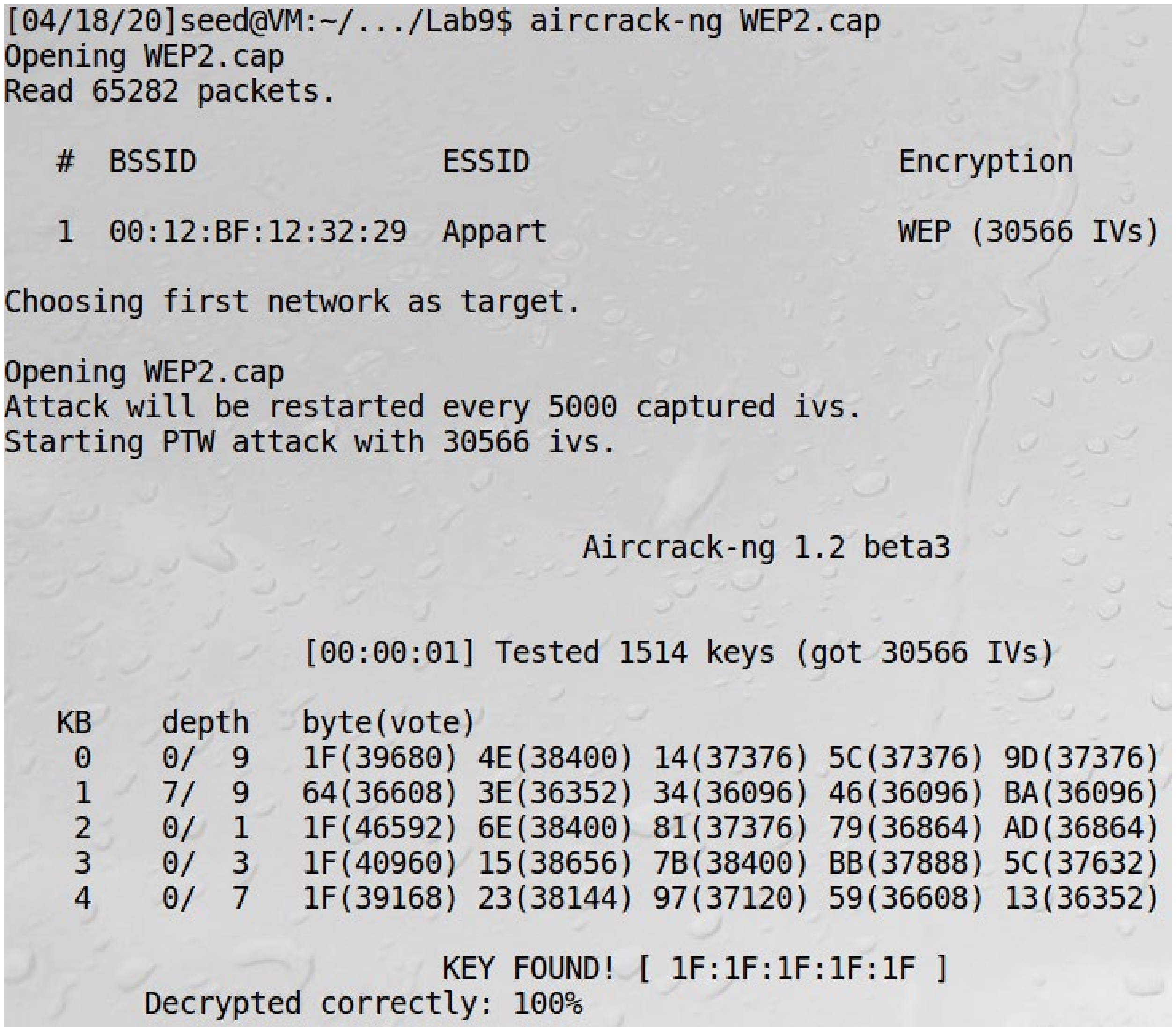
Task 1: Cracking the WEP Password

To use Aircrack-ng to crack the WEP protocol using the given PCAP file, I ran the command `aircrack-ng WEP2.cap` as shown in Figure 1 to obtain the correct password, `1F:1F:1F:1F:1F`.

