Lab 10: Secret-Key Encryption

The goal of this lab assignment is for students to get familiar with the concepts of secret-key encryption. Please refer to the following link at SEEDLabs:

https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Encryption/

The lab description is at:

https://seedsecuritylabs.org/Labs_20.04/Files/Crypto_Encryption/Crypto_Encryption.pdf

Labsetup.zip files:

https://seedsecuritylabs.org/Labs_20.04/Files/Crypto_Encryption/Labsetup.zip

Please review the list of tasks below and work only on the required tasks.

If you have any questions, please let the instructor know.

Deliverables:

Upload a Word document containing each Task's terminal outputs (Use screenshots):

Please include all source code, shell commands, and running results in the lab report. That is, please describe what you have done and what you have observed, including screenshots and code snippets. Do not forget to include your observations when required.

Note: Include your code (or a screenshot of it) when (1) first calling the program and (2) when modifying the code. Work only on the required tasks as follows:

- Task 1 – 30pts
- Task 2 – 20pts
- Task 6 – 30pts
- Task 7 – 20pts

Total: 100pts

Read the next page for more instructions →

For **task 7**, please **use Python code** and the following instead:

Plaintext (total 21 characters): This is a top secret.

Ciphertext (in hex format):

d350530328a77cb5cf0648ad638750d9f814b6897faf6d9d5b21336186370868

IV (in hex format): aabbccddeeff00998877665544332211

Goal: find the password that will result in the same ciphertext.

Useful Python code (read lines and pad them with '#' up to a length of 16):

```
# Using readlines()
file1 = open('words.txt', 'r')
Lines = file1.readlines()

for line in Lines:
        line = line.strip()
        if (line.isalpha()):
                if len(line) < 16:
                        while len(line) < 16:
                        line = line + '#'
```

Refer to the slide with example code on how to do the encryption with padding (CBC mode needs padding)

Programming using Cryptography APIs

```
#!/usr/bin/python3

from Crypto.Cipher import AES
from Crypto.Util import Padding

key_hex_string = '00112233445566778899AABBCCDDEEFF'
iv_hex_string  = '000102030405060708090A0B0C0D0E0F'
key = bytes.fromhex(key_hex_string)
iv  = bytes.fromhex(iv_hex_string)
data = b'The quick brown fox jumps over the lazy dog'
print("Length of data: {0:d}".format(len(data)))

# Encrypt the data piece by piece
cipher = AES.new(key, AES.MODE_CBC, iv)                      ①
ciphertext  = cipher.encrypt(data[0:32])                     ②
ciphertext += cipher.encrypt(Padding.pad(data[32:], 16))     ③
print("Ciphertext: {0}".format(ciphertext.hex()))

# Encrypt the entire data
cipher = AES.new(key, AES.MODE_CBC, iv)                      ④
ciphertext = cipher.encrypt(Padding.pad(data, 16))           ⑤
print("Ciphertext: {0}".format(ciphertext.hex()))

# Decrypt the ciphertext
cipher = AES.new(key, AES.MODE_CBC, iv)                      ⑥
plaintext = cipher.decrypt(ciphertext)                       ⑦
print("Plaintext: {0}".format(Padding.unpad(plaintext, 16)))
```

- We use **PyCryptodome** package's APIs.
- Line:
  1. Initialize cipher
  2. Encrypts first 32 bytes of data
  3. Encrypts the rest of the data
  4. Initialize cipher (start new chain)
  5. Encrypt the entire data
  6. Initialize cipher for decryption
  7. Decrypt