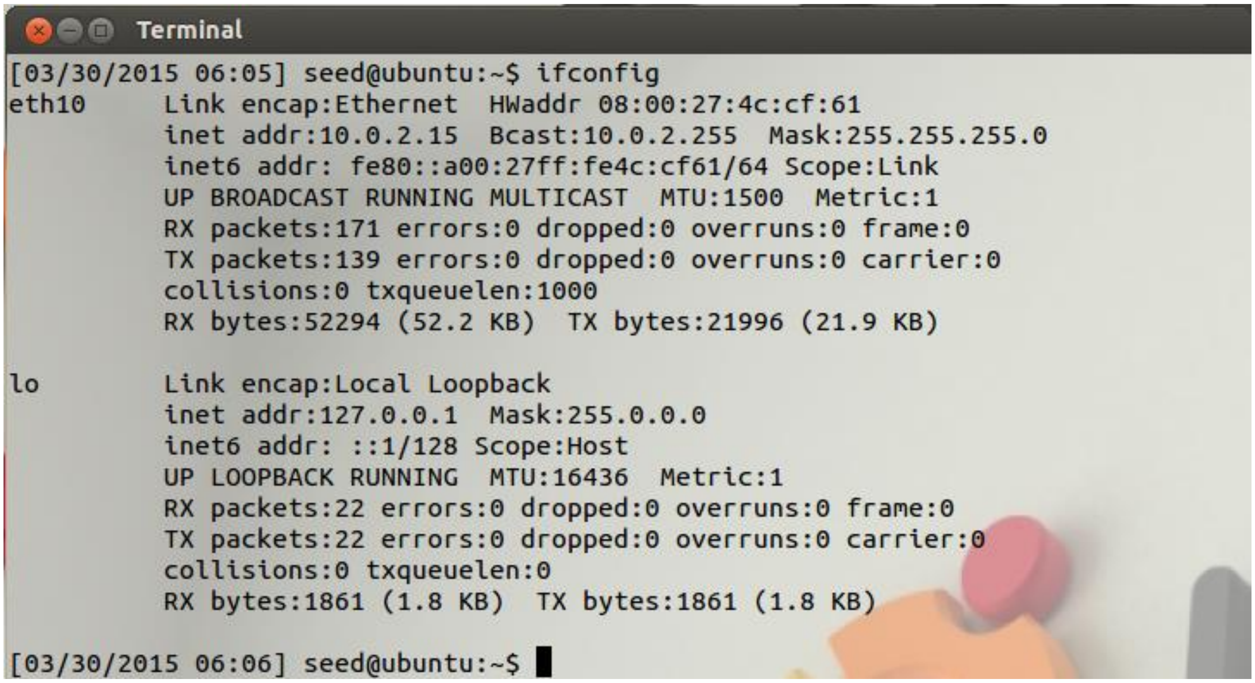
Lab 6. Crypto Lab I – Secret-Key Encryption

Bharath Darapu

Lab Environment:
In this lab we will be using 1 Virtual Machines, VM_1:



```
[03/30/2015 06:05] seed@ubuntu:~$ ifconfig
eth10     Link encap:Ethernet  HWaddr 08:00:27:4c:cf:61
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe4c:cf61/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:171 errors:0 dropped:0 overruns:0 frame:0
          TX packets:139 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:52294 (52.2 KB)  TX bytes:21996 (21.9 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:22 errors:0 dropped:0 overruns:0 frame:0
          TX packets:22 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:1861 (1.8 KB)  TX bytes:1861 (1.8 KB)

[03/30/2015 06:06] seed@ubuntu:~$
```

VM_1: 10.0.2.15

Installing OpenSSL:
OpenSSL is already downloaded into /seed//openssl-1.0.1, now we will configure it using the following commands as super user:
$ /seed/openssl-1.0.1
$ ./config
$ make
$ make test
$ sudo make install

If we have /usr/local/ssl then our configuration ran successfully.

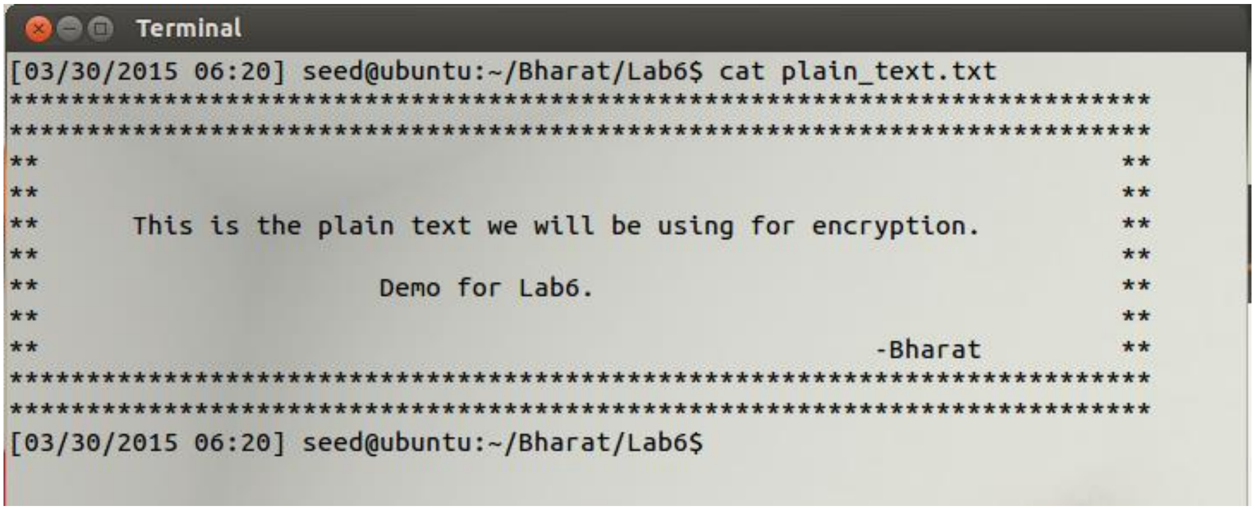
Installing a hex editor:
A hex editor called ‘GHex’ is already installed and configured in our pre built VMs.

