

# Exploring Symmetric Key Encryption Modes

---

## Getting Started

Boot your Linux system or VM, log in, and then open a terminal window and start the lab:

```
cd labtainer/labtainer-student  
start.py symkeylab
```

It may help to stretch the resulting bash terminal window to the right to provide for more output space.

Note the terminal displays the paths to two files on your Linux host:

- 1) This lab manual
- 2) The lab report template

On most Linux systems, these are links that you can right click on and select “Open Link”. **If you chose to edit the lab report on a different system, you are responsible for copying the completed files back to the displayed path on your Linux system before using “./stop.py” to stop the lab for the last time.**

Familiarize yourself with questions in the lab report template before you start.

In this lab, you will explore the concept of encryption modes and their properties. To perform this exploration, you will be using an open source encryption product known as OpenSSL. You will explore the properties of these modes, seeing a visual representation of the state of the ciphertext, and exploring error propagation during decryption in these various modes.

**Note:** You can use the commands “man openssl” and “man enc” to get additional help on how to use the openssl command line tool. Appendix A contains a quick reference sheet of commonly used Unix commands and appendix B explains hexadecimal.

Use the “ll” command to list the content of the directory.

## Task 1: Warm-up: Encrypt and then decrypt a file (any file)

In this task you are only getting used to the syntax of the `openssl` command.

Do the following:

1. Create a small text file “plain.txt”.
2. To encrypt the text file as “cipher.txt” type:

```
openssl CIPHER -e -in plain.txt -out cipher.txt -K KEY -iv IV
```

replacing:

- *CIPHER* with a specific cipher and CBC mode of operation, *e.g.* **aes-128-cbc**. (To see all the options use “man enc”).
- *plain.txt* is the name of the plain text file you just created
- *cipher.txt* is the name of the output file which will be the ciphertext
- *KEY* with a hexadecimal representation of a symmetric key (your choice)
- *IV* with a hexadecimal representation of an initialization vector (your choice)

**Be sure to use the -K option, and not the -k option.** The latter may not produce an error, but may cause issues later in the lab.

3. Observe the ciphertext you’ve created using `cat`, `more`, `less`
4. You should have observed that the encrypted file is gibberish that will not always display well. To see the actual hex values of the ciphertext, enter the following:  
`hexdump -C FILENAME`
5. List the contents of the directory using the *long* option (i.e., enter “`ls -l`”) to see the sizes for your original file and the encrypted file.

**Record the size of the plaintext file in item 1 of the report.**

**Record the size of the corresponding ciphertext file in item 2 of the report.**

**Note that item 3 asks a follow-up question. You can answer the question now or later (which is true for all the follow-up questions).**

6. You can decrypt the ciphertext you’ve created using the following command.  
**Be sure to output to a new plaintext file**, and not overwrite the original plaintext.

## Exploring Symmetric Key Encryption Modes

```
openssl enc CIPHER -d -in cipher.txt -out plainmod.txt -K KEY -iv IV
```

7. Compare the decrypted plaintext with the original plaintext using the `diff` command, as shown below (replacing `ORIGINAL` and `UNENCRYPTED` with the file names you used):

```
diff -a plain.txt plainmod.txt
```

[Note: if the two files are different then you did something wrong. If `diff` returns nothing, then there were no differences, which is what you would expect.]

## Task 2: Encryption Modes

In this task, you will explore the differences in security attained by several modes of encryption. You will use a web browser on your host Linux system (to view files that you create and modify on the “symkeylab” computer. You will have noticed that the symkeylab computer contains a “index.html” and “nps-logo.bmp” file. To see the page, start the firefox browser using this command:

```
./start_firefox
```

You will see the NPS logo, the (nps-logo.bmp file), and a broken link to a modified version of the logo, (a nps-logo\_mod.bmp file that you will be creating).

### 1. ECB Mode

- a. Observe the NPS logo.

**When looking at the NPS logo with the eye of a cryptanalyst, what patterns do you see? Record your observations in item #4 of the report.**

- b. Encrypt `nps-logo.bmp` using AES in **ECB** mode (with the option **aes-128-ecb**) to create a ciphertext (Name your ciphertext as “nps-logo\_mod.bmp”. [Because ECB mode does not require an IV, you do not need to provide one.]
- c. List the contents of the directory using the *long* option to see the sizes for the plaintext logo file and the encrypted file.

