black hat®
EUROPE 2019

DECEMBER 2-5, 2019

EXCEL LONDON, UK

#BHEU @BLACK HAT EVENTS

Trust in Apple's Secret Garden: Exploring & Reversing Apple's Continuity Protocol

Ta-Lun Yen
talun_yen@trendmicro.com @evanslify

whoami

- From Taiwan
- Independent Security Researcher
  - Threat Researcher @ TXOne Networks (Trend Micro), 2019/11-present
- Focused on protocol analysis, wireless, hardware
- Previously: HITCON 2018, 2019
- Powerlifting

Agenda

- Overview
  - Background
  - Continuity Protocol introduction
- Prior Studies
  - Current status of Continuity’s Security
- Our attack scenario
  - Fingerprinting / Tracking / Metadata leak
  - Breaking MAC Rotation
- Demo

Apple’s unboxing experience

Apple Continuity

Continuity Protocol

- A proprietary protocol used by Apple’s devices, based on BLE & Wi-Fi
- Integrated with iCloud (Public Key Infrastructure)
- For users to move seamlessly between devices
  - Phone calls, clipboards, hotspot, camera, airdrop, messages

Why this topic?

- Research was intended to re-implement Continuity on Linux
- Switched from Mac to Linux 2018/9
  - I missed AirDrop / Instant Hotspot
  - Hotspot has gimmicks, not working 100% of time
    - Settings menu has to be open, but sometimes still fails

Why this topic?

- Continuity protocol hasn’t been discussed before
- Fired up Ubertooth & PacketLogger
- It’s more interesting to study its security/privacy implications

Why this topic?

Responsible Disclosure

- Initially reported to Apple (8/5/2019) and reviewed prior to presentation during HITCON 2019
- Resubmitted to Apple in relation to Black Hat EU presentation on 11/21/2019
- Wi-Fi - CVE-2019-8854

  Impact: A device may be passively tracked by its WiFi MAC address

  Description: A user privacy issue was addressed by removing the broadcast MAC address.

Related Work & References

- Garman et al. (2016) Dancing on the Lip of the Volcano: Chosen Ciphertext Attacks on Apple iMessage https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/garman
- Martin Vigo (2017) DIY Spy Program: Abusing Apple's Call Relay Protocol. https://www.martinvigo.com/diy-spy-program-abusing-apple-call-relay-protocol/
- Celosia & Cunche (2019) Fingerprinting Bluetooth-Low-Energy Devices Based on the Generic Attribute Profile. https://dl.acm.org/citation.cfm?id=3358617https://dl.acm.org/citation.cfm?id=3358617
- Becker et al. (2019) Tracking Anonymized Bluetooth Devices. https://content.sciendo.com/view/journals/popets/2019/3/article-p50.xml
- Stute et al. (2018) One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol. https://arxiv.org/abs/1808.03156
- Martin et al. (2019) Handoff All Your Privacy: A Review of Apple's Bluetooth Low Energy Continuity Protocol. https://www.cmand.org/furiousmac/

Related Work & References

- Disclaimer
  - This research was done prior to joining Trend Micro
  - Some findings of this research are similar to (but not based on) the one released by Martin et al. in April 2019

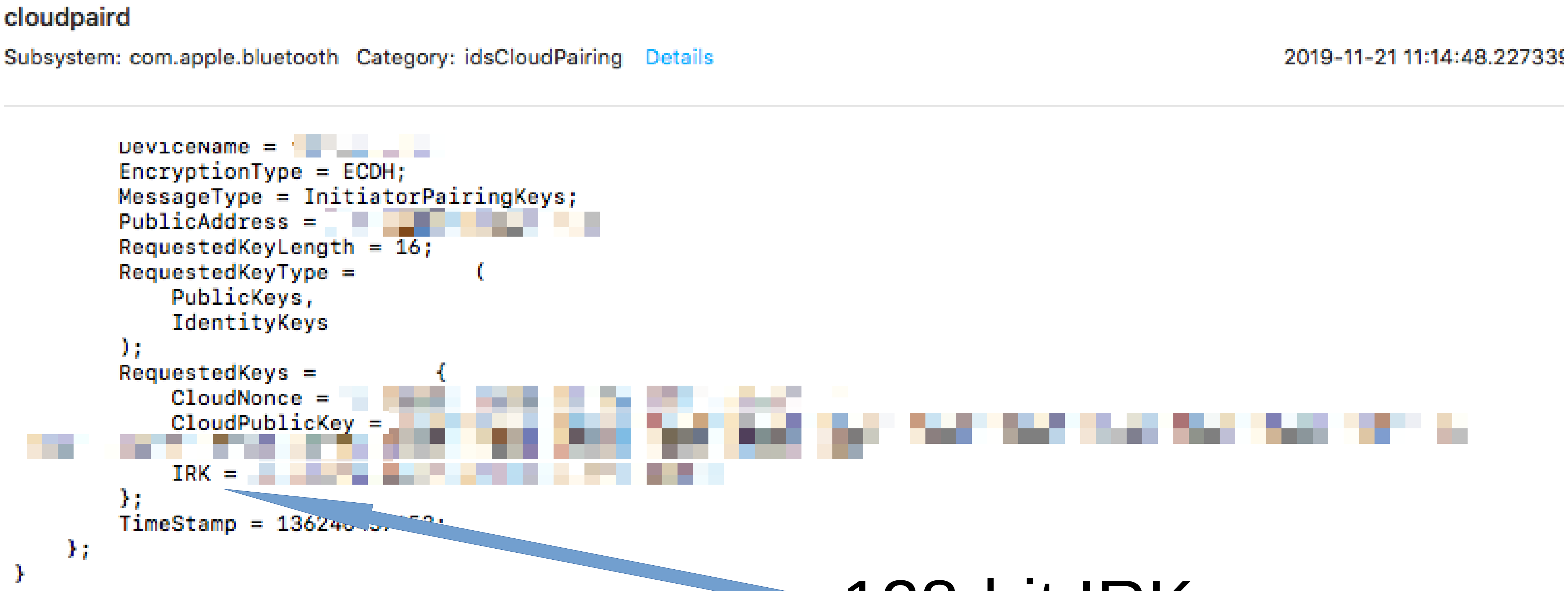
Overview - Glossary

Bluetooth Low Energy

- Workhorse of the Continuity protocol
- Can be used to bootstrap another protocol in Continuity
- Out-of-band pairing via iCloud
- Use “Private resolvable address” while broadcasting

Out-of-Band via iCloud

- Device “onboard” to each iDevice after iCloud login



cloudpaird

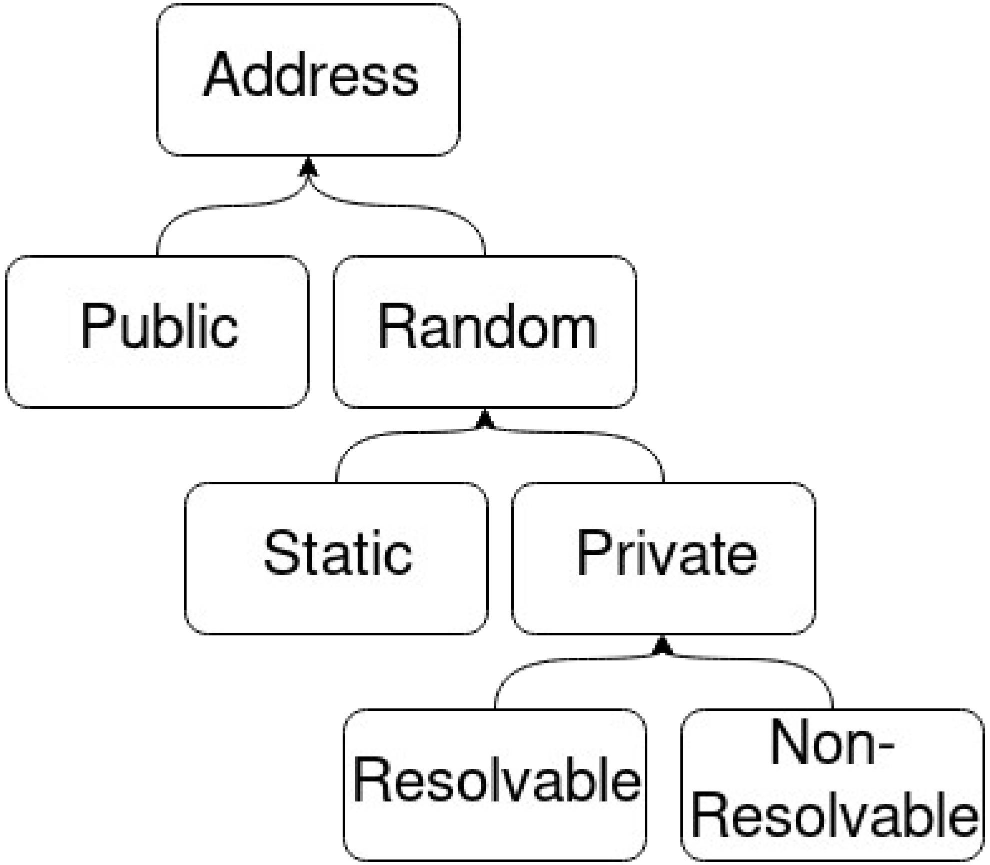
Subsystem: com.apple.bluetooth Category: idsCloudPairing Details 2019-11-21 11:14:48.22733

```
        DeviceName = '[redacted]
        EncryptionType = ECDH;
        MessageType = InitiatorPairingKeys;
        PublicAddress = [redacted]
        RequestedKeyLength = 16;
        RequestedKeyType =          (
            PublicKeys,
            IdentityKeys
        );
        RequestedKeys =          {
            CloudNonce = [redacted]
            CloudPublicKey = [redacted]
            IRK = [redacted]
        };
        TimeStamp = 1362[redacted];
    };
}
```

