Device Identification in the Presence of MAC Randomization

Ihab Zhaika
The Hebrew University
Jerusalem, Israel
ihab.zhaika@mail.huji.ac.il

David Hay
The Hebrew University
Jerusalem, Israel
dhay@cs.huji.ac.il

***Abstract*—Wi-Fi (IEEE 802.11) is the most-used protocol for wireless internet access on customer premises. The MAC address of each connected device, which used to be static, is being recently randomized (by the device's operating system) as frequently as daily to prevent tracking and fingerprinting of devices and users.**

While this feature might be useful in public areas, it disturbs some day-to-day functionalities, such as firewalls, parental control, and similar applications that require a static identifier per device. In this work, we present methods to ensure the functionalities of these applications, even when the MAC address is changed every time the device connects to the network. Our methods work even if the latest MAC randomization techniques are applied and provide these device identifications only to the gateway router. (Potentially malicious) devices that are connected to the same LAN, still see the randomized MAC address, and therefore, cannot identify or track the device.

I. Introduction

Nowadays, nearly all gateway routers at homes provide Wi-Fi connectivity, enabling their owners to easily connect mobile and IoT devices at high speed, in a relatively wide coverage area.

The ubiquity of Wi-Fi and its emergence as a universal wireless interface makes it a great candidate for *device tracking*: the ability to trace and identify a specific device among many others, within one network, across several networks, and in public spaces. Devices may be tracked without permission or the knowledge of the device owner/user, thus exposing them to various attack vectors and infringing their privacy. The most common unique identifier that is being used to track devices is their *Medium Access Control (MAC)* addresses. Traditionally, MAC addresses are unique and are assigned to the device by the Wi-Fi chip manufacturer; these addresses do not change over time and are used by the device to connect to any network (unlike IP addresses, for example, which are logical addresses that depend on the network the device connect to). When MAC addresses are constant and known to the tracker, tracking and identification are relatively easy as demonstrated in [1], [2], where one of the first proof-of-concept experiments of such technology has used garbage bins in the city of London to track users in public spaces for targeted advertisement purposes [2].

To tackle this issue and provide privacy to its users, major operating systems have included some mechanism of *MAC randomization*: generating MAC address randomly to hide the unique MAC address that was assigned by the manufacturer. This is done in two levels, each addressing different attack vectors.

First, MAC addresses are being randomized on the Wi-Fi management frames, used to bootstrap the connection of the device to the Wi-Fi network (through a Wi-Fi access point). These management frames, or more specifically, Wi-Fi probes by the devices, can be intercepted by an eavesdropper in close physical proximity. Thus, to eliminate tracking by these eavesdroppers, each burst of probes is sent with a newly-generated random MAC address. We note that the probe requests contain, in plain text, meta-data in form of Information Elements (IEs), which are key-value pairs that describe the capabilities of the device's NIC.

MAC addresses are only being randomized *after* the device has established a connection with the access point and has joined the Wi-Fi network. Such MAC address randomization is meant to eliminate the possibility for another device in the local network or the network owner (which is, sometimes, the communication service provider) to maliciously track the device. In such a scenario, MAC addresses can be randomized on every frame (or burst of frames) as the device may maintain long-lived connections over time. Thus, randomized MAC addresses can be changed either after a few hours or remain constant (but different, for each Wi-Fi network the device may join).

In recent years, MAC randomization become ubiquitous. iOS provides MAC address randomization since iOS 14 [3], implying that, as of January 2022, more than 93% of Apple devices worldwide, and more than 98% introduced in the last four years, use MAC randomization [4]; Android provides MAC address randomization since Android 8 [5], consisting, by 2021, of more than 50% of the Android OS market [6], [7].

The initial versions of MAC randomization in operating systems were turned off by default and were applied per Service Set Identifier (SSID), implying the generated MAC address is persistent as long as the SSID has not changed. However, more recently, several operating systems, such as Android 11, set MAC randomization on by default and give the option to enable a *daily* MAC address randomization that keeps changing the MAC address even in the same network with the same SSID.

