Security and Privacy Analysis of Tile's Location Tracking Protocol

Akshaya Kumar
Georgia Institute of Technology

Anna Raymaker
Georgia Institute of Technology

Michael A. Specter
Georgia Institute of Technology

Abstract

We conduct the first comprehensive security analysis of Tile, the second most popular crowd-sourced location-tracking service behind Apple's AirTags. We identify several exploitable vulnerabilities and design flaws, disproving many of the platform's claimed security and privacy guarantees: Tile's servers can persistently learn the location of all users and tags, unprivileged adversaries can track users through Bluetooth advertisements emitted by Tile's devices, and Tile's anti-theft mode is easily subverted.

Despite its wide deployment—millions of users, devices, and purpose-built hardware tags—Tile provides no formal description of its protocol or threat model. Worse, Tile *intentionally* weakens its antistalking features to support an antitheft use-case and relies on a novel "accountability" mechanism to punish those abusing the system to stalk victims.

We examine Tile's accountability mechanism, a unique feature of independent interest; no other provider attempts to guarantee accountability. While an ideal accountability mechanism may disincentivize abuse in crowd-sourced location tracking protocols, we show that Tile's implementation is subvertible and introduces new exploitable vulnerabilities. We conclude with a discussion on the need for new, formal definitions of accountability in this setting.

1 Introduction

Tile is one of the most popular Bluetooth-based location-tracking services. It has mature applications on both Android and iOS, and as of September 2021, Tile had sold over 40 million devices and had over 425,000 paying users [39]. The company has partnered with 19 third-party manufacturers—including Dell, Bose, and Fitbit—to embed its protocol into laptops, headphones, and smartwatches. In June 2021, Tile devices were integrated into Amazon's Sidewalk network, which claims coverage of 90% of the US population [2,6]. In November 2021, the family communication service Life360 [32] acquired Tile for $205 million, expanding Tile's network by 33 million smartphones [31].

In this paper, we present the first comprehensive security analysis of Tile's location tracking protocol. Our analysis reveals that Tile is vulnerable to several attacks that compromise the privacy and security of its users. For example, we find that Tile's servers collect location information on millions of devices, effectively running a mass surveillance network (§6.1). Furthermore, we demonstrate that a third-party RF/network adversary can easily monitor a user's physical movements by passively observing the Bluetooth advertisements emitted by the user's Tile tracker (§6.2). We also show that malicious users can exploit Tile trackers to stalk their victims (§6.3), and discover that Tile's novel Anti-Theft Mode and accountability mechanisms are susceptible to abuse and subversion (§6.4 and §6.5). We provide a detailed analysis and experimental verification for these vulnerabilities, showing that exploitation is within the reach of a motivated individual or malicious server.

Although Tile is one of the oldest Bluetooth-based crowd-sourced location tracking system, it competes with similar services from Apple, Samsung, and Google. Jointly with Tile, these vendors have sold hundreds of millions of devices.

As these cryptographic services, collectively known as *Offline-Finding* (OF) networks, become increasingly widespread and integrated into everyday devices, their potential for misuse is growing. Bluetooth-based tracking tags have facilitated several criminal incidents, including theft, stalking, physical assault, and even murder [3, 9, 10, 16, 34, 41, 42]. In these cases, a malicious user places their tag on a target (person or item) to monitor them via the OF network.

To prevent such abuse of OF networks, designers implemented "anti-stalking" features, allowing potential victims to detect rogue tags that may be traveling with them. These include Apple's Item Safety Alerts, Samsung's Unknown Tag Alerts, and Tile's Scan and Secure feature.

While these features help detect rogue tags, the questions remain: *What recourse does a stalking victim have once they have discovered a malicious tag? How can abusers of OF networks be held accountable?* This is where Tile differentiates itself from other OF providers: it is the *only* company that

Adversary	Attack Compromises			
	Unlinkability (tag indistinguishability)	Anti-stalking alerts (tag detectability)	Location privacy (location indistinguishability)	Framing resistance (framing resistance)
Passive RF	✓			
Active RF	✓			✓
Malicious owner		✓		
Malicious tag		✓		✓
Platform server	✓		✓	✓

Table 1: A summary of attacks by adversary type. We show what kind of adversary is capable of executing what sort of attack; e.g., a passive RF adversary can break Tile's unlikability/tag indistinguishability property, allowing the attacker to link Bluetooth advertisements over arbitrary time periods to a device. The viability of these attacks may be dependent on the configuration of the tag. Our results are for the Tile Mate 2022. We define adversary types and security properties in §2.2.

claims to offer accountability.

To contextualize Tile's accountability guarantees, we first describe its *Anti-Theft* mode. In February 2023, Life360 CEO Chris Hulls claimed in an independent blog post [22] that, "*an unintended consequence of these [antistalking] initiatives has been the neutering of the ability of Bluetooth locators to help recover stolen items after a theft.*" The rationale was that a thief could run an antistalking scan, detect the tag, and discard it, eliminating its usefulness in recovering stolen items. Soon after, Tile launched their *Anti-Theft mode* feature, making Tile trackers undetectable by anti-stalking features.

To discourage abuse of the Anti-Theft mode for stalking, Tile implemented an accountability mechanism. Users of the Anti-Theft mode must verify their identity with a government-issued ID and selfie, and acknowledge that their information will be shared with law enforcement at Tile's discretion. They must also consent to a $1 million fine if convicted of using Tile trackers for stalking.

While there are legal and practical questions surrounding the enforceability Tile's fine, there is academic interest in understanding the technical barriers and tradeoffs in implementing accountability in OF systems. For example, an accountability mechanism must support dispute resolution, allowing a victim to prove they are not attempting to frame an honest owner, and avoid universally puncturing anti-surveillance and other privacy guarantees. Understanding Tile's design can provide a useful case study in real-world design of accountability in Offline Finding systems.

In addition to accountability, Tile makes other strong claims about their platform's security and privacy. Tile's privacy policy [29] claims that "...[A]ny information transmitted across our network is anonymous. You are the only one with the ability to see your Tile location and your device location." However, Tile's privacy policy also links to Life360's general privacy policy, which appears to cover their entire suite of products and admits to a more expansive set of collection activities (including location data). [1]

Despite their bold claims, there is little public information about the design of Tile's system. Tile's support page [43] and privacy policy [29] provide only a vague description of its system and threat model. Therefore, we reverse engineer Tile's publicly available Android application to analyze the security of its OF network. For unknowable parts of Tile's private infrastructure, we make optimistic assumptions. We show that the attacks we identify are exploitable even under these optimistic assumptions.

Our Contributions. We present the first comprehensive security analysis of Tile's OF protocol. In particular, we:

- reverse-engineer and reconstruct Tile's protocols for tag registration, location reporting, Bluetooth advertisement generation, and its anti-stalking feature;
- identify and experimentally validate several vulnerabilities that allow adversaries with different capabilities to violate unlinkability, location privacy, and framing resistance, as summarized in Table 1;
- analyze Tile's novel Anti-Theft mode and accountability mechanism, showing that both can be circumvented and misused for stalking or framing attacks;
- compare Tile with other contemporary OF networks, such as Apple's Find My and Google's Find My Device, and show that it provides substantially weaker security.

Outline. We begin §2 with background on OF systems and their security, and state Tile's claims of security. We then summarize prior work in §3. In §4, we describe our reverse engineering methodology. In §5, we detail Tile's protocol as discovered in our methodology, covering registration, the OF protocol, custom cryptography, and a brief discussion of the parts we could not confirm. We present the attacks we identified against Tile's protocol in §6. We compare Tile's security and privacy guarantees with those of other major service providers like Apple and Samsung, and provide lessons learned and recommendations for incorporating accountability in OF systems in §7. Finally, we conclude in §8.

[1] The Federal Trade Commission (FTC) has indicated via regulatory action that statements in privacy policies do not override a falsehood elsewhere [11].

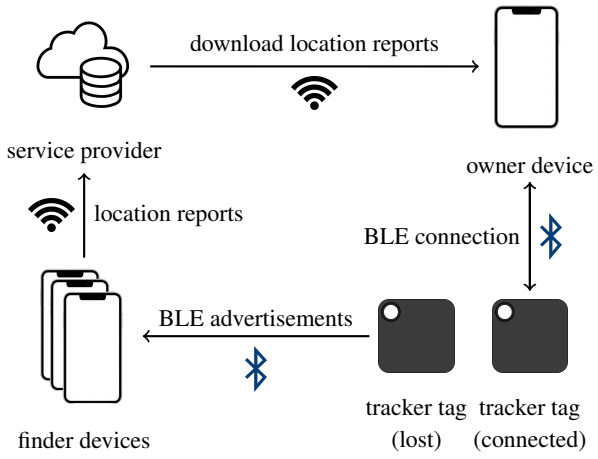

Figure 1: The components involved in an OF protocol.

2 Background

We begin this section with an overview of a generic OF system, explaining the various entities involved, their interactions, and communication channels. We then describe the threat model and security requirements commonly considered in the literature for both deployed and proposed OF systems, and with an overview of Tile's security claims.

2.1 Offline finding systems

We briefly sketch the devices and steps used by OF systems. We begin by broadly describing Bluetooth Low Energy (BLE) communication that forms the core of OF protocols.

Bluetooth Low Energy (BLE). BLE communication occurs through advertisements or full connections. Advertisements are connectionless and used for device discovery and connection initiation. Advertising packets contain devices' identity, capabilities, and services. These packets are received by nearby devices scanning for BLE advertisements. Connections allow bidirectional communication, and are established through the Generic Attribute Profile (GATT) protocol [4].

