Proceedings on Privacy Enhancing Technologies ; 2021 (3):164–181

Ellis Fenske*, Dane Brown, Jeremy Martin*, Travis Mayberry*, Peter Ryan, and Erik Rye*

Three Years Later: A Study of MAC Address Randomization In Mobile Devices And When It Succeeds

Abstract: Mobile device manufacturers and operating system developers increasingly deploy MAC address randomization to protect user privacy and prevent adversaries from tracking persistent hardware identifiers. Early MAC address randomization implementations suffered from logic bugs and information leakages that defeated the privacy benefits realized by using temporary, random addresses, allowing devices and users to be tracked in the wild. Recent work either assumes these implementation flaws continue to exist in modern MAC address randomization implementations, or considers only dated software or small numbers of devices.

In this work, we revisit MAC address randomization by performing a cross-sectional study of 160 models of mobile phones, including modern devices released subsequent to previous studies. We tested each of these phones in a lab setting to determine whether it uses randomization, under what conditions it randomizes its MAC address, and whether it mitigates known tracking vulnerabilities.

Our results show that, although very new phones with updated operating systems generally provide a high degree of privacy to their users, there are still many phones in wide use today that do not effectively prevent tracking.

Keywords: MAC randomization, privacy, device identifiers

DOI 10.2478/popets-2021-0042

Received 2020-11-30; revised 2021-03-15; accepted 2021-03-16.

***Corresponding Author: Ellis Fenske:** USNA, E-mail: fenske@usna.edu

Dane Brown: USNA, E-mail: dabrown@usna.edu

***Corresponding Author: Jeremy Martin:** MITRE, E-mail: jbmartin@mitre.org

***Corresponding Author: Travis Mayberry:** USNA, E-mail: mayberry@usna.edu

Peter Ryan: MITRE, E-mail: peterryan@mitre.org

***Corresponding Author: Erik Rye:** CMAND, E-mail: rye@cmand.org

1 Introduction

Although mobile devices and the wireless networks that support them provide a variety of benefits to their users, these networks were not designed with privacy in mind. Wi-Fi radios constantly send *Probe Requests* as a means of scanning for available access points. This has the side effect of announcing their presence and potentially trackable identity information to every nearby eavesdropper. In response to these privacy concerns mobile device manufacturers and software developers deploy a class of techniques called *Media Access Control (MAC) address randomization* for probe requests in which previously persistent network identifiers are randomized to prevent user tracking.

MAC address randomization saw its first major deployment in Apple's iOS version 8.0, released in September of 2014. Other manufacturers followed suit but at a slower and more sporadic pace [23]. In 2017, Martin et al. [21] performed the first study of the effectiveness of Wi-Fi MAC address randomization in the wild and discovered that although Apple had deployed randomization across its lineup of mobile devices, the majority of Android devices did not use any randomization. In addition, several weaknesses in the implementations of randomization were highlighted, including the novel discovery that Request to Send (RTS) and Clear to Send (CTS) frames could leak the MAC address of all known devices at the time.

Recent research has continued to use these identified weaknesses to make broad claims regarding the ineffectiveness of MAC address randomization [12, 28, 31]. However, [21] also made recommendations that could be implemented by manufacturers and standards bodies to realize the potential privacy assurances of MAC randomization. Since then, there have been significant changes in the deployment of randomization that make it uncertain how vulnerable modern devices are to being tracked. More manufacturers include the ability to randomize MAC addresses. Previously discovered issues have been fixed or mitigated. Interestingly, a new type of randomization is being used by some devices called *post-association randomization* [9]. In contrast with normal *pre-association* MAC address randomization, wherein a device uses a random

MAC address only before it associates with an Access Point (AP), devices that use post-association randomization also use a random address for all transmitted frames even after they are associated with an AP.

Unfortunately, most implementations of MAC address randomization are undocumented and their source code is proprietary. This makes it difficult for researchers and users of the devices themselves to understand what privacy guarantees, if any, the devices provide. In order to provide an accurate assessment of the modern state of MAC address randomization as well as a case study in how privacy technologies in mobile devices are adopted and deployed over time, we conducted a large-scale study to examine trends in MAC address randomization.

We obtained and tested 160 distinct mobile phones manufactured between 2012-2020 in a laboratory setting to assess trends over time. Our device set includes representatives from 18 manufacturers, including brands with large market penetration like Apple, Samsung, Huawei, Xiaomi, and Motorola. From these tests, we classify devices according to whether they use MAC address randomization, from what space the random addresses are generated, and how resilient they are against known de-anonymization techniques [18, 21, 24].

We also provide the first analysis of privacy achieved through post-association MAC address randomization. We identify from our data set a small number of devices which currently allow post-association randomization and examine the effectiveness of these techniques.

2 Background and Related Work

2.1 Previous work

Previous studies highlighted several shortcomings in the implementation of MAC address randomization. The goal of this work is to evaluate the current state of the art with respect to these vulnerabilities; we first discuss our intended adversarial model then briefly recall the vulnerabilities we investigate.

2.2 Adversarial Model

In this work we investigate the privacy implications of currently deployed MAC address randomization technology. Our threat model encompasses adversaries able to interact with a mobile device wirelessly without physical access or a software presence on the device. The adversary's goal is to uniquely identify and/or track a single device over time and from location to location, based on its wireless emissions.

This captures an upper bound on privacy; users choose which apps to install, but they cannot alter the manner in which their device advertises itself and interacts with other devices at the link layer. Any further app-based tracking can compound privacy issues, but are inherently orthogonal to our model.

In addition to the contents of Wi-Fi messages, it may also be interesting to consider metadata about the transmissions: transmit power, inter-frame arrival time [24], etc. We chose not to include that in this study in order to limit it to a reasonable scope. We leave this to future work.

2.2.1 Inconsistent Use of Randomization

The prior work conducted by Martin et al. [21] in early 2017 concluded that while Apple had recently introduced MAC address randomization in iOS 8, the vast majority of Android phones did not employ any form of MAC address randomization. Other Android phones that did employ randomization occasionally reverted to broadcasting their hardware MAC address for a small percentage of probe requests, even while unassociated. Between these two issues, over 96% of Android phones were ineffective at hiding their hardware MAC address.

Furthermore, implementation details differed even between devices that perform MAC randomization. For example, iOS devices randomized the entire MAC address, with the exception of the two functional local/universal and unicast/multicast bits. In contrast, many Android devices used a fixed 3-byte Company Identifier (CID) prefix and only randomized the last 3 bytes. Devices also differed in how often they rotated to new random addresses and how often they sent probe requests containing these addresses. These inconsistencies provide an adversary with mechanisms to isolate and identify unique devices based on these distinct behaviors.

2.2.2 Persistent Identifiers

In addition to the MAC address, other probe request fields can be used as persistent identifiers [21, 29]. In particular, devices that support Wi-Fi Protected Setup (WPS) transmit a Universally Unique IDentifier-Enrollee (UUID-E) in probe requests, which is static and persistent even when the random MAC address changes. The UUID-E is deterministically derived from the device's MAC address [8, 29], allowing an attacker to pre-compute UUID-Es for future MAC address de-anonymization. This technique has been used to recover a device's MAC address in public packet capture repositories despite source MAC anonymization [20].

2.2.3 Sequence Numbers

Probe requests also include a 12-bit sequence number field, which is typically initialized at 0 and incremented with each successive probe request sent, modulo 2^{12}. Prior work [21, 29] demonstrated that sequence number monotonicity decreases the efficacy of MAC address randomization, as it allows a nearby adversary to link random MAC address changes by comparing sequence number values.

For example, an attacker that observes a probe request with MAC address, sequence number tuple (MAC_A, seq), followed shortly thereafter by $(MAC_B, seq + 1)$ can be relatively certain that these two observations represent the same device that has "rotated" random MAC addresses. While this technique is ineffective if the device goes unobserved for significant periods of time, it allows an adversary with sustained surveillance of a target device to correlate its random MAC addresses.

2.2.4 Device Signatures

Gentry and Pennarun [15] describe a mechanism to passively identify the device model of Wi-Fi clients by analyzing the Information Elements (IEs) they include when transmitting probe request frames. Following a similar model as [6], which identifies device models by analyzing the Dynamic Host Configuration Protocol (DHCP) Request message options, [15] introduces the idea of a "device signature" – the IEs and their order that a client transmits in its probe requests. For example, a probe request containing the ordered IEs "SSID" (0), "Supported Data Rates" (1), and "Extended Supported Data Rates" (50) equates to a device signature of "0,1,50".

The authors report that approximately 60% of device models could be identified by signature in two real-world case studies, note that many devices exhibit multiple signatures, and discover that Operating System (OS) updates can change the signature emitted by devices. We re-examine, validate, and update these results in this work. In a similar vein, Matte and Cunche [22] develop a tool called "Panoptiphone" to evaluate the uniqueness of a device's IE fingerprint.

2.2.5 Active Attacks

In [29], Vanhoef *et al.* study the effect of the Access Network Query Protocol (ANQP) on MAC address randomization. This protocol is used to identify, query, and select Hotspot 2.0 networks [30], typically without user interaction. The authors establish Hotspot 2.0 honeypots in an attempt to elicit the hardware MAC addresses from devices that randomize their source MAC address when not associated with an AP. The authors found that Windows and Linux devices used their hardware MAC address when sending ANQP requests when in an unassociated state, while verifying that Apple devices continue to use a random identifier.

