Cryptography Lab 1

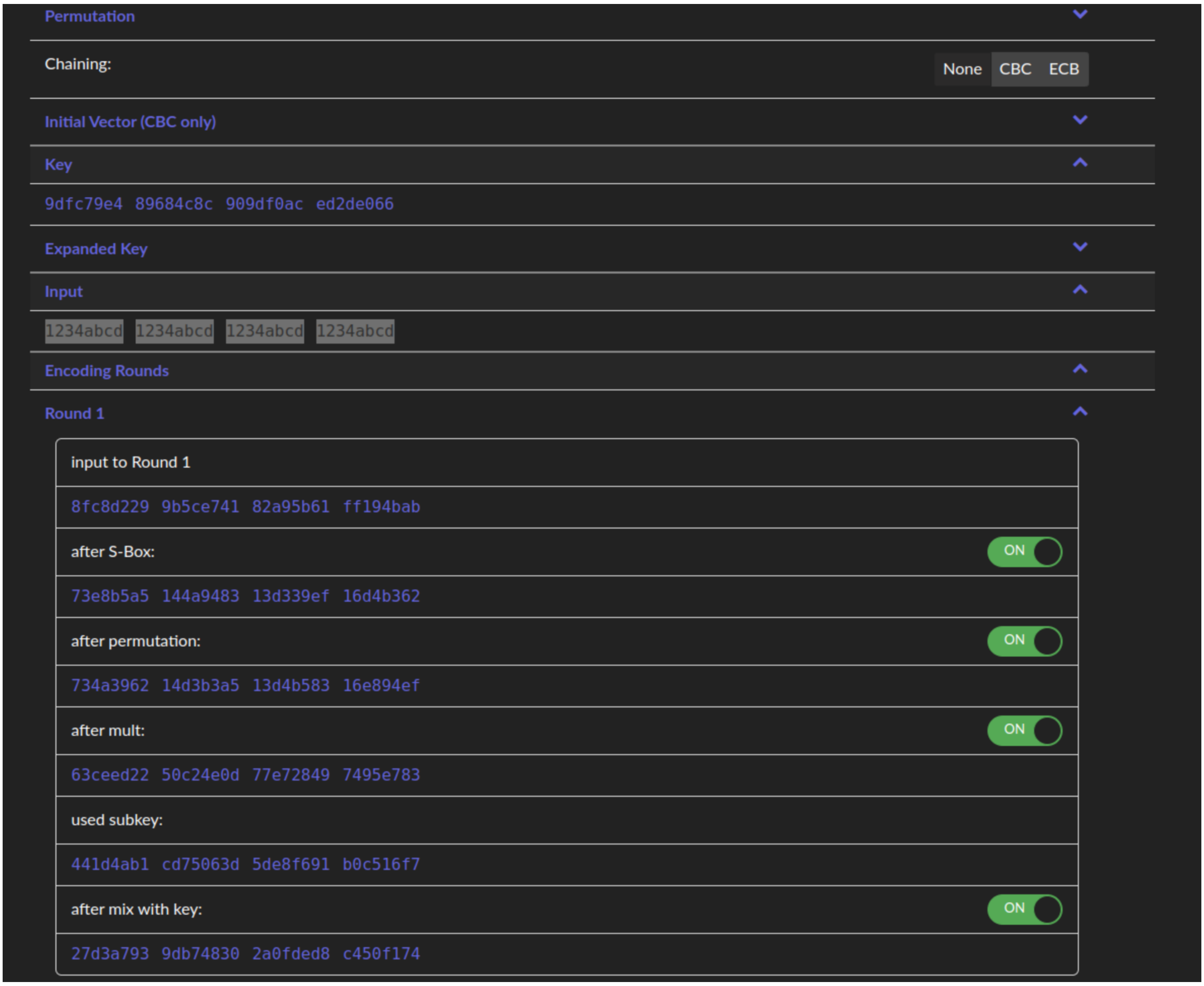
Name: Albert Ferguson **SID:** 13611165

Part 1. AES