**Record the size of the plaintext logo file in item 5 of the report.**

**Record the size of the encrypted logo file in item 6 of the report.**

## Exploring Symmetric Key Encryption Modes

Note: When you refresh the page in the web browser, it fails to display the modified file because it does not recognize it as a valid image.

- d. You can visualize the ciphertext by tricking a browser into thinking that the encrypted file is still a valid image file. To do this, you need to do a little “preprocessing” to make the file viewable. BMP images have a 54-byte header that informs the image viewer about the image, such as the image size, and its dimensions. You need to replace the encrypted BMP header in the ciphertext you created with a valid BMP header from `nps-logo.bmp`.

To replace the 54-byte header with one command, do the following

```
dd if=nps-logo.bmp of=nps-logo_mod.bmp bs=1 count=54 conv=notrunc
```

- e. After modifying the header, go back to the web browser and refresh the web page to view the encrypted image.

**Using item 7 of the report, describe the visualization of the encrypted logo.**

**Note that item 8 of the report asks a follow-up question.**

### 2. CBC Mode

- a. Encrypt `nps-logo.bmp` again, but this time use **CBC** mode to create a ciphertext (with the **aes-128-cbc** option). [This time you need to provide an IV.]
- b. List the contents of the directory using the *long* option to see the size of the ciphertext file you just created.

**Record the size of the CBC-encrypted logo file in item 9 of the report.**

- c. Using the `dd` command described above, modify the header of the new ciphertext to have the same 54 bytes as the original BMP file.
- d. View the encrypted ciphertext using the web browser refresh.

**Using item 10 of the report, describe the visualization of the encrypted logo.**

## Exploring Symmetric Key Encryption Modes

### 3. CFB Mode

- a. Encrypt the `nps-logo.bmp` file again, but this time use the **CFB** mode (with the **aes-128-cfb** option). [You need to provide an IV.]
- b. List the contents of the directory using the *long* option to see the size of the ciphertext file you just created.

**Record the size of the CFB-encrypted logo file in item 11 of the report.**

- c. Using the `dd` command described above, modify the header of the new ciphertext to have the same 54 bytes as the original BMP file.
- d. View the encrypted ciphertext using the web browser refresh.

**Using item 12 of the report, describe the visualization of the encrypted logo.**

### 4. OFB Mode

- a. Encrypt the `nps-logo.bmp` file again, but this time use the **OFB** mode (with the **aes-128-ofb** option). [You need to provide an IV.]
- b. List the contents of the directory using the *long* option to see the size of the ciphertext file you just created.

**Record the size of the OFB-encrypted logo file in item 13 of the report.**

- c. Using the `dd` command described above, modify the header of the new ciphertext to have the same 54 bytes as the original BMP file.
- d. View the encrypted ciphertext using the web browser refresh.

**Using item 14 of the report, describe the visualization of the encrypted logo.**

**Note that items 15, 16 and 17 of the report ask follow-up questions.**

## Task 3: Error Propagation During Decryption

This task will help you develop an understanding of the ability of various cipher modes of operation to recover from corruption. You will encrypt a text file in two different modes, change a single bit in the middle of the encrypted file, decrypt the corrupted file, and then view the effects of the corrupted bit on the plaintext file.

### 1. Introduction

- a. List the contents of the directory using the *long* option so you can see the size of the `declare.txt` file (in bytes).

**Using item 18 of the report, record the size of `declare.txt`.**

**Calculate the number of AES blocks it will take to encrypt `declare.txt`. Write your answer in item 19 of the report. Show your work.**

**If each character in a text file is represented in ASCII format, where each character is represented in one byte, how many characters does it take to fill up an AES block? Write your answer in item 20 of the report.**

### 2. ECB Mode

- a. Encrypt `declare.txt` using AES-128 in **ECB** mode (with the **aes-128-ecb** option).
- b. Open the encrypted file using hexedit, as shown below, replacing ENCRYPTEDFILE with the name you chose:  
`hexedit ENCRYPTEDFILE`
- c. Click <Enter> to select a byte to go to. When prompted with “New position? 0x”, enter 1230, to take you to near the middle of the encrypted file. You should end up with “00001230” as an address on the left-hand, as shown in the next figure.

