Framing Frames: Bypassing Wi-Fi Encryption by Manipulating Transmit Queues

D. Schepers, A. Ranganathan, and **M. Vanhoef**

Real-World-Crypto 2023, Tokyo, Japan

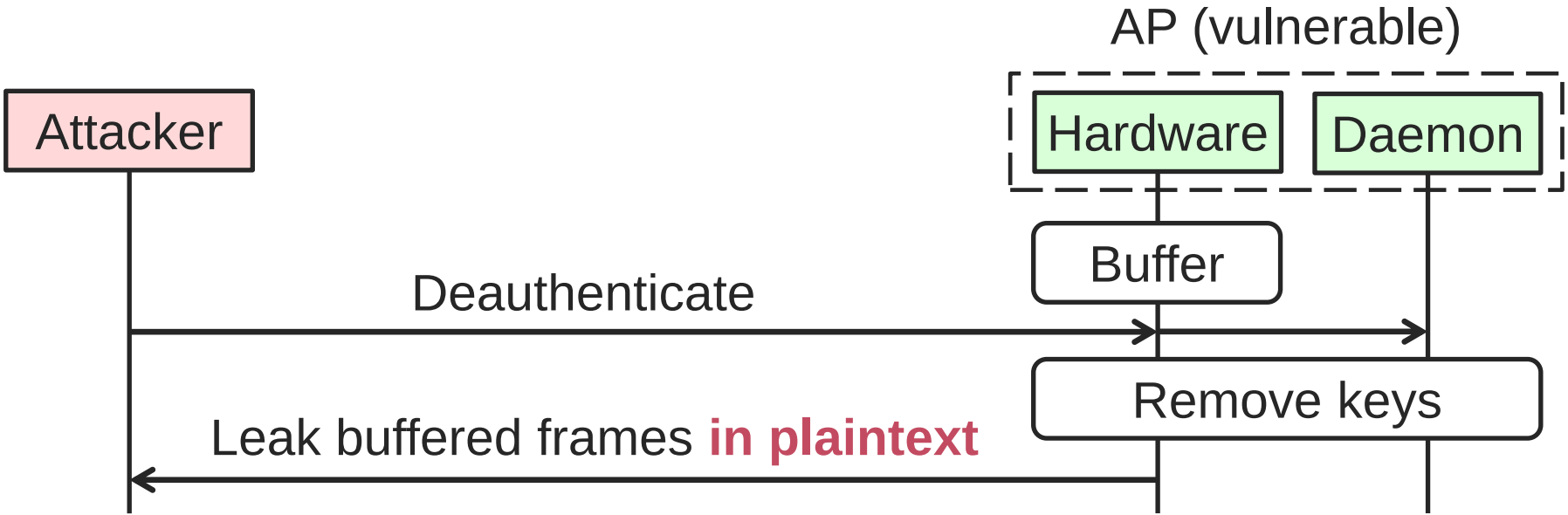
Paper accepted at USENIX Security ‘23

KU LEUVEN DistriNet

History of Wi-Fi

- WEP (1999): quickly broken [FMS01]
- WPA1/2 (~2003)
  - Offline password brute-force
  - **KRACK** & **Kraken** [VP17,VP18]
- WPA3 (2018):
  - **Dragonblood** side-channels [VR20]

Background: Kr00k implementation flaw

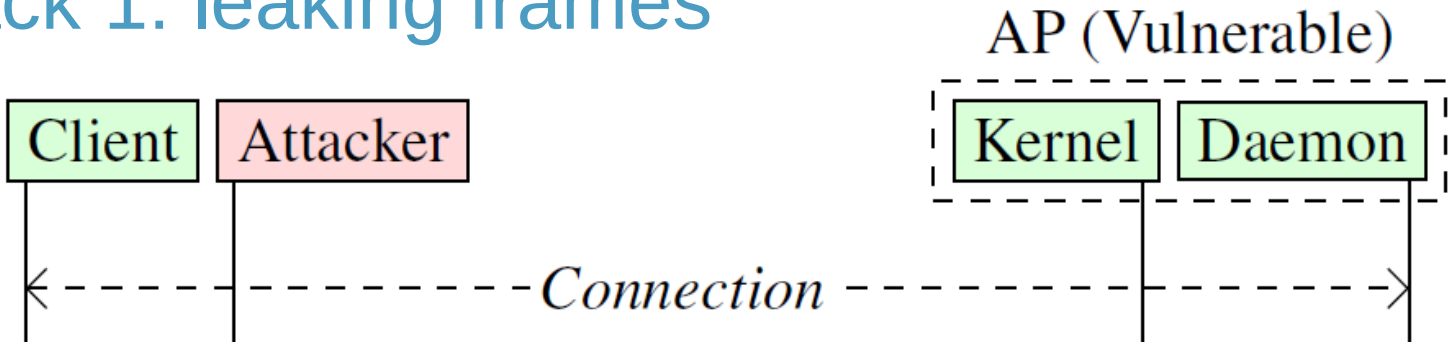


Research question: **how are security contexts managed**?