1.1 AES Encryption

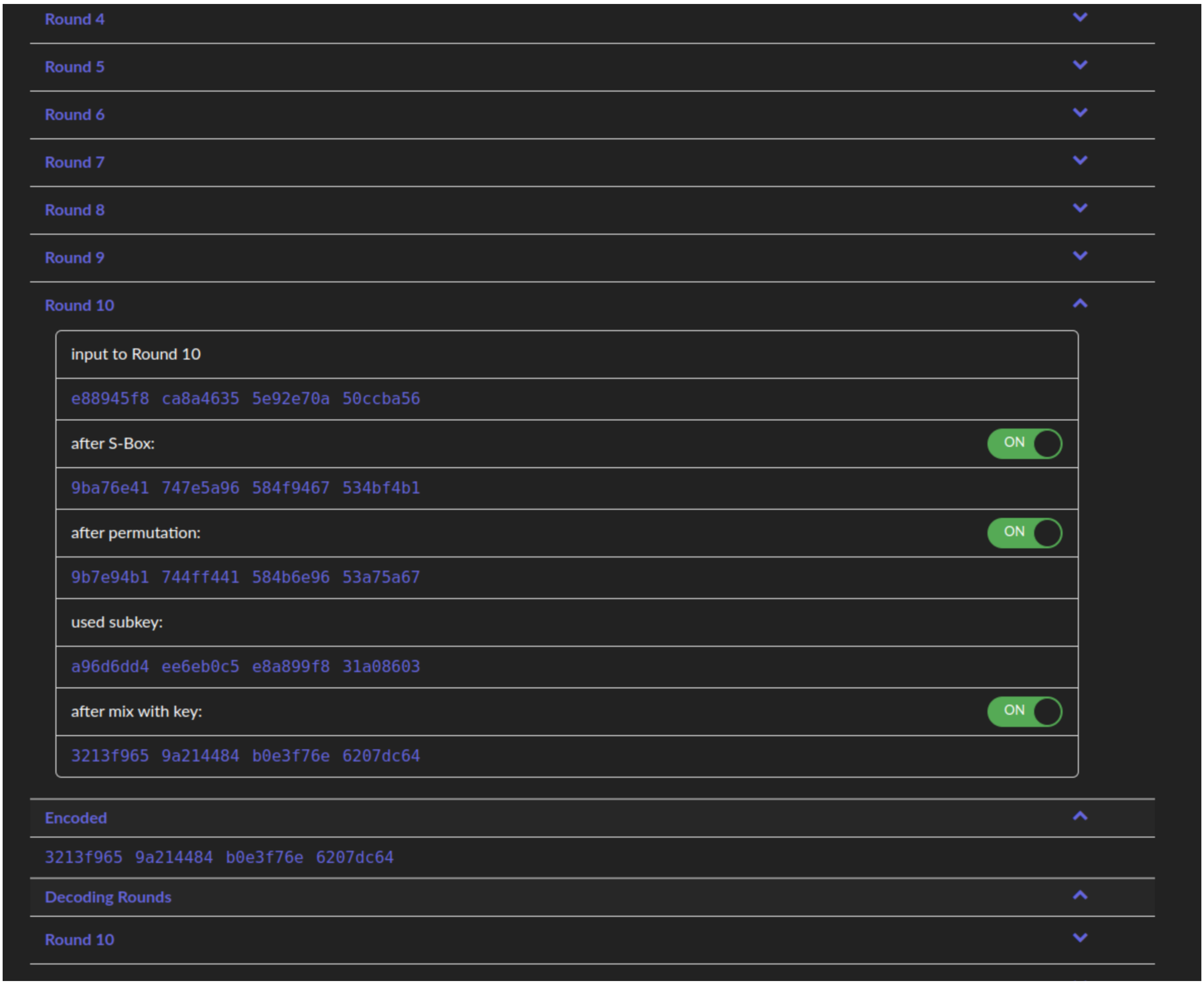
The following uses https://www.cryptool.org/en/cto/aes-step-by-step.

