

Secret-Key Encryption Lab

Lab Environment

```
[09/04/24]seed@VM:~/.../Labsetup$ docker-compose -f docker-compose.yml build
Building oracle-server
Step 1/8 : FROM handsonsecurity/seed-ubuntu:dev AS builder
dev: Pulling from handsonsecurity/seed-ubuntu
da7391352a9b: Pulling fs layer
14428a6d4bcd: Pulling fs layer
da7391352a9b: Extracting [>] 294.9kB/28.56M
B28.17MB/28.56MBwnload complete
da7391352a9b: Extracting [====>] 2.359MB/28.56M
Bd39fdffe330: Download complete
da7391352a9b: Extracting [=====>] 4.129MB/28.56M
Bbb168ce59cc: Download complete
da7391352a9b: Pull complete
14428a6d4bcd: Pull complete
2c2d948710f2: Pull complete
5d39fdffe330: Pull complete
56b236c9d9da: Pull complete
1bb168ce59cc: Pull complete
588b6963c007: Pull complete
c3c83e840346: Pull complete
Digest: sha256:f30e4224cf90ab83606285e4e1b12ef3879d3f6ec24aee991c00aeb52295551
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:dev
---> 89212aee292b
Step 2/8 : COPY . /oracle
---> 6bb3c291da94
Step 3/8 : WORKDIR /oracle

---> Running in 2c8f110821c1
Removing intermediate container 2c8f110821c1
---> e7dbd1181aca
Step 4/8 : RUN make
---> Running in 27349914174d
da7391352a9b: Already exists
14428a6d4bcd: Already exists
2c2d948710f2: Already exists
5d39fdffe330: Already exists
56b236c9d9da: Already exists
1bb168ce59cc: Already exists
588b6963c007: Already exists
small: Pulling from handsonsecurity/seed-ubuntu
Digest: sha256:53d27ec4a356184997bd520bb2dc7c7ace102bfe57ecfc0909e3524aabf8a0be
Status: Downloaded newer image for handsonsecurity/seed-ubuntu:small
---> 1102071f4a1d
Step 6/8 : COPY --from=builder /oracle/build /oracle
---> 2acf14da4c3f
Step 7/8 : WORKDIR /oracle
---> Running in d3fc16d846de
Removing intermediate container d3fc16d846de
---> 490334c3b96d
Step 8/8 : CMD ./server
---> Running in 1ce714e0fdc1
Removing intermediate container 1ce714e0fdc1
---> dcc805874ca8

Successfully built dcc805874ca8
Successfully tagged seed-image-encryption:latest
```

```
[09/04/24]seed@VM:~/.../Labsetup$ docker-compose up
Creating network "net-10.9.0.0" with the default driver
Creating oracle-10.9.0.80 ... done
Attaching to oracle-10.9.0.80
oracle-10.9.0.80 | Server listening on 3000 for known_iv
```

Task 2: Encryption using Different Ciphers and Modes

First, I created plain.txt with text - "My name is Malithi"

- Cipher type = -aes-128-cbc

```
[09/06/24]seed@VM:~/.../Labsetup$ openssl enc -aes-128-cbc -e -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
```

Cipher text:

```
1 xbnRŠ0ì"yÓãçm/$ÇLA?PÎi
```

- Cipher type = -bf-cbc

```
[09/06/24]seed@VM:~/.../Labsetup$ openssl enc -bf-cbc -e -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Cipher text:

```
1 FãûE-V;Đ3+Ÿ* >VQ0|¥k
```

- Cipher type = -aes-128-cfb

```
[09/06/24]seed@VM:~/.../Labsetup$ openssl enc -aes-128-cfb -e -in plain.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Cipher text:

```
Êÿ~K RßñÆíP|5P!
```

