Encrypted IP Over Apple AirTags is "Practical"

Shreyas Minocha
Georgia Institute of Technology

George Ari Hosono
Georgia Institute of Technology

Akshaya Kumar
Georgia Institute of Technology

Michael A. Specter
Georgia Institute of Technology

Abstract

Apple's Find My is a crowd-sourced location tracking network with hundreds of millions of devices. AirTags are small battery-operated devices with no internet capabilities that transmit Bluetooth advertisements. These advertisements are picked up by nearby finder devices like iPhones and iPads that submit their own locations along with the observed advertisement data to Apple's servers. Apple's protocol is designed to ensure the anonymity of Find My users and encryption of location reports. In addition to their own location, finder devices also encrypt a "status" byte, which is included in the Bluetooth data transmitted by accessories. We develop a protocol that allows users to use this byte to transmit and receive arbitrary data, such as TCP packets, over the Find My protocol. This can also serve as a covert communication channel since the Find My protocol encrypts the byte we use to transmit data.

1 Introduction

When Apple launched its crowd-sourced location tracking protocol, Find My, in 2019 [8], it was notable because of its scale and Apple's strong claims about its cryptographic privacy guarantees. Several works have studied the security of crowd-sourced location tracking networks [5, 7, 8, 11, 13, 21]. In the field of censorship circumvention, one line of work has attempted to enable covert communication by imitating other protocols [10] or using them as covert channels [17]. Some prior work has attempted to use Apple's Find My network to transmit data from sensors and other internet-less devices [6, 2, 18]. However, to our best knowledge, this is the first work to employ Apple's Find My network as infrastructure for *two-way communication* between remote parties. Our work builds upon a body of unconventional networking protocols [19, 9], including the seminal 1990 RFC *IP Over Avian Carriers* [20]. We developed a new covert communication mechanism that, because of Find My's cryptographic properties, allows us to perform encryption at the network layer.

2 Background

2.1 Apple Find My

Heinrich et al. [8] reverse-engineered and analyzed the security of Apple's Find My protocol. In this section, we provide a brief overview of Find My, especially as it relates to our networking protocol. Apple's protocol involves four kinds of components: accessories, small, battery-powered Bluetooth devices; owners, devices such as phones that are paired with accessories; finders, iPhones and iPads that assist in finding lost accessories around them; and Apple's servers.

Accessories must start by pairing with owner devices. Through the pairing process, these devices share a "master beacon" key pair on the NIST P224 elliptic curve, which is used to generate ephemeral P224 key pairs.

When accessories enter "lost" mode by losing Bluetooth connection with their owner devices, they broadcast Bluetooth Low Energy (BLE) advertisements, which include just 31 bytes of data. Each advertisement encodes an ephemeral P224 public key, a *"status byte"* (typically used to indicate e.g. low battery), and a "hint" byte[1].

Finder devices that pick up these BLE advertisements use an Elliptic Curve Diffie–Hellman (ECDH)-based scheme to non-interactively derive a shared secret with the accessory and owner devices. They do this using a randomly sampled P224 key pair and include its public key in the unencrypted portion of the secret. Next, they encrypt their physical location, *together with the status byte from the BLE advertisement* under a key derived from the shared secret using AES-GCM. Finally, they upload the encrypted location report along with a SHA256 hash of the accessory's ephemeral public key to Apple's servers.

To retrieve location reports, the owner fetches reports for their lost accessory's ephemeral public keys by querying their SHA256 hashes. They then compute the shared secret with the finder device using its public key. Finally, they use this secret to derive the symmetric encryption key and decrypt the

[1] According to [8], the hint byte is always `0x00` in iOS reports.

location information (and status byte).

2.2 Find My Implementation Details

Several practical aspects of Apple's Find My network influence its efficacy as a data link layer protocol.

2.2.1 Uploading Reports

Find My is designed to allow only Apple devices to submit location reports to its servers. Requests to the report submission endpoint are authenticated; among other things, they include an Elliptic Curve Digital Signature Algorithm (ECDSA) signature over the request body, which is likely verified by Apple's servers [8]. The private key used for signing request bodies is stored in Apple's Secure Enclave Processor (SEP) and it cannot easily be extracted from it. Unfortunately, this means that we cannot use the location data field to transmit encrypted payload data. Future work could explore the feasibility of doing this on Apple devices where we have root access, e.g., through a "jailbreak".

