Key Reinstallation Attacks: Breaking the WPA2 Protocol

Mathy Vanhoef — @vanhoefm

Black Hat Europe, 7 December 2017

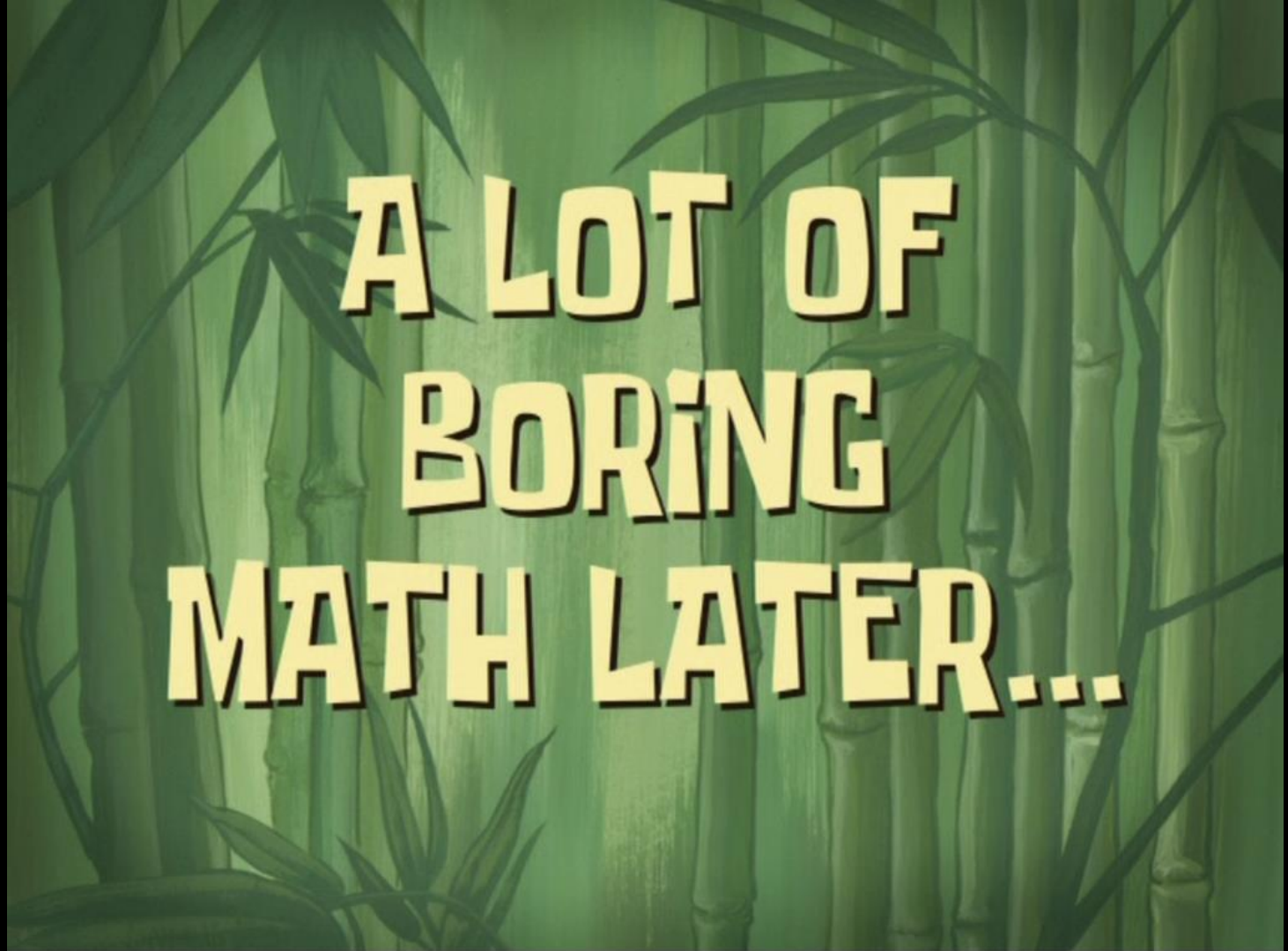
KU LEUVEN DistriNet

Introduction

PhD Defense, July 2016:

“You recommend WPA2 with AES, but are you sure that’s secure?”

Seems so! No attacks in 14 years & proven secure.

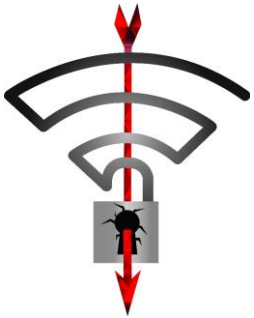

A LOT OF
BORING
MATH LATER...

Introduction

```
/* install the PTK */
if ((*ic->ic_set_key)(ic, ni, k) != 0) {
        reason = IEEE80211_REASON_AUTH_LEAVE;
        goto deauth;
}
ni->ni_flags &= ~IEEE80211_NODE_TXRXPROT;
ni->ni_flags |= IEEE80211_NODE_RXPROT;
```

Key reinstallation when ic_set_key is called again?

Overview



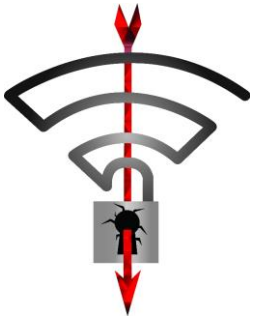
Key reinstalls in
4-way handshake

Misconceptions

Practical impact

Lessons learned

Overview



Key reinstalls in 4-way handshake

Misconceptions

Practical impact

Lessons learned

The 4-way handshake

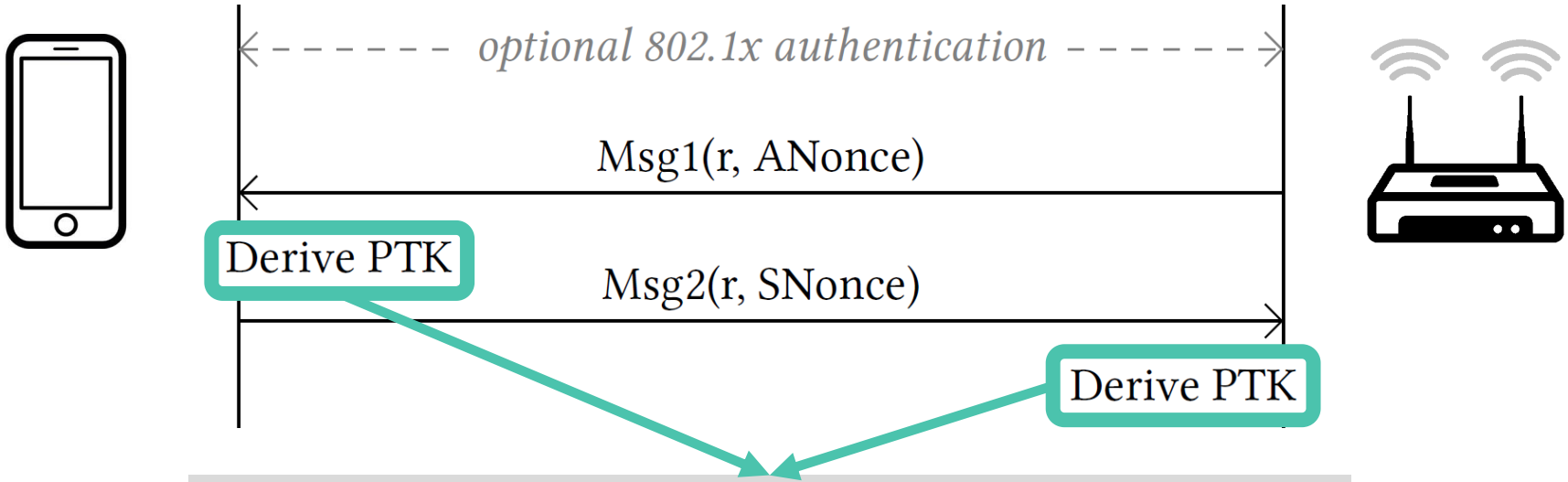
Used to connect to any protected Wi-Fi network

- Provides mutual authentication
- Negotiates fresh PTK: pairwise temporal key

Appeared to be secure:

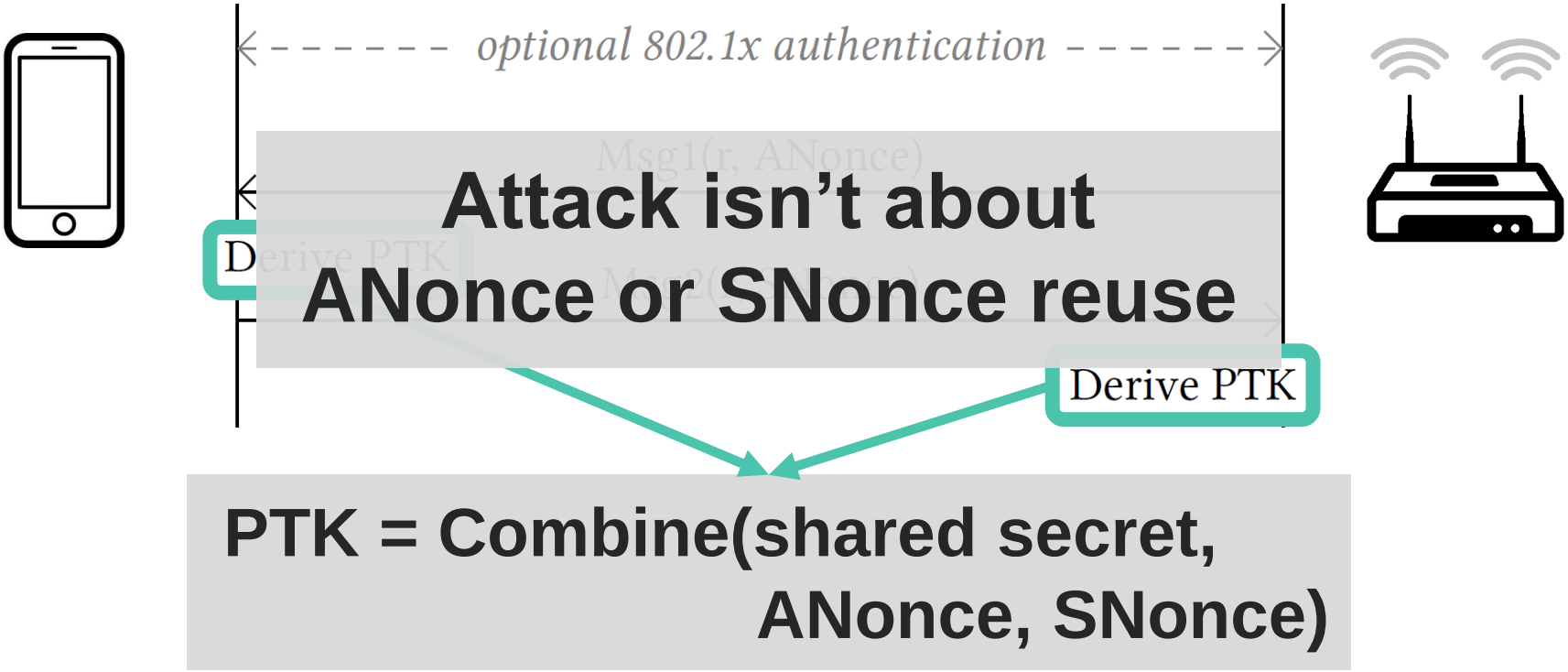
- No attacks in over a decade (apart from password guessing)
- Proven that negotiated key (PTK) is secret[1]
- And encryption protocol proven secure[7]

4-way handshake (simplified)

