FragAttacks:

Aggregation & Fragmentation Flaws in Wi-Fi

WAC4 ’21, 15 August 2021 (co-located with CRYPTO)

Mathy Vanhoef

NEW YORK UNIVERSITY

Advancements in Wi-Fi security

- WPA3 is continously being updated
  - Preventing recent Dragonblood [VR20] attack
  - Securing hotspots using asymmetric crypto

Advancements in Wi-Fi security

- WPA3 is continously being updated
  - Preventing recent Dragonblood [VR20] attack
  - Securing hotspots using asymmetric crypto
- Operating channel validation [VBDOP18]
- Beacon protection [VAP20]
- KRACK patches proven secure [CKM20]

Despite these major advacements,
found **flaws in all networks** (incl. WPA2/3)

Design flaws

Implementation Flaws

Design flaws

Implementation Flaws

Aggregation

Mixed key

Fragment cache

Implementation Flaws

Background

Sending small frames causes high overhead:

This can be avoided by **aggregating frames**:

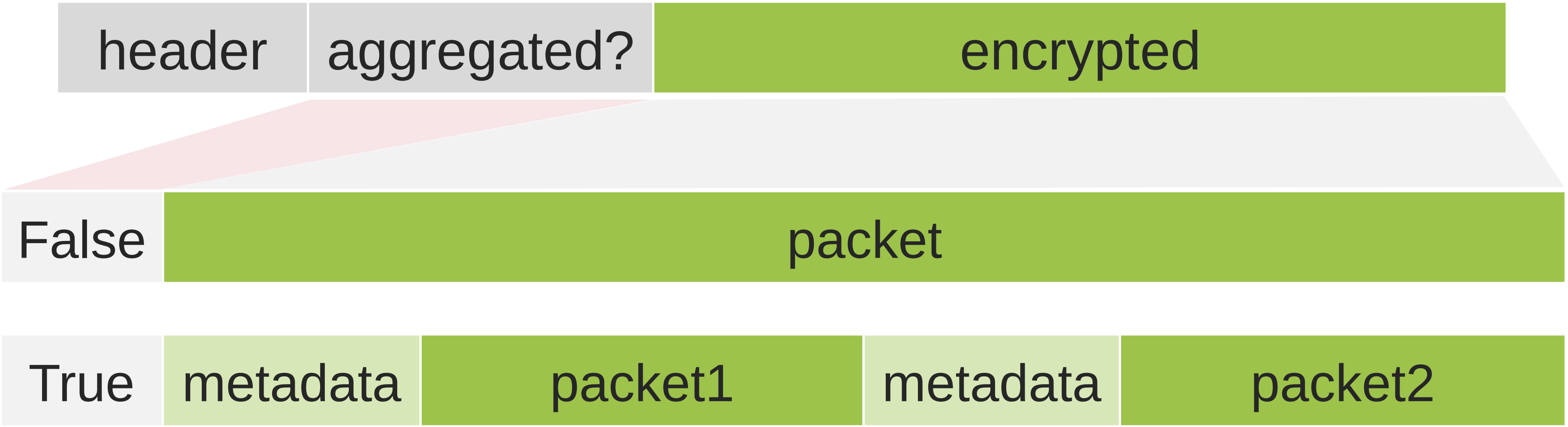
Background

Sending small frames causes high overhead:

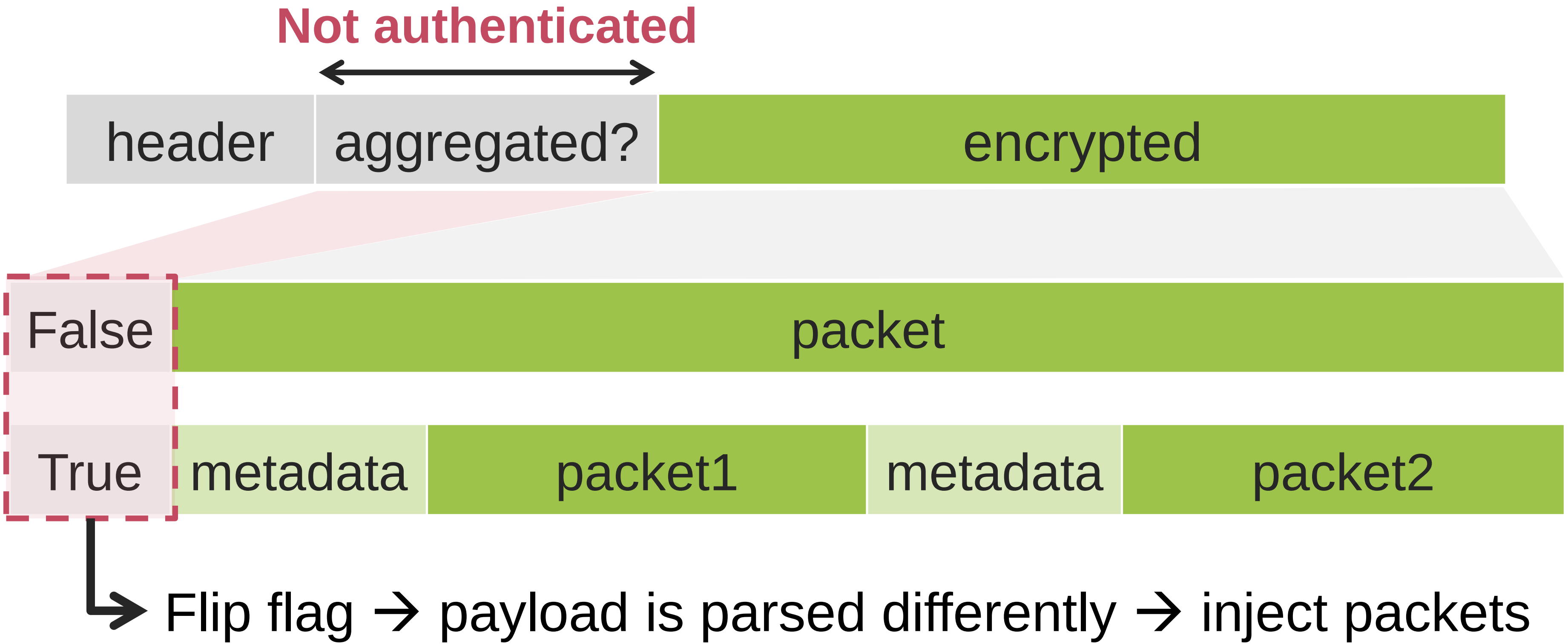
This can be avoided by **aggregating frames**:

Problem: how to recognize aggregated frames?