First off, generate a 32 hex-key, `9dfc79e4 89684c8c 909df0ac ed2de066`. Then some simple input, `1234abcd 1234abcd 1234abcd 1234abcd`. The resultant 1st round appears as,

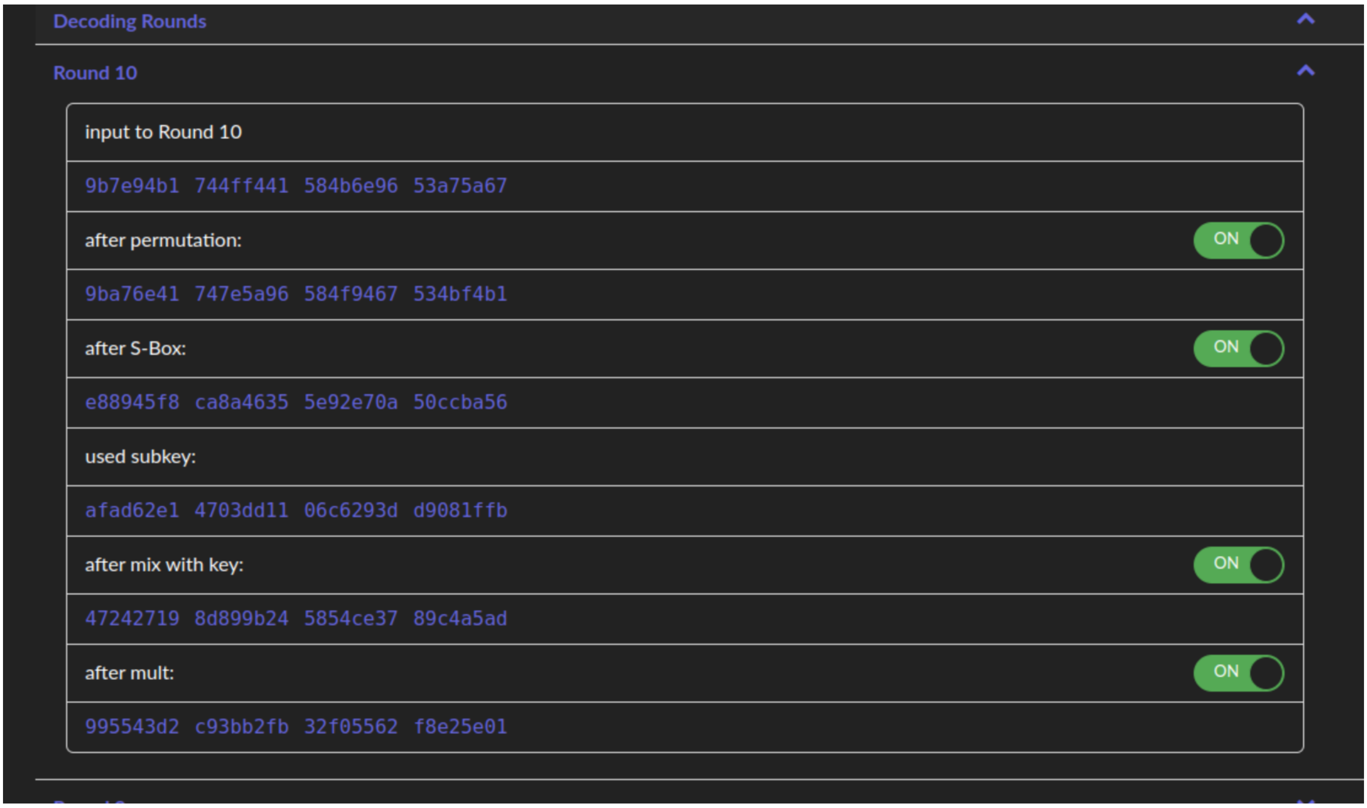


The difference between the first and last round, `r1` and `rn`, is that the 10th round skips the column mixing step. This is visible as the "*after mult*" step in the below screenshot,

> The column mixing step applies a matrix multiplication to the previous permutation step. Hence, the naming.