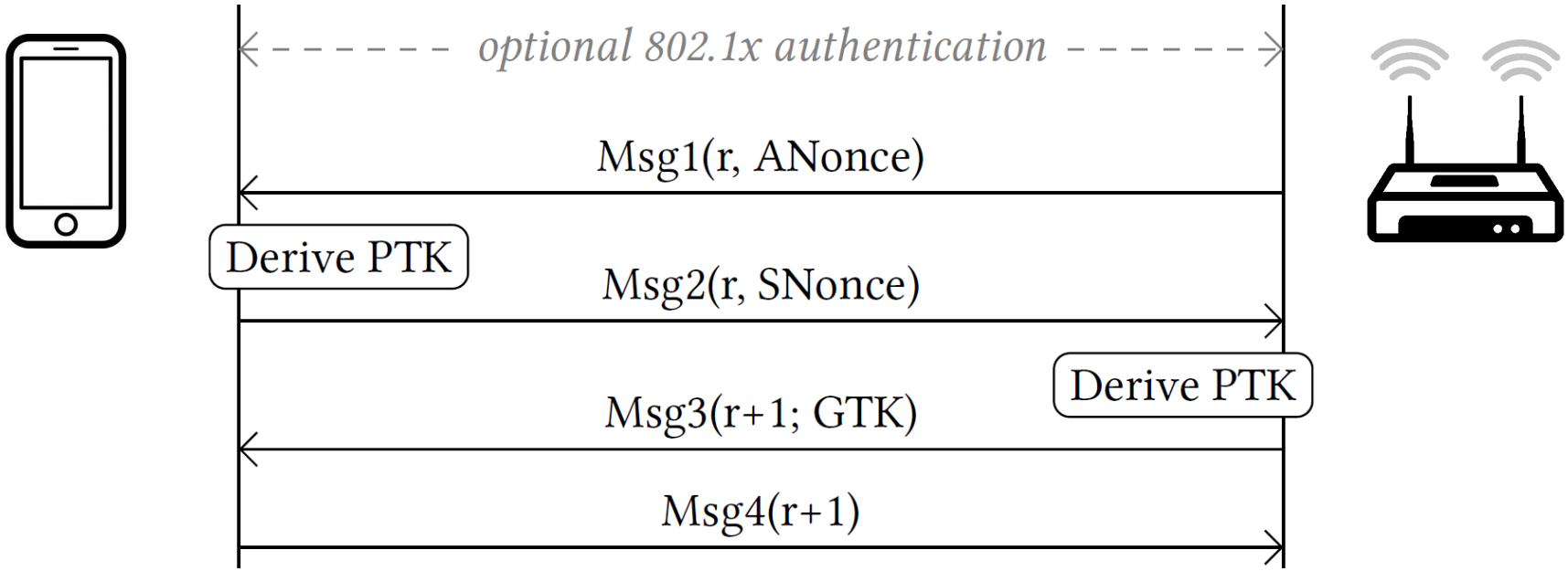


PTK = Combine(shared secret, ANonce, SNonce)

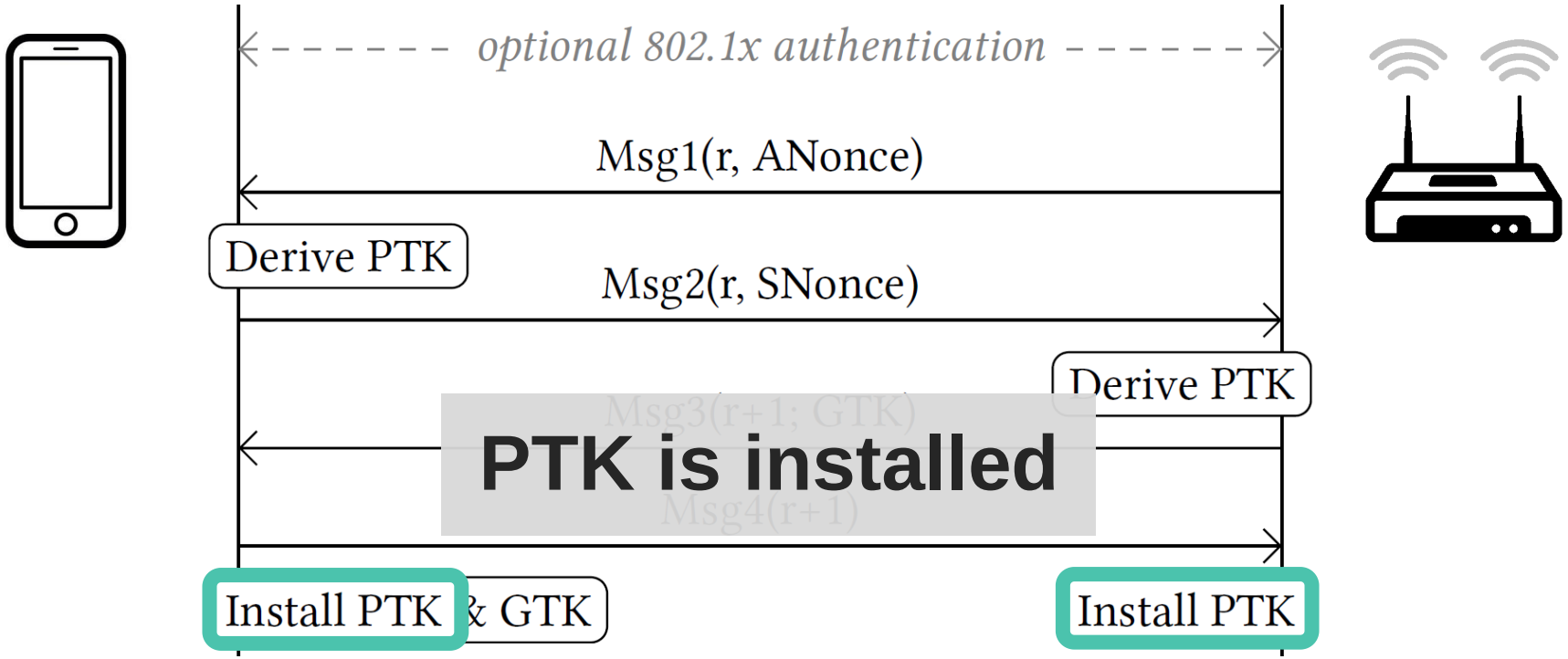
4-way handshake (simplified)



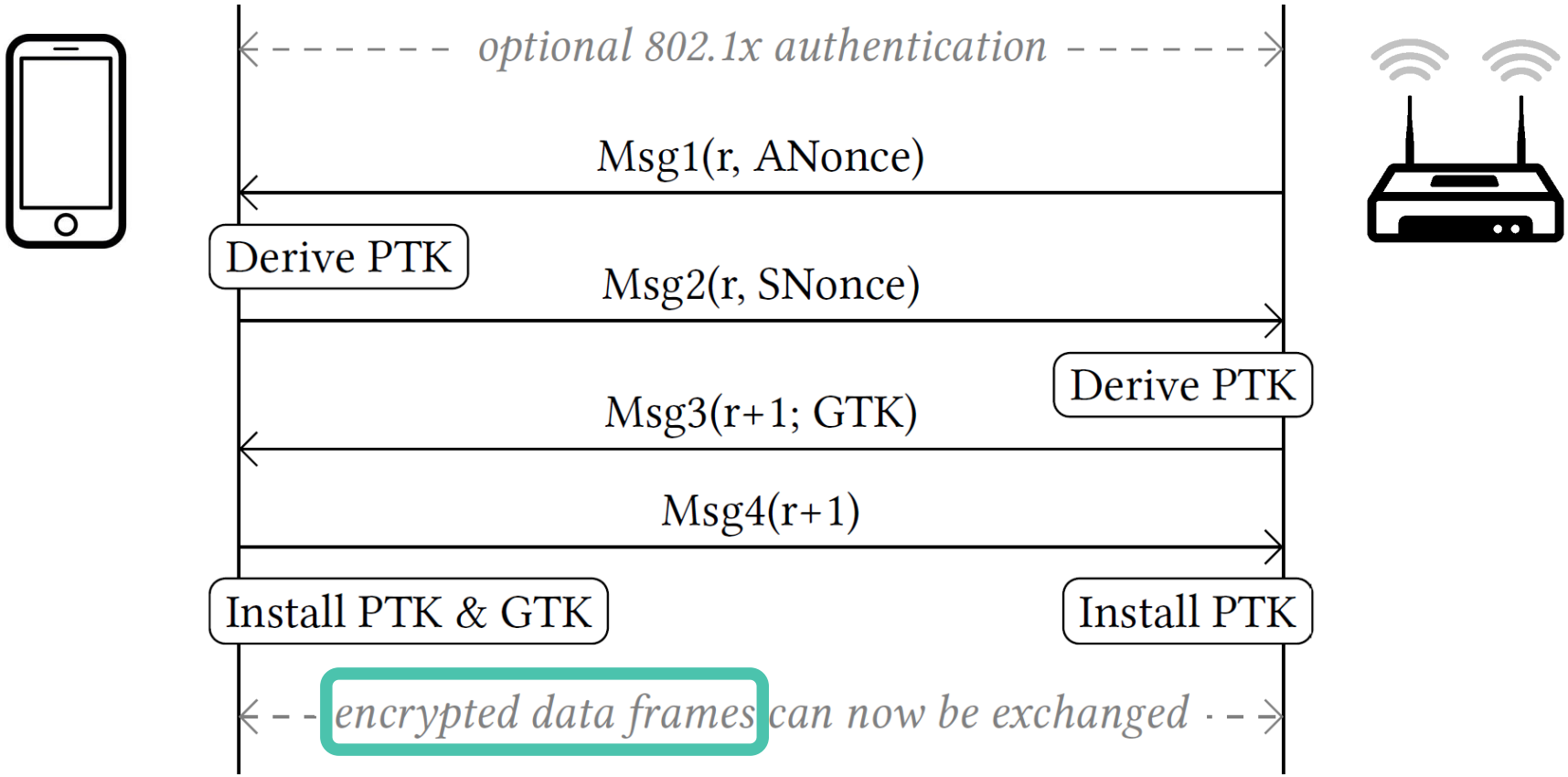
4-way handshake (simplified)



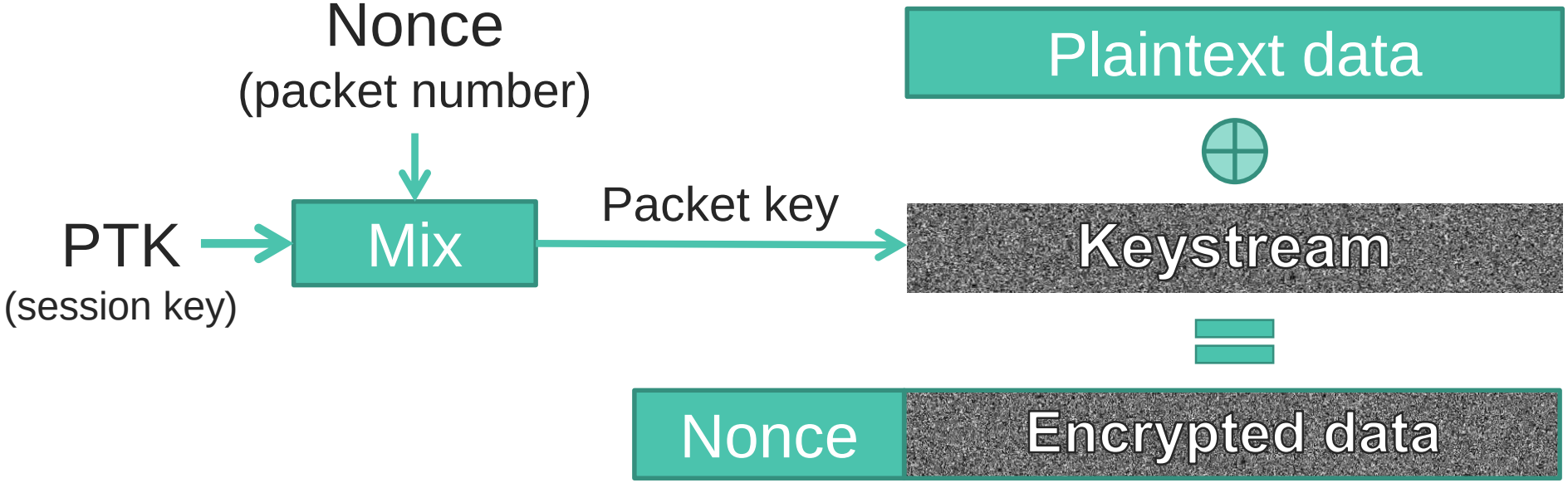
4-way handshake (simplified)



4-way handshake (simplified)

