Guillaume Celosia* and Mathieu Cunche

Discontinued Privacy: Personal Data Leaks in Apple Bluetooth-Low-Energy Continuity Protocols

Abstract: Apple Continuity protocols are the underlying network component of *Apple Continuity* services which allow seamless nearby applications such as activity and file transfer, device pairing and sharing a network connection. Those protocols rely on Bluetooth Low Energy (BLE) to exchange information between devices: Apple Continuity messages are embedded in the payload of BLE advertisement packets that are periodically broadcasted by devices. Recently, Martin et al. identified [1] a number of privacy issues associated with Apple Continuity protocols; we show that this was just the tip of the iceberg and that Apple Continuity protocols leak a wide range of personal information.

In this work, we present a thorough reverse engineering of Apple Continuity protocols that we use to uncover a collection of privacy leaks. We introduce new artifacts, including identifiers, counters and battery levels, that can be used for passive tracking, and describe a novel active tracking attack based on Handoff messages. Beyond tracking issues, we shed light on severe privacy flaws. First, in addition to the trivial exposure of device characteristics and status, we found that HomeKit accessories betray human activities in a smarthome. Then, we demonstrate that AirDrop and Nearby Action protocols can be leveraged by passive observers to recover e-mail addresses and phone numbers of users. Finally, we exploit passive observations on the advertising traffic to infer Siri voice commands of a user.

Keywords: Bluetooth Low Energy; Privacy; Tracking; Activity inference; Inventory attacks; Perceptual hashing; Guesswork; Protocol.

DOI 10.2478/popets-2020-0003
Received 2019-05-31; revised 2019-09-15; accepted 2019-09-16.

***Corresponding Author: Guillaume Celosia:** Univ Lyon, INSA Lyon, Inria, CITI,
F-69621 Villeurbanne, France,
E-mail: guillaume.celosia@insa-lyon.fr
Mathieu Cunche: Univ Lyon, INSA Lyon, Inria, CITI,
F-69621 Villeurbanne, France,
E-mail: mathieu.cunche@insa-lyon.fr

1 Introduction

Smart devices interacting with each other are bringing new types of applications that simplify configuration procedures and enhance users experience. Those new applications include sending a file to a nearby device, transferring an activity to another device, network connection sharing, etc. Major vendors have developed protocols to enable those features: *Google Nearby* [2], *Microsoft Connected Devices Platform* (CDP) [3] and protocols used by *Apple Continuity* [4]. The family of protocols developed by Apple, called *Apple Continuity protocols*, can be found in all Apple products but also in devices from third-party companies[1]. Thus, Apple Continuity protocols are embedded in more than one billion active devices [7], including smartphones, laptops, earphones, smartwatches and smarthome appliances. Within those devices, Apple Continuity protocols enable a range of services such as activity transfer, remote printing and smarthome monitoring.

Apple Continuity protocols rely on Bluetooth Low Energy (BLE) for the transport of information over the air: messages of continuity protocols are carried by BLE advertisement packets that are broadcasted and thus made available to all nearby devices.

Wireless communications functionalities of smart devices can represent privacy threats for users. In particular, Wi-Fi and Bluetooth/BLE signals can be used for users tracking [8, 9] and to infer other private attributes [10–12]. To remedy to the tracking issue, the Bluetooth Core Specification version 4.0 introduced the LE Privacy feature [13, Vol 3, Part C, sec. 10.7] that defines the use of temporary and random link layer identifiers. Several works [14, 15] have highlighted privacy issues associated with BLE showing that devices can still be tracked despite LE Privacy provisions. Furthermore, several serious issues have been recently discovered [1] in Apple Continuity protocols, allowing an attacker to track a device based on passive and active attacks.

1 Apple certified vendors [5] and HomeKit accessories manufacturers [6].

In this paper, we pursue on the path started by Martin et al. [1] and present a collection of new privacy issues in Apple Continuity protocols. Based on a detailed reverse engineering of Apple Continuity protocols, we demonstrate how information exposed in cleartext in BLE advertisement packets can be used for tracking but also to reveal personal information and users activities. Our contributions are outlined as follows:

– We present an extensive reverse engineering of Apple Continuity protocols along with the format of corresponding messages (Section 3);
– We identify new fields of Apple Continuity messages leverageable for passive tracking (Section 4);
– We introduce a novel active attack that can be used to track a user and to link devices belonging to the same iCloud account (Section 5);
– We find that several fields expose characteristics of the device (model, OS version, color, etc.) and information on the device status that could be leveraged to infer the ongoing activity (Section 6);
– We show that messages broadcasted by HomeKit accessories include a `Global State Number` (GSN) that leaks the activity in a smarthome (Section 7);
– We discover that AirDrop and Nearby Action messages include hashed e-mail addresses and phone numbers, and we demonstrate how they can be recovered through a *guesswork* attack (Section 8);
– We found that Siri messages include a perceptual hash of voice commands and demonstrate how it can be exploited to infer a command (Section 9);
– We provide a discussion on the impact of the discovered attacks (Section 10);
– We discuss potential solutions to remedy to the identified issues (Section 11).

2 Background

2.1 BLE protocol

BLE is a 2.4 GHz ISM band radio communication standard that has been introduced in 2010 as part of the Bluetooth Core Specification version 4.0 [13, Vol 6]. BLE operates on 40 physical channels: 3 of those channels are dedicated to the discovery mechanism called *advertisement* while the 37 remaining channels are used for data transmissions.

The *advertisement* discovery mechanism allows BLE devices to advertise their presence and characteristics to nearby devices. It relies on the broadcasting of *advertisement packets* on the 3 advertisement channels of BLE. An advertisement packet is composed of a header and a payload. The payload contains an advertising device address and up to 31 bytes of data, organized in Advertisement Data (AD) structures that carry information about the device. An AD structure is composed of a 1-byte field indicating the length of the AD (excluding itself), followed by a 1-byte field specifying the type[2] of the AD and finally, a sequence of up to 29 bytes of data (see Fig. 1). One AD structure type, called `Manufacturer Specific Data` (code `0xFF`), is dedicated to the transport of data for custom applications defined by the manufacturer.

2.2 BLE privacy

While advertising, BLE devices are identified by a Bluetooth device address, a 48-bit identifier found within the Advertising Address (`AdvA`) field of the advertising payload. As part of the LE Privacy feature, BLE has introduced random addresses in addition to the globally unique MAC address [16]. Thus, there are 4 types of device addresses in BLE:

Public device address: The address uniquely allocated to the device by the manufacturer in accordance to the IEEE MAC address specifications [17, sec. 8.2].

Random Static device address: A randomly generated address that can be renewed after each power cycle and that shall not change during the use of the device.

Random Non-resolvable device address: A randomly generated address that can be renewed at any time.

Random Resolvable device address: An address generated using a random value and an Identity Resolution Key (IRK), that can only be resolved by devices knowing the IRK.

Furthermore, the Bluetooth Core Specification [18, Vol 3, Part C, App. A] recommends to renew Random Non-resolvable and Random Resolvable addresses at most every 15 minutes.

2.3 Continuity protocols

Continuity protocols are network mechanisms used to enable short range communications for services running on mobile devices and connected appliances. They constitute the network element of *Apple Continuity* ser-

2 https://www.bluetooth.com/specifications/assigned-numbers/generic-access-profile vices [4], and similar protocols have been developed by major vendors: *Google Nearby* [2] and *Microsoft CDP* [3]. Typical continuity services include activity transfer, multimedia content streaming, file sharing, device pairing, monitoring and control of connected home appliances to name a few.

Continuity protocols typically rely on wireless technologies such as Bluetooth/BLE and Wi-Fi to carry information between devices. As they are directly used by applications, such protocols can be seen as part of the Application Layer of the OSI model. However, they sometimes include features of other layers such as the Session and Transport Layers (sequence number and reconnection). Because they only involve device-to-device communications, the Network Layer is limited to a bare minimum. Both the Data Link and Physical Layer are handled by the underlying wireless protocol (e.g. Bluetooth, BLE or Wi-Fi).

3 Reverse engineering of Apple Continuity protocols

3.1 Methodology

The analysis presented in this paper required an in-depth understanding of Apple Continuity protocols for which most specifications are not public. Hence, we had to rely on additional sources of information namely BLE traffic traces, outputs of Apple debugging tools and disassembled binaries.

Although limited in details and quantity, there exist some official documents about Apple Continuity protocols and associated services. Such documents include the iOS security guide [19] and the HomeKit Accessory Protocol (HAP) specifications [20].

As Apple Continuity messages are embedded in BLE advertisement packets, capture and generation of those packets were at the core of our study. Capture and generation of advertisement packets were respectively done using a sniffer based on the *bluepy*[3] python library and a bash script based on *hcitool*[4]. In both cases, we used a CSR v4.0 Bluetooth USB dongle as transceiver. In order to isolate signals from a device, we placed the monitoring system next to it and we relied on the *Received Signal Strength Indicator* (RSSI) to filter out packets coming from other devices.

All the tests were performed in our laboratory against Apple devices owned by the authors and their institutions. Those tests involved a range of Apple devices including iPhone smartphones, iPad tablets, AirPods earphones, MacBook laptops, Apple Watch smartwatches that were running across different versions of Apple iOS, macOS and watchOS operating systems. We considered products from partner companies too that are compatible with the Apple ecosystem such as the *Eve Motion* sensor and the *Osram Smart+* lightbulb.

We also relied on debugging tools provided by Apple, *PacketLogger* and *PacketDecoder*. Those tools, available on the Apple Developer[5] website, provide information on the content of received BLE frames, as well as on steps in the corresponding protocol.

Finally, we had recourse to disassembling to gain additional knowledge on Apple Continuity protocols. We used the *Hopper*[6] software to disassemble several macOS binaries such as *PacketLogger*, *PacketDecoder*, *CoreSpeech*, *HomeKit Accessory Simulator* and *sharingd*. Based on the disassembled codes, we were able to identify the precise format of continuity messages, and the signification of some codes (device model, activity level, action type, etc.). We were also able to analyze the implementation of some functionalities such as the perceptual hash of Siri voice commands (see Section 9).

