



KRACKATTACK

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

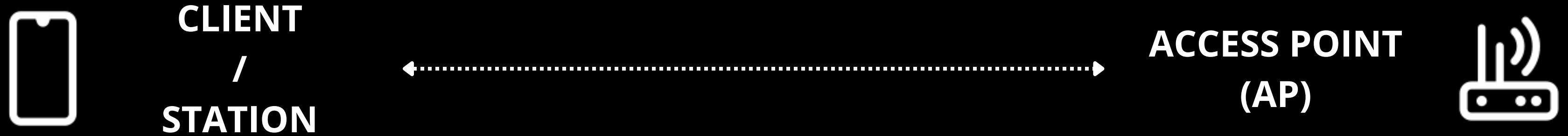
Mitigation

WPA2 Handshake

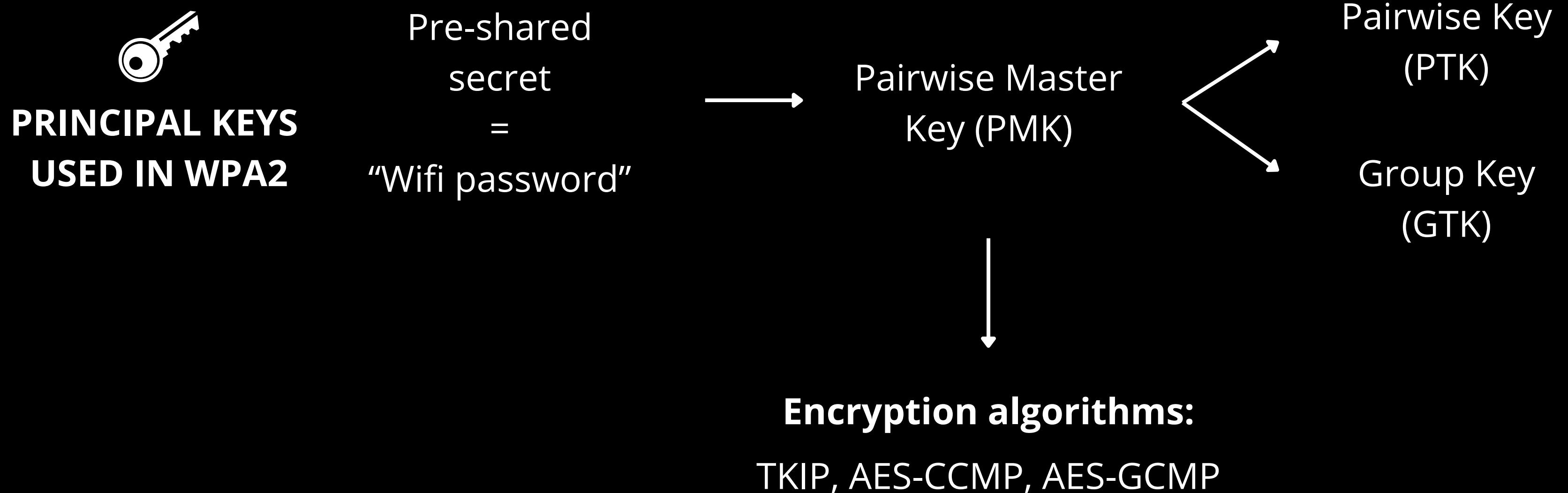
- ▶ **WPA2** (Wi-Fi Protected Access 2): encrypted security protocol that protects internet traffic on wireless networks.
- ▶ WPA2 addresses earlier flaws and offers more powerful encryption. It's a standard for Wi-Fi network security.
- ▶ **Key hierarchy** with many keys for unicast and multicast traffic.
- ▶ **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.



