NETWORK AUDIT AND SECURITY PROJECT

REALIZATION OF A ”BRUTEFORCE” ATTACK FOR ACCESS TO A WPA-PSK PROTECTED WIFI NETWORK

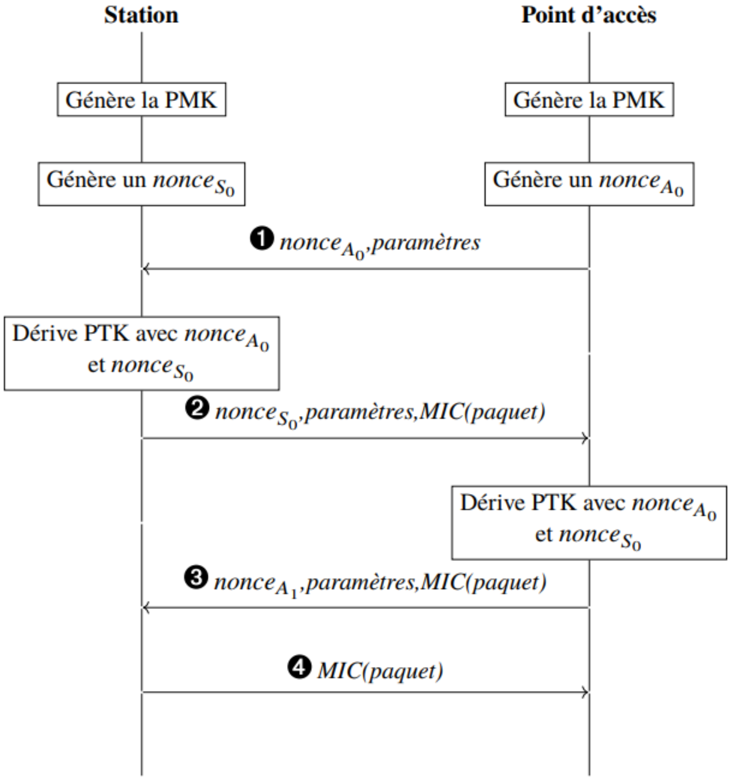
Author
Viet-Huy HA
Anh-Kiet DUONG

Contents

1 Introduction

Wireless security is a crucial aspect of staying safe online. Connecting to the internet over insecure links or networks is a security risk that could potentially lead to data loss, leaked account credentials, and the installation of malware on your network. Using the proper Wi-Fi security measures is critical – but in doing so, it's important to understand the differences between different wireless encryption standards, including WEP, WPA, WPA2, and WPA3.

Wi-Fi Protected Access (WPA) is a security standard for computing devices with wireless internet connections. It was developed by the Wi-Fi Alliance to provide better data encryption and user authentication than Wired Equivalent Privacy (WEP), which was the original Wi-Fi security standard. Since the late 1990s, Wi-Fi security types have gone through multiple evolutions to improve them.



Station
Point d'accès
Génère la PMK
Génère la PMK
Génère un $nonce_{S_0}$
Génère un $nonce_{A_0}$
❶ $nonce_{A_0}$,paramètres
Dérive PTK avec $nonce_{A_0}$ et $nonce_{S_0}$
❷ $nonce_{S_0}$,paramètres,MIC(paquet)
Dérive PTK avec $nonce_{A_0}$ et $nonce_{S_0}$
❸ $nonce_{A_1}$,paramètres,MIC(paquet)
❹ MIC(paquet)

Figure 1: The 4-way handshake WPA/WPA2

In this project, we will learn how to attack WPA-PSK security by ”brute force”. Through the analysis of the captured packet, we will reconstruct step by step the original WPA-PSK encryption, thereby creating a password dictionary and performing the attack.

2 Analysis and Attack

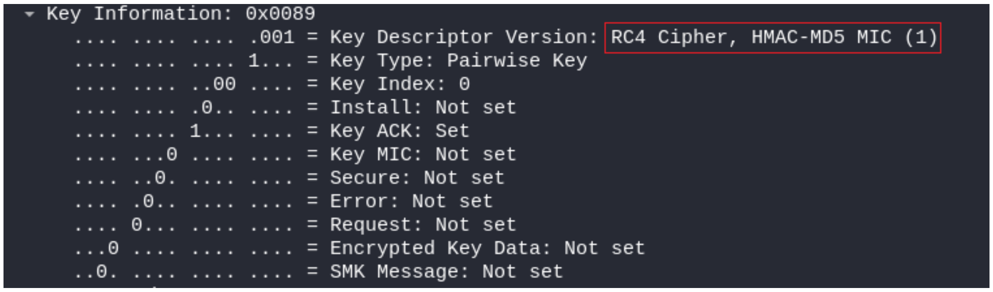
In this section, we will go into detail about analyzing target packets with two types of encryption WPA (TKIP) and WPA2 (AES) respectively. From there, build an attack method and create a complete tool. And we will use Wireshark as a support tool to analyze and get information from the captured package

2.1 WPA

We'll start with the captured package **capture_wpa.pcap**. This file is taken from the **Audit & Sécurité réseaux** page at `http://p-fb.net/`.