Aggregation design flaw

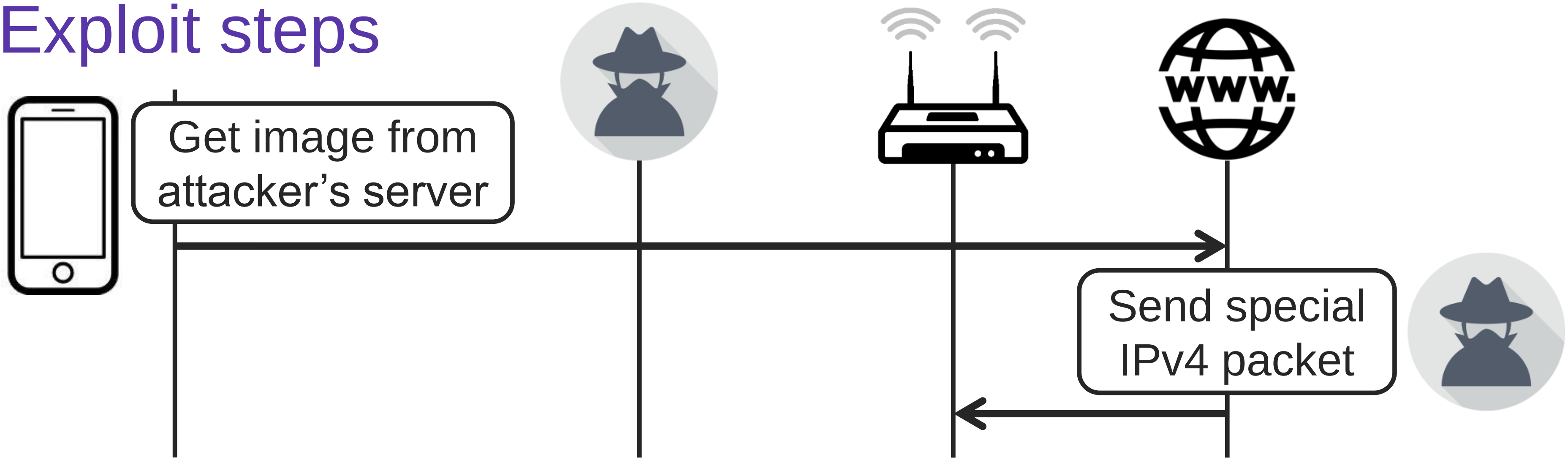


Aggregation design flaw

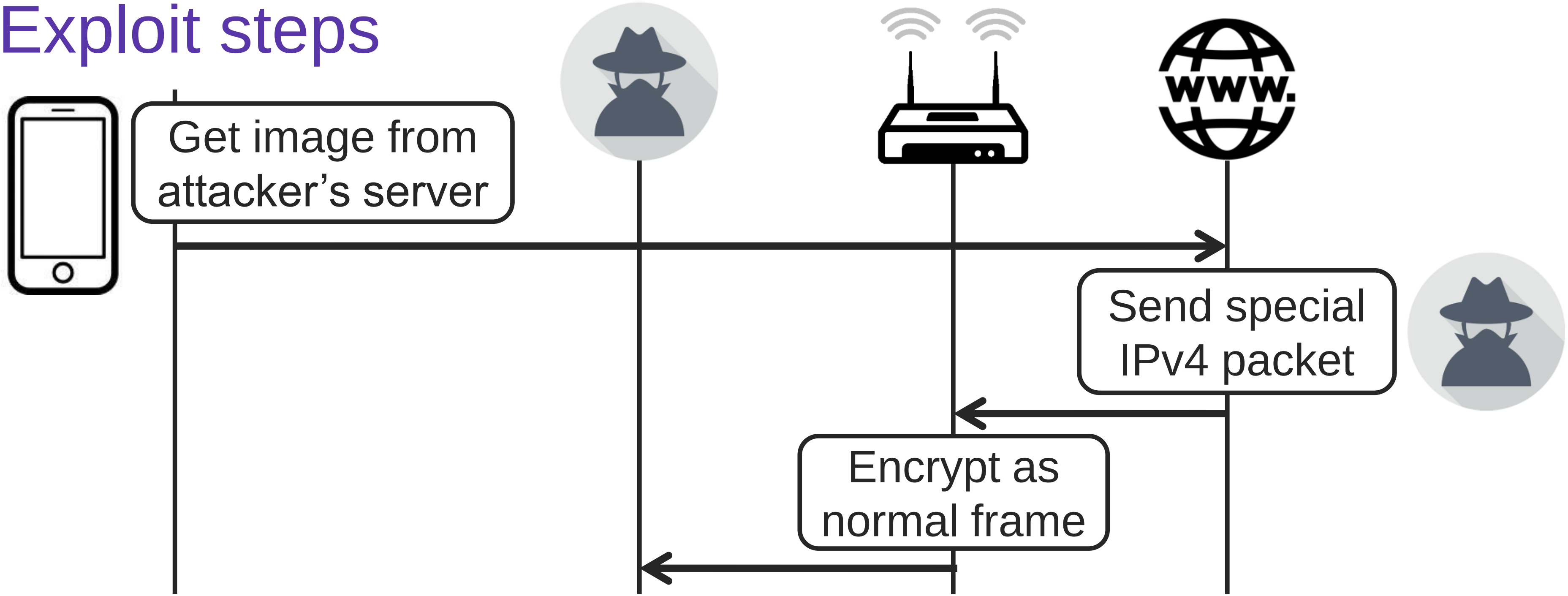


Flip flag → payload is parsed differently → inject packets

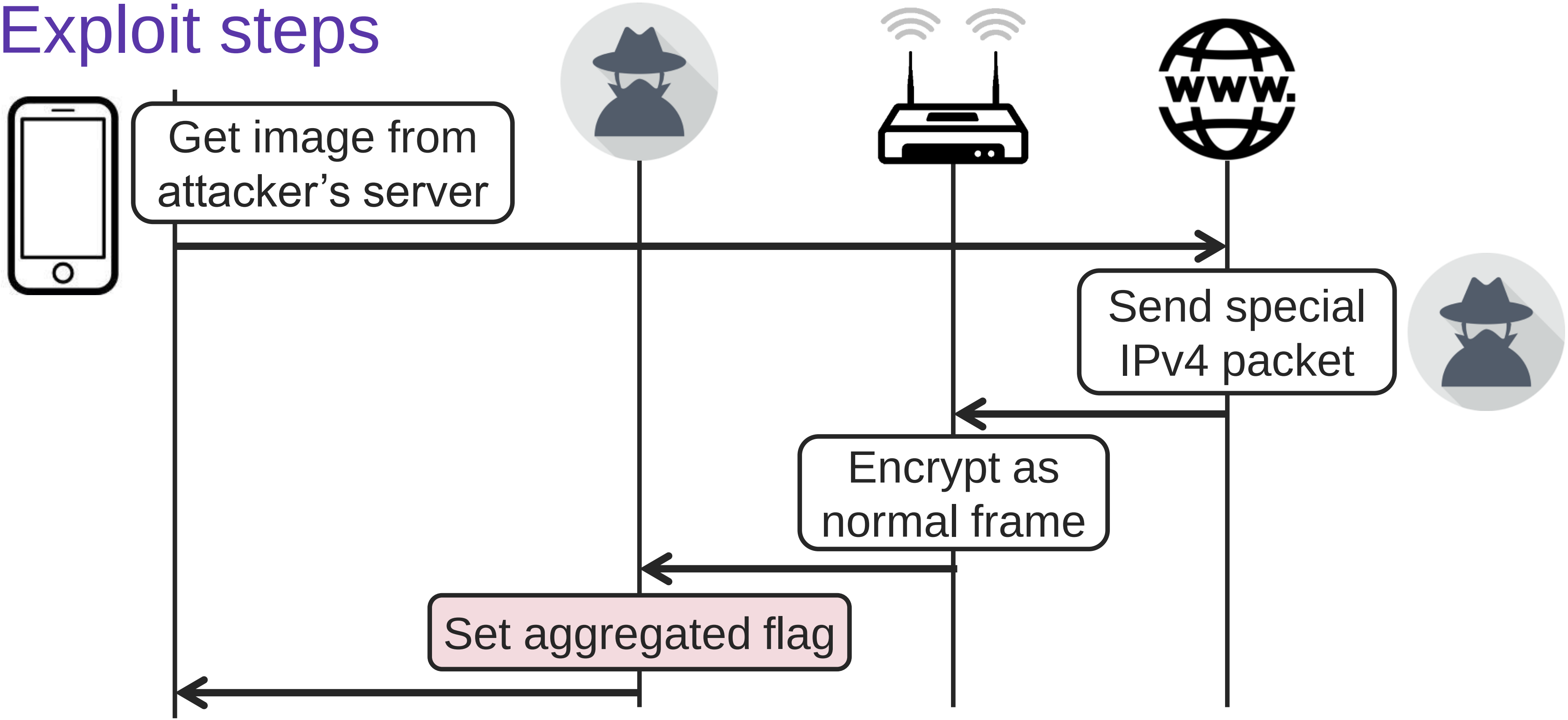