3.2 General features of Apple Continuity messages

Apple Continuity protocols use the `Manufacturer Specific Data` AD structure of BLE advertisement packets to transmit data to nearby devices. This AD starts with 2 bytes, indicating the company identifier of Apple (code `0x004C`), followed by one or several Apple Continuity messages accounting for up to 27 bytes. Those Apple Continuity messages follow a Type-Length-Value (TLV) format outlined in Fig. 1 (*Type* and *Length* are encoded on one byte each). We have identified a total of 12 different types of Apple Continuity messages (see Table 3).

Note that, for a given device, the Apple Continuity messages included in BLE advertisement packets can vary over time depending on its status and

3 https://ianharvey.github.io/bluepy-doc/scanner.html
4 https://github.com/pauloborges/bluez/blob/master/tools/hcitool.c

5 https://developer.apple.com/
6 https://www.hopperapp.com/

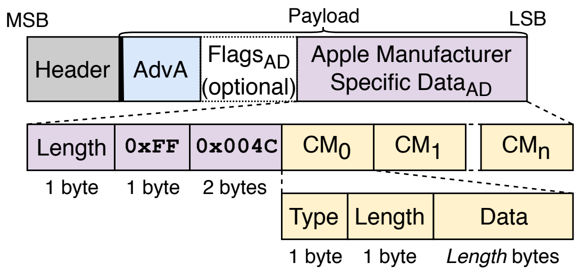

Fig. 1. Structure of a BLE advertisement packet used to carry data of Apple Continuity protocols. The data are stored in a `Manufacturer Specific Data` AD structure (`0xFF`) which starts with the company identifier of Apple (`0x004C`), followed by one or several continuity messages (CM) presented as a Type-Length-Value (TLV) format. *Flags* is an optional AD structure that is not specific to Apple Continuity protocols.

activity. For instance, advertisement packets generated by an iPhone will always include a Nearby Info message, and will occasionally include a Handoff message when a compatible application is running.

3.2.1 Content protection

The Apple Continuity messages are never encrypted as a whole and their content is thus exposed in clear as part of advertisement packets. However, Proximity Pairing and Handoff messages include an encrypted payload (through AES-128 and 256 in ECB mode). Other elements of data are not transmitted in clear but are hashed using SHA-256 then truncated. This is the case of Wi-Fi SSIDs, e-mail addresses and phone numbers found in AirDrop [19, p. 45] and some Nearby Action messages (see Section 8). Finally, Handoff, Nearby Action and Nearby Info messages are authenticated through an `Auth Tag` that is computed using AES-256 in GCM mode [19, p. 42].

3.2.2 Use of random device address

We found that, in most cases, advertisement packets including Apple Continuity messages are using Random Non-resolvable or Random Resolvable addresses, and that the Bluetooth Core Specification recommendation on the maximum lifetime of 15 minutes is enforced.

However, we identified several exceptions: AirPrint and AirPlay messages are using Public addresses, thus exposing the device MAC address; AirDrop and "Hey Siri" messages use a Random Non-resolvable device address that never changes until the user turns off then on the Bluetooth interface of his device.

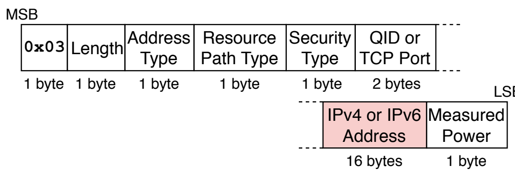

Fig. 2. Format of the AirPrint message.

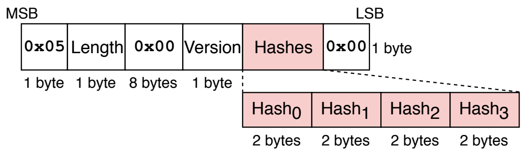

Fig. 3. Format of the AirDrop message.

3.3 Apple Continuity protocols

In this section, we give an overview of Apple Continuity messages found in BLE advertisement packets. For each message type, we present the associated service, the data format as well as the basics of the corresponding protocol (see Table 3 for a summary of the Apple Continuity protocol messages).

AirPrint
AirPrint is a feature included in Apple devices to discover compatible[7] printers and print documents via a wireless network. AirPrint messages are transmitted each time the user tries to print a document through AirPrint. Those messages (see Fig. 2) include fields describing the remote printer, including its complete IP address (IPv4 or IPv6) and port that can be leveraged as a first step toward a more elaborated attack [22].

AirDrop
AirDrop is a service which enables the transfer of files between Apple devices over Wi-Fi and Bluetooth. AirDrop messages are transmitted each time the user attempts an AirDrop transfer. Those messages (see Fig. 3) are composed of several 2-byte long fields containing hashed identifiers. Those hashes are the 16 most significant bits (MSB) of the SHA-256 digests of e-mail addresses and phone numbers configured in the user's iCloud account (see Section 8).

HomeKit
HomeKit is a framework for the monitoring and control of connected home appliances supporting the HomeKit Accessory Protocol (HAP) [20]. HomeKit messages are constantly transmitted by powered HomeKit devices

7 This feature is built in most popular printer models, such as the ones listed on the Apple website [21].

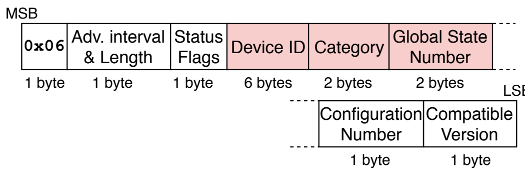

Fig. 4. Format of the HomeKit message.

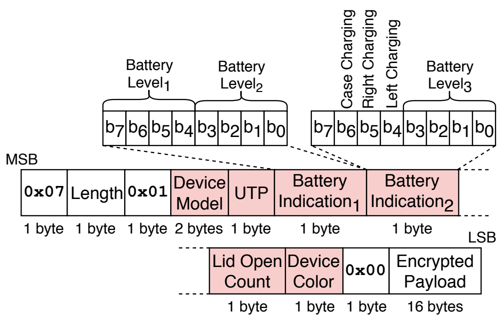

Fig. 5. Format of the Proximity Pairing message.

and include fields (see Fig. 4) identifying the device and its category, as well as a `Global State Number` (GSN) that is incremented each time the device state changes.

Proximity Pairing

Proximity Pairing is a feature to facilitate the pairing of audio devices with an iPhone or an iPad [23]. Proximity Pairing messages are constantly transmitted by active Apple audio devices (earphones and headphones). Those messages (see Fig. 5) include an encrypted payload concatenated to fields describing the device attributes (model and color), its position (in ear/case) via the `UTP` field as well as its current status: battery level, charging status and lid open counter[8].

"Hey Siri"

Siri is the Apple virtual assistant that uses voice commands issued by a user to answer questions and perform actions. "Hey Siri" messages are broadcasted each time the user submits a command via voice activation. Those messages (see Fig. 6) include fields associated with the expressed Siri voice command: a `Perceptual Hash` (see Section 9) and indicators of `Signal-to-Noise Ratio` (SNR) and `Confidence`. Those messages also include a field describing the `Device Class`.

AirPlay

AirPlay allows wireless streaming of multimedia content between compatible devices. AirPlay messages are

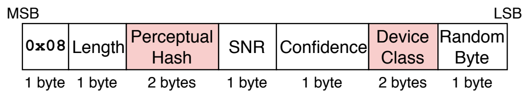

Fig. 6. Format of the "Hey Siri" message.

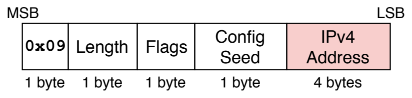

Fig. 7. Format of the AirPlay message.

transmitted each time the user tries to initiate the streaming of a content via AirPlay. Similarly to AirPrint, AirPlay messages (see Fig. 7) include the complete IP address of the AirPlay target in cleartext that could be leveraged by an attacker in the context of network intelligence gathering [24].

Magic Switch

The Magic Switch protocol is undocumented but appears to be related to the watchOS operating system. As observed by [1], only Watches send those messages. Magic Switch messages are transmitted when the Watch has lost the Bluetooth connection to its paired iPhone and when its screen is on. Those messages (see Fig. 8) include a 2-byte `Data` field that can expose the user to tracking (see Section 4) followed by a `Confidence on Wrist` field that appears to be an indicator that the watch is worn (see Table 9 in Appendix A).

Handoff

Apple Handoff [25] allows activities to be transferred between devices associated with the same iCloud account. Handoff messages are transmitted each time the user interacts – i.e. opens, runs or closes – with a Handoff-enabled application [26] (Mail, Safari, Maps, Contact, etc.). Those messages (see Fig. 9) include an `Initialization Vector` (IV), a payload encrypted with AES-256 in ECB mode and an `Auth Tag` generated with AES-256 in GCM mode [19, p. 42]. As reported by [1], we observed that the IV is not random and is incremented every time the payload changes.

Instant Hotspot

Instant Hotspot is a feature to share the cellular connectivity of an iPhone or an iPad over Wi-Fi with nearby devices associated with the same iCloud account. In-

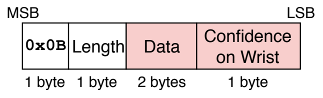

Fig. 8. Format of the Magic Switch message.

8 A similar 1-byte lid close counter is suspected to be also included and used the same way the lid open counter is used.

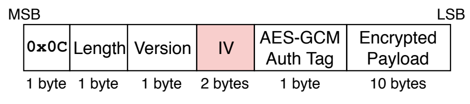

Fig. 9. Format of the Handoff message.

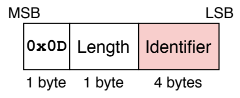

Fig. 10. Format of the Tethering Target Presence message (Instant Hotspot).

stant Hotspot relies on two types of messages: Tethering Target Presence and Tethering Source Presence.

Tethering Target Presence messages are transmitted each time the user does a Wi-Fi scan to discover surrounding Wi-Fi networks. In practice, a scan is performed when the user opens or navigates in the Wi-Fi and network settings menu. Those messages (see Fig. 10) include a 4-byte `Identifier` field that is generated from a *Destination Signaling Identifier* (DSID) tied to the iCloud account [19, p. 45]. As observed by [1], the DSID is rotated only once every 24 hours raising tracking concerns (see Section 4).

