ECE/CS 5560

Lab2: Symmetric Key Encryption Lab

Covered Topic

- Secret-key encryption
- Substitution cipher and frequency analysis
- Encryption modes, IV, and paddings
- Common mistakes in using encryption algorithms
- Programming using the crypto library

Make sure you complete Lab 1 before you proceed to this lab!

Task 1

- Goal: File encryption with different ciphers and modes
- Use **openssl** to encrypt files

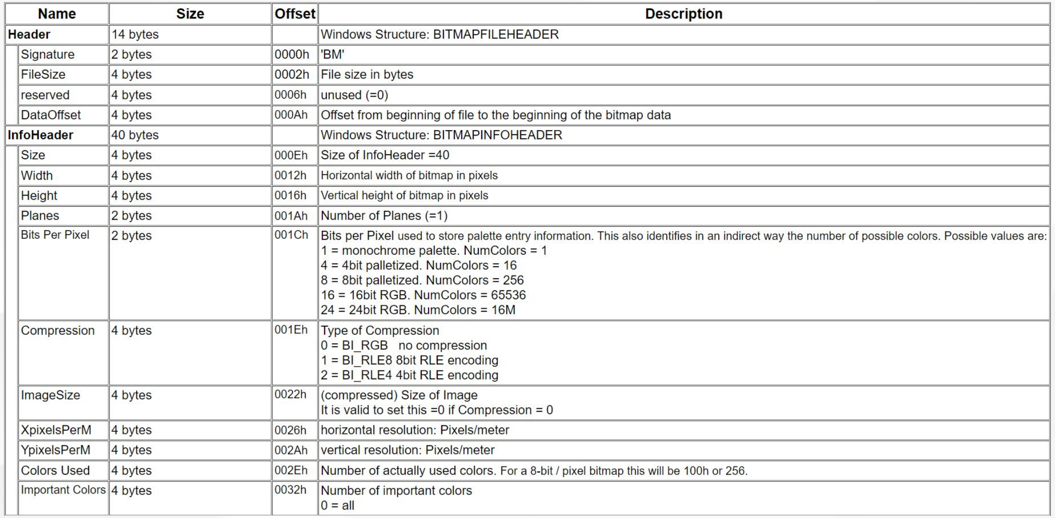
```
$ openssl enc -ciphertype -e -in plain.txt -out cipher.bin
                   -K 00112233445566778889aabbccddeeff
                   -iv 0102030405060708
```

-ciphertype supported cipher type can be found using “man enc” command

-e	encrypt
-in	input file name
-out	output file name
-K/-iv	key/iv value in HEX

Task 2

- Goal: Encrypt picture file using ECB and CBC, observe encryption difference
- Use a picture of your choice or the pic_original.bmp in lab_symmetric.zip (picture should be in **.bmp format**)
- Only encrypt the **body** of the picture but keep the header



Name	Size	Offset	Description
Header	14 bytes		Windows Structure: BITMAPFILEHEADER
Signature	2 bytes	0000h	'BM'
FileSize	4 bytes	0002h	File size in bytes
reserved	4 bytes	0006h	unused (=0)
DataOffset	4 bytes	000Ah	Offset from beginning of file to the beginning of the bitmap data
InfoHeader	40 bytes		Windows Structure: BITMAPINFOHEADER
Size	4 bytes	000Eh	Size of InfoHeader =40
Width	4 bytes	0012h	Horizontal width of bitmap in pixels
Height	4 bytes	0016h	Vertical height of bitmap in pixels
Planes	2 bytes	001Ah	Number of Planes (=1)
Bits Per Pixel	2 bytes	001Ch	Bits per Pixel used to store palette entry information. This also identifies in an indirect way the number of possible colors. Possible values are: 1 = monochrome palette. NumColors = 1 4 = 4bit palletized. NumColors = 16 8 = 8bit palletized. NumColors = 256 16 = 16bit RGB. NumColors = 65536 24 = 24bit RGB. NumColors = 16M
Compression	4 bytes	001Eh	Type of Compression 0 = BI_RGB no compression 1 = BI_RLE8 8bit RLE encoding 2 = BI_RLE4 4bit RLE encoding
ImageSize	4 bytes	0022h	(compressed) Size of Image It is valid to set this =0 if Compression = 0
XpixelsPerM	4 bytes	0026h	horizontal resolution: Pixels/meter
YpixelsPerM	4 bytes	002Ah	vertical resolution: Pixels/meter
Colors Used	4 bytes	002Eh	Number of actually used colors. For a 8-bit / pixel bitmap this will be 100h or 256.
Important Colors	4 bytes	0032h	Number of important colors 0 = all

