Lab 11: 4-way Handshake Over EAPoL and WiFi Cracking With aircrack-ng

Before attempting this lab, please make sure you have completed all of the material in the lessons tab.

Create a copy of this google document lastname_lab11 (File > Make a Copy) to record all of your assignment answers in.

> ⚠️ Failure to use answer document properly will result in a 10pt deduction from final score.

The table of contents for this lab is found below.

This week, we talked about EAP and the importance of which encryption protocols we should use to secure our wireless networks. In this lab, we're going to take a closer look at a capture file of a machine connected to the internet over WiFi. We're going to see what information we can extract from this capture session and see how easy it is to launch a dictionary attack against a network capture file.

Part 1: Understanding the 4-way Handshake

Based on this weeks lecture, we know that EAP (Extensible Authentication Protocol) is used frequently in our network connections and allows for many different ways to authenticate on a wireless network.

The 4-way handshake is the process of exchanging 4 messages between an access point (authenticator) and the client device (supplicant) to generate some encryption keys which can be used to encrypt actual data sent over Wireless medium.

To understand the 4 stages of the handshake we need to first understand the keys that are being used:

- MSK (Master Session Key)
- PMK (Pairwise Master Key)
- GMK (Group Master Key)
- PTK (Pairwise Transit Key)
- GTK (Group Temporal Key)
- ANonce
- SNonce
- MIC

Let's start by talking about the keys which are generated during the 4-way handshake and work our way towards the keys and other variables needed in order to generate these keys.

PTK (Pairwise Transit Key):

Pairwise transit key is used to encrypt all unicast traffic between a client station and the access point. PTK is unique between a client station and access point. To generate PTK, client device and access point need the following information.

`PTK = PRF (PMK + Anonce + SNonce + Mac (AA)+ Mac (SA))`

`Anonce` is a random number generated by an access point (authenticator), `Snonce` a random number generated by the client device (supplicant). MAC addresses of supplicant (client device) and MAC address of authenticator (access point). `PRF` is a pseudo-random function which is applied to all the input.

PTK is dependent on another high-level key PMK (pairwise master key) which is discussed below.

GTK (Group Temporal Key):

Group temporal key is used to encrypt all broadcast and multicast traffic between an access point and multiple client devices. GTK is the key which is shared between all client devices associated with 1 access point. For every access point, there will be a different GTK which will be shared between its associated devices.

GTK is dependent on another high-level key GMK (group master key) discussed below.

PMK (Pairwise Master Key):

What is PMK and why we need it? Now we know what is PTK and GTK. PTK is generated with the help of PMK. As we discused above in order to generate PTK, we need the following input.

```
PTK = PRF (PMK + Anonce + SNonce + Mac (AA)+ Mac (SA))
```

Pairwise master is key generated from master session key (MSK). In case of WPA2/PSK when device authenticates with an access point the PSK becomes PMK.

> Point to Remember: PMK resides on all stations as in AP and client devices, so we do not need to share this information. We use this information to create PTK which are used for unicast data encryption.

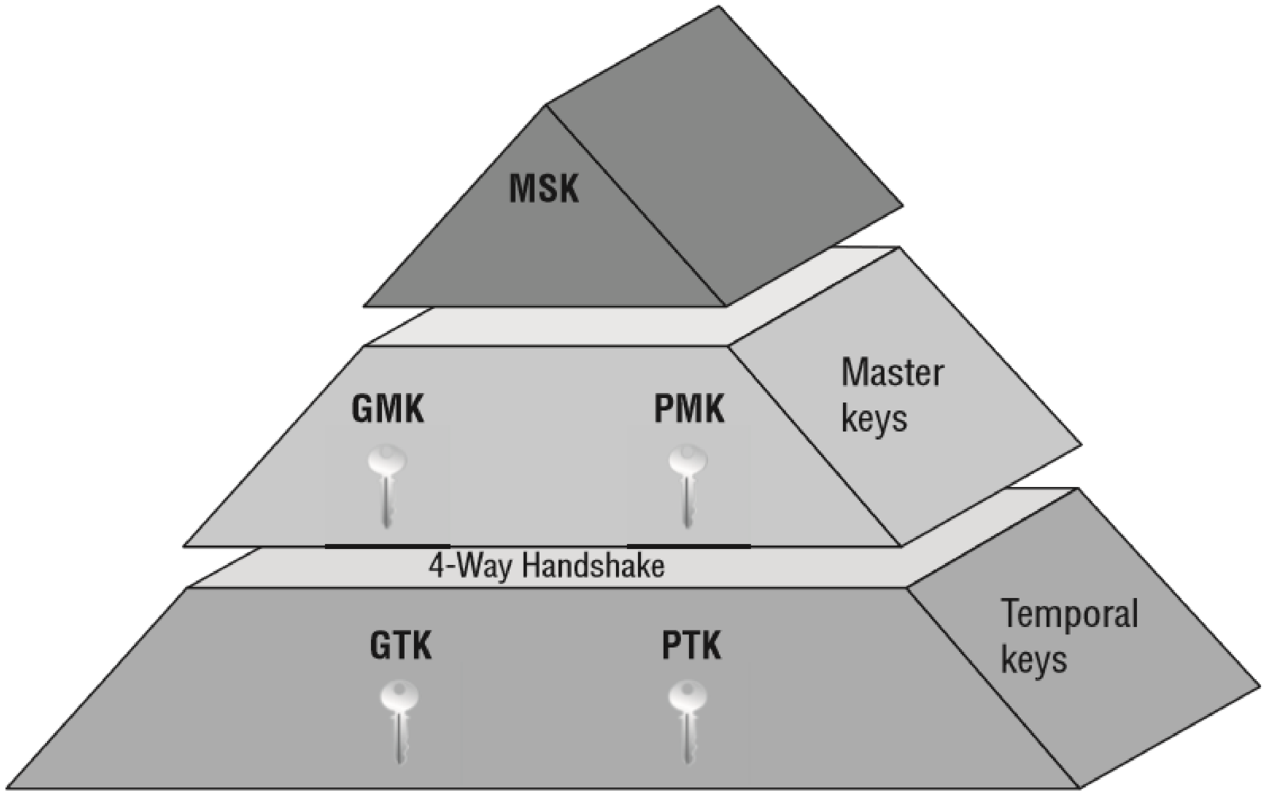
GMK (Group Master Key):