Tethering Source Presence messages are transmitted by devices capable of sharing a cellular connection in response to Tethering Target Presence messages coming from devices associated with the same iCloud account. Those messages (see Fig. 11) include fields representing the current state of the device: `Battery Level`, nature of cellular connection (`Network Type`) and cellular `Signal Strength` (as reported by [1]).

Nearby
Although it is not documented, the Nearby protocol appears to be used to inform nearby devices about the presence and state of a device. This protocol uses two types of messages: Nearby Action and Nearby Info.

Nearby Action messages are transmitted each time the user takes an action covered by the Nearby protocol such as setting up a speaker, sharing a Wi-Fi password or answering a call (see Table 11 in Appendix A for an extended list of Nearby Action types). Those messages include a field describing the `Action Type` as well as a

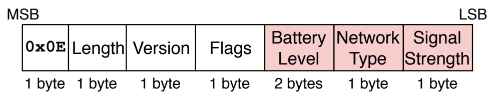

Fig. 11. Format of the Tethering Source Presence message (Instant Hotspot).

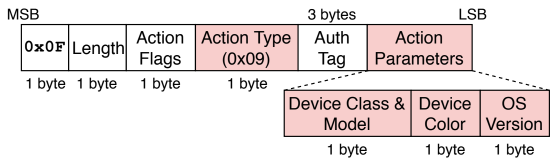

Fig. 12. Format of the Nearby Action (iOS Setup) message.

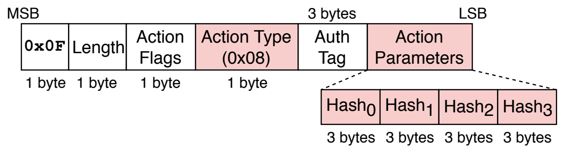

Fig. 13. Format of the Nearby Action (Wi-Fi Password) message.

set of `Action Parameters`. In the cases of the iOS Setup (see Fig. 12) and Wi-Fi Password (see Fig. 13) actions, parameters include characteristics of the device such as its class, model, color and OS version but also hashed e-mail addresses and phone numbers of users configured into their iCloud accounts (see Section 8).

Nearby Info messages are continuously transmitted by iPhones, iPads, MacBooks and Watches regardless of the device status (locked or active). Those messages (see Fig. 14) include an `Auth Tag` as well as an `Activity Level` field representing the current state of the device.

4 Passive tracking

The use of random identifiers is a countermeasure against tracking. However, the content of frames can be used to fingerprint a device or to link distinct addresses of a device [8, 27]. This type of issue has been recently demonstrated [1, 14, 15] with BLE advertisement packets which contain static identifiers and counters. In this section, we complement previous works by presenting a number of new artifacts contained in Apple Continuity messages that can be leveraged for passive tracking.

4.1 Identifiers and counters

Public MAC address: AirPrint and AirPlay messages are transmitted by devices using their public MAC ad-

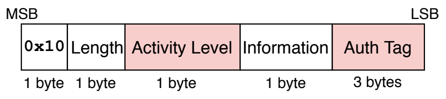

Fig. 14. Format of the Nearby Info message.

dresses. Similarly, we observed that a MacBook continuing an activity via Handoff broadcasts Nearby Info messages with its public MAC address.

Extended lifetime of Random Resolvable addresses: When a device is in power saving mode, it can keep the same Random Resolvable address for a duration that exceeds the Bluetooth Core Specification recommended maximum duration of 15 minutes. Some Random Resolvable addresses that carry Nearby Info messages have been observed for more than 4 days.

Stable Random Non-resolvable addresses: Advertisement packets carrying AirDrop and "Hey Siri" messages use Random Non-resolvable addresses that are not changed over time. We observed the same values for more than 7 days.

Bad synchronization in pseudonym changes: As reported in [1, 14], we observed that the change of the `Auth Tag` in Nearby Info messages is not completely synchronized with the change of the device address: for a short duration after the address change, the old `Auth Tag` is used with the new device address.

Stable identifier in AirDrop and Nearby Action (Wi-Fi Password) messages: AirDrop and Nearby Action (Wi-Fi Password) messages include hashed e-mail addresses and phone numbers[9] that constitute stable identifiers.

Stable identifier in Tethering Target Presence messages: Tethering Target Presence messages carry a 4-byte long `Identifier` tied to the iCloud account that is rotated only once every 24 hours and thus constitutes a stable identifier.

Stable data in Magic Switch messages: The 2-byte long `Data` field found in Magic Switch messages appears to be stable over time. We observed that such a field can remain unchanged during more than 30 minutes.

Non-reset IV: The IV of Handoff messages is not reset when the random device address changes and thus can be leveraged to link consecutive addresses.

4.2 AirPods battery levels and Lid Open Count

Proximity Pairing messages broadcasted by AirPods expose information about battery levels and `Lid Open Count` (see Section 3.3) that could be exploited for tracking [28] a set of AirPods. Those messages include three battery level indicators (one for each earphone and one for the case) which are presented with a resolution of 10 levels. The messages also include a charging status, as well as a `Lid Open Count` field that is a 1-byte counter incremented each time the case lid is opened. Those information are only broadcasted when the AirPods are outside the case and the battery level of the case is only exposed when the lid is open.

Proximity Pairing messages broadcasted by a set of AirPods have been recorded during a day: the AirPods were used intermittently and put back in their case after each usage session. Fig. 15 presents the evolution of the artifacts exposed by the set of AirPods during this day. Throughout the day, the battery level of the case decreases and the `Lid Open Count` increases. Battery levels of both AirPods earphones decrease while in use and are brought back to 100% by being stored in the case. Furthermore, it appears that the battery level is quite identical for both earphones.

Alone, the `Lid Open Count` holds a potential for tracking: it is a counter taking 16 values which can be used to identify AirPods during a session of use (since it is stable) and between two sessions (it is incremented by 1). During a session, the battery levels of the earphones take one of 10 values that evolves at a slower rate than the address change. Together, the `Lid Open Count` and the battery level of earphones can take a total of $16 \times 10 = 160$ different values.

Furthermore, the global energy of the system could be used to fingerprint the system. By modeling the energy transfer between the case and the AirPods, it would be possible to create a global energy fingerprint that could be leveraged to track a set of AirPods between usage sessions. This possibility is left for future work.

5 Active tracking/linking

Some Apple Continuity protocols are interactive and, as such, the reception of a message may trigger a reaction under the form of a new message emission. This mechanism can be used to mount active tracking attacks in which the attacker will replay messages to force a device to reveal its presence and identity.

In most cases, a device will only react to messages coming from a known source: a device it has been previously paired with or a device belonging to the same

9 As reported by [1], the `Hash3` field of Nearby Action (Wi-Fi) Password messages contains the first 3 bytes of the SHA-256 hash of the SSID the client device is attempting to join.

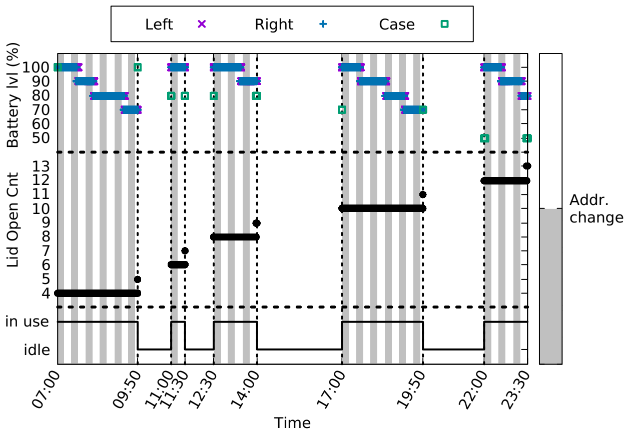

Fig. 15. Evolution of AirPods battery levels and `Lid Open Count` from Proximity Pairing messages. Different device addresses are represented by grey/white areas. Those data represent a day during which AirPods are used intermittently and the case is never charged. Battery levels and `Lid Open Count` remain stable during periods longer than the lifetime of the random device addresses.

iCloud account. This feature implies that some messages will only trigger a reaction from specific devices.

This kind of replay attack against an Apple Continuity protocol was first introduced by Martin et al. who showed [1] that replaying Tethering Target Presence messages will trigger a reaction from a source associated with the same iCloud account. Due to the rotating DSID, a Tethering Target Presence message can only be replayed for at most 24 hours.

5.1 Replay of corrupted Handoff messages

We discovered a new replay attack based on Handoff messages that are used to enable activity transfer between devices (see Section 3.3).

Handoff messages are far more common than Instant Hotspot ones and, in the Handoff protocol, source and destination roles can be endorsed by iPhones, iPads, MacBooks and Watches. The Handoff replay attack can thus target a wider range of devices than the Instant Hotspot one.

The attack relies on resynchronization mechanism in which a device initiates a new synchronization procedure through a connection request when it receives a corrupted Handoff message: an authentication tag inconsistent with the payload and the IV, or an IV not greater than the last one received. This behavior is implemented in the *sharingd* binary and we have confirmed it on our devices (see Table 1).

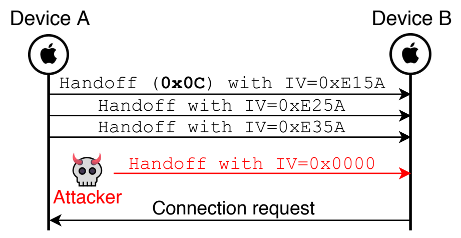

Fig. 16. Replay of Handoff messages with a bad IV. By replaying a Handoff message including an IV less than the last one received (`0xE35A`), the attacker forces *Device B* to initiate a new synchronization procedure with *Device A* triggering a connection request.

This mechanism can be leveraged as follows: 1) the attacker captures a Handoff message emitted by a *Device A*, 2) the attacker later replays this message with a modified IV field (for instance, by setting it to zero) then 3) a second *Device B* (associated with the same iCloud account) will respond to this message by initiating a connection thus revealing its presence and its current device address (see Fig. 16). Using this attack, the attacker is able to detect the presence and the current address of any Handoff-compatible device in range associated with the same iCloud account.

