Over-the-Air Runtime Wi-Fi MAC Address Re-randomization

Hongyu Jin
Networked Systems Security Group
KTH Royal Institute of Technology
Stockholm, Sweden
hongyuj@kth.se

Panos Papadimitratos
Networked Systems Security Group
KTH Royal Institute of Technology
Stockholm, Sweden
papadim@kth.se

ABSTRACT

Medium Access Control (MAC) address randomization is a key component for privacy protection in Wi-Fi networks. Current proposals periodically change the mobile device MAC addresses when it disconnects from the Access Point (AP). This way frames cannot be linked across changes, but the mobile device presence is exposed as long as it remains connected: all its communication is trivially linkable by observing the randomized yet same MAC address throughout the connection. Our runtime MAC re-randomization scheme addresses this issue, reducing or eliminating Wi-Fi frames linkability without awaiting for or requiring a disconnection. Our MAC re-randomization is practically 'over-the-air': MAC addresses are re-randomized just before transmission, while the protocol stacks (at the mobile and the AP) maintain locally the original connection MAC addresses - making our MAC layer scheme transparent to upper layers. With an implementation and a set of small-scale experiments with off-the-shelf devices, we show the feasibility of our scheme and the potential towards future deployment.

KEYWORDS

MAC spoofing, privacy, unlinkability, mix-zone

1 INTRODUCTION

Wi-Fi networks are a corner-stone for Internet access, at homes, offices, and public areas, enabling seamless connectivity for laptops, smartphones, smart wearables, etc., each identified by a Medium Access Control (MAC) address, a globally unique identifier assigned to each Network Interface Card (NIC). MAC addresses are hard-coded by the NIC manufacturers. When a device transmits Wi-Fi frames, it specifies its MAC address in the Source Address (SA)/Transmitter Address (TA) field in the frame header [1]. Transmitted (Tx) Wi-Fi frames can be easily monitored by NICs that can operate in 'monitor/promiscuous' mode. A monitoring device can track devices in an area by observing received (Rx) Wi-Fi frames with the same SA/TA MAC address. Multiple monitoring devices can track devices (and, consequently, their users) across numerous points-of-interest targeted users (devices) are likely to visit.

MAC address randomization [4, 15] emerged as a countermeasure to such so-called sniffing attacks. Instead of the hard-coded MAC address, the driver/firmware reads software-generated MAC addresses, refreshed periodically or based on specific conditions (e.g., per SSID or per connection). With encryption covering Wi-Fi frame payloads, device transmissions are still easily linkable while the same MAC address is used. Once the device is assigned a new randomized MAC address, a local eavesdropper cannot link traffic to that by the same device with an earlier randomized MAC address. Thus, such relatively shorter-term MAC addresses significantly reduce linkability of traffic and thus user presence/activities.

The level of privacy protection (unlinkability) is not high: while connected to an Access Point (AP), given the device uses the same MAC address throughout the connection, inference attacks on user activities are possible. More frequent MAC randomization is possible by manually disconnecting and re-connecting to APs, at the expense of inefficient communication, without fundamentally solving the problem at hand (see Sec. 2).

This motivates our proposal of over-the-air runtime MAC re-randomization, that is, MAC randomization within and repeatedly throughout the connection/session without affecting the communication quality, i.e., without disconnections and no data rate or reliability reduction. Our scheme only randomizes the MAC addresses in the headers of Tx frames on the verge of transmission, while the Rx devices recover the original/actual (termed *base*) MAC addresses immediately upon frame reception. This makes our scheme transparent to upper layer protocols. At the same time, an eavesdropper only sees the ephemeral re-randomized MAC addresses in the Tx frame headers. The re-randomization is performed synchronously by all stations connected to an AP, drastically growing the anonymity set to be equivalent to the set of connected devices.

We provide a detailed description of our scheme, including Sequence Number (SN) and nonce (this term is used interchangeably with Packet Number (PN) and Initialization Vector (IV)) reset approaches (Sec. 3), and analyze its security and privacy (Sec. 4). We implement our scheme with off-the-shelf Wi-Fi NICs at the driver level (Sec. 5). A set of small-scale experimental results shows that our scheme does not deteriorate communication performance (Sec. 6), before our concluding remarks (Sec. 7).