3 Lab Tasks

3.1 Task 1: Encryption using different ciphers and modes

For this task we will be using 3 different encryption schemes each with 3 different modes. The **encryption schemes** we will be using are:
a. DES
b. AES
c. Blow Fish
And the **models** we will be using are:
a. CBC
b. CFB
c. ECB

For encrypting we need plain text. And for that I will be using a text file (plain_text.txt). And the contents are as follows:

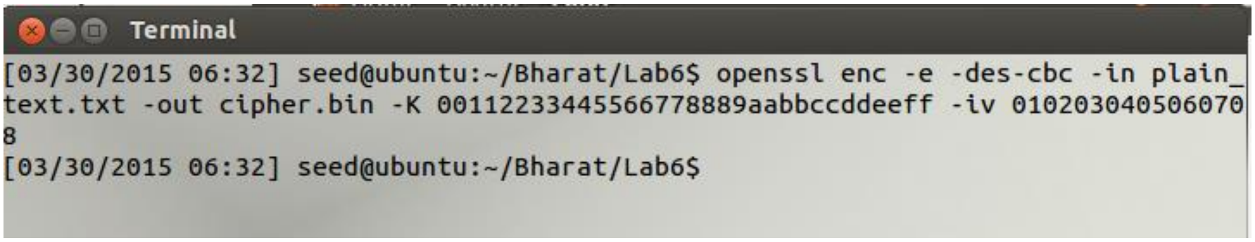


```
[03/30/2015 06:20] seed@ubuntu:~/Bharat/Lab6$ cat plain_text.txt
**************************************************************************
**************************************************************************
**                                                                      **
**                                                                      **
**      This is the plain text we will be using for encryption.        **
**                                                                      **
**                      Demo for Lab6.                                  **
**                                                                      **
**                                                     -Bharat          **
**************************************************************************
**************************************************************************
[03/30/2015 06:20] seed@ubuntu:~/Bharat/Lab6$
```

plain_text.txt file contents

3.1.1 Encryption Scheme: DES, Mode: CBC

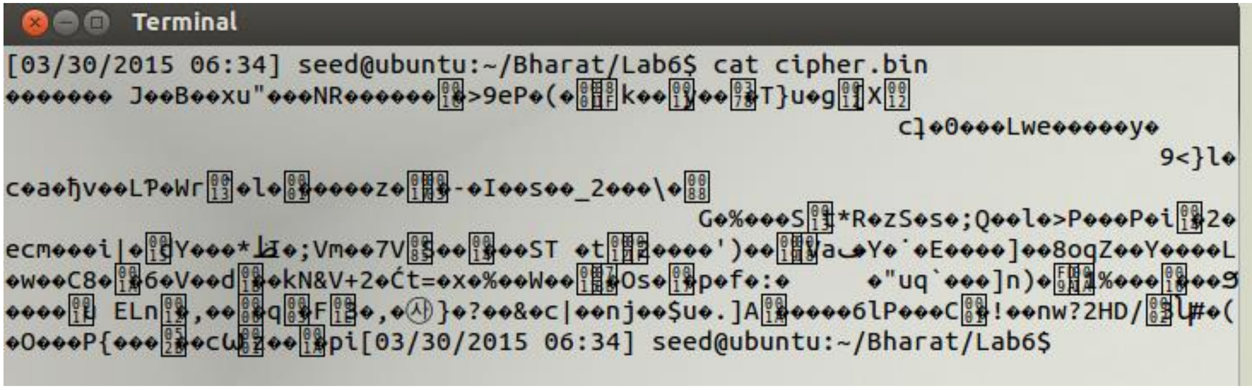
For this task we will be using the following command to **encrypt** the contents of plain_text.txt file:
'openssl enc -e -des-cbc -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708'



```
[03/30/2015 06:32] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -des-cbc -in plain_
text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 010203040506070
8
[03/30/2015 06:32] seed@ubuntu:~/Bharat/Lab6$
```

des-cbc encryption

Now let us check the contents of the encrypted file (cipher.bin)

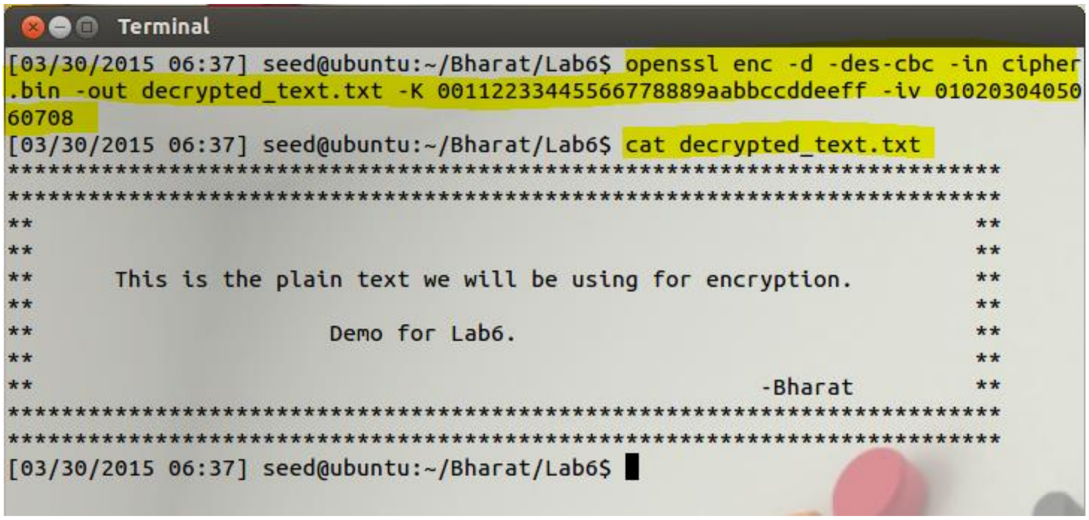


```
[03/30/2015 06:34] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
������� J��B��xu"���NR������□�>9eP�(�□□k��□y��□�T}u�g□]X□
                                                               c]�0���Lwe������y�
                                                                                  9<}l�
c�a�ђv��LP�Wr□�l�□�����z�□�-�I��s��_2���\�□
                                                   G�%���S□*R�zS�s�;Q��l�>P���P�i□�2�
ecm���i|�□Y���*ﻻ�;Vm��7V□��□��ST �t□����')��□Va ﺐ�Y�˙�E����]��8oqZ��Y����L
�w��C8�□6�V��d□�kN&V+2�Ćt=�x�%��W��□�Os�□p�f�:�     �"uq`���]n)�□%���□��ƽ
����□ ELn□,��□q□F□�,�㉦}�?��&�c|��nj��$u�.]A□�����6lP���C□!��nw?2HD/□Ц#�(
�O���P{���□�cЩ□��□pi[03/30/2015 06:34] seed@ubuntu:~/Bharat/Lab6$
```

cipher text des-cbc contents

Now lets try to **decrypt** the above text using the following command and check if we get back our plain text:
'openssl enc -d -des-cbc -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708'.