An overview of OF protocols. We summarize the components involved in an OF protocol and their interactions in Figure 1. First, a user buys a *tracker tag*, a small battery-powered device with low memory and computing power that has BLE capabilities but no access to the internet. Then, they pair the tracker tag with *owner devices* like smartphones or tablets that are more powerful than tags; they have GPS and internet capabilities. A user's owner devices share the same account. Owner devices and associated tags communicate with each other over a BLE connection. A tag is considered "lost" when it loses Bluetooth connectivity with its owner device (otherwise, it is considered "connected"). Similarly, an owner device is considered lost if it disconnects from the internet. A lost tag/device sends out BLE advertisements containing an identifier to be discovered by finder devices. *Finder devices* are bystander devices on the service provider's network that have GPS and internet capabilities and scan for BLE advertisements to discover lost tags/devices.

Once a lost tag/device is detected, the finder records its own location and sends it to the service provider's servers along with the lost tag/device's identifier. The *service provider* maintains a database of location reports and identifiers submitted by finder devices. Finally, an owner device can retrieve the locations of their lost tags/devices by querying the service provider for their tags' identifiers.

2.2 Security properties and threat models

The security of OF networks is an emerging area of research, with several recent papers studying existing networks and proposing new constructions. In this section, we present security properties and corresponding threat models, adopting definitions from prior work, and define *framing resistance*, a new definition that is a necessary precursor to accountability in OF systems.

Tag indistinguishability against passive and active RF adversaries. If a tag consistently broadcasts a unique identifier (e.g., a static MAC address), then an adversary can easily link the identifier to its owner and track their movements.

A *passive RF adversary* is an attacker equipped with a Radio Frequency (RF) detector, allowing inspection of Bluetooth emissions from nearby tracker tags. The passive RF adversary is also referred to as the "tracking adversary" [13], and could use information learned from Bluetooth advertisements to "track" the corresponding owner. In the worst case, an RF adversary could set up a network of RF sniffing devices to listen for BLE advertisements broadcast by several thousand trackers across a large area, deanonymize corresponding owners, and monitor their movements. An *active RF adversary* has the capabilities of the passive RF adversary, but can additionally emit arbitrary Bluetooth packets.

Manufacturers protect the privacy of owner devices by designing their tags to emit periodically changing identifiers such that any two identifiers broadcast by a tag are cryptographically indistinguishable from identifiers broadcast by two different tags. This property is formally referred to as *tag indistinguishability* [13, 35].

Tag detectability against malicious owners and tags. A *malicious owner* is an adversary that surreptitiously attaches its own tag to a victim and leverages the OF network to monitor the victim's movements. This type of adversary is also known as the "stalking adversary" [13]. One can think of a malicious owner as an honest-but-curious adversary who does not deviate from the protocol specification but tries to learn information it is not privy to. This cannot be fully prevented, as it is a natural consequence of the ability to track tags, but OF

networks should be designed to be resistant to such abuses.

To prevent network abuse for stalking, manufacturers implement anti-stalking features that guarantee *tag detectability* [13], allowing a user's device to determine if identifiers recorded at different locations and times originated from the same tag. Apple and Google's OF systems disable the periodic rotation of a tag's identifier once it is separated from the owner's device, forcing it to advertise static identifiers. Intuitively, this weakens the tag indistinguishability property, but allows the victim's device to alert the user if the same (unknown) identifier is seen over a suspiciously long time and distance.

A *malicious tag* adversary builds counterfeit tags to bypass stalking protections (e.g., by continuing to periodically rotate tag identifiers even in the lost mode). These counterfeit tags otherwise mimic the functionality of manufacturer-produced tags and can be tracked using the provider's network, but remain undetected by anti-stalking features. Ideally, an OF system should offer tag detectability against both malicious owners and malicious tags.

Location indistinguishability against the platform server. To prevent the server from learning the locations of the finders and owners based on the reports submitted, an OF system should provide *location indistinguishability* [13, 35]. This property guarantees that the server cannot learn location information from reports submitted by finder devices. Providers like Apple and Google achieve location indistinguishability by end-to-end encrypting location information using a public key embedded in BLE advertisements emitted by a tag.

Accountability and framing resistance. While stalking resistance allows for detection of a stalker's tag, an *accountability mechanism* allows the victim, platform, or an external authority to disincentivize misuse of the platform by punishing bad behavior. This can be achieved by cryptographically tying the identity of owners to the protocol so that, in the event of misuse, a victim collaborating with appropriate authorities can pierce the tag indistinguishability property to punish the perpetrator.

A core challenge is ensuring that an accountability mechanism does not allow an attacker to provide convincing proof that an innocent user owns a tag — we call this property *framing resistance*.

2.3 Tile's claims of security

Although there is no public, formal description of their protocol or threat model, Tile makes several claims about their system's security properties via their privacy policy [29], support pages [24, 26, 28], blog [23, 25], and FAQ [27].

Tag indistinguishability. Tile claims that "any information transmitted across our network is anonymous" [29] and "Location updates are automatic and anonymous, so it's completely secure" [26].

While explaining the results of Scan and Secure [27], Tile claims that "The Reference ID [returned by Scan and Secure] is a unique and encrypted identifier of the Tile or Tile-enabled device that intermittently changes to ensure the privacy of the Tile owner. For example, if another person running the Scan and Secure feature detected a Tile that you own, they would not be able to identify you as the Tile owner."

Location indistinguishability. In its privacy policy [29], Tile states that "You are the only one with the ability to see your Tile location and your device location." Furthermore, they claim that, "You won't be able to see where other people's Tiles are, and they won't be able to see yours" [26].

Tag detectability. Tile guarantees that their Scan and Secure feature allows users to "detect nearby Tiles and Tile-enabled devices" that may be traveling with them.

Tag (un)detectability in Anti-Theft mode. Tile promises that once a user activates the Anti-Theft mode, all their Tiles "will no longer be discoverable by our Scan and Secure feature and certain other 3rd party scanning tools" [24].

Accountability. To prevent the misuse of the Anti-Theft mode for stalking, Tile claims to have implemented "strict safety measures." This includes requiring a user to verify their identity using a government ID and a live photo before activating Anti-Theft mode. Tile also states that they will impose a "$1 million fine for any individual convicted in a court of law for using Tile devices to illegally track any individual without their knowledge or consent." It is unclear how enforceable the above agreement is or how such a proposal would work in a region without the rule of law.

Somewhat alarmingly, Tile has made inconsistent statements regarding when it will share information with law enforcement. On the one hand, Tile specifies in the FAQ page [27] that they will "work with law enforcement through a properly issued court order to identify the owner of a suspicious Tile using the Reference ID." On the other hand, they require users of the Anti-Theft mode to agree that their "personal information can and will be shared with law enforcement at our discretion, even without a subpoena."

Tile asserts that their system can generate legally admissible proof that a device used for stalking is owned by a particular user, without detailing the validity of this proof. This capability is directly presented as a solution to the weaknesses introduced by the Anti-theft mode. While Tile's policy avoids explicitly using the term "accountability," this assertion is specifically framed around Tile's protocol for identifying and punishing perpetrators of malicious tracking.

3 Related work

Previous studies have examined Tile's protocol only in passing, within the broader context of Bluetooth-based tracking technologies. None have analyzed its overall threat model,

protocol, potential for surveillance, or accountability mechanism, anti-stalking, and Anti-theft features. We are the first to analyze Tile's protocol exhaustively and find the vulnerabilities presented in §6.

Prior work on Tile. Weller et al. [47] performed the first security and privacy analysis of several Bluetooth trackers, including Tile. They reported that the app sends metadata—including the currently used Wi-Fi name and Wi-Fi MAC address—to the server. Garg et al. [14] outlined a set of desirable security guarantees for crowd-sourced location tracking systems and analyzed whether various services, including Tile, met these guarantees, but provided limited analysis on each. Various works [15,38,45,46] have analyzed the forensic information stored by Tile on a user's phone. These studies revealed that an entity with temporary physical access to a user's Tile-enabled phone can recover (timestamped) location information about the user, their Tiles, and even other users for the past 30 days.

Heinrich et al. [21] and Turk et al. [44] examined Tile as part of a broader study on the effectiveness of anti-stalking mechanisms employed by various OF service providers. They concluded that the guarantees provided by Tile's Scan and Secure feature were inadequate in their threat models.

Security analysis of other OF networks. Several works have analyzed the security of widely deployed OF protocols, including Apple and Samsung's implementations.

Heinrich et al. [20] reverse-engineered Apple's FindMy protocol and provided its first technical specification. Their work revealed design and implementation flaws that could potentially result in location correlation attacks and unauthorized retrieval of location histories. They also implemented the OpenHaystack tool [19] that helps create and use custom FindMy devices. Mayberry et al. [36] and Heinrich et al. [18] analyzed the security of Apple's Item Safety Alerts and presented ways to circumvent these alerts using custom tags. Bräunlein showed the use of the FindMy network for covert communication [5]. Yu at al. [50] and Liu et al. [33] bfound several vulnerabilities in their analysis of Samsung's OF protocol which focused on the unlinkability of tags and privacy of location reports. The security of both Apple's and Samsung's OF protocols has also been analyzed in a broader threat model [1]. Google's protocol was recently reverse-engineered by Botteger et al. [7], revealing vulnerabilities that allow a malicious owner to evade detection and compromise privacy. All these vulnerabilities resulted from bad design, just as we will see is the case with Tile.

Proposed (cryptographic) solutions for OF systems. Mayberry et al. [35] formalized security notions for a crowd-sourced location tracking system. They propose a modification of the cryptographic protocol underlying Apple's FindMy network that achieves their security notions. In particular, their protocol leverages partially blind signatures to guarantee that that attackers cannot add non-server authorized malicious tags to the network.