## Exploring Symmetric Key Encryption Modes

```
jkhosali@testSVN: ~/Downloads
00000FC0 94 C6 13 DA C4 89 13 27 4E 9C 38 71 E2 C4 89 F3 DF E1 D4 FC 34 A2 22 A7 ..... 'N.8q.....4."
00000FD8 A5 34 40 43 72 48 48 44 29 CD F2 C2 FD 65 AA C7 EB D5 20 7E 24 F7 48 13 .4McRKHd)....e.....$.H.
00000FF0 B7 05 02 24 62 EC CF 62 E1 63 31 21 86 61 72 CE 39 E7 80 58 A7 6E 9D CA ...$.b..c.i.l.ar.9..X.n..
00001008 21 82 52 4A 55 55 75 5D E7 5C 18 86 59 B1 2F 24 64 6C 5F D1 B2 AC 58 66 !.RJUUu]...\Y./$dl...Xf
00001020 14 00 10 42 56 58 9C 15 DB 84 32 76 D5 34 2D D3 E4 88 20 44 B5 47 21 4A ...BVX....2v.4....D.G.IJ
00001038 01 52 02 A1 8C 31 4A 08 08 29 11 A9 A2 30 46 11 84 00 90 84 31 2B 58 BA ..R...1J...)...0F...1+X.
00001050 6F F1 1F 0B 0B EA 6B 6A 5A B7 3E 23 6B 27 D7 6D 9A BA 46 13 44 55 15 1A o....k]Z.>#k'.m..F.DU..
00001068 3B 44 1E 08 A3 B1 AC 36 12 40 72 4B 00 25 C5 EB 97 2C DE 2C 0F 67 0E 1C ;D....6.@rK.%.%,.,g..
00001080 D6 20 99 D4 AE E7 0E 7B 39 00 A5 54 0F 96 EC 5B F1 D7 26 48 4F 4D 4C 14 {9..T...[.&HOML.
00001098 6A 22 02 30 85 31 8A 31 9F 5D CA 98 C2 18 91 BA E9 6E D9 B9 96 8B 1C 9E j".0.1.1].....n.....
000010B0 BD A4 BC DE A4 8E EE 60 E9 B6 1F BF 59 B4 51 64 A6 05 A5 B4 4B 10 42 3F .....Y.Qd.....K.B?
000010C8 BA FD 00 6D E9 CA A8 ED 31 76 14 07 7C C5 9A 33 AD 5E 93 4E 3D 7C CB 96 ...m....iv..[.3.^N=]..
000010E0 2C DF 5C A6 C9 70 78 5B 61 49 37 BD F5 85 ED DC 5F EF 6C DD 36 C5 95 64 ,\...px[aI7.....l.6..d
000010F8 CB FB 71 68 50 70 AF 10 00 54 E4 AF 98 FF 7B 71 93 E4 01 57 B4 4B F5 97 ..qkPp...T....{q...W.K..
00001110 1E 58 F2 ED 6F 49 58 BF DF 88 96 76 74 67 7A C4 3E 4E 08 8F 46 BD 2D DB .X..oIX....vtgz.>N..F.-.
00001128 5F 9C 94 4C 36 FD F4 E6 87 DF 6F 09 B9 68 9D 0B 2E AA 6D E7 9A 99 D2 A2 _..L6....o..h....m....
00001140 E8 C0 94 30 24 D5 82 83 A5 34 ED D2 00 FD B1 35 7F CD 72 DF D0 81 09 69 _..0$....4....5..r....i
00001158 ED 3B 44 1F 51 5A 76 48 F7 1F 5C 7D 74 DB 01 9E D4 15 88 94 00 88 84 98 ;..QZvK...}t.....
00001170 9A 0E 0E B5 76 AB 36 2D E8 A6 D8 D0 59 27 81 46 02 6A 2D AF 42 63 99 5C ...v.6....Y'.F.j-.Bc.\
00001188 25 CF 5D 36 6F F9 0E 74 E5 3A 2F EA 7F 53 1D 67 F3 0B 32 82 33 FF FA EA %.]6o..t.:/.S.g..2.3...
000011A0 B5 DD 76 51 9E 5F 6E 4F 54 05 00 80 69 98 85 D8 74 C8 80 AB 6E 1B DB 23 ..VQ.._nOT...t...n..#
000011B8 43 D1 F2 77 F7 EC BA 69 5F 34 AB 73 FF 26 07 9F BA 73 AD 61 0A 46 68 7C C..w...l.4.s.&...s.a.Fh]
000011D0 F7 30 4E 9C 38 71 E2 C4 89 13 27 4E 9C FF 0E A7 E4 A7 01 29 81 22 46 39 ..0N.8q....'N.....)."F9
000011E8 14 44 2C A7 14 C0 01 00 DF CD 35 77 52 D7 5D 2D 5C 3E 4B BE B3 2D 1C 08 .D.....5wR..]->K.-..
00001200 9B 03 53 A9 10 80 80 B9 21 4B 12 8A 10 8B 22 C3 60 30 64 B3 A9 B1 AC 98 ..S.....IK.....".0d....
00001218 69 69 69 52 88 63 29 46 24 22 31 0D 43 D3 34 42 48 28 14 36 2D 93 10 C2 i!iR.c)F$"1.C.4BH(.6....
00001230 5A A4 94 86 C3 11 87 C3 CE B9 00 00 44 64 8C 51 4A A5 94 15 39 3C 11 2D ...VQ.._nOT...t...n..#
00001248 8B 0B 21 34 4D 8F 44 22 8C D1 63 D6 63 15 54 1C 66 01 12 80 20 10 85 B1 ...14M.D"...c.c.T.f..9<-
00001260 0A 73 05 00 00 63 E7 42 48 A4 89 19 B5 1A 1B 4E 1E 2E 09 08 B5 1E 45 A6 ...c.BH.....N.....E.
00001278 10 21 24 02 02 12 46 A9 42 8F 1B B6 0B B2 22 6C 52 F5 24 66 A4 0A AB 2C .!$...F.B....."lR.$f...
00001290 82 34 0D 40 55 48 AC BA 24 50 25 21 39 AD B9 2B CD 44 56 56 6A C5 EE 92 .4.@UH..$P%19...+DIVVj...
000012A8 10 93 43 62 27 23 4A CE C1 9D E6 65 6A F4 50 91 A5 78 ED 52 A0 F4 97 12 ..Cb'J....ej.P..x.R....
000012C0 4F 8A C7 E3 60 00 80 84 20 10 14 42 5A 9C 78 5D 32 6F E3 5F 6F BD 6E 8C 0....'BZ.x]2o..n..
000012D8 1E D5 FF 9A 5B BA 5E 3A 78 C0 B2 1F 6F FD 38 67 E9 EA B5 05 97 77 0E 35 ...[^:X...g.8g....w.5
000012F0 E8 D6 A6 79 7E 4A 3D A7 97 EC 9B B3 A1 4C 13 40 08 02 22 71 24 24 25 69 ...y-J=.....L.@..q$%$%l
--- declare_mod.png ---@x1230/0xFCCB4-----
```

