

APPLIED CRYPTOGRAPHY

LAB-4

Date: 07/10/2022

Submitted by: Mahika Gupta

SRN: PES1UG20CS243

Task 1: Encryption using Different Ciphers and Modes

1. Using AES-128-cbc:

```
PES1UG20CS243:Mahika~/.../Labsetup$>cat>plain.txt  
Hi this is a message from Mahika!
```

```
^C
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>cat plain.txt  
Hi this is a message from Mahika!
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>■
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cbc -e -in plain.txt -  
out cipher.bin \-K 0011223344556677889aabccddeff \-iv 0102030405060708  
hex string is too short, padding with zero bytes to length  
PES1UG20CS243:Mahika~/.../Labsetup$>cat cipher.bin  
500`U!00.m00P0000060 u00f0000R*F >a0002-}000TPES1UG20CS243:Mahika~/.../Labset  
up$>■
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cbc -d -in cipher.bin  
-out decrypted.txt \-K 0011223344556677889aabccddeff \-iv 0102030405060708  
hex string is too short, padding with zero bytes to length  
PES1UG20CS243:Mahika~/.../Labsetup$>cat decrypted.txt  
Hi this is a message from Mahika!  
PES1UG20CS243:Mahika~/.../Labsetup$>■
```

2. Using AES-128-ecb:

```
PES1UG20CS243:Mahika~/.../Labsetup$>rm decrypted.txt  
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-ecb -e -in plain.txt -  
out cipher.bin \-K 00112233445566778889aabbccddeeff  
PES1UG20CS243:Mahika~/.../Labsetup$>cat cipher.bin  
*00{h*e*ö00{övd00t]ll0&0Ca0$jl0PES1UG20CS243:Mahika~/.../Labsetup$>■
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-ecb -d -in cipher.bin  
-out decrypted.txt \-K 00112233445566778889aabbccddeeff  
PES1UG20CS243:Mahika~/.../Labsetup$>cat decrypted.txt  
Hi this is a message from Mahika!  
PES1UG20CS243:Mahika~/.../Labsetup$>■
```

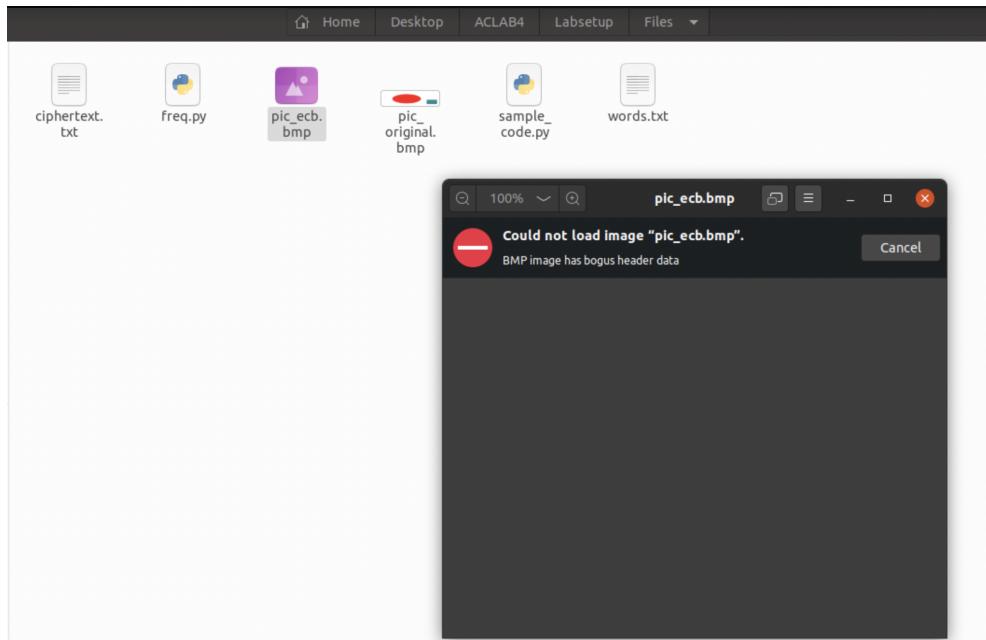
3. Using DES:

