Analysis of Protected Management Frames and WPA3's SAE-PK

Mathy Vanhoef - @vanhoefm

Workshop on Attacks in Cryptography (WAC), co-located with CRYPTO

14 August 2022, Santa Barbara, US (presented online)

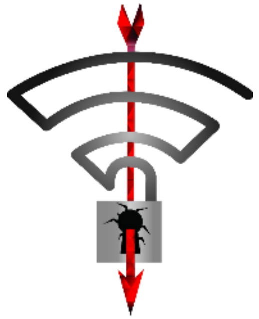
Agenda

Early 2000 Wi-Fi Protected Access (WPA and WPA2)

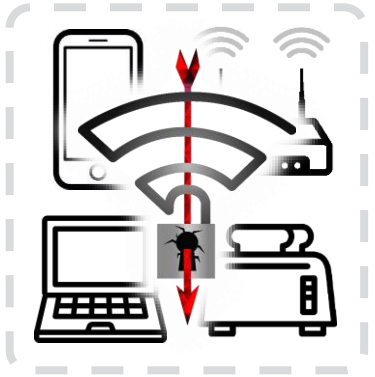
Affected by various design flaws:

- Vulnerable to offline **dictionary attacks**: a captured handshake can be used to brute-force the password
- **No forward secrecy**: can decrypt previously captured traffic after learning the password
- **Unprotected management frames**: can spoof beacons, deauthentication frames, & other non-data frames

2017 Key reinstallation attacks (KRACK)



- Flaw in the standard → all devices affected
- Motivated standard bodies to improve Wi-Fi security



Practical impact:

- Decrypt frames sent by a vulnerable device
- Replay frames towars a vulnerable device

2018 Wi-Fi Protected Access 3 (WPA3)

1. Simultaneous Authentication of Equals (**SAE**) handshake
   - Also called the **Dragonfly** handshake

Provides mutual authentication

Negotiates session key

Forward secrecy & prevents offline dictionary attacks

Protects against server compromise

2018 Wi-Fi Protected Access 3 (WPA3)

1. Simultaneous Authentication of Equals (**SAE**) handshake
   - Also called the **Dragonfly** handshake
2. Usage of Management Frame Protection (**MFP**)
   - Protection of deauthentication and disassociation frames to **prevent denial-of-service attacks**
   - Protection of “robust” action frames (frames used to manage the connection)

2019 **Dragonblood** attacks against WPA3

Main flaw is a **side-channel leak** in the Dragonfly handshake

- Time it takes for the AP to respond to (partial) handshake leaks info about the password
- Leaked info enables **offline brute-force attack**
- Attack against Raspberry Pi **feasible in practice**!

Agenda

1. Introduction
2. **Management Frame Protection**
    - ➢ Based on collaboration with Domien Schepers and Aanjhan Ranganathan, Northeastern University (WiSec'22)
3. The SAE-PK Protocol

WPA3 and Management Frame Protection (MFP)

WPA3 requires usage of management frame protection

Background: in Wi-Fi there are three types of frames:

1. **Management** frames: scanning for networks, disconnecting, advanced sleep mode, etc.
2. **Control** frames: acknowledgements, request to send, etc.
3. **Data** frames: transporting higher-layer data

WPA2 only required protection of data frames…

Management Frame Protection (MFP)

Provides confidentiality, integrity, and replay protection for:

- **Deauthentication** frames
- **Disassociation** frames
- **Robust action** frames

Prevent Denial-of-Service attacks

Management frame protection has exisisted since 2009 (!!)

- Defined in 802.11w amendment and **optional under WPA2**
- Not used in practice because few devices supported it…
- …and because (client) implementations were often buggy

Security of Management Frame Protection (MFP)

1. How secure is the MFP standard?
2. How secure are implementations?

- Security of the standard:
  - **Manual specification analysis**: are all frames protected? Are the rules complete, consistent, and secure?
- Security of implementations:
  - **Code inspection and tests**: looking for denial-of-service attacks

Analysis of standard (part one)

Several management frames are still left unprotected:

- **ATIM frames**: power management in ad hoc networks
- **Timing Advertisement frames**: used in vehicular networks
- **Beacons**: announces a network and its properties, used by all Wi-Fi networks
  - There is a recent beacon protection defense [VAP20]
  - But this defense is optional for WPA3 networks

Abusing beacons [VAP20]

Beacons contain the maximum allowed transmit power

- Abuse to **lower transmission power of victims**

Cisco extension to limit transmission power of clients

- Linux: can be abused to **forcibly disconnect a victim** by setting a negative transmission limit

They contain transmission back-off parameters (CSMA/CA)

- Abuse to **lower the bandwith of clients**

Other beacon attacks [VAP20]

Other attacks: partial **machine-in-the-middle**

- Bypasses channel operating validation in Linux

Battery depletion attacks

- Spoof beacons to **make clients stay awake**

→ Solution: use beacon protection!