We also reevaluate a novel privacy attack described in [21], in which the authors are able to successfully confirm the presence of 100% of devices they study, spanning all major manufacturers and then-current OS versions. In this attack, an adversary with *a priori* knowledge of a victim's hardware MAC address they wish to track can determine whether that device is nearby by sending RTS frames to that MAC address. The authors of [21] discovered that a nearby device will respond with a CTS frame, even if it is unassociated and sending probe requests with a random source MAC. Because the CTS response to the RTS query confirms the presence of the hardware MAC address in question, this privacy attack negates the value of using a random source MAC address while unassociated. In order to determine whether device manufacturers have addressed this potential source of location privacy disclosure, we conduct the same experiments as [21] using modern hardware and OSs.

2.3 Limitations of Previous Studies

The previous study by Martin et al. [21] was limited by the fact that it was based only on data captured in open, public spaces ("wild data") and was not performed under laboratory conditions. In [21], the authors identified device models and randomization behaviors using WPS fields contained in management frames. Because each device was observed for a limited time period in an uncontrolled Radio Frequency (RF) environment being used in an unknown manner, nuanced device behavior was impossible to determine.

Interestingly, as we report in §4.3, almost all devices produced in the last few years do not send WPS elements, meaning that the methodology of Martin et al. [21] would not be possible today.

Because Hotspot 2.0 deployments have drastically increased [4] since the publication of [29], as well as the fact that prior work did not study Android devices, we reexamine this potential vector for hardware MAC address disclosure in our study.

2.4 IEEE Standards

Pre-association MAC address randomization was formally specified by the IEEE Standards Association Standards Board in June 2018 with the approval of the 802.11aq Pre-Association Service Discovery Task Group amendment to the 802.11-2016 standard [3]. The Randomized and Changing MAC addresses Topic Interest Group (RCM TIG) was

established in March 2019. Although the RCM TIG has begun to evaluate and address post-association MAC address randomization, there is no formal standard dictating *how* post-association MAC address randomization should be implemented. Each manufacturer is free to decide how to implement this technique on their own.

3 Methodology

We performed 647 individual passive measurements of pre-association probe request behavior on our corpus of 264 mobile devices, spanning 160 individual device models selected for diversity with respect to manufacturer, operating system, and release date. We tested multiple devices of the same model when possible to ensure confidence and robustness of our results. Additionally, to assess differences in behavior due to changes in OS versions, we tested the same device before and after applying updates. To assess the more time-consuming active attacks, we selected representative samples from our corpus. These samples are weighted heavily towards modern devices that have not previously been studied. Because some devices do not support ANQP, we were able to complete fewer of these tests. We tested 62 and 118 distinct devices for ANQP and RTS vulnerabilities, respectively.

The manual nature of our methodology likely introduces some error into our data set. However, the size and relative consistency of our results give us strong confidence in our conclusions about general trends with respect to behavioral characteristics. Further, we note that the privacy technology we wish to assess is explicitly designed to prevent identification of any given wireless signal accurately to its source device. Since many devices deploy these technologies successfully, individual RF-shielded tests like those we performed are required to accurately assess individual device behavior.

The corpus, while comprehensive, is not representative of the distribution of devices existing in the real world. This is in part because the true distribution of currently active mobile devices is not precisely known, even at a regional granularity. Instead, we use the experiments performed on our corpus to determine ground truth for hundreds of individual devices, including but not limited to those we have reason to believe are the most widely deployed in practice based on data collected from web requests [25].

The breakdown of the individual tests by device release year, manufacturer, and OS version is presented in Figure 1. We select a single canonical representative for all figures in the following for every unique device-model OS-version pair, not displaying tests which were identical repeats of previous tests for double-checking and consistency purposes. The results and analysis are presented in detail in §4.

To further validate the results of these tests in practice and collect real-world data on the probe requests that appear in the wild, we also performed large-scale untargeted collection in public places, primarily in the mid-Atlantic region of the United States, discussed further in §3.2.

3.1 Pre-association Experiments

We performed the following individual tests to assess pre-association behavior.

Setup: All phones were factory reset before testing. Since unassociated Wi-Fi devices rotate through channels while sending probe requests, we connected three Wi-Fi capture cards tuned to non-overlapping channels 1, 6, and 11 of the 2.4 GHz band to capture all of the requests.

The experiment was broken down into one test for each of two device states, *action* and *idle*:

Action/Idle: We chose to gather probe requests from both the action (device in active use) and idle (device unused and eventually locked) states due to annotated Wi-Fi behavioral differences described by OS vendors while a device is in active use, when the screen is off or on, and whether a device is connected to a network [2]. We focused on device behavior in an unassociated state; therefore, our testing methodology focuses on the differences in behavior observed from the action versus idle states. To replicate the action state, we had a small robot with a stylus inside the box which maintained regular, periodic contact with the screen of the device. This kept the device from locking and emulated consistent active use. For the idle state, we placed the device in the enclosure and did not touch it, letting the screen lock naturally, emulating the behavior of a device in a pocket or otherwise not in use.

In each test the device was left inside an RF-shielded enclosure until either we collectively received 200 probe requests or 20 minutes elapsed. Earlier informal experiments led us to determine that 200 probe requests would be enough to characterize the pre-association probe requests from a device.

OS Updates: After the experiments were concluded on these factory-reset devices, we attempted to update their OS to the most recent version possible, and repeated the tests described above. We also examined, for each OS, device settings that could be modified or reset by an update that could reasonably affect probe request behavior and found none. When a device could be updated, we consider both tests, one on each OS, as distinct independent tests throughout all of the analysis that follows.

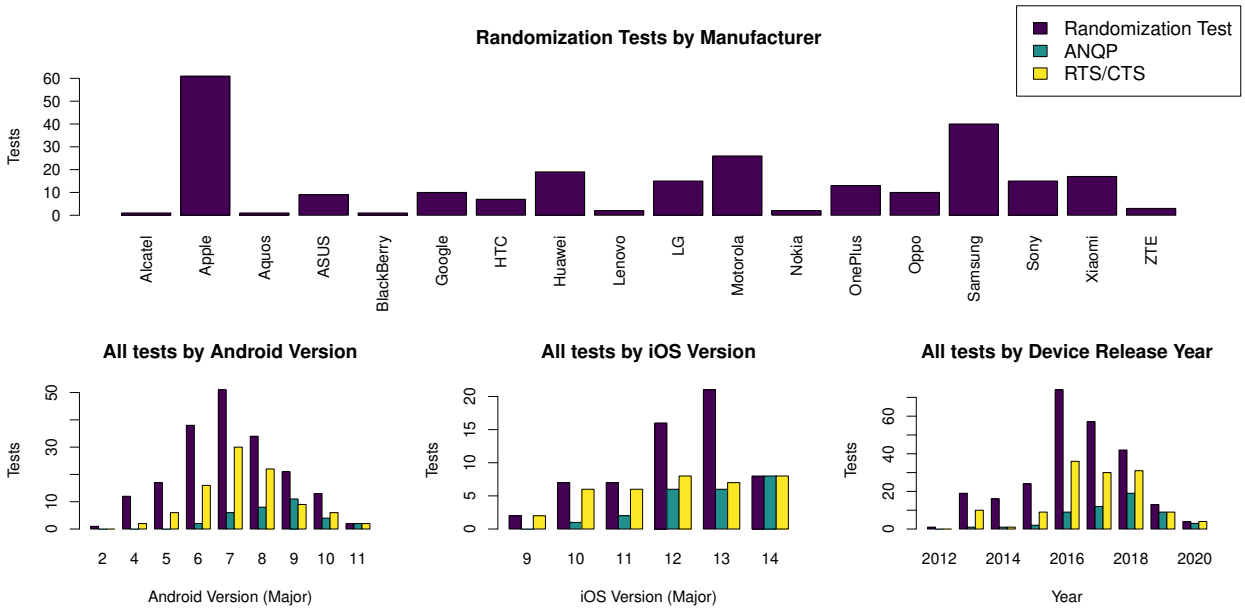

Fig. 1. Corpus Tests by OS Version, Release Year, Manufacturer

3.2 Wild Capture Data

To validate our experiments and assess the state of probe requests in the wild, we collected probe requests en masse from a variety of locations, primarily focused on the mid-Atlantic region of the United States. We created Wi-Fi collection kits, each with a Raspberry Pi and an Alfa AWUS036NEH NIC connected to a 7 dBi omni-directional antenna. The Pi was configured to start a Wi-Fi collection of management frames on power-up on channel 6 (2437 MHz), chunking in 50 MB files.

When connected to the Internet, the Pi would auto-upload collected packet captures to a server for cleaning (*e.g.* omission of malformed frames) and analysis. Using these kits, we collected 20 million probe requests between 2019 and 2020 for the purpose of assessing whether our individual enclosed-environment experiments reasonably reflected the state of privacy technologies in the wild.