128-bit IRK

Private resolvable address

- A way to randomize MAC, remain recognizable to a few clients
- Address change on each on/off cycle & timeout
- AES-128 key (IRK) to identify devices



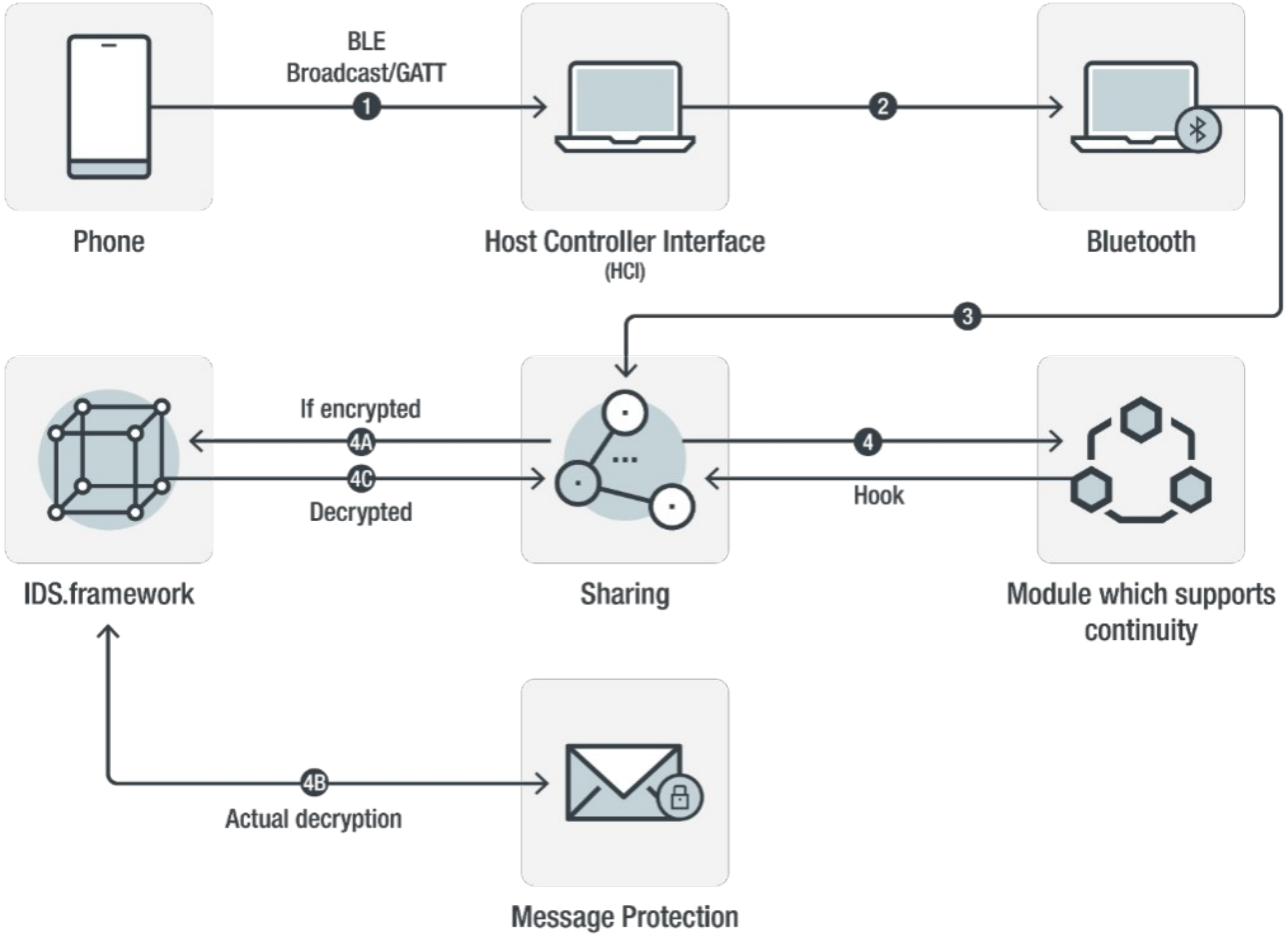
GATT

- Generic Attribute Profile
- Used to transfer data in BLE

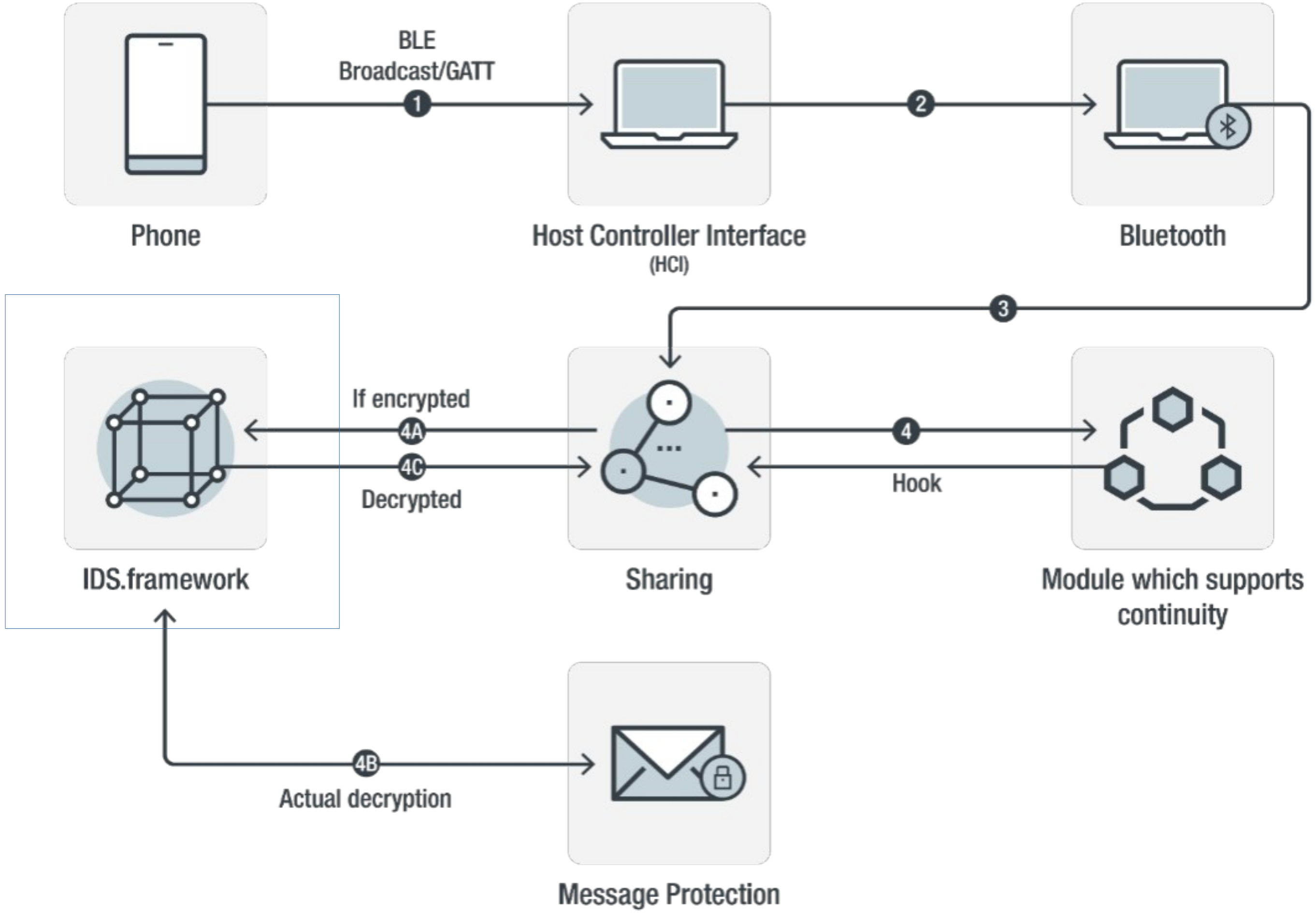
- 128-bit UUID to identify specific resource
  - One ID for device name, one for battery level, etc
  - Specific (2) ID for Continuity

