Who Tracks the Trackers? Circumventing Apple's Anti-Tracking Alerts in the Find My Network

Travis Mayberry
Ellis Fenske
Dane Brown
mayberry@usna.edu
fenske@usna.edu
dabrown@usna.edu
US Naval Academy
United States

Jeremy Martin
Christine Fossaceca
jbmartin@mitre.org
cfossaceca@mitre.org
MITRE
United States

Erik C. Rye
Sam Teplov
Lucas Foppe
rye@cmand.org
CMAND
United States

ABSTRACT

Apple's Find My protocol allows lost devices, such as AirTags, to relay their location to their owners via a network of over a billion active Apple devices. This convenient feature for device owners may also be a tool for malicious actors to cheaply and effectively track unknowing targets. Apple has introduced a featured known as "item safety alerts" to prevent AirTags from being used this way. We demonstrate that it is possible to create a custom device, with similar features to an AirTag in terms of cost, size, and battery life, which can participate in and be tracked by Apple's Find My network while not triggering any item safety alerts. This implies that Apple's protection mechanism is insufficient. We suggest natural mitigations for two of our malicious tracker techniques but note that the third would require substantially altering the Find My protocol to defend against.

CCS CONCEPTS

• **Hardware → Wireless devices**; • **Networks → Network privacy and anonymity**.

KEYWORDS

tracking,privacy,apple,ble,airtags

ACM Reference Format:
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 Privacy in the Electronic Society (WPES '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3463676.3485616

1 INTRODUCTION

In 2019, Apple introduced a feature for their devices termed "Offline Finding (OF)". The goal of this feature is to allow users to locate their lost devices even when they are in a location where they do not have internet access (outside of cellular reception, no nearby trusted Wi-Fi networks, etc.). Before this feature was added, the only way to locate a lost device was to have it "phone home" to Apple over the internet, after which the user can login with their iCloud account and check its location, have it play a sound, mark it as "lost", or remotely erase the device.

With OF, when a device loses internet connectivity it starts to send advertising messages over Bluetooth Low Energy (BLE) using Apple's Continuity protocol, which is described in Martin et al. [14]. Nearby Apple devices, (i.e, bystanders), will receive these BLE messages and log the BLE advertisement data along with the location (latitude/longitude coordinates) where the lost device was observed. When a *bystander* device gains a connection to the internet, it will forward the token and corresponding location coordinates to Apple for the owner of the lost device to later retrieve and locate their device.

This feature is very similar to what is already done by other manufacturers of token tracking technology such as Tile[1], SmartTag[3], and Chipolo[2]. In 2021 Apple released their own version of these tokens, which they named "AirTags", that participate in the same OF protocol described above. At this time, they renamed the technology from "Offline Finding" to "Find My," to reflect that the network would contain many types of devices and not just offline iPhones. In contrast with Tile, the largest BLE tracking competitor with around 35 million devices in its finding network [10], Apple has over 1 billion iPhones currently in active use [12]. This makes it much more likely that a bystander will have a compatible device to report the location of a lost AirTag, compared to a lost Tile.

In designing AirTags, Apple attempted to mitigate potential abuses of their technology. Since AirTags are very small, cheap, and can be located without requiring any internet access, they would be easy to surreptitiously place in someone's backpack, purse, car, etc. This could allow, for instance, a stalker, oppressive regime, or criminals to track the location of a target without their knowledge and without expending many resources. To prevent this, Apple has included in iOS another feature called "item safety alert" where an iPhone can identify that the same AirTag is following it from location to location and thereby alert the user that they are being tracked. The victim is then assisted in finding the tracking device and disabling it. The iOS device can connect to the Airtag over Bluetooth and cause it to beep loudly, and an iPhone 11 or newer

WPES '21, November 15, 2021, Virtual Event, Republic of Korea.

ACM ISBN 978-1-4503-8527-5/21/11…$15.00
https://doi.org/10.1145/3463676.3485616

can use ultra wideband to point the user directly toward the Airtag like a compass.

Popular media reports that this anti-tracking feature can be unreliable [7, 13]. Apple's anti-tracking technology is only available on iPhone devices (i.e., an Android user will not be able to detect tracking), and in addition, alerts are not always shown to the user in a reasonable amount of time. Tracking can go on for hours or sometimes days without triggering an alert [7]. In this paper, we present the first analysis of the item safety alert feature, experimentally determining under what criteria these alerts are shown to the user. We determine that alerts require an iPhone to see the same AirTag several times, with the user being in multiple discrete locations separated by a minimum distance.

