Security for Mobile Devices

Rules in radio communications

- Assume everyone is listening to everything you send.
- Assume anyone wants to jam you.
- Assume anyone wants to impersonate you.

Rules in radio communications

- Assume everyone is listening to everything you send.
- Assume anyone wants to jam you.
- Assume anyone wants to impersonate you.

Now try to send messages securely.

Number stations

Old school radio security (cold-war era):

https://www.youtube.com/watch?v=qyRpT-u44pU

- Messages are sent by a high-powered fixed installation (shortwave band).
- The transmitter can cover half a continent.
- Only the intended receivers can decode the message.

Number stations

Old school radio security (cold-war era):

https://www.youtube.com/watch?v=qyRpT-u44pU

- Receivers have plausible deniability (for reception, common SW receivers were used).

- Because they don’t have to transmit, they do not reveal their locations.

Number stations

Old school radio security (cold-war era):

https://www.youtube.com/watch?v=qyRpT-u44pU

As good as it gets:

- The message has to go through, even with glitches
  - low throughput must be used.
- Modern communications uses ARQ (Automatic Repeat reQuest)
  - but this means the receiver must reveal his position.

Number stations

Still used (according to the FBI):

http://www.bbc.com/future/story/20170801-the-ghostly-radio-station-that-no-one-claims-to-run

It also fits with a series of **arrests across the United States** back in 2010. The FBI announced that it had broken up a “long term, deep cover” network of Russian agents, who were said to have received their instructions via coded messages on shortwave radio – specifically 7887 kHz.

Number stations

Some are still operational.
Search for **UVB-76**.

Try it at home!

Broadcast area	Russia
Frequency	4625 kHz Shortwave
Format	Repeated buzzing sound
Language(s)	Russian
Former callsigns	УВБ-76, МДЖБ, ЖУО3
Former frequencies	4625 kHz
Owner	Russian Armed Forces
Sister stations	The Pip, The Squeaky Wheel

https://en.wikipedia.org/wiki/UVB-76

Why some locations should be kept secret

While in a secret locations, don’t post check-ins on Facebook.

Should be obvious, right?

Why some locations should be kept secret

While in a secret locations, don't post check-ins on Facebook.

Should be obvious, right?

Right !?

https://www.theguardian.com/world/2018/jan/28/fitness-tracking-app-gives-away-location-of-secret-us-army-bases

Fitness tracking app Strava gives away location of secret US army bases

Data about exercise routes shared online by soldiers can be used to pinpoint overseas facilities

Latest: Strava suggests military users 'opt out' of heatmap as row deepens

▲ A military base in Helmand Province, Afghanistan with route taken by joggers highlighted by Strava. Photograph: Strava Heatmap

Why some locations should be kept secret

While in a secret locations, don't post check-ins on Facebook.

Should be obvious, right?

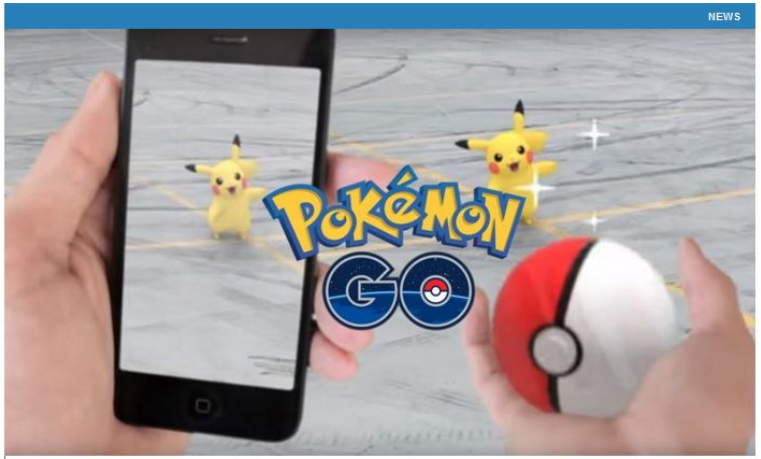
Right !?

Oh, come on…

https://www.cultofmac.com/438174/china-is-worried-pokemon-go-will-uncover-secret-military-bases/

China is worried *Pokémon Go* will uncover secret military bases

BY LUKE DORMEHL • 5:54 AM, JULY 15, 2016



All your base are belong to Pikachu.
Photo: Niantic Labs

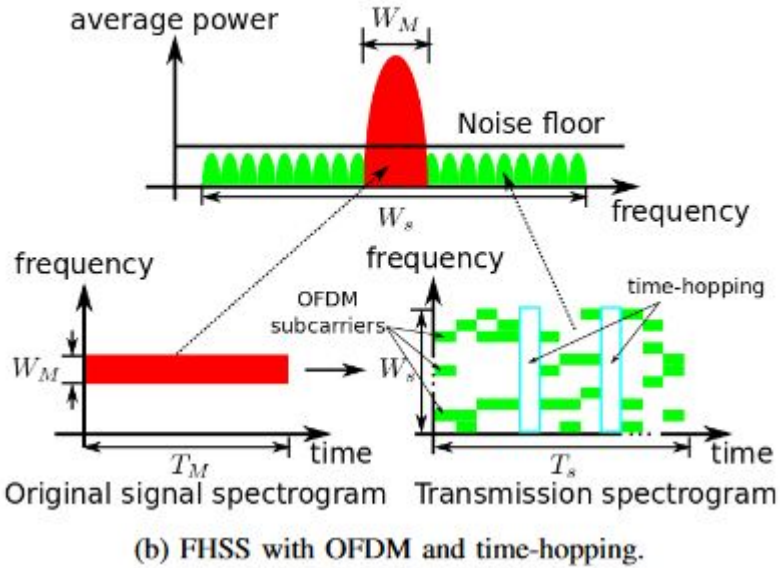
With *Pokémon Go* mania running wild, did you really think the worst that might happen was some would-be Ash Ketchum stumbling across a dead body?

