Cryptography - Lab 2 - AES and Wi-Fi Authentication Crack

Javier Antxon Garrues Apecechea

Student ID: 24647808

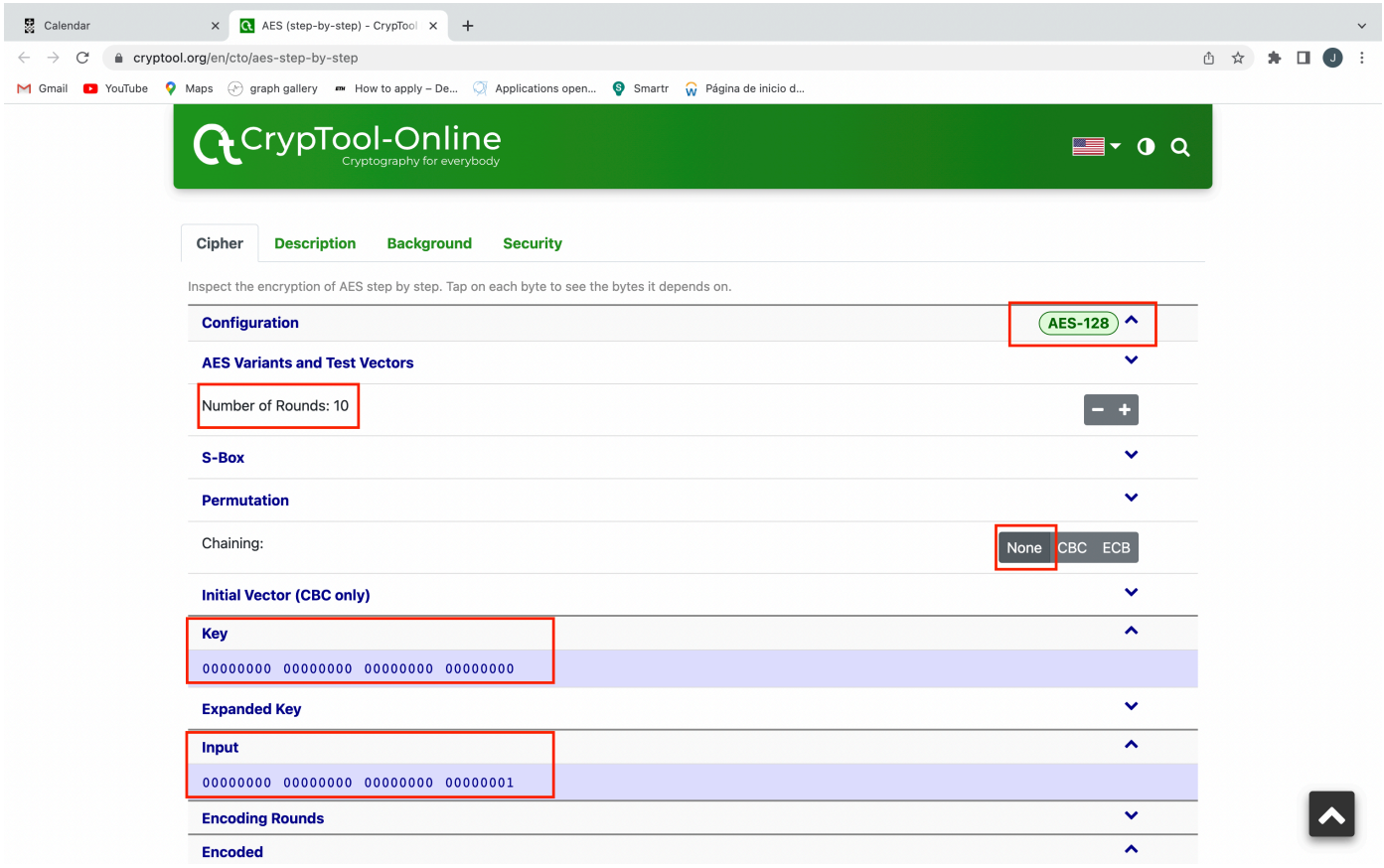
In this lab, we will understand the steps that take place during **AES** and **graphically** visualize them. At last, we will use **aircrack-ng** to crack **WEP** and **WPA** wifi passwords.

Part One: Step by Step AES

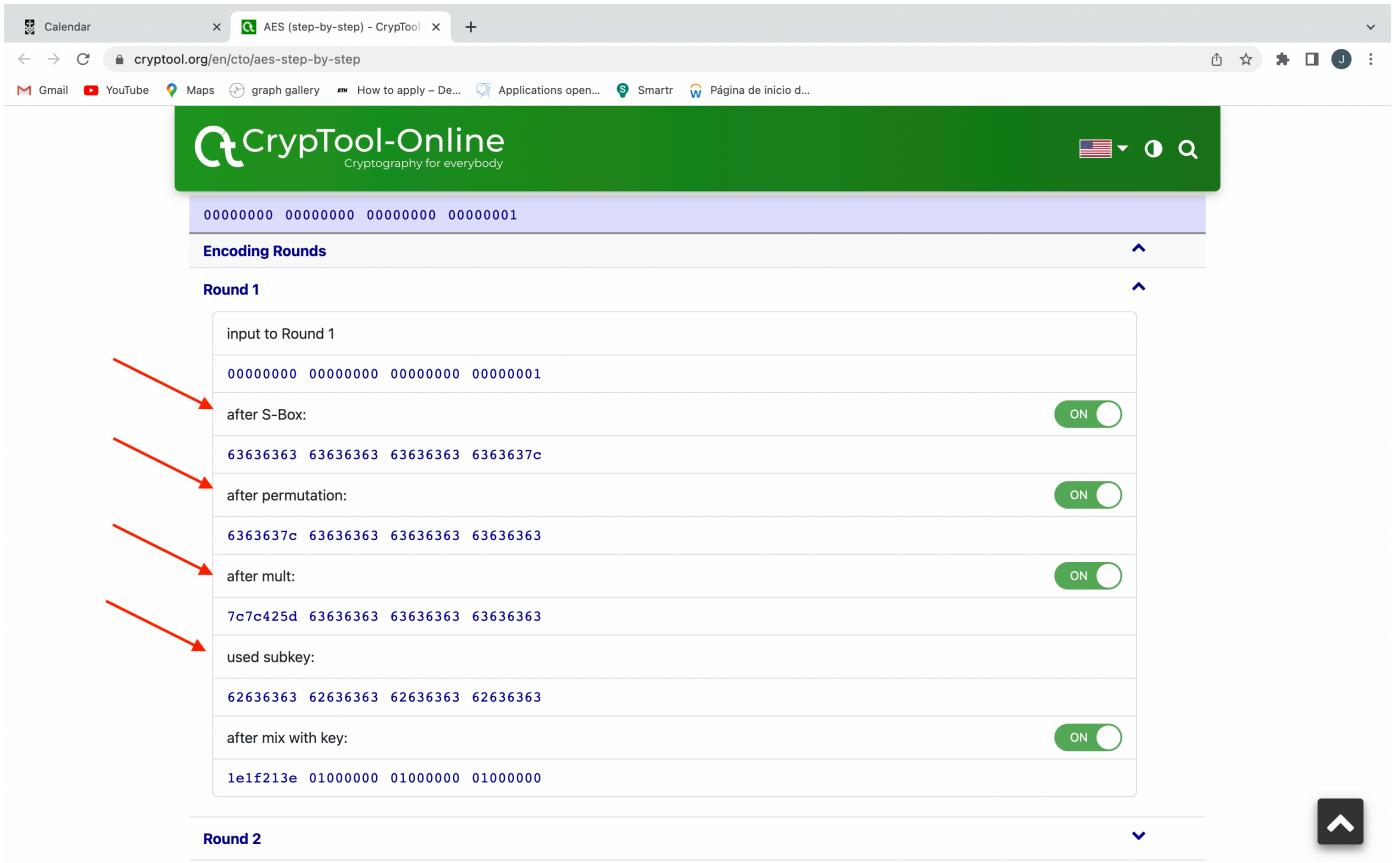
To do this part we will visit the following online website: https://www.cryptool.org/en/cto/aes-step-by-step.

Once there, we will choose an input text and a key (both of them 128-bits length). Furthermore, we will select 10 Rounds and non-chaining.

```
Input text: 00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 01
Key: 00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
```



Then, we will proceed with the encryption and have a look at **Round 1**:



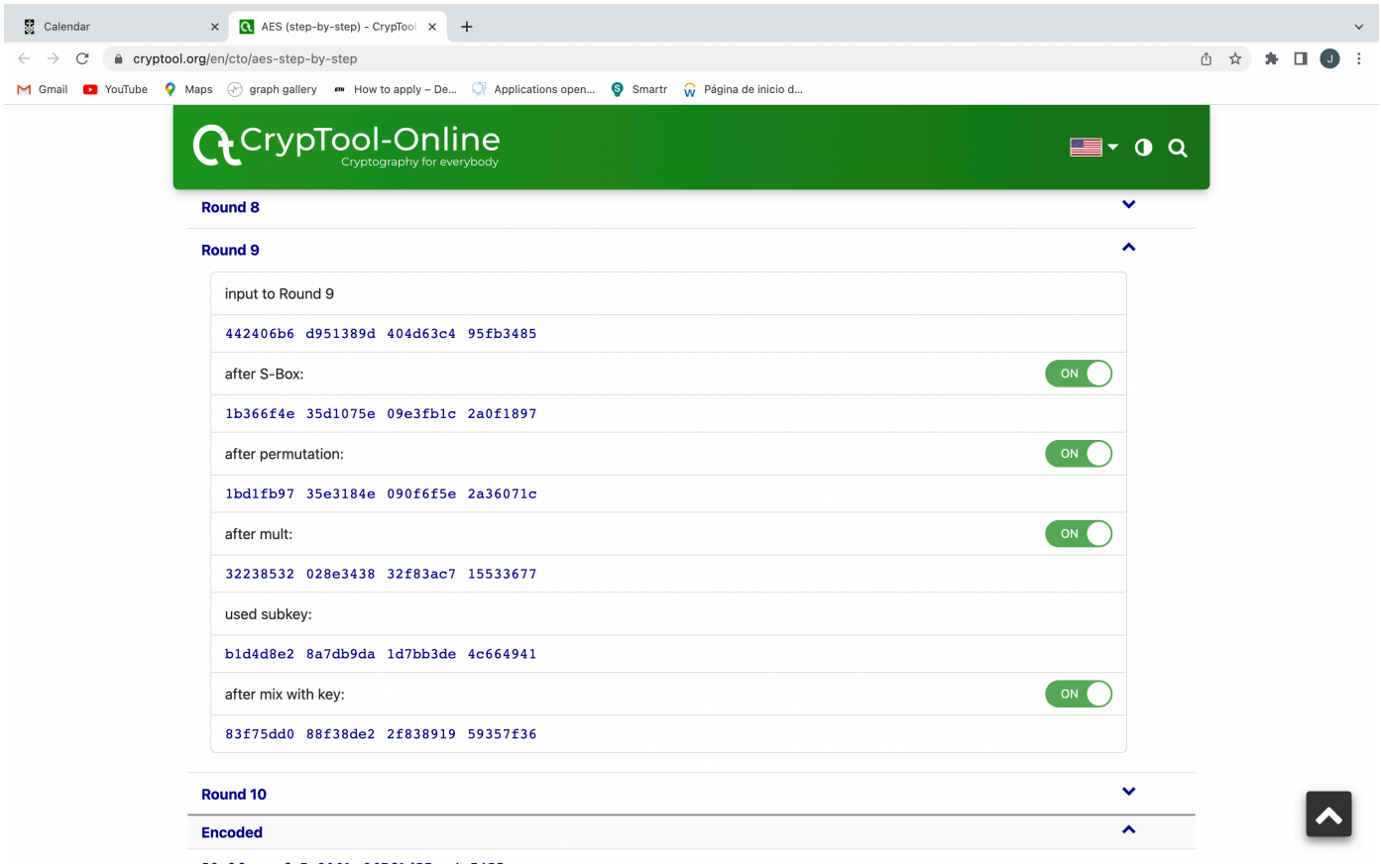
As we can see, there are **4 operations**. S-box, permutation, multiplication and addition (XOR) of the subkey.

It is clear that the S-box is constant during the substitution as every **00** is substituted by **63** and 01 is substituted by 7c.

