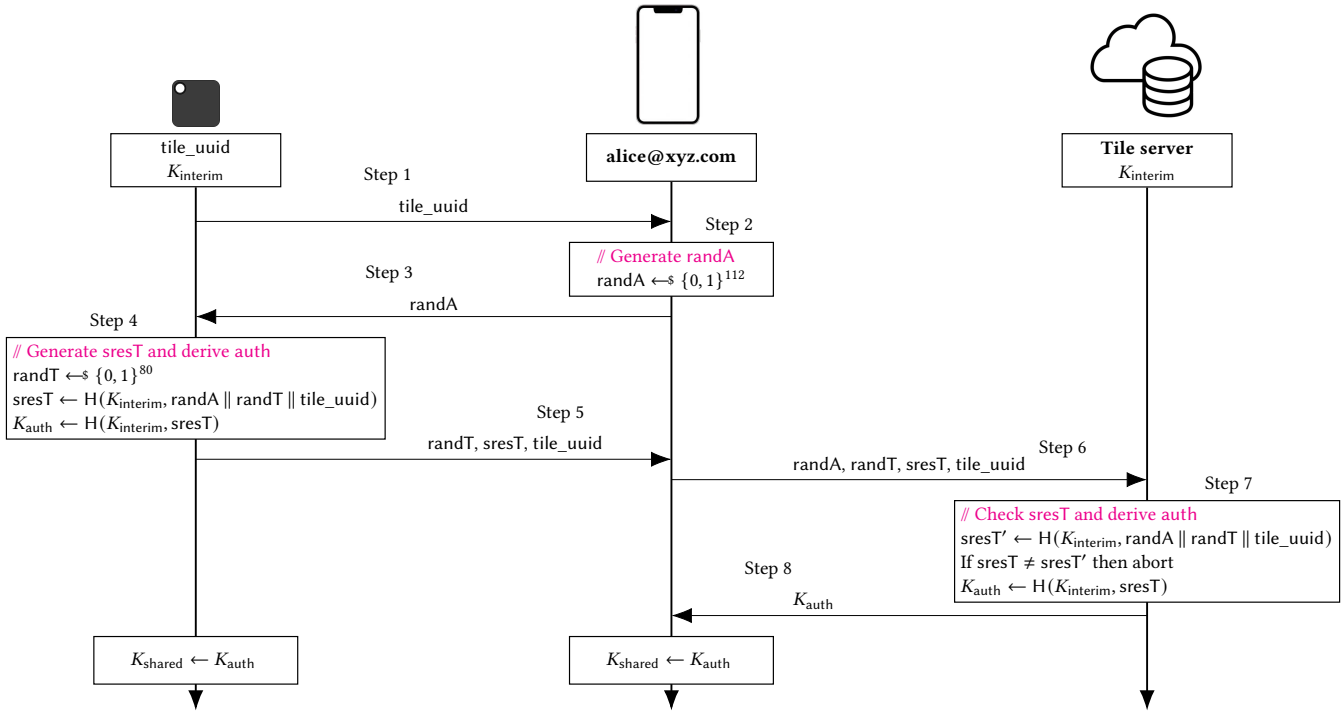

Figure 6: Registration for Tile's protocol.

$(pk_0, SKN_0, SKS_0) \leftarrow K_{\text{shared}}$
$SKN_i \leftarrow \text{KDF}(SKN_{i-1}, \text{“update”})$ // KDF = ANSI-X9.63-KDF
$u_i \,\|\, v_i \leftarrow \text{KDF}(SKN_i, \text{“diversify”})$ // $|u_i| = |v_i| = 288$
$u_i \leftarrow (u_i \bmod q - 1) + 1$
$v_i \leftarrow (v_i \bmod q - 1) + 1$
$id_i \leftarrow pk_0^{u_i} \cdot g^{v_i}$

Figure 7: Derivation of connected-mode identifiers for Apple's FindMy protocol.

$K_{\text{ephemeralId}} \leftarrow K_{\text{shared}}$
$str \leftarrow 0\text{xFF} \,\|\, \text{rotExp} \,\|\, \text{timeCtr} \,\|\, 0\text{x00} \,\|\, \text{rotExp} \,\|\, \text{timeCtr}$
$r'_i \leftarrow \text{KDF}(K_{\text{ephemeralId}}, str)$ // KDF = AES-256-ECB.Enc
$r_i \leftarrow r'_i \bmod n$
$id_i \leftarrow r_i \cdot \mathbb{G}$

Figure 8: Derivation of connected-mode identifiers for Google's FindHub protocol.

by applying SHA-256 to the master key K_{master} derived during pairing. Then it generates poolSize number of unique identifiers by applying AES-CBC-256 to IV and the epoch counter x_i and truncating the output to 8 bytes. Here, IV is the value set by the server during pairing, and x_i is derived deterministically from i.