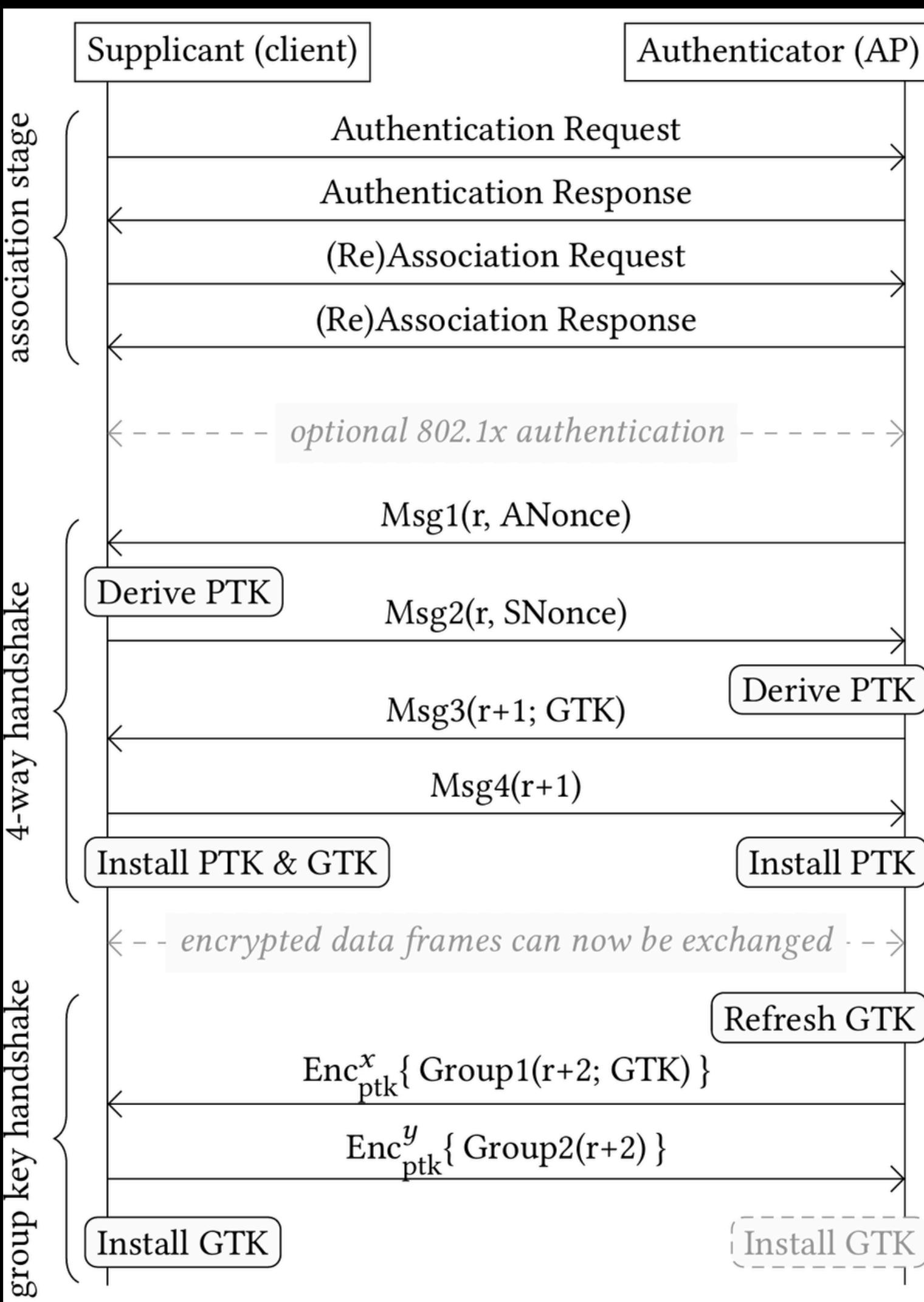
WPA2 Handshake: terminology



WPA2 Handshake: terminology



WPA2 Handshake: 4-way Handshake





WPA2 Handshake



KRACK Attack



LAB 0: Simulation



Mininet



LAB 1: Access Point (AP) testing



LAB 2: Client (STA) testing



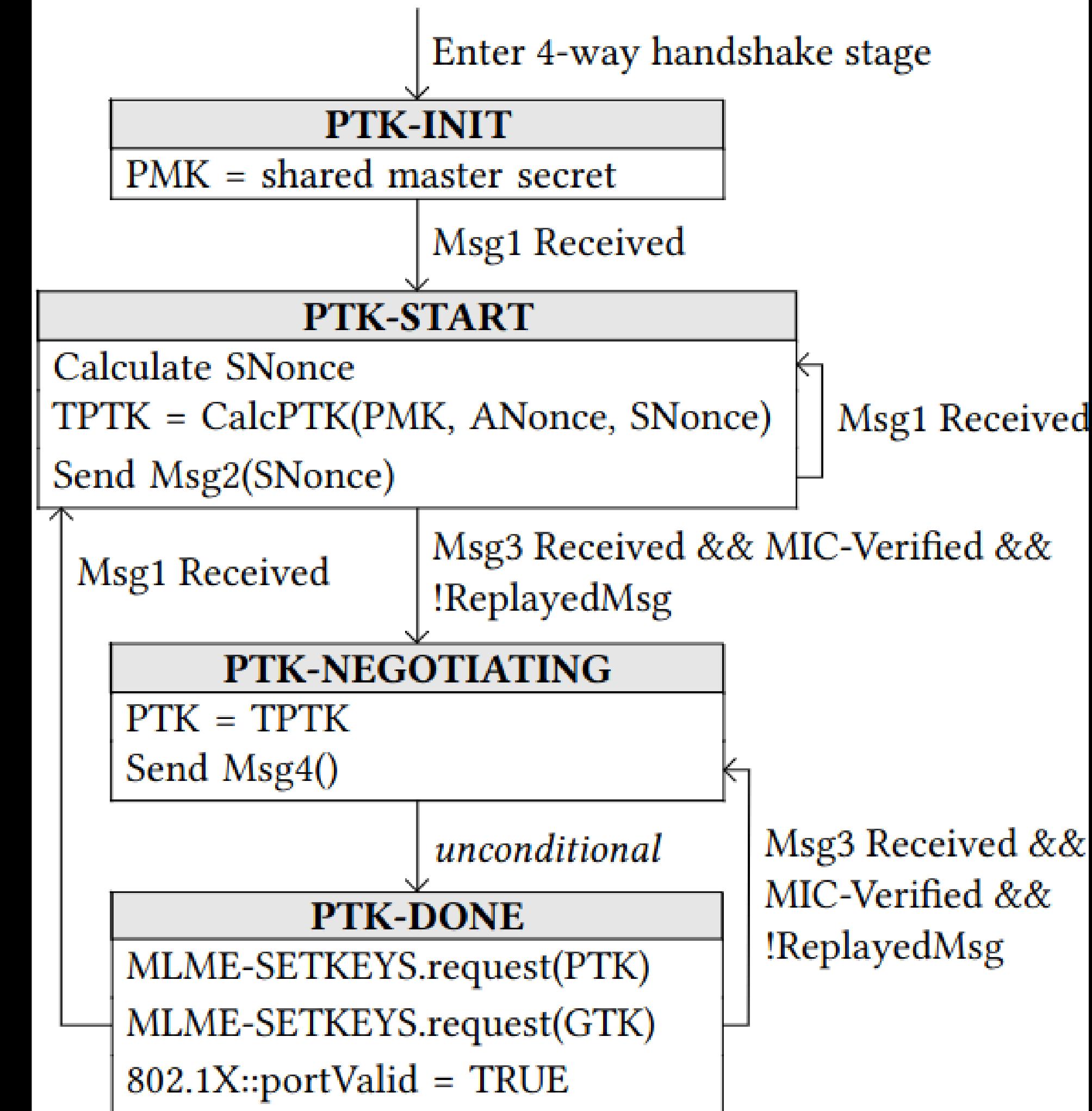
Mitigation

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.



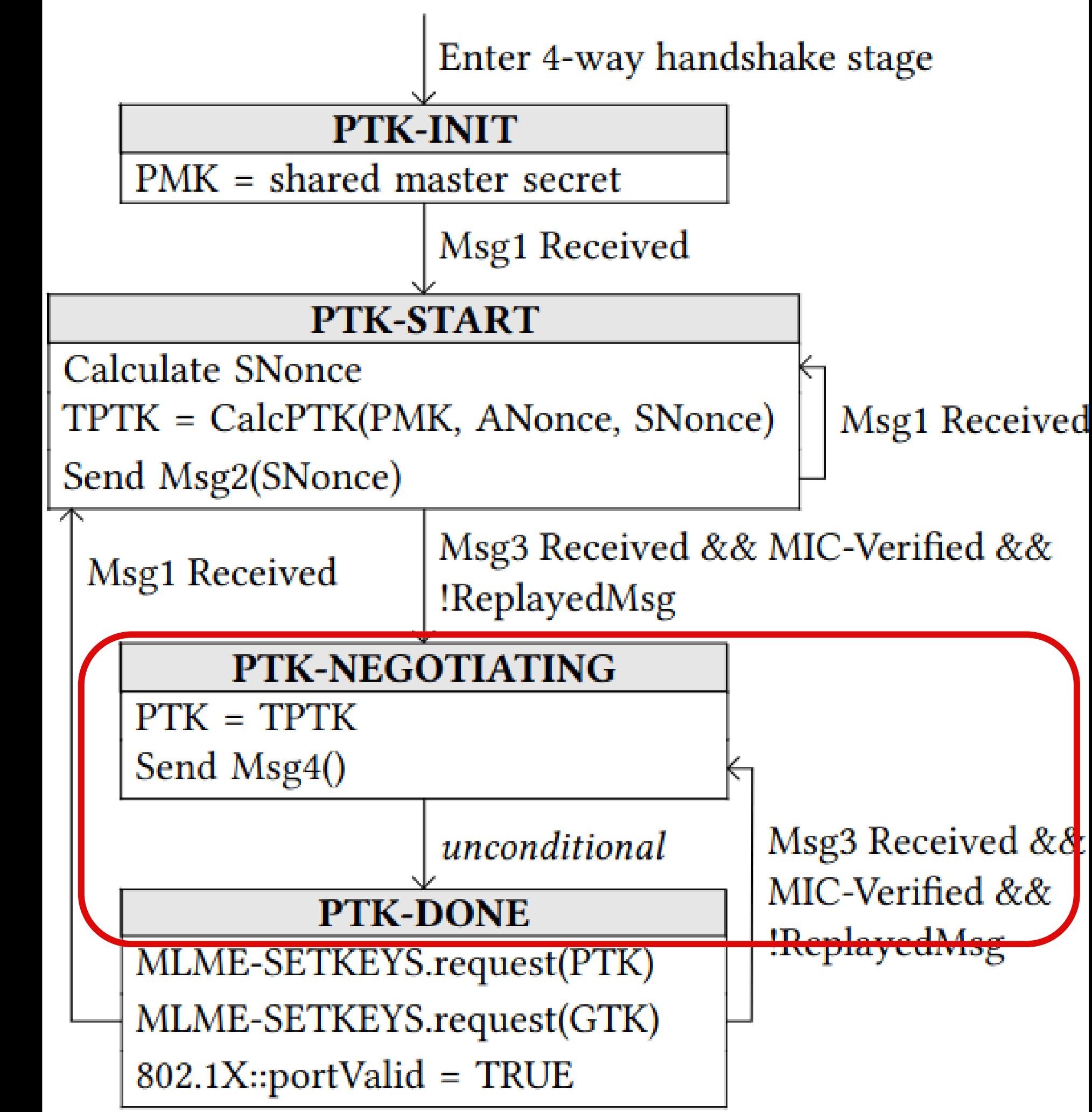
The client will subsequently **reuse packet numbers** when sending frames protected using TKIP, CCMP, or GCMP.



