Secret-key encryption (CS 915)

Overview

The learning objective of this lab is for you to get familiar with the concepts of secret-key encryption and some common attacks on encryption. From this lab, you will gain first-hand experience with encryption algorithms, encryption modes, paddings, and the initial vector (IV). Moreover, you will learn to use tools and write programs to encrypt/decrypt messages. Many common mistakes have been made by developers in using encryption algorithms and modes. These mistakes weaken the strength of the encryption and eventually lead to vulnerabilities. This lab exposes you to some of these mistakes and asks you to launch attacks to exploit those vulnerabilities.

Lab setup

- Download files here

This lab is based on the Secret-Key Encryption Lab with some adaptations. All the attacks should be performed within a SEED VM. Refer to the module page on how to set up a VM.

Outline

- Task 1: Familiarize yourself with OpenSSL encryption/decryption
- Task 2: Encryption mode - ECB vs CBC
- Task 3: Padding
- Task 4: Frequency analysis

Task 1: familiarize yourself with OpenSSL encryption/decryption

You can check the OpenSSL manual by typing **man openssl** and **man enc**

Encrypt using a specified key

```
$ openssl enc -ciphertype -e -in words.txt -out words.txt.enc \
-K 00112233445566778889aabbccddeeff \
-iv 01020304050607080102030405060708
```

Replace **-cipertype** with a specific type, such as -aes-128-cbc, -aes-128-cfb, -aes-128-ecb ec. Try to decrypt the ciphertext and see if you recover the same plaintext.

Check all supported cipertypes by typing “**man enc**”. Some common options for the openssl enc command are included below:

```
-in <file>     input file
-out <file>    output file
-e             encrypt
-d             decrypt
-K/-iv         key/iv in hex is the next argument
-[pP]          print the iv/key (then exit if -P)
```

Encrypt using a password

```
$ openssl enc -aes-128-ecb -e -in words.txt -out words.txt.enc
```

The above command works but gives the following warning: “deprecated key derivation used”. Find out what is -pbkdf2 and why it is needed.

Task 2: Encryption mode - ECB vs CBC

The file pic_original.bmp is included in the lab setup files, and it is a simple picture. You need to encrypt this picture to protect the secrecy of the content.

If you simply encrypt pic_original.bmp to produce an encrypted file, you will find you can’t open the encrypted file as a picture (try it), since the metadata is lost.

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. You can use the **bless** hex editor tool (already installed on the SEED VM) to view/edit a binary file.

The following is an example of how to separate the header and body of an image file.

```
$ head -c 54 p1.bmp > header
$ tail -c +55 p2.bmp > body
$ cat header body > new.bmp
```

Based on the above example, encrypt pic_original.bmp in two methods: 1) AES-128-CBC; 2) AES-128-ECB. Do you expect to see any difference in the output between the two methods?

For each method, display the encrypted picture using a picture viewing program (you can use **eog** that is already installed on the SEED VM).

Task 3: Padding

The PKCS#5 padding scheme is widely used by many block ciphers. We will conduct the following experiments to understand how this type of padding works.

Let us create three files, which contain 5 bytes, 10 bytes, and 16 bytes, respectively. We can use the following "echo -n" command to create such files. The following example creates a file f1.txt with length 5 (without the -n option, the length will be 6, because a newline character will be added by echo).

```
$ echo -n "12345" > f1.txt
```

We then use "openssl enc -aes-128-cbc -e" to encrypt these three files using 128-bit AES with CBC mode. Please describe the size of the encrypted files.

We would like to see what is added to the padding during the encryption. To achieve this goal, we will decrypt these files using "openssl enc -aes-128-cbc -d". Unfortunately, decryption by default will automatically remove the padding, making it impossible for us to see the padding. However, the command does have an option called "-nopad", which disables the padding, i.e., during the decryption, the command will not remove the padded data. Therefore, by looking at the decrypted data, we can see what data are used in the padding. Please use this technique to figure out what paddings are added to the three files.