Recall that the server possesses all the inputs required to generate a tag's identifiers. The server generates these identifiers analogously

$(K_{\text{master}}, \text{seed}, IV) \leftarrow K_{\text{shared}}$
$K_{\text{privId}} \leftarrow \text{KDF}_0(K_{\text{master}}, \text{“privacy”})$ // KDF_0 = SHA-256
For $i = 1$ to poolSize do
  $x_i \leftarrow 0 \,\|\, (i \wedge 255) \,\|\, \text{seed} \,\|\, 0 \,\|\, (i \wedge 255)$
  $id_i \leftarrow \text{KDF}_1(K_{\text{privId}}, IV, x_i)[:64]$ // KDF_1 = AES-CBC-256

Figure 9: Derivation of connected-mode identifiers for Samsung's protocol.

to the tag and maintains a map of identifiers to the corresponding owner. This map will become relevant during the lost mode interactions which we describe in Appendix D.

Advertisement data. In addition to the 8-byte id_i, the tag includes a status byte to encode the state of the tag, a 3-byte time counter timeCtr which increments every 15 minutes, a 4-byte authentication tag which is produced by MAC-ing the status byte, id_i, and timeCtr and truncating the output. Additionally, the advertisement payload contains an E2E field which is set to 0 by default and 1 when the owner enables end-to-end encryption. We describe this optional feature in Appendix D. After every 15-minute interval, a tag in the connected mode updates id_i, timeCtr, the authentication tag, and the broadcast MAC address.

C.4 Tile's protocol

Tile's protocol does not differentiate between lost and connected modes; tags broadcast similarly in both modes, rotating identifiers every 15 minutes. We show Tile's identifier generation process in

$K_{\text{seed}} \leftarrow \text{KDF}(K_{\text{shared}}, \text{tile\_uuid} \,\|\, \text{“identity”})$
For $i = 1$ to 8640 do
  $id_i \leftarrow \text{KDF}(K_{\text{seed}}, i)[:64]$

Figure 10: Derivation of connected-mode identifiers for Tile's protocol. Above, KDF **is** HMAC-SHA-256.

$(pk_0, SKN_0, SKS_0) \leftarrow K_{\text{shared}}$
$SKS_i \leftarrow \text{KDF}(SKS_{i-1}, \text{“update”})$ // KDF = ANSI-X9.63-KDF
$u_i \,\|\, v_i \leftarrow \text{KDF}(SKS_i, \text{“diversify”})$ // $|u_i| = |v_i| = 288$
$u_i \leftarrow (u_i \bmod q - 1) + 1$
$v_i \leftarrow (v_i \bmod q - 1) + 1$
$id_i \leftarrow u_i \cdot pk_0 + v_i \cdot \mathbb{G}$

Figure 11: Derivation of lost-mode identifiers for Apple's FindMy protocol. Identifiers rotate every $\Delta_{\text{lost}} = 24$ hours.

Figure 10. After registration, tags generate 8640 identifiers from K_{shared}. First, a seed key K_{seed} is generated by applying a KDF to K_{shared}. Then, the i^{th} identifier id_i for $i \in [1, 8640]$ is generated by applying KDF to K_{seed} and i and truncating the output to 8 bytes. Once a tag has rotated through 8640 identifiers, it cycles back through the same values.

D Lost mode implementations

D.1 Apple's FindMy

We now discuss Apple's implementation of lost mode interactions in its Find My network. The implementation for generating lost-mode identifiers is the same as the one for generating connected-mode identifiers, except the separated mode key *SKS* is used instead of the near mode key *SKN*. Moreover, the identifiers rotate every 24 hours instead of every 15 minutes. Additionally, tags encode their state in the advertisement payload. Bit 2 of byte 2 of the advertisement payload indicates the mode of the tag. It is set when connected to an owner device (indicating near mode) and unset after being disconnected for 15 minutes (indicating lost mode). Nearby finder devices only report locations for tags in lost mode. Similarly to the near mode, lost mode advertisements use the first six bytes of the identifier *id* as the advertising address. However, the advertisement also includes the remaining bytes of the identifier in the advertisement payload. In lost mode, the identifier is a public key and is used by nearby finders to encrypt location reports.

D.2 Samsung's SmartThings Find

We now describe Samsung's implementation of the lost mode in SmartThings Find.

