Network Security Lab

Group 8

Christian Sassi, Matteo Beltrami, Luca Pedercini

- WPA2 Handshake
- KRACK Attack
- LAB 0: Simulation
- Mininet
- LAB 1: Access Point (AP) testing
- LAB 2: Client (STA) testing
- Mitigation

WPA2 Handshake

KRACK Attack

LAB 0: Simulation

Mininet

LAB 1: Access Point (AP) testing

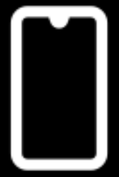
LAB 2: Client (STA) testing

Mitigation

WPA2 Handshake

- **WPA2** (Wi-Fi Protected Access 2): encrypted security protocol that protects internet traffic on wireless networks.
- WPA2 addresses earlier flaws and offers more powerful encryption. It's a standard for Wi-Fi network security.
- **Key hierarchy** with many keys for unicast and multicast traffic.
- **4-way handshake** used for authentication and to create all required keys. The wireless client and access point (AP) will have a secure connection, and all traffic will be encrypted.

WPA2 Handshake: terminology



CLIENT / STATION

ACCESS POINT (AP)

WPA2 Handshake: terminology

PRINCIPAL KEYS USED IN WPA2

Pre-shared secret = “Wifi password”

Pairwise Master Key (PMK)

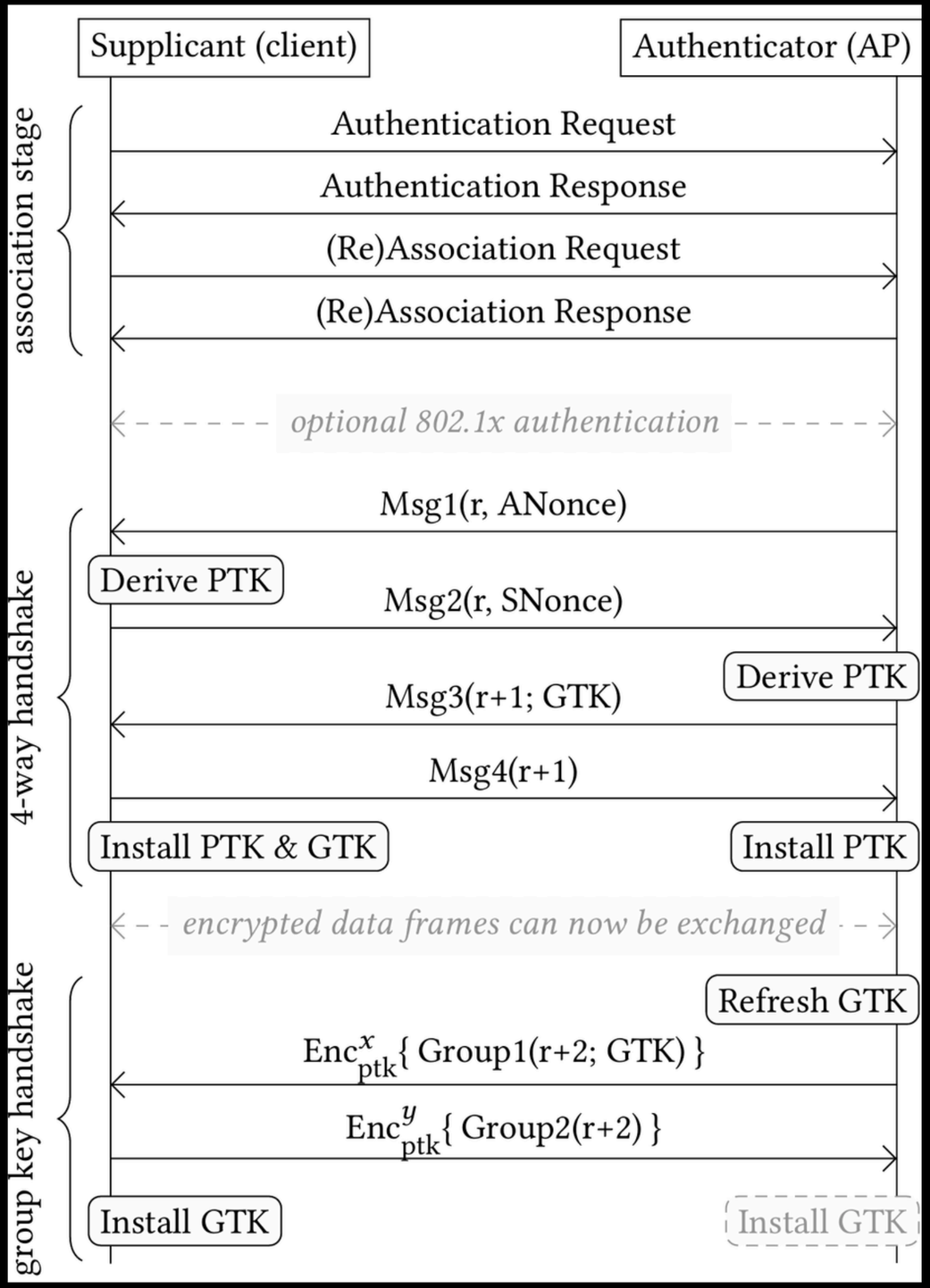
Pairwise Key (PTK)

Group Key (GTK)

Encryption algorithms:

TKIP, AES-CCMP, AES-GCMP

WPA2 Handshake:
4-way Handshake



WPA2 Handshake

KRACK Attack

LAB 0: Simulation

Mininet

LAB 1: Access Point (AP) testing

LAB 2: Client (STA) testing

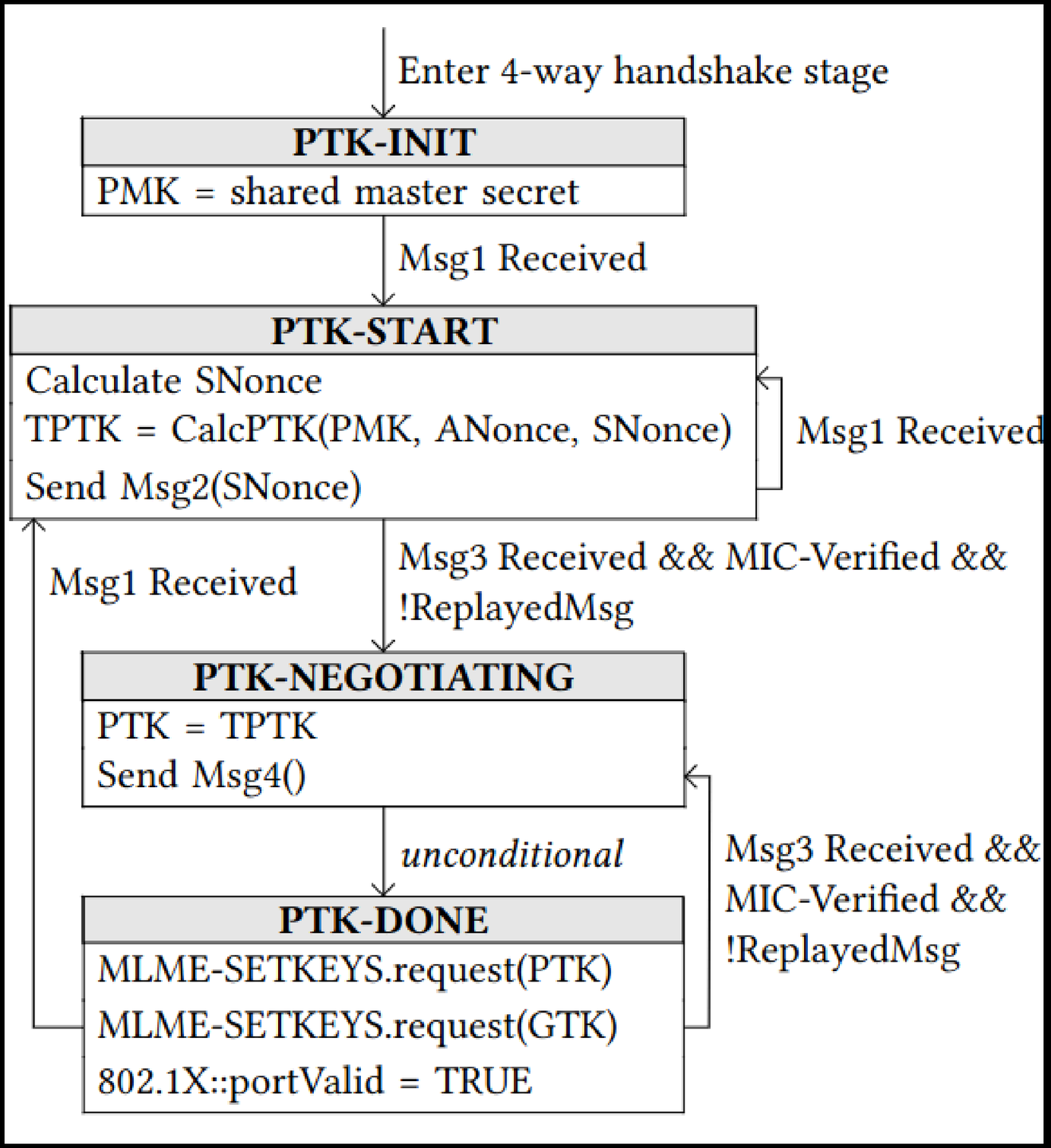
Mitigation

KRACK Attack

- **KRACK** (Key Reinstallation Attack) is a **critical vulnerability** affecting the WPA2 Wi-Fi security protocol.
- Discovered in 2017 by Mathy Vanhoef, a Belgian security researcher.
- KRACK exploits a **weakness** in the **four-way handshake** process used by the WPA2 protocol
- The attacker can force the **installation** of an **already used encryption key**, allowing the data traffic to be **decrypted** and **manipulated**.
- Affected devices are some Access Points, but above all **Android and Linux** clients (as they run wpa_supplicant).

KRACK Attack:
how it works

Supplicant
State Machine



KRACK Attack: how it works

When a client receives a **retransmitted message 3** of the 4-way handshake, it will **reinstall** the **already in-use pairwise key** (possibly, also the group key).

↓

If the client reinstalls one of these keys, the associated packet number (PN) is likely reset.

↓

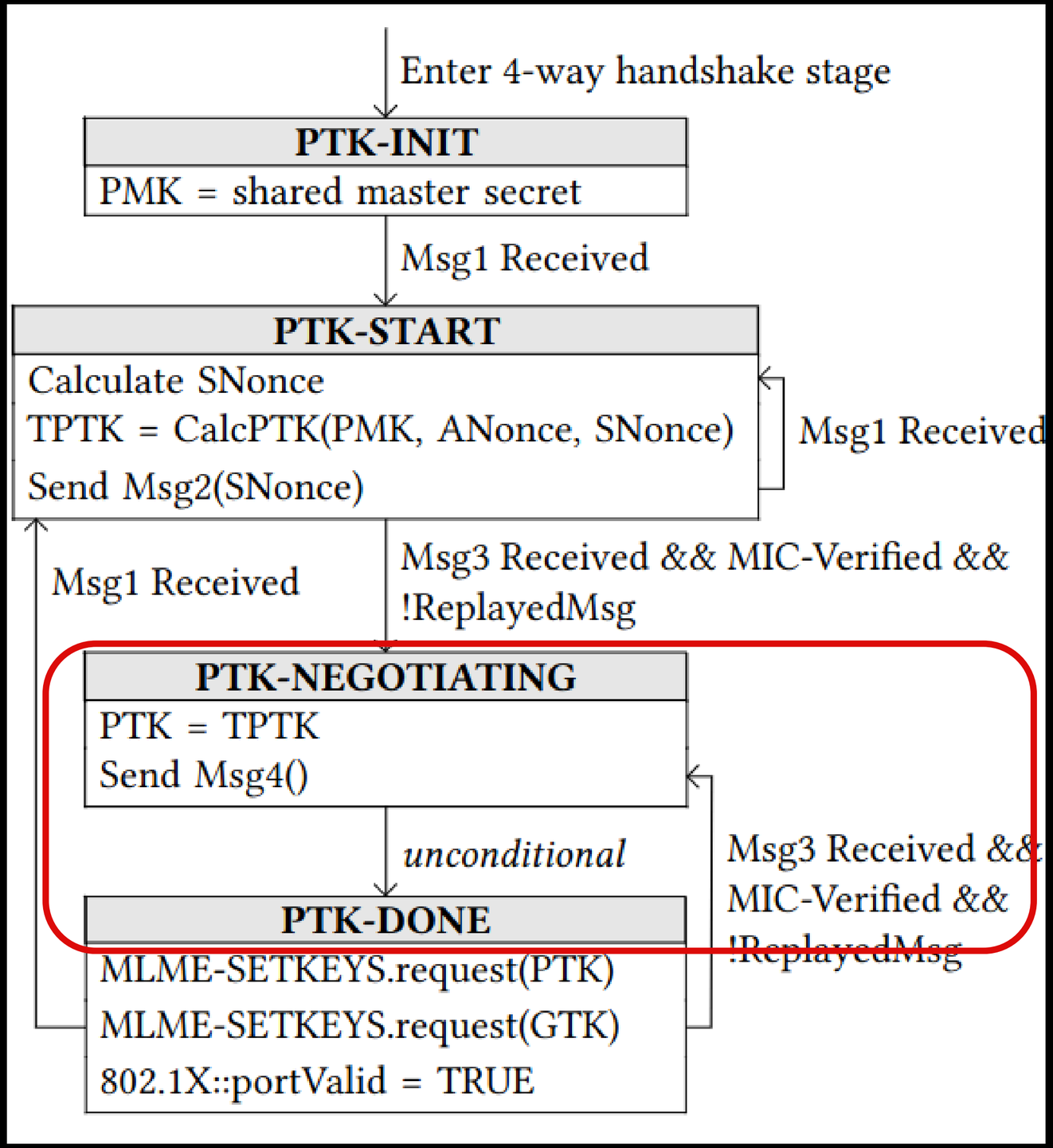
The client will subsequently **reuse packet numbers** when sending frames protected using TKIP, CCMP, or GCMP.

↓

This causes **NONCE REUSE** (sometimes also called **Initialization Vector** reuse).