Modern solutions to hide the transmission

A transmitter can either transmit the payload straight ahead, or use a frequency-hopping spread-spectrum solution.

This makes easier to hide the signal under the noise floor.

Demo: http://websdr.yo3ggx.ro:8765/



(b) FHSS with OFDM and time-hopping.

https://arxiv.org/pdf/1506.00066.pdf

Walkie talkies… and the illusion of privacy

Regular walkie talkies (PMR446 devices) have a short range (few blocks, less than 1 Watt) and can be used by anyone.

These also allow “coded channels”... but these are not secure.

Professional radios

These are not secure either (mostly).

The communication is protected by law (enforced by ANRCTI):

- Protected against interference
- Protected against unauthorized transmissions

Demo 173.030MHz .

Professional radios

Shouldn’t this communication be secure?

Professional radios

Shouldn’t this communication be secure?

Not necessarily - the communications are not sensitive, and making them encrypted would also make them more unreliable.

Professional radios

Shouldn’t this communication be secure?

Not necessarily - the communications are not sensitive, and making them encrypted would also make them more unreliable.

But some services still need secure communications.
This is why the police, army, etc. use encrypted channels.

Radios in aviation

Airplanes report their position and bearing over ADS-B.
(plaintext, unencrypted, unauthenticated protocol).

Airplane pilots are trained to rely more on the automated TCAS warnings than on Traffic Control's instructions.
(TCAS = traffic collision avoidance system)

The data must be confirmed by the airplane's radar before taken into account.
Why?

Amateur radios

By law, amateur radio operators must not use encryption.

People use the allocated frequencies just for fun.

Demo: http://aprs.fi/

Satellite communications - ISEE-3

- **International Sun-Earth Explorer-3.**
- Launched in 1978.

- Became the first spacecraft to visit a comet, passing through the plasma tail of Giacobini-Zinner comet within about 7,800 km (4,800 mi) of the nucleus on September 11, 1985.

- NASA suspended routine contact with ISEE-3 in 1997.

Satellite communications - ISEE-3

- “Reboot” effort appeared in 2014.
- No encryption, no authorization needed.
- Obsolete hardware had to be rebuilt.
- Required a 70-meter antenna to transmit the signal.

https://en.wikipedia.org/wiki/Goldstone_Deep_Space_Communications_Complex

Satellite communications - ISEE-3

- Control was regained (by a team of volunteers) for a few days.
- The satellite went unresponsive shortly after.

Satellite communications - Voyager 2

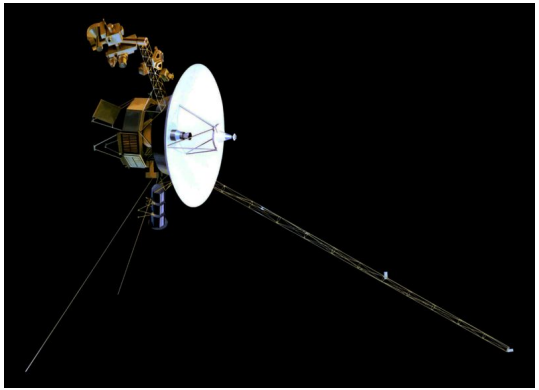
- Voyager 2 was launched in 1977.

- In 2010, the memory chips on the Voyager 2 probe got corrupted.
- A team of engineers had to send the “reboot” command to the probe.

- The probe is so far, it has a Round-Trip Time of 26 hours!
- The probe reached interstellar space in December 2018 and it’s still transmitting data.

Earlier this month, engineers suspended Voyager 2's science measurements because of an unexpected problem in its communications stream. A glitch in the flight data system, which formats information for radioing to Earth, was believed to be the problem. Engineers were able to replicate the glitch in a computer lab, showing that a single bit flip was responsible. NASA plans to reset Voyager's memory tomorrow.

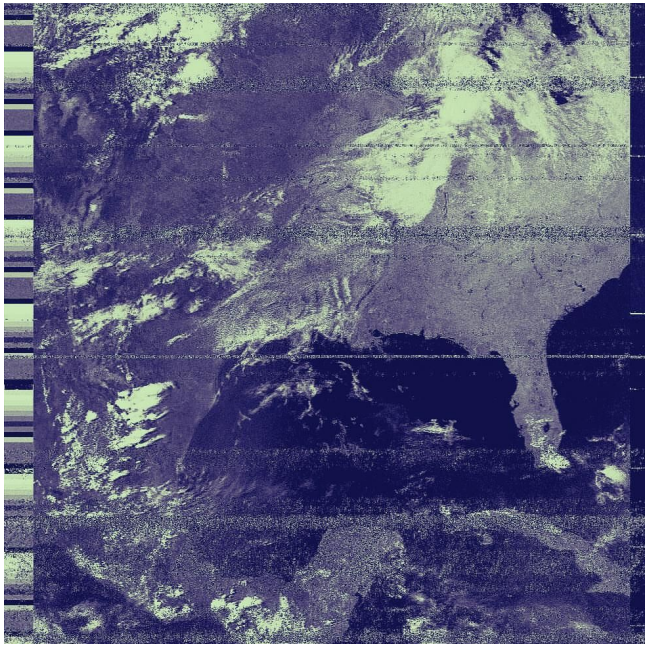
The spacecraft is so far away it takes nearly 13 hours for a radio signal from Earth, traveling at the speed of light, to reach it, and another 13 hours to receive a response.