Task 2

- Encrypt the whole picture

```
$ openssl enc -ciphertype -e …
```

- Retrieve the header from original image and body from encrypted picture

```
$ head -c 54 [original_picture] > header
$ tail -c +55 [encrypted_picture] > body
```

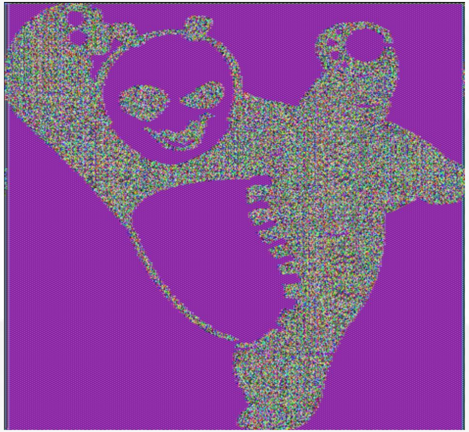
- Combine header and body into a .bmp file

```
$ cat header body > [new_picture]
```

- Open [new_picture] using image viewer (eog on our VM)

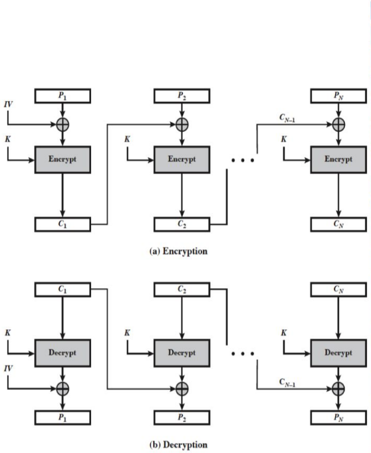
Task 2

- Perform encryption using both ECB and CBC
- Report any observations you may have
- Provide explanation for your observations



Task 3

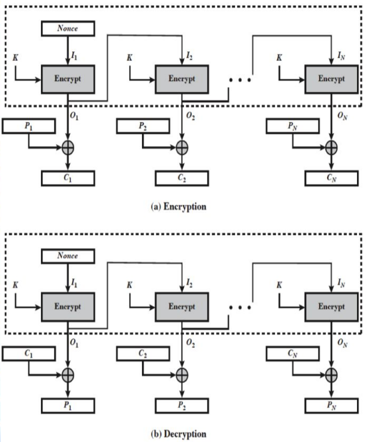
- Goal: Implement CBC and OFB mode using AES encryption
- Code for performing AES encryption/decryption is provided in lab_symmetric.zip
- Use Figure 7.4 and 7.6 in the lab document as a guide



(a) Encryption

(b) Decryption

Figure 7.4 Cipher Block Chaining (CBC) Mode



(a) Encryption

(b) Decryption

Figure 7.6 Output Feedback (OFB) Mode

Task 3

- Hints:
  - Convert key, iv, and plaintext/ciphertext to **HEX value** to perform the encryption/decryption
  - Make sure to use **correct padding** for CBC encryption, you are allowed to use Crypto.Util.Padding package for creating and removing pad
  - Use encryption values in Task3.txt of lab_symmetric.zip to verify your result

Task 4

- Goal: Observe paddings
- 4.1: only answer the questions about paddings for different ciphers
- 4.2: use aes-128-cbc
  - Generate files with 5 bytes, 10 bytes, and 16 bytes
  - Encrypt all files to ciphertexts
  - Decrypt all ciphertexts to plaintext **with nopad**
  - Compare hex value for plain1 and plain2

```
$ openssl enc -ciphertype -e
```

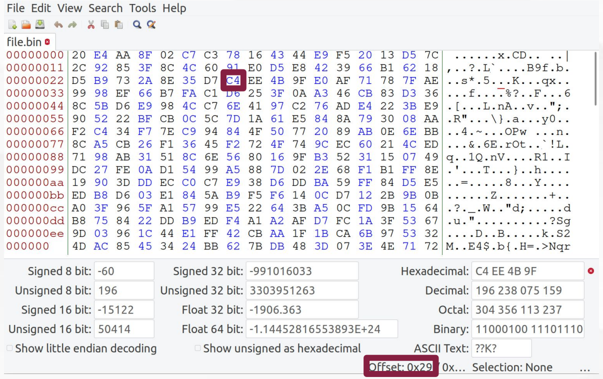
```
$ openssl enc -ciphertype -d -nopad
```

```
$ hexdump -C [plaintext]
```