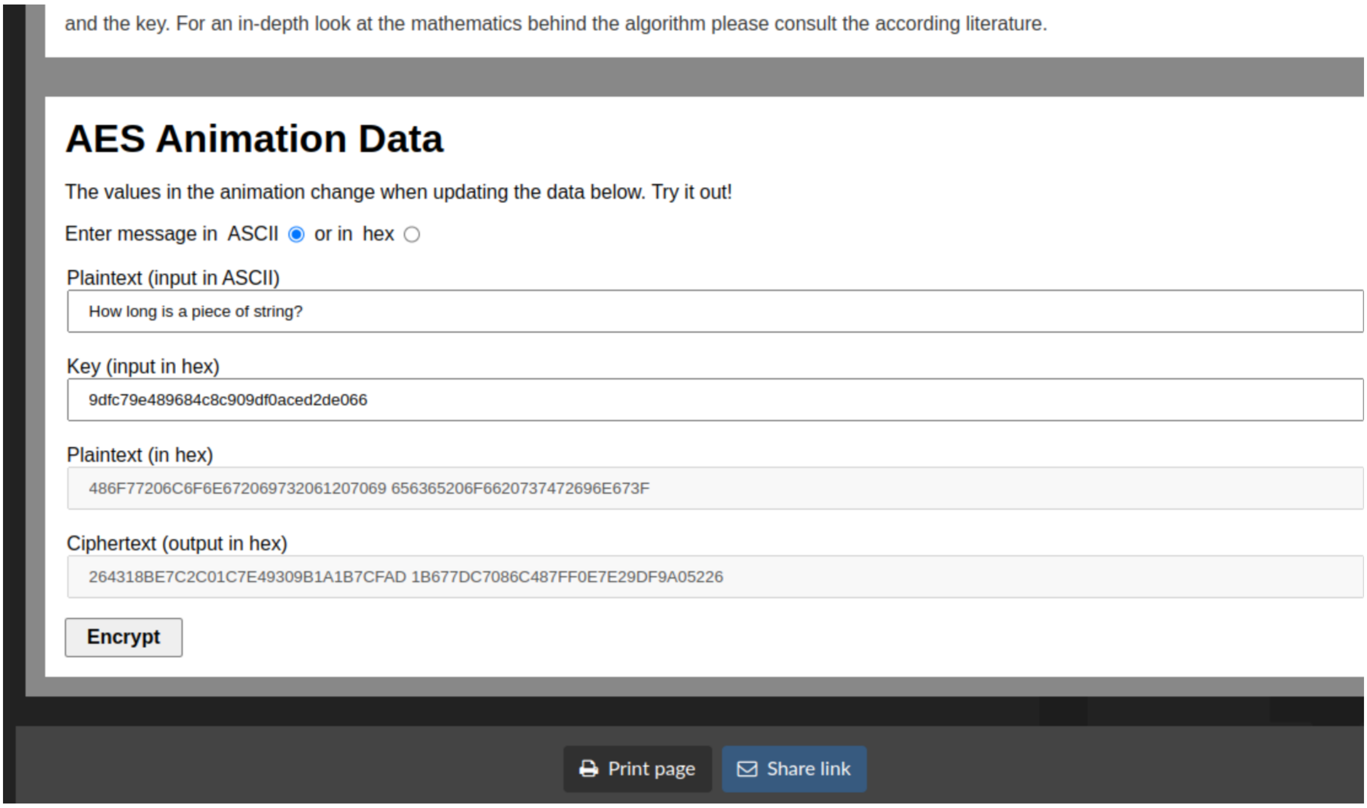
The resultant encoded output is, `3213f9659a214484b0e3f76e6207dc64`. The decoding uses the same atomic steps as the encode but in a different order. Where encode would take input > S-box > permutate > multiply > apply subkey. Then decode would take input > permutate > S-box > apply subkey > multiply. Likewise with encoding, the `nth` round does not execute the multiply step.