Task 3: Encryption Mode – ECB vs. CBC

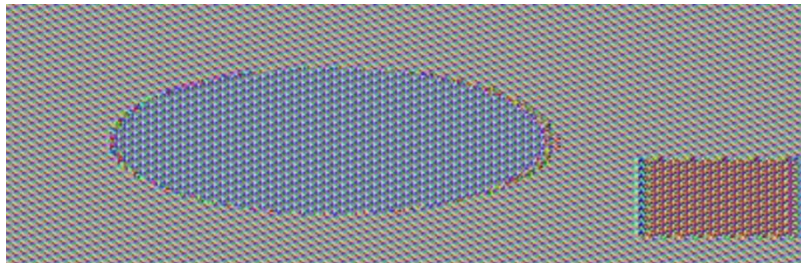
Encrypt the file using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining

```
[09/06/24]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out encrypt_
ecb.bmp -K 00112233445566778889aabbccddeeff
[09/06/24]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in pic_original.bmp -out encrypt_
cbc.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/06/24]seed@VM:~/.../Files$ head -c 54 pic_original.bmp > header_ecb
[09/06/24]seed@VM:~/.../Files$ tail -c +55 encrypt_ecb.bmp > body_ecb
[09/06/24]seed@VM:~/.../Files$ cat header_ecb body_ecb > new_ecb.bmp
[09/06/24]seed@VM:~/.../Files$ head -c 54 pic_original.bmp > header_cbc
[09/06/24]seed@VM:~/.../Files$ tail -c +55 encrypt_cbc.bmp > body_cbc
[09/06/24]seed@VM:~/.../Files$ cat header_cbc body_cbc > new_cbc.bmp
[09/06/24]seed@VM:~/.../Files$ █
```

Original image

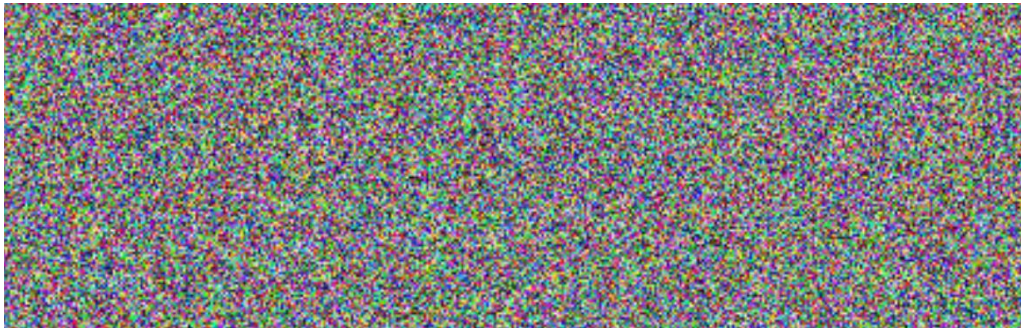


Encrypted image using ECB



Still, we can identify the original data from the encrypted image. That means we can see the patterns in the plaintext in the ciphertext. This is revealing information to an attacker. ECB works well with random strings.

Encrypted image using CBC



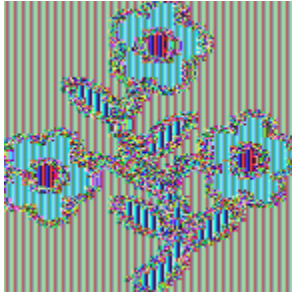
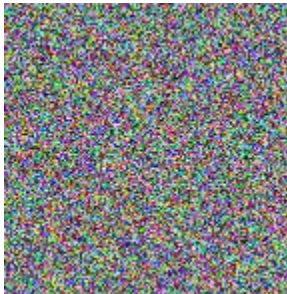
In CBC plaintext is combined with the previous ciphertext block, identical plaintext blocks produce different ciphertext blocks, which hide patterns in the plaintext. So, we can clearly see difference in original image and the encrypted image.