We go on to show that it is cheap and easy to design a custom device which participates in the OF network like an AirTag, and can be located through Apple's servers, but which will never trigger an item safety alert and cannot be identified as a tracker by nearby iOS devices. We show that this can be accomplished with three different techniques. Two of these attacks can be discovered and mitigated with potential software fixes to the item safety alert feature. Toward that end, we have disclosed our findings to Apple in June of 2021. However, we believe the third technique to be robust and difficult to detect, even if the attack is known to be in use.

2 FIND MY PROTOCOL

Specifications for the Find My protocol were briefly released by Apple in October 2020, but were quickly moved behind a portal requiring viewers to enroll with Apple as a partnering device manufacturer [4]. There are currently no publicly available specifications from Apple. Fortunately, Heinrich et al. [11] have also reverse engineered the main portions of the protocol and [8, 14] have implemented a Wireshark dissector to process Apple's BLE Continuity protocol frames. We update the dissector, providing additional Find My protocol dissection support and contribute our work back to the community. We reproduce the relevant details of Heinrich et al. [11] for our work in this section. We focus on the protocol as it applies to AirTags, since we are primarily investigating their behavior and the behavior of other Apple devices around them, but most of the details are the same for any lost device participating in the Find My protocol.

We first introduce some terminology. We refer to the messages sent by an AirTag in a lost state as **lost messages**. A third-party Apple device participating in the Find My network is a **bystander**. The reports that a bystander sends to Apple after receiving a lost message from an AirTag, which include the latitude and longitude of the lost device, are **location reports**.

2.1 Lost Messages

When an AirTag is first paired with an iCloud account, the AirTag and another device previously enrolled with the iCloud account (owner's iPhone or iPad for instance) jointly create a Elliptic Curve Diffie-Hellman (ECDH) public key on the curve P-224 P and two 256-bit symmetric keys SKS and SKN. These keys are used to generate rotating temporary public keys that the AirTag broadcasts in its BLE messages while it is lost. The keys are known to the AirTag as well as the iCloud user and are stored in their iCloud keychain.

Figure 1: Breakdown of the byte structure of a lost message in the Separated state.

Byte	Value	Comment
MAC addr	Public key	Bytes 0-5 of PW_j
0	`0x12`	Payload Type
1	`0x19`	Length
2		Battery info, see discussion
3-24	Public key	Bytes 6-27 of PW_j
25	Key overflow	Bits 0-1 of byte 0 of PW_j
26	Byte 5 of P_i	Unknown purpose

Crucially, this key is *not known* to Apple and is cryptographically sealed when stored in the iCloud keychain.

At any point in time, an AirTag can be in three possible states: *Connected*, *Nearby* or *Separated*. An AirTag starts out in a Connected state when it is initially paired. It remains in this state as long as the device it was paired with is nearby and it can maintain a BLE connection. If the AirTag ever loses the BLE connection with the paired device, it transitions to the Nearby state.

In the Connected or Nearby state, the AirTag broadcasts a short advertising message with a public key derived from SKN (Secret Key Nearby) and the Master Public Key P. This key, and the BLE MAC address used in the message, rotates every 15 minutes to prevent adversarial tracking via a static identifier. In these states, the BLE advertisement contains only a partial public key and nearby bystanders will ignore these messages. In these states, the AirTag is considered "not lost," i.e., it is either directly connected via BLE to its owner's device or it has recently been in the Connected state.

After 15 minutes in the Nearby state, if the AirTag does not transition back to the Connected state it goes into the Separated state. The AirTag now considers itself "lost". This is the most interesting state to us because as the AirTag now becomes active in the Find My network and other devices will start to report the lost AirTag's location upon observation of the "lost" BLE advertisement frames.

From this point, the AirTag will broadcast the full BLE advertising message that we refer to as a lost message. The structure of these messages can be seen in Figure 1. In this state, it advertises an ECDH public key PW_j that is derived from SKS (Secret Key Separated) and the Master Public Key P. This process is deterministic, incrementing the key once for each rotation period.