In principle, our Handoff-based replay attack is similar to the Instant Hotspot replay attack introduced by Martin et al. in [1]. However, because our attack relies on Handoff messages rather than Instant Hotspot ones, it is far easier to exploit. First, Handoff messages are emitted whenever the user interacts with one of the several compatible applications; whereas Instant Hotspot messages are only sent when a device searches for Wi-Fi connectivity. Second, Handoff messages affect a much wider range of devices than Instant Hotspot ones: the source of Instant Hotspot message is necessarily an iPhone, an iPad or a MacBook; and the destination can only be a device with cellular connectivity, e.g. an iPhone or an iPad. On the other hand, Handoff messages are emitted by all types of Apple devices (excluding AirPods) and thus can be received by all types of devices with the exception of AirPods.

5.2 Experimental evaluation of replay attacks

We evaluated the success of both Handoff and Instant Hotspot [1] replay attacks on a set of Apple devices. Each device was first linked to an iCloud account. We then captured a message (Handoff or Instant Hotspot) emitted by a secondary device associated to the same iCloud account. The message was replayed in proximity of the evaluated device and we monitored the wireless channel for a response. Results of this evaluation are presented in Table 1. We can observe that the Handoff-based attack affects all devices with the exception of the AirPods, while the Instant Hotspot attack only affects devices capable of sharing cellular connectivity such as iPhones and *Cellular* capable iPads [29].

5.3 Device linking

In addition to tracking, those replay attacks can be leveraged to link devices associated with the same iCloud account and thus belonging to the same user. Indeed, both Instant Hotspot and Handoff replay attacks trigger a reaction from other devices associated with the same iCloud account. An attacker having captured messages emitted by a particular device can then replay those messages later in order to detect and identify other devices belonging to the same user.

As each device potentially exposes different types of information, device linking increases the amount of information that can be gathered on an individual. Furthermore, device linking could be exploited to identify points of interest of the user by detecting devices that were left at home or at the office.

6 Exposed device status and characteristics

Some Apple Continuity messages include fields providing information on the status of a device (idle, screen on, playing video, etc.) or its characteristics. The following characteristics can be found in continuity messages:

Class and category are exposed by the `Category` field of HomeKit messages (see Table 4) and the `Device Class` of "Hey Siri" and Nearby Action messages (see Table 8 and Table 12).

Model is exposed by the `Device Model` fields of Proximity Pairing and Nearby Action messages (see Table 5 and Table 13).

Color is exposed by the `Device Color` field of Proximity Pairing and Nearby Action messages (see Table 7 and Table 14).

OS version is exposed by the `OS Version` field of Nearby Action messages (see Table 15).

Furthermore, the message type can be leveraged to fingerprint the device. For instance, Proximity Pairing are only emitted by earphones and headsets, while Magic Switch messages are only emitted by watches.

Similarly, elements describing the state of the device can be found in continuity messages:

Position (in ear/case) of the AirPods is exposed by the `UTP` field of Proximity Pairing messages (see Table 6).

Activity of the device is exposed by the `Activity Level` field of Nearby Info messages (see Table 16) and by the types of broadcasted continuity messages (e.g. AirPlay messages betray multimedia content streaming, AirPrint messages are involved in printing tasks, etc.).

State changes are exposed by the `Lid Open Count` field of Proximity Pairing and `Global State Number` field of HomeKit messages.

Cellular connectivity is exposed by the `Network Type` (see Table 10) and `Signal Strength` fields of Tethering Source Presence (Instant Hotspot) messages.

Battery status is exposed by the `Battery Indication` fields of Proximity Pairing messages and the `Battery Level` field of Tethering Source Presence (Instant Hotspot) messages.

6.1 Applications

Inventory attacks: The information provided on the nature of the device can be used in inventory attacks [30] where the attacker leverages the list of owned devices to infer information on a subject such as wealth or medical condition. To the best of our knowledge, Apple Continuity protocols are not yet included in medical devices. However, the ownership of Apple devices has been shown [31] to be a reliable indicator of wealth.

Visual identification: Physical characteristics of the device such as color, model and class can be used for visual identification. It could serve to establish a link [32] between a visual identity (a person holding a device) and its radio identity (the device address found in BLE advertisement packets).

Event correlation: By exposing status changes in real-time, it is possible to correlate the identity of a device with some triggered events: sending a message to an e-mail address will trigger a state change associated with a device address.

Activity monitoring: Exposure of a detailed description of device status has the potential for exposing in real-time the activity of a user on its device but also

Table 1. Experimental evaluation of replay attacks based on Handoff and Instant Hotspot messages against different Apple devices.

Apple device	Model	OS version	Vulnerable to... ...corrupted Handoff messages	...Instant Hotspot messages
AirPods (2nd generation)	A1602, A2031/2	1A691		
iPad (5th generation)*	A1822	iOS 12.3.1	✓	
iPad Mini 3*	A1599	iOS 11.4	✓	
iPhone 6	A1586	iOS 11.4.1	✓	✓
iPhone 8	A1905	iOS 12.4	✓	✓
MacBook Air (13", 2014)	A1466	macOS 10.12.3	✓	
MacBook Pro (13", 2015)	A1502	macOS 10.13.6	✓	
Watch Series 2	A1757	watchOS 5.0.1	✓	
Watch Series 3	A1858	watchOS 5.1.3	✓	

* Only supports Wi-Fi (i.e. not *Cellular* capable).

in its smart environment (see Section 7). This information could be leveraged to learn the habits of users or to implement activity-based advertising systems [33].

7 Leaking smarthome activity

HomeKit messages emitted by appliances contain information that can be leveraged to infer the activity in a smarthome. Those messages include a `Global State Number` (GSN) that is used to keep track of device state changes [20, sec. 7.4.6.3] and that is incremented each time the state is modified.

For a number of smarthome devices, a state change is the consequence of a user command or a reaction to an environment modification induced by the user. Therefore, the change of state in smarthome appliances can be an indicator of the presence and activity of a user [34].

We illustrate this inference of activity through an experiment conducted in our lab. We considered two HomeKit accessories that transmit HomeKit messages over BLE: a motion sensor (*Eve Motion*) and a lightbulb (*Osram Smart+*). The devices were installed in the office of one of the authors and the HomeKit messages transmitted over BLE were recorded by a sniffing device. The GSN was extracted from the captured BLE frames and translated into a temporal sequence of state changes for each device. The corresponding traces are shown in Fig. 17, in parallel with the time of presence and activity of the occupant.

It is possible to leverage the GSN to get insight on the activity in the office. First, any movement in the field of the motion sensor will induce a change of GSN, thus revealing activity and presence in the room. Similarly, turning on or off the lightbulb will be reflected by a change of GSN. Note that, manually turning on/off the light using a physical switch also triggers a GSN change.

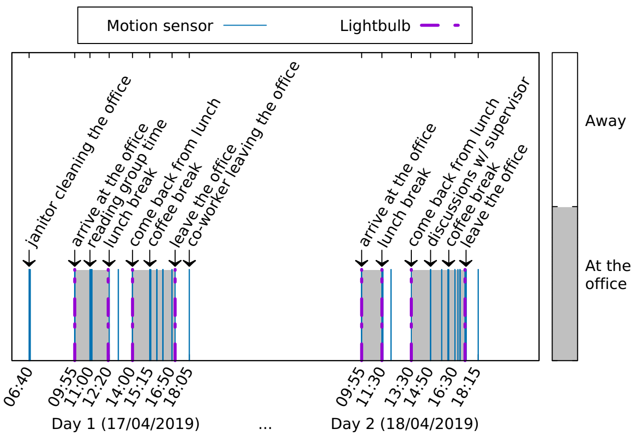

Fig. 17. Representation of the GSN changes extracted from HomeKit messages during two days. For both the motion sensor and lightbulb, GSN changes can be leveraged as an indicator of the presence and activity of a user.

As shown on Fig. 17, GSN of the motion sensor is a good indicator of presence in the office, while the data provided by the lightbulb reveal the boundaries of the office work activity. This attack does not allow the direct inference of a specific activity, rather, it exposes a coarse grain information: moving in the room, turning off a light. HomeKit messages are broadcasted by various types of devices including thermostat, door sensor and air quality monitor (see Table 4). Applying this attack to the whole range of HomeKit-enabled devices has the potential to reveal detailed information on smarthome activities [34].

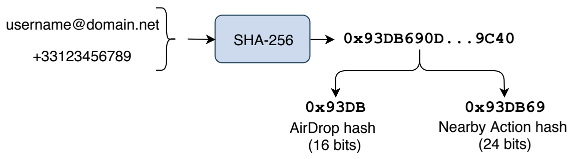

Fig. 18. Description of the process used to compute hashed identifiers found in AirDrop and Nearby Action messages. The identifier (an e-mail address or a phone number) is first hashed with SHA-256 then truncated to keep only the 16 and 24 MSB respectively for AirDrop and Nearby Action.

8 Leaked e-mail addresses and phone numbers

AirDrop and Nearby Action messages can include hashed identifiers such as e-mail addresses, phone numbers and SSIDs (see Section 3.3). Although those identifiers are not directly exposed, their values can be recovered through *guesswork* attacks [35, 36].

By analyzing the *sharingd* binary, we found that the identifiers are hashed using the SHA-256 function (without any salt) and are then truncated to 16 or 24 bits respectively for AirDrop and Nearby Action messages (see Fig. 18).

We confirmed the previous information through experiments with a device linked to an iCloud account on which two e-mail addresses and one phone number were configured. After triggering the transmission of AirDrop and Nearby Action messages, corresponding hashes were extracted from captured messages. Those hashes were matching with the 16 (respectively 24) MSB of the SHA-256 digests of the identifiers.

8.1 Recovering hashed identifiers

As a cryptographic hash function, SHA-256 is resistant to pre-image attacks. In other words, given a digest d, finding x such that SHA-256$(x) = d$ is computationally infeasible. However, when the set from which the values are drawn is small enough, re-identification through a brute force attack may become practical. Brute force attack for re-identification is also called *guesswork* and has been successfully employed against hashed e-mails and other digital identifiers [35–37].