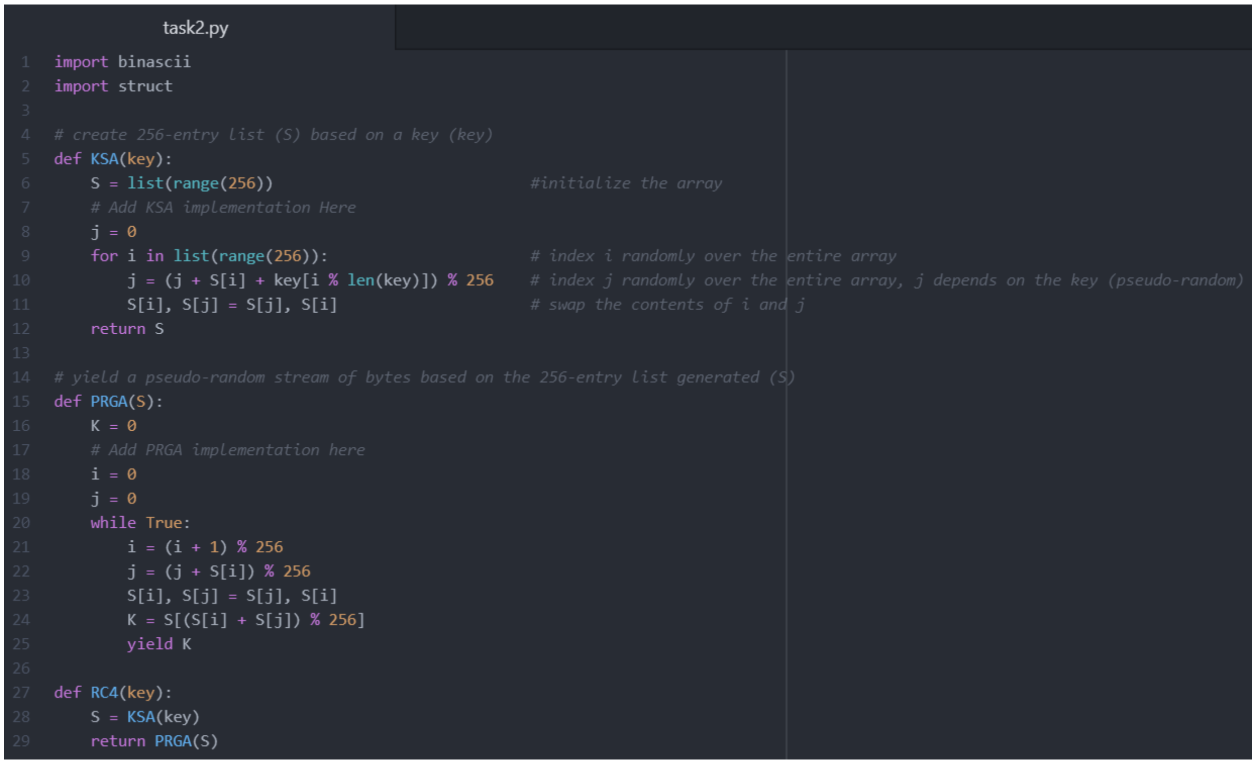


```
[04/18/20]seed@VM:~/.../Lab9$ aircrack-ng WEP2.cap
Opening WEP2.cap
Read 65282 packets.

   #  BSSID              ESSID                     Encryption

   1  00:12:BF:12:32:29  Appart                    WEP (30566 IVs)

Choosing first network as target.

Opening WEP2.cap
Attack will be restarted every 5000 captured ivs.
Starting PTW attack with 30566 ivs.


                                 Aircrack-ng 1.2 beta3


                 [00:00:01] Tested 1514 keys (got 30566 IVs)

   KB    depth   byte(vote)
    0    0/  9   1F(39680) 4E(38400) 14(37376) 5C(37376) 9D(37376)
    1    7/  9   64(36608) 3E(36352) 34(36096) 46(36096) BA(36096)
    2    0/  1   1F(46592) 6E(38400) 81(37376) 79(36864) AD(36864)
    3    0/  3   1F(40960) 15(38656) 7B(38400) BB(37888) 5C(37632)
    4    0/  7   1F(39168) 23(38144) 97(37120) 59(36608) 13(36352)

                         KEY FOUND! [ 1F:1F:1F:1F:1F ]
        Decrypted correctly: 100%
```

Figure 1: Cracking WEP Protocol with Aircrack-ng

Task 2: Cracking the WEP Packet

The RC4 algorithm is implemented in my code, `task2.py` as shown in Figure 2.

