

Symmetric Key Encryption Lab

4.2 Task 1.1: Encryption Mode – ECB vs. CBC

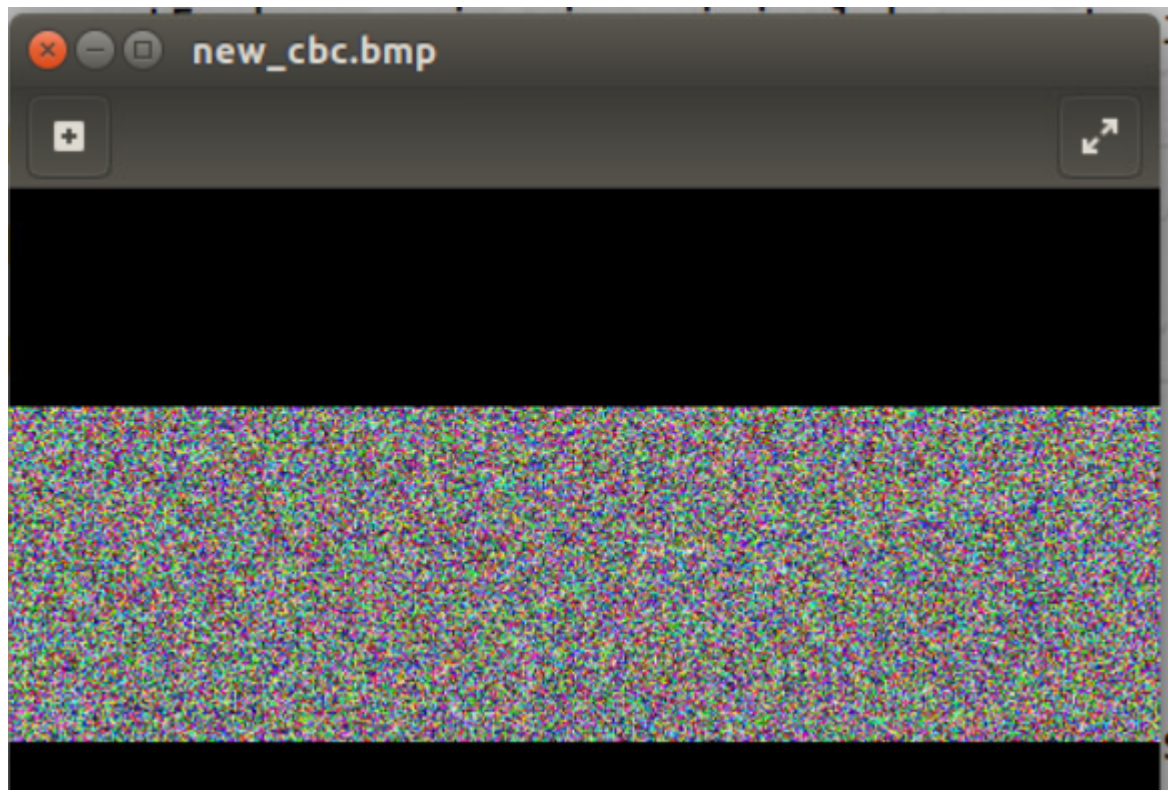
Now we are encrypting a picture with two different ciphers.

- 1) ECB
- 2) CBC

Original picture:



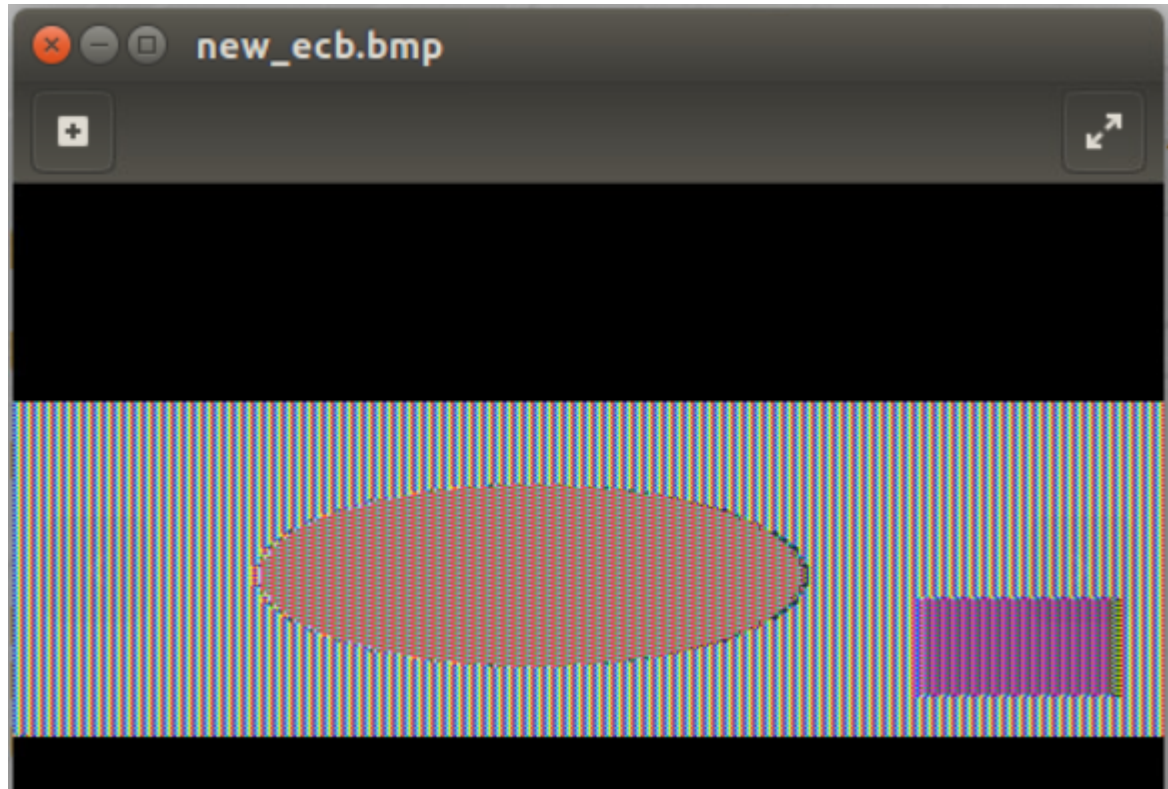
Encryption mode :- CBC:



The CBC encryption algorithm has completely changes the original file.

In this algorithm we cant derive any use full information. Because it encrypts all the byte in the picture

Encryption mode:-ECB:



In ECB the picture is similar to original picture.

We can guess this simple what the picture it was.

Yes, we can see the information here. Even it is encrypted.

4.3 Task 2: Encryption Mode – Corrupted Cipher Text

Here first we are taking plane text file, And after encrypting the data we are in introducing an error in the 30th byte from the file in encrypted file.

For introducing a error we are using “Bless command”

Encryption using AES-128-CBC :

Original File

The screenshot shows the Bless tool interface for the file 'en_text_cbc.txt'. The hex dump displays the following data:

Offset	Hex	ASCII
00000000	53 61 6C 74 65 64 5F 5F F3 6F 2F 16 97 F0 23 82 C6 27	Salted_.o/...#...'
00000012	F3 73 FC B9 75 BE B9 6C 3A B1 20 15 DE DB 91 2C 67 2F	.s..u..l:.,g/
00000024	1C 71 45 E2 8D 2F AF 9E 92 39 08 0B BC 8A 3B ED FA 30	.qE../...9....;..0
00000036	B9 DB A0 7F F3 1A F9 4D 66 20 97 FF C2 53 54 81 36 31Mf ...ST.61
00000048	E4 D7 46 A1 90 C9 91 AD E0 34 B9 53 6F 11 6F 04 03 3F	..F.....4.So.o..?
0000005a	12 77 77 A0 17 C9 A9 EB 7E 94 38 C4 BE D5 D4 17 47 CE	.ww.....~.8.....G.
0000006c	7C 91 FE 99 9F 38 CA D1 2D B7 60 38 6A A8 24 05 8D 8F8..-.`8j.\$...
0000007e	0C BF	..

Below the hex dump, various conversion options are available:

- Signed 8 bit: 83
- Unsigned 8 bit: 83
- Signed 16 bit: 21345
- Unsigned 16 bit: 21345
- Offset: 0x0 / 0x7f

Error File

The screenshot shows the Bless tool interface for the file 'en_text_cbc_error.txt'. The hex dump displays the following data:

Offset	Hex	ASCII
00000000	53 61 6C 74 65 64 5F 5F F3 6F 2F 16 97 F0 23 82 C6 27	Salted_.o/...#...'
00000012	F3 73 FC B9 75 BE B9 6C 3A B1 20 15 FF DB 91 2C 67 2F	.s..u..l:.,g/
00000024	1C 71 45 E2 8D 2F AF 9E 92 39 08 0B BC 8A 3B ED FA 30	.qE../...9....;..0
00000036	B9 DB A0 7F F3 1A F9 4D 66 20 97 FF C2 53 54 81 36 31Mf ...ST.61
00000048	E4 D7 46 A1 90 C9 91 AD E0 34 B9 53 6F 11 6F 04 03 3F	..F.....4.So.o..?
0000005a	12 77 77 A0 17 C9 A9 EB 7E 94 38 C4 BE D5 D4 17 47 CE	.ww.....~.8.....G.
0000006c	7C 91 FE 99 9F 38 CA D1 2D B7 60 38 6A A8 24 05 8D 8F8..-.`8j.\$...
0000007e	0C BF	..