Related Works: To the best of our knowledge, the only existing work that addresses MAC address rotation during Wi-Fi connection is RoMA [6]. However, RoMA does not provide seamless MAC address rotation from the perspective of the AP, because different IP addresses are used for virtual interfaces; the number of rotating MAC addresses during a connection is limited by the maximum number of supported Wi-Fi NIC virtual interfaces. Cryptographically generated [5] randomized addresses were not Wi-Fi compatible. The current MAC randomization protocol, work-in-progress with an IETF draft [15], defines 46-bit randomization (with 2 fixed bits defining unicast and locally administered addresses). The latest OSes implement their variants of MAC randomization, with major differences reflected on randomized MAC address lifetime (e.g., per SSID or per connection) [4, 8, 15]. We refer readers to [4, 8] for a detailed introduction to MAC randomization policies.

2 PROBLEM STATEMENT

MAC randomization has been adopted by the latest mobile devices (e.g., Apple and Android based [7, 13]) to prevent tracking connections to different APs/SSIDs in public areas. However, the same

randomized MAC address is used throughout a connection. This way, an eavesdropping/sniffing attacker can easily identify a device (period of) presence: by eavesdropping the appearance and disappearance of a specific mobile MAC frame address in the network. For example, a person at work connects her/his device to an AP in the office area having the same MAC address throughout the day. Therefore, randomized MAC addresses do not prevent inferring user activity or linking the same randomized MAC address over different connections (e.g., over multiple days).

A straightforward improvement is to periodically force disconnection from the AP and re-connection with a fresh randomized MAC address; triggered manually by the user or automatically by the system. However, such an approach would not be very effective because the attacker could easily link the two (old and new) MAC addresses by observing the appearance of a new MAC address in the network immediately after the disappearance of an old MAC address.

Inspired by the mix-zone approach [3, 10] for identity and location protection, where users change pseudonyms in a mix-zone synchronously, devices connected to an AP can synchronize their disconnection and re-connection with freshly randomized MAC addresses. Such synchronized actions can be performed periodically to render MAC address linking based on the appearance/disappearance timing impossible; because multiple MAC addresses would disappear and appear at the same time. However, such forced dis- and re-connections could cause substantial delays at each reconnection. Moreover, some devices might refuse to disconnect and change MAC address at a given point - e.g., due to ongoing communication (e.g., important video calls) of critical nature - at the expense of remaining linkable. Having several devices refraining from changing their MAC addresses shrinks the anonymity set for devices that did change their address.

These limitations motivate us to propose the runtime MAC re-randomization scheme described in Sec. 3: an effective scheme allowing fast changes across randomized MAC addresses without degrading communication performance, and preventing frame linking based on continuous Sequence Numbers (SNs) and WPA2/3 nonces. Our scheme protects against any eavesdroppers (including devices connected to the same AP) and retains the same level of privacy protection against the connected AP as that of classic MAC randomization [4, 8, 15]. Fingerprinting attacks based on traffic analysis [8, 9, 11] are out of the scope of this paper and part of future work.

3 OUR SCHEME

We build on top of MAC randomization, to ensure that a roaming user/mobile device (termed *station*) that connects to a new AP selects a new randomly chosen MAC address. We term this the *base MAC address* for the current connection to the AP. The base address of every station is known and kept by the AP, clearly, not needing to change/randomize further. The first novelty of our scheme is to not use the base address but rather assign a new periodically *re-randomized MAC address* for Tx frames. Re-randomized addresses are computed with a hash function based on a common secret, the WPA2/3 Pairwise Transient Key (PTK), between each station and the AP (Sec. 3.1). Computations are performed on both the AP and each station independently, based on identical inputs, thus resulting in identical new, re-randomized MAC addresses. For each new re-randomization, SNs and WPA2/3 nonces are also reset to prevent linking based on these (otherwise continuous) values (Sec. 3.2). In order to prevent nonce reuse, we propose a controlled nonce reset to make sure the 48-bit nonce wrap interval is significantly longer than a PTK lifetime. Our scheme is transparent to the upper layers because it ensures that only the re-randomized MAC addresses are exposed over the air, while the protocol stacks operate with the base MAC address (Sec. 3.3).

