Recent Wi-Fi attacks & defenses:
general lessons learned & open problems

Mathy Vanhoef - @vanhoefm

Summer School on real-world crypto and privacy

17 June 2022, Šibenik, Croatia

Presentation Outline

Recent attacks:

- Key reinstallation attacks in WPA2 (= KRACK)
- Side-channel leaks in WPA3 (= Dragonblood)
- Fragmentation issues in WPA* (= FragAttacks)

New defenses:

- Opportunistic wireless encryption (Wi-Fi Alliance)
- Beacon protection & channel validation (our work ☺)
- SAE-PK to protect hotspots (Wi-Fi Alliance)

Presentation Outline

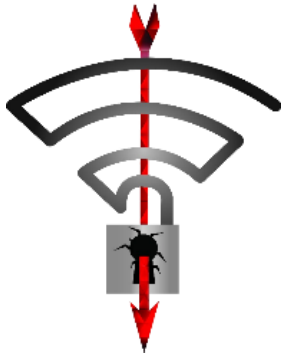
Recent attacks:

- **Key reinstallation attacks in WPA2 (= KRACK)**
- Side-channel leaks in WPA3 (= Dragonblood)
- Fragmentation issues in WPA* (= FragAttacks)

New defenses:

- Opportunistic wireless encryption (Wi-Fi Alliance)
- Beacon protection & channel validation (our work ☺)
- SAE-PK to protect hotspots (Wi-Fi Alliance)

Advancements in Wi-Fi security



2017 Key reinstallation attacks (**KRACK**)

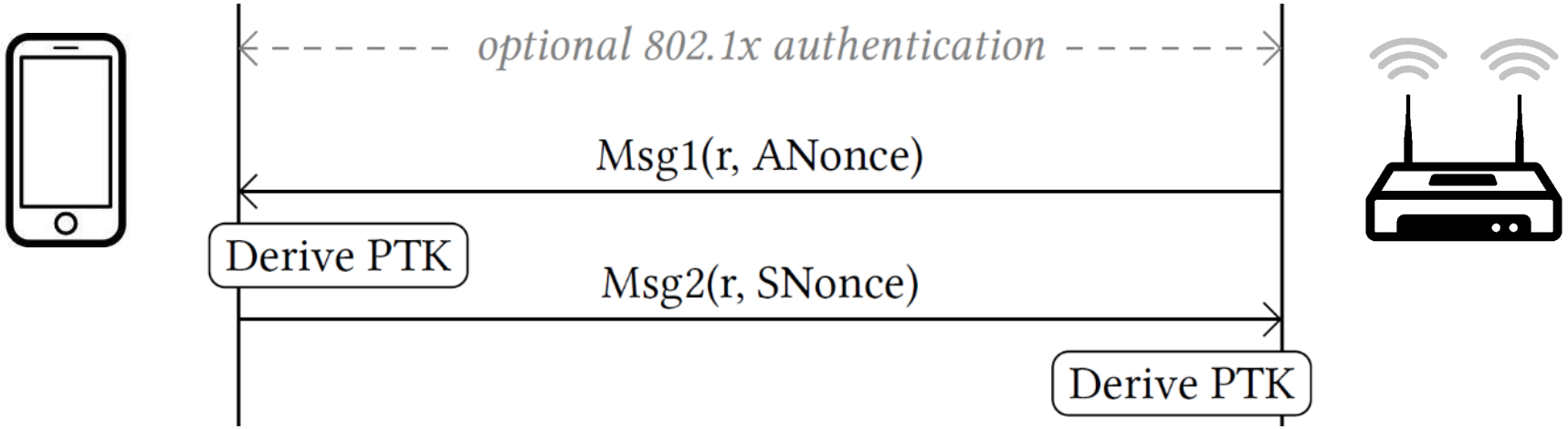
- Flaw in various Wi-Fi handshake → all devices affected
- We will focus on the 4-way handshake

4-way handshake is to connect to any protected Wi-Fi network

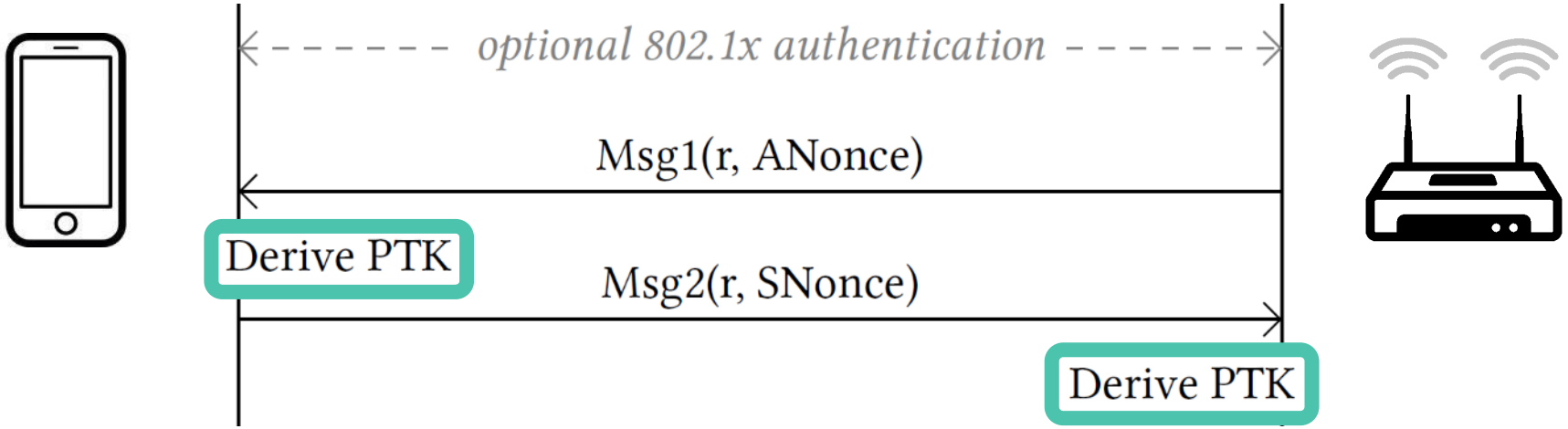
- Provides **mutual authentication**
- Negotiates fresh **P**airwise **T**ransient **K**ey (**PTK**) = session key

4-way handshake (simplified)

4-way handshake (simplified)

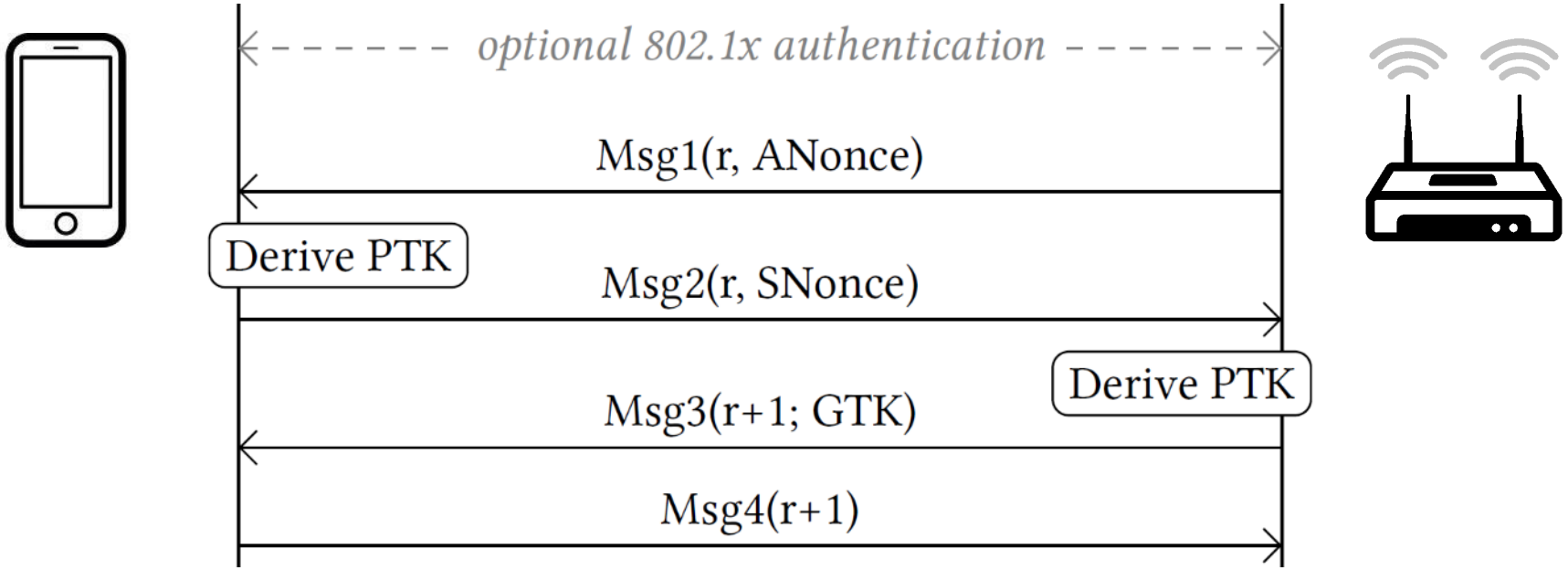


4-way handshake (simplified)

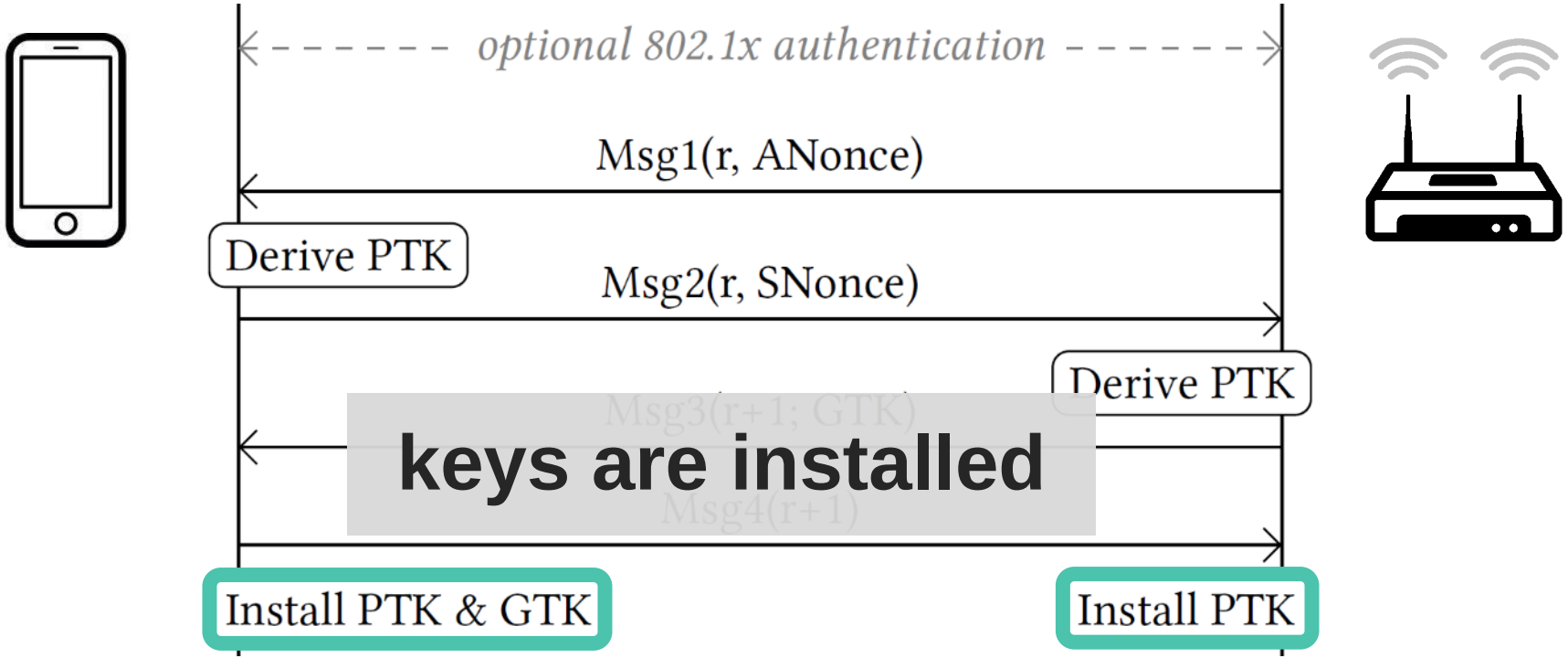


› Once we have a PTK (= session key) all messages are protected using a **Message Integrity Code (MIC) = MAC**