Prior studies suggest that there is a median delay of 13 minutes [18] to 25 minutes [8] between the generation and uploading of location reports. This delay depends on the density, connectivity, and battery levels of surrounding finder devices [6, 18]. This suggests that *IP Over AirTag* may currently not be suited for real-time applications like video conferencing.

Finder devices upload a limited number of reports per public key per day [8]. They only make up to 96 submissions per day, each including up to 200 reports [6].

2.2.2 Fetching reports

Although only Apple devices are supposed to support fetching reports for accessories, these requests, unlike report submissions, are *not* authenticated with a signature signed by SEP-protected keys. With an Apple ID, one may use open-source software such as FindMy.py [1, 7] together with tools like anisette-v3-server [3, 16] to fetch reports. By design, anyone may fetch encrypted reports for any hashed public key, regardless of whether they have the corresponding private key.

Apple's servers support fetching reports for up to 255 hashed public keys in a single request. Although Apple's rate-limiting on report fetching has not been studied (and doing so may present ethical concerns), folklore [12] suggests that one request (with up to 255 hashed public keys) every 30 minutes is likely safe from rate limiting.

Apple's servers don't return reports older than seven days [8].

3 Design

Since the status byte in Find My Bluetooth advertisements is encrypted as-is and included in reports fetched from Apple, it can be used to transmit arbitrary payloads [6]. We use this byte, together with Find My's existing cryptographic design, to design a networking protocol—*IP Over AirTag*—that transmits data over Find My.

In our protocol, a sender broadcasts Bluetooth advertisements that encode the recipient's public key and set the status byte to one byte of the payload. These advertisements are picked up by finder devices (as well as other Bluetooth devices) in the sender's vicinity. Finders encrypt the payload and their physical location for the recipient before uploading them to Apple's servers. The recipient can then fetch reports for the public keys advertised by the sender, use the corresponding private keys to decrypt them, and extract the payload bytes. Finally, the recipient can send a response by mirroring the message transmission process.

When two parties send each other messages through this mechanism, data transmission operates at a bandwidth of one byte per unique Bluetooth advertisement. Granzow et al. [6] were able to do this at a rate of 1.1 bit/s, or about 7 seconds per byte. On the other hand, receiving payloads takes on the order of minutes because of the delay between finder devices creating reports and uploading them (see 2.2.1).

Since one request to Apple's servers can fetch reports for up to 255 keys, 255 makes for a natural *frame* size for *IP Over AirTag*. Senders can transmit data in frames of up to 255 bytes, which recipients can conveniently fetch by making one request per frame. Note that this is not a strict requirement of our protocol. However, both parties must always agree on how many bytes have been sent by both of them and whose turn it is to send. This could be achieved, for instance, by including the length of the payload in the frame (as in UDP) or by switching sender-recipient roles after each frame (as in TCP in stop-and-wait mode with piggybacking of outgoing data onto ACK packets). This limitation stems from the fact that both parties must have a consistent view of which keys are being advertised at any moment.

Suppose *IP Over AirTag* is used to send and receive TCP packets. TCP headers are between 20 and 60 bytes long, which is well within the frame size limit. UDP headers are even smaller at just 8 bytes per datagram. Alternatively, our protocol could be used to just transmit data from one party to another, similar to other work on AirTag-based data transmission [6, 2, 18]. When used in this mode, our per-byte key rotation mechanism offers several practical benefits over recent prior work [6].

3.1 Properties

3.1.1 One Public Key Per Byte

While prior work that uses the status byte for transmitting the payload relies on report timestamps and repeated polling for reports [6], we instead designed a key rotation protocol. This lets us send advertisements for each byte of payload data with a unique public key.

This means that even the sender can fetch reports for the public keys they advertised to check whether the corresponding bytes were uploaded.

Additionally, successful data transmission needs just one report per advertised public key, thus avoiding the per-key upload limitations. Unlike in Granzow et al. [6], the receiver doesn't need to worry about Apple's servers sending only up to 2000 reports per key either.

The key rotation also reduces the likelihood of malicious devices around the sender interfering with data transfer by sending advertisements with conflicting data. A device that passively listens for the sender's advertisements and rebroadcasts them with alternate data would have no way of predicting subsequent public keys. Therefore, in case of conflicts, the recipient could prioritize the status bytes in reports that were generated and sent earlier.