Exploit steps



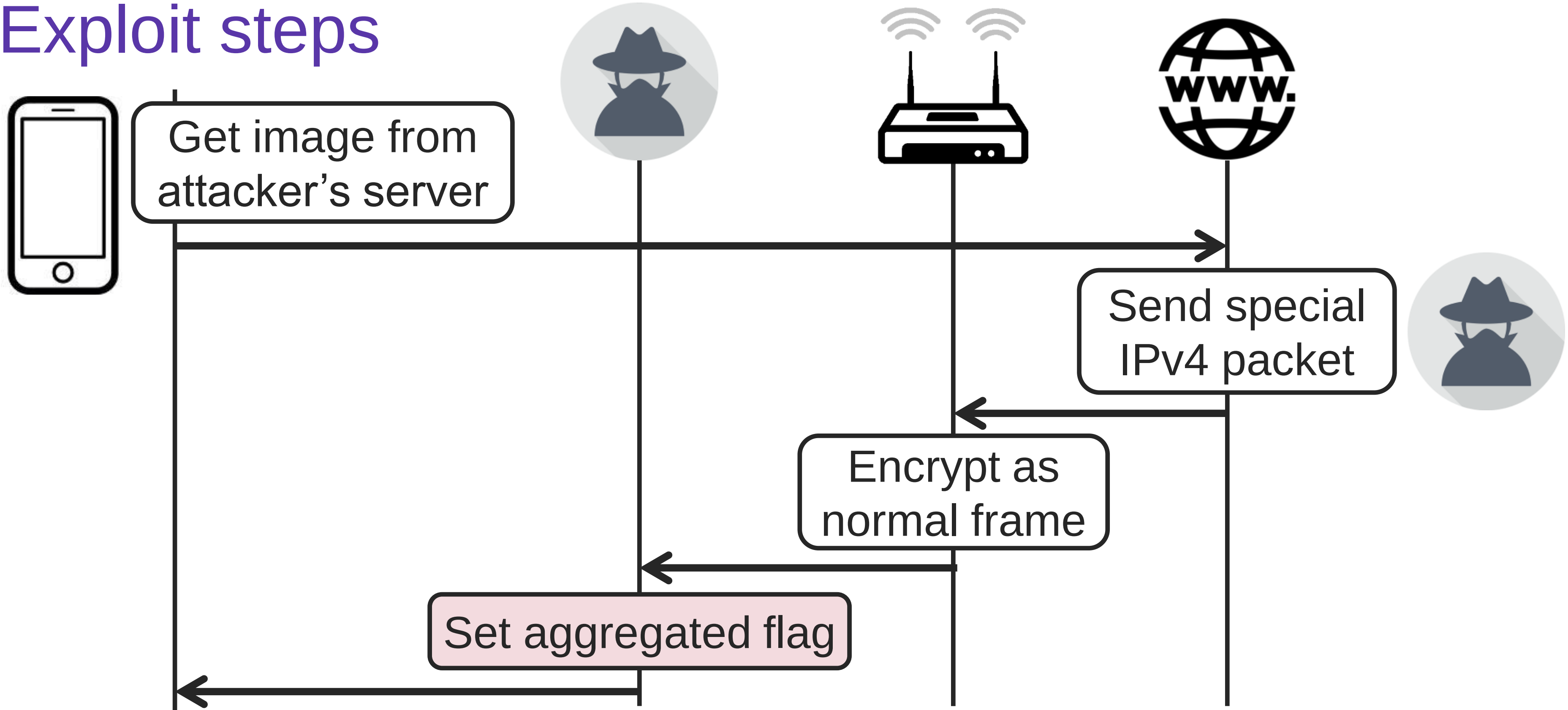
Exploit steps



Exploit steps

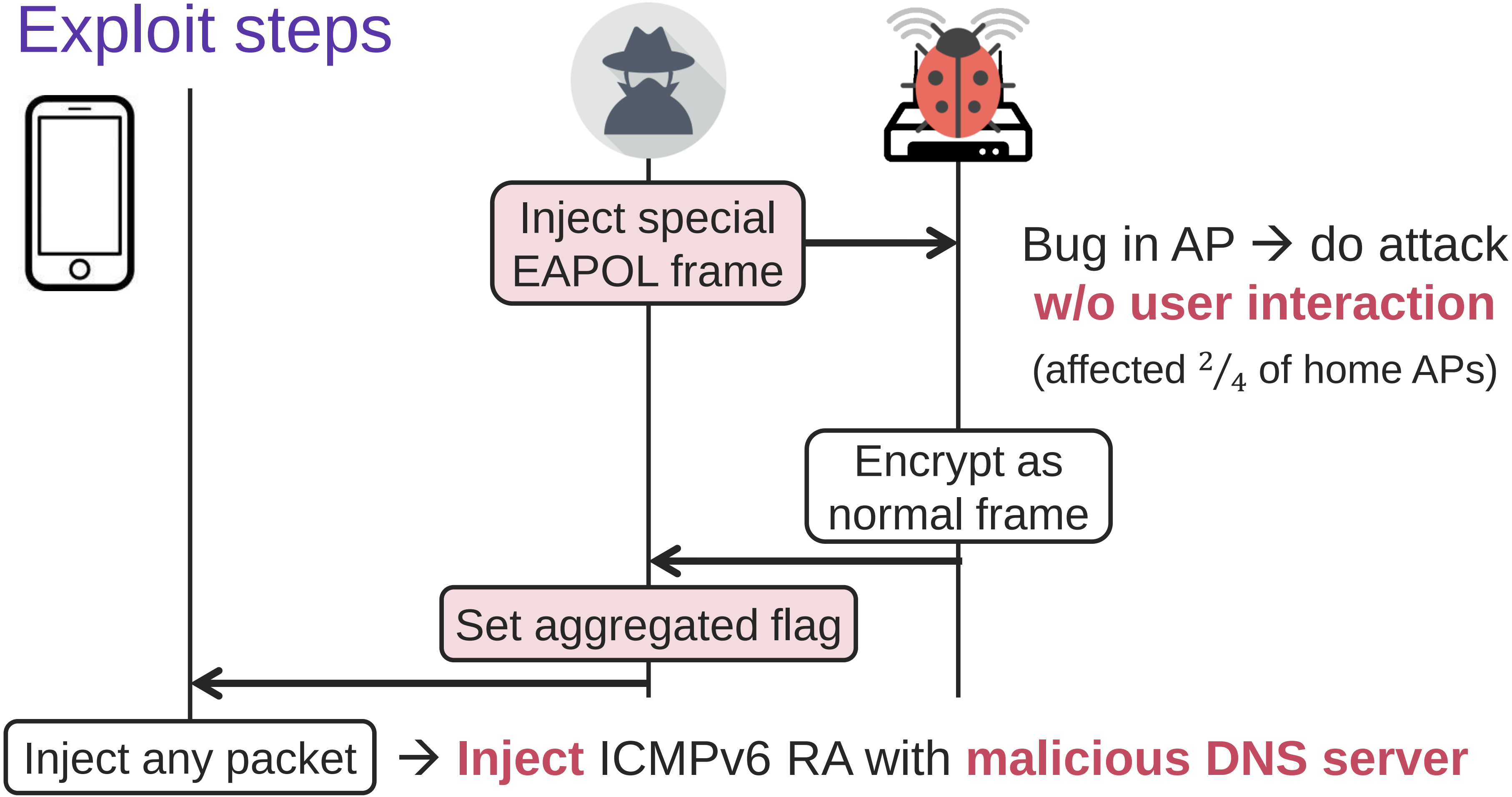


Exploit steps



Inject any packet → **Inject** ICMPv6 RA with **malicious DNS server**