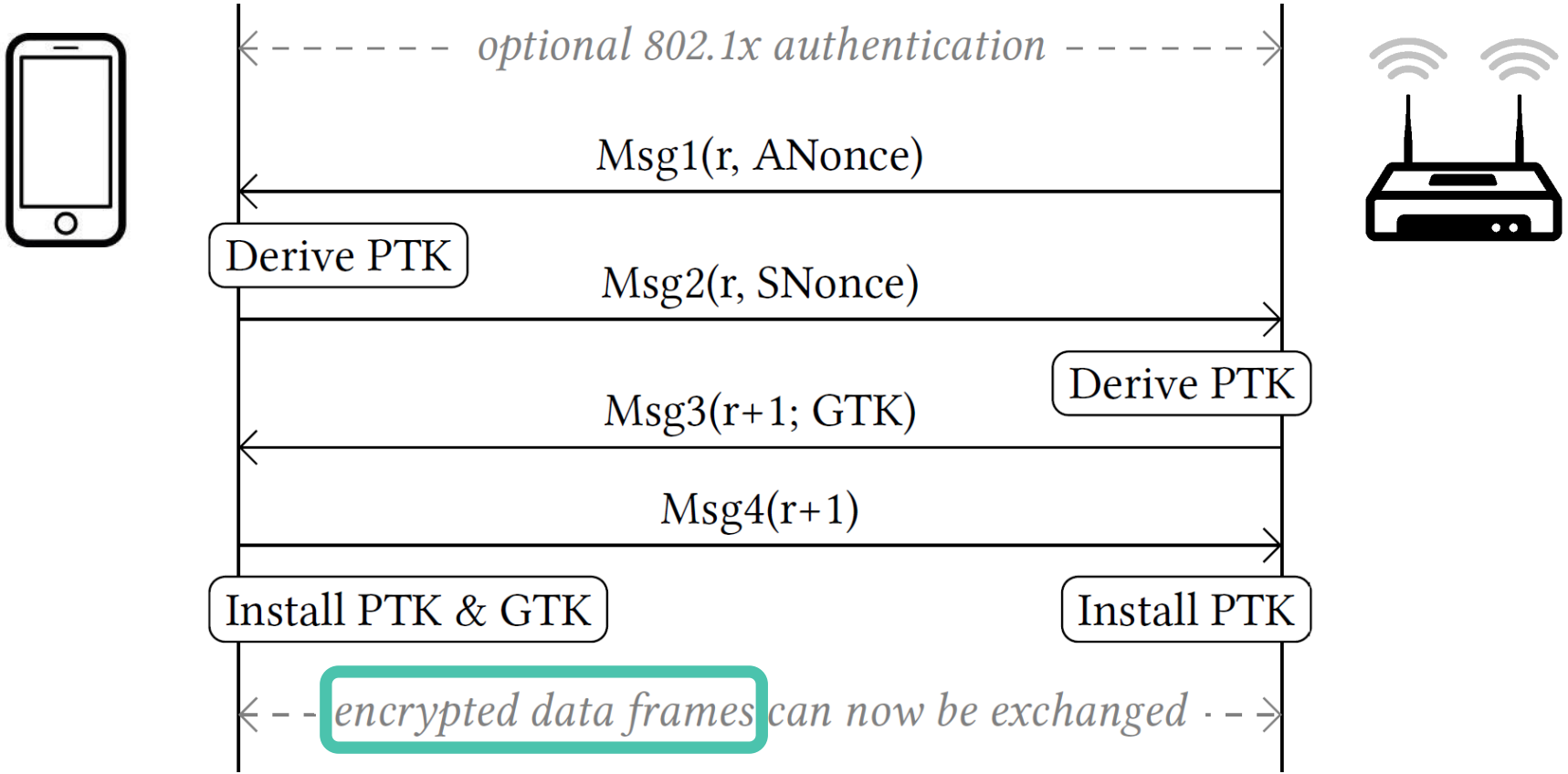
4-way handshake (simplified)



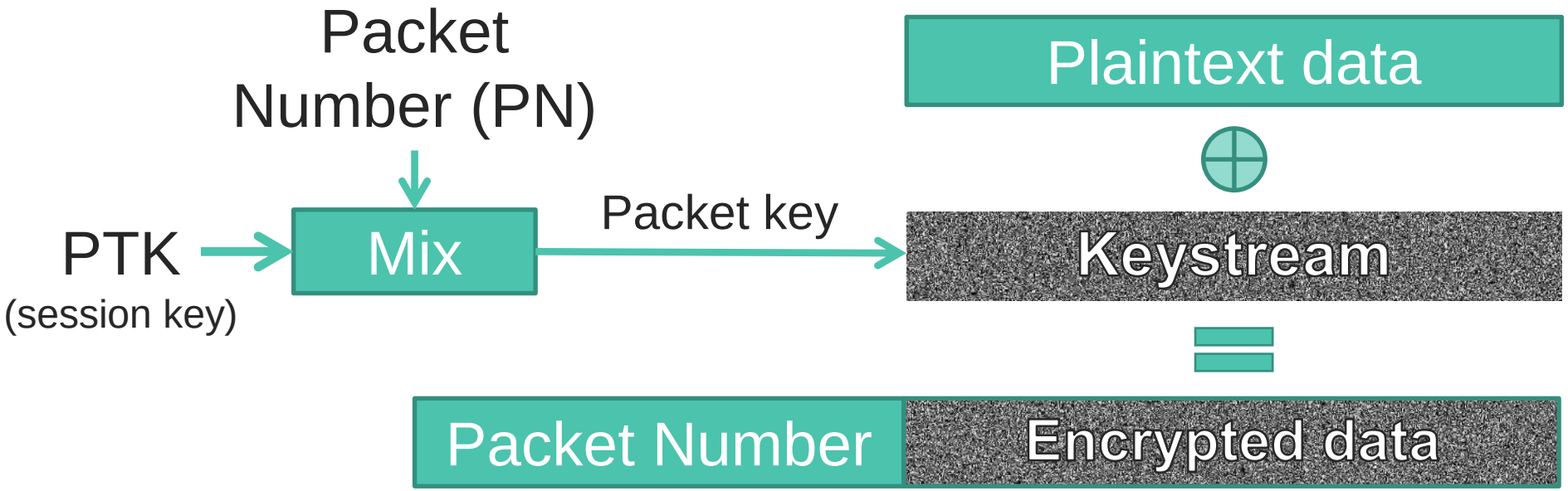
4-way handshake (simplified)



4-way handshake (simplified)

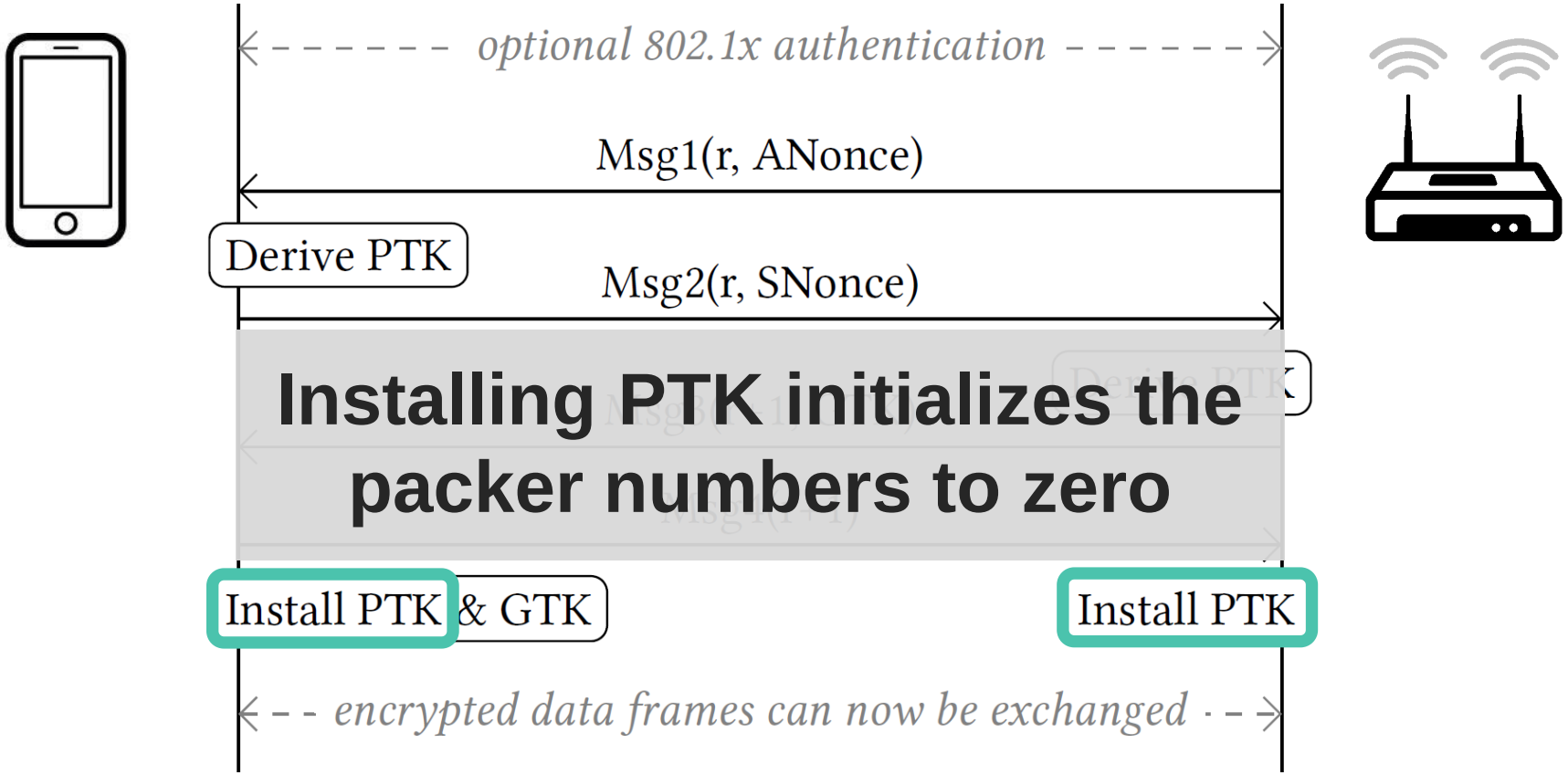


Frame encryption (simplified)



→ This high-level construction is used in **all WPA versions**

4-way handshake (simplified)

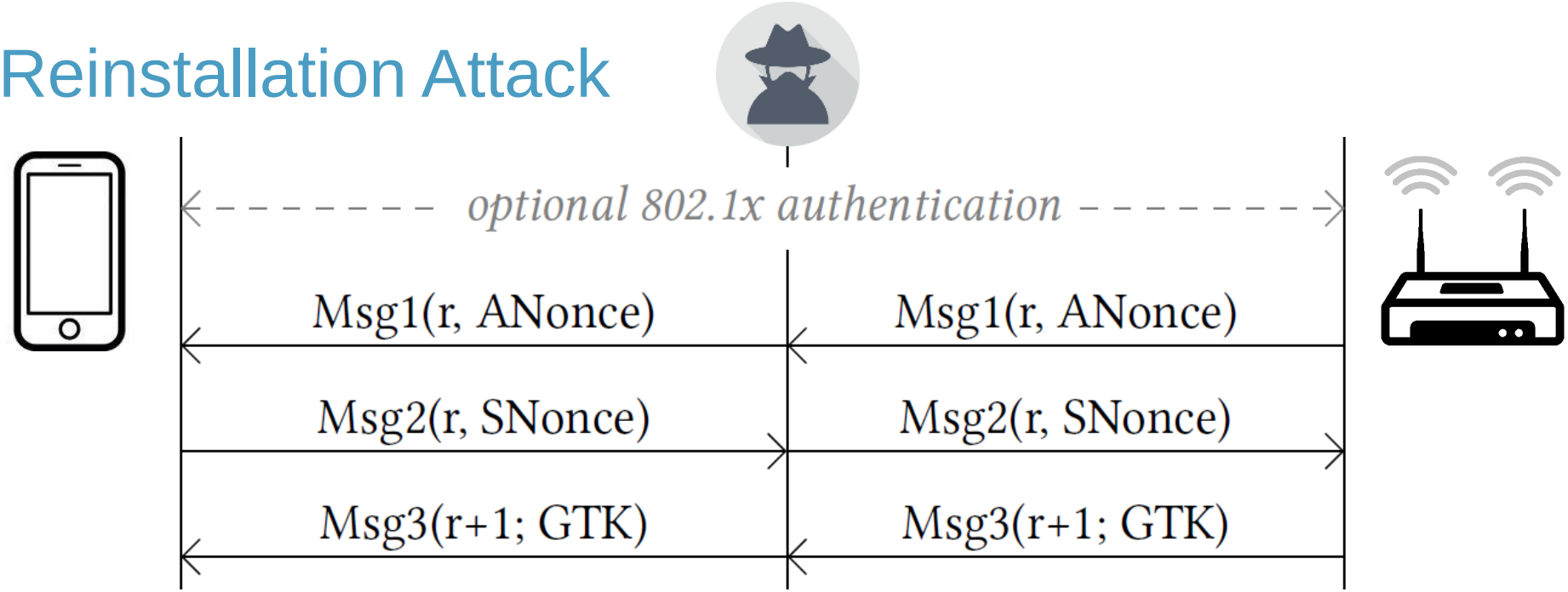


Reinstallation Attack

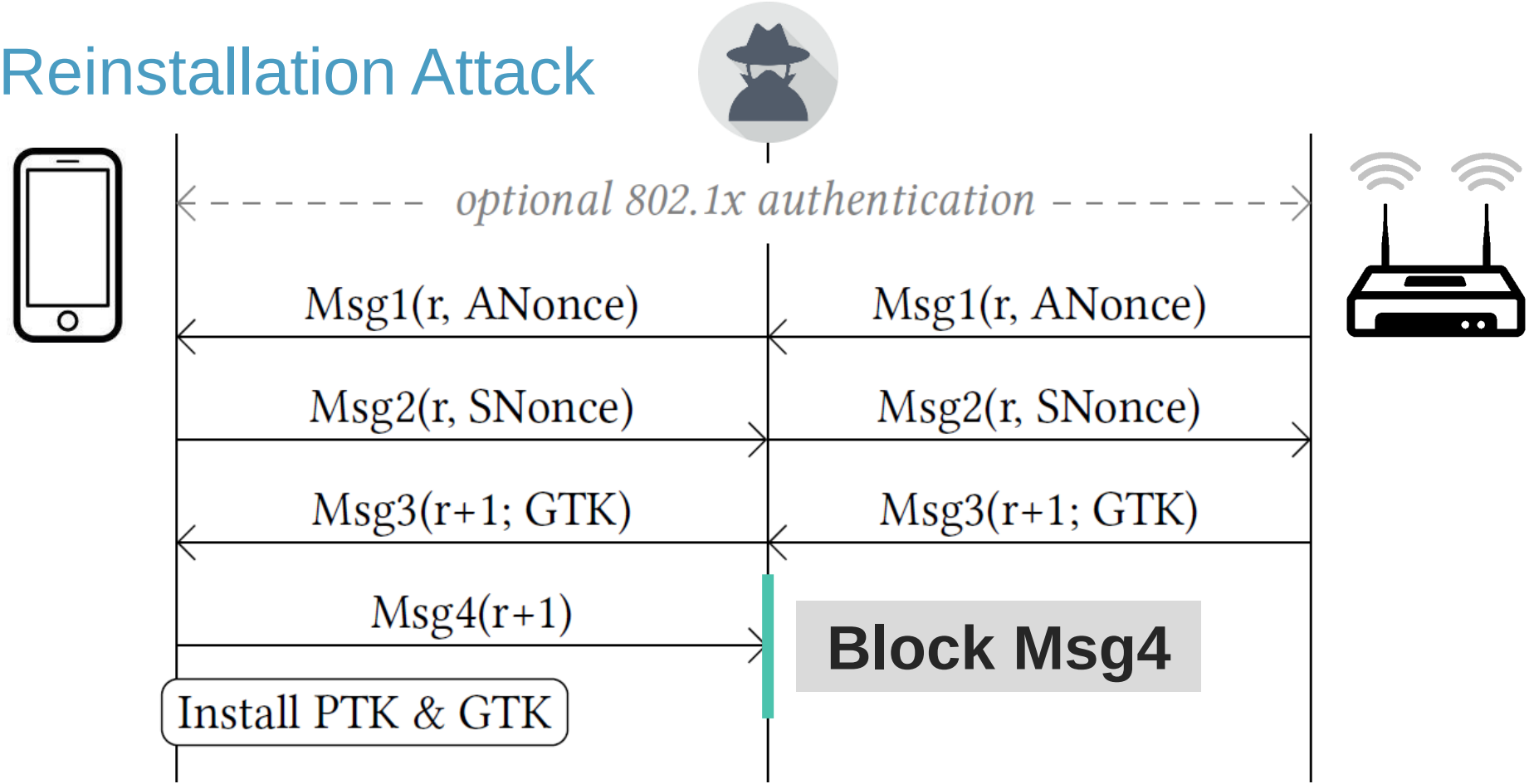
= Adversary establishes a **Multi-Channel Machine-in-the-Middle** position