978-1-6654-3540-6/22

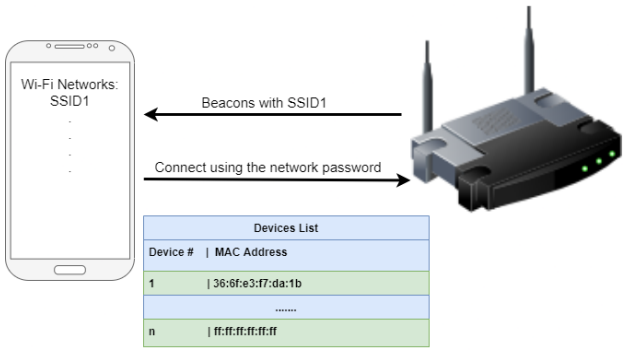

Fig. 1: The common setting, where *persistent* MAC address is used to identify each device. Such device identification does not work when MAC address randomization is in use.

However, the recent shift to MAC address randomization has a significant adverse effect on *legitimate network appliances* that are using (persistent) MAC addresses to *identify* devices. More specifically, persistent MAC addresses are extensively used by *security appliances* for setting permissions of devices for activities within the network or creating network management use cases, such as port forwarding. Persistent MAC addresses are useful to troubleshoot network performance, especially if you need to inspect a certain device's network behavior over several days. Additionally, persistent MAC addresses are essential in pay-per-use day passes (e.g., public Wi-Fi), guest network login (e.g., in hotels) and parental control services (e.g., in-home Wi-Fi) [8]. Fig. 1 illustrates the common settings in which security (and other) appliances are mapping the (persistent) MAC address with the device itself.

One approach to identifying devices is to try to infer what is the device based on its traffic characteristics (on all layers). These passive approaches (in which the device is identified by observing the traffic and not by taking any action) are probabilistic and tend to perform poorly when several devices of the same manufacturer/model exist in the network. The performance of these methods, in general, does not suffice for security appliances as a substitute for the MAC address. Section II surveys these methods in details.

In this work, we are taking an *active approach*, in which the behavior of the *Wi-Fi access point* is *slightly* modified to allow device identification by the Wi-Fi access point itself. It is important to notice that our approach has the following three advantages: *(i)* No modification is needed to the device itself or its operating system; *(ii)* only the Wi-Fi access point can identify the device, while malicious eavesdroppers or other entities within the Wi-Fi network still observe randomized MAC addresses that can change frequently; and *(iii)* there is no tracking capabilities across networks, thus a network owner cannot correlate devices' traffic as they move between networks (even if it owns all of them, as in the case of a communication service provider). It is important to note that our approach assumes that *the device trusts the access point*, and therefore, does not try to hide its identity from the access point. Attack vectors in which the access point may be malicious are not covered in this paper. Moreover, we do not deal directly with cases in which a malicious device tries to impersonate other devices that have previously joined the network (which can happen also when using MAC randomization).

Specifically, we are proposing two active methods and discussing the advantages and disadvantages of each of the methods.

In the first method, we require access points to generate multiple SSIDs, so that each device connects to the network through a unique SSID. This is done by a small modification to the access point code, as most access points supports multiple SSIDs anyway. Interestingly, the generated SSIDs contain non-printable ASCII characters, therefore they appear to the user as if they are the same. As devices try to reconnect to previously-visited SSIDs, we can identify the device by the SSID it has used. See illustration in Fig. 2(a). Section III discusses this approach in details.

Alternatively, our second method is to use a single SSID per network but allocate a unique password for each connected device. When a device tries to re-join the network, it uses its allocated password, and therefore, can be identified. See illustration in Fig. 2(b). Section IV discusses this method in details. Generally speaking, we show that, when available, our password-based method is better and is also resilient to impersonation attacks.