3.1 MAC Computation

Our scheme assumes the base MAC address randomization is handled by the station itself. This can be easily supported by mobile devices nowadays. For example, the *macchanger* tool can randomize the MAC address in Linux systems. Although current default Apple and Android policies don't change MAC addresses for each new connection to the same SSID without user intervention (e.g., by 'forgetting' the network or resetting the network settings) [7, 13], it is clear that this can be supported (currently restricted by system policies).

After the AP and a station are connected (thus, shared keys computed), the MAC re-randomization is performed periodically, every T seconds; T is the lifetime of each re-randomized MAC address. Transitions to a new MAC address are synchronized across all stations connected to an AP. The computation of the re-randomized MAC address is based on a hash function with three input values, carried out by the AP and each station locally, based on the corresponding pair-wise shared secret (key).

- The first input is the base (static) MAC address, used to initiate the current connection to the AP.
- The second input is the PTK computed during the authentication phase [14], only known by the AP and the corresponding station, thus the shared secret for each AP-station pair. In WPA2, a PTK (thus the re-randomized MAC addresses) can be derived by any eavesdropper that monitored a connection from its very beginning (i.e., recorded all handshake frames) and knows the passphrase. Thus, WPA2 is not fully secure per se and cannot be used to enhance privacy beyond the standard MAC randomization approach [4, 8, 15] with our scheme. However, the Dragonfly/Simultaneous Authentication of Equals (SAE) handshake protocol in WPA3 requires active participation to compute the PTK, essentially preventing the eavesdropper from computing the PTK and consequently the re-randomized MAC addresses of other stations, even if she were present from the beginning of their connection [14].
- The last input is a unique index of the T interval, i.e., the validity period of the re-randomized MAC address. The index is computed with $\lfloor t_{epoch}/T \rfloor$, where t_{epoch} is the Unix (or epoch) time in seconds (counted from January 1, 1970).

The three values are concatenated and hashed to produce a digest whose first six bytes are used as the basis for a random MAC address. In order to derive a valid MAC address, we have to set to the first byte bit-0=0 (unicast) and bit-1=1 (locally administered) [1, 15]. With this adjustment, the six bytes are ready to be used as the newly re-randomized MAC address. Each station only needs to maintain

the currently used re-randomized MAC address (for the current T interval) together with the base address, and convert between the base one and the re-randomized one during frame transmission and reception. The AP needs to (i) compute for each station its re-randomized MAC address using the same inputs and function as described above, and (ii) maintain a table of base and re-randomized MAC addresses for MAC conversion for Wi-Fi frames sent/received to/from the stations. The AP does not need to randomize (let alone re-randomize) its MAC address because there is no need to protect the AP privacy.

3.2 Sequence Number and Nonce Reset

To prevent linking Wi-Fi frames based on their continuous SNs and nonces, we have to reset them along with each MAC address re-randomization. The SN is simply reset to 0, a standard operation that usually happens when the 12-bit incremental SN reaches its maximum value (i.e., 4095; or 2047 allowed on some devices [4]). In our scheme, the SN reset happens together with the MAC re-randomization too.

In WPA2/3, 48-bit nonces (essentially PNs) are used to ensure the same plaintext does not result in the same ciphertext [12]. Unlike SNs, resetting the nonce to 0 would create a vulnerability, due to nonce reuse. A straightforward approach is to set the nonce to a random value with the new MAC address, and increment as usual, from that point until the next MAC re-randomization. The large enough 48-bit nonce space would naturally render nonce reuse highly unlikely. This can be augmented by checking whether frames within the upcoming T interval could potentially reuse earlier own nonces (depending on the bit rate and frame size) and randomize again if needed. This has a privacy implication: frames using previously seen nonces shrink the anonymity set because frames with identical nonces do not originate from the same station (MAC addresses).

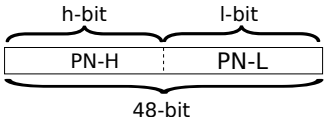

Figure 1: 48-bit nonce/PN comprises h/l-bit PN-H/PN-L.

