Aes-cbc-IV(Poison IV)

Problem: Can you give me the IV to make the current plain text that is encrypted with aes to decrypt to the second plain text. Send the IV needed to chain the plain text to nc 192.168.3.5 64444

Given:

**Original**:

Pass: sup3r31337. Don't loose it!

**Encrypted**:

4f3a0e1791e8c8e5fefe93f50df4d8061fee884bcc5ea90503b6ac1422bda2b2b7e6a975bfc555f44f7dbcc30aa1fd5e

**IV**:

19a9d10c3b155b55982a54439cb05dce

**Decrypted**:

Pass: notAs3cre7. Don't loose it!

Hint: Study up on aes-cbc first block byte swapping

Steps)

1. Understanding aes-cbc. Aes is a type of encryption that encrypts blocks of data. To learn more about aes and the different types visit [link](https://en.wikipedia.org/wiki/Advanced_Encryption_Standard).



AES is a block cipher, which means that plaintext is split into blocks: every block is encoded with an encryption key of an equal length (128, 192 or 256 bits in case of AES). By itself, a block cipher is only suitable for secure transmission of one block; in order to encode larger amounts of data, various modes of operation were introduced. CBC (Cipher Block Chaining) is one of such modes. To encrypt a block in CBC mode each block's plaintext is XORed with the preceding block's ciphertext (or IV for the first block), then encoded with a chosen algorithm (AES in our case). CBC is widely-used, but because of its properties it's vulnerable to byte-flipping attacks: when you change a byte in a block's ciphertext, the byte in the same position of the next block's plaintext gets changed because of the XOR operation.

1. Understand the problem given. Firstly, we are given the original plain text. Second, we don’t know any key size. Third, we are given the encrypted version of the original plain text. Fourth, which is important, the initial vector used for the encryption. Lastly, the decryption version of text that we want to obtain.

What we want: We want to rearrange the first block of the plain text to be the desired plain text after decryption. The initial vector that we are given would produce the original plain text when used to decrypt so we want to make a new initial vector that would change the first block of the aes decryption to the decrypted plain text.

1. The way we start to get our answer is by using the mathematical equation used by aes.

C = Enc(IV xor M)

C = Enc(IV' xor M')

So, IV xor M == IV' xor M'

C=cipher text

IV=initial vector

IV’=new initial vector to produce our wanted decrypted text (need to find)

M=first plain text

M’ = second plan text

1. Start a script to rearrange the IV to produce the desired plain text.

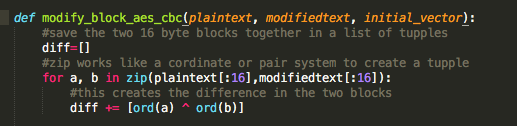
First, create a list variable. Ie such as diff=[]

Second, use a for loop and zip to make a list of tupples. This is basically a coordinate type system that would look like a list like the following:

[x,y],[x1,y2],[x2,y2],…

Third, in the for loop make your list contain (x[plaintext] xor y[modifiedtext])

Such as :

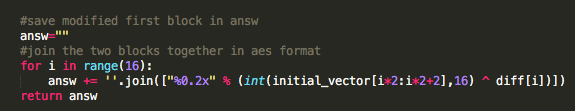


Now, we have a list generated of the given plaintext and the desired plaintext xored together.

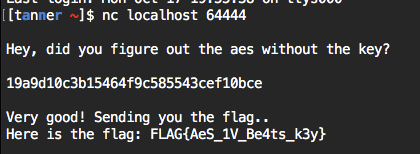
We will begin to generate the new IV now. First a variable will be needed to store the new IV. Such as answ=””

Now we want to loop through 0 to 16 which is the first 16 bytes. In this loop we want to join the bytes using a certain equation to generate the new IV that swaps the bytes to the desired letters. We use the initial vector and swap it around to generate the new IV that gives the desired plain text.

We want to take the IV turn it to a decimal value and swap it using the following equation for the index: i\*2:i\*2\*2. Then xor that part of the IV with the list we created earlier. This looks like the following:



1. Then you want to connect to the server and send it the IV you have created and try to get back the flag.



Sample Script:

def modify\_block\_aes\_cbc(plaintext, modifiedtext, initial\_vector):

#save the two 16 byte blocks together in a list of tupples

diff=[]

#zip works like a cordinate or pair system to create a tupple

for a, b in zip(plaintext[:16],modifiedtext[:16]):

#this creates the difference in the two blocks

diff += [ord(a) ^ ord(b)]

#save modified first block in answ

answ=""

#join the two blocks together in aes format

for i in range(16):

answ += ''.join(["%0.2x" % (int(initial\_vector[i\*2:i\*2+2],16) ^ diff[i])])

return answ

#modies first block aes-cbc $1=plaintext $2=modifiedtext $3=given Initial Vector

print modify\_block\_aes\_cbc('Pass: sup3r31337','Pass: notAs3cre7','19a9d10c3b155b55982a54439cb05dce')