

Fragment and Forge:

Breaking Wi-Fi Through Frame Aggregation and Fragmentation

USENIX Security '21 (video made on April 20, 2021)



Mathy Vanhoef



NEW YORK UNIVERSITY

Advancements in Wi-Fi security

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

Advancements in Wi-Fi security

- › WPA3 is continuously 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 advancements,
found **flaws in all networks** (incl. WPA2/3)

Design
flaws

Implementation
Flaws

Design
flaws

Implementation
Flaws

Aggregation

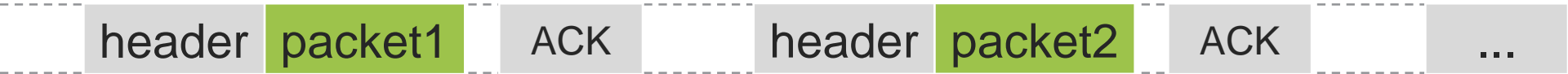
Mixed
key

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:

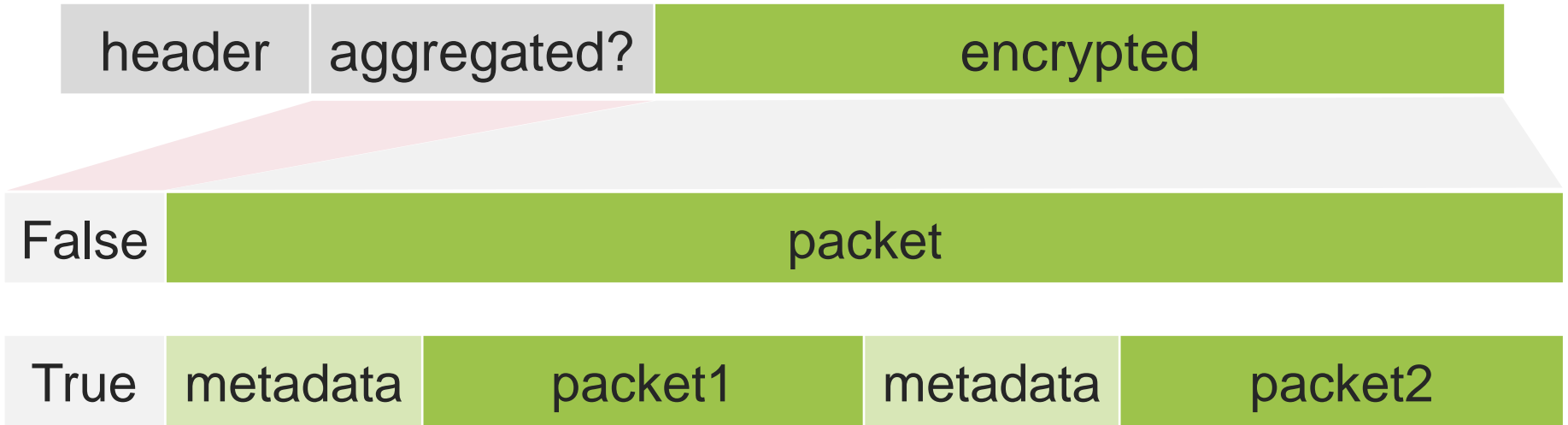


This can be avoided by **aggregating frames**:



Problem: how to recognize aggregated frames?