Let us consider a hash function $h(.)$ that produces digests of length l and a digest d that has been obtained by hashing the value $\bar{x}$, i.e. $h(\bar{x}) = d$. The objective of the guesswork is to find all the pre-images of d in a set S of possible values. A guesswork attack will proceed by hashing all the elements in S and returning any element $s \in S$ such that $h(s) = d$. In our case, $h(.) = \text{MSB}_k(\text{SHA-256}(.))$, where $\text{MSB}_k(.)$ represents the function returning the k MSB of the input.

The cost in term of time of the guesswork can simply be expressed as: $c = m/z$, where m is the size of S (the number of element to be tested) and z is the hashing speed of the attacker.

Another parameter to consider is the potential false positives returned by the guesswork: value x such as $h(x) = d$ but $x \neq \bar{x}$. Indeed, m, the size of the set S, is potentially larger than the number of hash values n; thus it is likely that a given hash can have several pre-images in the set S. This case where $m > n$ falls under the *many-to-one* model described in [36], and the average number of elements of S matching a given digest is $r = m/n$. In the general case, and assuming that the identifier used to produce the hash is included in the set ($\bar{x} \in S$), the average number of identifiers returned by the guesswork [36] is the following:

$$r = \begin{cases} m/n & \text{if } m > n \\ 1 & \text{otherwise} \end{cases}$$

Since the attacker is only interested in the value $\bar{x}$, the identifier associated with the iCloud account, the number r should be kept as small as possible.

8.1.1 A posteriori confirmation

In the case where the guesswork returns several identifiers, the attacker will need additional information to narrow down the one used by the iCloud account. One possible approach to perform this confirmation is to exploit the device status information leaked by Nearby Info messages through an event correlation attack (see Section 6.1). The attacker can send messages to each identifier while monitoring for a device that goes from idle (code `0x03`) to active (code `0x07`). The attacker can also rely on visual observations: the target will pick up the phone or read the message.

8.2 Identifier sets for the guesswork

The guesswork attack requires a set of elements, S, that will be tested against a hashed value. As previously discussed, the size of this set will have an impact on efficiency of the guesswork (cost and number of false positives). Thus, the set S should be as small as possible.

Below, we discuss the potential approaches available to the attacker to build such a set of identifiers.

8.2.1 Phone numbers

Phone numbers follow a general format that corresponds to 8.11×10^{14} possible values [37]. This space can be significantly reduced if the attacker has additional information on the target such as the region in which the phone number is registered. This set can be further reduced by selecting a subdivision of the geographical regions or number ranges corresponding to specific operators. In particular, the attacker could focus on phone numbers of mobile operators as iCloud are usually configured with a mobile phone number. A resourceful attacker can also have access to a database of all registered phone numbers in a region.

In order to evaluate a potential attack, we considered several sets that an attacker could use: 1) all possible values following the number format of a region, 2) all possible values following the mobile number format of a region and 3) all registered mobile/landline numbers in the region. In the following, we selected France as region.

8.2.2 E-mail addresses

E-mail addresses can be up to 307 characters[10] long [36] corresponding to 2^{1666} different values. Thus, it is not realistic to envision an exhaustive search over all the possible values, and the attacker will have to rely on dictionary of practical size for the attack. To build such a dictionary, an attacker could either obtain a list of existing e-mails or try to generate e-mail addresses by assembling names and domains. For the sake of simplicity, we will focus on the first case (list of existing e-mails), and point the interested readers to existing works on synthesis of e-mail addresses and identifiers [36].

We considered several lists that could be used by an attacker: 1) all existing e-mail addresses worldwide, 2) addresses found in leaked database [38], 3) addresses of Gmail users, 4) addresses of university department and 5) addresses of PETS 2017 participants.

10 54 characters for the username and 253 characters for the domain.

8.3 Simulation results

The outcome of the hypothetical attacks for different sets of phone numbers and e-mail addresses are summarized in Table 2. We assumed that the attacker is able to test identifiers at the rate of 2000kH/s[11]

The attack cost appears to be reasonable and practical in every cases, as the worst-case scenario (all the existing e-mail addresses) can be tested within an hour.

The number of matching values ranges from 1 up to several thousands and depends on the size of the digest. For a 16-bit long digest of AirDrop, the list of values returned by the attack can contain several thousands of identifiers. However, with the 24 bits of the Nearby Action, the set is much smaller and can be, in some cases, reduced to a single element. When the set of potential value is small enough (several thousands of identifiers) the hashed identifier can be recovered without ambiguity in a matter of seconds.

9 Voice Assistant commands

When receiving voice commands, a Siri-enabled device will emit "Hey Siri" messages (see Section 3.3) including a digest of the command under the form of a *perceptual hash*. In the following, we discuss how this data can be leveraged to gain knowledge on spoken commands.

9.1 Perceptual hash

Commonly used for content identification [39], perceptual hashing is a technique to compute the digest of signal such as an image or a sound. A key property of perceptual hashing is that two perceptually similar signals $x \sim x'$ should produce digests at a close distance, $d(h(x), h(x')) < \varepsilon$, where ε is a threshold representing the robustness of the hashing algorithm [40].

For audio signals, one state of the art technique is the *Philips Robust Hash* (PRH) [39]. This algorithm produces a 32-bit long digest of an audio signal and works as follows. The signal is decomposed in overlapping frames and the following process is applied to each frame: switching to the frequency domain using a Fast Fourier Transform (FFT), decomposition of the signal

11 This hashing speed has been observed using *hashcat* (v5.1.0-951-g6caa786) in straight (dictionary) attack mode running on an Intel® Core™ i7-5600U CPU @ 2.60 GHz.

Table 2. Evaluation of hypothetical guesswork attacks on hashed phone numbers and e-mail addresses. For each set of identifiers, the corresponding size is used to evaluate the cost of the attack and the average number of values that will be returned as matching the digest (assuming the set includes the original identifier). Hashing speed of the attacker is assumed to be 2000kH/s.

Set type	Set description	Set size	Guesswork time	Avg. # of matching identifiers	
				16 bits (AirDrop)	24 bits (Nearby Action)
Phone numbers	France	10^9	500s	15259	60
	France, mobile	2×10^8	100s	3052	12
	France, registered mobile	74.8×10^6	37s	1141	4
	France, registered landline	38.4×10^6	19s	586	2
E-mail addresses	All existing addresses	4.7×10^9	1h	71716	280
	Leaked database [38]	773×10^6	387 s	11795	46
	Gmail users	1.5×10^9	750 s	22888	89
	University Dept.	7000	< 1	1	1
	PETS 2017 participants	240	< 1 s	1	1

into 33 non-overlapping bands and computation of the energy in each band. It then generates a 32-bit digest in which the bits depend on the energy variation between consecutive bands and frames. Interested readers can refer to [39] for a detailed presentation of the algorithm.

9.2 Observations on Siri's perceptual hash

The perceptual hash included in "Hey Siri" messages is 16-bit long, so twice as short as the output of the *standard* PRH algorithm. By analyzing the *CoreSpeech* binary, we identified the function in charge of computing the perceptual hash. This function, called `pHash(.)`, takes as input an audio signal and returns a 16-bit integer. We were unable to fully reverse engineer the function, however we were able to identify several characteristics: it uses a FFT, it computes energy in the frequency domain and bits of the hashes are computed based on the differences between consecutive elements of a buffer. All those features suggest that this code implements a variant of the PRH algorithm.

We analyzed the stability of the hashes among different users using an experiment involving two male users. We found that the same commands produce significantly different hashes when spoken by two different users (see Table 17 in Appendix A). This suggests that the perceptual hash depends both on the command and the user. In fact, it seems that features such as the speaking speed, voice tone as well as the pronunciation have an impact on the generated digest.

9.3 Exploiting Siri's perceptual hash

As noted by Knospe [40], the compactness of the digest prevents the reconstruction of the original audio signal. However, because it is a robust and identifying representation of an audio signal, the digest could still reveal information about the voice command. In the following, we show how perceptual hashes can be leveraged by a passive attacker to infer commands spoken by the user.

9.3.1 Dictionary attack on perceptual hashes

Perceptual hashing implies that a given command should produce the same output modulo some small variations. Therefore, knowing the digest of a given command, it would be possible for an attacker to infer the command issued to the voice assistant.

We assume that the attacker has captured a perceptual hash $\bar{y}$ that has been produced when the target spoke the command $\bar{x}$ to the voice assistant, i.e. $\bar{y} = h(\bar{x})$. The aim of the attacker is to identify the command $\bar{x}$ based on the captured hash $\bar{y}$. We further assume that the attacker has a dictionary that contains a list of commands and their corresponding digest: $\mathcal{D} = \{(x_i, y_i)\}_{0 \leq i < n}$, where y_i is the digest of the command x_i and n is the size of the dictionary.

Using the dictionary $\mathcal{D}$, the attacker can perform a custom dictionary attack by finding the command $x_k \in \mathcal{D}$ such that $y_k = h(x_k)$, and y_k is close enough to $\bar{y}$: $k = \text{Argmin}(d(y_i, \bar{y}))_{0 \leq i < n}$ and $d(y_k, \bar{y}) \leq \varepsilon$. This attack proceeds as described in Algorithm 1. It takes as input the intercepted hash and the dictionary, and returns a command of the dictionary or nothing if it has not found a close enough hash.

```
**Input:** $\mathcal{D} = \{(x_i, y_i)\}_{0 \le i < n}$, $\bar{y}$, $\varepsilon$
**Output:** $x'$ such that $y' = \mathcal{D}(x')$ and
        $y' = \text{Argmin}(d(y, \bar{y}))_{y \in \mathcal{D}}$ and
        $d(y', \bar{y}) < \varepsilon$
$x' = \varnothing$; $r = +\infty$;
for $(x, y) \in \mathcal{D}$ do
    if $d(y, \bar{y}) <= \varepsilon$ then
        if $d(y, \bar{y}) < r$ then
            $x' = x$;
            $r = d(y, \bar{y})$;
        end
    end
end
```

Algorithm 1: Dictionary attack on perceptual hashes.