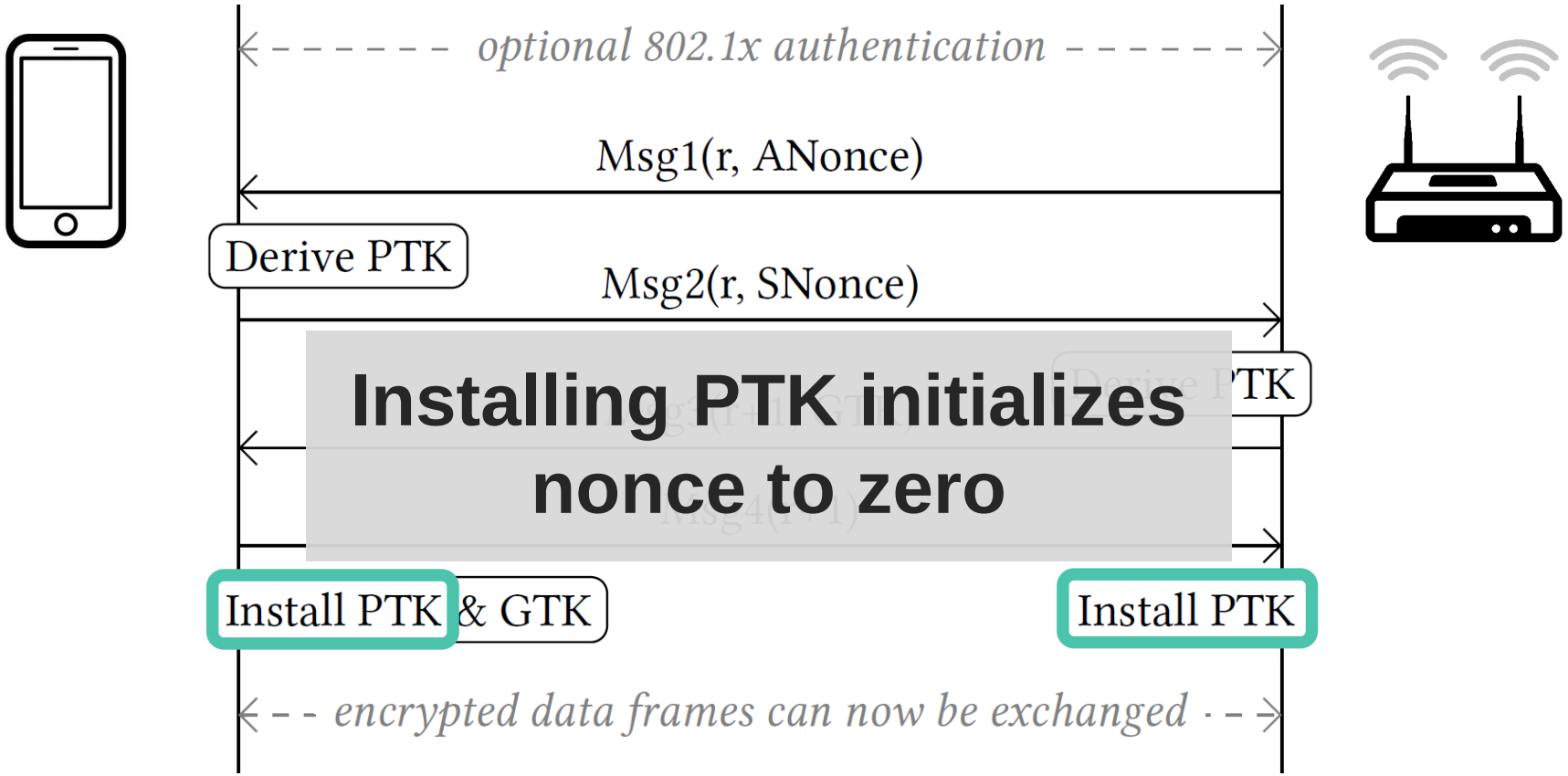


Frame encryption (simplified)

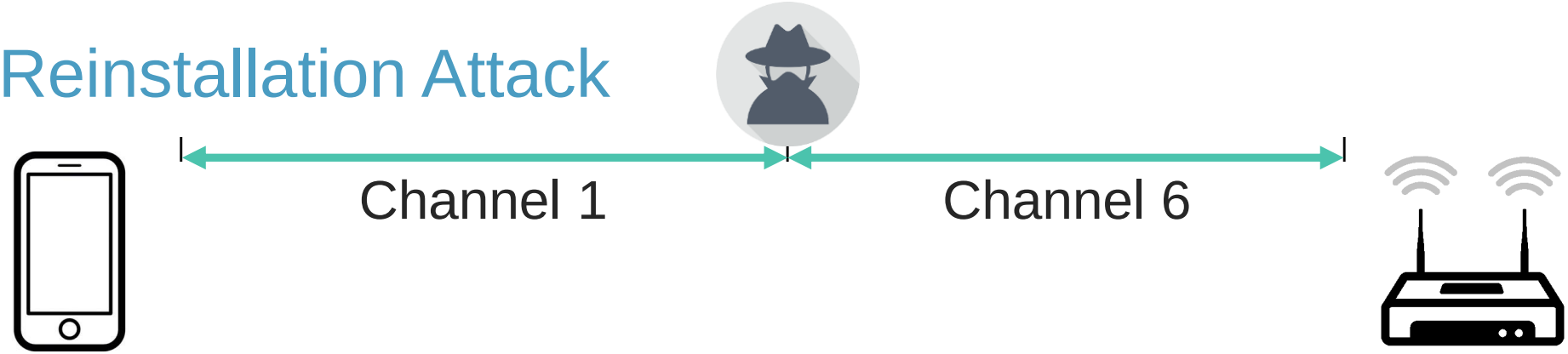


→ Nonce reuse implies keystream reuse (in all WPA2 ciphers)

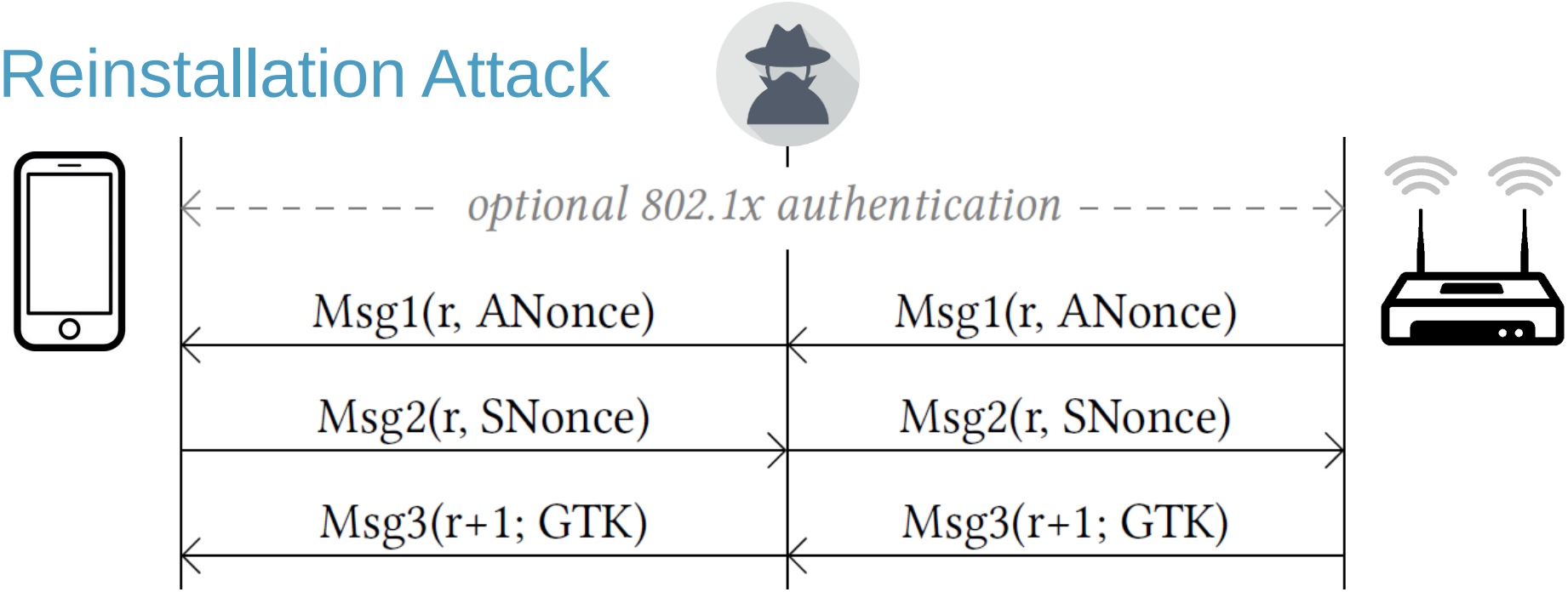
4-way handshake (simplified)