Group master key is used in a 4-way handshake to create GTK discussed above. GTK is generated on every access point and shared with the devices connected to this AP.

MSK (Master Session Key):

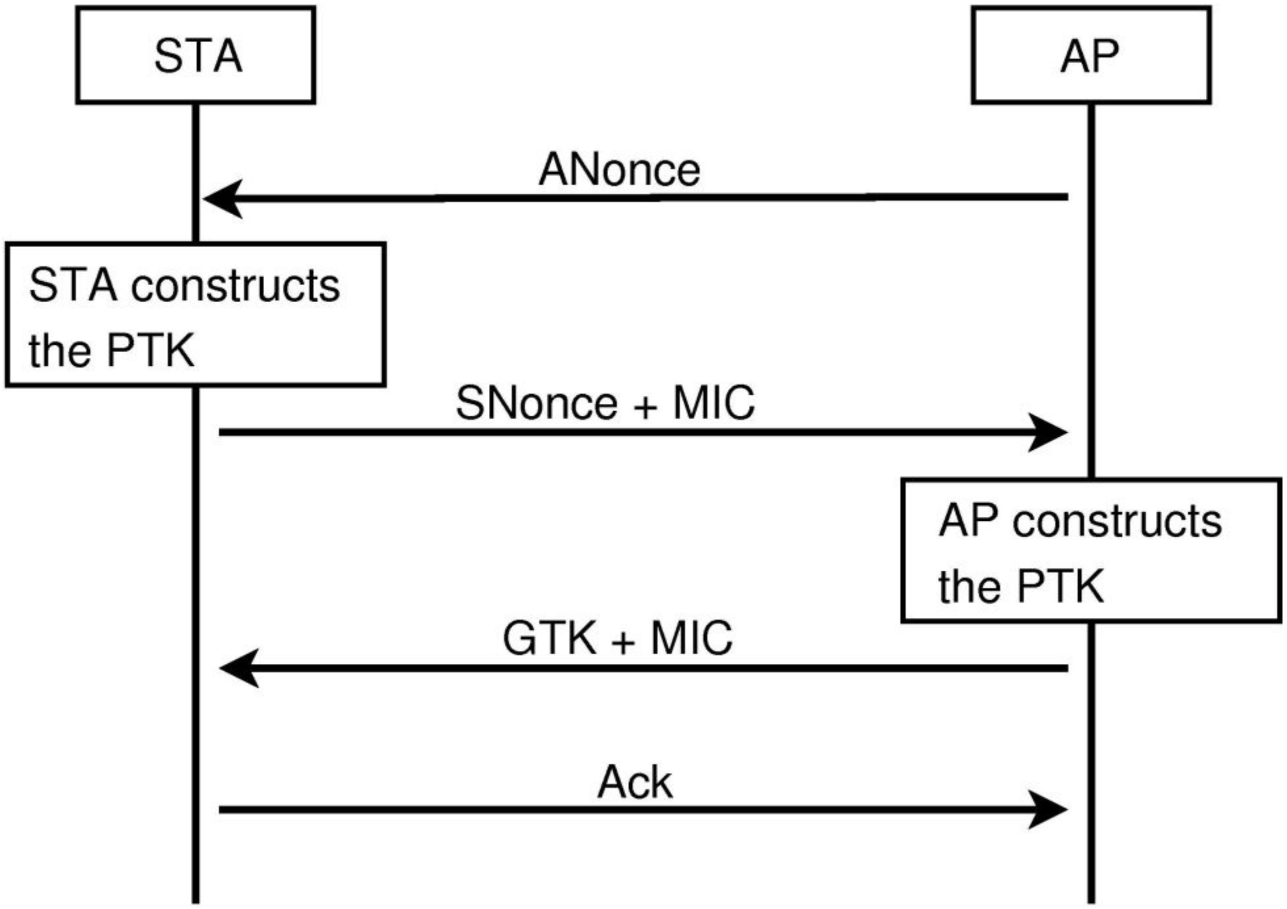
The master session is the first key which is generated either from 802.1X/EAP or derived from PSK authentication.

