FragAttacks: Fragmentation & Aggregation Attacks against Wi-Fi

Mathy Vanhoef

Draft version 1, 8 March 2021

NEW YORK UNIVERSITY

Table of contents

Design flaws:

Table of contents

Aggregation Attack

CVE-2020-24588

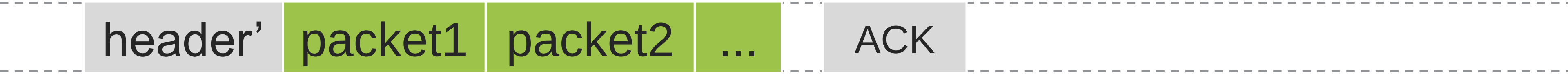
Background

Sending small frames causes high overhead:

Background

Sending small frames causes high overhead:

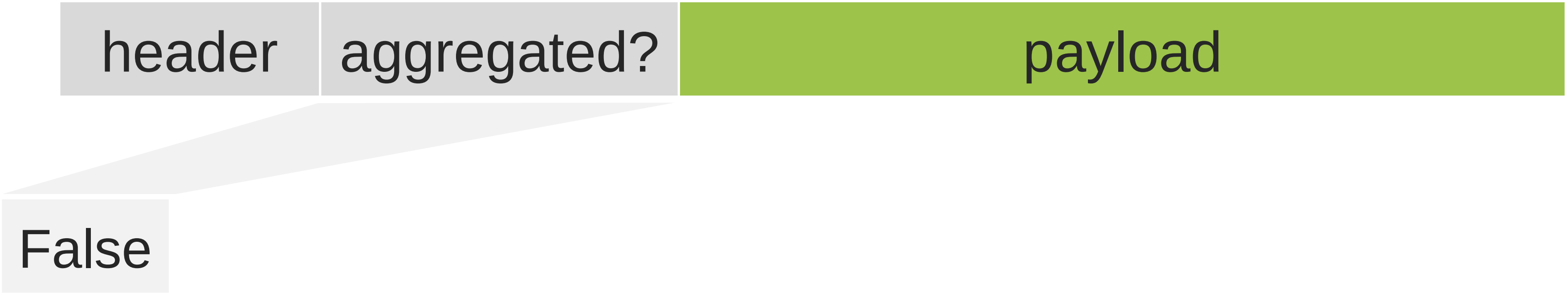
This can be avoided by **aggregating frames**:



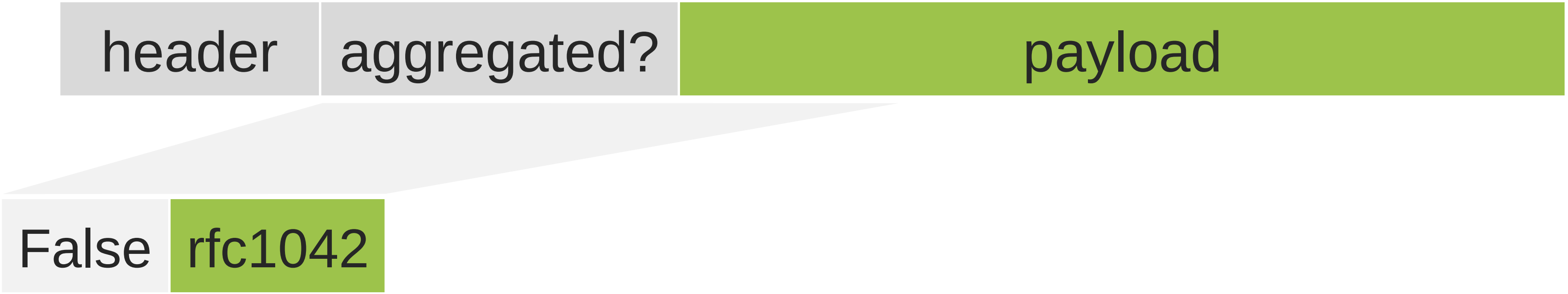
Problem: how to recognize aggregated frames?

Recognizing aggregated frames

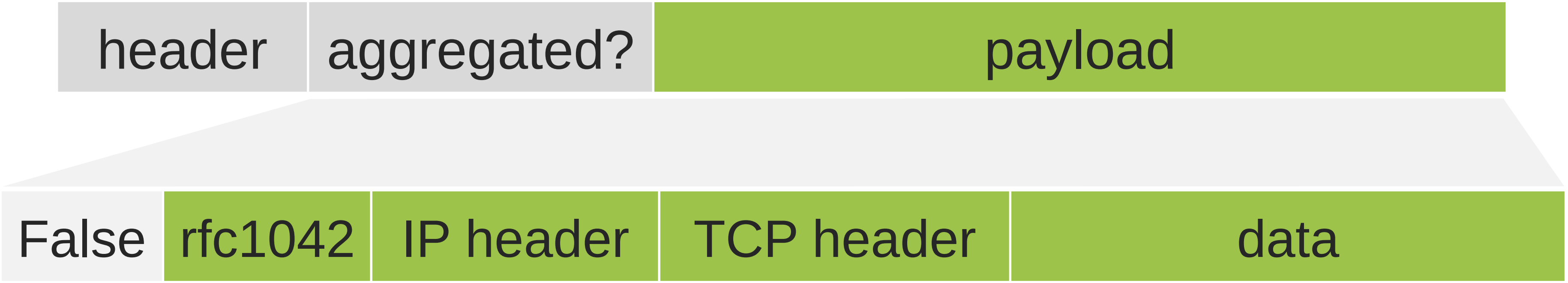
Recognizing aggregated frames



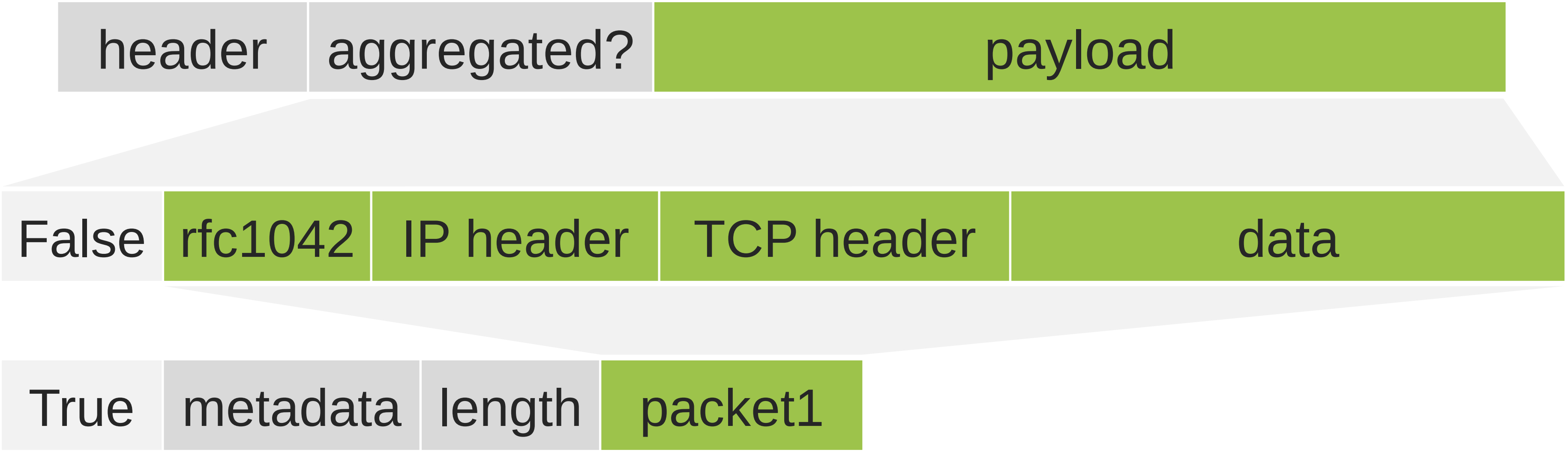
Recognizing aggregated frames



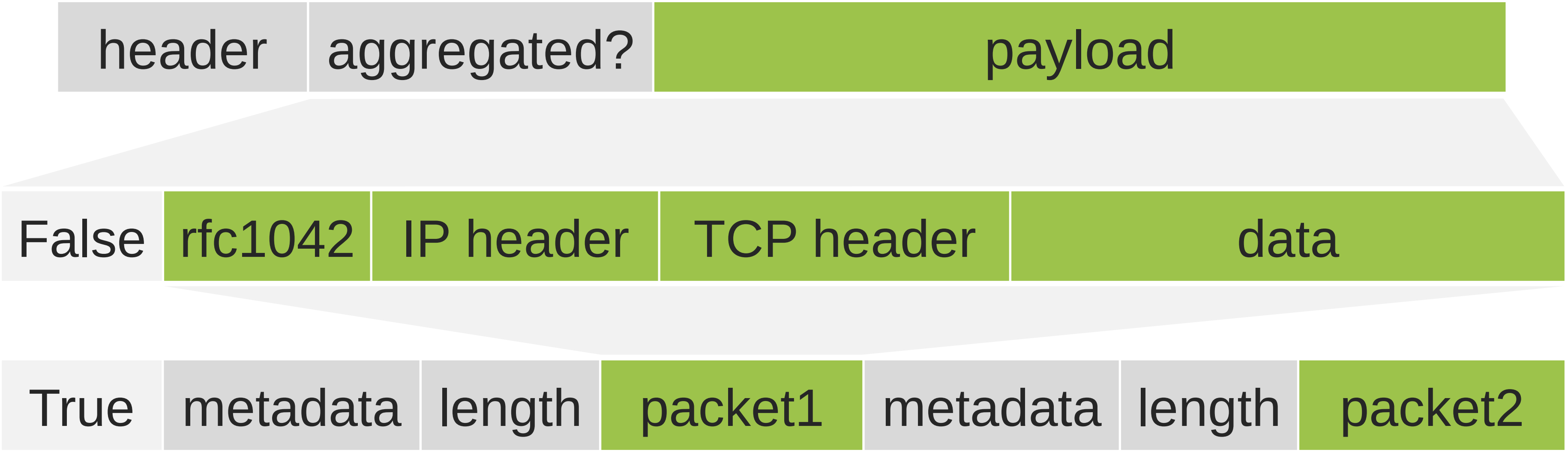
Recognizing aggregated frames



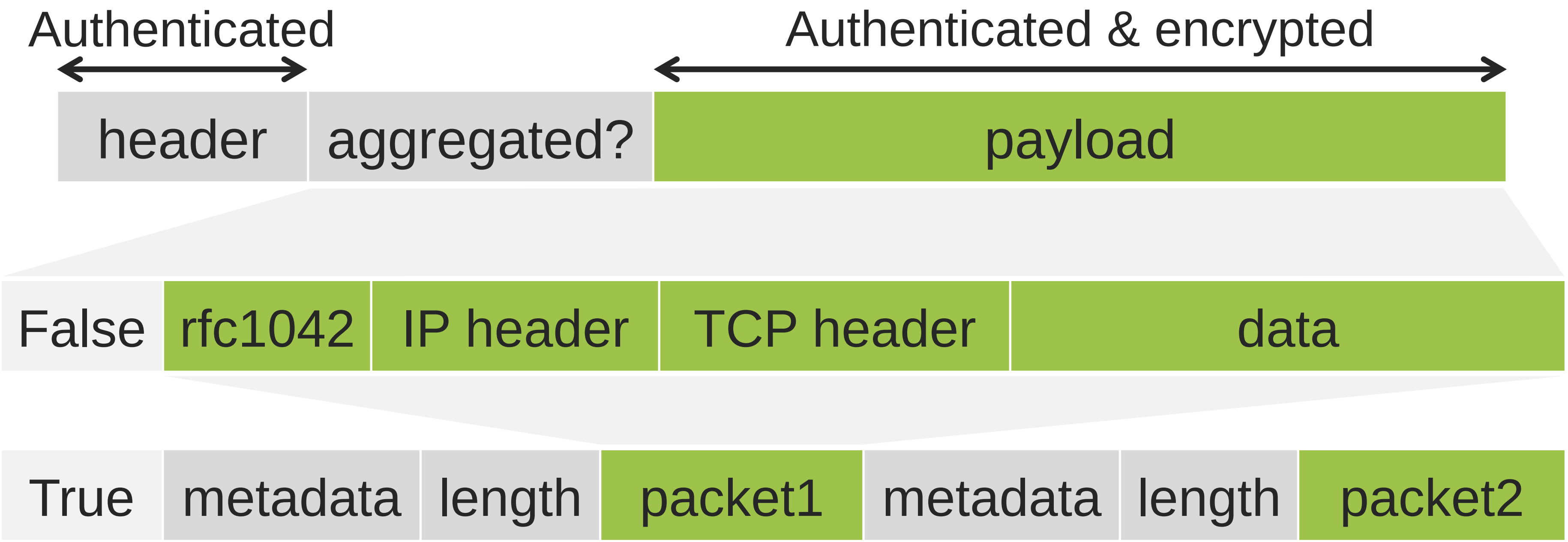
Recognizing aggregated frames



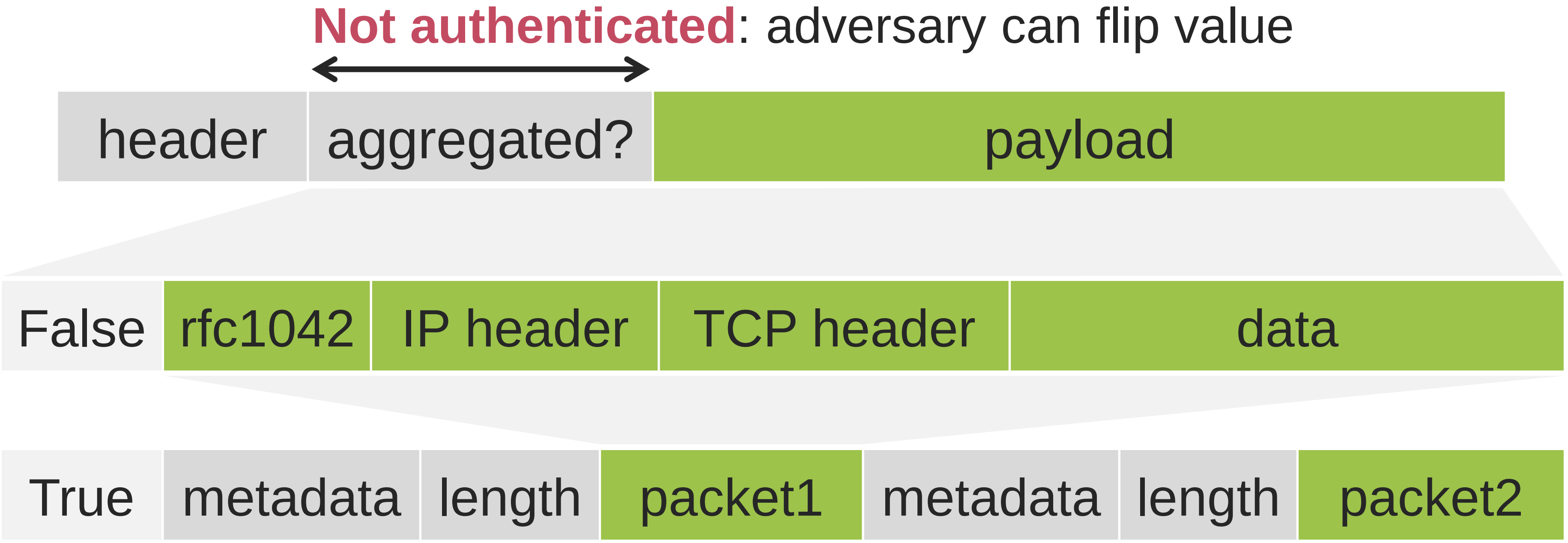
Recognizing aggregated frames



Recognizing aggregated frames



Recognizing aggregated frames



What happens if we flip this flag?

