Jeremy Martin*, Douglas Alpuche, Kristina Bodeman, Lamont Brown, Ellis Fenske*, Lucas Foppe, Travis Mayberry*, Erik Rye*, Brandon Sipes, and Sam Teplov

Handoff All Your Privacy – A Review of Apple's Bluetooth Low Energy Continuity Protocol

Abstract: We investigate Apple's Bluetooth Low Energy (BLE) Continuity protocol, designed to support interoperability and communication between iOS and macOS devices, and show that the price for this seamless experience is leakage of identifying information and behavioral data to passive adversaries. First, we reverse engineer numerous Continuity protocol message types and identify data fields that are transmitted unencrypted. We show that Continuity messages are broadcast over BLE in response to actions such as locking and unlocking a device's screen, copying and pasting information, making and accepting phone calls, and tapping the screen while it is unlocked. Laboratory experiments reveal a significant flaw in the most recent versions of macOS that defeats BLE Media Access Control (MAC) address randomization entirely by causing the public MAC address to be broadcast. We demonstrate that the format and content of Continuity messages can be used to fingerprint the type and Operating System (OS) version of a device, as well as behaviorally profile users. Finally, we show that predictable sequence numbers in these frames can allow an adversary to track Apple devices across space and time, defeating existing anti-tracking techniques such as MAC address randomization.

Keywords: BLE, Bluetooth, privacy, tracking

DOI 10.2478/popets-2019-0057
Received 2019-02-28; revised 2019-06-15; accepted 2019-06-16.

***Corresponding Author: Jeremy Martin:** The MITRE Corporation, E-mail: jbmartin@mitre.org
Douglas Alpuche: U.S. Naval Academy (USNA)
Kristina Bodeman: USNA
Lamont Brown: USNA
***Corresponding Author: Ellis Fenske:** USNA, E-mail: fenske@usna.edu
Lucas Foppe: USNA
***Corresponding Author: Travis Mayberry:** USNA, E-mail: mayberry@usna.edu
***Corresponding Author: Erik Rye:** CMAND, E-mail: rye@cmand.org
Brandon Sipes: USNA
Sam Teplov: USNA

1 Introduction

The ubiquity of wirelessly connected mobile devices in the day-to-day lives of people globally has brought with it unprecedented risk of privacy violation for modern consumers. Mobile devices constantly transmit and receive information even while not in active use, and many of the protocols driving this communication are not designed with privacy in mind.

Tracking concerns and privacy leakages in 802.11 Wi-Fi are well-known and have been extensively studied over the last decade. Since Wi-Fi clients must actively probe for nearby access points to connect to, an adversary can listen to these probes and use the device's MAC address (which is included in probes) to identify and track it as it moves from place to place. This is not an academic threat: there are multimillion-dollar companies [39, 60] whose business model relies on using Wi-Fi tracking data for targeted marketing, and they control large networks of Wi-Fi access points that gather information on all nearby devices. Users are largely unaware that these widely-deployed tracking capabilities exist and that their Wi-Fi devices might be leaking sensitive data.

In response to this threat, device and OS manufacturers began to provide MAC address randomization as a privacy enhancement. Rather than using the same MAC address consistently, which enables correlation over multiple observations, devices employing MAC randomization instead choose random values, and change them periodically. While the principle itself is sound, many implementations of MAC address randomization have proven ineffective in practice [47, 64]. Defeating MAC address randomization is largely possible due not to flaws in Wi-Fi itself, but because of extraneous information in higher-layer protocols. Many technologies are not privacy-aware and leak information that can be used to track users and devices, despite the MAC address being effectively hidden through randomization.

Bluetooth, in both of its current protocol instantiations, also uses MAC addresses as hardware identifiers. BLE, which we examine exclusively in this study, has included mechanisms for a device to generate and use ran-

dom MAC addresses, enhancing the potential privacy benefit to clients by increasing the difficulty of tracking unique devices. Unfortunately, it also suffers from the same problem as Wi-Fi: manufacturers and OSes implement features built on top of BLE that leak sensitive information which can be used to track users [35], defeating the purpose of MAC randomization itself.

In this work we investigate one such technology – Apple's Continuity protocol. Continuity is designed to support the seamless transfer and synchronization of data between multiple iOS and macOS devices. We show that in exchange for simplifying the user experience and allowing synchronization between devices, the messages that comprise Continuity constantly leak sensitive information; not only identifiers that could be used to defeat randomization and track devices wirelessly, but also behavioral information that reveals user device activity.

1.1 Contributions

- To the best of our knowledge, we are the first to reverse engineer Apple's Continuity protocol. We describe the format, contents, and behavior of Continuity messages.
- We identify several recognizable messages that leak behavioral data, including when a user locks or unlocks their phone, when they are touching the screen, when they visit certain apps or settings pages, when they receive text messages or answer a phone call, and even when they use the copy/paste feature.
- We show that observing these messages allows an adversary to accurately fingerprint the model type and OS version of a device. We also present a novel application of a known, hard-to-detect, active reconnaissance technique that can further increase the precision of these fingerprints.
- Finally, we demonstrate how the predictable sequence numbers sent with these messages can be exploited to defeat anti-tracking technologies like MAC address randomization and detect the existence of a device nearby even after not observing it for several days.

2 Background and Related Work

2.1 Wireless Device Tracking and Privacy

Wireless layer-2 identifiers – in particular, MAC addresses – have a rich and lengthy history of being exploited for tracking mobile devices, and hence, their owners. The broadcast nature of wireless communication, the tendency for software designers to program devices to proactively advertise their availability and seek out services, and the persistence and uniqueness of global MAC addresses combine to form a serious threat to user privacy: our mobile devices constantly broadcasting trackable identifiers for all to hear without our interaction.

In order to detect nearby wireless networks, 802.11 Wi-Fi clients broadcast a special type of wireless frame known as a *probe request*. The purpose of these frames is to solicit responses from local access points that provide network connectivity. Contained in each frame is the client's source MAC address; this 48-bit value uniquely identifies the client seeking network service to access points that provide it. Unfortunately, this also provides adversaries a unique identifier to track users [20, 24, 30, 48, 50]. To address the privacy concerns inherent in the use of the MAC address assigned by the device manufacturer (a *globally unique* MAC address), manufacturers have shifted to broadcasting ephemeral, random MAC addresses while the device is in an unassociated state; unfortunately, MAC address randomization in 802.11 can often be defeated [47, 64].

MAC addresses are used as layer-2 identifiers in Bluetooth communication as well, and carry the same privacy and tracking risks as Wi-Fi MAC addresses [20, 38, 43, 65]. Similar to Wi-Fi, Bluetooth devices implement MAC address randomization as a mechanism to evade tracking and preserve user privacy. In fact, randomized MAC addresses have been part of the BLE standard since its introduction (known then as Bluetooth Smart) [4]. As we focus exclusively on BLE in this work, we first discuss BLE MAC address structure and randomization in greater detail.

2.2 Bluetooth (Classic vs. Low Energy)

The term *Bluetooth* is often used colloquially to refer to two distinct, non-interoperable technologies. The original Bluetooth standard is now referred to as Bluetooth "Classic" to recognize its chronological precedence, or Bluetooth Basic Rate (BR)/Enhanced Data Rate (EDR) in relation to the rate at which a device implementing this protocol transmits data. BLE, formerly known by the marketing name "Bluetooth Smart", on the other hand, is so named due to the lower power consumption needs of devices implementing it compared to Bluetooth BR/EDR. Despite their typical use in connecting peripheral devices at close range, both Bluetooth Classic and BLE are capable of transmitting up to 100m in an open area. The current BLE version, 5.0, is rated up to 400m [14].

Generally speaking, Bluetooth Classic is preferred for applications in which a constant flow of data occurs between the paired devices; for example, Bluetooth headphones or speakers connected to a mobile phone would almost certainly utilize Bluetooth Classic. BLE, on the other hand, is typically used to send short messages advertising a device's existence and some parameters related to the device; for instance, Tile [16] and related devices use BLE for proximity sensing. In this study, we are interested only in Apple's BLE implementation, as it lends itself to device tracking, OS fingerprinting, and activity profiling.

While Bluetooth Classic and BLE operate in the 2.4 GHz unlicensed spectrum, each utilizes a different number and size of channels. BLE uses 40 2 MHz channels; 37 of these are used to send and receive data, while the remaining three are used to detect a device's presence by sending advertisement frames at regular intervals [4]. With exception of our Generic Attributes (GATT) queries in Section 4.3, we are concerned in this work with messages continuously sent in the advertisement channels that are designed to be received by nearby stations.

2.3 BLE Addressing and Randomization

Every Bluetooth interface is assigned a globally-unique, *public* MAC address by the device manufacturer; Bluetooth MAC addresses are EUI-48 identifiers and are obtained from the Institute of Electrical and Electronics Engineers (IEEE) Registration Authority in the same manner as 802.11 MAC addresses. In order to provide users with a measure of privacy and prevent privacy leakages [46] associated with revealing a public device address, devices are also permitted to use *random* device addresses, which are split into two categories – *static* random addresses and *private* random addresses, the latter of which is further divided into two subcategories.

Static random addresses are, as their name implies, random addresses that are long-lived; the Bluetooth Core Specification mandates that these BLE MAC addresses remain unchanged after initialization [4]. Power cycling a device may change its static random address, but changing a device's static address will cause any devices that have previously connected to it to fail to automatically connect to the previous static address. Static random addresses are identifiable by having the two highest order bits set to 1, at least one of the remaining 46 low-order bits set to 1, and at least one of the lower 46 bits set to 0 (*i.e.*, two set bits followed by 46 0s or 46 1s are not allowable static random addresses.)

Non-resolvable private random addresses provide additional privacy compared to using a public MAC address, as the random address is used in lieu of the public, globally-unique MAC address of the device. Non-resolvable random addresses are identifiable by the two most significant bits being set to 0, and the remaining 46 lower order bits containing at least one 0 and one 1 bit [4]. Finally, as their name implies, non-resolvable private addresses do not aid in authenticating two devices to each other.