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -des -e -in plain.txt -out cipher.bin \-K 00112233445566778889aabbccddeeff \-iv 0102030405060708  
hex string is too long, ignoring excess  
PES1UG20CS243:Mahika~/.../Labsetup$>cat cipher.bin  
0$C+00c0T/j0m00IxF008g0dXh000PES1UG20CS243:Mahika~/.../Labsetup$>■
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -des -d -in cipher.bin -out decrypted.txt \-K 00112233445566778889aabbccddeeff \-iv 0102030405060708  
hex string is too long, ignoring excess  
PES1UG20CS243:Mahika~/.../Labsetup$>cat decrypted.txt  
Hi this is a message from Mahika!  
PES1UG20CS243:Mahika~/.../Labsetup$>■
```

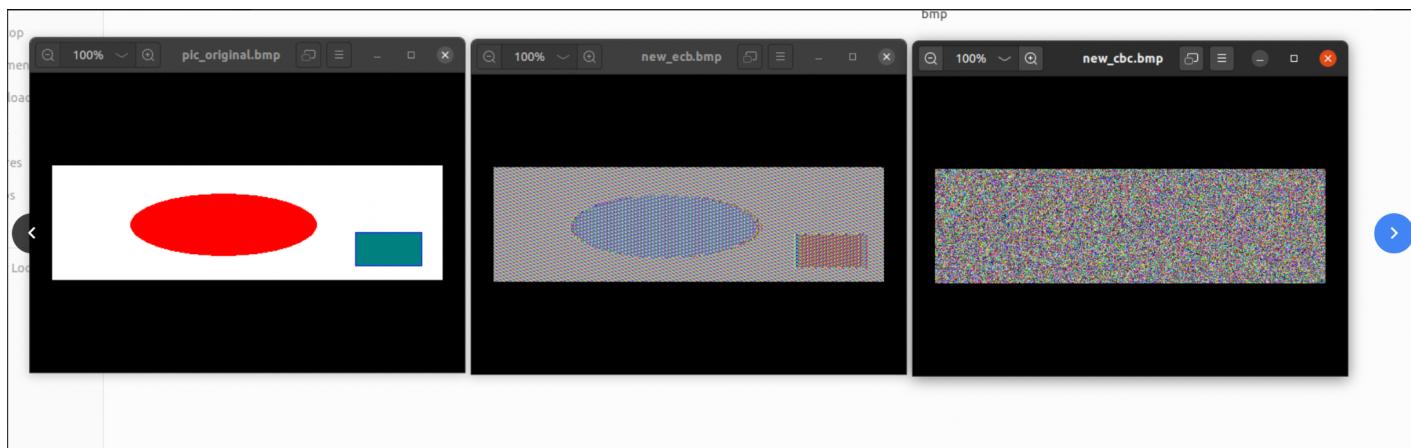
Task 2: Encryption Mode – ECB vs. CBC

```
PES1UG20CS243:Mahika~/.../Files$>openssl enc -aes-128-ecb -e -in pic_original.bmp -out pic_ecb.bmp \-K 00112233445566778889aabbccddeeff \-iv 0102030405060708  
warning: iv not used by this cipher  
PES1UG20CS243:Mahika~/.../Files$>■
```



```
PES1UG20CS243:Mahika~/.../Files$>openssl enc -aes-128-cbc -e -in pic_original.bmp -out pic_cbc.bmp -K 00112233445566778889aabcccddeeff -iv 0102030405060708  
hex string is too short, padding with zero bytes to length  
PES1UG20CS243:Mahika~/.../Files$>ls  
ciphertext.txt  pic_cbc.bmp  pic_original.bmp  words.txt  
freq.py        pic_ecb.bmp  sample_code.py  
PES1UG20CS243:Mahika~/.../Files$>■
```

After adding headers to the image files:



Here we can observe that after using ecb for encryption, the image is still slightly distinguishable. However, on encrypting with cbc, the image is completely imperceptible.

Task 3: Padding:

```
PES1UG20CS243:Mahika~/.../Labsetup$>echo -n "12345" > f1.txt
PES1UG20CS243:Mahika~/.../Labsetup$>echo -n "1234567890" > f2.txt
PES1UG20CS243:Mahika~/.../Labsetup$>echo -n "1234567890abcdef" > f3.txt
PES1UG20CS243:Mahika~/.../Labsetup$>ls -l f*.txt
-rw-rw-r-- 1 seed seed 5 Oct 7 11:49 f1.txt
-rw-rw-r-- 1 seed seed 10 Oct 7 11:49 f2.txt
-rw-rw-r-- 1 seed seed 16 Oct 7 11:49 f3.txt
PES1UG20CS243:Mahika~/.../Labsetup$>■
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cbc -e -in f1.txt -out
f1.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
PES1UG20CS243:Mahika~/.../Labsetup$>cat f1.bin
0|0f00m0Fr00dPES1UG20CS243:Mahika~/.../Labsetup$>^C
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cbc -e -in f2.txt -out
f2.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
PES1UG20CS243:Mahika~/.../Labsetup$>cat f2.bin
0g0Y00A5t2FA00PES1UG20CS243:Mahika~/^C
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cbc -e -in f3.txt -out
f3.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
PES1UG20CS243:Mahika~/.../Labsetup$>cat f3.bin
0.%0;0$0fs000000Gdo0|F00P0PES1UG20CS243:Mahika~/.../Labsetup$>■
```

```
1234567890abcdefPES1UG20CS243:Mahika~.../Labsetup$>xxd p1.txt  
00000000: 3132 3334 350b 0b0b 0b0b 0b0b 0b0b 12345.....  
PES1UG20CS243:Mahika~.../Labsetup$>xxd p2.txt  
00000000: 3132 3334 3536 3738 3930 0606 0606 0606 1234567890.....  
PES1UG20CS243:Mahika~.../Labsetup$>xxd p3.txt  
00000000: 3132 3334 3536 3738 3930 6162 6364 6566 1234567890abcdef  
00000010: 1010 1010 1010 1010 1010 1010 1010 1010 .....  
PES1UG20CS243:Mahika~.../Labsetup$>█
```

In p1.txt padding is done by 0b.

In p2.txt it is done by 06.

In p3.txt it is done by 10.

Task 4: Error Propagation – Corrupted Cipher Text

```
PES1UG20CS243:Mahika~/.../Labsetup$>nano error.txt
PES1UG20CS243:Mahika~/.../Labsetup$>cat error.txt
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse in dui
et velit dictum dignissim. Suspendisse placerat erat ac ipsum tempor, id ves-
tibulum urna bibendum. Cras placerat vehicula sem, ut ultrices felis. Etiam
congue posuere mi, sed tincidunt lorem euismod sit amet. Phasellus congue ma-
ssa ac metus tempor sollicitudin. Aenean cursus massa at quam suscipit vulpu-
tate. Nunc eget bibendum lorem. Sed eros libero, pulvinar ut dolor eu, cursu-
s volutpat tellus. Vestibulum et neque condimentum, pharetra velit sodales,
rhoncus enim. Sed rhoncus sapien sed facilisis interdum. Nunc suscipit liber-
o a ipsum gravida sollicitudin. Nulla turpis risus, ornare sed nunc nec, fin-
ibus venenatis ipsum. Donec congue malesuada congue. Praesent mollis consequ-
at nisi sed pretium. Donec a lacus ut neque convallis tincidunt.