WPA2 Handshake:
how it works

Supplicant
State Machine



WPA2 Handshake

KRACK Attack

LAB 0: Simulation

Mininet

LAB 1: Access Point (AP) testing

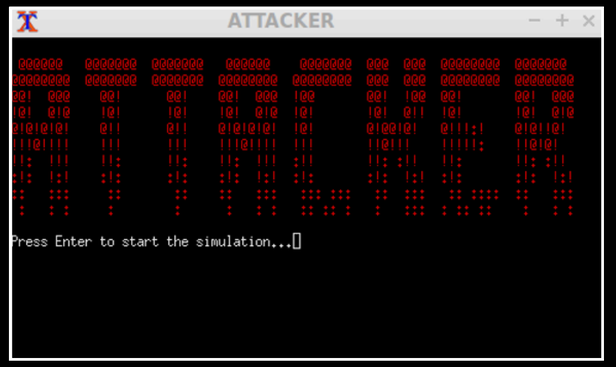
LAB 2: Client (STA) testing

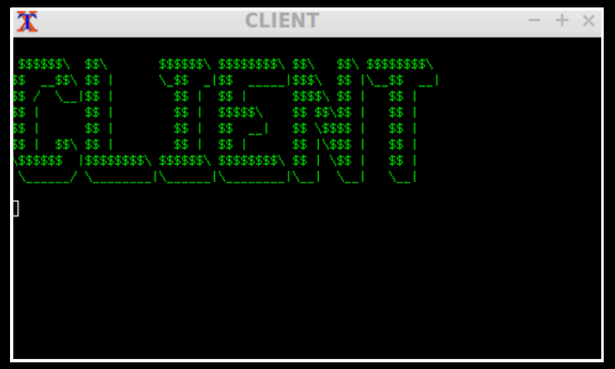
Mitigation

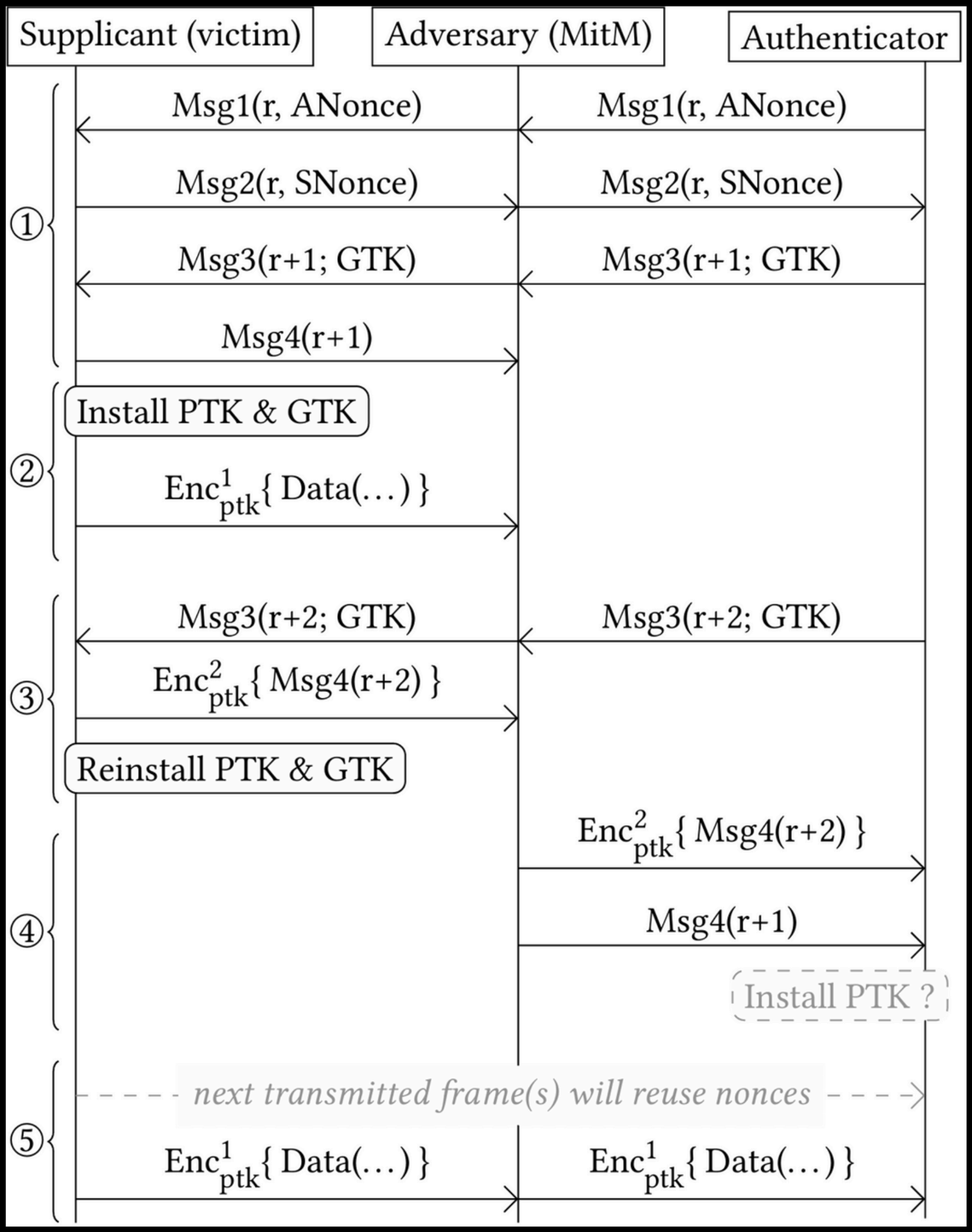
LABS


```
krack@krack:~$ cd network-security-lab
krack@krack:~$ chmod +x launcher.sh
krack@krack:~$ ./launcher.sh
```

LAB 0: Simulation







WPA2 Handshake

KRACK Attack

LAB 0: Simulation

Mininet

LAB 1: Access Point (AP) testing

LAB 2: Client (STA) testing

Mitigation

Mininet

- Realistic **virtual network,** running real kernel, switch and application code
- Used to simulate possible attack scenarios, running APs and clients with desired wpa_supplicant version.