Exploit steps



Aggregation

Mixed key

Fragment cache

Implementation Flaws

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:

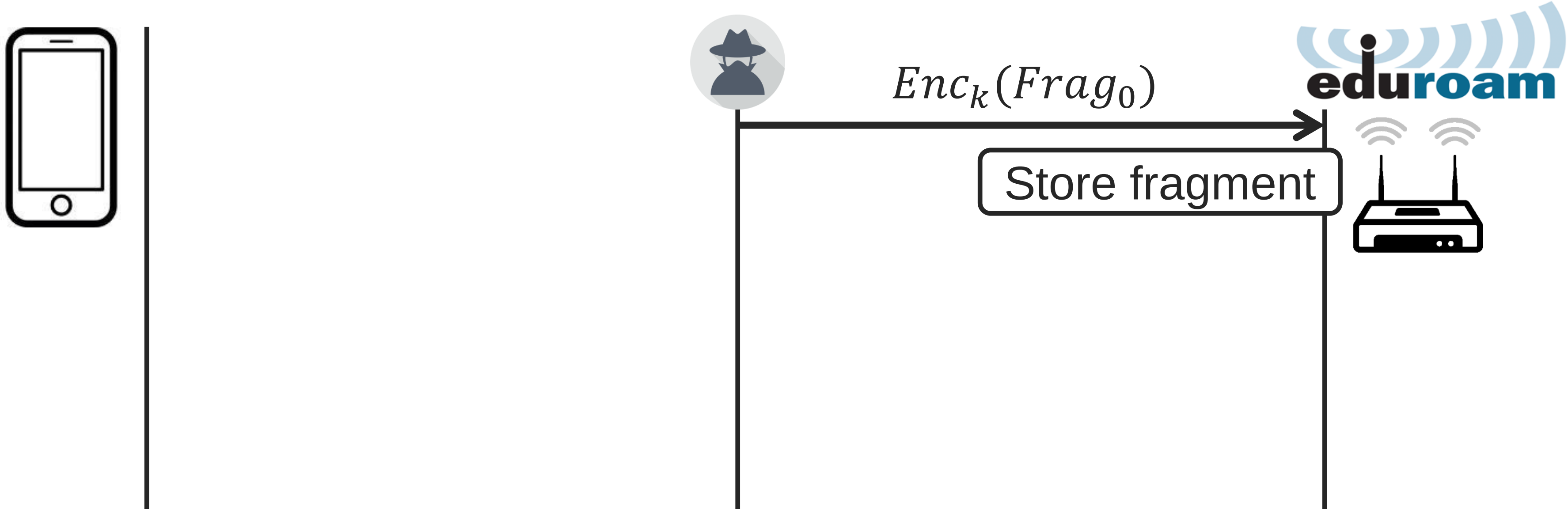
→ Protected header info defines place in original frame