2.1.1 Detect type of encryption

For the first, we need to detect type of encryption. We will look at the "Key Information" section in the first package to detect the type of encryption in use. And we got $\boxed{RC4\,Cipher,\ HMAC-MD5\,MIC}$



```
▾ Key Information: 0x0089
    .... .... .... .001 = Key Descriptor Version: RC4 Cipher, HMAC-MD5 MIC (1)
    .... .... .... 1... = Key Type: Pairwise Key
    .... .... ..00 .... = Key Index: 0
    .... .... .0.. .... = Install: Not set
    .... .... 1... .... = Key ACK: Set
    .... ...0 .... .... = Key MIC: Not set
    .... ..0. .... .... = Secure: Not set
    .... .0.. .... .... = Error: Not set
    .... 0... .... .... = Request: Not set
    ...0 .... .... .... = Encrypted Key Data: Not set
    ..0. .... .... .... = SMK Message: Not set
```

Figure 2: Encryption Type

2.1.2 PMK

The first step of WPA encryption is the calculation of the PMK. So we will build a dictionary for PSK and look up the SSID from the captured package

1. **Build PSK dictionary**

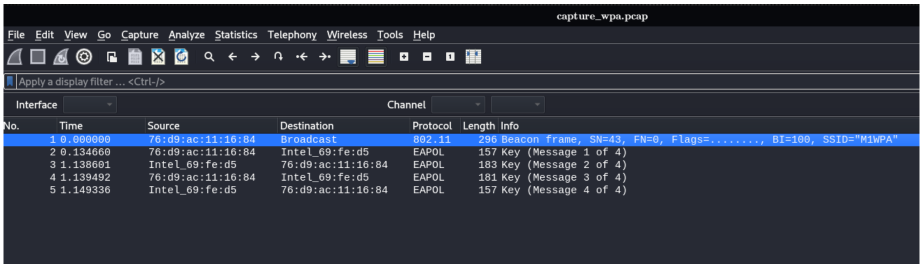
   Based on the suggestion given by the project, we know that all letters are lowercase and start with four 'a' characters, and PSK length is 8. So we will build a dictionary for PSK with any set of 4 lowercase characters. This is Python code:

```python
import itertools

letters = 'abcdefghijklmnopqrstuvwxyz'

words = [''.join(x) for x in itertools.product(letters, repeat=4)]

print(len(words))
with open('pwd_list.txt', 'w') as file:
    for word in words:
        file.write(''.join(word) + '\n')

```

Listing 1: PSK generate

2. **SSID**

   After opening the package with Wireshark, we can immediately see the $\boxed{SSID} = \boxed{WPAM1}$



capture_wpa.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Interface
Channel

```
No.  Time      Source             Destination        Protocol Length Info
   1 0.000000  76:d9:ac:11:16:84  Broadcast          802.11      296 Beacon frame, SN=43, FN=0, Flags=........, BI=100, SSID="M1WPA"
   2 0.134660  76:d9:ac:11:16:84  Intel_69:fe:d5     EAPOL       157 Key (Message 1 of 4)
   3 1.138601  Intel_69:fe:d5     76:d9:ac:11:16:84  EAPOL       183 Key (Message 2 of 4)
   4 1.139492  76:d9:ac:11:16:84  Intel_69:fe:d5     EAPOL       181 Key (Message 3 of 4)
   5 1.149336  Intel_69:fe:d5     76:d9:ac:11:16:84  EAPOL       157 Key (Message 4 of 4)
```

Figure 3: SSID

2.1.3 PTK

After obtaining the PMK, we will proceed to calculate the PTK. To achieve this we need to search for $Nonce_A$, $Nonce_S$, MAC_{Client}, $MAC_{Authenticator}$, Finnaly, use this information to calculate the PTK using the PRF_512 function.

$$\boxed{PTK = PRF\_512(PMK, "Pairwise\ key\ expansion", LowerMAC\|\|HigerMAC\|\|LowerNonce\|\|HigherNonce)}$$

1. $Nonce_A$, $Nonce_S$, MAC_{Client}, $MAC_{Authenticator}$

   Based on analyzing the captured package using Wireshark, we got the following information:

   $\boxed{Nonce_A} = \boxed{7c67f224a6e08193230feeb0eff9a07ec6cbf0163f962ba34d31dbdb2bc69d8d}$

   $\boxed{Nonce_S} = \boxed{eea4124e3facf8e0270db587fceee4da1c2a689be96b931fc26d35b4c7dbbbae}$

   $\boxed{MAC_{Client}} = \boxed{76 : d9 : ac : 11 : 16 : 84}$

   $\boxed{MAC_{Authenticator}} = \boxed{00 : 0e : 35 : 69 : fe : d5}$

2. **PRF_512**

   Based on the tutorial we have built the PRF_512 calculation function as follows:

```python
    def PRFn(self, K, A, B, n = 512):
        i = 0
        R = b''
        while len(R)*8 < n:
            data = A + b'\x00' + B + bytes([i])
            r = hmac.new(K, data, sha1).digest()
            R += r
            i += 1
        return R[:n//8]

```

Listing 2: PRF_512

With:

A = "Pairwise key expansion",

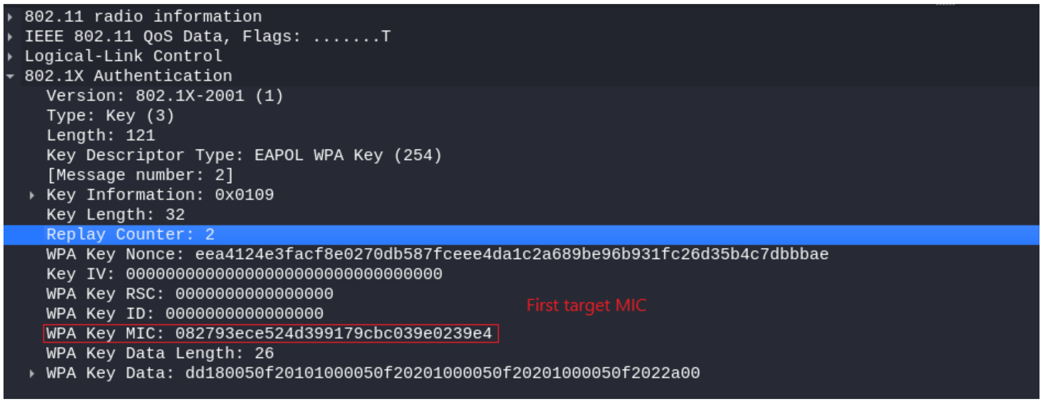
B = $LowerMAC\|\|HigerMAC\|\|LowerNonce\|\|HigherNonce$