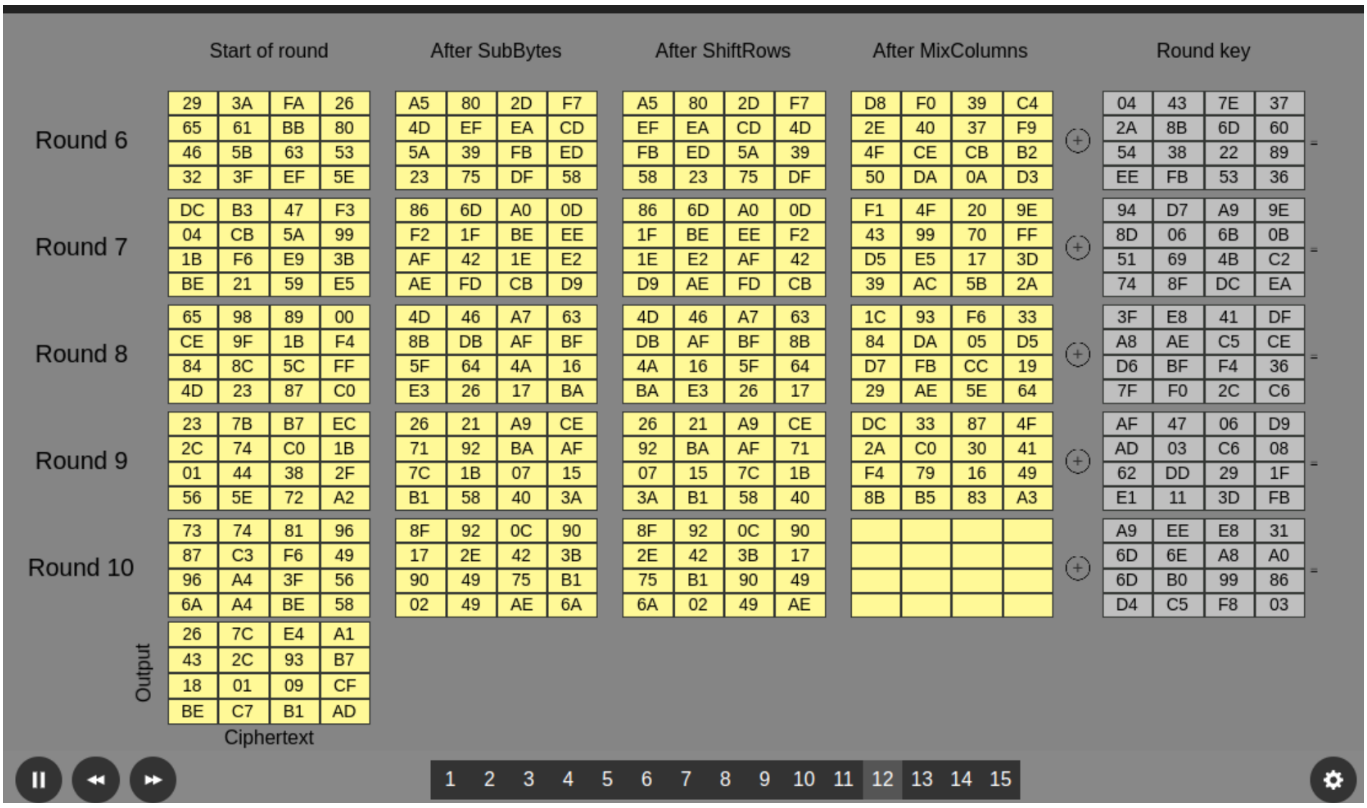
1.2 AES Encryption (Visualisation)

The following uses https://www.cryptool.org/en/cto/aes-animation.

As in #1, the same 32 hex-key is used, , `9dfc79e4 89684c8c 909df0ac ed2de066`. This time, using a valid UTF-8 plaintext string, `"How long is a piece of string?"`. The resultant config and ciphertext is then,



The animation pauses on the 12th slide to show the various outputs of each round-step,



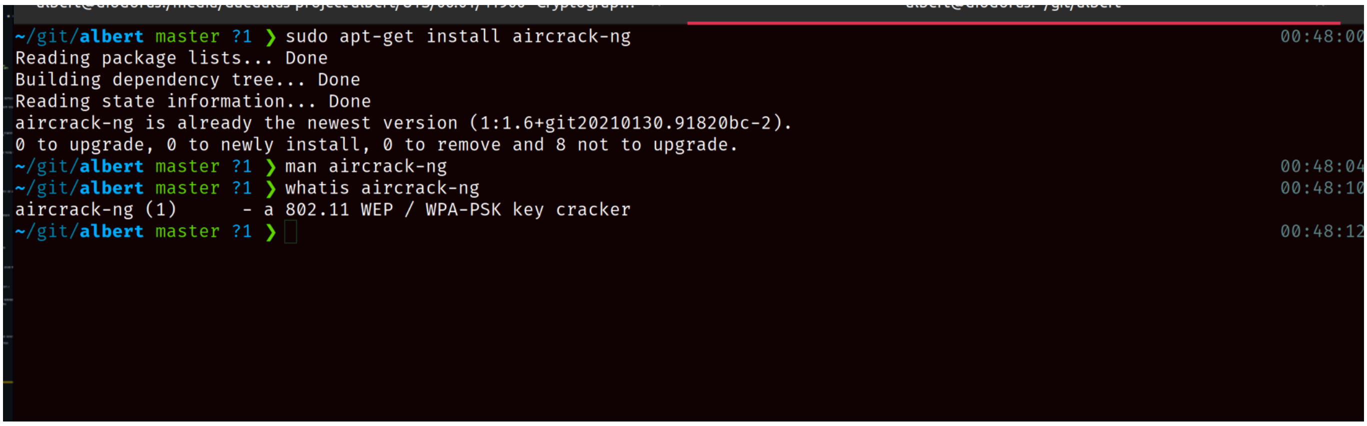
Example one only allows manual hex inputs and limits the input to the length of the key. I am no bothering with typing it out manually for comparison (tempting though). However, if I did then I would expect each round tab to contain matching 32 hex values with the above screenshot. Where each row from the screenshot is a segment in the round table.