WPA2 Handshake

KRACK Attack

LAB 0: Simulation

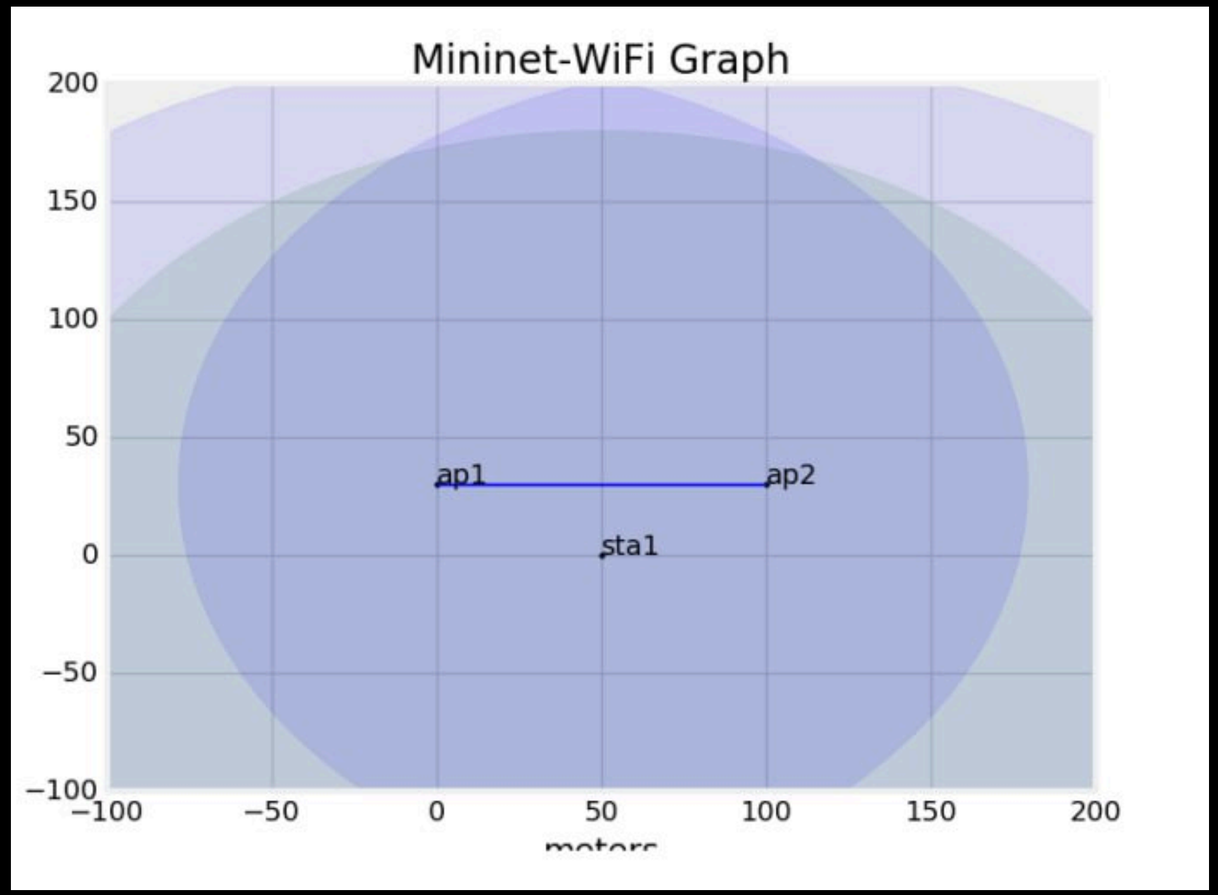
Mininet

LAB 1: Access Point (AP) testing

LAB 2: Client (STA) testing

Mitigation

LAB 1: Access Point (AP) testing



Scenario

GOAL

Verify that AP2 is vulnerable to KRACK attacks as it reinstalls keys and reuse nonces

↓

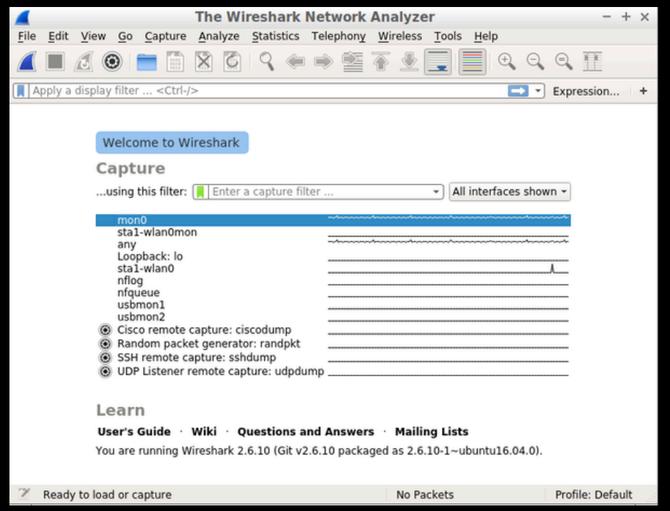
As AP2 implements the protocol IEEE 802.11r, Fast BSS Transition (FT)

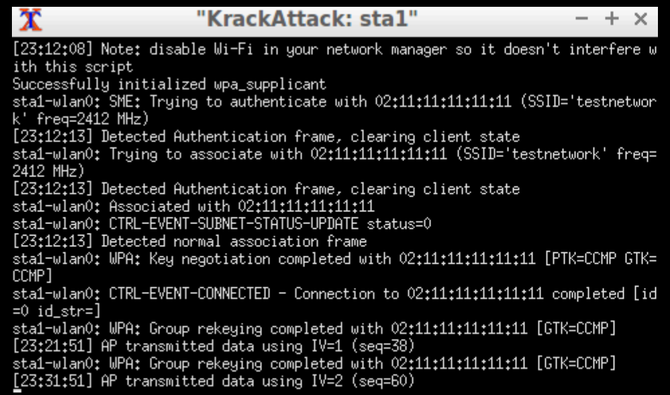
LAB 1: Access Point (AP) testing

From your **launcher**, start LAB 1

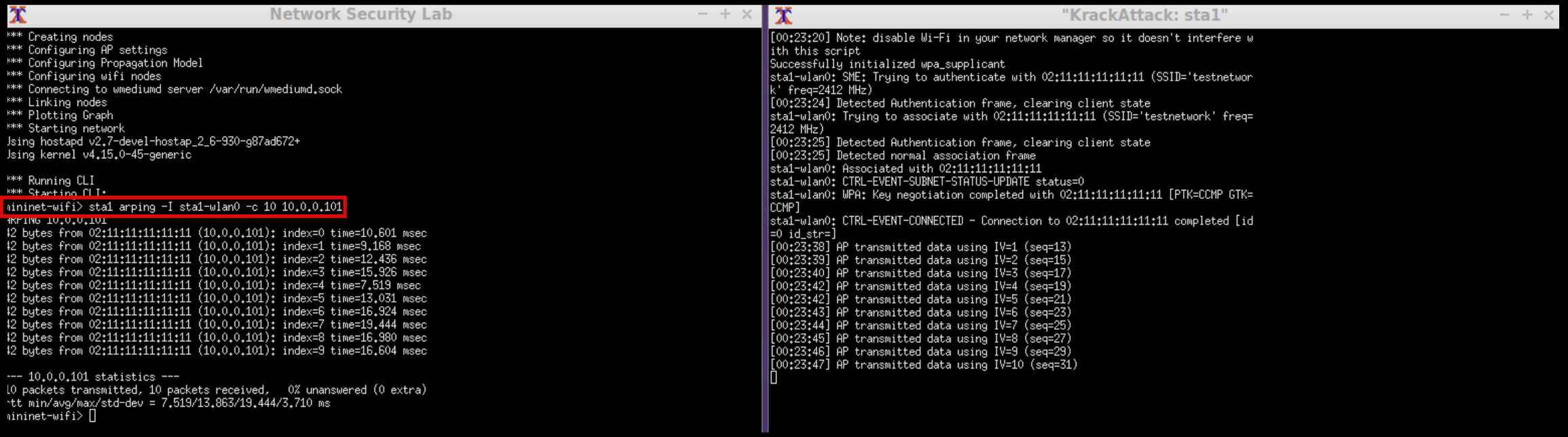
Five windows:

- Figure representing the scenario
- WIreshark, to capture the transmitted packets
- Three XTerm terminals
  - Mininet console
  - Two terminals to implement the KRACK attack