Resolvable private random addresses are the final type of random address in Bluetooth, and are the type used by Apple to provide MAC address privacy; as such, this is the address type we consider in this work. Resolvable private addresses provide the ability for devices to authenticate each other based on the use of a 128-bit key, known as an *Identity Resolving Key (IRK)*. When a device is configured to use a resolvable private address, it generates 22 pseudorandom bits (*prand*), and uses its *local IRK* and *prand* as inputs into a one-way security function, the output of which is a 24-bit hash [4]. The resolvable private address is created by setting the most significant bit to 0, second-most significant bit to 1, concatenated with the 22 bits of *prand*, followed by the 24-bit hash result. The device then uses this resolvable private address as the source MAC address for a set period of time. Oftentimes the period is 15 minutes, as [4] recommends 15 minutes as the minimum time interval between private random address changes. Resolvable private random addresses have the advantage of allowing potential peers to determine if they already know a device. If the potential peer has the IRK (exchanged during initial pairing) of the remote device it wishes to connect to, it can determine whether an advertised random address belongs to that device or not through the following process: the would-be peer computes the 24-bit hash value given the *prand* value from the resolvable private address, and the pre-shared IRK. If the value computed locally matches the lower 24 bits of the resolvable address, the peer's identity has been confirmed to be that associated with the IRK when key exchange was initially done.

2.4 Related Work

Significant previous work exists related to tracking mobile devices via 802.11 Wi-Fi MAC addresses [24, 30, 39, 50, 53, 55, 58], tracking via cellular identifiers [40, 45, 49, 53–55, 61, 63] and attempting to correlate randomized 802.11 MAC addresses to the same physical device [47, 55, 64]. By contrast, our work fo-

cuses on the BLE technology, and specifically uses flaws in Apple's Continuity protocol (Section 2.5) to track devices despite randomization of the Bluetooth hardware identifier. More closely related to this study is previous work on tracking users via Bluetooth identifiers and the discovery of privacy leakages in Bluetooth protocols [22, 32, 35, 38, 42, 43, 65]. Of these studies, most focus on Bluetooth Classic [38, 43, 65], whereas we study Apple's BLE Continuity protocol messages exclusively. Fawaz *et al.* [35] develop a tool aimed at preventing privacy leakages in BLE devices by restricting who can discover, scan, and connect to BLE; to date, this tool has not been widely adopted outside of a laboratory setting. In [32], Das *et al.* examine the privacy leakages present in wearable fitness tracking devices, as well as the ability to track these devices and therefore their owners. Unlike [32], Apple devices implement BLE MAC address randomization, which significantly increases the difficulty of tracking them. Korolova and Sharma [42] examine the feasibility of *cross-application tracking* in Android and iOS devices: the ability for an application to fingerprint applications running on nearby devices by actively scanning those devices.

In our work, we do not rely on information gleaned from external applications installed on our devices, but rather we are able to track users based on OS-default features alone. Like our study, Stute *et al.* examine a proprietary protocol used by Apple to enhance interoperability between iOS and macOS devices. In [62], Stute *et al.* reverse engineer Apple's Apple Wireless Direct Link (AWDL) protocol, an extension to 802.11 that enables AirDrop and other Apple services. While AWDL leverages BLE as a discovery mechanism, the authors are interested in using the AWDL implementation for tracking purposes, rather than BLE as in our work.

Contemporaneously and most closely related to our work, Becker *et al.* [22] examine tracking devices using randomized BLE identifiers. While they examine OSes we do not (Windows, Android), their Apple evaluation considers the BLE messages to be largely uninterpretable data; we exhaustively reverse-engineer the Continuity protocol, revealing both the structure of the messages as well as what actions are required to produce them. This affords us the ability to behaviorally profile users, fingerprint major iOS version and device type, and greatly enhances the potential to track users despite the use of anonymized MAC addresses, as detailed in Sections 4 and 5.

Finally, an expansive body of literature deals with fingerprinting OS type and version remotely, often by soliciting replies from targets via crafted ICMP and TCP messages [23, 25, 28, 29, 33, 41, 44, 56, 57, 59], via DHCP options and User-Agent strings [5], or in mobile devices by examining properties of the radio transmission [36, 51], MAC and upper layer protocols [26, 34, 37], or both [66]. Because the scope of our study is restricted to Apple devices, our fingerprinting focus is on differentiating between major release versions of Apple's iOS and macOS operating systems. Our fingerprinting capability is limited to within BLE transmission distance of the target device, but requires no active transmissions to elicit a reply from the target and is derived from variations in the format of Apple's Continuity messages themselves.

2.5 Apple Continuity

Continuity is an umbrella term used by Apple to describe interoperability features between various devices within its ecosystem; for example, the ability to copy text on an iPhone and paste that same text on a MacBook linked via the same iCloud account [2, 11]. Continuity was introduced in iOS 8 and OS X Yosemite, although some features require more recent software versions [1]. Continuity features include:

- **Handoff**, which allows users to start tasks, such as writing an email, and continue on another device.
- **Universal Clipboard**, which allows the copying of data from one Apple device and pasting it on another.
- **iPhone Cellular Calls**, giving users the ability to make calls using their iPhone's cellular connection while on their Mac, iPad, or iPod.
- **Instant Hotspot**, which supports turning an iPhone or iPad into a secure hotspot other Apple devices may use without requiring a password.
- **Auto Unlock**, allowing users to unlock a Mac with their Apple watch.
- **Continuity Camera**, which transfers photos taken on an iPhone, iPad, or iPod touch to a Mac.

Continuity features are enabled by the transmission of special BLE advertisement messages sent between devices on the same iCloud account; as such, all Continuity-enabled devices are BLE-capable. In this work, we reverse-engineer the Apple-proprietary message formats and describe the operation of those Continuity messages that enable our OS fingerprinting and user tracking techniques in Section 4.

3 Methodology

Our analysis was conducted using open-source software and off-the-shelf commodity hardware. We perform tests against a wide range of iPhone, iPad, iPod, airPod, and Apple Watch devices across major and minor iOS versions. Additionally we experiment with a sample of macOS laptops. Passive collection testing was implemented using an Ubuntu environment and an Ubertooth One USB receiver [17] with *2018-12-R1* firmware [19]. When utilizing the Ubertooth, we run the `ubertooth-btle` software [18] to collect BLE advertisement frames. While running `ubertooth-btle`, we set the `-q` option with `DLT_BLUETOOTH_LE_LL_WITH_PHDR` to ensure we can capture the Received Signal Strength Indicator (RSSI) value for each frame. The `stdout` of `ubertooth-btle` is piped directly to Wireshark where we are able to conduct live analysis.

As our reverse engineering observations revealed the frame format, message type, and data attributes of the Apple Continuity protocol, we modified Wireshark's dissection logic in the `packet-bthci_cmd.c` file in order to properly dissect Continuity messages.

In section 4.3, we describe a technique to elicit model-granularity details from Apple devices, and discuss replaying two previously observed types of Continuity messages. To carry out these active techniques, we utilize a Sena Technologies UD100 Bluetooth USB adapter, along with the software `gatttool` and `hcitool`. We use `gatttool` to query our devices' GATT, and `hcitool` to spoof arbitrary Continuity frames.

3.1 Ethical Considerations

Our collection methodology is entirely passive. At no time did we attempt to decrypt any user data, alter normal network behavior, or attempt to track any individuals not associated with our research team without their prior knowledge. Additionally, in order to evaluate the privacy flaws we present in this work, we conduct a variety of experiments on lab devices owned by the authors and the authors' institutions. These devices were allowed to communicate with legitimate network services. Given the nature of our data collection, we consulted with our Institutional Review Board (IRB).

The primary concerns of the IRB centered on: i) the information collected; and ii) whether the experiment collects data "about whom" or "about what." Because we limit our analysis to BLE advertisement frames we do not observe Personally Identifiable Information (PII). Further, humans are incidental to our experimentation as our interest is in OS and hardware profiling, device usage, and the analysis of the randomization of BLE device layer-2 MAC addresses, or "what."

Finally, in consideration of beneficence and respect for persons, our work presents no expectation of harm, while the concomitant opportunity for network measurement and security provides a societal benefit. Our experiment was therefore determined by the IRB to not be human subject research.

4 Analysis

This section is divided into three parts:

- Analysis of passively collected data to reverse engineer the frame format and data attributes of Apple's BLE Continuity framework. We show through this effort, that Continuity features leak a significant amount of data related to user behavior.
- Active attacks transmitting tailored BLE frames to elicit responses from Apple devices revealing further details regarding user information and behavior.
- A comprehensive evaluation of the effectiveness of these attacks with respect to an adversary's ability to identify and track devices and users.

Since we analyze many different types of Continuity messages, each with different flaws, we organize our findings by calling attention specifically to the following categories, based on what information is leaked:

- **OS fingerprinting**
- **Device fingerprinting**
- **Tracking**
- **User / Device Activity**
- **Device attributes**

When a flaw is observed, we classify it into one or more of these categories and describe how it can be exploited by a potential adversary.

4.1 Passive Analysis Reverse Engineering

We evaluate Apple's proprietary Continuity protocol by inspecting BLE frames emitted from iOS and macOS devices across Apple's ecosystem and OS versions.

As described in Section 3, we collect BLE frames passively using an Ubertooth with `stdout` piped to Wireshark (using our custom dissector), allowing for real-time dynamic analysis via a live capture display. Apple Continuity frames are transmitted on all three BLE advertisement channels; as such, our Ubertooth collection setup requires monitoring only a single advertisement channel.