To thwart nonce reuse and the potential privacy issue above, we propose a controlled nonce/PN reset approach. As shown in Figure 1, we split the PN field into PN-H and PN-L, h-bit and l-bit long, respectively, where $h + l = 48$. For the corresponding T interval, we set PN-H = $\lfloor t_{epoch}/T \rfloor \ mod\ 2^h$, and PN-L is simply reset to 0 with each new MAC re-randomization. In other words, PN-H is incremented by 1 every T, and PN-L is the nonce space used for the frames within the T interval. h and l values can be chosen based on Eq. (1), where L_{frame} is the average frame size, and the interval between two PN wraps is $2^h * T$. We show in Sec. 4 that nonce reuse is essentially impossible with practical frame sizes and bit rates.

$$l = \lceil log_2(\frac{Bitrate * T}{L_{frame}}) \rceil,\ h = 48 - l \tag{1}$$

3.3 Over-the-Air MAC Conversion

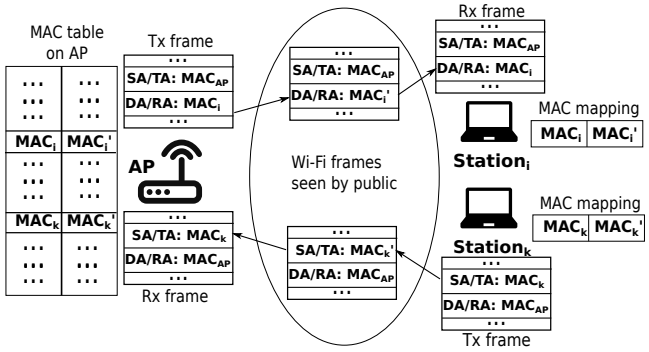

Figure 2: Over-the-air MAC conversion.

Figure 2 illustrates the over-the-air MAC conversion. The AP maintains a MAC address table, one row per connected mobile station; with the base MAC addresses used for establishing connections in the left column and the currently valid re-randomized MAC addresses in the right column. Immediately before transmitting a Tx frame, the AP performs a lookup in the MAC address table with the Destination Address (DA)/Receiver Address (RA) address (a base address). If there is a match, the DA/RA address is converted to the corresponding re-randomized MAC address. For an Rx frame, the AP lookup is carried out with the SA/TA address at the right column of the MAC table, immediately upon Rx frame reception. If there is a match, the SA/TA address is converted to the corresponding base MAC address, transparently to the upper layers, so that the Rx frame payload is properly decapsulated. Each station only maintains its own re-randomized MAC addresses, apart from the base address. Similarly, each station converts the SA/TA address on a Tx frame and the DA/RA address (that matches own re-randomized addresses) on an Rx frame. In summary, only the re-randomized MAC addresses are exposed over the air while the AP and station protocol stacks use the station base MAC address. The MAC address conversion has to be performed before the Frame Check Sequence (FCS) is calculated, usually done at the hardware level; otherwise, the corresponding frames would be deemed corrupted and dropped by receivers.

4 SECURITY AND PRIVACY ANALYSIS

We provide a security analysis on nonce reuse and a privacy analysis on the unlinkability of MAC addresses, based on the unencrypted MAC header fields over the synchronized re-randomizations. Traffic (metadata) analysis or device fingerprinting are out of scope.

4.1 Preventing Nonce Reuse

Nonce reuse with a same PTK is prevented by choosing proper h and l values based on system parameters (Eq. (1)). We explain, with the help of a demanding example, why nonce reuse is practically impossible with our scheme. Figure 3 shows the interval between two nonce wraps (i.e., PN-H wraps to zero) considering two extreme values for average data frame size (50 $bytes$, near minimal) and bit rate (10 $Gbps$). Even so, the wrap happens every 60 days or more,

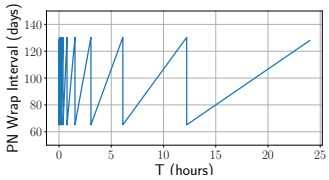

Figure 3: Interval between two PN wraps, as a function of T ($1s \sim 24h$), for 50 *byte* data frames and 10 *Gbps* bit rate.