Spoofing aggregated frames

False	rfc1042	IPv4 hdr	TCP hdr	data

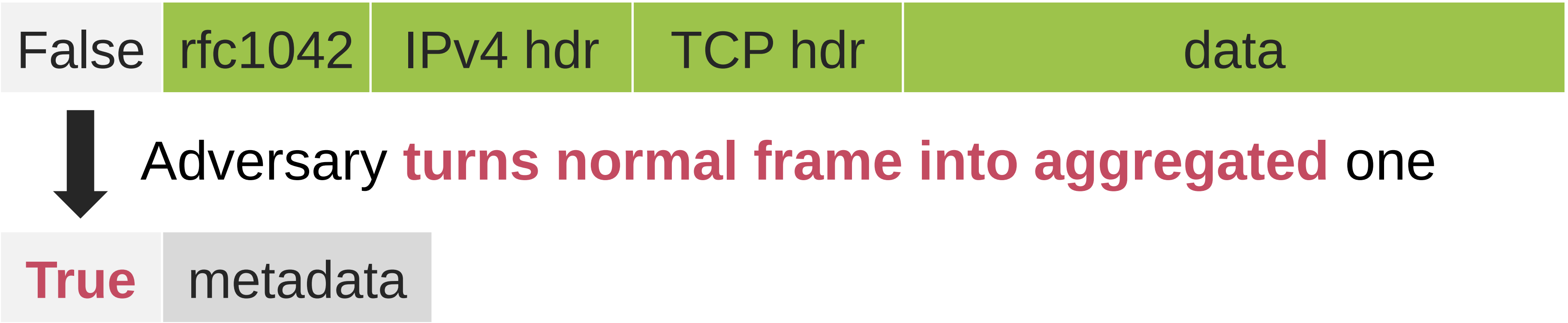
Spoofing aggregated frames

False	rfc1042	IPv4 hdr	TCP hdr	data

↓ Adversary **turns normal frame into aggregated** one

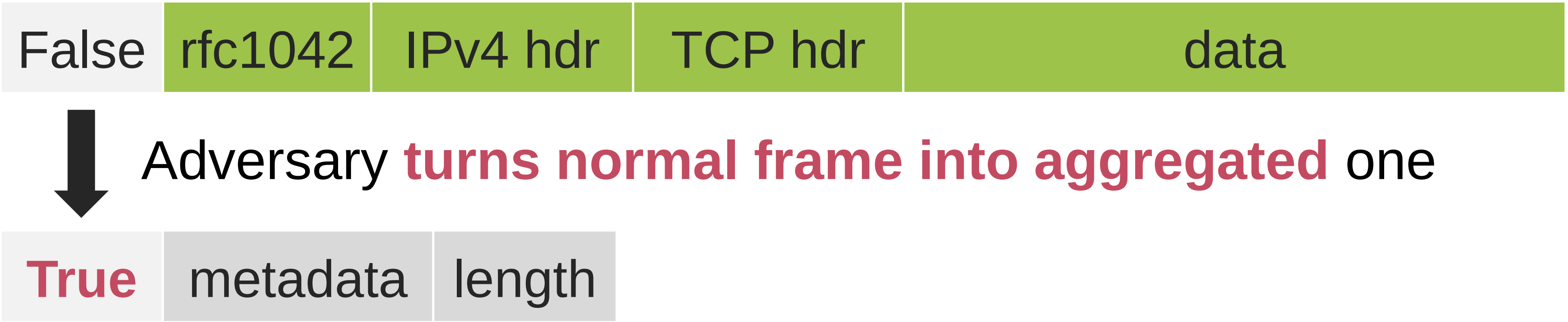
True

Spoofing aggregated frames



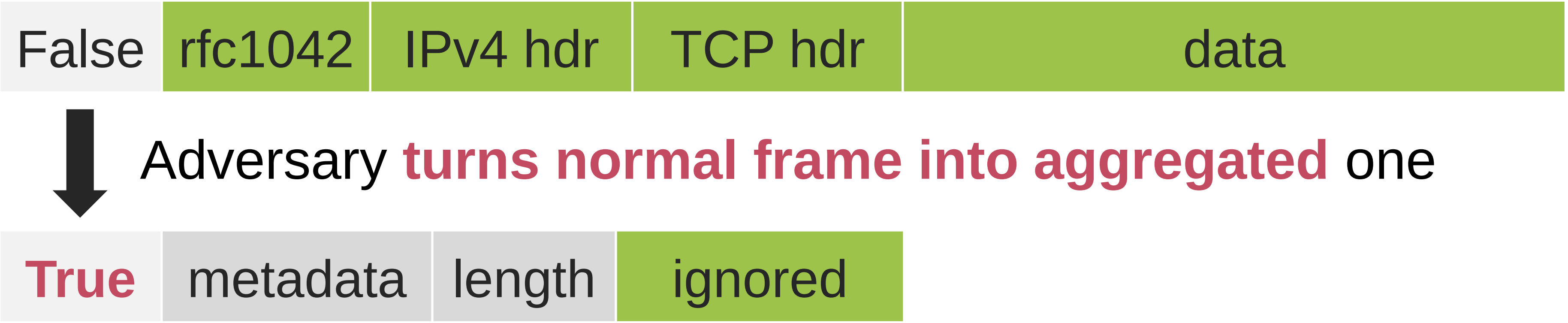
- Metadata = rfc1042 & IPv4 hdr → 1st sub-packet is ignored

Spoofing aggregated frames



- Metadata = rfc1042 & IPv4 hdr → 1st sub-packet is ignored
- Length = IPv4 ID field: determines start of 2nd sub-packet

Spoofing aggregated frames



- Metadata = rfc1042 & IPv4 hdr → 1st sub-packet is ignored
- Length = IPv4 ID field: determines start of 2nd sub-packet

Spoofing aggregated frames

False	rfc1042	IPv4 hdr	TCP hdr	data

↓ Adversary **turns normal frame into aggregated** one

True	metadata	length	ignored	metadata	length	packet

- Metadata = rfc1042 & IPv4 hdr → 1st sub-packet is ignored
- Length = IPv4 ID field: determines start of 2nd sub-packet
- Control IP ID & part of TCP data → **inject arbitrary packets**

How to control IPv4 ID and TCP data?

Send IPv4 packets to the victim

- Client: if there's a firewall, trick client to visit our website
  - Allows adversary to send IP/TCP packets, not any others
- AP: can attack AP if a client uses predictable IP IDs

All major operating systems affected

- 802.11n mandates support for aggregated frames
- Excluding Net/OpenBSD and certain IoT devices

Recap: can inject arbitrary packets

1. Inject special IPv4 frame to the victim
2. Adversary will intercept the corresponding Wi-Fi frame, **set the aggregated flag**, and forward it to the victim

Example attacks:

- Inject IPv6 Router Advertisement with our DNS server
- Victim will use our DNS server → **intercept IP-based traffic**
- Against IPv4-only clients we performed a port scan

Mixed Key Attack

CVE-2020-24587

Background

Large frames have a high chance of being corrupted:

Background

Large frames have a high chance of being corrupted:

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

Problem: **how to (securely) reassemble** the fragments?

Reassembling plaintext fragments

header	fragment1
header	fragment2
header	fragment3

Reassembling plaintext fragments

header	s	fragment1
header	s	fragment2
header	s	fragment3

- All Wi-Fi frames have an incremental sequence number s
- Fragments of a frame have the **same sequence number** s

Reassembling plaintext fragments

header	s	0	fragment1
header	s	1	fragment2
header	s	2	fragment3

- All Wi-Fi frames have an incremental sequence number s
- Fragments of a frame have the **same sequence number s**
- All fragments also have a **fragment number** ...

Reassembling plaintext fragments

header	s	0	More	fragment1
header	s	1	More	fragment2
header	s	2	Last	fragment3

- All Wi-Fi frames have an incremental sequence number s
- Fragments of a frame have the **same sequence number s**
- All fragments also have a **fragment number** ...

... and a flag to **identify the last** fragment

Reassembling encrypted fragments

header	s	n	0	More	fragment1
header	s	$n+1$	1	More	fragment2
header	s	$n+2$	2	Last	fragment3

- All encrypted frames have a packet number to detect replays
  - Cannot reuse sequence number (not unique & too small)

Reassembling encrypted fragments

header	s	n	0	More	fragment1
header	s	$n+1$	1	More	fragment2
header	s	$n+2$	2	Last	fragment3

Authenticated   Authenticated

- All encrypted frames have a packet number to detect replays
  - Cannot reuse sequence number (not unique & too small)
- Everything except sequence number is authenticated
- Drop if packet & fragment numbers are not consecutive

Problem: key renewal

› Session key can be periodically renewed ...

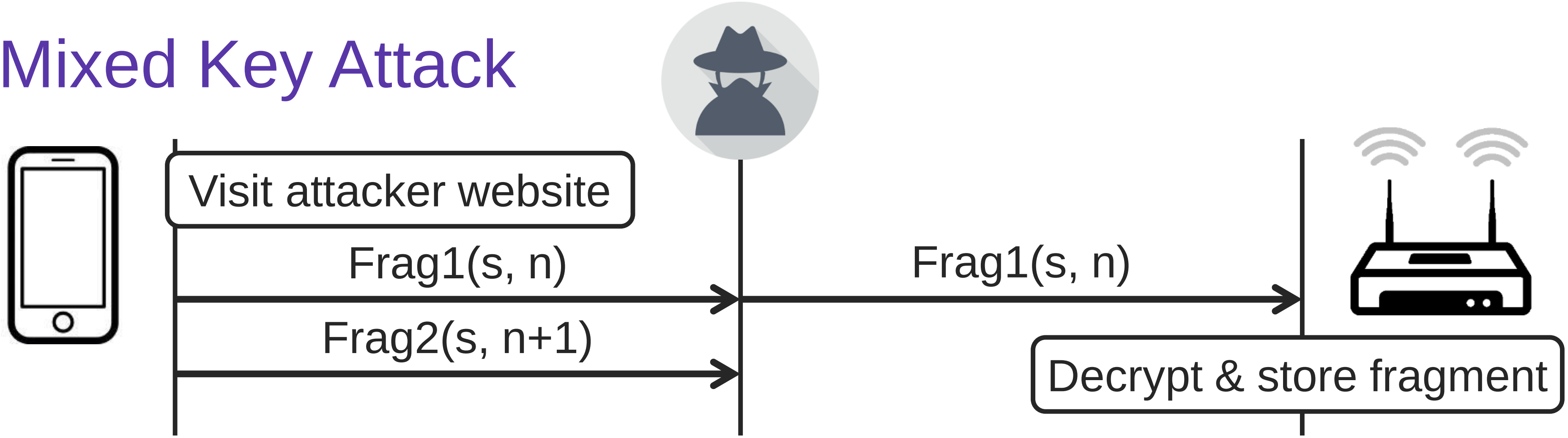
› ... or updated when roaming between APs

› Receiver is allowed to **reassemble fragments encrypted under different keys** (i.e. mixed keys)

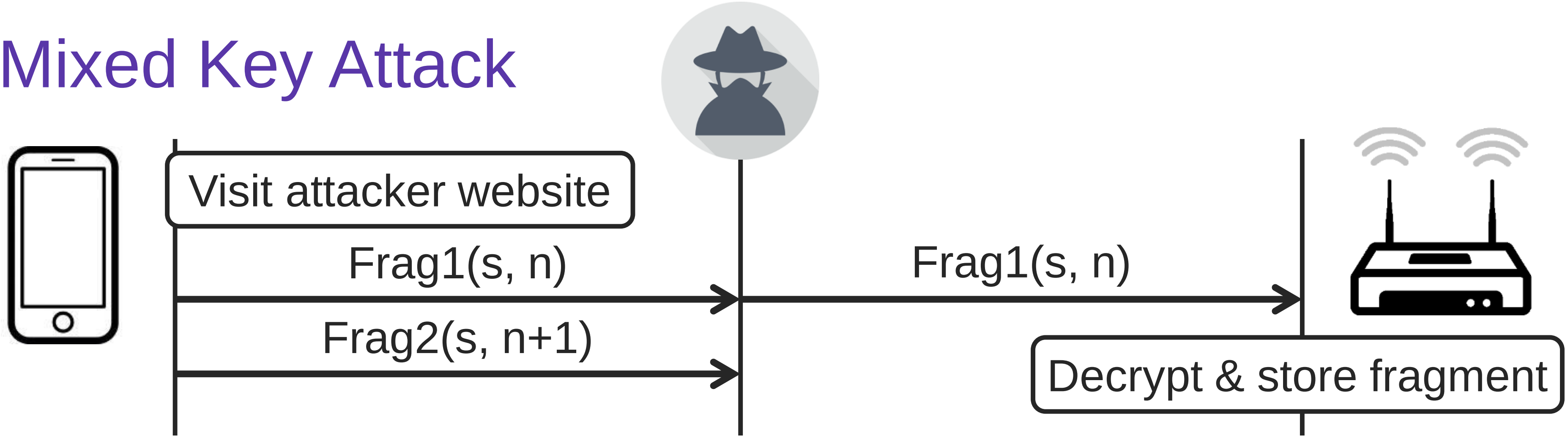
› Attacker can trick victim to combine fragments (from two different frames) encrypted under mixed keys