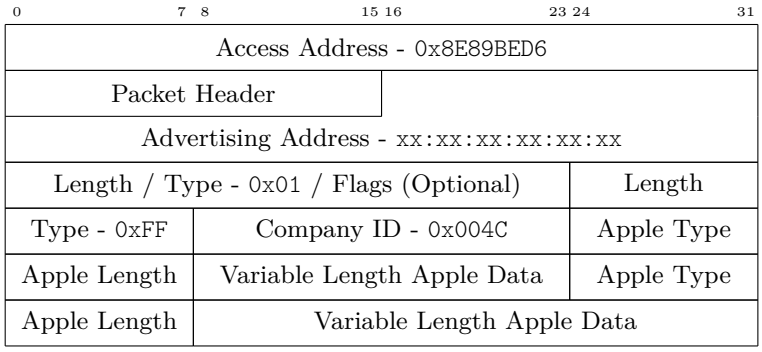

Fig. 1. Apple BLE Frame Format with Hardcoded Field Values

4.1.1 BLE Advertisement Frame Prevalence

As a preliminary study, we first sought to understand the prevalence of BLE in representative public locations. Our goal was to understand what types of devices, OSes, and ecosystems were employing BLE and whether privacy countermeasures such as MAC address randomization were frequently deployed. Table 1 depicts two distinct measurements: the first is a multi-hour single collection, while the second comprises several distinct collections over two days. In both cases our results indicate that only two major ecosystems are commonly observed utilizing BLE MAC address randomization – Apple and Microsoft Windows devices. It should be noted that the counts for *Random* MAC addresses are skewed, and therefore should not be interpreted as the number of distinct devices observed. This is a reflection of the interval policy used by Apple and Microsoft, in which the random BLE MAC address rotates every 15 minutes.

Table 1. Advertisement Frames

		Test 1	Test 2
		Count	
Address Type	Public	26	57
	Random	726	1,518
Company ID†	Apple	692	1296
	Microsoft	30	201
	Garmin	2	9
	Samsung	0	3
	All Others	2	9

† Randomized Devices Only

Microsoft's Connected Devices Platform (MS-CDP) discovery protocol [52] provides a framework to allow users to verify and authenticate devices and exchange messages between devices. As the protocol is delineated in a published specification we center our analysis on reverse engineering Apple's Continuity protocol. To refine our analysis to focus solely on Apple BLE traffic, we filter for BLE frames containing Apple's Company ID (0x004C) [15].

Device Fingerprinting

While the Company ID is required as per specification [15], it allows for simple identification of BLE traffic generated by Apple devices.

4.1.2 Configuration Settings - Disabling Continuity

An underlying condition for Continuity messages to be sent is the user having their device associated with an iCloud account to which at least two devices are registered; because users regularly neglect to remove old Apple products from their iCloud account years after discarding or retiring the device, this condition is routinely met even when the user may only actively use one Apple device.

While they are enabled by default, Handoff messages described in Section 4.1.6 are unique in that they can be explicitly turned off in the Settings Menu. Some message types require a device with a cellular connection to be associated with the iCloud account in order to be generated by the user, namely the WiFi Settings and Instant Hotspot messages, outlined respectively in Sections 4.1.7 and 4.1.8, which need a cellular-capable device to act as a hotspot.

Activating Airplane Mode does not disable the transmission of Continuity messages in either iOS 11 or 12, regardless of whether Airplane Mode is activated from the Control Center or Settings Menu. Similarly, disabling Bluetooth from the Control Center in iOS 11 or 12 does not discontinue the transmission of Continuity messages [3]. We determined that the only way to stop transmission of all Continuity messages is to disable Bluetooth from the Settings Menu.

Table 2. Most Commonly Observed Continuity Messages

Type	Value	iOS Version					Vulnerability				
		8	9	10	11	12	OS FP	Device FP	Tracking	Activity	Attributes
Watch Connection	11	N	N	N	Y	Y	✗	✓	✗	✗	✗
Handoff	12	Y	Y	Y	Y	Y	✗	✗	✓	✓	✓
Wi-Fi Settings	13	8.1+	Y	Y	Y	Y	✗	✗	✓	✓	✗
Instant Hotspot	14	8.1+	Y	Y	Y	Y	✗	✗	✓	✓	✓
Wi-Fi Join Network	15	N	N	N	Y	Y	✗	✗	✓	✓	✗
Nearby	16	N	N	Y	Y	Y	✓	✗	✓	✓	✓

4.1.3 Overall Message Structure

After segregating all Apple traffic from our collection by filtering for Apple's Company ID, we observe that Apple's Continuity frames adhere to a simple Type-Length-Value (TLV) structure delineated in Figure 1. Of note, multiple Continuity message types are often concatenated together in this TLV format, allowing them to be passed in a single advertisement frame.

Device Fingerprinting

We observe that optional BLE advertisement flags indicate a device category, allowing us to delineate MacBooks from mobile devices (iPhone, iPad, iPod, and watches). Specifically, we investigate the following flags:

- Simultaneous LE and BR/EDR to Same Device Capable Host (**H**)
- Simultaneous LE and BR/EDR to Same Device Capable Controller (**C**)
- Peripheral device is LE only (**LE**)

Mobile devices were observed with flags **H, C,** and **LE** set to `1,1,0`, whereas MacBooks were set to `0,0,1`. AirPods lacked any flags and were thereby easily identifiable as the only device type with no flag attributes.

After adding the identified TLV structure, our custom-defined Apple BLE fields, and associated attributes to the `packet-bthci_cmd.c` Wireshark dissector, we proceed to reverse engineer the most commonly observed Continuity messages. For each new message type and attribute we update the dissector, recompile, and reevaluate across multiple devices and OSes. Table 2 highlights the message types and corresponding values we observed. Also annotated are the message type mappings to applicable iOS versions. The message types' descriptive names were generated by our group in an attempt to properly categorize the message.

4.1.4 Other

While iBeacon messages uniquely identify iBeacon nodes, these devices, as their name implies are not meant to be anonymous and we conduct no further analysis of the iBeacon message type.

Additionally, AirDrop and AirPlay messages were observed, albeit infrequently and were not examined in our study; [62] *et al.* provide a thorough investigation of privacy leakages and tracking mechanisms enabled by the AirDrop protocol. Though we did not make direct use of any of these message types, we reverse engineered enough of their format to allow us to detect and ignore them in our study.

Device Fingerprinting

Lastly, we observed that AirPods transmit a unique message type and were trivially identified via observation of these messages.

4.1.5 Watch Connection

An Apple Watch transmits message type `11` when the watch has lost a Bluetooth connection with the paired iPhone.

Device Fingerprinting

This message distinctly identifies Apple Watches as they are solely transmitted from an Apple Watch.

4.1.6 Handoff

Handoff messages occur when a user interacts with a Handoff-enabled application such as Mail, Maps, Safari, Calendar, Contacts, Pages, Numbers, Keynote, and third-party applications such as Airbnb and Google Chrome [8]. In addition to being triggered by user interaction with Continuity-enabled applications, Handoff messages are also observed when these applications are opened or closed. Unlike other message types, Handoff messages can be disabled explicitly through the settings page, though they are enabled by default. Handoff messages are also only observed when tied to an iCloud account that contains two or more Apple devices, as Handoff cannot work with a lone device.

Having identified the user behaviors that generate Handoff messages, we focus on recovering privacy-sensitive information with the message itself. Figure 2 depicts the Handoff frame, indicated by a Type field of `0x0C`.

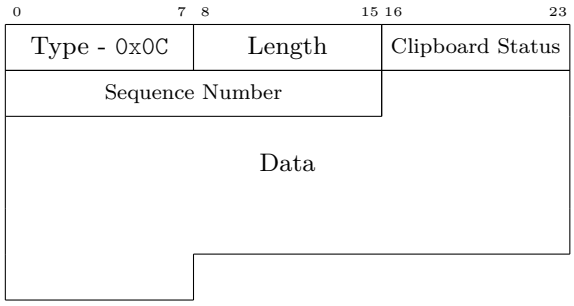

Fig. 2. Handoff Message - Frame Format

Device Attributes

The Handoff message contains a one-byte field that contains an indicator we call "clipboard status" indicating when a user has recently copied data to the Universal Clipboard and is now available to transfer to nearby Apple devices. This behavior was observed across iOS 10, 11, and 12. A value of 0x08 indicates that data are stored in the Universal Clipboard, and 0x00 if not.

Tracking

A two-byte sequence number follows the clipboard status field. The sequence number increments only when a new action occurs in a Handoff-enabled application, the application is opened/closed, the phone is unlocked, or the phone is rebooted. As such, the sequence number increments monotonically, slowly, and at a rate proportional to Handoff-enabled app use, allowing for long-duration tracking. The remaining bytes in the frame appear to be encrypted data and provide no specific details about the user's activity that we could infer.

Importantly, sequence numbers are not affected by MAC address randomization. We observe MAC address changes that preserve the sequence number and encrypted Handoff data before and after the expiration of the *private_addr_int* timer. This allows for the trivial association of two random MAC addresses, and, because addresses change on a fixed 15 minute schedule, allows a passive adversary to prepare for the next rotation 15 minutes hence. This behavior was consistently observed across all iOS major and minor versions we tested, from iOS 9 through 12.3. We describe this tracking vulnerability in detail in Section 5.

User Activity

The sequence number also carries information about private user behavior. Since it increases only in response to user-generated events, it serves as a crude, real-time measurement of user activity. Measuring the sequence number of a device at two different times allows an adversary to infer how much the phone was used during that time period, leaking information about the activity between measurements.

4.1.7 Wi-Fi Settings

Another Continuity message type, which we refer to as "Wi-Fi Settings", is transmitted when the user navigates to the Wi-Fi Settings page in iOS or clicks on the Wi-Fi and network status icon on the top of their screen in macOS. While the settings page is open, an Apple device will continuously transmit Wi-Fi Settings frames, as depicted in Figure 3. This feature requires that a device other than the one in use is registered to the same iCloud account, and that the second device has a cellular radio and is Instant Hotspot capable.

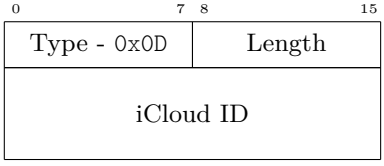

Fig. 3. Wi-Fi Settings Message - Frame Format

User Activity

This message indicates to an observer that the user is currently on the Wi-Fi Settings page and is likely configuring or about to connect to a Wi-Fi network.