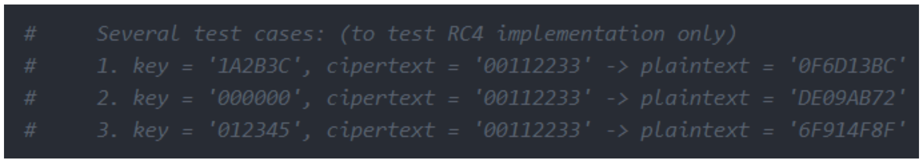


task2.py

```python
import binascii
import struct

# create 256-entry list (S) based on a key (key)
def KSA(key):
    S = list(range(256))                        #initialize the array
    # Add KSA implementation Here
    j = 0
    for i in list(range(256)):                  # index i randomly over the entire array
        j = (j + S[i] + key[i % len(key)]) % 256    # index j randomly over the entire array, j depends on the key (pseudo-random)
        S[i], S[j] = S[j], S[i]                 # swap the contents of i and j
    return S

# yield a pseudo-random stream of bytes based on the 256-entry list generated (S)
def PRGA(S):
    K = 0
    # Add PRGA implementation here
    i = 0
    j = 0
    while True:
        i = (i + 1) % 256
        j = (j + S[i]) % 256
        S[i], S[j] = S[j], S[i]
        K = S[(S[i] + S[j]) % 256]
        yield K

def RC4(key):
    S = KSA(key)
    return PRGA(S)
```

Figure 2: RC4 Algorithm Python Code

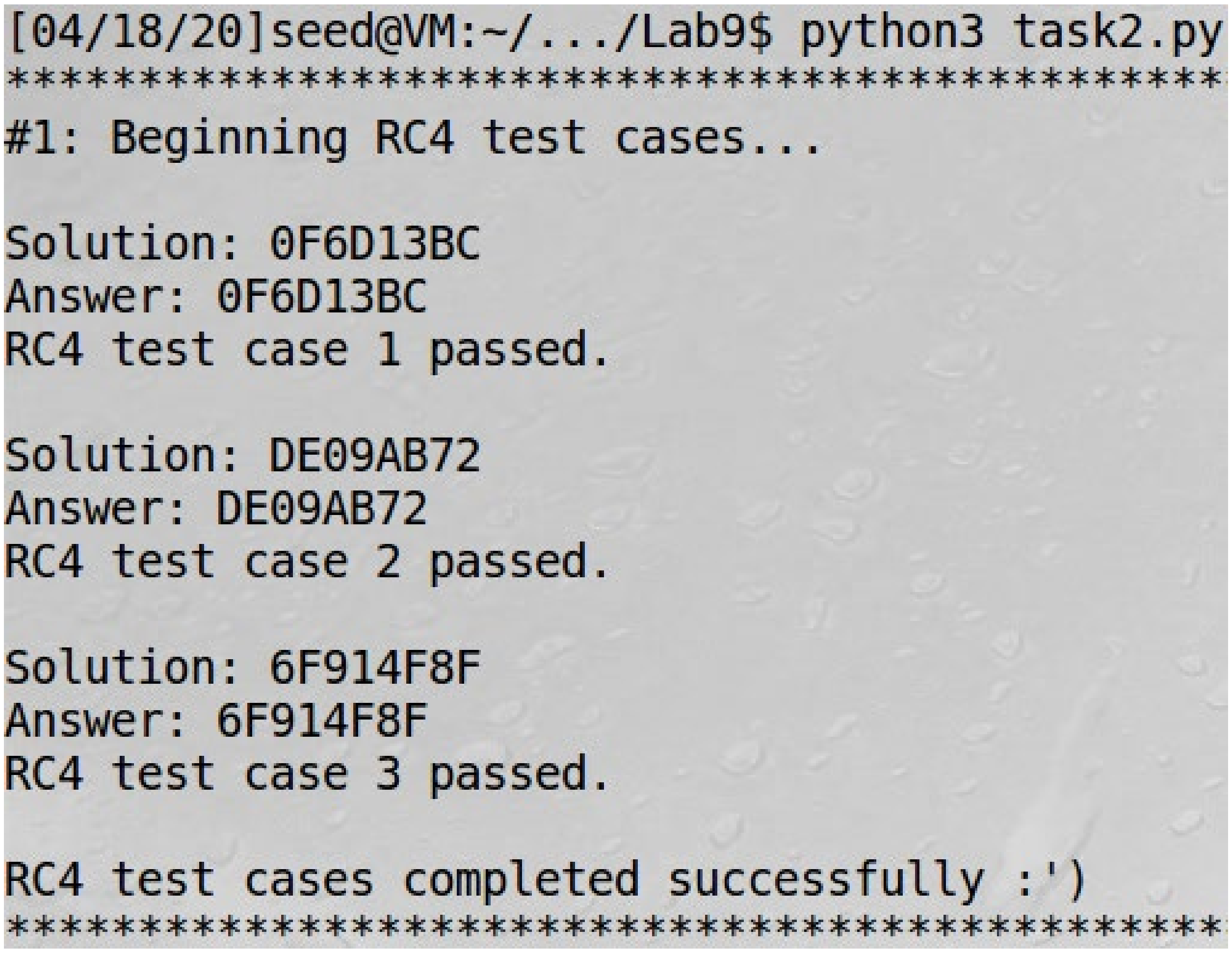
There were three test cases provided for testing to verify that my RC4 implementation was correct as shown in Figure 3.



```
#     Several test cases: (to test RC4 implementation only)
#     1. key = '1A2B3C', cipertext = '00112233' -> plaintext = '0F6D13BC'
#     2. key = '000000', cipertext = '00112233' -> plaintext = 'DE09AB72'
#     3. key = '012345', cipertext = '00112233' -> plaintext = '6F914F8F'
```