In addition, we compared this dataset to the data recovered by a similar collection effort from 2016 with 60 million probe requests to assess large-scale trends over time. We made no attempt to ensure these samples were representative, and in particular cannot determine between mobile devices and other devices that send probe requests: laptops, Internet of Things devices, etc. Devices consistently nearby the collection kits (for example, owned by a collector) are overrepresented. Given these problems, many of which we consider to be inherent to bulk wild data collection of devices deploying privacy technology, we are careful to limit the conclusions we draw from this data. Nonetheless, we find the large wild corpus very useful to confirm the existence of certain phenomena in the real world, and we discuss our results further in §4.1.

3.3 Ethical Considerations

In this study, we capture 802.11 frames from devices owned by our research group in a laboratory setting, as well as "wild" data that was primarily captured near our institution in a major metropolitan area. The data captured outside the laboratory was collected in an entirely passive manner; at no time did we stimulate a device not owned by our group, alter the normal flow of traffic, or attempt to decrypt any encrypted data. We consulted with our Institutional Review Board (IRB) in order to obtain a determination regarding the information collected, and whether this information contains data "about whom" or "about what". In this study, we limit the frames we consider from our outside collection corpus to 802.11 management frames. Management frames do not contain Personally Identifiable Information (PII), and although we obtain MAC addresses from 802.11 devices, there is no reasonable way to map these addresses back to an individual. Because the purpose of our study is to evaluate the efficacy of MAC address randomization, humans are incidental to the experiment, and our work presents no expectation of harm. As such, our IRB determined that our experiment was not human subject research.

3.4 Active Attack Experiments

Vanhoef *et al.* demonstrated in [29] that ANQP, a protocol designed to solicit authentication and network information from APs by wireless devices, also leaked the hardware MAC addresses of devices in certain situations. ANQP is used by mobile devices, in particular, to determine whether or not to attempt to authenticate to Hotspot 2.0 networks based on their cellular service provider, if the service provider supports data offloading via Hotspot 2.0. These devices are provisioned by the cellular carrier and device manufacturer to automatically query and attempt to authenticate to trusted 802.11 networks. As in [29], we establish a Hotspot 2.0 honeypot and observe the ANQP requests from our test devices in order to ascertain their vulnerability to this type of privacy attack. While prior work studied only Apple mobile devices (as well as Linux and Windows computers), we extend this work to Hotspot 2.0-capable Android devices, in order to evaluate the extent to which it preserves the privacy of the hardware MAC address.

Martin *et al.* introduced a novel attack in [21] in which a device using random MAC addresses while probing for APs could be induced to respond to an RTS frame sent to its hardware MAC address. In fact, *every* device the study tested sent a CTS frame in response to an RTS with the correct hardware MAC address, indicating a pervasive privacy problem. Although the MAC address space is too large to feasibly brute force and discover the hardware MAC address of every device within transmission range, an attacker with *a priori* knowledge of a device's MAC address that they wish to track can detect whether this device is within transmitter range. We reevaluate the effectiveness of this attack on modern devices with current OSs in a laboratory environment in order to determine whether this tracking vulnerability has been mitigated.

Our test protocol involves sending 802.11 RTS frames to the hardware MAC address of each test device over a period of two minutes on a single 802.11 2.4 GHz channel. The test devices are not associated to any network, and thus, if they are capable of pre-association MAC address randomization, they are sending probe requests with a random source MAC address. Because CTS frames sent in response to RTS messages do not contain a source MAC address but rather only the destination that is cleared to send its traffic, we declare a device vulnerable to RTS stimulation attacks if we receive CTS frames addressed to our transmitter.

Scheme	% (2016)	% (2019-2020)
Hardware Addr	**81.6%**	**56.0%**
46-bit (Apple)	**10.7%**	**25.6%**
46-bit (Google)	**0.0%**	**0.7%**
46-bit (Samsung)	**0.0%**	**1.5%**
46-bit (Other)	**4.1%**	**7.5%**
`DA:A1:19`	**0.4%**	**6.6%**
`02`	**0%**	**0.2%**
Motorola	**3%**	**1.7%**
`92:68:C3`	**0.2%**	**0.4%**

Table 1. Probe Requests in the Wild

4 Passive Analysis

4.1 Wild Capture Data

Martin et al. [21] primarily used large-scale passive collection of probe requests to classify and identify probe request behavior. While in past studies, many probe requests included WPS information, providing ground truth with which to analyze these frames collected from the wild, we found that of the frames sent with random addresses, less than 0.5% included WPS information. This means WPS information has been almost entirely phased out of the mobile device ecosystem, and severely limits the conclusions we may draw from our captures in the wild, since we cannot necessarily link multiple randomized addresses to the same device. In addition, this collection methodology does not allow researchers to identify a device shifting its behavior over time or in response to certain events, which we observed in our controlled experiments, and cannot determine relative frequency of probe requests between devices with random addresses. Finally, in some cases we cannot know from in-the-wild captures whether or not a probe request is using a random address, since some devices select random addresses without the Universal/Local (U/L) bit set, meaning their random addresses are indistinguishable from legitimately assigned hardware addresses.

We present our results in the form of percentages of the total probe requests collected, organized by randomization scheme in Table 1. For probe requests with local addresses, we categorized each based on its prefix according to the schemes we observed and in the case of 46-bit randomization, made an attempt to classify the manufacturer of the source device using signatures as described in §4.5. We further note that every scheme with the local bit set can appear at random as an address selected by the 46-bit randomization scheme. However, the only frames which can be confused with non-negligible frequency with a 46-bit randomization scheme are from the `02` scheme, which only appears on a select number of Samsung devices in our experiments, which are identifiable by signature. Therefore, we only classify as `02` in Table 1 frames with

02 addresses and Samsung-specific signatures, noting that probe requests from Samsung devices using 46-bit randomization that randomly select an address with first byte 02 are incorrectly classified.

We caution the reader against drawing conclusions from the results in Table 1 since not only is our corpus a non-representative sample of behavior in the wild, a number of factors mentioned above complicate conclusions drawn by any study based on modern in-the-wild capture data in 2020. Finally, since our intent is to measure and identify the output of a privacy-preserving technology, as the technology becomes more effective our attempts to assess and measure behavior in the real world should become less effective by design; preventing an in-the-wild approach to assessment is a natural side effect of effective privacy technology, if not its intended result. Due to these complications, we use these results primarily to justify the necessity of the individual, isolated, per-device experiments described in §3.1.

Given the experimental data, we used a mix of custom software and manual inspection to classify behavior of devices both tested individually from the corpus and from the collection of wild packet captures. While we initially expected to observe clear delineations of privacy protection introduction by OSs and device manufacturers, discovering such straightforward demarcations often proved elusive. In our data, contemporaneous mobile devices produced by a single manufacturer and running the same OS version often exhibited different privacy behaviors. Therefore, we highlight the fragmentation and diversity of behaviors in the mobile device ecosystem, and identify general trends both by manufacturer and over time.

4.2 MAC Address Randomization

As in previous studies [21, 29], we observed several distinct MAC address randomization behaviors from the devices in our experiments. These behaviors varied across devices and also across OS versions on the same devices.

Structurally, randomized MAC addresses take different forms based upon the implementation – some parts of the MAC address may be fixed, with others chosen at random. For instance, some devices use a fixed CID while randomizing the remaining 24 bits. Other devices use random addresses without the U/L bit set, indicating that the MAC address is drawn from an Institute of Electrical and Electronics Engineers (IEEE)-allocated Organizationally Unique Identifier (OUI) and globally-unique, when it is in fact not [17]. We refer to the bitwise structure of randomized addresses for a device as *Randomization Schemes*.

On a behavioral level, not all devices deploy randomization consistently. We observed devices that use randomized addresses only while in the idle state, and devices that occasionally "cycle through" their hardware address in between random ones. Randomization inconsistencies are discussed further in §4.2.2. We refer to these behavioral classifications as a device's *Randomization Type*.

4.2.1 Randomization Schemes

We observed several distinct randomization schemes throughout our experimentation. From a privacy perspective, no particular scheme is meaningfully more effective than any other if deployed consistently, since while some devices randomize more bits of the address than others, all randomize at least the final 24 bits. We observed no unique statistical properties of the randomization besides the determination of which bits were fixed and which were random; bits are always drawn uniformly at random under any scheme. Different randomization schemes allow for limited device profiling, particularly if a scheme is used for few devices, and this wide array of schemes characterizes the fragmentation among approaches manufacturers have taken towards providing privacy.

Hardware MAC Address. Though this is not properly a randomization scheme (as no randomization occurs), we include it here for completeness. The standard behavior for non-Apple devices released before 2015 was to simply send all Wi-Fi messages including probe requests with the device's hardware MAC address. Surprisingly, many modern devices still exhibit this behavior. We observed it from devices released as recently as 2018, and on a single device running Android 9. Many devices on Android 7 and 8 use this scheme. We were not able to identify any devices using only this scheme with Android 10 or 11, though this may be due to the release of Android 10 being phased in first to modern or flagship devices which are more likely to implement a randomization scheme.

46-bit Randomization. Many devices deploy so-called 46-bit randomization, wherein all bits of the address are random except for the two least-significant bits of the first byte, which are required to be fixed by [17]. We observed this scheme in all Apple devices, which is a fact well-understood in literature. Furthermore, modern Android devices use 46-bit randomization: nearly every device we observed on Android 10 and 11 uses this scheme, as well as around half of the devices we tested on Android 9.