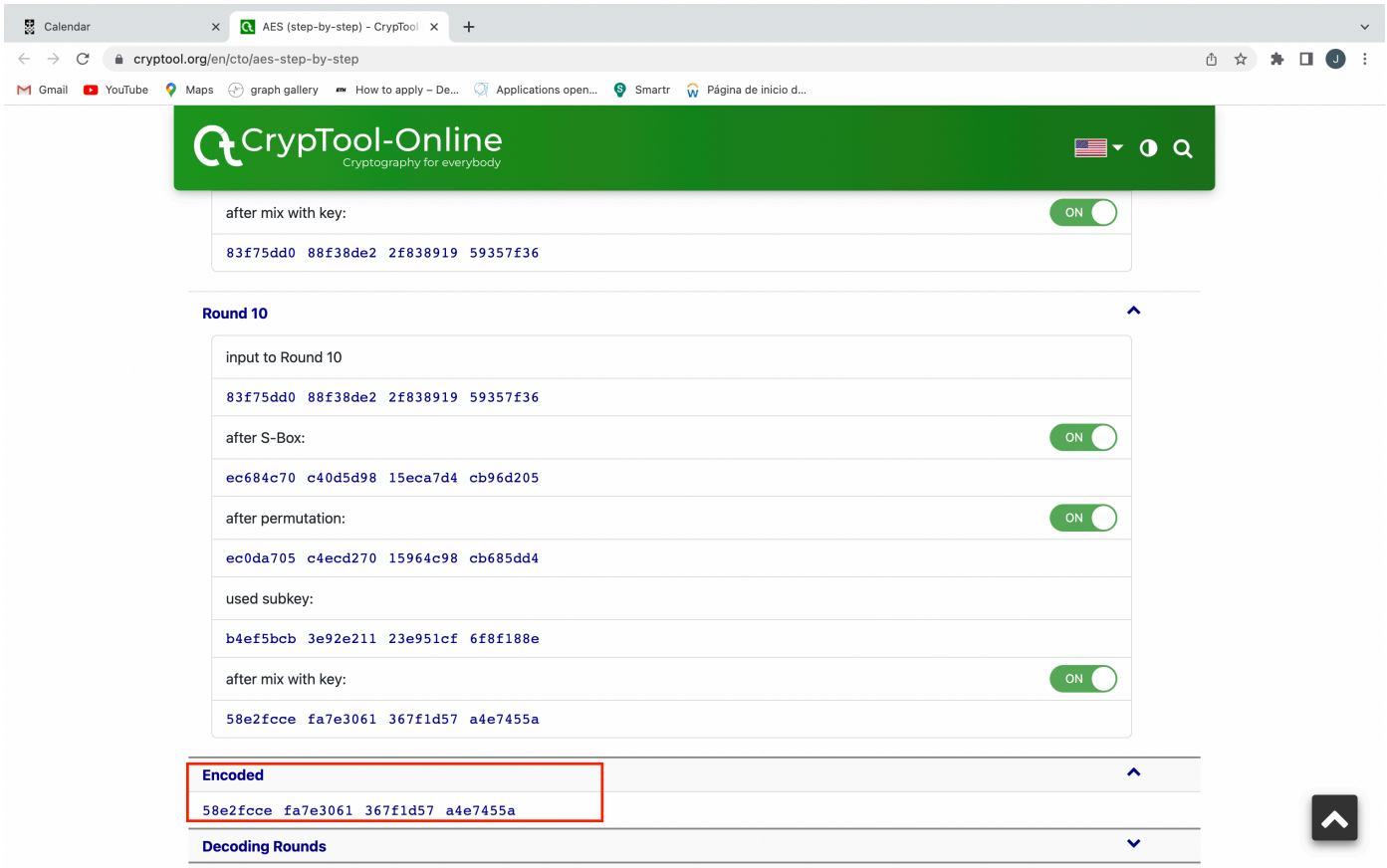
Then, the **permutation** is applied over the **rows** of the state matrix, even though every 32-bit word here visualized is a column of the matrix. As all the values are 63, apart from the one in the bottom left corner, all the rows stay constant but the last one. This one goes from **63 63 63 7c** to **7c 63 63 63.** This makes the first 32-bit word (or column) vary and go from **63 63 63 63** to **63 63 63 7c**.

Then a multiplication is performed (we will see this process in depth in the next section). And finally, the subkey is XORed column-wise completing the first round.

All the rounds have the same number of steps from **Round 1 to Round 9**:



On the other hand, **Round 10** has one step less has the mixing columns (multiplication) isn't performed:



The subkeys used in each round were obtained through expansion of the original key producing the expanded key:

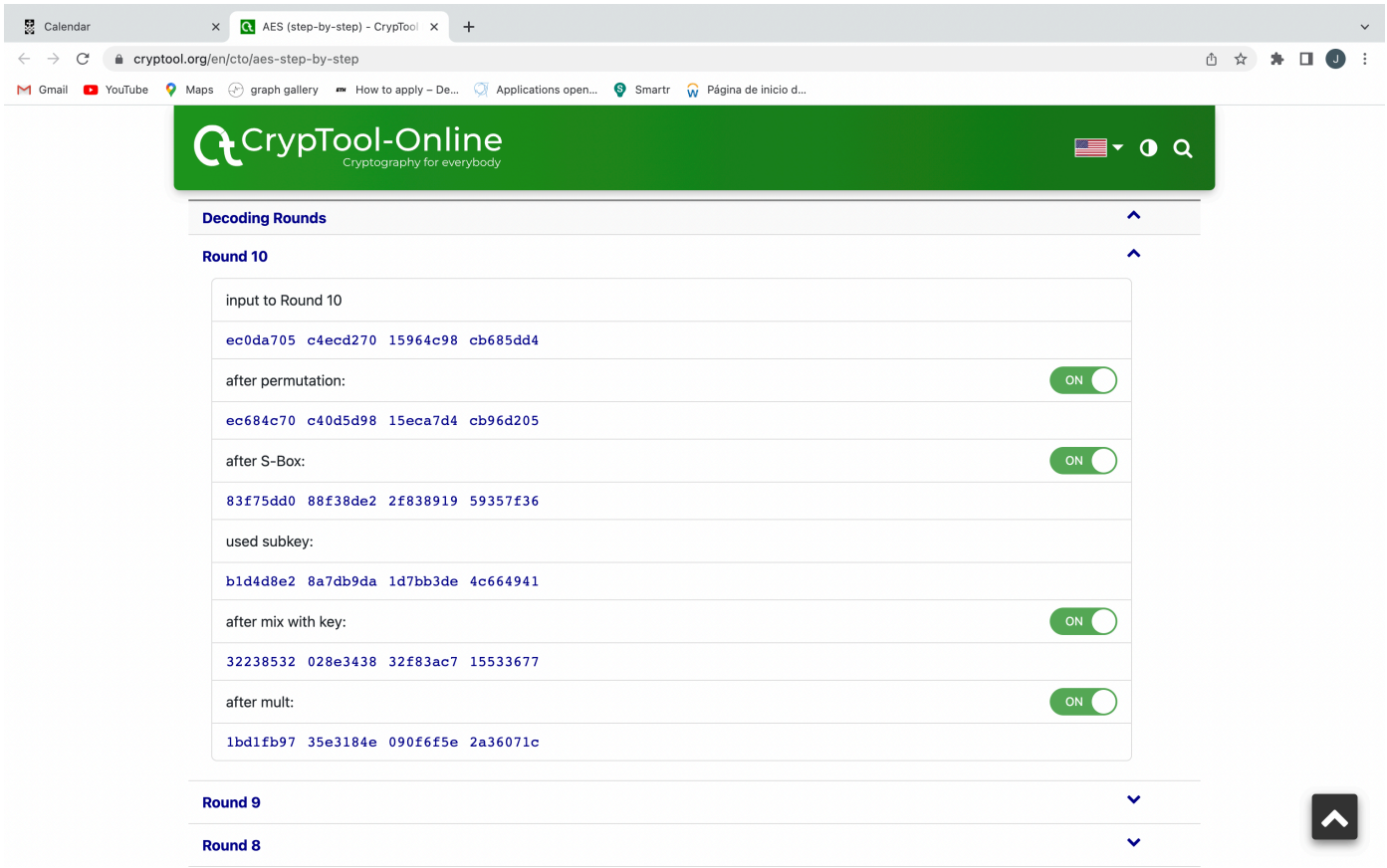
```
Expanded key: 00000000 00000000 00000000 00000000 62636363 62636363 62636363 62636363
9b9898c9 f9fbfbaa 9b9898c9 f9fbfbaa 90973450 696ccffa f2f45733 0b0fac99 ee06da7b
876a1581 759e42b2 7e91ee2b 7f2e2b88 f8443e09 8dda7cbb f34b9290 ec614b85 1425758c
99ff0937 6ab49ba7 21751787 3550620b acaf6b3c c61bf09b 0ef90333 3ba96138 97060a04
511dfa9f b1d4d8e2 8a7db9da 1d7bb3de 4c664941 b4ef5bcb 3e92e211 23e951cf 6f8f188e
```

As it is clear, the intial 128 bits stay constant and the rest are obtained through a sequence of operations that will be studied in depth in the following section.

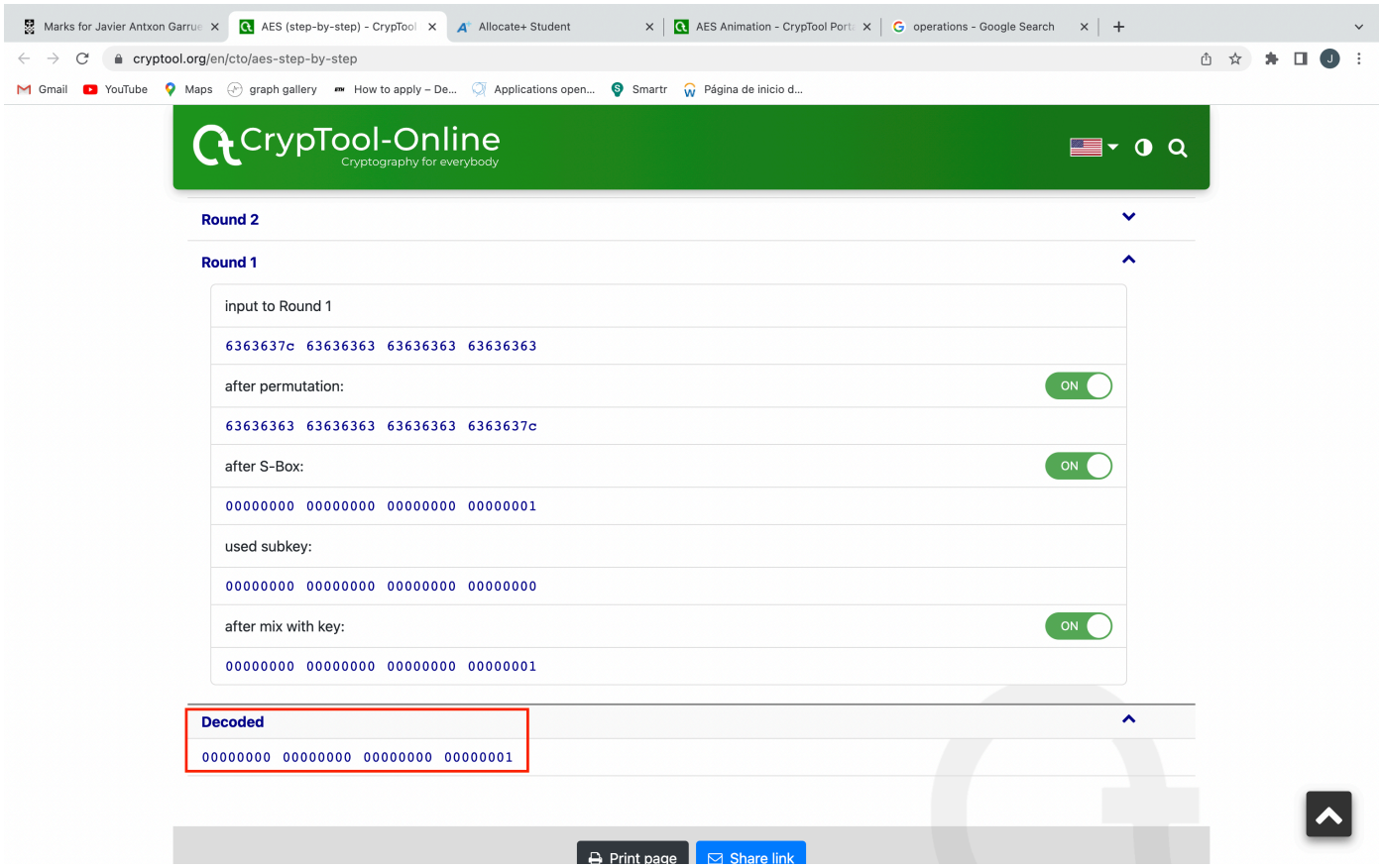
To conclude this part of the lab, we note that the **encoded** input **text** is:

```
Encoded text: 58e2fcce fa7e3061 367f1d57 a4e7455a
```

The decoding rounds operate similarly but in inverse order. Let's visualize one of them to understand this idea:



In this case, it will be **Round 1** the one that will have one operation less than the rest of rounds:



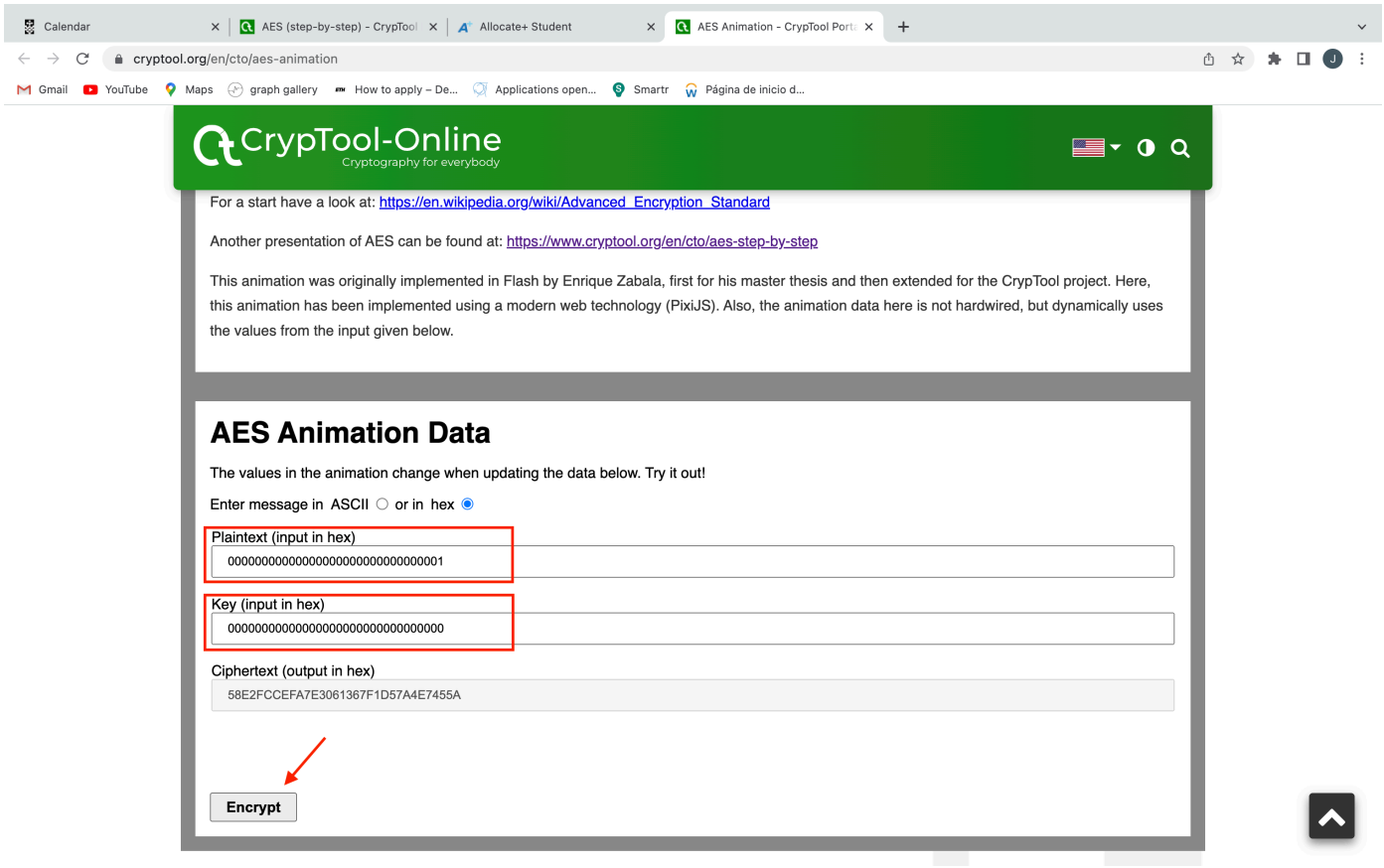
We can note that the **decoded text** is indeed the original **input text** of the encryption process.