Figure 3: RC4 Algorithm given test cases

These three test cases were run and completed successfully as shown in Figure 4. The full code for the test cases is in the `task2.py` file.

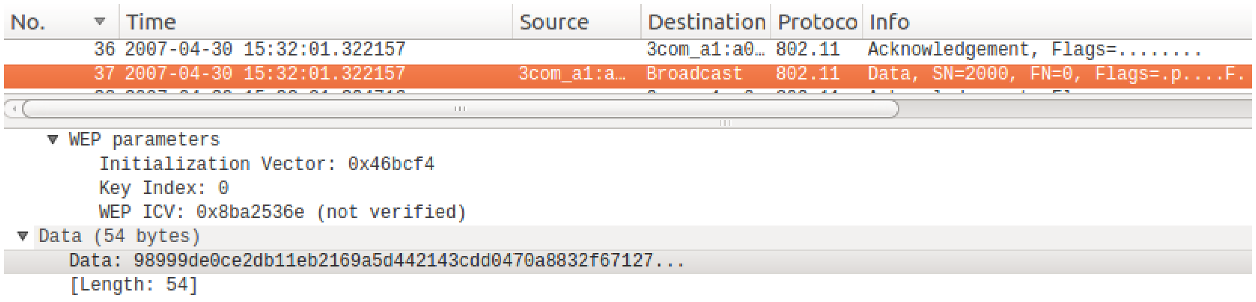


```
[04/18/20]seed@VM:~/.../Lab9$ python3 task2.py
***********************************************
#1: Beginning RC4 test cases...

Solution: 0F6D13BC
Answer: 0F6D13BC
RC4 test case 1 passed.

Solution: DE09AB72
Answer: DE09AB72
RC4 test case 2 passed.

Solution: 6F914F8F
Answer: 6F914F8F
RC4 test case 3 passed.

RC4 test cases completed successfully :')
***********************************************
```

Figure 4: RC4 Test Cases Success

I chose to crack broadcast WEP packet where SN=2000. Key details of the packet from the PCAP can be seen in Figure 5, from which I extracted the following:

- Initialization Vector (IV): `0x46bcf4`
- Encrypted WEP ICV: `0x8ba2536e`
- Encrypted Data: `98999de0ce2db11eb2169a5d442143cdd0470a8832f6712745fb4ffacdcc9ff99681c1da2f8c479ef446300eaa68aaca018b6a0a985c`