Reinstallation Attack

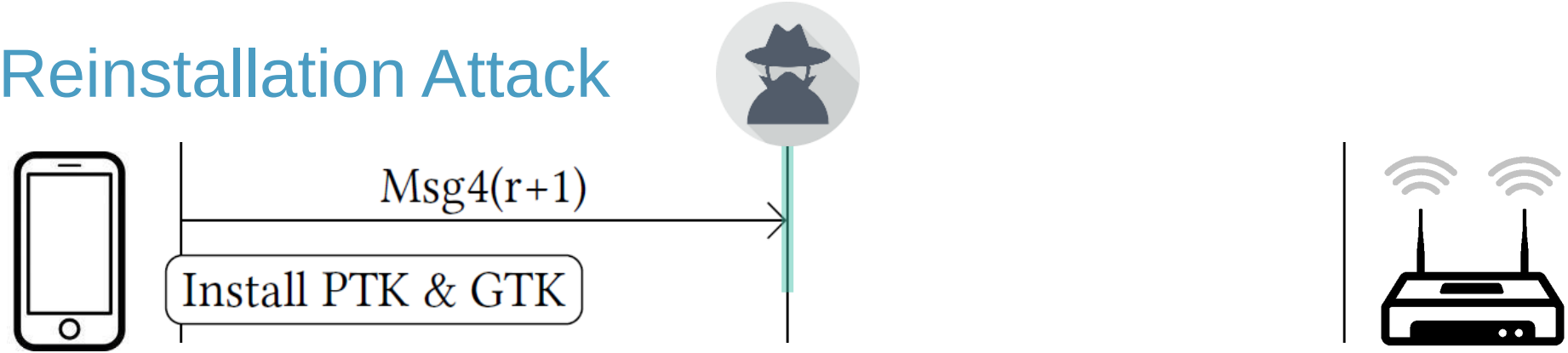
Reinstallation Attack



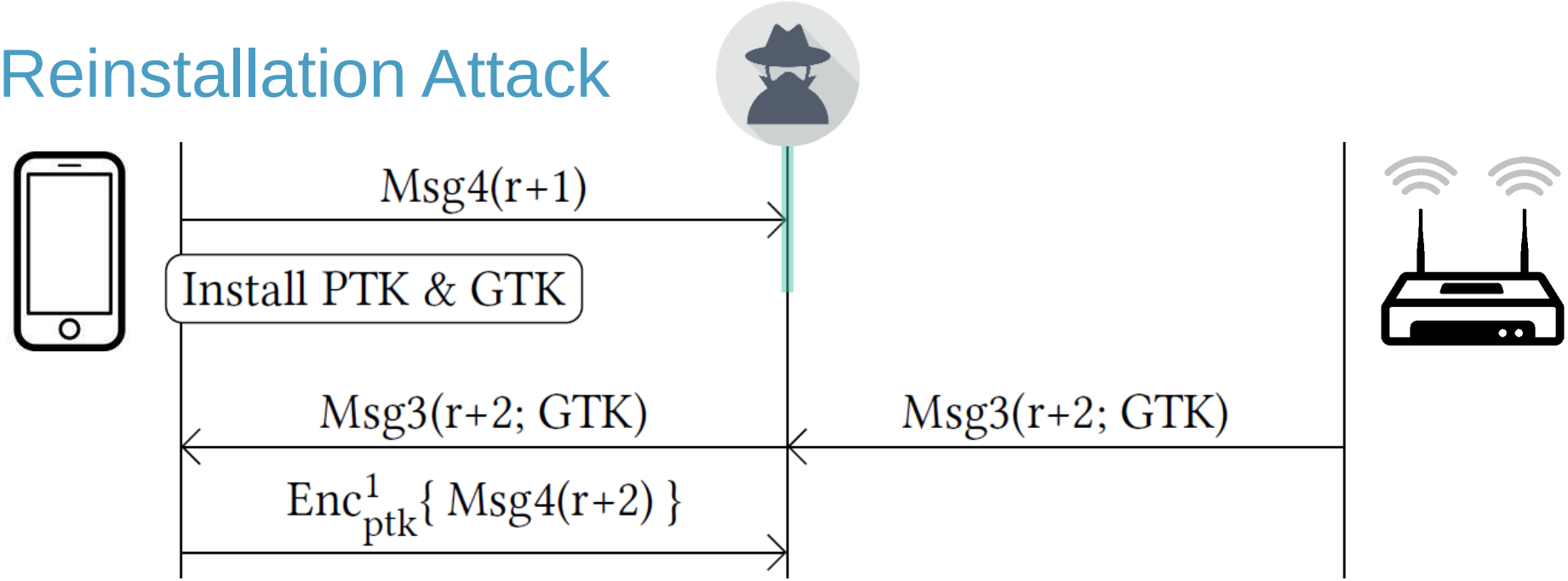
Reinstallation Attack



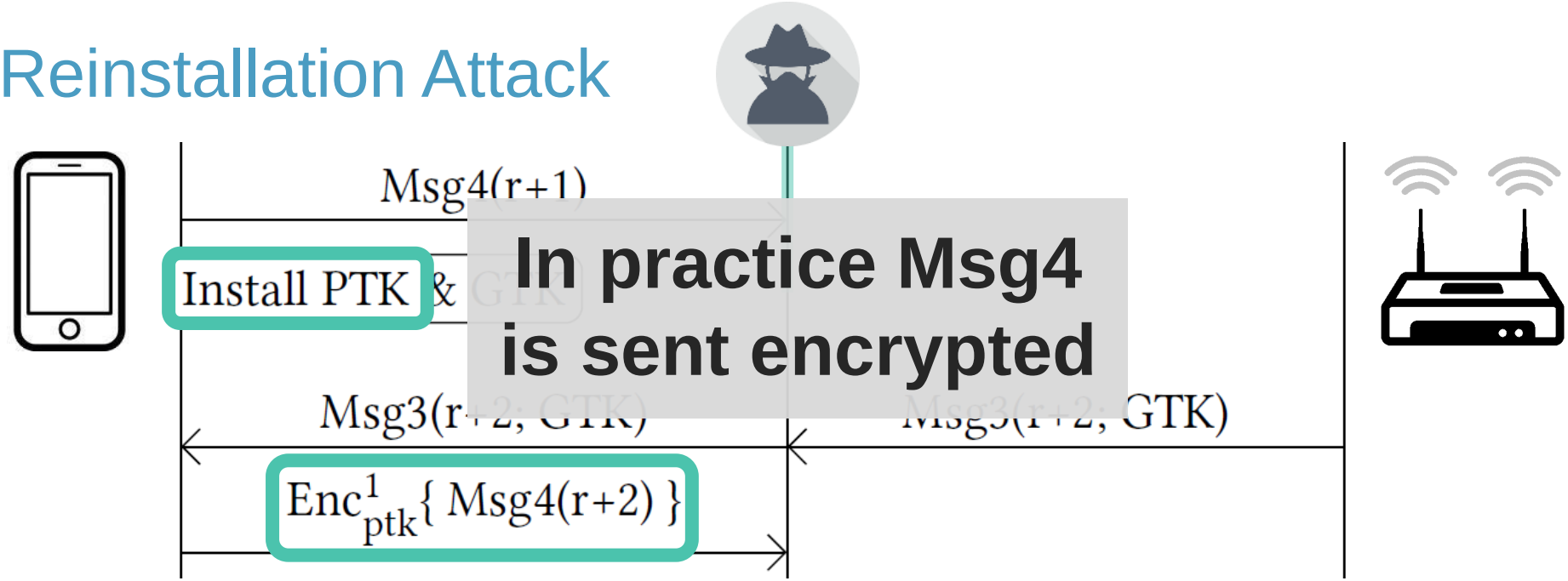
Reinstallation Attack



Reinstallation Attack

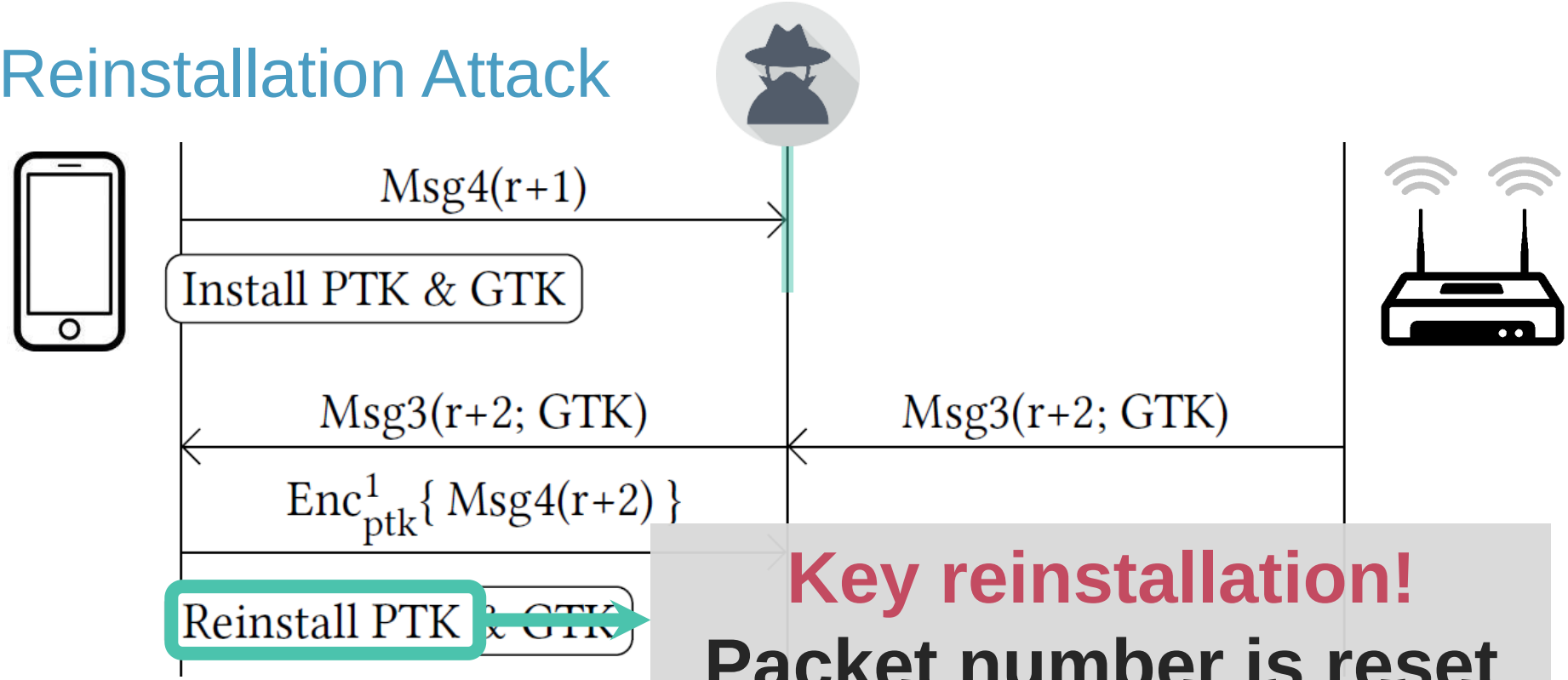


Reinstallation Attack



In practice Msg4 is sent encrypted

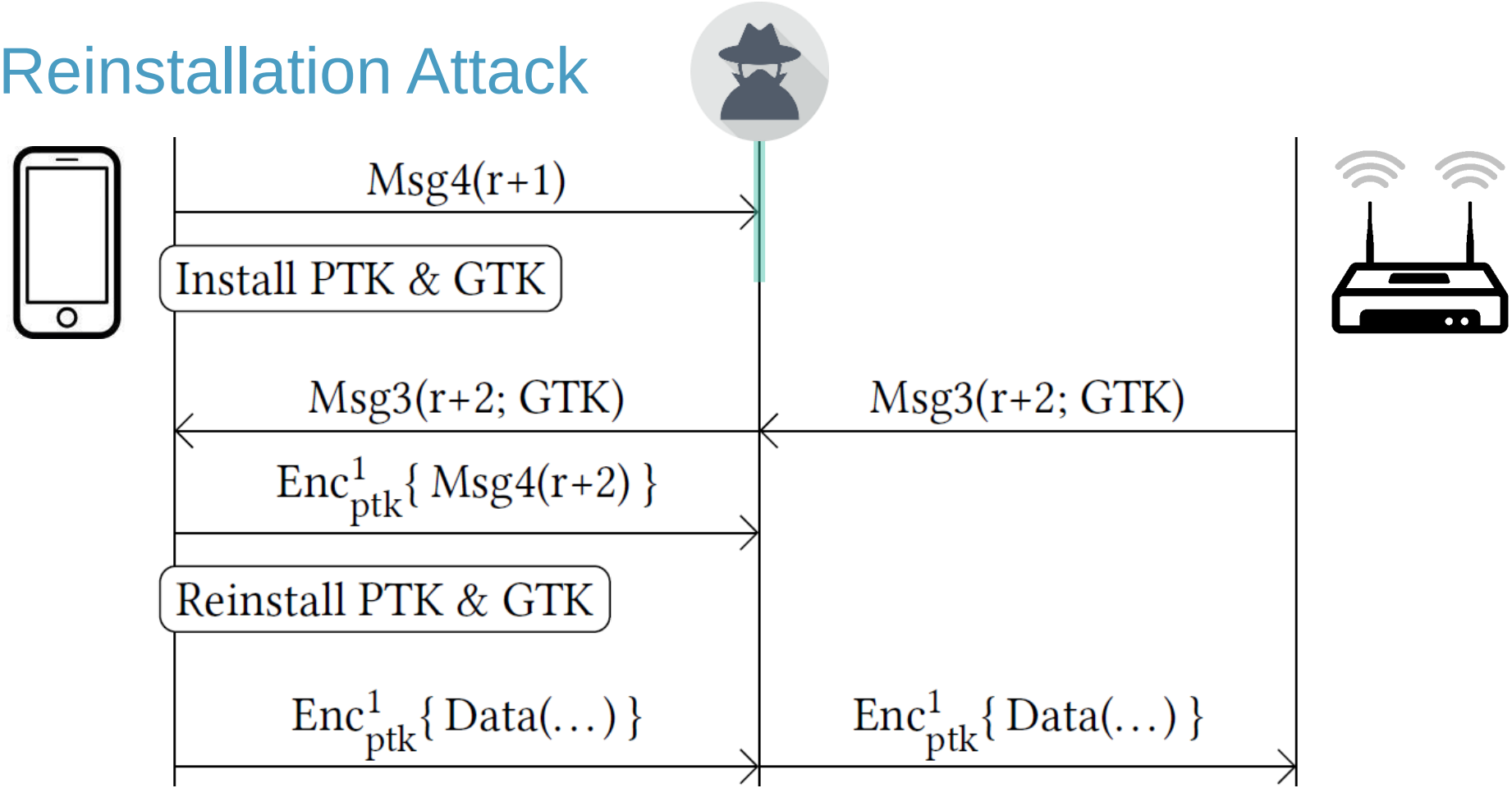
Reinstallation Attack



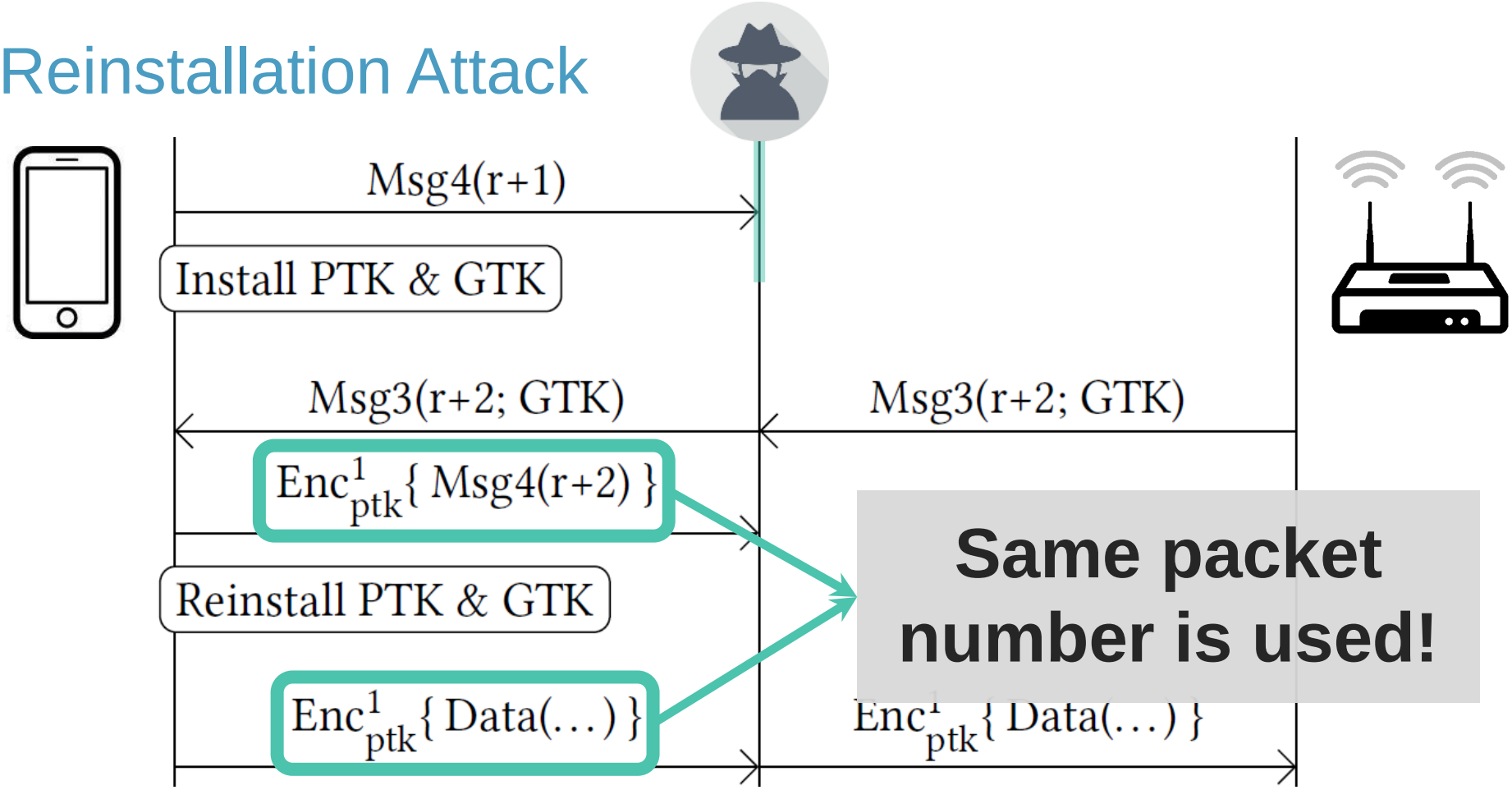
Key reinstallation!

Packet number is reset

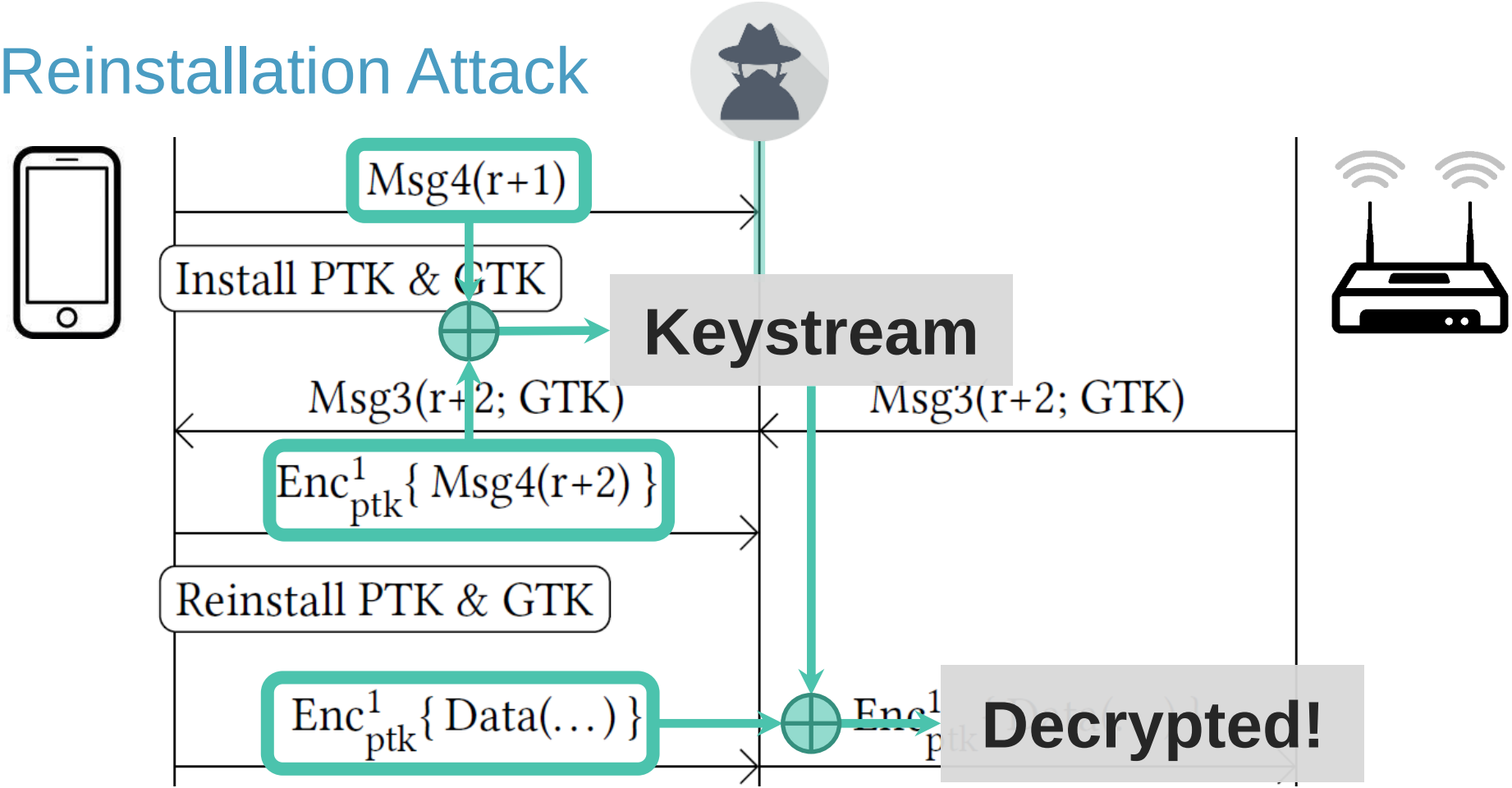
Reinstallation Attack



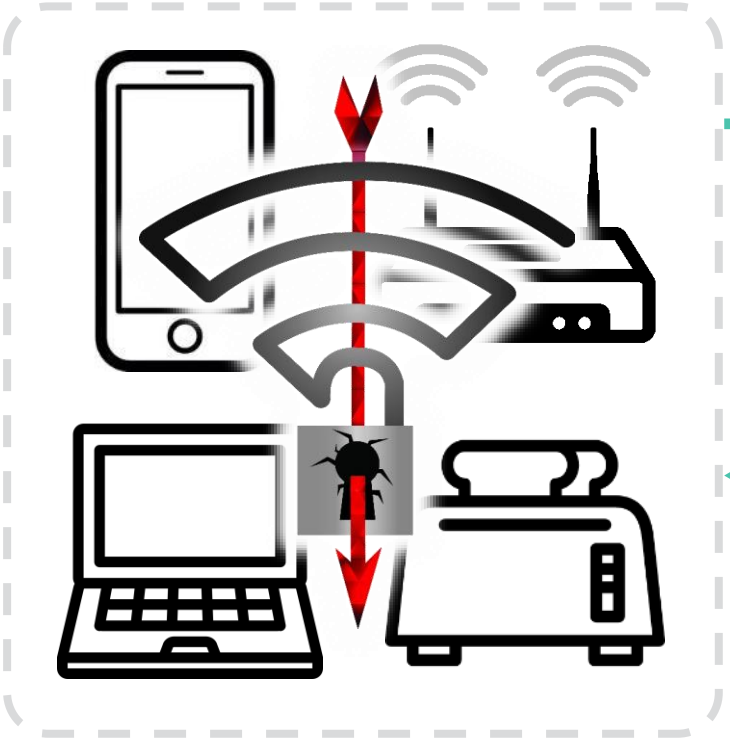
Reinstallation Attack



Reinstallation Attack



General impact



Transmit packet number reset

Decrypt frames sent by victim

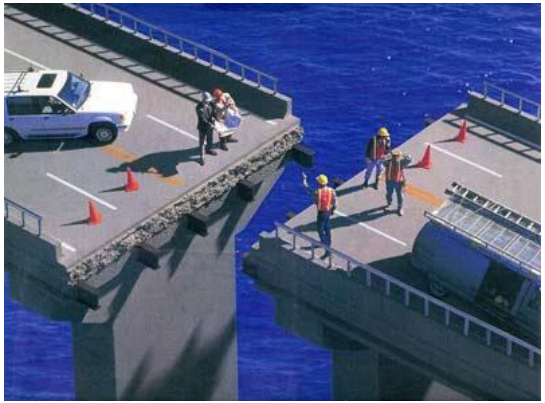
Receive replay counter reset

Replay frames towards victim

Root cause

- 4-way handshake proven secure
- Encryption protocol proven secure

} Combined in a state machine



State machine was not proven secure!

What about key reinstallations in other protocols?

What does reinstallation mean?

- Wi-Fi *installs* keys, then *enables* Rx and/or Tx for that key. Few protocols use this terminology or separation.

The *attack technique* does apply to other protocols:

- Narrow definition: replaying, or causing the retransmission of, a handshake message to trigger key/nonce reuse.
- Broader but more vague definition: **manipulating the handshake to cause key/nonce reuse**.

→ Similar flaws appear unlikely in other protocol standards

Key reinstallations in protocol implementations?