Both of our proposed methods yield high accuracy in comparison to the passive methods, and even more, can be orthogonal to the passive methods. Our methods work on any type of device's operating system and do not need any training phase. Importantly, the proposed methods can be applied easily by the customer premises equipment (CPE) manufacturers without the need for expensive or special equipment. In our experiments, we implemented the solutions in OpenWRT [9], an open-source operating system based on Linux for network traffic routers, that requires only slight modification to existing code. Finally, with our methods, operating systems may opt to use the more-secure non-persistent MAC randomization, as our methods can identify devices no matter how frequently randomized MAC addresses are being generated.

II. Related Work

In this section, we survey *passive approaches* for device identification that does not rely on MAC addresses, and therefore, can be used in the presence of MAC randomization.

One group of these passive approaches rely on *physical layer characteristics*, such has carrier frequency offset [10], waveform and wave modulation [10], and clock Skew variation [11], [12].

Another group has used additional MAC header information (other than MAC address) [13], such as the fragment number and the sequence number (if not reset between probes). Layer 2 parameters are also used for reverse hash-mapping on *Wi-Fi Protected Setup (WPS)* information [13] and by analyzing IEs in the probe requests, probe responses, or beacons [13]–[15].

((a)) Unique SSID per device.

((b)) Unique password per device.

Fig. 2: Illustration of our two active approaches for device identification: (a) The access point uses many SSIDs so that each device is connected through a unique SSID and can be identified by it; (b) The access point uses a single SSID but with many passwords so that each device can be identified by the password it uses.

Devices have also been identified based on time-based characteristics [16], such as probe requests timing in which its inter-burst intervals, burst length and size, and inter-frame arrival time. Finally, geospatial characteristics (e.g., the distance of the device from the access point) were used to identify devices that are either static or have limited mobility [14], [17].

Most importantly, all of these methods are probabilistic and perform poorly when the set of the devices are from the same model and running the same software versions (as these devices have very similar meta-data). Our active approaches, on the other hand, are deterministic and can be used to distinguish between two identical devices.

III. Identification Using Multiple SSIDs

Our first active approach involves using multiple SSIDs per Wi-Fi network. These SSIDs are broadcasted in beacons by the Wi-Fi access point (AP), and each device picks its SSID (only a single device can use a specific SSID) and tries to join the network with that SSID. Wi-Fi devices today try to rejoin previously visited SSIDs, implying future connections of the device to the network are done with the same SSID. Thus, upon a successful connection, we can map the device with the SSID (See Fig. 2(a)). We note that this approach do not work if the user chooses not to connect to Wi-Fi networks automatically. In such a case, one can use our second approach, which is described in Section IV.

SSIDs can be broadcasted as *visible* or *hidden*. Next, we discuss how these two methods affect the identification process and what is required from the device (or the device's user) for the approach to work.

A. Visible (Public) SSIDs

The AP is broadcasting many beacons, each with a unique SSID. All SSIDs differ only in some *non-printable Unicode characters* (everywhere in the string), and therefore, they look the same to the end-user. As mentioned before, once an SSID is picked, the same one is used in the next attempts to rejoin the network, and thus, the SSID can identify the device.

Notice that when a device joins the network for the first time, it has a user interface that lists all the available SSIDs in the area (e.g., the WiFiManager in Android). In our case, the list consists of many SSIDs that look identical. The user will click on one of the SSID and will join the network. We note that typically upon re-joining the network the user should not click any SSID.

Special care needs to be taken to ensure that two devices do not connect on the same SSID (thus identifying both devices as one). This is done by ordering the (identical) SSID, such that SSIDs, which have been already used, appear down on that list (so the user should scroll to connect to them). The Wi-Fi manager in most operating systems, among them iOS and Android, are sorting the SSIDs by the signal strength and by the alphabetic order, in our case the signal strength is the same since the beacons are transmitted from the same source which means that the order of the list is determined by the alphabetic order, implying the order can be easily manipulated by the AP.

For this approach to work, the AP should be continuously transmitting multiple beacons with multiple SSIDs, which include both unused SSIDs (that are used by new devices) and the SSIDs that were used by some devices to connect (so that they can re-join the network). To improve the re-joining time, used SSIDs can have more frequent beacons than unused ones.