Mode transitions. A SmartTag enters *offline mode* if it has been disconnected from its owner for 15 minutes, and *overmature offline mode* if it remains offline for 24 hours. The tag's mode is encoded in bits 5–7 of byte 0 of the advertisement payload (010 for offline mode and 011 for overmature offline mode). Lost-mode identifiers are generated using the same procedure as connected-mode identifiers. Finder devices only report locations for tags that are in either offline or overmature offline mode.

Optional end-to-end encryption. By default, finder devices upload location reports to the server in plaintext over TLS. The server associates each report with the correct owner by using the identifier mapping described in Appendix C.3. When an owner later queries the locations of its tags, the server returns all reports corresponding to that owner.

Samsung, however, provides an optional end-to-end (E2E) encryption mode that users can explicitly enable. Tags with E2E enabled indicate this setting via a dedicated field in their advertisements. When a user enables the E2E mode, the user is first required to choose a 6-digit PIN. Then, the owner device generates a key pair (pk_e, sk_e), encrypts sk_e using the hash of the PIN and a random IV, producing ciphertext c, and sends (pk_e, c, IV) to the server. Finders that encounter such tags request pk_e from the server by querying the recorded identifier, encrypt the location under pk_e, and send the ciphertext together with the tag's identifier back to the server. Finally, the owner decrypts the reports locally using sk_e.

D.3 Google's Find My Device Network

We now discuss Google's implementation of lost mode interactions in its Find My Device Network. Google's implementation for generating lost-mode identifiers is the same as the one for generating connected-mode identifiers. However, in the lost mode, tags in Google's FMDN enter the Unwanted Tracking (UT) mode where they slow down the rate of MAC addresses randomization to once in 24 hours.

End-to-end encryption of location data. Recall that the identifiers broadcast by a tag are ephemeral public keys. By default, a FMDN finder encrypts location reports using the observed identifier, discarding any reports for locations with 100m of their home location. Each submission to the server includes the encrypted location report, the (truncated) identifier of the tag, and the UT status observed in the advertisement.

Mode transitions. In Apple's protocol, a tag automatically switches between connected and lost modes based on Bluetooth proximity to its owner device. In contrast, Google's FMDN relies on the server to control this transition. When a finder device submits a location report for the tag, the server checks the included UT status. If UT mode is not enabled, the server extracts the truncated identifier from the report and checks the last time the corresponding owner submitted a location report for it. If more than 30 minutes have passed since the owner of the tag reported its location, the server responds to the finder with an HTTPS message containing the unwanted tracking protection key $K_{\text{uwTracProtec}}$ (see Appendix B.2) instructing it to set the UT byte of the tag. The UT bit can only be cleared when the tag reconnects to its owner device.

E Challenges beyond the protocol level

In this section we describe works highlighting some additional risks and challenges beyond the protocol layer.

BLE security analyses. Even if a protocol achieves unlinkability at the cryptographic or protocol level, the physical layer often leaks side channels that allow efficient and reliable fingerprinting of devices. Characteristics such as static identifiers in advertisements, radio signal patterns, timing behavior, and hardware design choices

can all be exploited to link broadcasts or infer device identities. A growing body of work shows that these physical-layer attacks are not only practical but also scalable, undermining unlinkability guarantees by revealing long-term identifiers or persistent correlations across broadcasts. We summarize these works here.

BLE advertisements include a static UUID that enables finders to filter devices for OF purposes, but also exposes a fingerprinting vector. Zuo et al. [82] demonstrated large-scale device fingerprinting using wide-area scanning, identifying Tile among the top 10 device types. Moreover, attackers can exploit signal strength of scanned advertisements to further distinguish devices of the same type.

Ludant et al. [52] showed that current Bluetooth chip design enables linking BLE advertisements to Bluetooth Classic (BTC) frames through their shared clock, revealing the device's global identifier (BDADDR). The attack required BLE and BTC to be simultaneously used, which is quite common (e.g., a user using an AirTag and their Airpods). Tested against Apple's Find My, this attack linked ephemeral tag identifiers to user identities. More generally, this physical-layer vulnerability lets attackers correlate tag broadcasts even when protocols aim to provide unlinkability.

Zhang and Lin [81] presented another BLE fingerprinting attack that is effective despite randomized MACs and payloads. The attack exploits communication patterns between a paired peripheral and central when allow-lists are used for automatic reconnection. Because the central and peripheral change MAC addresses asynchronously, an attacker can link these patterns to continuously track the peripheral over long periods.

Leu et al. [47] demonstrated a distance-reduction attack against the Ultra-Wideband (UWB) ranging feature in Apple's AirTags, allowing an adversary to convince a phone that the tag is nearby when it is not. While the implications for the security notions we consider are not immediately clear, this result highlights how the threat landscape may expand as UWB becomes more integrated into OF systems.