2.1.4 MIC generation

The next step, once we have the necessary information, we will proceed to calculate the MIC from the corresponding EAPOL. Continuing to use Wireshark and based on the original data exchange diagram, we look at the packages in turn and get the following target MIC values and EAPOL (include MIC):

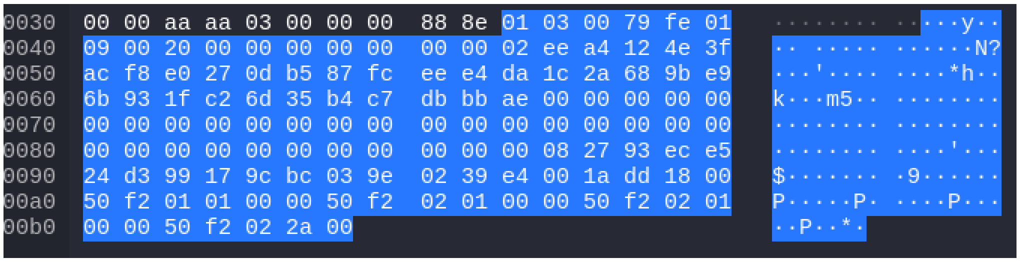
First target MIC and First EAPOL

From second frame, we have:



```
▸ 802.11 radio information
▸ IEEE 802.11 QoS Data, Flags: .......T
▸ Logical-Link Control
▾ 802.1X Authentication
    Version: 802.1X-2001 (1)
    Type: Key (3)
    Length: 121
    Key Descriptor Type: EAPOL WPA Key (254)
    [Message number: 2]
  ▸ Key Information: 0x0109
    Key Length: 32
    Replay Counter: 2
    WPA Key Nonce: eea4124e3facf8e0270db587fceee4da1c2a689be96b931fc26d35b4c7dbbbae
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: 0000000000000000
    WPA Key ID: 0000000000000000
    WPA Key MIC: 082793ece524d399179cbc039e0239e4
    WPA Key Data Length: 26
  ▸ WPA Key Data: dd180050f20101000050f20201000050f20201000050f2022a00
```

Figure 4: First target MIC



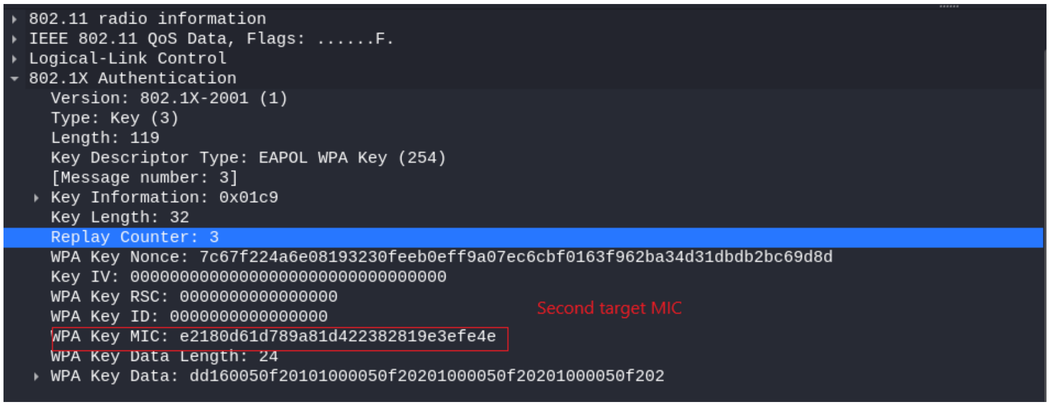
```
0030  00 00 aa aa 03 00 00 00  88 8e 01 03 00 79 fe 01   ........ .....y..
0040  09 00 20 00 00 00 00 00  00 00 02 ee a4 12 4e 3f   .. ..... ......N?
0050  ac f8 e0 27 0d b5 87 fc  ee e4 da 1c 2a 68 9b e9   ...'.... ....*h..
0060  6b 93 1f c2 6d 35 b4 c7  db bb ae 00 00 00 00 00   k...m5.. ........
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0080  00 00 00 00 00 00 00 00  00 00 00 08 27 93 ec e5   ........ ....'...
0090  24 d3 99 17 9c bc 03 9e  02 39 e4 00 1a dd 18 00   $....... .9......
00a0  50 f2 01 01 00 00 50 f2  02 01 00 00 50 f2 02 01   P.....P. ....P...
00b0  00 00 50 f2 02 2a 00                               ..P..*.
```