Tracking

A four-byte data field representing an iCloud-derived ID trivially links all other devices tied to the same iCloud account. The derived ID is rotated on a 24 hour basis for all devices on the account where each device remains synchronized with all other devices on the same account; as the ID is ephemeral, however, it is not trackable for more than 24 hours.

4.1.8 Instant Hotspot

Instant Hotspot is an Apple Continuity feature that allows users to share cellular network connectivity among iCloud-linked devices by creating a bridged Wi-Fi connection. The Instant Hotspot feature allows Apple devices to seamlessly identify and connect to these personal hotspot-enabled devices through the use of the Instant Hotspot message. Devices configured to support Instant Hotspot will automatically begin transmitting Instant Hotspot messages when another device on the same iCloud account is nearby and is transmitting Wi-Fi Settings messages containing the correct iCloud-derived ID. The hotspot-enabled device continues to broadcast Instant Hotspot messages as long as the other device remains on the Wi-Fi Settings menu.

Device Activity

This message is inherently descriptive of the device's ability to support acting as an Access Point (AP), and assuming a connection is established, the device will be recognizable in the 802.11 domain as an AP.

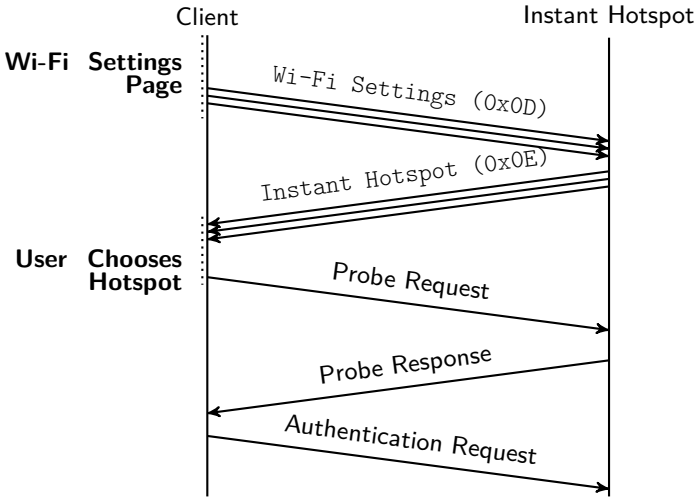

Fig. 4. Instant Hotspot Discovery and Connection Setup

Tracking

The Instant Hotspot connection process, described in an abbreviated form in Figure 4, highlights several tracking issues. First, upon attempting to connect to an Instant Hotspot network, the client searches for the Hotspot by sending directed and broadcast probe request frames. This is similar to the well-documented probe request privacy flaw in which a user can be profiled based on the Service Set IDentifier (SSID) of the networks for which it actively searches [21, 31].

Upon receiving a probe request from a would-be client, the Instant Hotspot device transmits probe responses and beacon frames using a deterministically-generated, locally-administered MAC address rather than its true, *global public* MAC address. The probe responses and beacons transmitted by the Instant Hotspot device provide a second tracking mechanism through the inclusion of a vendor-specific Information Element (IE) that includes a reversible permutation of the Bluetooth and Wi-Fi globally-unique MAC addresses as shown in [47].

Finally, the client device transmits an 802.11 authentication frame, in which the source address is the *global public* MAC address of the device. Strangely, we observe that probe request frames also use the *global public* MAC address of the device; this is unusual because iOS devices generally randomize MAC addresses in probe requests when in an un-associated state.

Device Attributes

The Instant Hotspot message reveals surprising device characteristics in plaintext as shown in Figure 5. Specifically, the device's battery life, cellular service type (e.g. LTE, 3G, EV-DO), and cellular service quality (measured as a function of number of bars) are all transmitted unencrypted.

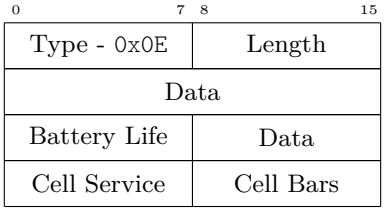

Fig. 5. Instant Hotspot Message - Frame Format

Tracking

Lastly, due to the nature of how an Instant Hotspot message is elicited we can trivially determine that the initiator device (Wi-Fi Settings message transmitter) and the Instant Hotspot device are associated to the same iCloud account.

4.1.9 Wi-Fi Join Network

An additional Wi-Fi themed Continuity message type, in which we annotate as "Wi-Fi Join Network", is transmitted when a user attempts to connect to an encrypted network from the Wi-Fi Settings page. We call attention to the fact that this message is only sent when a password is required; therefore, open and captive portal-enabled networks will not generate Wi-Fi Join Network messages.

User / Device Activity

We note that the observation of a Wi-Fi Join message indicates the intent by a user to connect to an encrypted network regardless of whether the proper credentials are entered.

Tracking

A similar tracking flaw to the one described in Section 4.1.8 allows an adversary to link the BLE communication (Wi-Fi Join message) to the frames observed in the 802.11 (Wi-Fi) domain. Specifically, as delineated in Figure 6 the observer can compare the timestamps of the Wi-Fi Join message to the authentication frames collected at the same time. As the authentication frame contains the *global public* MAC address [47], anonymization is broken in both the Wi-Fi and BLE protocols.

Due to the state in which this message is sent in the process (prior to authentication), we can retrieve the client's *global public* MAC address even when a user attempts to authenticate with an invalid passcode.

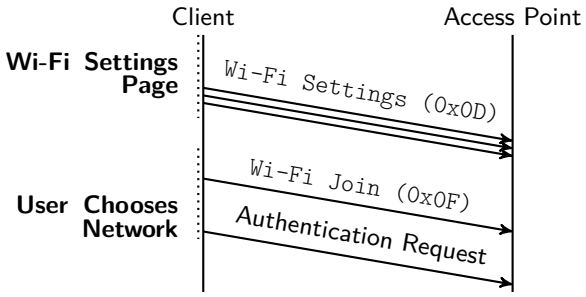

Fig. 6. Wi-Fi Join – Authentication Frame Global MAC Exposed

Another glaring implementation flaw centers on a three-byte SSID field in the Wi-Fi Join frame, depicted in Figure 7. This field represents the first three bytes of a SHA256 hash of an SSID the client device is attempting to join. An adversary can pre-compute hashes of nearby SSIDs, allowing for trivial correlation.

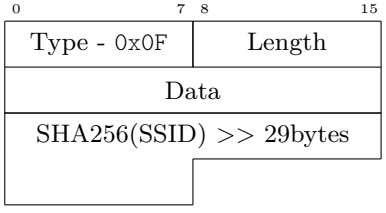

Fig. 7. Wi-Fi Join Network Message - Frame Format

4.1.10 Nearby

Nearby messages, presumably intended to keep *nearby* devices aware of the state of other devices in the same iCloud ecosystem, are transmitted frequently by Continuity-enabled Apple devices. As of iOS 12 and macOS Mojave, Nearby messages **never** stop transmitting, and are sent at a rate of over 200 frames per minute. Because of the frequency and consistency with which Nearby messages are transmitted and the user-behavioral data contained in their payload, Nearby messages represent a serious privacy and tracking issue.

Support for Nearby messages began with release of iOS 10. Observable changes in the frame format and usage correspond with each iOS major version release thereafter. In earlier implementations (iOS 10 and 11, and macOS High Sierra) Nearby messages timeout after a period of inactivity, meaning that devices cease to transmit Nearby messages when a device is left inactive after approximately 30 seconds. As of iOS 12 and macOS Mojave, devices continuously transmit Nearby messages while BLE services are not disabled and the device is on, and so long as the lid is not closed on a macOS device.

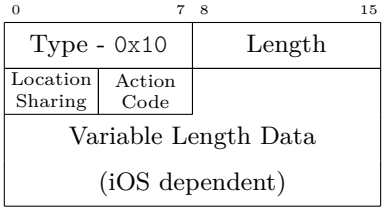

Fig. 8. Nearby Message - Frame Format

User / Device Activity

The Nearby message contains a one-byte field, referenced in Figure 8, of which the least significant nibble we designate the "Action Code" field, as it indicates the *action* or *state* of the Apple device. The most significant nibble, not observed in use prior to iOS 12.3, indicates if the device has been configured to "Share My Location" with family and friends. As of iOS 12.3 this field is set to `1` when the user specifically selects this device as the sharer of the locational data. Every iCloud account has a single device selected as its location sharer, and this field is set to `1` for this selected device, and `0` for all other devices on the account. This behavior is observed regardless of whether location services is on.

Our research highlighted in Table 3 describes seven commonly observed Action Codes.

Table 3. Action Codes

Type	Description
1	iOS recently updated
3	Locked Screen
7	Transition Phase
10	Locked Screen, Inform Apple Watch
11	Active User
13	Unknown
14	Phone Call or Facetime

Action Code 11 A user is actively interacting with the associated device (iOS or macOS). This Action Code will continue to be transmitted until a period of ∼30 seconds of inactivity, upon which it will then send Nearby messages with Action Code 7, and later Action Code 3 as the screen becomes locked (iOS).

Action Code 3 An iOS device in a screen locked state, indicating a lack of interaction between the user and the device.

Action Code 10 Informs a paired Apple Watch that the connected mobile device is in a locked screen state. In this scenario the mobile device will send only Action Code 10 messages vice the previously mentioned Action Code 3, likely to indicate that the phone will forward notifications to the watch when the device is locked and that the watch should display the notifications.

Action Code 7 Observed after ∼30 seconds of user-device (iOS or macOS) inactivity or when the device transitions from a locked state to that of an active state (caused by waking the device).

Action Code 14 Observed after the release of iOS 12.3 when transmitted from iOS devices in an active phone call or Facetime session.

Action Code 13 Observed in the wild after the release of iOS 12.3, however we were unable to reproduce this message in our laboratory experiments.

Action Code 1 We observed this Action Code rarely. In our experiments, we observed this Action Code from iOS devices recently updated to a new iOS version which have not been rebooted after the full update. Additionally, we often observed these from locked macOS devices, however we were unable to definitively attribute an action to these messages.