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

WPA2 Handshake: how it works

Suplicant
State Machine





WPA2 Handshake



KRACK Attack



LAB 0: Simulation



Mininet



LAB 1: Access Point (AP) testing

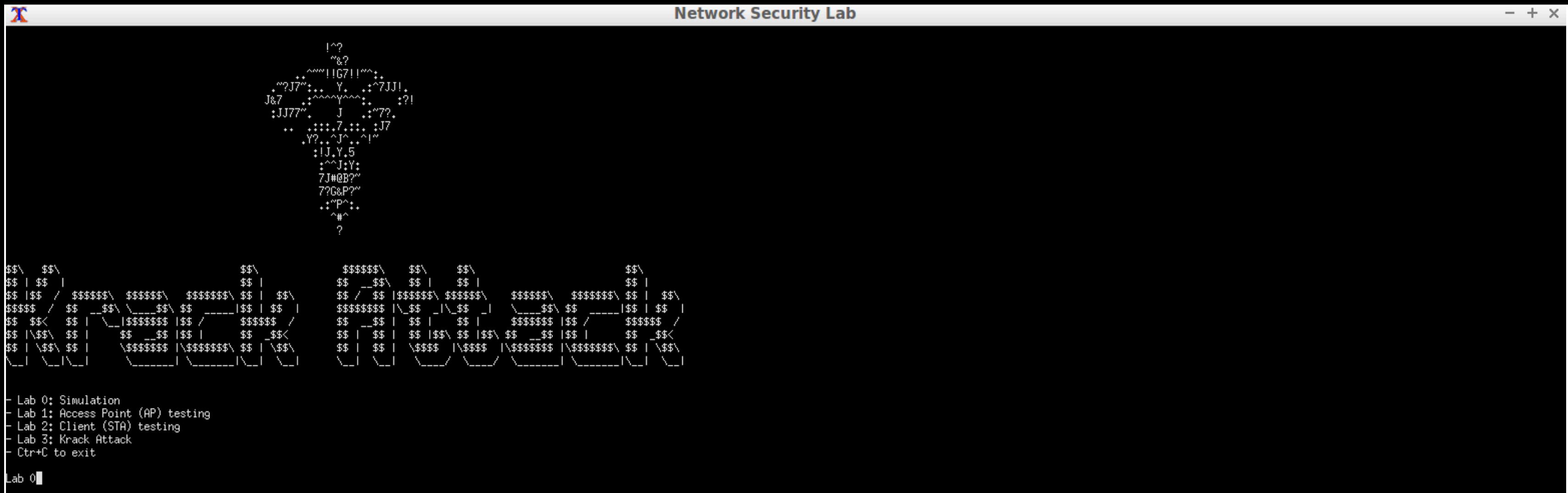


LAB 2: Client (STA) testing



Mitigation

LABS

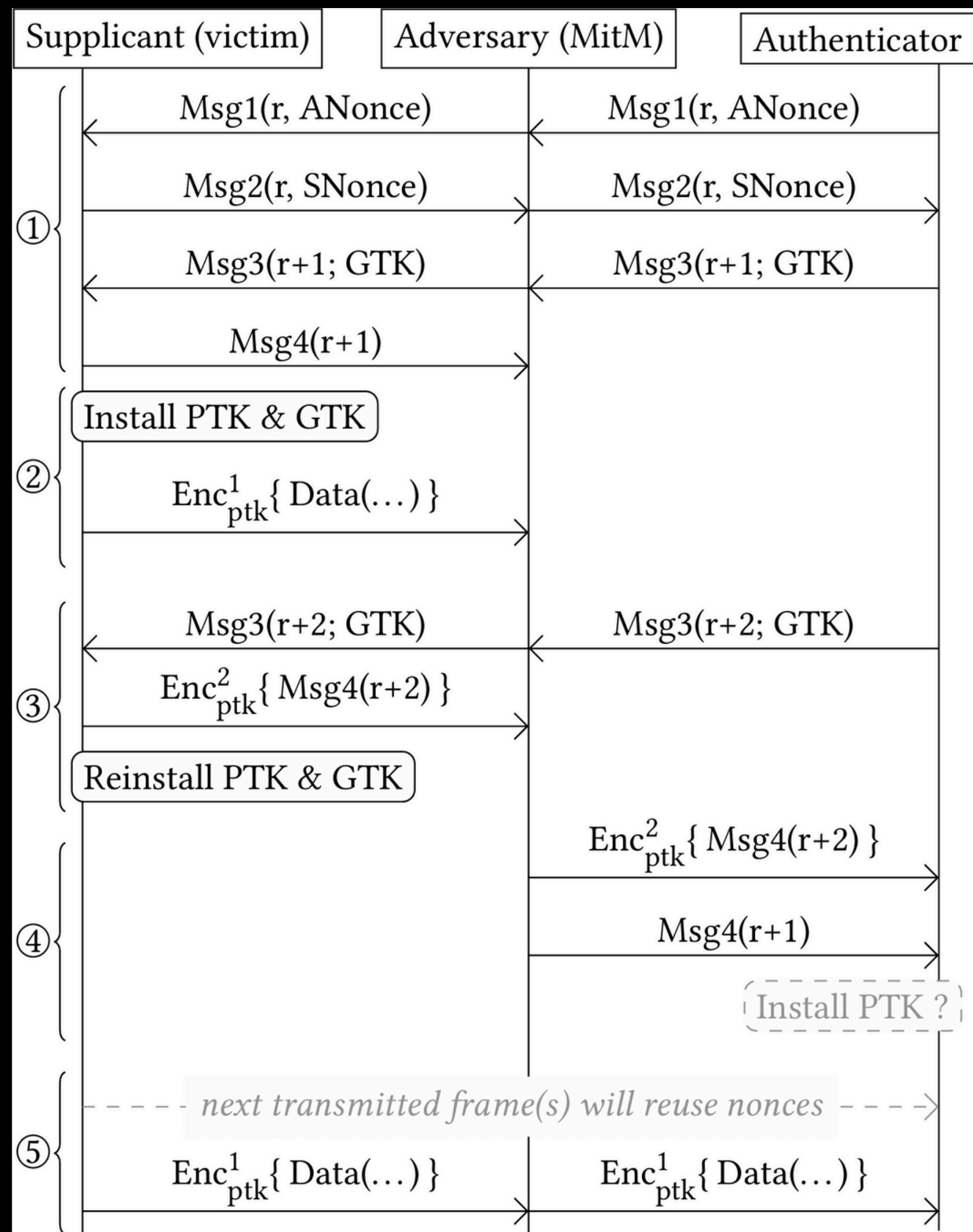
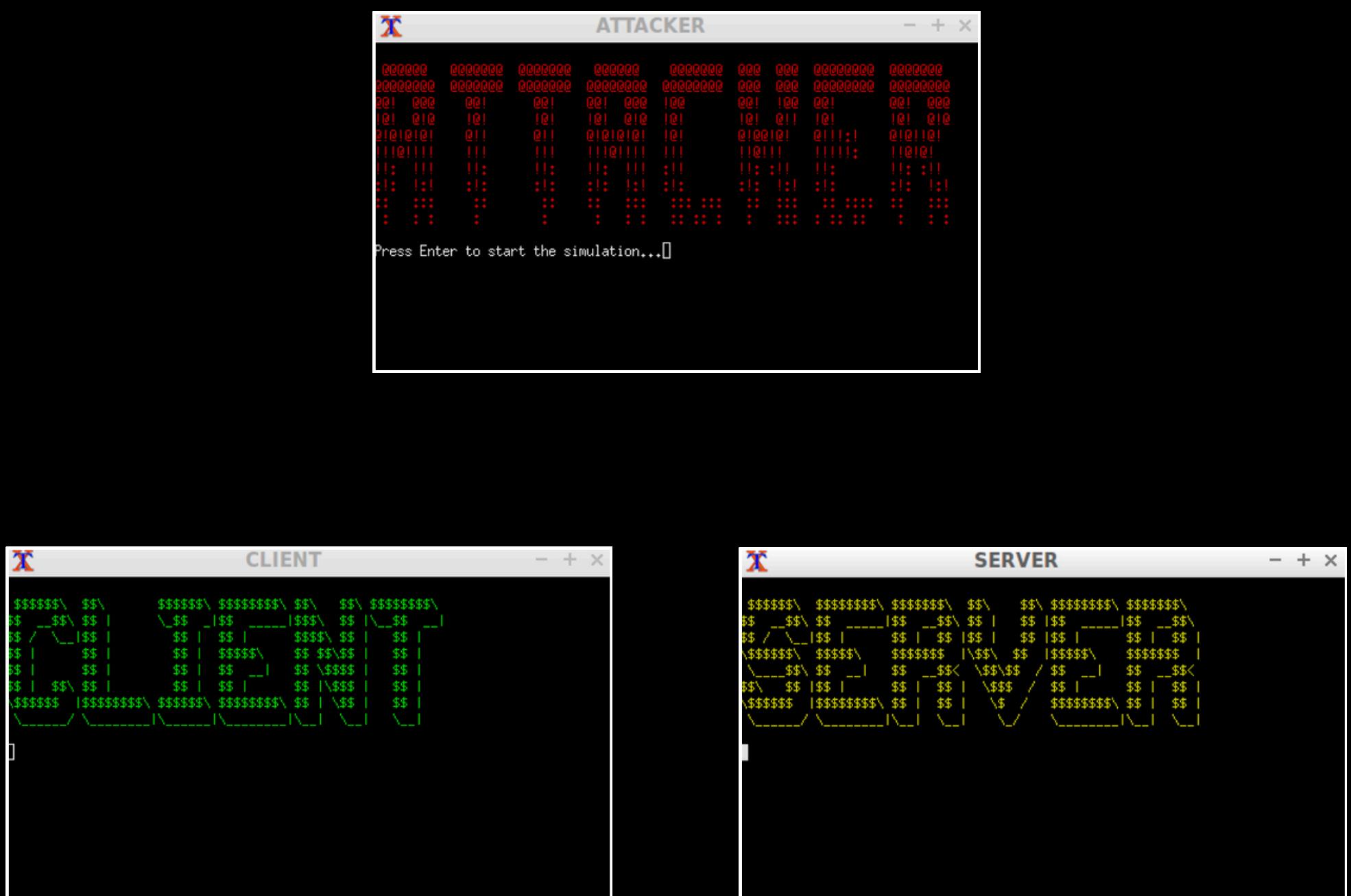


```
krack@krack:~$ cd network-security-lab
```

```
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

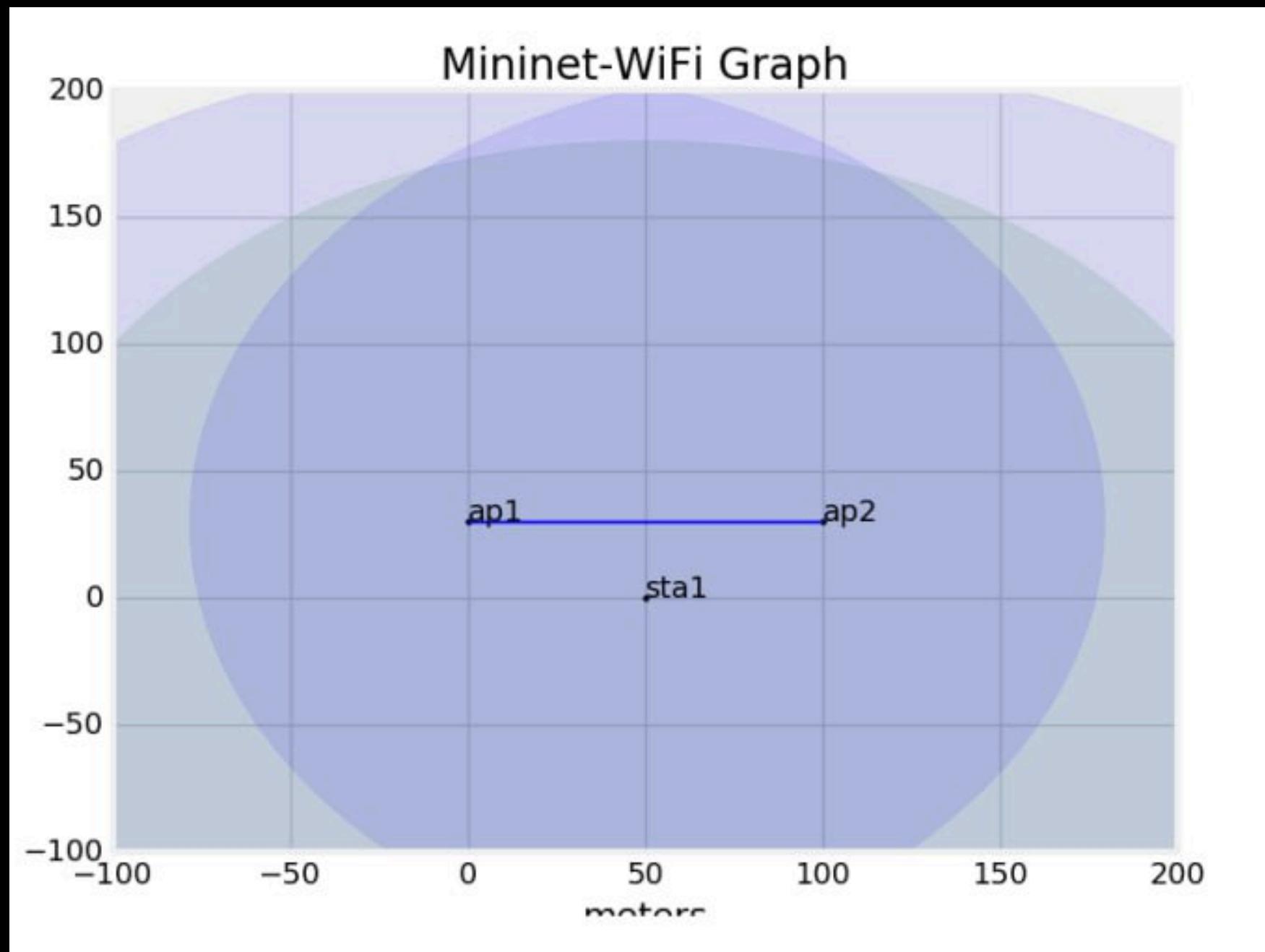


LAB 2: Client (STA) testing



Mitigation

LAB 1: Access Point (AP) testing



Scenario

GOAL

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



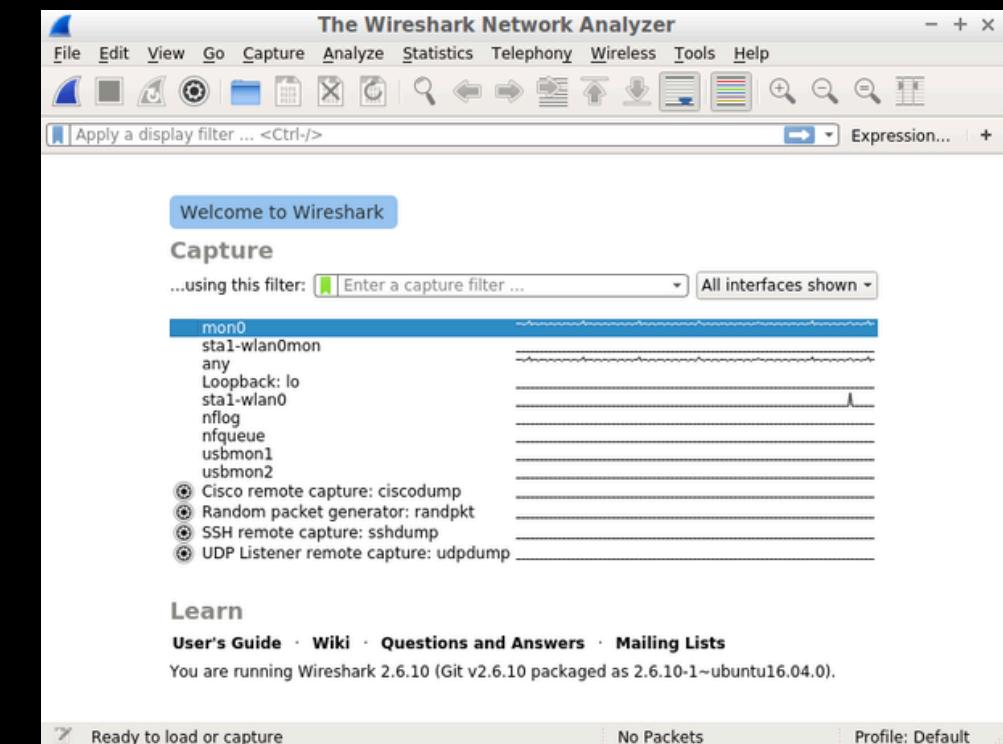
As AP2 implements the protocol IEEE 802.11r, Fast BSS Transition (FT)

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



Network Security Lab