It should be noted that padding data may not be printable, so you need to use a hex tool to display the content. You can use either hexdump or xxd to display a file in hex format.

```
$ hexdump -C p1.txt
00000000 31 32 33 34 35 36 37 38 39 49 4a 4b 4c 0a |123456789IJKL.|
$ xxd p1.txt
00000000: 3132 3334 3536 3738 3949 4a4b 4c0a 123456789IJKL.
```

Task 4: frequency analysis

It is well-known that monoalphabetic substitution cipher (also known as monoalphabetic cipher) is not secure, because it can be subjected to frequency analysis. In this task you are given a cipher-text that is obtained from using a substitution cipher; namely, each letter in the original text is replaced by another letter, where the replacement does not vary (i.e., a letter is always replaced by the same letter during the encryption). Your task is to find out the original text using frequency analysis. It is known that the original text is an English article.

In the following, we describe how we encrypt the original article, and what simplification we have made.

- Step 1: let us generate the encryption key, i.e., the substitution table. We will permute the alphabet from a to z using Python, and use the permuted alphabet as the key. See the following program.

```
#!/bin/env python3
import random
s = "abcdefghijklmnopqrstuvwxyz"
list = random.sample(s, len(s))
key = ''.join(list)
print(key)
```

- Step 2: let us do some simplification to the original article. We convert all upper cases to lower cases and then removed all the punctuations and numbers. We do keep the spaces between words, so you can still see the boundaries of the words in the ciphertext. In real encryption using a monoalphabetic cipher, spaces will be removed. We keep the spaces to simplify the task. We did this using the following command:

```
$ tr [:upper:] [:lower:] < article.txt > lowercase.txt
$ tr -cd '[a-z][\n][:space:]' < lowercase.txt > plaintext.txt
```

- Step 3: we use the **tr** command to do the encryption. We only encrypt letters, while leaving the space and return characters alone.

```
$ tr 'abcdefghijklmnopqrstuvwxyz' 'sxtrwinqbedpvgkfmalhyuojzc' \
< plaintext.txt > ciphertext.txt
```

We have created a ciphertext using a different encryption key (not the one described above). It is called ciphertext.txt in the lab setup file. Your job is to use the frequency analysis to figure out the encryption key and the original plaintext.

Guidelines. Using the frequency analysis, you can find out the plaintext for some of the characters quite easily. For those characters, you may want to change them back to their plaintext, as you may be able to get more clues. It is better to use capital letters for plaintext, so for the same letter, we know which is plaintext and which is ciphertext. You can use the tr command to do this. For example, in the following, we replace letters a, e, and t in in.txt with letters X, G, E, respectively; the results are saved in out.txt.

```
$ tr 'aet' 'XGE' < in.txt > out.txt
```

There are many online resources that you can use. We list four useful links in the following:

- https://www.dcode.fr/frequency-analysis This website can produce the statistics from a ciphertext, including the single-letter frequencies, bigram frequencies (2-letter sequence), and trigram frequencies (3-letter sequence), etc.
- https://en.wikipedia.org/wiki/Frequency_analysis This Wikipedia page provides frequencies for a typical English plaintext.
- https://en.wikipedia.org/wiki/Bigram Bigram frequency.
  https://en.wikipedia.org/wiki/Trigram Trigram frequency.

After Lab

Complete the post-lab assignment as below and submit a report of not more than 4 pages.

The report should contain answers to the following questions. (You must explain answers **in your own words** and demonstrate a good understanding of what you write. Simply citing answers from online resources without showing understanding will get zero marks).

- Task 1: When you encrypt data using a password, you observe a warning “deprecated key derivation used”. Explain why you have such a warning. Furthermore, explain what is -pbkdf2 and why it is needed. [20]

- Task 2: You encrypt pic_original.bmp in two methods: 1) AES-128-CBC; 2) AES-128-ECB. Use diagrams to explain how these two modes work. Describe any difference in the output between the two methods and explain why. [20]

- Task 3: You use "openssl enc -aes-128-cbc -e" to encrypt the three files using 128-bit AES with CBC mode. Describe the sizes of the encrypted files and explain why you have obtained such sizes. Describe what padding has been used in the encryption and explain how you verify that through the decryption process. [20]

- Task 4: Write down the substitution letters in the key (the top row being the plaintext letters) and explain how you have obtained them. [40]

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z