Key/nonce reuse in implementations is more common:

Key reinstallation against Wi-Fi access points in IWD[CCS22] and MediaTek[OPCDE18]

Keystream reuse in VoLTE[RKHP20]

Nonce reuse in buggy TLS libraries[BZDSJ16]

→ **When fuzzing**/auditing libraries, **check for key/nonce reuse** (that is possibly triggered by handshake manipulation)

Presentation Outline

Recent attacks:

- Key reinstallation attacks in WPA2 (= KRACK)
- **Side-channel leaks in WPA3 (= Dragonblood)**
- Fragmentation issues in WPA* (= FragAttacks)

New defenses:

- Opportunistic wireless encryption (Wi-Fi Alliance)
- Beacon protection & channel validation (our work ☺)
- SAE-PK to protect hotspots (Wi-Fi Alliance)

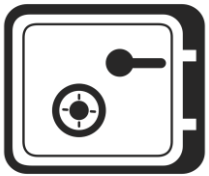
After KRACK we got a new handshake ☺

Late 2018: release of Wi-Fi Protected Access 3 (WPA3)

- Uses a **Password Authenticated Key Exchange (PAKE)**
- Simultaneous Authentication of Equals (**SAE**)

Provides mutual authentication

Negotiates session key



Forward secrecy & prevents offline dictionary attacks

Protects against server compromise

After KRACK we got a new handshake ☺

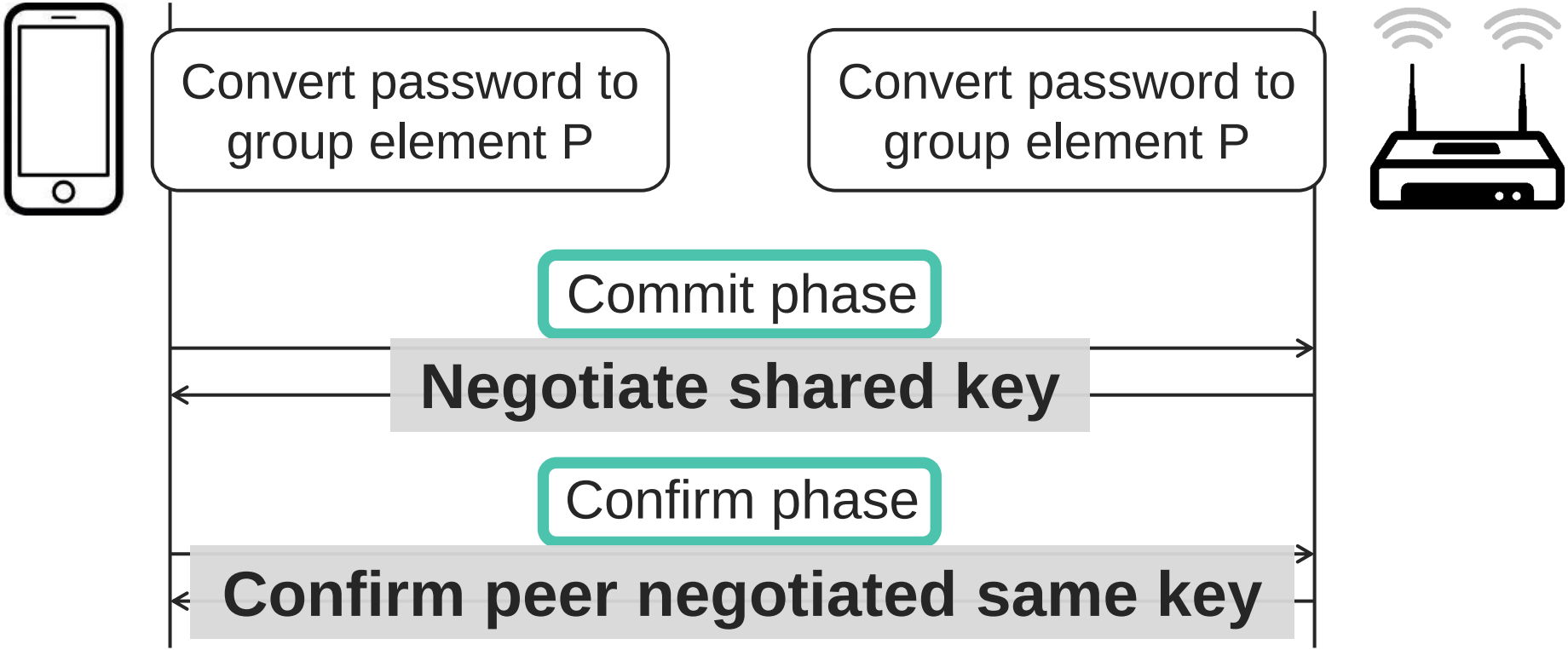
Late 2018: release of Wi-Fi Protected Access 3 (WPA3)

- Uses a **Password Authenticated Key Exchange (PAKE)**
- Simultaneous Authentication of Equals (**SAE**)

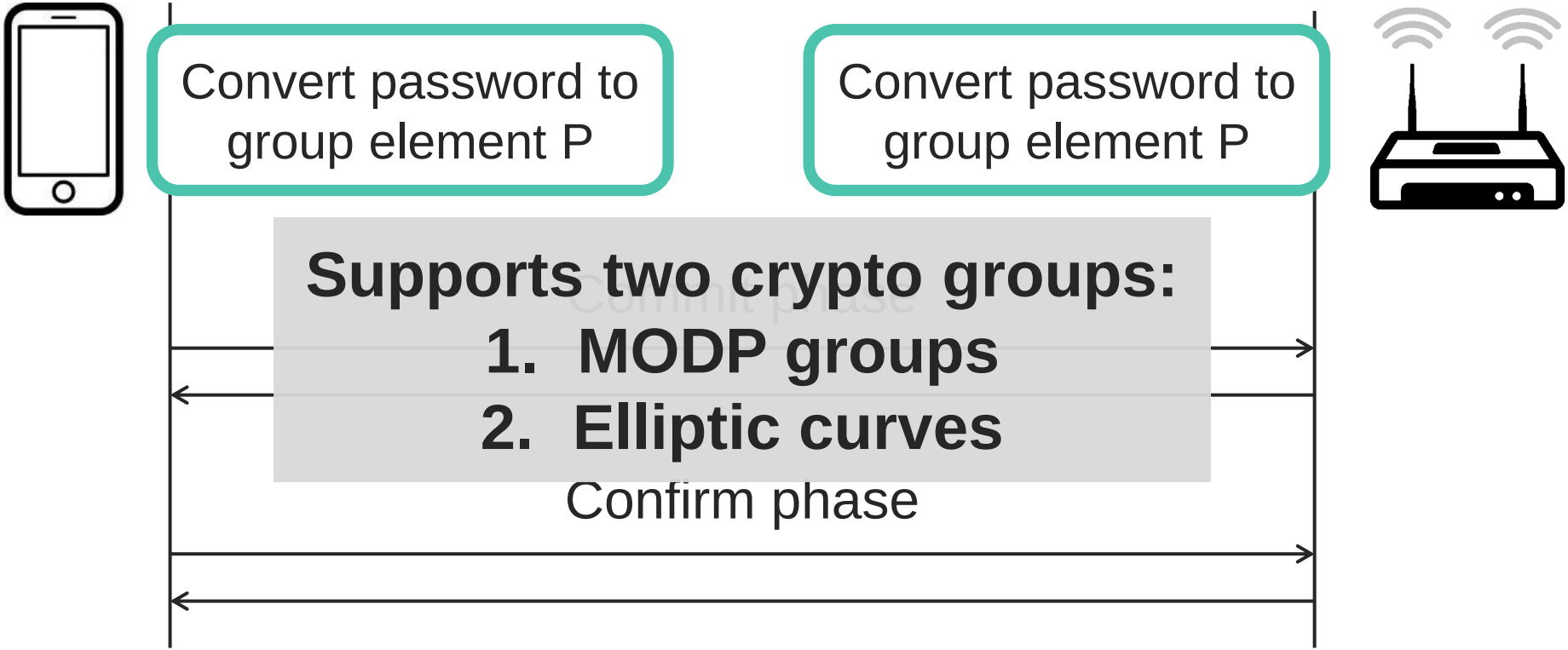
Also called the “Dragonfly” handshake

- Originally for mesh networks (2008 / 2011)
- Made part of WPA3 without academic feedback
- Vulnerable to **Dragonblood** side-channels

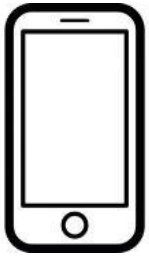
Dragonfly



Dragonfly

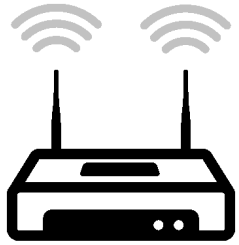


Dragonfly



Convert password to group element P

Convert password to group element P

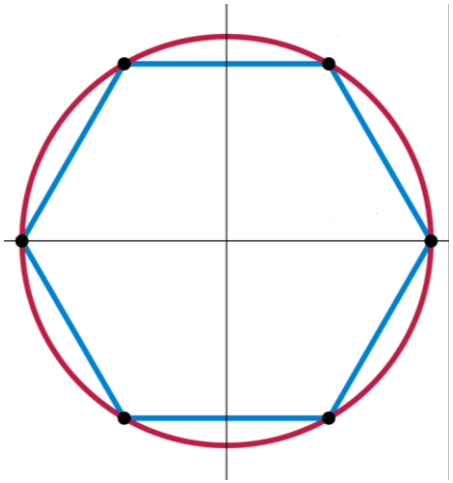


Supports two crypto groups:

1. **MODP groups**
2. **Elliptic curves**

Confirm phase

What are MODP groups?



Operations performed on integers x where:

- $x < p$ with p a prime
- $x^q \bmod p = 1$ must hold
- q = #elements in the group

→ All operations are **MOD**ulo the **P**rime (= MODP)

Convert password to MODP element

```
for (counter = 1; counter < 256; counter++)
    value = hash(pw, counter, addr1, addr2)
    if value >= p: continue
```

$P = value^{(p-1)/q}$

```
    return P
```

Convert value to a MODP element

Convert password to MODP element

```
for (counter = 1; counter < 256; counter++)
    value = hash(pw, counter, addr1, addr2)
    if value >= p: continue
    P = value^((p-1)/q)
    return P
```

Problem for groups 22-24: high chance that value >= p

Convert password to MODP element

```
for (counter = 1; counter < 256; counter++)
    value = hash(pw, counter, addr1, addr2)
    if value >= p: ???
    P = value^((p-1)/q)
    return P
```

Convert password to MODP element

```
for (counter = 1; counter < 256; counter++)
    value = hash(pw, counter, addr1, addr2)
    if value >= p: continue
    P = value^((p−1)/q)
    return P
```

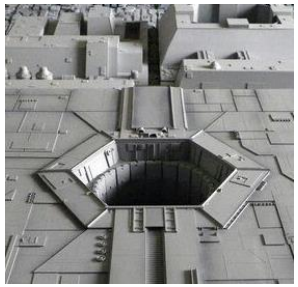
Convert password to MODP element

```
for (counter = 1; counter < 256; counter++)
    value = hash(pw, counter, addr1, addr2)
    if value >= p: continue
    P = value^((p-1)/q)
return P
```

#iterations depends on password

No timing leak countermeasures,
despite warnings by IETF & CFRG!

IETF mailing list in 2010



“[..] **susceptible to side channel (timing) attacks** and may leak the shared password.”

“not so sure how important that is [..] **doesn't leak the shared password** [..] not a trivial attack.”

Leaked information: #iterations needed

Client address	addrA
Measured	

Leaked information: #iterations needed

Client address	addrA
Measured	
Password 1	
Password 2	
Password 3	

Leaked information: #iterations needed

Client address	addrA
Measured	
~~Password 1~~	
Password 2	
Password 3	

What information is leaked?

```
for (counter = 1; counter < 256; counter++)
    value = hash(pw, counter, addr1, addr2)
    if value >= p: continue
    P = value^((p−1)/q)
```

What information is leaked?

```
for (counter = 1; counter < 256; counter++)
    value = hash(pw, counter, addr1, addr2);
    if value >= p: continue
    P = value^((p-1)/q)
```

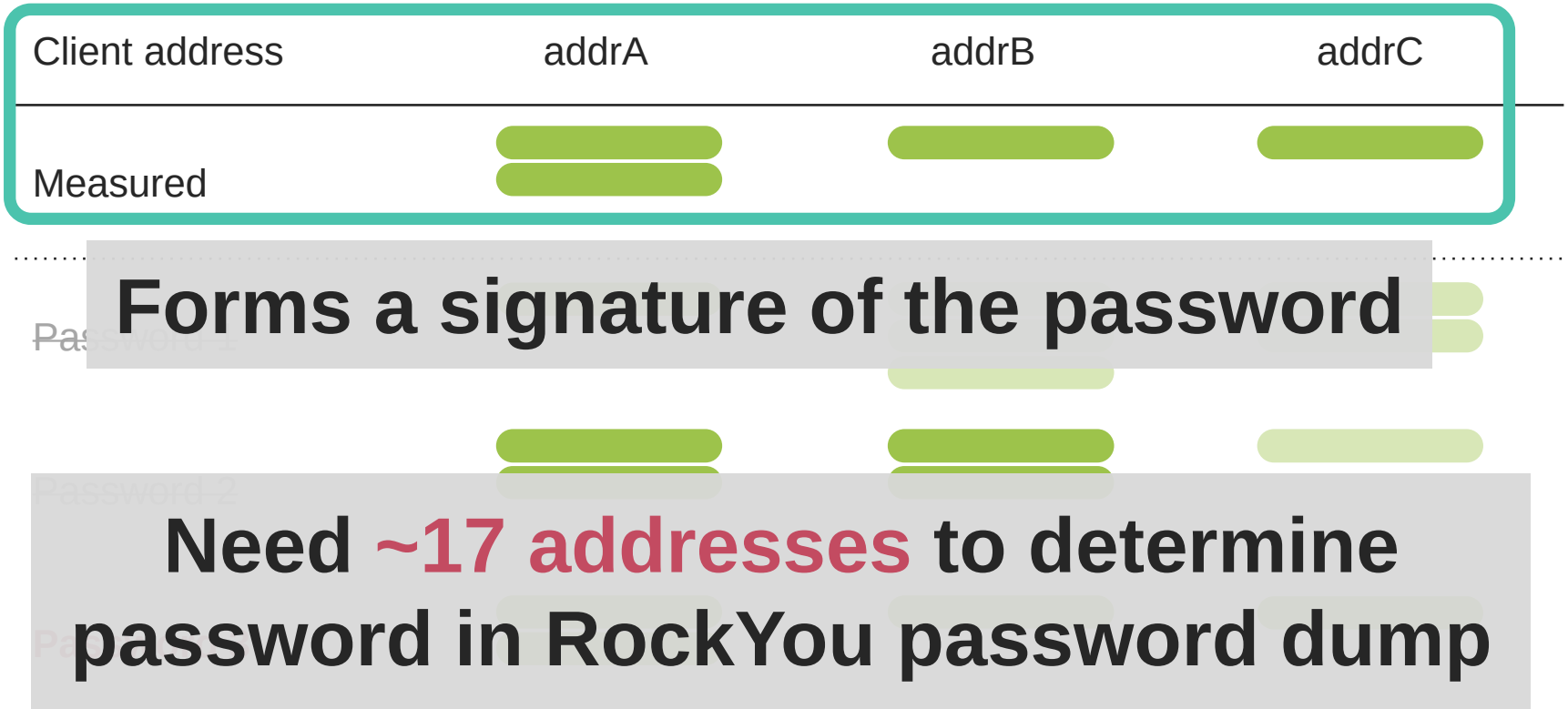
Spoof client address to obtain different execution & leak new data