Part Two: An AES encryption Visualisation

In this section, we will select the same input text and key used in the last part of the lab and visualize graphically the process by using the online website: https://www.cryptool.org/en/cto/aes-animation.

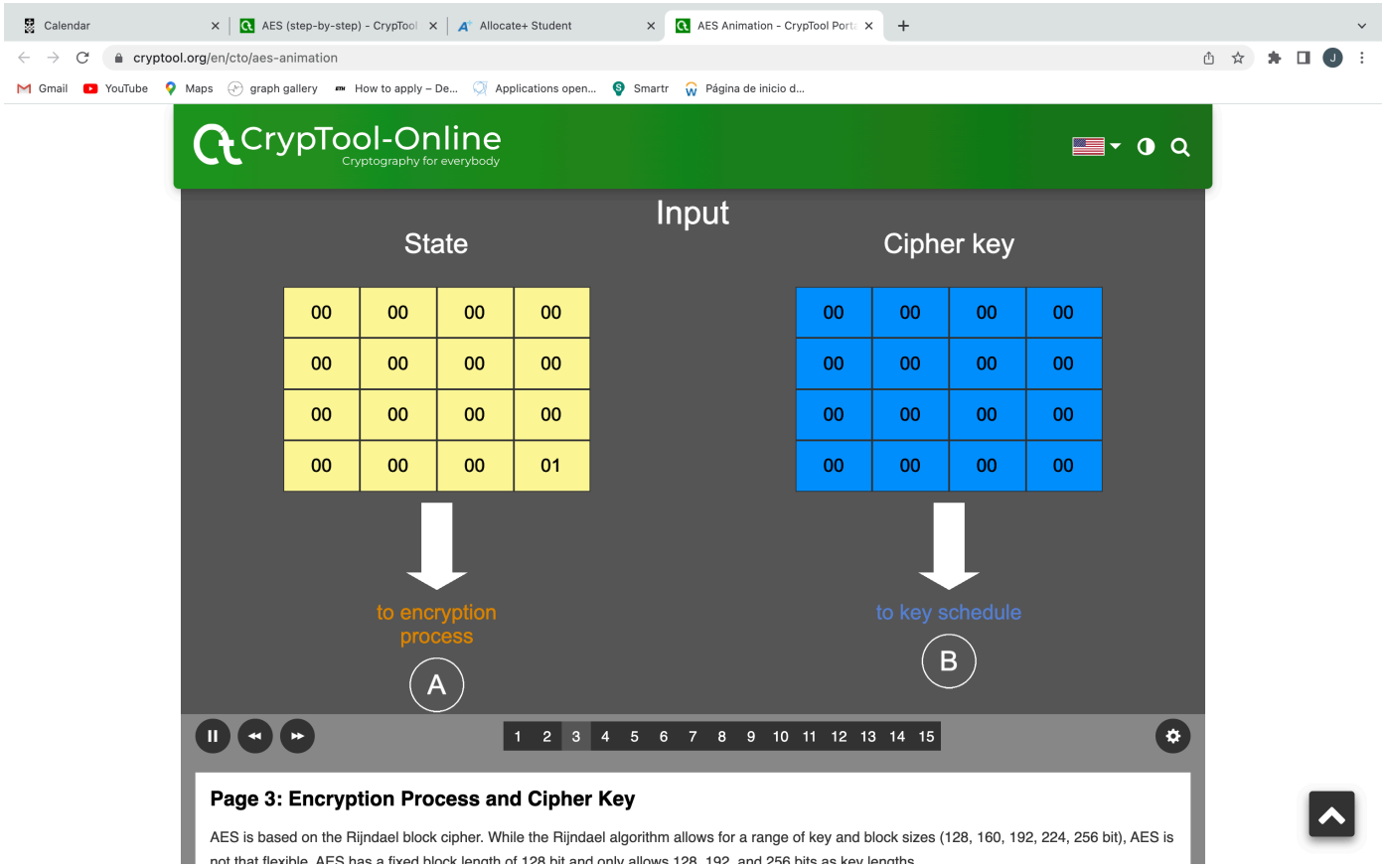
First, we type in the aforementioned input text and key:

```
Input text: 00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 01
Key: 00 00 00 00   00 00 00 00   00 00 00 00   00 00 00 00
```

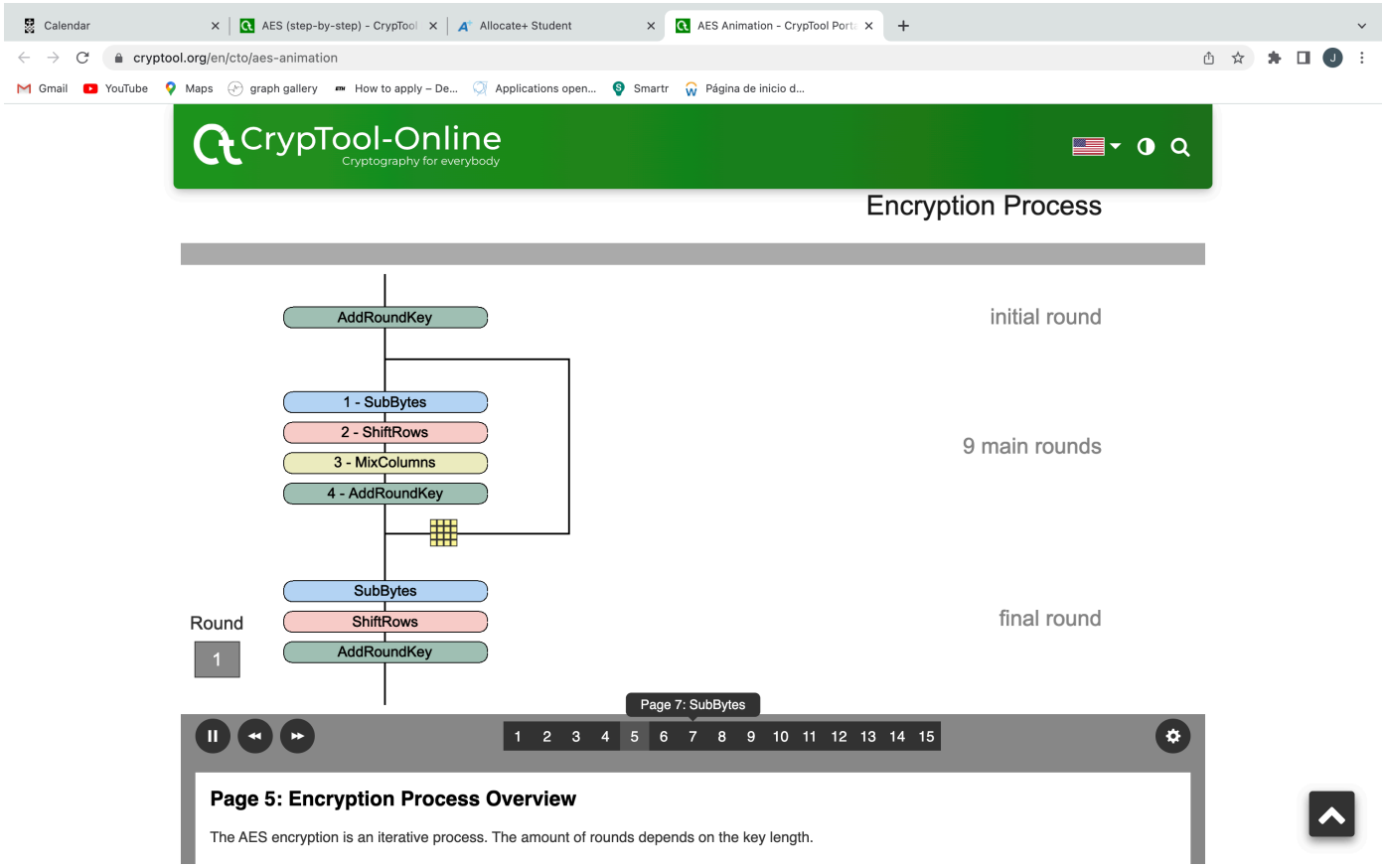


Then, we click on encrypt to obtain the ciphertext. Following this, we can proceed to visualize each of the steps and describe them briefly.

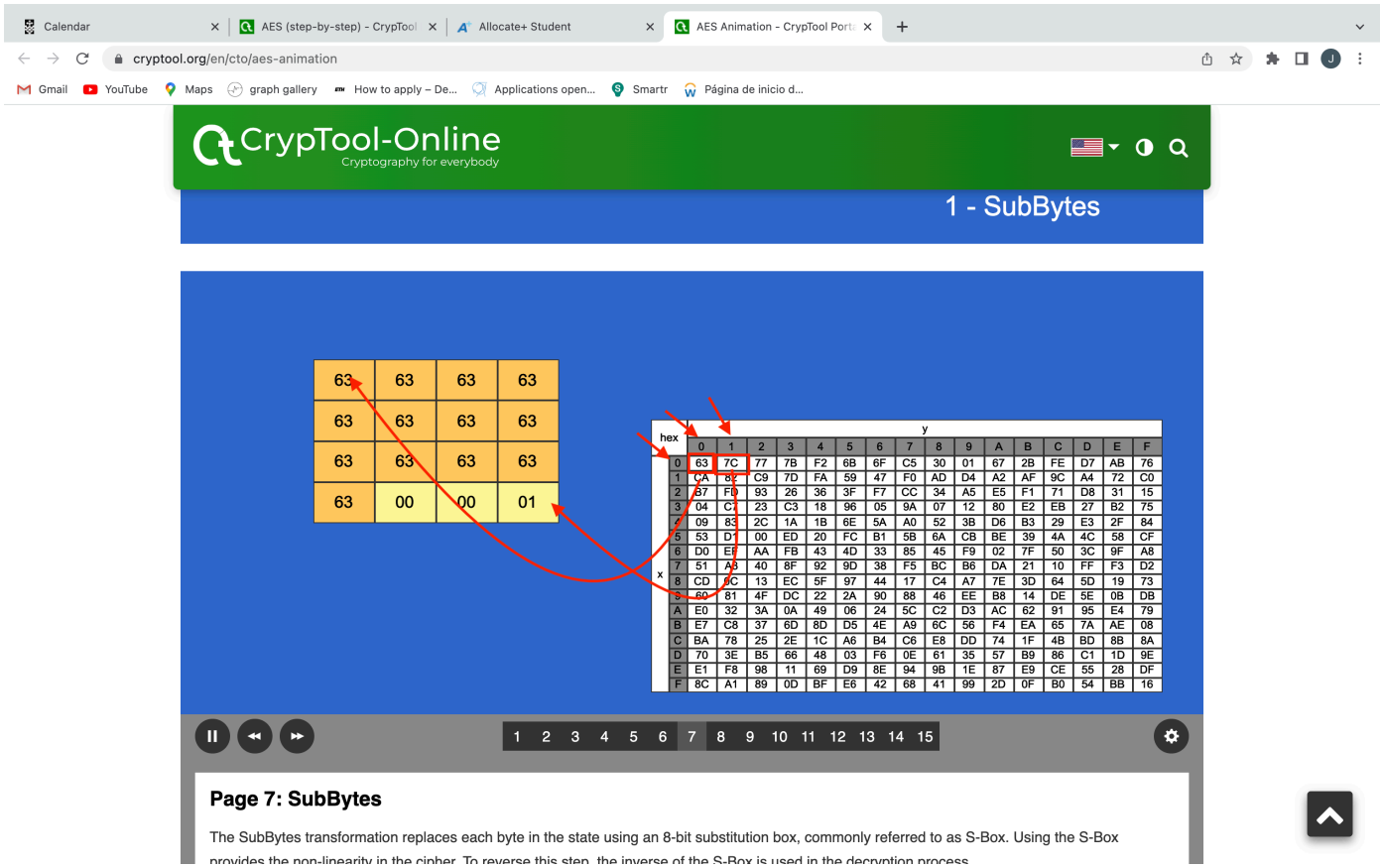
First, the two matrices are created with the given values. As we mentioned before, the 32-bit words are positioned as columns of the matrix.



Then, a schema of the process is shown explaining the number of rounds the matrix will undergo and pointing out the peculiarity of the final round.