Device Attributes

iOS 12 and macOS Mojave Nearby messages contain additional data, highlighted in Table 4, in which the state of the Wi-Fi radio can be inferred, namely whether Wi-Fi is enabled or disabled. If the first byte of the data field is `0x18` the Wi-Fi radio is off, whereas when the value is `0x1C` the Wi-Fi radio is on.

Table 4. Nearby Message (iOS) – Data Field

Feature	iOS Version		
	10	11	12
Length (bytes)	1	4	4
Byte 1	0x00	0x10	0x18 || 0x1C
Byte 2-4	-	Data	Data

†: 0x18 (Wi-Fi Off), 0x1C (Wi-Fi On)

OS Fingerprinting

The variable nature of the Nearby message data field allows for OS version fingerprinting. Specifically, the length of the frame and the value of the first byte distinctly reveal the iOS major version. As such we can infer a device's iOS major version as either iOS 10, 11, or 12 with 100% accuracy. Similarly, we can identify macOS as Mojave or pre-Mojave OS versions, however we must first evaluate the BLE flags as described in section 4.1.3 in order to separate iOS from macOS devices.

Tracking

Similar to the privacy flaw discussed in Section 4.1.6, the Nearby data field does not immediately change when the MAC address is periodically rotated, allowing for trivial tracking of a device across MAC address changes. This is further compounded with the behavior observed in iOS 12 and macOS Mojave where Nearby messages are continuously transmitted.

4.2 macOS BLE MAC Randomization Breaks Itself

For macOS devices running High Sierra and Mojave, we observed that Nearby messages change when Handoff messages are being sent concurrently. Normally, all Apple Continuity messages utilize the current *resolvable private random address*. This remains true with macOS under circumstances where only Nearby messages are being transmitted by the macOS device. However, when a Handoff message or Wi-Fi Settings message is sent, the Nearby messages switch to the ***global public*** MAC address. The observed Handoff or Wi-Fi Settings messages continue to use the properly randomized address. Nearby messages are transmitted with global MAC address continuously while these other messages are being sent, then revert back to the randomized address.

In practice, this means active use of Safari or Google Chrome on a Handoff-enabled device will generate a constant flow of Nearby messages with the global MAC address. Furthermore, the data fields of both normal and these Handoff-concurrent Nearby messages match, allowing for trivial correlation of the current randomized address to the device's real MAC address. This passive observation technique entirely circumvents BLE MAC randomization for any macOS device so long as the device has another device associated to its iCloud account, Handoff has not been disabled, and any Handoff-enabled application is in use.

This macOS implementation flaw correlates the current random MAC address, public MAC address, and current Handoff sequence number, all of which are useful for tracking a device. Further, knowledge of the public Bluetooth MAC provides an adversary with the ability to detect its presence in any setting by initiating a connection attempt without needing sequence numbers at all. Finally, the Bluetooth MAC is often offset ±1 from the Wi-Fi MAC address [27, 45], enabling a passive adversary to obtain an 802.11 identifier through BLE collection.

4.3 Device Stimulation & Active Analysis

A feature unique to BLE is GATT [6], a framework to discover, read, and write information to and from BLE devices. Each BLE device that supports GATT has a GATT *profile*; GATT profiles define the type of *services* that a device provides. Within each service are well-defined *characteristics*, and each characteristic may contain several fields and values that describe that characteristic. For example, "Blood Pressure" is a GATT service that describes blood pressure and other data related to blood pressure readings from a monitor [7]; within the

blood pressure service are the "blood pressure measurement" and "intermediate cuff" characteristics that describe a blood pressure reading, and cuff pressure value during a blood pressure reading, respectively. Each of these characteristics has a number of fields with associated values, such as "timestamp" and "heart rate" that provide further information to a remote device querying them. Services and characteristics are uniquely identified by Universally Unique IDentifiers (UUIDs), in order to standardize their meaning across a myriad of manufacturers.

Device Fingerprinting

While not a vulnerability in its own right, we discovered that Apple devices support GATT queries and provide detailed model information when it is requested. All Apple products we tested (iPhone, iPad, Apple Watch, and MacBook) supported the "Device Information" service (UUID `0x180A`), and responded to its "Model Number String" characteristic (UUID `0x2A24`) with Apple's identifier string that uniquely identifies the device model [10, 12]. For example, an iPhone 7 we tested returned the identifier "iPhone9,1", and a mid-2015 15-inch MacBook Pro with Retina display returned "MacBookPro11,4." We also note that while an adversary must actively transmit data in order to retrieve the "Model Number String" GATT characteristic, she may do so using a random source MAC address, and the user of the device is unaware that they have received a GATT query, as no prompt appears asking them to approve the data transmission. Because of this, it is exceptionally difficult to detect and prevent an adversary from querying an Apple device model without disabling Bluetooth entirely. Although using the "Device Information" service to obtain the device model is itself not novel (indeed, this is its purpose), in Section 5, we show that this knowledge assists in tracking individual devices. Devices that respond with a different "Model Number String" than our tracking target can be excluded from the pool of potential new identities after a change in random MAC address.

The remainder of our active attacks focus on two complimentary types of Apple Continuity messages – Wi-Fi Settings and Instant Hotspot. Wi-Fi Settings messages are generated when a user navigates to the Wi-Fi Settings page of their iOS device, or when a user clicks on the Wi-Fi and network status icon on the top of their screen on macOS. These messages trigger Instant Hotspot messages in response from devices linked to the same iCloud account that can act as a hotspot, and leads to the device appearing in the user's list of available networks when viewed from a MacBook.

Device Attributes

In order to enumerate the possible values and meaning for the cellular field in the Instant Hotspot message described in Section 4, we spoof previously-captured Instant Hotspot messages from laboratory iPhones in response to Wi-Fi Settings messages from a MacBook Pro on the same iCloud account. By enumerating the possible values for the 1-byte cellular service field and observing the type of service displayed for our spoofed device in the laptop's available networks list, we were able to exhaustively classify each value without having a device actually receiving that type of service (*e.g.*, LTE, 3G, EV-DO, etc.) We note that the source MAC address in our spoofed messages must be a resolvable private random address that the MacBook can resolve, for this reason, we choose to replay the source MAC observed.

Tracking

We demonstrate that spoofing Wi-Fi Settings messages provides a tangible benefit for an attacker. Because a device on the same iCloud account that can provide Instant Hotspot service will respond without user intervention when it receives a correct Wi-Fi Settings message from a laptop, we are able to replay previously-captured Wi-Fi Settings messages hours later. As with Instant Hotspot message spoofing, the source MAC address must be one that can be resolved by the iPhone or iPad that can provide the hotspot service, and we note that the iCloud ID field in the message changes daily at a fixed time per iCloud account. As such, these messages may be replayed for a maximum of 24 hours.

Tracking

Finally, we measure the effect of incoming SMS messages and phone calls on devices running in order to determine whether an adversary with knowledge of their target's phone number could stimulate their device in order to discover whether it is in close proximity.

All of the iOS versions we tested began sending Handoff messages when a phone call was accepted, or the Messages application was opened in response to an SMS message. iOS 9 did not send Nearby messages, as Nearby messages are not a feature of any iOS earlier than 10. In addition to sending Handoff messages when a user takes an incoming call or opens the Messages app in response to an SMS message, iOS 10 and 11 also send Active User Nearby messages in addition to the Handoff messages that were sent in iOS 9.

Concerningly, the latest iOS version, iOS 12, proved the most useful for targeted tracking of users because it required *no interaction on behalf of the user* in order to determine whether a phone call was incoming or a text had been received. Because iOS 12 devices transmit

Nearby messages constantly, an attacker merely needs to observe a change in the Nearby message Action Codes. Pre-iOS 12.3, a change from the locked state (Nearby Action Code 3) to the transition state (Nearby Action Code 7), indicated an incoming call or text, as the screen is illuminated when a call or text is received. An adversary with the ability to send a text message or initiate a call to a device can trivially identify the device if it is in close proximity, defeating the private, random MAC address and allowing for the type of user tracking we outline in Section 5. Finally, the version of iOS at time of publication, 12.3, makes detecting an incoming phone or Facetime call even more trivial by introducing Nearby Action Code 14; a device transmitting this Action Code is in an active call.

5 Evaluation

Next we evaluate how effectively the above flaws, in concert, can be used by an adversary to track a device.

5.1 User Tracking

As described above, BLE devices periodically rotate their MAC address in order to prevent tracking. Ideally, the MAC address is regularly set to a fresh random value and potential adversaries eavesdropping on a device at different times and/or locations will not be able to correlate these observations and confidently identify them as belonging to the same device. In our tests we find that Apple devices rotate the addresses every 15 minutes, as recommended by the BLE specification [4]. However, the data leakage described above can be used, in many cases, to defeat randomization and track a device even through MAC address changes.

The main flaw which allows this is the fact that Handoff messages are sent out regularly and with monotonically increasing sequence numbers. The first implication of this is if an adversary is present at the time that a MAC address is changed to a new random value, this transition is identifiable because the sequence number and data fields of Handoff messages do not change even when the MAC address does. We have observed that after a MAC address change the sequence number stays constant until a new Handoff-related action is performed by the user before continuing to increment.

Similarly, the four-byte data field in iOS 11 and 12 Nearby messages, rather than immediately rotating when the MAC address changes, remains constant for one to two frames after the MAC address change. As annotated in Section 4.1.10, Nearby messages never stop transmitting in iOS 12 or macOS Mojave, making this information leak a more powerful tracking method.

5.1.1 Longer Time Frames

Over longer time scales, it is very likely that an adversary will not be present to observe every MAC address change, and will eventually lose track of the MAC address of a particular device. Due to the slow monotonic increase of the sequence number and the relatively large sequence number space it is possible to track devices by using the sequence number itself as an identifier.