Fragment cache design flaw

Fragments aren’t removed after disconnecting:

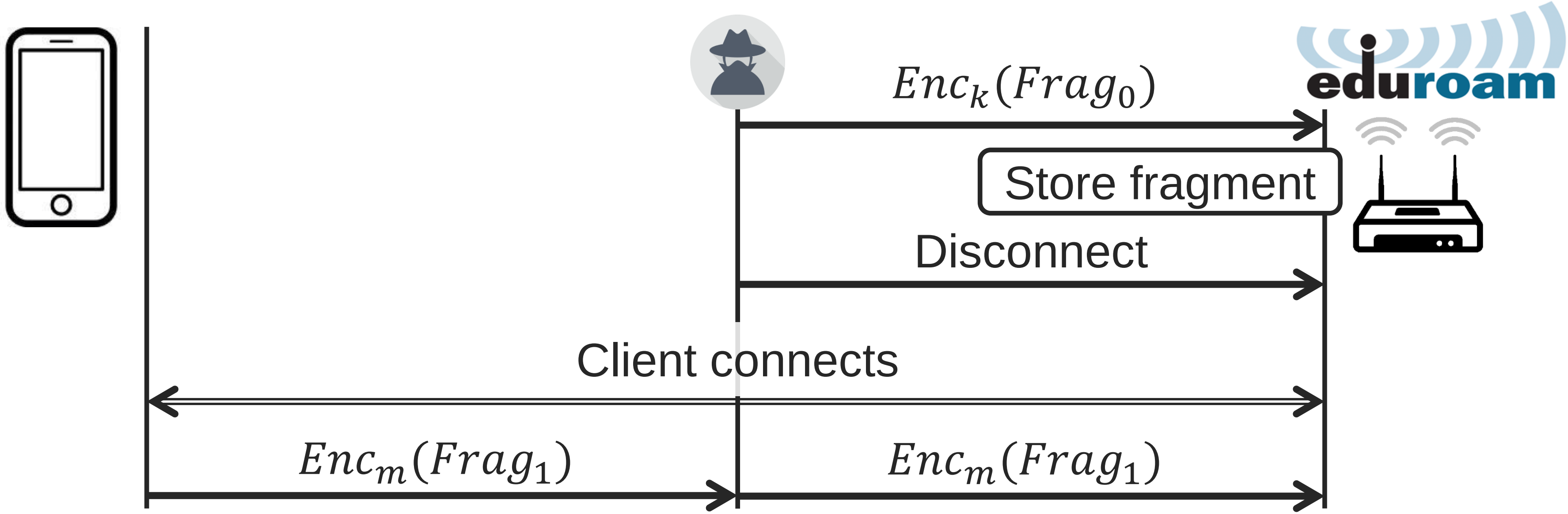
Fragment cache design flaw

Fragments aren't removed after disconnecting:



Fragment cache design flaw

Fragments aren't removed after disconnecting:



- Attacker's $Frag_0$ and client's $Frag_1$ is reassembled

Summary of impact

Abuse to **exfiltrate or inject packets** assuming:

1. Hotspot-like network where users distrust each other
2. Client sends fragmented frames (rare unless Wi-Fi 6)

Summary of impact

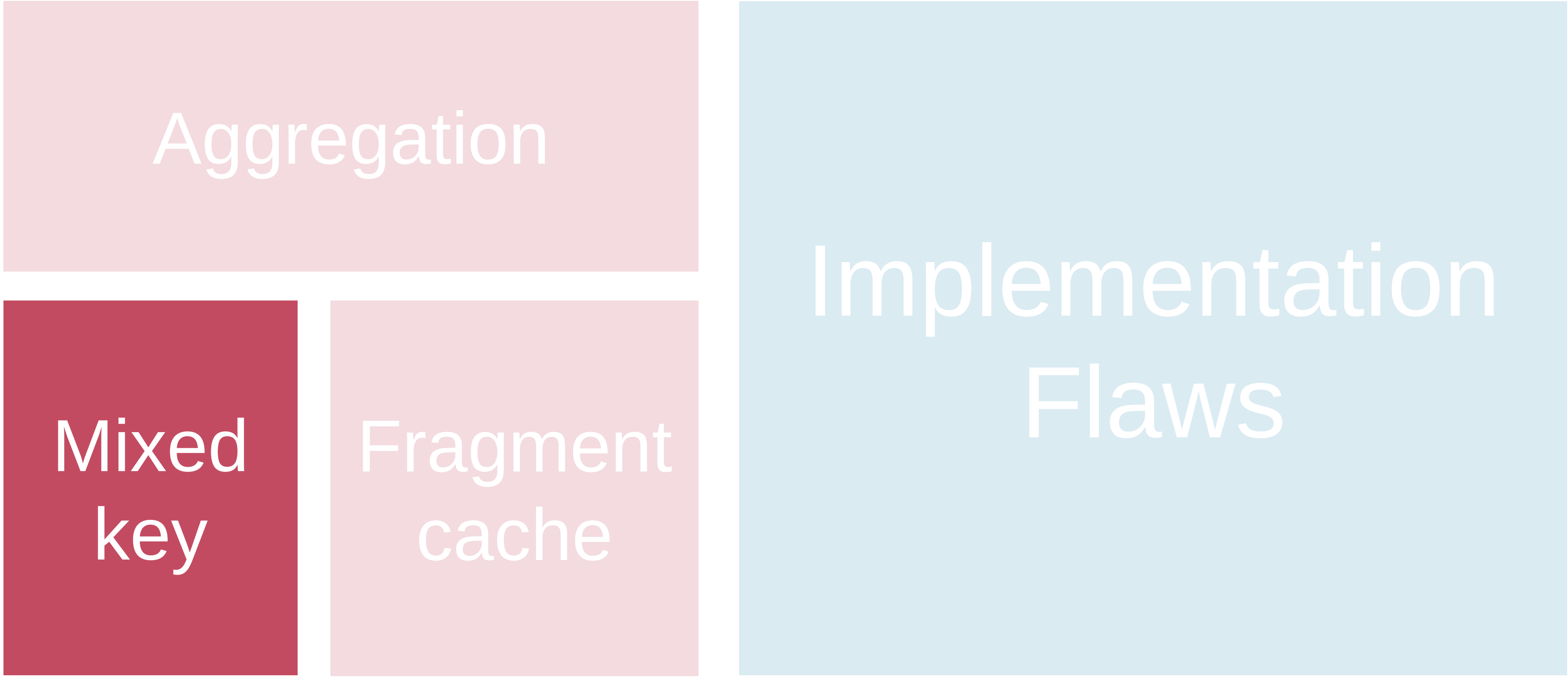
Abuse to **exfiltrate or inject packets** assuming:

1. Hotspot-like network where users distrust each other
2. Client sends fragmented frames (rare unless Wi-Fi 6)

Even the ancient **WEP protocol is affected**!

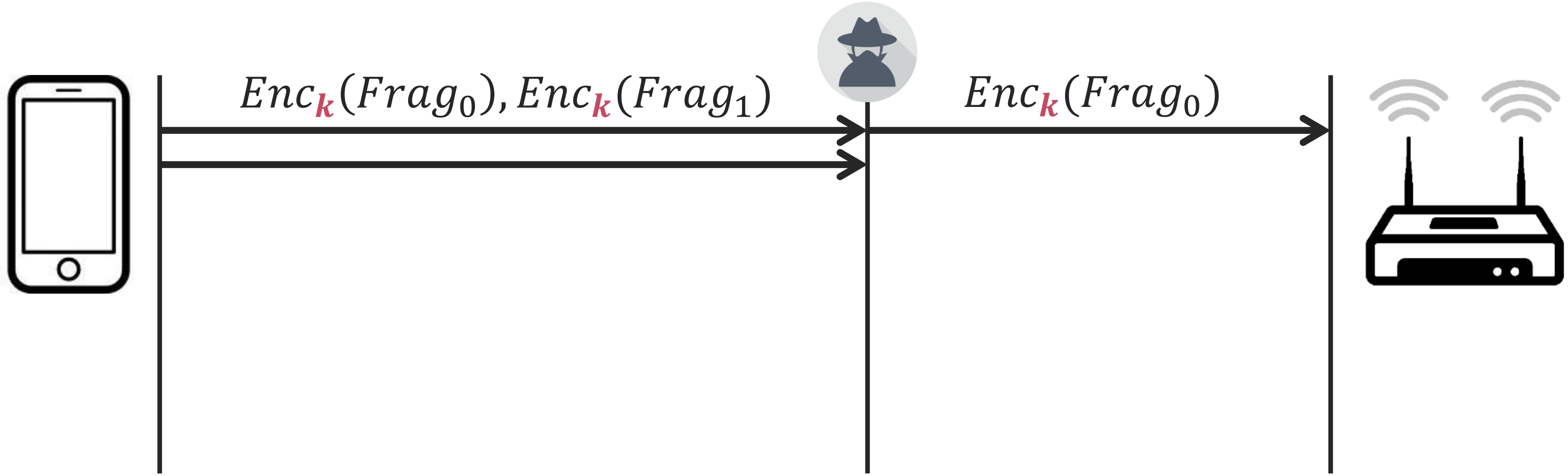
› WEP is also affected by the mixed key design flaw