Figure 5: First EAPOL

$$\boxed{MIC1} = \boxed{082793ece524d399179cbc039e0239e4}$$

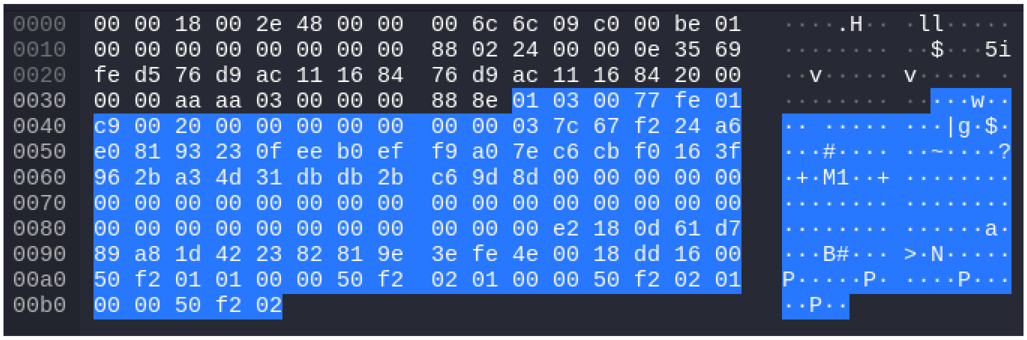
Second target MIC and Second EAPOL

From third frame, we have:



```
▸ 802.11 radio information
▸ IEEE 802.11 QoS Data, Flags: ......F.
▸ Logical-Link Control
▾ 802.1X Authentication
    Version: 802.1X-2001 (1)
    Type: Key (3)
    Length: 119
    Key Descriptor Type: EAPOL WPA Key (254)
    [Message number: 3]
  ▸ Key Information: 0x01c9
    Key Length: 32
    Replay Counter: 3
    WPA Key Nonce: 7c67f224a6e08193230feeb0eff9a07ec6cbf0163f962ba34d31dbdb2bc69d8d
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: 0000000000000000
    WPA Key ID: 0000000000000000
    WPA Key MIC: e2180d61d789a81d422382819e3efe4e
    WPA Key Data Length: 24
  ▸ WPA Key Data: dd160050f20101000050f20201000050f20201000050f202
```

Figure 6: Second target MIC



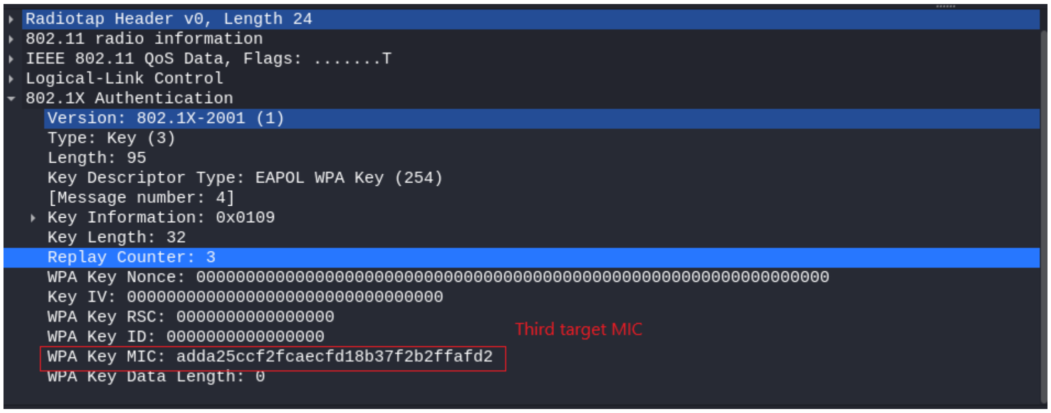
```
0000  00 00 18 00 2e 48 00 00  00 6c 6c 09 c0 00 be 01   ·····.H·· ·ll·····
0010  00 00 00 00 00 00 00 00  88 02 24 00 00 0e 35 69   ········ ··$···5i
0020  fe d5 76 d9 ac 11 16 84  76 d9 ac 11 16 84 20 00   ··v····· v····· ·
0030  00 00 aa aa 03 00 00 00  88 8e 01 03 00 77 fe 01   ········ ·····w··
0040  c9 00 20 00 00 00 00 00  00 00 03 7c 67 f2 24 a6   ·· ····· ···|g·$·
0050  e0 81 93 23 0f ee b0 ef  f9 a0 7e c6 cb f0 16 3f   ···#···· ··~····?
0060  96 2b a3 4d 31 db db 2b  c6 9d 8d 00 00 00 00 00   ·+·M1··+ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 00 00 e2 18 0d 61 d7   ········ ······a·
0090  89 a8 1d 42 23 82 81 9e  3e fe 4e 00 18 dd 16 00   ···B#··· >·N·····
00a0  50 f2 01 01 00 00 50 f2  02 01 00 00 50 f2 02 01   P·····P· ····P···
00b0  00 00 50 f2 02                                     ··P··
```

Figure 7: Second EAPOL

$$\boxed{MIC2} = \boxed{e2180d61d789a81d422382819e3efe4e}$$

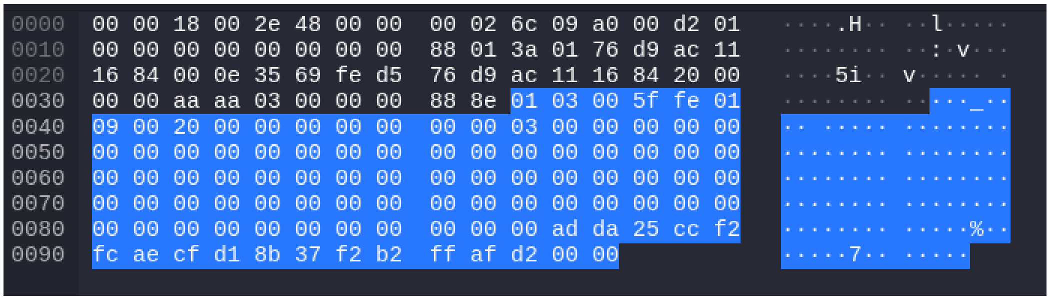
Third target MIC and Second EAPOL

From fourth frame, we have:



```
▸ Radiotap Header v0, Length 24
▸ 802.11 radio information
▸ IEEE 802.11 QoS Data, Flags: .......T
▸ Logical-Link Control
▾ 802.1X Authentication
    Version: 802.1X-2001 (1)
    Type: Key (3)
    Length: 95
    Key Descriptor Type: EAPOL WPA Key (254)
    [Message number: 4]
  ▸ Key Information: 0x0109
    Key Length: 32
    Replay Counter: 3
    WPA Key Nonce: 0000000000000000000000000000000000000000000000000000000000000000
    Key IV: 00000000000000000000000000000000
    WPA Key RSC: 0000000000000000
    WPA Key ID: 0000000000000000
    WPA Key MIC: adda25ccf2fcaecfd18b37f2b2ffafd2
    WPA Key Data Length: 0
```