Experiment with new one

```
[09/06/24]seed@VM:~/.../Files$ openssl enc -aes-128-ecb -e -in index.bmp -out encrypt_ecb.bmp
-K 00112233445566778889aabbccddeeff
[09/06/24]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in index.bmp -out encrypt_cbc.bmp
-K 00112233445566778889aabbccddeeff -iv 0102030405060708
hex string is too short, padding with zero bytes to length
[09/06/24]seed@VM:~/.../Files$ head -c 54 index.bmp > header_ecb
[09/06/24]seed@VM:~/.../Files$ tail -c +55 encrypt_ecb.bmp > body_ecb
[09/06/24]seed@VM:~/.../Files$ head -c 54 index.bmp > header_cbc
[09/06/24]seed@VM:~/.../Files$ tail -c +55 encrypt_cbc.bmp > body_cbc
[09/06/24]seed@VM:~/.../Files$ cat header_ecb body_ecb > new_ecb.bmp
[09/06/24]seed@VM:~/.../Files$ cat header_cbc body_cbc > new_cbc.bmp
[09/06/24]seed@VM:~/.../Files$
```

Original image



Encrypted image using ECB**Encrypted image using CBC****Task 5: Error Propagation – Corrupted Cipher Text**

1. Create a text file that is at least 1000 bytes long.
2. Encrypt the file using the AES-128 cipher.
3. Unfortunately, a single bit of the 55th byte in the encrypted file got corrupted. You can achieve this corruption using the bless hex editor.
4. Decrypt the corrupted ciphertext file using the correct key and IV.

Encrypt test file using AES-128 ECB

```
[09/11/24]seed@VM:~/.../Labsetup$ openssl enc -aes-128-ecb -e -in test.tex -out output.bin  
-K 00112233445566778889aabbccddeeff
```


Name: Malithi	Index Number: 856558563	Date: 9/12/2024
---------------	-------------------------	-----------------

Corrupt 55th (0A) byte of output. bin as 00 using bless

```

00000000 33 31 15 09 32 2A A5 DF B4 3F E9 1D 84 6E C0 B7 31..2*...?...n..
00000100 32 85 23 76 33 D1 98 02 AE 18 21 99 AF 2D 02 EC 2.#v3.....l..-..
00000200 7B F9 6D A1 7A D9 EC EC A1 81 65 73 04 D6 6C 97 {.m.z.....es..l.
00000300 0C 34 DC E7 39 BF 00 C4 46 C9 04 AD 52 87 43 9A .4..9...F...R.C.
00000400 07 66 AC E9 52 7C 76 E6 55 4F 7F E6 C4 34 CC 23 .f..R|v.U0..4.#
00000500 F7 1A 9D 5A 9D 9A F1 26 C7 BA C0 57 0D E9 A2 DF ...Z...&...W....
00000600 D7 BD FA B2 C5 9A AC BA 46 28 AC 8C 05 94 B5 4D .....F(.....M
00000700 3B 14 44 E4 F9 D2 EE 75 6C A8 56 3D 87 2C B9 47 ;.D....u\l.V=.,.G
00000800 47 F3 EF 41 B3 45 D7 70 FC 02 6F 0C AD 00 C7 A7 G..A.E.p..o.....
00000900 28 90 DD 44 96 26 46 3E 92 5E 6C 8B A9 1B C6 A3 (.D.&F>.^l.....
00000A00 36 85 0B 3E B4 36 73 91 12 B6 B7 CF 3B BD CD 11 6..>.6s.....;...
00000B00 D2 3B C2 DE 0D B5 71 70 FA 2D A4 42 47 08 A2 C3 .;....qp.-.BG...
00000C00 E5 8D E3 CF 8B 69 47 77 54 2A 44 81 99 6B 6D AB .....iGwT*D..km.
00000D00 75 CE B3 F8 3E E9 BD A6 EE 63 6E 57 ED A9 91 34 u...>....cnW...4
00000E00 7E 63 B4 B7 DC 94 AA F0 6B 1C C4 D3 29 C8 4D 6B ~c.....k...).Mk
00000F00 CE B1 D5 91 EB 07 61 DF 28 0B CE 24 6C 5C C3 72 .....a(..$l\..r
00001000 5E DC AB 97 95 2A 0A 03 0A 66 A1 5E 26 26 33 09 ^.....*...f.^&&3.
00001100 C0 CD FC 84 20 80 7E 23 3F 8A AC A0 4A 83 75 82 .....~#?...J.U.
00001200 79 0C 77 7E 15 22 76 CA 77 FE 06 96 0B 0B E4 DF y.w~."v.w.....
00001300 C7 B9 46 89 4B 24 30 1A D9 AE D6 CE 68 0A 76 B5 ..F.K$0.....h.v.
00001400 CB 3D B6 2D CE CD 08 61 CA FD D4 E7 B0 41 A7 CE .=-...a.....A..
00001500 49 5E F7 FE 35 CA 73 8E 67 DF 20 0D 9E 59 54 66 I^..5.s.g....YTf
00001600 84 4D 7E F5 20 3C 70 51 08 0B F9 99 64 8C 41 61 .M~..<pQ.....d.Aa
00001700 A6 8F 00 6E AF 39 2A 0B AD 9F B3 E1 5D 40 EB E5 ...n.9*.....]@..
00001800 2B 4A 81 DD 11 E4 55 AF 80 D7 D0 79 E1 28 4D DA +J....U....y.(M.