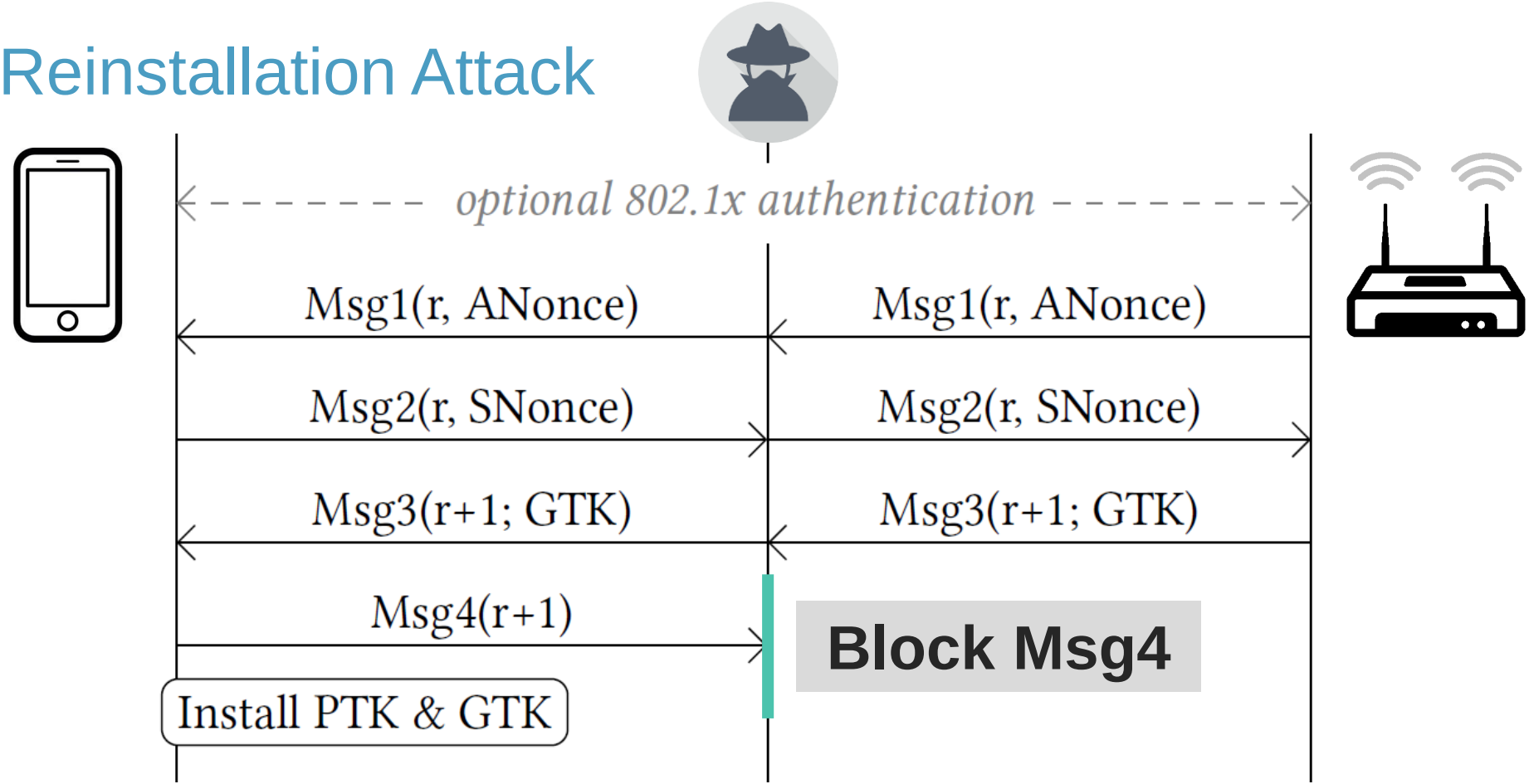
Reinstallation Attack



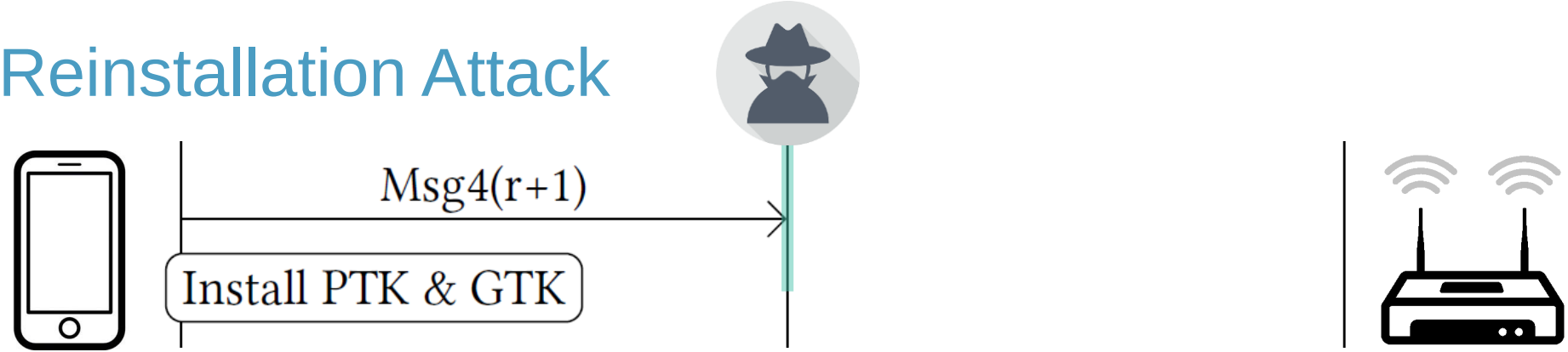
Reinstallation Attack



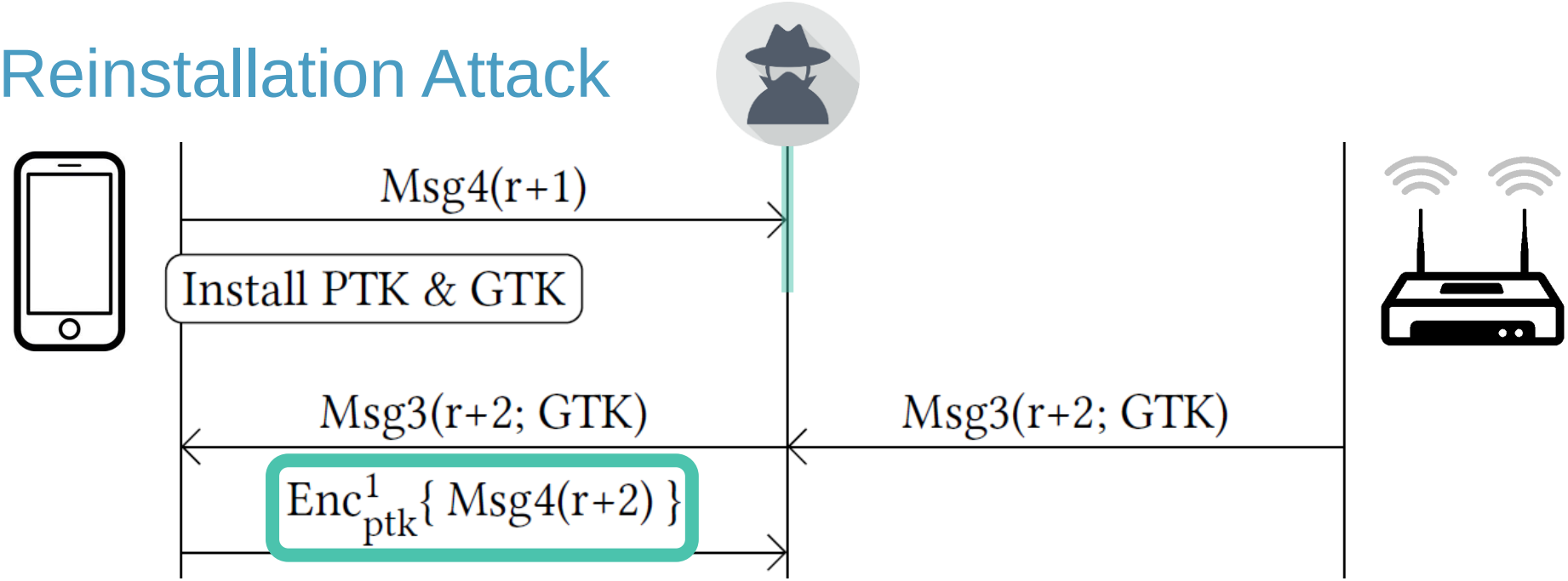
Reinstallation Attack



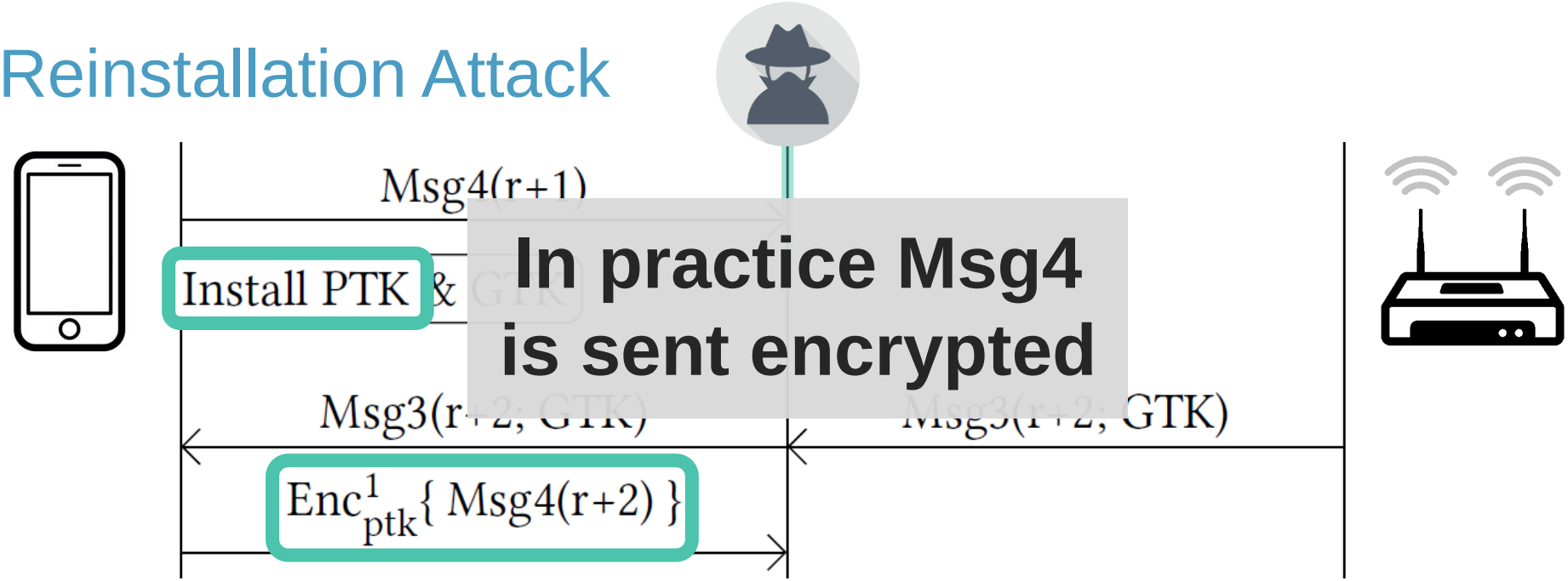
Reinstallation Attack



Reinstallation Attack

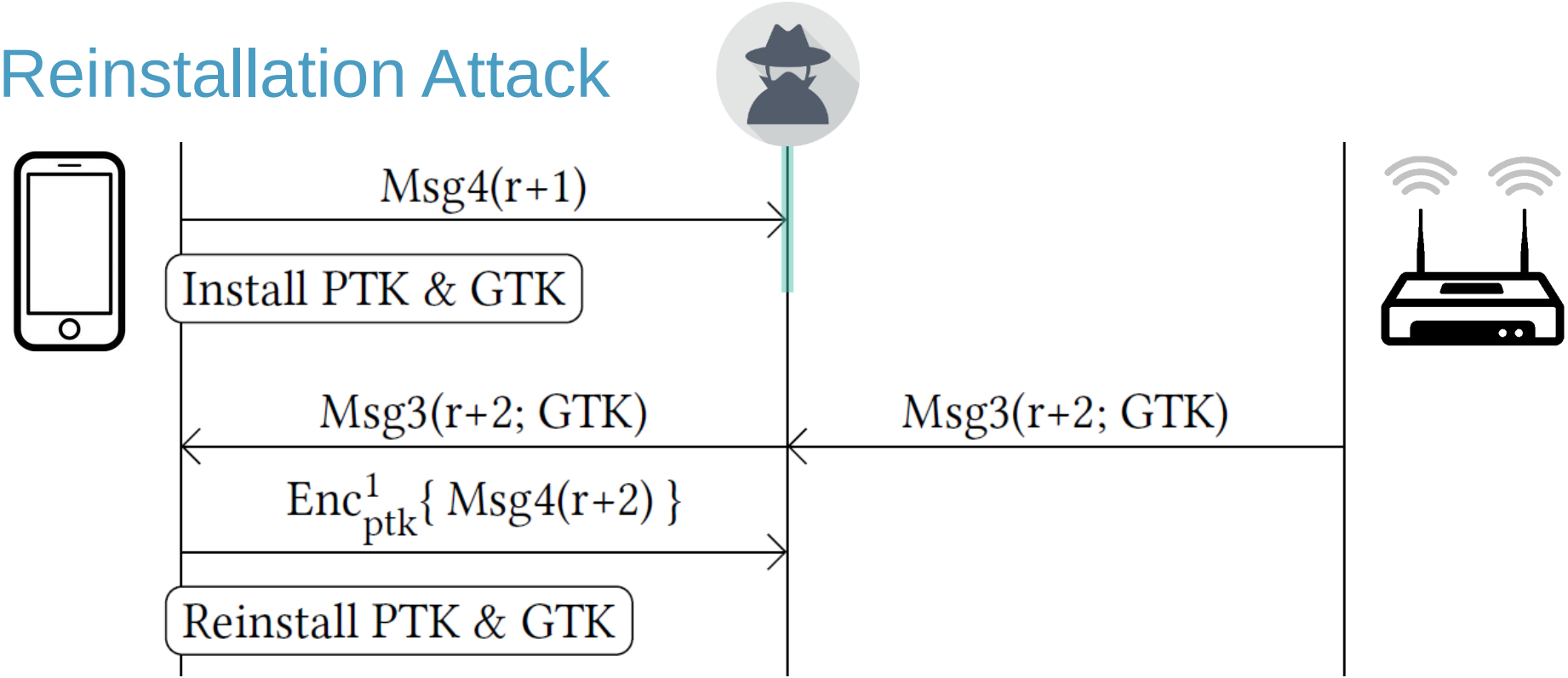


Reinstallation Attack

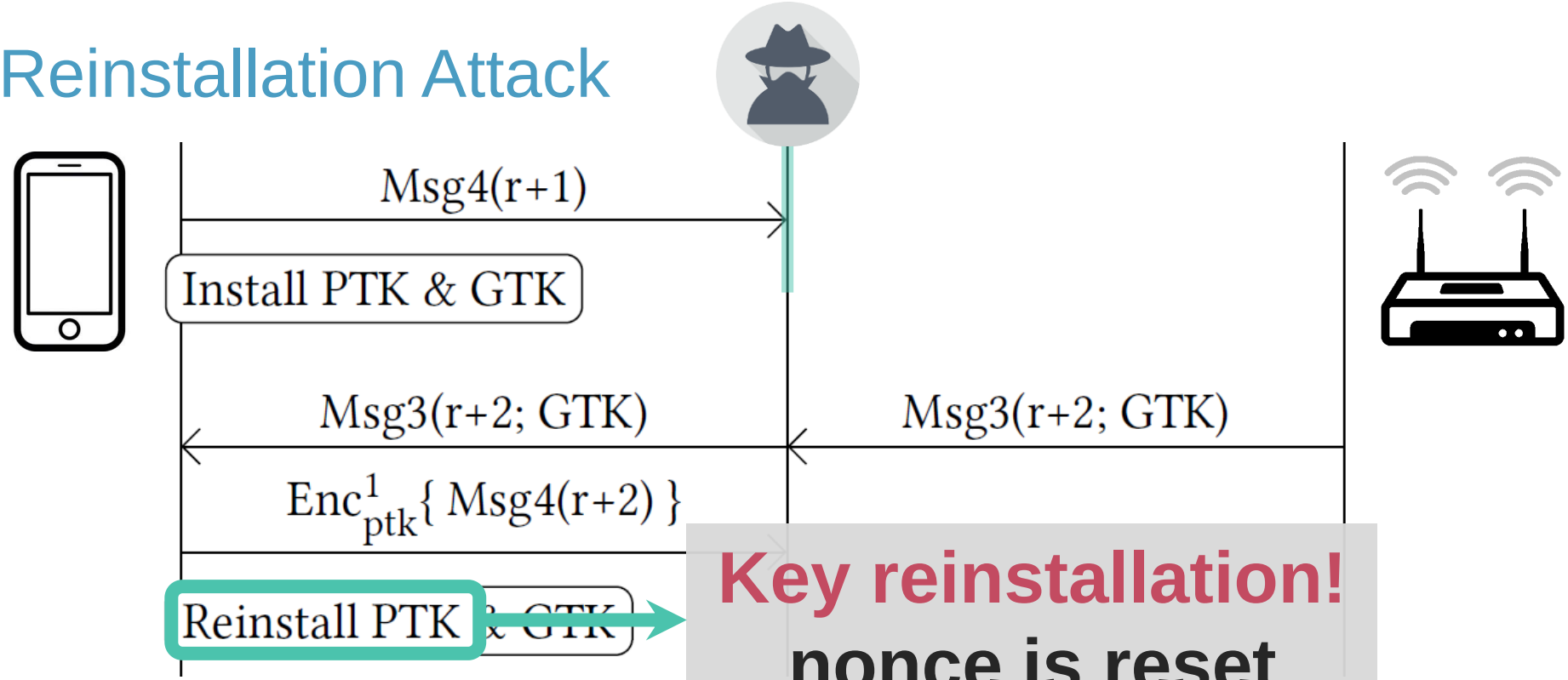


In practice Msg4 is sent encrypted

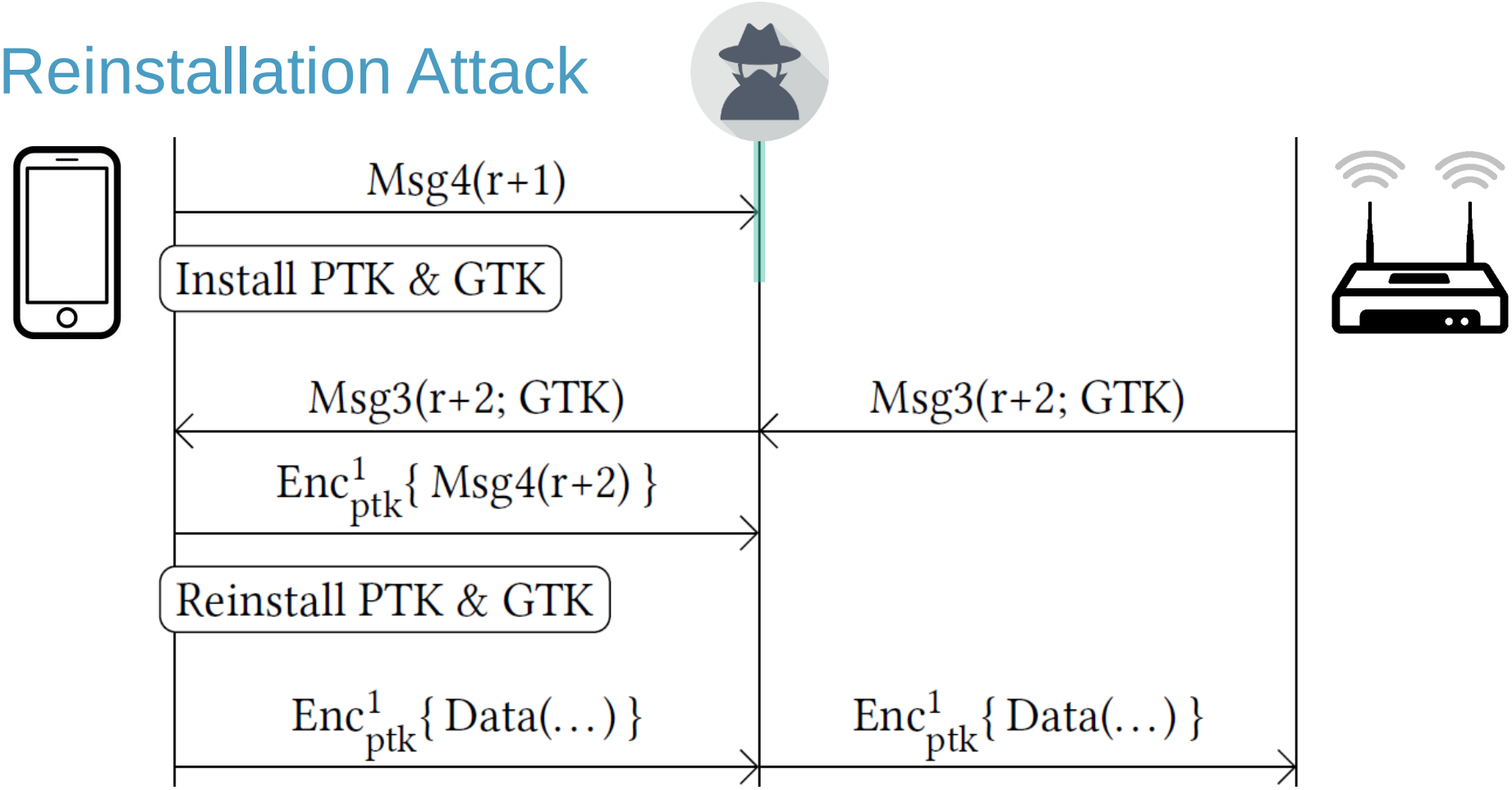
Reinstallation Attack



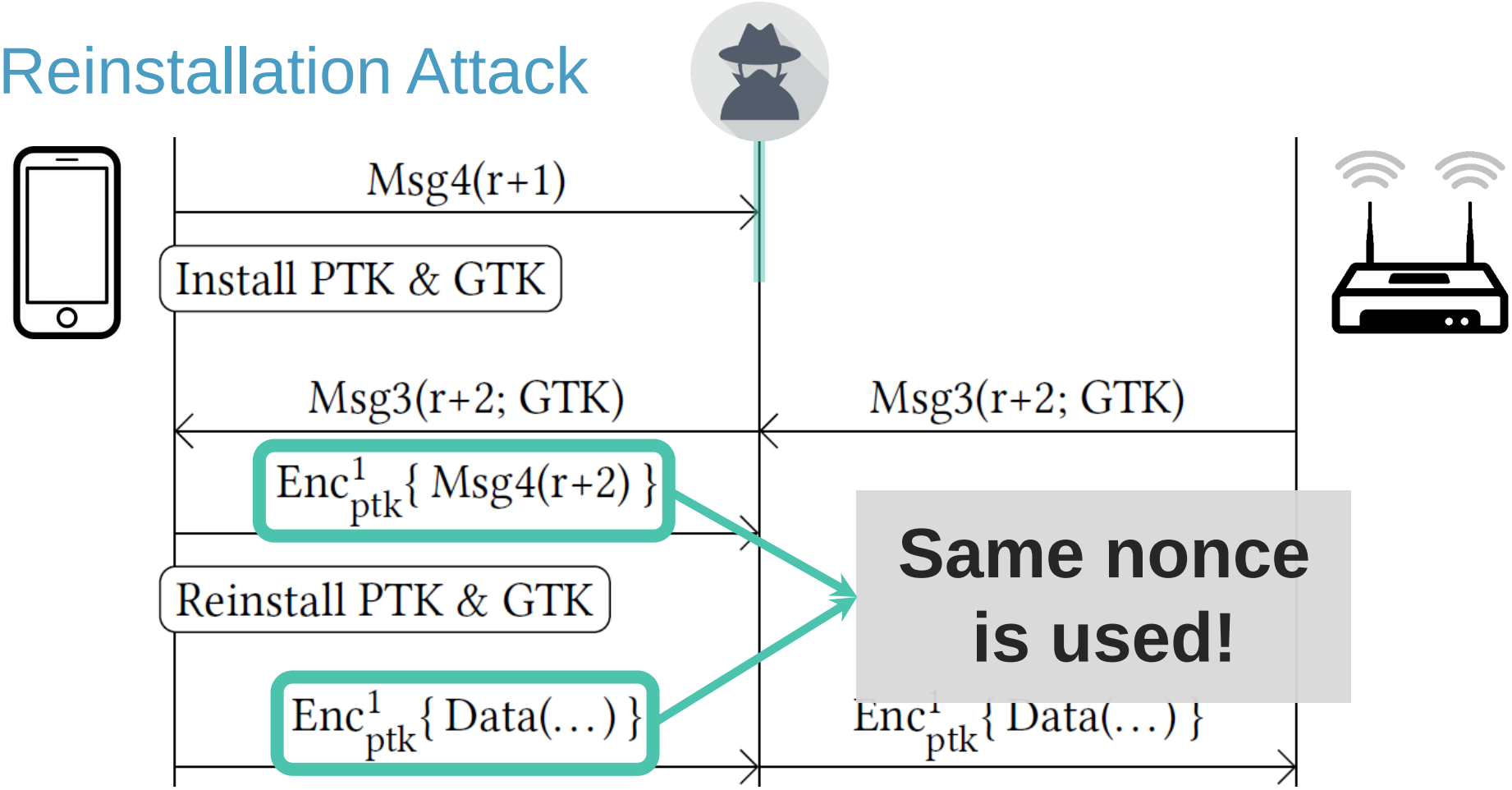
Reinstallation Attack



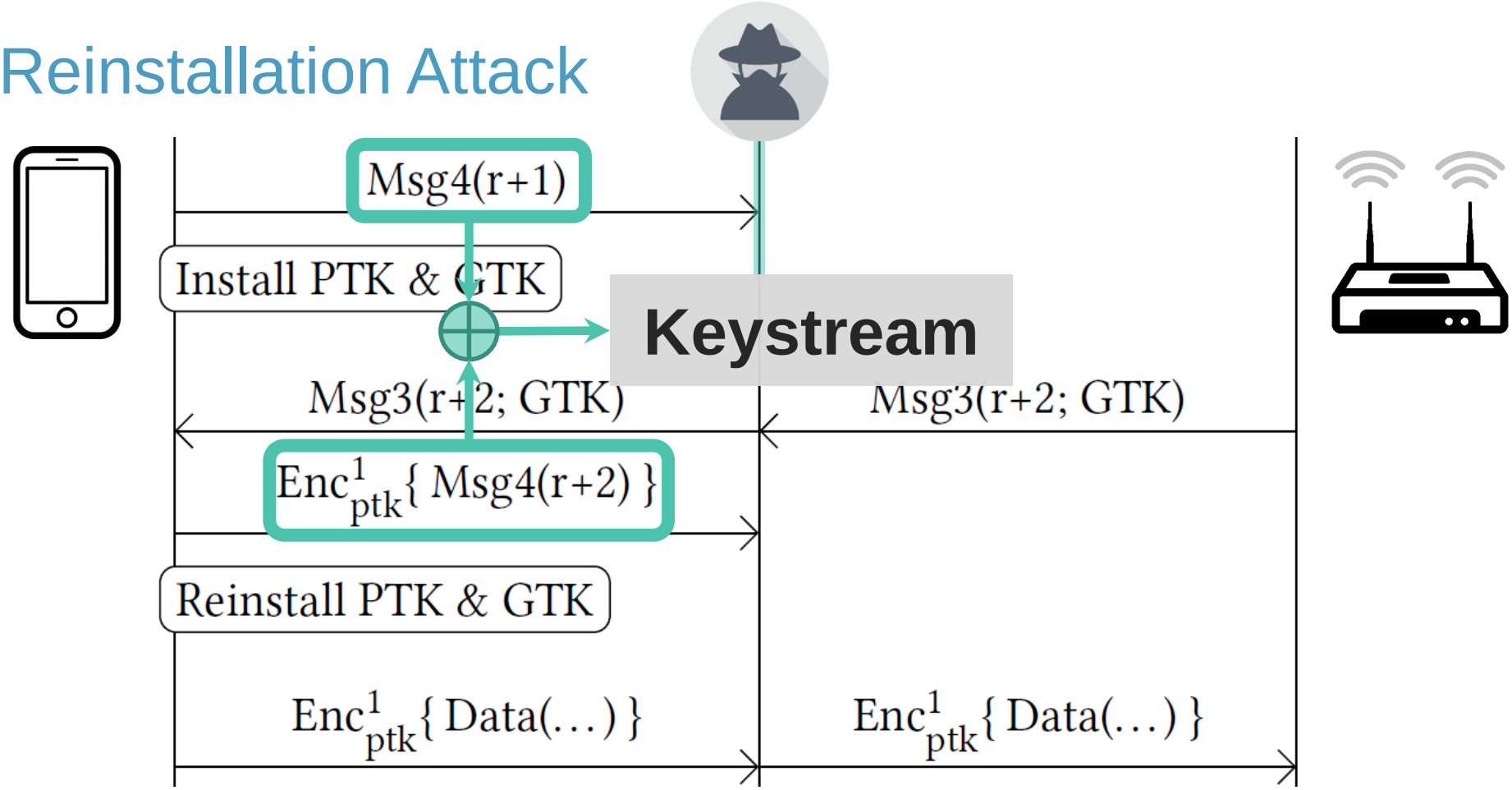
Reinstallation Attack



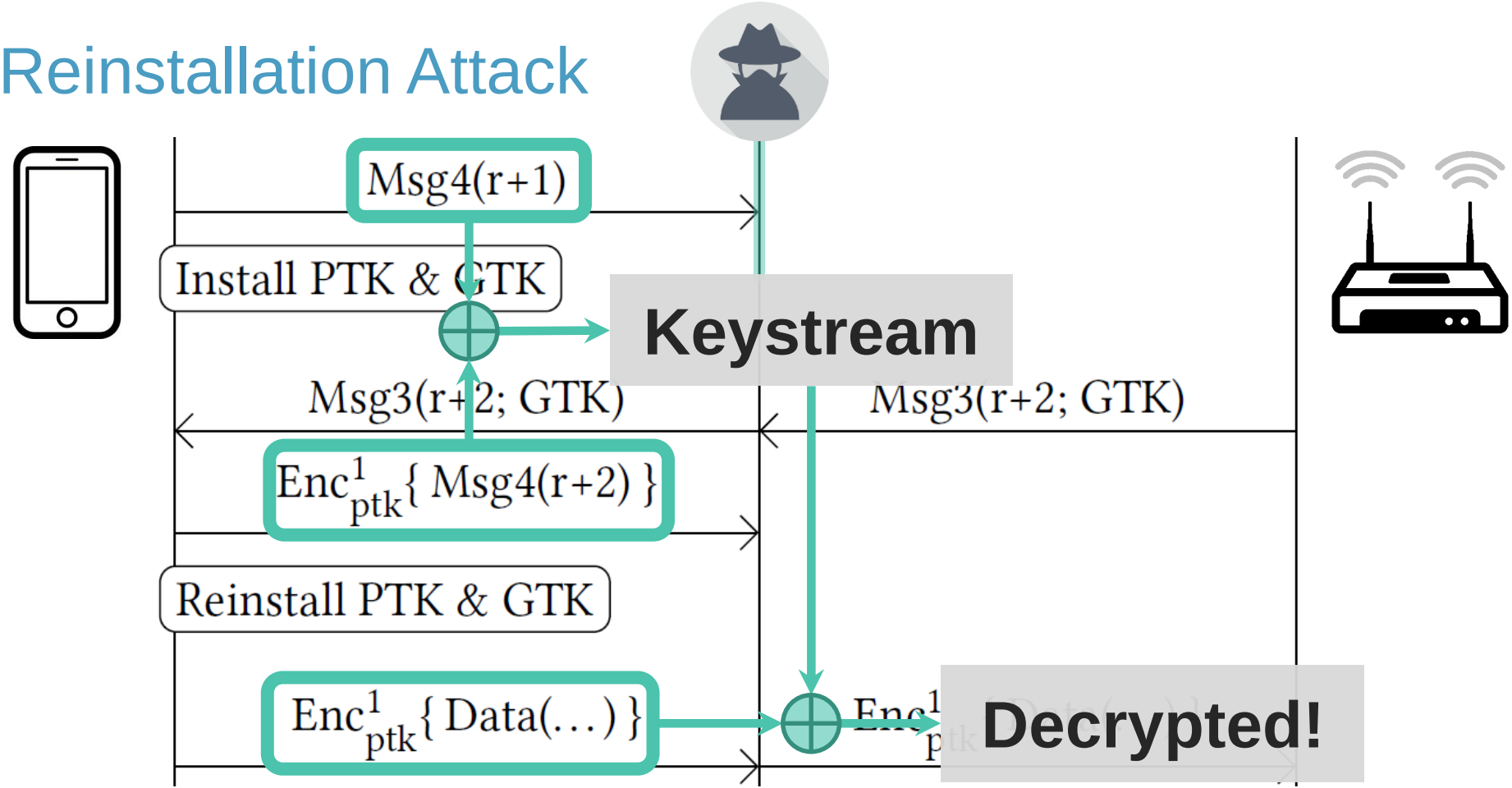
Reinstallation Attack



Reinstallation Attack



Reinstallation Attack



Key Reinstallation Attack

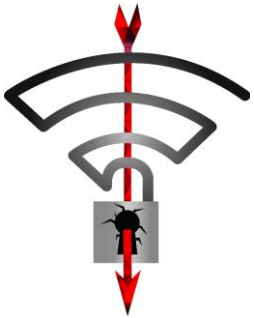