Aggregation design flaw



Aggregation design flaw

Not authenticated



False

packet

True

metadata

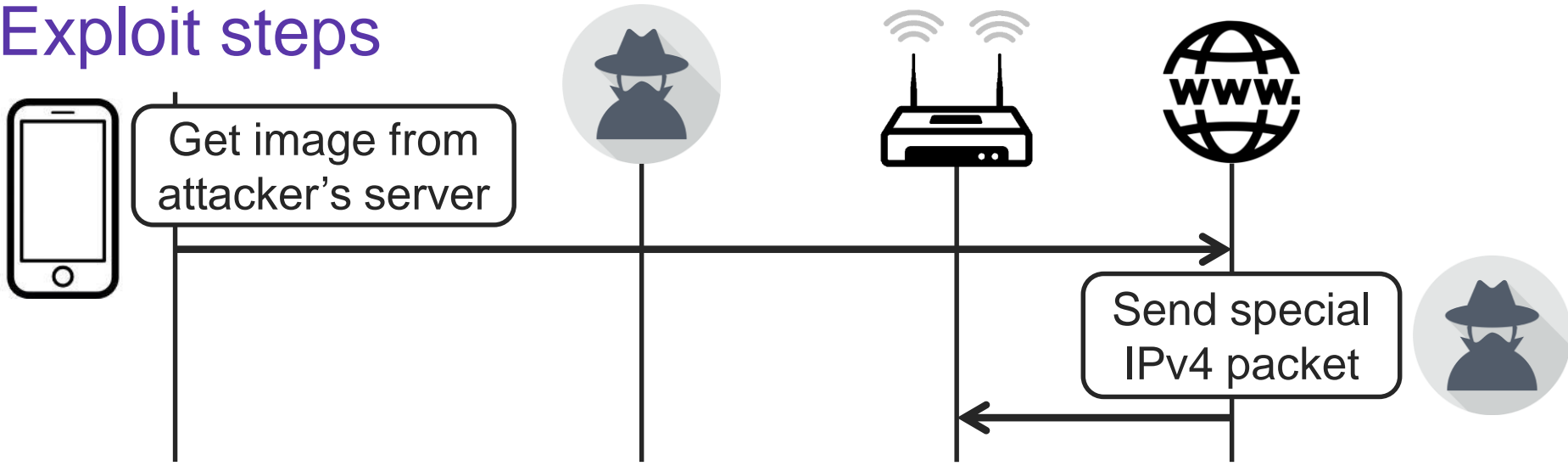
packet1

metadata

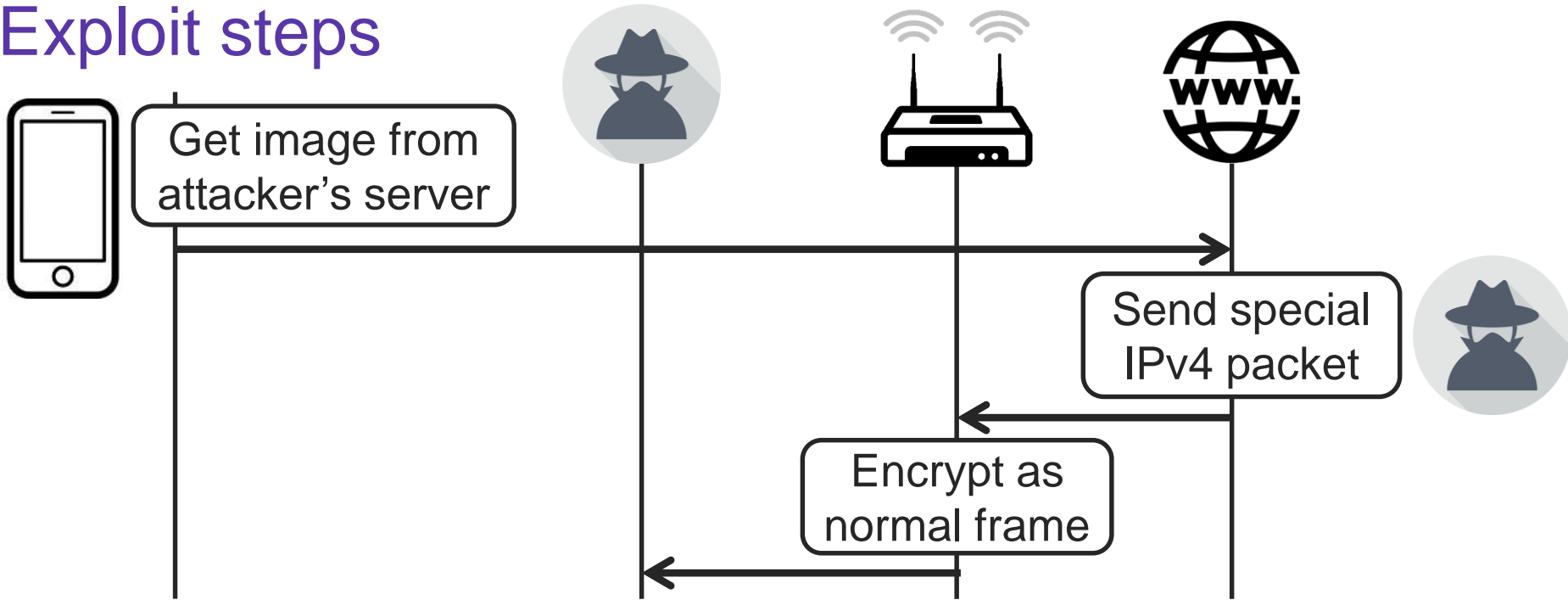