```

Decrypt the file output.bin which is encrypted by AES-128 ECB

```
[09/11/24]seed@VM:~/.../Labsetup$ openssl enc -aes-128-ecb -d -in output.bin -out decrypted.txt -K 00112233445566778889aabbccddeeff
```

Original text

Independence Day, known colloquially as the Fourth of July, is a federal holiday in the United States which commemorates the ratification of the Declaration of Independence by the Second Continental Congress on July 4, 1776, establishing the United States of America.

After decrypting - ECB

[Independence Day, known colloquially as the Four00#0000Žy0tA¹€{!+35 federal holiday in the United States which commemorates the ratification of the Declaration of Independence by the Second Continental Congress on July 4, 1776, establishing the United States of America.

After decrypting using ECB, it can't be able to recover the whole message. It has some changes in "th of July is a".

Encrypt test file using AES-128 CBC

```
[09/11/24]seed@VM:~/.../Labsetup$ openssl enc -aes-128-cbc -e -in test.txt -out output1.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Corrupt 55th (EC) byte of output1. bin as 00 using bless

Name: Malithi

Index Number: 856558563

Date: 9/12/2024

```
000000 15 5D 83 29 80 99 84 5B 7F D0 D2 2B F9 0A 15 22 .].)...[.+.+"
000010 16 E5 59 F5 F1 98 3A 07 52 4C 78 45 0A 10 3E B9 ..Y....:RLxE...>
000020 3E A2 8C D3 A2 1C A5 9D D1 B6 7F BA D6 63 52 29 >.....cR)
000030 2A F8 0B A1 81 42 00 2E CD C0 58 BF 8D 05 27 6F *...B....X... 'o
000040 2F 16 86 59 5C 14 8B 84 F0 A4 E0 22 52 4F 26 F9 /.Y\....."RO&
000050 E1 74 42 0D A4 2B 40 C4 4F A2 60 7C 77 D2 AF 26 .tB...+@.O.`|w...&
000060 CD 2B FD 4B 0D DE 36 27 E6 20 54 2B C9 1E EF 2D .+.K..6!.T+...-
000070 33 D0 73 DB AC FF 20 55 BE 11 CA 11 D4 B8 41 4B 3.s....U.....AK
000080 D0 41 48 F5 27 2B 21 20 07 D5 F2 EF 79 DC AE 21 .AH.'+l.....y..!
000090 BC 70 8E 90 F7 6D AF A2 40 FB 29 3C CF 19 30 90 .p...m..@.)<..0.
0000A0 9F 4F F1 F9 14 1D 90 CB 50 18 D0 7C 58 C4 3B 98 .O.....P..|X.;.
0000B0 4E F1 0B BE 20 20 AC D0 F0 E8 56 18 C1 1C 4B D8 N.....V...K.
0000C0 98 D5 DC 22 19 3C 58 2D 06 3C 32 CA EC 03 80 AF ...<X-<2....
0000D0 6A CE 2C ED 36 CF 90 6D C7 01 AF EC 40 6A 34 36 j.,.6..m....@j46
0000E0 BC 7E 1C BE 84 77 21 9B 41 49 42 2D 59 02 26 59 ~...w|.AIB-Y.&Y
0000F0 9E F6 2C DE 6E 5C E5 98 67 AF C4 77 AA 1D D3 CA ...n\..g..w....
000100 57 7F 8C 10 7D 7D 0A 49 EA 83 A4 30 CF BE 4C 48 W..}}.I...0..LH
000110 AD 72 CB 38 B4 38 26 0D F2 12 2A 89 79 FA 16 CF .r.8.8&...y...
000120 60 2F EE EC A2 E2 DE 07 A0 2E D2 55 A4 B8 10 21 `/. ....U...!
000130 A9 C8 4A 96 0F 6F 98 56 2E D5 DE EA 24 24 DC 92 ./...o.V...$$..
000140 2F E4 BC 7F 1C EF B7 23 5A 60 12 0C 36 0B 56 51 /...#Z`..6.VQ
000150 F9 48 08 DE D4 A4 26 CA 88 E4 C3 AF D1 8B 08 50 .H..J&.....P
000160 F4 A9 98 2D 7D 8D CE BA B1 3D EA A4 25 A8 17 6B ...-}....=..%.k
000170 5F EE 07 9F 09 EF D5 10 B1 C4 1E F4 78 77 95 41 _.....xw.A
000180 48 73 C8 46 80 E1 31 B9 29 F2 2D 96 BE 2C CB 75 Hs.F..1.)..-.,.u
```

Decrypt the file output1.bin which is encrypted by AES-128 CBC

