CENG489 FALL 2018 – ASSIGNMENT 1 PART 2

In this assignment, you will write a program that will attack the cipher block chaining mode of AES to create ciphertext that will decrypt to a desired plaintext without knowledge of the secret encryption key or the IV value. Remember in block ciphers, we need to apply padding to the end of plaintext messages when their size is not a multiple of the block size. All that you will have access to as the attacker will a **sample ciphertext** that has been encrypted by the system and a function that tells you during decryption whether the plaintext was padded properly, i.e. prints an **error message when the plaintext has not been padded** properly. You will write your program in Python, so let's start with the environment setup.

Getting ready: Setup the runtime environment

In this assignment, you will be using Python's cryptography library, pycrypto. You can download it at https://pypi.org/project/pycrypto/#files and install it using the following commands after changing to the download directory:

python setup.py build
python setup.py install

After installing the library, you should run the provided file *vulnerable.py* to install the provided encryption and decryption functions, so that your program can use them. Although various padding standards can be used, in this assignment plaintext messages will be padded using the following rule:

If one byte of padding is needed, set the last byte of the block to x01
If two bytes of padding is needed, set the last two bytes of the block to x02
…
If 15 bytes of padding is needed, set the last 15 bytes of the block to x0f
If the last block size is 16, create an additional block filled completely with the byte value x10

Therefore, any padding byte value greater than x10 will be invalid.

e.g. The plaintext block "abcdefgh" will be padded as follows "abcdefgh\x08\x08\x08\x08\x08\x08\x08\x08"

*Remember AES encrypts plaintext blocks of size 128 bits (16 bytes).

The function *decr* takes an input parameter *c* as ciphertext, and returns the message "SUCCESS" if the decryption has been successful or the message "ERROR" if the padding in the decrypted text is invalid.

The vulnerability

Figure 1 below shows the encryption of a plaintext of 3 blocks using the CBC mode of AES and Figure 2 shows the decryption of a ciphertext of the same size.