Part 2. Wi-Fi Cracking

2.1 Setup

Installing the dependencies is as simple as,

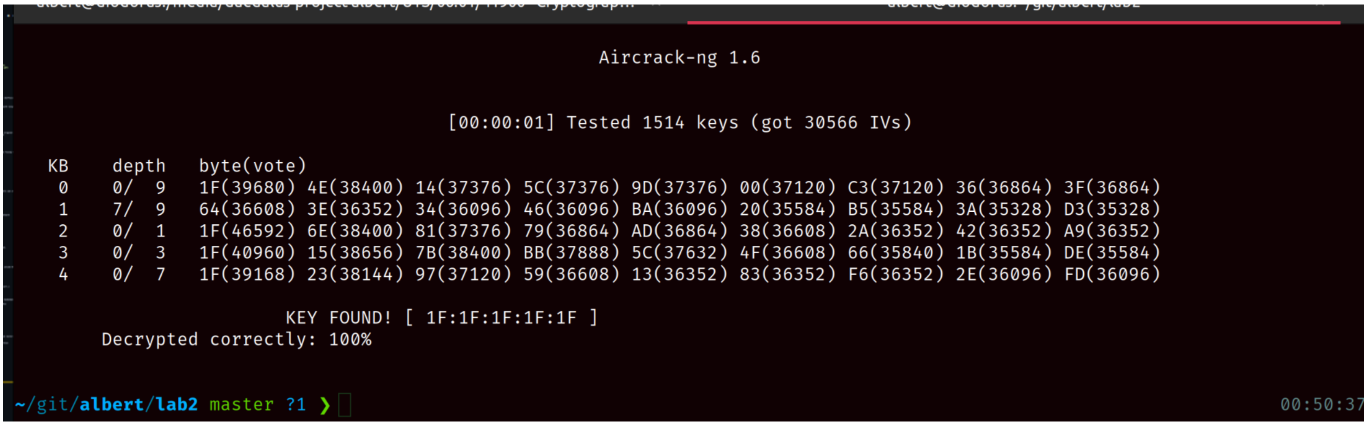
```
sudo apt-get install aircrack-ng
```



2.2 Cracking pre-prepared WEP network dumps

Running the following I can crack the provided dumps using PTW and Korek modes,

```
aircrack-ng wep_64_ptw.cap
# then later
aircrack-ng -K wep_KoreK.ivs
```