```
*** Creating nodes
*** Configuring AP settings
*** Configuring Propagation Model
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Linking nodes
*** Plotting Graph
*** Starting network
Using hostapd v2.7-devel-hostap_2.6-930-g87ad672+
Using kernel v4.15.0-45-generic

*** Running CLI
*** Starting CLI:
mininet-wifi> █
```

"KrackAttack: sta1"

```
[23:12:08] Note: disable Wi-Fi in your network manager so it doesn't interfere with this script
Successfully initialized wpa_supplicant
sta1-wlan0: SME: Trying to authenticate with 02:11:11:11:11:11 (SSID='testnetwork' freq=2412 MHz)
[23:12:13] Detected Authentication frame, clearing client state
sta1-wlan0: Trying to associate with 02:11:11:11:11:11 (SSID='testnetwork' freq=2412 MHz)
[23:12:13] Detected Authentication frame, clearing client state
sta1-wlan0: Associated with 02:11:11:11:11:11
sta1-wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
[23:12:13] Detected normal association frame
sta1-wlan0: WPA: Key negotiation completed with 02:11:11:11:11:11 [PTK=CCMP GTK=CCMP]
sta1-wlan0: CTRL-EVENT-CONNECTED - Connection to 02:11:11:11:11:11 completed [id=0 id_str=]
sta1-wlan0: WPA: Group rekeying completed with 02:11:11:11:11:11 [GTK=CCMP]
[23:21:51] AP transmitted data using IV=1 (seq=38)
sta1-wlan0: WPA: Group rekeying completed with 02:11:11:11:11:11 [GTK=CCMP]
[23:31:51] AP transmitted data using IV=2 (seq=60)
```

"wpa_cli: sta1"