Finally, the key rotation property allows a sender to broadcast multiple advertisements in parallel since bytes are ordered not by report timestamps but by the key derivation process.

We note that our key rotation protocol does *not* provide forward-secrecy or post-compromise security; it is not secure in the event that one or more ephemeral secret keys are compromised.

3.1.2 Finder-to-Recipient Encryption

In our protocol, the payload is encrypted between finder devices and the recipient. Finder devices compute an ECDH shared secret with the recipient using the advertised public key and their own key pair. The shared secret is used to derive an AES-GCM key, which is used to encrypt the payload (including the finder's location and the status byte) for the recipient.

However, the payload *will* be observable in the plain by Bluetooth-capable devices in the physical vicinity of the sender. Depending on the application, it may be advisable to use protocols like TLS higher up in the stack.[2]

[2]We note that the Bible offers somewhat conflicting advice on this issue. On the one hand, there is "Do not plot harm against your neighbor, who lives trustfully near you" (Proverbs 3:29), and on the other, there is "Beware of your friends; do not trust anyone in your clan. For every one of them is a deceiver, and every friend a slanderer" (Jeremiah 9:4). We turn, instead, to commandment three of Biggie's *Ten Crack Commandments*, which advises to "never trust nobody".

3.1.3 High-Precision Geolocation

Another unique feature of our protocol is that every byte is accompanied by fairly accurate geographical coordinates of the finder (and by virtue of Bluetooth's low range, the sender) [8]. However, this location is encrypted for the recipient. While this may be detrimental to anonymity on the internet, it may have unexpected trust and safety benefits that warrant further inquiry. The design of anonymity networks over *IP Over AirTag* remains an open problem. We also note that IPv4 and IPv6 addresses are geolocatable, albeit with lower precision, through geolocation databases that use data from regional internet registries.

3.2 Cryptographic Design

Any party wishing to communicate over *IP Over AirTag* samples an *identity* P224 key pair. The public key of the identity key pair must be published to a well-known location. For two parties to communicate over *IP Over AirTag*, they must already know each other's identity public keys. However, they cannot directly use identity keys to communicate since all single-byte messages from all senders to the recipient would appear as reports under the same public key. From the recipient's perspective, it would be impossible to tell which messages came from which sender.

To ensure that senders can address (i.e., "send" by way of broadcasting Bluetooth advertisements) reports to public keys where no one else is likely to send reports, we develop a key negotiation process. Suppose Alice and Bob's identity keys are $\left(\mathrm{sk}_I^A, \mathrm{pk}_I^A\right)$ and $\left(\mathrm{sk}_I^B, \mathrm{pk}_I^B\right)$ respectively. They must first negotiate *channel* P224 key pairs $\left(\mathrm{sk}_C^A, \mathrm{pk}_C^A\right)$ and $\left(\mathrm{sk}_C^B, \mathrm{pk}_C^B\right)$. They must learn each other's channel public keys, but no curious third parties that know both pk_I^A and pk_I^B should learn of the new public keys. If the new public keys are known outside of the sender and the recipient, third parties may be able to advertise them and cause conflicting reports to be sent. To achieve this, we derive a shared secret using ECDH, interpret it as a field scalar, and multiply both parties' secret scalars and public points by it [4]. We use a hash function to perform the ECDH key exchange non-interactively since communicating new keys over a globally writeable medium (reports submitted to identity public keys) with a one-byte bandwidth is infeasible. Since the order of hash inputs must be consistent from both Alice and Bob's perspective, let $\mathrm{pk}_I^{\min}$ be $\min\left(\mathrm{pk}_I^A, \mathrm{pk}_I^B\right)$ and $\mathrm{pk}_I^{\max}$ be $\max\left(\mathrm{pk}_I^A, \mathrm{pk}_I^B\right)$. In practice, min and max could be computed by lexicographically comparing the serialized forms of pk_I^A and pk_I^B. For Alice, the process of generating new keys looks as follows:

$$s = H\left(\mathrm{pk}_I^{\min} \parallel \mathrm{pk}_I^{\max} \parallel \mathrm{sk}_I^A \cdot \mathrm{pk}_I^B\right) \tag{1}$$

$$\left(\mathrm{sk}_C^A, \mathrm{pk}_C^A\right) = \left(s \times \mathrm{sk}_I^A, s \cdot \mathrm{pk}_I^A\right) \tag{2}$$

$$\mathrm{pk}_C^B = s \cdot \mathrm{pk}_I^B \tag{3}$$

After both Alice and Bob compute their own channel key pairs and each other's channel public keys, they effectively have a "private channel" for their communication. Alice (and only Alice) can decrypt reports addressed to her channel public key, and since Bob is the only one (besides Alice) who knows her channel public key, he is likely to be the only one who will address messages to that key up to a collision in H.