The main change we should notice in the two commands in the flags. For encryption we use '-e' and for decruption we use '-d'. And of course our input/output files change and the key, iv(initial vector) remain the same.

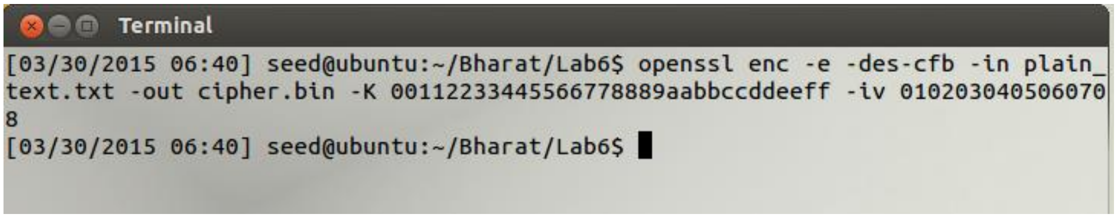


```
[03/30/2015 06:37] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -des-cbc -in cipher
.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 01020304050
60708
[03/30/2015 06:37] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
**************************************************************************
**************************************************************************
**                                                                      **
**                                                                      **
**      This is the plain text we will be using for encryption.         **
**                                                                      **
**                      Demo for Lab6.                                  **
**                                                                      **
**                                                      -Bharat         **
**************************************************************************
**************************************************************************
[03/30/2015 06:37] seed@ubuntu:~/Bharat/Lab6$ █
```

des-cbc decryption

3.1.2 Encryption Scheme: DES, Mode: CFB

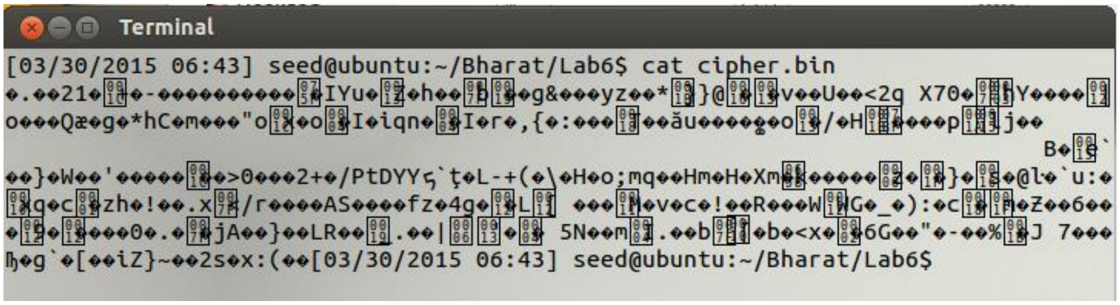
For this task we will be using the following command to encrypt the plain text:
‘openssl enc -e -des-cfb -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708’



```
[03/30/2015 06:40] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -des-cfb -in plain_
text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 010203040506070
8
[03/30/2015 06:40] seed@ubuntu:~/Bharat/Lab6$ █
```

des-cfb encryption

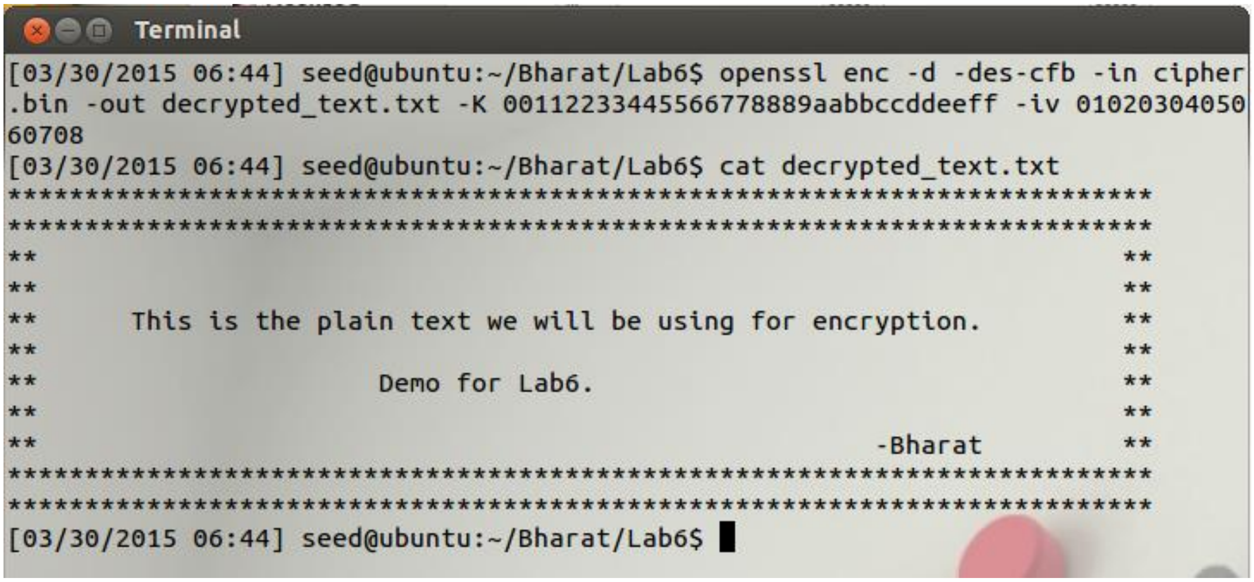
The encrypted text will look like:



```
[03/30/2015 06:43] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
�.��21�□�-����������□�IYu�□�h��□b□�g&���yz��*□}@□□v��U��<2q X70�□□Y����□
o���Qæ�g�*hC�m���"o□�o□�I�iqn�□�I�r�,{�:���□��ău�����ȣo□/�H□□���p□□j��
                                                                     B�□`
��}�W��'�����□�>0���2+�/PtDYYς`ţ�L-+(�\�H�o;mq��Hm�H�Xm□�����□�□}�□�@l�`u:�
□q�c□zh�!��.x□/r����AS����fz�4g�□L□ ���□�v�c�!��R���W□G�_�):�c□□�Z��6��
�□�□���0�.�□jA��}��LR��□.��|□□�□ 5N��m□.��b□�b�<x�□6G��"�-��%□J 7���
ђ�g`�[��iZ}~��2s�x:(��[03/30/2015 06:43] seed@ubuntu:~/Bharat/Lab6$
```

cipher.bin (encrypted text) contents

Now lets try to decrypt the above text using the following command and check if we get back our plain text: ‘openssl enc -d -des-cfb -in cipher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708’