› Can be abused to **exfiltrate data** under specific conditions

Mixed Key Attack

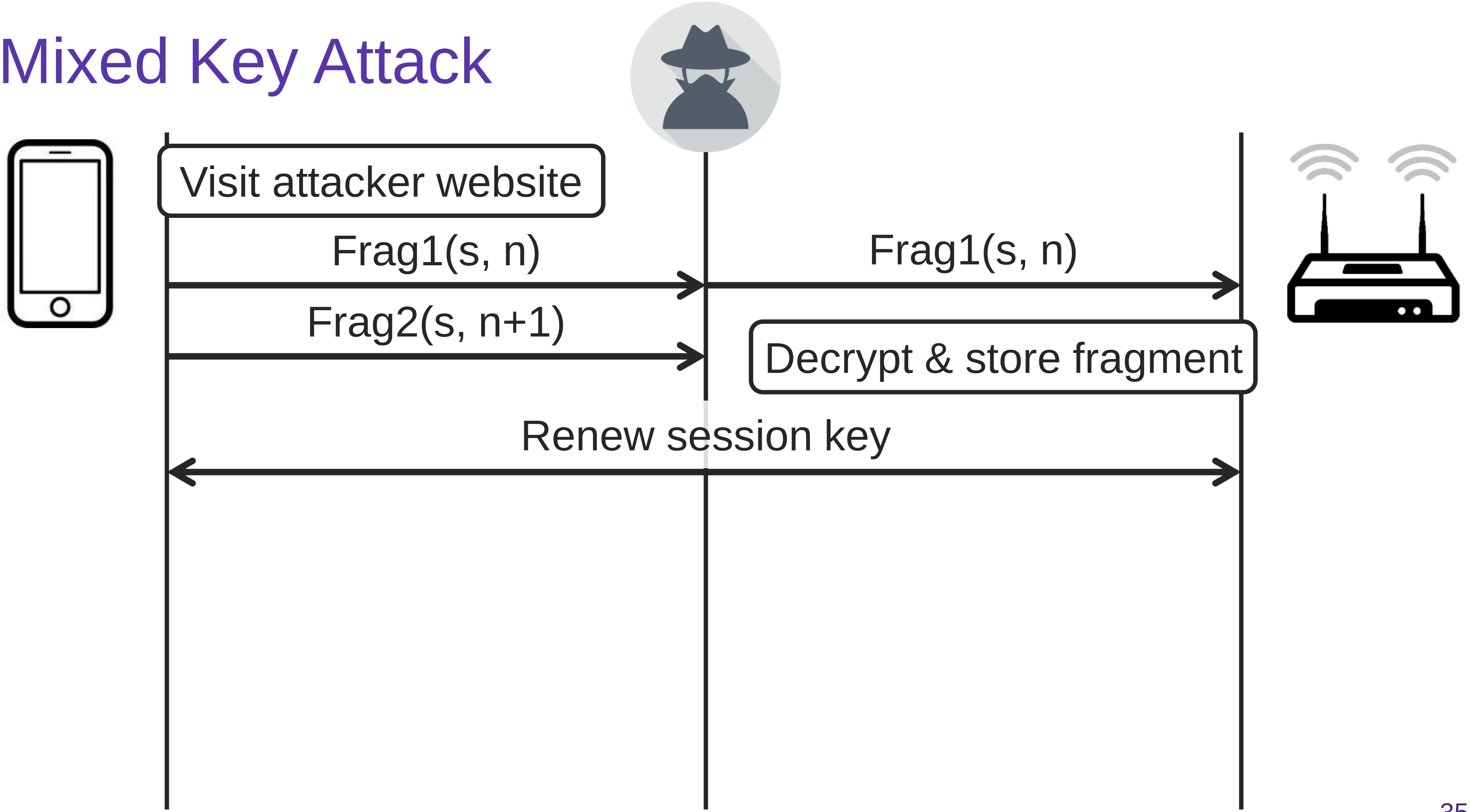


Mixed Key Attack

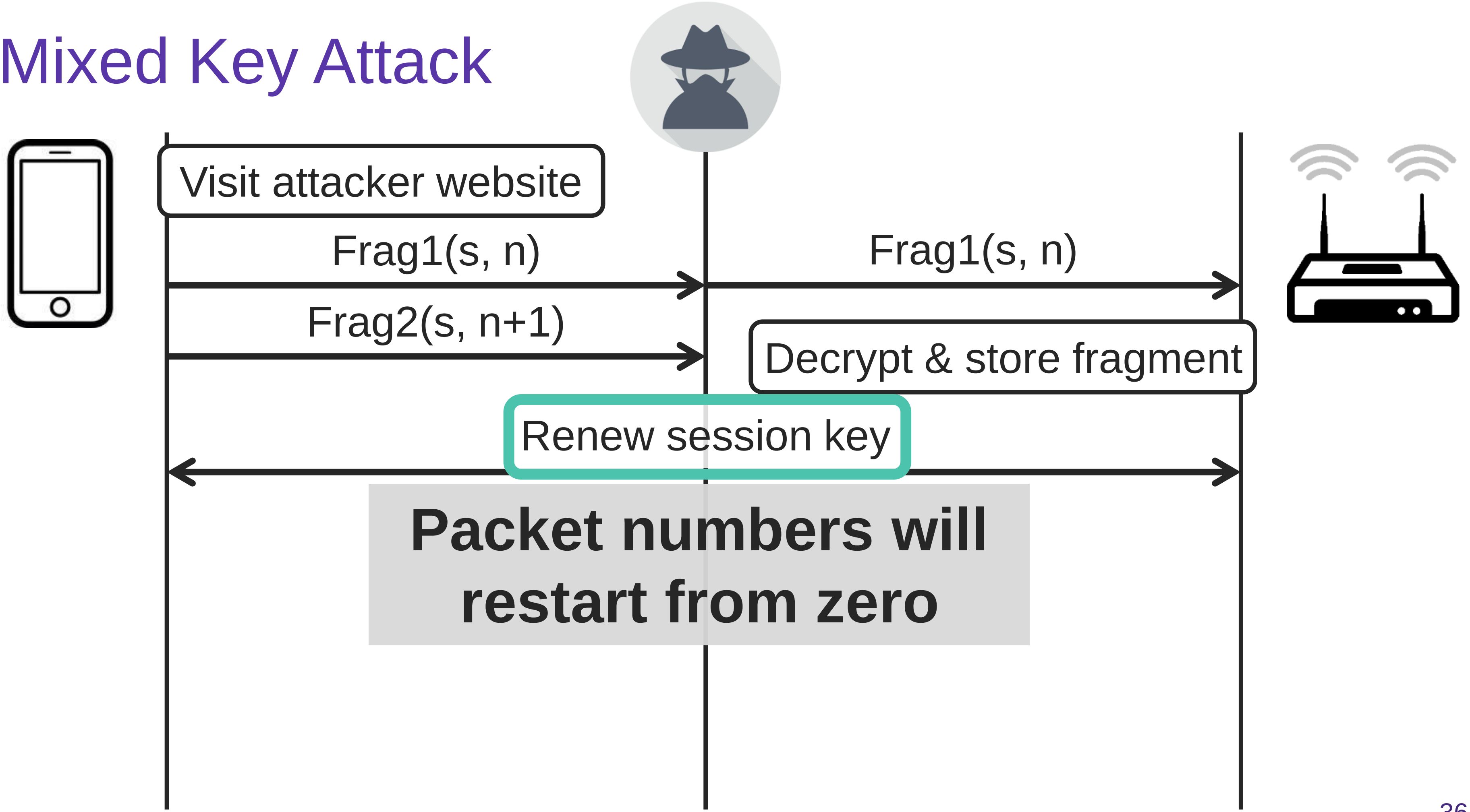


Header (Frag1)	Payload (Frag2)
192.168.1.2 to 3.5.1.1	GET /image.png HTTP/1.1

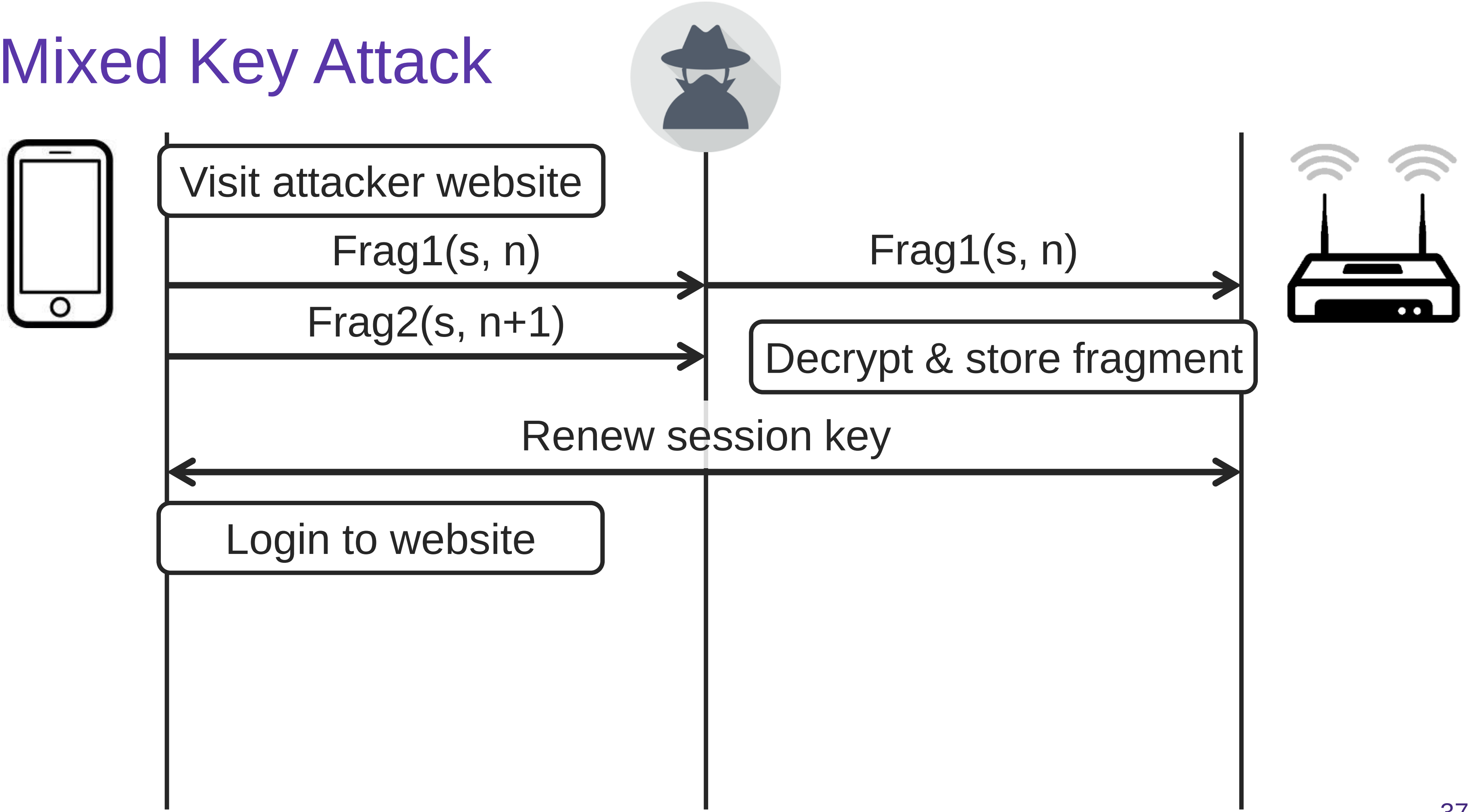
Mixed Key Attack



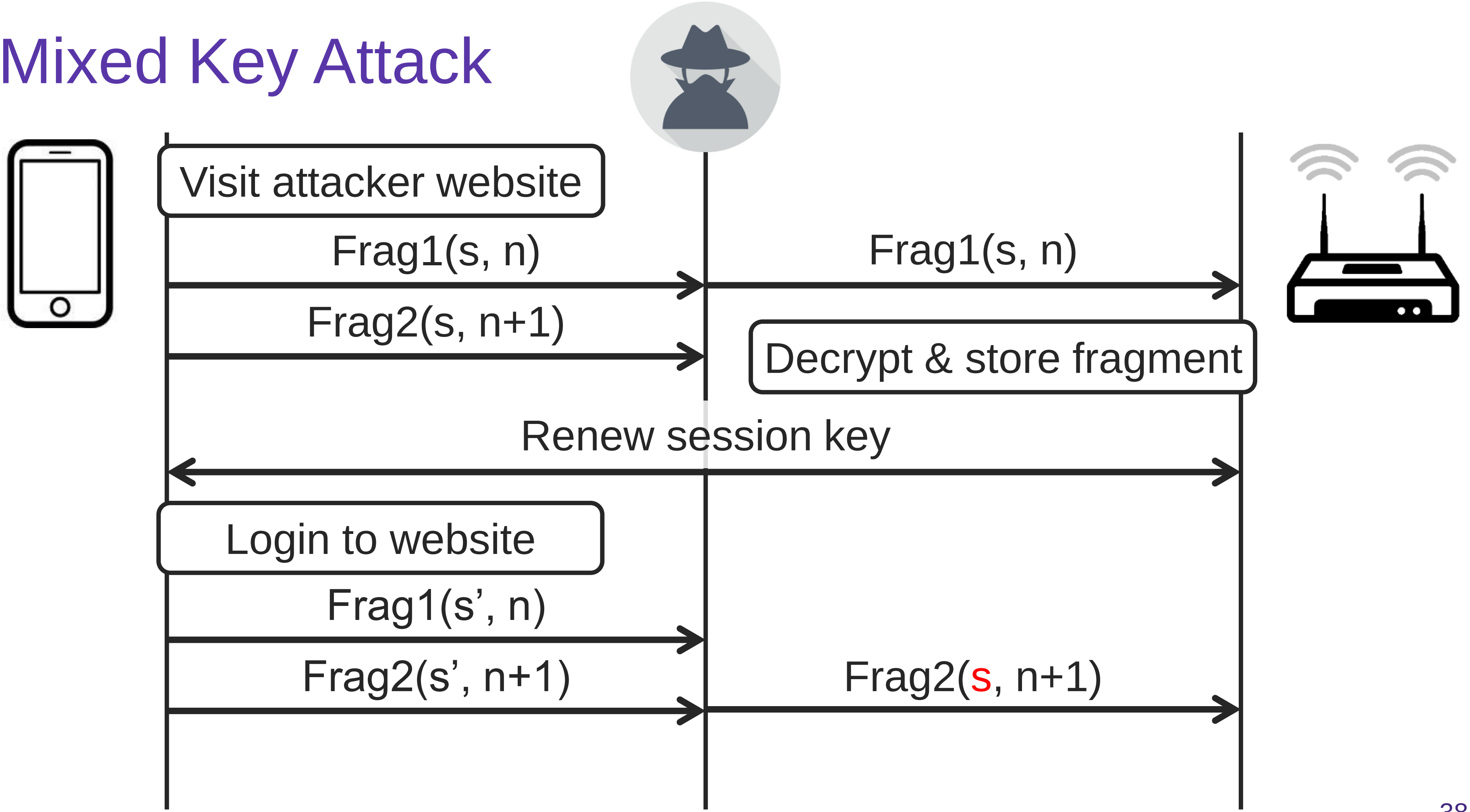
Mixed Key Attack



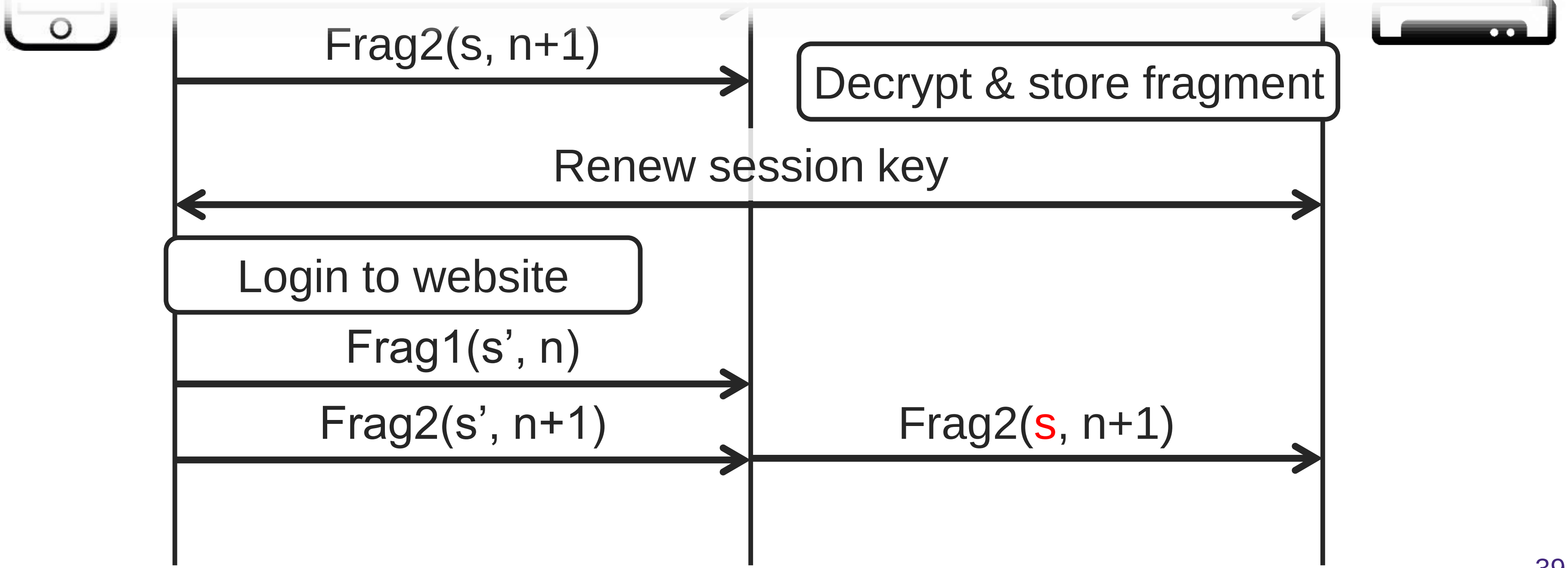
Mixed Key Attack



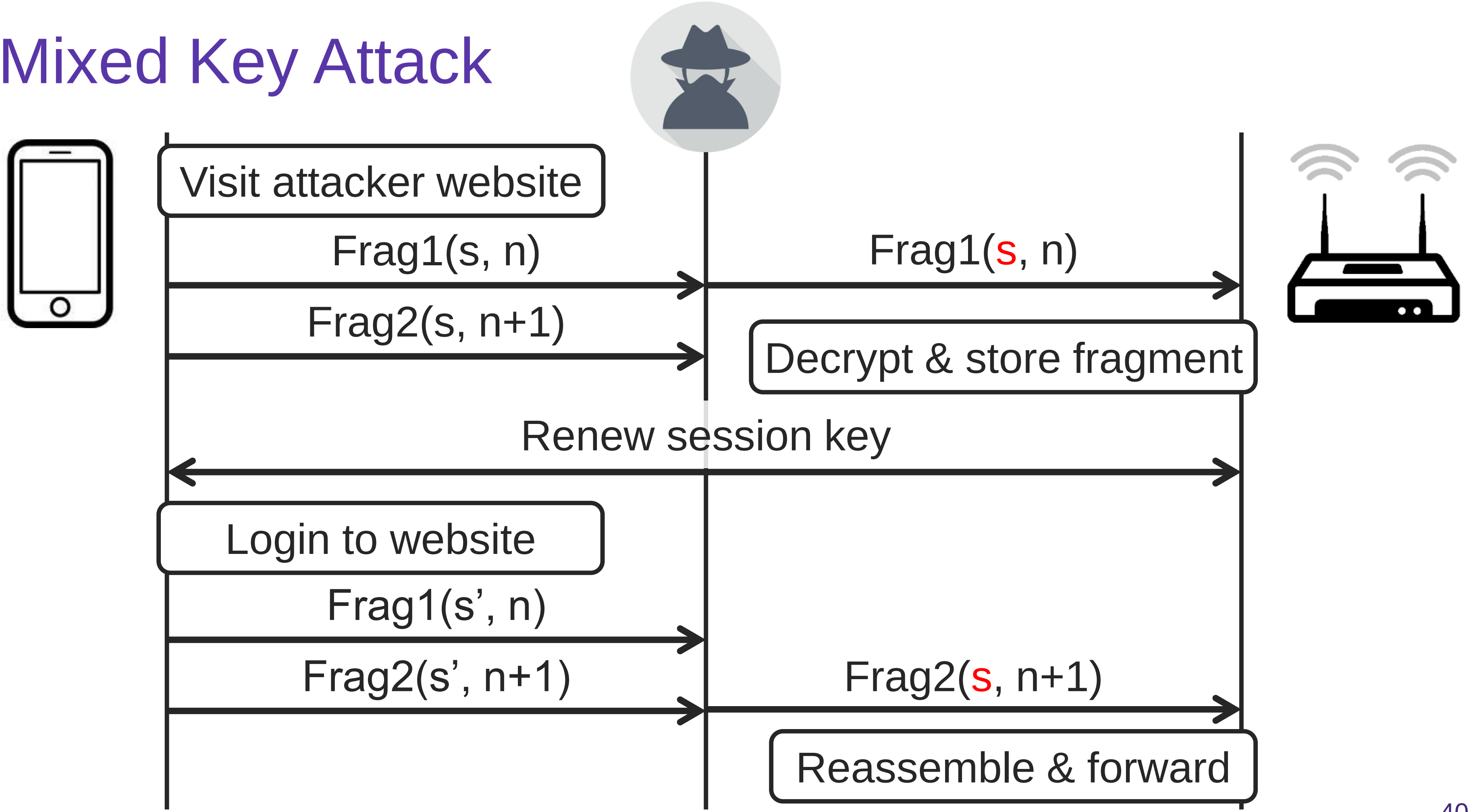
Mixed Key Attack



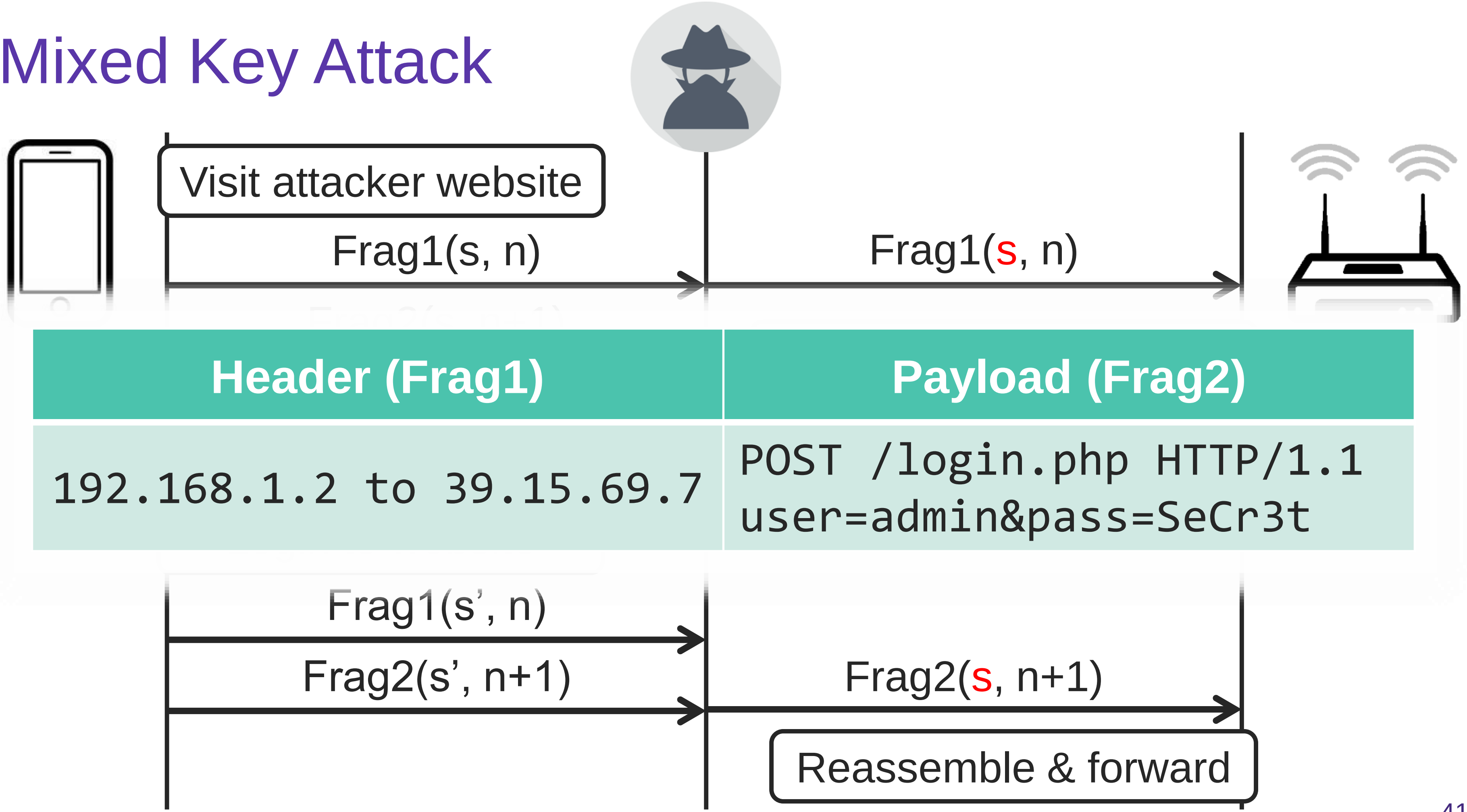
Header (Frag1)	Payload (Frag2)
192.168.1.2 to 39.15.69.7	POST /login.php HTTP/1.1 user=admin&pass=SeCr3t



Mixed Key Attack



Mixed Key Attack



Header (Frag1)	Payload (Frag2)
192.168.1.2 to 39.15.69.7	POST /login.php HTTP/1.1 user=admin&pass=SeCr3t

Data exfiltration

Header (Frag1)	Payload (Frag2)
`192.168.1.2 to 3.5.1.1`	`GET /image.png HTTP/1.1`

Data exfiltration

Header (Frag1)	Payload (Frag2)
192.168.1.2 to 3.5.1.1	GET /image.png HTTP/1.1
192.168.1.2 to 39.15.69.7	POST /login.php HTTP/1.1 user=admin&pass=SeCr3t

Data exfiltration

Header (Frag1)	Payload (Frag2)
`192.168.1.2 to 3.5.1.1`	`GET /image.png HTTP/1.1`
`192.168.1.2 to 39.15.69.7`	`POST /login.php HTTP/1.1` `user=admin&pass=SeCr3t`

