Network System security

Assignment-1 Report

Submitted By

Anjali Singh

M.Tech Cyber Security

2023JCS2565

Submitted To

Mr. Vireshwar Kumar

Indian Institute Of Technology, Delhi

Content

Background Details

Advanced Encryption Standard (AES) is the most popular block cipher utilized for the encryption of electronic data. AES performs operations on 16 bytes of input data at a time using a 128-bit key. AES has been shown to be robust against attacks. However, if the secret key is itself weak (i.e., easily predictable), then AES cannot provide the same security guarantee. Now we will be validating the following statements through the given test cases.

0.1 System specifications

I will be using Macbook Air M2 , hence the time taken for all the codes will be according to these specifications.

0.2 Receiving cipher for required Plaintext

We were given *get_encrypt.py* file which helped us to receive the ciphertext for the given plaintext . I ran the given code for :

The output I received was as follows:

Check submitted files (0/0)

```
All required files submitted!
```

1) Get Encrypted (0/0)

```
Plain text : b'First Plaintextt'
Encryption with 16 set bits :
b'pR\xb0\xa3\xeb\xc0iI}\x99\x89\x13\xb9\x0e\xc2.'
Encryption with 32 set bits :
b'Z}\x8d~\xad\xf9W\xffE\x1cxF3,}\xbe'
Encryption with 48 set bits :
b'\xea$\x17h\xfb\xb3\xab;\xba\xb8\x02\xbb\x01\x8c\xf4\xa9'
Plain text : b'Second Plaintextt'
Encryption with 16 set bits :
b'e^\xa1\xbf\xf1\x84\x19up\x91\x8e\t\xa8\x13\xce.\x97'
Encryption with 32 set bits :
b'Oq\x9cb\xb7\xbd\'\xc3H\x14\x7f\\"1q\xbeR'
Encryption with 48 set bits :
b'\xff(\x06t\xe1\xf7\xdb\x07\xb7\xb0\x05\xa1\x10\x91\xf8\xa9\xc4'
Plain text : b'Third Plaintextt'
Encryption with 16 set bits :
b'bS\xab\xa2\xfb\xc0iI}\x99\x89\x13\xb9\x0e\xc2.'
Encryption with 32 set bits :
b'H|\x96\x7f\xbd\xf9W\xffE\x1cxF3,}\xbe'
Encryption with 48 set bits :
b'\xf8%\x0ci\xeb\xb3\xab;\xba\xb8\x02\xbb\x01\x8c\xf4\xa9'
```

Now we will use above information of plaintext and ciphertext to obtain the key.

1 Case1- 16

1.1 Problem Statement

The first secret key has 16 least significant bits (LSBs) that are random (i.e.,either 0 or 1). The rest of the bits in this secret key are 0's .

1.2 Approach

For the given Case, I used a brute-force approach and tried to find out the last 16 bits to get the full key that could be used to find out the ciphertext for the given plaintext. The ciphertext and plaintext pair that I used to find out the key was obtained with help of above mentioned *get_encrypt.py* file.

I found the key within seconds using this approach.

1.3 Code Discussion

The code that I used for finding 16-bit key using the bruteforce method is as follows:

```
# For 16-bit key
from random import randbytes
import pyaes

def brute_force():
    key = b"\x00" * 14
    plaintext = b'First Plaintextt'
    ciphertext =b'pR\xb0\xa3\xeb\xc0iI}\x99\x89\x13\xb9\x0e\xc2.'

    for i in range(2**16):  # Try all possibilities for the last 4
        bytes
        last_two_bytes = i.to_bytes(2, byteorder='big')
        f_key = key + last_two_bytes
        aes = pyaes.AESModeOfOperationCTR(f_key)
```

```
        deciphered = aes.decrypt(ciphertext)

        if deciphered == plaintext:
            print("Key found:", f_key)


brute_force()
```

In this code we are using a loop and changing the last 16 bits everytime to obtain the final 128 bit bit key.

1.4 Deciphered Key

The **16-bit** key that I obtained is as follows:

```
b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xe9E'
```

2 Case2- 32

2.1 Problem Statement

The first secret key has 32 least significant bits (LSBs) that are random (i.e.,either 0 or 1). The rest of the bits in this secret key are 0's .

2.2 Approach

For the given Case, I used a brute-force approach and tried to find out the last 16 bits to get the full key that could be used to find out the ciphertext for the given plaintext. The ciphertext and plaintext pair that I used to find out the key was obtained with help of above mentioned *get_encrypt.py* file.

Founding the key using this approach took lot of time.

2.3 Code Discussion

The code that I used for finding 32-bit key using the bruteforce method is as follows:

```
# For 32-bit key
from random import randbytes
import pyaes

def brute_force():
    key = b"\x00" * 12
    plaintext = b'First Plaintextt'
    ciphertext =b'Z}\x8d~\xad\xf9W\xffE\x1cxF3,}\xbe'

    for i in range(2**32):  # Try all possibilities for the last 4
        bytes
        last_four_bytes = i.to_bytes(4, byteorder='big')
        f_key = key + last_four_bytes
        aes = pyaes.AESModeOfOperationCTR(f_key)
```

```
        deciphered = aes.decrypt(ciphertext)

        if deciphered == plaintext:
            print("Key found:", f_key)


brute_force()
```

2.4 Deciphered Key

The **32-bit Key** that I obtained is as follows:

b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xef\xcc\xb5d'

3 Case3- 48

3.1 Problem Statement

The first secret key has 48 least significant bits (LSBs) that are random (i.e.,either 0 or 1). The rest of the bits in this secret key are 0s .

3.2 Approach

I tried many different approches for this part but ws unable to figure out finding the key.

3.3 Code Discussion

3.4 Deciphered Key