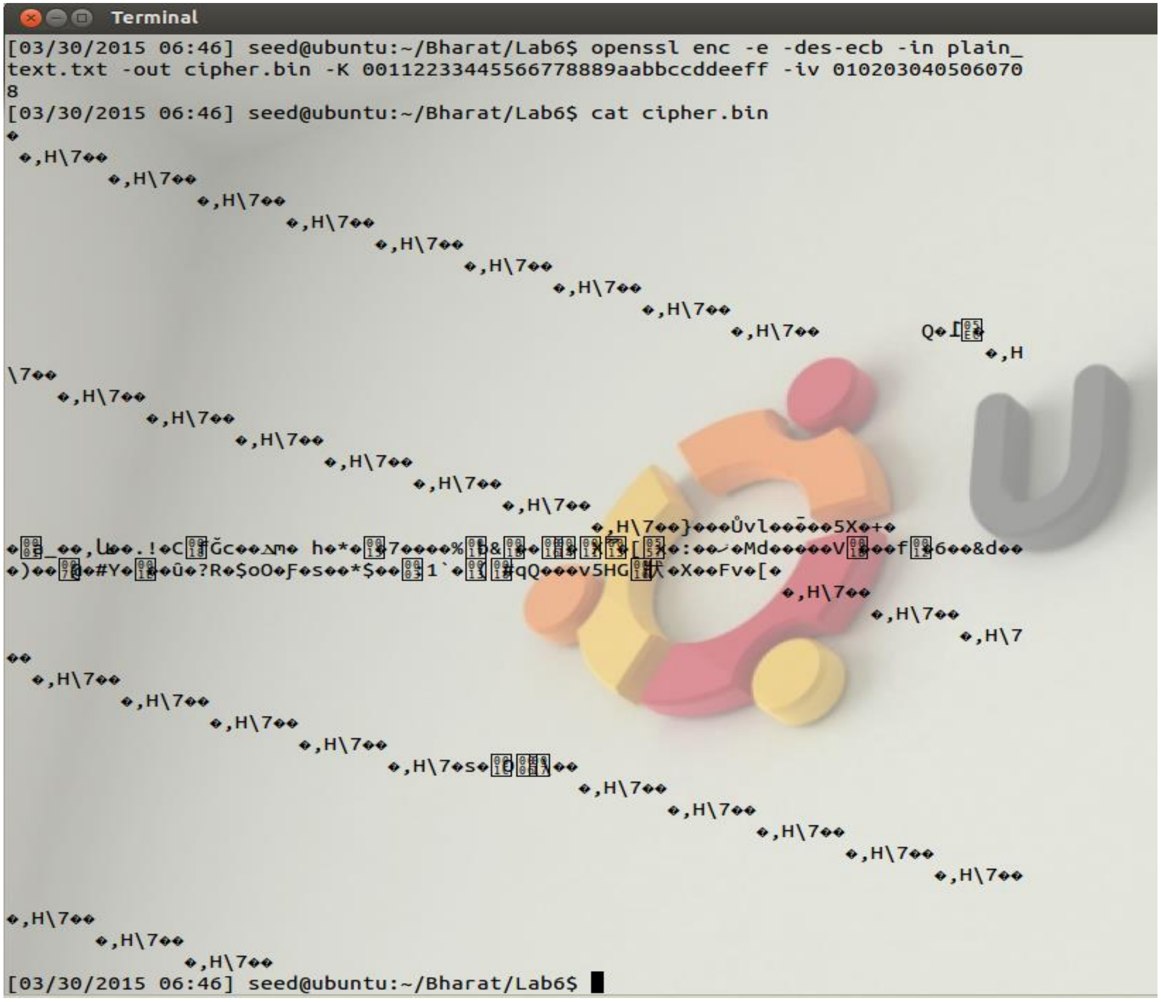


```
[03/30/2015 06:44] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -des-cfb -in cipher
.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 01020304050
60708
[03/30/2015 06:44] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
******************************************************************************
******************************************************************************
**                                                                          **
**                                                                          **
**      This is the plain text we will be using for encryption.             **
**                                                                          **
**                       Demo for Lab6.                                     **
**                                                                          **
**                                                     -Bharat              **
******************************************************************************
******************************************************************************
[03/30/2015 06:44] seed@ubuntu:~/Bharat/Lab6$ █
```