Eldridge et al. [13] formalize strong security definitions for privacy-preserving abuse-resistant location tracking protocols that guarantee robust stalker detection whilst maintaining user privacy. They also construct a new OF protocol using secret-sharing and lattice-based algorithms as building blocks. Engineers at Google and Apple have created Detecting Unwanted Location Trackers (DULT) [30], a draft IETF specification. David et al. [12] present advanced security and privacy notions for OF protocols and present XDHIES, a cryptographic tool that enables achieving these properties.

Anti-stalking studies. Several studies have analyzed anti-stalking countermeasures implemented by service providers. Heinrich et al. [18] analyzed Apple's anti-stalking mechanism and found that, while Apple's solution protects iOS users, their anti-stalking application for Android was insufficient. They developed Airguard, an Android application that better detects the presence of rogue AirTags around Android users. Airguard also detects rogue tags from other manufacturers, including Samsung and Tile. Müller et al. [37] analyzed the potential for misuse of various trackers and developed a similar application.

In recent work [21], Heinrich et al. measured the misuse of Bluetooth trackers, particularly Apple, Tile, Samsung, and Chipolo, for unwanted tracking and stalking. The study used data from the AirGuard app and a large user survey. It reported that over 40% of stalking victims had been subjected to location tracking by tracker tags. Turk et al. [44] also investigated the effectiveness and limitations of several service providers' antistalking mechanisms. They highlighted that in many cases, victims do not use these features, even if they suspect that they are being stalked.

Tracking BLE devices. Celosia et al.'s analysis of the BLE advertising mechanism [8] showed that advertisements may contain information that uniquely identifies a Bluetooth device, despite MAC address randomization. They also note that many devices continue to broadcast static addresses, even though identification and subsequent tracking of Bluetooth devices emitting static MAC addresses is a known issue.

4 Methodology

In order to gain a thorough understanding of Tile's infrastructure, we began by decompiling the most recent version (2.125.0) of their Android application as found on the Google Play Store. We used a Google Pixel 3XL, jailbroken using Magisk [48], as our test device. For our experiments, we used the Tile Mate 2022 running the pre-activation firmware version 48.04.16.0 and post-activation firmware version 48.04.28.0.

We then studied the decompiled code to learn various aspects of Tile's protocol, including the registration process, generation of cryptographic secrets, location reporting and

retrieval, Scan and Secure, and Anti-Theft protocols. Additionally, we performed dynamic analysis to inspect BLE messages exchanged between the Tile and other devices, and network traffic analysis to inspect HTTP messages sent between tags, devices, and the server. Our experimental methodology for verifying the vulnerabilities we describe in §6 overlapped with the dynamic analysis we performed in the reverse engineering process. We describe any differences in experimental methodology in §6.

Static analysis. We performed static analysis on Tile's Android application to study the closed-source implementation of Tile's OF protocol. In particular, we used the JADX [17] decompiler to convert the Tile APK into Java-like source code. We then analyzed the obtained source code to piece together Tile's OF protocol.

Android Bluetooth log analysis. By enabling the Bluetooth HCI snoop log option, Android records Bluetooth communication between the mobile device and peripherals. This allowed us to examine data exchanged between a phone and a tag over BLE.

Network analysis. We analyzed the messages exchanged between a Tile-enabled Android client with other devices/parties by monitoring network traffic. To monitor HTTPS traffic between our test device and the server, we used the BurpSuite tool to set up a proxy to man-in-the-middle and decrypt all incoming and outgoing traffic from our device.

Limitations of our study. Our study focuses exclusively on Tile's most common tracker, the Tile Mate 2022. Tile manufactures several other models and also collaborates with third-party manufacturers. Our findings might not apply universally across all devices. Furthermore, our analysis does not include access to Tile's server backend, limiting our understanding of server-side processes. We make best-case assumptions regarding those operations in our analysis of their protocol.

5 Tile's offline finding protocol

In this section, we outline the OF protocol executed between a Tile tracker (tag), a phone that is paired and connected to the tracker (owner), the Tile server(s), and a network of bystander devices using the Tile application (finders). We break the protocol down into multiple phases — owner/finder registration (§5.1), tag activation (§5.2), tag-owner interactions (§5.3), tag-finder-server interactions (§5.4), owner-server interactions (§5.5), Scan and Secure (§5.6), the Anti-Theft mode (§5.7), tag transfers and sharing (§5.8), Community Information (§5.9), and account deletion (§5.10).

We begin by providing a brief overview of the protocol. The first step for the owner, the tag, and the finder is registration. For the owner and the finder, this involves creating an account with Tile using an email address and a password over an active network connection. The tag is activated by registering it with an owner device. The owner acts as a proxy between the tag and the server during this step. An activated tag transmits BLE advertisements containing a unique and rotating identifier. These advertisements are picked up by finder devices that extract the identifier from the BLE advertisements and store them in a local database. Once the finder connects to the internet, it uploads the recorded identifiers along with its current location to the Tile server. Finally, the owner of a tag can query the server for location information about its tag. We explain each of these steps in detail below.

5.1 Owner/Finder registration

We now describe the registration process for owner and finder devices. In the Tile ecosystem, these devices are smartphones or tablets.

To register, an owner first downloads the Tile Android application and creates an account using an email address and password. During this process, the owner's Android device generates a static 16-byte client_uuid that is unique to the particular device. Then the device sends a POST request to the Tile API server at production.tile-api.com/api/v1/users, containing client_uuid and the owner's email ID and password. The user is sent a 6-digit verification code on the email ID they provided. Notably, the user isn't required to prove ownership of the email; this step is skippable. The response from the server includes a 16-byte user_uuid assigned by the server and contains a status field whose value is set to "ACTIVATED". The user_uuid is used to identify the owner in future communications with the server. Next, the device issues another POST request at /api/v1/tiles/generate_tileUUID, containing the tile_uuid, the user_uuid, and tile_type where tile_uuid is set to client_uuid, user_uuid is the value previously returned by the server, and tile_type is set to "PHONE". The server's response overwrites tile_uuid with a fresh 16-byte value prefixed with the string "p!" for device-type identification. This completes registration for the owner.

5.2 Tag activation

We now describe the process by which a Tile tracker is activated and associated with an owner device. We summarize the interactions between the owner device, the tracker tag, and the server during the tag activation phase in Figure 2. We describe each step below.

Tile Device Information (TDI) [Steps 1,2 in Figure 2]. Before a Tile tracker is activated, it broadcasts advertisements containing the pre-activation UUID 'FEEC'. The tracker owner initiates a Bluetooth scan through the Tile Android app to activate it. The results of this scan are filtered using the 'FEEC' UUID. The owner requests the tileId, model, firmware, and hardware_version from the tag over the custom Tile Device Information (TDI) service (UUID 180A). The

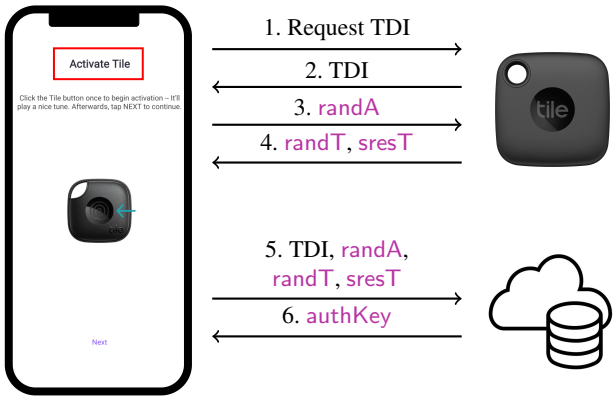

Figure 2: An overview of the various steps involved in Tile tracker activation.

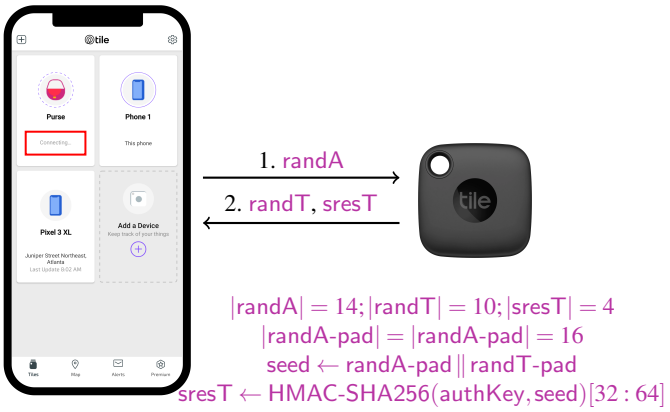

Figure 3: The authentication protocol used by the Tile tracker to authenticate itself to the owner device.

corresponding characteristic UUIDs for these variables are defined in Table 4 in Appendix B.1.

Tag authentication [Steps 3,4 in Figure 2]. During the activation process, the owner establishes a shared secret called the authKey between itself, the server, and the Tile tag. To initiate this process, the owner's phone connects with the GATT server on the tag. After the connection is established, the tracker exposes its 'FEED' service to the owner, and the owner and the tracker execute a challenge-response protocol for authenticating the tag to the server.

The owner samples a 14-byte uniformly random value called randA using Java's SecureRandom random number generator and sends it to the tag. Subsequently, the Tile tag generates a 10-byte random value randT. The tag then computes a 4-byte value sresT by applying the HMAC-SHA256 hash function to randA, randT, and tileId using a 16-byte interim key interimAuthKey that is provisioned to a vendor by Tile and stored on device at manufacture time. Essentially, during this step, the tag proves knowledge of the interimAuthKey to the server (using the owner device as a proxy) to authenticate itself as a legitimate device.