Due to the size constraints of a BLE advertising message being extremely limited, the structure of a lost message is very compact. In order to send the entire 28 byte public key, the first 6 bytes of PW_j are encoded within the MAC address of the AirTag. Since this address is randomized and local, it must have the least significant bits of byte 0 of the MAC address set to `0b11`. Because of this, the corresponding two bits of PW_j are sent instead in byte 25 of the advertisement frame.

Byte 2 contains the battery status of the AirTag, encoded in bits 6 and 7. It can have values $0-3$ for Full, Medium, Low and Very Low, respectively. Importantly, this byte is also required by the spec to have bit 5 set to 1. We will revisit this fact later.

Byte 26 is described in the specification as a "hint" byte. It is set to byte 5 of the key P_i (the Nearby State Key). The purpose of this byte is not given. In our experiments we have determined

that setting it to arbitrary values, or always 0, has no effect on the protocol.

Because none of the information in this message is identifying (the public key is random), a bystander does not learn anything about who owns the AirTag. Since the key rotates every 15 minutes in Nearby state and every 24 hours in Separated state, it cannot be tracked long-term by any identifiers in the message. This is also why we can create our own devices that act as AirTags: since the message only contains a random public key, and no authentication information, it is trivial to forge these messages and have them match the distribution of messages from a normal AirTag.

2.2 Location Reports

When a bystander device receives a lost message, it reconstructs the ECDH public key, encrypts its own current location with that key and sends it to Apple's servers along with a hash of the public key. We refer to this as the location report. An owner attempting to locate the AirTag will then calculate the public keys the AirTag has used over the past 7 days (the owner can do this because they have P and SKS in their iCloud keychain), hashes them and sends them to the Apple server to see if any location reports were submitted matching those hash values. Note that there is no throttling or limiting on the number of keys that a client can request from Apple's servers; it is possible to request hundreds or thousands of them in a few seconds.

If a report is retrieved, the owner decrypts the report with their corresponding private key, generated from SKS. Because SKS is known only to the owner of the device, only the owner can correctly decrypt these lost reports and obtain the location of the AirTag. Even Apple cannot learn an AirTag's location because SKS is never sent to them and is stored in the iCloud keychain sealed by a key that is stored in the Secure Enclave [5] of the owner's device.

3 ITEM SAFETY ALERTS

Anticipating the potential for abuse of these low-cost AirTags as malicious unwanted tracking devices, Apple has implemented anti-tracking technology in iPhones which displays a warning upon receiving consistent lost messages from the same AirTag over time.

We observed through experimentation that these tracking alerts appear only when the AirTag is co-located with the device over a period of time and also across a minimum distance (about a mile). Upon release, these alerts were much less reliable, requiring several hours to display (seen in our experiments and also reported in news media [7]), but an update was deployed in June for iOS 15 beta that lowered the amount of time to about 30 minutes [15]. An example of an alert can be seen in Appendix A.

We attempted to determine more precisely what the criteria were for a device to trigger an alert. We were able to observe iPhones recording lost messages through the debug console log, which shows that an AirTag beaconing over an long period of time will eventually be flagged as "suspicious." We have yet to determine the exact mechanisms by which these messages were flagged as such. We leave it to future work to determine this, potentially via reverse engineering.

4 DEFEATING ALERTS

We identified, implemented, and tested three primary methods to prevent item safety alerts. All were implemented on an Espruino Puck.js [6], a small, low-cost, general-purpose computing platform able to send BLE messages. The Puck is approximately the same size as an AirTag and actually uses the same Bluetooth System on a Chip (SoC) (nRF52832). We release each method as a fully functional open-source implementation (link redacted), including a Wireshark dissector (link redacted) to parse and verify Find My BLE lost messages.

Rather than derive a key from information stored by the iCloud account as described in the genuine Find My protocol, we generate a random P-224 keypair and advertise it from our device as outlined in Section 2.1. Since Apple's Find My app only allows users to retrieve location reports for legitimate Apple devices that have undergone the pairing process, we instead use the OpenHaystack client [16], an open-source project that allows querying Apple's Find My servers for arbitrary public keys.

However, OpenHaystack provides an additional advantage in that it displays multiple location report positions over time. By contrast, Apple's Find My application depicts only the location of the most recent location report. Appendix A contains several maps generated by tracking our Espurino Puck devices. All tracking experiments were performed by the authors of this paper using our devices to track ourselves. No non-consensual tracking was performed.