decryption and the contents

3.1.3 Encryption Scheme: DES, Mode: ECB

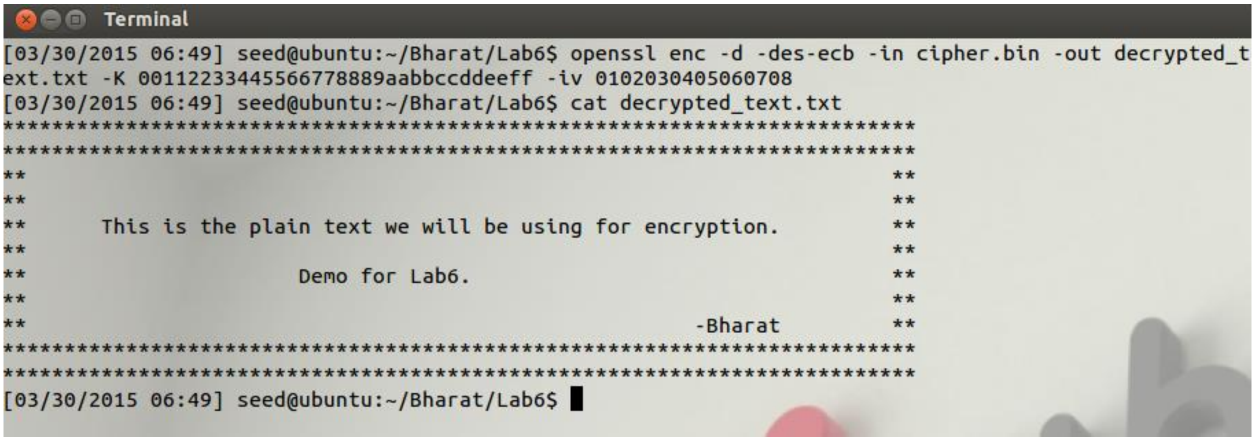
For this task we will be using the following command: ‘openssl enc -e -des-ecb -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708’ and the encrypted text is also shown:



```
[03/30/2015 06:46] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -des-ecb -in plain_
text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 010203040506070
8
[03/30/2015 06:46] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
◆
 ◆,H\7◆◆
        ◆,H\7◆◆
               ◆,H\7◆◆
                      ◆,H\7◆◆
                             ◆,H\7◆◆
                                    ◆,H\7◆◆
                                           ◆,H\7◆◆
                                                  ◆,H\7◆◆
                                                         ◆,H\7◆◆         Q◆I□
                                                                              ◆,H
\7◆◆
    ◆,H\7◆◆
           ◆,H\7◆◆
                  ◆,H\7◆◆
                         ◆,H\7◆◆
                                ◆,H\7◆◆
                                       ◆,H\7◆◆
                                              ◆,H\7◆◆}◆◆◆Ůvl◆◆ō◆◆5X◆+◆
◆□_◆◆,Ŀ◆◆.!◆C□Ğc◆◆⍙m◆ h◆*◆□7◆◆◆◆%□&□◆□□□[□◆:◆◆︶◆Md◆◆◆◆◆V□◆◆f□6◆◆&d◆◆
◆)◆◆□◆#Y◆□◆û◆?R◆$oO◆Ƒ◆s◆◆*$◆◆□1`◆□□qQ◆◆◆v5HG□◆X◆◆Fv◆[◆
                                                              ◆,H\7◆◆
                                                                     ◆,H\7◆◆
                                                                            ◆,H\7
◆◆
  ◆,H\7◆◆
         ◆,H\7◆◆
                ◆,H\7◆◆
                       ◆,H\7◆◆
                              ◆,H\7◆s◆□□◆◆
                                             ◆,H\7◆◆
                                                    ◆,H\7◆◆
                                                           ◆,H\7◆◆
                                                                  ◆,H\7◆◆
                                                                         ◆,H\7◆◆

◆,H\7◆◆
       ◆,H\7◆◆
              ◆,H\7◆◆
[03/30/2015 06:46] seed@ubuntu:~/Bharat/Lab6$ █
```

DES-ECB encryption

We will use the following command to decrypt and then see the decrypted text:
‘openssl enc -d -des-ecb -in cipher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708’

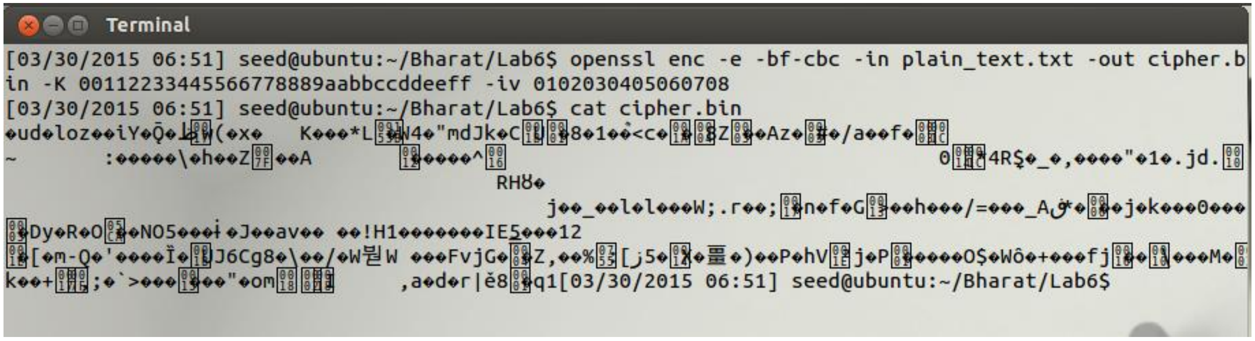


```
[03/30/2015 06:49] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -des-ecb -in cipher.bin -out decrypted_t
ext.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 06:49] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
***************************************************************************
***************************************************************************
**                                                                       **
**                                                                       **
**       This is the plain text we will be using for encryption.         **
**                                                                       **
**                       Demo for Lab6.                                  **
**                                                                       **
**                                                        -Bharat        **
***************************************************************************
***************************************************************************
[03/30/2015 06:49] seed@ubuntu:~/Bharat/Lab6$ █
```

des-ecb decryption

3.1.4 Encryption Scheme: Blow Fish, Mode: CBC

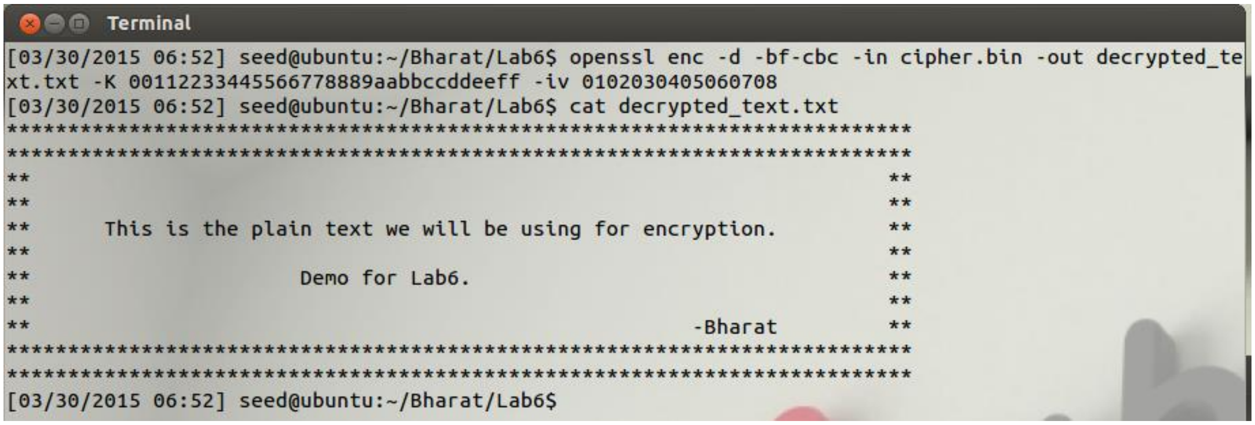
For this task we will be using the following command: ‘openssl enc -e -bf-cbc -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708’ and the encrypted text is also shown:



```
[03/30/2015 06:51] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -bf-cbc -in plain_text.txt -out cipher.b
in -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 06:51] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
�ud�loz��iY�Q̄�▯▯(�x�   K���*L▯▯N4�"mdJk�C▯▯▯8�1��<c�▯▯▯Z▯�Az�▯�/a��f�▯
~       :�����\�h��Z▯��A        ▯�����^▯                                    0▯▯4R$�_�,����"�1�.jd.▯
                                 RHȣ�
                                    j��_��l�l���W;.r��;▯n�f�G▯��h���/=���_Aی*�▯�j�k���0���
▯Dy�R�O▯�NO5���ɨ�J��av�� ��!H1�������IE5���12
▯[�m-Q�'����Ĩ�▯J6Cg8�\��/�W뷛W ���FvjG�▯Z,��%▯[ز5�▯�▯�)��P�hV▯j�P▯����0$�Wô�+���fj▯�▯���M�▯
k��+▯;�`>���▯��"�om▯▯      ,a�d�r|ě8▯q1[03/30/2015 06:51] seed@ubuntu:~/Bharat/Lab6$
```