In theory, the channel public keys should suffice for communication between Alice and Bob. However, finder devices have limits on the number of reports they upload for a particular public key per day [8, 6]. Additionally, reordering data bytes based on report times [6] is error-prone and relatively unreliable. In light of these limitations, we adopt a key rotation mechanism somewhat analogous to AirTags' own key rotation mechanism, which is motivated, instead, by the risk of stalking.

$$s_0 = H\left(\mathrm{pk}_C^{\min} \parallel \mathrm{pk}_C^{\max} \parallel \mathrm{sk}_C^A \cdot \mathrm{pk}_C^B\right) \tag{4}$$

$$\left(\mathrm{sk}_0^A, \mathrm{pk}_0^A\right) = \left(\mathrm{sk}_C^A, \mathrm{pk}_C^A\right) \tag{5}$$

$$s_{i+1} = H\left(\mathrm{pk}_i^{\min} \parallel \mathrm{pk}_i^{\max} \parallel \mathrm{sk}_i^A \cdot \mathrm{pk}_i^B\right) \tag{6}$$

$$\left(\mathrm{sk}_{i+1}^A, \mathrm{pk}_{i+1}^A\right) = \left(s_i \times \mathrm{sk}_i^A, s_i \cdot \mathrm{pk}_i^A\right) \tag{7}$$

$$\mathrm{pk}_{i+1}^B = s_i \cdot \mathrm{pk}_i^B \tag{8}$$

Thus, no bytes are ever addressed to channel public keys. Instead, byte m_1 from Alice to Bob is addressed to pk_1^B, byte m_2 to pk_2^B, and so on.

3.3 System Design

We developed a platform-agnostic Rust library that implements our key negotiation and rotation protocols. We integrated it with open-source tools [1, 3] that allow conveniently fetching Find My reports to develop a high-level framework for building *IP Over AirTag* applications. Additionally, we developed firmware for Nordic nRF52833 devices that reads BLE data over a serial port and broadcasts it as an advertisement. This allows us to run the *IP Over AirTag* code on an internet-connected laptop (an internet connection is necessary to fetch reports) and use the nRF52833 devices to broadcast BLE advertisements with custom Bluetooth addresses (as required by the Find My protocol). We note that the *IP Over AirTag* code could also be used on a device like the Espressif Systems ESP32, which is equipped with both WiFi and Bluetooth, eliminating the need for a connection to a laptop.

Senders and receivers start with the knowledge of each other's identity public keys and agree upon who will transmit the first frame. They both derive a stream of keys—their own key pairs and the other party's corresponding public keys—as described in 3.2.

Find My is typically an acknowledgment-less protocol, i.e., an accessory receives no notice when a finder uploads a report for it. However, in our setting, the sender can simply fetch reports for the public keys it advertised. Since our key rotation mechanism ensures a one-to-one mapping between data bytes and keys, finding any reports for a public key signals that the corresponding data byte was sent successfully, even though the sender can't decrypt the report payloads themselves. When an application requires high reliability at the *IP Over AirTag* level, it may use this property to re-advertise any keys that weren't picked up by finders and block until it detects that all payload bytes have been uploaded.

In practice, it is important for recipients to know how many bytes of data were sent by the other party. Suppose Bob prematurely fetches reports when only the first 50 of the 100 bytes transmitted by Alice have been uploaded. It is important that Bob doesn't start transmitting data to Alice's 51st public key because Alice would never fetch reports for that public key.

To ensure that both parties' views of the active keys remain synchronized, we use keys in groups of 255, i.e., the frame size. Thus, even if Alice transmits only 100 bytes, she advances her keys by 255. Likewise, when Bob receives 50 bytes, he also advances his keys by 255 (the discrepancy between the data sent and received is for the transport layer protocol to handle). Additionally, we require switching of sender and recipient roles after each frame.