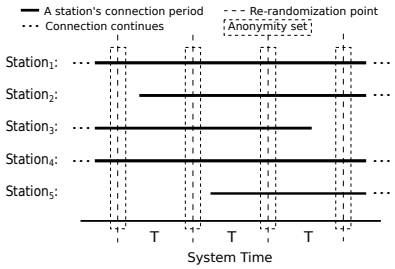

Figure 4: MAC re-randomization example with five stations.

much longer than the expected lifetime of a PTK. With larger frame sizes and lower bit rates, the number of theoretically possible frame transmissions would be lower. This implies lower l and higher h values, thus even longer wrap intervals than those in Figure 3.

4.2 MAC Address Unlinkability

Figure 4 shows an example of MAC re-randomization for five station connections. As per Sec. 2, if stations re-randomize/change their MAC address independently, thus in all likelihood at distinct points in time, an eavesdropper could easily link any two consecutive MAC addresses of each station (possibly based on continuing activity across the MAC address change). However, with our synchronized MAC re-randomization, the anonymity set after each MAC change would be the same as the number of connected stations; thus linking the old and new MAC addresses of the same device would be hard. The initial connection to the AP always stands out due to the handshake frames, but the subsequent frames will not be linkable based on MAC addresses. Hiding a disconnection is possible if a station leaves the network without sending disconnection frames and T is relatively short; thus the eavesdropper would not know whether the station left or it was just idle, without any communication for a short period until the next re-randomization point. This would incur slightly more overhead on the AP, needing to maintain the disconnected stations until their removal due to inactivity timeouts. Moreover, unified SN and nonce resets (i.e., reset to identical values by each station and the AP) make linking based on these (otherwise continuous) values impossible. Last but not least, we emphasize our scheme does not enhance privacy against connected APs: they learn the same information the standard MAC randomization provides.

5 IMPLEMENTATION

We implement our MAC re-randomization and run experiments on four identical small form factor computers with Atheros AR5414 NICs [2], Intel Atom D510 dual-core 1.66GHz CPUs, and 1 GB RAM. They are all installed with Debian OS 11.8, with Linux kernel version 5.10.0-26. AR5414 is a legacy NIC model that supports only IEEE 802.11 a/b/g, not the latest IEEE 802.11 n/ac/ax (Wi-Fi 4/5/6). However, we choose this available to us model due to Linux support through the fully open-source driver, ath5k, without any non-free binary firmware. Therefore, we have full control of the device through its driver. This makes AR5414 perfect for testing the feasibility of our approach, as a first step towards a larger scale deployment and experiments with the latest devices. We implement our scheme by modifying the ath5k driver and the mac80211 module provided through Linux backports driver version 5.10.168[1].

5.1 MAC Re-randomization

Our scheme requires each station to randomize its base MAC address used for connection through system tools, e.g., *macchanger*, to set a randomized MAC address for the wireless interface in Linux. For the re-randomized MAC computation after the established connection, we use the SHA256 function from the Linux kernel crypto module to compute hash digests, as described in Sec. 3, retaining the first six bytes. Then, we set bit-0=0 and bit-1=1 of the first byte respectively, to ensure the MAC address is valid. At the AP, we use the Linux kernel hash table[2] to maintain the MAC address table. For each station, the AP maintains two entries (i.e., key-value pairs) in its hash table. The keys of the two entries are the base MAC and the re-randomized MAC addresses for each connected station. The values of the two entries are identical: a *struct* comprises both the base and the re-randomized MAC addresses. The two entries are used to look up MAC addresses to convert to for Tx or Rx frames. We declare a 10-bit hash table in our implementation, storing 2^{10} entries for at most 512 stations. With AR5414, FCS can be correctly calculated by the hardware after the Tx frame MAC address conversion by the driver.

5.2 Resetting Sequence Number and Nonce

We found that the AR5414 would assign SNs on the hardware by overwriting the driver-assigned SNs. Therefore, we had to disable hardware-assigned SNs to respect the driver-assigned SNs (Sec. 3.2). We were able to reset nonces in the driver without any issue.