```
[09/11/24] seed@VM:~/.../Labsetup$ openssl enc -aes-128-cbc -d -in output1.bin -out decrypte
d1.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

After decrypting – CBC

Independence Day, known colloquially as the Fourth of July, is a federal holiday in the United States which commemorates the ratification of the Declaration of Independence by the Second Continental Congress on July 4, 1776, establishing the United States of America.

After decrypting using CBC, it can't be able to recover the whole message. It has some changes in "th of July is a".

Encrypt test file using AES-128 CFB

```
[09/11/24] seed@VM:~/.../Labsetup$ openssl enc -aes-128-cfb -e -in test.txt -out output2.bin
-K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Corrupt 55th (0B) byte of output1. bin as 00 using bless

```
000000 CE E8 EB 40 B1 5A D4 B5 CA F0 DC 78 55 54 3D 14 ...@.Z.....xUT=.
000010 72 3E 9D CC 2C 42 2F 1F 6D 2C 44 07 4A AA 6F E8 r>...B/.m,D.J.o.
000020 5A E4 7F 6B 33 40 E9 B1 B2 C1 89 4D A5 FA C1 59 Z.k3@.....M...Y
000030 2D 1A 52 D9 93 36 00 55 FE 35 16 A9 34 53 CF 52 -.R..6.U.5..4S.R
000040 A2 39 A4 A8 52 AE 3D 09 C1 FC CC B6 07 75 0C 8E .9..R.=.....u..
000050 DD 81 D7 58 66 77 7C 67 0B 7A 96 70 59 AC EE FC ...Xfw|g.z.pY...
000060 5E 6A 77 3E D8 73 EC 6E 4F D9 D5 54 E9 AE 37 05 ^jw>.s.n0..T..7.
000070 E7 BB 8A C3 8A B3 B8 F7 3F 96 D1 89 DA F2 AC AA .....?.....
000080 07 08 A9 0F 7A 73 BA E5 35 A8 E1 2F 1A E6 9A 6D ....zs..5./...m
000090 9E C8 9E 3F 8B FA 3E D0 08 AD 91 99 5E 84 5D F2 ...?..>.....^..]
0000A0 9F 97 DF 4B B1 6E B3 28 A8 CA 83 96 9D 28 AE 66 ...K.n.(.....(.f
0000B0 EE 11 3E A1 7A F1 1A 30 BC 74 CB 27 09 0F 47 CE ..>.z..0.t.'..G.
0000C0 8B E9 9F EE AA 35 D0 6E CB BA 40 F1 85 04 49 0B .....5.n..@...I.
0000D0 22 0F 4D 1F CF 9A 6D 20 D0 BB 05 C9 A4 C0 2B 9B ".M...m.....+.
0000E0 65 D0 F9 85 25 65 C1 71 52 89 1B F3 35 D5 86 BA e...%e.qR...5...
0000F0 1C 08 2A 86 BD 8E ED 1B FC 26 F4 95 CD 70 60 CB ..*.....&...p'.
000100 D6 26 78 27 76 FC 0B D7 0F 48 61 35 E5 51 EF 08 .&x'v....Ha5.Q..
000110 F6 39 7F 72 D1 96 9D A3 A3 FB 26 83 AE 8D A6 FA .9r.....&.....
000120 15 E1 F5 D1 11 91 BF 05 C0 86 C6 8A 66 DF E2 A4 .....f...
000130 5F E4 0E 40 58 13 90 AE 56 B2 86 8C 5F E1 D2 AB _..@X...V.....
000140 40 1A E8 CA BF 81 8F 4E 67 41 65 E5 70 3A BE 9E @.....NgAe.p:...
000150 61 1E A7 0A 6D 2A 69 AF 80 D0 16 A2 3E 63 FF 6A a...m*1.....>c.j
000160 9C 92 7F 23 18 0C 06 61 97 62 48 07 87 68 1A B6 ..#...a.bH..h..
000170 87 BA C9 A7 AF CB 76 28 8C 4B B0 CA 74 ED 8A 0C .....V(.K..t...
000180 B2 43 9B 89 8F DA DA DB 0A BF A8 27 B5 C5 5F 83 .C.....'.._..
```

Decrypt the file output2.bin which is encrypted by AES-128 CFB

Name: Malithi

Index Number: 856558563

Date: 9/12/2024