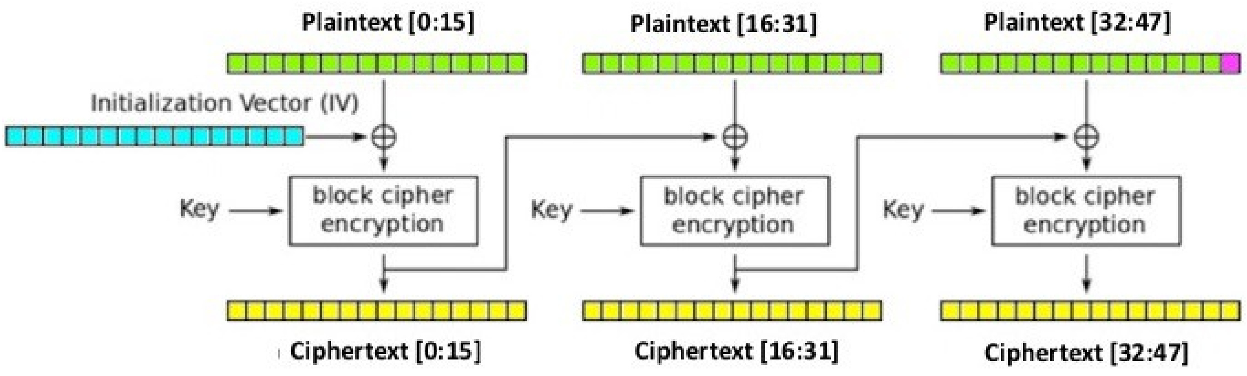

Figure 1. AES encryption in CBC mode

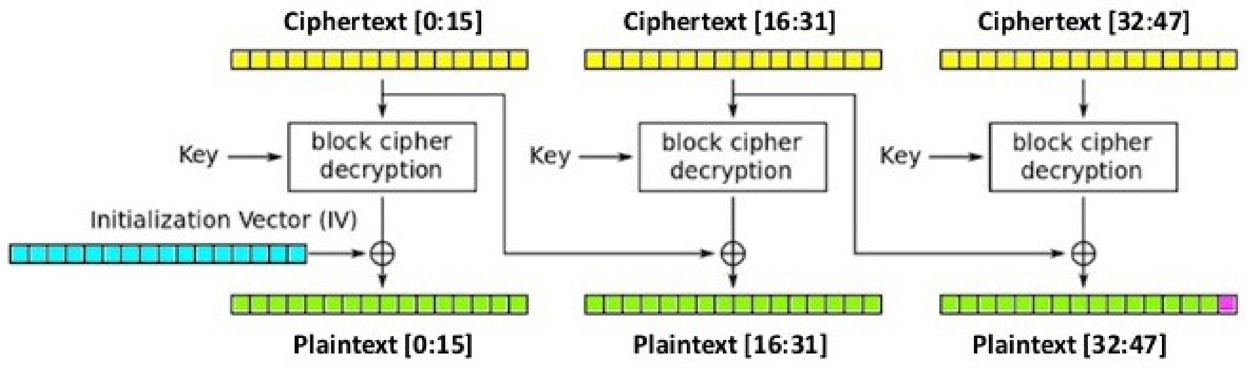

Figure 2. AES decryption in CBC mode

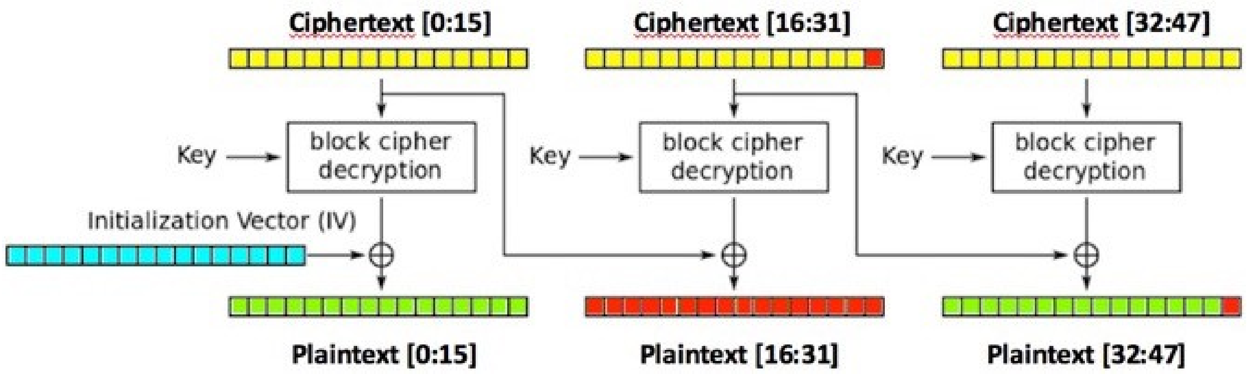

Figure 3. Effect of modifying one byte in ciphertext on decryption

If we change byte 31 of the ciphertext, the second plaintext block will change completely and the last byte of the third block will also change as seen in Figure 3.