packet2

→ Flip flag → payload is parsed differently → inject packets

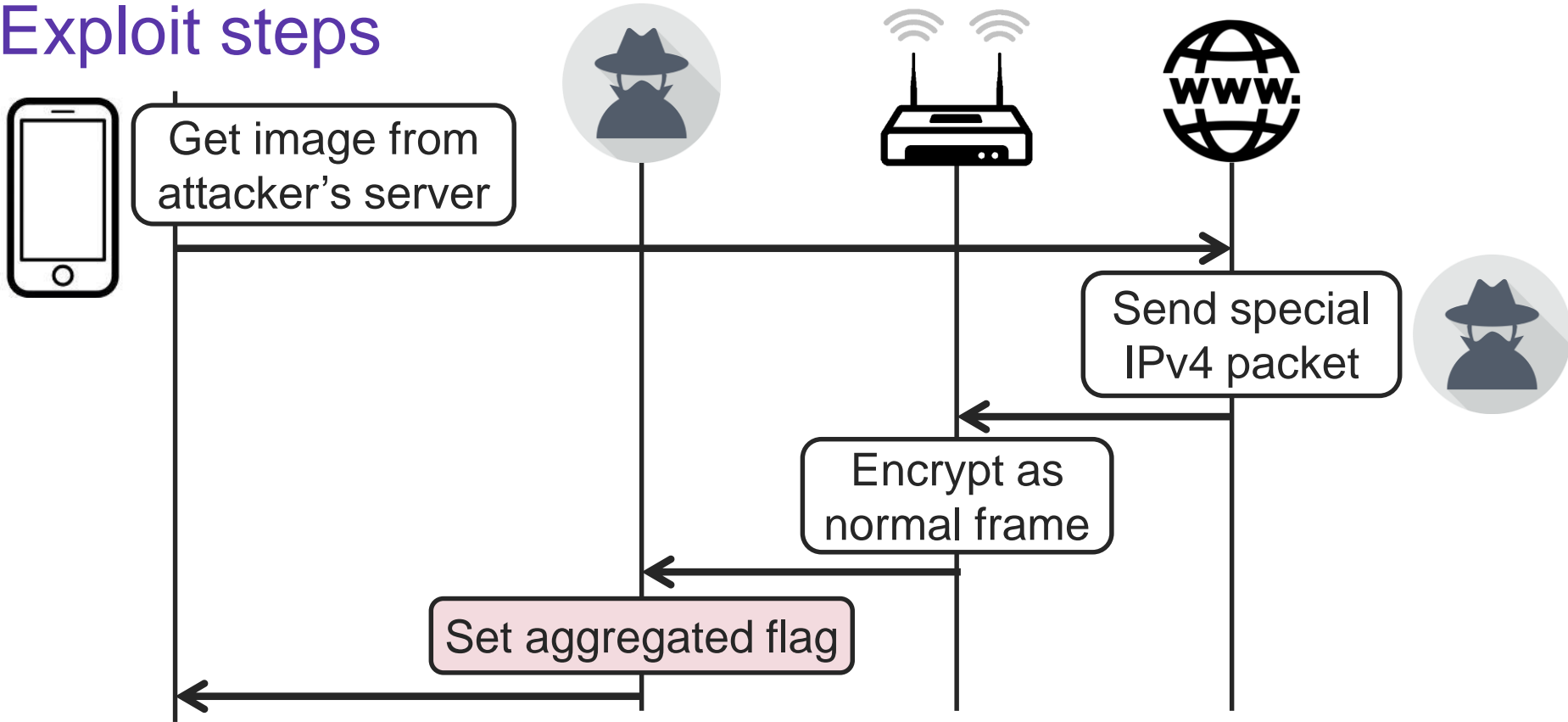
Exploit steps



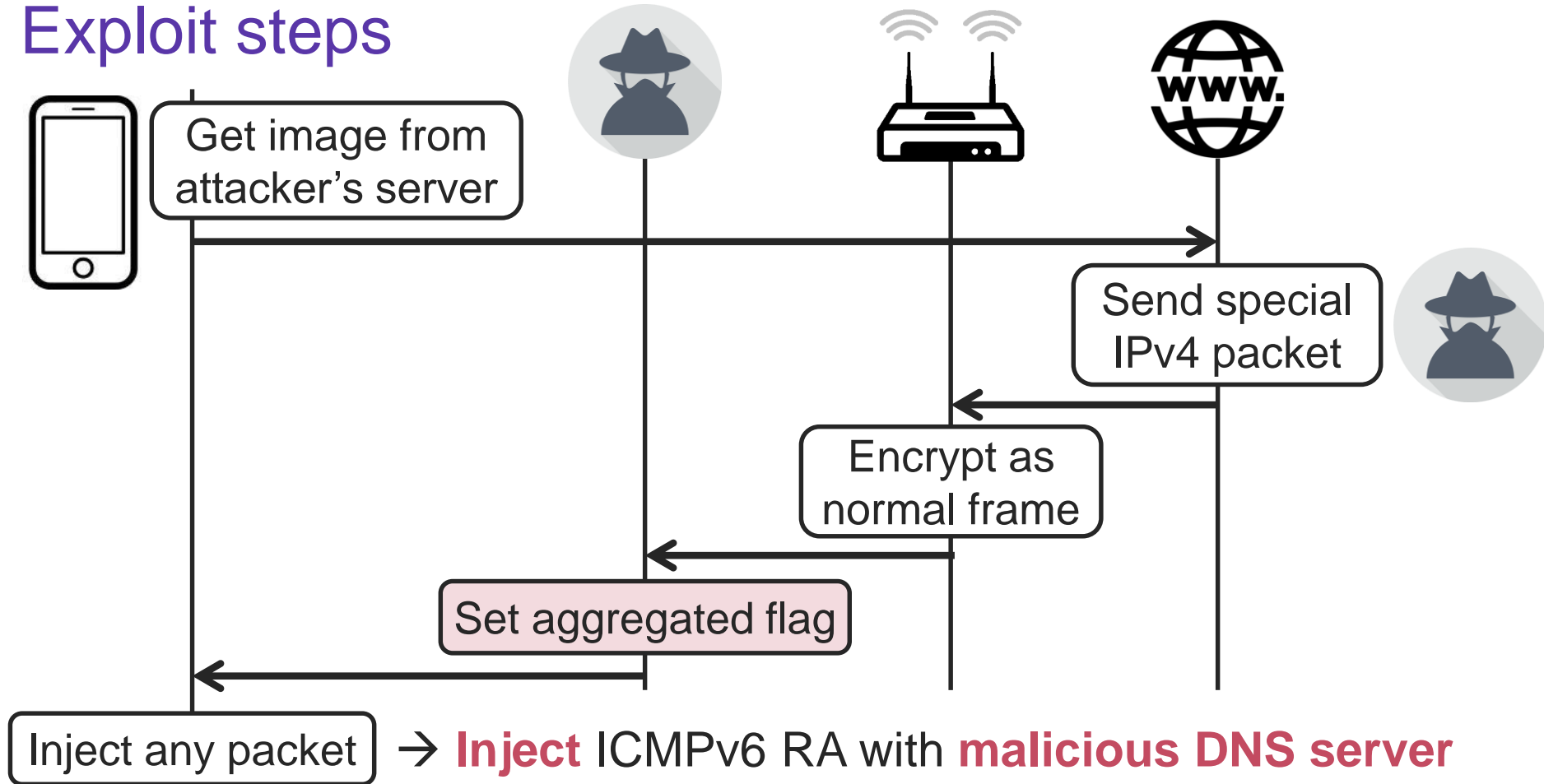
Exploit steps



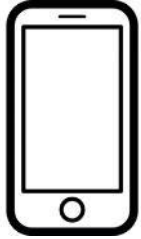
Exploit steps



Exploit steps



Exploit steps



Inject special EAPOL frame

Bug in AP → do attack
w/o user interaction
(affected $\frac{2}{4}$ of home APs)