```
[09/11/24]seed@VM:~/.../Labsetup$ openssl enc -aes-128-cfb -d -in output2.bin -out decrypte
d2.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

After decrypting – CFB

Independence Day, known colloquially as the Fourth of Auly, is a         g»l             in the United States which commemorates the ratification of the Declaration of Independence by the Second Continental Congress on July 4, 1776, establishing the United States of America.

After decrypting using CFB, it can't be able to recover the whole message. It has some changes in the letter "J" and "federal holiday".

Encrypt test file using AES-128 OFB

```
[09/11/24]seed@VM:~/.../Labsetup$ openssl enc -aes-128-ofb -e -in test.txt -out output3.bin
-K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

Corrupt 55th (AF) byte of output3. bin as 00 using bless

```
00000000 CE E8 EB 40 B1 5A D4 B5 CA F0 DC 78 55 54 3D 14 ...@.Z....xUT=.
00000010 1D 74 F2 69 D4 2C 2C D3 94 37 6A 90 3C 18 3D F6 ..t.i.,,,7j.<.=.
00000020 B4 01 50 18 45 04 2C 75 61 CE 0E A5 7B 52 DC DA ..P.E.,ua...{R..
00000030 7A 4D 92 F5 03 B6 00 53 57 08 4D 3A B6 0D 35 35 zM....SW.M:...55
00000040 5D 14 B8 B4 BD 15 24 94 46 C6 28 8F DA A5 AA 32 ]....$.F.(...2
00000050 3C 31 E0 DF BE 04 16 7E 7C 92 DE 12 0E 78 15 EB <1....-|....x..
00000060 81 3E B6 B4 9A DC 14 75 1E 67 83 8F B0 F9 8A 4D .>....u.g....M
00000070 7C E4 3B A8 33 C5 5A CE DE 3C 73 BF 99 48 46 FE |.j.3.Z.<s...HF.
00000080 6F 57 CE ED 8E AE 7F E3 6F A2 5B F0 7E F5 81 95 oW....O.[.~...
00000090 FA 0B FC 6D 33 33 61 83 E2 83 20 E5 8B 25 56 C9 ...m33a.....%V.
000000A0 B4 EC 9C 1B 7F 15 70 45 56 3D CB CC 10 D4 E2 C8 .....pEV=.....
000000B0 D5 FE E6 C9 E8 FB DD FA 66 24 05 DC 0F E8 3A F8 .....f$......
000000C0 20 60 AE F2 11 60 F6 1E 4B 57 0A 32 DF 2E 81 0A .....KW.2....
000000D0 2F A5 88 54 31 1B BE 8B 6C FB 14 E7 58 AF AC 8E /..T1...l...X..
000000E0 1D FF EB B6 1D B0 E9 47 FD 2D 33 B8 6D BC F1 7A .....G.-3.m..z
000000F0 6A D8 1F CF 74 8F 2F 31 CE D0 29 07 65 1F AB 8B j...t./1..).e...
00000100 12 CA AC 06 A3 0D 0C 02 F0 62 B5 BB 3B 70 3F 22 .....b...;p?..
00000110 95 E0 12 22 32 28 16 53 57 BD 56 32 7B 40 B4 E2 ..."2(.SW.V2[@..
00000120 BA FE E4 09 83 F1 70 16 6C C4 BA 78 21 69 9C A1 .....p.l.x!i..
00000130 12 99 69 B7 2A 9F 34 07 A5 86 9A 0C 96 A5 DD 8A ..i*.4.....
00000140 13 1E C6 5E 95 EA 38 49 D9 BF 05 B9 0D A8 BB AD ...^..8I.....
00000150 96 D1 BE 39 49 57 82 AB A2 B2 B4 1B AE D9 00 5B ...9IW.....[
00000160 C7 7B FC 8F FC E4 18 3F 3F CB 31 47 3C 92 E2 D8 .{.....??16<...
00000170 EF E1 38 D8 30 5F F8 9F 15 A5 9A D4 4F BF E6 40 ..8.0.....0..@
00000180 BE 4F AD A0 86 70 77 89 66 FC 51 FC 68 0A 39 C3 .0...pw.f.Q.h.9.
```

Decrypt the file output3.bin which is encrypted by AES-128 OFB