Sed interdum, nunc eu hendrerit semper, lectus neque gravida dui, a blandit
lorem dolor et magna. Suspendisse luctus consequat urna ut venenatis. Nunc q-
uis rutrum eros, in cras.
PES1UG20CS243:Mahika~/.../Labsetup$>wc -c error.txt
1004 error.txt
PES1UG20CS243:Mahika~/.../Labsetup$>
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cbc -e -in error.txt
-t out enc.bin -K 00112233445566778899aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
PES1UG20CS243:Mahika~/.../Labsetup$bless enc.bin
```

File Edit View Search Tools Help

enc.bin

00000000	96 85 D6 6E 21 AE 54 03 8B 96 2E 8C F2 06 C9 B0 71 18 B4 C6 C0 17 3B 97 15 02 B2 BB AE F7 92 DE 90 21 45 11 3D FA 03 DO 63 60 18
0000002b	1B 15 53 E5 96 8C F0 6A 89 F1 1E 3E 7D B1 4F EE 49 E7 E5 B6 E5 9F E3 48 8C 3D 0D 9C F5 61 18 D9 16 6E 12 CC E7 DD F6 4D 3F 34 ED
00000056	68 56 AD ED CA 51 9D BE BA 16 5F DB D7 7A 16 71 42 88 BD 35 D5 75 24 E7 D6 18 69 D3 47 FA 4D B7 01 B2 47 6C CE 38 2A 03 08
00000081	75 D6 46 84 1D E8 A4 77 55 0F F6 F7 D9 D1 17 39 99 FB 6D F2 3C F6 F4 6B A2 11 04 B4 83 9B 27 D0 3A 08 6C 92 71 82 DF E0 9D AB D8
000000ac	08 BC 7A EC FD BB C9 48 F3 OF 79 0E EE D2 77 D8 E2 3E 00 54 4C 1D 7C F1 46 14 OF A6 27 94 94 59 BF FD 9F 5F 16 8C F9 16 A2 63 A8
000000d7	2E 9A BD F3 AE 6B E5 A5 C9 EB 39 9D 2E D4 9F FB 5D 7E 19 5E 4B 00 70 FB 63 79 55 D7 TE 4C 70 FF 1F DB 31 88 C5 0E 4C 79 EC 1B
00000102	3D 5C 41 52 57 AE OF 75 C8 36 47 D9 4C C5 E4 E7 38 6A CC 1C A3 88 80 12 8E 79 E2 D2 59 39 F2 BF 9F 46 79 49 7D 95 B5 5D 60 F6 73
0000012d	55 1C 20 8C 2A 70 Ff A7 07 B1 62 42 F5 C2 15 88 BF 4E 86 08 AD 54 6B 15 84 81 OA BA 03 7C 70 BD C3 CD A4 9A 0F E7 1F F5 FF F5 8C U
00000158	4B 94 56 AB CF F3 AF C6 C4 6F 3C 46 77 F9 27 9A 6B 0D D4 BF D8 A8 E3 40 AD A1 91 DB D5 95 5F 37 85 80 76 21 88 19 3A 74 9E 45 4C K
00000183	86 CB CD 7C 42 A3 CF 2A DD 82 65 51 5F D4 67 C1 0B D7 CE AE 6D 0E 5F 54 04 FC FE 84 08 CF 1C 94 14 17 C2 F4 D1 1F 8F 6A C4
000001ae	B4 BE 69 03 48 63 55 C8 F7 D8 E5 3A 27 E2 8C CD B4 E8 6F 88 BB 1F 42 DE 62 CD 34 TA 01 BB 15 CB 96 37 EB 53 E3 0D CD 08 11 13 89
000001d9	BD 67 F5 GF F2 C5 14 08 71 93 ED EB E7 54 B2 8B 45 ED 14 4F 77 7D 48 E6 1F 2C 91 89 7B AC FF B3 29 10 54 F5 63 4D 57 F6 63 EC 53
00000204	01 72 2D 16 DF 73 62 5D A8 54 62 CA BE 4E 40 37 D7 B8 13 90 31 4A F5 08 19 C5 D4 14 A0 16 02 E6 61 D0 85 DF 88 9C 33 41 C0 F8 F8
0000022f	9A C5 1C 5E FD F8 7B 80 70 19 0E CO D0 93 C5 32 F9 58 B5 B1 F5 98 61 A9 27 C5 77 37 AA 2B 6A 7E 54 38 16 74 DF A4 44 00 3F 4E
0000025a	D2 F2 89 1C ED 4A 8D BE 3A BB 55 38 D3 DF 4F 55 F1 21 79 38 C4 9E 16 7E DB AA 80 F7 E6 EE 47 D5 2B 92 OE 68 72 C6 EF 84 29 78 88