⬇ Adversary **mixes different fragments**

`192.168.1.2 to 3.5.1.1`	`POST /login.php HTTP/1.1` `user=admin&pass=SeCr3t`

Data exfiltration

Header (Frag1)	Payload (Frag2)
`192.168.1.2 to 3.5.1.1`	`GET /image.png HTTP/1.1`
`192.168.1.2 to 39.15.69.7`	`POST /login.php HTTP/1.1` `user=admin&pass=SeCr3t`

⬇ Adversary **mixes different fragments**

`192.168.1.2 to 3.5.1.1`	`POST /login.php HTTP/1.1` `user=admin&pass=SeCr3t`

→ Login **info is sent to attacker's server**

Experiments

All major operating systems are vulnerable

- Specifics may depend on driver being used
- All four tested home routers affected (Asus, Linksys, D-Link)
- One out of three professional APs affected (LANCOM)

Practically all clients are vulnerable

- Data exfiltration not possible when exploiting a client
- Possible attacks depend on target (non-trivial, see paper)

Practicality

Non-trivial exploitation requirements:

1. In our example AP must be vulnerable
2. Client must send fragmented frames
3. Session key is periodically refreshed
4. Client sends IP packets to attacker’s server

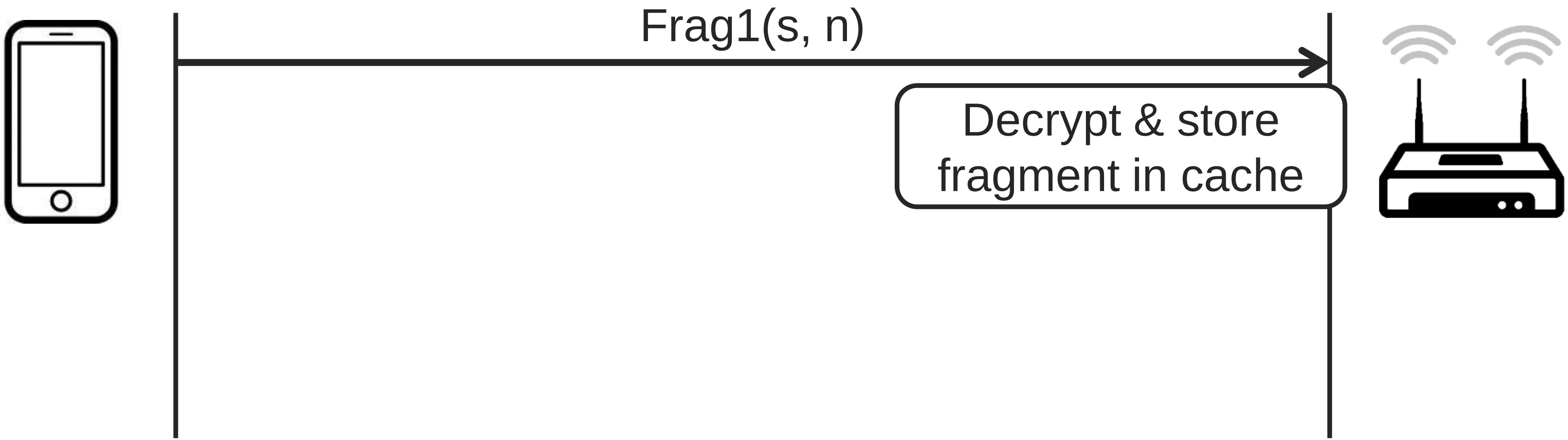
When combined with **implementation bugs**,
this attack becomes **more practical**