We discussed above keys from bottom to top and how keys are dependent on other keys. This is the view from top to bottom.



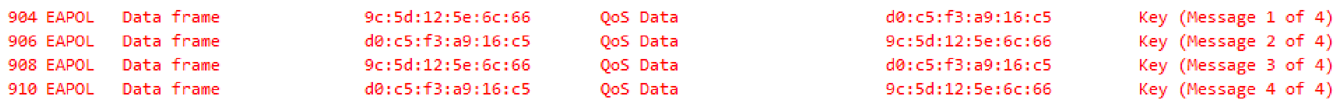
1. The first level key is generated is MSK during the process of 802.1X/EAP or PSK authentication.
2. The second level key is generated from MSK is PMK and GMK. PMK is used to generate PTK and GMK is used to create GTK.
3. Third level keys are the actual keys used for data encryption.

We can visually represent this handshake below (here, STA is the client and AP is the access point):

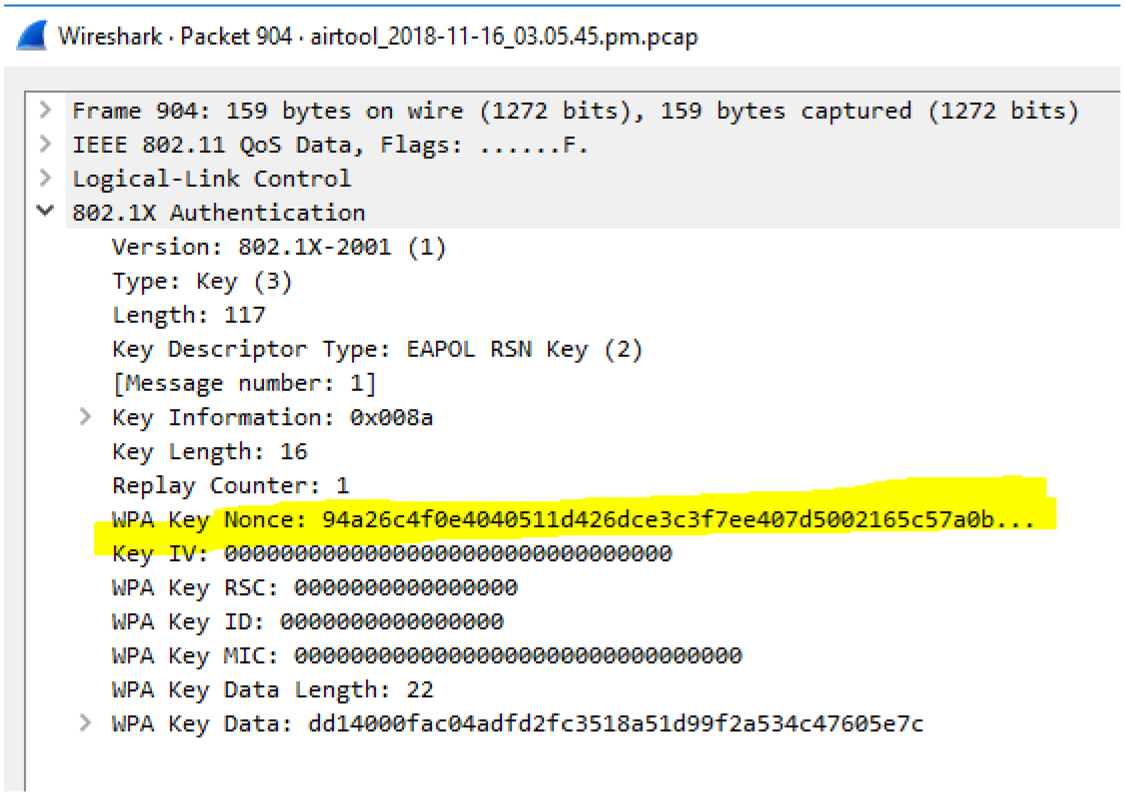


Device States:

A device going through states from authentication to association. Once the device is authenticated and associated and now security will be checked, and 4-way handshake will start.


```
904 EAPOL   Data frame              9c:5d:12:5e:6c:66       QoS Data                d0:c5:f3:a9:16:c5       Key (Message 1 of 4)
906 EAPOL   Data frame              d0:c5:f3:a9:16:c5       QoS Data                9c:5d:12:5e:6c:66       Key (Message 2 of 4)
908 EAPOL   Data frame              9c:5d:12:5e:6c:66       QoS Data                d0:c5:f3:a9:16:c5       Key (Message 3 of 4)
910 EAPOL   Data frame              d0:c5:f3:a9:16:c5       QoS Data                9c:5d:12:5e:6c:66       Key (Message 4 of 4)
```

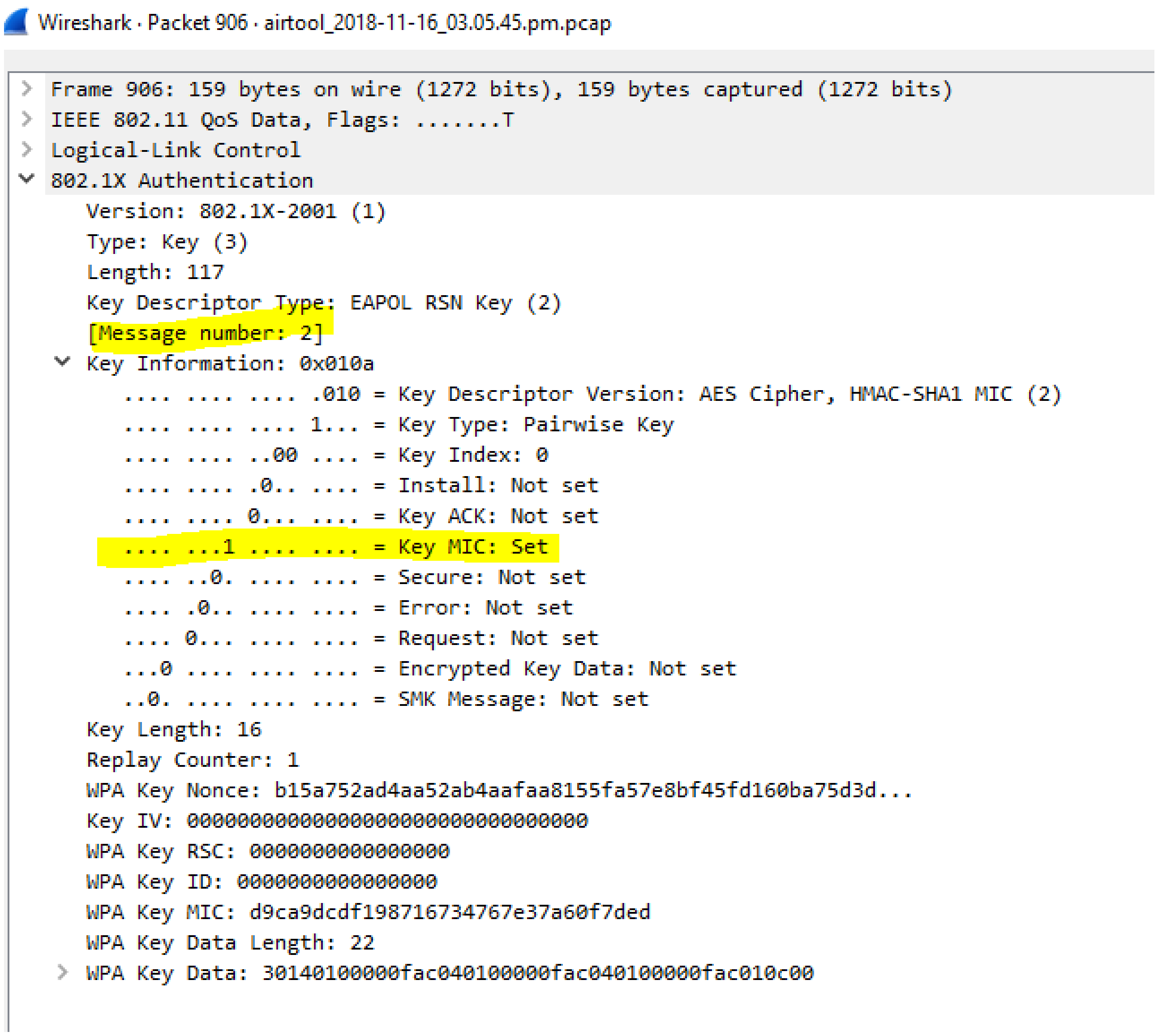
Message 1: AP sends to the client his ANONCE. Now the client has everything he needs to create the PTK because he got the ANONCE, it was the only thing that was missing for him.