Tests were run on various devices, including iPhones 7, 8 and 12 and iPad Pro (4th Gen). Behavior was tested and confirmed to be identical on iOS 14.5 (the first version to support Airtags), 14.5.1 and 14.6.

4.1 Approach 1: Bit Flipping

As we observed in Section 2.1, bits 5, 6, and 7 of byte 2 in a lost message contain battery information. Crucially, while AirTags set bit 5 to 1, iPhones can also report themselves as lost, in which case they generate lost messages as well, but with byte 2 set to `0x00`. iPhones do not consider lost iPhones to be potential tracking devices, presumably because they are expensive with a relatively short battery life. This means we can simply configure our devices to send lost messages using the iPhone format (with bit 5 set to 0) rather than the AirTag format, in which case they do not produce item safety alerts.

Mitigation: We reported this vulnerability to Apple and expect mitigation against this technique to be trivial and very effective; item safety alerts can simply be generated even if a lost iPhone (or adversary device posing as an iPhone) is used as a tracking device.

4.2 Approach 2: Rotating Keys

While separated-state AirTags use the same lost message over a long period of time, we face no such limitation with our custom advertising devices. Instead, we found that we could simply rotate the advertised public key regularly over time, and then query all keys on our fixed rotation schedule to recover all relevant location reports for our devices, presenting an effective tracking technique. We observed that simply rotating keys can still trigger item safety alerts if the number of keys is small, but if many keys are used

Keys (n)	Rotation time (s)	Repeat time (r)	Alert?
5	900	4500	Yes
15	900	13500	Yes
25	20	500	Yes
50	900	45000	No
100	20	2000	No
100	900	90000	No

Table 1: Key Rotation Experiments

the devices can be used for tracking purposes indefinitely without triggering alerts. We performed a series of experiments with our devices, each time carrying the device for many hours across long distances (for experiments with many keys, we carried these devices for weeks across thousands of miles as depicted in Appendix A. Each time, we programmed the device to send lost messages on a n-key rotation, sending an advertising message every two seconds and rotating keys every s seconds, so that the first key used by the device is re-used after $r = ns$ seconds, with our results presented in Table 1. From our results, we suspect the primary factor in preventing the alerts is number of keys, rather than time between repeated keys, and that a rotation cycle with length between $25 - 50$ is currently sufficient to prevent the alerts.

Mitigation: This technique only works because the item safety alerts are not recognizing repeated keys as tracking threats if they occur too far apart or with too many other lost devices seen in the meantime. The algorithm could be tuned to recognize repeated keys more aggressively. The Puck can only store a limited number of keys due to memory constraints (slightly more than 100 with our implementation) so with proper parameters the item safety alerts could certainly recognize our tracker.

4.3 Approach 3: Generating Keys on the Puck

Since memory is limited (the Puck can only store about 100 KB of data) if we want to avoid repeating keys the next logical step is to organically generate keys on the Puck. This is more difficult than it may sound, due to the limited hardware resources available. The Puck normally runs javascript code as its input, but our implementations of Elliptic Curve operations using the on-board Javascript interpreter were too slow to be practical (30 minutes to calculate one public key). For our implementation, we ultimately built a custom version of the Espruino firmware which included versions of the Elliptic Curve functions from OpenSSL. This firmware had to be hand-tuned to fit into the flash memory of the Puck.

We were then able to create a javascript program that could run on the Puck which creates ECDH public keys deterministically from an initial seed. Each key is calculated as SHA256($seed + counter$), where $counter$ starts at 0 when the Puck is initialized and increments every 15 minutes when the key rotates. This ensures that a key is never repeated, making detection of the tracker more difficult.

Mitigation: We considered a few approaches to mitigation in this case. First, devices could detect when they consistently observe lost messages near them, even if they do not use the same key. However, this presents a problem: in dense areas it is not unreasonable to expect lost devices to be present in many locations, and it would not take many false positives for these alerts to become an irritant rather than a helpful resource to users.

Moreover, the strategy of the malicious tracker could easily be adjusted to hide from this type of detection by only broadcasting lost messages for short periods and then going inactive for some time. It would reduce the granularity of location reports slightly in exchange for being much harder to detect. The Puck even has an on-board accelerometer, so a more sophisticated tracker program could stay inactive until it detects movement.