```
root@krack:/~network-security-lab/labs/lab-1# █
```

LAB 1: Access Point (AP) testing

The image shows two terminal windows side-by-side. The left window is titled "Network Security Lab" and contains the following text:

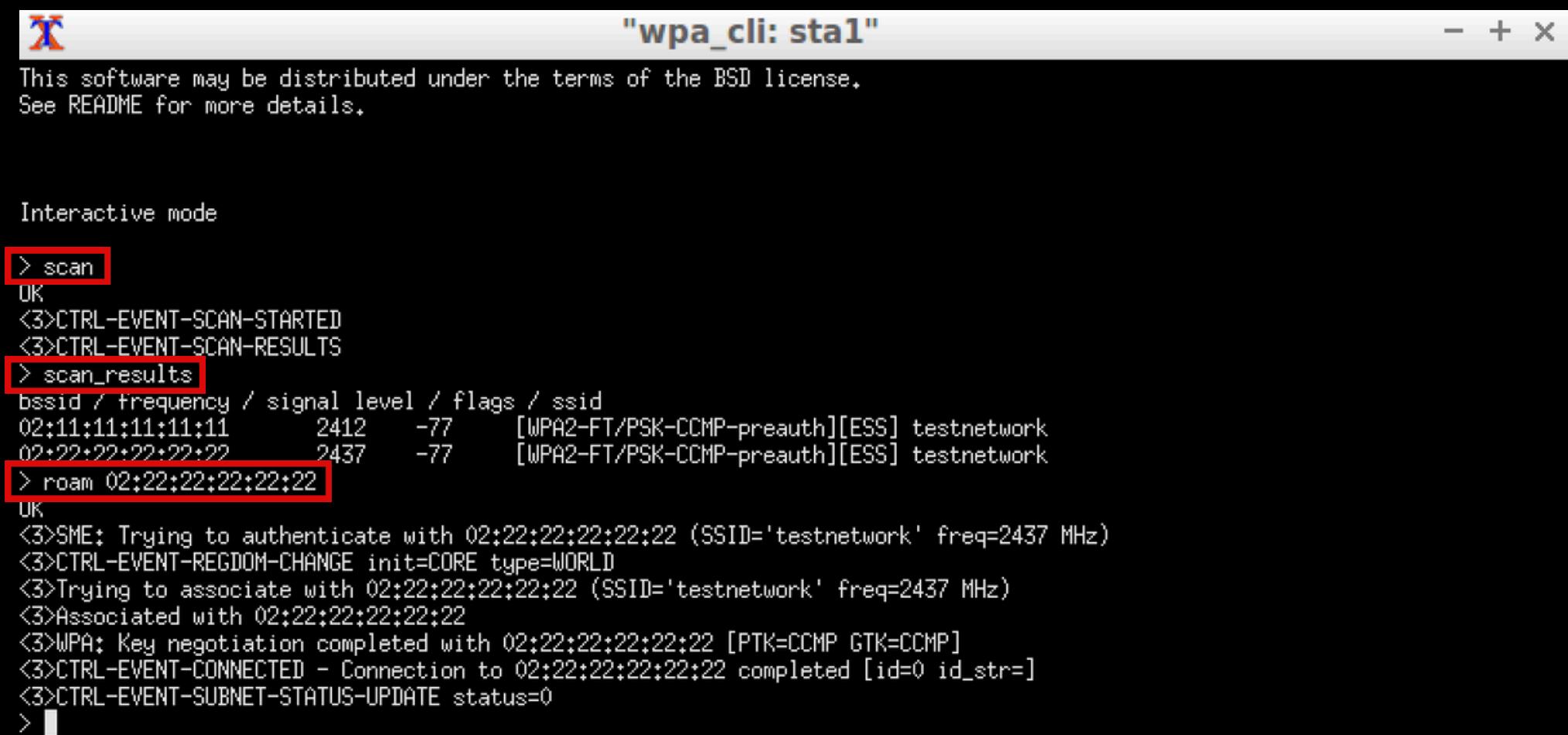
```
*** Creating nodes
*** Configuring AP settings
*** Configuring Propagation Model
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Linking nodes
*** Plotting Graph
*** Starting network
Using hostapd v2.7-devel-hostap_2_6-930-g987ad672+
Using kernel v4.15.0-45-generic
*** Running CLI
*** Starting CLI*
mininet-wifi> sta1 arping -I sta1-wlan0 -c 10 10.0.0.101
arping 10.0.0.101
12 bytes from 02:11:11:11:11 (10.0.0.101): index=0 time=10.601 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=1 time=9.168 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=2 time=12.436 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=3 time=15.926 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=4 time=7.519 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=5 time=13.031 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=6 time=16.924 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=7 time=19.444 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=8 time=16.980 msec
12 bytes from 02:11:11:11:11 (10.0.0.101): index=9 time=16.604 msec
--- 10.0.0.101 statistics ---
10 packets transmitted, 10 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 7.519/13.863/19.444/3.710 ms
mininet-wifi> 
```

The right window is titled "KrackAttack: sta1" and contains the following text:

```
[00:23:20] Note: disable Wi-Fi in your network manager so it doesn't interfere with this script
Successfully initialized wpa_supplicant
sta1-wlan0: SME: Trying to authenticate with 02:11:11:11:11 (SSID='testnetwork' freq=2412 MHz)
[00:23:24] Detected Authentication frame, clearing client state
sta1-wlan0: Trying to associate with 02:11:11:11:11 (SSID='testnetwork' freq=2412 MHz)
[00:23:25] Detected Authentication frame, clearing client state
[00:23:25] Detected normal association frame
sta1-wlan0: Associated with 02:11:11:11:11
sta1-wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
sta1-wlan0: WPA: Key negotiation completed with 02:11:11:11:11 [PTK=CCMP GTK=CCMP]
sta1-wlan0: CTRL-EVENT-CONNECTED - Connection to 02:11:11:11:11 completed [id=0 id_str=]
[00:23:38] AP transmitted data using IV=1 (seq=13)
[00:23:39] AP transmitted data using IV=2 (seq=15)
[00:23:40] AP transmitted data using IV=3 (seq=17)
[00:23:42] AP transmitted data using IV=4 (seq=19)
[00:23:42] AP transmitted data using IV=5 (seq=21)
[00:23:43] AP transmitted data using IV=6 (seq=23)
[00:23:44] AP transmitted data using IV=7 (seq=25)
[00:23:45] AP transmitted data using IV=8 (seq=27)
[00:23:46] AP transmitted data using IV=9 (seq=29)
[00:23:47] AP transmitted data using IV=10 (seq=31)
```

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

LAB 1: Access Point (AP) testing



The screenshot shows a terminal window titled "wpa_cli: sta1". The window contains the following text:

```
This software may be distributed under the terms of the BSD license.  
See README for more details.  
  
Interactive mode  
> scan  
OK  
<3>CTRL-EVENT-SCAN-STARTED  
<3>CTRL-EVENT-SCAN-RESULTS  
> scan_results  
bssid / frequency / signal level / flags / ssid  
02:11:11:11:11:11 2412 -77 [WPA2-FT/PSK-CCMP-preauth][ESS] testnetwork  
02:22:22:22:22:22 2437 -77 [WPA2-FT/PSK-CCMP-preauth][ESS] testnetwork  
> roam 02:22:22:22:22:22  
OK  
<3>SME: Trying to authenticate with 02:22:22:22:22 (SSID='testnetwork' freq=2437 MHz)  
<3>CTRL-EVENT-REGDOM-CHANGE init=CORE type=WORLD  
<3>Trying to associate with 02:22:22:22:22 (SSID='testnetwork' freq=2437 MHz)  
<3>Associated with 02:22:22:22:22  
<3>WPA: Key negotiation completed with 02:22:22:22:22 [PTK=CCMP GTK=CCMP]  
<3>CTRL-EVENT-CONNECTED - Connection to 02:22:22:22:22 completed [id=0 id_str=]  
<3>CTRL-EVENT-SUBNET-STATUS-UPDATE status=0  
> █
```

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

LAB 1: Access Point (AP) testing

The screenshot shows two terminal windows. The left window, titled "Network Security Lab", displays the output of a "arping" command to 10.0.0.101. The right window, titled "KrackAttack: sta1", shows a log of wireless traffic. A red box highlights several error messages indicating IV reuse and AP vulnerability. A red arrow points from the text "AP vulnerable!" to the highlighted area.

```
*** Plotting Graph
*** Starting network
Using hostapd v2.7-devel-hostap_2.6-930-g87ad672+
Using kernel v4.15.0-45-generic

*** Running CLI
*** Starting CLI:
mininet-wifi> sta1 arping -I sta1-wlan0 -c 10 10.0.0.101
ARPING 10.0.0.101
42 bytes from 02:11:11:11:11 (10.0.0.101): index=0 time=10.601 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=1 time=9.168 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=2 time=12.436 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=3 time=15.926 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=4 time=7.519 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=5 time=13.031 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=6 time=16.924 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=7 time=19.444 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=8 time=16.980 msec
42 bytes from 02:11:11:11:11 (10.0.0.101): index=9 time=16.604 msec
--- 10.0.0.101 statistics ---
10 packets transmitted, 10 packets received, 0% unanswered (0 extra)
rtt min/avg/max/std-dev = 7.519/13.024/24.770/5.457 ms
mininet-wifi> sta1 arping -I sta1-wlan0 -c 10 10.0.0.102
ARPING 10.0.0.102
42 bytes from 02:22:22:22:22 (10.0.0.102): index=0 time=24.770 msec
Timeout
Timeout
Timeout
Timeout
Timeout
Timeout
Timeout
Timeout
42 bytes from 02:22:22:22:22 (10.0.0.102): index=1 time=13.856 msec
--- 10.0.0.102 statistics ---
10 packets transmitted, 2 packets received, 80% unanswered (0 extra)
rtt min/avg/max/std-dev = 13.856/19.313/24.770/5.457 ms
mininet-wifi>
```