An alternate design could include the payload length in the frames (as in UDP), which would allow parties to send more than one frame at a time while ensuring that both parties use the right keys.

4 Discussion

Our work develops a protocol that allows using the Apple Find My network's "data muling" [2] properties for reliable two-way communication. It achieves this while maintaining the encryption of payload data between finders and recipients. It also offers ways to improve prior work on one-way transmission of arbitrary data over Find My. Our key rotation mechanism allows for more reliable data transmission, verification of transmission by senders, parallel advertisement of public keys, and more. Our protocol's use of multiple keys—one key per byte—also means that we can exploit the ability to fetch reports for up to 255 keys.

Before *IP Over AirTag* can fully replace protocols like IPv6, a measurement of the geographic and environmental factors that influence protocols like Find My, and therefore unnatural networking protocols like *IP Over AirTag*, is in order. This includes measuring or modeling the density of finder devices in parts of the world with low population density and where Apple has a relatively low share of the consumer electronics market.

Future work could explore the feasibility of using zero-day vulnerabilities in finder devices to control location report data before it's encrypted and uploaded. Among other things, this

would allow for a 10× increase in the transmission bandwidth of *IP Over AirTag* and would eliminate the mandatory sharing of the sender's geolocation with the receiver.

We hope to find ways to lift the requirement for communicating parties to share their identity public keys out-of-band, which would allow us to implement *IP Over AirTag* servers. We can also imagine improvements to our cryptographic protocol (e.g., offering better security when ephemeral keys are compromised) and our network protocol (e.g., improving the key synchronization mechanism).

Finally, we would also love to see an analysis of *IP Over AirTag*'s efficacy as a censorship-resistant communication protocol.

Despite its relatively lower efficiency in environments with few finder devices, our protocol encourages users to leave their basements and be around others, which we consider socially valuable.

5 Conclusion

We developed a protocol to use Apple's Find My network for arbitrary communication between two parties. Additionally, our cryptographic protocol improves upon previous designs for AirTag-based one-way data transmission schemes [6]. We are releasing our proof-of-concept code and our Rust library for interfacing with Find My as open-source software [14, 15]. We hope that protocols like *IP Over AirTag* will encourage more internet users to "touch grass". Finally, we hope that this work will encourage the development of more unnatural networking protocols that build on the legacy of *IP Over Avian Carriers* [20].

References

[1] Mike Almeloo. *malmeloo/FindMy.Py*. Dec. 2023. URL: https://github.com/malmeloo/FindMy.py.

[2] Alex Bellon, Alex Yen, and Pat Pannuto. "TagAlong: Free, Wide-Area Data-Muling and Services". In: *Proceedings of the 24th International Workshop on Mobile Computing Systems and Applications*. HotMobile '23. New York, NY, USA: Association for Computing Machinery, Feb. 2023, pp. 103–109. ISBN: 979-8-4007-0017-0. DOI: 10.1145/3572864.3580342. URL: https://dl.acm.org/doi/10.1145/3572864.3580342.

[3] Dadoum. *Dadoum/anisette-v3-server*. URL: https://github.com/Dadoum/anisette-v3-server.

[4] Edward Eaton, Douglas Stebila, and Roy Stracovsky. *Post-Quantum Key-Blinding for Authentication in Anonymity Networks*. 2021. URL: https://eprint.iacr.org/2021/963.

[5] Harry Eldridge et al. "Abuse-Resistant Location Tracking: Balancing Privacy and Safety in the Offline Finding Ecosystem". In: *33rd USENIX Security Symposium (USENIX Security 24)*. 2024, pp. 5431–5448. ISBN: 978-1-939133-44-1. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/eldridge.

[6] Max Granzow et al. "Leveraging Apple's Find My Network for Large-Scale Distributed Sensing". In: *Proceedings of the 22nd Annual International Conference on Mobile Systems, Applications and Services*. MOBISYS '24. New York, NY, USA: Association for Computing Machinery, June 2024, pp. 666–667. ISBN: 979-8-4007-0581-6. DOI: 10.1145/3643832.3661412. URL: https://dl.acm.org/doi/10.1145/3643832.3661412.

[7] 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. New York, NY, USA: Association for Computing Machinery, June 2021, pp. 374–376. ISBN: 978-1-4503-8349-3. DOI: 10.1145/3448300.3468251. URL: https://doi.org/10.1145/3448300.3468251.