Other Wi-Fi handshakes also vulnerable:

- Group key handshake
- FT handshake
- TDLS PeerKey handshake

For details see our CCS'17 paper[12]:

- "Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2"

Overview



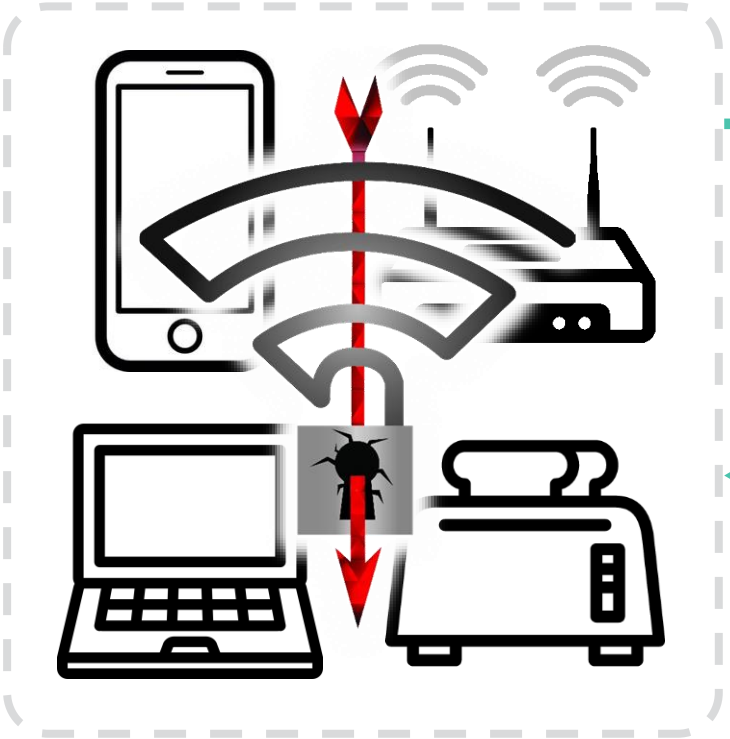
Key reinstalls in
4-way handshake

Misconceptions

Practical impact

Lessons learned

General impact



Transmit nonce reset

Decrypt frames sent by victim

Receive replay counter reset

Replay frames towards victim

Cipher suite specific

AES-CCMP: No practical frame forging attacks

WPA-TKIP:

- Recover Message Integrity Check key from plaintext[4,5]
- **Forge/inject** frames sent by the device under attack

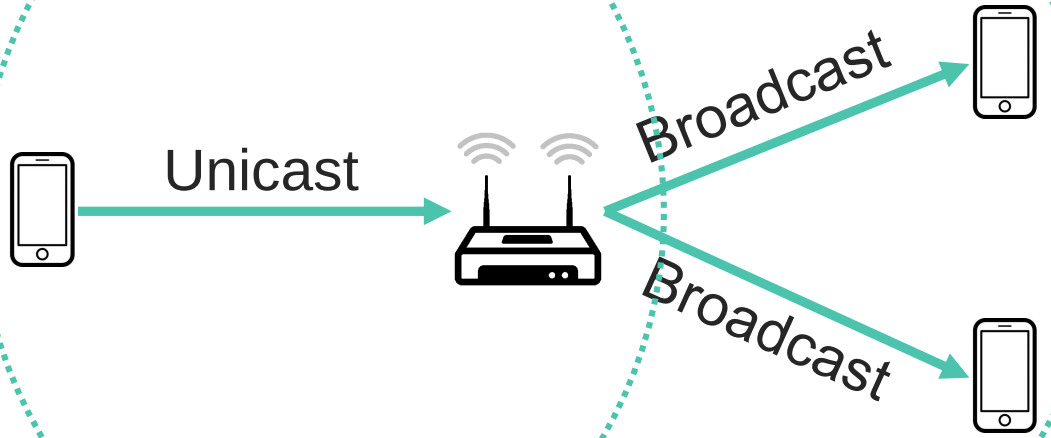
GCMP (WiGig):

- Recover GHASH authentication key from nonce reuse[6]
- **Forge/inject** frames in **both directions**