Overview - Continuity

Continuity protocol stack



Continuity protocol stack



IDS.framework

- Apple’s directory service for every(!) device
- Integral part of iMessage/Continuity’s encryption
- Able to fetch any device’s public key with corresponding phone #/email

Encryption

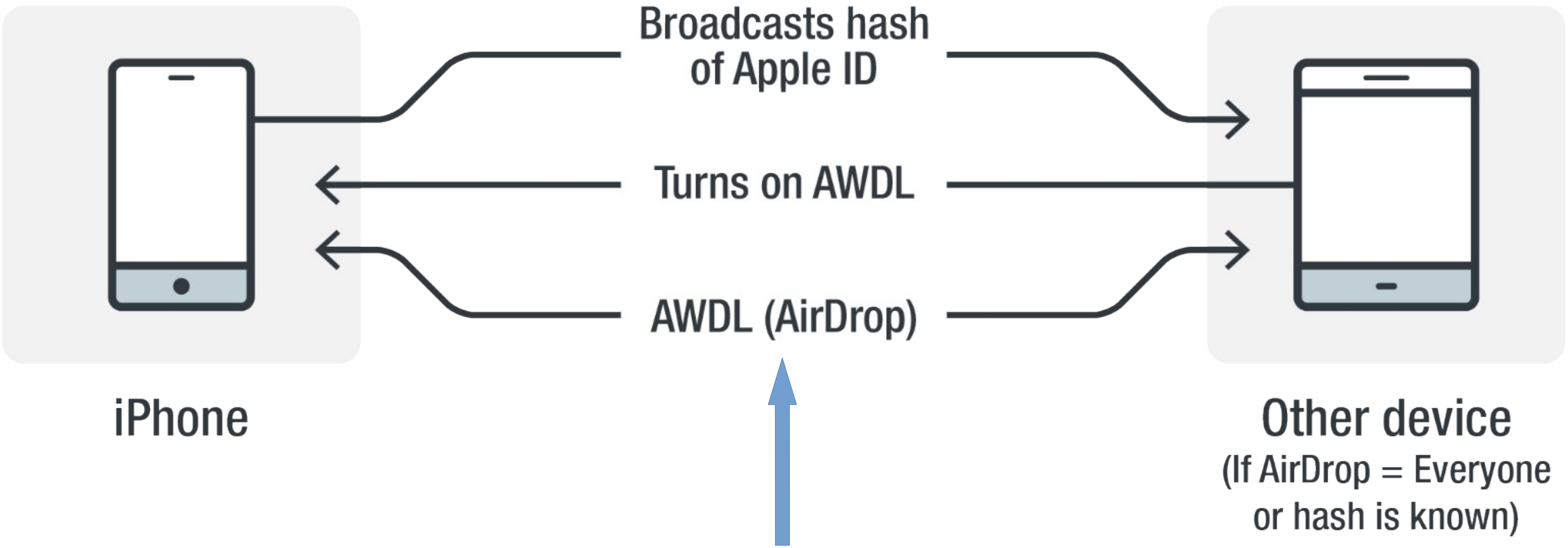
- GATT Exchanges are encrypted
- RSA-1280 to decrypt AES-128 in payload
- Key obtainable through IDS (→iCloud) & Keychain

Current Status of Continuity’s Security

Current Status of Continuity’s Security

- Protocols with vulnerabilities before
- Both used daily & might affect daily lives
  - AirDrop
    - Send files to other iDevices without hassle
  - Call Relay
    - Make calls from other iDevice

AirDrop (AWDL)


Broadcasts hash of Apple ID
Turns on AWDL
AWDL (AirDrop)
iPhone
Other device
(If AirDrop = Everyone or hash is known)

Peer-to-Peer Wi-Fi (Wi-Fi Direct)

Status of AWDL

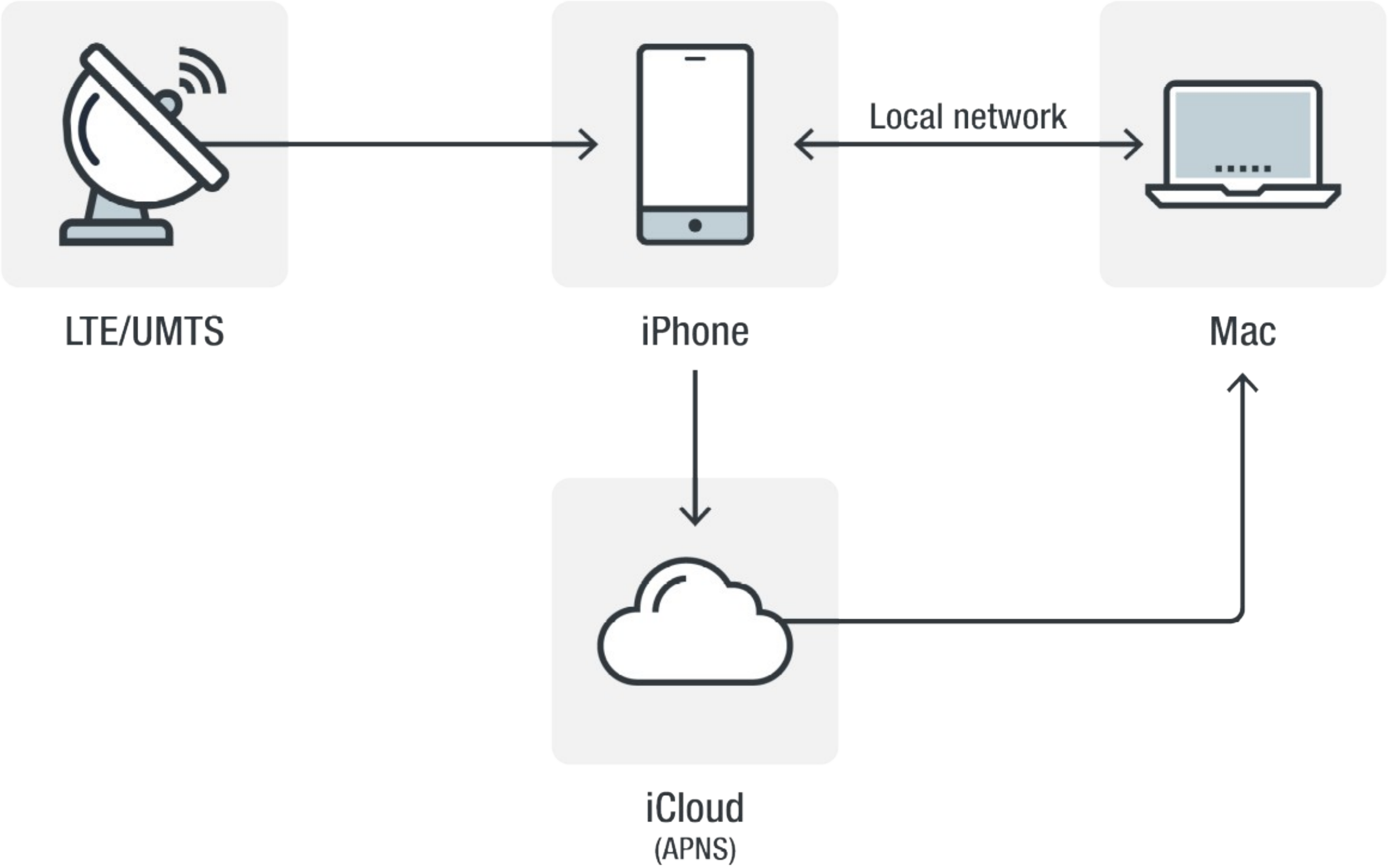
- User Tracking (CVE-2019-8567, CVE-2019-8620)
- MitM Attack (CVE-2019-8612)

- Research &
  Open Source re-implementation

* https://github.com/seemoo-lab/opendrop
* https://github.com/seemoo-lab/owl
* Stute et al. One Billion Apples' Secret Sauce: Recipe for the Apple Wireless Direct Link Ad hoc Protocol

```
n -f owl.png
```