DA:A1:19. Many Android devices fix a Google CID, DA:A1:19, and randomize the remaining half of the MAC address. We observed this behavior in both modern and legacy devices running Android 6.1.1 through Android 9, from a variety of manufacturers. Although DA:A1:19 is a CID belonging to Google, we observe that modern

Google devices consistently use 46-bit randomization on OS releases 8 and above, meaning that this scheme is predominantly used by non-Google Android manufacturers.

Motorola. Motorola devices have distinct randomization schemes compared to Android devices produced by other manufacturers. Of the 21 Motorola devices we tested, only one effectively applied randomization in the action state, the Moto Z 4th gen, which deployed 46-bit randomization in action and idle states. Motorola devices select addresses at random within their own assigned OUI. They usually also rotate through the device's actual hardware MAC address alongside the random addresses, as identified in previous work. This means that these devices cycle through random MAC addresses which all appear as universal addresses (*i.e.* the U/L bit in the address is not set).

`92:68:C3`. Consistent with previous work [21], we observed the Motorola Nexus 6 as the only device using this scheme, which fixes the first 3 bytes, randomizing the remaining 3 bytes, similar to the `DA:A1:19` scheme above.

`02`. A few Samsung devices fix the first byte of the MAC address as `02`, then randomize the remaining 40 bits on Android versions 8 and 9. However, when these devices were upgraded to Android 10, where possible, their scheme switched to 46-bit randomization. We observed this behavior from the Galaxy J7, Galaxy J7 Prime, Galaxy Note 9, Galaxy S9 and S9+, and Galaxy S10 and S10+. However, we emphasize that many similar Samsung devices on the same OS versions do not use this scheme: the Galaxy J7 Max, released in the same year as the J7 devices above and on Android 8.1.0, uses its hardware MAC address, as does the Galaxy S8, even on Android 9. Since this behavior has been phased out in Android 10 and is only used by a small number of devices on specific OSs, this randomization scheme provides a very precise device fingerprint.

4.2.2 Inconsistent Use of Randomization

As reported in previous studies, we observed many devices that randomize MAC addresses, will at some point "rotate through" their hardware MAC address. We observed this behavior in both the action and idle states in Motorola devices and those using the `DA:A1:19` scheme. In many cases, the devices used their hardware MAC address frequently, every few seconds, while other devices properly randomized their addresses for long periods of time (30-40 minutes) before using the hardware MAC address (primarily Motorola devices) in a few frames.

Disabling Wi-Fi: Although it is widely known that disabling Wi-Fi on Android and iOS devices does not prevent all Wi-Fi interactions (e.g. devices can still survey nearby APs for location information), we did not see a significant number of devices transmitting probe requests with Wi-Fi disabled. We tested the devices in our corpus using the same methodology described in §3 but with Wi-Fi switched off in the device settings. Under these circumstances, for example, the Sony Xperia X Compact sent probe requests using its hardware address, and the 4th Generation Motorola Moto Z sent probe requests with random addresses. Only a few of the devices in our corpus sent any probe requests at all with Wi-Fi disabled, all of which were Android devices. This means that in most cases, although a user cannot do anything to directly solve the MAC address randomization issues we identify in this work, they can disable Wi-Fi entirely to mitigate them temporarily.

4.2.3 Idle vs Action State

We observed 11 Huawei, LG, Motorola, OnePlus, and Sony devices randomizing consistently but only while in the idle state. Notably, these devices used their hardware MAC address at least some of the time while in the action state.

In particular, we highlight among this group the 7th Generation Motorola Moto G on Android 10 and 9, marking it as the only one of our ten Android 10 devices that did not consistently deploy 46-bit randomization at all times. All other devices exhibiting idle-only randomization were running Android 6 and 7. The devices in this class that could be upgraded to Android 8 randomized properly when this update was applied, with the exception of the aforementioned Motorola device.

We note that while it is known that many Android devices cycle through their hardware address while deploying the Motorola and `DA:A1:19` schemes [21], many devices successfully deploy randomization in the idle state without ever using their hardware address. This is a surprising behavior characteristic that previous work could not identify without individual controlled tests of these devices.

4.2.4 OS Updates

We were able to apply major OS version updates to 51 devices from our corpus. With respect to randomization type, 13 of these improved with an OS update, one degraded, and the remaining 37 behaved similarly before and after the update was applied. Only four devices, all Samsung, changed randomization scheme, from `02` to a 46-bit scheme after the update, randomizing consistently regardless of scheme.

Of the 13 that improved, ten devices randomized consistently after all updates were applied, while three improved marginally, beginning to randomize their addresses inconsistently after the updates, whereas with the previous OS version they used their hardware address.

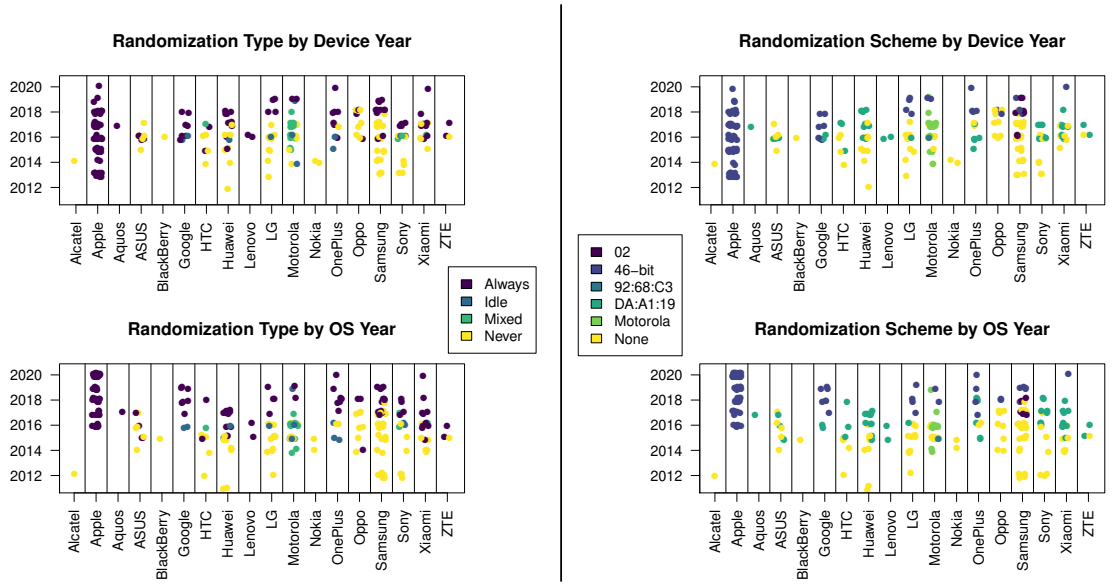

Fig. 2. Randomization Scheme and Type, by Manufacturer

Finally, we identified one model whose randomization scheme degraded with an OS update. The two Asus ZenFone 3 devices we tested used the `DA:A1:19` scheme on Android versions 6.1.1 and 7, but when upgraded to Android 8, used their hardware address.

4.2.5 Device Chipsets

In an attempt to explain dissimilar behavior from similar devices, we annotated the device chipset for each device in our corpus and considered whether hardware differences could plausibly explain the inconsistent behavior among similar devices, or within a single device over time. In some cases we found the chipset could plausibly explain device behavior; however, chipset role in randomization schemes remains ambiguous and unconfirmed.

Discrepancies may be explained better by product line: the Samsung S and J series devices that deploy randomization use the `02` scheme on Android 8 and 9, while the A series devices we tested uses a 46-bit scheme on Android 9. However, in many cases the device behavior remains unpredictable even with knowledge of the low-level chipset information: there remain some sets of devices that share a manufacturer, device release year, OS version, and chipset, but deploy different randomization schemes. For example, the Sony Xperia XZ and Sony Xperia X Performance were both released in 2016 and use the Snapdragon 820 chipset, but one deploys the `DA:A1:19` scheme in the idle state only, and the other uses its hardware MAC address, both on Android 7.1.1.

4.2.6 Results

To characterize inconsistent behavior (regardless of the randomization scheme in use) we define a *randomization type* to be a characterization of when, and how consistently, randomization is deployed, falling into one of four categories:

- *Always*, where devices always use randomized addresses while not associated.
- *Idle*, where devices consistently use randomized addresses in the idle state, but use their hardware address intermittently or always during the action state.
- *Mixed*, where devices use their hardware address at least once in each state, but were also observed using randomized addresses.
- *Never*, where devices never use randomized addresses and instead always use their hardware address.

Our results are presented in Figure 2 by manufacturer over time, presenting how both randomization type and scheme changes, both by hardware and software since either can determine device behavior. To present changing hardware over time, we sort device models by the year they were released. We observe some clear trends from the data: Motorola, Samsung, and Apple stand out as distinct, while the remaining (Android) manufacturers follow a general trend of using the hardware address in 2014-2015, a gradual, fragmented transition to `DA:A1:19` around 2015-2016, and a similarly gradual transition towards 46-bit randomization in 2018 and later. Apple devices are consistent throughout, Samsung devices began randomizing recently but have transitioned through the `02` scheme to consistent 46-bit randomization very quickly.