Encrypt as normal frame

Set aggregated flag

Inject any packet

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

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:



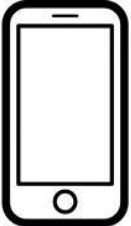
→ Protected header info defines place in original frame

Fragment cache design flaw

Fragments aren't removed after disconnecting:

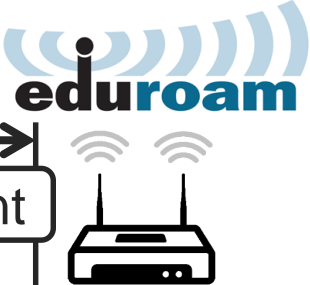
Fragment cache design flaw

Fragments aren't removed after disconnecting:



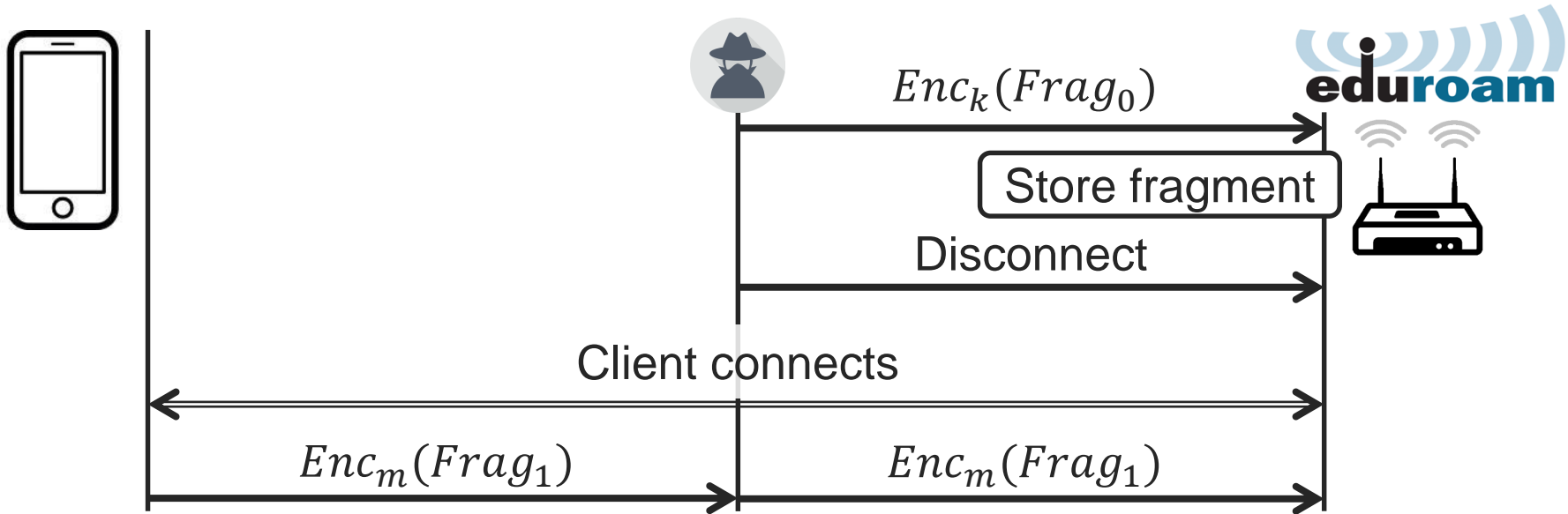
$Enc_k(Frag_0)$

Store fragment



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

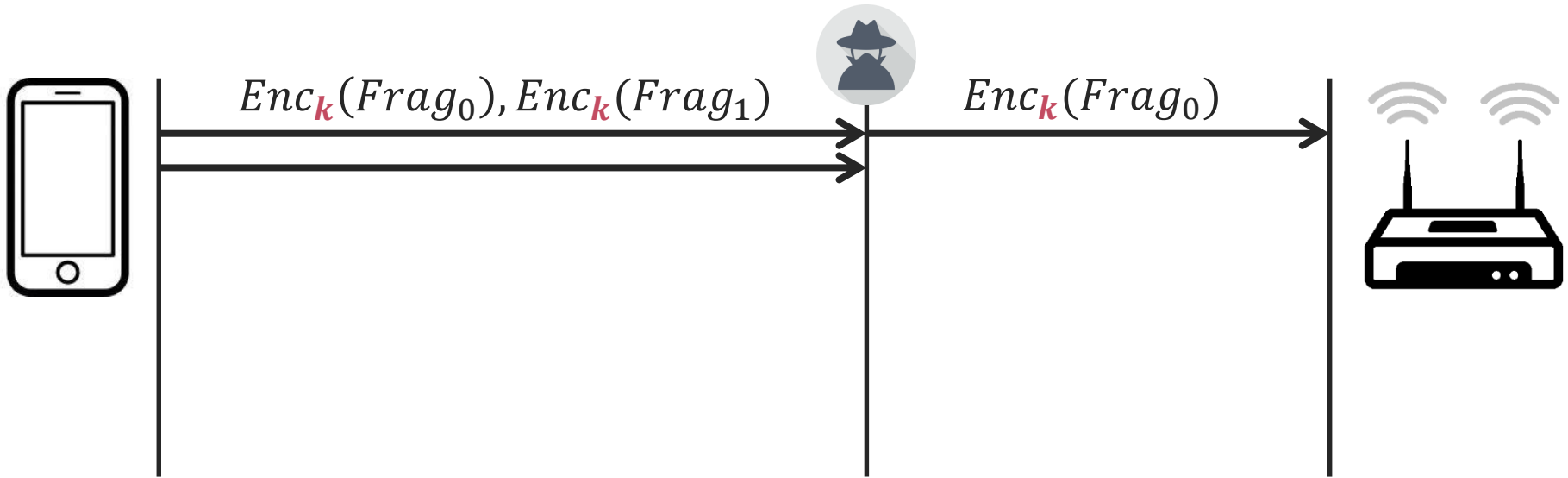
Mixed
key

Fragment
cache

Implementation
Flaws

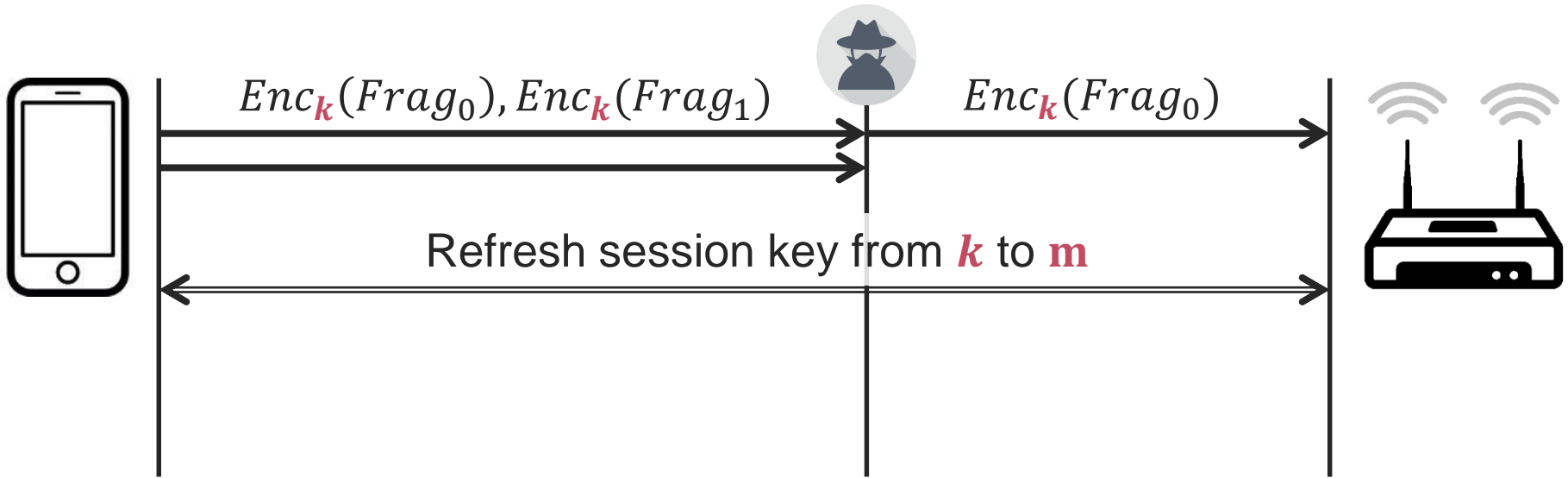
Mixed key design flaw

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



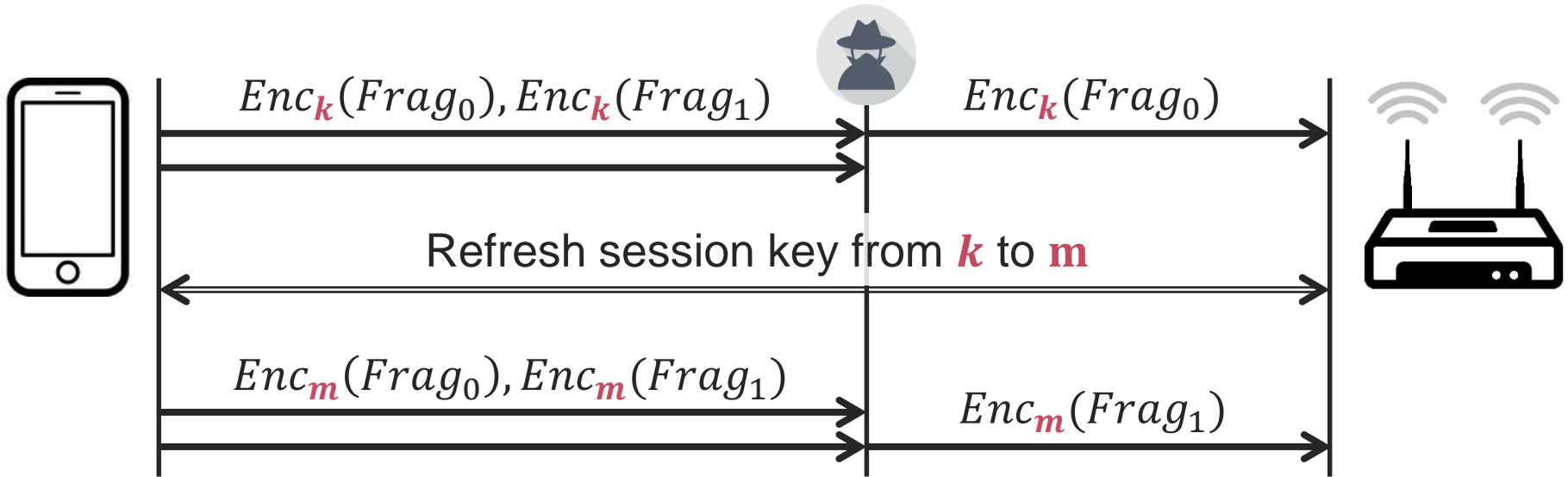
Mixed key design flaw

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