00000285	E7 BF C2 65 8F 4C BF 4B 5D 1F B5 EE 4F 1A 03 43 C7 EF 92 60 2E 80 43 7C 5A CF 99 34 5D C0 54 82 27 F2 98 16 90 A8 49 41 83
000002b0	EF 91 3B 10 E7 84 F6 E6 90 6C EA 92 54 E8 38 8C 5A 28 75 D8 7A EE 70 CA BE 6D B8 D8 F7 0D 23 EA 91 5F 60 68 22 BD C2 DB 4C 9D 4B
000002db	DE 50 AC 14 78 1F 59 CB 48 9F 45 1E E6 1B 75 F2 44 39 CB 2E 2B B3 DA 91 CD 90 3C 76 0A 14 6F FE F7 3C EF 43 21 45 AC E3 OA CA 1C A9 0A 70 76 C4 D2 BD B
00000306	42 97 CF 3A A6 04 8E 3F BA 19 B3 F8 86 F8 CA DA 91 CD 90 3C 76 0A 14 6F FE F7 3C EF 43 21 45 AC E3 OA CA 1C A9 0A 70 76 C4 D2 BD B
00000331	D1 3A 80 91 07 A0 40 CO FF E8 64 2C 43 1D F8 11 12 58 54 E9 3B 60 58 16 DC D6 AF BA 38 FE 79 40 66 4A 34 06 FD E0 4C 11 F7 2A EB
00000325c	B8 51 76 48 34 09 23 A3 4E CB B9 1A BC 3A 5B 02 67 E2 EA 10 04 82 88 3E 6D BD F9 E5 57 76 74 CE 1E 88 84 9A 16 BA 86 05 05 23 BF
00000387	A6 3E 8E 93 33 46 91 88 30 7A 20 61 F1 7D AF 10 63 06 62 C4 DA 51 21 1B 0E 2C B6 A2 C3 33 5A 6F 49 D5 96 72 D9 D4 95 BC 79 26 D4
000003b2	C2 BE 78 38 3A D1 66 6C B5 87 6E 8A CD BD 2D 49 06 76 97 77 AF 09 6D 3D 40 6A 8C FB FD EB 45 7E 4D B4 08 94 80 C0 00 33 7E D8 76
000003dd	82 CC A0 9F 68 AB 51 71 96 7A 95 71 71 D1 D3 DC 87 89 41

Signed 8 bit: -106
Unsigned 8 bit: 150
Signed 16 bit: -27003
Unsigned 16 bit: 38533

Signed 32 bit: -1769613714
Unsigned 32 bit: 2525353582
Float 32 bit: -2.162263E-25
Float 64 bit: -3.56613809140253E-200

Hexadecimal: 96 85 D6 6E
Decimal: 150 133 214
Octal: 226 205 326
Binary: 10010110 10

Show little endian decoding Show unsigned as hexadecimal ASCII Text: ??n
Offcial: 0x00 / 0x00

The screenshot shows a hex editor interface with a menu bar (File, Edit, View, Search, Tools, Help) and a toolbar with various icons. The main window displays a file named "enc.bin*" with a list of memory addresses and their corresponding hex values. The addresses are listed on the left, and the hex values are listed on the right. One byte at address 0000002b is highlighted in red, specifically the byte at index 4 (value E4). The rest of the bytes are in blue.

Address	Value											
00000000	96	85	D6	6E	21	AE	54	03	8B	96	2E	80
0000002b	16	15	53	E4	96	8C	F0	6A	89	F1	1E	3F
00000056	6E	56	AD	ED	CA	51	9D	BE	BA	16	20	5F
00000081	75	D6	46	84	1D	E8	A4	77	55	0F	F6	F1
000000ac	08	BC	7A	EC	FD	B8	C9	48	F3	0F	79	0F
000000d7	2E	9A	BD	F3	AE	6B	E5	A5	C9	E8	39	9I
00000102	3D	5C	41	52	57	AE	0F	75	C8	36	47	D9
0000012d	55	1C	20	8C	2A	70	FF	A7	07	B1	62	42
00000158	4B	94	56	AB	CF	F3	AF	C6	C4	6F	3C	46
00000183	86	CB	CD	DF	C6	7C	42	A3	CF	2A	DD	81
000001ae	B4	BE	69	03	48	63	55	C8	F7	D8	E5	32
000001d9	BD	67	F5	6F	F2	C5	14	08	71	93	ED	E9
00000204	01	72	2D	16	DF	73	62	5D	AD	54	62	C1

In the bless file, one byte is changed.

1. Using AES-128-ecb