Secret key establishment [Steps 5,6 in Figure 2]. The tag derives the permanent secret key—authKey—by applying the HMAC-SHA256 hash function to the interimAuthKey and the sresT value obtained above. The authKey is the shared secret between the owner device and the tag. However, the owner device does not know the interimAuthKey and cannot derive authKey itself. Instead, the owner device creates an HTTPS POST request to the server containing the information obtained during TDI as well as randA, randT, and sresT. We describe the structure of the request body in Appendix A.

The server first checks that randA, randT, and sresT form an unused and valid authentication triplet for the queried Tile tracker by checking the relation between them. If the check fails, the server rejects the query. Otherwise, the server derives the 16-byte authKey analogously to the tag, and returns the authKey to the owner. This step completes the establishment of the authKey between the tag, the owner, and the server. It is important to note that the server knows the authKey for every tag. The authKey is subsequently used to produce pseudonymous identifiers for the tag to broadcast.

Generation of pseudonymous identifiers. The owner device, tag, and server use the authKey to derive 8640 identifiers, called privateIds, for the tag. The privateIds are generated as follows where tileId is the unique identifier for the tracker tag, identityBytes is the little Endian byte encoding of the string "identity", and ctr is a counter initialized to 0 and incremented after each invocation of HMAC-SHA256. The concatenation tileId || identityBytes is padded to 32 bytes. The output of HMAC-SHA256 is truncated to the first 64 bits.

$$\text{seed} \leftarrow \text{HMAC-SHA256}(\text{authKey}, \text{tileId} \,||\, \text{identityBytes})$$
$$\text{privateId} \leftarrow \text{HMAC-SHA256}(\text{seed}, \text{ctr})[0:64]$$

The tag continuously emits BLE advertisements containing a privateId, starting at the value corresponding to the counter 0, and rotating to the next value every 15 minutes. With 8640 unique values, this rotation cycle repeats every 90 days.

5.3 Tag-Owner interactions

In this subsection, we discuss the interactions between the owner device, the tag, and the server after the tag has been activated. Most of these interactions facilitate ancillary features of the tag, such as ringing the tag and using the tag to (reverse) ring the owner. In particular, these steps—with the exception of location reporting, which we describe in this section—are not relevant to the OF protocol. We include them in §B.2 for completeness.

Establishing a connected channel: Tag authentication. Following tag activation, all tag-owner communications are transmitted over a connected channel. In order to establish a connection, the owner and the tracker authenticate each other

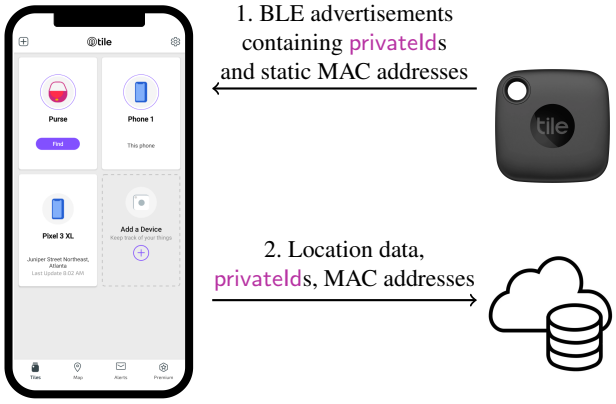

Figure 4: Location reporting in the lost mode.

using Tile's custom challenge-response protocol. We describe owner authentication in Appendix B.2. Tag authentication follows the protocol described in Figure 3. This protocol is similar to Steps 3 and 4 in Figure 2. The only difference is that the tag uses the authKey instead of the interimAuthKey to derive sresT from randA and randT as follows. Bytes 4-7 of the HMAC-SHA256 output are assigned to sresT.

$$\text{seed} \leftarrow \text{randA-pad} \,||\, \text{randT-pad}$$
$$\text{sresT} \leftarrow \text{HMAC-SHA256}(\text{authKey}, \text{seed})[32:64]$$

Here randA-pad and randT-pad are obtained by padding randA and randT with 0s to make them 16 bytes long. Essentially, the tag proves its knowledge of authKey to the owner in this step.

Location reporting by the owner. Once a connection has been established, the owner device periodically sends location reports to the server through an HTTPS POST request containing its location information—altitude, latitude, longitude, and timestamp—along with a list of tileIds and corresponding authentication data—randA, randT, sresT—for each connected tag. We detail the request body in Appendix A. Interestingly, the owner periodically uploads location reports for a Tile tag to the server even when it is connected to the tag. This step is not necessary for the functionality of an OF protocol and poses privacy risks that we will discuss in §6.1.

5.4 Tag-Finder-Server interactions

We now outline the core of Tile's OF protocol. Specifically, we describe the interactions between a finder, a tracker tag, and the server. We summarize this phase in Figure 4.

Tag-Finder interactions. An activated tag rotates through privateIds every 15 minutes regardless of its connection to the owner. This means that Tile tags do not differentiate between lost and connected modes—they are *always* findable.

The advertisements emitted by a tag contain the tag's rotating privateId. A finder/bystander device running the Tile app executes a service that periodically scans for BLE advertisements in the background. The app filters BLE advertisements using the 'FEED' service UUID to get advertisements from nearby Tile tags. The finder device then parses the advertisement data and retrieves the tag's privateId and MAC address. It checks if the privateId corresponds to any of its own tags. If not, the finder adds the privateId and MAC address to a database of seen tags.

Finder-Server interactions. Periodically, the finder device also runs a service that batches multiple scanned advertisements and computes its own location at the time. It then uploads its location and the batched advertisements along with its own clientID to Tile's servers via an HTTPS POST request. We describe the request body in Appendix A. The uploaded advertisement data contains the *static* MAC address of the advertising tag as well as the corresponding privateId. Finder devices also uniquely identify themselves as the user uploading reports to Tile's server.

5.5 Owner-Server interactions

When the owner wants to track their Tile, they log in and query the tiles/location/history/{tileId} endpoint to retrieve location history for their Tile with Id tileId. The owner includes its user_uuid in the request header. The server first verifies whether the requested tileId is registered to the requesting user's user_uuid. If the verification succeeds, the server returns the location data for the corresponding Tile as reported by finders; otherwise, it rejects the query.

5.6 Scan and Secure

The Scan and Secure feature allows a user to trigger a scan for unknown Tiles or Tile-enabled devices that may be following the user. Unlike similar features provided by other service providers like Apple and Google, this feature is not built into the operating system or available to non-Tile users. It is not even enabled in the app by default. To run a scan, a user must manually trigger it by clicking the "Scan and Secure" button under the app's settings. The scan takes approximately 10 minutes and requires the user to be physically moving while performing the scan and monitoring results on the screen.

During the scan, the app performs six consecutive Bluetooth scans, filtering results based on the 'FEED' service UUID. It then parses the advertised data and retrieves the privateId from each advertisement. For each privateId, the app checks it against its local database of privateIds to determine whether the privateId corresponds to a connected/paired tracker. If so, it categorizes the corresponding result as a "Known Tile". All other advertisements are categorized as having come from an "Unknown Tile". For unknown trackers, the app sends an HTTPS POST request to the Tile server containing the corresponding privateIds for each scan. We provide the format of the request body in Appendix A.

The server responds with a subset of the privateIds that do not correspond to Tiles that are in the Anti-Theft mode. We explain the Anti-Theft mode shortly.

Finally, the app then displays two lists: one for "Known Tiles", where it shows the user-assigned local names of connected trackers, and another for "Unknown Tiles", where it displays the privateIds returned by the server along with the number of times (out of six) each privateId appeared.

5.7 Anti-Theft mode

Anti-Theft mode allows users to make their Tile trackers invisible to the Scan and Secure feature. To enable this mode, users must verify their identity by providing a government-issued ID and live photos. Tile relies on two third-party verification services, Berbix and Persona, to handle this process. Once verification is complete and the user agrees to Tile's terms, the server updates the user's Anti-Theft status to "enabled".

As a result, Tiles owned by that user are excluded from Scan and Secure results. In particular, when the server receives a list of privateIds during a Scan and Secure scan, it filters out any privateIds associated with Tiles in Anti-Theft mode, returning only those that do not have this setting enabled.

5.8 Transfers and Unlimited Sharing

Tile provides a feature for users to transfer ownership of their trackers to another user and share their Tiles with other users.

We discuss transfers first. To initiate a transfer, the current owner selects the Tile they wish to transfer, navigates to "More Options," chooses "Transfer," and enters the recipient's email address. This action triggers an HTTPS POST request to the Tile production API server, which includes the recipient's email address. Upon successful transfer, the original owner receives an email notification, though the recipient is not explicitly notified. Instead, the transferred Tile automatically appears in the recipient's list of connected Tiles when they log in or reopen the Tile app. Notably, the server forwards the authKey from the original owner to the new recipient—there is no attempt to update the authKey after transfer.

Now we discuss Tile's "Unlimited Sharing" feature. Tile allows users to share their Tiles with other users. In order to share a Tile, a user selects the Tile they wish to share, chooses "Unlimited Sharing" and enters the email address of the new shared owner. Then the app sends an HTTPS request to the server containing the tileId of the Tile and the email address of the new owner. The server associates the corresponding tag with the new owner and sends them the authKey. We describe the server's response body in Appendix A.