```
[09/11/24]seed@VM:~/.../Labsetup$ openssl enc -aes-128-ofb -d -in output3.bin -out decrypte
d3.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
```

After decrypting – OFB

Independence Day, known colloquially as the Fourth of   uly, is a federal holiday in the United States which commemorates the ratification of the Declaration of Independence by the Second Continental Congress on July 4, 1776, establishing the United States of America.

By comparing decrypted images from ECB, CBC, CFB, and OFB, OFB was able to recover the original message with only one change in the letter 'J'.

Name: Malithi	Index Number: 856558563	Date: 9/12/2024
---------------	-------------------------	-----------------

Task 6: Initial Vector (IV) and Common Mistakes

Task 6.1. IV Experiment

1. Different IV – Key is 00112233445566778889aabbccddeeff

```
0000000 19 DB 7E D7 EE 1F C1 C0 EA FF 72 8A 15 15 A4 8A  ..~.....r.....
0000010 4C BB 9B 17 FD B5 C4 B2 42 DC 02 75 FB A8 8C 2E  L.....B..u....
```

Key is 00112233445566778889aabbccddeeff

```
0000000 14 B3 2F B2 FD 03 B4 C8 C1 42 59 7F 1D 84 2F B2  ../.....BY../.
0000010 69 C5 22 7D D3 1E B6 BA B6 33 88 D7 05 03 21 69  i."}.....3....!i
```

2. Same IV

Same key and same plaintext provide same ciphertext.

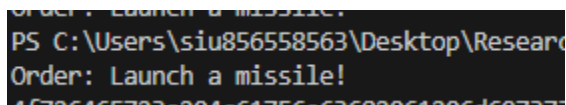
Task 6.2. Common Mistake: Use the Same IV

First get the output stream from the encryption of the first plaintext p1.

Output = p1 XOR c1

Then get p2 by,

P2= Output XOR c2



- Plaintext is = ‘order: Launch a missile!’

If we replace **OFB** in this experiment with **CFB** (Cipher Feedback), only part of the p2 can be revealed but that overlaps with p1.

Task 6.3. Common Mistake: Use a Predictable IV

Attacker Eve knows the message contains either ‘yes’ or ‘no’ and can see the ciphertext and the initialization vector (IV).

P1 – first plaintext, p2- second plaintext, c1-ciphertext for p1, c2- ciphertext for p2, IV first initial vector and IV_NEXT next initial vector.

Name: Malithi	Index Number: 856558563	Date: 9/12/2024
---------------	-------------------------	-----------------

If p1 = "yes"

P2= "yes" XOR IV_NEXT

IV_NEXT generated by using IV and C1. Then IV_NEXT can be used to generate the next plaintext input.

Bob's secret message is either "Yes" or "No", without quotations.

Bob's cyphertext: 54601f27c6605da997865f62765117ce

The IV used: d27d724f59a84d9b61c0f2883efa7bbc

(Hex:6432376437323466353961383464396236316330663238383365666137626263)

Next IV: d34c739f59a84d9b61c0f2883efa7bbc

(Hex: 6433346337333966353961383464396236316330663238383365666137626263)

Your plaintext: 11223344aabbccdd

Your ciphertext: 05291d3169b2921f08fe34449ddc3611

Next IV: cd9f1ee659a84d9b61c0f2883efa7bbc

Your plaintext: <your input>