```
PES1UG20CS243:Mahika~.../Labsetup$>openssl enc -aes-128-ecb -e -in enc.bin  
-out err_dec.txt -K 00112233445566778899aabbcdddeeff  
PES1UG20CS243:Mahika~.../Labsetup$>cat err_dec.txt  
0000t#0mI0S0g000M0000k@  
o3gN000_]0o000Mn      0M000t000PV00t!e+0\00s00Cy000r:0;L0b0f1H'0  
000y00u0h0000C00000E)0,#|000<000>0?0)0j000A]fd000 c36b,0h3|0  
000J0Z0000S0m0n0J00000000PI0H]$0襲00007`  
0x0000i00N0o000^0I0(000Z290ebT0D0?z00<0I+0j0000000q002000000審/00`000$0:  
?00{;0X076000[R"0000C=0X0s0x00S0Ga00`0S0&30y00<000     0000/00A030^00e00%0  
L  
0v[000050000w00F+D0K0V9Vy]>!000ffF\0600000_00000Q00000N0Å0000000d00000;n600  
0'0T_0"E0a+0I00/P000h-0c000GÉ07C- d01[00050 N000r0x0E00  
0          W0N  
        oa*eo  
        @u  
W|0500  
i00000l0d0>r0X0x00>0{0000/^00Ud0"R0N+'0[M80]@Pal0'00000+iaq0_80000000Z000y00  
0V0000S$^0I-0060u00000a0%xsM  
0#0'00lh  
J0%1c0X0&000y`0R0^N0?V0  
?0p>0X000000000000  
n0x0e00070055[0[00e^00#J000X0y{P00H00K00)00600R000  
_s00040s00  
t0D  
鍤 )LJ0#iw0$00A000&004)e0St0G00E090000S00  
3:Mahika~.../Labsetup$>|
```

We can see that the text is not recoverable on corrupting the encrypted file.

2. Using AES-128-cbc:

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cbc -e -in enc.bin  
-out err_dec.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708  
hex string is too short, padding with zero bytes to length  
PES1UG20CS243:Mahika~/.../Labsetup$>cat err_dec.txt  
F0`M0\00d$0+))00!0B0?0J00  
0/07'F00000F0A0000k0`000=0ZQ0000000=0tU0~-00000b000Z00000K0~0o0000@00p0H  
j00=00F0LB0FNR";01{$00_Gp0@0ai000瘞*000%0+00J0Q0F0_0!J.00W000aA0%TS0cj000  
0'|?10_5<:m5x0{100l00e0000sl0%  
{Z0090Y0800,807  
00K0iS0S00?o0!00300R000\1gj000_00  
|aJ00K5;0~00H00_0Gr0}00k0M00n000D$  
S0h0R0o090;0*H0Tu#00%000j0I00E0000j000;0T80*0000[00d/000Σ00HD%j0!0\050ZQm~  
00*00P0j0U^[F0g'RY SWU00200$0MK00Nu5^#000Q00w000000L*f0S00Z  
00n0K004000u00b70  
u)0l0`%-0q000;0T0&0p0-0f0000(b0/0f0q00t0e000Z000A000"003Y0v00&!0·0?0-0000}0`  
0,P0000Qf0000W-0$000`00H0h,0;00n^000?:4|00D000>00+H00000U$0`W0000  
4x0{00'0"  
U060Y  
000o000&~0.0"0m0000W000o0o090  
000000&(90$$q)003R7W?0]00D000ulCQ00Gk0$0  
h_  
T0x_0600:0g00gWV0D_"0000c<j00j0o~00?0000c009PES1UG20CS243:Mahika~/.../Labse  
tup$>
```