https://www.zdnet.com/article/nasa-to-reset-voyager-2-memory/

Satellite communications - weather

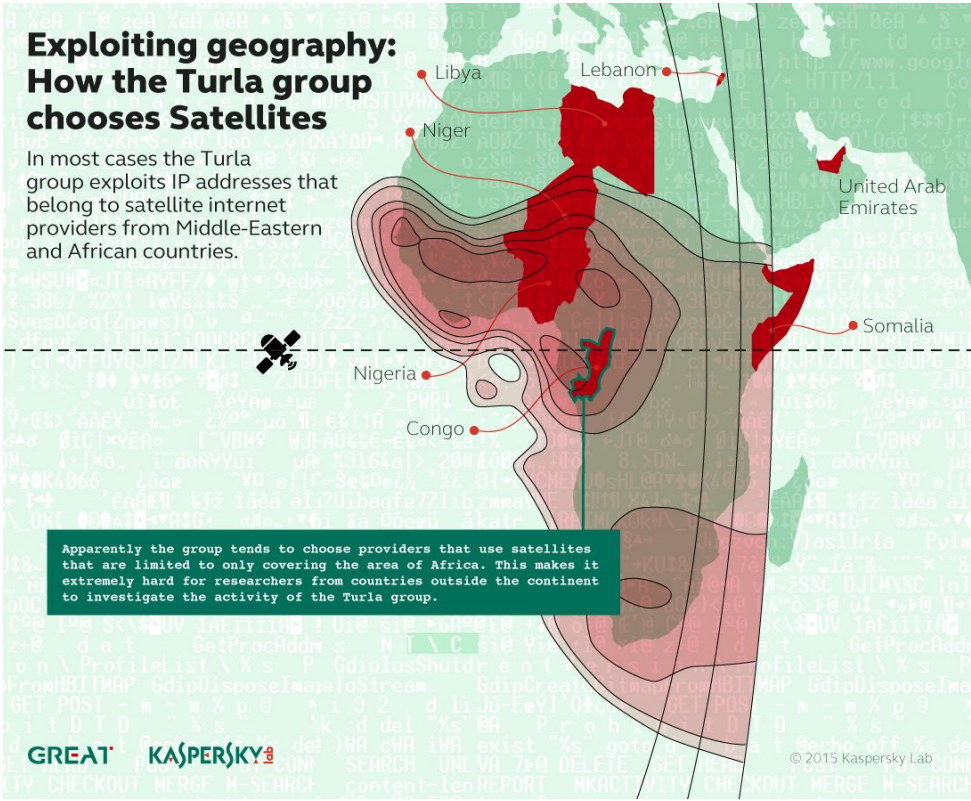
- Weather satellites broadcast the data continuously.
- Because of the interference (and because it’s not critical information) it doesn’t require encrypted transmission.
- It’s orders of magnitude simpler to recover the data this way.



- Because it’s unencrypted, it can be sniffed using a $10 dongle and a Raspberry Pi.