→ Design flaws have been **part of Wi-Fi since 1997**


Aggregation
Mixed key
Fragment cache
Implementation Flaws

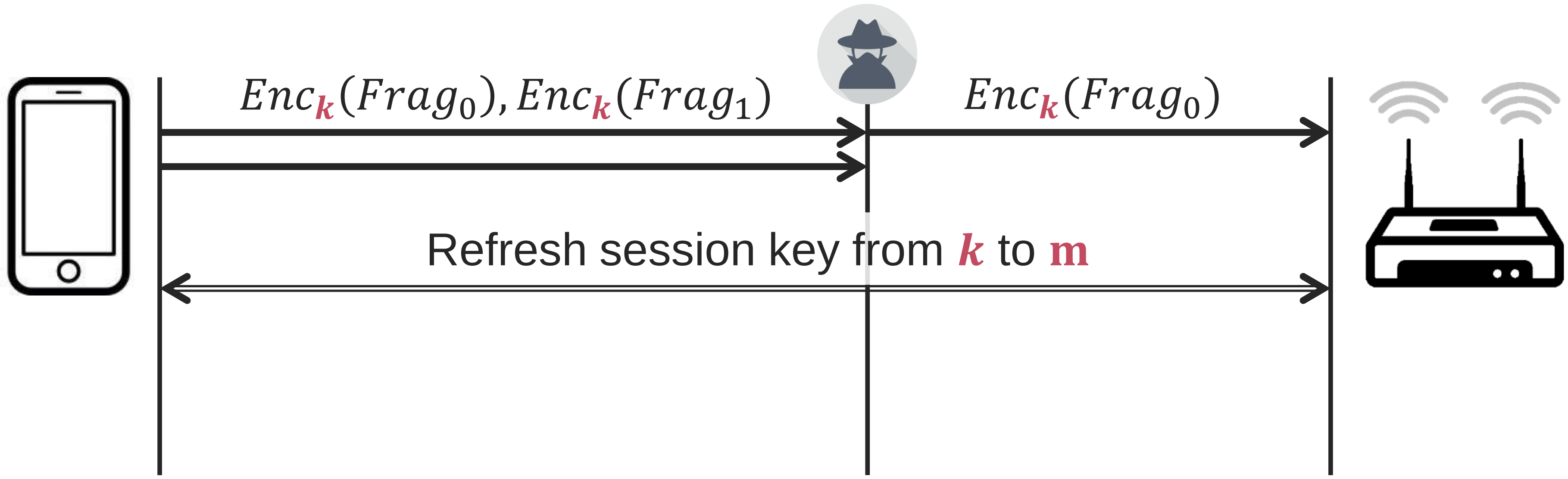
Mixed key design flaw

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



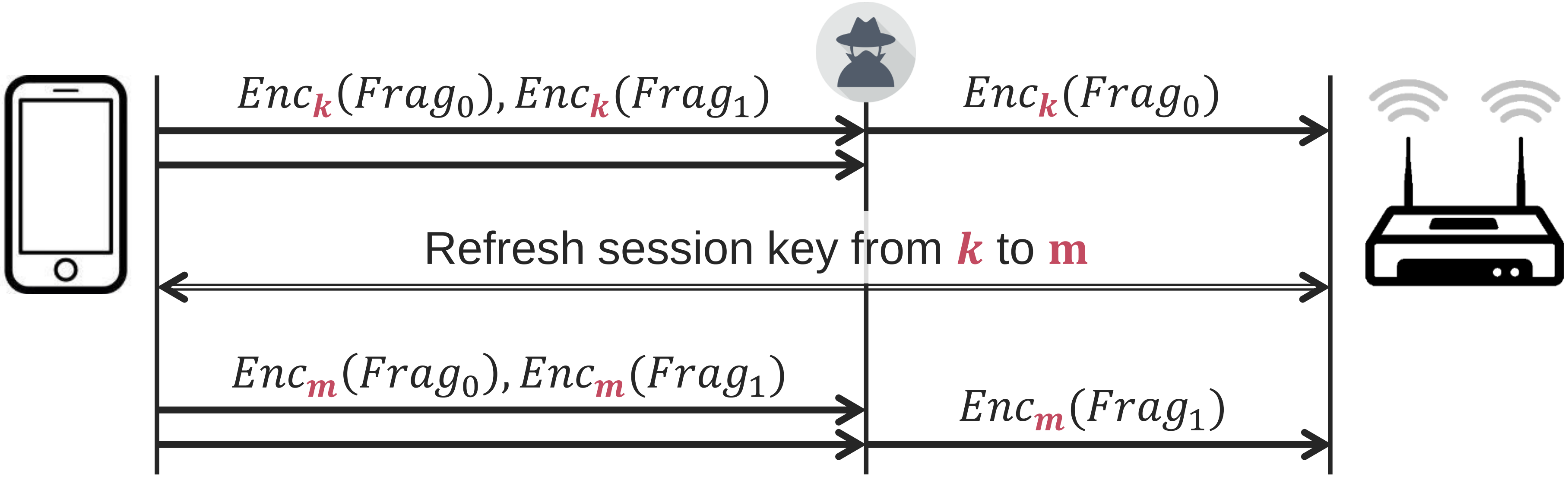
Mixed key design flaw

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



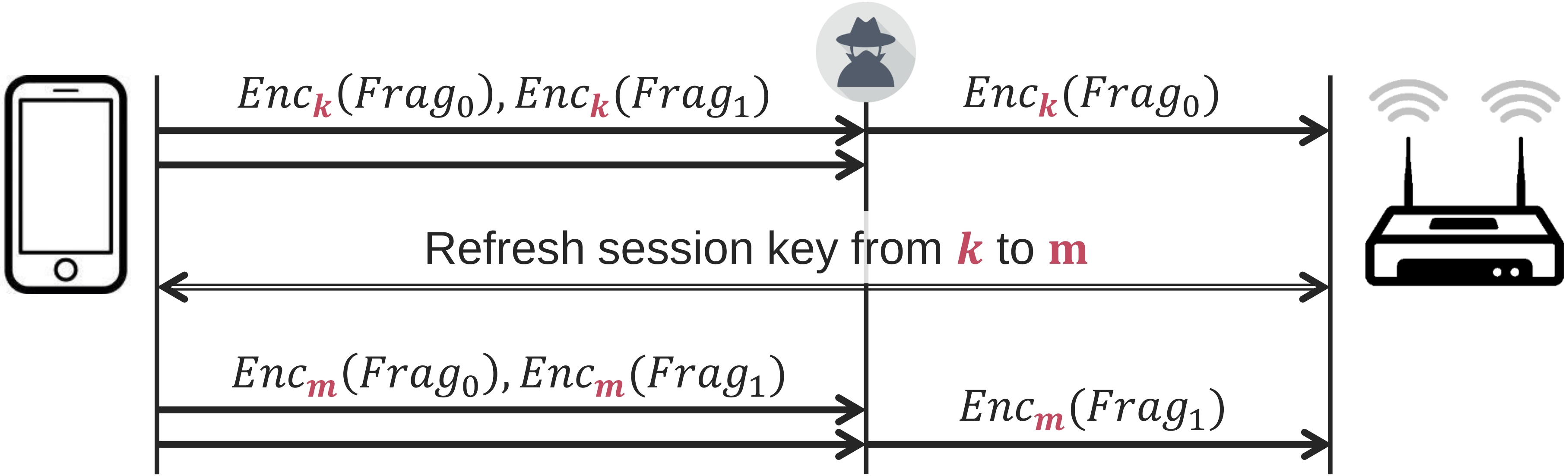
Mixed key design flaw

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



Mixed key design flaw

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



→ Can **mix fragments of different frames**

Summary of impact

Abuse to **exfiltrate data** assuming:

1. Someone sends fragmented frames (rare unless Wi-Fi 6)
2. Victim will connect to server of attacker
3. Network periodically refreshes the session key

Summary of impact

Abuse to **exfiltrate data** assuming:

1. Someone sends fragmented frames (rare unless Wi-Fi 6)
2. Victim will connect to server of attacker
3. ~~Network periodically refreshes the session key~~
   - **Combine with implementation flaw** to avoid this condition

Design flaws

Implementation Flaws

Design flaws

Plaintext frames

Broadcast fragments

Cloacked A-MSDUs

Mixed fragments

EAPOL forwarding

Out of order frag

Out of order fragments

Trivial frame injection

Plaintext frames wrongly accepted:

- Depending if **fragmented**, **broadcasted,** or while **connecting**

Trivial frame injection

Plaintext frames wrongly accepted:

- Depending if **fragmented**, **broadcasted,** or while **connecting**
- Sometimes frames that **resemble a handshake** message
- Examples: Apple and some Android devices, some Windows dongles, home and professional APs, and many others!

