Network Security Lab
Experiment#1
Secret Key Encryption

Dr. Ahmed Awad & Eng. Ibrahim Amryeh

February 7, 2021

1 Objectives

The purpose of this experiment is to get familiar with the basic tools for symmetric encryption. This will be accomplished through applying tasks related to modes of operations, the impact of padding, and the use of Initialization Vector (IV).

2 Pre-Lab

1. Read about DES and AES.
2. Read about ECB and CBC modes of operations.
3. Read about Pseudo Random number generators.

3 Procedure

3.1 Installing OpenSSL

a. Download the latest version of OpenSSL from: **https://www.openssl.org/source/openssl-3.0.0-alpha11.tar.gz.sha1**.

b. Uncompress the folder you have downloaded.

c. Execute the following set of commands to install OpenSSL:
sudo ./config
sudo make
sudo make test
sudo make install

d. Make sure that all the above steps have worked successfully.

3.2 Installing Hex Editor

a. Provide an example of two tools that manipulate files of binary format.

b. Install the tool named **bless** and show the command needed for this purpose.

c. Run bless tool and show that it works properly.

3.3 Text Encryption

a. Prepare a plaintext file name **Plain1.txt** that includes the following data (in Hex format): **5468617473206D79204B756E67204675**. Use bless editor to prepare this file and print out the plaintext as characters.

b. Encrypt the plaintext in the file Plain1.txt using AES with 128-bit key (in ECB mode) with a value of **5468617473206D79204B756E67204675**. Use an Initialization Vector (IV) value of **0102030405060708**. Show the ciphertext. To do so, you need to use the command **enc** in the openssl library.

c. Change the IV of the previous step and show the ciphertext. State your conclusions.

d. Is there padding in the previous encryption?

e. Decrypt the ciphertext you have just created using the same key and IV value.

f. Encrypt the plaintext in the file Plain1.txt using AES with 128-bit key with a value of **5468617473206D79204B756E67204675** with an IV value of **0102030405060708** . using the following modes of operations: CBC, CFB. Change the IV and state your conclusions.

g. Decrypt the ciphertext you created in the previous step.

3.4 Image Encryption

a. download a bmp image from the Internet.

b. Encrypt the downloaded image using ECB mode of operation with 3DES. Choose the key value to be **0E329232EA6D0D73**.

c. Show the encrypted image using some picture viewing software.

d. Can you derive any useful information about the original picture from the encrypted image?

e. Decrypt the encrypted image and show the original picture.

3.5 Corrupted Ciphertext

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

b. Encrypt the file using AES 128 bit cipher with ECB mode of operation.

c. Now, assume that a single bit of the 30th byte in the encrypted file got corrupted. You can do that manually using the bless hex editor.

d. Decrypt the corrupted ciphertext using the key and IV you used in the encryption.

e. Repeat the previous steps using CBC, CBF, and OFB modes of operations. State your conclusions.

3.6 Padding Analysis

a. Create a plaintext in a textfile that is 20 octets long.

b. Encrypt the plaintext using AES with the following modes of operations: ECB, CBC, CFB, and OFB. Choose the key and IVs that you prefer.

c. Report which modes have paddings and which modes do not have. Explain why?

3.7 Pseudo Random Number Generation

a. What is the purpose of pseudo random number generators in security?

b. The randomness is usually measure by entropy which defines how many bits of random numbers does a system have. To get that on your Linux machine, execute the following command: **cat /proc/sys/kernel/random/entropy_avail**. What is the output?

c. Move and click your mouse, type something, and re-run the previous command. State your conclusions.

d. To generate a 16 bytes of a pseudo random number, use the following command **head -c 16 /dev/random — hexdump**.

e. Run again the above command and explain why the program is just waiting?