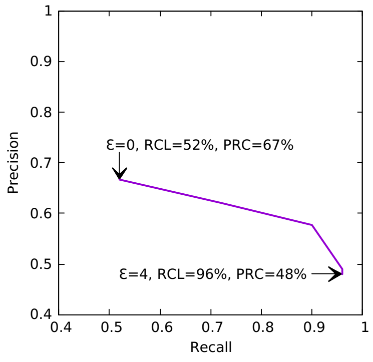

Fig. 19. Performance evaluation of the dictionary attack on perceptual hashes. The precision and recall are presented as a parametric curve computed using the threshold ε. The distance threshold ε covers the interval $[0; 4]$.

9.3.2 Building a dictionary of commands and digests

The attack requires that the attacker holds a dictionary of commands and hashes. Such a dictionary could be created during a learning phase prior to the attack. During this phase, the attacker is both able to capture voice commands and perceptual hashes, for instance by being physically present in the same room. Another approach could be to monitor BLE traffic and to correlate perceptual hashes with observable events such as network traffic and state changes of smarthome appliances.

We considered the possibility of using a voice synthesizer to build this dictionary. However, as noted in Section 9.2, the values of the perceptual hashes depend on the speaker. Thus, it is not possible to build a universal dictionary. This issue could be tackled with the help of machine learning to mimic the voice of the target [41]. We leave this direction for future work.

9.3.3 Evaluation

We evaluated the performance of the dictionary attack using an experiment with the Siri assistant. A dictionary, A, of 100 commands (see Table 19 in Appendix A) was built based on a list of common Siri commands [42]. Each command was then spoken to the device and the corresponding perceptual hash was recorded.

We then played a second set B of commands composed of two sets: B_1 including 50 commands from the dictionary A, and B_2 including 50 commands not in the dictionary A (see Table 19 in Appendix A). Based on the corresponding hashes, we ran the dictionary attack for various values of the threshold ε.

We measured how successful the attack was by its precision and recall [43]. The precision is defined as the proportion of correctly matched commands among all matched commands, and the recall is defined as the proportion of correctly matched commands among B_1. Fig. 19 presents the precision and recall of the dictionary attack. A precision as high as 67% with a recall of 52% can be obtained for a threshold of $\varepsilon = 0$, corresponding to an exact match of the perceptual hashes. Relaxing this constraint by taking larger values of ε will lead to higher recall at the cost of a precision reduction.

Due to the small size of this dictionary, the performance presented above is likely to be overestimated. Indeed, if the size of the dictionary increases, the distance between hashes will diminish and increase the probability of collision. There is an upper bound[12] for the number of distinguishable commands: $2^{16-2.\varepsilon}$. For $\varepsilon = 2$, the number of distinguishable commands is at most $2^{12} = 4096$.

10 Impact of the attacks

In this section, we discuss the impact of the presented attacks on user privacy. A first element to consider is the feasibility of the attacks regarding to the range. In fact, BLE was designed to provide a range of up to 100 meters in outdoor environments [44], but it is usually shorter in

12 Hashes can be seen as words of a code of length 16 with correction capacity ε. Thus, the number of distinguishable hashes is the dimension of the code k. From the Singleton bound and the definition of minimal distance: $k \le 16 - 2.\varepsilon$

practice. We measured the BLE range of several Apple devices and found that they can all be received at least to 61 and 38 meters respectively in outdoor and indoor environments (see Table 18). This means that our attacks can be performed from a significant distance like from the other side of the street or from another room.

A second element to consider is the context in which the attack can be performed, as some messages may only be available in certain conditions. For instance, Nearby Info messages are continuously available thus constantly exposing the user to the corresponding tracking threat. On the other hand, AirDrop messages, that can be harvested for identifiers, are only available upon a file transfer. Table 3 summarizes the context in which the messages are broadcasted along with corresponding threats.

Most of those attacks are straightforward to implement provided the aforementioned conditions are met. However, hashed identifier and voice command recovery may be more complex to implement. They both require prior knowledge under the form of a dictionary and the voice command attack can be negatively impacted by speech variations.

A last aspect to consider is the privacy risk [45] associated with the elements of information obtained by a potential attacker. Through activity inference and tracking, users may be affected by stalking, surveillance and burglary. E-mail addresses and phone numbers may expose the user to spear-phishing and malicious account recovery. Finally, most information elements could be leveraged to profile the user as they reveal identifiers, activities, whereabouts and owned devices.

11 Recommendations

This section presents a set of recommendations for the design and implementation of continuity protocols, that complements the ones introduced in [1] and [14].

Encryption and content minimization of continuity messages: When possible – e.g. when devices are associated with the same iCloud account and thus share cryptographic keys, or when such keys are exchanged during a pairing procedure between two devices – content of Apple Continuity messages should be encrypted[14]. In case the data cannot be encrypted, the content exposed in cleartext should be kept to a bare minimum. For instance, by avoiding the exposure of device status and characteristics.

Timestamps: Timestamping Apple Continuity messages is a simple countermeasure that could be included to defeat replay attacks (see Section 5). Most devices have a local clock and messages already include authentication tags. Including a timestamp with a coarse granularity (seconds or minutes) would be enough to prevent long term tracking while avoiding to expose users to clock-based fingerprinting [46].

Synchronization between continuity protocols and device address changes: Identifiers, counters and payloads included in continuity messages can expose users to tracking (see Section 4) if they are not rotated exactly at the same time as the BLE device address. The rotation of the address and the content of Apple Continuity messages should be carefully synchronized. Note that, the BLE device address and continuity protocols do not belong to the same network layer, which could make the synchronization challenging from a technical point of view.

12 Related work

Several works have analyzed the pitfalls of device address randomization in Wi-Fi [8, 27] and BLE [1, 14, 15]. The payload of advertisement packets can include static identifiers that can be trivially used for tracking. Similarly, predictable fields and temporary identifiers found in those frames can be used to link two consecutive pseudonyms [8, 14, 15, 27]. We show that analogous pitfalls affect the payload of Apple Continuity protocols.

The recent work [1] by Martin et al. is close to ours, as it demonstrates several privacy issues with Apple Continuity protocols. Nevertheless, we reveal additional issues not identified by Martin et al.. First we extend the replay attack of Martin et al. and present a variant that can be used in a larger number of cases. Then, we identify additional artifacts that can be exploited for tracking and fingerprinting; in particular, battery levels and counter broadcasted by AirPods. We demonstrate an attack exposing smarthome activity. We show that, in addition to the SSID, continuity messages can also include e-mail addresses and phone numbers. Finally, we discover that Siri commands could leak over BLE.

Another undocumented protocol of Apple, the *Apple Wireless Direct Link* (AWDL) has been reverse engineered by Stute et al. [47] who independently discovered the hashed identifiers in AirDrop messages, but

14 This is the approach followed by Microsoft within the CDP protocol [3, sec. 2.2.2.2.3].

Table 3. Summary of Apple Continuity messages (CM) with their conditions of emission and their associated privacy threats.

Apple CM type	Code	Message emission	Privacy threats			
			Activity inference	Device attributes	Tracking	User info
AirPrint	0x03	On user action	✓		✓	
AirDrop	0x05	On user action	✓		✓	✓
HomeKit	0x06	Constantly	✓	✓		
Proximity Pairing	0x07	Constantly		✓	✓	
"Hey Siri"	0x08	On user action		✓		
AirPlay	0x09	On user action	✓		✓	
Magic Switch	0x0B	On user physical[13] action		✓	✓	
Handoff	0x0C	On user action			✓	
Instant Hotspot						
Tethering Target Presence	0x0D	On user action	✓		✓	
Tethering Source Presence	0x0E	Reaction to *Target Presence*		✓	✓	
Nearby						
Nearby Action	0x0F	On user action	✓	✓	✓	✓
Nearby Info	0x10	Constantly	✓		✓	

[14] The Watch loses the Bluetooth connection to its paired iPhone and its screen is on.

not those that we found in Nearby Action messages. Furthermore, their work does not consider the recovery of those identifiers; rather, it focuses on the exploitation for illegitimate activation of AirDrop mechanism on nearby phones. The same way, *Google Nearby*, a continuity protocol of Google, has been reverse engineered and analyzed [48] to reveal a number of security flaws.

Network traffic generated by connected devices has been leveraged to reveal the activity of users. In [12], authors showed how the BLE traffic from a fitness tracker can leak the physical activity of the wearer. An active attack to infer state changes of a Bluetooth device is demonstrated in [49]. Inference of human activities based on connected devices and smarthome traffic has been demonstrated by several works [34, 50, 51]. As opposed to those works, we do not rely on an analysis of the traffic features as the information are readily available in cleartext to a passive attacker.

13 Conclusion

We presented a collection of privacy issues in the Apple Continuity protocols. Those issues range from mild leakages, such as the exposure of device model, to serious leakages such as the exposure of personal identifiers like e-mail addresses and phone numbers. Furthermore, we show that including those messages in advertisement packets undermines anti-tracking provisions of the BLE standard. To make matter worst, most of those issues can be exploited by a passive attacker.

Several severe security and privacy flaws have been recently exposed in Google [48] and Apple [1, 47] continuity protocols. In all cases, the specifications were not public and the authors had to rely on reverse engineering to understand the system prior to identify the flaws. This is yet another demonstration that security by obscurity does not work. This also shows that even companies with extended resources and dedicated security/privacy teams cannot only rely on internal scrutiny of their systems to avoid such issues.

We believe that those technologies could benefit a joint standardization effort with security researchers in order to specify dedicated protocols, or at least to set up guidelines for the design and implementation.

Responsible disclosure

The vulnerabilities identified were reported to *Apple*, *Osram* and *Eve* on May 29th, 2019.

Acknowledgements

This work was supported by the INSA Lyon - SPIE ICS IoT chair and the H2020 SPARTA Cybersecurity Competence Network project. The authors would like to thank the reviewers for their careful, constructive and insightful comments towards improving this paper.