5.3 Enabling ACK/RTS/CTS Frames

5.3.1 ACK Frames. Time-sensitive ACK frames acknowledge data frames, triggered upon reception by the hardware, not the driver (the hardware-driver communication would introduce extra delay). Usually, NICs in *station* mode only acknowledge frames that specify RA/DA as the Rx station local addresses. In our implementation, given the transmitted frames carry the re-randomized MAC addresses that do not match the local addresses, the Rx hardware would not trigger ACK frames, resulting in a series of retransmissions of non-ACKed data frames.

This can be solved by setting a BSSID mask (originally used to enable multiple virtual interfaces) on AR5414 [2]: after each MAC re-randomization, setting a BSSID mask (computed with inputs the base and the newly re-randomized MAC addresses) ensures ACK frames are properly triggered. However, this opens up a privacy

[1] https://cdn.kernel.org/pub/linux/kernel/projects/backports/stable/
[2] https://lwn.net/Articles/510202/

issue. An attacker could inject frames by specifying the (eavesdropped) base MAC address of a station as the DA/RA. These injected frames would trigger ACK frames and expose the presence of the device with the targeted (sought by the adversary) base MAC address.

In order to fundamentally solve the problem, we directly update the main MAC address register [2] on the hardware with the re-randomized MAC address. This allows ACK frames to be triggered only for the currently used re-randomized MAC address, not for the base MAC address. This is the expected correct behavior of our scheme: stations should not respond to any frame destined for their base MAC addresses. Otherwise, the presence of the device with the specific base MAC address would be exposed.

5.3.2 RTS/CTS Frames. Similar to ACK frames, time-sensitive Request to Send (RTS)/Clear to Send (CTS) frames[3] are also handled by the hardware [2]. A connected station sends an RTS, awaits for a CTS sent in response by the AP and only then transmits data. Most APs would disable RTS/CTS frames due to performance uncertainty; e.g., OpenWrt disables RTS/CTS (confirmed by installing the OpenWrt firmware on two different ASUS router models), and dd-wrt sets the default RTS threshold to 2347 *bytes*[4] (i.e., effectively disabled, given the maximum Wi-Fi frame size being 2346 *bytes*). However, in order to demonstrate the usability of RTS/CTS frames in our implementation, we do enable RTS/CTS frames by setting a moderate threshold on the stations. We see RTS/CTS frames properly triggered with the re-randomized MAC addresses written to the hardware register (see Sec. 6).

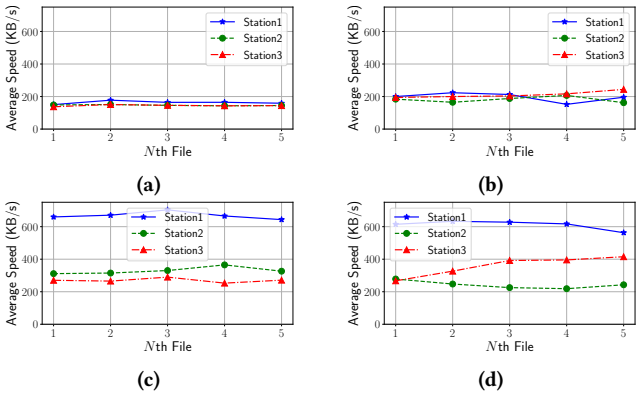

Figure 5: (a, c) Vanilla driver and (b, d) our scheme. Average copy speed of 50 *MB* files to the server (public IP) with (a, b) RTS/CTS off and (c, d) RTS_threshold=500 *bytes*.

```
interface=<INTERFACE>
driver=nl80211
ssid=<SSID>
hw_mode=g
channel=9
# The next two lines enable WPA3
wpa=2
wpa_key_mgmt=SAE
rsn_pairwise=CCMP
wpa_passphrase=<PASSWORD>
ieee80211w=2
```

Listing 1: *hostapd*

```
network={
    ssid=<SSID>
    psk=<PASSWORD>
    key_mgmt=SAE
    ieee80211w=2
}
```

Listing 2: *wpa_supplicant*

6 EXPERIMENTAL EVALUATION

We find that our scheme and the modified driver do not deteriorate the communication performance in the experimental setup. Due to the legacy hardware and the limited number of devices available, we do not present an extensive evaluation. Our goal is to show the feasibility of our scheme by comparing with the vanilla driver, serving as a stepping stone to a larger scale experiment with the latest devices in the future.