```
[00:28:40] AP transmitted data using IV=1 (seq=64)
[00:28:40] IV reuse detected (IV=1, seq=64), AP is vulnerable!
[00:28:41] Replaying Reassociation Request
[00:28:41] Detected FT reassociation frame
[00:28:41] AP transmitted data using IV=1 (seq=67)
[00:28:41] IV reuse detected (IV=1, seq=67), AP is vulnerable!
[00:28:42] Replaying Reassociation Request
[00:28:42] Detected FT reassociation frame
[00:28:42] AP transmitted data using IV=1 (seq=70)
[00:28:42] IV reuse detected (IV=1, seq=70), AP is vulnerable!
[00:28:43] Replaying Reassociation Request
[00:28:43] Detected FT reassociation frame
[00:28:43] AP transmitted data using IV=1 (seq=73)
[00:28:43] IV reuse detected (IV=1, seq=73), AP is vulnerable!
[00:28:44] Replaying Reassociation Request
[00:28:44] Detected FT reassociation frame
[00:28:44] AP transmitted data using IV=1 (seq=76)
[00:28:44] IV reuse detected (IV=1, seq=76), AP is vulnerable!
[00:28:45] Replaying Reassociation Request
[00:28:45] AP transmitted data using IV=2 (seq=78)
[00:28:45] Detected FT reassociation frame
[00:28:46] Replaying Reassociation Request
[00:28:46] Detected FT reassociation frame
[00:28:47] Replaying Reassociation Request
[00:28:47] Detected FT reassociation frame
[00:28:48] Replaying Reassociation Request
[00:28:48] Detected FT reassociation frame
[00:28:49] Replaying Reassociation Request
[00:28:49] Detected FT reassociation frame
[00:28:50] Replaying Reassociation Request
[00:28:50] Detected FT reassociation frame
[00:28:51] Replaying Reassociation Request
[00:28:51] Detected FT reassociation frame
[00:28:52] Replaying Reassociation Request
[00:28:52] Detected FT reassociation frame
[00:28:53] Replaying Reassociation Request
[00:28:53] Detected FT reassociation frame
[00:28:54] Replaying Reassociation Request
[00:28:54] Detected FT reassociation frame
[00:28:55] Replaying Reassociation Request
```

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

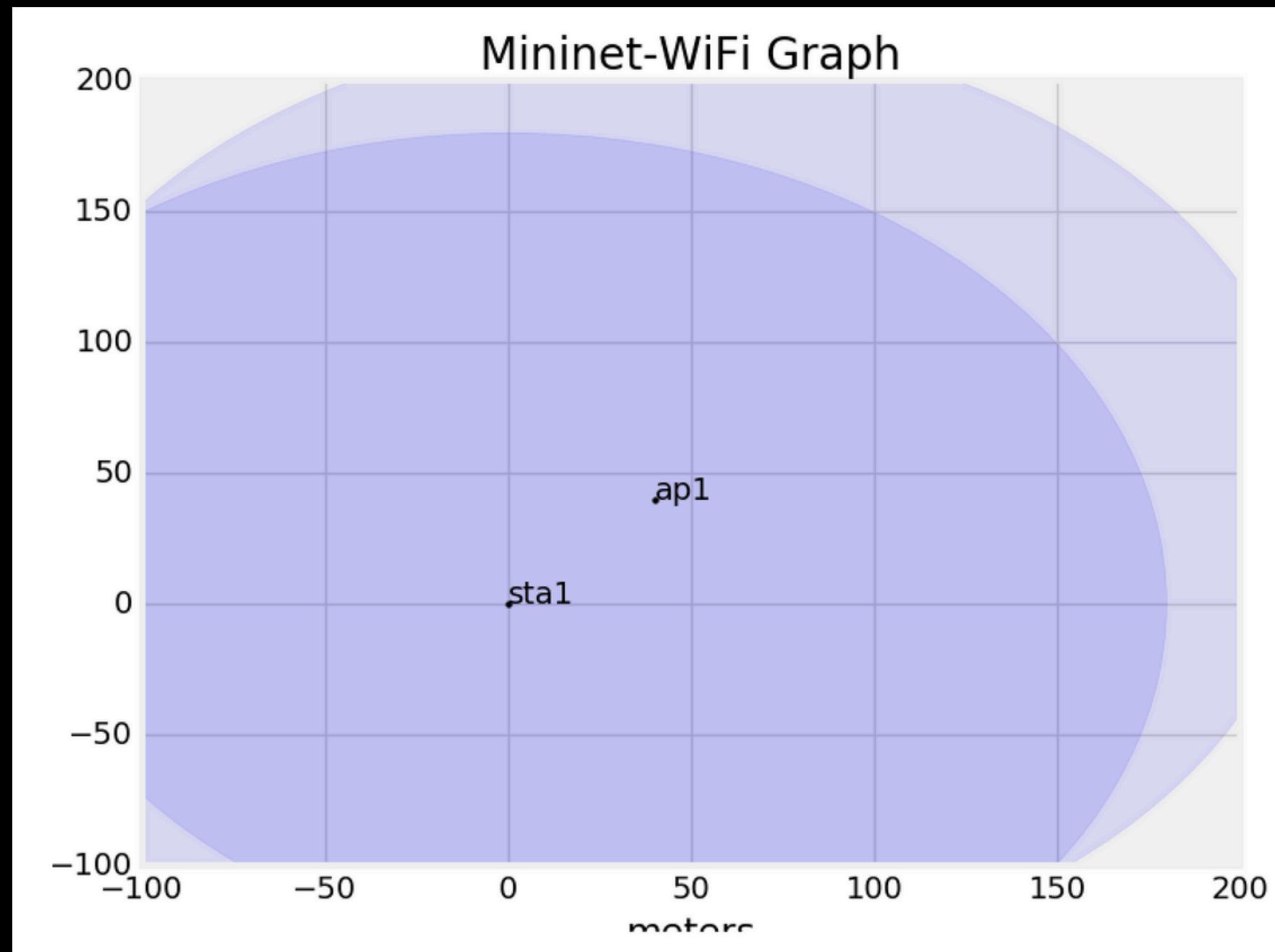


LAB 2: Client (STA) testing



Mitigation

LAB 2: Client (STA) testing



Scenario

GOAL

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



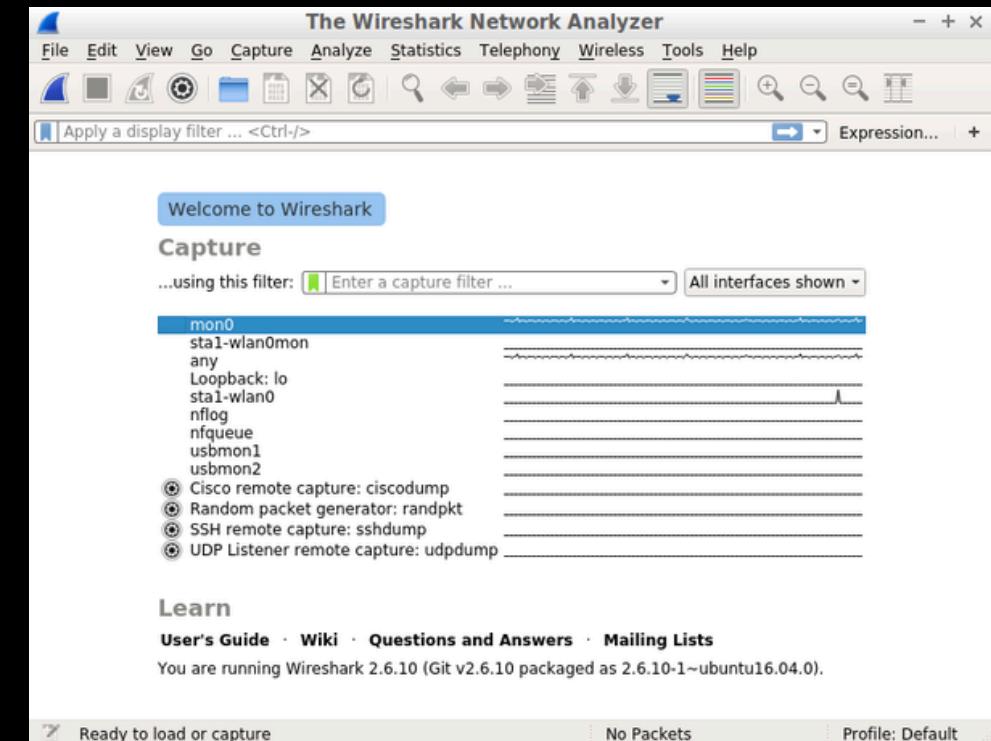
As the client uses a vulnerable version of
wpa_supplicant

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



```
X Network S
*** Creating nodes
*** Configuring Propagation Model
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Plotting Graph
*** Starting network
Using wpa_supplicant v2.10

Using kernel v4.15.0-45-generic

*** Running CLI
*** Starting CLI:
mininet-wifi> []
```

```
X "AP: ap1"
[01:04:07] Note: disable Wi-Fi in network manager & disable hardware encryption.
Both may interfere with this script.
[01:04:07] Starting hostapd ...
Configuration file: /home/krack/network-security-lab/krackattacks-scripts/krackattack/hostapd.conf
Using interface ap1-wlan0 with hwaddr 02:11:11:11:11 and ssid "testnetwork"
ap1-wlan0: interface state UNINITIALIZED->ENABLED
ap1-wlan0: AP-ENABLED
[01:04:08] Ready. Connect to this Access Point to start the tests. Make sure the
client requests an IP using DHCP!
[01:04:09] Reset PN for GTK
[01:04:11] Reset PN for GTK
[01:04:13] Reset PN for GTK
[01:04:15] Reset PN for GTK
[01:04:17] Reset PN for GTK
[01:04:19] Reset PN for GTK
[01:04:21] Reset PN for GTK
[01:04:23] Reset PN for GTK
[01:04:25] Reset PN for GTK
```

```
X "Connection: sta1"
root@krack:/home/krack/network-security-lab/labs/lab-2# []
```

LAB 2: Client (STA) testing

The image displays three terminal windows illustrating the setup and operation of a wireless network. The first window shows the configuration of a 'Network S' node, including the creation of nodes, configuration of propagation models, and the use of wpa_supplicant v2.10. The second window shows the logs for an Access Point (AP) named 'ap1', detailing its connection to a client. The third window shows the logs for a client station (sta1) attempting to connect to the AP, where it successfully negotiates a key.