Figure 8: Third target MIC



```
0000  00 00 18 00 2e 48 00 00  00 02 6c 09 a0 00 d2 01   ····.H·· ··l·····
0010  00 00 00 00 00 00 00 00  88 01 3a 01 76 d9 ac 11   ········ ··:·v···
0020  16 84 00 0e 35 69 fe d5  76 d9 ac 11 16 84 20 00   ····5i·· v····· ·
0030  00 00 aa aa 03 00 00 00  88 8e 01 03 00 5f fe 01   ········ ·····_··
0040  09 00 20 00 00 00 00 00  00 00 03 00 00 00 00 00   ·· ····· ········
0050  00 00 00 00 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 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 00 00 ad da 25 cc f2   ········ ·····%··
0090  fc ae cf d1 8b 37 f2 b2  ff af d2 00 00            ·····7·· ·····
```

Figure 9: Third EAPOL

$$\boxed{MIC3} = \boxed{adda25ccf2fcaecfd18b37f2b2ffafd2}$$

MIC Calculation

We will calculate the MIC from each PSK in the dictionary and EAPOL without MIC, then compare it with the target MIC, thereby finding the correct password.

$$\boxed{MIC = HMAC_{MD5}(KCK, EAPOL_{noMIC})}$$

With KCK:

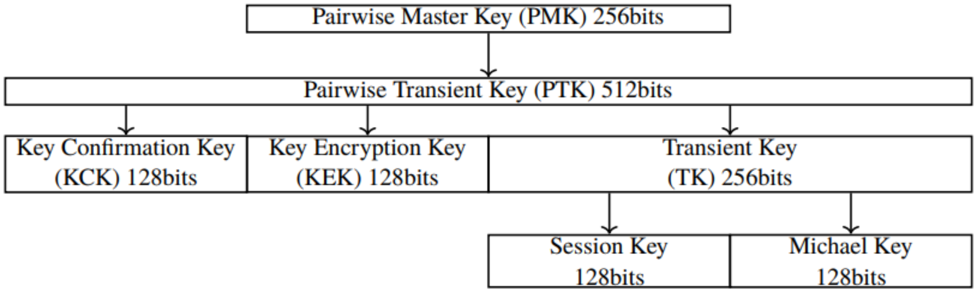

Pairwise Master Key (PMK) 256bits
Pairwise Transient Key (PTK) 512bits
Key Confirmation Key (KCK) 128bits
Key Encryption Key (KEK) 128bits
Transient Key (TK) 256bits
Session Key 128bits
Michael Key 128bits

Figure 10: Key in WPA

2.1.5 Bruteforce Attacking

This is last step in our project. After getting all of information, we proceed to program script for doing brute force PSK. We obtained following result:

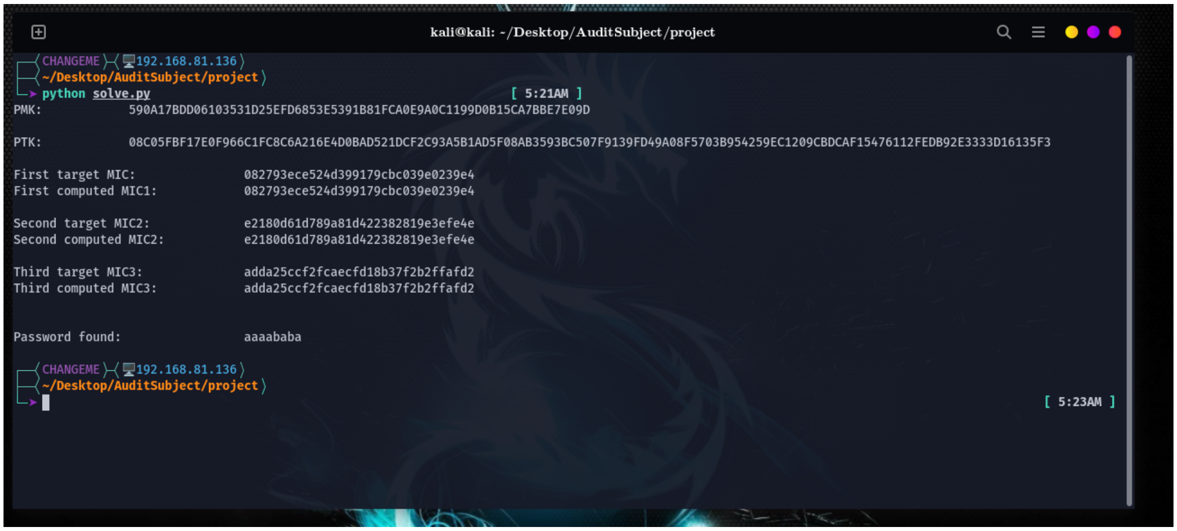

```
kali@kali: ~/Desktop/AuditSubject/project
┌─⟨CHANGEME⟩─⟨192.168.81.136⟩
├─⟨~/Desktop/AuditSubject/project⟩
└─> python solve.py                                               [ 5:21AM ]
PMK:            590A17BDD06103531D25EFD6853E5391B81FCA0E9A0C1199D0B15CA7BBE7E09D

PTK:            08C05FBF17E0F966C1FC8C6A216E4D0BAD521DCF2C93A5B1AD5F08AB3593BC507F9139FD49A08F5703B954259EC1209CBDCAF15476112FEDB92E3333D16135F3

First target MIC:               082793ece524d399179cbc039e0239e4
First computed MIC1:            082793ece524d399179cbc039e0239e4

Second target MIC2:             e2180d61d789a81d422382819e3efe4e
Second computed MIC2:           e2180d61d789a81d422382819e3efe4e

Third target MIC3:              adda25ccf2fcaecfd18b37f2b2ffafd2
Third computed MIC3:            adda25ccf2fcaecfd18b37f2b2ffafd2


Password found:                 aaaababa

┌─⟨CHANGEME⟩─⟨192.168.81.136⟩
├─⟨~/Desktop/AuditSubject/project⟩
└─>                                                                                                           [ 5:23AM ]
```