B. Hidden SSIDs

In this setting, SSIDs are not broadcasted by AP, but the user should insert the SSID manually. These hidden SSIDs are supported by most of the commercial AP, where the SSID can be obtained in various methods, such as scanning a QR code using Zebra Crossing library [18] or through a captive portal that has an SSID with non-printable Unicode characters and instructions on how to join the network with this SSID.

When the user tries to connect to a hidden SSID, the device broadcasts a probe request with the explicit SSID. To

connect automatically, user devices keep transmitting these probe requests periodically, where the frequency varies between devices, operating systems, and the device state; for example, the frequency may be tens of minutes when the device screen is off, and a few seconds when a user launches Android's Wi-Fi manager. To tackle this issue we suggest that the AP should transmit beacons with SSIDs that were used in the past, along with the guest SSID used to connect a new device. Hence, previously-connected devices will seamlessly rejoin the network with their correct SSID (as in the case of visible SSID); if the network is password-protected, new devices cannot use these SSID to connect to the network (and should use the QR or captive portal).

IV. Identification Using Multiple Passwords

This approach uses unique passwords for connecting to the Wi-Fi network and, unlike previous approaches, yields a *deterministic identification* if applied correctly. In a nutshell, in this approach, the AP uses a single SSID and assigns a unique password for each new device. As, upon rejoining the network, the device uses its pre-assigned password, the AP is able to identify it by keeping the mapping between the devices and the assigned password (see Fig. 2(b)).

More specifically, the usage of passwords depends on the authentication protocols employed by the AP (none of these were designed to have multiple passwords for the same AP/SSID).

A. WPA/WPA2

We are utilizing the 4-way handshake authentication flow of the 802.11 standard [19] but with a unique password for each user. To do so, we are required to modify the codebase in the AP (but not in the device). First, by the standard, both sides need to generate a *Pre Shared Key (PSK)* with a *Password-Based Key Derivation Function (PBKDF2)*. The PSK generated by PBKDF2 depends on the passphrase and the SSID, implying each device will have a different PSK. Thus, at the beginning of the authentication flow, the AP should iterate through the list of previously-generated PSKs, and decrypt the message from the device with each one of them until the right PSK is found. Then, the authentication flow continues as specified in the 802.11 with the right PSK for the device.

Similar to the case of hidden SSID, we have a few options to generate a passphrase for each device, including manually giving a different plain-text passphrase for each device, using a QR code, or through a captive portal.

B. WPA3

The WPA3 [20] standard, which is starting to emerge in recent operating systems, supports multiple passwords per SSID, or *Password Identifiers (PIs)*, implying one can assign a PI for each device and get the same functionality out of the box. Notice that, as WPA3 uses Simultaneous Authentication of Equals (SAE) [21], our previous approach with WPA/WPA2 and PBKDF2 does not apply under WPA3.

No.	Name	Passphrase	MAC List	QR 2.4 GHZ	QR 5.x GHZ
1	David-phone	zFtgeuZ4PB zXt4vwQatr	ac:c1:ee:33:50:8b		
2	Win-PC	znmigEa79fL Gw	ec:08:6b:01:18:03		
3	iPhone	JApj7hHgCE TjMX5T4Nri 3V	a4:4b:d5:bd:6a:8b		
4	Ihab-Android	eX5oI0dsJbR 1OCQcg57Ir aBT	7e:60:45:6b:c4:e3 4a:2e:bc:1c:e6:a4		

Fig. 3: Our dashboard is used to generate unique passwords (and their corresponding QR codes), as well as to map MAC addresses to devices. Notice that the device named "`Ihab-Android`" is using MAC randomization, and therefore, it is associated with many MAC addresses.