Call Relay



* Vigo, 2017

Integration is hard

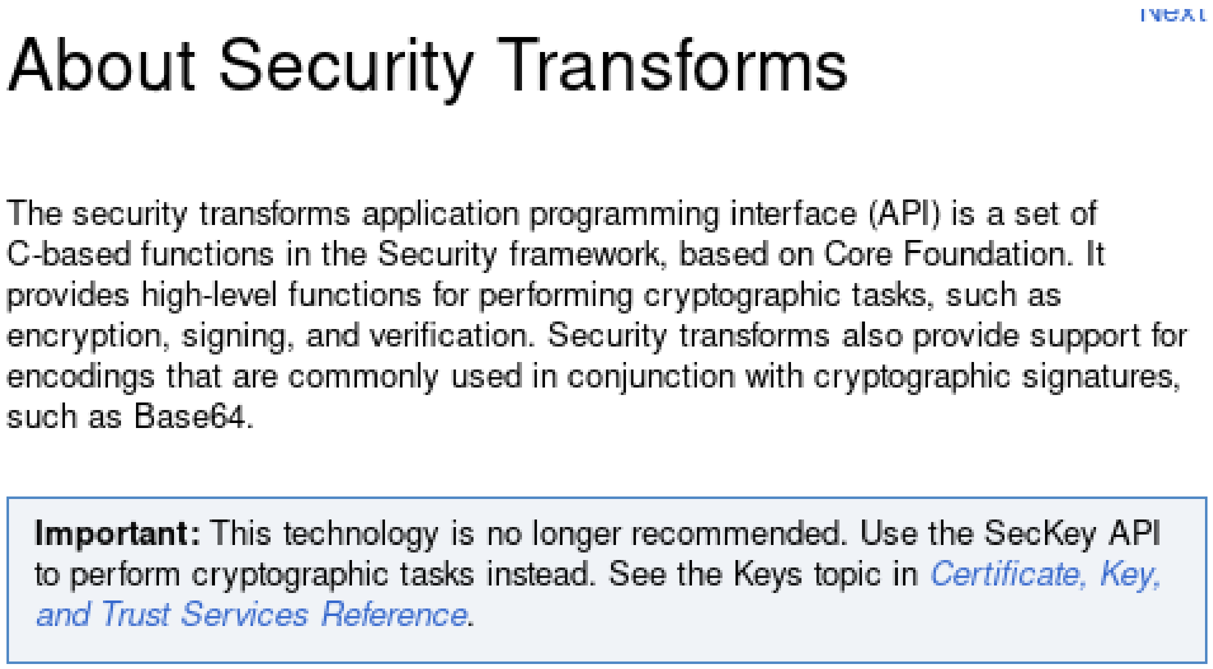
- Call Relay
  - CVE-2016-4635: User interface inconsistencies in handling of relayed calls
  - CVE-2016-4721: Caller spoofing on multiparty calls
  - CVE-2016-4722: End call packet spoofing
  - CVE-2016-7577: Facetime memory corruption

Protocol Implementation

Technical Details ahead

Message Verification & Decryption

- Relies on Security.framework
  - Apple says obsolete
- Security Transforms
  - SecVerifyTransformCreate
  - SecDecryptTransformCreate

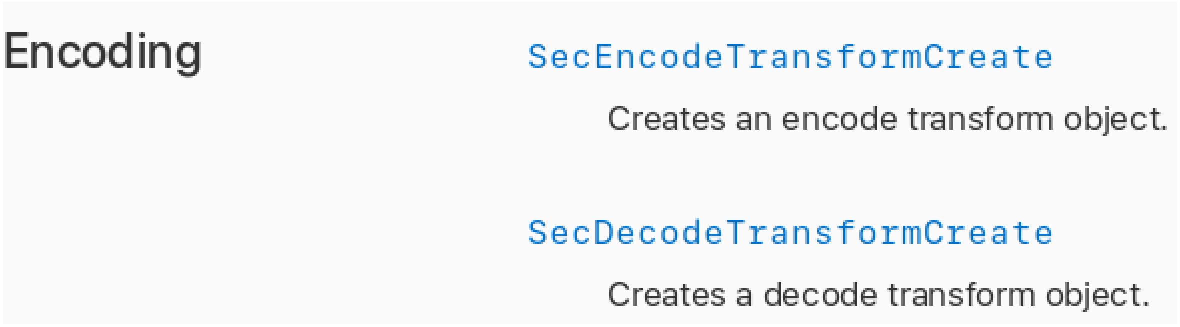


Next

About Security Transforms

The security transforms application programming interface (API) is a set of C-based functions in the Security framework, based on Core Foundation. It provides high-level functions for performing cryptographic tasks, such as encryption, signing, and verification. Security transforms also provide support for encodings that are commonly used in conjunction with cryptographic signatures, such as Base64.

Important: This technology is no longer recommended. Use the SecKey API to perform cryptographic tasks instead. See the Keys topic in *Certificate, Key, and Trust Services Reference*.



Encoding

`SecEncodeTransformCreate`

Creates an encode transform object.

`SecDecodeTransformCreate`

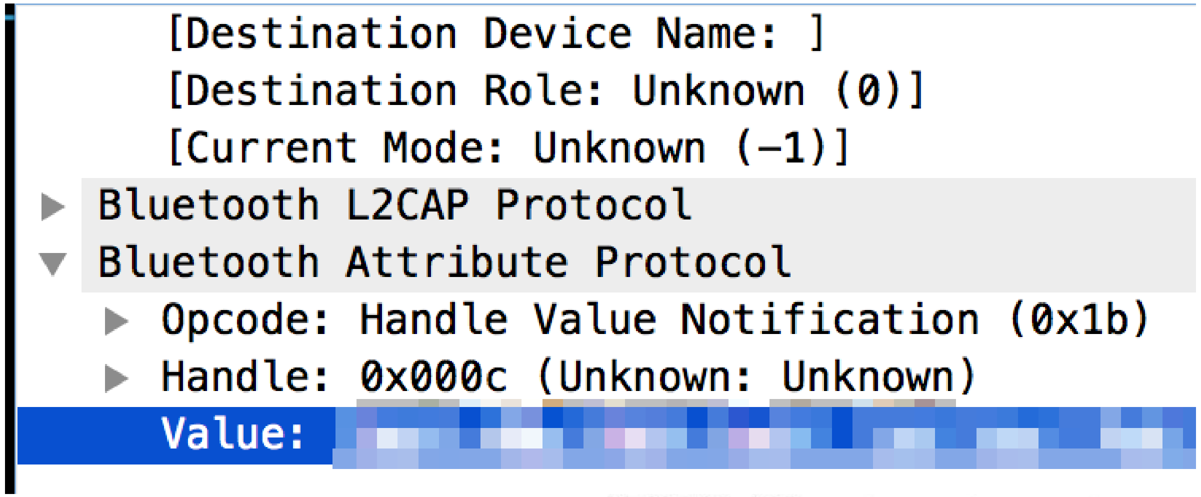
Creates a decode transform object.

Message Verification & Decryption

- SecMPVerifyAndExposeMessage
  - SecMPVerifyMessageContents(payload)
    - Used when sizeof(payload) > 17
- Raw payload from HCI
  - Calls SecKeyDigestAndVerifyWithError
  - Actual decryption is called if verified

GATT Data Structure

- Data can be split into multiple packets
- Payload length at 0x38-0x39
- 0x39-end = Payload + Signature
- Total Length (bthci_acl.length) - Payload length = Signature length

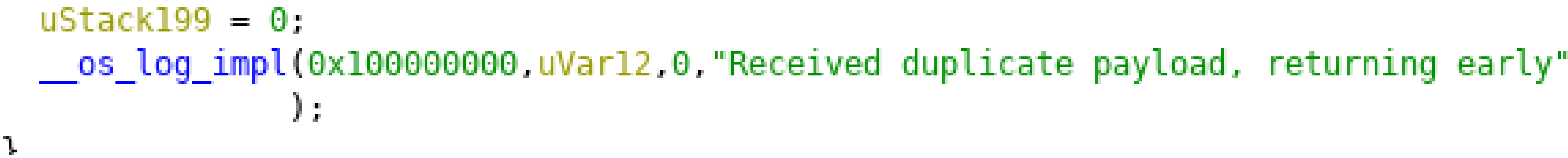


```
    [Destination Device Name: ]
    [Destination Role: Unknown (0)]
    [Current Mode: Unknown (-1)]
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Handle Value Notification (0x1b)
  ▶ Handle: 0x000c (Unknown: Unknown)
    Value: [redacted]
```

Protection

- Messages are signed, but no MAC
- iMessage shared IDS with continuity
  - Huffman table is used in iMessage, but not Continuity
  - iMessage Chosen Ciphertext Attack
- Fixed by hashing every payload and storing it in IDSMessageHashStore, fails when dupes are received

* Garmin et al. 2018



```
  uStack199 = 0;
  __os_log_impl(0x100000000,uVar12,0,"Received duplicate payload, returning early"
                );
}
```

Decryption

- ~160 bytes = RSA-encrypted payload
  - ~16 bytes → AES-128 Key
  - 16~ bytes → Ciphertext A
- 160~ bytes → Ciphertext B
- AES-128 CTR, PK = 1
  - aes_decrypt(ciphertext A + ciphertext B) → gzip → binary plist

Github Project

- https://github.com/evanslify
- Currently a little script to play with broadcast only
- To-do
  - Release de-encryption & encryption
  - Emulate Hotspot behavior