BF-CBC encryption

WE will use the following command to decrypt and then see the decrypted text:
‘openssl enc -d -bf-cbc -in cipher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708’

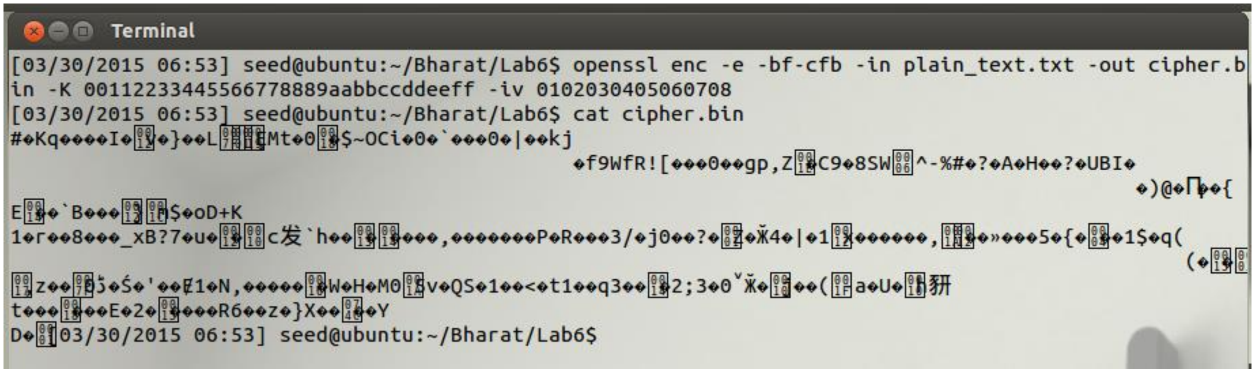


```
[03/30/2015 06:52] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -bf-cbc -in cipher.bin -out decrypted_te
xt.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 06:52] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
***************************************************************************
***************************************************************************
**                                                                       **
**                                                                       **
**       This is the plain text we will be using for encryption.         **
**                                                                       **
**                       Demo for Lab6.                                  **
**                                                                       **
**                                                        -Bharat        **
***************************************************************************
***************************************************************************
[03/30/2015 06:52] seed@ubuntu:~/Bharat/Lab6$
```

bf-cbc decryption

3.1.5 Encryption Scheme: Blow Fish, Mode: CFB

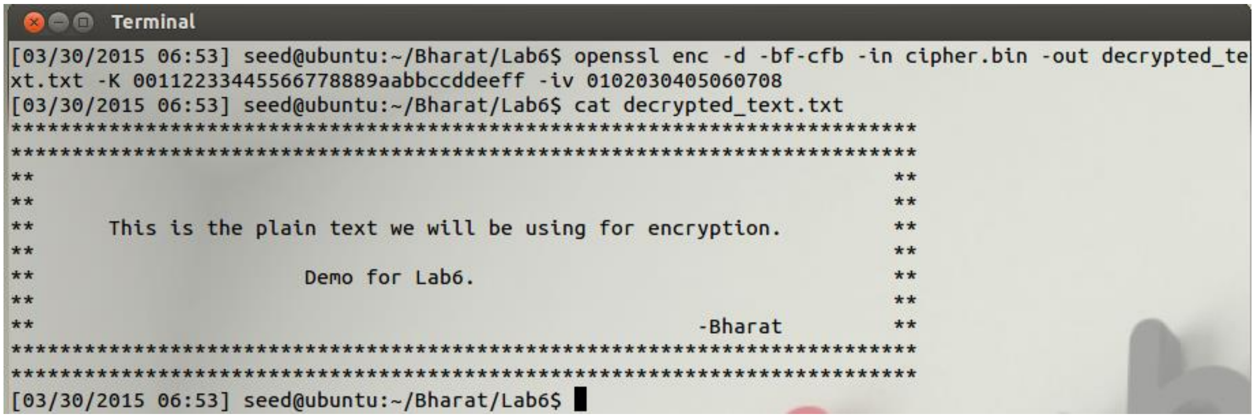
For this task we will be using the following command: ‘openssl enc -e -bf-cfb -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708’ and the encrypted text is also shown:



```
[03/30/2015 06:53] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -bf-cfb -in plain_text.txt -out cipher.b
in -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 06:53] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
#�Kq����I�□�}��L□□□Mt�0□$~OCi�0�`���0�|��kj
                                                  �f9WfR![���0��gp,Z□C9�8SW□^-%#�?�A�H��?�UBI�
                                                                                            �)@�Γ�{
E□�`B���□□$�oD+K
1�r��8���_xB?7�u�□□c发`h��□□���,�������P�R���3/�j0��?�□�Ж4�|�1□������,□�»���5�{�□�1$�q(
                                                                                              (�□□
□z��□ӟ�Ś�'��Ɇ1�N,�����□W�H�M0□v�QS�1��<�t1��q3��□2;3�0ˇЖ�□��(□a�U�□犴
t���□��E�2�□���R6��z�}X��□�Y
D�□03/30/2015 06:53] seed@ubuntu:~/Bharat/Lab6$
```

BF-CFB encryption

We will use the following command to decrypt and then see the decrypted text:
‘openssl enc -d -bf-cfb -in cipher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708’