Leaked information: #iterations needed

Leaked information: #iterations needed

Leaked information: #iterations needed

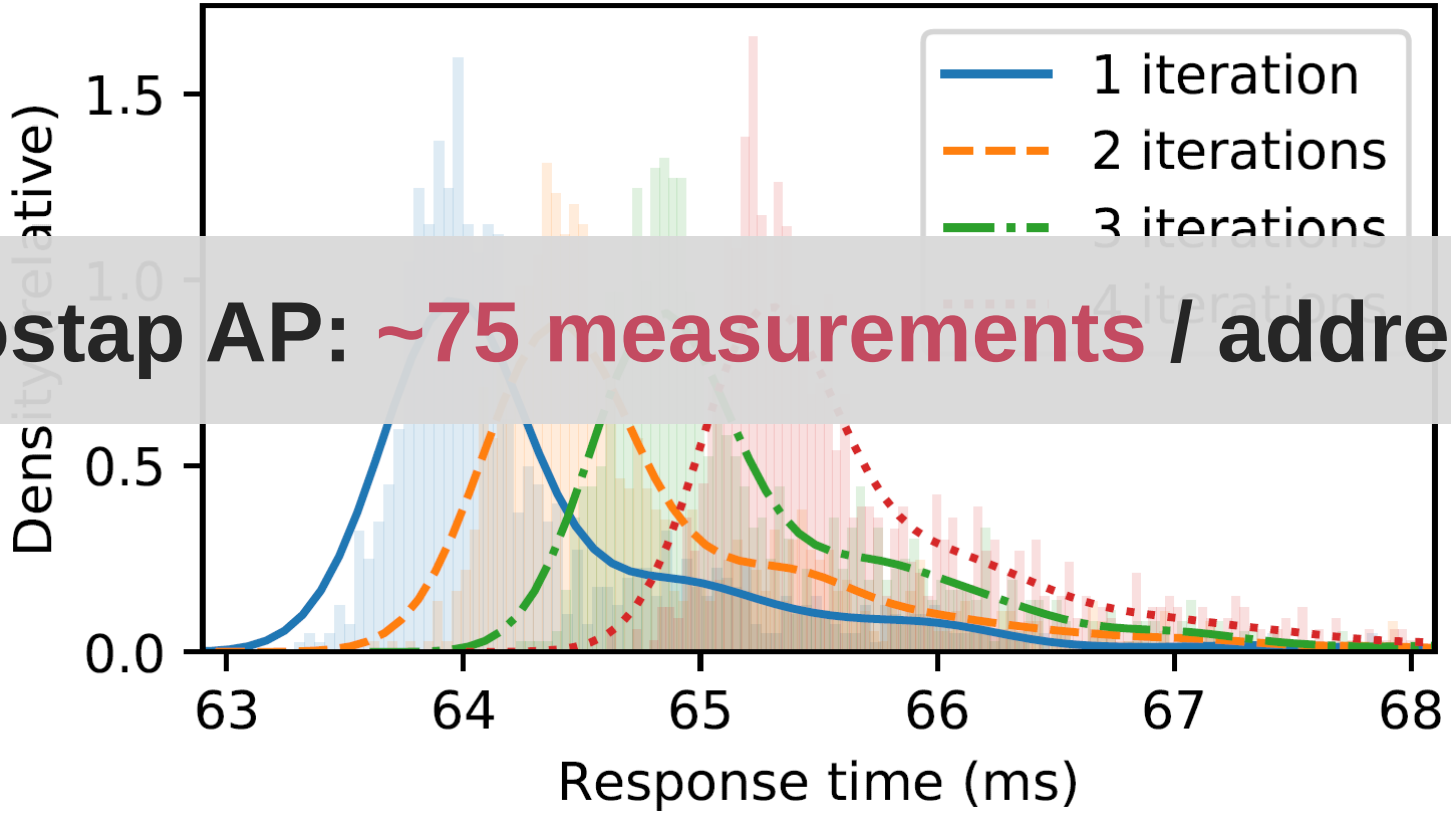
Leaked information: #iterations needed



Forms a signature of the password

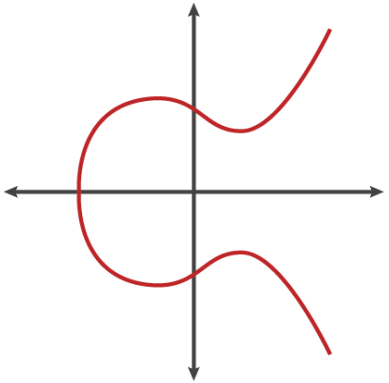
Need ~17 addresses to determine password in RockYou password dump

Raspberry Pi 1 B+: differences are measurable



Hostap AP: ~75 measurements / address

What about elliptic curves?



Operations performed on points (x, y) where:

- $x < p$ and $y < p$ with p a prime
- $y^2 = x^3 + ax + b \bmod p$ must hold

Similar algorithm to **convert password to point (x,y)**:

- EAP-PWD: vulnerable to the same timing attack.
- WPA3: always does 40 loops. But **variance of the execution time** may still leak info & **cache attacks** are possible.

Fixing the root cause

Improve password conversion algorithm

- Use **hash-to-element conversion** instead
  - Simplified Shallue-Woestijne-Ulas (S-SWU)
- Easier to implement in constant time

Newly certified devices must implement hash-to-element

- Still called WPA3 → not easy to tell what a device supports
- **WPA3 > WPA2** so you should always switch to WPA3!

Presentation Outline

Recent attacks:

- Key reinstallation attacks in WPA2 (= KRACK)
- Side-channel leaks in WPA3 (= Dragonblood)
- **Fragmentation issues in WPA* (= FragAttacks)**

New defenses:

- Opportunistic wireless encryption (Wi-Fi Alliance)
- Beacon protection & channel validation (our work ☺)
- SAE-PK to protect hotspots (Wi-Fi Alliance)

Background

Large frames have a high chance of being corrupted:

Avoid by **fragmenting** & only retransmitting lost fragments:

Background

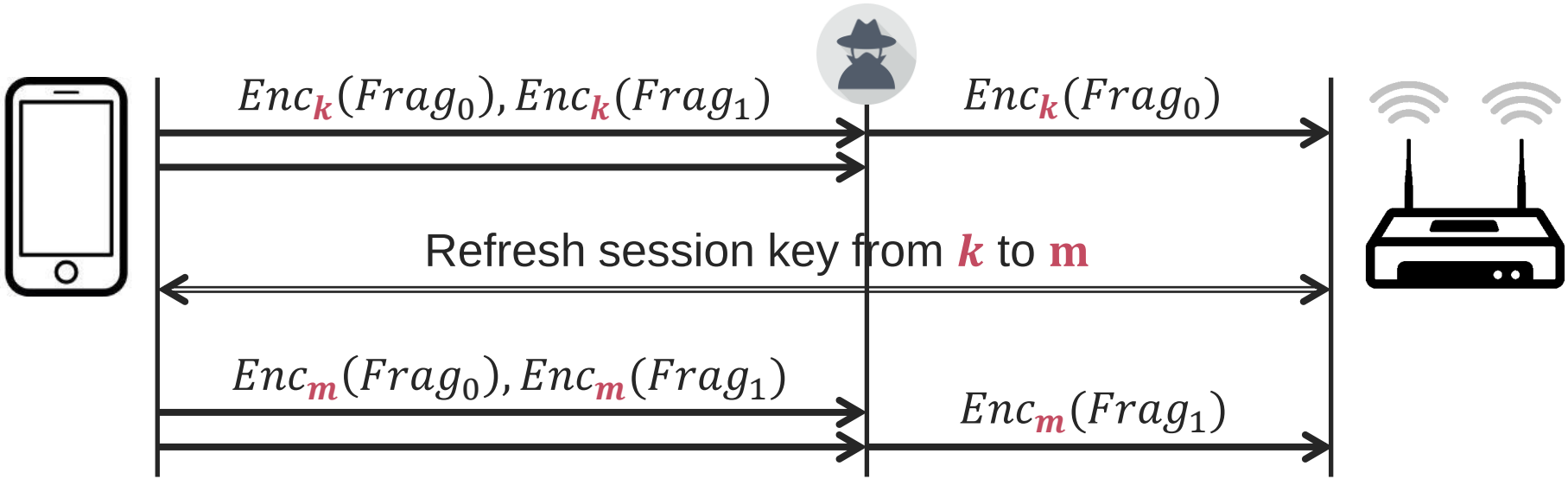
Large frames have a high chance of being corrupted:

Avoid by **fragmenting** & only retransmitting lost fragments:

→ **Protected header** info defines place in original frame

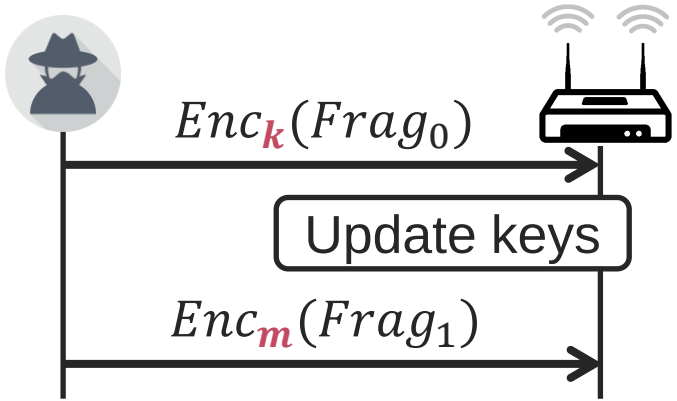
Mixed key design flaw

Fragments decrypted with **different keys are reassembled**:



→ Can **mix fragments of different frames**

Root cause: bad managing of security contexts



- **Receiver** doesn't securely handle security context changes.
- Can sometimes also mix plaintext with encrypted frames

We also discovered various implementation flaws:

- Can sometimes mix plaintext with encrypted fragments
- Devices accept specially-constructed **plaintext frames**
  - For instance, fragmented plaintext frames, …

How does a transmitter handle security contexts?

Kr00k attack discovered by ESET during KRACK tests

› Send disassociation to make station delete security context

Queued frames still get transmitted. Two variants:

1. Frames get encrypted using an all-zero key[RSA20]
2. Frames have security header but plaintext data[BH20]

→ Need to securely **manage security context changes** throughout the whole network stack.

Presentation Outline

Recent attacks:

- Key reinstallation attacks in WPA2 (= KRACK)
- Side-channel leaks in WPA3 (= Dragonblood)
- Fragmentation issues in WPA* (= FragAttacks)

New defenses:

- **Opportunistic wireless encryption (Wi-Fi Alliance)**
- Beacon protection & channel validation (our work ☺)
- SAE-PK to protect hotspots (Wi-Fi Alliance)

Security for open networks?

Problem: open networks don't use encryption.

Goal: prevent passive attacks. Inspired by:

- RFC 7258: "Pervasive Monitoring Is an Attack"
- RFC 7434: "Opportunistic Security: Some Protection Most of the Time"

→ Wi-Fi Alliance solution: **Diffie-Hellman to negotiate keys** without authentication of the network.

Opportunistic Wireless Encryption (OWE)

Based on RFC 8110 by D. Harkins & W. Kumari:

- Perform a **Diffie-Hellman** key exchange to negotiate pairwise master key (PMK). Use this in 4-way handshake.
- Clients can **reconnect using previous PMK** if the AP still remembers the PMK of the client (likely easy to DoS).
- Mandates usage of **Management Frame Protection** (MFP), which prevents common disconnection attacks.

Analysis of OWE

Reasons to use:

- Clients are harder to disconnect due to usage of MFP
- Requires active attacks to intercept traffic

Is it worth the effort?

- It's unknown who is passively monitoring Wi-Fi. How do we know they won't move to active attacks?

→ Cost/benefit seems **open to discussion**

Presentation Outline

Recent attacks:

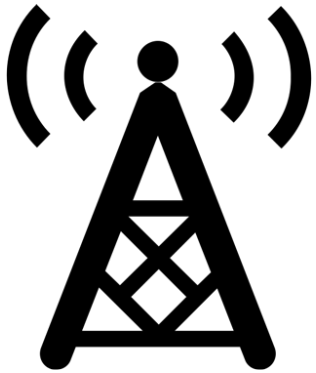
- Key reinstallation attacks in WPA2 (= KRACK)
- Side-channel leaks in WPA3 (= Dragonblood)
- Fragmentation issues in WPA* (= FragAttacks)

New defenses:

- Opportunistic wireless encryption (Wi-Fi Alliance)
- **Beacon protection** & channel validation (our work ☺)
- SAE-PK to protect hotspots (Wi-Fi Alliance)

Background: beacons

- Wi-Fi networks use beacons to announce their presence
- They are sent every ~100 ms by an Access Point



Contains properties of the network:

- Name of the network
- Supported bitrates (e.g. 11n or 11ac)
- Regulatory constraints (e.g. transmission power)
- ...