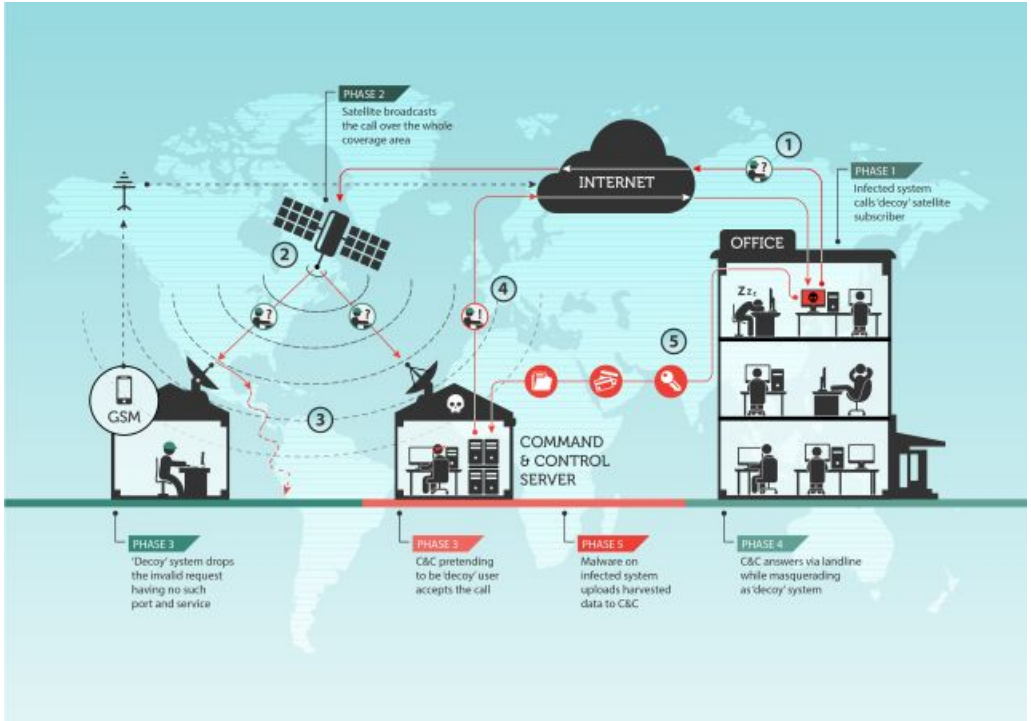
Satellite communications - Turla

https://securelist.com/satellite-turla-apt-command-and-control-in-the-sky/72081/



Satellite communications - Turla

https://securelist.com/satellite-turla-apt-command-and-control-in-the-sky/72081/



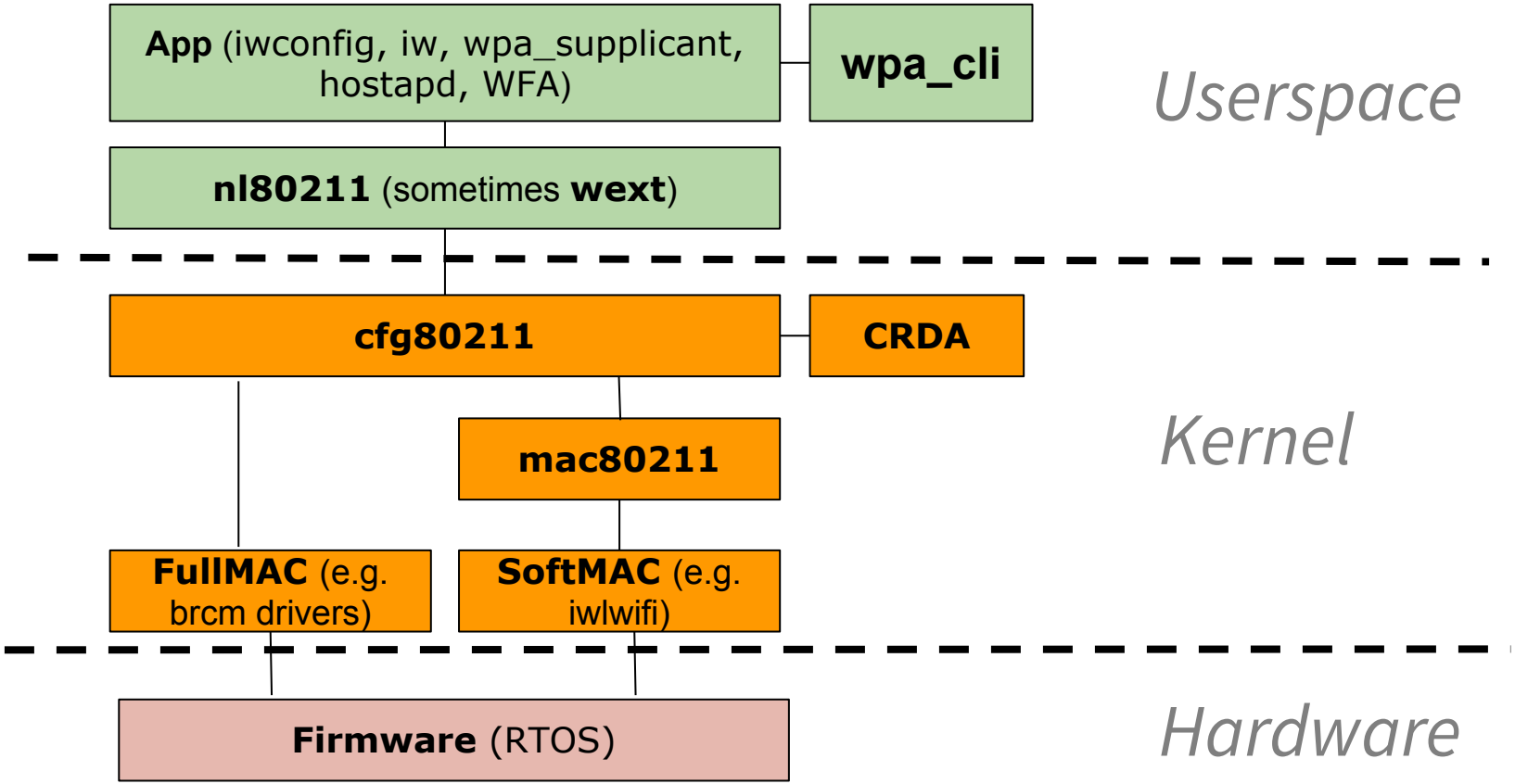
Wifi in your system

And some security …

Components of a Wifi stack

- Userspace configuration
- Driver
- Hardware (firmware, ASIC)
- And … Protocol design/Standard

Linux flow (because is all I know)



Mac80211 register with netdev

```
static const struct net_device_ops ieee80211_dataif_ops = {
    .ndo_open               = ieee80211_open,
    .ndo_stop               = ieee80211_stop,
    .ndo_uninit             = ieee80211_uninit,
    .ndo_start_xmit         = ieee80211_subif_start_xmit,
    .ndo_set_rx_mode        = ieee80211_set_multicast_list,
    .ndo_change_mtu         = ieee80211_change_mtu,
    .ndo_set_mac_address    = ieee80211_change_mac,
    .ndo_select_queue       = ieee80211_netdev_select_queue
};
```

Wifi driver interfacing with mac80211:

```
static struct ieee80211_ops il3945_mac_ops __read_mostly = {
        .tx                     = il3945_mac_tx,
        .start                  = il3945_mac_start,
        .stop                   = il3945_mac_stop,
        .add_interface          = il_mac_add_interface,
        .remove_interface       = il_mac_remove_interface,
        .change_interface       = il_mac_change_interface,
        .config                 = il_mac_config,
        .configure_filter       = il3945_configure_filter,
        .set_key                = il3945_mac_set_key,
        .conf_tx                = il_mac_conf_tx,
        .reset_tsf              = il_mac_reset_tsf,
        .bss_info_changed       = il_mac_bss_info_changed,
        .hw_scan                = il_mac_hw_scan,
        .sta_add                = il3945_mac_sta_add,
        .sta_remove             = il_mac_sta_remove,
        .tx_last_beacon         = il_mac_tx_last_beacon,
        .flush                  = il_mac_flush,
};
```