Our Attack Scenario

Attack on Continuity

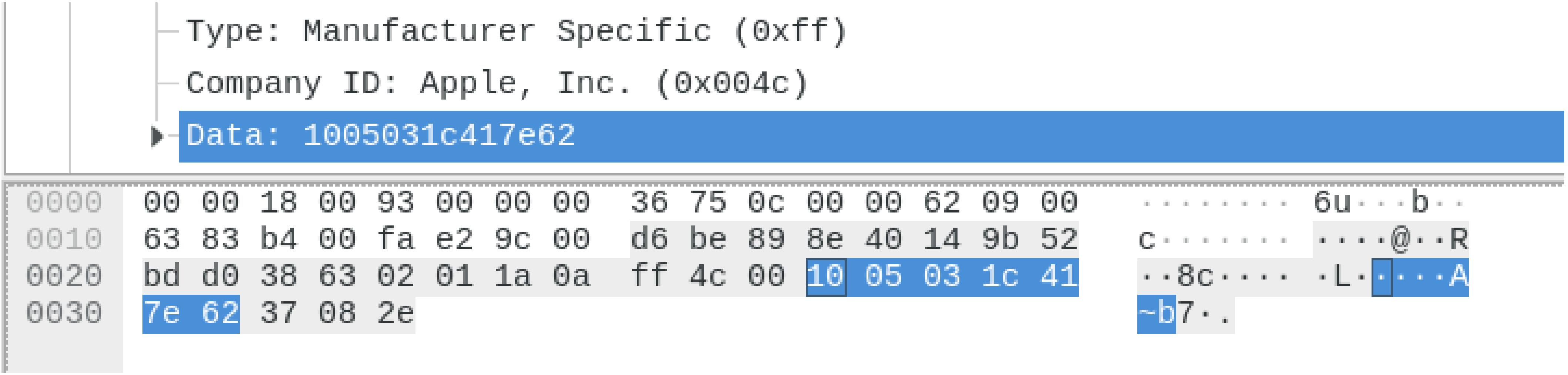
- Exploits parts in-between different protocols
- Some behaviors which leaks device usage, identity
- Allows adversaries to track specific device
- De-anonymization

- Any BLE sniffer can serve as a tracking platform

Attack Overview

- Prerequisite: Format of Continuity broadcast
- Privacy Leak
  - Device Fingerprinting
    - OS Version, device type
  - Activity, Battery levels, etc
- Breaking MAC Randomization
- Spoofing

Continuity Broadcast Format



```
Type: Manufacturer Specific (0xff)
Company ID: Apple, Inc. (0x004c)
Data: 1005031c417e62

0000  00 00 18 00 93 00 00 00  36 75 0c 00 00 62 09 00   ········ 6u···b··
0010  63 83 b4 00 fa e2 9c 00  d6 be 89 8e 40 14 9b 52   c······· ····@··R
0020  bd d0 38 63 02 01 1a 0a  ff 4c 00 10 05 03 1c 41   ··8c···· ·L·····A
0030  7e 62 37 08 2e                                     ~b7·.
```

10	05	03	1c	41	7e	62
Type	Length	Payload				

*17 types as of XCode 10.2 (PacketDecoder)

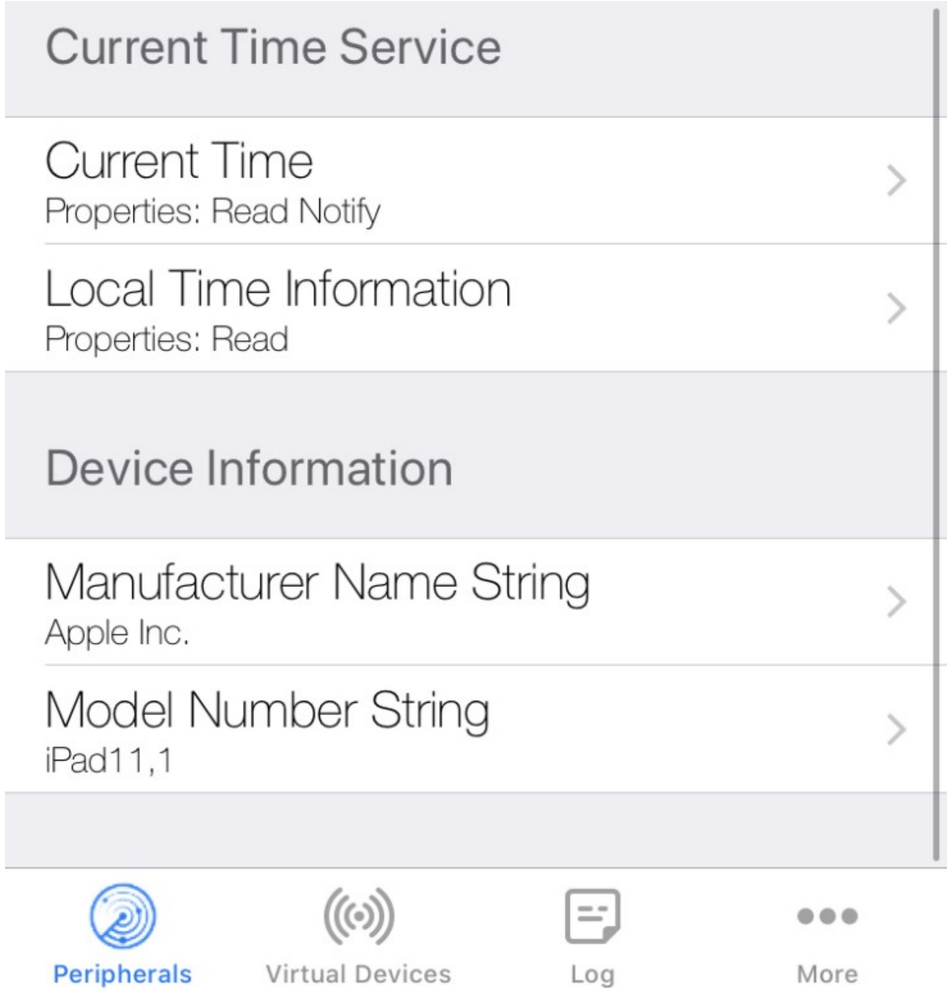
Privacy Leak - Device Fingerprinting

- OS Version / Device type
- Specific types emitted by specific device
  - e.g. iPad Wi-Fi cannot emit Tethering Source
- https://support.apple.com/en-gb/HT204689

Type	ID	
AirPlay Target	0x09	Apple TV
AirPrint	0x03	Printer
Handoff	0x0c	iOS => 8
Tethering Source	0x0e	iOS => 8.1
Nearby	0x10	iOS => 10

Privacy Leak - Active Fingerprinting

- Not a vulnarability itself
- Have to connect to device and interrogate it
- Able to get model number via GATT attributes



Current Time Service

Current Time
Properties: Read Notify

Local Time Information
Properties: Read

Device Information

Manufacturer Name String
Apple Inc.

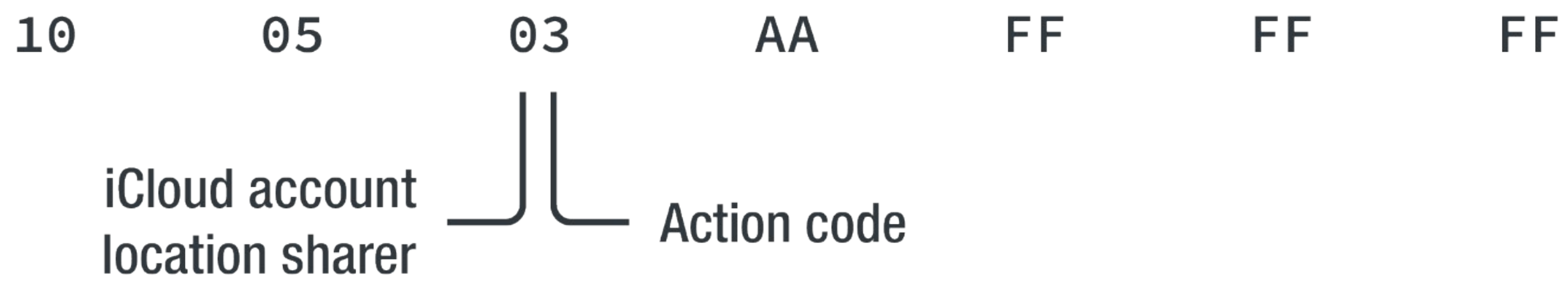
Model Number String
iPad11,1

* Martin et. al, 2019

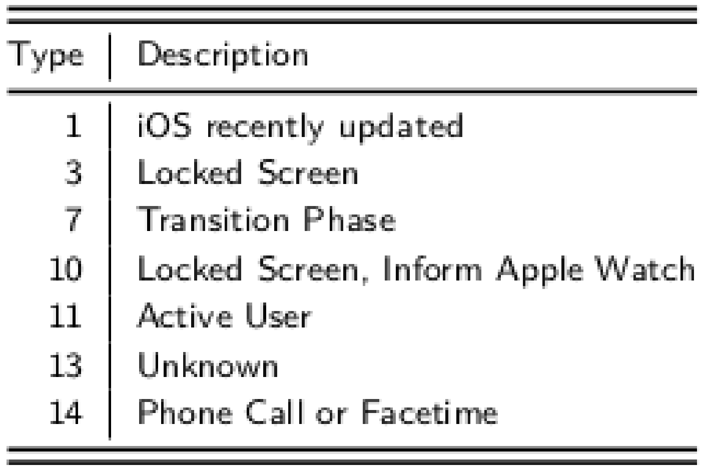
black hat EUROPE 2019

Privacy Leak - Nearby

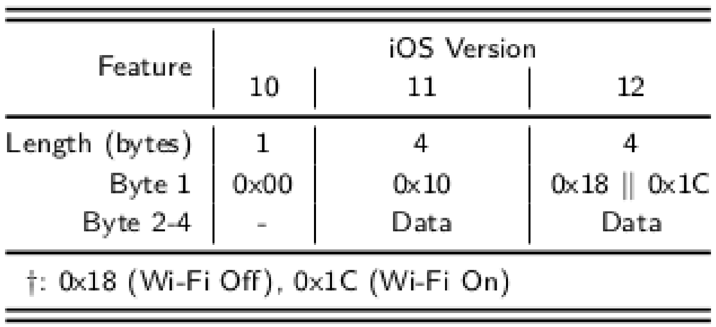
- OS Version leak
  - Format of Wi-Fi field
- Metadata, Usage leak
  - Action values



```
10      05      03      AA      FF      FF      FF
```