Figure 11: Final Result

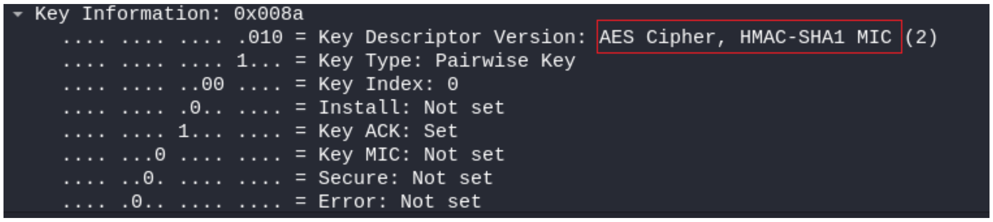
As we can see, all of MICs were matched. Then we got $\boxed{PSK} = \boxed{aaaababa}$

2.2 WPA2

In this section, we'll start with the captured package **wpa-Induction.pcap**. This file is taken from `https://wiki.wireshark.org/SampleCaptures`.
Similar to the WPA part, we also follow the same steps

2.2.1 Detect type of encryption

Look at the Key Information section, we got $\boxed{AESCipher, HMAC - SHA1MIC}$



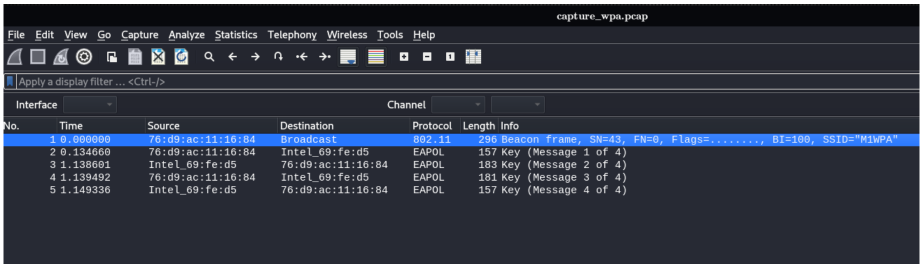
```
▾ Key Information: 0x008a
    .... .... .... .010 = Key Descriptor Version: AES Cipher, HMAC-SHA1 MIC (2)
    .... .... .... 1... = Key Type: Pairwise Key
    .... .... ..00 .... = Key Index: 0
    .... .... .0.. .... = Install: Not set
    .... .... 1... .... = Key ACK: Set
    .... ...0 .... .... = Key MIC: Not set
    .... ..0. .... .... = Secure: Not set
    .... .0.. .... .... = Error: Not set
```

Figure 12: Encryption Type

2.2.2 PMK

1. **Build PSK dictionary**
   Assume that we knew first 5 characters are "Induc". We will create PSK dictionary with the same above way

2. **SSID**
   $\boxed{SSID} = \boxed{Coherer}$



capture_wpa.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Interface
Channel

```
No.   Time       Source              Destination         Protocol Length Info
    1 0.000000   76:d9:ac:11:16:84   Broadcast           802.11     296 Beacon frame, SN=43, FN=0, Flags=........, BI=100, SSID="M1WPA"
    2 0.134660   76:d9:ac:11:16:84   Intel_69:fe:d5      EAPOL      157 Key (Message 1 of 4)
    3 1.138601   Intel_69:fe:d5      76:d9:ac:11:16:84   EAPOL      183 Key (Message 2 of 4)
    4 1.139492   76:d9:ac:11:16:84   Intel_69:fe:d5      EAPOL      181 Key (Message 3 of 4)
    5 1.149336   Intel_69:fe:d5      76:d9:ac:11:16:84   EAPOL      157 Key (Message 4 of 4)
```

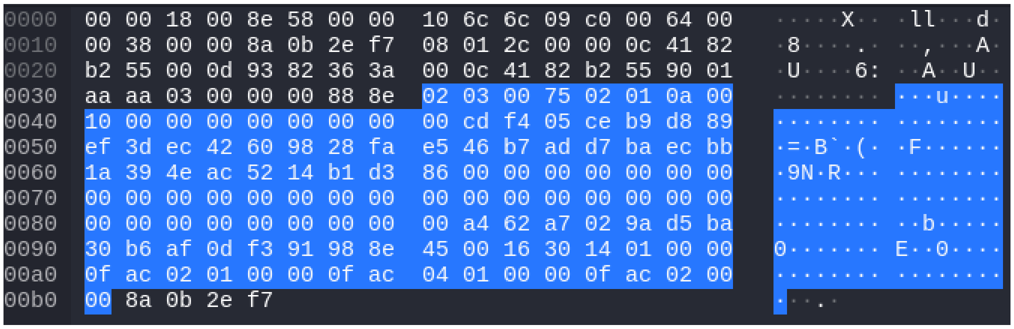
Figure 13: SSID

2.2.3 PTK

$Nonce_A$, $Nonce_S$, MAC_{Client}, $MAC_{Authenticator}$
Based on analyzing the captured package using Wireshark, we got the following information:
$\boxed{Nonce_A} = \boxed{3e8e967dacd960324cac5b6aa721235bf57b949771c867989f49d04ed47c6933}$
$\boxed{Nonce_S} = \boxed{cdf405ceb9d889ef3dec42609828fae546b7add7baecbb1a394eac5214b1d386}$
$\boxed{MAC_{Client}} = \boxed{00:0c:41:82:b2:55}$
$\boxed{MAC_{Authenticator}} = \boxed{00:0d:93:82:36:3a}$