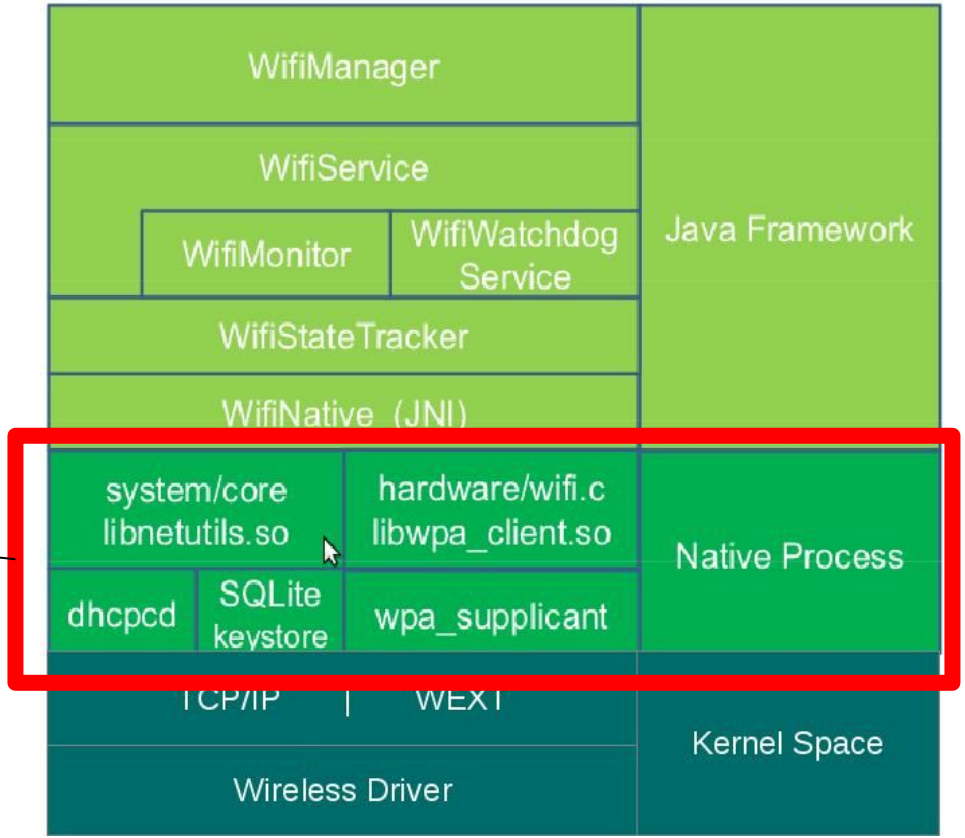
https://github.com/balena-os/linux-artik7/blob/master/drivers/net/wireless/iwlegacy/3945-mac.c#L3472

Just as a curiosity, I don't know this …

Source:

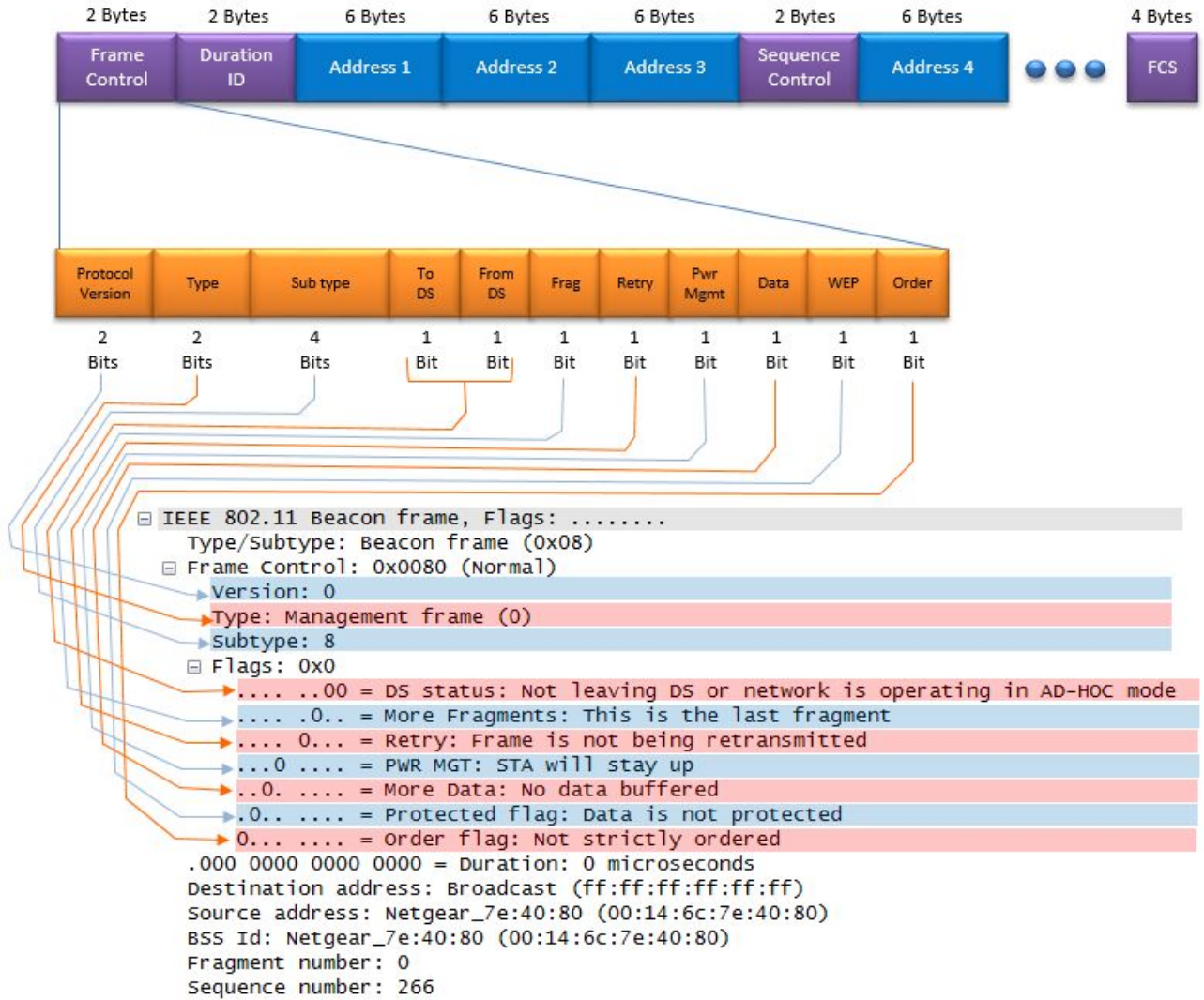
https://mitulmodi.wordpress.com/2012/03/21/android-wifi-architecture-wext/