Handshake specific

Group key handshake:

› Client is attacked, but only AP sends real broadcast frames



Handshake specific

Group key handshake:

- Client is attacked, but only AP sends real broadcast frames
- Can only replay broadcast frames to client

4-way handshake: client is attacked → replay/decrypt/forge

FT handshake (fast roaming = 802.11r):

- Access Point is attacked → replay/decrypt/forge
- **No MitM required, can keep causing nonce resets**

Implementation specific

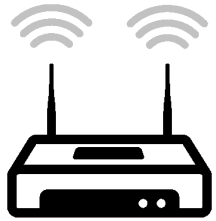
iOS 10 and Windows: 4-way handshake not affected

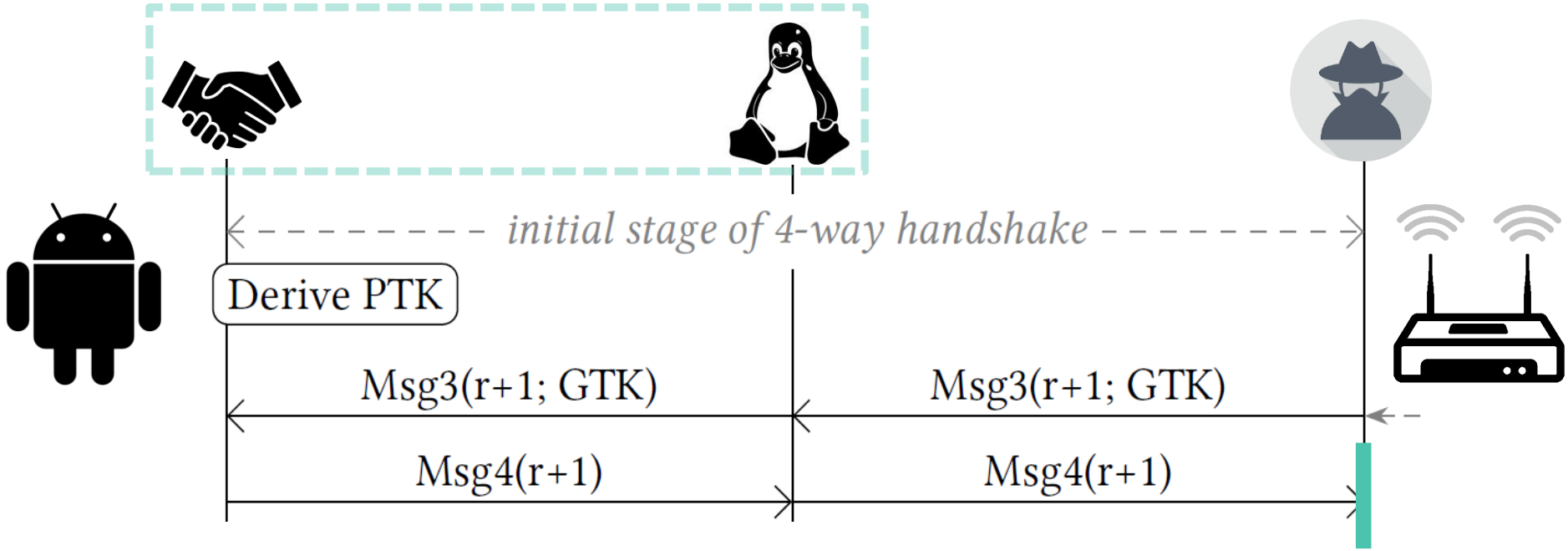
- **Cannot decrypt unicast traffic** (nor replay/decrypt)
- But group key handshake is affected (replay broadcast)
- Note: iOS 11 does have vulnerable 4-way handshake[8]