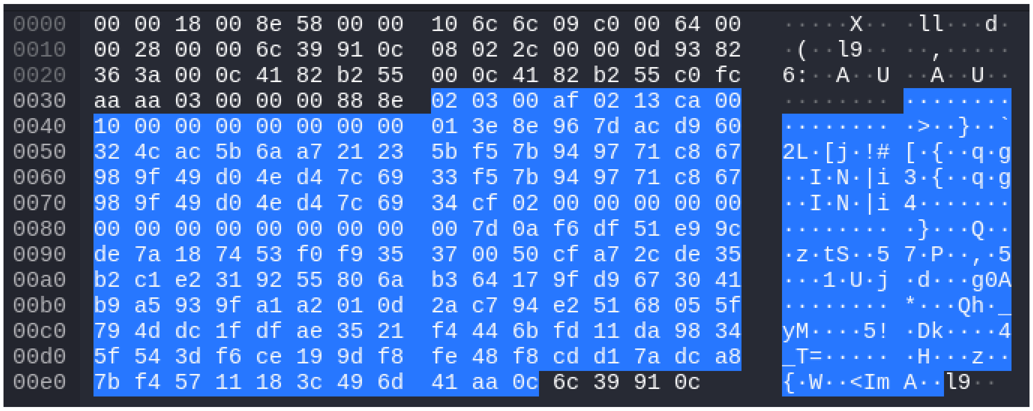
2.2.4 MIC generation

Similar to the previous section, we capture using Wireshark:
$\boxed{MIC1} = \boxed{a462a7029ad5ba30b6af0df391988e45}$
$\boxed{MIC2} = \boxed{7d0af6df51e99cde7a187453f0f93537}$
$\boxed{MIC3} = \boxed{10bba3bdfbcfde2bc537509d71f2ecd1}$



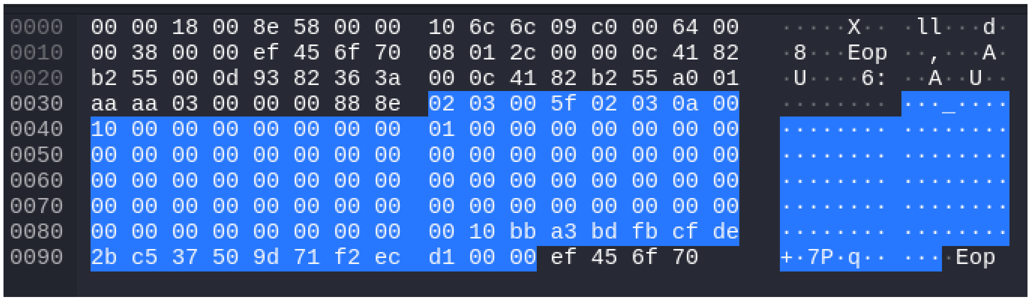
```
0000  00 00 18 00 8e 58 00 00  10 6c 6c 09 c0 00 64 00   ·····X·· ·ll···d·
0010  00 38 00 00 8a 0b 2e f7  08 01 2c 00 00 0c 41 82   ·8····.· ··,···A·
0020  b2 55 00 0d 93 82 36 3a  00 0c 41 82 b2 55 90 01   ·U····6: ··A··U··
0030  aa aa 03 00 00 00 88 8e  02 03 00 75 02 01 0a 00   ········ ···u····
0040  10 00 00 00 00 00 00 00  00 cd f4 05 ce b9 d8 89   ········ ········
0050  ef 3d ec 42 60 98 28 fa  e5 46 b7 ad d7 ba ec bb   ·=·B`·(· ·F······
0060  1a 39 4e ac 52 14 b1 d3  86 00 00 00 00 00 00 00   ·9N·R··· ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 a4 62 a7 02 9a d5 ba   ········ ··b·····
0090  30 b6 af 0d f3 91 98 8e  45 00 16 30 14 01 00 00   0······· E··0····
00a0  0f ac 02 01 00 00 0f ac  04 01 00 00 0f ac 02 00   ········ ········
00b0  00 8a 0b 2e f7                                     ···.·
```