The so-called HAL



mac80211 - wrapper for all hardware - functions

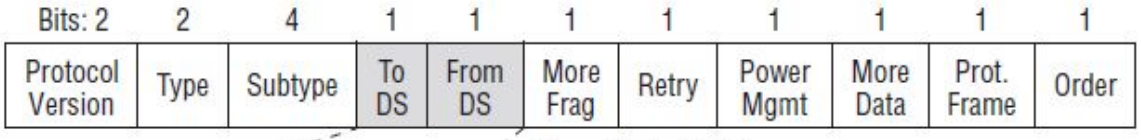
- Find interface
- Remove Ethernet header and add LLC & dot11 header
- Fill (SA DA RA ToDS/FromDS SEQ_CTRL)
- Fill (CONTROL DURATION)
- Rate adaption (based on RX-VECTOR feedback from hardware) → set rate (legacy, HT, VHT, MCS, BW)
- Flag frame for encryption
  - *“Software”* using CPU (e.g. ARM Crypto extensions)
  - Offload to firmware



- **What the … are those acronyms?**

Clarification of MAC addresses

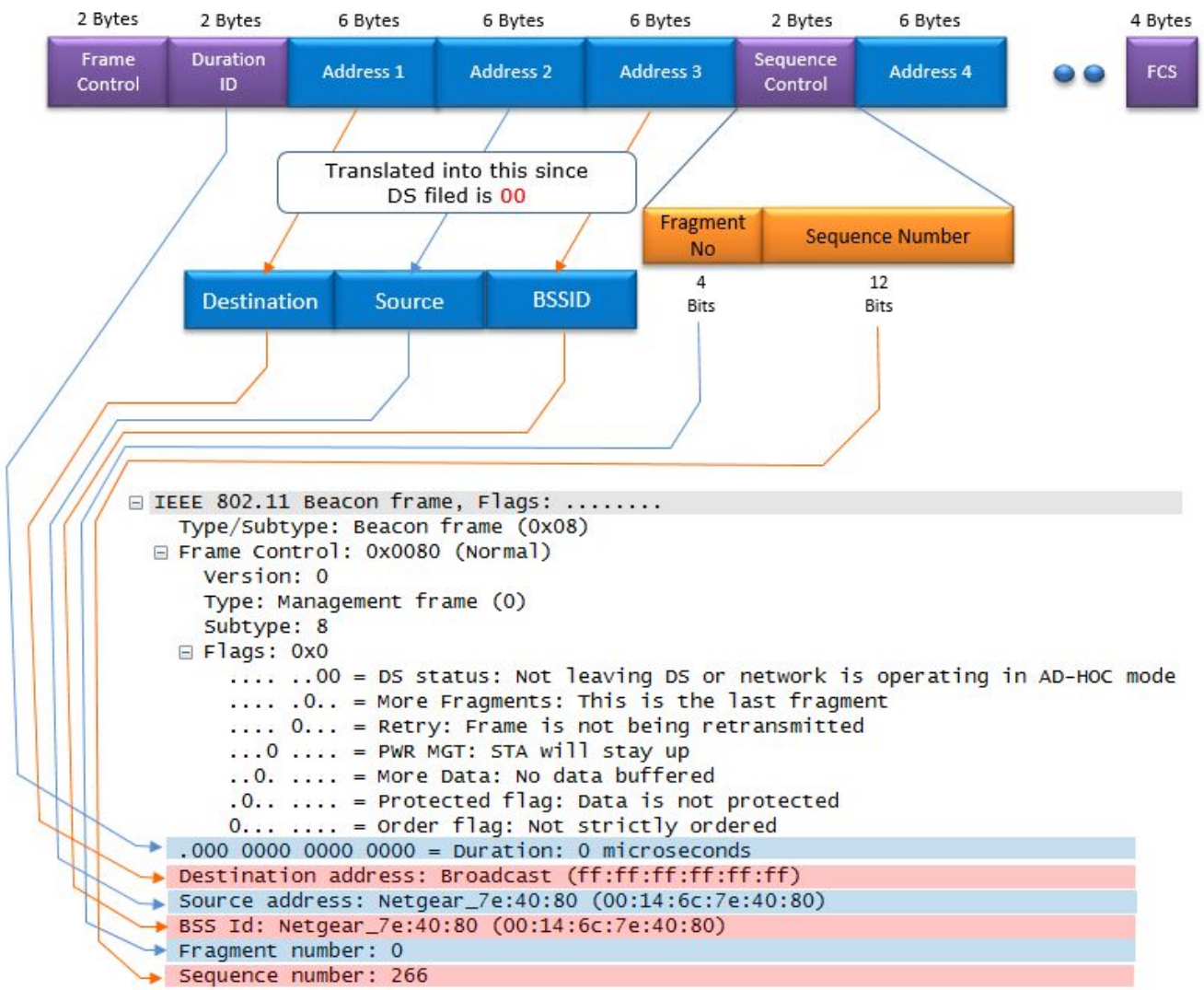
FIGURE 3.20 802.11 MAC addressing



To DS	From DS	Address 1	Address 2	Address 3	Address 4
0	0	RA = DA	TA = SA	BSSID	N/A
0	1	RA = DA	TA = BSSID	SA	N/A
1	0	RA = BSSID	TA = SA	DA	N/A
1	1	RA	TA	DA	SA

- SA = MAC address of the original sender (wired or wireless)
- DA = MAC address of the final destination (wired or wireless)
- TA = MAC address of the transmitting 802.11 radio
- RA = MAC address of the receiving 802.11 radio
- BSSID = L2 identifier of the basic service set (BSS)

- 802.11 frames typically use only 3 of the MAC address fields
- Frames send within WDS requires all 4 MAC address fields.



- **What the … are those acronyms? (cont.)**

Ok...And the firmware?