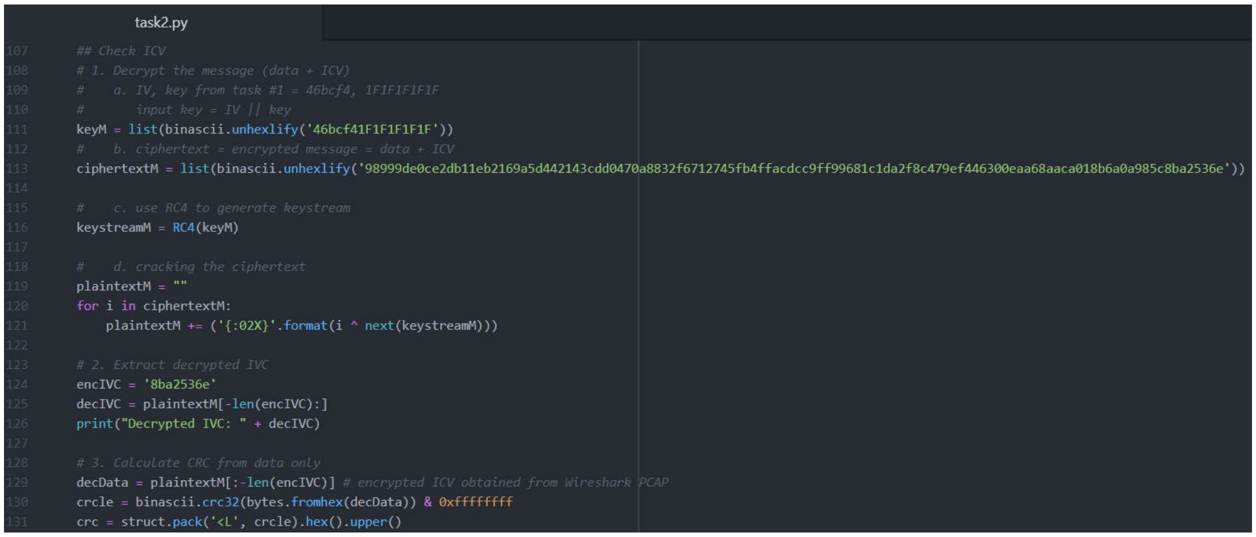
```
No.     Time                       Source      Destination  Protoco  Info
     36 2007-04-30 15:32:01.322157              3com_a1:a0…  802.11   Acknowledgement, Flags=........
     37 2007-04-30 15:32:01.322157  3com_a1:a…  Broadcast    802.11   Data, SN=2000, FN=0, Flags=.p....F.

  ▼ WEP parameters
      Initialization Vector: 0x46bcf4
      Key Index: 0
      WEP ICV: 0x8ba2536e (not verified)
▼ Data (54 bytes)
    Data: 98999de0ce2db11eb2169a5d442143cdd0470a8832f67127...
    [Length: 54]
```

Figure 5: PCAP of WEP Broadcast Packets

ICV can be used to verify the integrity of the packet. The captured decrypted ICV will only match the calculated CRC of the message if the decryption is correctly done. To verify my decryption:

1. Decrypt data and IVC: The data shown in the PCAP in Figure 5 is that of the encrypted data and IVC, thus we begin by decrypting the encrypted IVC concatenated with the encrypted data to obtain the decrypted data and IVC.
2. Extract decrypted IVC: The decrypted IVC can be extracted from the end of the decrypted message and is the same length as the encrypted IVC shown in the Figure 5.
3. Calculate CRC of decrypted data: The CRC of the decrypted data can then be calculated.
4. Compare the values of step 2 and 3: If the decrypted IVC is the same as the CRC of the decrypted data, we can verify that the decryption is correctly done.