Fragment Cache Attack

CVE-2020-24586

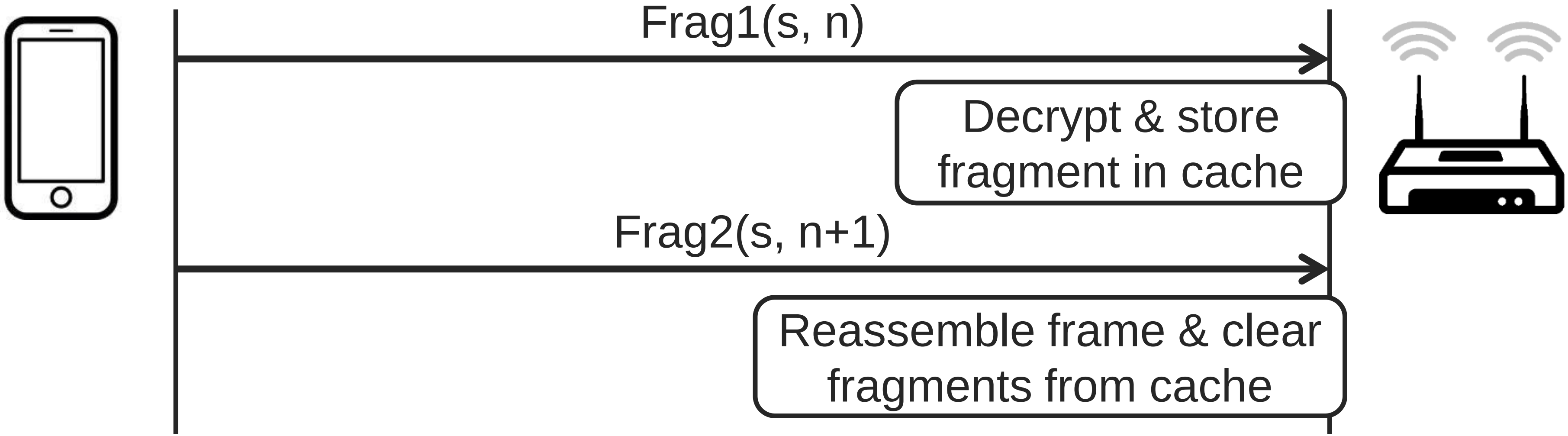
Background: fragment cache

Incomplete fragments are stored in memory = **fragment cache**



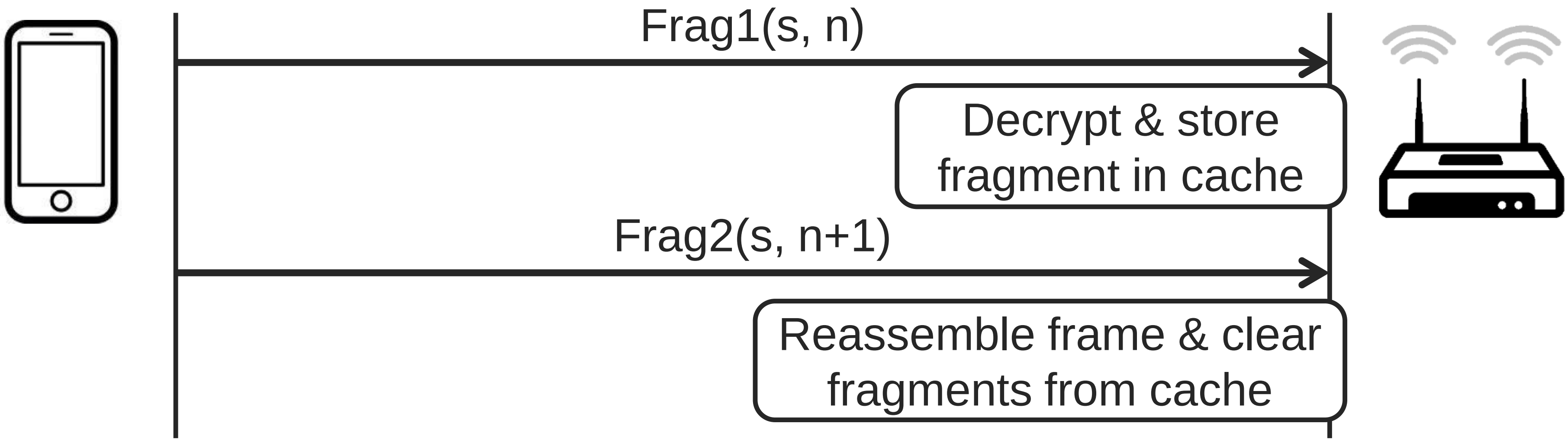
Background: fragment cache

Incomplete fragments are stored in memory = **fragment cache**



Background: fragment cache

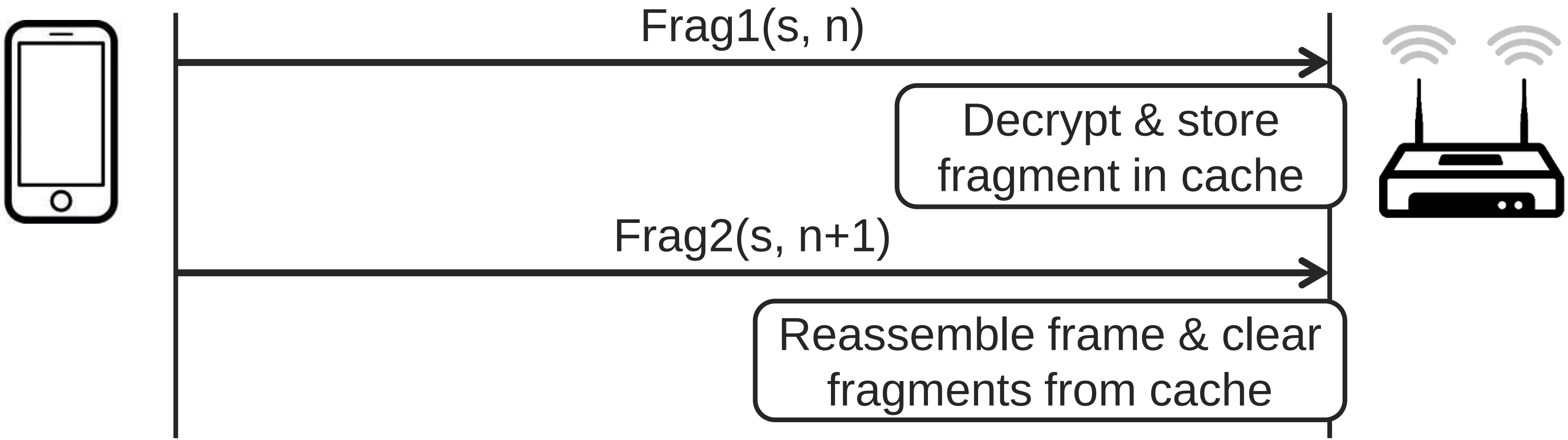
Incomplete fragments are stored in memory = **fragment cache**



Problem: **fragment cache isn't cleared** when (re)connecting to a different network

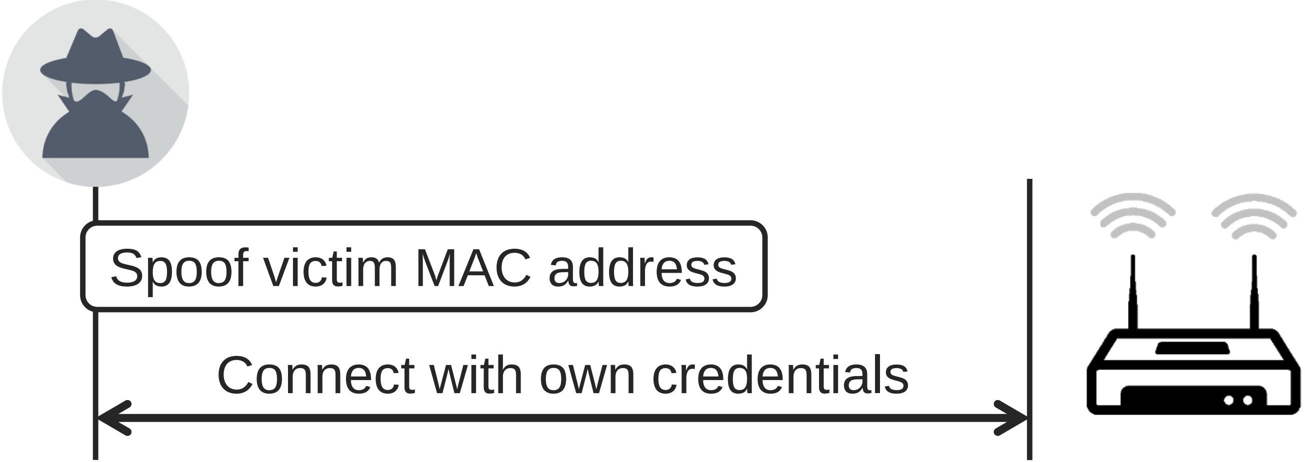
Background: fragment cache

Incomplete fragments are stored in memory = **fragment cache**

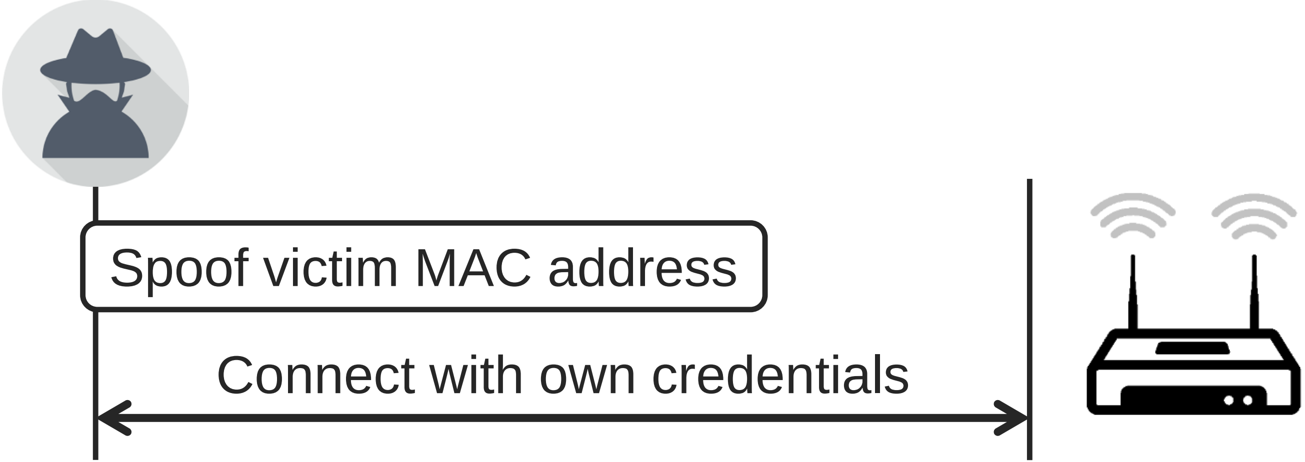


→ Can be **abused to exfiltrate** (or inject) data

Cache Attack



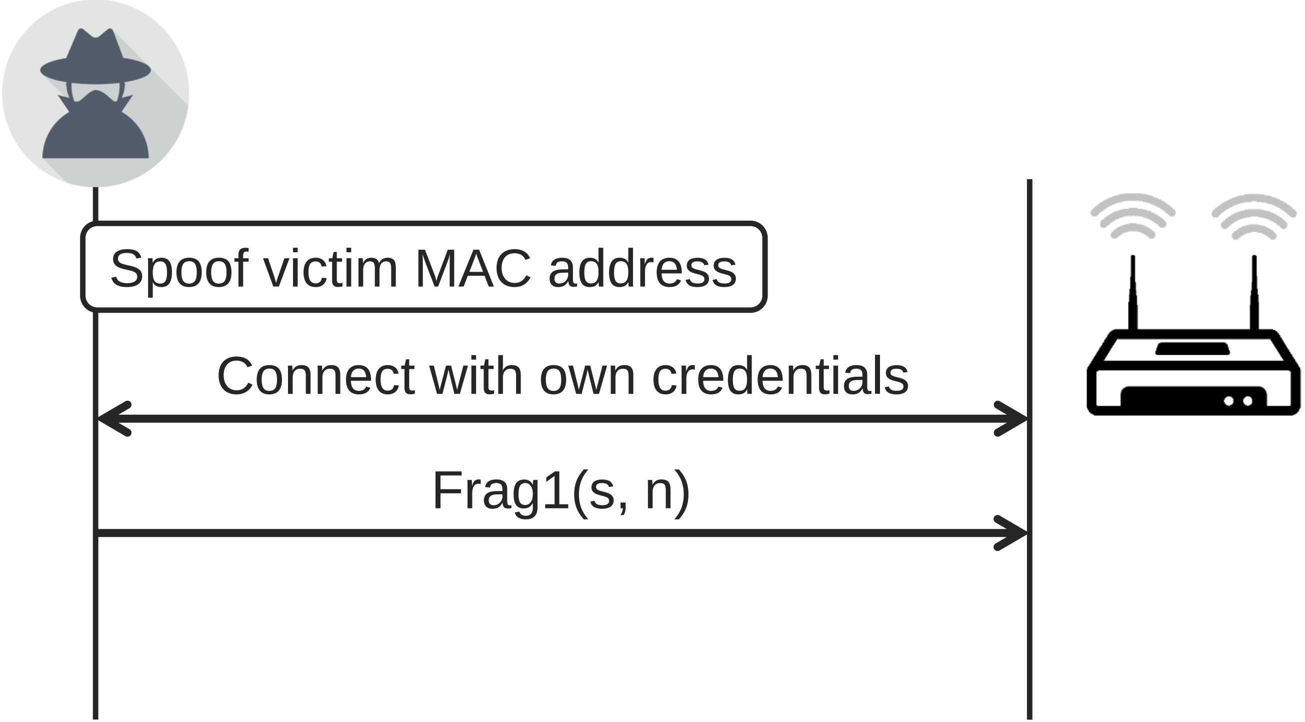
Cache Attack



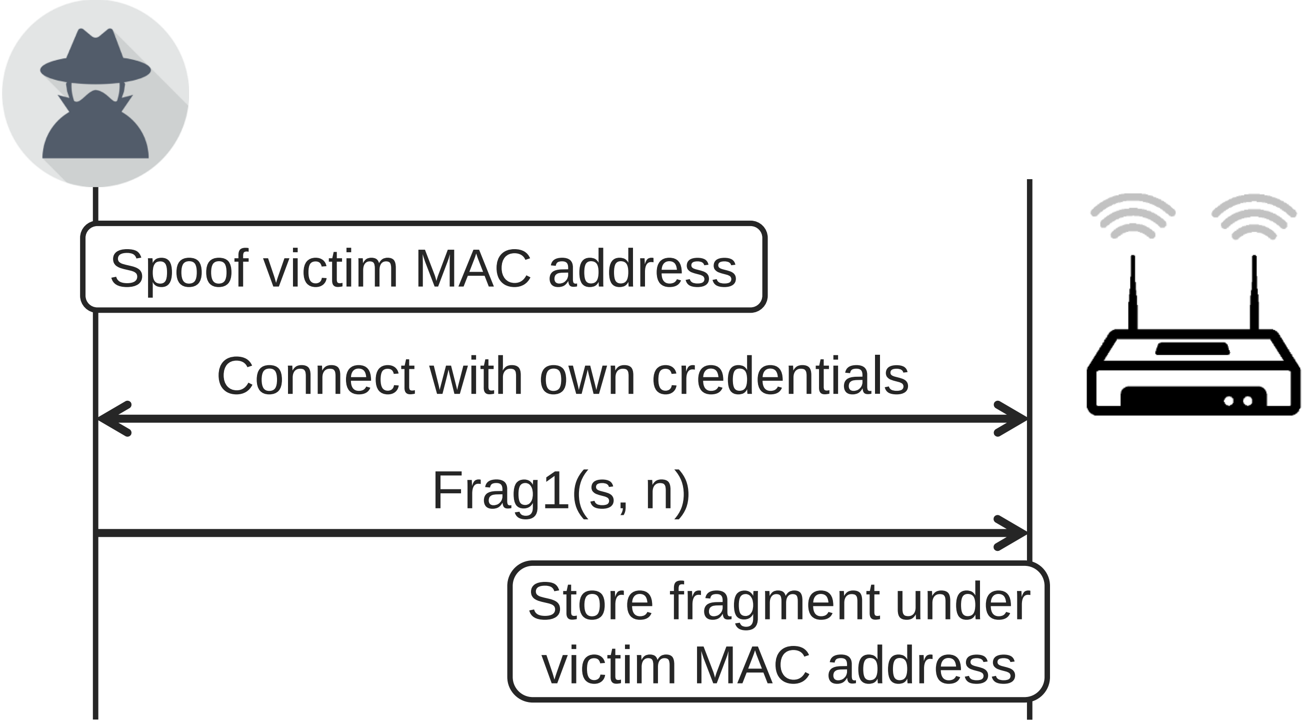
Target is an **enterprise (hotspot) network**