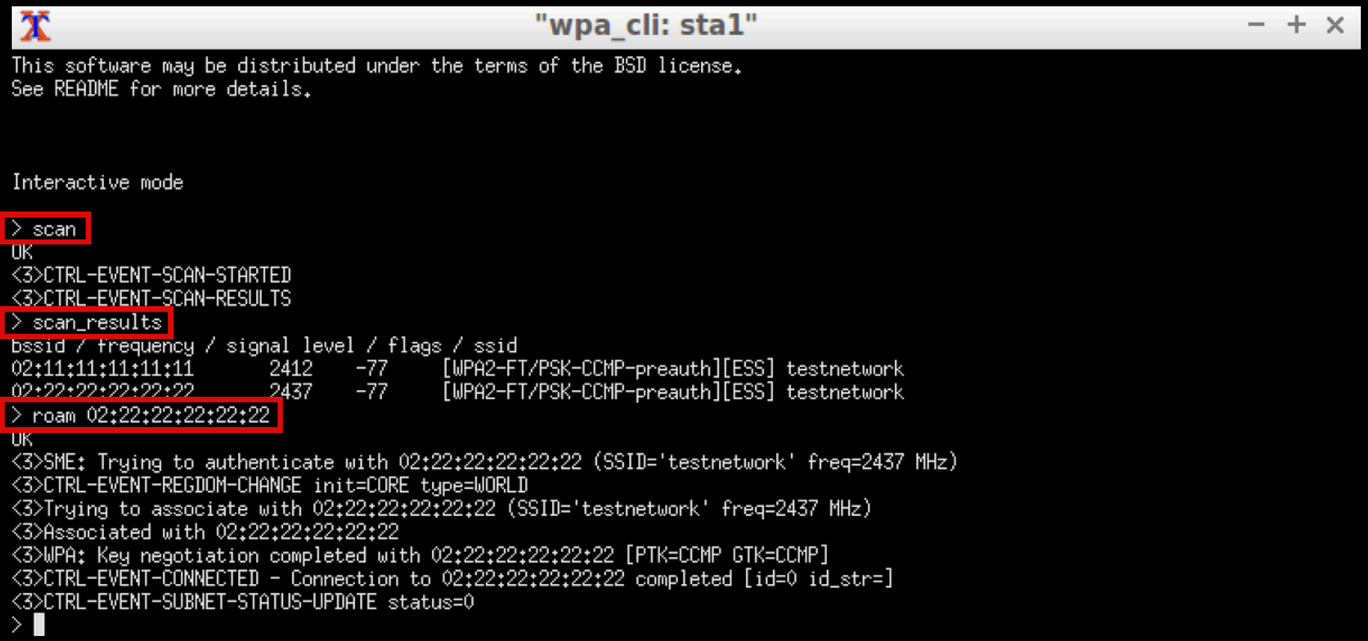


LAB 1: Access Point (AP) testing



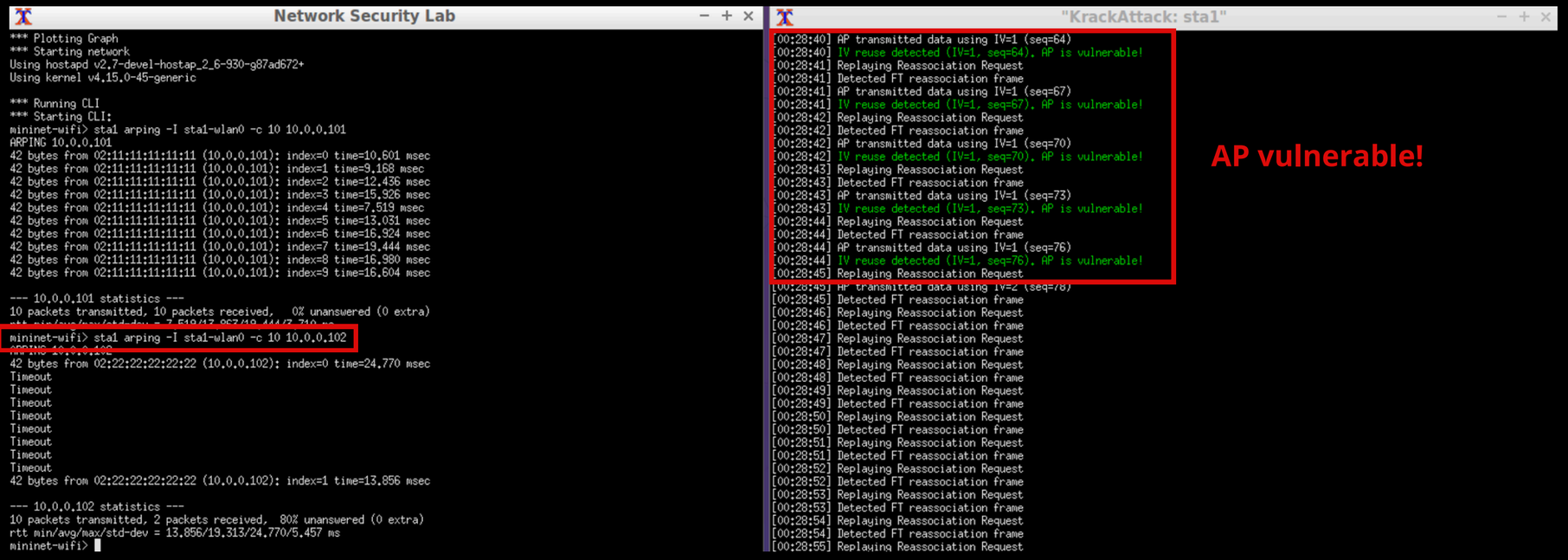

```
mininet-wifi> sta1 arping -i sta1-wlan0 -c 10 10.0.0.101
```

LAB 1: Access Point (AP) testing



- `wpa_cli -i sta1-wlan0`
- `scan`
- `scan_results`
- `roam 02:22:22:22:22:22`

LAB 1: Access Point (AP) testing




```
mininet-wifi> sta1 arping -I sta1-wlan0 -c 10 10.0.0.102
```