V. Implementation Details

The Access Point (AP) is implemented by modifying the SSID and password handling (specifically, PSK generation and searching) mechanisms in HostAPD [22], which is a user-space daemon for access points and authentication servers. When configuring HostAPD, the AP owner selects some SSID for the network, so we are utilizing this SSID to craft a set of similar SSIDs that contain non-printable Unicode characters. This has resulted in modifications to several routines within hostAPD such as probe request/response handling, as well as association and authentication. We note that on the authentication side we have developed methods to find the PSK corresponding to the selected SSID, as described in Section IV-A.

To generate beacons, each one with a unique SSID, we were not able to use HostAPD as the minimum interval between two beacons under HostAPD is 100 milliseconds (which is prohibitive when one has 100s of SSIDs). Instead, we have used an external Arduino device (with ESP8266 Wi-Fi module) that requires only 2 milliseconds between two beacons. We note that our Arduino device was used only for sending beacons, while all the standard 802.11 protocols were run on the hostAPD module. More details on the devices we have used for our experimental results are in Section VI.

To generate the passwords we have used a dashboard as shown in Figure 3 that runs on the AP. It allows the AP administrator to generate passwords dynamically, to track the devices that are connected using a given password, and to generate a standard QR code that can be given to users to join the network.

VI. Experimental Results

Our experiments include a *Wi-Fi access point* and devices that connect to that access point.

For the access point, we have conducted the experiment on OpenWRT-based routers [9]. OpenWRT is an open-source Linux-based operating system for embedded devices, whose main usage is for Wi-Fi equipment such as routers and switches; with OpenWRT, one can modify the Wi-Fi stack and add custom packages to provide an even more customized experience. Specifically, we have used *TP-Link Archer C7 V5* [23] router, although any router that supports OpenWRT, supports also our modifications. For ease of implementation, we have also connected an Arduino with an ESP8266 Wi-Fi module, to the TP-Link router. With Arduino, it is easier to generate unique SSIDs, such that the inter-beacon time is minimized (about 2 milliseconds between beacons compared to hundreds of milliseconds in other modules). We note that the mapping between devices and SSIDs/passwords, managing SSIDs, and passwords, was done at the AP with HostAPD [22]—a user-space daemon for access point and authentication server.

The devices that we have examined were either Android devices (Android 6.0 with persistent MAC address, as well as Android 11.0 where MAC addresses are randomized daily) or iPhone devices (running iOS 14.x and iOS 15.x).

A. The number of visible SSIDs

When an SSID is used per device, the number of SSIDs supported by the AP determines the number of users that can connect to the network. This number is determined by both the frequency of beacons (in the access point) and by the time the connecting device needs in order to scan these beacons and identify all the SSIDs. Moreover, the connecting device may declare (namely, after a few seconds of not observing beacons) that an SSID is no longer available and removes it from its SSID list.

As mentioned before, we are using Arduino with ESP8266 to transmit the beacons due to its ability to send beacons in high frequency. However, as the frequency of beacons increases, their load on the Wi-Fi networks increases until it becomes prohibitive. On the other hand, if the frequency is too low (namely, more than a few seconds between beacons of the same SSID), the device's Wi-Fi manager might remove the SSID from its list (implying they in fact become invisible). Thus, in order to eliminate this problem, we require that within 1 second, all SSIDs beacons will be transmitted.

Moreover, as shown in Fig. 4, we have conducted an experiment with Galaxy S21 Android phone that uses the latest Broadcom Wi-Fi chip (Wi-Fi 6E) to measure the required scan cycles until we cover the transmitted beacons. As the number of concurrent SSIDs increases, the required number of scans also increases, implying a longer time for a user to find its required beacon that contains the specific unique SSID that was used to connect in the past. As each scan takes 12 seconds in Galaxy S21, we stress that at most 20 SSIDs can be supported (to allow scanning all of them within 36 seconds). This implies an inter-beacon time of 50 milliseconds that does not pose any significant load on the network. This limits the network to 20 users.