David et al. [50] showed yet another failure of unlinkability in practice through a novel physical-layer vulnerability they call the Battery Insertion Attack. In this attack, an adversary can restart a beacon at a chosen time and leverage the resulting timing patterns between identifier rotations to link broadcasts, even when periodic rotation is used. The key insight is that attackers observe not only the identifiers themselves but also the precise moments at which they change. Since existing unlinkability definitions do not model timing information, the authors propose a stronger notion—timed-sequence indistinguishability—which accounts for both the content and timing of broadcasts.

Chen et al. [16] demonstrate a large-scale privacy attack that exploits a BLE vulnerability in Apple's Find My network. The vulnerability arises because, although the Find My protocol specifies that tags in lost mode must broadcast using static random addresses, this requirement is difficult to enforce in practice. As a result, the network also accepts broadcasts using other address types, namely non-resolvable and resolvable private addresses. The attack allows adversaries to remotely transform ordinary devices—including desktops, laptops, smartphones, and IoT devices running Linux, Windows, or Android—into trackable "tags", making it possible to uniquely locate them over the long term.

Unwanted tracking detection in practice. While we discussed earlier detectability definitions, here we focus on prior work that reversed engineered currently deployed anti-stalking mechanisms, proposed new ones, and strived to understand the stalking risks of tracking devices through user and measurement studies.

Prior work investigated the broader IoT ecosystem risks and analyzed the ways in which these technologies can be abused. Stephenson et al. [66] performed a measurement study on websites about how abusers use IoT devices, including Bluetooth trackers, for interpersonal abuse. Meanwhile, Wei et al. [76] conducted a measurement study on TikTok videos that give anti-privacy and anti-security advice on using technology for surveillance and control in interpersonal relationships. Stephenson et al. [65] conduct a series of semi-structured interviews with people with first or second-hand experience with IoT abuse to understand how survivors identify abuse, strategies to mitigate it, and challenges in doing so. Finally, Ceccio et al. [14] performed a survey of spy devices and detection tools available, showing how spy devices are easy to acquire, but hard to detect with existing mechanisms.

Some work looked into understanding how anti-stalking tools available specifically for tracking devices work. Heinrich et al. [36] reverse-engineered Apple's anti-stalking protection mechanism. The authors noted how Apple's protocol at the time was not sufficient to protect the users from being stalked, since notifications were delivered too late, when user's private locations, like their home, was most likely already revealed to the abuser. Turk et al. [73] described the mechanisms available to detect AirTags, SmartTags, Tile, and Chipolo trackers. They pointed out weaknesses in such mechanisms, ranging from how hard to use, to the challenge in distinguishing between honest and malicious usage of these trackers. Shafquat et al. [63] investigated the anti-stalking mechanism in Find My. The work considered how different factors, such as device model, local time, etc. play a role in the time it takes to trigger safety alert.

Additionally, some previous work showed attacks against current anti-stalking protocols. Shafquat et al. [63] showed an attack where tampering with some bytes in AirTags BLE advertisements lead to trackers not triggering safety alert notifications. Mayberry et al. [55] showed how to create tracking tags that participate in the Find My network, but that do not trigger safety alerts, based on attacks that involve bit-flipping and rotating keys faster. These attacks, combined with the ability to create fake or cloned tags, as shown in [37, 13, 58, 10], expose great deficiencies in the existing anti-stalking tools.

Given the weaknesses on manufacturer-produced anti-stalking tools, previous work also builds new applications and techniques to detect malicious trackers. AirGuard [36] is an Android app for detecting tags following the Find My protocol. This included detection for fake tags using the OpenHaystack framework [37]. It was proposed due to a gap between the ability of Apple and Android devices in detecting AirTags. Meanwhile, BLE-Doubt [12] is an Android app to detect trackers from multiple vendors. HomeScout [56] extends AirGuard to detect Tile and SmartTags too and BL(u)E CRAB [18] includes an application to improve trackers detection by relying on more factors beyond time and distance. Finally, Despres et al. [21] propose an algorithm to detect trackers even when these rotate MAC addresses.

Finally, several works performed user studies to analyze the usability of the anti-stalking features. Heinrich et al. [39] analyzed data from AirGuard users and conducted an online survey among such users about the misuse of Bluetooth trackers and protections against it. Turk and Hutchings [72] conducted a user experiment based on the Assassins game to study how easy it is to use trackers and how usable anti-stalking features are. Gerhardt et al. [27, 26] ran a user study looking into when participants receive an anti-stalking notification, to evaluate the reliability of unwanted tracking detection and how users react to receiving these notifications.