Wireshark · Packet 904 · airtool_2018-11-16_03.05.45.pm.pcap

```
> Frame 904: 159 bytes on wire (1272 bits), 159 bytes captured (1272 bits)
> IEEE 802.11 QoS Data, Flags: ......F.
> Logical-Link Control
v 802.1X Authentication
     Version: 802.1X-2001 (1)
     Type: Key (3)
     Length: 117
     Key Descriptor Type: EAPOL RSN Key (2)
     [Message number: 1]
   > Key Information: 0x008a
     Key Length: 16
     Replay Counter: 1
     WPA Key Nonce: 94a26c4f0e4040511d426dce3c3f7ee407d5002165c57a0b...
     Key IV: 00000000000000000000000000000000
     WPA Key RSC: 0000000000000000
     WPA Key ID: 0000000000000000
     WPA Key MIC: 00000000000000000000000000000000
     WPA Key Data Length: 22
   > WPA Key Data: dd14000fac04adfd2fc3518a51d99f2a534c47605e7c
```

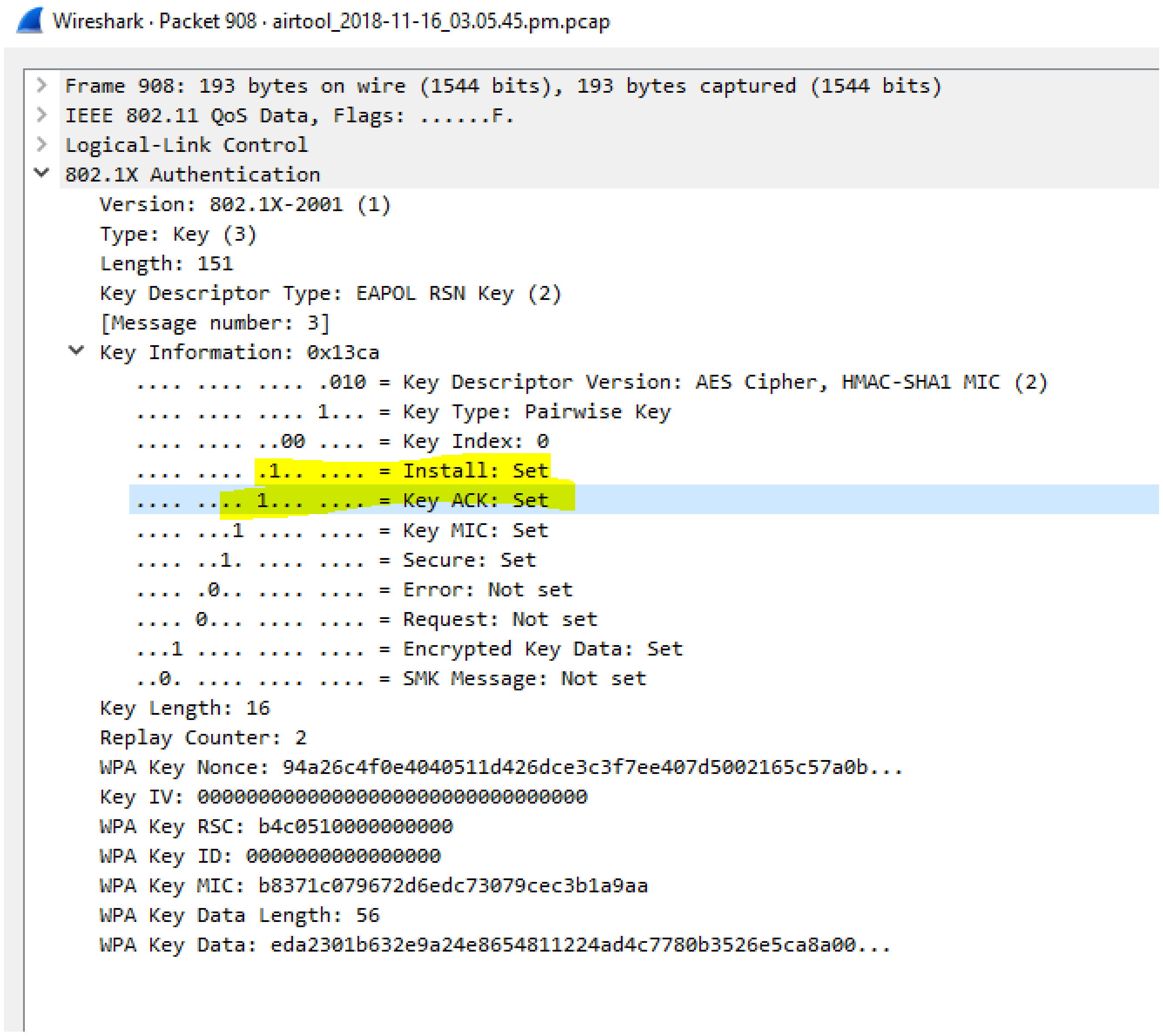
Message 2: The client sends to the AP his SNONCE with a MIC, the MIC is mainly for the AP to recognize that this message is really from this client, its like a signature (a high level algorithm signature). Now, after the AP got the message he has everything he needs to create the PTK and that is what he does.