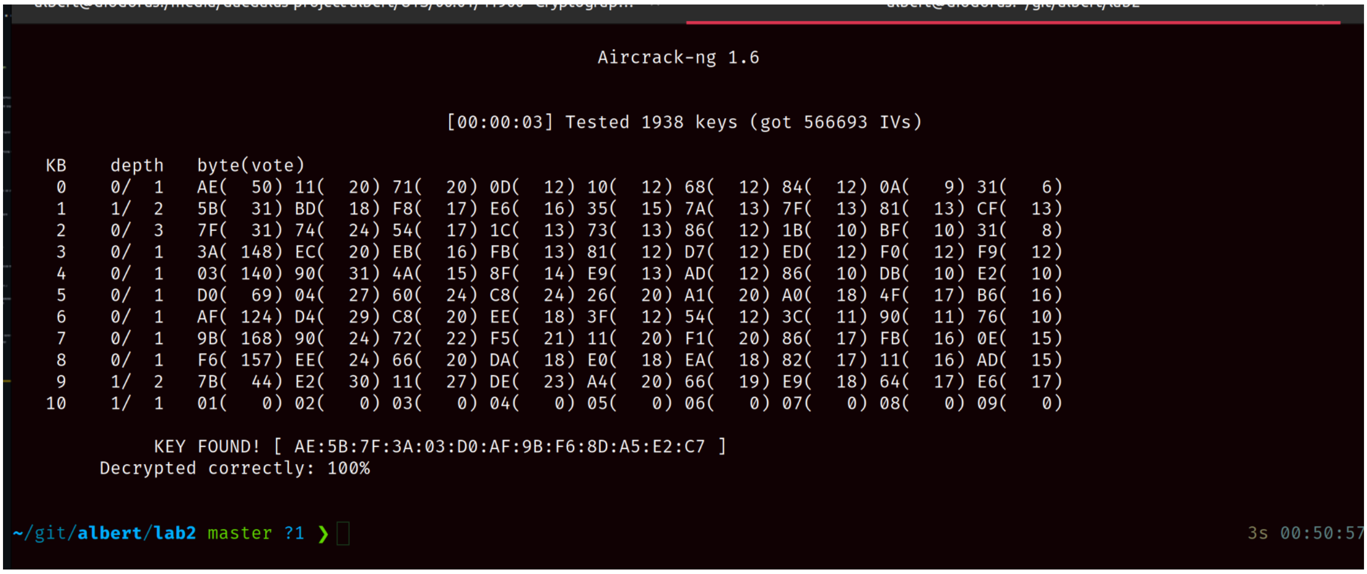
```
                              Aircrack-ng 1.6

                 [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) 00(37120) C3(37120) 36(36864) 3F(36864)
    1    7/  9   64(36608) 3E(36352) 34(36096) 46(36096) BA(36096) 20(35584) B5(35584) 3A(35328) D3(35328)
    2    0/  1   1F(46592) 6E(38400) 81(37376) 79(36864) AD(36864) 38(36608) 2A(36352) 42(36352) A9(36352)
    3    0/  3   1F(40960) 15(38656) 7B(38400) BB(37888) 5C(37632) 4F(36608) 66(35840) 1B(35584) DE(35584)
    4    0/  7   1F(39168) 23(38144) 97(37120) 59(36608) 13(36352) 83(36352) F6(36352) 2E(36096) FD(36096)

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

~/git/albert/lab2 master ?1 ❯                                                                   00:50:37
```



```
                              Aircrack-ng 1.6

                 [00:00:03] Tested 1938 keys (got 566693 IVs)

   KB    depth   byte(vote)
    0    0/  1   AE(  50) 11(  20) 71(  20) 0D(  12) 10(  12) 68(  12) 84(  12) 0A(   9) 31(   6)
    1    1/  2   5B(  31) BD(  18) F8(  17) E6(  16) 35(  15) 7A(  13) 7F(  13) 81(  13) CF(  13)
    2    0/  3   7F(  31) 74(  24) 54(  17) 1C(  13) 73(  13) 86(  12) 1B(  10) BF(  10) 31(   8)
    3    0/  1   3A( 148) EC(  20) EB(  16) FB(  13) 81(  12) D7(  12) ED(  12) F0(  12) F9(  12)
    4    0/  1   03( 140) 90(  31) 4A(  15) 8F(  14) E9(  13) AD(  12) 86(  10) DB(  10) E2(  10)
    5    0/  1   D0(  69) 04(  27) 60(  24) C8(  24) 26(  20) A1(  20) A0(  18) 4F(  17) B6(  16)
    6    0/  1   AF( 124) D4(  29) C8(  20) EE(  18) 3F(  12) 54(  12) 3C(  11) 90(  11) 76(  10)
    7    0/  1   9B( 168) 90(  24) 72(  22) F5(  21) 11(  20) F1(  20) 86(  17) FB(  16) 0E(  15)
    8    0/  1   F6( 157) EE(  24) 66(  20) DA(  18) E0(  18) EA(  18) 82(  17) 11(  16) AD(  15)
    9    1/  2   7B(  44) E2(  30) 11(  27) DE(  23) A4(  20) 66(  19) E9(  18) 64(  17) E6(  17)
   10    1/  1   01(   0) 02(   0) 03(   0) 04(   0) 05(   0) 06(   0) 07(   0) 08(   0) 09(   0)

             KEY FOUND! [ AE:5B:7F:3A:03:D0:AF:9B:F6:8D:A5:E2:C7 ]
        Decrypted correctly: 100%

~/git/albert/lab2 master ?1 ❯                                                                3s 00:50:57
```