- Setup hardware registers
- DMA frame to/from Linux queues
- Keep statistics
- Reports to mac80211 RX info (e.g. RSSI) for rate adaption & carrier sense
- Implement DCF or EDCA
- Take care of bureaucracy such as L2 simple ACK/BlockACK, TSF, DTIM

Oh...interesting...hardware...I don't care

- GoogleProjectZero project: BCM4339 exploit
- ROM - used to store firmware code
- RAM - data processing (heap, WiFi structures etc., mac80211 data, whatever) - firmware is **downloaded by the driver!!**
- Wi-Fi management frames encode most of their information in Information Elements (IEs)- structured as TLVs:
  - Cisco CCKM or 802.11r FT is trigger
  - Information embedded in management frames was vulnerable, could trigger a stack overflow → **CVE-2017-6957**

Oh … I want to read

https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_4.html

https://googleprojectzero.blogspot.com/2017/04/over-air-exploiting-broadcoms-wi-fi_11.html

https://bugs.chromium.org/p/project-zero/issues/detail?id=1051

Linux (and Android) was easy because

- You can get **ioctl** access - http://androidxref.com/7.1.1_r6/xref/system/sepolicy/ioctl_macros
- BCM provides free access to debug tools: https://android.googlesource.com/platform/hardware/broadcom/wlan/+/master/bcmdhd/dhdutil/Android.mk

Same research was carried on iOS

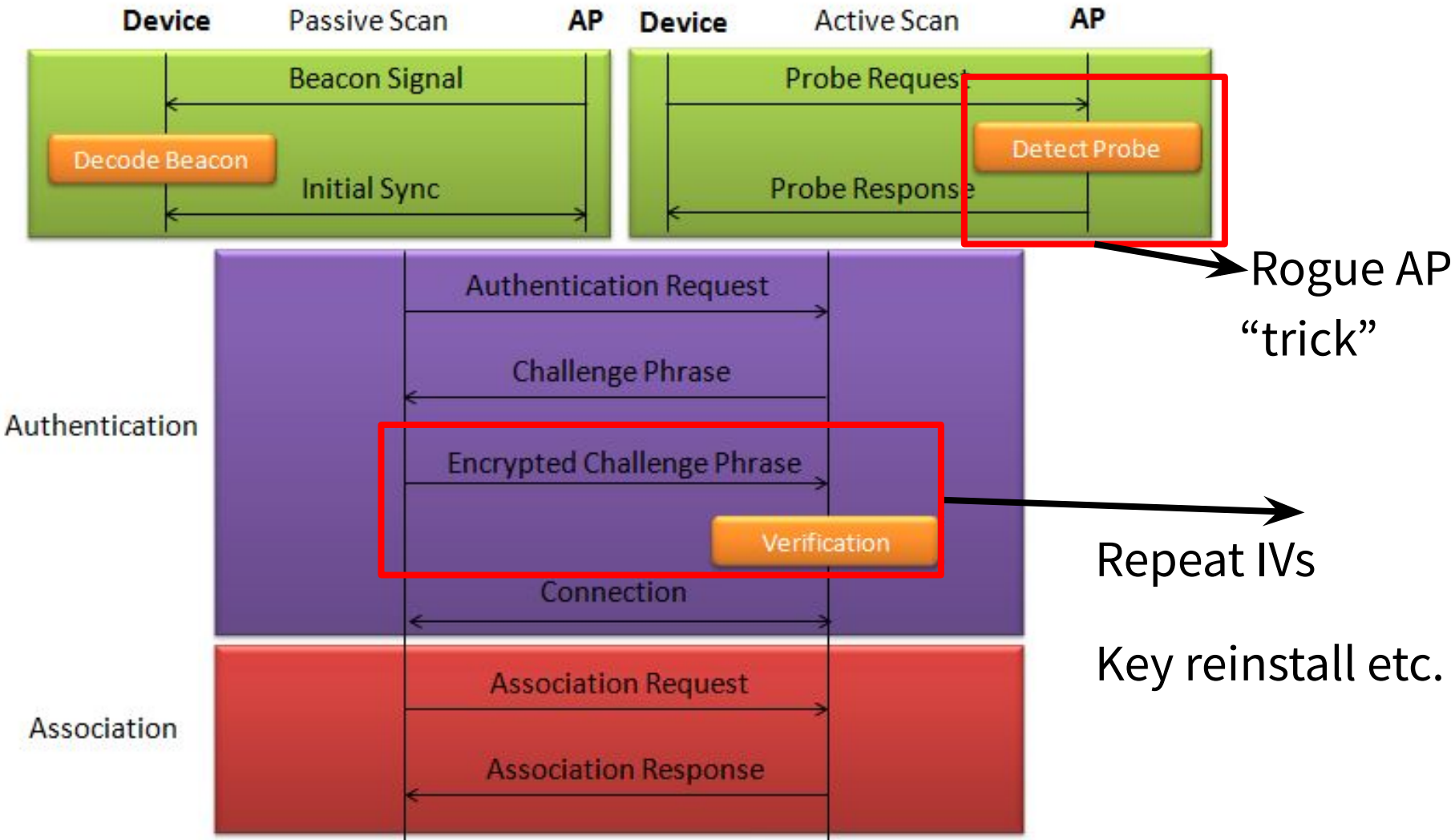
https://googleprojectzero.blogspot.com/2017/09/over-air-vol-2-pt-1-exploiting-wi-fi.html

https://googleprojectzero.blogspot.com/2017/10/over-air-vol-2-pt-2-exploiting-wi-fi.html

https://googleprojectzero.blogspot.com/2017/10/over-air-vol-2-pt-3-exploiting-wi-fi.html

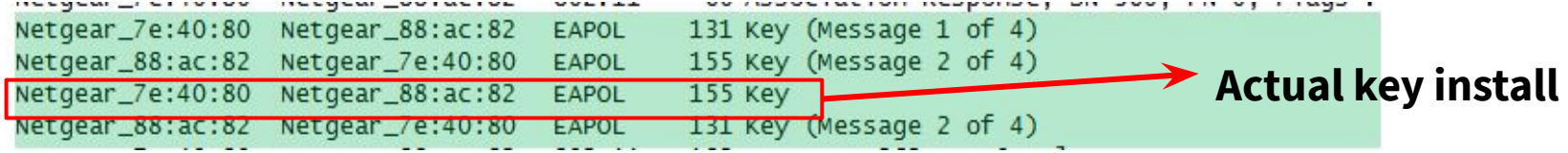
Wifi is worse: Known protocols for security