WPA2 Handshake

KRACK Attack

LAB 0: Simulation

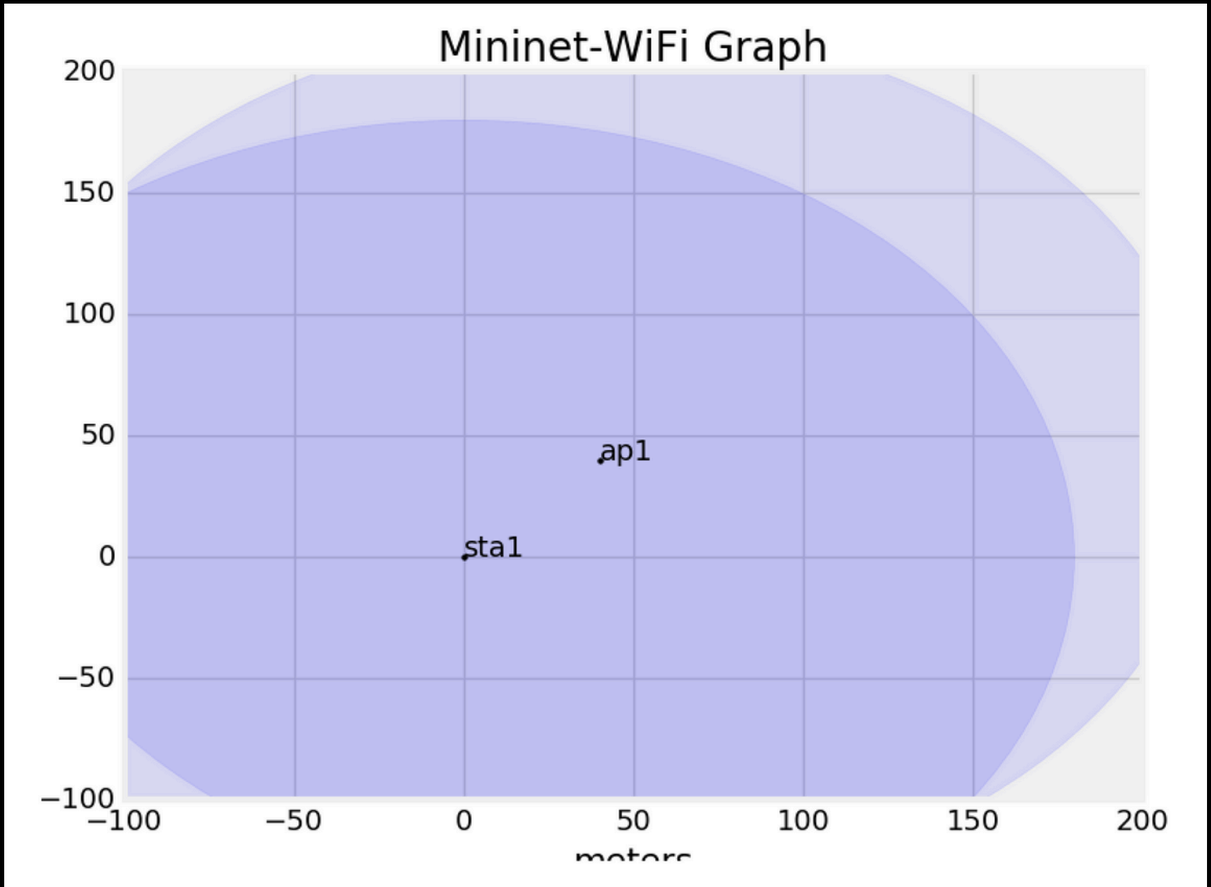
Mininet

LAB 1: Access Point (AP) testing

LAB 2: Client (STA) testing

Mitigation

LAB 2: Client (STA) testing



Scenario

GOAL

Verify that STA1 is vulnerable to KRACK attacks as it reinstalls keys and reuse nonces

↓

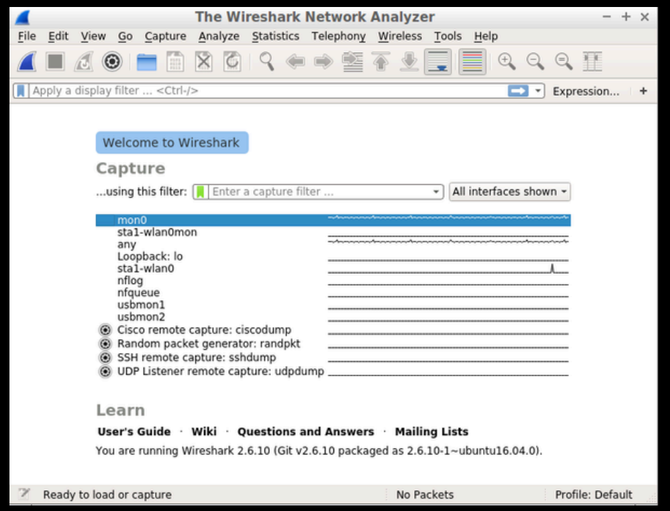
As the client uses a vulnerable version of wpa_supplicant

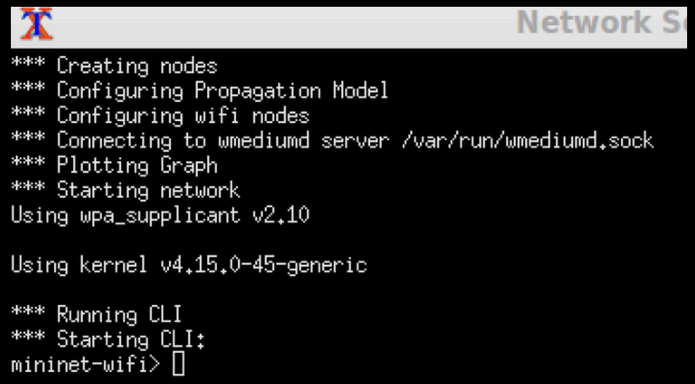
LAB 2: Client (STA) testing

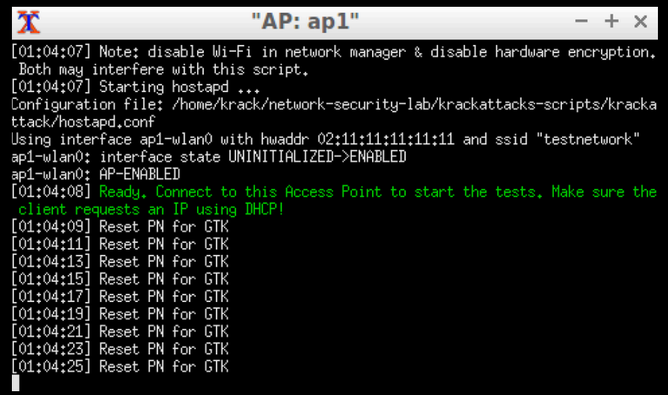
From your **launcher**, start LAB 2

Five windows:

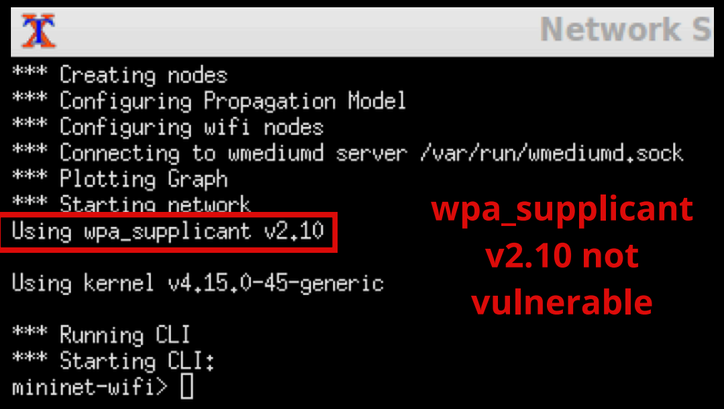
- Figure representing the scenario
- WIreshark, to capture the transmitted packets
- Three XTerm terminals
  - Mininet console
  - Two terminals to implement the KRACK attack



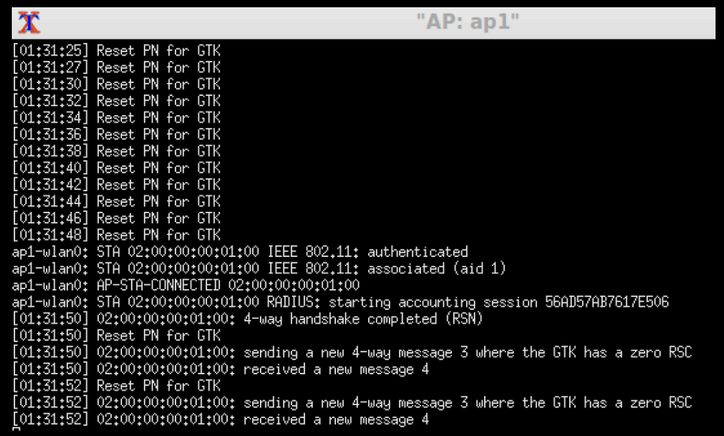


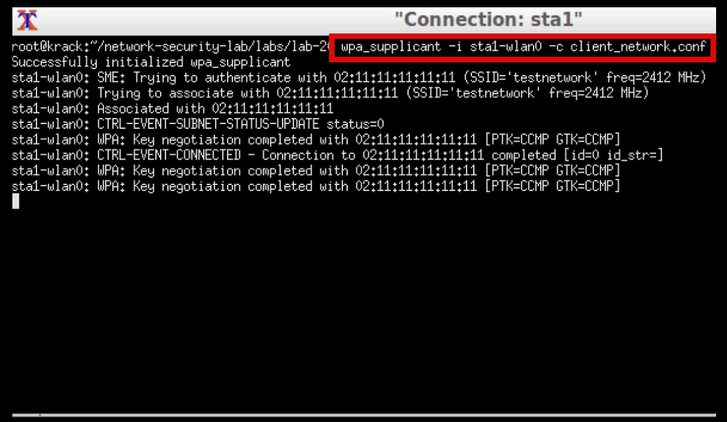


LAB 2: Client (STA) testing



wpa_supplicant v2.10 not vulnerable





CLIENT / STA not vulnerable!