Source: M. Vanhoef, P. Adhikari, and C. Pöpper. Protecting Wi-Fi Beacons from Outsider Forgeries. In *13th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2020.

Analysis of standard: part two [SRV22]

Part two: **inspecting the MFP rules**

- There are ~10 import rules, each described in a paragraph:

"A STA with dot11RSNAProtectedManagementFramesActivated equal to false shall transmit and receive unprotected individually addressed robust Management frames to and from any associated STA and shall discard protected individually addressed robust Management frames received from any associated STA.

A STA with dot11RSNAProtectedManagementFramesActivated equal to true and dot11RSNAUnprotectedManagementFramesAllowed equal to true shall transmit and receive unprotected individually addressed robust Management frames to and from any associated STA that advertised MFPC = 0 and shall discard protected individually addressed robust Management frames received from any associated STA that advertised MFPC = 0.

…

Management frame protection cannot be applied until the PTK and IGTK has been established with the STA. A STA shall not transmit robust Action frames until it has installed the PTK for the peer STA, or in the case of group addressed frames, has installed the IGTK. The STA shall discard any robust Action frames received before the PTK and IGTK are installed."

First issue: complex conditionals in the rules

"A STA with **dot11RSNAProtectedManagementFramesActivated equal to true** and **dot11RSNAUnprotectedManagementFramesAllowed equal to true** shall transmit and receive unprotected individually addressed robust Management frames to and from any associated **STA that advertised MFPC = 0** and shall discard protected individually addressed robust Management frames received from any associated STA **that advertised MFPC = 0**."

- Whether a rule applies is conditional on:
  - Whether the client and access point **are capable** of MFP
  - Whether the client and access point **require** MFP
  - Whether the client **advertised support** of MFP
  - Whether the AP **advertised support** of MFP
  - Whether the AP and client possess the necessary **keys**

Analysis of standard: part two [SRV22]

We discovered that this complexity leads to:

- **Undefined scenarios** as well as **contradictory rules**
- Shortcomings lead to new denial-of-service vulnerabilities

- Example contradictory rule: handling unprotected Deauth frames once connected
  - Some rules say to drop them, some say to start a protected SA Query
- Example undefined scenario: how a client should handle broadcast frames when the AP doesn't support MFP

First issue: the standard is hard to understand

- We also discovered insecure rules:

"The receiver **shall process unprotected** individually addressed Disassociation and Deauthentication frames before the PTK and IGTK are installed."

- Adversary can **cause the handshake to fail** by spoofing Disassociation or Deauthentication frames
- Suggested defense:
  - **Start a timer instead of disconnecting**
  - Stop the timer when the handshake progresses. This effectively ignores the unprotected Deauth or Disassoc frame.

Addressing issues in the standard

Long-term fix: require MFP so the rules become simpler

- Avoids all the conditional statements in the rules

Short-term fix: we proposed updates to simplify the rules in the standard & offer defenses:

- Presented at TGm meetings of the IEEE 802.11
- Currently still being discussed by the IEEE

MFP security part two: implementation analysis

We also **audited implementations of MFP**

- Goal was to find **disconnection attacks**
- Causes the victim to reconnect & perform a new handshake
- This facilitates attacks that target the connection process (KRACK, FragAttacks, Dragonblood, etc.)

MFP security part two: implementation analysis

Methodology:

1. First focus on open-source implementations (e.g., Linux)
2. Inspect code paths that lead to disconnection calls
3. Can these be triggered by spoofing plaintext frames?
4. Test whether discovered attacks also work against other implementations

Testing framework: github.com/domienschepers/wifi-framework

Implementation analysis: disconnection attacks

Found several implementation vulnerabilities:

- Spoofing **4-way handshake** messages: are accepted on Linux even if sent in plaintext. Causes client to disconnect.
- Spoofing **802.1X messages**: accepted on Linux even if sent in plaintext. Various messages will disconnect the client.
- Spoofing **beacon frames**: specifying an invalid bandwidth or channel witch announcement causes the client to disconnect (Linux, macOS, iOS, iPadOS, and Windows).