- d. There will be a group of two hex digits to the right of "00001230". Change the right-most digit of this pair such that only one bit is modified. **Refer to the table in Appendix C** to determine whether to change the hex digit up or down one to ensure that only one bit is modified in the ciphertext.

Enter <Ctrl-X> to **Save the change** when you are done and then exit the hex editor.

- e. Decrypt the ciphertext **without** overwriting the original file.  
f. Use the diff command (with the "-a" option, as shown earlier) to show where the original file is different from the decrypted file. **Using item 21 of the report, describe the corruption of the decrypted file.**

**Note that item 22 is a follow-up question.**

## Exploring Symmetric Key Encryption Modes

### 3. CBC Mode

- a. Encrypt declare.txt again using AES-128 in **CBC** mode (with option **aes-128-cbc**).
- b. Use hexedit again to modify the ciphertext in the same location such that only one bit is modified.
- c. Decrypt the ciphertext **without** overwriting the original file.
- d. Use the diff command (with the “-a” option, as shown earlier) to show where the original file is different from the decrypted file.  
**Using item 23 of the report, describe the corruption of the decrypted file.**

**Note that item 24 is a follow-up question.**

## Submission

After finishing the lab, go to the terminal on your Linux system that was used to start the lab and type:

```
stop.py symkeylab
```

If you modified the lab report on a different system, you must copy those completed files into the directory paths displayed when you started the lab, and you must do that before typing “./stop.py”. When you stop the lab, the system will display a path to the zipped lab results on your Linux system. Provide that file to your instructor, e.g., via the Sakai site.