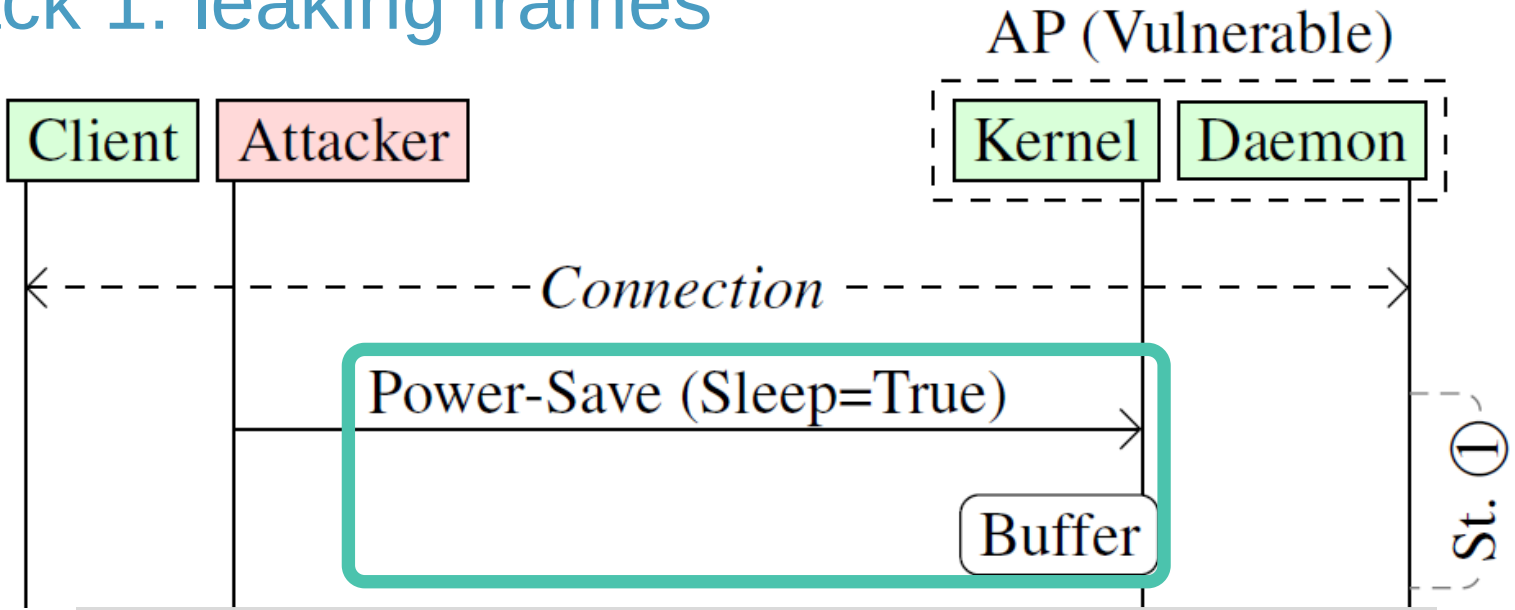
New attack 1: leaking frames

Attack 1: leaking frames

Attack 1: leaking frames

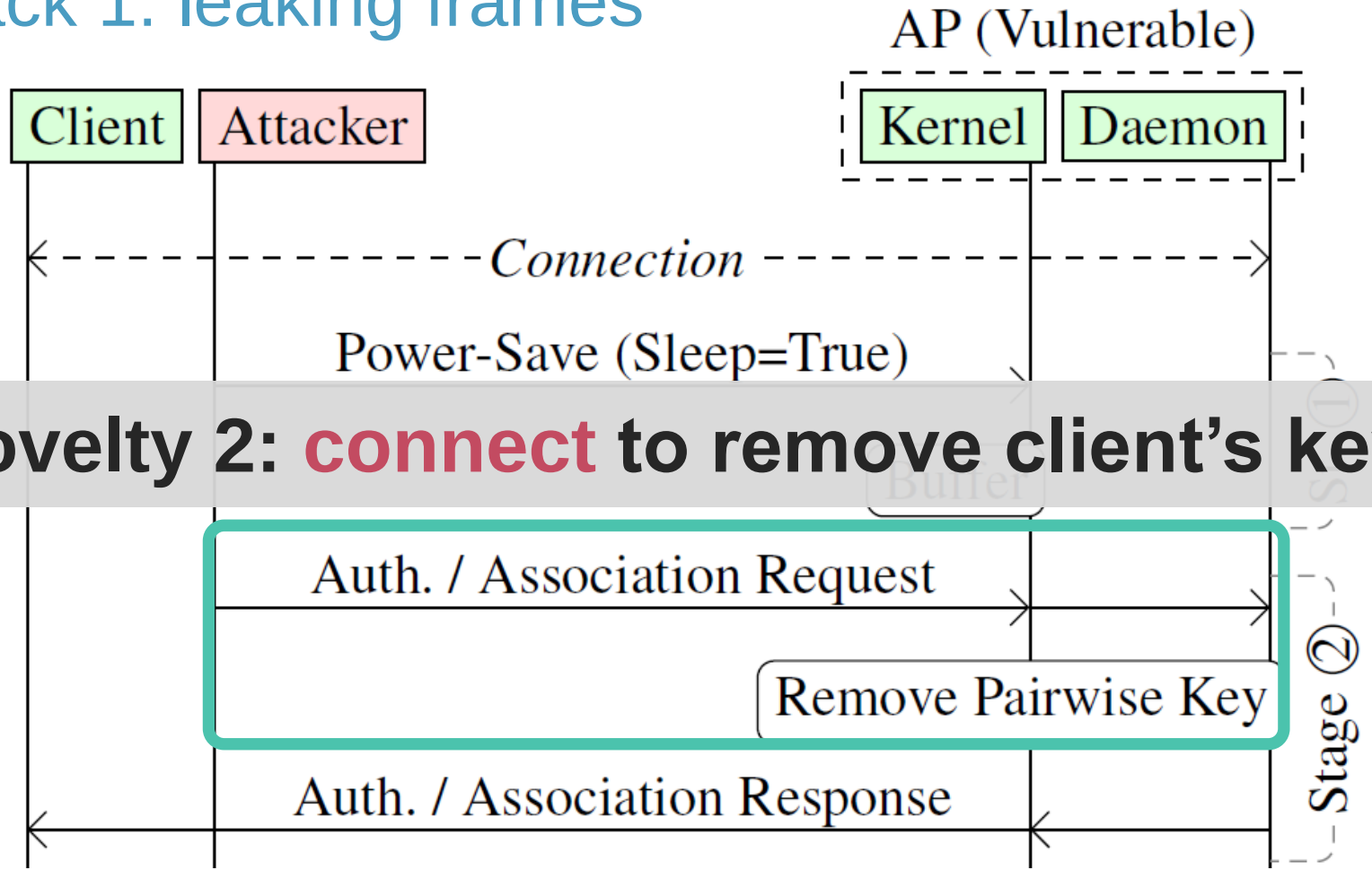


Attack 1: leaking frames



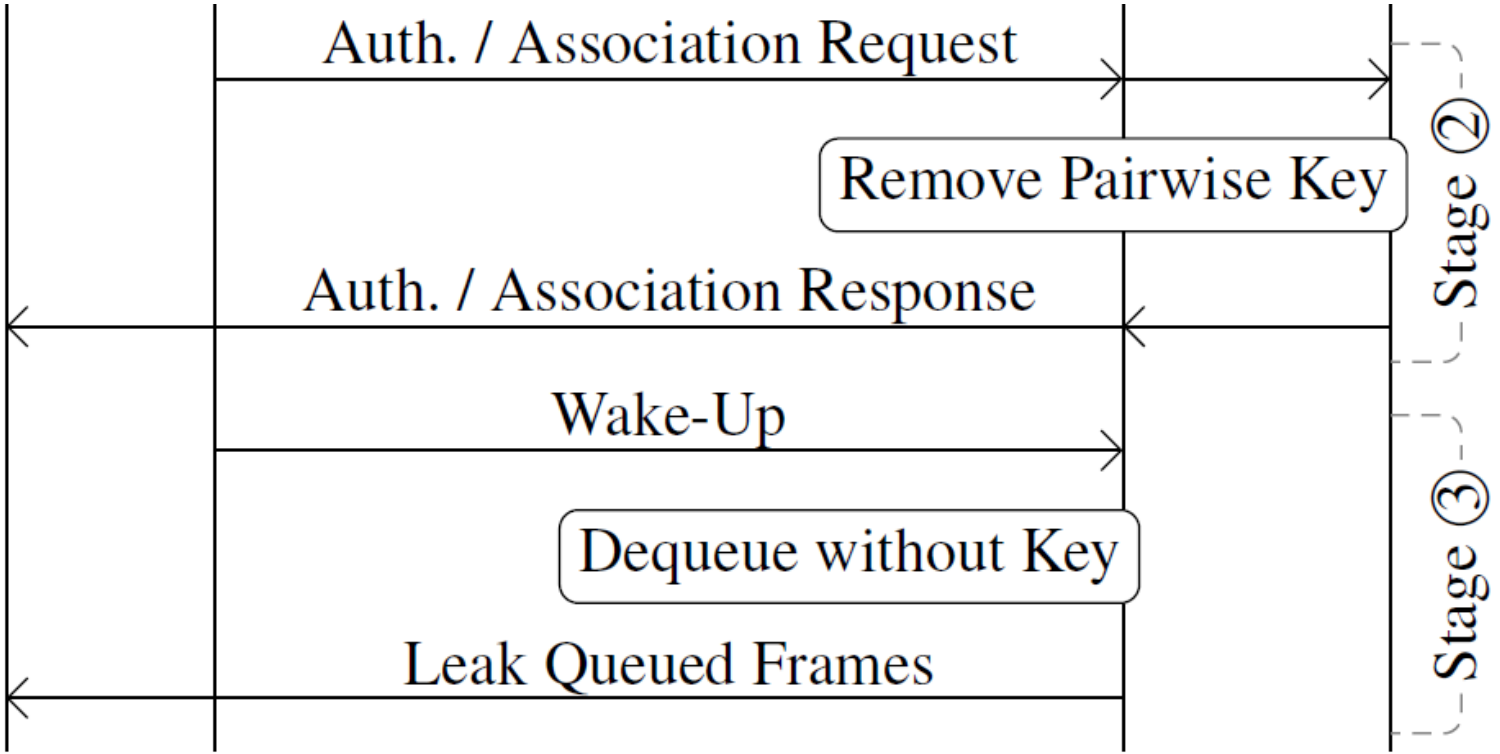
Novelty 1: controlled buffering

Attack 1: leaking frames

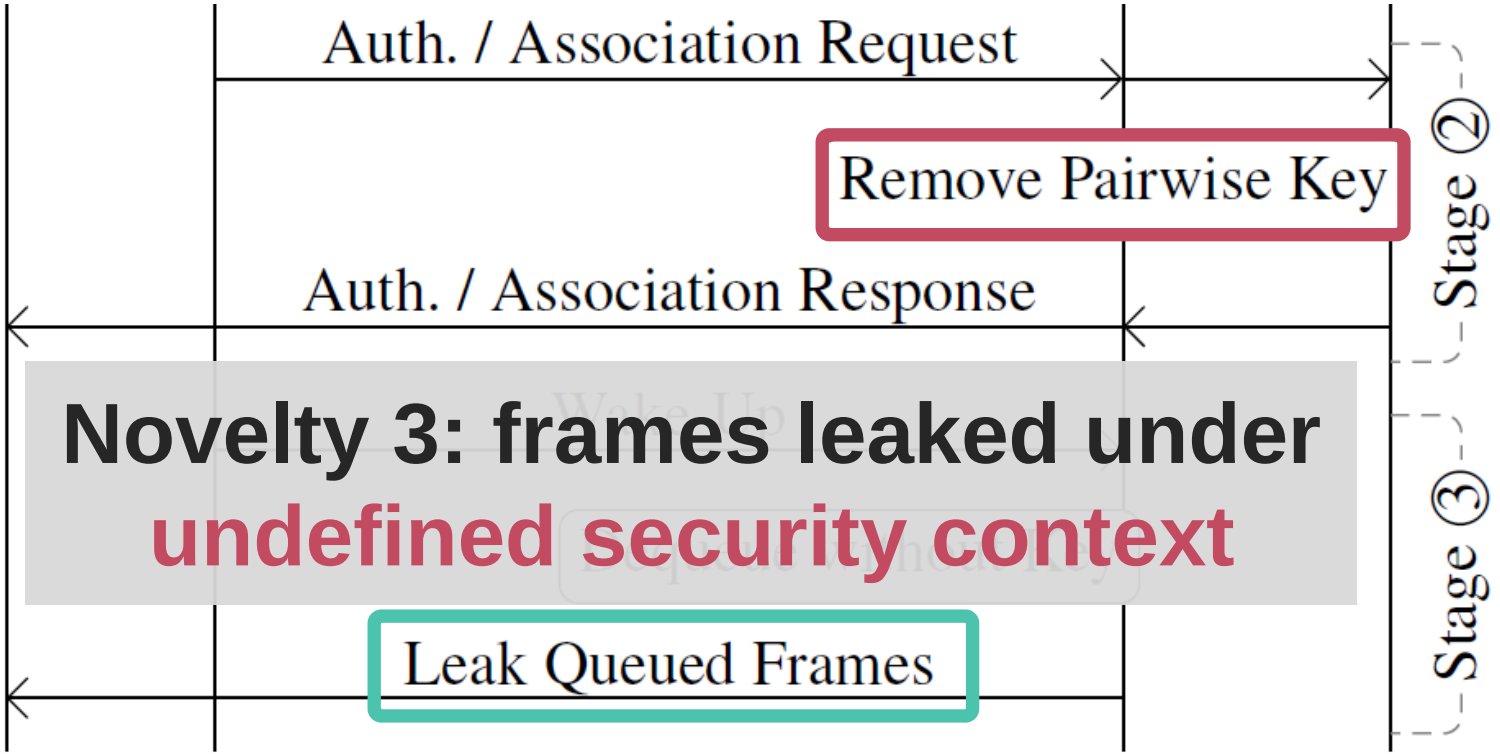


Novelty 2: connect to remove client's keys

Attack 1: leaking frames



Attack 1: leaking frames



Novelty 3: frames leaked under undefined security context

Undefined security context: FreeBSD example