References

[1] 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. *Proceedings on Privacy Enhancing Technologies*, 2019(4):34–53, 2019.

[2] Google. Nearby. URL https://developers.google.com/nearby/. Accessed: 2019-05-25.

[3] Microsoft. *Microsoft Connected Devices Platform Protocol Version 3*. 2019. URL https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-cdp/f5a15c56-ac3a-48f9-8c51-07b2eadbe9b4. Accessed: 2019-05-25.

[4] Apple. All your devices. One seamless experience. . URL https://www.apple.com/macos/continuity/. Accessed: 2019-05-25.

[5] Apple. MFi Program. . URL https://developer.apple.com/programs/mfi/. Accessed: 2019-05-25.

[6] Apple. Home accessories. The list keeps getting smarter. . URL https://www.apple.com/ios/home/accessories/. Accessed: 2019-05-25.

[7] Apple. Apple Reports Record First Quarter Results. 2016. URL https://www.apple.com/newsroom/2016/01/26Apple-Reports-Record-First-Quarter-Results/. Accessed: 2019-05-25.

[8] Mathy Vanhoef, Celestin Matte, Mathieu Cunche, Leonardo S. Cardoso, and Frank 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*, ASIA CCS '16, pages 413–424, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4233-9. 10.1145/2897845.2897883.

[9] Taher Issoufaly and Pierre Ugo Tournoux. BLEB: Bluetooth Low Energy Botnet for large scale individual tracking. In *2017 1st International Conference on Next Generation Computing Applications (NextComp)*, pages 115–120. IEEE, 2017.

[10] Ben Greenstein, Ramakrishna Gummadi, Jeffrey Pang, Mike Y Chen, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall. Can Ferris Bueller Still Have His Day Off? Protecting Privacy in the Wireless Era. In *HotOS*, 2007.

[11] Mathieu Cunche, Mohamed-Ali Kaafar, and Roksana Boreli. Linking wireless devices using information contained in Wi-Fi probe requests. *Pervasive and Mobile Computing*, 11:56–69, April 2014. ISSN 1574-1192. 10.1016/j.pmcj.2013.04.001.

[12] Aveek K Das, Parth H Pathak, Chen-Nee Chuah, and Prasant 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.

[13] Bluetooth SIG. *Bluetooth Core Specification v4.0*. 2010. URL https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=456433. Accessed: 2019-05-25.

[14] Johannes K Becker, David Li, and David Starobinski. Tracking Anonymized Bluetooth Devices. *Proceedings on Privacy Enhancing Technologies*, 2019(3):50–65, 2019.

[15] Guillaume Celosia and Mathieu Cunche. Saving Private Addresses: An Analysis of Privacy Issues in the Bluetooth-Low-Energy Advertising Mechanism. In *Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. ACM, 2019.

[16] Martin Woolley. Bluetooth Technology Protecting Your Privacy. 2015. URL https://www.bluetooth.com/blog/bluetooth-technology-protecting-your-privacy/. Accessed: 2019-05-25.

[17] IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*, pages 1–74, June 2014. 10.1109/IEEESTD.2014.6847097.

[18] Bluetooth SIG. *Bluetooth Core Specification v5.1*. 2019. URL https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=457080. Accessed: 2019-05-25.

[19] Apple. *iOS Security - iOS 12.3*. 2019. URL https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf. Accessed: 2019-05-25.

[20] Apple. *HomeKit Accessory Protocol Specification (Non-Commercial Version) - Release R2*. 2019. URL https://developer.apple.com//homekit/specification/. Accessed: 2019-08-20.

[21] Apple. About AirPrint. 2019. URL https://support.apple.com/en-us/HT201311. Accessed: 2019-05-25.

[22] Ang Cui, Michael Costello, and Salvatore Stolfo. When Firmware Modifications Attack: A Case Study of Embedded Exploitation. 2013.

[23] Apple. Connect and use your AirPods. 2019. URL https://support.apple.com/en-us/HT207010. Accessed: 2019-05-25.

[24] Dorene Kewley, Russ Fink, John Lowry, and Mike Dean. Dynamic approaches to thwart adversary intelligence gathering. In *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, volume 1, pages 176–185. IEEE, 2001.

[25] Apple. Handoff. . URL https://developer.apple.com/handoff/. Accessed: 2019-05-25.

[26] Apple. Use Handoff to continue a task on your other devices. 2019. URL https://support.apple.com/en-us/HT209455. Accessed: 2019-05-25.

[27] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. A Study of MAC Address Randomization in Mobile Devices and When It Fails. *Proceedings on Privacy Enhancing Technologies*, 2017(4):365–383, 2017.

[28] Lukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Diaz. The leaking battery. In *Data Privacy Management, and Security Assurance*, pages 254–263. Springer, 2015.

[29] Apple. Use Instant Hotspot to connect to your Personal Hotspot without entering a password. 2019. URL https://support.apple.com/en-us/HT209459. Accessed: 2019-08-20.

[30] Kassem Fawaz, Kyu-Han Kim, and Kang G Shin. Protecting Privacy of BLE Device Users. In *USENIX Security Symposium*, pages 1205–1221, 2016.

[31] Marianne Bertrand and Emir Kamenica. Coming Apart? Cultural Distances in the United States Over Time. Technical report, National Bureau of Economic Research, 2018.

[32] Le T Nguyen, Yu Seung Kim, Patrick Tague, and Joy Zhang. IdentityLink: User-Device Linking through Visual and RF-Signal Cues. In *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 529–539. ACM, 2014.

[33] Matthias C Sala, Kurt Partridge, Linda Jacobson, et al. An Exploration into Activity-Informed Physical Advertising Using PEST. In *International Conference on Pervasive Computing*, pages 73–90. Springer, 2007.

[34] Bogdan Copos, Karl Levitt, Matt Bishop, and Jeff Rowe. Is Anybody Home? Inferring Activity From Smart Home Network Traffic. In *2016 IEEE Security and Privacy Workshops (SPW)*, pages 245–251. IEEE, 2016.

[35] Joseph Bonneau. The Science of Guessing: Analyzing an Anonymized Corpus of 70 Million Passwords. In *2012 IEEE Symposium on Security and Privacy*, pages 538–552. IEEE, 2012.

[36] Levent Demir, Amrit Kumar, Mathieu Cunche, and Cedric Lauradoux. The Pitfalls of Hashing for Privacy. *IEEE Communications Surveys & Tutorials*, 20(1):551–565, 2018.

[37] Matthias Marx, Ephraim Zimmer, Tobias Mueller, Maximilian Blochberger, and Hannes Federrath. Hashing of personally identifiable information is not sufficient. *SICHERHEIT 2018*, 2018.

[38] Troy Hunt. The 773 Million Record "Collection #1" Data Breach. 2019. URL https://www.troyhunt.com/the-773-million-record-collection-1-data-reach/. Accessed: 2019-05-25.

[39] Jaap Haitsma and Ton Kalker. A Highly Robust Audio Fingerprinting System. In *Ismir*, volume 2002, pages 107–115, 2002.

[40] Heiko Knospe. Privacy-enhanced perceptual hashing of audio data. In *2013 International Conference on Security and Cryptography (SECRYPT)*, pages 1–6. IEEE, 2013.

[41] Gopala Krishna Anumanchipalli, Kishore Prahallad, and Alan W Black. Festvox: Tools for Creation and Analyses of Large Speech Corpora. In *Workshop on Very Large Scale Phonetics Research, UPenn, Philadelphia*, page 70, 2011.

[42] Sparhandy. Siri commands - endless functions of your virtual assistant. URL https://www.sparhandy.de/apple/info/siri-commands/. Accessed: 2019-05-25.

[43] Hao Fu, Aston Zhang, and Xing Xie. Effective Social Graph Deanonymization Based on Graph Structure and Descriptive Information. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(4):49, 2015.

[44] Jon Gunnar Sponas. Things You Should Know About Bluetooth Range. 2018. URL https://blog.nordicsemi.com/getconnected/things-you-should-know-about-bluetooth-range. Accessed: 2019-08-20.

[45] Nina Gerber, Benjamin Reinheimer, and Melanie Volkamer. Investigating People's Privacy Risk Perception. *Proceedings on Privacy Enhancing Technologies*, 2019(3):267–288, 2019.

[46] Chrisil Arackaparambil, Sergey Bratus, Anna Shubina, and David Kotz. On the Reliability of Wireless Fingerprinting using Clock Skews. In *Proceedings of the third ACM conference on Wireless network security*, pages 169–174. ACM, 2010.

[47] Milan Stute, Sashank Narain, Alex Mariotto, Alexander Heinrich, David Kreitschmann, Guevara Noubir, and Matthias Hollick. A Billion Open Interfaces for Eve and Mallory: MitM, DoS, and Tracking Attacks on iOS and macOS Through Apple Wireless Direct Link. page 18, 2019.

[48] Daniele Antonioli, Nils Ole Tippenhauer, and Kasper Rasmussen. Nearby Threats: Reversing, Analyzing, and Attacking Google's' Nearby Connections' on Android. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, February 2019.

[49] Guillaume Celosia and Mathieu Cunche. Detecting smartphone state changes through a Bluetooth based timing attack. In *Proceedings of the 11th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 154–159. ACM, 2018.

[50] Noah Apthorpe, Dillon Reisman, Srikanth Sundaresan, Arvind Narayanan, and Nick Feamster. Spying on the Smart Home: Privacy Attacks and Defenses on Encrypted IoT Traffic. 2017.

[51] Sandra Siby, Rajib Ranjan Maiti, and Nils Tippenhauer. IoTScanner: Detecting and Classifying Privacy Threats in IoT Neighborhoods. *arXiv preprint arXiv:1701.05007*, 2017.

A Appendix

Table 4. Extended list of HomeKit `Category` codes.