Unlike the transfer feature, neither the original owner nor the new shared owner is explicitly notified about the sharing of the Tile. The tile appears on the new user's screen and is displayed as a shared tile. To stop sharing a Tile with a user, an owner selects the Tile on the app's home screen, chooses

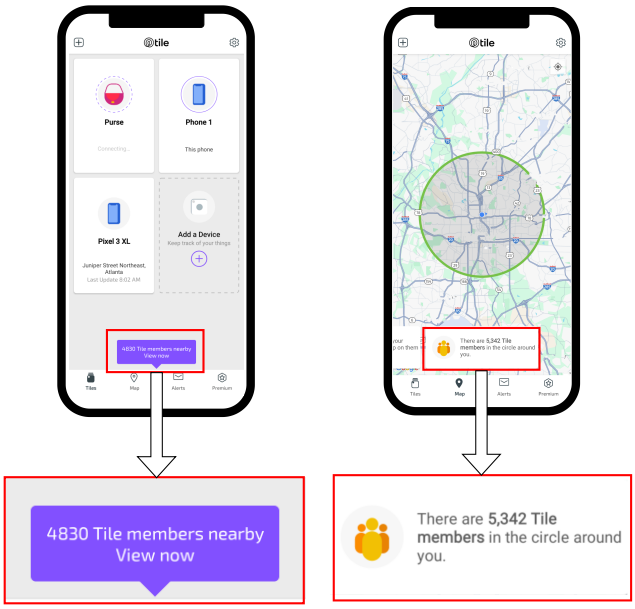

Figure 5: Tile's community information feature.

"Unlimited Sharing" option, and selects the corresponding shared user. This initiates an HTTPS DELETE request including the Tile's tileId and the shared owner's email. The tag is removed from the shared owner's home screen. Importantly, the authKey is not updated even after the original owner stops sharing their Tile with another user.

5.9 Community information

Tile has a feature called Community Information that is not directly part of its OF protocol. This feature displays the number of Tile users in a 5-mile radius around the owner. We show a screenshot of the feature in Figure 5. When a user opens the Tile app, their client sends an HTTPS GET request to the Tile API server at /api/v1/community/stats, including the latitude and longitude of the client's current location. The server responds with the number of Tile users in the 5-mile radius around the client. We describe the server's response format in Appendix A. We note that this feature is not useful for the functionality of the OF network.

5.10 Account deletion

To delete a Tile account, the user must navigate to the Settings menu and select 'Delete Account' under 'Manage Account'. The process requires the user to enter their password and acknowledge that their Tiles will no longer be functional and cannot be reactivated, and that account deletion is a permanent action. Additionally, users must confirm that, although their account details and personal information will be removed from Tile's servers, "Tile may need to retain certain

data" per Tile's privacy policy. Finally, the user is required to type in 'DELETE' to confirm account deletion. This triggers an HTTPS DELETE request to the Tile API server at /api/v1/users/user_uuid. Upon deletion, the server sends an HTTP 202 Accepted status message.

6 Privacy and security analysis

We now detail vulnerabilities that we identified against Tile's OF protocol, classifying them into those that (1) allow the platform server to persistently learn the location information of owners and finders by breaking the location indistinguishability property (§6.1), (2) compromise owner privacy by breaking the tag indistinguishability property (§6.2), (3) enable stalking by breaking the tag detetctability property (§6.3), (4) allow the framing of honest users by breaking the accountability mechanism (§6.4), and (5) allow the circumvention of the Anti-theft mode (§6.5).

To enable independent verification, we describe the experimental methodology we used to confirm the practical exploitability of the vulnerabilities we report.

6.1 Violating location indistinguishability

We outlined in §2.2 that an OF protocol should ensure location indistinguishability against the platform server. However, vulnerabilities in Tile's implementation undermine this property, allowing (1) Tile's server to run a mass surveillance network and (2) any third-party attacker with the Tile application to systematically deanonymize and track Tile users. These issues transform Tile's infrastructure into a global tracking network.

1. Tile's servers continuously collect unencrypted location information of its users. Tile's protocol allows the platform server to persistently collect unencrypted location information about all Tile users. For reports submitted by finder devices, this also includes the static MAC address of the corresponding tag. Other service providers (like Apple and Google) encrypt the location using the destination's public key and only allow their tracker tags to be discoverable by non-owners when in the "lost" mode i.e., the trackers are only detectable by finder devices when disconnected from the owner's device.

This compromises location indistinguishability and directly contradicts the claim made in Tile's Security and Privacy Policy [29], which tells users that they are the *only* ones who can access the location of their device(s) and Tile(s).

We verified our claim as part of the network traffic analysis we performed to reverse engineer the Android application. While intercepting the HTTPS traffic between the Tile Android client and Tile's servers, we confirmed that location reports were transmitted over TLS in plaintext (see Appendix A).

2. The Community information feature compromises users' location privacy. The Community Info feature may allow an attacker to deanonymize the identities of Tile users. Community stats display the number of Tile users in a 5-mile radius. An attacker could query the API endpoint on various latitude and longitude values and use these aggregate values to deanonymize users. We note that attacks in this setting have been well documented in the literature [40, 49].

The service provider could mitigate this attack by rate limiting these queries or obfuscating the results. However, implementing rate limiting is non-trivial since an adversary could just create multiple accounts and make one query using each account. Due to ethical and legal concerns, we chose not to send any irregular traffic to Tile's servers. Hence, we were unable to confirm if this vulnerability is exploitable.

6.2 Violating tag indistinguishability

If an OF protocol does not guarantee tag indistinguishability against passive RF adversaries, then it can be transformed into a pervasive surveillance infrastructure. To motivate this, consider a situation in which an attacker (e.g., a government agency, cell service provider, or app developer) with an RF receiver is able to uniquely identify a BLE advertisement as having been broadcast by their target's tag. The attacker could deploy RF receiver networks—for example, by using an app or SDK with fine-grained Bluetooth access—to log target devices' advertisements. By correlating these identifiers across time and geographic location, particularly in high-traffic areas like transit hubs or city centers, the attacker could construct detailed movement profiles of individuals without their knowledge or consent.

As outlined in §2.3, Tile's privacy policy [29] explicitly claims that all communications over their network are anonymous and that no one can learn where others' Tiles are. The following four vulnerabilities not only break the tag indistinguishability property as defined in the literature but also directly contradict Tile's claimed privacy guarantees.

1. The Tile Mate 2022 broadcasts static MAC addresses. MAC addresses broadcast by the Tile device are static. This allows a passive RF adversary to record MAC address broadcast by a Tile Mate and use them to identify and monitor the user's location. As a result, the work put into rotating privateIds in Bluetooth advertisements does not guarantee any privacy/anonymity to the owner of a Tile Mate.

To confirm that the Tile Mate 2022 broadcasts static MAC addresses, we recorded all BLE communication between our test phone and nearby peripherals. Over multiple iterations spanning several days, we captured BLE advertisement packets from the Tile Mate 2022 and analyzed the corresponding log files. We consistently observed the same static MAC address in all advertisement packets emitted from the device.

2. privateIds do not guarantee tag indistinguishability even if MAC addresses are randomized. There are 8640 unique privateId values corresponding to a particular Tile/Tile-

enabled device. A device rotates its privateId every 15 minutes. Instead of regenerating a new set of privateId values once the protocol uses all 8640 values, Tile begins re-using the same privateIds. As a result, TileIDs are repeated every 90 days. A passive RF adversary could record all the privateIds emitted by a tag over 90 days and use these values to uniquely identify the tag even if it has randomized MAC addresses.

To verify our claim, we conducted the following experiment. We programmatically logged in to our test user account on the Tile Android app and retrieved the authKey associated with a specific Tile Mate 2022 registered to the account. The algorithm used for generating all (8640) unique privateId values is detailed in §5.2. We used the algorithm to generate all 8640 unique values for our test device, and recorded the BLE advertisements emitted by the Tile Mate over 90 days by scanning for BLE advertisements. We compared the values captured with the set of previously generated 8640 privateIds, and observed that the privateIds repeated after 90 days, confirming that Tile reuses the same set of values.

This reuse pattern—together with the collision resistance property of the HMAC-SHA256 hash function used to generate the privateIds— allows an adversary to link privateIds over time and track the device. This is a realistic adversary: Individuals are likely to visit the same location at the same time of day (e.g., one's home or office) over long periods of time, and Tile devices are intended to be used for over a year.

3. The Scan and Secure feature does not guarantee the privacy of the owners of detected Tiles. Tile claims that the results of Scan and Secure present a "unique and encrypted identifier" that ensures the privacy of the corresponding owner [27]. We find that this claim is both technically incorrect and substantively misleading.

As described in §5.6, Tile's Scan and Secure feature's results contain the privateIds that appear during the six Bluetooth scans (as long as they do not correspond to a Tile in the Anti-theft mode), even if the privateId was recorded only once. Contrary to Tile's claim, the identifiers in the Scan and Secure result (privateIds) are pseudorandom, not encrypted.

Furthermore, the identifier does not provide privacy to the Tile owner, especially in the case of Tile Mate 2022, as the MAC address is static. The static MAC address is embedded in the advertisements recorded during the scan and allows the scanner to permanently fingerprint a device. This architectural flaw not only invalidates Tile's privacy guarantees but creates systemic surveillance risks—allowing any scanner to deanonymize innocent bystanders whose Tiles were incidentally detected during scans.

4. The Transfer/Unlimited Sharing features compromise the authKey. As discussed in §5.8, the transfer and/or sharing process does not regenerate or change the authKey—meaning that both users gain access to the same authKey used by the tag. Since privateIds are deterministically derived from the authKey, either recipient can generate all privateIds associated with the shared or transferred tag.

While transferring ownership is a permanent action, sharing a Tile can be revoked. However, even if a shared user's access is later revoked, they can still leverage the authKey or derived privateIds to track the original owner of the Tile.