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



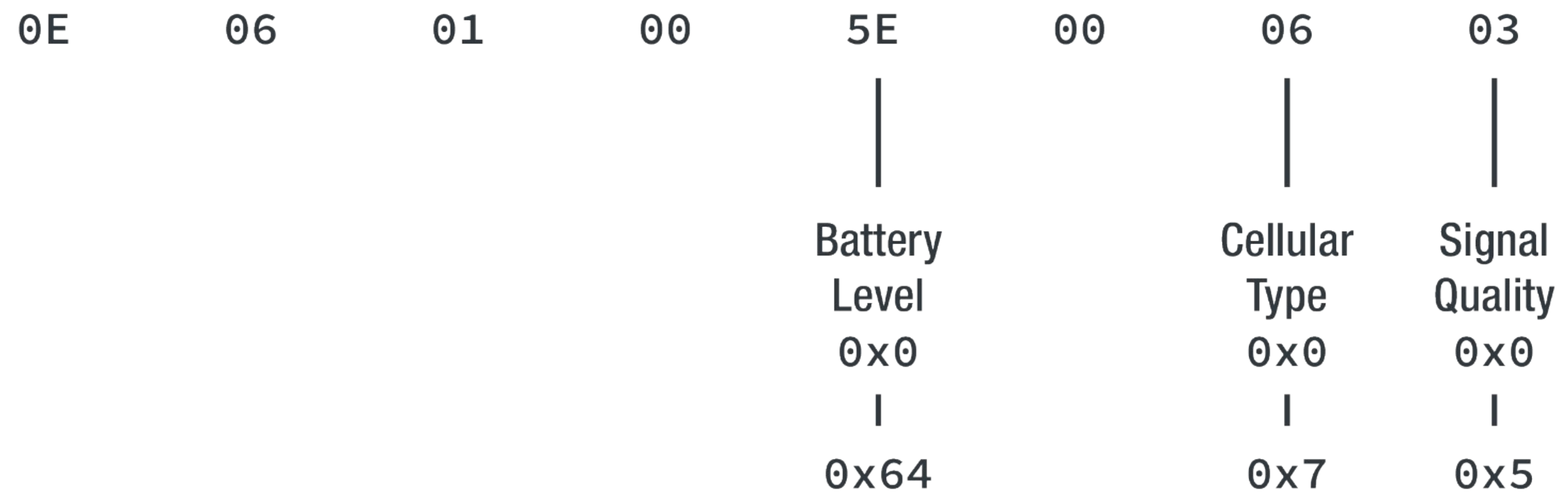
Feature	iOS Version		
	10	11	12
Length (bytes)	1	4	4
Byte 1	0x00	0x10	0x18 || 0x1C
Byte 2-4	-	Data	Data

†: 0x18 (Wi-Fi Off), 0x1C (Wi-Fi On)

* Martin et. al, 2019

Privacy Leak - Instant Hotspot

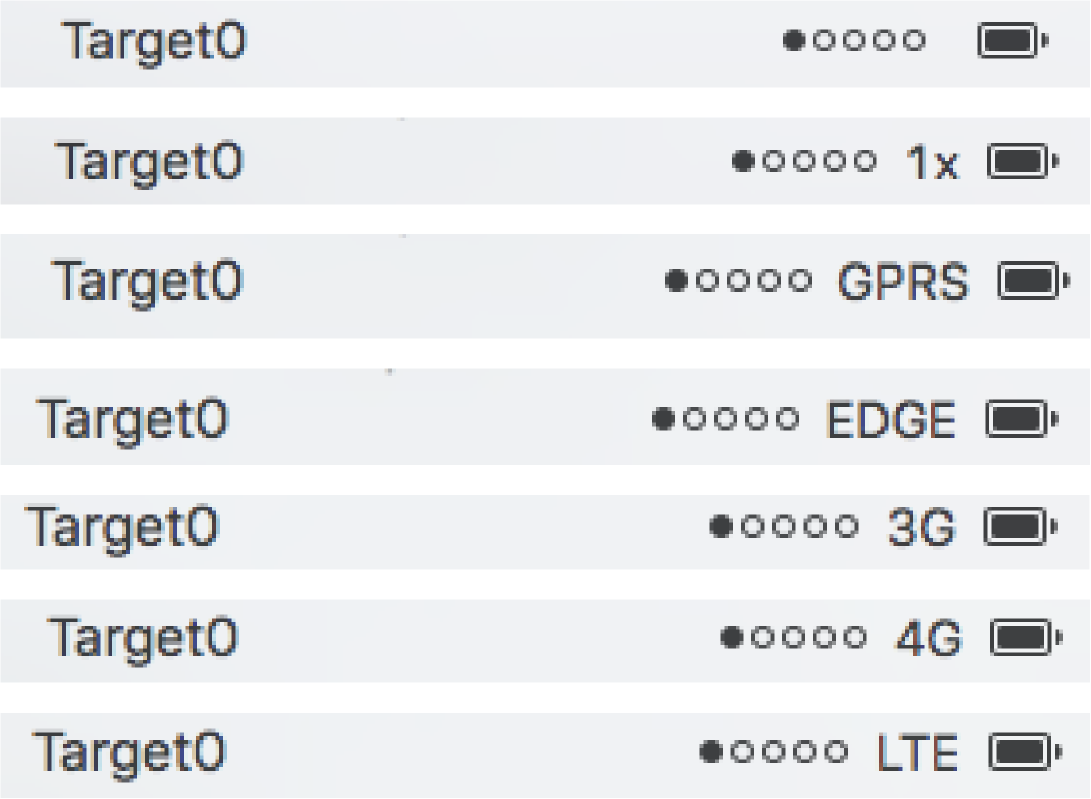
- Type 0x0E, Starts broadcasting after device under same Apple ID sends Tethering Source Presence (type 0x0D)
- Leaks info from broadcast



```
0E    06    01    00    5E        00    06         03
                        |               |          |
                     Battery         Cellular    Signal
                      Level            Type      Quality
                       0x0             0x0         0x0
                        |               |          |
                       0x64            0x7         0x5
```

Spoofing Instant Hotspot

- Replay & Changing bytes is possible
  - Ubertooth, faux slave mode
  - Broadcast with Public MAC
  - Find related device with known MAC



Target0 ●○○○○

Target0 ●○○○○ 1x

Target0 ●○○○○ GPRS

Target0 ●○○○○ EDGE

Target0 ●○○○○ 3G

Target0 ●○○○○ 4G

Target0 ●○○○○ LTE

Breaking MAC Randomization

Overview

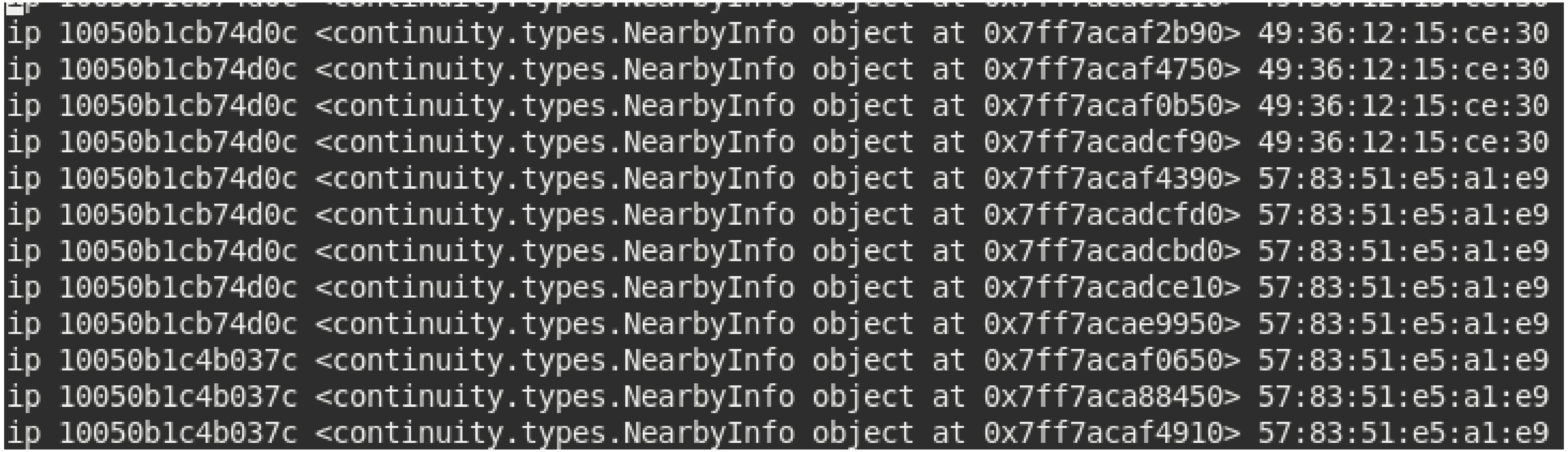
- Our objective
  - To track device regardless of MAC randomization
- Breaking MAC Rotation
  - Nearby
  - Handoff
  - IRK
- Connection between private MAC – public MAC
  - Hotspot

BLE Spec

- BLE Spec recommends rotation per 15 minutes
  - Observed >15 minuted interval

- Is there any other way to track devices?