Second, the messages could include some kind of authentication so that they cannot be trivially forged. To preserve the anonymity properties of the Find My protocol, this would require an anonymous authentication protocol to allow devices to ignore lost messages which do not come from a legitimate AirTag. In our estimation, such an approach would require a significant redesign of the devices so that they contain forgery-resistant credentials, and the protocol itself, since the advertising message does not contain additional space that could be used for authentication information; the public keys are already large enough that they do not fit in the message payload itself, so must be partially contained within the advertising MAC address of the messages.

Finally, Apple may be able to limit queries for location reports. If we were not able to use OpenHaystack to query for arbitrary public keys, our trackers would not work as we have described them here. Apple could limit queries cryptographically, potentially restricting access to their servers to the first-party Find My App through use of the Secure Enclave. However, this would prevent any device without a Secure Enclave from using the service. Even this is not a complete solution because an attacker could still attempt to retrieve the master key from an Airtag (it does not have any secure hardware) and derive keys from that which would be trackable using the Find My App. Researchers have been able to connect to the debug pins of the microcontroller and retrieve the contents of the flash memory [9], making this approach likely feasible.

5 CONCLUSION

Apple's Find My network has orders of magnitudes more devices participating in it than its nearest competitors. This makes it an extremely powerful tool for cheaply and accurately locating lost devices such as AirTags. Apple has worked extensively to prevent their AirTags from being used maliciously as covert tracking devices, implementing "item safety alerts" into iOS to warn users if they are being tracked by an AirTag.

However, we have shown that Apple's threat model for anti-tracking is dangerously incomplete. In their efforts to make the protocol privacy-protecting for AirTag owners, they make it possible for third-party devices to participate in the network due to a lack of authentication of lost devices. We have demonstrated that it is possible to create a device with approximately the same size, cost, and battery-life of an AirTag which can be tracked using the Find My network while also running custom code.

We have presented three strategies that a malicious tracker can use to avoid detection by item safety alerts. Two of these strategies can be fixed with simple software updates to iOS, but the third seems to require substantial redesigning of the Find My protocol. We have released our tracker software and analytical tools to the public as open source implementations, including custom Espruino firmware and Wireshark dissector.

REFERENCES

[1] [n.d.]. Find Your Keys, Wallet & Phone with Tile's App and Bluetooth Tracker Device. https://www.thetileapp.com/

[2] [n.d.]. Find your keys, wallet or anything you don't want to lose. https://chipolo.net/en-us/

[3] [n.d.]. Use the Samsung Galaxy SmartTag and SmartTag+. https://www.samsung.com/us/support/answer/ANS00088244/

[4] 2021. Create Innovative Accessories. https://mfi.apple.com/

[5] 2021. Keychain data protection. https://support.apple.com/guide/security/keychain-data-protection-secb0694df1a/web

[6] 2021. Puck.js. https://www.espruino.com/Puck.js

[7] Geoffrey A. Fowler. 2021. Apple's AirTag trackers made it frighteningly easy to 'stalk' me in a test. *Washington Post* (May 2021). https://www.washingtonpost.com/technology/2021/05/05/apple-airtags-stalking/

[8] furiousMAC. [n.d.]. furiousMAC/continuity: Apple Continuity Protocol Reverse Engineering and Dissector. https://github.com/furiousMAC/continuity

[9] Ghidraninja. 2021. Yesss!!! After hours of trying (and bricking 2 AirTags) I managed to break into the microcontroller of the AirTag! /cc @colinoflynn @LennertWo pic.twitter.com/zGALc2S2Ph. https://mobile.twitter.com/ghidraninja/status/1391148503196438529?s=20

[10] Todd Haselton. 2021. Here's how Apple's AirTag trackers compare to Tile, and why the company is so upset with Apple. https://www.cnbc.com/2021/04/27/apple-airtags-versus-tile-tracker-how-they-compare.html

[11] 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* 3 (2021), 227–245.