This attack is particularly relevant in cases of Intimate Partner Violence (IPV), where Bluetooth trackers have been reported to aid the perpetrator. In this setting, an abuser might gain temporary access to their victim's phone, transfer or share the victim's Tile to themselves, obtain the authKey, and then transfer the Tile back or revoke their own shared access. With the authKey in hand, they could track/stalk their victim by generating all privateId values, then using RF receivers to track the victim using these values.

6.3 Violating tag detectability

The tag detectability property requires that unwanted tracking attempts be reliably detected by potential victims. With the growing use of Bluetooth-based trackers for stalking, this property is imperative in OF systems.

We identify fundamental flaws in Tile's design that violate this property, allowing malicious actors to evade detection and stalk users. The tag detectability property is trivially broken for the majority of users without Tile's app: To detect a malicious actor, a victim must install Tile's application, creating additional barriers to discovery. And, even if the app is installed, Scan and Secure has a number of usability issues: Scan and Secure is not a background process, and must be triggered manually in a sub-menu on the app. Worse, the scan duration is ephemeral, lasts about 10 minutes, and will not continue once the user navigates away. As we will discuss in §7, this reactive, opt-in approach contrasts sharply with the continuous and automatic background scanning implemented in competing systems.

Furthermore, Tile's Anti-Theft mode may be used to trivially violate tag detectability—Tiles in the Anti-Theft mode will not be detected by its Scan and Secure feature (unless, as we will discuss in §6.5, the user uses a modified app). Tile intentionally made this trade-off to support their anti-theft use case which is in direct tension with the tag detectability property. Though such tensions may be theoretically resolvable through careful design, Tile's implementation worsens the system's already weak detectability, and remains an open area of research.

6.4 Violating framing resistance

As discussed in §2.2, one of the main technical challenges of implementing an accountability mechanism for OF networks is achieving framing resistance. Below we describe replay attacks that exploit Tile's protocol to frame honest users.

1. Scan and Secure results are spoofable. As we outlined in §5, the Scan and Secure feature initiates 6 consecutive Blue-

tooth scans to detect advertisements from Tile or Tile-enabled devices, requiring the user to be in motion. Tile then extracts the privateIds embedded in the scanned advertisements and forwards the discovered privateIds to the server. The server returns only those privateIds that are not associated with Tiles in Anti-Theft mode.

An attacker with a compromised authKey (e.g., as described in §6.2) could execute a derive-then-replay attack to frame the compromised user. It would first derive privateIds from the compromised authKey, then broadcast these values at chosen locations. If a user runs the Scan and Secure feature in that location, the app will display a false positive, suggesting the presence of a Tile where none may exist. Indeed, there is no way to prove that the detected privateId was emitted by a legitimate Tile device.

2. Users can be framed for misconduct. We introduced framing resistance in §2.2. We claim that both active RF adversaries and the platform server can frame honest Tile users for misconduct.

An active RF adversary would first passively listen for and collect all the privateIds broadcast by an honest Tile. It would then simply replay the privateIds of the honest tag at an arbitrary location. Tile's Scan and Secure would record these replayed privateIds and display false positives to a user. Finally, Tile's server has the authKey for all Tile devices, and can therefore frame any user.

6.5 Circumventing the Anti-Theft mode

Tile uniquely markets its Anti-theft mode as a solution for theft protection, claiming tags in the Anti-theft mode cannot be discovered via Scan and Secure. However, our analysis reveals that this protection is superficial—while tags in Anti-Theft mode are excluded from the user interface, the system *still* records and transmits their privateIds during scanning operations. This design choice allows for trivial circumvention that fundamentally undermines Tile's advertised guarantees.

Anti-Theft mode can be circumvented by Tile users. Our analysis in §5.6 reveals the Scan and Secure feature records *all* observed privateIds, including those from tags in the Anti-Theft mode. The server merely filters these tags from the displayed results. A user with a modified app can easily circumvent the Anti-Theft mode by displaying all privateIds recorded during the scan.

To verify that Tile's Anti-Theft Mode can be circumvented, we conducted an experiment using two test accounts (A and B) on our test Android device. We enabled the Anti-Theft Mode in account A, and while logged in to account B on a different device, we ran Scan and Secure while in the vicinity of tags registered under account A. While analyzing network traffic from B, we confirmed that the HTTPS POST request sent to Tile's server contained privateIDs emitted by A's anti-theft-mode-enabled tag.

7 Discussion

A comparison of the security of major OF networks. To contextualize the security implications of Tile's system, we compare its protocol with three competing players in the OF landscape—Apple, Google, and Samsung—as well as the IETF's DULT draft standard. We evaluate these protocols with respect to the security properties defined in §2.2, and divide them across their privacy properties (Table 2) and their anti-stalking and accountability features (Table 3).

We start by discussing privacy properties. Each of the three major vendors and the DULT specification implements end-to-end encryption for location data under keys only accessible to the reporting finder and owner of the lost tag. Crucially, platform servers learn nothing about user locations—a fundamental privacy guarantee that Tile fails to provide. Previously, Samsung was shown to be inadequate on both accounts [50]—tag indistinguishability and location indistinguishability—but has since updated its protocol to improve tag indistinguishability and optionally provide location indistinguishability if a user explicitly enables it in their system settings. Tile is alone in its continued use of static MAC addresses, a choice which renders much of Tile's already limited defenses against passive RF adversaries moot.

Next, we consider antistalking and accountability properties. For the former, we evaluate whether providers' anti-stalking algorithms successfully detect rogue tags as well as whether they are always enabled by default and alert users automatically. All service providers except Tile have implemented antistalking algorithms that guarantee tag detectability at the operating system level, ensuring that these scans always run in the background and alert the user automatically. Samsung limits this protection to its own devices, while Apple, Google, and DULT extend coverage to all iOS and Android smartphones. Tile's reliance on manual, user-initiated scans creates dangerous detection gaps. Naturally, since Tile operates only at the application level, it lacks OS-level privileges and cannot support background scanning unless integrated into Apple's or Google's protocols.

Tile is the only deployed OF network that attempts to provide accountability. However, as demonstrated in §6, its implementation is naive and introduces framing vulnerabilities absent in other systems. Apple and the DULT specification allow a non-owner with physical access to a tag to query partial account information of the owner. This feature is optional; there is nothing cryptographically tying this information to advertisements, and it does not appear to have much use as an accountability mechanism.

The need for increased transparency. All major OF systems, including those by Apple, Samsung, and Tile, required reverse engineering for initial security analyses. These analyses exposed various vulnerabilities and limitations across the systems. Increased transparency will enable security researchers

Table 2: Privacy properties of different OF networks.

Privacy Property	Tile	Apple	Google	Samsung	RFC (DULT)
Tag indistinguishability	×	✓	✓	×	✓
Location indistinguishability	×	✓	✓	✓*	✓
Tag broadcasts randomized MAC addresses	×‡	✓	✓	✓	✓

*Samsung was reported insecure by Yu et al. [50], but has been fixed since to provide location indistinguishability in modern phones, and optionally also for its custom tags. ‡ Result for Tile Mate 2022.

Table 3: Antistalking and accountability properties of different OF networks.

Anti-Stalking Property	Tile	Apple	Google	Samsung	RFC (DULT)
Tag detectability (stalking alerts)	×	✓	✓	✓	✓
Tag detection triggered automatically by iOS/Android	×	✓	✓	✓*	✓
Attempted accountability	✓	×	×	×	×
Framing resistant	×	✓	✓	✓	✓

* This feature is only available in Samsung phones, through Samsung's proprietary SmartThings app. In Samsung phones, it is enabled by default.

to efficiently review and identify potential weaknesses *before* the system is deployed. Transparency can also foster trust between users and service providers and better allow users to understand the inherent risks of the systems they depend on.

Security guarantees provided by service providers should be well-defined. Our work demonstrates that many of Tile's security claims were (a) incorrect (user anonymity, location privacy), (b) insinuated, but substantively wrong (detection of rogue tags using the Scan and Secure feature is valid/unspoofable), (c) correct, but vulnerable to an active attacker (accountability for the abuse of the Anti-Theft mode). These findings highlight a broader imperative: service providers should offer clear and accessible definitions for the security properties they attempt to guarantee, alongside a well-defined threat model outlining the conditions under which these properties hold. Without this context, users have little chance of making informed choices about their own security and privacy needs.

Cryptographic accountability in OF networks. The design of Tile's accountability mechanism is flawed, subvertible, and allows for framing attacks, but these design issues are informative. Tile's protocol lacks clarity on what kind of evidence can be produced to implicate a user, who can access this evidence, and under what circumstances this evidence holds validity. The challenge of incorporating accountability mechanisms into OF networks remains a worthwhile research direction. Future work that establishes strong and practical definitions of accountability can enable victims to work with service providers or trusted third parties to identify misuse, prove harm, and, ultimately, disincentivize stalking and abuse.

Fundamental tensions in OF security goals. The security notions for offline finding networks appear to balance three seemingly conflicting goals: (1) tag indistinguishability, which prevents an attacker from linking Bluetooth advertisements to a particular tag; (2) tag detectability, which requires that a victim that has seen a sufficient number of advertisements from the same tag should be able to link them to form an alert; and (3) accountability, which requires that if a user detects a malicious tag, then they should be able to collaborate with appropriate authorities to punish the (correct) tag's owner. Current implementations prioritize different subsets of these properties at the expense of others. Further systematization, formalization of tradeoffs, and development of cryptographic systems that attempt to satisfy all three requirements remain open research problems worthy of future work.