Nearby

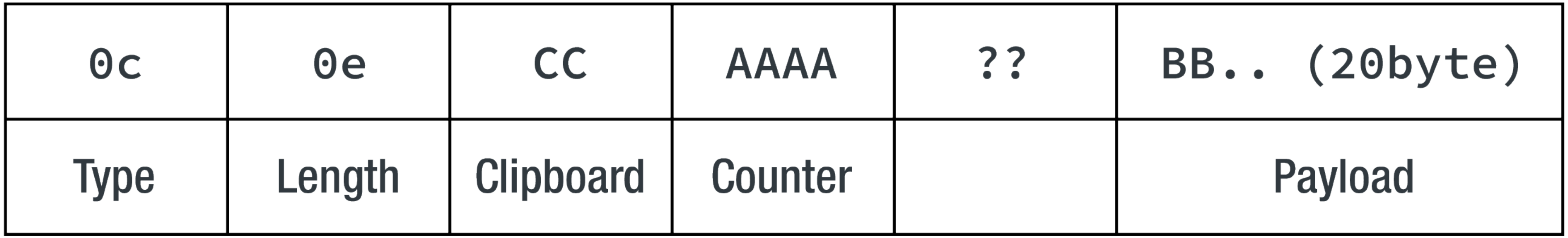


```
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acaf2b90> 49:36:12:15:ce:30
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acaf4750> 49:36:12:15:ce:30
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acaf0b50> 49:36:12:15:ce:30
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acadcf90> 49:36:12:15:ce:30
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acaf4390> 57:83:51:e5:a1:e9
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acadcfd0> 57:83:51:e5:a1:e9
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acadcbd0> 57:83:51:e5:a1:e9
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acadce10> 57:83:51:e5:a1:e9
ip 10050b1cb74d0c <continuity.types.NearbyInfo object at 0x7ff7acae9950> 57:83:51:e5:a1:e9
ip 10050b1c4b037c <continuity.types.NearbyInfo object at 0x7ff7acaf0650> 57:83:51:e5:a1:e9
ip 10050b1c4b037c <continuity.types.NearbyInfo object at 0x7ff7aca88450> 57:83:51:e5:a1:e9
ip 10050b1c4b037c <continuity.types.NearbyInfo object at 0x7ff7acaf4910> 57:83:51:e5:a1:e9
```

- Payload from Nearby is not changed immediately
  - iPad Mini 5th, iOS 12.3.1 & iPhone 7, iOS 12.4.1
  - iPhone 11, iOS 13.2.3
- Track device’s next random MAC with same payload

Handoff

- Move app states between devices seamlessly
- Payload contains App’s identifier (encrypted)



0c	0e	CC	AAAA	??	BB.. (20byte)
Type	Length	Clipboard	Counter		Payload

Handoff - Implementation

- Payload with AES-256-GCM
  - Keys can be sent via P2P or iCloud
  - One key per device

- No GCM tag validation
- IV = Counter

Handoff - Implementation

- Increment counter, +1 per “action”
- Actions
  - Notes, Browsers, Messages
  - Goodbye (when returning to “desktop”)

- <u>Counter will NOT reset between MAC change</u>

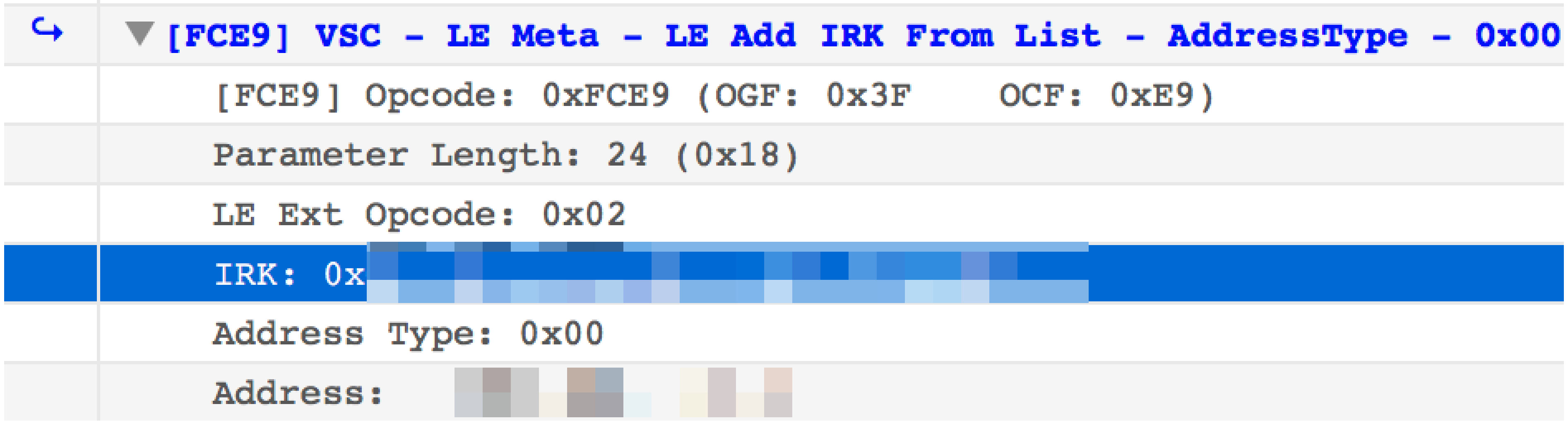
IV Reuse?

- +1 per “action”
- 0x0000 – 0xFFFF

- 50k-ish after 2 years of usage
- Keys rotate when IV reaches 0

IRK Changing

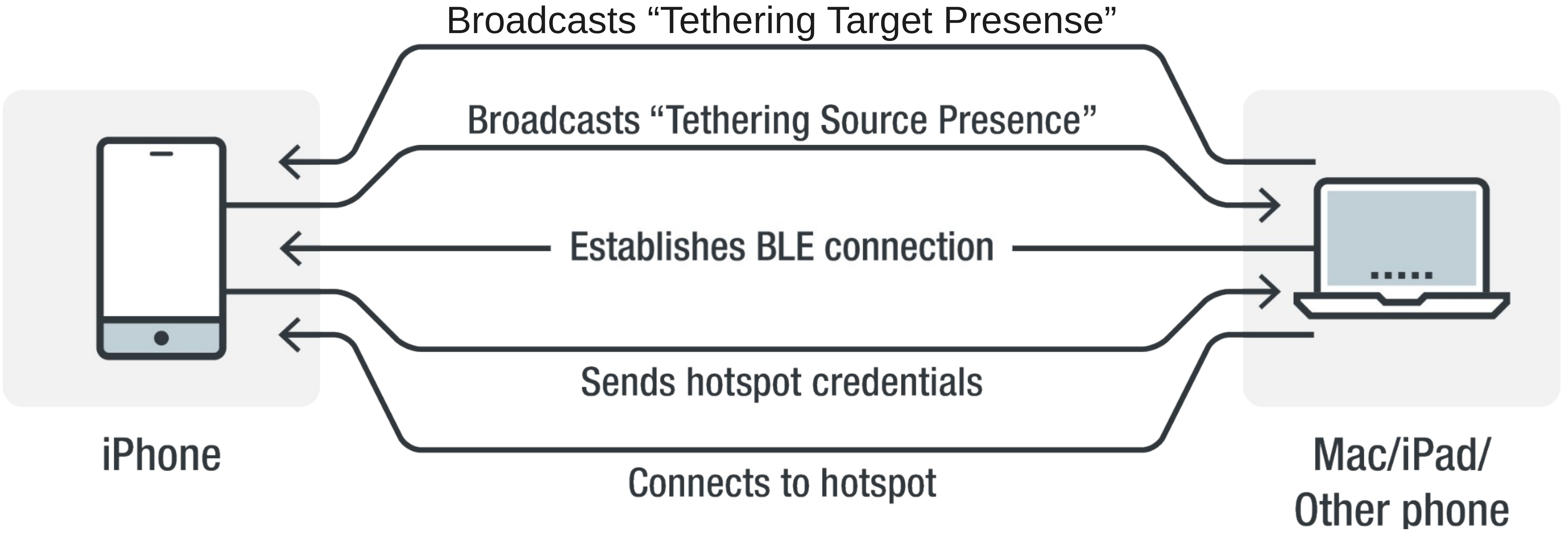
- Not observed at all, even after device reset
- Can be retrieved in Console.app/PacketLogger
- IRK synced to other iDevice