How the frame is leaked depends on kernel version & driver:

Version	driver (vendor)	Leakage
13.0	run (Ralink)	Plaintext
13.1	run (Ralink)	WEP with all-zero key
13.1	rum (Ralink)	CCMP with group key
13.1	rtwn (Realtek)	CCMP with group key

- Malicious insiders know the group key!
- Linux, NetBSD, open Atheros firmware also affected

Root cause

Standard isn't explicit on how to manage buffered frames

- Should drop buffered frames when refreshing/deleting keys

Lesson: include transmit queue in formal Wi-Fi models

- Because buffered frames are not yet encrypted (unlike TLS)
- [CKM20] modelled transmit queue but not key deletion!

[CKM20]: A Formal Analysis of IEEE 802.11's WPA2 by C. Cremers, B. Kiesl, and N. Medinger (USENIX Security)

New attack 2:

Bypassing client isolation

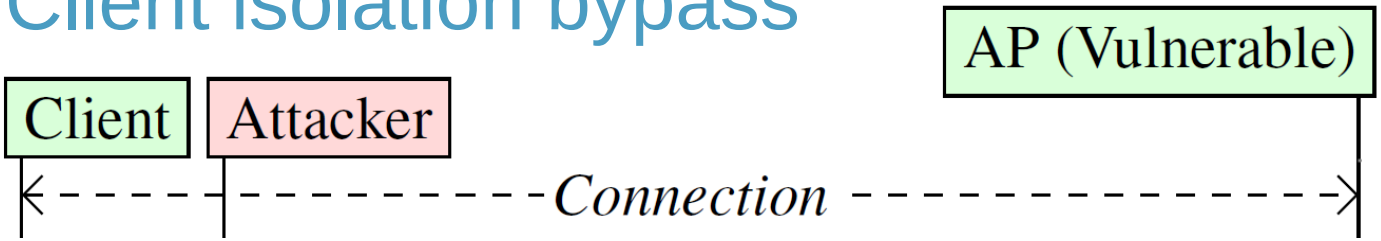
Attack 2: bypassing Wi-Fi client isolation

Target is networks that use **client isolation**. Examples:

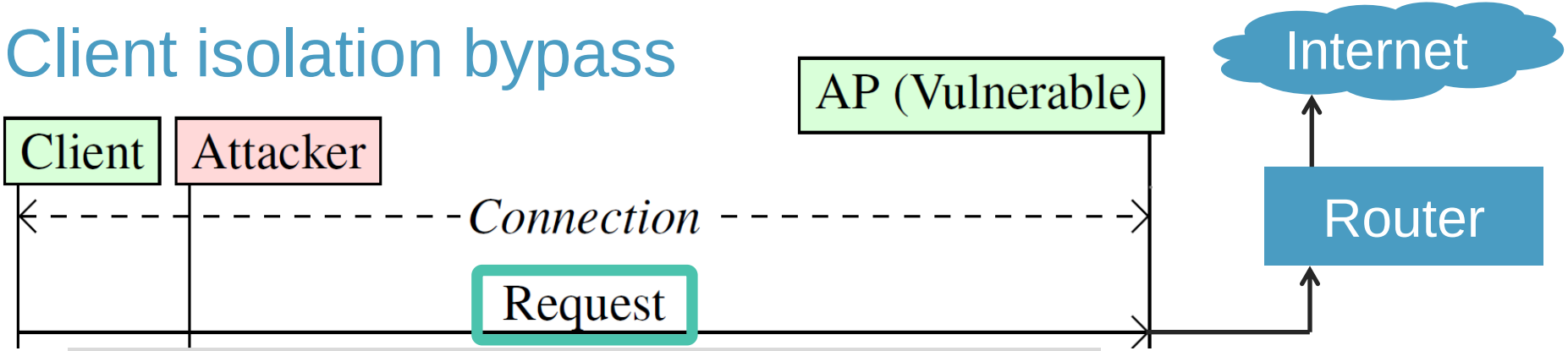
- Company network with malicious/compromised clients
- Public hotspots that require authentication