Following this, the first substitution takes places making use of the created S-box:



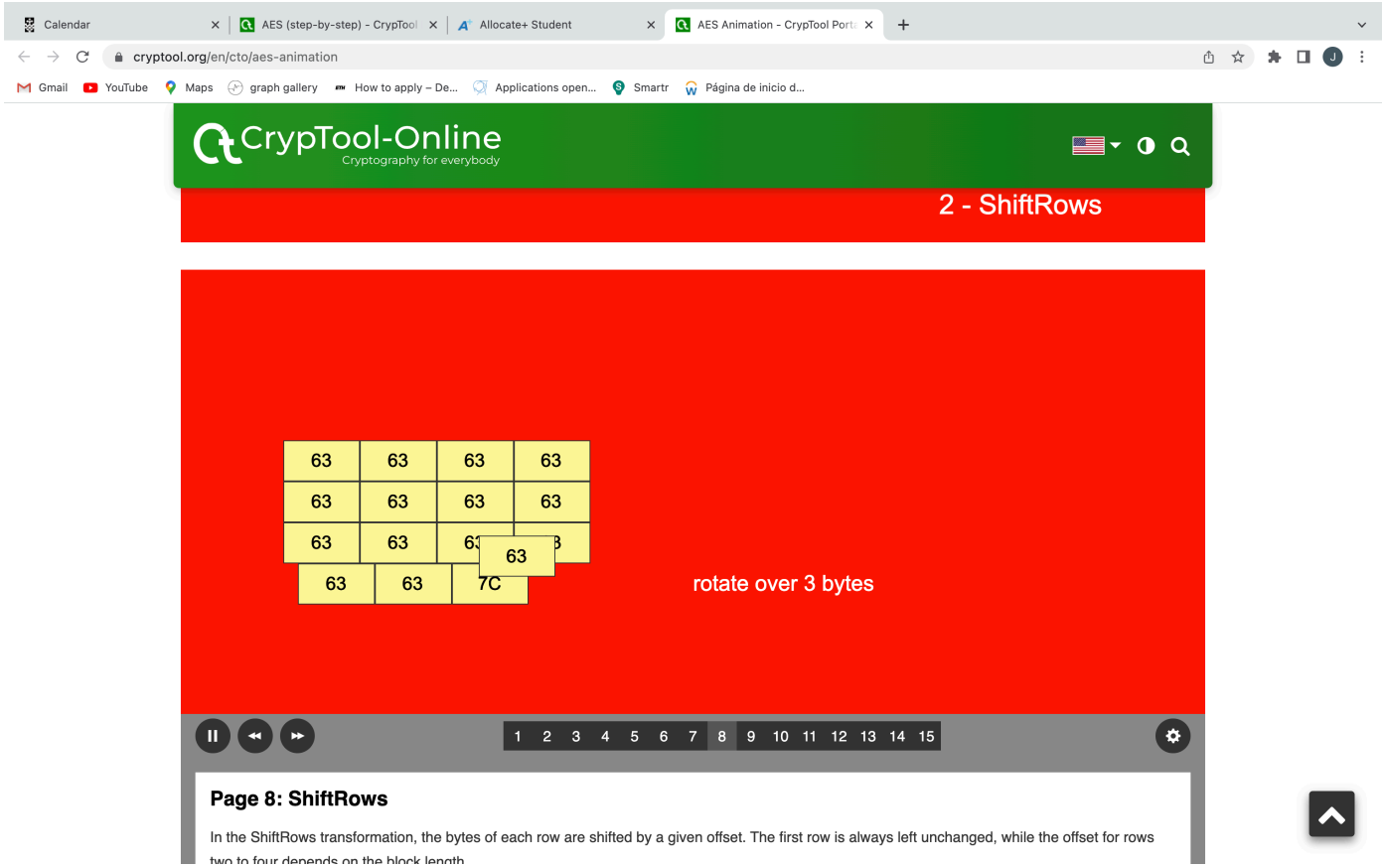
Then, the row-permutation is performed:

1st Row: no permutation

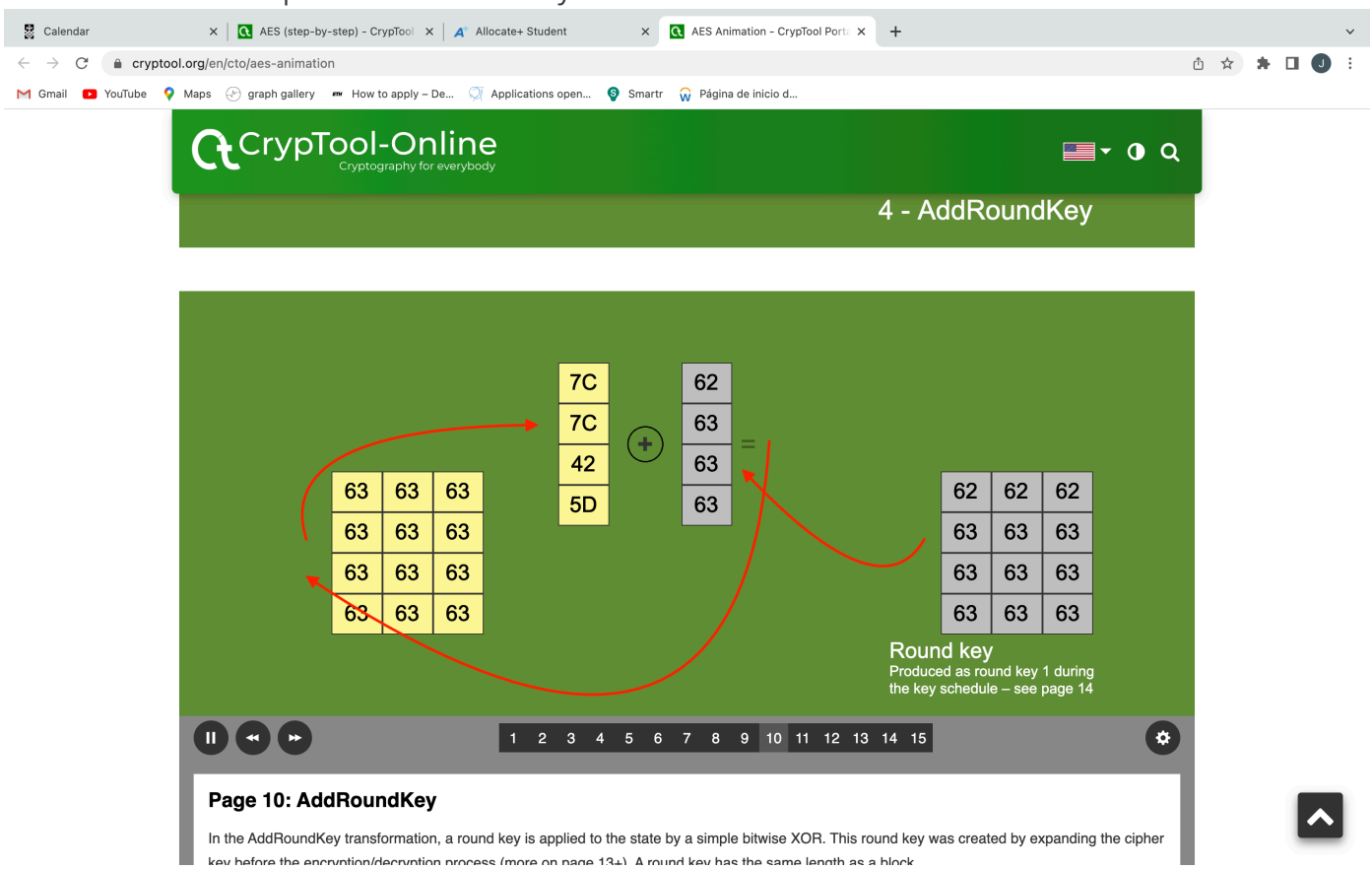
2nd Row: 1-shift (byte) to the left

3rd Row: 2-shift (byte) to the left

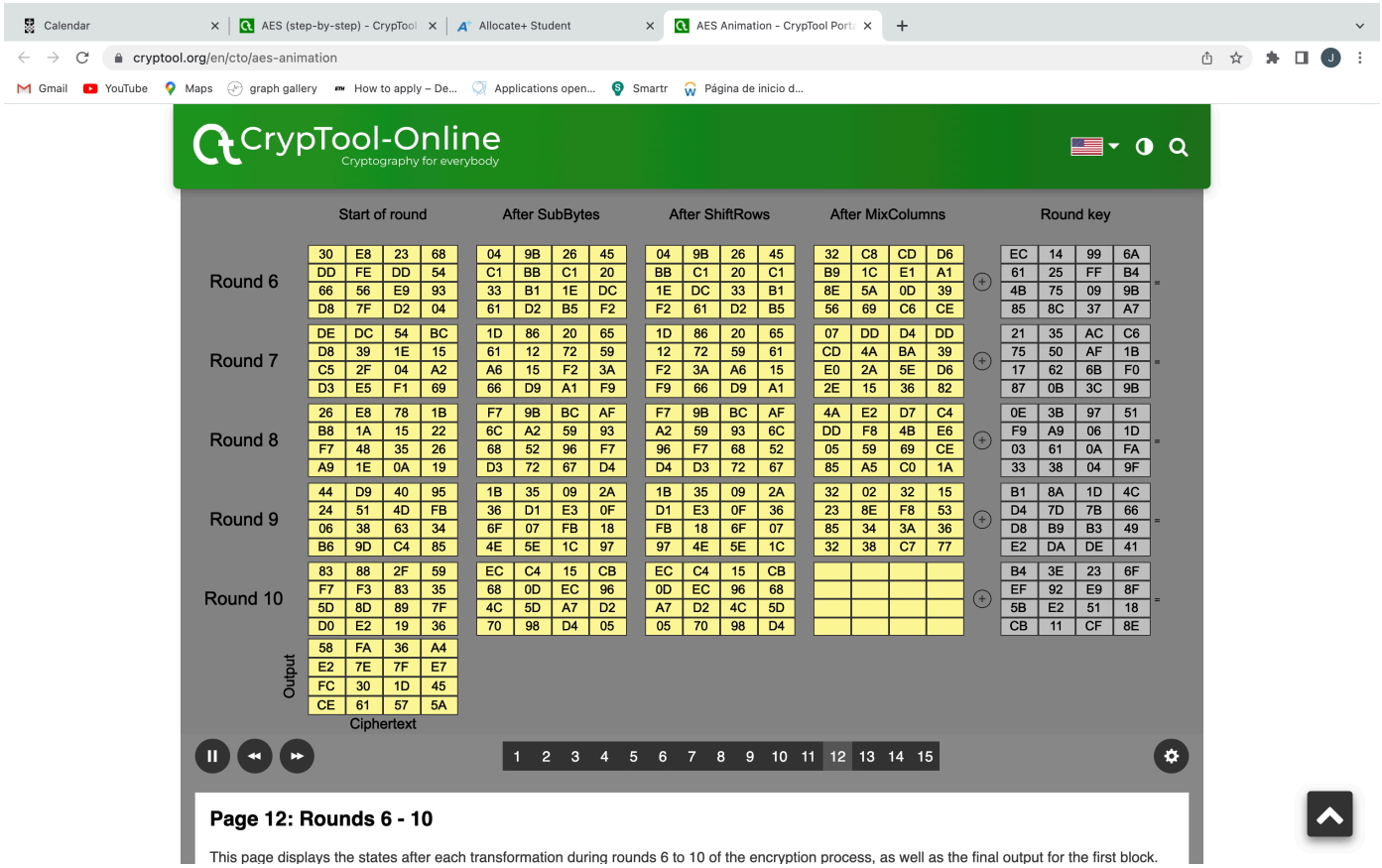
4th Row: 3-shift (byte) to the left



Once that is done, the last step is performed. This consists of XORing each column of the resulting state matrix with the correspondent round subkey:



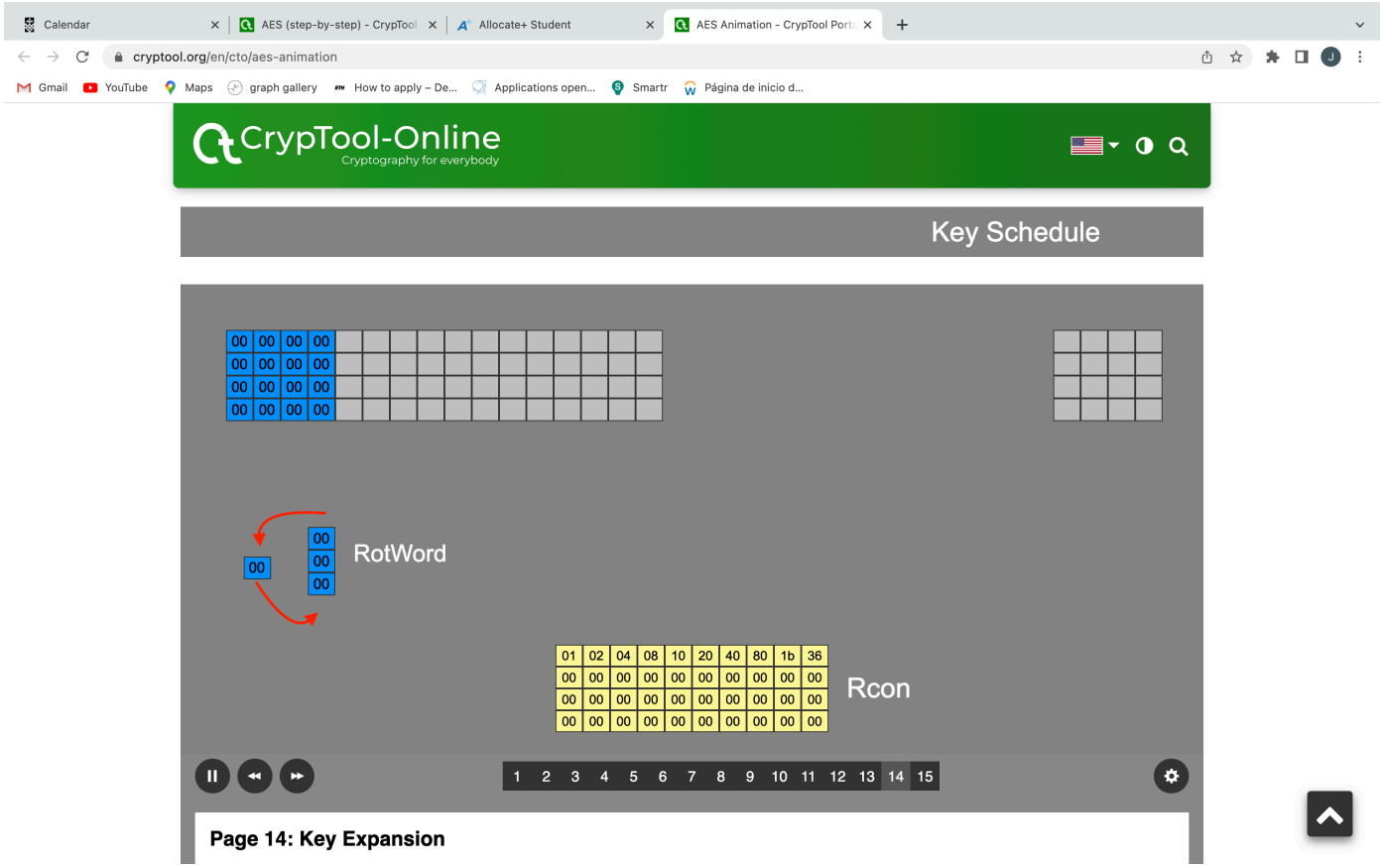
Overall transformation states of the matrix from **Round 6** to **Round 10**:



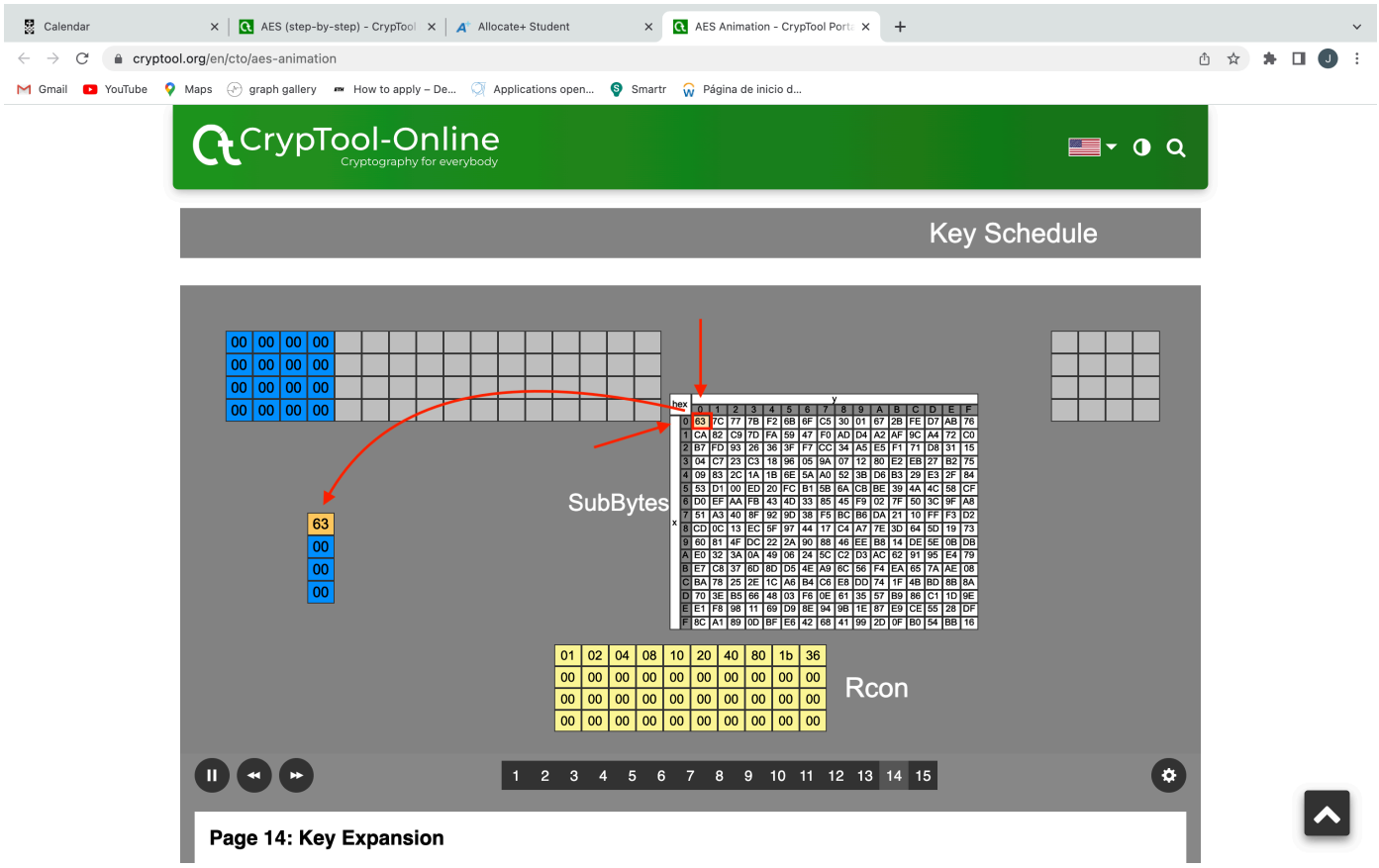
Note that the **MixColumns** transformation **isn't executed** in the **last round**.

To conclude the visualization, we are shown the procedure followed to obtain the different subkeys. This is called the "**Key Expansion**" or "**Key Schedule**". Let's see what is the process experience by one of the columns:

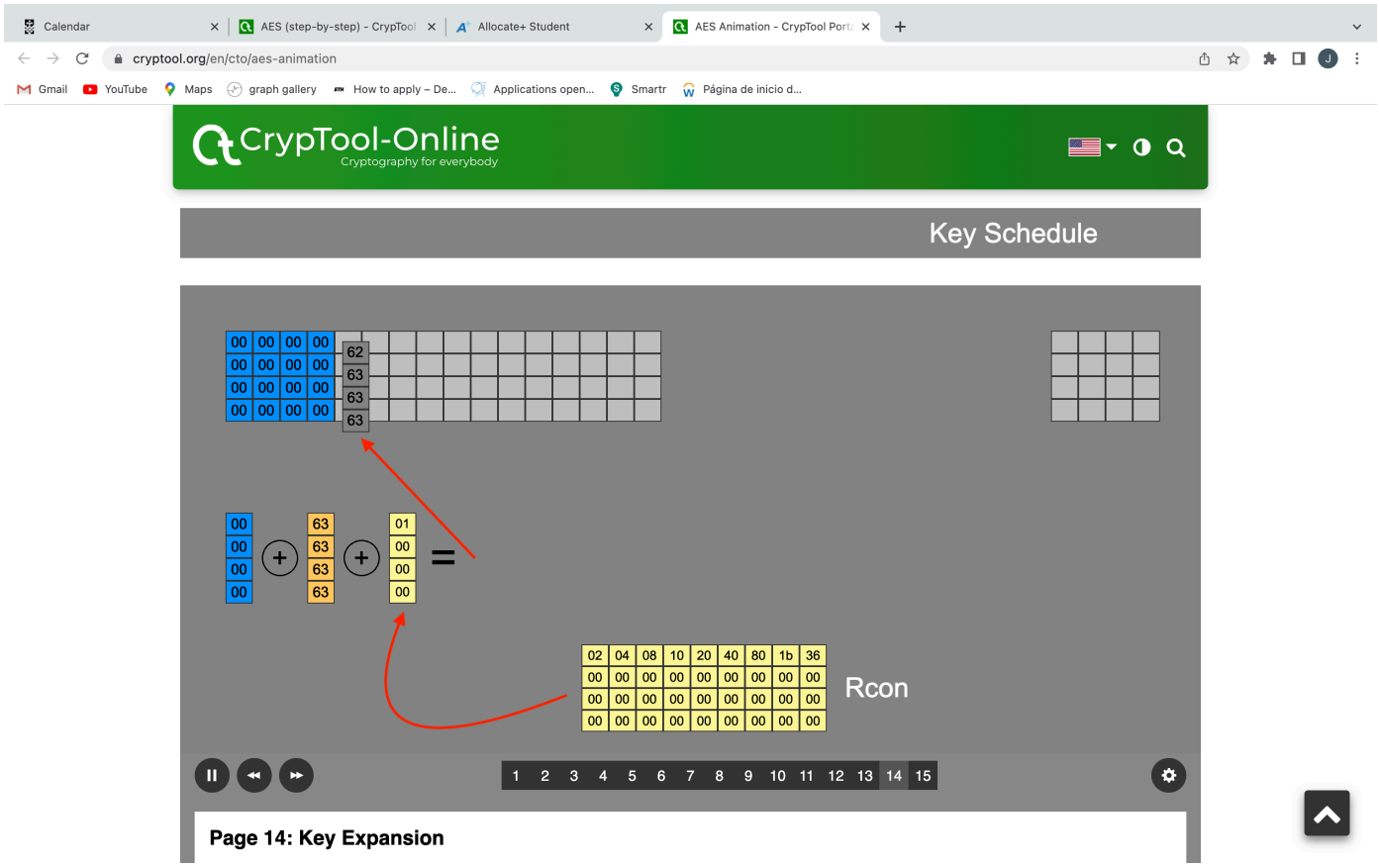
First, a rotation is performed. The first element (byte) becomes the last one:



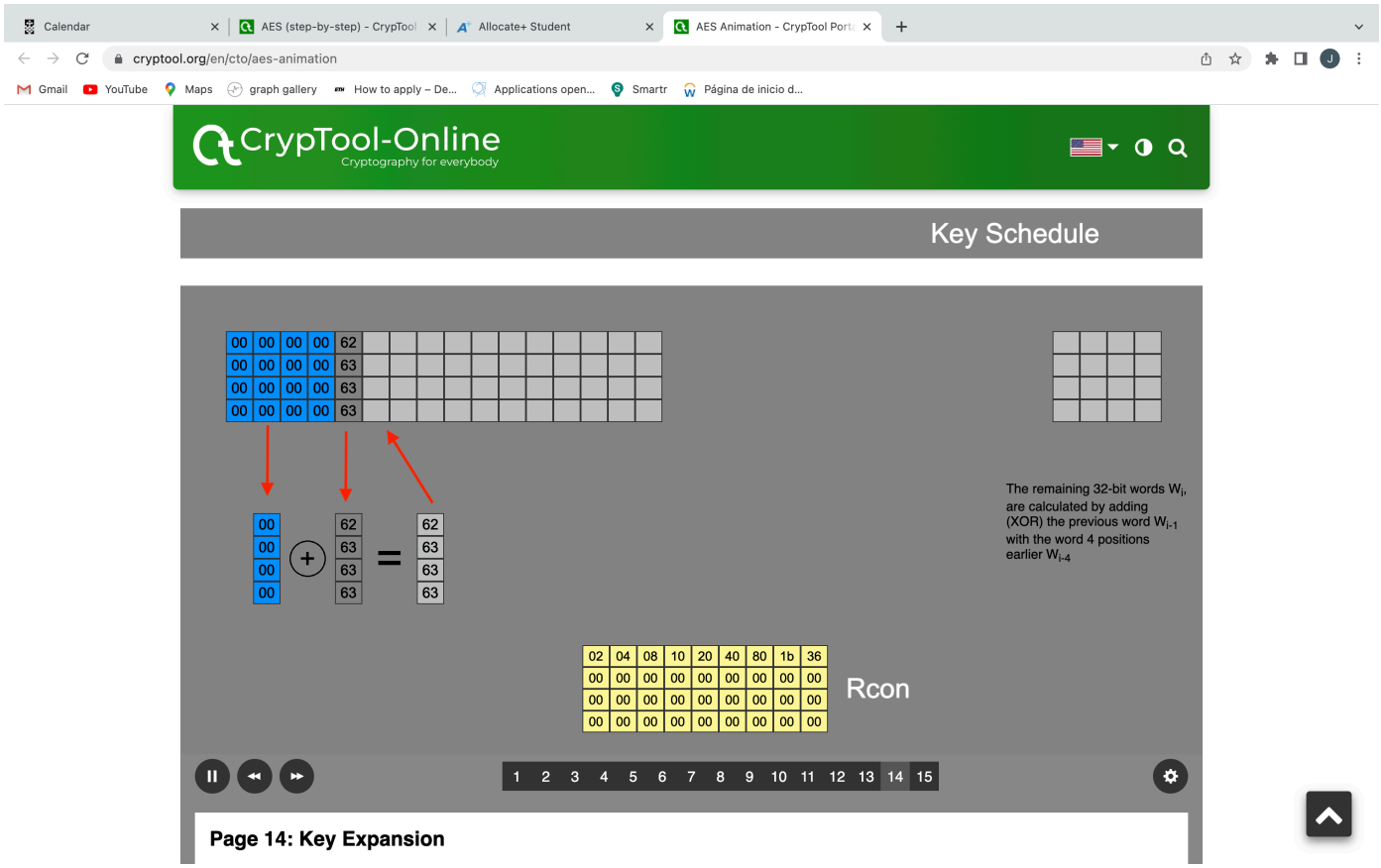
Then, every byte of the column is substituted using the S-box before shown:

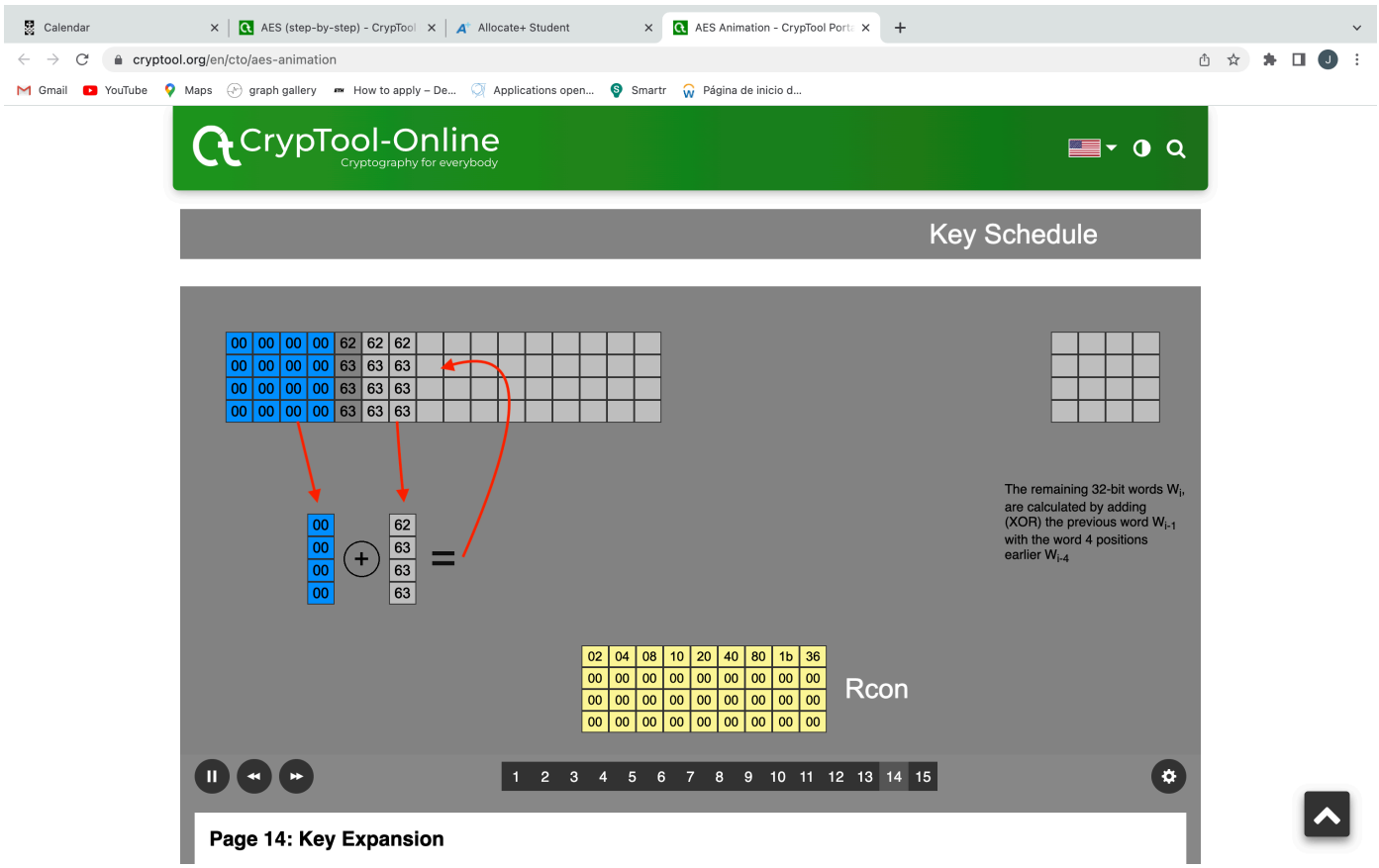


After that, the substituted column (**Xi**) is **XORed** with the (**Xi-4**) column and with the correspondent column of the **Rcon** matrix.



The last step is repeated until a subkey is completed (4 columns).

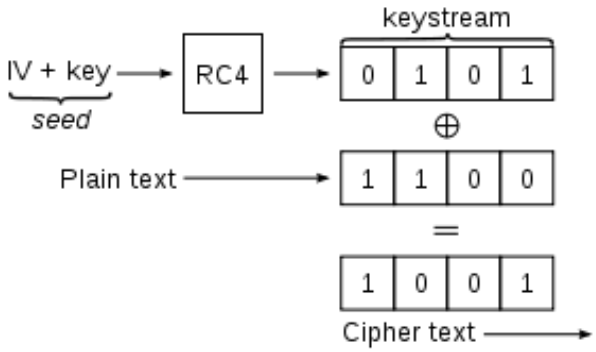




Then, the whole process begins from the beginning untill all the subkeys have been generated.

Part Three: Wi-Fi Authentication Crack

In this part of the lab we will use the tool **aircrack-ng** to crack WEP and WPA keys. WEP is much simpler than WPA. When using WEP, every packet is encrypted with RC4 using a specific key computed from a root key and a IV (Initialization Vector). The problem **doesn't rely** on the encrypted data being sent simultaneously with the **unencrypted IV** (this is a common practice), but in the short lenght of these IVs making repetition something common when sending lots of packets (apart from the poorness of the RC4 to generate the keystream). One could ask why would you then use IVs if they are not secret? The answer to this is to ensure that encrypting the same message twice yields two completely different ciphertexts. We could have a look at why having the same message encrypted with two different IVs can allow us to crack the password:



Suppose we have two messages P1 and P2 that have the same IV (as we saw this is possible due to the IVs being too short and inkecting requests in the network to generate thousands of packets). Then, these two P1 and P2 have the same seed (IV + key) and, therefore, the same key. Naming the keytream as K, we have that: P1 = M1 Xor K and P2 = M2 Xor K (being M1 and M2 the two plaintexts). At last we just need to compute P1 Xor P2 and we obtain (1):

P1 **Xor** P2 **=** (K **Xor** M1) **Xor** (K **Xor** M2) **=** (M1 **Xor** K) **Xor** (K **Xor** M2) **=** M1 **Xor** (K **Xor** K) **Xor** M2**=**M1 **Xor** M2

And from the two xored paintexts the keystream should be easily derived. Then a dictionary of IVs - Keystreams could be created by repeating this process several times. Once this is done, everytime a encrypted message is received the IV is extracted and checked at the dictionary. If there is an entry associated to it the message can be decrypted, if not the process done at (1) is repeated and the keystream is added to the dictionary.

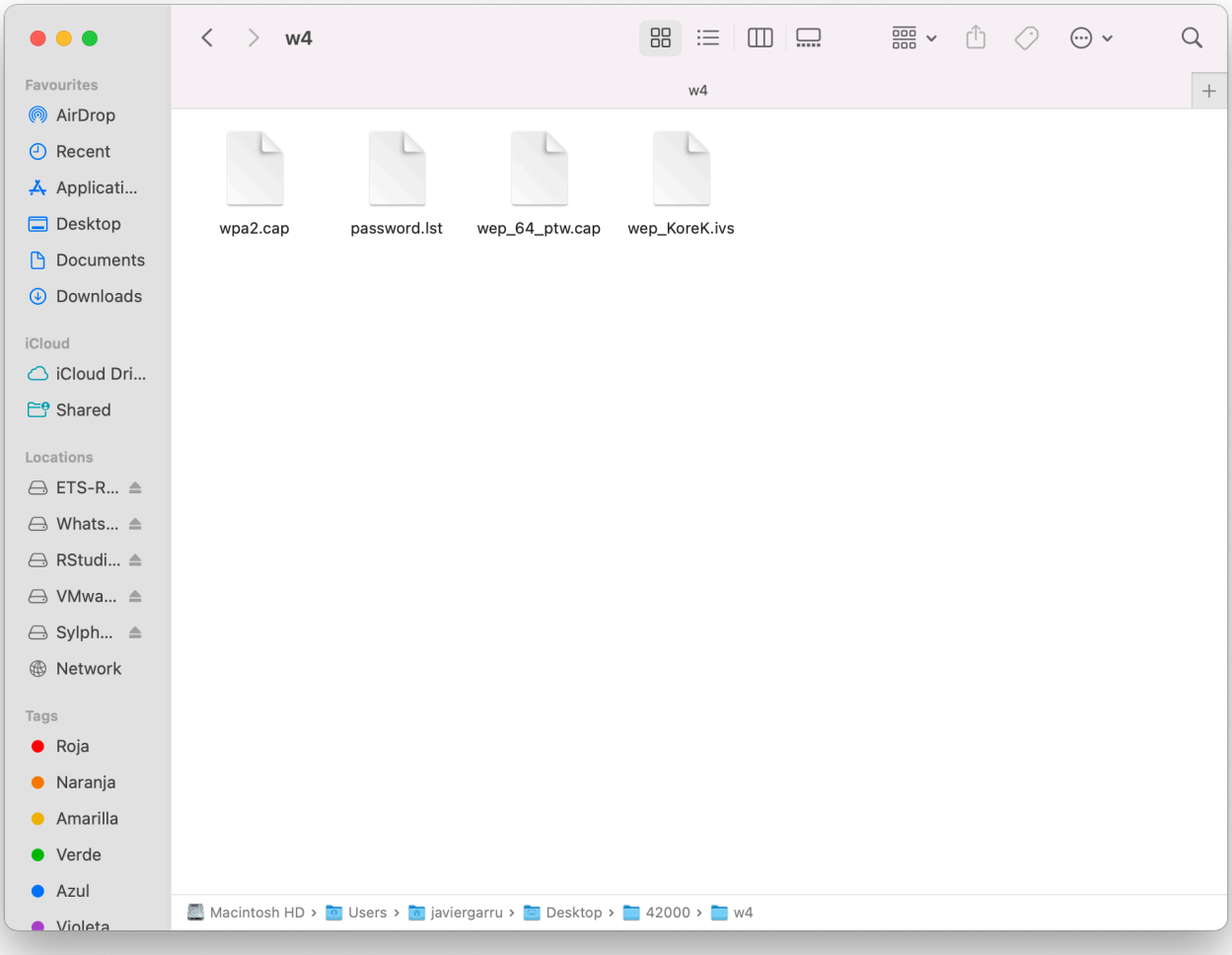
The two methods used in this lab are based on this structure and add brute force and statistical methods exploiting vulnerabilities of RC4 to obtain the Rootkey from a dictionary of IVs - keystreams.

Cracking a WEP key:

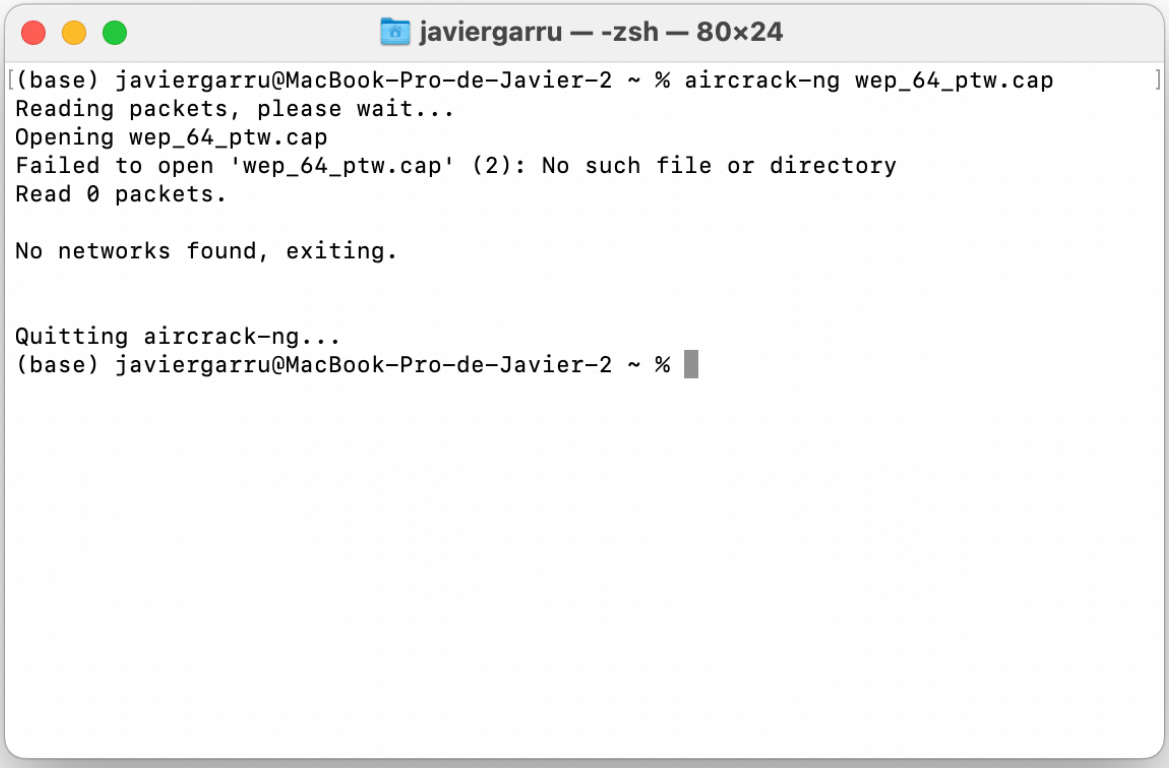
We will use **airodump-ng**. By capturing enough packets we are able to crack the password. The method used by default is called **PTW**, but there is another method called **FMS/KoreK** which we could use in case of not being able to crack the password with the first one. The PTW method is an extension of the FMS method with the constraint of only working with ARP requests. The **FMS/KoreK** is also based on the FMS but it's

older and requires **more packets** than PTW. After this short introduction, let's proceed with the cracking.