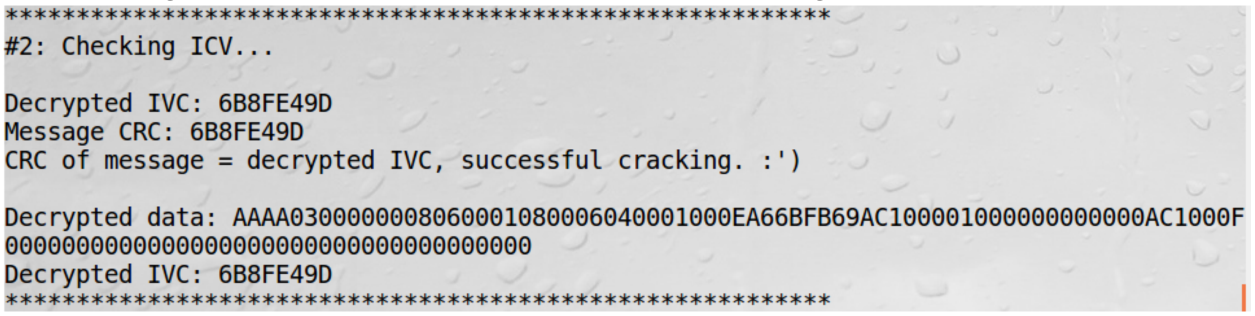
This was completed using Python as shown in the `task2.py` code in Figure 6.



```python
## Check ICV
# 1. Decrypt the message (data + ICV)
#    a. IV, key from task #1 = 46bcf4, 1F1F1F1F1F
#       input key = IV || key
keyM = list(binascii.unhexlify('46bcf41F1F1F1F1F'))
#    b. ciphertext = encrypted message = data + ICV
ciphertextM = list(binascii.unhexlify('98999de0ce2db11eb2169a5d442143cdd0470a8832f6712745fb4ffacdcc9ff99681c1da2f8c479ef446300eaa68aaca018b6a0a985c8ba2536e'))

#    c. use RC4 to generate keystream
keystreamM = RC4(keyM)

#    d. cracking the ciphertext
plaintextM = ""
for i in ciphertextM:
    plaintextM += ('{:02X}'.format(i ^ next(keystreamM)))

# 2. Extract decrypted IVC
encIVC = '8ba2536e'
decIVC = plaintextM[-len(encIVC):]
print("Decrypted IVC: " + decIVC)

# 3. Calculate CRC from data only
decData = plaintextM[:-len(encIVC)] # encrypted ICV obtained from Wireshark PCAP
crcle = binascii.crc32(bytes.fromhex(decData)) & 0xffffffff
crc = struct.pack('<L', crcle).hex().upper()
```

Figure 6: Decrypt WEP Broadcast Packet Python Code

After running `task2.py`, we obtain the results as shown in Figure 7.



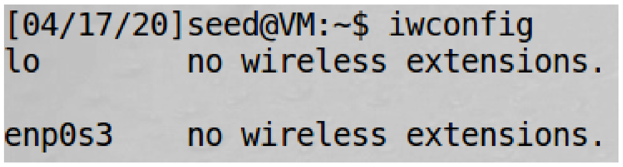
```
**********************************************************
#2: Checking ICV...

Decrypted IVC: 6B8FE49D
Message CRC: 6B8FE49D
CRC of message = decrypted IVC, successful cracking. :')

Decrypted data: AAAA0300000008060001080006040001000EA66BFB69AC100001000000000000AC1000F
0000000000000000000000000000000000000
Decrypted IVC: 6B8FE49D
**********************************************************
```

Figure 7: Successfully cracked WEP Packet

We are able to successfully crack the WEP packet and obtain the decrypted data, which is "`AAAA0300000008060001080006040001000EA66BFB69AC100001000000000000AC1000F0000000000000000000000000000000000000`", as well as the decrypted ICV, "`6B8FE49D`" as shown in Figure 7.

Task 3: Capturing the Four-Way Handshake

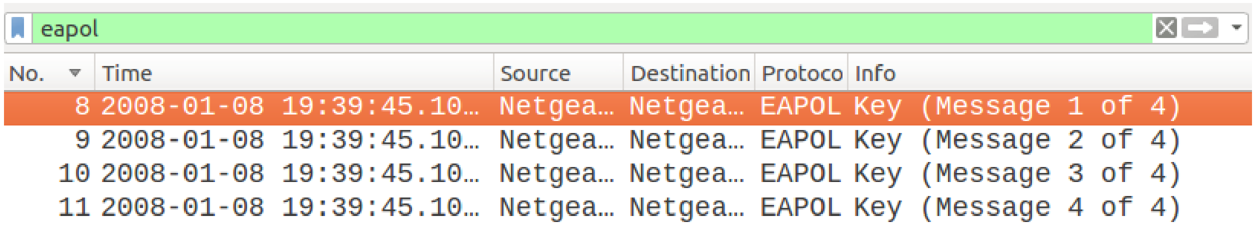
To list out all available wireless cards connected in the Ubuntu 16.04 SEED Virtual Machine, I ran the command, `iwconfig`. As shown in Figure 1, there were no wireless cards that I could enable Monitor Mode on.



```
[04/17/20]seed@VM:~$ iwconfig
lo        no wireless extensions.

enp0s3    no wireless extensions.
```

Figure 1: No wireless cards

As such, I used the given `WEP.cap` file to identify the four-way handshake as shown in Figure 2.