Below the hex dump, various conversion options are available:

- Signed 8 bit: -37
- Unsigned 8 bit: 219
- Signed 16 bit: -9327
- Unsigned 16 bit: 56209
- Offset: 0x1f / 0x7f

Output after corrupting 30th byte

```
-0`8j0$00[09/05/21]seed@SEED:~$ cat de_text cbc error.txt  
00000000 D0Z$00u0 ssignment and Uhis is Lakshmi Sai Tejaswi Pathuri. I'm from wichita state university.
```

Encryption using AES-128-ECB

Original File

The screenshot shows the Bless tool interface for the file 'en_text_ecb.txt'. The hex dump displays the first 80 bytes of the file, which correspond to the ASCII text 'Salted__..b..N....f2....g@.Z.i|..q).eX.....nAm.\..3....._g....6.._...x.9.'.....3\^.{.....[...ZoFQ.<...[.W.ZX..x D.'. The byte at offset 0x0012 (index 18) is highlighted in red, indicating it is the 30th byte (index 29) of the file. The tool also shows various numerical representations of the selected byte (0x53) and the entire selected range (0x53 to 0x7F).

Offset	Hex	ASCII
00000000	53 61 6C 74 65 64 5F 5F EC 10 62 B8 A8 4E FE E2 14 A4	Salted__..b..N....
00000012	D8 66 32 A7 98 19 B4 67 40 D9 5A E9 69 7C CA 7F 71 29	.f2....g@.Z.i ..q)
00000024	D7 65 58 F5 DA F2 81 17 09 C2 A4 86 6E 41 6D 8C 5C D4	.eX.....nAm.\.
00000036	C9 9D 33 04 A2 E8 14 CE D0 5F 67 DB D5 AC 7F 36 16 DF	..3....._g....6..
00000048	08 B7 5F F5 D0 B9 78 06 39 C8 27 86 E1 AA A8 B5 00 84	.._...x.9.'.....
0000005a	FF 8E 15 9C 33 5C 5E 90 7B C4 B4 10 81 D1 5B DC FF CA3\^.{.....[...
0000006c	E7 5A 6F 46 51 C1 3C 02 B2 5B BC 57 BE 5A 58 CC F1 78	.ZoFQ.<...[.W.ZX..x
0000007e	44 CD	D.

Selected range: 0x53 to 0x7F

Offset: 0x0 / 0x7f Selection: None INS

Error File

The screenshot shows the Bless tool interface for the file 'en_text_ecb.txt*'. The hex dump displays the first 80 bytes of the file. The byte at offset 0x0012 (index 18) is highlighted in red, indicating it is the 30th byte (index 29) of the file. The tool also shows various numerical representations of the selected byte (0x7C) and the entire selected range (0x7C to 0x9F).

Offset	Hex	ASCII
00000000	53 61 6C 74 65 64 5F 5F EC 10 62 B8 A8 4E FE E2 14 A4	Salted__..b..N....
00000012	D8 66 32 A7 98 19 B4 67 40 D9 5A E9 52 7C CA 7F 71 29	.f2....g@.Z.R ..q)
00000024	D7 65 58 F5 DA F2 81 17 09 C2 A4 86 6E 41 6D 8C 5C D4	.eX.....nAm.\.
00000036	C9 9D 33 04 A2 E8 14 CE D0 5F 67 DB D5 AC 7F 36 16 DF	..3....._g....6..
00000048	08 B7 5F F5 D0 B9 78 06 39 C8 27 86 E1 AA A8 B5 00 84	.._...x.9.'.....
0000005a	FF 8E 15 9C 33 5C 5E 90 7B C4 B4 10 81 D1 5B DC FF CA3\^.{.....[...
0000006c	E7 5A 6F 46 51 C1 3C 02 B2 5B BC 57 BE 5A 58 CC F1 78	.ZoFQ.<...[.W.ZX..x
0000007e	44 CD	D.

Selected range: 0x7C to 0x9F

Offset: 0x1f / 0x7f Selection: None INS

Output after corrupting 30th byte

```
5/21]seed@SEED:~$ cat de_text_ecb_error.txt  
000q%7yj~00#0Gssignment and this is Lakshmi Sai Tejaswi Pathuri. I'm from wichita state university.
```

DELIVERABLES FOR “ECB “AND “CBC”

ECB

- Here in ECB, I can derive the information from 30th byte
- ECB is basic encryption, but the data can be guessed easily.
- ECB encrypt the data by byte by byte

CBC

- Here in CBC, I can derive the information from 30th byte and in the middle, there is a word corrupted
- CBC is advance level encryption, it is very hard to guess the data
- CBC encrypt the data with reference to previous encrypted information.