Wireshark · Packet 906 · airtool_2018-11-16_03.05.45.pm.pcap

```
> Frame 906: 159 bytes on wire (1272 bits), 159 bytes captured (1272 bits)
> IEEE 802.11 QoS Data, Flags: .......T
> Logical-Link Control
v 802.1X Authentication
     Version: 802.1X-2001 (1)
     Type: Key (3)
     Length: 117
     Key Descriptor Type: EAPOL RSN Key (2)
     [Message number: 2]
   v Key Information: 0x010a
        .... .... .... .010 = Key Descriptor Version: AES Cipher, HMAC-SHA1 MIC (2)
        .... .... .... 1... = Key Type: Pairwise Key
        .... .... ..00 .... = Key Index: 0
        .... .... .0.. .... = Install: Not set
        .... .... 0... .... = Key ACK: Not set
        .... ...1 .... .... = Key MIC: Set
        .... ..0. .... .... = Secure: Not set
        .... .0.. .... .... = Error: Not set
        .... 0... .... .... = Request: Not set
        ...0 .... .... .... = Encrypted Key Data: Not set
        ..0. .... .... .... = SMK Message: Not set
     Key Length: 16
     Replay Counter: 1
     WPA Key Nonce: b15a752ad4aa52ab4aafaa8155fa57e8bf45fd160ba75d3d...
     Key IV: 00000000000000000000000000000000
     WPA Key RSC: 0000000000000000
     WPA Key ID: 0000000000000000
     WPA Key MIC: d9ca9dcdf198716734767e37a60f7ded
     WPA Key Data Length: 22
   > WPA Key Data: 30140100000fac040100000fac040100000fac010c00
```

Message 3: The AP sends to the client the GTK because he is going to be his new client. The client get the GTK and install it.



Wireshark · Packet 908 · airtool_2018-11-16_03.05.45.pm.pcap

```
> Frame 908: 193 bytes on wire (1544 bits), 193 bytes captured (1544 bits)
> IEEE 802.11 QoS Data, Flags: ......F.
> Logical-Link Control
v 802.1X Authentication
     Version: 802.1X-2001 (1)
     Type: Key (3)
     Length: 151
     Key Descriptor Type: EAPOL RSN Key (2)
     [Message number: 3]
   v Key Information: 0x13ca
        .... .... .... .010 = Key Descriptor Version: AES Cipher, HMAC-SHA1 MIC (2)
        .... .... .... 1... = Key Type: Pairwise Key
        .... .... ..00 .... = Key Index: 0
        .... .... .1.. .... = Install: Set
        .... .... 1... .... = Key ACK: Set
        .... ...1 .... .... = Key MIC: Set
        .... ..1. .... .... = Secure: Set
        .... .0.. .... .... = Error: Not set
        .... 0... .... .... = Request: Not set
        ...1 .... .... .... = Encrypted Key Data: Set
        ..0. .... .... .... = SMK Message: Not set
     Key Length: 16
     Replay Counter: 2
     WPA Key Nonce: 94a26c4f0e4040511d426dce3c3f7ee407d5002165c57a0b...
     Key IV: 00000000000000000000000000000000
     WPA Key RSC: b4c0510000000000
     WPA Key ID: 0000000000000000
     WPA Key MIC: b8371c079672d6edc73079cec3b1a9aa
     WPA Key Data Length: 56
     WPA Key Data: eda2301b632e9a24e8654811224ad4c7780b3526e5ca8a00...
```

Message 4: The client sends to the AP that everything is OK and installed.

```
EAPOL   Data frame          d0:c5:f3:a9:16:c5     QoS Data          9c:5d:12:5e:6c:66       Key (Message 4 of 4)
```

Why it is important to understand the stages in the messages?

If we want to crack Wi-Fi encryption and we don't have the passphrase (PSK), we need to capture the handshake between the client and the AP that we want to connect to. We can force the client to reconnect by simply attacking it with deauthentication messages, and then capture the handshake.

Using a long list of potential passwords, you can use aircrack-ng to run a dictionary attack against your handshake messages. Aircracki-ng takes the handshake that you capture (specifically the first two messages) and separate the MIC from the other parameters that all together is equal to the PTK `(PTK = PMK + ANONCE + SNONCE + MAC(AA) + MAC(SA))`. Then aircrack-ng runs the list with your potential passwords (your PSK (pre-shared key)) and puts it together with the other parameters, and then checks if the MIC that the command gets from the combination is equal to the original MIC. As you'll see ~ this process can take a lot of time depending on the complexity of the password.