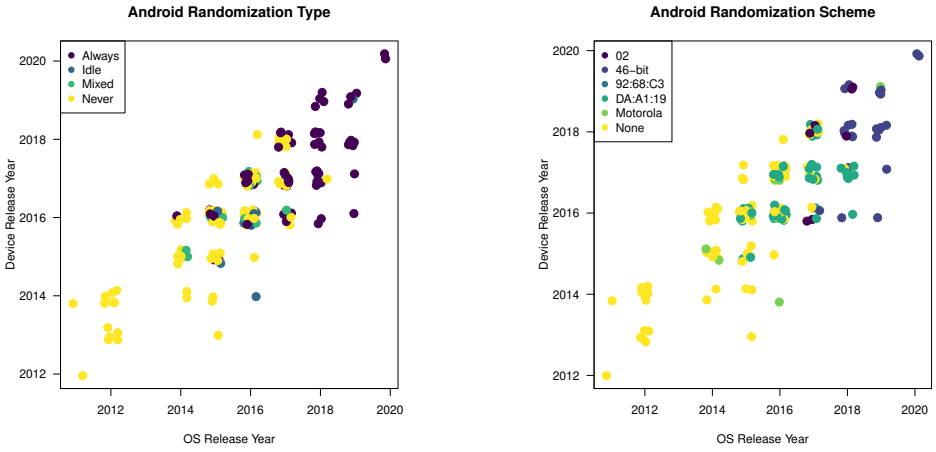

Fig. 3. Android Randomization Trends

The behavior of Motorola devices is unlike those from any other manufacturer: not only do they use their own schemes, but much of the diversity in randomization type comes from Motorola devices, as they provide the bulk of the "mixed" classifications (caused by Motorola randomization schemes where the devices rotate through the hardware address alongside random addresses). Of the two Android 10 Motorola devices we tested, one had typical Motorola device behavior (a Motorola scheme in the idle state and the hardware address in the action state), while the other used 46-bit randomization, indicating Motorola devices may be trending towards 46-bit randomization.

In Figure 3 we present the trends with respect to both the hardware and software release date over time for Android devices. Apple devices are omitted, since they all consistently deploy 46-bit randomization.

4.3 WPS and UUID-E

Although prior work [20, 21, 29] leveraged the WPS IE in probe requests to deanonymize devices using random MAC addresses, it is non-existent in modern OSs and equipment. We observed 24 devices broadcasting WPS information, including the unique persistent UUID-E field, but only from legacy devices. In our corpus WPS is entirely absent since Android 8 and rare in Android 7.

4.4 Sequence Numbers

We observed two primary meaningful characterizations of sequence numbers in our experiments: monotonicity and maximum value. Some devices use strictly monotonic sequence numbers modulo some maximum, while others randomize them over time. Some devices limit their sequence numbers in probe requests to 2047, while others send messages with sequence numbers as high as 4095. Even random sequence numbers increase monotonically in bursts, resetting to a random new starting point periodically.

While IEEE 802.11 standards allow for sequence numbers as high as 4095, many devices restrict these sequence numbers to a maximum of 2047. Sequential sequence numbers reset to 0 after reaching the maximum value. This property provides some fingerprinting capabilities for passive adversaries (for *e.g.*, if a sequence number higher than 2047 is observed, this precludes many devices as a potential source of the signal).

Many modern devices randomize their sequence numbers in tandem with the MAC address to prevent an adversary from trivially linking sequential randomized addresses with the same or a nearby sequence number [5]. This behavior is common in modern devices, but our data and our experiments after applying OS updates suggest a strong correlation between hardware release year and sequence number randomization that does not exist with software version, especially for Apple devices. We speculate that sequence number randomization is in many cases implemented in hardware, and cannot be introduced with OS updates. Sequence number randomization does not appear to depend on the the particular maximum sequence number, but we observed that no modern devices that randomized sequence numbers did not also randomize their MAC address.

We present our sequence number analysis in Figure 4. We drop from the sequence number maximum plots those devices for which the maximum value could not be determined from our experiments (*i.e.* we neither observed a sequence number larger than 2047, or a clear reset after reaching 2047). We note that the maximum sequence number is cleanly determined by manufacturer and hardware release year, with no clear consensus among manufacturers, while sequence number randomization has been adopted consistently by newer devices of all types.

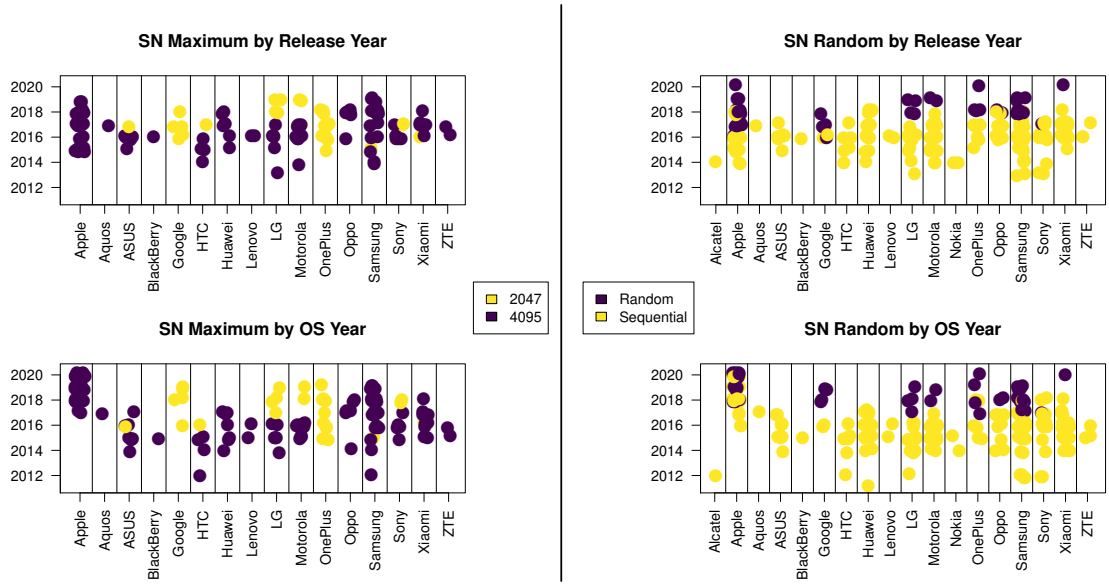

Fig. 4. Sequence Number Behavior, by Manufacturer

4.5 Device Signatures

Probe request device signatures are a privacy concern because they allow a passive adversary to perform device fingerprinting (*e.g.*, determining a device's OS, manufacturer, device model, or radio chipset). Ideally, signatures should be entirely generic, giving adversaries no information about the properties of the device sending a particular probe request. In our observations, we found that device signatures for most devices allow fairly granular device binning, allowing adversaries to determine a small class of devices that could have produced any given probe request. Unlike other privacy features we identify, it is difficult to identify a best practice for making signatures more private other than general heuristics (*e.g.* include fewer information elements) that may impact device performance. According to our observations, no particular method for producing generic signatures has seen wide adoption. We briefly characterize the effectiveness of fingerprint-based device binning based on our experiments for a few major manufacturers, but a fully exhaustive characterization of device signatures and fingerprinting techniques is outside the scope of this work and has been given elsewhere [15, 29]. Briefly, a signature is an ordered list of IEs present in a probe request along with certain bitmasks which have high entropy between distinct device models, and low entropy within probe requests from a single device type, making them robust identifiers [15, 21, 29].

4.5.1 Device Binning

Both Apple and Samsung exhibit the same general pattern with respect to device signatures: every generation of devices uses the same set of signatures, and each generation of devices use signatures distinct from other generations. We provide the device classes and signatures for Apple and Samsung in Table 2. Other manufacturers employ a variety of schemes, and some vary by device within manufacturer. For example, the three OnePlus devices we tested all appeared as uniquely identifiable to the model within our corpus, and all Oppo devices we tested were identifiable as such. Motorola device signature patterns varied by device, with some identifiable and some generic.

4.5.2 Trends

Our focus primarily remains on whether or not there has been an improvement in this area in the mobile device ecosystem from the perspective of privacy technology. In previous work analyzing signatures, authors recommend that signatures should be made generic [21]. Unfortunately, there is no explicit guidance for what information elements should be included in a properly generic signature. This has led to fragmentation. Google, for example, has modified their device signatures in order to minimize the number of information elements with the explicit intention of protecting user privacy [5], reducing entropy in their devices' probe requests. Unfortunately, while the set of information elements (0, 1, and 3) they select is small, that precise set is used almost exclusively by Google devices, allowing manufacturer fingerprinting. Since Google devices are relatively uncommon [25], this could provide passive adversaries with effective tracking capabilities.

The most common signature in our corpus and in the wild, simply consists of three information elements, which appears very generic: (0,1,50). Unfortunately, this signature is characteristic of older Android devices, most of which use their hardware MAC address, so signature-based fingerprinting provides no additional information in this case.