6.1 Experimental Setup

One device acts as the AP and the other three act as stations, deployed in a small office. $Station_1$ is in line-of-sight, and $Station_2$ and $Station_3$ are in non-line-of-sight of the AP. We use *hostapd* to set up the AP and *wpa_supplicant* for the stations. The AP is configured with WPA3-Personal. The stations are assigned static IPv4 addresses, and access the Internet wirelessly, routed through the AP. We set $T = 30s$, and both h and l to 24, thus 24-bit PN-H/L. The values are sufficient to prevent nonce reuse for the bit rate supported by the devices. We provide the full *hostapd* and *wpa_suppplicant* configurations below.

6.2 Wi-Fi Frame Monitoring

We also place another device in the same space, with its wireless interface set to *monitor* mode on the same channel as the AP and three stations. Real-time monitoring data through Wireshark[5] confirms that the MAC addresses are refreshed every $T = 30s$. Once a new T interval is reached, older MAC addresses no longer show up in the captured frames, while the ongoing communication is undisrupted. We also observe that ACK/RTS/CTS frames, handled by the hardware, are properly triggered thanks to the hardware overwritten MAC address registers.

6.3 Performance Evaluation

In order to compare the performance of the vanilla driver and our implementation, we conduct several experiments, copying files between a local server and the stations, and downloading files from the Debian website. We repeatedly run *scp* copy or *wget* download commands (i.e., a new command is executed after the previous one concludes), to make sure all three stations communicate with the AP simultaneously. We show the average speed for five operations (commands run) for each station.

Figure 5 shows the results for repeatedly copying (uploading) a 50 *MB* file from the stations to a local server with a public IP address. We see the results are similar for the two driver versions, while enabling RTS/CTS improves the performance. The improvement is more evident for $Station_1$, which is in line-of-sight of the AP.

[3]They are intended to make the shared medium access more efficient, solving the hidden and exposed terminal problems.
[4]https://wiki.dd-wrt.com/wiki/index.php/Advanced_wireless_settings
[5]https://www.wireshark.org/

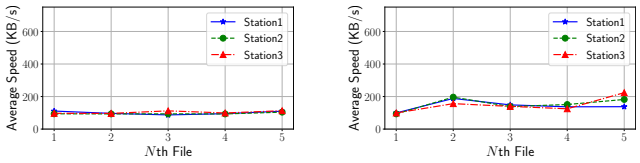

Figure 6: (a) Vanilla driver and (b) our scheme. Average copy speed of 50 *MB* files from the server (public IP).

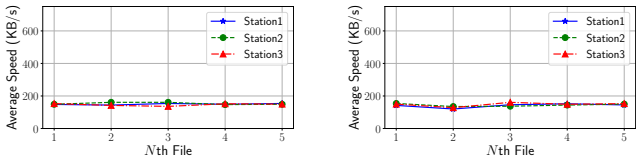

Figure 7: (a) Vanilla driver and (b) our scheme. Average download speed of 60 *MB* images from the official Debian website.

Figure 6 shows the results for stations copying (downloading) a 50 *MB* file from the same local server. In this scenario, we didn't enable RTS/CTS on the stations because the majority of heavy transmissions are from the AP to the stations. The results are again similar for the two driver versions. For completeness in this context, we also evaluate the speed of downloading a 60 *MB* ISO file[6] from the official Debian website over the Internet. Figure 7 shows the results for the two driver versions are similar too.

The goal is achieved: the modified driver performs very similarly to the vanilla driver. We do not attempt to explain performance results in further detail, because the intent is not to evaluate the latest Wi-Fi hardware performance or an optimized AP (unlike our device that acted as AP). Extensive performance evaluation is left for future work, after a successful implementation of our scheme on the latest NICs that support IEEE 802.11 n/ac/ax (Wi-Fi 4/5/6).