To illustrate and quantify the effectiveness of this tracking capability we performed four sets of measurements that demonstrate the predictability of sequence numbers and the likelihood that a device can be uniquely identified in a public place by their sequence number. First, we measured how the sequence numbers for devices owned by members of our research team increase over time during regular daily use. Second, we measured the distribution of sequence numbers in public places. Third, we estimated the probability of a second device coincidentally having a sequence number close to the target, representing a false positive for a tracking adversary. Finally, we moved a device belonging to a member of our research team through a variety of public places over time and capture BLE signals, attempting to identify this targeted user's device.

Sequence Number Trajectories. Since sequence numbers increment when specific user actions are taken on the device (use of a Handoff-enabled app, device setting manipulation, SMS/call activity) we hypothesized the rate at which sequence numbers increase is stable and predictable based on the usage patterns of individual users. This would mean the sequence numbers of devices may be predicted with high accuracy even when devices are not under observation, and devices that leave an adversary's collection region may be re-identified when returning to collection proximity.

Five members of our research team (four students, one faculty) used their iPhones normally over a period of one week including the weekend and recorded their sequence numbers as close to hourly as possible. We present our results in Figure 9, displaying the trajectories as well as $|u|$, the size of the projected window of sequence numbers for that user over time, as described below for accuracy estimates. We expect sequence numbers to increase more quickly with the use of Handoff-enabled apps on the device. In addition, we left a device on but unused for the duration of the experiment and observed that its sequence number did not increment.

Given the predictable per-user slope of the observed sequence numbers, we believe that over a period of a few days to a week the sequence number of a particular device can be predicted with a high degree of accuracy.

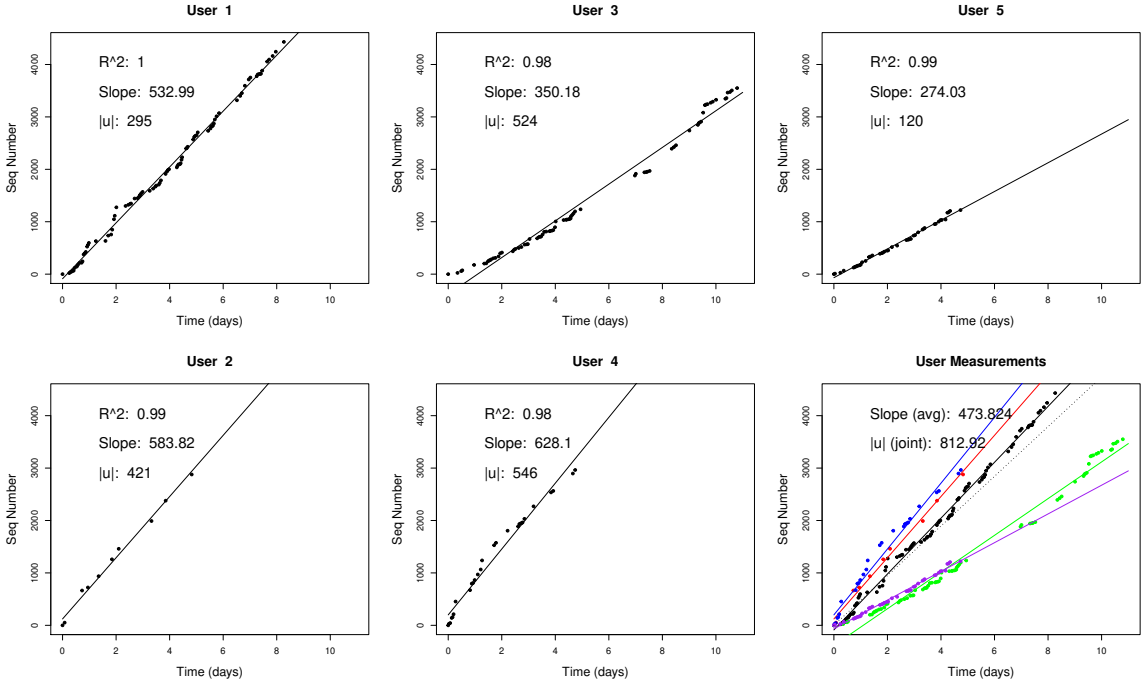

Fig. 9. Regressions of our collected data on sequence numbers. |u| is calculated with 90% prediction intervals

We note that while we have evidence to suggest this predictability occurs in certain cases (among members of our research team in a given week), we only conjecture that this is the case more generally.

Sequence Number Distributions. Additionally, we took passive captures of sequence numbers in the wild at four distinct public locations. We hypothesize that sequence numbers are uniformly distributed in the space $[0, 65535]$ since, while sequence numbers begin at zero, they increase consistently through regular use. We passively collected BLE signals and removed redundant measurements by ignoring multiple Handoff messages from the same MAC address. Our results are presented in Appendix A.

Estimates. In order to provide estimates on the accuracy of this tracking method we use our measurement data and data from external sources wherever possible. In cases where the data is unavailable we make conservative independence assumptions, noting that in general more precise measurements of the distributions required for these estimations will result in more accurate tracking capabilities than we outline here.

We estimate collision probabilities under our different attack models against iPhones. Our passive reconnaissance outlined in Section 4.1 allow an adversary to bin devices by OS version, while the active attacks from Section 4.3 allow the adversary to determine the granular hardware submodel of a device as well.

We use statistics from Mixpanel [13] to estimate the proportion of Apple devices of each type in the wild as of February 25, 2019 and statistics from Apple [9] as of February 24, 2019 to estimate the proportion of devices with each iOS installed. We note that iPhone 7 is the most common hardware model and in order to estimate the proportional size of the largest bin we assume:

– That the distributions of iOS version number and iPhone hardware model are independent. Given this, we select the most prevalent iOS version, version 12.
– That iPhone 7 devices are split evenly between the two hardware sub-models (iPhone9,3 vs iPhone 9,1) [10]

For an estimate of the adversary's accuracy in re-identifying a device, we consider two cases: targeted, where an adversary takes measurements of a specific device and wishes to track it over time, and untargeted, where an adversary does not have data but attempts to determine if a device is a previously observed device or a new one. The adversary must calculate a window of plausible sequence numbers associated with a previously observed device to determine whether a new measurement is the previous device or a new one, and the adversary may incorporate knowledge about the expected use patterns of its target to determine this window. In the targeted case, where an adversary has made many measurements of a specific user over time we estimate as the size of the window $u = 421$ the median size of

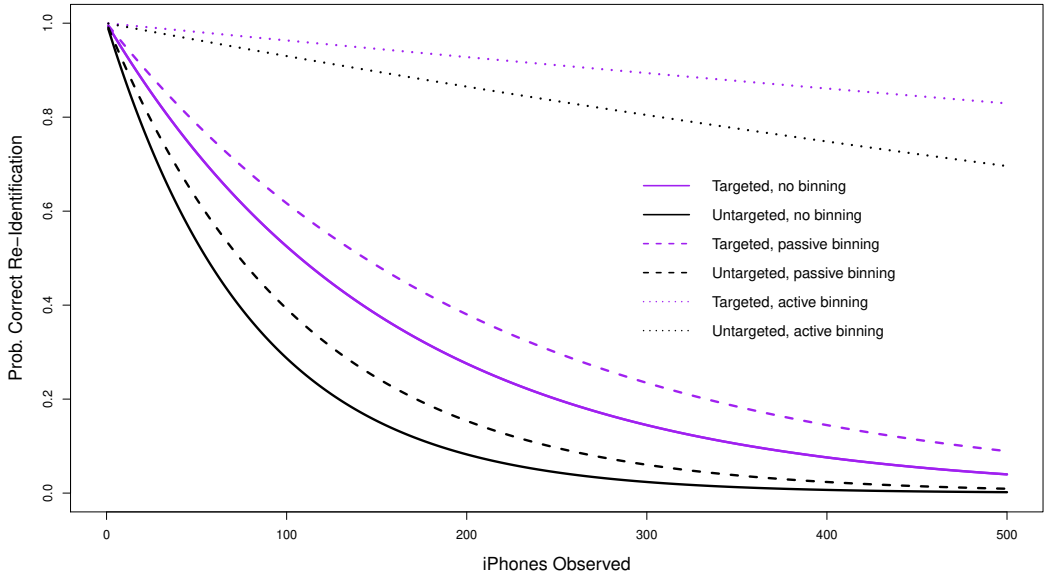

Fig. 10. Re-identification accuracy under different attack scenarios

the largest 90%-prediction interval size from our five experiments, and in the untargeted case, where an adversary assumes the device increment rate sits between our largest and smallest observed rates we take the more conservative convex closure of all 90% prediction intervals, $[u_{\text{MIN}}, u_{\text{MAX}}]$ which gives $u = 813$.

In this setting we can calculate the probability that a target device will be the only one in a given location with a sequence number in the target window as

$$\left(1 - \frac{u}{65536}\right)^n$$

where n is the number of devices that are identically configured to the target (i.e., look the same according to all of our binning techniques).

We calculate the likelihood that a user is correctly re-identified (i.e., that an in-bin sequence number collision does not occur) under each possible attack scenario in Figure 10.

Sequence Number Collisions. We performed eight short measurements in public locations, presented in Figure 11, to verify the viability of our tracking methods in a real-world scenario. Each measurement was in a different location in the same city, lasting up to 40 minutes each. User 1 from our trajectory experiments took their iPhone 6S with a known sequence number and moved to each location and another researcher captured nearby Handoff traffic, with a third applying methods described in Section 4.3 in an attempt to identify the hardware models of devices in close proximity. In the last three experiments, a different user was targeted who had an iPhone 6. Nearly every device we observed in the wild was running iOS 12 so we do not include software binning in our results. We note that our experimental setup was not able to accurately identify hardware models for most devices in the first five experiments, so we ignore hardware profiles for these measurements and consider them a test of tracking without binning. For measurements six, seven, and eight we made significant improvements to our collection software and identified the hardware model of 81%, 70%, and 75% of devices respectively. The devices with hardware profiles that are unknown or that match the target device we take as collisions, and we note that in total 90 of 465 total devices were successfully binned by hardware, and of these 90, four were in the same bin as the target device.