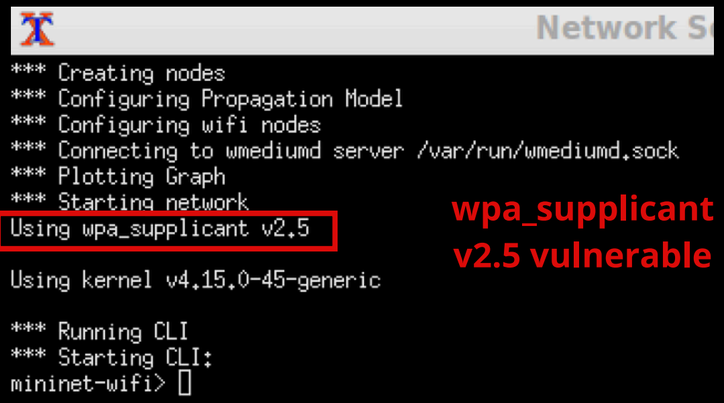
```
> wpa_supplicant -i sta1-wlan0 -c client_network.conf
```

LAB 2: Client (STA) testing

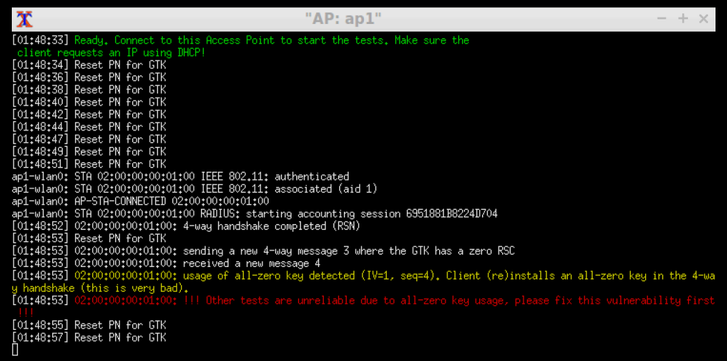
The previous client was not vulnerable. It has a non-vulnerable *wpa_supplicant* version.
Let's **downgrade** the ***wpa_supplicant*** to a vulnerable version (v 2.5).

```
krack@krack:~$ cd network-security-lab/krackattacks-scripts/krackattack/hostap-wpa_supplicant-2.5/wpa_supplicant
krack@krack:~$ make clean
krack@krack:~$ make
krack@krack:~$ sudo make install
krack@krack:~$ wpa_supplicant -v # You should have the 2.5 installed now
```

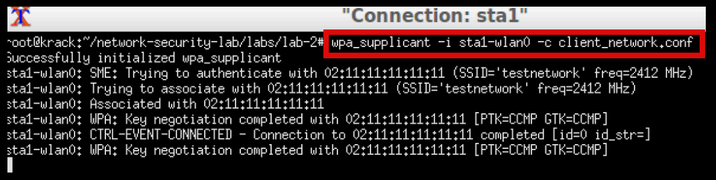
LAB 2: Client (STA) testing



wpa_supplicant v2.5 vulnerable



CLIENT / STA vulnerable!




```
> wpa_supplicant -i sta1-wlan0 -c client_network.conf
```

LAB 2: Client (STA) testing

Live testing

LAB 3: extra

Demonstration based on the paper

Key Reinstallation Attacks:
Forcing Nonce Reuse in WPA2
(CSS 2017)

Made by Mathy Vanhoef

Attacking Android & Linux

KRACK Attacks: Bypassing WPA2 against Android and Linux

WPA2 Handshake

KRACK Attack

LAB 0: Simulation

Mininet

LAB 1: Access Point (AP) testing

LAB 2: Client (STA) testing

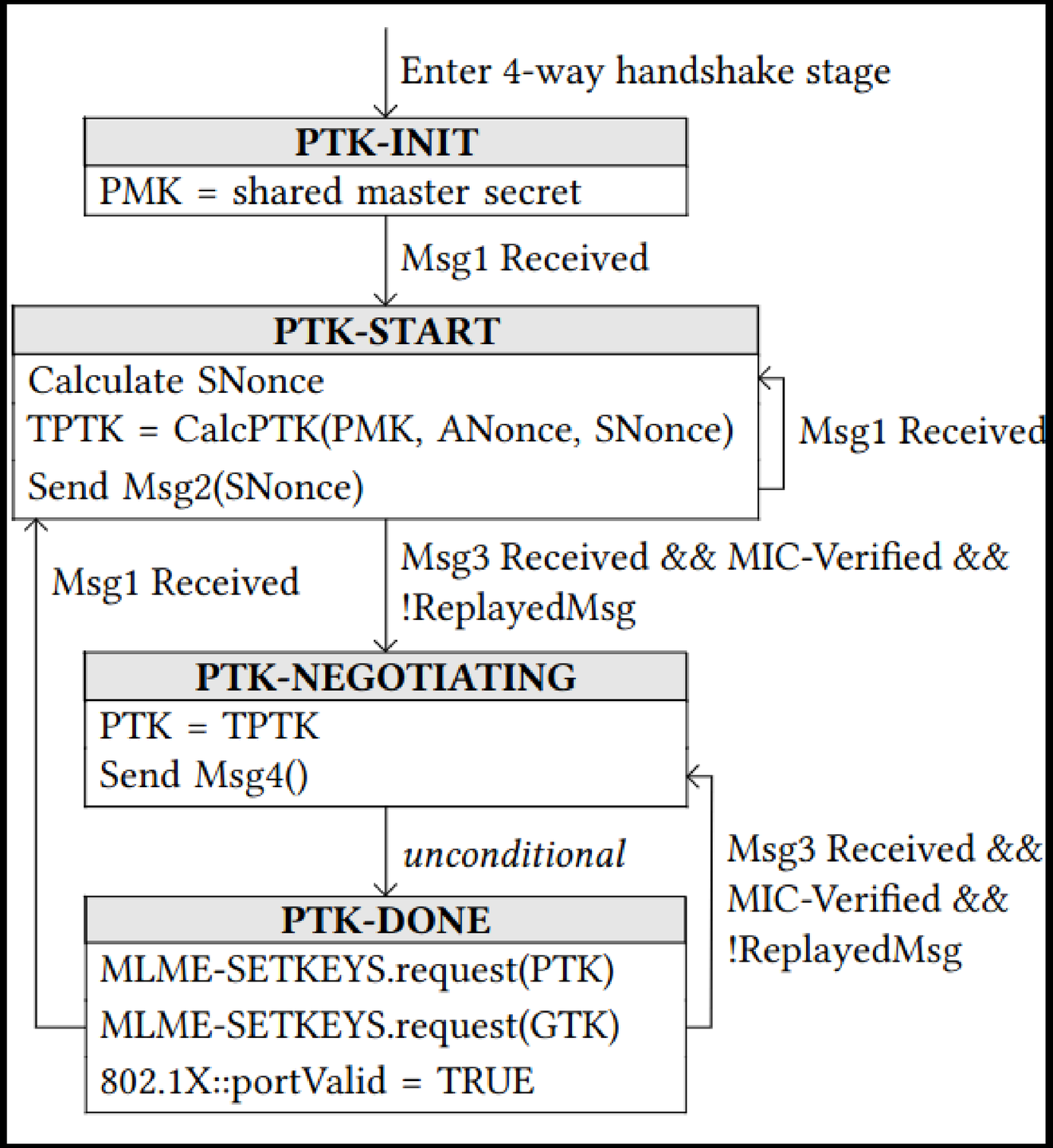
Mitigation

Mitigation

ANY IDEAS ?