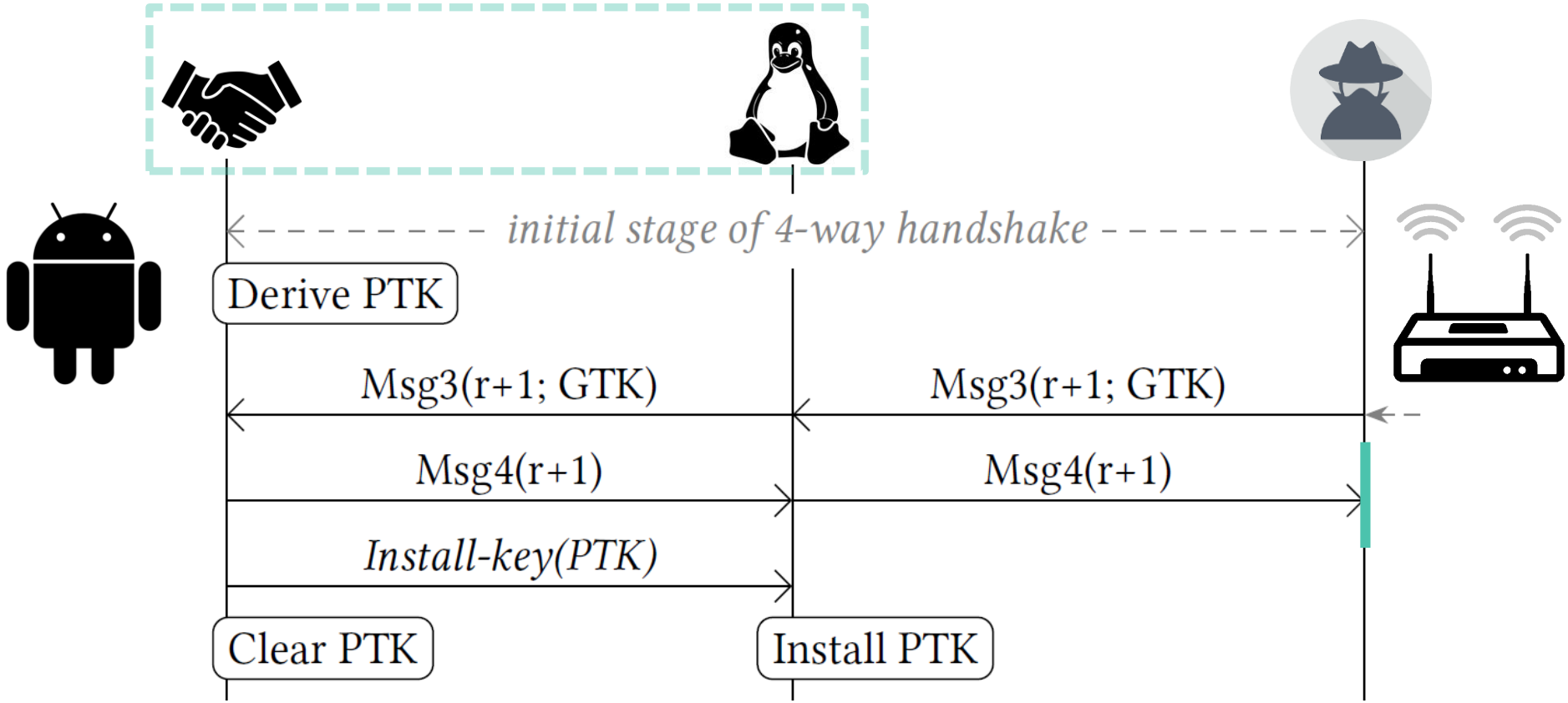
wpa_supplicant 2.4+

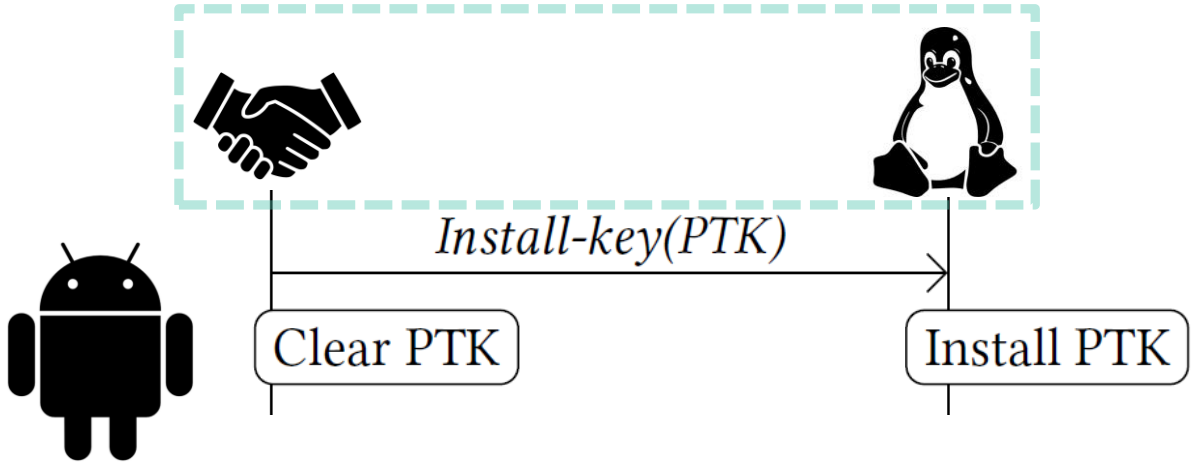
- Client used on Linux and Android 6.0+
- On retransmitted msg3 will **install all-zero key**

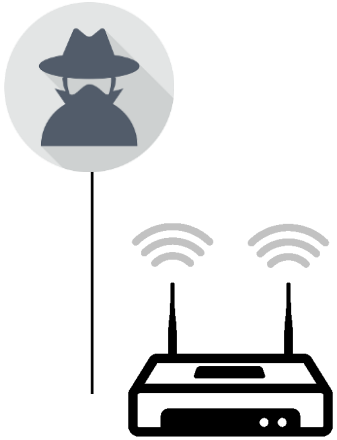
Android (victim)

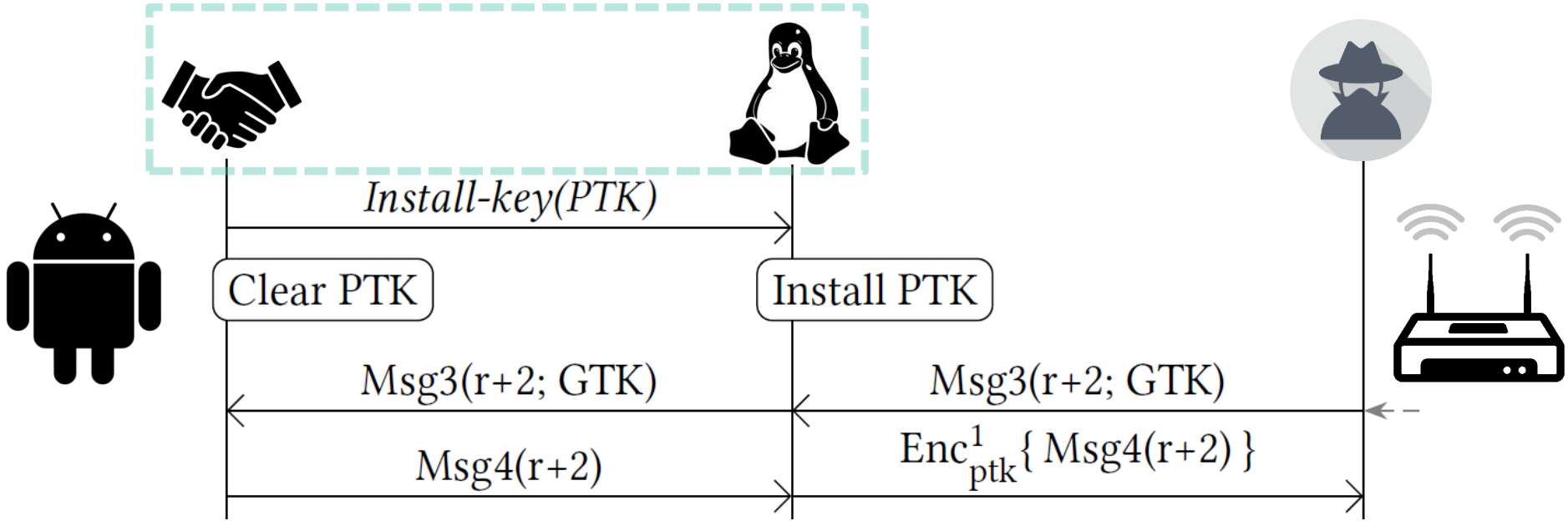


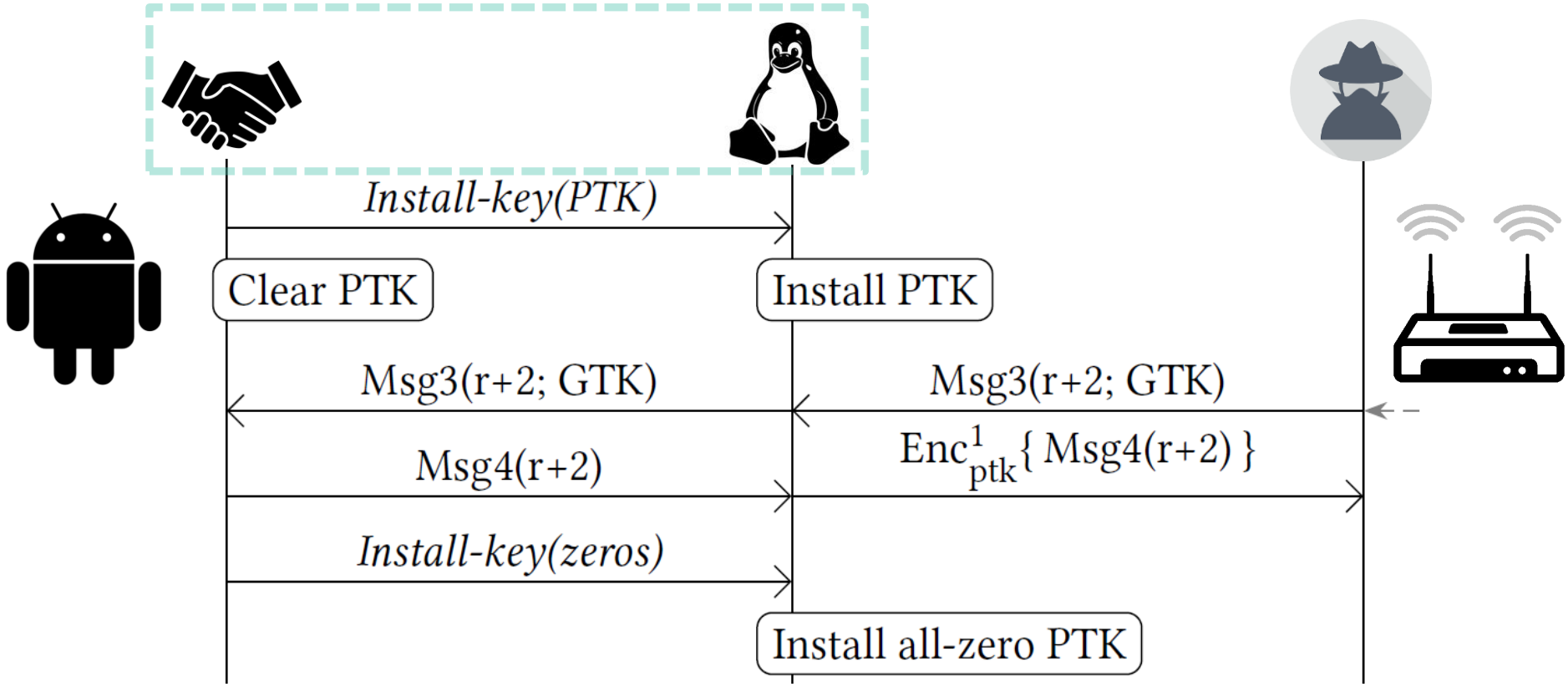

initial stage of 4-way handshake
Derive PTK
Msg3(r+1; GTK)
Msg3(r+1; GTK)
Msg4(r+1)
Msg4(r+1)