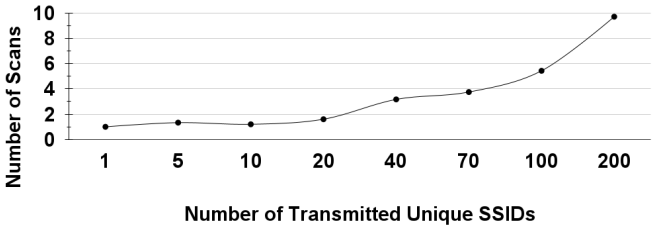

Fig. 4: The number of scans that are required on average to see the whole set of the transmitted unique SSIDs. Each scan takes around 12 seconds in Galaxy S21.

In a more realistic scenario, we assume that a subset of the users is using the AP at a given time, so we do not have to transmit all the possible unique SSIDs that we have in the system. Instead, we will give different time quotas for different SSIDs, according to their class: SSID used by active devices, SSID used by the non-active device (that are currently not online), and unused SSIDs reserved for new devices. With such time quotas, we can push the limit on the number of users that can connect to the network to a few tens; yet, unlike our password-based approach, this approach does not scale to hundreds of users.

B. Network speed

One concern of transmitting multiple and frequent beacons is that their transmission adversely affects the network speed. However, this depends on the channel in which the beacons are transmitted. Specifically, the naïve solution is to transmit the beacons on the same channel of the data, thus reducing data bandwidth. An alternative solution is to use beacons on a channel other than the data channel; as AP listens only to one channel (namely, the data channel), it might miss all probes. To circumvent this issue, we change the channel ID *on the beacon header* to the data channel ID. This results in beacons sent on one channel but responses (and afterward, data) sent on another channel. Such isolation between beacons and data improves the network speed and implies that our frequent beacon transmissions have a negligible effect on performance.

C. Identification using multiple passwords

This method yields a 100% success rate as long as the users do not share the password with each other. To help to prevent the users from sharing the plain-text password, the generated passwords are long and random making it tedious for users to remember; moreover, passwords can be hidden from users since the primary way of connecting to the network is by QR codes.

We have also checked the impact on the network speed, which turns to be negligible as suitable PSK is found only when the device is connecting (or re-connecting) to the network and is not related to the data exchange between the AP and the client. Furthermore, up to 4,000 unique passwords can be used before the client drops the authentication flow due to a timeout (We note that this number may vary, as it depends

on the access point used.) Finally, the PSK list is generated only once per AP boot, and therefore, can be done lazily.

VII. Conclusions

In this paper, we have developed two methods for device identifications to circumvent the problems that arise from the emerging MAC randomization trend. Our approach is built on features of 802.11 Wi-Fi standard [19]—namely, using multiple SSIDs and multiple passwords per SSID—and requires only small modifications to the access point code (no modification is required from the connecting device). We note that, if needed, the two approaches can be used together, to push even further the number of users that can be successfully identified in a deterministic manner.

When comparing the proposed methods, using a unique password per device is a clear winner in most cases: it is easier to use and easier to implement, it is seamless to the user (it sees only a single SSID), and yet always succeeds to identify the device in spite of its randomized MAC address. However, these methods require a mechanism to generate and distribute unique passwords to the users, which sometimes do not exist. Moreover, in case Wi-Fi password sharing mechanisms are in use (as available in the latest iOS version [24]), this method can result in two devices getting the same password, and therefore, being identified as one. Thus, our unique SSID per device approach complements the password-based approach to allow device identification also in such cases.

Acknowledgements

We would like to thank Shaul Levi, Ray Oei, and Muhammad Ali from Gamgee B.V for the useful discussions on the problem. This research was supported in part by a research grant from Cisco Systems and by the Federmann Cyber Security Research Center in conjunction with the Israel National Cyber Directorate.

References

[1] B. Bonné, A. Barzan, P. Quax, and W. Lamotte, "Wifipi: Involuntary tracking of visitors at mass events," in *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2013, pp. 1–6.