Mitigation

Hint: look at client's wpa_supplicant state machine



Mitigation

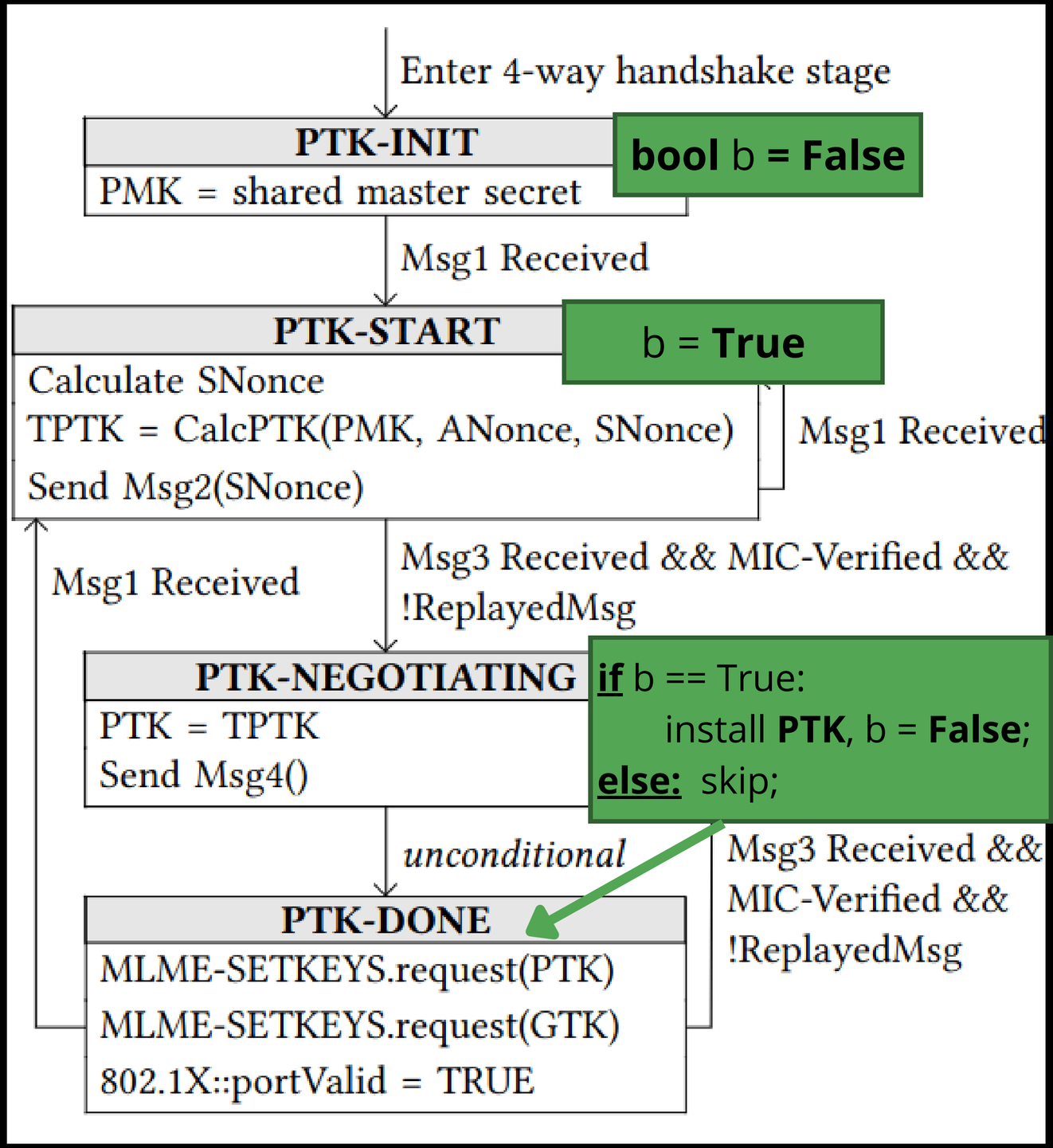
Mathy Vanhoef proposed solution
(and actually implemented)

Assure that a particular key is only installed once during a handshake execution

Add a **boolean variable** to the state machine, initialized as false and set to true when generating a fresh PTK in **PTK-START**

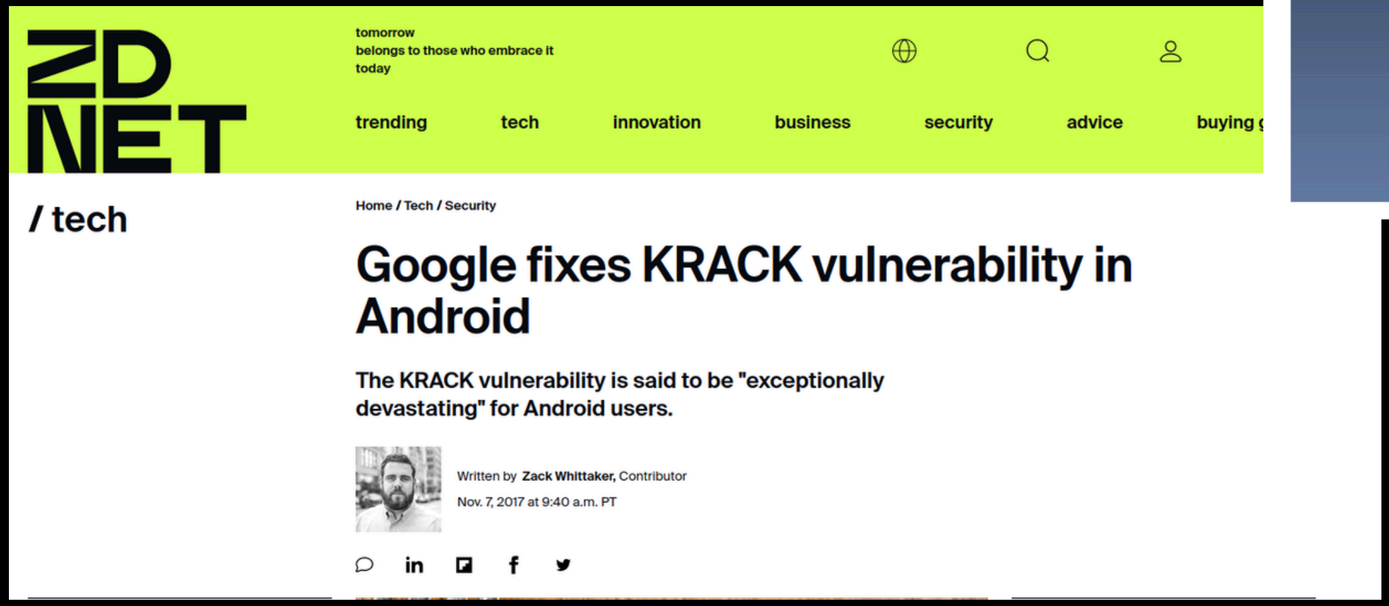
When entering in **PTK-DONE:**

- if **true -->** install PTK, set boolean to false
- if **false -->** PTK is **NOT installed**



Mitigation

The proposed solution was actually implemented on all affected devices, which **in 2017 received an important security patch** to solve KRACK Vulnerability.



THANKS !