Table 2. Device Classes and Signature Fingerprints

Device Classes	Example Signature
iPhone 6/6+/6S/SE	0,1,50,3,45,127,107,221(0x17f2,10),221(0x50f2,8),221(0x1018,2),htcap:0021,htagg:17,htmcs:000000ff,extcp:0400088400000040
iPhone 7/7+/8/8+/X	0,1,50,3,45,127,107,htcap:402d,htagg:17,htmcs:000000ff,extcp:0000088400000040
iPhone XS/XS Max	0,1,50,3,45,127,221(0x17f2,10),221(0x50f2,8),221(0x1018,2),htcap:402d,htagg:1b,htmcs:000000ff,extcp:0000080400000040
iPhone 11	0,1,50,3,45,127,255,221(0x17f2,10),221(0x50f2,8),221(0x1018,2),htcap:402d,htagg:1b,htmcs:000000ff,extcp:0000080400000040
Galaxy J2 Core/J3/J4	0,1,50,3,45,127,221(0x50f2,8),htcap:016f,htagg:17,htmcs:000000ff,extcp:0000000000000040
Galaxy Note 8/Note 9/S8/S9/S9+	0,1,50,3,45,127,221(0x904c,4),221(0x50f2,8),221(0x1018,2),221(0x904c,92),htcap:11ef,htagg:1b,htmcs:0000ffff,extcp:0000088001400040
Galaxy S10/S10+	0,1,50,3,45,127,255,255,221(0x904c,4),221(0x50f2,8),221(0x1018,2),221(0x506f9a,22),htcap:002d,htagg:1b,htmcs:0000ffff,extcp:00004880014000400021

To develop an objective criterion for a generic signature, we identified the most common information elements appearing in signatures. We analyzed our experimental results and wild capture data, identified all distinct signatures from each, and counted the occurrence of each information element.

Since the results from both datasets gave similar rankings, we took the most common information elements from this list (0, 1, 45, 50), present in 80−90% of all signatures we observed in our lab and in the wild, and considered any signature which was a subset of these elements to be generic. Under this definition, nearly all modern devices we observed used non-generic signatures, marking these identifiers as the one privacy technology for which we have not observed marked improvement.

Inconsistent use of Signatures. As with MAC address randomization, we observed many devices using different (generally shorter) signatures in the idle state. 18 distinct devices consistently used generic signatures in the idle state, even under our strict definition of generic. These are all Android devices from a variety of manufacturers, both new and old. In addition, many devices use many signatures while in a single state. Crucially, to prevent against fingerprinting attacks, device signatures must be generic consistently, which was nearly never the case in our experiments.

5 Active Attacks

5.1 ANQP

While Vanhoef et al. [29] investigated the use of randomized and hardware MAC addresses in ANQP requests sent by Apple devices and Linux and Windows computers, we extend this analysis to include a variety of Android devices. Like [29], we find that all Apple devices we tested use a randomized MAC address when sending ANQP requests to query Hotspot 2.0-capable APs. However, we find a wide variety of behaviors among Android devices, as displayed in Figure 5. Some manufacturers, like Google, OnePlus, and Samsung exhibit a distinct transition from using the device's hardware MAC address in earlier-released models, to using a randomized MAC address in later models. This suggests a conscious effort by the manufacturer to implement best practices suggested in [29]. The other manufacturers we assessed have not adopted the practice of using a randomized MAC address in ANQP frames; this leaves the device open to being stimulated by an adversary seeking to elicit nearby hardware MAC addresses. Finally, some devices we tested did not send ANQP requests at all; these behaviors are summarized in Table 3.

5.2 RTS/CTS

This work also updates [21] by examining whether devices have implemented privacy protections against the RTS/CTS attack the authors developed, which they found affected 100% of devices they studied. In other words, do modern devices still respond to RTS frames sent to their hardware MAC address even while probing for networks with a randomized source MAC address?

Figure 5 indicates that again, several manufacturers have implemented the recommended best practice from [21] and do not respond to RTS frames addressed to the device's hardware address with a CTS frame, or they do so with such infrequency that we failed to detect it during our two-minute observation period described in §3.4. In either case, an attacker is unable to determine whether their target is nearby. Apple and Samsung devices evince a clear split, after which their products are no longer vulnerable to the RTS device detection attack. All OnePlus phones we tested were unresponsive to RTS frames, though we have no models produced prior to 2016. Most manufacturers' devices are still vulnerable to this attack, including several models of LG and Motorola devices that were released in 2019. This indicates that while this privacy vulnerability has been patched by some manufacturers, many are either unaware of the threat or have not prioritized addressing it. The RTS response behavior of selected devices is listed in Table 3.

5.3 Forcing the Action State

As described in §4.2.3 and §4.5, many Android devices deploy privacy technology only when the device is in an idle state: *e.g.* randomized MAC address and generic signatures. These behaviors persist even in devices on Android 9 and 10, and in particular nearly all Motorola devices use their hardware address (whether by cycling through it, or simply using it exclusively) in the action state. In addition, we observed informally that devices which use

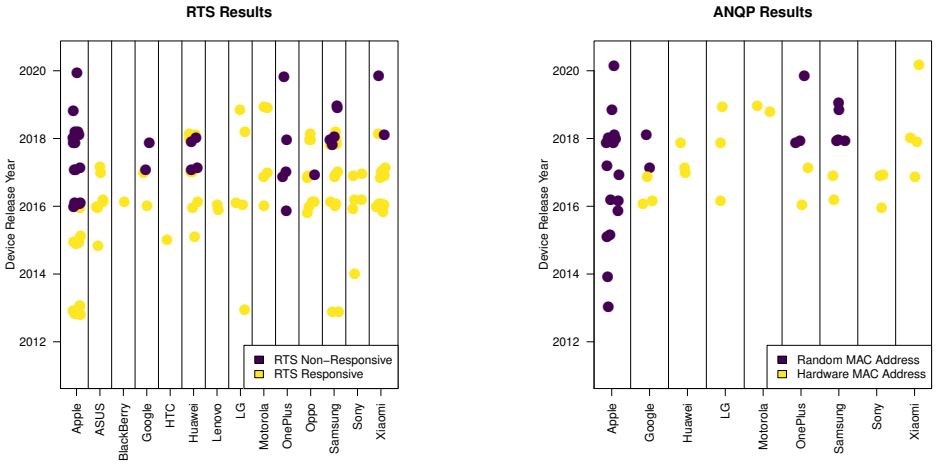

Fig. 5. Active Attacks

their hardware address in between random addresses do so more frequently in the action state. Since many devices lose privacy protections while in active use, we investigated further whether the device's action state could be elicited by active attacks or benign non-user behavior. For all of these devices, the action state behaviors began immediately when the screen was turned on by a user or when the device received an incoming phone call, meaning that an active attacker with the user's phone number could defeat these privacy technologies without user interaction. Text messages and push notifications did not induce a state change, though these signals could prompt the user to activate her own device to respond.

The aforementioned action "triggers" require a user-based activity to occur that is reasonably unlikely for an adversary to elicit in a non-obtrusive manner. As such, we sought to identify a non-interactive, inconspicuous triggering mechanism. We repeated the ANQP experiments performed in §5.1 to observe whether the mobile device switches from the idle to the action state when it transmits an ANQP request frame. Our first test, a Google Pixel 3a running Android 10 began sending action state device signatures immediately after transmitting an ANQP request.

We repeated the experiment on a Motorola Moto G (7th gen.) on Android 10 and a LG V20 on Android 7. Both of these devices properly employ randomization while in the idle state, but utilize the hardware MAC address in the action state. As with the Google Pixel 3a, each device transitioned to the action state upon executing ANQP queries, in this instance causing the devices to transmit probe requests with the hardware MAC address. It is important to note that devices do not immediately send ANQP queries in the idle state version of our ANQP experiments. In order to optimize power usage while the screen is off Android implements interval scan rates when searching for available Wi-Fi networks [1]. Our work does not explore power saving implementations in detail; rather, we note that while this attack extracts identifying information from these devices, it relies on the device initiating a scan for nearby APs, which can occur infrequently.

6 Post-association Randomization

Post-association randomization, wherein a random MAC address is used as a client's source MAC address *after* it connects to an AP, was introduced in 2015 in Windows 10 [16]. Four years elapsed before Android 10, introduced in fall 2019, became the first mobile operating system to introduce this feature for devices with supported hardware. Apple's iOS 14, released in 2020, also supports post-association randomization. Thus, an investigation into the privacy-preserving properties and discussion of privacy shortfalls of this type of MAC address randomization is a timely problem.

In contrast to the randomized MAC addresses considered heretofore, post-association randomization is intended to protect user privacy when sending 802.11 frames during and following association to an AP. Rather than use the device's hardware MAC address when transmitting, in all post-association randomization schemes, the device uses a random MAC address it has chosen for this connection for some period of time, although implementation differences exist. Regardless of implementation, this type of randomization is designed to protect user tracking by an adversary observing the same hardware MAC address on multiple networks, *e.g.*, at home and at the office. This type of randomization, while new, is a crucial development for location privacy. Devices are often associated to APs when users are stationary, and in many instances mobile devices can be trivially induced to associate to a previously-used Service Set Identifier (SSID) by an active attacker.