- Users don't trust each other
- Adversary has valid credentials

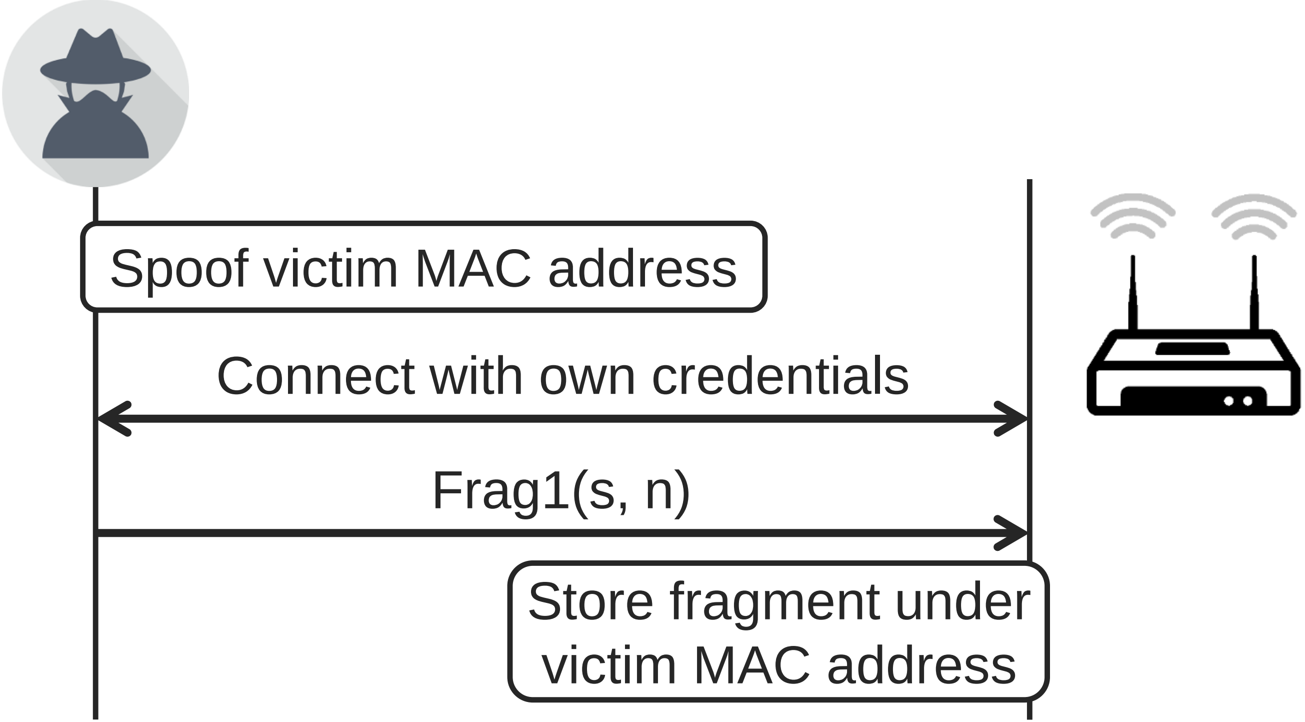
Cache Attack



Cache Attack

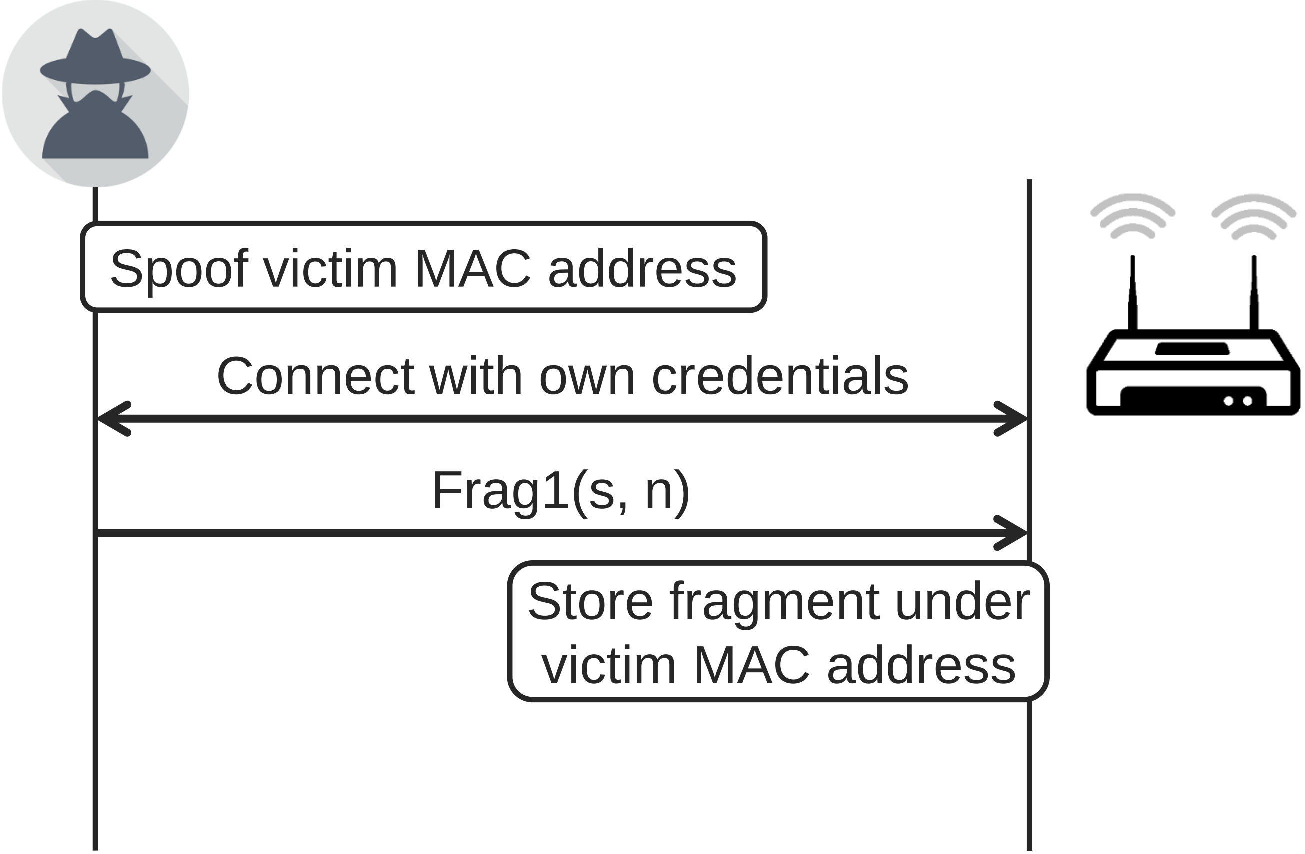


Cache Attack

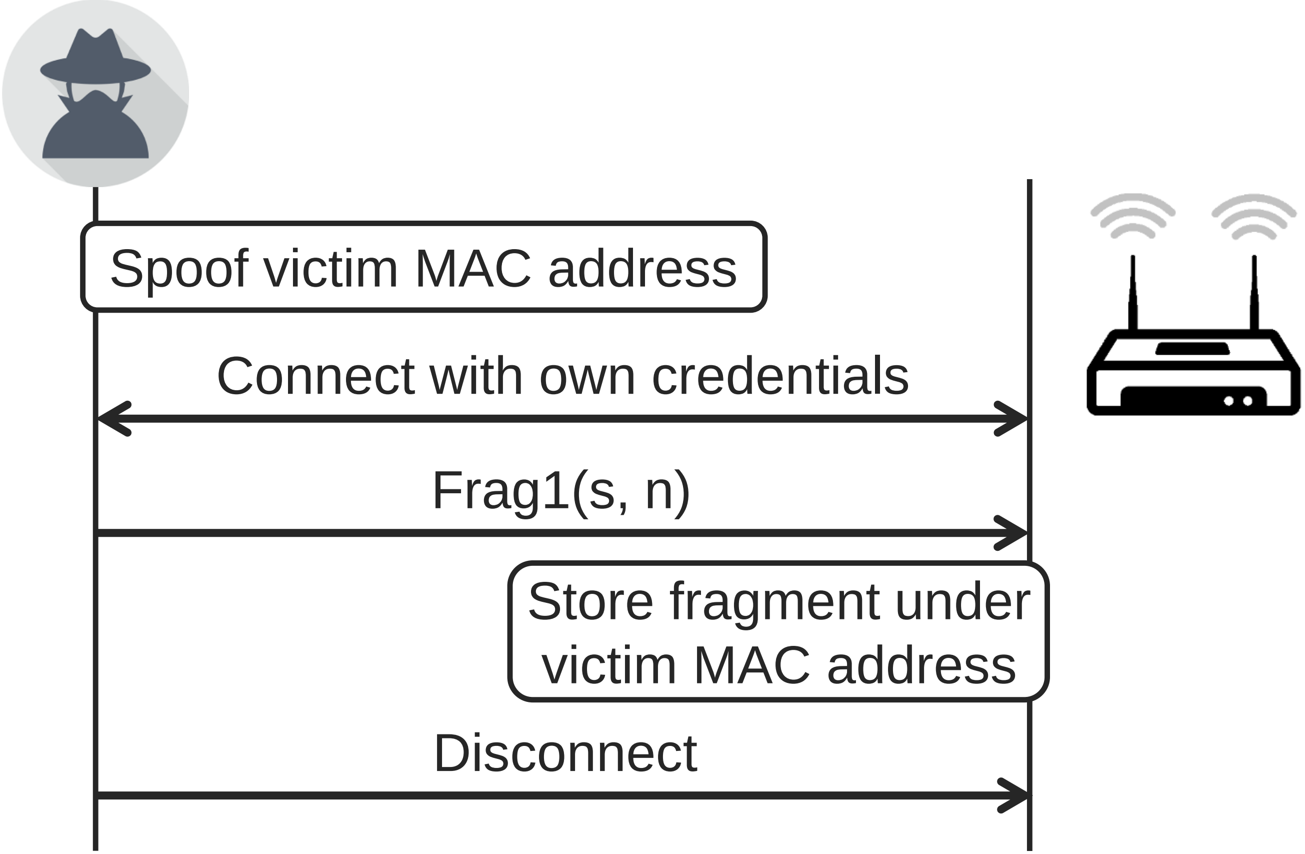


Header (Frag1)	Payload (Frag2)
`192.168.1.2 to 3.5.1.1`	

Cache Attack

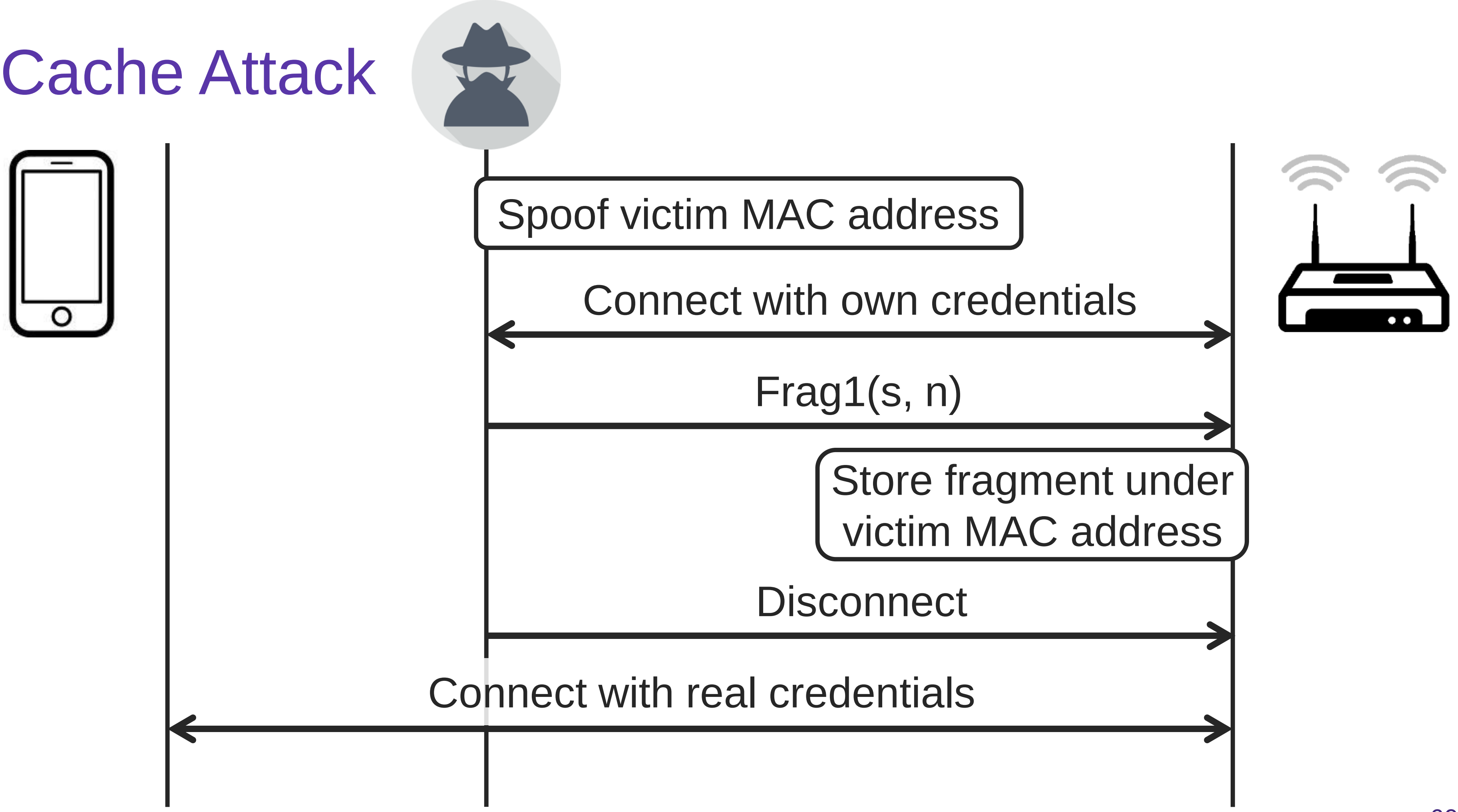


Cache Attack

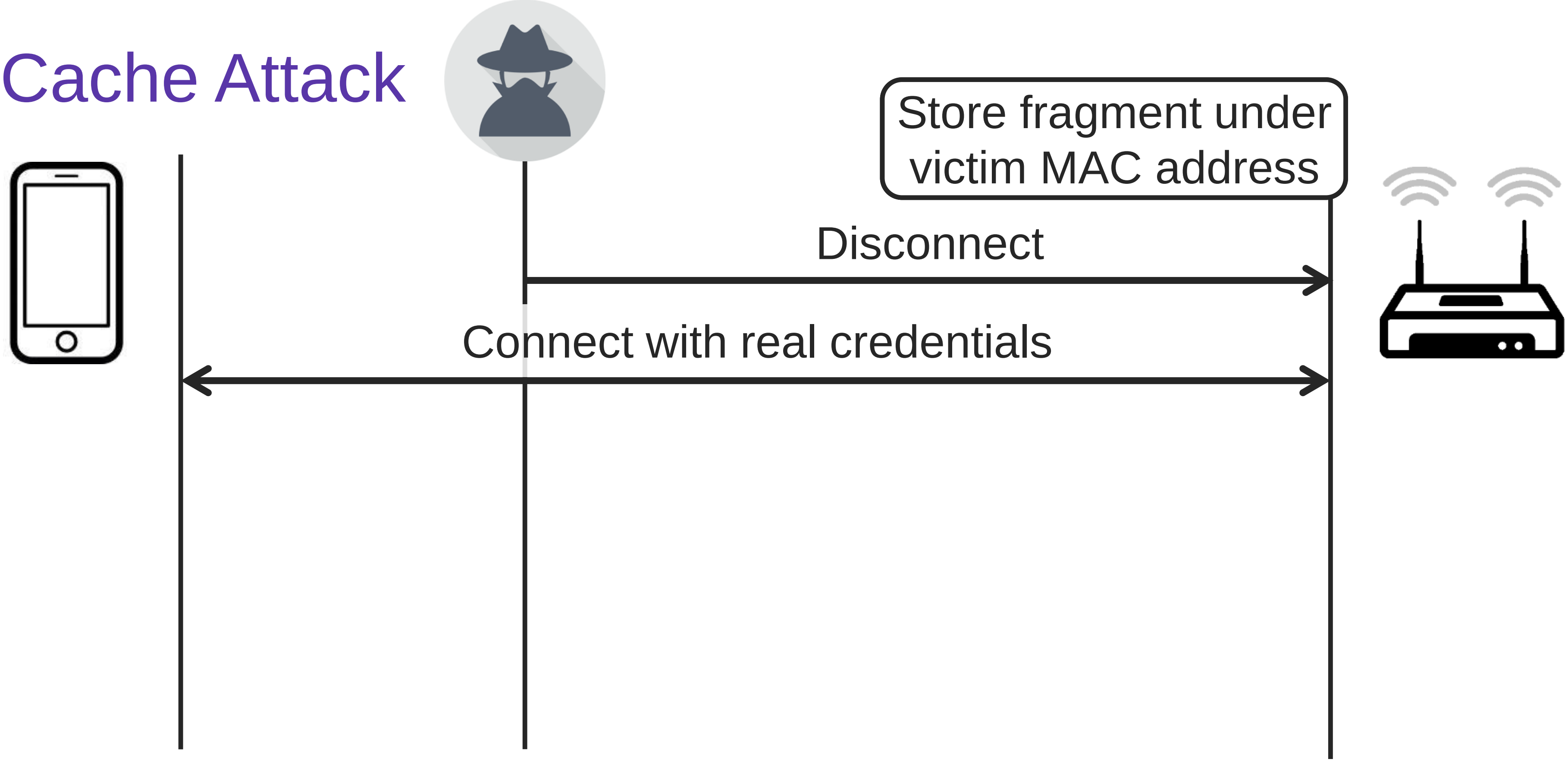


Vulnerability: AP **won't clear fragment** from it's memory

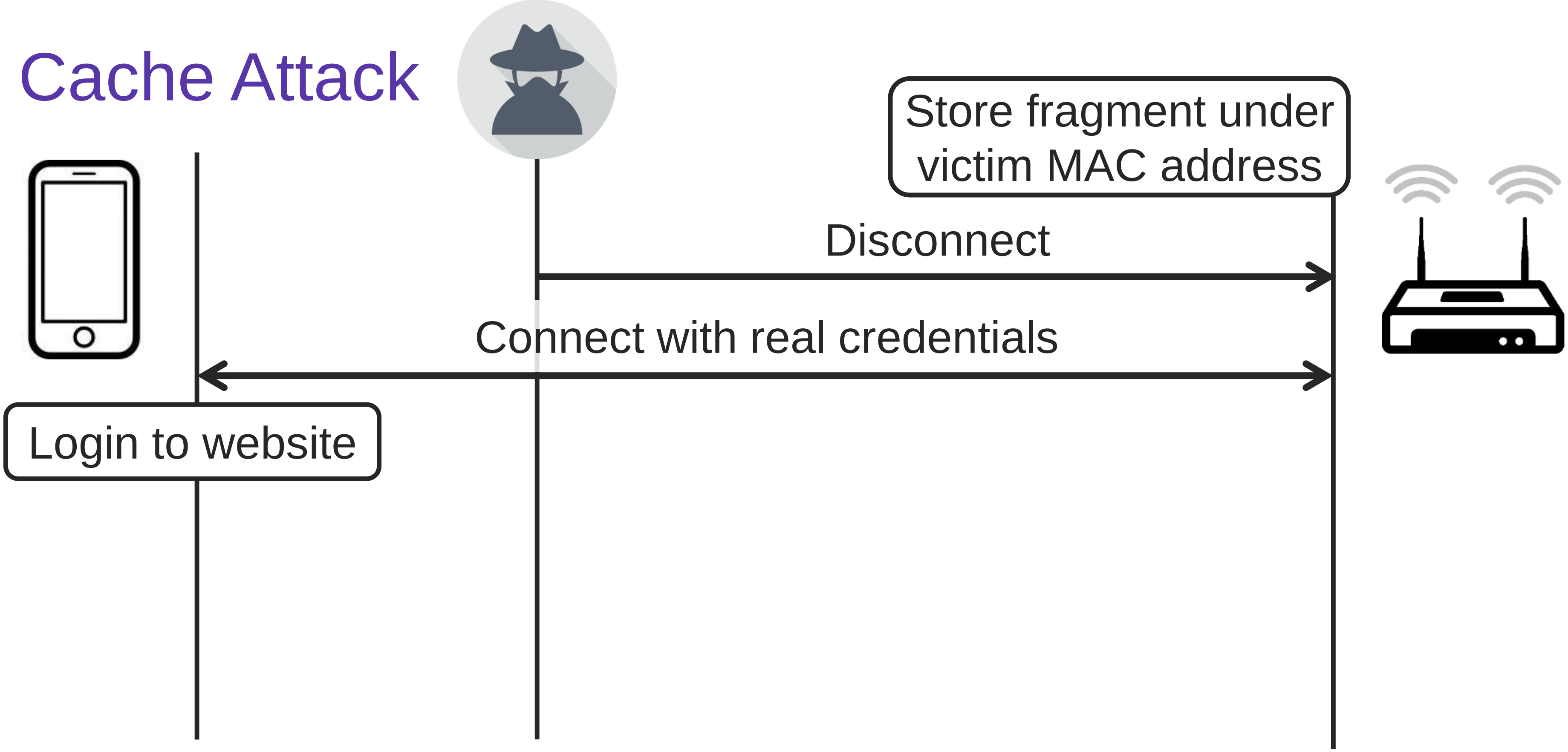
Cache Attack



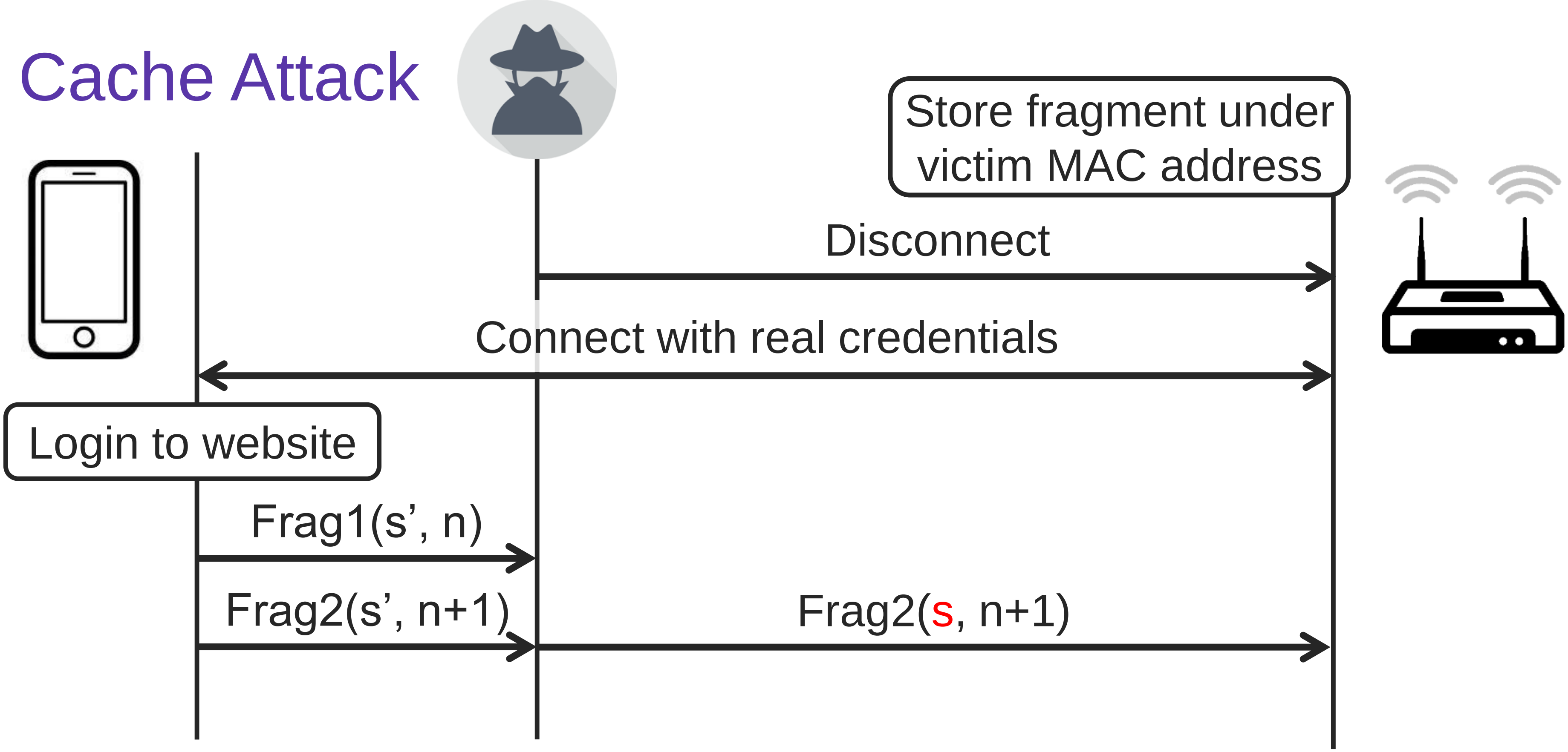
Cache Attack



Cache Attack

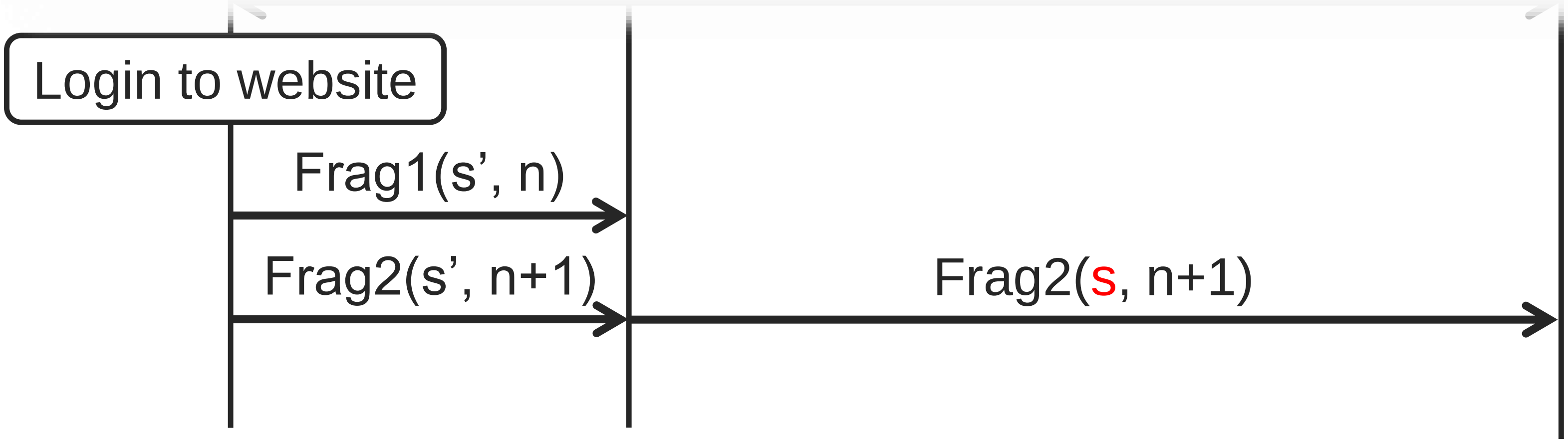


Cache Attack

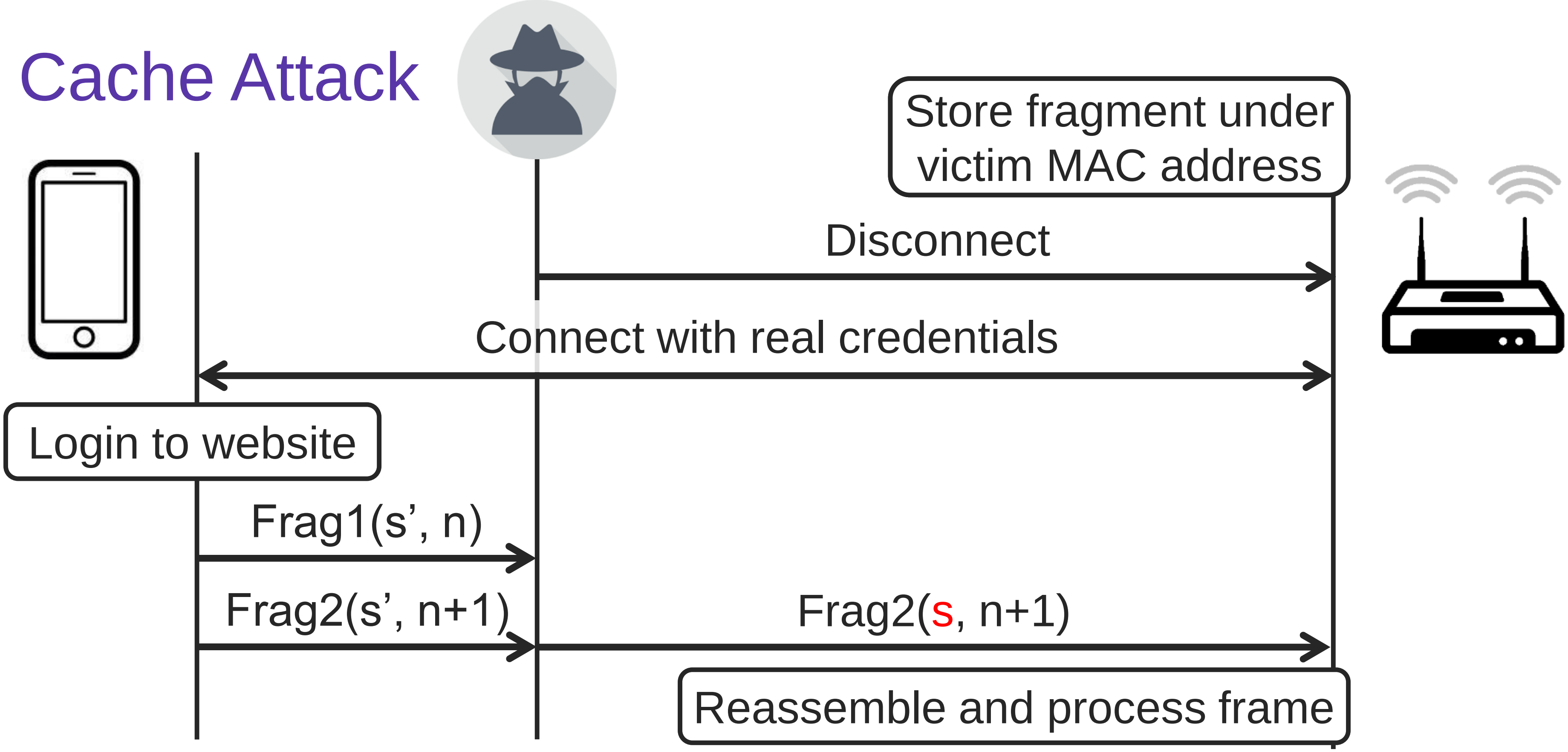


Cache Attack

Header (Frag1)	Payload (Frag2)
	POST /login.php HTTP/1.1 user=admin&pass=SeCr3t

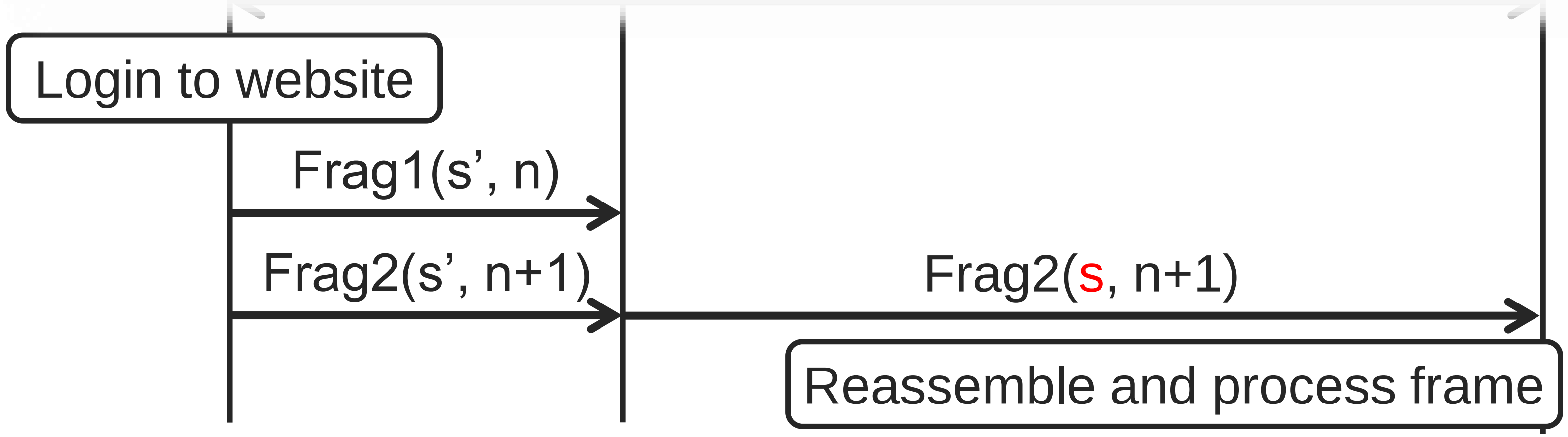


Cache Attack



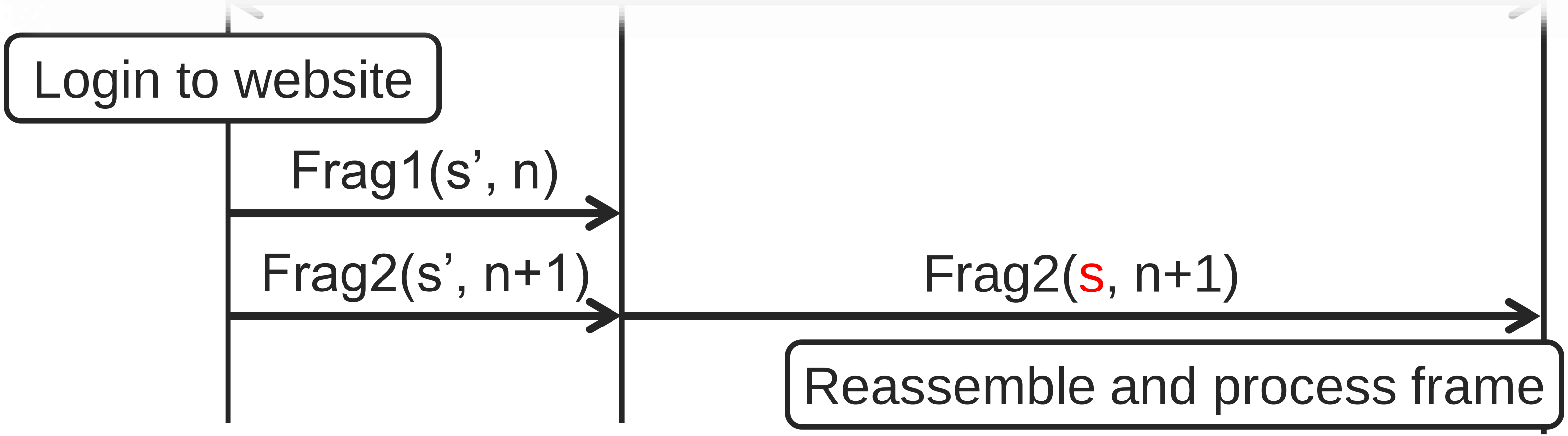
Cache Attack

Header (Frag1)	Payload (Frag2)
192.168.1.2 to 3.5.1.1	POST /login.php HTTP/1.1 user=admin&pass=SeCr3t



Cache Attack

Header (Frag1)	Payload (Frag2)
192.168.1.2 to 3.5.1.1	POST /login.php HTTP/1.1 user=admin&pass=SeCr3t



→ Login data in Frag2 is now **sent to the attacker**

Experiments

Roughly **half of all tested devices** are vulnerable

- Seems to depend on driver & network card
- My four home routes were all affected
- None of the three professional APs were affected

Also possible to exploit clients under non-trivial threat model

- Can inject packets towards the client
- See paper for details

Recap: attacker can exfilrate/inject packets

Fragments aren't cleared from memory after (re)connecting

- Adversary can poison the cache

Threat model and impact:

- Exploiting an AP: against enterprise network, client sends fragmented frames, can **exfiltrate client data**