```
Network S
*** Creating nodes
*** Configuring Propagation Model
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Plotting Graph
*** Starting network
Using wpa_supplicant v2.10

Using kernel v4.15.0-45-generic
*** Running CLI
*** Starting CLI:
mininet-wifi> [ ]
```

wpa_supplicant
v2.10 not
vulnerable

```
"AP: ap1"
[01:31:25] Reset PN for GTK
[01:31:27] Reset PN for GTK
[01:31:30] Reset PN for GTK
[01:31:32] Reset PN for GTK
[01:31:34] Reset PN for GTK
[01:31:36] Reset PN for GTK
[01:31:38] Reset PN for GTK
[01:31:40] Reset PN for GTK
[01:31:42] Reset PN for GTK
[01:31:44] Reset PN for GTK
[01:31:46] Reset PN for GTK
[01:31:48] Reset PN for GTK
ap1-wlan0: STA 02:00:00:00:01:00 IEEE 802.11: authenticated
ap1-wlan0: STA 02:00:00:00:01:00 IEEE 802.11: associated (aid 1)
ap1-wlan0: AP-STA-CONNECTED 02:00:00:00:01:00
ap1-wlan0: STA 02:00:00:00:01:00 RADIUS: starting accounting session 56AD57AB7617E506
[01:31:50] 02:00:00:00:01:00: 4-way handshake completed (RSN)
[01:31:50] Reset PN for GTK
[01:31:50] 02:00:00:00:01:00: sending a new 4-way message 3 where the GTK has a zero RSC
[01:31:50] 02:00:00:00:01:00: received a new message 4
[01:31:52] Reset PN for GTK
[01:31:52] 02:00:00:00:01:00: sending a new 4-way message 3 where the GTK has a zero RSC
[01:31:52] 02:00:00:00:01:00: received a new message 4
```

```
"Connection: sta1"
root@krack:/network-security-lab/labs/lab-2# wpa_supplicant -i sta1-wlan0 -c client_network.conf
Successfully initialized wpa_supplicant
sta1-wlan0: SME: Trying to authenticate with 02:11:11:11:11 (SSID='testnetwork' freq=2412 MHz)
sta1-wlan0: Trying to associate with 02:11:11:11:11 (SSID='testnetwork' freq=2412 MHz)
sta1-wlan0: Associated with 02:11:11:11:11
sta1-wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
sta1-wlan0: WPA: Key negotiation completed with 02:11:11:11:11 [PTK=CCMP GTK=CCMP]
sta1-wlan0: CTRL-EVENT-CONNECTED - Connection to 02:11:11:11:11 completed [id=0 id_str=]
sta1-wlan0: WPA: Key negotiation completed with 02:11:11:11:11 [PTK=CCMP GTK=CCMP]
sta1-wlan0: WPA: Key negotiation completed with 02:11:11:11:11 [PTK=CCMP GTK=CCMP]
```

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

```
Network S
*** Creating nodes
*** Configuring Propagation Model
*** Configuring wifi nodes
*** Connecting to wmediumd server /var/run/wmediumd.sock
*** Plotting Graph
*** Starting network
Using wpa_supplicant v2.5
Using kernel v4.15.0-45-generic
*** Running CLI
*** Starting CLI:
mininet-wifi> []
```

wpa_supplicant
v2.5 vulnerable

```
"AP: ap1"
[01:48:33] Ready. Connect to this Access Point to start the tests. Make sure the
client requests an IP using DHCP!
[01:48:34] Reset PN for GTK
[01:48:35] Reset PN for GTK
[01:48:38] Reset PN for GTK
[01:48:40] Reset PN for GTK
[01:48:42] Reset PN for GTK
[01:48:44] Reset PN for GTK
[01:48:47] Reset PN for GTK
[01:48:49] Reset PN for GTK
[01:48:51] Reset PN for GTK
ap1-wlan0: STA 02:00:00:00:01:00 IEEE 802.11: authenticated
ap1-wlan0: STA 02:00:00:00:01:00 IEEE 802.11: associated (aid 1)
ap1-wlan0: AP-STA-CONNECTED 02:00:00:00:01:00
ap1-wlan0: STA 02:00:00:00:01:00 RADIUS: starting accounting session 6951881B8224D704
[01:48:52] 02:00:00:00:01:00: 4-way handshake completed (RSN)
[01:48:53] Reset PN for GTK
[01:48:53] 02:00:00:00:01:00: sending a new 4-way message 3 where the GTK has a zero RSC
[01:48:53] 02:00:00:00:01:00: received a new message 4
[01:48:53] 02:00:00:00:01:00: usage of all-zero key detected (IV=1, seq=4). Client (re)installs an all-zero key in the 4-wa
y handshake (this is very bad).
[01:48:53] 02:00:00:00:01:00: !!! Other tests are unreliable due to all-zero key usage, please fix this vulnerability first
!!!
[01:48:55] Reset PN for GTK
[01:48:57] Reset PN for GTK
```

CLIENT / STA vulnerable!