→ Can trivially **inject frames**

Design flaws

Plaintext frames

Mixed fragments

Out of order frag

Broadcast fragments

EAPOL forwarding

Cloacked A-MSDUs

No fragmentation support

No fragmentation support

Some devices don't support fragmentation

- But they **treat fragmented frames as full frames**
- Examples: OpenBSD and Espressif chips

→ Abuse to **inject frames** under right conditions

→ **All devices are vulnerable** to one or more flaws

Created tool to test devices

Has **45+ test cases** for both **clients and APs**:

Command	Short d
Sanity checks	
`ping`	Send a normal ping.
`ping I,E,E`	Send a normal fragmented ping.
Basic device behaviour	
`ping I,E,E --delay 5`	Send a normal fragmented ping with a
`ping-frag-sep`	Send a normal fragmented ping with fr
`ping-frag-sep --pn-per-qos`	Same as above, but also works if the tar
A-MSDU attacks (§3)	
`ping I,E --amsdu`	Send a ping encapsulated in a normal (
`amsdu-inject`	Simulate attack: send A-MSDU frame w
`amsdu-inject-bad`	Same as above, but against targets that
Mixed key attacks (§4)	
`ping I,F,BE,AE`	Inject two fragments encrypted under a
`ping I,F,BE,AE --pn-per-qos`	Same as above, but also works if the tar
Cache attacks (§5)	
`ping I,E,R,AE`	Inject a fragment, try triggering a *reass*
`ping I,E,R,E`	Same as above, but with a longer delay
`ping I,E,R,AE --full-reconnect`	Inject a fragment, *deauthenticate* and re
`ping I,E,R,E --full-reconnect`	Same as above, but with a longer delay

Non-consecutive PNs attack (§6.2)	
`ping I,E,E --inc-pn 2`	Send a fragmented ping with non-
Mixed plain/encrypt attack (§6.3)	
`ping I,E,P`	Send a fragmented ping: first fragn
`ping I,P,E`	Send a fragmented ping: first fragn
`ping I,P`	Send a plaintext ping.
`ping I,P,P`	Send a fragmented ping: both frag
`linux-plain`	Mixed plaintext/encrypted fragmer
Broadcast fragment attack (§6.4)	
`ping I,D,P --bcast-ra`	Send a unicast ping in a plaintext b
`ping D,BP --bcast-ra`	Same as above, but frame is sent d
A-MSDU EAPOL attack (§6.5)	
`eapol-amsdu I,P`	Send a plaintext A-MSDU containir
`eapol-amsdu BP`	Same as above, but the frame is se
`eapol-amsdu-bad I,P`	Send malformed plain. A-MSDU co
`eapol-amsdu-bad BP`	Same as above, but the frame is se

Command	Short d
A-MSDU attacks (§3)	
`ping I,E --amsdu-fake`	If this test succeeds, the A-MSDU fl
`ping I,E --amsdu-fake --amsdu-spp`	Check if the A-MSDU flag is authen
Mixed key attacks (§4)	
`ping I,F,BE,E`	In case the new key is installed relat
`ping I,E,F,AE`	Variant if no data frames are accept
`ping I,E,F,AE --rekey-plain`	If the device performs the rekey har
`ping I,E,F,AE --rekey-plain --rekey-req`	Same as above, and actively reques
`ping I,E,F,AE --rekey-early-install`	Install the new key after sending me
`ping I,E,F,E [--rekey-pl] [--rekey-req]`	Same as above 4 tests, but with lon
`ping I,F,BE,AE --freebsd`	Mixed key attack against FreeBSD c
Cache attacks (§5)	
`ping I,E,R,AE --freebsd [--full-reconnect]`	Cache attack specific to FreeBSD im
`ping I,E,R,AP --freebsd [--full-reconnect]`	Cache attack specific to FreeBSD im
`ping I,E,R,AP [--full-reconnect]`	Cache attack test where 2nd fragme

Mixed plain/encrypt attack (§6.3)	
`ping I,E,E --amsdu`	Send a normal ping as a fragment
`ping I,E,P,E`	Ping with first frag. encrypted, sec
`linux-plain 3`	Same as linux-plain but decoy frag
Broadcast checks (extensions of §6.4)	
`ping I,P --bcast-ra`	Ping in a plaintext broadcast frame
`ping BP --bcast-ra [--bcast-dst]`	Ping in plaintext broadcast frame c
`ping BP [--bcast-dst]`	Ping in a plaintext frame during th
`eapfrag BP,BP`	Experimental broadcast fragment a
A-MSDU EAPOL attack (§6.5)	
`eapol-amsdu[-bad] BP --bcast-dst`	Same as `eapol-amsdu BP` but easie
AP forwards EAPOL attack (§6.6)	
`eapol-inject 00:11:22:33:44:55`	Test if AP forwards EAPOL frames b
`eapol-inject-large 00:11:22:33:44:55`	Make AP send fragmented frames
No fragmentation support attack (§6.8)	
`ping I,D,E`	Send ping inside an encrypted seco
`ping I,E,D`	Send ping inside an encrypted first

→ Available at https://github.com/vanhoefm/fragattack

Discussion

Design flaws took **two decades** to discover

- Without modified drivers some attacks will fail

Discussion

Design flaws took **two decades** to discover

- Without modified drivers some attacks will fail
- Fragmentation & aggregation wasn’t considered important

Discussion

Design flaws took **two decades** to discover

- Without modified drivers some attacks will fail
- Fragmentation & aggregation wasn't considered important

Long-term lessons:

- **Adopt defences early** even if concerns are theoretic
- Isolate **security contexts** (data decrypted with different keys)
- **Keep fuzzing** devices. Wi-Fi Alliance can help here!

Conclusion

- Discovered three **design flaws**
- Multiple **implementation flaws**
- Several flaws are **trivial to exploit**
- More info: www.fragattacks.com