Implementation analysis: other attacks

- **Packet counter for group key** is not set when connecting to a network, allows for replay of management frames
- **Flooding** a lot of SAE (Dragonfly) handshake messages **crashes** the D-Link DIR-X1860 router
  - Out-of-memory flaw in the driver
  - Other routers with the same driver likely also vulnerable

Agenda

Introduction to SAE-PK

Goal of SAE Public Key:

- Authenticate a Wi-Fi hotspot using a password…
- …but prevent an adversary from cloning the network

→ Accomplished by using asymmetric crypto

High-level overview of SAE-PK

Based on public key crypto:

1. The Access Point (AP) generates a public/private key pair
2. The Wi-Fi password is derived from the public key
3. The public key is sent to the client when connecting
4. Clients use the password to verify the public key of the AP
5. AP proves possession of the corresponding private key

→ The **password forms a signature** of the public key

The SAE-PK password

The SAE-PK password is the truncated output of:

Hash(SSID || Modifier M || public key)

- SSID (Service Set Identifier): name of the Wi-Fi network
- Modifier M: starts from a random value and is incremented until the output starts with 3 or 5 zero bytes.
  - Number of required zero bytes is controlled by a security parameter
- Public key: point on an elliptic curve

The SAE-PK password

The SAE-PK password is the truncated output of:

Hash(SSID || Modifier M || public key)

Output is converted into a human-readable form

- Example password: **`2udb-slxf-3ijn`**`-dbu3`
- Password length is variable and decided by administrator
- Shortest allowed password length encodes 52 bits of the hash output (excluding the leading 3 or 5 zero bytes)

Attack: creating a rogue clone of the network?

Find a modifier M & public key that result in the same password

Hash(SSID || Modifier M || public key)

What is the complexity of this in the best case?

- Hash output must start with at least 3 zero bytes → 2^{24}
- Remaining output must equal the password → 2^{52}

Total time complexity of 2^{76} to perform a naïve attack

Time-memory trade-off attack

An attacker is essentially inverting the hash function:

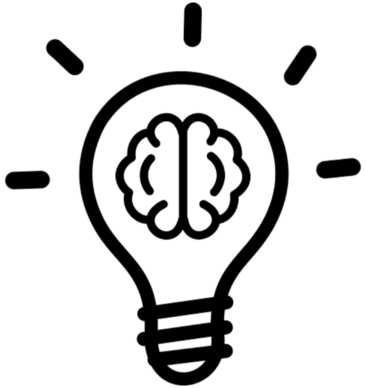
- We can construct **rainbow tables** to optimize the attack.
- The SSID is part of the hash input, so every table only will work **against a specific network name**.
- On verge of practicality based on theoretical estimates.
  - E.g., a table of ~6TB can break a password in ~2 weeks on AWS.
  - **More research is needed**, these are very rough estimates.

→ Mitigate attacks using a long password or by making the truncated output start with at least 5 zero bytes.

Downside of computed table

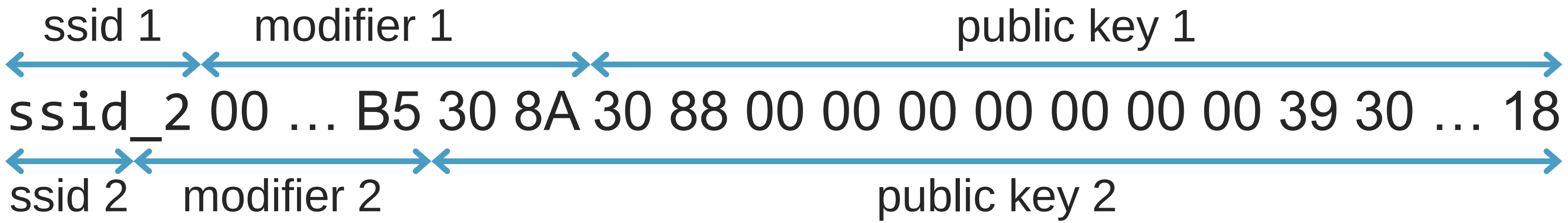
Major downside: computed table can only be used to attack a single SSID.