Part 2: Analyzing a Wireless Capture File

Ok, now that we have some context for what we're about to do, let's jump right in.

Step 2.1 Download this .cap file

Step 2.2 Open the .cap file with Wireshark.

> ⚠️ Because this file is a .cap and not .pcap file, you might need to open it *from* Wireshark instead of double-clicking on the file to auto-launch Wireshark.

Step 2.3 Open the .cap file with Wireshark.

Step 2.4 Use the filter string `eapol` to filter for EAPoL traffic.

⁉️ Question 1 - How many sets of 4-way handshake exchanges are there (4 messages in one set)

⁉️ Question 2 - Submit a screenshot of your filtered results.

⁉️ Question 3 - What protocol is listed under the protocol column?

Step 2.5 Open the first packet by double-clicking on it.

Step 2.6 Expand the `802.1x Authentication` field in Wireshark.

⁉️ Question 4 - What encryption key type is being used for this key exchange?

⁉️ Question 5 - Based on that encryption key type, what possible encryption standards are being used for this communication?

⁉️ Question 6 - Look through the first set of key exchanges under the `802.1x Authentication > Key Information` field. What changes do you notice during the handshake.

Part 3: Key Cracking with aircrack-ng

Because I can't assume that everyone in our class has a wireless network card that supports monitor mode ~ we won't be able to generate and capture our own WEP traffic and then solve for the initialization vector (IV) and private key to gain unauthorized access to the network. So instead, we'll take the capture pact from Part 2 of this lab and launch a dictionary attack against it to try and brute force a private key.

Step 3.1 Launch and long into your Kali machine you created in our previous lab.

Step 3.2 Drag and drop the net.cap file you downloaded in Part 2 of this lab into your Kali machine.

> This operation might only be possible if you have VMWare tools installed for this machine (most of the time, this is done by default). If you can't drag and drop, access this lab from the Kali environment and download the .cap file.

Step 3.3 Open a new terminal and run the below command to download our dictionary file we'll use for our attack:

```
wget --no-check-certificate --content-disposition
https://github.com/mikeconti/csf432-
fall2020/blob/master/labs/lab11/files/pass-dictionary.txt.gz
```

You can download this file here from your browser.

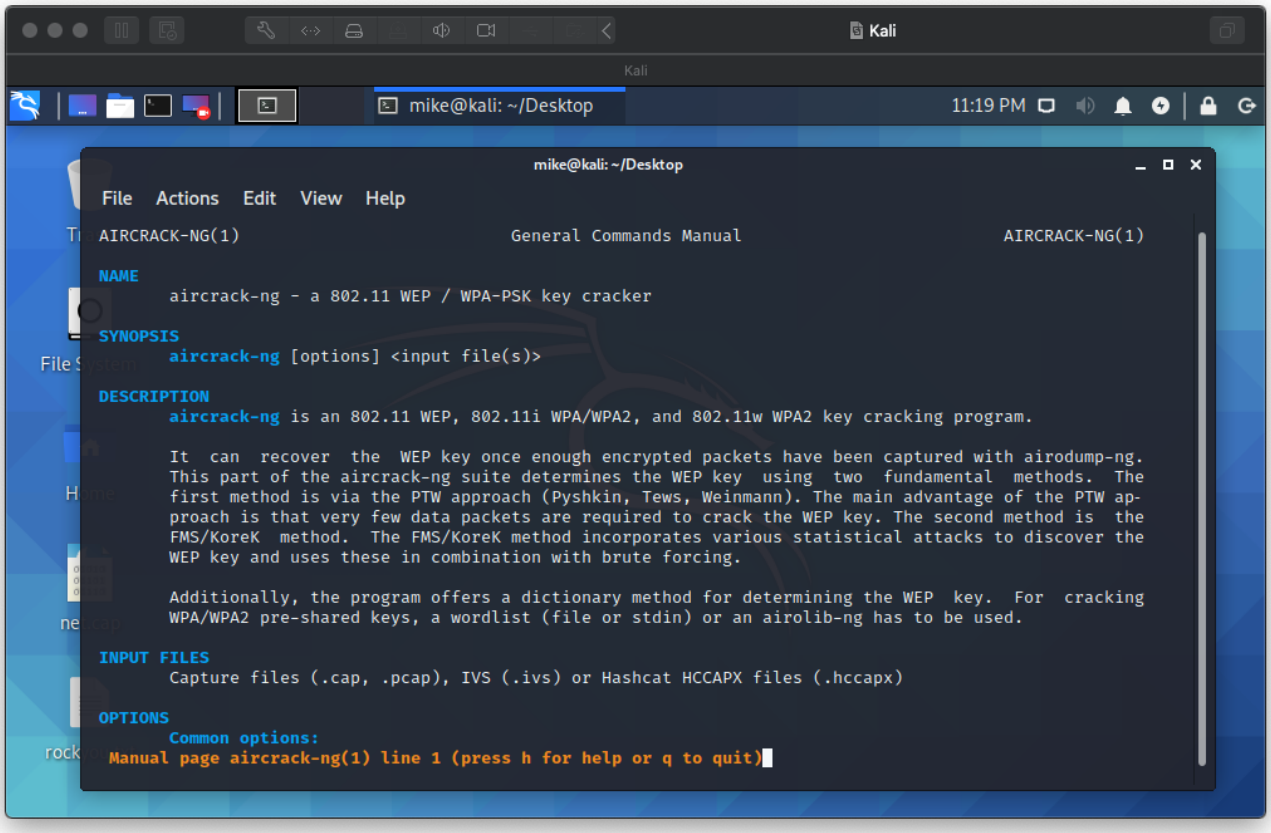
Step 3.4 Unpack/decompress this file in Kali. Make sure the plaintext .txt file is located on your desktop.