```
"Connection: sta1"
root@krack:/network-security-lab/labs/lab-2# wpa_supplicant -i sta1-wlan0 -c client_network.conf
Successfully initialized wpa_supplicant
sta1-wlan0: SME: Trying to authenticate with 02:11:11:11:11:11 (SSID='testnetwork' freq=2412 MHz)
sta1-wlan0: Trying to associate with 02:11:11:11:11:11 (SSID='testnetwork' freq=2412 MHz)
sta1-wlan0: Associated with 02:11:11:11:11:11
sta1-wlan0: WPA: Key negotiation completed with 02:11:11:11:11:11 [PTK=CCMP GTK=CCMP]
sta1-wlan0: CTRL-EVENT-CONNECTED - Connection to 02:11:11:11:11:11 completed [id=0 id_str=]
sta1-wlan0: WPA: Key negotiation completed with 02:11:11:11:11:11 [PTK=CCMP GTK=CCMP]
```

> 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: ForcingNonceReuse 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



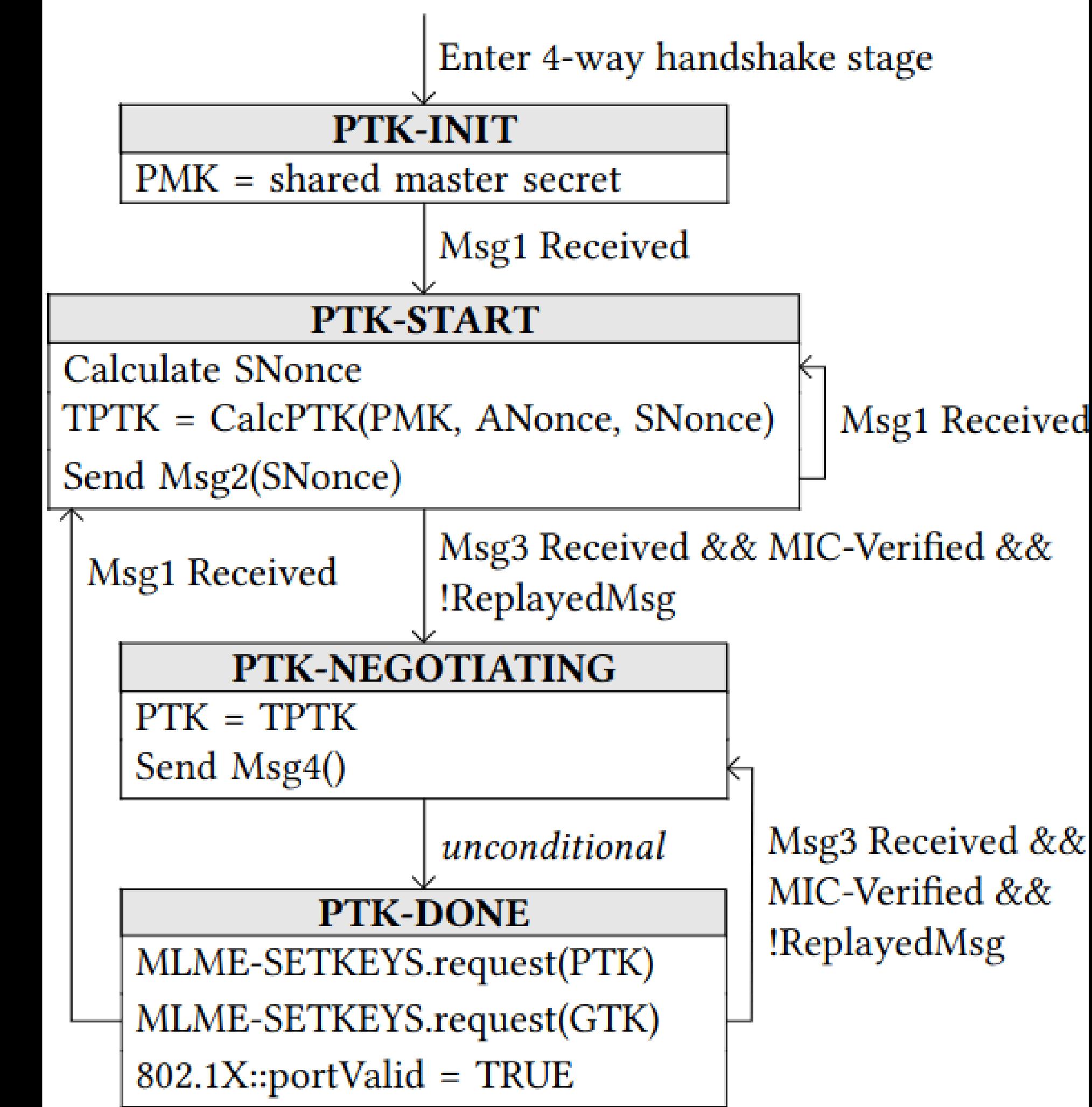
Mitigation

Mitigation

ANY IDEAS ?

Mitigation

Hint: look at client's
wpa_supplicant state
machine



Mitigation

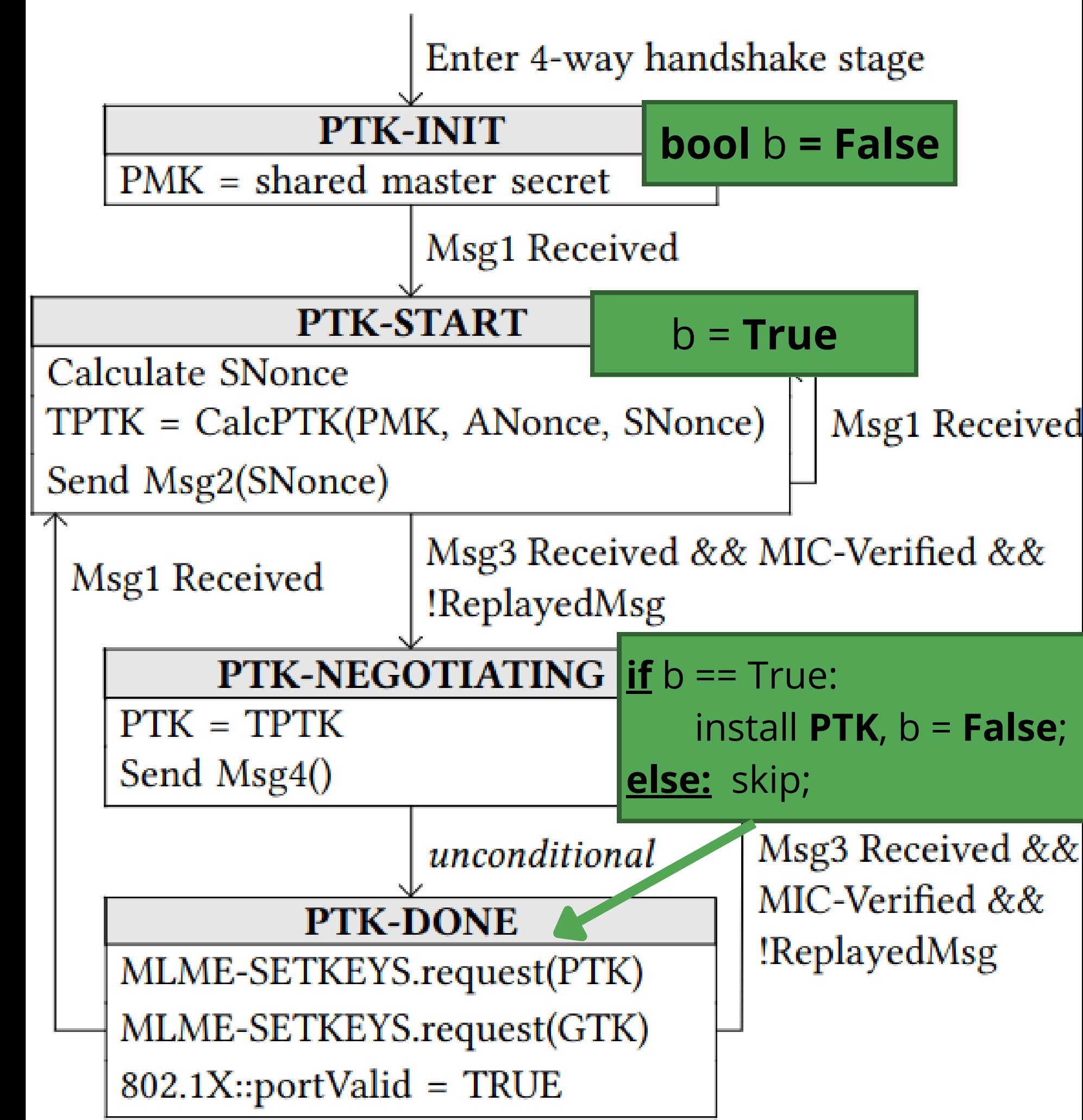
Mathy Vanhoef proposed solution
(and actually implemented)

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.



ZDNET
tomorrow
belongs to those who embrace it
today

trending tech innovation business security advice buying guides

Home / Tech / Security

Google fixes KRACK vulnerability in Android

The KRACK vulnerability is said to be "exceptionally devastating" for Android users.

Written by Zack Whittaker, Contributor
Nov. 7, 2017 at 9:40 a.m. PT

Share:



BLEEPINGCOMPUTER

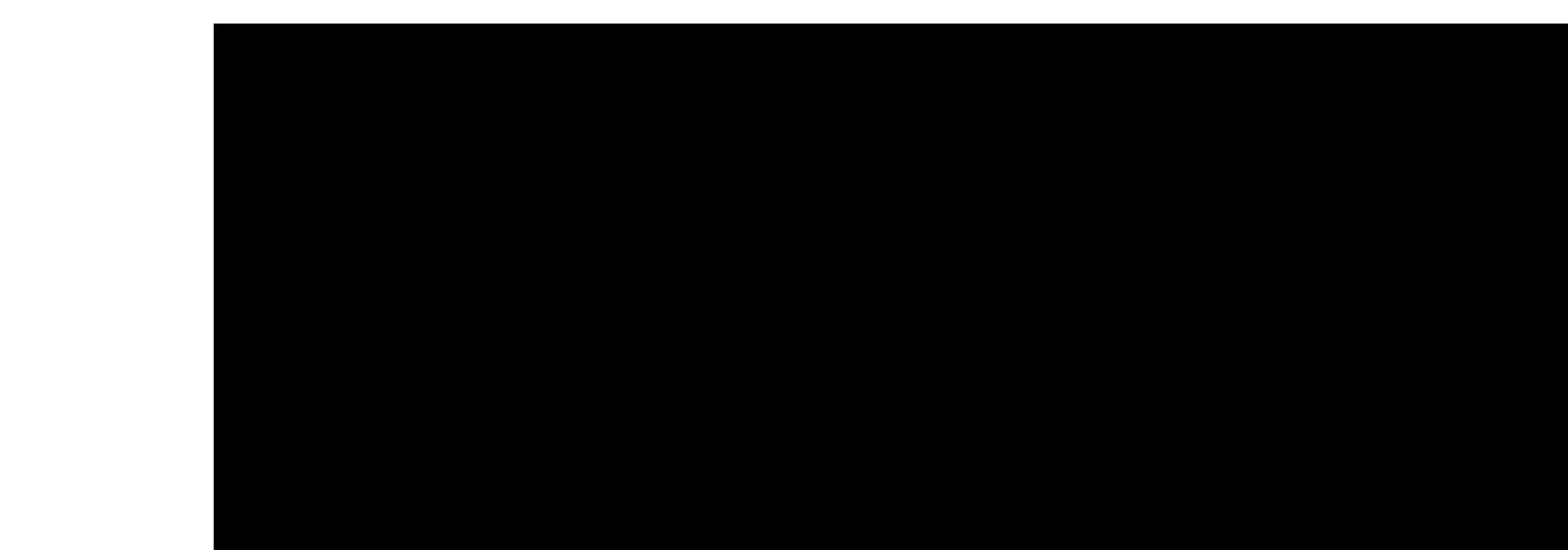
NEWS ▾ TUTORIALS ▾ VIRUS REMOVAL GUIDES ▾ DOWNLOADS ▾ DEALS ▾ VPN

Home > News > Security > Microsoft Quietly Patched the Krack WPA2 Vulnerability Last Week

Microsoft Quietly Patched the Krack WPA2 Vulnerability Last Week

By Lawrence Abrams

October 16, 2017 06:15 PM 4





THANKS !