6.1 Windows 10

While we focus primarily on mobile devices, we mention Windows 10 briefly because it was the first widely available implementation of post-association randomization, and as a point of comparison for other implementations. Post-association randomization is not enabled by default in Windows 10. A user may either enable this setting for specific networks individually or as a global setting applied to all 802.11 networks; in either case, the device chooses a random MAC address to use when connected to each SSID to which it connects. Windows 10 also affords the user with the option to change the random MAC address for use with a specific SSID daily, rather than the random MAC address generated when the device first connected.

6.2 Android 10/11

Unlike Windows 10, in Android 10 and 11, post-association MAC address randomization is enabled by default for new connections if the device hardware supports it – though it may be disabled by the user. Random MAC addresses are generated per-SSID; this enables users to move from one AP to another in an Extended Service Set (ESS) while using the same MAC address. MAC address reuse for the same SSID prevents needing to reauthenticate to each new AP within an ESS and the potential to be denied because of MAC address filtering if a new MAC address were to be generated for each associated Basic Service Set Identifier (BSSID). Android stores SSID-random MAC address mappings in a root-accessible file called `WifiConfigStore.xml`.

Android's post-association randomization landscape is fragmented, owing to the variety of device manufacturers developing unique Android flavors for their products and disparate underlying hardware capabilities. For example, some devices were incapable of using a random MAC address after association despite running Android 10. This included the original Google Pixel, released in 2016, which we speculate is limited by its relatively dated hardware. Of note, the Xiaomi Mi 9 Lite, released in September of 2019 does not implement post-association MAC address randomization, indicating that widespread adoption may still require manufacturer support.

Some Android devices leak traceable identifiers after establishing a wireless connection with post-association randomization. These identifiers can potentially undermine the privacy benefits afforded by post-association randomization by allowing an adversary to correlate two random MAC addresses on different networks. Of the devices we tested running Android 10, the OnePlus6, Samsung Galaxy Note 9 and Samsung Galaxy S10 fall into this category. Following association to our network, we observe hostname and operating system information passed in DHCP Discover and Request messages. In the case of all three devices, the phone model is the default hostname (*e.g.*, Galaxy Note 9), while `android-dhcp-10` was additionally passed as the Vendor Class Identifier DHCP option. While the combination of these identifiers distinguishes the phone's device model and operating system version, users that modify their hostname potentially increase their ability to be tracked, depending upon its uniqueness. Other devices, primarily newer-model Google Pixels, leaked only their OS version in DHCP messages; similar to the Samsung and OnePlus models we tested, Pixels identify themselves as `android-dhcp-10` devices in DHCP messages, which are broadcast on the local network.

We also note that trackable identifiers exchanged in HTTP sessions (*e.g.* cookies) are available to adversaries capable of sniffing traffic at the AP in the case of an encrypted network, and to any observer within transmission range for open networks.

6.3 iOS 14 Beta and Production Release

Although iOS 14 has been officially released as of the time of writing, Apple had previously announced that devices running it, iPadOS 14, or watchOS 7, would implement post-association randomization [10]. We installed the iOS 14 beta release in order to evaluate the features present prior to the official release of iOS 14 and compare the findings to the production release.

Unlike Android's implementation of post-association MAC address randomization, the iOS 14 beta release ensures that a new random address is selected for each distinct SSID when the user "forgets" the network, or after 48 hours have elapsed. This address is used for all subsequent connections to networks advertising this SSID until the user "forgets" the network. At this point, the random MAC address-SSID mapping ceases to exist, and a new random MAC address will be generated should the user decide to connect again in the future. Like Android, Apple's post-association randomization implementation is enabled by default for all new networks (per-SSID) a user joins, but can be manually disabled by the user. This behavior diverges from Android, which permanently links a MAC address to each SSID.

Recent press releases from Apple [10] and other sources [26] suggested that the 48 hour rotation may not be enforced in the production release. In our analysis of the official iOS 14 release, we confirm that Apple demurred from the aggressive 48 hour rotation schedule and instead maintain a connection-persistent address unless a user forgets the network, prioritizing usability over privacy in this case.

Despite enabling post-association randomization in iOS 14, we observe traceable identifiers appearing in DHCP

traffic that could be used by an on-network (or nearby, if the network is unencrypted) adversary to correlate users with the random MAC addresses their device is using.

6.4 Active Attacks

Vanhoef *et al.* present attacks on pre-association randomization that induce a client randomizing its MAC address to reveal its hardware MAC address [29]. We extend those attacks to post-association MAC address randomization.

6.4.1 Leveraging Connection-Persistence

Nearly all devices implementing post-association MAC address randomization can be tracked by taking advantage of the scope of the per-network MAC address to network mapping, because all but Windows 10's "change daily" setting use a persistent MAC address for an indefinite amount of time. This enables an adversary to either follow devices as they reconnect to these networks over time and space, or perform an Evil-Twin attack using well-known SSIDs. As [29] notes, the increasing prevalence and usage of Hotspot 2.0 networks provide a wealth of SSIDs suitable for use in Evil-Twin attacks, as cellular and cable providers often offer Hotspot access (*e.g.* "xfinitywifi", "attwifi"). Additionally, free Wi-Fi services offered at popular regional locations like cafes or airports, municipal Wi-Fi deployments, and industry-specific networks like "eduroam" [13] present additional Evil-Twin SSID targets, allowing an attacker to detect the presence of a device in an arbitrary physical location with knowledge of one of that device's previously-used connections. While an adversary does not obtain the victim's true MAC address, the persistence of the random MAC address makes it just as valuable for targeted user tracking.

Nearly all implementations of post-association MAC address randomization are vulnerable to tracking persistent random MAC addresses. Android 10/11 permanently joins a device-SSID pair to a randomized MAC address until a factory reset.

Windows 10 devices implementing post-association MAC address randomization keep a persistent address for one day or indefinitely, but these addresses can be refreshed by forgetting and later rejoining, a network.

6.4.2 Leveraging EAP Identities

Wi-Fi networks utilizing Extensible Authentication Protocol (EAP) [7] authentication methods provide a final tracking vector. In addition to trackable identifiers in network protocols (§6), clients configured to authenticate using an EAP method to certain networks provide another trackable identifier in the EAP identity. EAP-Evil-Twin attacks using EAP authentication are trivial to conduct; `EAPhammer` [14], for example, is a penetration-testing tool designed to steal EAP identities and credentials for networks employing Protected Extensible Authentication Protocol (PEAP) and EAP-Tunneled Transport Layer Security (TTLS). Generally, EAP identities are the username of the device owner and are unique within the enterprise to which the customer belongs and remain static over time. Other EAP methods, like EAP-Subscriber Identity Module (SIM) and -Authentication and Key Agreement (AKA), provide a trackable identifier in the form of mobile subscriber's International Mobile Subscriber Identity (IMSI) [11, 21, 27], although some mechanisms to protect the IMSI exist [19, 27].

Regardless of the EAP method employed, most EAP identities provide a static identifier that can be linked to a connection-specific MAC address as in §6.4.1, providing a second traceable identifier despite post-association MAC address randomization.

7 Summary of Modern Devices

We summarize the current state of the modern devices from our corpus: devices whose hardware release date and OS release date were in 2018 or later. For devices we tested on multiple OS versions, we give the results for our test after the device has been fully updated. Our results are given in Table 3. In summary, for passive attacks all recommendations have been consistently applied in nearly all modern devices, with the notable exception of the signatures. The Google Pixel 3 uses intentionally low-information signatures, but we consider these signatures non-generic as discussed in §4.5. We highlight the 7th Generation Motorola Moto G, the only modern device we tested that did not consistently use random addresses even after all available updates were applied, though the overwhelming general trend we observe is towards consistent, well-deployed 46-bit randomization, with random sequence numbers.

We review the work of Vanhoef et al. [29] and observe that some Android devices (LG and Motorola) remain susceptible to ANQP attacks, while the attack is ineffective against other modern Android devices. Many devices still respond to RTS messages, but we note this attack does not extract private information from a device, simply allows an adversary to confirm its presence once the adversary has already identified the hardware MAC by other means.

Post-association randomization was deployed in Android 10 and iOS 14. While most devices we tested employed post-association MAC address randomization we observe that the Xiaomi Mi 9 Lite did not, indicating that it is not seamlessly integrated across manufacturers.