8 Conclusion

We reverse engineer Tile's OF protocol and provide an in-depth security and privacy analysis. Tile's unique Anti-Theft mode intentionally weakens anti-stalking safeguards, and includes an accountability mechanism for penalizing abusers of the feature. We identify several attacks that compromise the privacy of Tile's users and find that the Scan and Secure anti-stalking feature is inadequate for protecting individuals from being tracked by Tile devices. Additionally, we find that the Anti-Theft mode is easily circumventable, and that a number of adversaries can subvert the accountability mechanism to frame honest users. Ultimately, argue that introducing to accountability OF networks is an open problem, advocate for future research in implementing useful solutions that balance privacy and accountability.

9 Ethical Considerations

Process. To ensure that our actions did not impact Tile's servers, we sent no unnatural traffic and only performed attacks against our own devices and infrastructure. Any communication with Tile's servers were performed following their protocol for normal use.

Disclosure. We disclosed our findings to Tile on November 13, 2024, by contacting CEO Chris Hulls (chris@life360.com) and the support team (support@life360.com), as a direct vulnerability disclosure channel was not available. We committed to adhere to the industry standard of 90 days (through February 11, 2024) before publicly disclosing our findings and offered remediation assistance. Tile acknowledged the vulnerabilities and engaged in dialogue until February 4, 2025, after which communications ceased. Following an additional notice on February 5 declaring our intent to publish, we extended the embargo period by an additional 60 days (until April 7, 2025) to give them sufficient time to address issues.

Mitigations. As part of our disclosure, we offered to provide Tile with recommendations for mitigating the vulnerabilities we identified. Some mitigations are feasible via firmware updates: for location indistinguishability, we suggest adopting end-to-end encryption of location data, as implemented by Samsung in response to a related issue; for tag indistinguishability, we recommend replacing static with randomized MAC addresses, deriving privateIds in real time using an incrementing counter to avoid long-term reuse, and updating the authKey upon tag transfer or sharing to reduce linkability. Other mitigations require more substantial architectural changes: since Tile's application lacks the OS-level privileges needed for continuous scanning (unlike Apple's and Google's implementations), rebuilding the application around OS-native frameworks is required to improve tag detectability. Finally, with the above mitigations in place, Tile could, in principle, strengthen framing resistance by verifying whether a privateId could have been generated by any valid authKey for the corresponding timestamp, although this approach would be computationally inefficient in practice.

References

[1] Hosam Alamleh, Michael Gogarty, David Ruddell, and Ali Abdullah S. AlQahtani. Securing the invisible thread: A comprehensive analysis of ble tracker security in apple airtags and samsung smarttags, 2024.

[2] Amazon Staff. Echo, tile, and level devices join amazon sidewalk. Amazon news blog, May 2021.

[3] Lindsey Bever. She tracked her boyfriend using an airtag — then killed him, police say, Jun 2022.

[4] Bluetooth Special Interest Group (SIG). Generic attribute profile (gatt). Bluetooth Core Specification v5.4, Part F, 2022.

[5] Fabian Bräunlein. Send my: Arbitrary data transmission via apple's find my network. https://positive.security/blog/send-my, May 2021.

[6] Businesswire. Amazon invites developers to test sidewalk and build the next billion connected devices. Businesswire, March 2023.

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

[8] Guillaume Celosia and Mathieu Cunche. Saving private addresses: an analysis of privacy issues in the bluetooth-low-energy advertising mechanism. In *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '19, page 444–453, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Hartley Charlton. Apple's airtag item trackers increasingly linked to criminal activity, Dec 2021.

[10] Samantha Cole. Police records show women are being stalked with apple airtags across the country, Apr 2022.

[11] Federal Trade Commission. FTC's Twitter Case: Enhancing Security Without Compromising Privacy, 2022. Accessed: 2024-10-21.

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

[13] Harry Eldridge, Gabrielle Beck, Matthew Green, Nadia Heninger, and Abhishek Jain. Abuse-Resistant location tracking: Balancing privacy and safety in the offline finding ecosystem. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5431–5448, Philadelphia, PA, August 2024. USENIX Association.

[14] Chinmay Garg, Aravind Machiry, Andrea Continella, Christopher Kruegel, and Giovanni Vigna. 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, page 311–322, New York, NY, USA, 2021. Association for Computing Machinery.

[15] Valentin Gazeau and Qingzhong Liu. Catch me if you can: Analyzing geolocation artifacts left by the tile application on iphones. *Acta Scientific Computer Sciences*, 2(10):38–43, 2020.

[16] Thomas Germain. Alleged stalking victims accuse tile of advertising its devices as women trackers, Aug 2024. Accessed: 2024-10-21.

[17] Github. jadx - dex to java decompiler. https://github.com/skylot/jadx.

[18] Alexander Heinrich, Niklas Bittner, and Matthias Hollick. Airguard - protecting android users from stalking attacks by apple find my devices. In *Proceedings of the 15th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '22, page 26–38, New York, NY, USA, 2022. Association for Computing Machinery.

[19] Alexander Heinrich, Milan Stute, and Matthias Hollick. Openhaystack: a framework for tracking personal bluetooth devices via apple's massive find my network. In *Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WiSec '21, page 374–376, New York, NY, USA, 2021. Association for Computing Machinery.

[20] Alexander Heinrich, Milan Stute, Tim Kornhuber, and Matthias Hollick. Who can find my devices? security and privacy of apple's crowd-sourced bluetooth location tracking system. *CoRR*, abs/2103.02282, 2021.

[21] Alexander Heinrich, Leon Würsching, and Matthias Hollick. Please unstalk me: Understanding stalking with bluetooth trackers and democratizing anti-stalking protection. *Proceedings on Privacy Enhancing Technologies*, 2024:353–371, 2024.

[22] Chris Hulls. Tile is taking a different approach to the bluetooth industry's theft and stalking problems, 2024. Accessed: 2024-10-21.

[23] Tile Inc. How does tile anti-theft mode work?, 2025. Accessed: 21-Oct-2024.

[24] Tile Inc. Tile anti-theft mode, 2025. Accessed: 21-Oct-2024.

[25] Tile Inc. Tile anti-theft mode. https://www.tile.com/en-ca/blog/tile-anti-theft-mode, 2025.

[26] Tile Inc. Tile network, 2025. Accessed: 21-Oct-2024.

[27] Tile Inc. Tile scan and secure faq, 2025. Accessed: 21-Oct-2024.

[28] Tile Inc. Tile scan and secure overview, 2025. Accessed: 21-Oct-2024.

[29] Tile Inc. Tile security & privacy policy. https://support.thetileapp.com/hc/en-us/articles/201259973-Tile-Security-Privacy-Policy, 2025. Accessed: 21-Oct-2024.

[30] Brent Ledvina, Zachary Eddinger, Ben Detwiler, and Siddika Parlak Polatkan. Detecting Unwanted Location Trackers. Internet-Draft draft-detecting-unwanted-location-trackers-01, Internet Engineering Task Force, December 2023. Work in Progress.

[31] Life360. Life360 acquires Tile trackers, Nov 2021. Accessed: 20-Oct-2024.

[32] Life360 Inc. Life360: Location sharing, safety & family tracking app. https://www.life360.com/, 2025.

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

[34] Ryan Mac and Kashmir Hill. Are apple airtags being used to track people and steal cars?, Dec 2021.

[35] Travis Mayberry, Erik-Oliver Blass, and Ellis Fenske. Blind my - an improved cryptographic protocol to prevent stalking in apple's find my network. *Proc. Priv. Enhancing Technol.*, 2023:85–97, 2023.

[36] Travis Mayberry, Ellis Fenske, Dane Brown, Jeremy Martin, Christine Fossaceca, Erik C. Rye, Sam Teplov, and Lucas Foppe. Who tracks the trackers? circumventing apple's anti-tracking alerts in the find my network. In *Proceedings of the 20th Workshop on Workshop on Privacy in the Electronic Society*, WPES '21, page 181–186, New York, NY, USA, 2021. Association for Computing Machinery.

[37] Katharina OE Müller, Louis Bienz, Bruno Rodrigues, Chao Feng, and Burkhard Stiller. Homescout: Anti-stalking mobile app for bluetooth low energy devices. In *2023 IEEE 48th Conference on Local Computer Networks (LCN)*, pages 1–9. IEEE, 2023.

[38] Lauren R. Pace, LaSean A. Salmon, Christopher J. Bowen, Ibrahim Baggili, and Golden G. Richard. Every step you take, i'll be tracking you: Forensic analysis of the tile tracker application. *Forensic Science International: Digital Investigation*, 45:301559, 2023.

[39] Sarah Perez. Tile secures another $40 million to take on Apple AirTag with new products, Sep 2021. Accessed: 20-Oct-2024.

[40] Apostolos Pyrgelis, Carmela Troncoso, and Emiliano De Cristofaro. What does the crowd say about you? evaluating aggregation-based location privacy. *Proceedings on Privacy Enhancing Technologies*, 2017:156 – 176, 2017.

[41] CBS Chicago Team. Man killed girlfriend after she removed airtag he'd secretly placed in her car, prosecutors say, Jul 2023.

[42] FOX 7 Austin Digital Team. Texas man uses apple airtag to track down stolen truck, then shoots, kills suspect: police, Apr 2023.

[43] Tile Inc. How tile works. Accessed: 21-Oct-2024.

[44] Kieron Ivy Turk, Alice Hutchings, and Alastair R. Beresford. Can't keep them away: The failures of anti-stalking protocols in personal item tracking devices. In *Security Protocols Workshop*, 2023.

[45] Christopher Vance. Android locating location data: Tile app, 2019. Accessed: 2024-10-21.

[46] Christopher Vance. ios tile app, part 2: Custom artifact, 2020. Accessed: 2024-10-21.