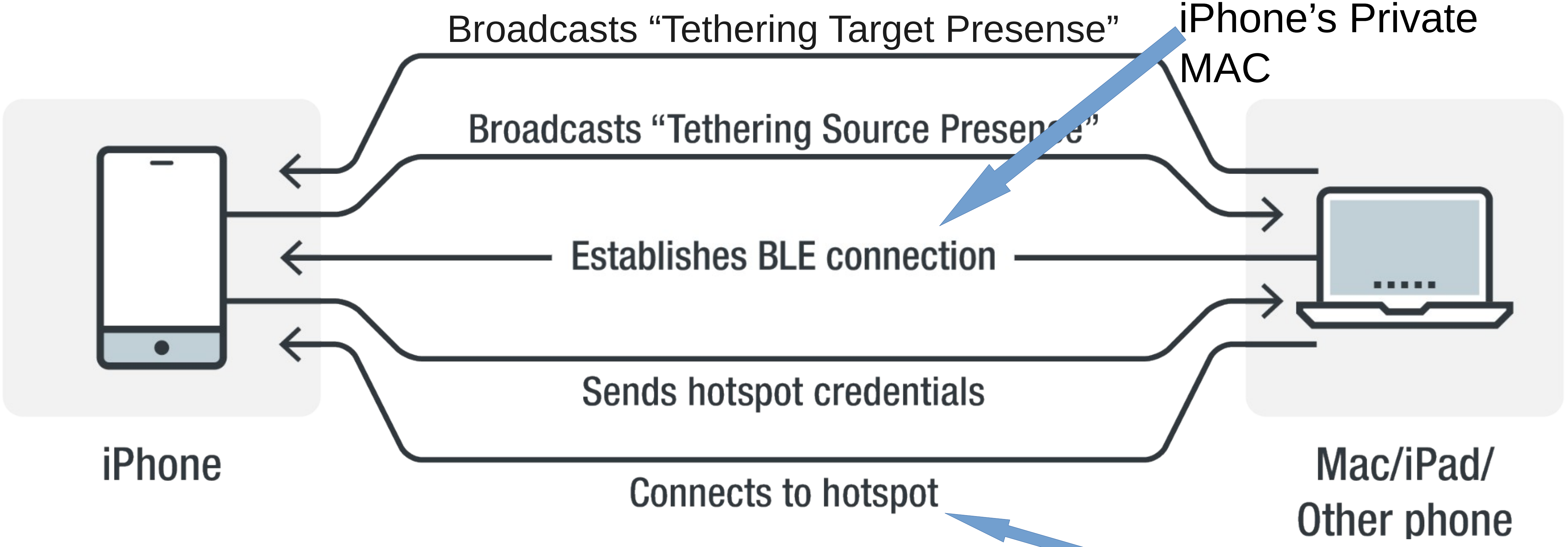
```
↪ ▼ [FCE9] VSC - LE Meta - LE Add IRK From List - AddressType - 0x00
        [FCE9] Opcode: 0xFCE9 (OGF: 0x3F    OCF: 0xE9)
        Parameter Length: 24 (0x18)
        LE Ext Opcode: 0x02
        IRK: 0x[redacted]
        Address Type: 0x00
        Address: [redacted]
```

Connection between private MAC – public MAC

Hotspot Flow


Broadcasts “Tethering Target Presense”
Broadcasts “Tethering Source Presence”
Establishes BLE connection
Sends hotspot credentials
Connects to hotspot
iPhone
Mac/iPad/
Other phone

Hotspot Attack Scenario



Broadcasts “Tethering Target Presense”
Broadcasts “Tethering Source Presence”
Establishes BLE connection
Sends hotspot credentials
Connects to hotspot
iPhone
Mac/iPad/
Other phone

Mac’s Public MAC to iPhone’s Private MAC

Wi-Fi MAC of both

Hotspot Attack Scenario

- Tethering Source Presence appears after Tethering Target
  - Device A is related to device B
  - One Private MAC & Both Public MAC
- Sniffing on both BLE&Wi-Fi
  - Probe Request/Response after BLE connection

Contextual De-Anonymization (Instant Hotspot)

Attack Demo

Practical Considerations

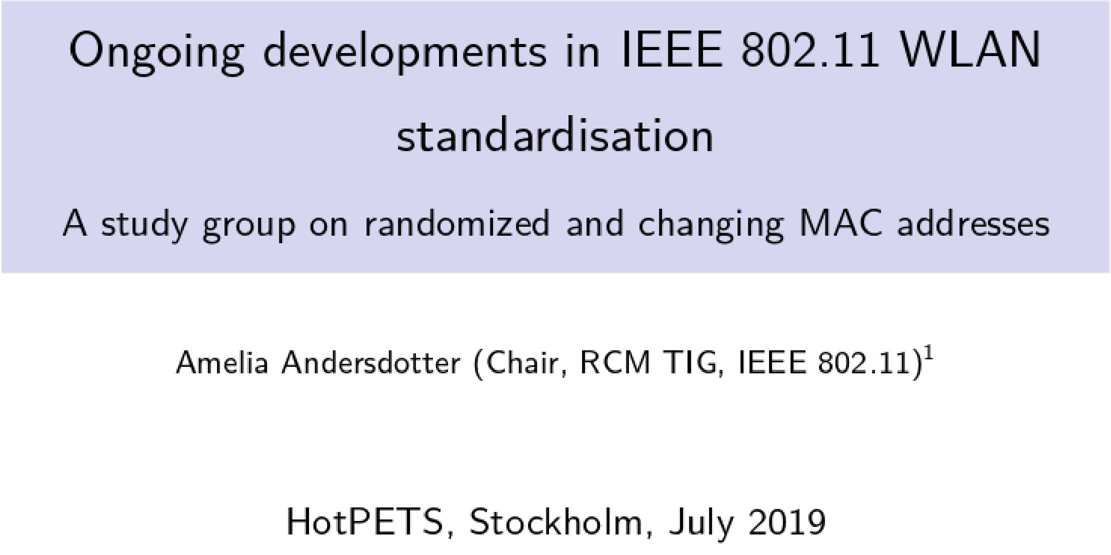
- Encrypt everything
  - Infrastructure are there already
  - Performance issues?
- But still no protection against
  - Attack against iCloud
  - Any compromised iDevice
- Wi-Fi Anonymization

Practical Considerations

- Cannot protect against
  - Attack against iCloud
    - IRK & Public Keys are stored on iCloud
  - Any compromised iDevice
    - IRK & Public Keys is reachable from any iDevice in same apple ID

Practical Considerations

- Wi-Fi Anonymization
  - In draft
  - MAC Randomization in Android Q
  - Management issues?

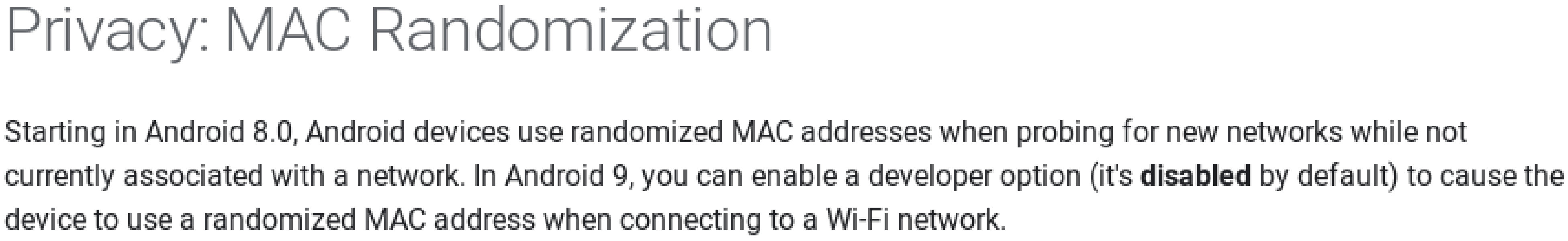


Ongoing developments in IEEE 802.11 WLAN standardisation

A study group on randomized and changing MAC addresses

Amelia Andersdotter (Chair, RCM TIG, IEEE 802.11)[1]

HotPETS, Stockholm, July 2019



Privacy: MAC Randomization

Starting in Android 8.0, Android devices use randomized MAC addresses when probing for new networks while not currently associated with a network. In Android 9, you can enable a developer option (it's **disabled** by default) to cause the device to use a randomized MAC address when connecting to a Wi-Fi network.

Black Hat Sound Bytes

- New approach to iDevice tracking
- Convenience implies degree of privacy hazards
- Review your implementation of new protocols carefully, especially when integrating with another protocol

Thank you!

Ta-Lun Yen
talun_yen@trendmicro.com @evanslify