Table 3. Modern Devices

Model	OS Version	Random Address	Generic Signature	Random Sequence Number	Random ANQP Address	RTS Unresponsive	Post-Assoc
Apple iPhone X	13.4.1	✓	X	✓	✓	✓	X
Apple iPhone XR	13.4	✓	X	✓	✓	✓	X
Apple iPhone XS	13.4	✓	X	✓	✓	✓	X
Apple iPhone XS Max	13.4	✓	X	✓	✓	✓	X
Apple iPhone 11	13.4.1	✓	X	✓	✓	✓	X
Apple iPhone 12	14.2	✓	X	✓	✓	✓	✓
LG V35 ThinQ	9	✓	X	✓	X	X	X
LG G8 ThinQ	10	✓	X	✓	X	X	✓
Google Pixel 3	10	✓	Attempted	✓	✓	✓	✓
Motorola Moto Z (4th gen.)	10	✓	X	✓	X	X	✓
Motorola Moto G (7th gen.)	10	Idle-Only	Idle-Only	X	X	X	✓
OnePlus 6	10	✓	X	✓	✓	✓	✓
OnePlus 8	11	✓	X	✓	✓	✓	✓
Oppo F7	9	✓	X	✓	?	X	X
Oppo F9 Pro	9	✓	X	✓	?	X	X
Samsung Galaxy A8 (2018)	9	✓	X	✓	?	✓	X
Samsung Galaxy Note 9	10	✓	X	✓	✓	✓	✓
Samsung Galaxy S9	10	✓	X	✓	✓	X	✓
Samsung Galaxy S9+	10	✓	X	✓	✓	X	✓
Samsung Galaxy S10	10	✓	X	✓	✓	✓	✓
Xiaomi Mi 9 Lite	10	✓	X	✓	X	✓	X

✓ : Privacy Protection Employed X : Privacy Protection Non-Existent ? : Test Inconclusive

8 Conclusion and Recommendations

We consider a primary takeaway of our work to be that randomization technology has not been deployed cleanly or consistently across the range of modern mobile devices. Not only do different OSes introduce these technologies differently, but different manufacturers implementing the same operating system have their own distinctions and idiosyncrasies. It appears as though manufacturers and OS developers saw the need to harden probe requests and implemented their own solutions independently, leading to fragmentation and significant differences in effective privacy from one device to another.

In time, these technologies appear to be converging towards consensus, but the path is slow and staggered, moving in fits and starts. In particular, we found the behavior of individual Android 7, 8, and 9 devices to be difficult to predict, with many pairs of similar devices with radically dissimilar privacy properties, devices whose privacy protections are eliminated with OS updates, and devices that provide effective protections only while the device is not actively in use. However, we observe a clear, strong trend toward effective deployment of privacy technologies with some protections in particular lagging behind: device signatures and mitigation against ANQP and RTS attacks. The most egregious passive and active attacks, which allow adversaries to recover a device's hardware address, are now mostly ineffective. Post-association randomization presents important new challenges, and again we see distinct implementations emerging from different device ecosystems, some of which still provide trackable identifiers to passive and active attackers.

While pre-association MAC address randomization privacy improvements have not majorly effected usability in practice, some networks are designed assuming a consistent and stable MAC address per device, which sits in direct conflict with the fundamental privacy goals of post-association MAC address randomization. This makes finding a compromise between privacy and usability particularly difficult in this context, which we have seen with shifting randomization policies in the case of Apple, or allowing network operators to disable privacy technology and deploying it off-by-default from Microsoft. While we advocate for on-by-default effective post-association MAC address randomization, we recognize this requires a significant change in the design of some enterprise networks, and may take time before it can be seamlessly deployed on all devices.

8.1 Recommendations

- **State**. Devices should deploy privacy protections consistently no matter the device state, noting that these states are fragile and may be easily manipulated in unexpected ways by an adversary to compromise user privacy, as we have shown above.
- **Pre-Association Randomization**. Manufacturers should agree on a fixed, small, low-information device signature for probe requests. We also recommend manufacturers fix the remaining vulnerabilities (RTS, ANQP, etc.) outlined in Table 3.
- **Post-Association Randomization**. We recommend the industry move towards the approach taken in the initial iOS 14 beta randomization model, where post-association MAC address randomization is on by default and random MAC addresses used for a given SSID rotate regularly without user interaction. Additionally, devices should ensure the messages they send in discovery protocols like DHCP do not contain persistent or model-specific identifiers.

Acknowledgment

Views and conclusions are those of the authors and should not be interpreted as representing the official policies or position of the U.S. Government. The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by the author. This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. The authors additionally thank Joey Han, Paul Slife, and Leo Bulgrin for technical assistance and feedback. Finally, for their hard work conducting our many individual experiments, we would like to thank John Baldwin, Will Cheshire, Byron Gallagher, Adie Geoghagan, Eduardo Gomez, Hugh Hajdik, Dylan Larkin, Trystin Martin, Eugene Om, Alex Psichas, Connor Schellenbach, Caroline Sears, Geno Shamugia, and Eric Towe.

References

[1] Wi-fi preferred network offload scanning, . https://source.android.com/devices/tech/connect/wifi-scan.

[2] Android wi-fi network selection, . https://source.android.com/devices/tech/connect/wifi-network-selection.

[3] 802.11aq-2018 - ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control and physical layer specifications amendment 5: Preassociation discovery. https://standards.ieee.org/standard/802_11aq-2018.html.

[4] Wifi certified passpoint® continues worldwide momentum. https://www.wi-fi.org/beacon/the-beacon/wi-fi-certified-passpoint-continues-worldwide-momentum.

[5] Changes to device identifiers in android o, Apr 2017. https://android-developers.googleblog.com/2017/04/changes-to-device-identifiers-in.html.

[6] Fingerbank, 2020. https://fingerbank.org/.

[7] B. Aboba, L. Blunk, J. Vollbrecht, J. Carlson, and H. Levkowetz. Extensible Authentication Protocol (EAP). RFC 3748 (Standards Track), 2004. http://www.ietf.org/rfc/rfc3748.txt.

[8] Wi-Fi Alliance. Wi-Fi Simple Configuration Protocol and Usability Best Practices for the Wi-Fi Protected Setup™ Program, 2020. https://www.wi-fi.org/download.php?file=/sites/default/files/private/wsc_best_practices_v2_0_1.pdf.

[9] Amelia Andersdotter. Ongoing developments in ieee 802.11 wlan standardization. *12th Workshop on Hot Topics in Privacy Enhancing Technologies (HotPETs 2019)*, 2019.

[10] Apple. Use private Wi-Fi addresses in iOS 14, iPadOS 14, and watchOS 7, 2020. https://support.apple.com/en-us/HT211227.

[11] Jaejong Baek, Sukwha Kyung, Haehyun Cho, Ziming Zhao, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. Wi not calling: Practical privacy and availability attacks in wi-fi calling. In *Proceedings of the 34th Annual Computer Security Applications Conference*, pages 278–288, 2018.

[12] Guillaume Celosia and Mathieu Cunche. Discontinued privacy: Personal data leaks in apple bluetooth-low-energy continuity protocols. *Proceedings on Privacy Enhancing Technologies*, 2020 (1):26–46, 2020.

[13] eduroam. eduroam, 2020. https://eduroam.org.

[14] Gabriel Ryan (s0lst1c3). EAPhammer, 2020. https://github.com/s0lst1c3/eaphammer.

[15] Denton Gentry and Avery Pennarun. Passive taxonomy of wifi clients using mlme frame contents. *arXiv preprint arXiv:1608.01725*, 2016.

[16] Christian Huitema. Experience with mac address randomization in windows 10. In *93th Internet Engineering Task Force Meeting (IETF)*, 2015.

[17] IEEE. Ieee standards for local and metropolitan area networks: overview and architecture. *IEEE Std 802–2001*, pages 802–1990, 2001.

[18] Oisín Kyne. Mac address de-anonymisation. *arXiv*, pages arXiv–1805, 2018.

[19] Malthankar, Rohan C., Sawant, Paresh B., Fernandes, Sitnikov, Sergey, Mathias, Arun G., Novak, and et al. Protection of the ue identity during 802.1x carrier hotspot and wi-fi calling authentication - apple inc., May 2018. http://www.freepatentsonline.com/y2018/0124597.html.

[20] Jeremy Martin, Erik Rye, and Robert Beverly. Decomposition of mac address structure for granular device inference. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 78–88. ACM, 2016.

[21] 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.

[22] Célestin Matte and Mathieu Cunche. Panoptiphone: How unique is your wi-fi device? In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 209–211, 2016.

[23] Célestin Matte and Mathieu Cunche. Spread of mac address randomization studied using locally administered mac addresses use historic. 2018.

[24] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. Defeating mac address randomization through timing attacks. In *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, pages 15–20, 2016.

[25] Scientia Mobile. Mobile overview report, 2020. https://www.scientiamobile.com/movr-mobile-overview-report/.

[26] Wi-Fi Now. Is Apple backpedaling on their new 'Private Wi-Fi' feature?, 2020. https://wifinowglobal.com/news-and-blog/is-apple-backpedaling-on-their-new-private-wi-fi-feature/.

[27] Piers O'hanlon, Ravishankar Borgaonkar, and Lucca Hirschi. Mobile subscriber wifi privacy. In *2017 IEEE Security and Privacy Workshops (SPW)*, 2017.

[28] Jiaxing Shen, Jiannong Cao, and Xuefeng Liu. Bag: Behavior-aware group detection in crowded urban spaces using wifi probes. *IEEE Transactions on Mobile Computing*, 2020.

[29] Mathy Vanhoef, Célestin 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*, pages 413–424. ACM, 2016.

[30] Wi-Fi Alliance. Hotspot 2.0 Specification Version 3.1, 2019.

[31] Fang-Jing Wu, Yunfeng Huang, Lucas Doring, Stephanie Althoff, Kai Bitterschulte, Keng Yip Chai, Lidong Mao, Damian Grabarczyk, and Ernoe Kovacs. Passengerflows: A correlation-based passenger estimator in automated public transport. *IEEE Transactions on Network Science and Engineering*, 2020.