> A dictionary attack is based on trying all the strings in a pre-arranged listing. Such attacks originally used words one would find in a dictionary

Aircrack-ng is a network software suite consisting of a detector, packet sniffer, WEP and WPA/WPA2-PSK cracker and analysis tool for 802.11 wireless LANs. Kali Linux has aircrack-ng installed by default ~ so no need to go through a painfull terminal download and install process here.

You can learn more about the options you have available to you by running the manual command below:

```
mike@kali:~/Desktop$ man aircrack-ng to see the manual page of the tool.
```



```
AIRCRACK-NG(1)                 General Commands Manual                 AIRCRACK-NG(1)

NAME
       aircrack-ng - a 802.11 WEP / WPA-PSK key cracker

SYNOPSIS
       aircrack-ng [options] <input file(s)>

DESCRIPTION
       aircrack-ng is an 802.11 WEP, 802.11i WPA/WPA2, and 802.11w WPA2 key cracking program.

       It  can  recover  the  WEP key once enough encrypted packets have been captured with airodump-ng.
       This part of the aircrack-ng suite determines the WEP key  using  two  fundamental  methods.  The
       first method is via the PTW approach (Pyshkin, Tews, Weinmann). The main advantage of the PTW ap-
       proach is that very few data packets are required to crack the WEP key. The second method is  the
       FMS/KoreK  method.  The FMS/KoreK method incorporates various statistical attacks to discover the
       WEP key and uses these in combination with brute forcing.

       Additionally, the program offers a dictionary method for determining the WEP  key.  For  cracking
       WPA/WPA2 pre-shared keys, a wordlist (file or stdin) or an airolib-ng has to be used.

INPUT FILES
       Capture files (.cap, .pcap), IVS (.ivs) or Hashcat HCCAPX files (.hccapx)

OPTIONS
       Common options:
 Manual page aircrack-ng(1) line 1 (press h for help or q to quit)
```

Step 3.5 Navigate to your Kali desktop in your terminal.

Step 3.6 Run the following command to crack the private key used for this network encryption.

```
mike@kali:~/Desktop$ aircrack-ng net.cap -w ./pass-dictionary.txt
```

Here the -w option specifies the path to our dictionary.

> Make sure that the pwd your are operating out of is the same location as your pass-dictionary.txt file or change the pathway to correctly target your pass-dictionary.txt file. If you get a "Please specify a dictionary (option -w)" error, this is likely the problem.

⁉️ Question 7 - Submit a screenshot of your terminal running the aircrack-ng program.

Step 3.7 Terminate the program with `ctrl+c`. No need to run the entire program.

⁉️ Question 8 - What are the three fields on the left hand side of the aircrack-ng display. Match each one of these fields with the key abbreviations described in Part 1 of this lab.

⁉️ Question 9 - In your own words, what is aircrack-ng doing and what is it solving for?

Because the capture file we are testing against is using WPA encryption standards, we're going to have a harder time getting the key we need to decode this network traffic. Keep in mind that if this capture file was using WEP, we could duplicate this exact process and easily gain unauthorized access to this network. While it might seem like no one in their right mind would be using WEP, there are still many networks that do use it as a result of an oversight or deprecated hardware.

It's trivial to scan for WEP networks while driving down the highway or around popular shopping districts. This is the exact scenario that resulted in the loss of 45 million TK Maxx customer records ~ source - file.

Part 4: Submission

Convert your network document into a **.PDF** and upload a single `lastname_lab11.pdf` file to Brightspace through the attachment uploads option. source - file