7 CONCLUSION AND FUTURE WORK

Our scheme re-randomizes MAC addresses while the stations are connected to the AP without disrupting ongoing communication. Synchronized MAC address transitions make linking based on timing information difficult, with the size of the anonymity set being the number of connected stations. SNs and nonces are also reset to ensure unlinkability. We show the feasibility of our scheme with a set of small-scale experiments. The benefit is a very significant reduction of user exposure to any eavesdropper: any long-lived connection/session with an AP essentially dissolves into a multitude of unlinkable ephemeral connections.

An immediate next step is to implement and evaluate our scheme on the latest NICs, with the challenge of using non-free binary firmware. The T value of our scheme is currently hard-coded into the driver; an extension is to have connected mobile stations receive the re-randomization schedule from the AP. Our scheme highly depends on time synchronization, especially for high bit rates (i.e., short frame intervals), otherwise station MAC address transitions would be easily linkable. Countermeasures for loosely synchronized devices would be necessary, along with a full-blown unlinkability analysis. Adaptation of our scheme in Wi-Fi ad-hoc mode is also a part of future work.

[6]https://cdimage.debian.org/cdimage/archive/5.0.10/powerpc/iso-cd/debian-5010-powerpc-businesscard.iso

ACKNOWLEDGMENTS

This work was supported by the Swedish Research Council project 2020-04621.

REFERENCES

[1] IEEE Std 802.11-2020. 2021. 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. (2021).
[2] AR5414 Dual-Band, Multi-Mode MAC/BB/Radio for IEEE 802.11 a/b/g Wireless LAN. 2005. https://atoma.spb.ru/sites/default/files/documents/ar5414_data_sheet_05_04.pdf
[3] Alastair R Beresford and Frank Stajano. 2003. Location privacy in pervasive computing. *IEEE Pervasive computing* 2, 1 (2003), 46–55.
[4] Ellis Fenske, Dane Brown, Jeremy Martin, Travis Mayberry, Peter Ryan, and Erik C Rye. 2021. Three Years Later: A Study of MAC Address Randomization In Mobile Devices And When It Succeeds. *PoPETs* 3 (2021), 164–181.
[5] Ben Greenstein, Damon McCoy, Jeffrey Pang, Tadayoshi Kohno, Srinivasan Seshan, and David Wetherall. 2008. Improving wireless privacy with an identifier-free link layer protocol. In *ACM MobiSys*. Breckenridge, Colorado.
[6] Johann Hugon, Mathieu Cunche, and Thomas Begin. 2022. RoMA: Rotating MAC Address for privacy protection. In *SIGCOMM Demo*. Amsterdam, Netherlands.
[7] MAC Randomization Behavior. 2023. https://source.android.com/docs/core/connect/wifi-mac-randomization-behavior
[8] Jeremy Martin, Travis Mayberry, Collin Donahue, Lucas Foppe, Lamont Brown, Chadwick Riggins, Erik C Rye, and Dane Brown. 2017. A Study of MAC Address Randomization in Mobile Devices and When it Fails. *PoPETs* 4 (2017), 365–383.
[9] Célestin Matte, Mathieu Cunche, Franck Rousseau, and Mathy Vanhoef. 2016. Defeating MAC address randomization through timing attacks. In *ACM WiSec*. Darmstadt, Germany.
[10] Panos Papadimitratos. 2019. *Mix-Zones in Wireless Mobile Networks*. Springer.
[11] Mathy Vanhoef, Célestin Matte, Mathieu Cunche, Leonardo S Cardoso, and Frank Piessens. 2016. Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms. In *Asia CCS*. Xi'an, China.
[12] Mathy Vanhoef and Frank Piessens. 2018. Release the Kraken: new KRACKs in the 802.11 Standard. In *ACM CCS*. Toronto, Canada.
[13] Wi-Fi privacy. 2021. https://support.apple.com/guide/security/wi-fi-privacy-secb9cb3140c/web
[14] WPA3 Specification. 2022. https://www.wi-fi.org/system/files/WPA3%20Specification%20v3.1.pdf
[15] Juan-Carlos Zúñiga, Carlos J. Bernardos, and Amelia Andersdotter. 2023. *Randomized and Changing MAC Address*. Technical Report. IETF. https://datatracker.ietf.org/doc/draft-ietf-madinas-mac-address-randomization/09/