NETWORK SECURITY HW3

GUR TELEM 206631848

1. Question - DNS Spoofing and MITM

See attached python file (`attacker.py`)

2. Question - AES, Modes of Operation

2.1. **A student changed the AES and replaced MC with rotate clockwise. How does the new and modified AES compare in resiliency to the original?** The new AES is much less resilient than the original one (it's likely because it was a student who made it).

Since in the new AES the MC is missing, each byte in the cipher corresponds to one specific byte in the plaintext and is completely independent of other bytes in the plaintext.

For each byte in the plaintext, the place of the corresponding byte in the cipher is deterministic and independent of the key.

This means that once we sent a plaintext with the i-th byte as x, and got that the corresponding byte in the cipher is y, we know that whenever we see a cipher with y in that spot, it corresponds to x in the place we tried in the plaintext.

An attack on the new AES can be done as follows:

Pre-attack preparations: Map each location of byte in a plaintext to the location of the corresponding byte in the cipher.

Let's call that map the locations map.

The attack:

(1) Send 256 plaintext messages to be encrypted. For each location in the plaintext, the byte there needs to be different in all of the messages. For simplicity the messages can be: `00 00 ... 00`, `01 01 ... 01`, ...
(2) For message `XY XY ... XY`, map XY to the resulting cipher from the message. Let's call that map key map.

Now that we have the two maps (key map per key and locations map for **all** keys).

When enchanting a wild cipher (see illustration below), we can decrypt it like so:

(1) For each byte ZK in the cipher, check in the locations map its original location.
(2) For the same ZK byte, also check which byte XY is mapped to it in ZK's location in the cipher under the key map.

Example 1. Example:

When sending the plaintext messages, the key map is:

00:

01	02	ff	f1
f4	1d	a6	65
00	43	3b	cc
a7	8a	95	42

, 01:

ff	01	02	f1
3b	00	43	cc
95	a7	8a	42
a6	f4	1d	65

, ...

And now we got an encrypted message:

cipher:

01	02	ff	f1
3b	00	43	cc
00	43	3b	cc
a6	f4	1d	65

Now we go over the bytes in the cipher: `01 02 ff f1 ...` and we can reverse map the original location using the locations map. For the plaintext bytes we use the key map: the `01`,`02`,`ff`,`f1` in the cipher all corresponds to the 00 (because I was lazy to add more bytes to the mapping), and in the plaintext those are 00.

As we said, the number of pairs (P, C) required is 256.

The space complexity is $2^8 \cdot 16$ bytes (the number of ciphers times the size of each)

The time complexity is 2^8 as we need that many encryption actions to generate the key map.

2.2. **Replace BS with Rotate Clockwise.** The new AES (the 2nd) is less secure than the original AES since we replaced the S-Box operation (BS - which is the only non linear) with a linear function (rotate clockwise).

The rotate clockwise is linear because for two blocks, rotating moves each byte to a new location deterministic and independent of the value and thus, xor(ing) after rotating will xor the same pairs of bytes.

Also note that the location of the xor(ed) results is the same whether we xor first or after (since xor works for each location independently).

So say in location i in the plaintext there were x, y. We can either xor them first and then move to the new location, or move each to the new location and only when xor, the xor and movement is independent of each other.

Thus the rotate clockwise is linear.

Since with the 2nd new AES is made of only linear operations, it can barely be called a cipher (as mentioned in the tutorials).

In conclusion, the new AES (the 2nd) is significantly less secure than the original AES.

2.3. **Before each KM, we'll do a Rotate Clockwise on the sub key. How does the lecturer's new AES compare to the original?** The lecturer's new AES is as secure (at least) as the original.

If we follow the effect of changing a single byte, it still affects **ALL** of the bytes in the cipher as demonstrated in the tutorials.

Assuming the scheduling of the key is good, each of the sub keys are pseudo random.

Combined with rotating clockwise (which is an injective function, meaning we don't lose uniformity/de-generalize the key) we stay with a key which is still pseudo random meaning that the new cipher is at least as strong as the original AES.

2.4. **Zoom's security breach.**

2.4.1. *What is the attribute of the encryption that lead to the researchers to decide the encryption is no good.* As demonstrated in the tutorial, using ECB, encrypting each block is independent so identical blocks will always be encrypted to the same cipher so images will still be coherent.

2.4.2. *Suggest a changed that Zoom can implement that would solve the security issue.* Changing the AES encryption to CBC mode (for example) will solve the issue.

CBC makes the same blocks to be encrypted to different ciphers depending on order.

2.5.

2.5.1. *ECB.* Let blocks B, C be different blocks of plaintext data. Let's set $M_1 = BB$, $M_2 = BC$.

In ECB mode, since the encryption of each block is independent, encrypting the message BB must result in two identical blocks of cipher, while BC must result in two differing blocks of cipher (otherwise the encryption wouldn't be one-to-one).

So, if the cipher is made of two identical halves, we know it's M_1 that was encrypted, if the cipher's blocks differ, them it's M_2 that was encrypted.

2.5.2. *Interleaved-CBC.* We want to cause the input of the AES to be identical for one message and different for the other (same as previous section). But here we have IV(s) being used.

But since our message is only made of two blocks, we can choose $M_1 = (B \oplus IV_0)\,(B \oplus IV_1)$ and $M_2 = (B \oplus IV_0)\,(C \oplus IV_1)$.

Xoring the blocks with IV_0 and IV_1 respectively will cause the blocks B, B and B, C to be the direct input to the AES (E in the diagram). Which means that again, if the cipher is made of two identical blocks, then M_1 was encrypted, otherwise, it's M_2 that was encrypted.

2.6. **One message is sent and the IV for the next message is published already. What to encrypt to figure out what the first message was?** M_1 was encrypted with IV_1. Meaning that $M_1 \oplus IV_1$ is the data that was encrypted.

So if $M_2 =$ "yes" $\oplus\, IV_1 \oplus IV_2$ is the message, the actual data that goes into the AES is "yes" $\oplus\, IV_1 \oplus IV_2 \oplus IV_2 =$ "yes" $\oplus\, IV_1$. So if the resulting cipher is the same as the cipher that was first sent, it means that the input for the AES was the same $\Longrightarrow$ so $M_1 =$ "yes".

If the result cipher is different than the first cipher, then $M_1 =$ "no".