Unlike our estimates, these measurements include devices of all types (watches, notebooks, etc) while in the estimates above our unit of measure is observed number of iPhones. These experiments test the targeted setting, so we use as our window the median size from our trajectory experiments $|u| = 421$.

We conclude, from our estimates and measurements:

- Active adversaries can reliably re-identify even the most common devices over time in public places without any long-term targeted data collection.
- Passive adversaries can reliably re-identify devices that are less common, by targeting specific users over time, or when the number of observed devices is low.
- Negative results about the presence of a given device are highly accurate, even for passive adversaries.

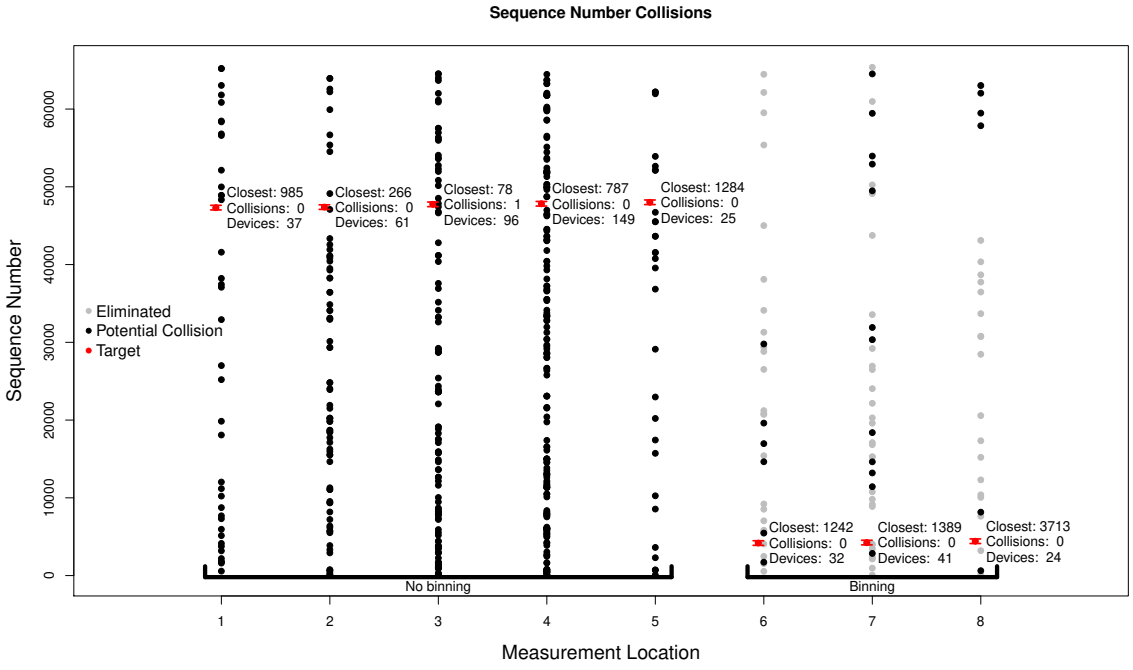

Fig. 11. Collision Measurements

5.2 Behavioral Data

We note the sequence number is a measurement of user interactions with the device that is persistent over time and regularly broadcast over BLE. Similarly, Nearby Action Codes explicitly broadcast information on how a device is being used in the moment. This means BLE traffic from Apple devices provides multiple sources of private behavioral data. In particular, this allows adversaries to make statistical inferences about the quantity of Handoff device interactions that have occurred between measurements. Behavioral statistics of this nature present an area for interesting future work, though we recommend mitigation strategies to Apple for future software versions changes so that this information can no longer be collected.

6 Remediation

In this section, we recap the flaws we discovered and discuss how they could be addressed. In most cases, there is nothing that a user can do to protect themselves. The fixes have to be addressed by Apple at the kernel or firmware level. We believe the most straightforward solution to most of these flaws is to either remove the plaintext information that is being leaked, if it is not necessary, or encrypt it with the shared encryption key used across all devices on the same iCloud account. We have notified Apple of our findings and hope to work with them to address the issues we have identified.

macOS Global MAC Address

We believe that this behavior is an unintended bug resulting in a serious tracking vulnerability. As such it should be easily correctable and we recommend Apple release an updated patch correcting the behavior.

MAC randomization

In general, the frequency with which MAC addresses are rotated (every 15 minutes) is a substantial flaw. If an adversary is within range of the device when it rotates, even fixing all of the flaws we have discussed so far, it is not usually difficult to link two subsequent randomized MAC addresses. This is because the rotation event is so infrequent that an adversary can observe that one MAC address stops transmitting at precisely the same time that another one starts and deduce that they are actually the same device. Even worse, since it always happens every 15 minutes, it can be predicted like clockwork. The ideal solution would be for every frame to have a new randomized MAC, but if that is too burdensome then devices should at least have a shorter period of rotation, and that rotation should be done stochastically instead of on a fixed timer.

GATT commands

An adversary can use GATT to query BLE-enabled Apple devices for their exact model even when their MAC address is being randomized. To reduce identifiability of devices, iOS could respond to this query with a less specific identifier (i.e., just "Apple" or "iOS"), or not respond to it at all since it is optional.

Nearby messages

The Nearby messages are responsible for the most consistent behavioral leakage. Since there is an Action Code for when the device locks, when the device unlocks, when the user interacts with the device and another one that constantly broadcasts while the screen is on, a passive adversary can always tell whether the device is:

- Locked
- Unlocked and being actively used
- Unlocked but not being actively used
- Currently in a phone or Facetime call

Encrypting the Action Codes would effectively stop someone from learning this behavior since the Nearby messages (at least in iOS 12 and macOS Mojave) are sent constantly at a regular interval. If the Code is hidden then all that could be seen is a regular transmission of a message with no link to specific behavior.

Handoff messages

The main vulnerability we have discovered in Handoff messages is the presence of a monotonically increasing two-byte sequence number which allows an adversary to defeat MAC address randomization and track a device over a long period of time. However, there appears to be no need for a sequence number in these frames at all. Handoff messages are sent over the BLE advertising channel, meaning that these messages are not part of an established communication link between devices. If a message is missed, there is no mechanism for a receiving device to ask for it to be repeated anyway. The sequence numbers should be removed.

We also found that the clipboard status was leaked in the clear. This message should be encrypted. Additionally, the data field in each Handoff message (which we believe *is* encrypted) stays constant when a device's MAC address rotates, allowing an adversary to link two subsequent MAC addresses. Therefore, this encryption should be refreshed whenever the MAC address changes.

Wi-Fi Settings and Instant Hotspot

When a user opens the Wi-Fi settings page on an idevice, a Wi-Fi Settings frame is sent. Nearby devices that are capable of Instant Hotspot respond with an Instant Hotspot frame containng information such as battery life of the device, how many bars of cell service it has, and what type of cellular network it is connected to. The Wi-Fi Settings frame also contains a cleartext iCloud ID, shared amongst all devices on the same iCloud account. This not only leaks information about the state of the phone, but may also allow an adversary to link different devices to the same owner.

The information about the device (battery life, etc.) should be encrypted. Similarly, there is no reason to use a static identifier that facilitates tracking and linking of devices. The identifier could also be encrypted with fresh randomness so that it does not appear the same in two different Wi-Fi Settings frames.

iOS devices also transmit a message when joining a network which includes a hash of the SSID that they are joining. Again, this hash should incorporate the shared encryption key so that it is cannot be easily reversed by an adversary.

7 Conclusion

In this work we present several flaws in Apple's Continuity protocols which leak device and behavioral data to nearby listeners. Individually, each flaw leaks a small amount of information, but in aggregate they can be used to identify and track devices over long periods of time, despite significant efforts in other parts of the BLE protocol to prevent this scenario (MAC randomization). Device designers use short range wireless communications protocols like BLE to generate a fluid user experience across many devices, a very attractive product feature given the proliferation of connected mobile devices. However, securely deploying these technologies presents difficult privacy challenges. Not only do these short-range transmissions inherently leak location information, it seems the most practical uses for these communications are real-time activity updates which are themselves user data. Finally, the short range nature of these technologies means that signals on two independent channels can often be identified as coming from the same device, meaning that privacy vulnerabilities in one wireless domain could entirely trivialize well-implemented safeguards in another as we have presented here.

Acknowledgment

We would like to thank the Apple privacy team who provided prompt feedback and guidance. Views and conclusions are those of the authors and should not be interpreted as representing the official policies or position of the U.S. Government. The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author. The authors additionally thank Dane Brown, Caroline Sears, Peter Ryan, and Robert Beverly for technical assistance and feedback.

References

[1] Apple Continuity Requirements. https://support.apple.com/en-us/HT204689, . Accessed: 2019-02-24.

[2] Apple Continuity Support. https://support.apple.com/en-us/HT204681, . Accessed: 2019-02-24.

[3] Use Bluetooth and Wi-Fi in Control Center with iOS 11 and Later. https://support.apple.com/en-us/HT208086, . Accessed: 2019-02-24.

[4] Bluetooth Core Specification. https://www.bluetooth.com/specifications/bluetooth-core-specification. Accessed: 2019-02-11.

[5] Fingerbank. https://fingerbank.org. Accessed: 2019-06-04.

[6] GATT Overview. https://www.bluetooth.com/specifications/gatt/generic-attributes-overview, . Accessed: 2019-02-21.

[7] GATT Specifications. https://www.bluetooth.com/specifications/gatt, . Accessed: 2019-02-21.

[8] Handoff Apps. https://support.apple.com/en-us/HT209455. Accessed: 2019-02-24.

[9] App store stats. https://developer.apple.com/support/app-store/. Accessed: 2019-02-24.

[10] The iPhone Wiki: Models. https://www.theiphonewiki.com/wiki/Models. Accessed: 2019-02-21.

[11] Apple macOS Continuity. https://www.apple.com/macos/continuity/. Accessed: 2019-02-24.

[12] Apple: Identify Your MacBook Pro Model. https://support.apple.com/en-us/HT201300. Accessed: 2019-02-21.