Task 5

- Goal: Observe error propagation for different cipher modes
- Hint:
  - Use 2000 bytes of 0x00 to better observe the result
  - Use bless tool to open the encrypted file, manually change the 42nd byte (offset 0x29)
  - Compare both plaintexts

```
$ truncate -s 2K file.bin
```



```
00000000 20 E4 AA 8F 02 C7 C3 78 16 43 44 E9 F5 20 13 D5 7C  ......x.CD.. ..|
00000011 2C 92 85 3F 8C 4C 60 91 E0 D5 E8 42 39 66 B1 62 18 ,..?.L`....B9f.b.
00000022 D5 B9 73 2A 8E 35 D7 C4 EE 4B 9F E0 AF 71 78 7F AE ..s*.5...K...qx..
00000033 99 98 EF 66 B7 FA C1 D6 25 3F 0A A3 46 CB 83 D3 36 ...f....%?..F...6
00000044 8C 5B D6 E9 98 4C C7 6E 41 97 C2 76 AD E4 22 3B E9 .[...L.nA..v..";.
00000055 90 52 22 BF CB 0C 5C 7D 1A 61 E5 84 8A 79 30 08 AA .R"...\}.a...y0..
00000066 F2 C4 34 F7 7E C9 94 84 4F 50 77 20 89 AB 0E 6E BB ..4.~...OPw ...n.
00000077 8C A5 CB 26 F1 36 45 F2 72 4F 74 9C EC 60 21 4C ED ...&.6E.rOt..`!L.
00000088 71 98 AB 31 51 8C 6E 56 80 16 9F B3 52 31 15 07 49 q..1Q.nV....R1..I
00000099 DC 27 FE 0A D1 54 99 A5 88 7D 02 2E 68 F1 B1 FF 8E .'...T...}..h....
000000aa 19 90 3D DD EC C0 C7 E9 38 D6 DD BA 59 FF 84 D5 E5 ..=.....8...Y....
000000bb ED B8 D6 03 E1 84 5A B9 F5 F6 14 0C D7 12 2B 9B 0B ......Z.......+..
000000cc A0 3F 96 5F A1 57 99 E5 22 64 3B A5 0C FD 9B 15 64 .?._.W.."d;.....d
000000dd B8 75 84 22 DD B9 ED F4 A1 A2 AF D7 FC 1A 3F 53 67 .u."..........?Sg
000000ee 9D 03 96 1C 44 E1 FF 42 CB AA 1F 1B CA 6B 97 53 32 ....D..B.....k.S2
000000   4D AC 85 45 34 24 BB 62 7B DB 48 3D 07 3E 4E 71 72 M..E4$.b{.H=.>Nqr
```

Signed 8 bit: -60 Signed 32 bit: -991016033 Hexadecimal: C4 EE 4B 9F

Unsigned 8 bit: 196 Unsigned 32 bit: 3303951263 Decimal: 196 238 075 159

Signed 16 bit: -15122 Float 32 bit: -1906.363 Octal: 304 356 113 237

Unsigned 16 bit: 50414 Float 64 bit: -1.14452816553893E+24 Binary: 11000100 11101110

□ Show little endian decoding □ Show unsigned as hexadecimal ASCII Text: ??K?

Offset: 0x29 0x... Selection: None ...

Task 6

- Goal: Decrypt cipher using same or predictable IV
- 6.1:
  - Use a **short file** (i.e. 20 bytes) for encryption, easy to compare and observe the result
  - You can ignore the warning “hex string is too short…”
- 6.2:
  - Use the sample_code.py in lab_symmetric.zip for OFB
  - Explain if we replace OFB with CFB, **no need to demonstrate**

Task 6

- 6.3:
  - Make sure you read through **Section 2: Lab Environment** at the beginning of the lab document
  - The plaintext that you input should be a **HEX value**, convert string to HEX first
  - Bob's secret message is **Yes** or **No** with capitalized **Y** and **N**
  - You also need to consider CBC padding

Questions?

Yousef Akbar

akbary@vt.edu

Office Hours: Mon. 1:30 - 3:30pm, Thu. 1:30 - 3:30pm

Zoom: https://virginiatech.zoom.us/j/6931202457