[2] K. Shubber, "Tracking devices hidden in london's recycling bins are stalking your smartphone," Aug 2013. [Online]. Available: https://www.wired.co.uk/article/recycling-bins-are-watching-you

[3] "Use private wi-fi addresses in ios 14, ipados 14, and watchos 7," Nov 2020. [Online]. Available: https://support.apple.com/en-us/HT211227

[4] A. Inc., "ios and ipados usage," 2022. [Online]. Available: https://developer.apple.com/support/app-store/

[5] "Privacy: Mac randomization in android." [Online]. Available: https://source.android.com/devices/tech/connect/wifi-mac-randomization

[6] "Android version market share worldwide." [Online]. Available: https://gs.statcounter.com/os-version-market-share/android

[7] "Android os version market share over time," Apr 2021. [Online]. Available: https://www.appbrain.com/stats/top-android-sdk-versions

[8] D. Zehavi, "Blog: The mac address is going away. now what?" 2020, https://levl.tech/resources/mac-address-is-going-away-now-what/.

[9] R. Brown, "Openwrt - routers operating system." [Online]. Available: https://openwrt.org/

[10] T. D. Vo-Huu, T. D. Vo-Huu, and G. Noubir, "Fingerprinting wi-fi devices using software defined radios," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, pp. 3–14.

[11] L. C. C. Desmond, C. C. Yuan, T. C. Pheng, and R. S. Lee, "Identifying unique devices through wireless fingerprinting," in *Proceedings of the first ACM conference on Wireless network security*, 2008, pp. 46–55.

[12] M. Cristea and B. Groza, "Fingerprinting smartphones remotely via icmp timestamps," *IEEE Communications Letters*, vol. 17, no. 6, pp. 1081–1083, 2013.

[13] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why mac address randomization is not enough: An analysis of wi-fi network discovery mechanisms," in *Proceedings of the 11th ACM on Asia conference on computer and communications security*, 2016, pp. 413–424.

[14] M. Cunche and C. Matte, "On wi-fi tracking and the pitfalls of mac address randomization," *Nouveaux défis de l'Internet des Objets: Interaction Homme-Machine et Facteurs Humains*, pp. 18–18, 2016.

[15] H. Li, H. Zhu, and D. Ma, "Demographic information inference through meta-data analysis of wi-fi traffic," *IEEE Transactions on Mobile Computing*, vol. 17, no. 5, pp. 1033–1047, 2017.

[16] C. Matte, M. Cunche, F. Rousseau, and M. Vanhoef, "Defeating mac address randomization through timing attacks," in *Proceedings of the 9th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, 2016, pp. 15–20.

[17] A. Zhang, Y. Yuan, Q. Wu, S. Zhu, and J. Deng, "Wireless localization based on rssi fingerprint feature vector," *International Journal of Distributed Sensor Networks*, vol. 11, no. 11, p. 528747, 2015.

[18] Zxing, "Zxing - QR code generator." [Online]. Available: https://github.com/zxing/zxing/wiki/Barcode-Contents{#}wi-fi-network-config-android-ios-11

[19] "Ieee 802.11-2012 - 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 (mac) and physical layer (phy) specifications," 2012. [Online]. Available: https://standards.ieee.org/standard/802_11-2012.html

[20] "Wpa3 authentication," 2018. [Online]. Available: https://spectrum.ieee.org/tech-talk/telecom/security/everything-you-need-to-know-about-wpa3

[21] D. Harkins, "Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks," in *2008 Second International Conference on Sensor Technologies and Applications (sensorcomm 2008)*. IEEE, 2008, pp. 839–844.

[22] "hostapd: ieee 802.11 ap, ieee 802.1x/wpa/wpa2/eap/radius authenticator," 2002. [Online]. Available: https://w1.fi/hostapd/

[23] "Ac1750 wireless dual band gigabit router." [Online]. Available: https://www.tp-link.com/us/home-networking/wifi-router/archer-c7/

[24] "How to share your wi-fi password from your iphone, ipad, or ipod touch," Sep 2021. [Online]. Available: https://support.apple.com/en-il/HT209368