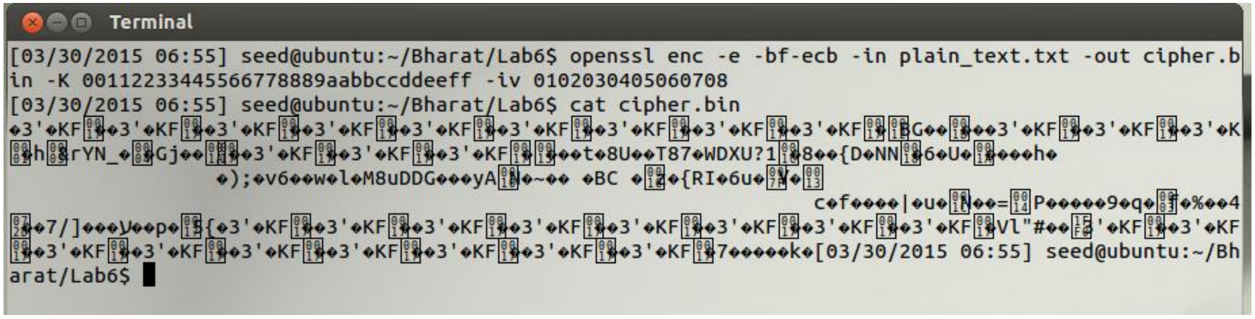


```
[03/30/2015 06:53] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -bf-cfb -in cipher.bin -out decrypted_te
xt.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 06:53] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
**************************************************************************
**************************************************************************
**                                                                      **
**                                                                      **
**      This is the plain text we will be using for encryption.         **
**                                                                      **
**                      Demo for Lab6.                                  **
**                                                                      **
**                                                  -Bharat             **
**************************************************************************
**************************************************************************
[03/30/2015 06:53] seed@ubuntu:~/Bharat/Lab6$ █
```

BF-CFB decryption

3.1.6 Encryption Scheme: Blow Fish, Mode: ECB

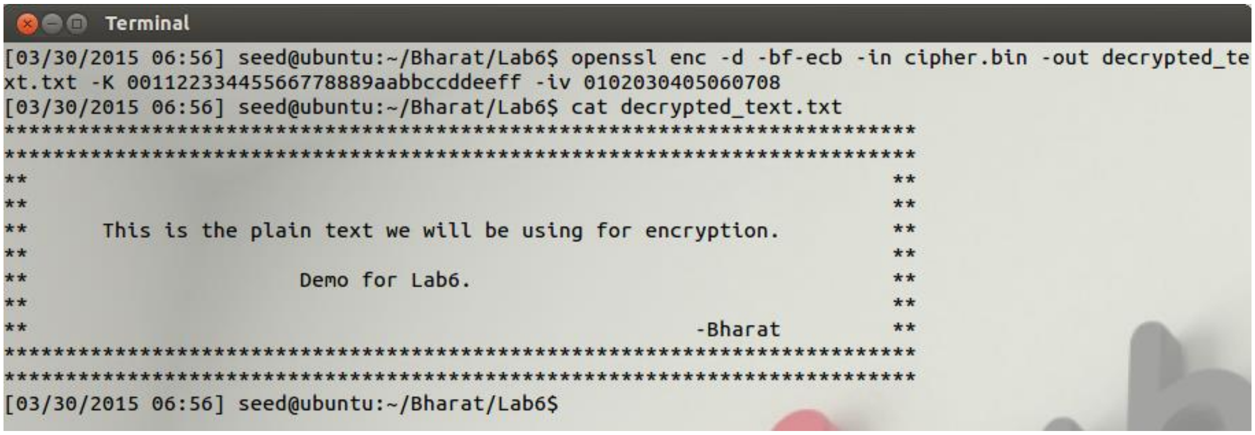
For this task we will be using the following command: ‘openssl enc -e -bf-ecb -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708’ and the encrypted text is also shown:



```
[03/30/2015 06:55] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -bf-ecb -in plain_text.txt -out cipher.b
in -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 06:55] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□□G��□��3'�KF□�3'�KF□�3'�K
□h□rYN_�□Gj��□�3'�KF□�3'�KF□�3'�KF□□��t�8U��T87�WDXU?1□8��{D�NN□6�U�□���h�
                    �);�v6��w�l�M8uDDG���yA□�~�� �BC �□�{RI�6u�□�□
                                                            c�f����|�u�□��=□P�����9�q�□�%��4
□�7/]���V��p�□{�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□Vl"#��□'�KF□�3'�KF
□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□�3'�KF□7�����k�[03/30/2015 06:55] seed@ubuntu:~/Bh
arat/Lab6$ █
```

BF-ECB encryption

We will use the following command to decrypt and then see the decrypted text:
‘openssl enc -d -bf-ecb -in cipher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708’



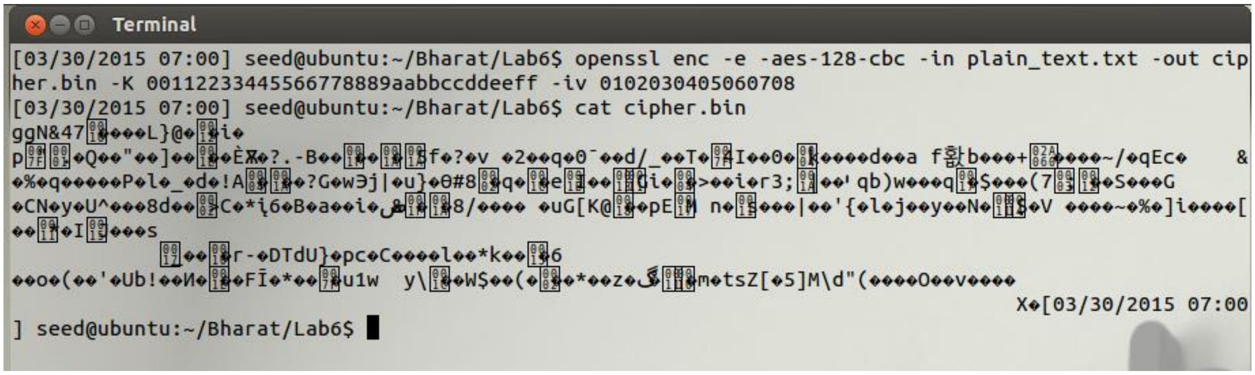
```
[03/30/2015 06:56] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -bf-ecb -in cipher.bin -out decrypted_te
xt.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 06:56] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
**************************************************************************
**************************************************************************
**                                                                      **
**                                                                      **
**       This is the plain text we will be using for encryption.        **
**                                                                      **
**                       Demo for Lab6.                                 **
**                                                                      **
**                                                     -Bharat          **
**************************************************************************
**************************************************************************
[03/30/2015 06:56] seed@ubuntu:~/Bharat/Lab6$
```

BF-ECB decryption

3.1.7 Encryption Scheme: AES -128, Mode: CBC

Unlike the above two encryption schemes, in AES there are multiple key versions. In order to demonstrate we will be using different versions for as we change the modes.

