Lab 2 - Playing with Symmetric Encryption

Submission

You need to write a report which includes:

1. A concise description of your findings for each task.
2. Answers to all questions given in the tasks.
3. The Python scripts you write to accomplish the tasks.

Submission Requirement

1. Use Markdown to write your report; a report template is provided.
2. You can use your favorite Markdown editor.
3. In your report, if code is included, all code should be placed in code blocks/fences. For example:

```
def somefunc(param1='', param2=0):
    '''A docstring'''
    if param1 > param2: # interesting
        print 'Greater'
    return (param2 - param1 + 1) or None
```

4. Do not submit the MD file; instead, export as **pdf** for submission.

Objective

The learning objective of this lab is for students to get familiar with the concepts in the **secret-key encryption**. After finishing the lab, students should be able to gain a first-hand experience on encryption algorithms, encryption modes, paddings, and initialization vector (IV). Moreover, students will be able to use tools and write programs to encrypt/decrypt messages.

Links FYI

- https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html
- https://pycryptodome.readthedocs.io/en/latest/src/examples.html
- https://nitratine.net/blog/post/python-encryption-and-decryption-with-pycryptodome/
- http://www.dreamincode.net/forums/topic/240431-how-to-work-with-bytes-in-python/
- https://pymotw.com/3/struct/

Lab Setup

Making sure Anaconda is installed on the computer. Add the following values to your system PATH environment variable if they are not there:

- path_to_Anaconda (for example, mine is **D:\Anaconda\anaconda3**)
- path_to_Anaconda\Library\bin

- path_to_Anaconda\Scripts

After configuring the environment, install PyCryptodome with the following command.

```
conda install -c conda-forge pycryptodome
```

Also, make sure that you are using the **python program from Anaconda**.

Drills

Task 1 - Encryption using different ciphers and modes

1. Create a python script using the following code, and run it

```
from Crypto.Cipher import AES
import binascii

key = b'mysecretpassword'
plaintext = b'Secret Message A'

encobj = AES.new(key, AES.MODE_ECB)
ciphertext = encobj.encrypt(plaintext)

# ciphertext in bytes
print(ciphertext)
# ciphertext in hex representation
print(binascii.hexlify(ciphertext))
# convert to string object
a = binascii.hexlify(ciphertext)
print(a.decode('ascii'))
```

2. Refer to https://pycryptodome.readthedocs.io/en/latest/src/cipher/cipher.html, modify the above code, and try the following encryption algorithms:
    1. Blowfish
    2. DES
    3. DES3
    4. ARC4

> You do **NOT** need to deliver anything for this task, but make sure you try various encryption algorithms.

Task 2 - Encryption Mode – ECB vs. CBC

The file pic *original.bmp* contains a simple picture. We would like to encrypt this picture, so people without the encryption keys cannot know what is in the picture. Please write a python script to encrypt the file using the **ECB** (Electronic Code Book) and **CBC** (Cipher Block Chaining) modes, and then do the following:

1. Let us treat the encrypted picture as a picture, and use a picture viewing software to display it. However, for the .bmp file, the first **54** bytes contain the header information about the picture,

we have to set it correctly, so the encrypted file can be treated as a legitimate .bmp file. We will replace the header of the encrypted picture with that of the original picture. You can either do this manually after encryption or program it as part of your python script.
2. Display the encrypted picture using any picture viewing software. Can you get any useful information about the original picture from the encrypted picture? Please explain your observations in the report.

In the case of AES, the IV has fixed length of **16** bytes. You may choose the values of the encryption key and the IV.

Some sample code for your reference:

```
def encrypt_bmp(key, iv, mode, in_filename, out_filename=None, chunksize=1024):
    if not out_filename:
        out_filename = in_filename + '.enc'

    if mode == AES.MODE_ECB:
        # no iv is needed for ECB!
        encryptor = AES.new(key, AES.MODE_ECB)
    elif mode == AES.MODE_CBC:
        encryptor = AES.new(key, AES.MODE_CBC, iv)
    else:
        print("invalid mode, program stopped")
        return

    with open(in_filename, 'rb') as infile:
        # extract bmp header
        header = infile.read(54)
        # open output file
        with open(out_filename, 'wb') as outfile:
            outfile.write(header)
            while True:
                chunk = infile.read(chunksize)
                if len(chunk) == 0:
                    break
                elif len(chunk) % 16 != 0:
                    # #pad the last block with spaces
                    chunk += b' ' * (16 - len(chunk) % 16)
                outfile.write(encryptor.encrypt(chunk))
```

> ### Task 2: Questions
>
> 1. Include both **the original picture** and **two encrypted pictures** in your report.
> 2. Include the python script in the report.

Task 3 - Corrupted Ciphertext

To understand the properties of various encryption modes, we would like to do the following exercise:

1. Create a text file that is at least **64** bytes long.

2. Use a python script to encrypt the file with the **AES-128** cipher.
3. Unfortunately, a single bit of the **30th** byte in the **encrypted** file got corrupted. (you can manually do this)
4. Decrypt the corrupted file (encrypted) using the correct key and IV.

> ### Task 3: Questions
>
> 1. How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC respectively?
> 2. Please explain why.

Task 4 - Find the key

You are given a plaintext and a ciphertext, and you know that **AES-128-CBC** is used to generate the ciphertext from the plaintext, and you also know that the numbers in the IV are all zero-bytes (should be a bunch of `\x00`, **NOT** the ASCII character '0'). Another clue that you have learned is that the key used to encrypt the plaintext is a word from a typical English dictionary, **words.txt**. Since the word has less than 16 characters (i.e. 128 bits), space characters (hexadecimal value `0x20`) are appended to the end of the word to form a key of 128 bits. Your goal is to write a program to find out this key. Some key info:

```
Plaintext (total 21 characters):
This is a top secret.
Ciphertext (in hex format):
3f814d00c3f1047f1dfa879115970472472a17eabdd9ba4fcd667743e1e03674
```

You should be aware that AES is a block cipher that operates on 128-bit blocks. You need to pad with last block using the **PKCS7** padding (see course slides for details).
Your job is to write a python script to find the encryption key!

> ### Task 4: Questions
>
> 1. What is the encryption key?
> 2. Include the python script in the report.