- Exploiting a client: non-trivial threat model, AP send fragmented frames, can **inject packets towards the client**

Implementation Flaws: trivial plaintext injection

Accepted plaintext frames (CVE-2020-26140 / 26143)

Accepting **plaintext frames** (CVE-2020-26140)

- Examples: some routers, some dongles on Linux/Windows

Accepting **fragmented plaintext frames** (CVE-2020-26143)

- Examples: many dongles on Windows, some FreeBSD APs

→ Can **inject frames** indepedent of network config

Plaintext broadcast fragments (CVE-2020-26145)

Some devices accept **plaintext broadcast fragments**

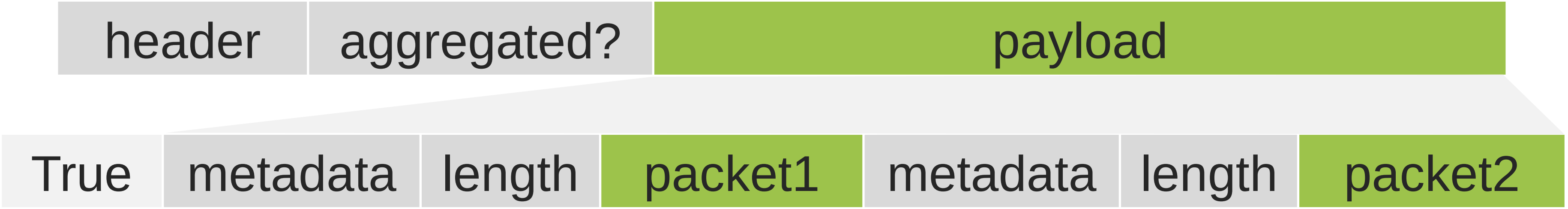
- Sometimes only accepted while connecting
- Treated as full frames!
- Examples: MacOS, iOS, and Free/NetBSD APs

→ Can **inject frames** indepedent of network config

Cloacked aggregated frames (CVE-2020-26144)

Implementations must accept plaintext EAPOL frames

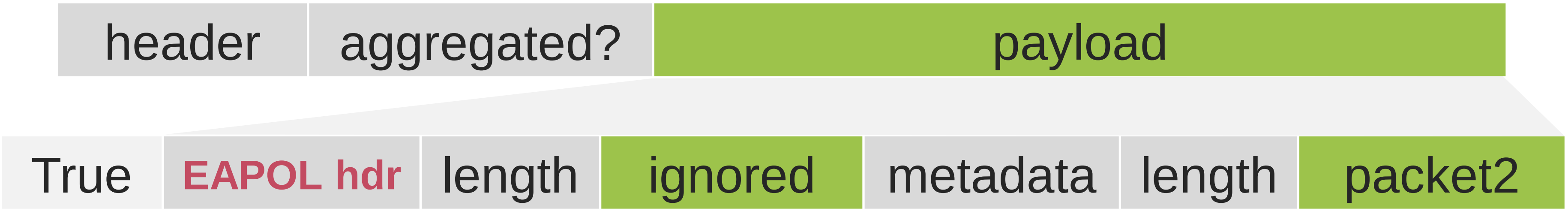
- Abuse to inject plaintext aggregated frames to low #devices:



Cloacked aggregated frames (CVE-2020-26144)

Implementations must accept plaintext EAPOL frames

- Abuse to inject plaintext aggregated frames to low #devices:



- Set metadata to start of EAPOL → plaintext frame is accepted

Cloacked aggregated frames (CVE-2020-26144)

Implementations must accept plaintext EAPOL frames

- Abuse to inject plaintext aggregated frames to low #devices:

- Set metadata to start of EAPOL → plaintext frame is accepted
- 1st sub-packet will be ignored, but **2nd one is processed**

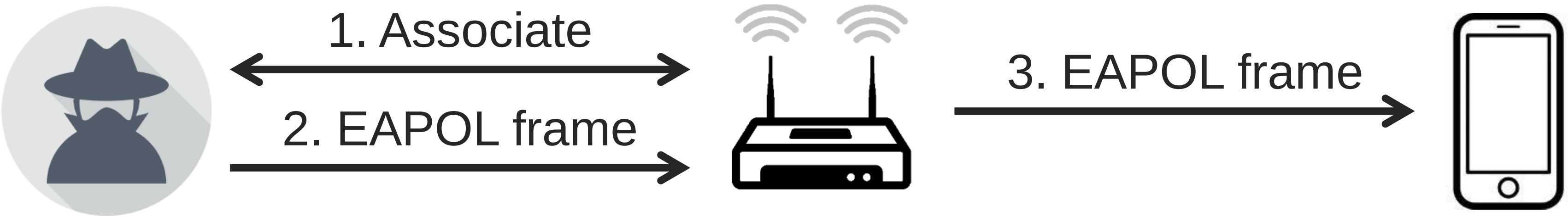
→ Can **trivially inject plaintext frames**

Implementation flaws with other impact

Pre-auth EAPOL forwarding (CVE-2020-26139)

Some APs forwards EAPOL frames before sender is authenticated

› Examples: Net/FreeBSD APs and $^2/_4$ home routers



→ Abuse to **inject frames** in combination with aggregation attack (CVE-2020-24588)