[13] Mixpanel Device Statistics. https://mixpanel.com/trends/report/iphone_models. Accessed: 2019-02-27.

[14] Things You Should Know About Bluetooth Range. https://blog.nordicsemi.com/getconnected/things-you-should-know-about-bluetooth-range. Accessed: 2019-02-28.

[15] Bluetooth company identifier list. https://www.bluetooth.com/specifications/assigned-numbers/company-identifiers. Accessed: 2019-02-24.

[16] tile. https://www.thetileapp.com/en-us/. Accessed: 2019-02-18.

[17] Ubertooth One. https://github.com/greatscottgadgets/ubertooth/wiki/Ubertooth-One, . Accessed: 2019-05-01.

[18] ubertooth-btle. https://github.com/greatscottgadgets/ubertooth/blob/master/host/README.btle.md, . Accessed: 2019-05-01.

[19] Ubertooth 2018-12-R1 Release Notes. https://github.com/greatscottgadgets/libbtbb/releases/tag/2018-12-R1, . Accessed: 2019-05-01.

[20] N. Abedi, A. Bhaskar, and E. Chung. Bluetooth and Wi-Fi MAC Address Based Crowd Data Collection and Monitoring: Benefits, Challenges and Enhancement. 2013.

[21] M. V. Barbera, A. Epasto, A. Mei, V. C. Perta, and J. Stefa. Signals from the Crowd: Uncovering Social Relationships through Smartphone Probes. In *Proceedings of the 2013 conference on Internet measurement conference*, pages 265–276. ACM, 2013.

[22] J. K. Becker, D. Li, and D. Starobinski. Tracking Anonymized Bluetooth Devices. *Proceedings on Privacy Enhancing Technologies*, 1:17.

[23] R. Beverly. A Robust Classifier for Passive TCP/IP Fingerprinting. In *International Workshop on Passive and Active Network Measurement*, pages 158–167. Springer, 2004.

[24] B. Bonné, A. Barzan, P. Quax, and W. Lamotte. WiFiPi: Involuntary Tracking of Visitors at Mass Events. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a*, pages 1–6. IEEE, 2013.

[25] J. Caballero, S. Venkataraman, P. Poosankam, M. G. Kang, D. Song, and A. Blum. FiG: Automatic Fingerprint Generation. 2007.

[26] J. Cache. Fingerprinting 802.11 Implementations via Statistical Analysis of the Duration Field. *Uninformed. org*, 5, 2006.

[27] J. Cache, V. Liu, and J. Wright. *Hacking exposed wireless: wireless security secrets & solutions*. Number Sirsi) i9780072262582. McGraw-Hill, 2007.

[28] Y.-C. Chen, Y. Liao, M. Baldi, S.-J. Lee, and L. Qiu. OS Fingerprinting and Tethering Detection in Mobile Networks. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, pages 173–180. ACM, 2014.

[29] M. Cristea and B. Groza. Fingerprinting smartphones remotely via ICMP timestamps. *IEEE Communications Letters*, 17(6):1081–1083, 2013.

[30] M. Cunche. I Know Your MAC Address: Targeted Tracking of Individual Using Wi-Fi. *Journal of Computer Virology and Hacking Techniques*, 2014.

[31] M. Cunche, M. A. Kaafar, and R. Boreli. I Know Who You Will Meet This Evening! Linking Wireless Devices Using Wi-Fi Probe Requests. In *2012 IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–9. IEEE, 2012.

[32] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra. Uncovering Privacy Leakage in BLE Network Traffic of Wearable Fitness Trackers. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pages 99–104. ACM, 2016.

[33] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee. Identifying Unique Devices Through Wireless Fingerprinting. In *Proceedings of the first ACM conference on Wireless network security*, pages 46–55, 2008.

[34] J. P. Ellch. Fingerprinting 802.11 Devices. Technical report, Naval Postgraduate School, Monterey, CA, 2006.

[35] K. Fawaz, K.-H. Kim, and K. G. Shin. Protecting Privacy of BLE Device Users. In *25th USENIX Security Symposium USENIX Security 16)*, pages 1205–1221, 2016.

[36] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker. Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting. In *USENIX Security Symposium*, volume 3, pages 16–89, 2006.

[37] D. Gentry and A. Pennarun. Passive Taxonomy of WiFi Clients Using MLME Frame Contents. *arXiv preprint arXiv:1608.01725*, 2016.

[38] M. Haase, M. Handy, et al. BlueTrack–Imperceptible Tracking of Bluetooth Devices. In *Ubicomp Poster Proceedings*, 2004.

[39] D. Holger. How 'Free' Wi-Fi Hotspots Can Track Your Location Even When You Aren't Connected, Nov 2018. URL https://www.pcworld.com/article/3315197/privacy/free-wi-fi-hotspots-can-track-your-location-even-when-you-arent-connected.html.

[40] B. Hong, S. Bae, and Y. Kim. GUTI Reallocation Demystified: Cellular Location Tracking with Changing Temporary Identifier. In *Symposium on Network and Distributed System Security (NDSS). ISOC*, 2018.

[41] T. Kohno, A. Broido, and K. C. Claffy. Remote Physical Device Fingerprinting. *IEEE Transactions on Dependable and Secure Computing*, 2(2):93–108, 2005.

[42] A. Korolova and V. Sharma. Cross-App Tracking via Nearby Bluetooth Low Energy Devices. In *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, pages 43–52. ACM, 2018.

[43] T. Liebig and A. U. K. Wagoum. Modelling Microscopic Pedestrian Mobility using Bluetooth. In *ICAART (2)*, pages 270–275, 2012.

[44] G. F. Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning*. Insecure, 2009.

[45] J. Martin, D. Rhame, R. Beverly, and J. McEachen. Correlating GSM and 802.11 Hardware Identifiers. In *IEEE Military Communications Conference*, 2013.

[46] J. Martin, E. Rye, and R. Beverly. Decomposition of MAC Address Structure for Granular Device Inference. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 78–88. ACM, 2016.

[47] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown. A Study of MAC Address Randomization in Mobile Devices and When it Fails. *Proceedings on Privacy Enhancing Technologies*, pages 365–383, 2017.

[48] C. Matte. *Wi-Fi Tracking: Fingerprinting Attacks and Counter-Measures*. PhD thesis, Université de Lyon, 2017.

[49] S. F. Mjølsnes and R. F. Olimid. Easy 4G/LTE IMSI catchers for Non-Programmers. In *International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, pages 235–246. Springer, 2017.

[50] A. Musa and J. Eriksson. Tracking Unmodified Smartphones using Wi-Fi Monitors. In *Proceedings of the 10th ACM conference on embedded network sensor systems*, pages 281–294. ACM, 2012.

[51] C. Neumann, O. Heen, and S. Onno. An Empirical Study of Passive 802.11 Device Fingerprinting. In *2012 32nd International Conference on Distributed Computing Systems Workshops*, pages 593–602. IEEE, 2012.

[52] Openspecs-Windows. [ms-cdp]: Connected devices platform protocol version 3. URL https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cdp.

[53] P. O'Hanlon, R. Borgaonkar, and L. Hirschi. Mobile Subscriber WiFi Privacy. In *Security and Privacy Workshops (SPW), 2017 IEEE*, pages 169–178. IEEE, 2017.

[54] C. Paget. Practical Cellphone Spying. *Def Con*, 18, 2010.

[55] R. Rajavelsamy, D. Das, and M. Choudhary. Privacy protection and mitigation of unauthorized tracking in 3GPP-WiFi interworking networks. In *Wireless Communications and Networking Conference (WCNC), 2018 IEEE*, pages 1–6. IEEE, 2018.

[56] D. W. Richardson, S. D. Gribble, and T. Kohno. The Limits of Automatic OS Fingerprint Generation. In *Proceedings of the 3rd ACM workshop on Artificial intelligence and security*, pages 24–34. ACM, 2010.

[57] E. C. Rye and R. Beverly. Sundials in the Shade: An Internet-Wide Perspective on ICMP Timestamps. In *International Conference on Passive and Active Network Measurement*, pages 82–98. Springer, 2019.

[58] P. Sapiezynski, A. Stopczynski, R. Gatej, and S. Lehmann. Tracking Human Mobility using wifi Signals. *PloS one*, 10 (7):e0130824, 2015.

[59] Z. Shamsi, A. Nandwani, D. Leonard, and D. Loguinov. Hershel: Single-packet OS Fingerprinting. In *ACM SIGMETRICS Performance Evaluation Review*, volume 42, pages 195–206. ACM, 2014.

[60] A. Soltani. Privacy Trade-Offs in Retail Tracking. *Tech@ FTC. URL https://wwwi.ftc.gov/news-events/blogs/techftc/2015/04/privacy-trade-offs-retai*, 2015.

[61] D. Strobel. IMSI catcher. *Chair for Communication Security, Ruhr-Universität Bochum*, 14, 2007.

[62] M. Stute, S. Narain, A. Mariotto, A. Heinrich, D. Kreitschmann, G. Noubir, and M. Hollick. A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link. In *USENIX Annual Technical Conference*, 2019.

[63] F. Van Den Broek, R. Verdult, and J. de Ruiter. Defeating IMSI Catchers. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, pages 340–351. ACM, 2015.

[64] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens. Why MAC Address Randomization is not Enough: An Analysis of Wi-Fi Network Discovery Mechanisms. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pages 413–424. ACM, 2016.

[65] M. Versichele, T. Neutens, M. Delafontaine, and N. Van de Weghe. The Use of Bluetooth for Analysing Spatiotemporal Dynamics of Human Movement at Mass Events: A Case Study of the Ghent Festivities. *Applied Geography*, 32(2): 208–220, 2012.

[66] Q. Xu, R. Zheng, W. Saad, and Z. Han. Device Fingerprinting in Wireless Networks: Challenges and Opportunities. *IEEE Communications Surveys & Tutorials*, 18(1):94–104, 2015.

A Histograms of sequence numbers captured in public locations