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

[48] Topjohn Wu. Magisk: The magic mask for android. GitHub repository and documentation, 2023. Accessed: 2024-03-15.

[49] Fengli Xu, Zhen Tu, Yong Li, Pengyu Zhang, Xiaoming Fu, and Depeng Jin. Trajectory recovery from ash: User privacy is not preserved in aggregated mobility data. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, page 1241–1250, Republic and Canton of Geneva, CHE, 2017. International World Wide Web Conferences Steering Committee.

[50] Tingfeng Yu, James Henderson, Alwen Tiu, and Thomas Haines. Security and privacy analysis of samsung's Crowd-Sourced bluetooth location tracking system. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5449–5466, Philadelphia, PA, August 2024. USENIX Association.

A HTTP request/response bodies

Body of the HTTPS request sent by the owner device to the server during the secret key establishment process of the tag activation phase §5.2.

```
{
    "tile_uuid": tileId,
    "name": "Mate",
    "rand_a": rand_a,
    "rand_t": rand_t,
    "sres_t": sres_t,
    "hw_version": 24.00,
    "model": TILE 24.00,
    "firmware_version": 48.04.16.0
}
```

Here, the corresponding values for `tile_uuid`, `name`, `hw_version`, `model`, and `firmware_version` are obtained during TDI and `rand_a`, `rand_t`, and `sres_t`, are obtained during tag authentication.

Location reports submitted by an owner for connected tags.

```
{"updates": [{
    "record_id": x,
    "location": {
        "altitude": x,
        "latitude": x,
        "longitude": x,
        "timestamp": x,
        ...},
    "tiles": [{
        "connected_auth_data": {
            "rand_a": rand_a,
            "rand_t": rand_t,
            "sres_t": sres_t,
            "tile_uuid": <8-byte UUID>},
        "discovery_timestamp": x,
        "record_id": x
        },
        {
        "client_data": {"tile_uuid": x},
        "discovery_timestamp": x,
        "record_id": x
        }]
    },
    ...
]}
```

Here, the corresponding values for `rand_a`, `rand_t`, and `sres_t` are obtained during tag authentication, and the location values include the owner's current location.

Location reports submitted by a finder for lost tags.

```
{"updates": [{
    "record_id": x,
    "location": {
        "altitude": x,
        "latitude": x,
        "longitude": x,
        "timestamp": x,
        ...},
    "tiles": [{
        "advertised_service_data": {
        "mac_address": MAC address,
        "payload_service_data": privateId,
        ...},
        "discovery_timestamp": x,
        "record_id": x
        },
        {
        "client_data": {"tile_uuid": x},
        "discovery_timestamp": x,
        "record_id": x
        }]
    },
    ...
]}
```

Here `payload_service_data` includes the privateId broadcast by the lost tag, and `mac_address` is the tag's static MAC address.

Request sent by a Scan and Secure client to the server after completing the scan.

```
[
    {"privateIds": [x1, x2, ...]},
    {"privateIds": [x1, x2, ...]},
    {"privateIds": [x1, x2, ...]},
    {"privateIds": [x1, x2, ...]},
    {"privateIds": [x1, x2, ...]},
    {"privateIds": [x1, x2, ...]}
]
```

Here, each `xi` is a privateId value seen during the corresponding scan.

The server's response for the Unlimited Sharing feature.

```
"result":{
    "tileType":"TILE",
    "tile_uuid": x,
    "user_uuid": x,
    "other_user_uuid":x,
    "other_user_email":x,
    ...
    }
```

Here `other_user_uuid` and `other_user_email` correspond to the shared owner's details.

The server's response for the Community Information feature.

```
{...
 "timestamp_ms": x,
 "result_code": 0,
 "result": {
   "timestamp": x,
   "center_latitude": lattitude,
   "center_longitude": longitude,
   "center_radius": 5.0,
   "tilers_around": numberOfTileUsers,
   "display_tilers_around": true,
   "display_tiles_found": false,
   ...
 }
}
```

Here the latitude and the longitude are the client's current location coordinates, the center radius is always set to 5 miles, and the `tilers_around` value represents the number of Tile users in the 5-mile radius around the client.

B Details of Tile's protocol

B.1 Tile Device Information

The owner requests the tileId, model, firmware, and hardware_version from the tag over the custom Tile Device Information (TDI) service (UUID 180A). The corresponding characteristic UUIDs for these variables are defined in Table 4.

Characteristic UUID	Value
9d410007-35d6-f4dd-ba60-e7bd8dc491c0	Tile Id
00002a24-0000-1000-8000-00805f9b34fb	Model number
00002a26-0000-1000-8000-00805f9b34fb	Firmware
00002a27-0000-1000-8000-00805f9b34fb	Hardware version

Table 4: Characteristic UUIDs and corresponding values shared between the tag and the user under the devInfoService (UUID 180A) during activation.

The tileId is an 8-byte static identifier of the tag that is derived from its static MAC address, the model is a 10-character alphanumeric identifier for the tracker of the form "xxxx yy.yy" where the first 4 characters constitute the vendor identifier assigned to a vendor by Tile and the last 5 characters stand for the model number, the value firmware is a 10-character numeric value of the form "xx.xx.xx.x" and represents the firmware version, and hardware_version is a 5-character numeric value of the form "xx.xx" and represents the hardware version.

Characteristic Name	Characteristic UUID
MEP_TOA_CMD	9d410018-35d6-f4dd-ba60-e7bd8dc491c0
MEP_TOA_RSP	9d410019-35d6-f4dd-ba60-e7bd8dc491c0

Table 5: UUIDs for MEP_TOA_CMD and MEP_TOA_RSP characteristics used MEP TOA communications.

B.2 Details of tag-owner interactions

The Tile Over-the-Air protocol. Once a Tile tracker has been activated, communications with it use the Multi-Endpoint Tile Over-the-Air (MEP TOA) protocol. MEP TOA is executed using two characteristics, MEP_TOA_CMD and MEP_TOA_RSP, defined under the 'FEED' service. The corresponding character UUIDs are given in Table 5. The former is used by the owner device to send messages to the tracker and the latter is used by the tracker to send messages to the owner device.

Communications using MEP TOA can be connectionless or connected. Connectionless communications are identified by a random 4-byte sequence called the toaToken. Near-mode interactions between the owner and the tracker are over a connected channel. We now describe the steps involved in establishing a connected channel.

Establishing a connected channel: Owner authentication. The second part is owner authentication. We summarize the steps involved in owner authentication in Figure 6. This step begins once the Tile server sends a TOA_OPEN_CHANNEL message to the tracker containing the channelPrefix and channelData to be used for communications over the connected channel. The owner first computes a key that we call the tagKey using the authKey as follows. Here randA is the value used in the tag authentication step and toaToken is the identifier for the current connectionless channel.

$$\text{seed} \leftarrow \text{randA} \,||\, \text{channelData} \,||\, \text{channelPrefix} \,||\, \text{toaToken}$$
$$\text{tagKey} \leftarrow \text{HMAC-SHA256}(\text{authKey}, \text{seed})[0:128]$$

Then the owner authenticates itself to the tracker tag by sending a fixed message $\text{msg} = [0\text{x}12, 0\text{x}13]$ along with a MAC tag over the message generated using tagKey as follows.

$$\text{seed} \leftarrow \text{ctrA} \,||\, 1 \,||\, \text{msgLen} \,||\, \text{msg}$$
$$\text{tag} \leftarrow \text{HMAC-SHA256}(\text{tagKey}, \text{seed})[0:32]$$

Here msgLen is the length of the message and ctrA is a counter maintained by the owner which is initialized to 0 and incremented each time a message is sent to the tracker. The tracker verifies the tag against the message by locally deriving tagKey from authKey.

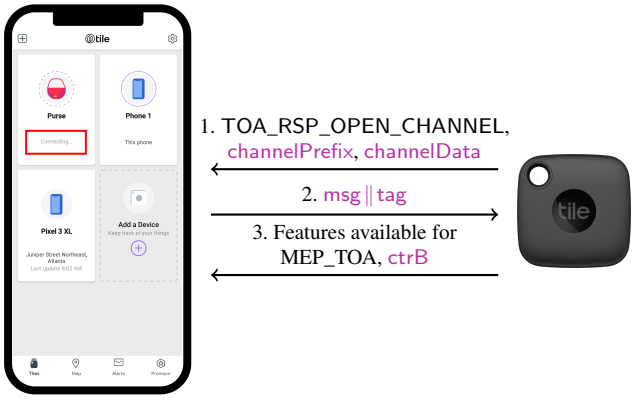

Figure 6: The authentication protocol used by the owner device to authenticate itself to the Tile tracker. The variables channelPrefix, channelData, msg, tag, and ctrB are defined under owner authentication.

If the verification is successful, the tag responds with the features available for MEP_TOA and the counter ctrB. The Tile Mate 2022 offers features such as ringing the Tile, (reverse) ringing the owner's phone. After this step, the owner and the tracker have established a connected channel. Each message the owner sends to the tag is authenticated with a MAC tag over the message generated using tagKey and ctrA.

Ringing the tracker. An owner can ring their Tile device by tapping on the corresponding tag on the app and clicking the "Find" button on the tag's screen. This will use the song characteristic (9d410002-35d6-f4dd-ba60-e7bd8dc491c0) to play a song on the selected Tile tag.

Reverse ringing the phone. The Find My Phone feature allows a Tile to ring the owner's phone using the reverse ring characteristic (9d410000-35d6-f4dd-ba60-e7bd8dc491c0) by double-pressing the button on the tag. The owner can tap the "Found it" button on the notification to stop the ring.