→ Adversary can connect to the network, but can't attack others

Client isolation bypass

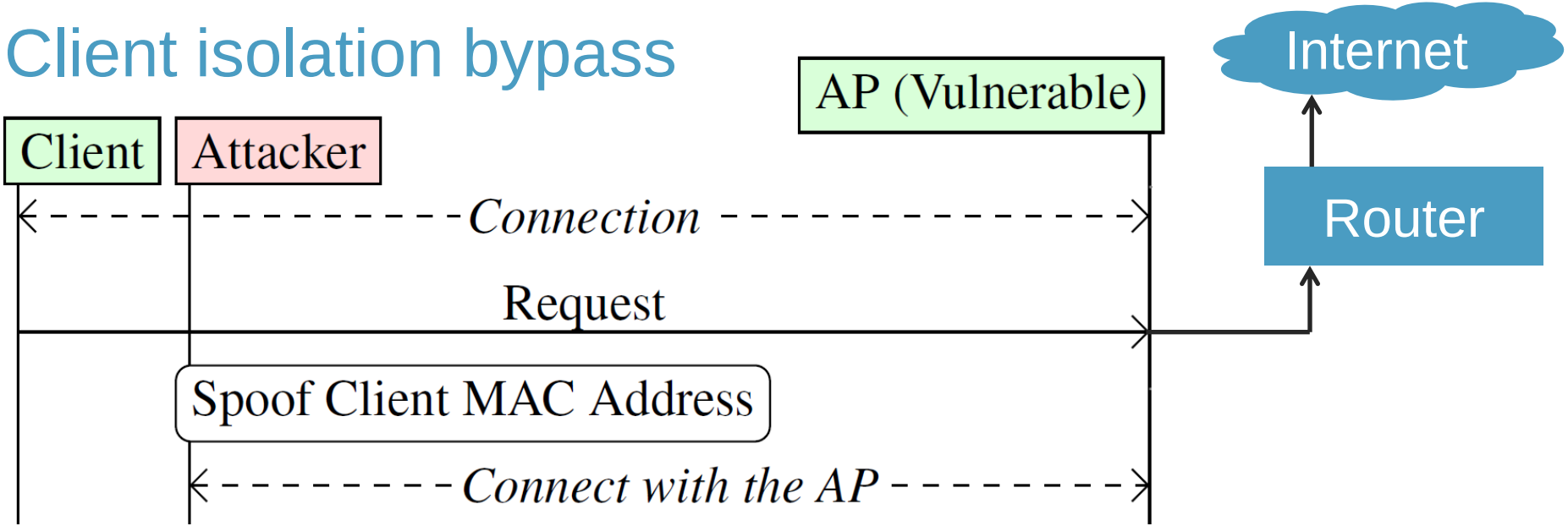


Client isolation bypass

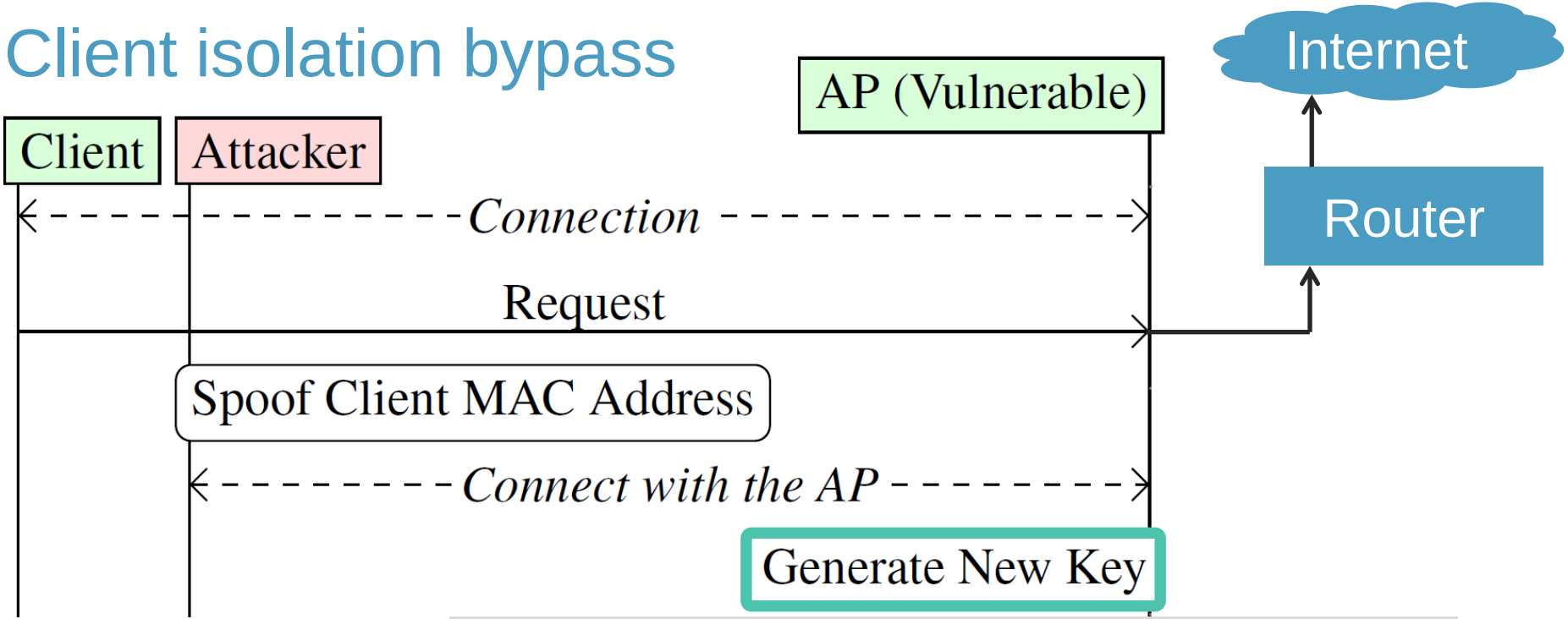


E.g., DNS or HTTP request

Client isolation bypass

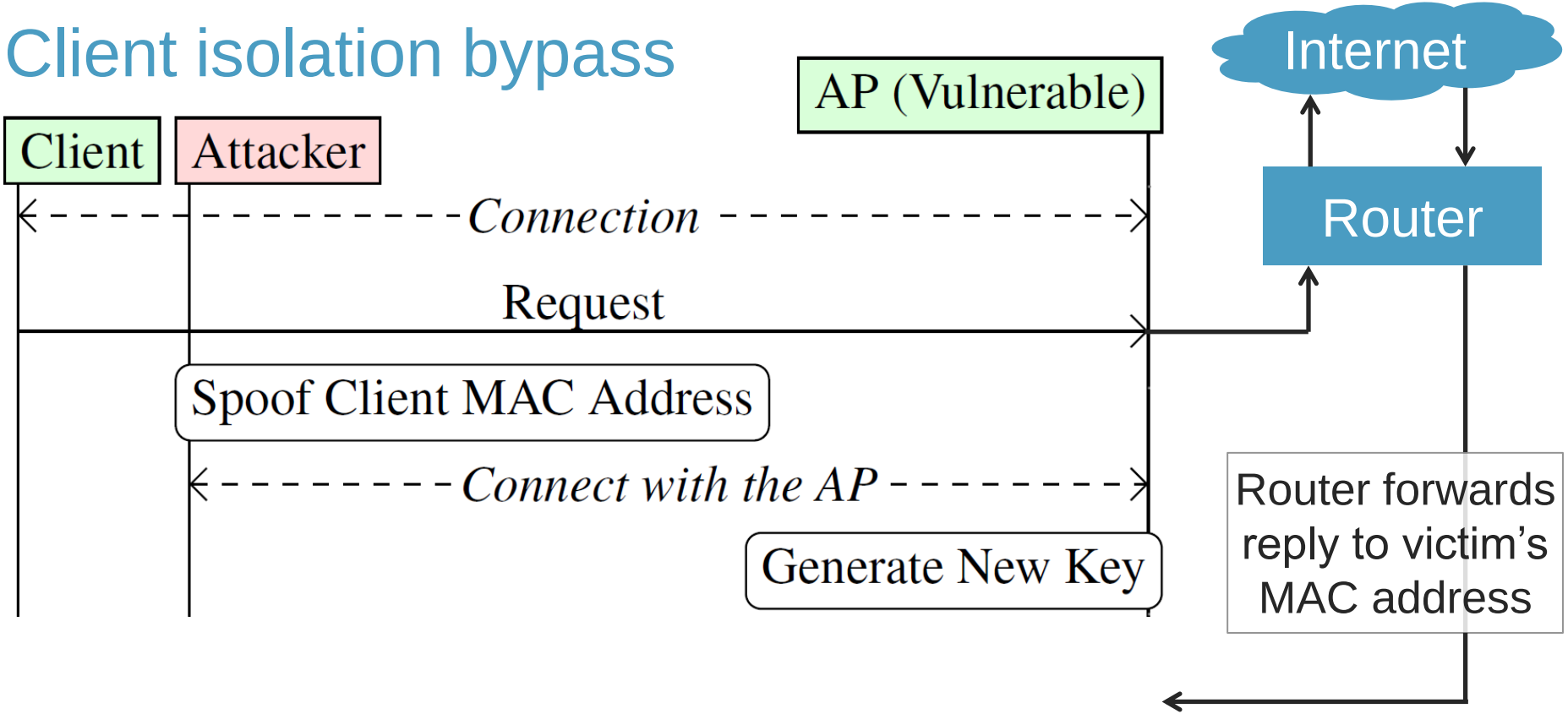


Client isolation bypass

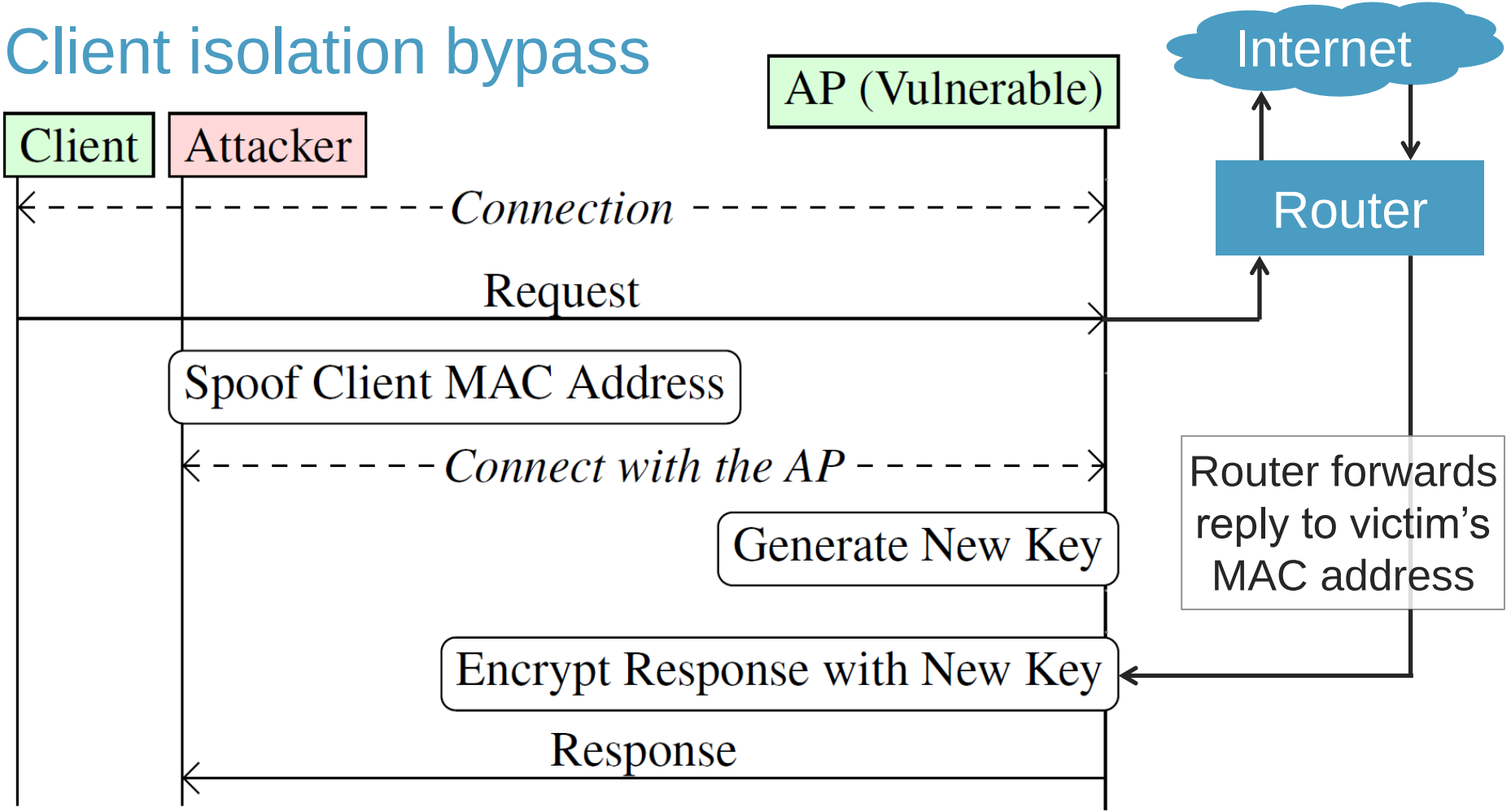


New key is associated with the victim's MAC address

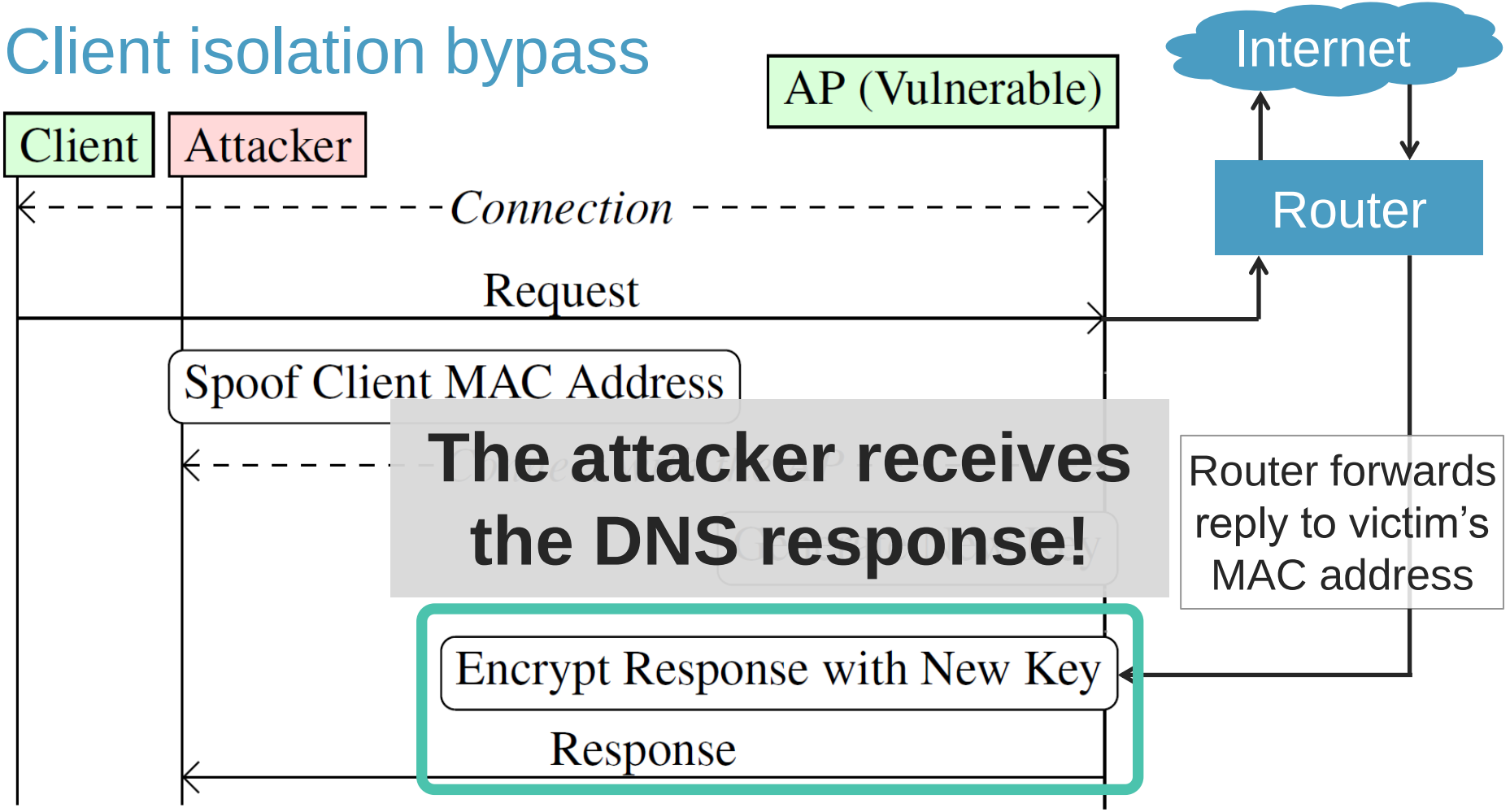
Client isolation bypass



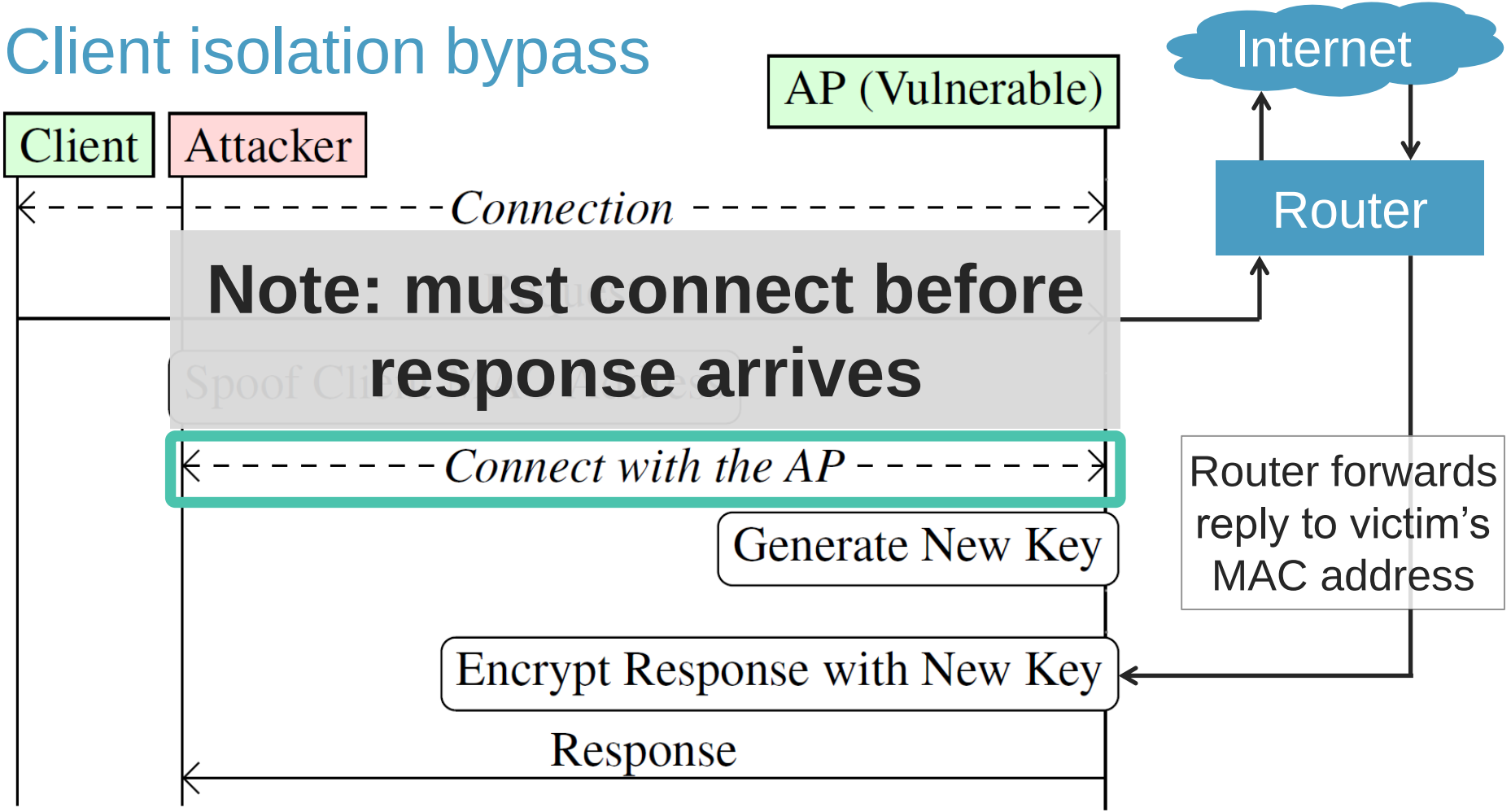