Beacons are not protected

```
› Tag: SSID parameter set: cisco
› Tag: Supported Rates 1(B), 2(B), 5.5(B), 11(B), 6, 9, 12, 18, [Mbit/sec]
› Tag: DS Parameter set: Current Channel: 1
› Tag: Traffic Indication Map (TIM): DTIM 0 of 0 bitmap
› Tag: Country Information: Country Code GB, Environment Unknown (0x04)
› Tag: Power Constraint: 3
› Tag: ERP Information
› Tag: Extended Supported Rates 24, 36, 48, 54, [Mbit/sec]
› Tag: QBSS Load Element 802.11e CCA Version
› Tag: RM Enabled Capabilities (5 octets)
› Tag: HT Capabilities (802.11n D1.10)
› Tag: RSN Information
› Tag: Mobility Domain
› Tag: HT Information (802.11n D1.10)
› Tag: Extended Capabilities (10 octets)
› Ext Tag: HE Capabilities (IEEE Std 802.11ax/D3.0)
› Ext Tag: HE Operation (IEEE Std 802.11ax/D3.0)
› Ext Tag: Spatial Reuse Parameter Set
```

- WPA version & channel: verified when connecting
- All other fields can be spoofed by an adversary

Defense: authenticate beacons [WiSec'20]

Rely on **symmetric encryption**

- Reuse existing crypto primitives of Wi-Fi
- Makes it easiers for vendors to adopt the defense

We **defend against outsider attacks**

- Adversary doesn't possess network credentials
- Similar to protection of broadcast Wi-Fi traffic

Beacon protection: new element

We add a **new type-length-value element** to beacons:

- Clients that do not recognize this element will ignore it
- Nonce: incremental number to **prevent replay attacks**
- Message Integrity Check: **CMAC or GMAC** over the beacon
  - Existing crypto primitive of management frame protection
  - All WPA3-capable devices already support it

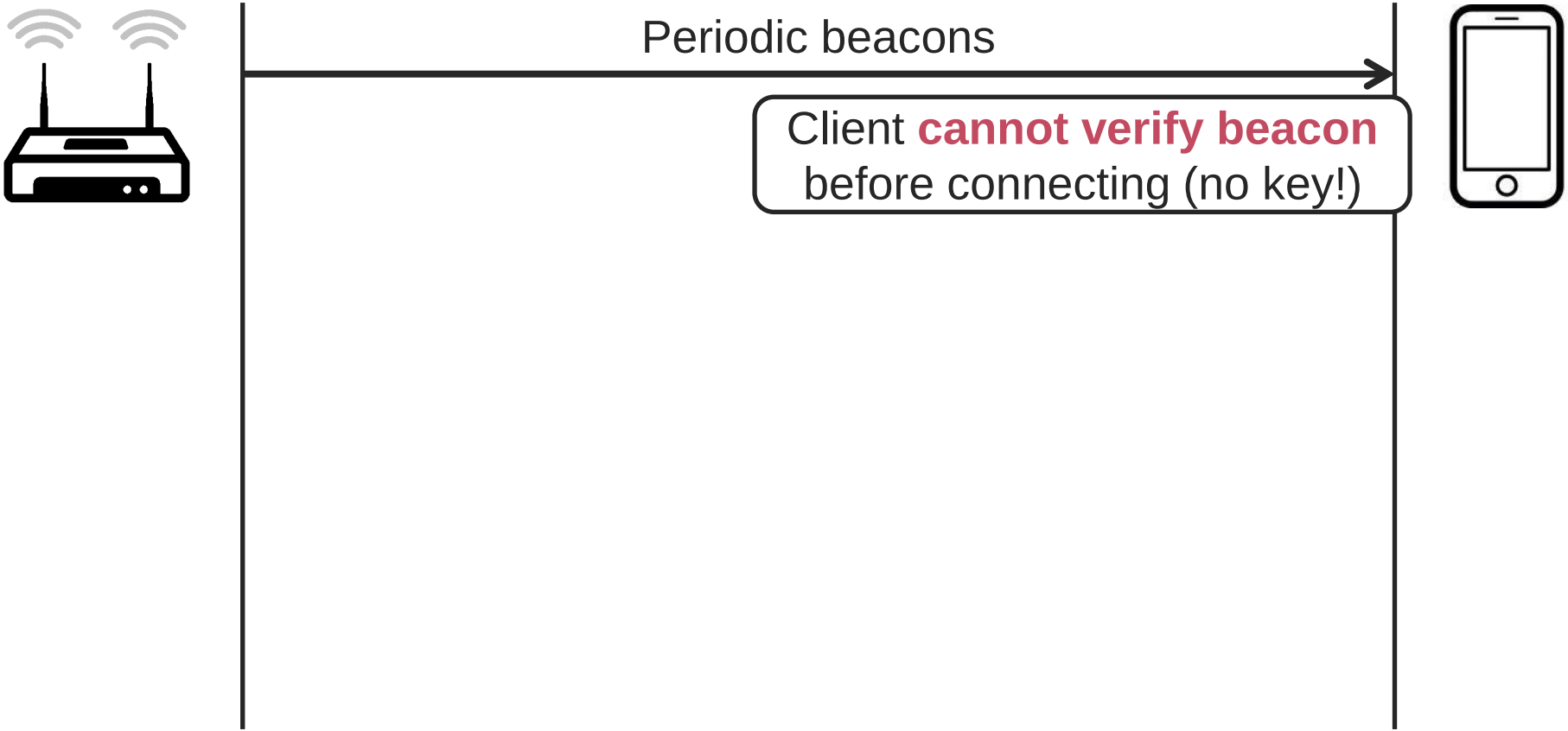
Key management

Key used to generate/verify the authenticity tag?

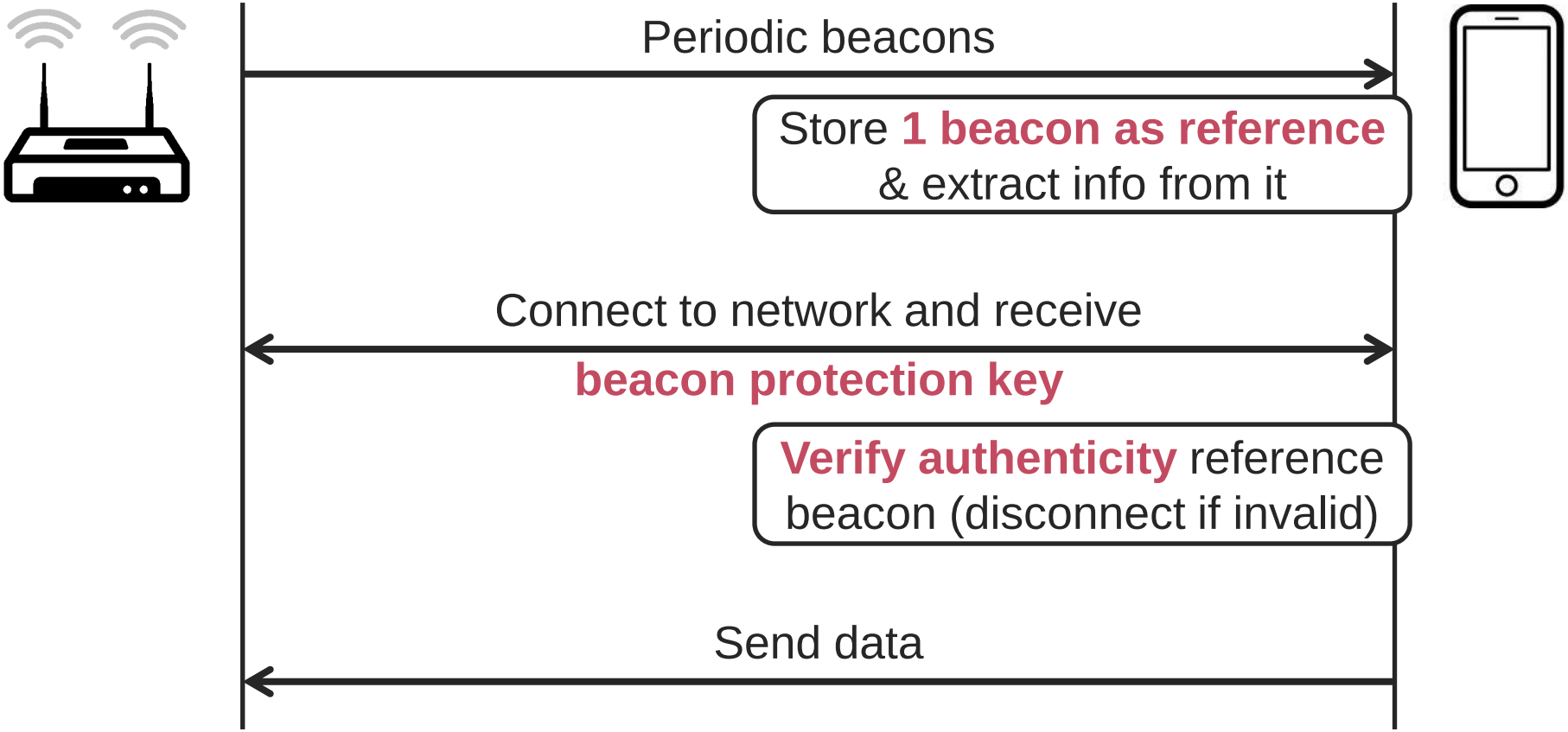
- AP generates a fresh **beacon protection key** when booting
- **AP always sends the beacon key** when a client connects
  - Older clients will ignore this key
  - New clients will enable beacon protection

→ Adversary can't manipulate handshake that transports the beacon key, **preventing downgrade attacks**.

Pre-authentication behavior

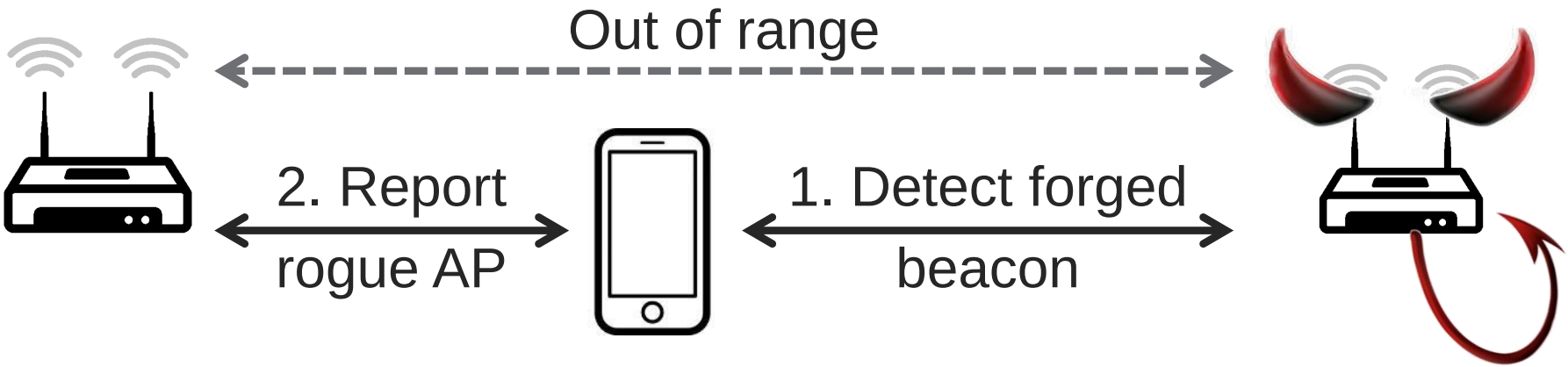


Pre-authentication behavior



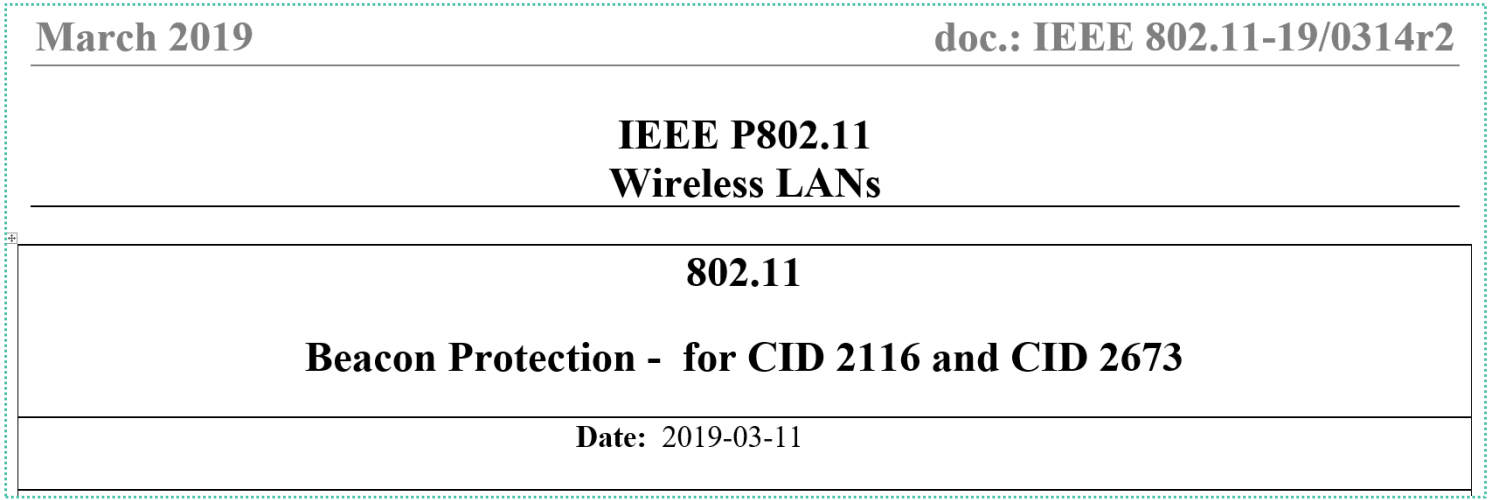
Reporting forged beacons

- Clients can report forged beacons to the AP
- Can now **detect far away rouge APs**



Specification

- Collaborated with industry to standardize our defense (Intel, Broadcom, Qualcomm and Huawei)
- **Now part of the 2020 update to the IEEE 802.11 standard**



March 2019 doc.: IEEE 802.11-19/0314r2

IEEE P802.11
Wireless LANs

802.11

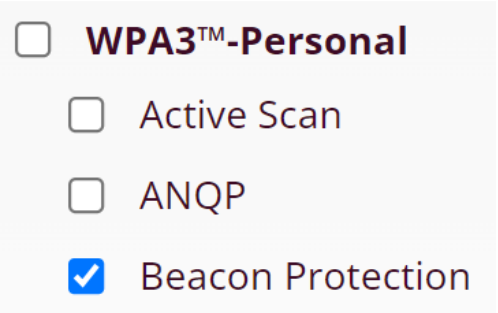
Beacon Protection - for CID 2116 and CID 2673