First, we will download the following documents from Canvas:



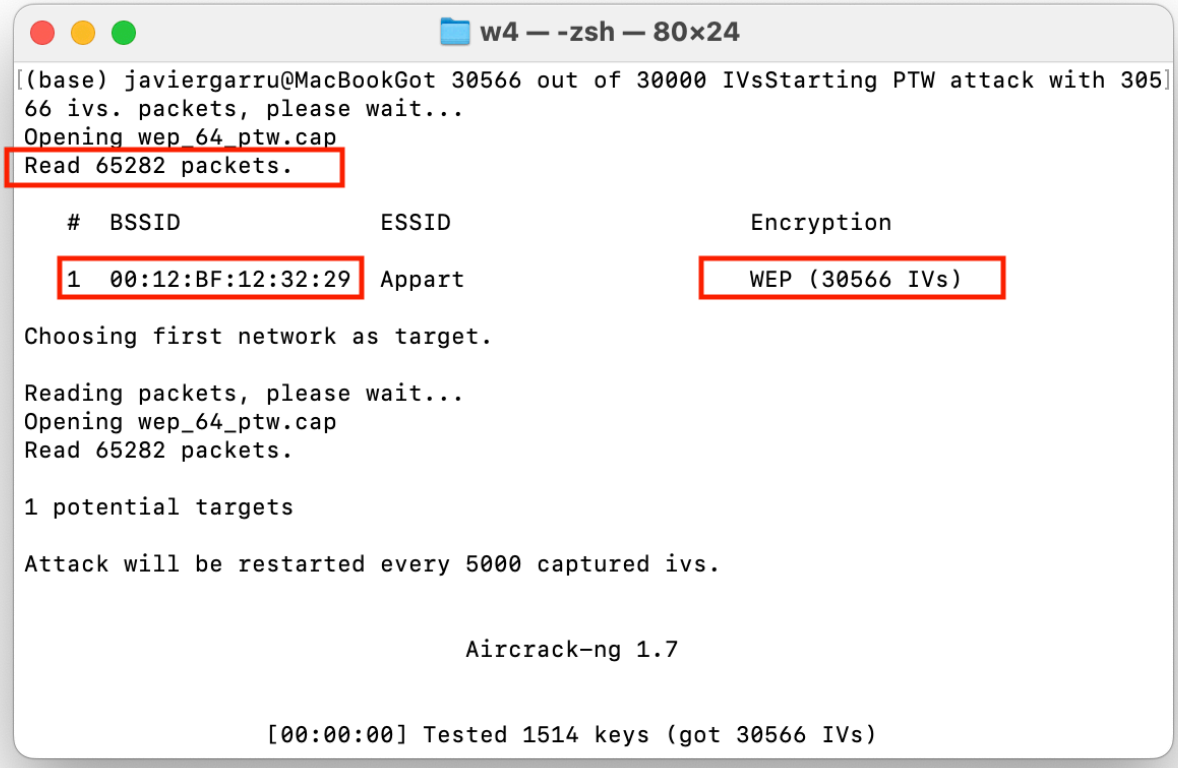
Then, we need to situate ourselves in the folder where we have imported the documents. If we wouldn't do this when executing the command we will see this error:

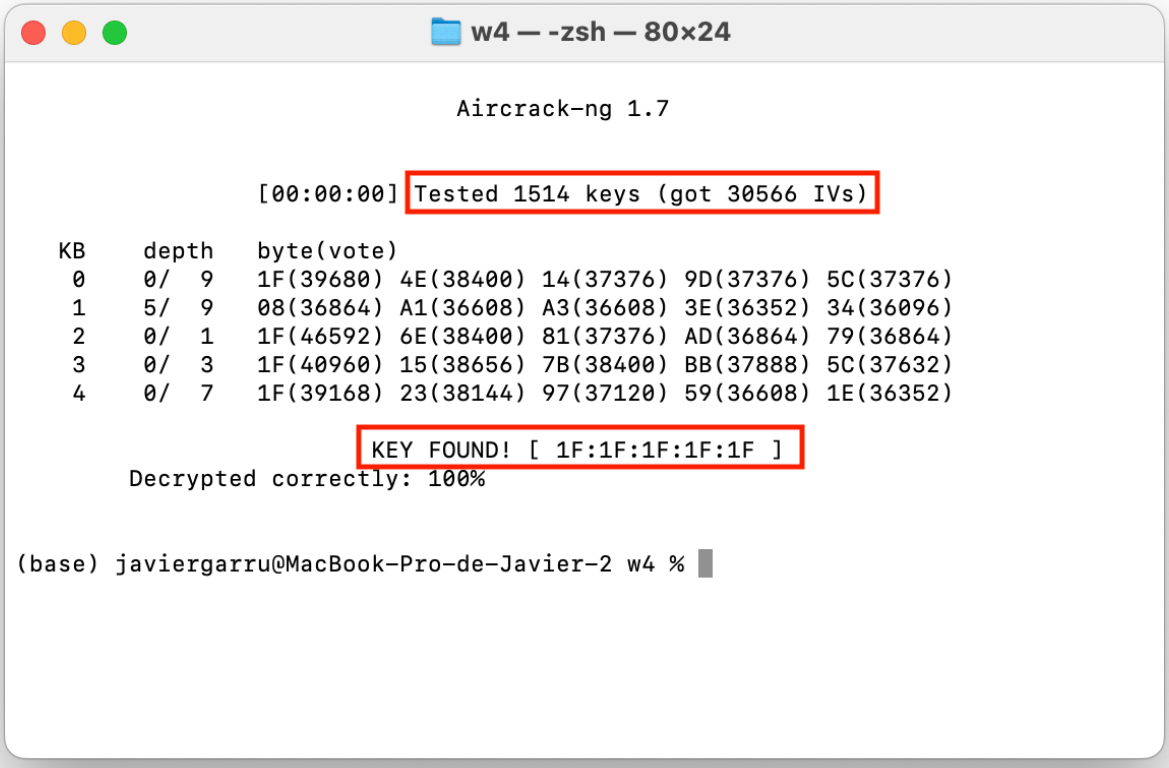


Once, in the correct folder we execute:

```
aircrack-ng wep_64_ptw.cap
```

This **wep_64_ptw.cap** is a packet capture file that contains a list of captured packets. In particular it contains **65282** packets. Once it has read the packets it finds the potential targets (or WEP protected wifi nets) receiving or sending the packets. Then, extracts the different IVs and proceeds to use brute force to try keys. This brute force attack takes place amongst a reduced number of possibilities due to a previous statistical work. We can have a look at the output of the execution:


w4 — -zsh — 80×24
[(base) javiergarru@MacBookGot 30566 out of 30000 IVsStarting PTW attack with 305
66 ivs. packets, please wait...
Opening wep_64_ptw.cap
Read 65282 packets.
BSSID ESSID Encryption
1 00:12:BF:12:32:29 Appart WEP (30566 IVs)
Choosing first network as target.
Reading packets, please wait...
Opening wep_64_ptw.cap
Read 65282 packets.
1 potential targets
Attack will be restarted every 5000 captured ivs.
Aircrack-ng 1.7
[00:00:00] Tested 1514 keys (got 30566 IVs)

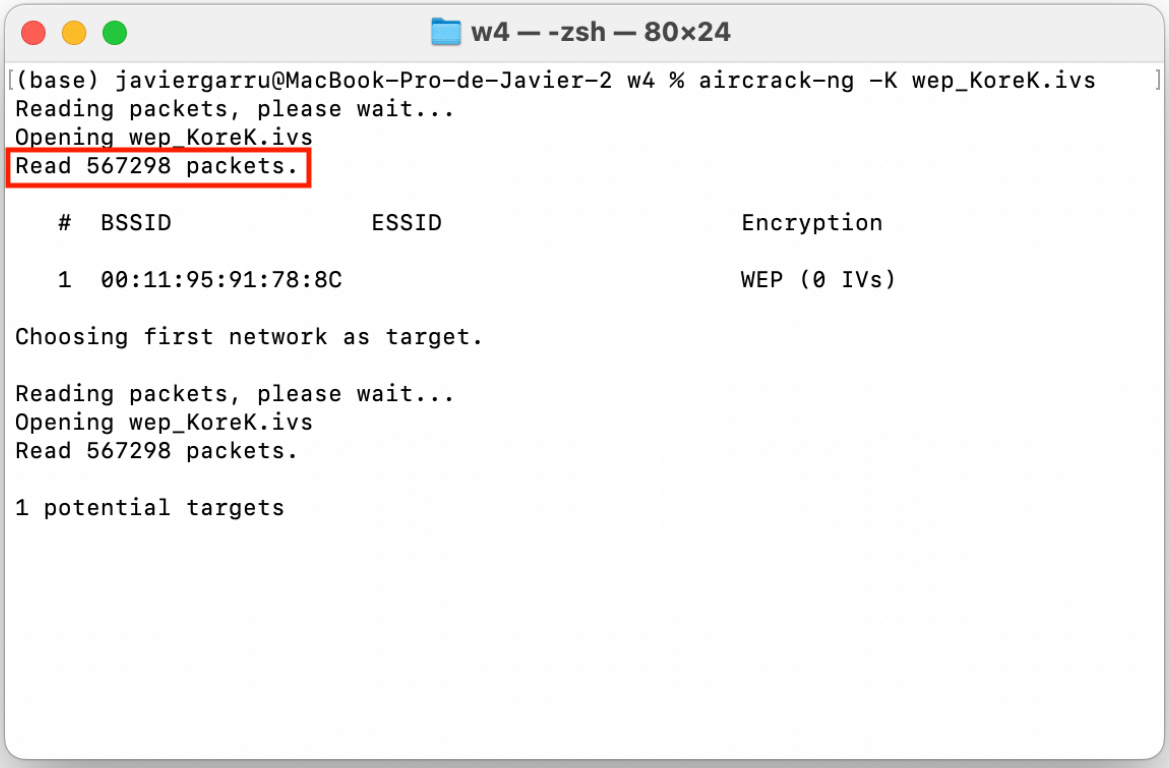

w4 — -zsh — 80×24
Aircrack-ng 1.7
[00:00:00] Tested 1514 keys (got 30566 IVs)
KB depth byte(vote)
0 0/ 9 1F(39680) 4E(38400) 14(37376) 9D(37376) 5C(37376)
1 5/ 9 08(36864) A1(36608) A3(36608) 3E(36352) 34(36096)
2 0/ 1 1F(46592) 6E(38400) 81(37376) AD(36864) 79(36864)
3 0/ 3 1F(40960) 15(38656) 7B(38400) BB(37888) 5C(37632)
4 0/ 7 1F(39168) 23(38144) 97(37120) 59(36608) 1E(36352)
KEY FOUND! [1F:1F:1F:1F:1F]
Decrypted correctly: 100%
(base) javiergarru@MacBook-Pro-de-Javier-2 w4 %

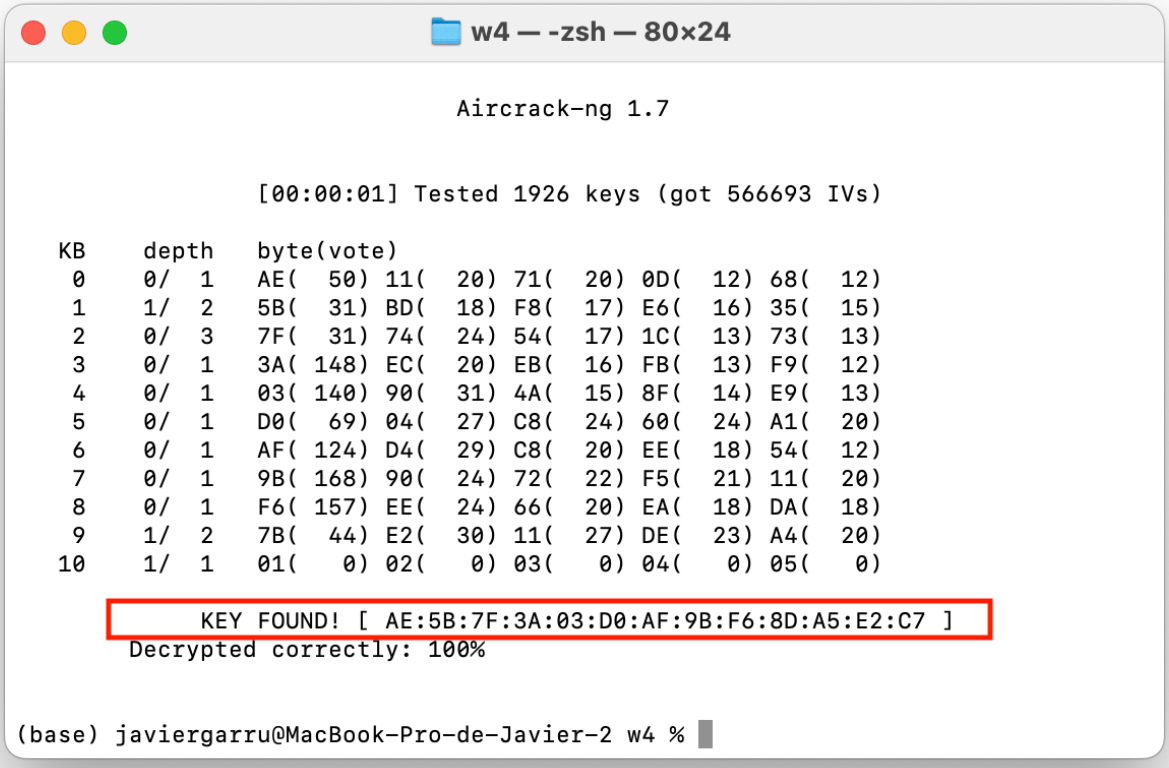
The found key is:

```
1F:1F:1F:1F:1F
```

We can now use the KoreK method. We will determine this by the **option -K** and we will also provide a file containing all the IVs:

```
aircrack-ng -K wep_KoreK.ivs
```





The found key is:

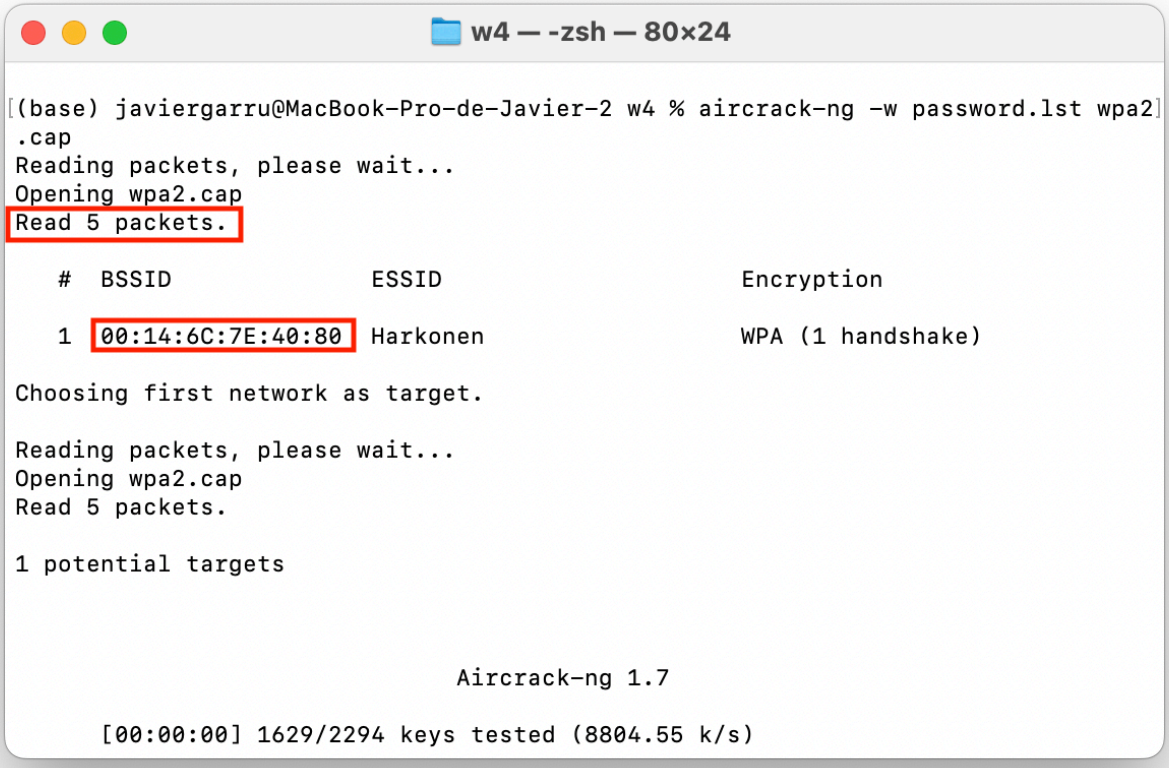
```
AE:5B:7F:3A:03:D0:AF:9B:F6:8D:A5:E2:C7
```

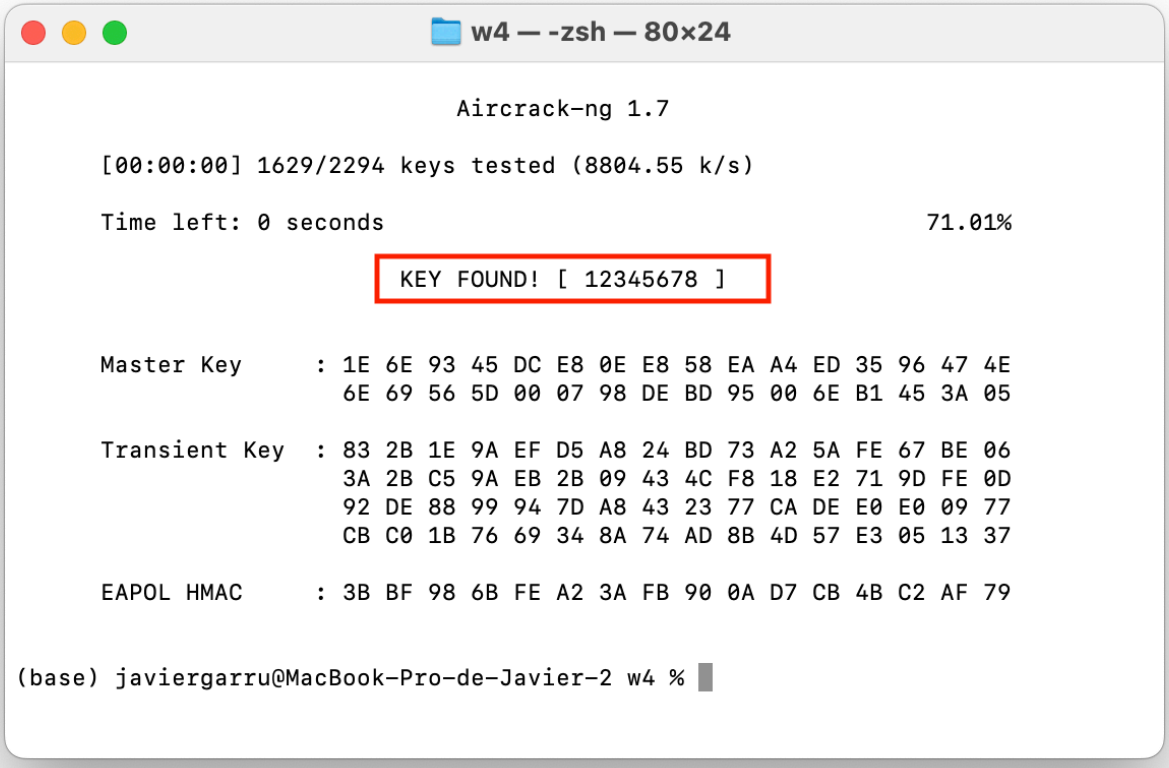
Cracking a WPA key:

The only type of WPA/WPA2 that aircrack can crack are the ones using pre-shared keys (PSK). WPA/WPA2 use IVs but the key is not static, then the colection of IVs becomes useless. Brute force would be an option if the password length would be known (and short) without capital letters, symbols or numbers for example.

In this case we will attack WPA by a dictionary attack. This is a list with potential passwords that will be tried. Again, a file containing packets (particularly now handshake packets) will be input in the aircrack tool. While on the last file used lots of packets were necessary, only 2 of the 4 handshake packets will be necessary to crack the password now:

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


w4 — -zsh — 80×24
(base) javiergarru@MacBook-Pro-de-Javier-2 w4 % aircrack-ng -w password.lst wpa2.cap
Reading packets, please wait...
Opening wpa2.cap
Read 5 packets.
BSSID ESSID Encryption
1 00:14:6C:7E:40:80 Harkonen WPA (1 handshake)
Choosing first network as target.
Reading packets, please wait...
Opening wpa2.cap
Read 5 packets.
1 potential targets
Aircrack-ng 1.7
[00:00:00] 1629/2294 keys tested (8804.55 k/s)


w4 — -zsh — 80×24
Aircrack-ng 1.7
[00:00:00] 1629/2294 keys tested (8804.55 k/s)
Time left: 0 seconds 71.01%
KEY FOUND! [12345678]
Master Key : 1E 6E 93 45 DC E8 0E E8 58 EA A4 ED 35 96 47 4E
6E 69 56 5D 00 07 98 DE BD 95 00 6E B1 45 3A 05
Transient Key : 83 2B 1E 9A EF D5 A8 24 BD 73 A2 5A FE 67 BE 06
3A 2B C5 9A EB 2B 09 43 4C F8 18 E2 71 9D FE 0D
92 DE 88 99 94 7D A8 43 23 77 CA DE E0 E0 09 77
CB C0 1B 76 69 34 8A 74 AD 8B 4D 57 E3 05 13 37
EAPOL HMAC : 3B BF 98 6B FE A2 3A FB 90 0A D7 CB 4B C2 AF 79
(base) javiergarru@MacBook-Pro-de-Javier-2 w4 %

The found key is:

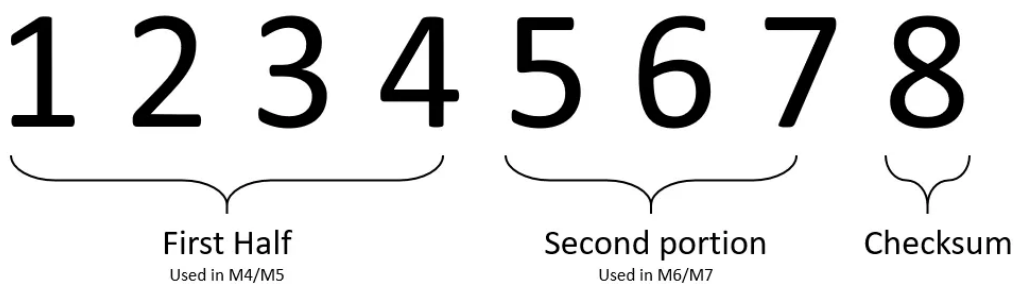
```
12345678
```

Questions:

Q1: What is the vulnerability of WPA2 Personal?

There are a few vulnerabilities in WPA2 Personal:

The first of these is based on the possibility of brute force attacking the **WPS Pin** (if WPS is enabled). WPS stands for WiFi Simple Config and was created to establish faster and easier connections between the router and the wireless devices. While WPS is usually used through pressing a physical button in the router and tapping on something or pressing a button in the wireless device, there is also a Pin associated to the protocol. This Pin is made of 8 numbers distributed in the following way:



And they are introduced in a sequential manner, meaning a brute force attack would only need to crack the first half and, once the AP would verify the first half it would request the second half, the following three digits as the eigth can be calculated.

The second of these vulnerabilities is called **KRACK- Key Reinstallation Attack**. The attack exploits the fact that when reconnecting to an AP the four-way handshake doesn't completely take place. In order to accelerate the reconnection, only the third packet of the handshake must me retransmitted and there is no control over the number of times this packets are resent when trying to reconnect to the user. The attack will work like this:

1. Attacker stops connection between user AP and User (this can be done through flooding for example).
2. User tries to reconnect to the AP and AP resends thrid packet of handshake.
3. Attacker impersonates AP and establishes connection with User. The attacker can provide internet connection through the clone AP to make the User believe everything is working as expected.
4. Attacker keeps sending the third packet of the handshake to the User and User keeps encrypting it and answering back,
5. Attacker uses the plain information and the encrypted packets associated to it to crack the encryption key.

One this is compromised the attacker can use software to sniff the communication of the user with the internet and use SSL strip to avoid the user visiting SSL encrypted webpages to steal data from the user.

The third and last of these vulnerabilities is the short **lenght of the keys** being used.

Q2: How does WPA3 solve WPA2 shortcomings?

There are several improvements:

1. Improved **key length**: WPA2 works with 64 and 128 bit keys while WPA3 works uses up to 192 bits.
2. Includes **mechanisms** to intrinsically **strengthen** *weak* **passwords** to avoid dictionary or brute force attacks.
3. **Automatically encrypts** data (even for open access APs) becoming more individualized that WPA2 as an intruder with the key cannot read the messages sent by other users.
4. Uses a **stronger handshake** protocol and uses **Simultaneous Authentication of Equals (SAE)** instead of Pre-shared keys (PSK).
5. **WPS Pins** are 384-bit hashes which are incredibly more secure than the original WPS Pins.

Q3: Is there any possible attacks against WPA3?

The biggest attack that has been found to be succesful against WPA3 is a **timing-based side-channel attack**. This kind of attack is based on the time an encryption algorithm or protocol takes to encrypt different kinds of texts or packets. Through analysis and statistical studies of this different periods of times and considering the encryption algorithm being used the key could be cracked.

One vulnerability that has been found in WPA3 is the possiblility to **downgrade** some devices using this protocol to its predecesor (WPA2) and then perform a classic KRACK attack. After downgrading to WPA2 the old handshake would take place when reconnecting to the AP and the classic attack could be performed.

As any other protocol, **classical vulnerabilities** such as bad configuration and use of poor or weak passwords are still present.

Q1: How to capture the four-way handshake?

To capture the four-way handshake we will have to follow these steps from **Linux** (Mac OS would be different due to airmon-ng and similar others not compatible):

1. Start the monitor mode on wireless interface.

```
airmon-ng start wlan0
```

2. Note the nearest WiFi networks.

```
airodump-ng mon0
```

3. Note the channel of the network we wish to attack (target network) and dump all the packets that will be transmitted through that channel to a capture local file.

```
airodump-ng --channel CHANNEL --write local_capture_file mon0
```

4. Deauthenticate all the users (or a specific user if we know its MAC) from the WiFi:

```
aireplay-ng --deauth 0 -a <AP\_MAC> -c <CLIENT\_MAC> mon0
```

Once this is done, we just need to wait until one of the deauthenticated users makes the four-way handshake with the WiFi again and we have captured it.

Another option to do this, an as the provided documentation indicates, would be by using Wireshark. On wireshark we would only need to set up the monitoring mode, connect to the access point and the filter by *"eapol"*. Once this was done we would only need to save the capture packets in a file and we would have completed the task.

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

The vulnerabilities used in this cracking are two: The relatively **short length** of the **IVs** being used and the **poorness** on the use of the **RC4 algorithm**:

1. To exploit the first vulnerability, a replay attack takes places to generate as many packets, and therefore as many IVs, as possible. This leads to a repetition of IV's enabling attackers to learn about the root key.
2. The second vulnerability is found in the way the keystreams are generated and is used to reduce the possibilities through statistical study of the IV's and/or their encrypted messages allowing brute force to work in a reduced possibility-range.

The short length of the key being used and the poor Integrity Check Value algorithm potentiate the negative effects of the above vulnerabilities.

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

In this cracking the fact that everytime a connection to a WPA2 WiFi is established a **four-way handshake** takes place is used to perform the attack. Once **two** of these **packets** are **captured** (2 and 3 or 3 and 4) the tool can begin the **attack** by using a **dictionary**. The second **vulnerability** used doesn't belong to the protocol by itself, but the **simplicity** of the **key** being used increases the chances of it being in the dictionary (password.lst) and being cracked quickly.

Q4: What have you learnt in this lab? (Lab summary)

In this lab I have learnt how the AES algorithm really works, understanding the process graphically. In addition to this, I have understood the vulnerabilities associated to the use of short IVs when working with an encryption algorithm. I have also learnt why shouldn't we choose a simple password for any of our accounts or devices as they could be easily cracked with the use of a dictionary. Something that I wasn't expecting to discover is the vulnerabilities associated to leaving the WPS option enabled on our routers and how WPA3 addresses WPA2's vulnerabilities but fails to become completely secure.