Non-consective packet numbers (CVE-2020-26146)

header	s	n	0	More	fragment1
header	s	$n+1$	1	More	fragment2
header	s	$n+2$	2	Last	fragment3

- Recap: fragments must have consecutive packet numbers
- **Almost nobody checks this!** Only Linux does.
- Can do mixed key "exfiltration" attack without periodic rekeys
  - Client must still use fragmentation & connect to attacker server

Mixed plain/encrypted fragments (CVE-2020-26147)

Many devices only require the **first fragment to be encrypted**

- Most Windows & Linux drivers, some Free/NetBSD drivers
- **Aggregation attack** possible without the victim needing to connect to the attacker's website (can inject packets)
- **Cache attack** against client possible when even when the AP doesn't send fragmented frames (can inject packets)

Some devices only require the **last fragment to be encrypted**

- Several Free/NetBSD drivers
- Trivial to **inject** data if fragmentation is used

No fragmentation support (CVE-2020-26142)

Some devices don't support fragmentation

- They **treat fragmented frames as full frames**
- Examples: OpenBSD and ESP12-F

Abuse to **inject frames** when:

- Another device sends fragmented frames
- This other device visits the attacker's server

Discussion

Practicality vs. impact

Perhaps we're lucky:

- Widespread flaws → relatively trickly to exploit in practice
- Trivial to exploit flaws → not widespread in practice (?)

Important concerns remain:

- Significant #devices affected by trivial to exploit flaws
- **Every Wi-Fi device affected** by one or more flaws
- Combining flaws increases practicality of certain attacks

→ **Patch now** before attack improve!

Conclusion

- Aggregation & fragmentation **design flaws**
- Several common **implementation flaws**
- Need **patched drivers** to test if affected
- **Impact varies** per device & network