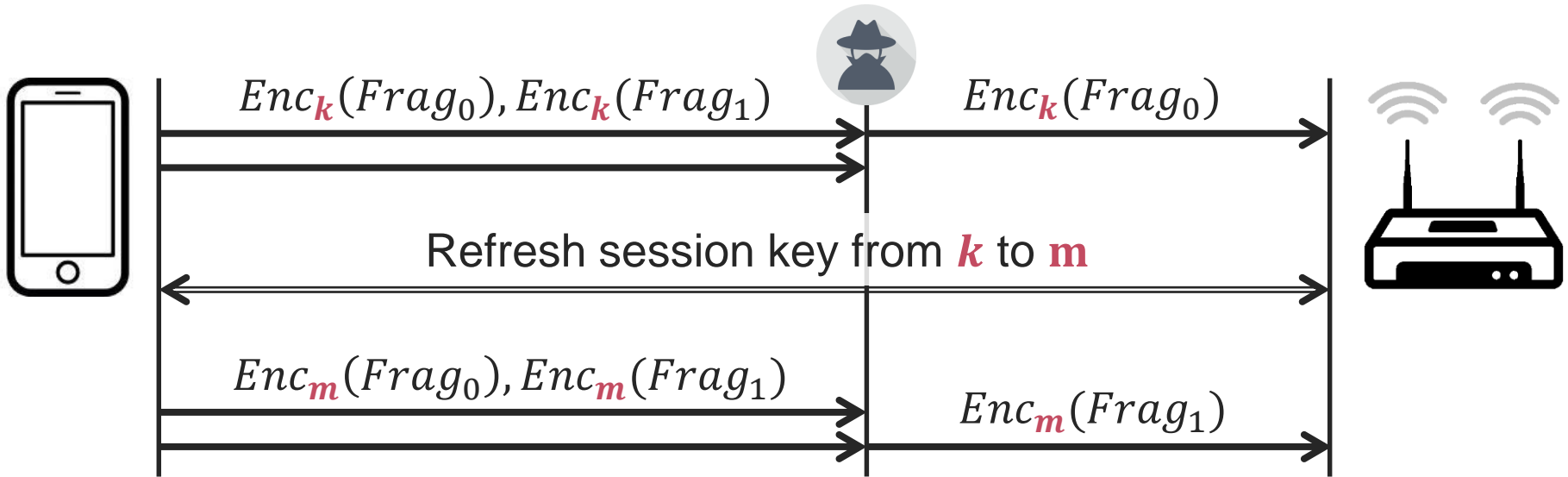
Mixed key design flaw

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

Mixed fragments

Broadcast fragments

EAPOL forwarding

Cloacked A-MSDUs

Out of order fragments

Out of order frag

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 c
<i>Sanity checks</i>	
ping	Send a normal ping.
ping I,F,E	Send a normal fragmented ping.
<i>Basic device behaviour</i>	
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
<i>A-MSDU attacks (§3)</i>	
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
<i>Mixed key attacks (§4)</i>	
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
<i>Cache attacks (§5)</i>	
ping I,E,R,AE	Inject a fragment, try triggering a reuss
ping I,F,R,E	Same as above, but with a longer delay
ping I,E,R,AE --full-reconnect	Inject a fragment, <i>deauthenticate</i> and re
ping I,E,R,E --full-reconnect	Same as above, but with a longer delay

<i>Non-consecutive PNs attack (§6.2)</i>	