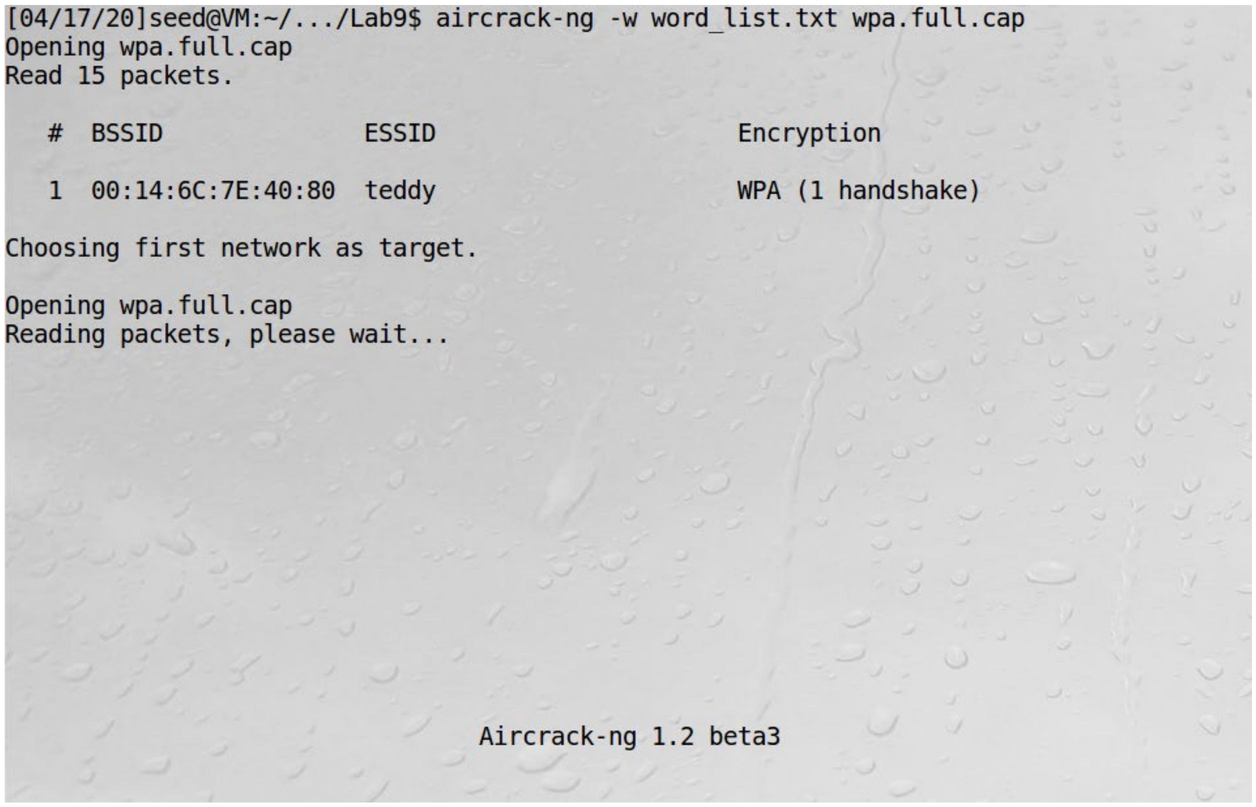


No.	Time	Source	Destination	Protoco	Info
8	2008-01-08 19:39:45.10…	Netgea…	Netgea…	EAPOL	Key (Message 1 of 4)
9	2008-01-08 19:39:45.10…	Netgea…	Netgea…	EAPOL	Key (Message 2 of 4)
10	2008-01-08 19:39:45.10…	Netgea…	Netgea…	EAPOL	Key (Message 3 of 4)
11	2008-01-08 19:39:45.10…	Netgea…	Netgea…	EAPOL	Key (Message 4 of 4)