[8] Alexander Heinrich et al. "Who Can Find My Devices? Security and Privacy of Apple's Crowd-Sourced Bluetooth Location Tracking System". In: *Proceedings on Privacy Enhancing Technologies* (2021). ISSN: 2299-0984. URL: https://petsymposium.org/popets/2021/popets-2021-0045.php.

[9] Bob Hinden and Brian E. Carpenter. *Adaptation of RFC 1149 for IPv6*. Request for Comments RFC 6214. Internet Engineering Task Force, Mar. 2011. DOI: 10.17487/RFC6214. URL: https://datatracker.ietf.org/doc/rfc6214.

[10] Amir Houmansadr, Chad Brubaker, and Vitaly Shmatikov. "The Parrot Is Dead: Observing Unobservable Network Communications". In: *2013 IEEE Symposium on Security and Privacy*. May 2013, pp. 65–79. DOI: 10.1109/SP.2013.14. URL: https://ieeexplore.ieee.org/document/6547102.

[11] Brent Ledvina et al. *Detecting Unwanted Location Trackers*. Internet Draft draft-detecting-unwanted-location-trackers-01. Internet Engineering Task Force, Dec. 2023. URL: https://datatracker.ietf.org/doc/draft-detecting-unwanted-location-trackers.

[12] maelp. *Scaling to Thousands of Devices · Issue #99 · malmeloo/FindMy.Py*. Jan. 2025. URL: https://github.com/malmeloo/FindMy.py/issues/99.

[13] Travis Mayberry, Erik-Oliver Blass, and Ellis Fenske. "Blind My - An Improved Cryptographic Protocol to Prevent Stalking in Apple's Find My Network". In: *Proceedings on Privacy Enhancing Technologies* 2023.1 (Jan. 2023), pp. 85–97. ISSN: 2299-0984. DOI: 10.56553/popets-2023-0006. URL: https://petsymposium.org/popets/2023/popets-2023-0006.php.

[14] Shreyas Minocha and George Ari Hosono. *shreyasminocha/ip-over-airtag: All Your Status Bytes Are Belong to Us*. 2025. URL: https://github.com/shreyasminocha/ip-over-airtag.

[15] Shreyas Minocha and George Ari Hosono. *shreyasminocha/offline-finding: Code to Interface with and Implement Apple's Offline Finding (AirTag) Protocol*. 2025. URL: https://github.com/shreyasminocha/offline-finding.

[16] SideStore. *SideStore/Omnisette-Server*. SideStore Team. Mar. 2025. URL: https://github.com/SideStore/omnisette-server.

[17] Zhen Sun and Vitaly Shmatikov. "Telepath: A Minecraft-based Covert Communication System". In: *2023 IEEE Symposium on Security and Privacy (SP)*. May 2023, pp. 2223–2237. DOI: 10.1109/SP46215.2023.10179335. URL: https://ieeexplore.ieee.org/document/10179335.

[18] Leonardo Tonetto et al. "Where Is My Tag? Unveiling Alternative Uses of the Apple FindMy Service". In: *2022 IEEE 23rd International Symposium on a World of Wireless, Mobile and Multimedia Networks* (2022). Ed. by Liming Luke Chen et al., pp. 396–405. DOI: 10.1109/WoWMoM54355.2022.00059. URL: http://www.scopus.com/inward/record.url?scp=85137103236&partnerID=8YFLogxK.

[19] David Waitzman. *IP over Avian Carriers with Quality of Service*. Request for Comments RFC 2549. Internet Engineering Task Force, Apr. 1999. DOI: 10.17487/RFC2549. URL: https://datatracker.ietf.org/doc/rfc2549.

[20] David Waitzman. *Standard for the Transmission of IP Datagrams on Avian Carriers*. Request for Comments RFC 1149. Internet Engineering Task Force, Apr. 1990. DOI: 10.17487/RFC1149. URL: https://datatracker.ietf.org/doc/rfc1149.

[21] Tingfeng Yu et al. "Security and Privacy Analysis of Samsung's Crowd-Sourced Bluetooth Location Tracking System". In: *33rd USENIX Security Symposium (USENIX Security 24)*. 2024, pp. 5449–5466. ISBN: 978-1-939133-44-1. URL: https://www.usenix.org/conference/usenixsecurity24/presentation/yu-tingfeng.