ping I,E,E --inc-pn 2	Send a fragmented ping with non-c
<i>Mixed plain/encrypt attack (§6.3)</i>	
ping I,E,P	Send a fragmented ping: first fragm
ping I,P,E	Send a fragmented ping: first fragm
ping I,P	Send a plaintext ping.
ping I,P,P	Send a fragmented ping: both frag
linux-plain	Mixed plaintext/encrypted fragmer
<i>Broadcast fragment attack (§6.4)</i>	
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
<i>A-MSDU EAPOL attack (§6.5)</i>	
eapol-amsdu I,P	Send a plaintext A-MSDU containi
eapol-amsdu BP	Same as above, but the frame is sei
eapol-amsdu-bad I,P	Send malformed plain. A-MSDU co
eapol-amsdu-bad BP	Same as above, but the frame is sei

Command	Short di
<i>A-MSDU attacks (§3)</i>	
ping I,E --amsdu-fake	If this test succeeds, the A-MSDU fi
ping I,E --amsdu-fake --amsdu-spp	Check if the A-MSDU flag is authen
<i>Mixed key attacks (§4)</i>	
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 m
ping I,E,F,AE [--rekey-pl] [--rekey-req]	Same as above 4 tests, but with lon
ping I,F,BE,AE --freebsd	Mixed key attack against FreeBSD c
<i>Cache attacks (§5)</i>	
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 fragm

<i>Mixed plain/encrypt attack (§6.3)</i>	
ping I,E,E --amsdu	Send a normal ping as a fragm
ping I,E,P,E	Ping with first frag. encrypted, secc
linux-plain 3	Same as linux-plain but decoy frag
<i>Broadcast checks (extensions of §6.4)</i>	
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 :
<i>A-MSDU EAPOL attack (§6.5)</i>	
eapol-amsdu[-bad] BP --bcast-dst	Same as eapol-amsdu BP but easie
<i>AP forwards EAPOL attack (§6.6)</i>	
eapol-inject 00:11:22:33:44:55	Test if AP forwards EAPOL frames t
eapol-inject-large 00:11:22:33:44:55	Make AP send fragmented frames
<i>No fragmentation support attack (§6.8)</i>	
ping I,D,E	Send ping inside an encrypted sec
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