Similar to ecb, the text is not recoverable.

3. Using AES-128-cfb:

```

PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cfb -e -in enc.bin
-out err_dec.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
PES1UG20CS243:Mahika~/.../Labsetup$>cat err_dec.txt
Lnrem ipsum dolo2ψ0y000^00IC03?000.0"0.0^0Iql{0I0002u]000媽0e000>00,j0^000000
u0o00T
0m00h00($Y0[ ,0#00<n0Vmnn00001cT' }00L0=0h000c.00&00)00h0!000quπ0C00`0IL0
00;-00000&0b000S0[zms0000meFu00007=V0+T0E/0N0ob
                                                @00GN00;0002000K000Hü 000Ж
00.00p-0ca0cL20\00"0>خr0j0^AE00(~000C=00
b00Gq0x0
    l).ç000      0000ak00
0[qT000 00(00001&00000B010,0"000d_0,u00000&m0'V00°000C>000tE0K0~00xw:I]0600
0r0000I0p000V+0006j00000L00p00{K000D00w-000600^ 0G030J90bB0f]yF
}xCA0E0077200B0000
c?:p:w0yT005
000000,em0000`0000P0k0j0諾 "0200F000000k000yc0š;0q/00m00n0<0%07ج0SĐ13_0,0-
000000`00H00000gm0q0W(-0.0d00050u0CP00'00m.000R0000X0A00@0I|yI?0/0$007F0+0
000Fg0Z00}010n00a'0xp50;0Y0e0d0X0,0E00u0Js_K00+d0=0fG00000P050VQ0xg06606df00
00200)0l0(0v=d
          0~X00$00F0200y00J|ES0^000?;0a!00h000i000d040c000C 0{0. 0H0om|
0
0A)006000/0I00000!0~0N0]0K0cD0 0$'00m;Z ?0PES1UG20CS243:Mahika~/.../Labset
up$>■

```

Only a few characters are discernible and recoverable.

4. Using AES-128-ofb:

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-ofb -e -in enc.bin  
-out err_dec.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708  
hex string is too short, padding with zero bytes to length  
PES1UG20CS243:Mahika~/.../Labsetup$>cat err_dec.txt  
Lnrem ipsum dolor sit amet, consectetur adipiscing elit. Suspendisse in dui  
et velit dictum dignissim. Suspendisse placerat erat ac ipsum tempor, id ves  
tibulum urna bibendum. Cras placerat vehicula sem, ut ultrices felis. Etiam  
congue posuere mi, sed tincidunt lorem euismod sit amet. Phasellus congue ma  
ssa ac metus tempor sollicitudin. Aenean cursus massa at quam suscipit vulpu  
tate. Nunc eget bibendum lorem. Sed eros libero, pulvinar ut dolor eu, cursu  
s volutpat tellus. Vestibulum et neque condimentum, pharetra velit sodales,  
rhoncus enim. Sed rhoncus sapien sed facilisis interdum. Nunc suscipit liber  
o a ipsum gravida sollicitudin. Nulla turpis risus, ornare sed nunc nec, fin  
ibus venenatis ipsum. Donec congue malesuada congue. Praesent mollis consequ  
at nisi sed pretium. Donec a lacus ut neque convallis tincidunt.
```

```
Sed interdum, nunc eu hendrerit semper, lectus neque gravida dui, a blandit  
lorem dolor et magna. Suspendisse luctus consequat urna ut venenatis. Nunc q  
uis rutrum eros, in cras.
```

```
PES1UG20CS243:Mahika~/.../Labsetup$█
```

**Using ofb, it is almost completely
recoverable, there are only minor changes.**

Task 5: Initial Vector (IV) and Common Mistakes:

PES1UG20CS243:Mahika~/.../Labsetup\$ cat plain.txt
Hi this is a message from Mahika!
PES1UG20CS243:Mahika~/.../Labsetup\$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher.bin \-K 00112233445566778889aabccddeff \-iv 0102030405060708
hex string is too short, padding with zero bytes to length
PES1UG20CS243:Mahika~/.../Labsetup\$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher1.bin \-K 00112233445566778889aabccddeff \-iv 0102030405060708
hex string is too short, padding with zero bytes to length
PES1UG20CS243:Mahika~/.../Labsetup\$ cat cipher.bin
500`U!00.m00P0000060 u00f000R*f >a002-}000TPES1UG20CS243:Mahika~/.../Labsetup\$>^C
PES1UG20CS243:Mahika~/.../Labsetup\$ cat cipher1.bin
500`U!00.m00P0000060 u00f000R*f >a002-}000TPES1UG20CS243:Mahika~/.../Labsetup\$>

```
PES1UG20CS243:Mahika~/.../Labsetup$xxd cipher.bin  
00000000: 35d8 de60 5521 a798 2e6d 9e95 5030 93db 5..`U!...m..P0..  
00000010: 14d6 e436 a309 75c4 f066 f783 efdd 0252 ...6..u..f.....R  
00000020: 2ad2 9320 3e61 a9ff ba32 2d7d f8d6 ec54 *.. >a...2-}....T  
PES1UG20CS243:Mahika~/.../Labsetup$xxd cipher1.bin  
00000000: 35d8 de60 5521 a798 2e6d 9e95 5030 93db 5..`U!...m..P0..  
00000010: 14d6 e436 a309 75c4 f066 f783 efdd 0252 ...6..u..f.....R  
00000020: 2ad2 9320 3e61 a9ff ba32 2d7d f8d6 ec54 *.. >a...2-}....T  
PES1UG20CS243:Mahika~/.../Labsetup$
```

Here we can see that on using the same iv value, we produce the same cipher text.

This can reveal some information about the cipher text to

the attacker.

Hence changing the iv value is important.