- **WEP** - *sept 1999* - should be abandoned :)
  - Aircrak-ng tutorial using ARP + IVs
- **WPA** (or WPA-PSK) - 802.11i - around 2003 - as w/a to WEP
  - TKIP for encryption, PSK for getting keys
- **WPA2** - 802.16i, 2004 first rel, 2006 - mainstream
  - AES (because US Gov), PSK for key sharing
  - KRACK & other board people
- **WPA3** - Jan 2018 - fresh!!
  - Should replace “safely” key generation


Device
Passive Scan
AP
Beacon Signal
Decode Beacon
Initial Sync
Device
Active Scan
AP
Probe Request
Detect Probe
Probe Response
Rogue AP
"trick"
Authentication
Authentication Request
Challenge Phrase
Encrypted Challenge Phrase
Verification
Connection
Repeat IVs
Key reinstall etc.
Association
Association Request
Association Response

Known WPA2 vulnerabilities - KRACK

- Exploits below 4-way handshake when associating



- Trick the vulnerable client to reinstall key already in use
- Force reset of packet numbers - go to a rogue AP
- Wifi has loses → retransmission is carried by AP
- Entire research at https://www.krackattacks.com/
- Catastrophic because this is design protocol issue, not hardware, nor crappy vendor

WPA2 - hobby cracks

- Prof Bill Bunachan implements KRACK using a RPI and 10$ wifi transceiver to crack a PBKDF2-SHA1, without need to capture entire association process
- Random guy on forum reads about WPA3 founds a way to crack WPA2

Source1: https://medium.com/@billatnapier/the-beginning-of-the-end-of-wpa-2-cracking-wpa-2-just-got-a-whole-lot-easier-55d7775a7a5a

Source 2: https://hashcat.net/forum/thread-7717.html

WPA3

- Replace PSK for getting key with Simultaneous Authentication of Equals (SAE) and 128-bit encryption in personal
- For enterprise: 256-bit GCMP
- Key derivation and confirmation: 384-bit-HMAC with SHA384
- Key establishment and authentication: ECDH + ECDSA using a 384-bit elliptic curve
- Robust management frame protection: BIP-GMAC-256

WPA3 - Dragonfly attack - not a joke

Dragonblood: A Security Analysis of WPA3's SAE Handshake

Mathy Vanhoef
New York University Abu Dhabi
Mathy.Vanhoef@nyu.edu

Eyal Ronen
Tel Aviv University and KU Leuven
eyal.ronen@cs.tau.ac.il

ABSTRACT

The WPA3 certification aims to secure Wi-Fi networks, and provides several advantages over its predecessor WPA2, such as protection against offline dictionary attacks and forward secrecy. Unfortunately, we show that WPA3 is affected by several design flaws, and analyze these flaws both theoretically and practically. Most prominently, we show that WPA3's Simultaneous Authentication of Equals (SAE) handshake, commonly known as Dragonfly, is affected by password partitioning attacks. These attacks resemble dictionary attacks and allow an adversary to recover the password by abusing timing or cache-based side-channel leaks. Our side-channel attacks target the protocol's password encoding method. For instance, our cache-based attack exploits SAE's hash-to-curve algorithm. The resulting attacks are efficient and low cost: brute-forcing all 8-character lowercase password requires less than 125$ in Amazon EC2 instances. In light of ongoing standardization efforts on hash-to-curve, Password-Authenticated Key Exchanges (PAKEs), and Dragonfly as a TLS handshake, our findings are also of more general interest. Finally, we discuss how to mitigate our attacks in a backwards-compatible manner, and explain how minor changes to the protocol could have prevented most of our attacks.

design and implementation flaws. For instance, when verifying the assumptions made by the formal proof of the SAE handshake [59], we discovered both timing and cache-based side-channel vulnerabilities in its password encoding method. We empirically confirmed all our findings against both open source and recently-released proprietary implementations of WPA3.

All combined, our work resulted in the following contributions:

- We provide a self-contained and high-level description of WPA3 and its SAE handshake (Section 2 and 3).
- We show that the anti-clogging mechanisms of SAE is unable to prevent denial-of-service attacks (Section 4). In particular, by abusing the overhead of SAE's defenses against already-known side-channels, a resource-constrained device can overload the CPU of a professional Access Point (AP).
- We present a dictionary attack against WPA3 when it is operating in transition mode (Section 5). This is accomplished by trying to downgrade clients to WPA2. Although WPA2's 4-way handshake detects the downgrade and aborts, the frames sent during the partial 4-way handshake provide enough information for a dictionary attack. We also present a downgrade attack against SAE, and discuss implementation-

https://papers.mathyvanhoef.com/dragonblood.pdf

WPA3 - Dragonfly attack

- Dragonforce - experimental tool that takes the information to recover from the timing attacks and performs a password partitioning attack.
- Dragonslayer - implements attacks against EAP-P (Extensible Authentication Protocol-Password)

"Nearly all of our attacks are against SAE's password encoding method, i.e., against its hash-to-group and hash-to-curve algorithm. Interestingly, a simple change to this algorithm would have prevented most of our attacks," the researchers say

WPA3 weakness - references

https://thehackernews.com/2019/04/wpa3-hack-wifi-password.html?m=1

https://medium.com/asecuritysite-when-bob-met-alice/wpa-3-dragonfly-out-of-the-frying-pan-and-into-the-fire-35240aef4376

Now you care? (instead of conclusions)

- Yes, I should care about who makes the hardware and how
- Yes, I should care who develops protocols
- Yes, I should read some papers
- Yes, Wi-Fi is worse than loses: MAC headers are always open
- Yes, capturing wifi from my neighbours can be fun

Questions

- Which component from WiFi stack is vulnerable to attacks?
- When is it not necessary to encrypt data messages?