### Appendix - Some Unix Commands

- cd** Change the current directory.  
    `cd destination`  
With no “destination” your current directory will be changed to your home directory. If you “destination” is “..”, then your current directory will be changed to the parent of your current directory.
- cp** Copy a file.  
    `cp source destination`  
This will copy the file with the “source” name to a copy with the “destination” name. The “destination” can also include the path to another directory.
- clear** Erase all the output on the current terminal and place the shell prompt at the top of the terminal.
- less** Display a page of a text file at a time in the terminal. (Also see more).  
    `less file`  
To see another page press the space bar. To see one more line press the Enter key. To quit at any time press ‘q’ to quit.
- ls** List the contents and/or attributes of a directory or file  
    `ls location`  
    `ls file`  
With no “location” or “file” it will display the contents of the current working directory.
- man** Manual  
    `man command`  
Displays the manual page for the given “command”. To see another page press the space bar. To see one more line press the Enter key. To quit before reaching the end of the file enter ‘q’.
- more** Display a page of a text file at a time in the terminal. (Also see less).  
    `more file`  
To see another page press the space bar. To see one more line press the Enter key. To quit at any time press ‘q’ to quit.
- mv** Move and/or Rename a file/directory  
    `mv source destination`  
The “source” file will be moved and/or renamed to the given

## Exploring Symmetric Key Encryption Modes

“destination.”

pwd      Display the present working directory  
pwd

### Appendix B - Hex Explained

Hex is short for hexadecimal. Numbers can be represented in many ways. We typically count in base-10, or decimal. Binary is counting in base-2. Hexadecimal is counting in base-16.

A standard way to describe these notations is to explain that each 'position' counts powers in these bases. Let  $X_b$  be a number whose value ( $X$ ) should be interpreted as being written in base- $b$ . The decimal number 13 might be written as  $13_{10}$ , for clarity.

Counting by powers:

$$1534_{10} = 4 \times 10^0 + 3 \times 10^1 + 5 \times 10^2 + 1 \times 10^3 = 4 + 30 + 500 + 1000$$
$$1011_2 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 1 + 2 + 8 = 11_{10}$$

The symbols we use to count by powers are all values less than base:

- Base 2 uses two symbols (0,1) to count.
- Base 10 uses ten symbols (0,1,2,3,4,5,6,7,8,9) to count.
- Base 16 uses symbols sixteen symbols: 0,1,2,3,4,5,6,7,8,9,a,b,c,d,e,f.

$$1534_{16} = 4 \times 16^0 + 3 \times 16^1 + 5 \times 16^2 + 1 \times 16^3 = 5428_{10}$$
$$1011_{16} = 1 \times 16^0 + 1 \times 16^1 + 0 \times 16^2 + 1 \times 16^3 = 4113_{10}$$
$$\text{beef}_{16} = 15 \times 16^0 + 14 \times 16^1 + 14 \times 16^2 + 11 \times 16^3 = 48879_{10}$$

Sometimes we write hex digits using a leading '0x' to signal that the number is in hex, since its not always clear: 0x1534, 0x1011, 0xbeef, etc.

A single hex digit can represent any four-digit binary value, or 4 bits:

$$0000_2 = 0x0, 0001_2 = 0x1, \dots, 1110_2 = 0xe, 1111_2 = 0xf$$

As an aside, just as 8 bits is a byte, we sometimes call 4 bits a nibble. Since a byte is eight bits, this means a byte is two nibbles, or two hex digits:

$$0x1f \text{ is the byte } 00011111_2$$

This makes hex a convenient notation for writing bytes of random data, like keys.

## Appendix C - Hex to Binary Translation

The following table provides the translation of hex to binary.

Hex digit	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111