2.3 Cracking pre-prepared WPA2 network dumps

Using the pre-prepared WPA2 handshake capture, I ran the following to execute a dictionary attack,

```
aircrack-ng -w password.lst wpa2.cap
```

The result was instant, and resolved the expected password (`12345678`),

```
                              Aircrack-ng 1.6

      [00:00:00] 21/2294 keys tested (666.67 k/s)

      Time left: 3 seconds                                      0.92%

                          KEY FOUND! [ 12345678 ]


      Master Key     : EE 51 88 37 93 A6 F6 8E 96 15 FE 73 C8 0A 3A A6
                       F2 DD 0E A5 37 BC E6 27 B9 29 18 3C C6 E5 79 25

      Transient Key  : EA 0E 40 46 33 C8 02 45 03 02 86 8C CA A7 49 DE
                       5C BA 5A BC B2 67 E2 DE 1D 5E 21 E5 7A CC D5 07
                       9B 31 E9 FF 22 0E 13 2A E4 F6 ED 9E F1 AC C8 85
                       45 82 5F C3 2E E5 59 61 39 5A E4 37 34 D6 C1 07

      EAPOL HMAC     : D5 35 53 82 B8 A9 B8 06 DC AF 99 CD AF 56 4E B6

~/git/albert/lab2 master ?1 ❯                                                00:53:28
```

Part 3. WPA2/3 Questions

> Q1: What is the vulnerability of WPA2 Personal?

There are several vulnerabilities, primarly the potential for brute-force attacks. Secondary is the "castle" defence, as a single bad-actor can snoop other clients' traffic once the network is breached.

> Q2: How does WPA3 solve WPA2 shortcomings?

In two ways primarily,

1. PSK is dropped in favour of Simultaneous Authentication Exchange (SAE) forces realtime-handshakes and creates forward secrecy
2. personalized encryption between clients, avoids bad-actors snooping other clients' traffic.

> Q3: Is there any possible attacks against WPA3?

Yes, although either non-trivial or novel. The "Dragonblood" analysis provide details various vulnerabilities. These revolve around the SAE "dragonfly" handshake and often rely on misimplementations of the underlying protoco. These include,

- various side-channel attacks (timing and cache based),
- DOS attacks during the SAE handshake
- and downgrade attacks during the handshake

These are revised as recently as 2020 based on this source.

Part 3. Further Questions

> Q1: How to capture the four-way handshake?

As described in the related reading. By using WireShark in monitor mode, the packets can be passively scanned once connected to the target AP. They can be filtered from by searching for the `EAPOL` protocol.

> Q2: What is the vulnerability of WEP used in this cracking?

WEP utilises an IV prefix to create "randomness". This is available in plaintext and easily gathered in significant quantities (40 - 85K packets). This lab provided the packet captures for both the PTW and FMS/Koreks methods. By XOR'ng packets with the same IV prefix, the original packet can be recovered and then the key is easily deduced.

> Q3: What is the vulnerability of WPA2 used in this cracking?

WPA2 utilises a 4-way handshake during it's setup. This can be captured passively and attacked offline to apply a brute-force dictionary attack. This lab provided the handshake capture and an example (password) dictionary to demonstrate.

> Q4: What have you learnt in this lab?

I learnt about the WEP, WPA2, and WPA3 specifications in depth leading up to this lab. This lab encouraged me to read further into the various vulnerabilities and explore the KRACKs, Dragonblood, and PTW papers in-depth. Overall, I have learnt both that wireless security is still a "new" domain with several exploits, and that I should immediately upgrade my personal network's security.

Summary

In summary, this lab had me revise my notes on AES and the various Wi-Fi standards. The initial AES part was interesting, as it provided a visual example for the algorithm. Ironically, I think this is the first and most concise example of the AES algorithm from all the examples in this subject so far. Although, I personally found the original Rijndael paper far more useful. The remainder of the lab was a quick practical on the usage of aircrack to demonstrate the vulnerabilities of WEP and WPA2. This was an excellent intro to the tool, and thanfully skipped the packet sniffing setup step that so many other labs would've encouraged.