- Simplified, the table inverts Hash(SSID || M || Public key)
- We need to pick a specific SSID when building the table



Idea! Can we change the SSID without changing the input to the hash function?

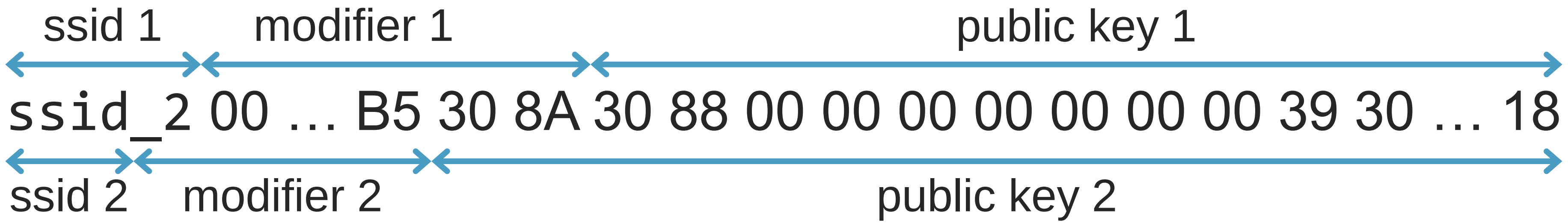
Different SSID but same hash input



- Network “`ssid_2`”: uses the modifier value “00 … B5 30 8A”
- Network “`ssid`”: uses the modifier value “_2 00 .. B5”
- Public key 1 starts in the middle of public key 2

Problem: how to embed public key 1 into public key 2?

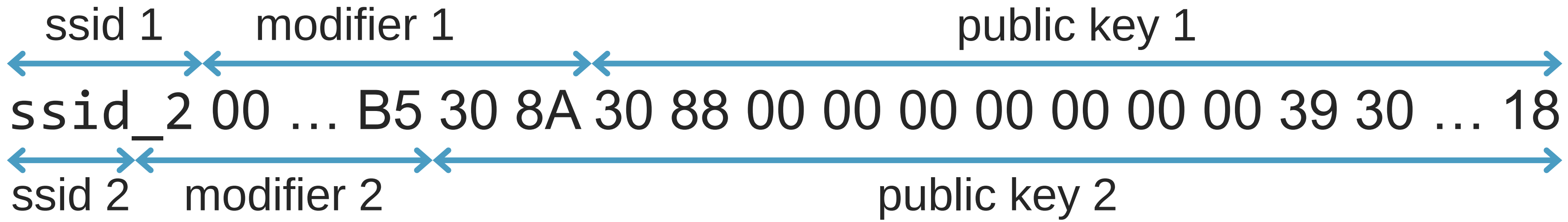
Different SSID but same hash input


ssid 1
modifier 1
public key 1
`ssid_2` 00 … B5 30 8A 30 88 00 00 00 00 00 00 00 39 30 … 18
ssid 2
modifier 2
public key 2

Public key is encoded according to RFC 5480. We **abuse parsing flexibilities** to embed one public key into another:

- CVE-2014-1569: encode arbitrary data within a length field
- CVE-2018-16151: accepts arbitrary parameters in the algorithm identifier

Different SSID but same hash input



```
ssid_2 00 … B5 30 8A 30 88 00 00 00 00 00 00 00 39 30 … 18
```

- A vulnerable client will accept public key 2 and public key 1
- A secure client accepts public key 1, but rejects public key 2 because it contains unexpected data

→ Against a vulnerable client a single precomputed table can target two SSIDs where one is the prefix of the other

Network-based attacks

ARP poisoning

- The SAE-PK standard allows client-to-client traffic
- An adversary can use ARP poisoning to intercept traffic

Abuse of the group key

- All clients have the symmetric key to protect broadcast traffic
- This means every client and spoof broadcast traffic

Conclusion

Security of **Management Frame Protection**:

- Standard is complex and difficult to understand
- Non-trivial to implement in practice
- DoS attacks still possible in practice

Security of **SAE Public Key**:

- Need to pick a long enough password…
- …or pick a strong enough security parameter

Thank you! Questions?

Security of **Management Frame Protection**:

- Standard is complex and difficult to understand
- Non-trivial to implement in practice
- DoS attacks still possible in practice

Security of **SAE Public Key**:

- Need to pick a long enough password…
- …or pick a strong enough security parameter