```
PES1UG20CS243:Mahika~/.../Labsetup$>openssl enc -aes-128-cbc -e -in plain.txt -out cipher2.bin \-K 00112233445566778889aabcccddeeff \-iv 0102030405060709
hex string is too short, padding with zero bytes to length
PES1UG20CS243:Mahika~/.../Labsetup$>xxd cipher.bin
00000000: 35d8 de60 5521 a798 2e6d 9e95 5030 93db  5..`U!....m..P0..
00000010: 14d6 e436 a309 75c4 f066 f783 efdd 0252  ...6..u..f.....R
00000020: 2ad2 9320 3e61 a9ff ba32 2d7d f8d6 ec54  *... >a...2-}....T
PES1UG20CS243:Mahika~/.../Labsetup$>xxd cipher2.bin
00000000: 82f4 575b 6f1f fd4a 0dc1 aab3 5ded 05a7  ..W[o...J....]...
00000010: 28be 9787 f4f1 c359 561e 9700 25f8 aead  (.....YV...%...
00000020: alfa a6b9 f4f1 407c 6573 a39d 5b4c 417f  .....@|es..[LA.
PES1UG20CS243:Mahika~/.../Labsetup$>
```

Here we can see that on changing the iv value, we produce 2 different cipher texts.

```
GNU nano 4.8          sample_code.py          Modified
#!/usr/bin/python3

# XOR two bytearrays
def xor(first, second):
    return bytearray(x^y for x,y in zip(first, second))

MSG = "This is a known message!"
HEX_1 = "a469b1c502c1cab966965e50425438e1bb1b5f9037a4c159"
HEX_2 = "bf73bcd3509299d566c35b5d450337e1bb175f903fafc159"

# Convert ascii string to bytearray
D1 = bytes(MSG, 'utf-8')

# Convert hex string to bytearray
D2 = bytearray.fromhex(HEX_1)
D3 = bytearray.fromhex(HEX_2)

r1 = xor(D1, D2)
r2 = xor(D2, D3)
r3 = xor(D2, D2)
print(r1.hex())
print(r2.hex())
print(r3.hex())

^G Get Help      ^O Write Out      ^W Where Is      ^K Cut Text      ^J Justify
^X Exit          ^R Read File      ^\ Replace       ^U Paste Text     ^T To Spell
```

```
PES1UG20CS243:Mahika~/.../Labsetup$>cd Files
PES1UG20CS243:Mahika~/.../Files$>ls
body_cbc      freq.py      new_ecb.bmp  pic_original.bmp
body_ecb      header      pic_cbc.bmp  sample_code.py
ciphertext.txt new_cbc.bmp  pic_ecb.bmp  words.txt
PES1UG20CS243:Mahika~/.../Files$>nano sample_code.py
PES1UG20CS243:Mahika~/.../Files$>python3 sample_code.py
f001d8b622a8b99907b6353e2d2356c1d67e2ce356c3a478
1b1a0d165253536c0055050d07570f00000c0000080b0000
000000000000000000000000000000000000000000000000000000000000000
PES1UG20CS243:Mahika~/.../Files$>
```

The 2 cipher texts are XOR-ed. Since the XOR produces 0's, we can conclude that the same plaintext produces the same cipher texts.

Task 7: Programming using the Crypto Library :

```
PES1UG20CS243:Mahika~/.../Labsetup$>gcc code.c -o findkey -lcrypto  
PES1UG20CS243:Mahika~/.../Labsetup$>./findkey  
Got key: median  
PES1UG20CS243:Mahika~/.../Labsetup$>■
```

Here the words.txt file is searched by brute force to find the key which is used for this encryption.