Date: 2019-03-11

Specification

- Collaborated with industry to standardize our defense (Intel, Broadcom, Qualcomm and Huawei)
- **Now part of the 2020 update to the IEEE 802.11 standard**

- Optional feature of WPA3
- Wi-Fi 7 **APs must support** beacon protection

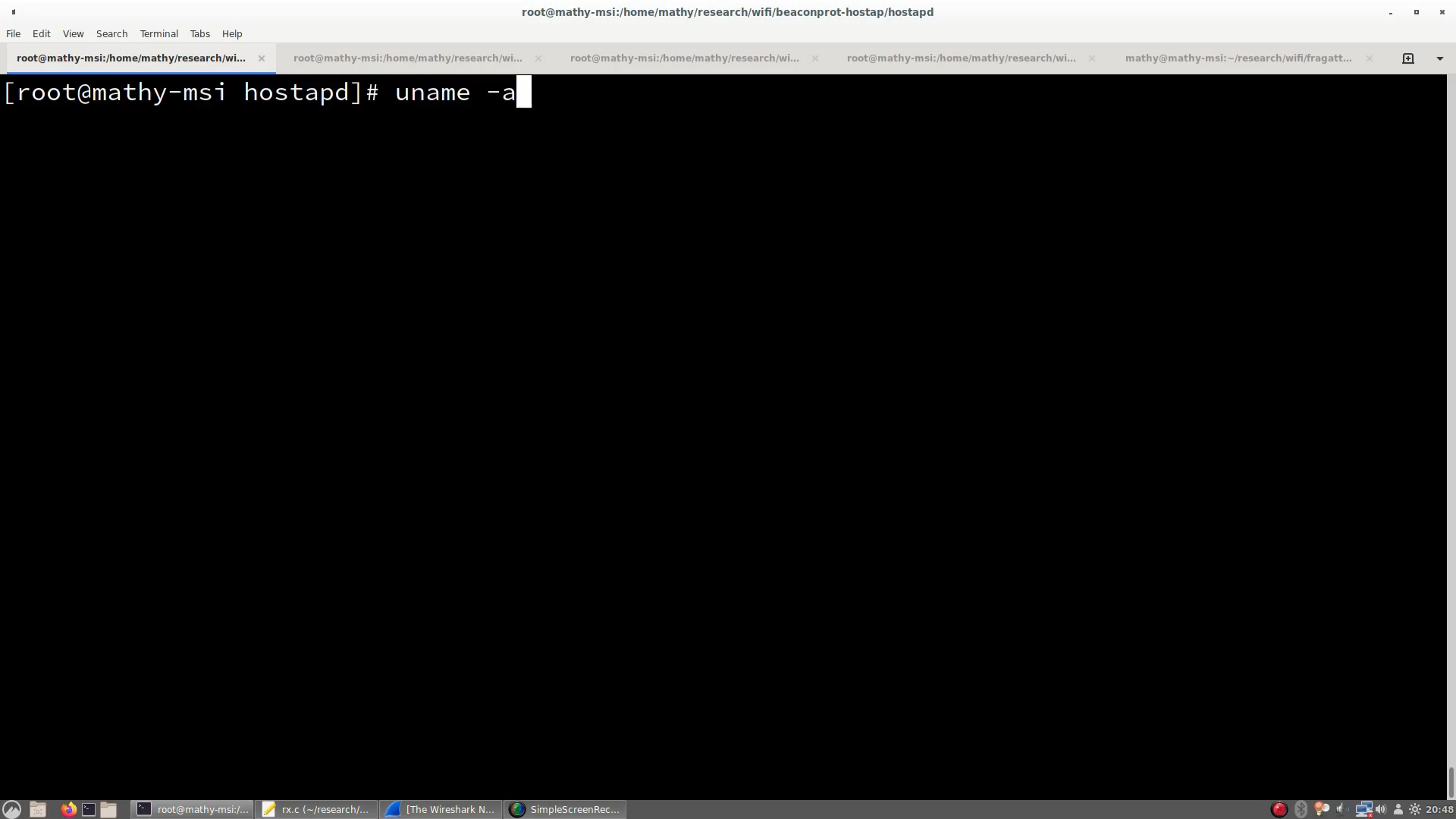


Implementation & demo

Has been independently (!) implemented by Linux

- Beacon signature calculated in hardware
- Requires firmware updates of Wi-Fi radios: beacons are usually generated in hardware.

DEMO


root@mathy-msi:/home/mathy/research/wifi/beaconprot-hostap/hostapd
File Edit View Search Terminal Tabs Help
root@mathy-msi:/home/mathy/research/wi...
root@mathy-msi:/home/mathy/research/wi...
root@mathy-msi:/home/mathy/research/wi...
root@mathy-msi:/home/mathy/research/wi...
mathy@mathy-msi:~/research/wifi/fragatt...
[root@mathy-msi hostapd]# uname -a
root@mathy-msi:/...
rx.c (~/research/...
[The Wireshark N...
SimpleScreenRec...
20:48

Presentation Outline

Recent attacks:

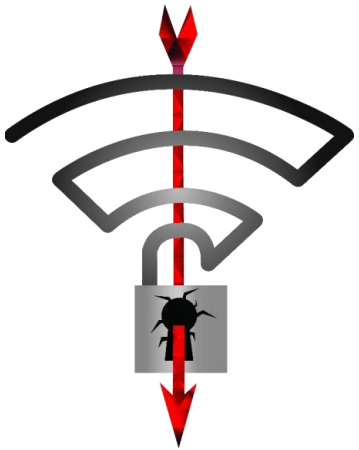
- Key reinstallation attacks in WPA2 (= KRACK)
- Side-channel leaks in WPA3 (= Dragonblood)
- Fragmentation issues in WPA* (= FragAttacks)

New defenses:

- Opportunistic wireless encryption (Wi-Fi Alliance)
- Beacon protection & **channel validation** (our work ☺)
- SAE-PK to protect hotspots (Wi-Fi Alliance)

Recent attacks use multi-channel MitM

- Network is cloned on **different channel**
- Allows adversary to reliably **block**, **delay**, or **modify** packets
- Used as the basis for advanced crypto attacks:



Attacks used special multi-channel MitM

AP is cloned on **different channel**

Connection succeeds &
can reliably manipulate frames!

Preventing multi-channel MitM [WiSec'18]

Verify operating channel when connecting to a network

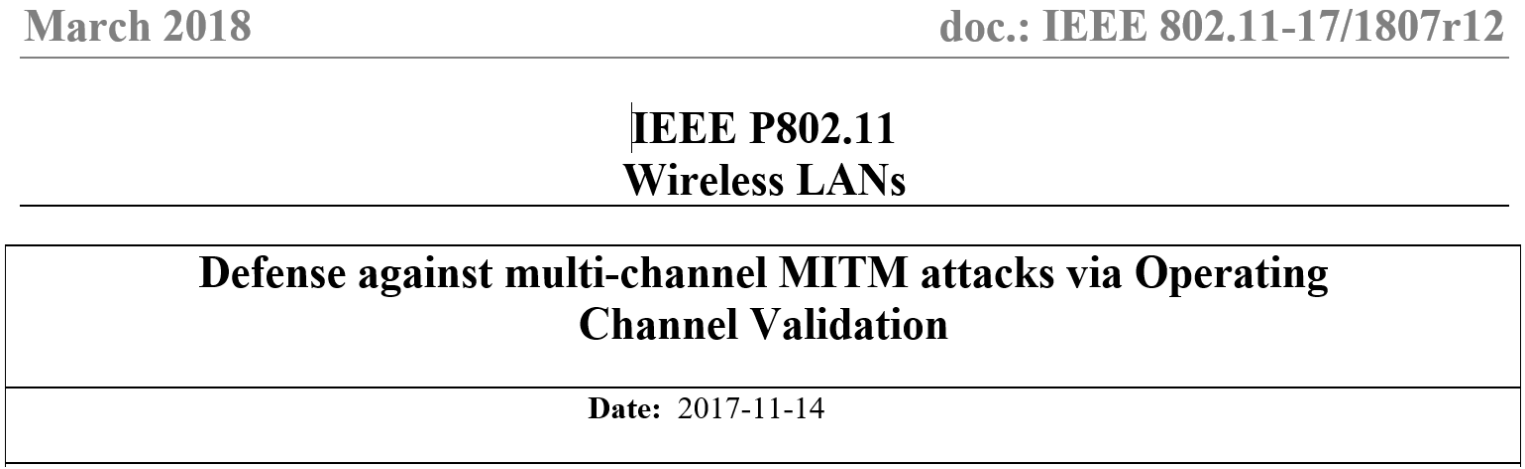
Also need to handle some edge cases

- After the clients wakes up from **sleep mode**
- When the network **switches channel** due to radar detection

→ Implemented on Linux in wpa_supplicant and hostapd

Specification

- Collaborated with industry to standardize defense (with Broadcom and Intel)
- **Now part of the 2020 update to the IEEE 802.11 standard**



March 2018 doc.: IEEE 802.11-17/1807r12

IEEE P802.11
Wireless LANs

Defense against multi-channel MITM attacks via Operating Channel Validation

Date: 2017-11-14

Specification

- Collaborated with industry to standardize defense (with Broadcom and Intel)
- **Now part of the 2020 update to the IEEE 802.11 standard**

- Recognized as an **optional feature of WPA3**
- Good initial step, hopefully becomes mandatory in future

Open questions

How to prevent a **repeater MitM** when devices are out of range?

Defense based on channel randomness & reciprocity?

› Could verify the "**channel signature**" between both devices

Presentation Outline

Recent attacks:

- Key reinstallation attacks in WPA2 (= KRACK)
- Side-channel leaks in WPA3 (= Dragonblood)
- Fragmentation issues in WPA* (= FragAttacks)

New defenses:

- Opportunistic wireless encryption (Wi-Fi Alliance)
- Beacon protection & channel validation (our work ☺)
- **SAE-PK to protect hotspots (Wi-Fi Alliance)**

WPA3 For Hotspots: SAE-PK

The Wi-Fi Alliance released SAE-PK in late 2020

- Extension of the SAE “Dragonfly” handshake
- Goal is to **secure hotspots using a password**…
- …but that password should not allow someone to create a rogue clone of the network!

Background: design history

They first requested comments from the community!

› https://www.wi-fi.org/security-development on July 2020:

> In addition, Wi-Fi Alliance has identified the following potential security protocol updates and will review all comments received:
>
> 15. Hash-to-element password generation, Client Privacy Mechanisms, Operation Channel Validation, and Beacon protection

Background: design history

They first requested comments from the community!

- https://www.wi-fi.org/security-development on July 2020
- This is an excellent step forward!

On the other hand, who knew about this?

- Getting the word out is still difficult
- Want to inspect drafts? Configure an e-mail warning when this page changes. Or follow me on Twitter **@vanhoef** ☺

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.

Is SAE-PK secure?

When not using the weakest security configuration:

- There are no known practical attacks…
- …on the other hand, **there is no formal analysis/proof**.

Open questions:

- Is some kind of security proof feasible?
- Are other attacks possible?
- Are there risks when implementing SAE-PK?

Major remaining issues & problems

The biggest issue: how to make Wi-Fi less complex?

- There are so many edge cases you will forget something…
- Backwards-compatibility at the price of security?

Complexity has further consequences:

- How to secure the network stack as a whole? User space, kernel, driver, firmware, hardware,… must interact securely.
- Modelling of protocols is inherently limited. But still useful!

How to access standards?

- 802.11 spec: https://standards.ieee.org/ieee/802.11/7028/
- Wi-Fi Alliance: https://www.wi-fi.org/security-development
- Draft IEEE docs: https://mentor.ieee.org/802.11/documents
  - Searchable using Google, use “search keywords site:mentor.ieee.org”

Other advice:

- Send an e-mail to ask for access to draft standards?
- Hostap implements many protocols: https://w1.fi/cvs.html
- Use mac80211_hwsim on Linux for virtual Wi-Fi interfaces

Thank you!
Questions?

References

- [OPCDE18]: M. Vanhoef. Presentation “Improved KRACK Attacks Against WPA2 Implementations” given at OPCDE Dubai, 2018.
- [CCS22]: C. M. Stone, S. L. Thomas, M. Vanhoef, J. Henderson, N. Bailluet, and T. Chothia. The Closer You Look, The More You Learn: A Grey-box Approach to Protocol State Machine Learning. To appear at the 29th ACM Conference on Computer and Communication Security (CCS 2022).
- [RKHP20]: Rupprecht, D., Kohls, K., Holz, T., & Pöpper, C. Call Me Maybe: Eavesdropping Encrypted LTE Calls With ReVoLTE. In *29th USENIX security symposium (USENIX security 2022).*
- [BZDSJ16]: Böck, H., Zauner, A., Devlin, S., Somorovsky, J., & Jovanovic, P. Nonce-Disrespecting Adversaries: Practical Forgery Attacks on GCM in TLS. In 10th USENIX Workshop on Offensive Technologies (WOOT 16).
- [RSA20]: R. Lipovsky and S. Svorencik. Kr00k: How KRACKing Amazon Echo Exposed a Billion+ Vulnerable WiFi Devices. At the RSA Conference, 2020. See also https://www.eset.com/afr/kr00k/
- [BH20]: R. Lipovsky and S. Svorencik. KrØØk: Serious Vulnerability Affected Encryption of Billion+ Wi-Fi Devices. At Black Hat USA, 2020. See also https://www.welivesecurity.com/2020/08/06/beyond-kr00k-even-more-wifi-chips-vulnerable-eavesdropping/
- [WiSec’20]: 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.
- [WiSec’18]: M. Vanhoef, N. Bhandaru, T. Derham, I. Ouzieli, and F. Piessens. Operating Channel Validation: Preventing Multi-Channel Man-in-the-Middle Attacks Against Protected Wi-Fi Networks. In 11th ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec), 2018.