Figure 14: First target MIC



```
0000  00 00 18 00 8e 58 00 00  10 6c 6c 09 c0 00 64 00   ·····X·· ·ll···d·
0010  00 28 00 00 6c 39 91 0c  08 02 2c 00 00 0d 93 82   ·(··l9·· ··,·····
0020  36 3a 00 0c 41 82 b2 55  00 0c 41 82 b2 55 c0 fc   6:··A··U ··A··U··
0030  aa aa 03 00 00 00 88 8e  02 03 00 af 02 13 ca 00   ········ ········
0040  10 00 00 00 00 00 00 00  01 3e 8e 96 7d ac d9 60   ········ ·>··}··`
0050  32 4c ac 5b 6a a7 21 23  5b f5 7b 94 97 71 c8 67   2L·[j·!# [·{··q·g
0060  98 9f 49 d0 4e d4 7c 69  33 f5 7b 94 97 71 c8 67   ··I·N·|i 3·{··q·g
0070  98 9f 49 d0 4e d4 7c 69  34 cf 02 00 00 00 00 00   ··I·N·|i 4·······
0080  00 00 00 00 00 00 00 00  00 7d 0a f6 df 51 e9 9c   ········ ·}···Q··
0090  de 7a 18 74 53 f0 f9 35  37 00 50 cf a7 2c de 35   ·z·tS··5 7·P··,·5
00a0  b2 c1 e2 31 92 55 80 6a  b3 64 17 9f d9 67 30 41   ···1·U·j ·d···g0A
00b0  b9 a5 93 9f a1 a2 01 0d  2a c7 94 e2 51 68 05 5f   ········ *···Qh·_
00c0  79 4d dc 1f df ae 35 21  f4 44 6b fd 11 da 98 34   yM····5! ·Dk····4
00d0  5f 54 3d f6 ce 19 9d f8  fe 48 f8 cd d1 7a dc a8   _T=····· ·H···z··
00e0  7b f4 57 11 18 3c 49 6d  41 aa 0c 6c 39 91 0c      {·W··<Im A··l9··
```

Figure 15: Second EAPOL



```
0000  00 00 18 00 8e 58 00 00  10 6c 6c 09 c0 00 64 00   ·····X·· ·ll···d·
0010  00 38 00 00 ef 45 6f 70  08 01 2c 00 00 0c 41 82   ·8···Eop ··,···A·
0020  b2 55 00 0d 93 82 36 3a  00 0c 41 82 b2 55 a0 01   ·U····6: ··A··U··
0030  aa aa 03 00 00 00 88 8e  02 03 00 5f 02 03 0a 00   ········ ···_····
0040  10 00 00 00 00 00 00 00  01 00 00 00 00 00 00 00   ········ ········
0050  00 00 00 00 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 00   ········ ········
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ········ ········
0080  00 00 00 00 00 00 00 00  00 10 bb a3 bd fb cf de   ········ ········
0090  2b c5 37 50 9d 71 f2 ec  d1 00 00 ef 45 6f 70      +·7P·q·· ····Eop
```

Figure 16: Third EAPOL

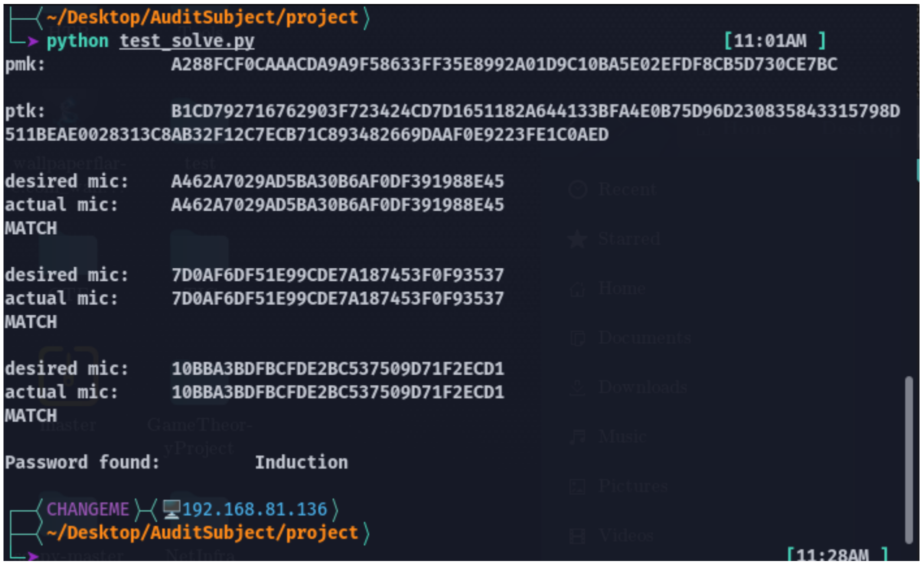
MIC Calculation

We will calculate the MIC from each PSK in the dictionary and EAPOL without MIC, then compare it with the target MIC, thereby finding the correct password.

$\boxed{MIC = HMAC_{SHA1}(KCK, EAPOL_{noMIC})}$

2.2.5 Bruteforce Attacking

We got result:



```
┌─⟨~/Desktop/AuditSubject/project⟩
└─➤ python test_solve.py                                                    [11:01AM ]
pmk:            A288FCF0CAAACDA9A9F58633FF35E8992A01D9C10BA5E02EFDF8CB5D730CE7BC

ptk:            B1CD792716762903F723424CD7D1651182A644133BFA4E0B75D96D230835843315798D
511BEAE0028313C8AB32F12C7ECB71C893482669DAAF0E9223FE1C0AED

desired mic:    A462A7029AD5BA30B6AF0DF391988E45
actual mic:     A462A7029AD5BA30B6AF0DF391988E45
MATCH

desired mic:    7D0AF6DF51E99CDE7A187453F0F93537
actual mic:     7D0AF6DF51E99CDE7A187453F0F93537
MATCH

desired mic:    10BBA3BDFBCFDE2BC537509D71F2ECD1
actual mic:     10BBA3BDFBCFDE2BC537509D71F2ECD1
MATCH

Password found:         Induction

┌─⟨CHANGEME⟩─⟨🖥192.168.81.136⟩
├─⟨~/Desktop/AuditSubject/project⟩
└─➤                                                                              [11:28AM ]
```

Figure 17: Final Result

As we can see, all of MICs were matched. Then we got $\boxed{PSK} = \boxed{Induction}$