Client isolation bypass



Client isolation bypass



Client isolation bypass



Experiments: home APs

All tested professional & home APs were vulnerable

→ **Design flaw** in Wi-Fi client isolation!

Fast security context override

Technique to quickly reconnect. Experiments:

- Minimum reconnect time: ~12 ms
- Average UDP response time: [Verizon]
  - Transatlantic connections: ~70 ms
  - Connections within Europe: ~13 ms
- TCP responses are retransmitted → trivial to intercept

[Verizon] Verizon IP latency statistics

Root cause

Client identity not authenticated across the network stack:

- Wi-Fi security: 802.1X identity (username)
- Packet routing: IP/MAC addresses

} Not bound to each other

→ Wi-Fi attacker can spoof client's identity on other layers

Other observation: client isolation was "bolted on" by vendors

- Not part of IEEE 802.11 standard → less studied

Fixing client isolation

Disallow recently-used MAC address unless:

- Certain amount of time has passed (incomplete defense)
- We're sure it's the same user as before (complete defense)
  - Based on 802.1X identity or cached keys (not always available)

Currently unclear what vendors will adopt

- Don't rely on client isolation for security
- Alternative: use VLANs to isolate groups

Conclusion

Standard is vague on how to manage buffered frames

- Can **leak frames** under different security context
- Important to **model/define transmit queues**

Can **bypass client isolation**

- All devices vulnerable → **design flaw**
- Hard to fully prevent