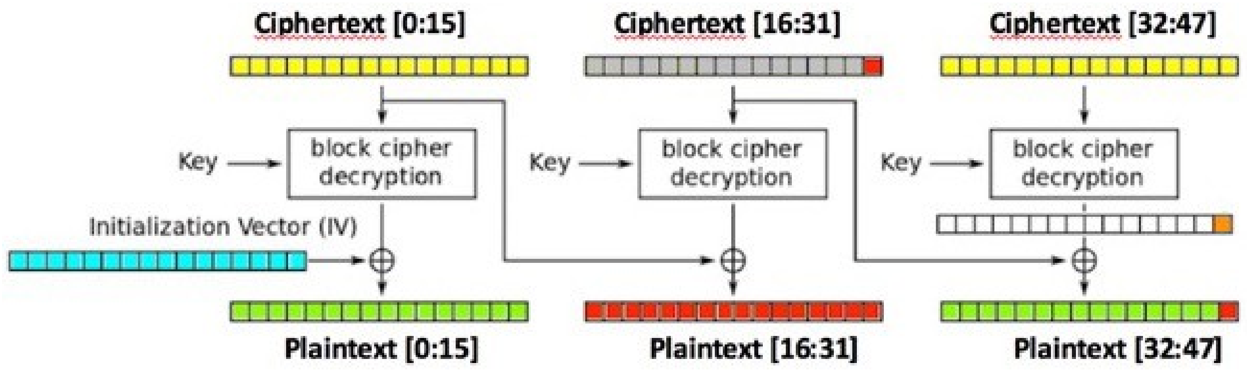

Figure 4. Calculation of the last byte of plaintext

If we change byte 31 of the ciphertext (i.e. Ciphertext[31]) and try to perform decryption, the padding of the plaintext will become invalid unless Plaintext[47] is equal to x01 (due to the padding scheme used). We can find the values resulting in a valid padding by trying all possible values between 0-255 (or x00-xff in hex) for byte 31 (i.e. replacing byte 31 of the ciphertext with all possible values) and observing the decryption result for the whole ciphertext. Obviously, one of the valid values will be the original value of Ciphertext[31] and the other will be the value that makes Plaintext[47] = 1.

Let T = D(C[32:47]) denote the block resulting from the AES decryption of the third block of the ciphertext.

Looking at Figure 4, we know that **Plaintext[47] = (T[15] ⊕ Ciphertext[31])**.

Then, **T[15] = Plaintext[47] ⊕ Ciphertext[31]**

After finding the value Ciphertext'[31] that makes Plaintext[47] = 1, we can calculate

T[15] = Ciphertext'[31] ⊕ 1

In order to calculate T[14], we need to apply padding with 2 characters at the end of the block and calculate Ciphertext'[30:31]. We can calculate Ciphertext'[31] = T[15] ⊕ 2 and try all possible values of Ciphertext'[30] to find the decryption that does not result in a padding error (i.e. results in the last two blocks of plaintext having the value 2). From the correct value found for Ciphertext'[30], we can calculate T[14] = Ciphertext'[30] ⊕ 2. We can continue like this until we find all byte values in T that are required to set the desired bytes in the plaintext.

In order to create a ciphertext block that will decrypt to a plaintext block desired by the attacker (instead of the original plaintext block), the attacker can find all byte values in T (i.e. T[0:15]) and XOR them with the desired plaintext block. Notice that the attacker does not need to know the key or the IV to achieve this and will be able to modify all but the first block of the plaintext by replacing the ciphertext blocks with those calculated as above.

Your task: Write an exploit for this system

Your task is to write a program that exploits the above vulnerability to create a ciphertext that will **end** in a desired plaintext. Your program will take two inputs:

1. C = A ciphertext created by the encryption machine
2. DP = The desired plaintext ending

You should output the ciphertext whose decryption will result in a plaintext ending with DP and the byte value x01 (for a single-character padding), e.g. if DP is “attacked”, the decryption of the ciphertext you output should end in “attacked\x01”.

Note that you can use the **ord** function in Python to find the integer value of a Unicode character.

Important: The given vulnerable.py contains a specific value of the key and IV, but we will run your program with different values when testing. Your implementation should not depend on those values. For the sake of simplicity, you can assume that the given ciphertext C will always consist of 48 bytes and DP will always consist of at least 3 bytes and at most 8 bytes. You can also assume that the length of the original plaintext will not be a multiple of the block size (i.e. the original padding length will be between 1-15 characters).

Submission

Your submission file should be named AESAttack.py and should be runnable as follows:

```
./python AESAttack.py C DP
```

where C and DP are the parameters explained above. Your program should output the whole modified ciphertext whose decryption will result in the desired plaintext ending.

Please be sure to include your program file in the single archive you will submit for both parts of the assignment. Also, you can include anything you want us to know about your program in your README file.