[12] Jacob Kastrenakes. 2021. Apple says there are now over 1 billion active iPhones. https://www.theverge.com/2021/1/27/22253162/iphone-users-total-number-billion-apple-tim-cook-q1-2021

[13] John Koetsier. 2021. How To Track People With Apple AirTags. https://www.forbes.com/sites/johnkoetsier/2021/04/22/how-to-track-people-with-apple-airtags/?sh=3565bf6269df

[14] Jeremy Martin, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske, Lucas Foppe, Travis Mayberry, Erik C Rye, Brandon Sipes, and Sam Teplov. 2019. Handoff All Your Privacy: A Review of Apple's Bluetooth Low Energy Continuity Protocol. *arXiv preprint arXiv:1904.10600* (2019).

[15] Philip Michaels. 2021. Apple updates AirTag to fix its biggest flaws. https://www.tomsguide.com/news/apple-updates-airtag-to-fix-its-biggest-flaws

[16] Seemoo-Lab. 2021. OpenHaystack. https://github.com/seemoo-lab/openhaystack

A EXAMPLE IMAGES

All of the following example images and location histories were generated in the course of experiments performed by members of our research group, and were created and are reproduced here for publication with consent and foreknowledge of the participants.

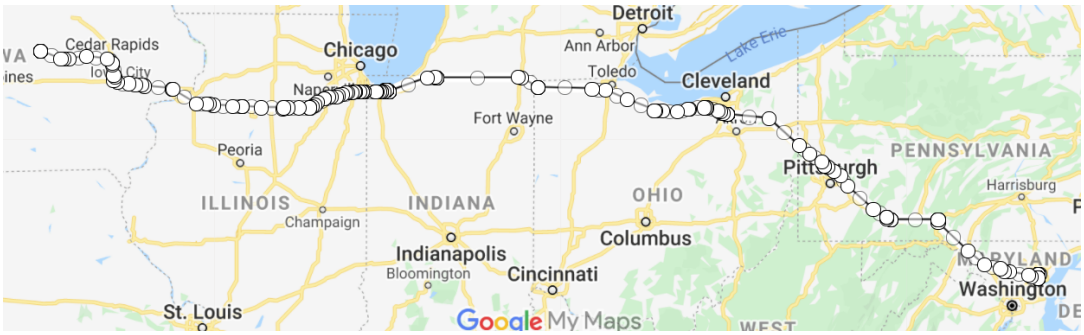

Figure 2: Traveling across the US. Note increased density of reports in urban/suburban areas from more bystanders.

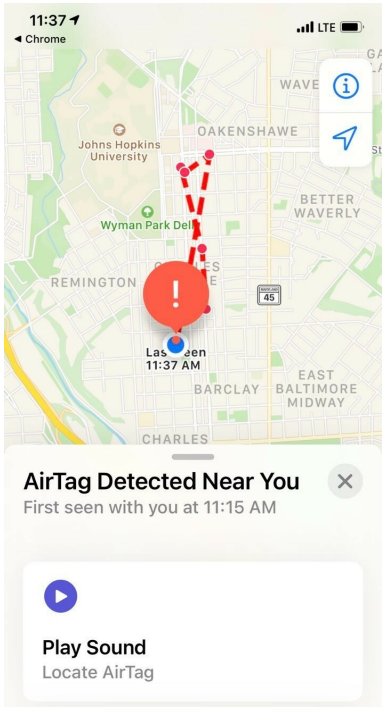

Figure 3: The item safety alert screen.

Figure 4: Tracking in a small urban area.

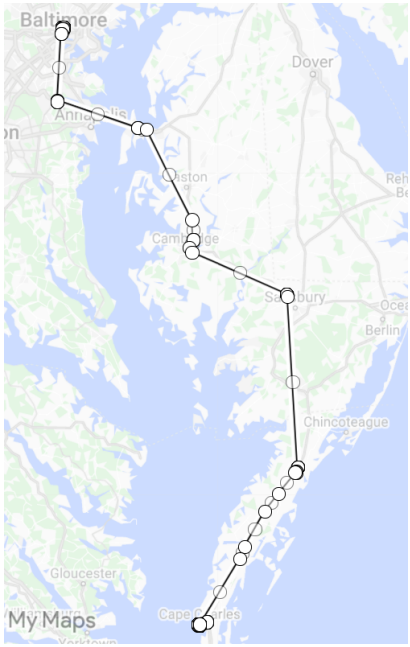

Figure 5: Tracking over a longer distance.