Figure 2: PCAP of Four-Way Handshake

Task 4: Cracking WPA2 WiFi Passphrase Using Aircrack-ng

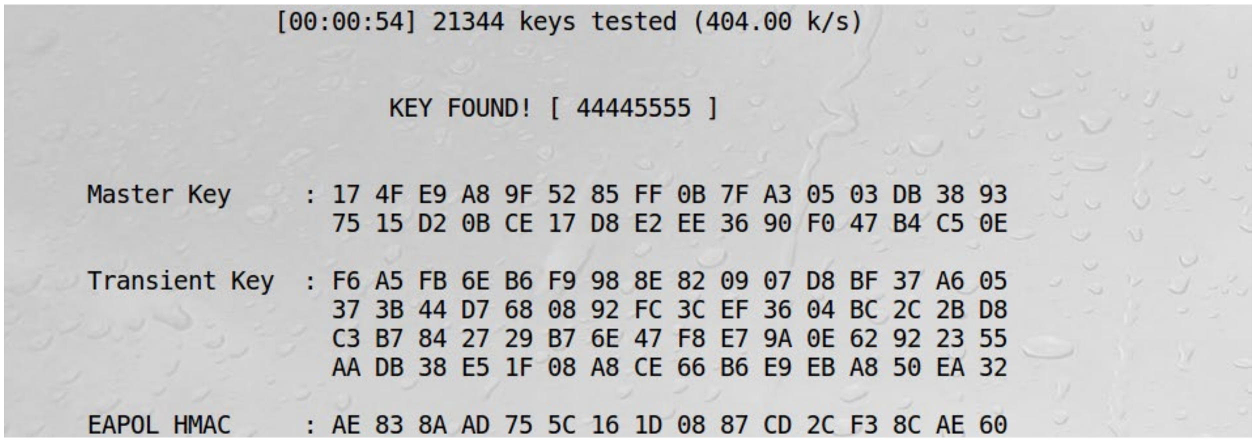
To use Aircrack-ng to crack the passphrase encrypted by the WPA2 with a given word list and given PCAP file, I ran the command `aircrack-ng -w word_list.txt wpa.full.cap` as shown in Figure 3.



```
[04/17/20]seed@VM:~/.../Lab9$ aircrack-ng -w word_list.txt wpa.full.cap
Opening wpa.full.cap
Read 15 packets.

   #  BSSID              ESSID                     Encryption

   1  00:14:6C:7E:40:80  teddy                     WPA (1 handshake)

Choosing first network as target.

Opening wpa.full.cap
Reading packets, please wait...



                              Aircrack-ng 1.2 beta3
```

Figure 3: Run Aircrack-ng

As shown in Figure 4, 21344 keys were tested before the key, `44445555` was found.



```
                [00:00:54] 21344 keys tested (404.00 k/s)


                       KEY FOUND! [ 44445555 ]


     Master Key     : 17 4F E9 A8 9F 52 85 FF 0B 7F A3 05 03 DB 38 93
                      75 15 D2 0B CE 17 D8 E2 EE 36 90 F0 47 B4 C5 0E

     Transient Key  : F6 A5 FB 6E B6 F9 98 8E 82 09 07 D8 BF 37 A6 05
                      37 3B 44 D7 68 08 92 FC 3C EF 36 04 BC 2C 2B D8
                      C3 B7 84 27 29 B7 6E 47 F8 E7 9A 0E 62 92 23 55
                      AA DB 38 E5 1F 08 A8 CE 66 B6 E9 EB A8 50 EA 32

     EAPOL HMAC     : AE 83 8A AD 75 5C 16 1D 08 87 CD 2C F3 8C AE 60
```

Figure 4: Completed WiFi cracking with Aircrack-ng

a. What is the difference between Monitor Mode and Promiscuous Mode?

Monitor Mode allows a wireless network interface card (NIC) to capture packets without associating with any access points. Monitor Mode only applies to wireless networks.

Promiscuous Mode allows you to sniff packets after associating with an access point. The user must be able to authenticate themselves at an access point to be associated to use Promiscuous Mode for packet sniffing. Promiscuous Mode can be used on both wired and wireless networks.

b. If the WiFi traffic is on-going, how to crack the WiFi password?

If the WiFi traffic is still live, a deauthentication attack can be executed to dissociate any wireless clients associated with a particular access point using aireplay-ng and the access point's MAC address. When the dissociated client attempts to reauthenticate with the network, we can capture the WPA/WPA2 handshakes. Suitable softwares to crack the WiFi password such as Aircrack-ng can then be used with a given word list and the PCAP file obtained to crack the WiFi password.