```

cipher_cons.py > ...
1  #!/usr/bin/python3
2  from sys import argv
3
4  __, first, second, third = argv
5  p1 = bytearray(first, encoding='utf-8')
6  padding = 16 - len(p1) % 16 # padding to match the block size as 128 bit
7  p1.extend([padding]*padding)
8  IV = bytearray.fromhex(second)
9  IV_NEXT = bytearray.fromhex(third)
10 p2 = bytearray(x ^ y ^ z for x, y, z in zip(p1, IV, IV_NEXT))
11 print(p2.decode('utf-8'), end='')

```

```

PS C:\Users\s1u856558563\Desktop>
Ydp
P2=
PS C:\Users\s1u856558563\Desktop>

```

To get the c2

```

[09/12/24]seed@VM:~/.../Files$ openssl enc -aes-128-cbc -e -in p2 -out c2 -K 001
12233445566778899aabbccddeeff -iv 6433346337333966353961383464396236316330663238
383365666137626263

```

C2

```

00000000 F3 83 84 AF B2 5B E0 C2 90 9E 5E B4 F8 21 5C 87 .....[....^...!\.

```

Name: Malithi	Index Number: 856558563	Date: 9/12/2024
---------------	-------------------------	-----------------

To compare

It provides the same c2.

f38384afb25be0c2909e5eb4f8215c87

We can clearly see first generated c2 and after generated one is equal.

After decrypting it generate the same message “yes”

```
12233445566778899aabbccddeeff -iv 6432376437323466353961383464396236316330663238  
383365666137626263
```

1 Yes|

Name: Malithi	Index Number: 856558563	Date: 9/12/2024
---------------	-------------------------	-----------------

Task 7: Programming using the Crypto Library

Plaintext (total 21 characters): This is top secret.

Ciphertext (in hex format): 764aa26b55a4da654df6b19e4bce00f4
ed05e09346fb0e762583cb7da2ac93a2

IV (in hex format): aabbccddeeff00998877665544332211

Following code is used to find the key from words.txt

```
crack_key.py > ...
1  #!/usr/bin/python3
2  from sys import argv
3  from Crypto.Cipher import AES
4  from Crypto.Util.Padding import pad
5
6  _, first, second, third = argv
7
8  assert len(first) == 21
9  data = bytearray(first, encoding='utf-8')
10 ciphertext = bytearray.fromhex(second)
11 iv = bytearray.fromhex(third)
12
13 with open('./words.txt') as f:
14     keys = f.readlines()
15
16 for k in keys:
17     k = k.rstrip('\n')
18     if len(k) <= 16:
19         key = k + '#'*(16-len(k))
20         cipher = AES.new(key=bytearray(key,encoding='utf-8'), mode=AES.MODE_CBC, iv=iv)
21         guess = cipher.encrypt(pad(data, 16))
22         if guess == ciphertext:
23             print("find the key:",key)
24             exit(0)
25
26 print("cannot find the key!")
```

```
./words.txt
python crack_key.py "This is a top secret." 764aa26b55a4da654df6b19e4bce00f4ed05e09346fb0e762583cb7da2ac93a2 aabbccddeeff00998877665544332211
```

```
PS C:\Users\siu856558563\Desktop>
find the key: Syracuse#####
```

Find key is 'Syracuse#####'