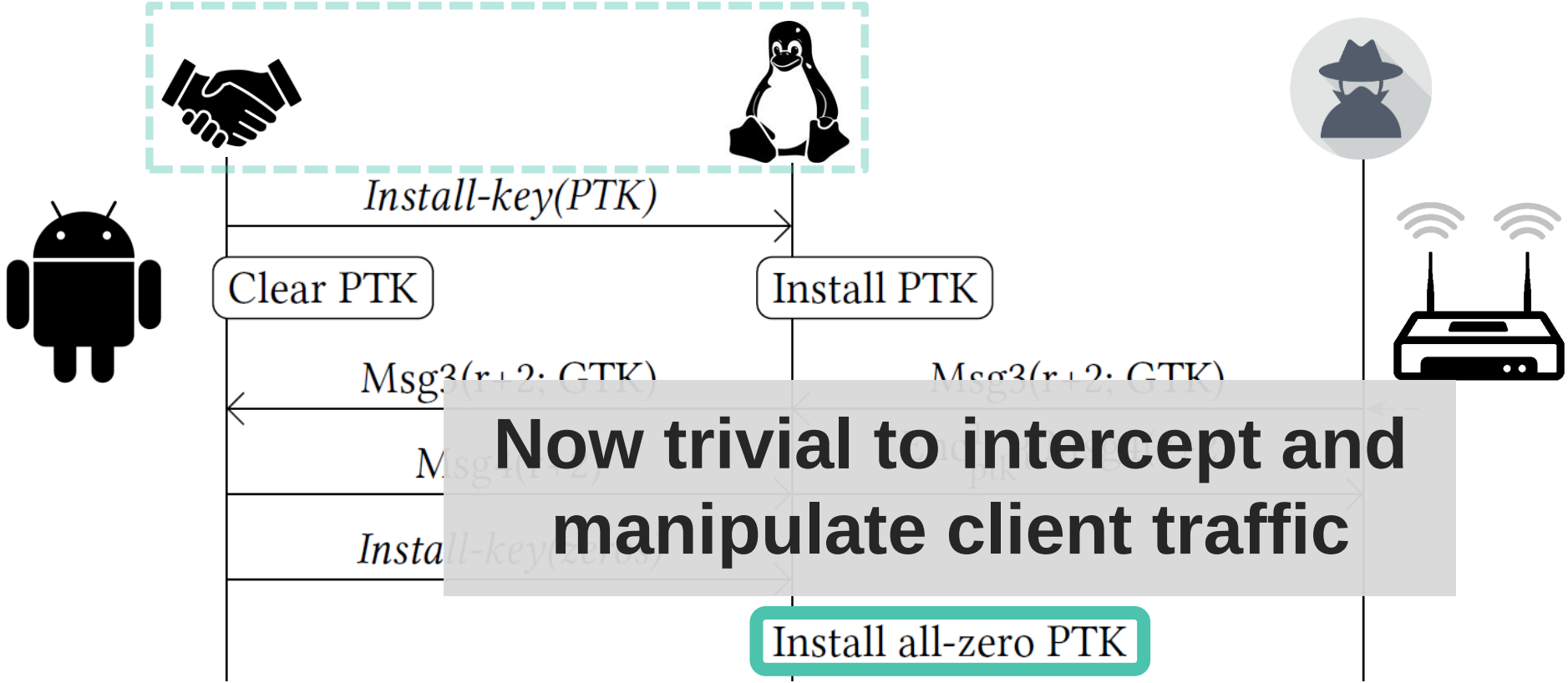

initial stage of 4-way handshake
Derive PTK
Msg3(r+1; GTK)
Msg3(r+1; GTK)
Msg4(r+1)
Msg4(r+1)
Install-key(PTK)
Clear PTK
Install PTK


Install-key(PTK)
Clear PTK
Install PTK




Install-key(PTK)
Clear PTK
Install PTK
Msg3(r+2; GTK)
Msg3(r+2; GTK)
Msg4(r+2)
Enc$^{1}_{ptk}$ { Msg4(r+2) }


Install-key(PTK)
Clear PTK
Install PTK
Msg3(r+2; GTK)
Msg3(r+2; GTK)
Msg4(r+2)
Enc$^{1}_{ptk}$ { Msg4(r+2) }
Install-key(zeros)
Install all-zero PTK



Now trivial to intercept and manipulate client traffic

Is your devices affected?

github.com/vanhoefm/krackattacks-scripts

- Tests clients and APs
- Works on Kali Linux

Remember to:

- Disable hardware encryption
- Use a supported Wi-Fi dongle!

Countermeasures

Problem: many clients won’t get updates

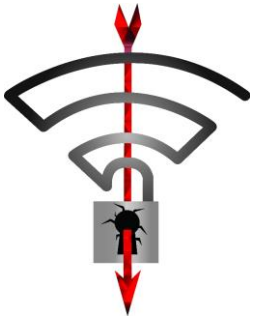
Solution: AP can prevent (most) attacks on clients!

- Don’t retransmit message 3/4
- Don’t retransmit group message 1/2

However:

- Impact on reliability unclear
- Clients still vulnerable when connected to unmodified APs

Overview



Key reinstalls in
4-way handshake

Misconceptions

Practical impact

Lessons learned

Misconceptions I

Updating only the client or AP is sufficient

› Both vulnerable clients & vulnerable APs must apply patches

Need to be close to network and victim

› Can use special antenna from afar

Must be connected to network as attacker (i.e. have password)

› Only need to be nearby victim and network

Misconceptions II

No useful data is transmitted after handshake

- Trigger new handshakes during TCP connection

Obtaining channel-based MitM is hard

- Nope, can use channel switch announcements

Attack complexity is hard

- Script only needs to be written once …
- … and some are (privately) doing this!

Misconceptions III

Using (AES-)CCMP mitigates the attack

› Still allows decryption & replay of frames

Enterprise networks (802.1x) aren't affected

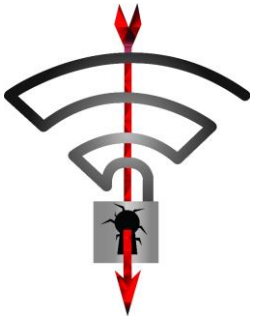
› Also use 4-way handshake & are affected

It's the end of the world!

› Let's not get carried away ☺

Image from "KRACK: Your Wi-Fi is no longer secure" by Kaspersky

Overview



Key reinstalls in
4-way handshake

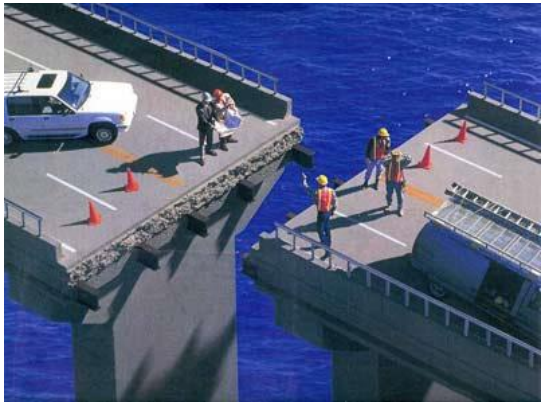
Misconceptions

Practical impact

Lessons learned

Limitations of formal proofs I

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

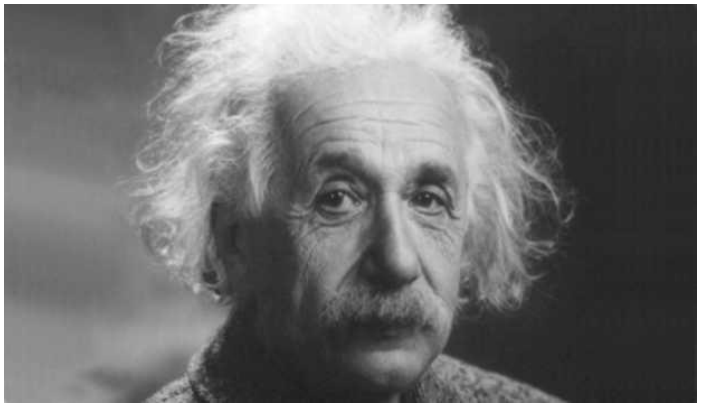


The combination was not proven secure!

Limitations of formal proofs II

Were the proofs too abstract?

- They did not model retransmissions
- **Abstract model ≠ real code**



“In theory, theory and practice are the same. In practice, they are not.”

Keep protocols simple I

The wpa_supplicant 2.6 case:

- Complex state machine & turned out to still be vulnerable
- Need **formal verification of implementations**

Discovered other vulnerabilities:

- Hostapd reuses ANonce during rekey
- $POPULAR_CLIENT reuses SNonce during rekey
- When combined, **rekeying reinstalls the existing PTK**

Keep protocols simple II

Network Operations Division Cryptographic Requirements[9]:

“**Re-keying introduces unnecessary complexity (and therefore opportunities for bugs** or other unexpected behavior) without delivering value in return.”

→ Keep the protocol and code simple!

Need rigorous specifications

Original WPA2 standard (802.11i amendment)

- State machine described in pseudo code
- Doesn't define when messages are accepted

```
StaProcessEAPOL-Key (S, M, A, I, K, RSC, ANonce, RSC, MIC, RSNE, GTK[N], IGTK[M], IPN)
    . . .
    if M = 1 then
        if Check MIC(PTK, EAPOL-Key frame) fails then
            State ← FAILED
        else
            State ← MICOK
        endif
    endif
    if K = P then
        if State ≠ FAILED then
```

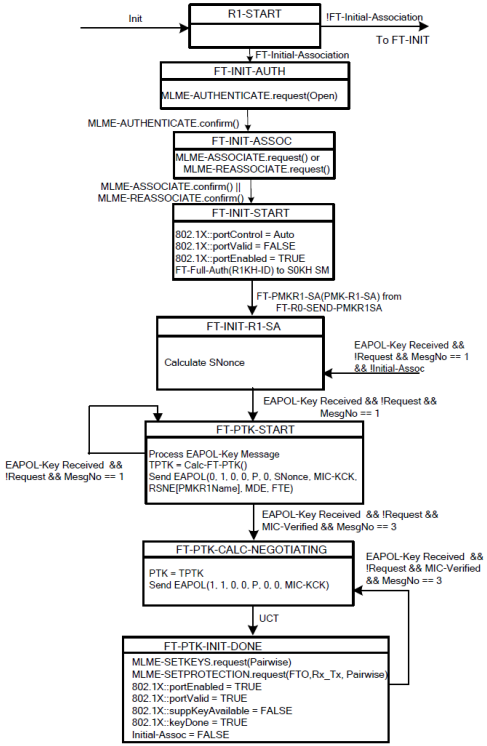
Need rigorous specifications

Original WPA2 standard (802.11i amendment)

- State machine described in pseudo code
- Doesn't define when messages are accepted

802.11r amendment (FT handshake)

- Better defines how/when to handle messages
- But **some terms and cases still unclear**



S1KH state machine

On a related note…

Workshop on:

Security Protocol Implementations: Development and Analysis (SPIDA)

Co-located with EuroS&P 2018

“focuses on improving development & analysis of security protocol implementations”

Thank you!

Questions?

krackattacks.com

References

1. C. He, M. Sundararajan, A. Datta, A. Derek, and J. Mitchell. A Modular Correctness Proof of IEEE 802.11i and TLS. In CCS, 2005.
2. S. Antakis, M. van Cuijk, and J. Stemmer. Wardriving - Building A Yagi Pringles Antenna. 2008.
3. M. Parkinson. Designer Cantenna. 2012. Retrieved 23 October 2017 from https://www.mattparkinson.eu/designer-cantenna/
4. E. and M. Beck. Practical attacks against WEP and WPA. In WiSec, 2009.
5. M. Vanhoef and F. Piessens. Practical verification of WPA-TKIP vulnerabilities. In ASIA CCS, 2013.
6. A. Joux. Authentication failures in NIST version of GCM. 2016.
7. J. Jonsson. On the security of CTR+ CBC-MAC. In SAC, 2002.
8. Apple. About the security content of iOS 11.1. November 3, 2017. Retrieved 26 November from https://support.apple.com/en-us/HT208222
9. US Central Intelligence Agency. Network Operations Division Cryptographic Requirements. Retrieved 5 December 2017 from https://wikileaks.org/ciav7p1/cms/files/NOD%20Cryptographic%20Requirements%20v1.1%20TOP%20SECRET.pdf
10. J. Salowey and E. Rescorla. TLS Renegotiation Vulnerability. Retrieved 5 December 2017 from https://www.ietf.org/proceedings/76/slides/tls-7.pdf
11. Bhargavan et al. Triple Handshakes and Cookie Cutters: Breaking and Fixing Authentication over TLS. In IEEE S&P, 2014.
12. M. Vanhoef and F. Piessens. Key Reinstallation Attacks: Forcing Nonce Reuse in WPA2. In CCS, 2017.