`Category` code	Description
0x0000	Unknown
0x0100	Other
0x0200	Bridge
0x0300	Fan
0x0400	Garage Door Opener
0x0500	Lightbulb
0x0600	Door Lock
0x0700	Outlet
0x0800	Switch
0x0900	Thermostat
0x0A00	Sensor
0x0B00	Security System
0x0C00	Door
0x0D00	Window
0x0E00	Window Covering
0x0F00	Programmable Switch
0x1000	Range Extender
0x1100	IP Camera
0x1200	Video Doorbell
0x1300	Air Purifier
0x1400	Heater
0x1500	Air Conditioner
0x1600	Humidifier
0x1700	Dehumidifier
0x1C00	Sprinklers
0x1D00	Faucets
0x1E00	Shower Systems

Table 5. Extended list of Proximity Pairing `Device Model` codes.

`Device Model` code	Description
0x0220	AirPods
0x0320	Powerbeats3
0x0520	BeatsX
0x0620	Beats Solo3

Table 6. Extended list of Proximity Pairing `UTP` codes.

`UTP` code	Description
0x01	In Ear
0x02	In Case
0x03	Airplane

Table 7. Extended list of Proximity Pairing `Device Color` codes.

`Device Color` code	Description
0x00	White
0x01	Black
0x02	Red
0x03	Blue
0x04	Pink
0x05	Gray
0x06	Silver
0x07	Gold
0x08	Rose Gold
0x09	Space Gray
0x0A	Dark Blue
0x0B	Light Blue
0x0C	Yellow

Table 8. Extended list of "Hey Siri" `Device Class` codes.

`Device Class` code	Description
0x0002	iPhone
0x0003	iPad
0x0009	MacBook
0x000A	Watch

Table 9. Extended list of Magic Switch `Confidence on Wrist` codes.

`Confidence on Wrist` code	Description
0x03	Not on wrist
0x1F	Wrist detection disabled
0x3F	On wrist

Table 10. Extended list of Tethering Source Presence `Network Type` codes (as reported by [1]).

`Network Type` code	Description
0x01	1xRTT
0x02	GPRS
0x03	EDGE
0x04	3G (EV-DO)
0x05	3G
0x06	4G
0x07	LTE

Table 11. Extended list of Nearby Action `Action Type` codes.

`Action Type` code	Description
0x01	Apple TV Tap-To-Setup
0x04	Mobile Backup
0x05	Watch Setup
0x06	Apple TV Pair
0x07	Internet Relay
0x08	Wi-Fi Password
0x09	iOS Setup
0x0A	Repair
0x0B	Speaker Setup
0x0C	Apple Pay
0x0D	Whole Home Audio Setup
0x0E	Developer Tools Pairing Request
0x0F	Answered Call
0x10	Ended Call
0x11	DD Ping
0x12	DD Pong
0x13	Remote Auto Fill
0x14	Companion Link Prox
0x15	Remote Management
0x16	Remote Auto Fill Pong
0x17	Remote Display

Table 12. Extended list of Nearby Action `Device Class` codes.

`Device Class` code	Description
0x2	iPhone
0x4	iPod
0x6	iPad
0x8	Audio accessory (HomePod)
0xA	Mac
0xC	AppleTV
0xE	Watch

Table 13. Extended list of Nearby Action `Device Model` codes.

`Device Model` code	Description
0x1	D22ish
0x2	SEish
0x3	JEXXish

Table 14. Extended list of Nearby Action `Device Color` codes.

`Device Color` code	Description
0x00	Unknown
0x01	Black
0x02	White
0x03	Red
0x04	Silver
0x05	Pink
0x06	Blue
0x07	Yellow
0x08	Gold
0x09	Sparrow

Table 15. Extended list of Nearby Action `OS Version` codes.

`OS Version` code	Description
0x09	Version 9
0x0A	Version 10

Table 16. Extended list of Nearby Info `Activity Level` codes.

`Activity Level` code	Description
0x00	Activity level is not known
0x01	Activity reporting is disabled
0x03	User is idle
0x05	Audio is playing with the screen off
0x07	Screen is on
0x09	Screen on and video playing
0x0A	Watch is on wrist and unlocked
0x0B	Recent user interaction
0x0D	User is driving a vehicle
0x0E	Phone call or Facetime*

* As reported by [1].

Table 17. Perceptual hashes (pHash) obtained from commands issued by two male users. Hamming distances are computed between pHash of $User_A$ and $User_B$ considered as binary values (sequences of bits).

Command	pHash		Hamming
	$User_A$	$User_B$	dist.
Call Mark.	0xEFF0	0xB608	9
Call Bob.	0xFDF0	0xAF08	8
Play some music.	0x9F82	0x9547	6
Search the web for 'privacy'.	0x39EF	0xFE81	10
Send a message to Mark.	0x1438	0xB3B4	8
Send a message to Bob.	0x10B0	0xB31C	8
Set a timer for 3 minutes.	0xB680	0x181F	11

Table 18. Experimental evaluation of the range of Apple Continuity BLE messages leveraging various Apple devices in different environment settings.

Apple device	Range (in meters)				
	Indoor (#walls*)				Outdoor
	0	1	2	3	
AirPods (2nd generation) Model: A1602, A2031/2	49	45	41	39	61
iPad (5th generation) Model: A1822	51	49	44	41	63
iPad Mini 3 Model:A1599	50	48	44	42	62
iPhone 6 Model: A1586	50	46	43	38	62
iPhone 8 Model: A1905	51	49	46	42	65
MacBook Air (13", 2014) Model: A1466	52	50	47	45	65
MacBook Pro (13", 2015) Model: A1502	52	51	49	47	67
Watch Series 2 Model: A1757	50	47	45	42	61
Watch Series 3 Model: A1858	51	49	46	43	62

* Walls are constituted of plasterboard and are about 15 centimeters thick.

Table 19. Complete list of Siri commands used for the dictionary attack on perceptual hashes.

Commands of dictionaries A and B_1 **(commands of B_1 are in bold)**.	**Activate Do Not Disturb.**; Add 'tomatoes' to the grocery list.; **Add a reminder.**; **Alarm in 5 hours.**; Alice is my mother.; Best comedy movies ?; **Best horror movies ?**; **Bob is my brother.**; Call 408 555 1212.; **Call Bob.**; **Call Mark.**; Call me sweetheart.; **Call the nearest restaurant.**; Can you recommend a movie ?; Cancel my event with Mark.; Decrease brightness.; FaceTime Audio call to Alice.; Find number of dad.; How are the markets doing ?; **How big is the biggest elephant ?**; How far away is Mars ?; How far away is Tokyo ?.; How humid is it in Paris ?; **How long do cats live ?**; **How many calories in an apple ?**; How many days until Christmas ?; **How many teeth does a cat have ?**; How old is Madonna ?; How old is Peter ?; How tall is Paris Hilton ?; Increase brightness.; **Inform my husband when I'm back home.**; **Inform my wife when I leave my home.**; **Is mom at home ?**; **Learn to pronounce my name.**; **Listen to Alicia Keys.**; **Locate my father.**; Note: 'Susan will be late tonight'.; **Open Instagram.**; **Open Spotify.**; **Play some music.**; Play the rest of this album.; Read my new messages.; Remind me today: call Kevin.; Search Google for pictures of Thor.; Search the web for 'computer'.; **Search the web for 'privacy'.**; **Send a message to Bob.**; **Send a message to Mark.**; **Send an e-mail to Susan.**; Set a timer for 3 minutes.; Show all my photos.; Show me best family movies.; **Show me my notes.**; Show me my photos from London.; Show me my photos of last week.; Show me new e-mail from Peter.; **Show me the latest tweets.**; Show me the nearest cinema.; Show me tweets from Peter.; **Show me videos of Avengers.**; Show my favorite photos.; Show my selfies.; **Square root of 49 ?**; **Susan is my sister.**; **Take a video.**; **Take me home.**; **Turn on Night Mode.**; **Turn on Wi-Fi.**; What day is in 5 days ?; What day was 2 days ago ?; **What did Dow close at today ?**; **What did Nikkei close at today ?**; What is my altitude ?; **What is the time zone in Miami ?**; **What time is it ?**; What's 7 plus 2 ?; **What's Kevin's address ?**; **What's the capital of France ?**; What's the date ?; **What's the definition of 'robot' ?**; **What's the Nasdaq today ?**; **What's the Nikkei price ?**; What's the temperature outside ?; What's the temperature tonight ?; **What's this song ?**; When am I meeting with Alice ?; When do I meet Bob ?; When is my next appointment ?; When is the sunrise ?; **When is the sunset ?**; When is the Super Bowl ?; Where died Bob Marley ?; **Where is my iPhone ?**; **Where is my next appointment ?**; **Where lives Susan ?**; Which movies are with Tom Hanks ?; **Who does this smartphone belong to ?**; Who is Sean Connery married to ?; **Who sings this ?**
Commands of dictionary B_2.	Andrew is my boyfriend.; Any new e-mail from Kevin ?; Call Abraham on speakerphone.; Call me a cab.; Call me king.; Cancel my event with Alice.; Compare Nikkei with Dow.; Deactivate Do Not Disturb.; Delete all alarms.; Delete the reminder 'project'.; Find some movie theaters near my home.; Flip a coin.; How many bones does a dog have ?; How many days until the birthday of dad ?; How many days until year 2020 ?; How small is the smallest dog ?; In which city lives Peter ?; Is 'Airplane mode' enabled ?; Navigate to Susan by car.; Open Facebook.; Open mail.; Open settings.; Pause the timer.; Play me my latest voicemail.; Play the Titanic soundtrack.; Play the trailer for 'Pearl Harbor'.; Play this song from the beginning.; Play top 10 songs from Aya Nakamura.; Read Calendar.; Runtime of Titanic ?; Search Wikipedia for 'dog'.; Show me my photos from Lisbon.; Show me my photos of yesterday.; Show me pictures of Hulk.; Show me the appointments for next month.; Show me the traffic.; Tell me a story.; Text Susan: 'I will be late'; Translate 'cat' from English in Russian.; Turn on Cellular Data.; What is the time at home ?; What's the current dew point ?; What's the Nasdaq price ?; What's the pressure outside ?; What's the visibility outside ?; When died Freddie Mercury ?; When is Kevin's birthday ?; Where is my MacBook ?; Where is the office of Sean ?; Which songs are from Imagine Dragons ?