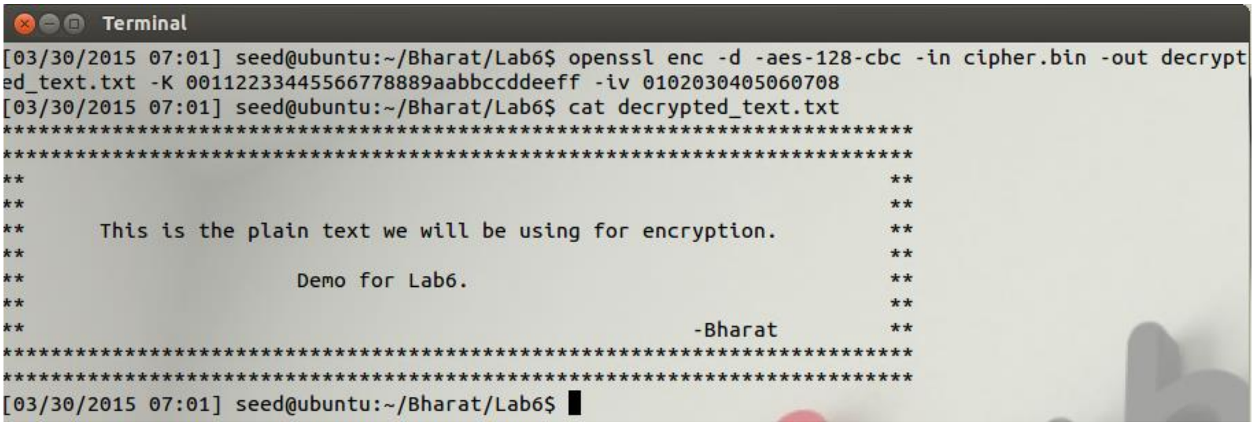
For this task we will be using the following command: ‘openssl enc -e –aes-128-cbc -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708’ and the encrypted text is also shown:



```
[03/30/2015 07:00] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-cbc -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:00] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
ggN&47▯�▯��L}@�▯i�
p▯▯�Q��"��]��▯�ÈӾ�?.-B��▯�▯▯f�?�v �2��q�0¯��d/_��T�▯I��0�▯����d��a f핤b���+▯���~/�qEc�    &
�%�q�����P�l�_�d�!A▯▯�?G�w∋j|�u}�0#8▯q�▯e▯��▯i�▯>��i�r3;▯��' qb)w���q▯$���(7▯▯�S���G
�CN�y�U^���8d��▯C�*į6�B�a��i�ھ▯▯8/���� �uG[K@▯�pE▯ n�▯���|��'{�l�j��y��N�▯�V ����~�%�]i����[
��▯�I▯���s
          ▯��▯r-�DTdU}�pc�C����l��*k��▯6
��o�(��'�Ub!��И�▯�FĪ�*��▯u1w  y\▯�W$��(�▯�*��z�ڰ▯m�tsZ[�5]M\d"(����O��v����
                                                                          X�[03/30/2015 07:00
] seed@ubuntu:~/Bharat/Lab6$ █
```

AES-128-CBC encryption

We will use the following command to decrypt and then see the decrypted text:
‘openssl enc -d -aes-128-cbc -in cipher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708’

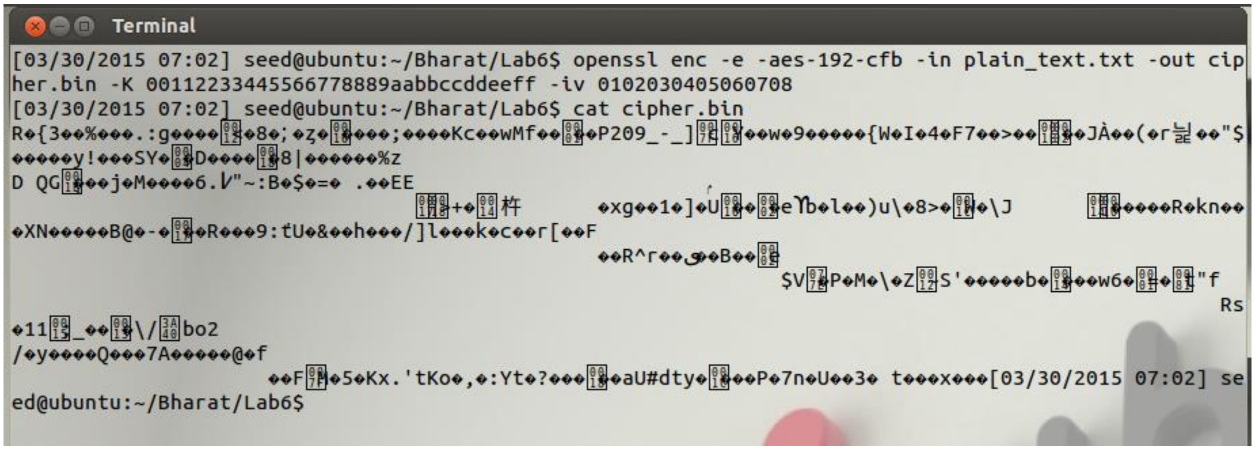


```
[03/30/2015 07:01] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -aes-128-cbc -in cipher.bin -out decrypt
ed_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:01] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
**************************************************************************
**************************************************************************
**                                                                      **
**                                                                      **
**       This is the plain text we will be using for encryption.        **
**                                                                      **
**                       Demo for Lab6.                                 **
**                                                                      **
**                                                     -Bharat          **
**************************************************************************
**************************************************************************
[03/30/2015 07:01] seed@ubuntu:~/Bharat/Lab6$ █
```

AES-128-CBC decryption

3.1.8 Encryption Scheme: AES -192, Mode: CFB

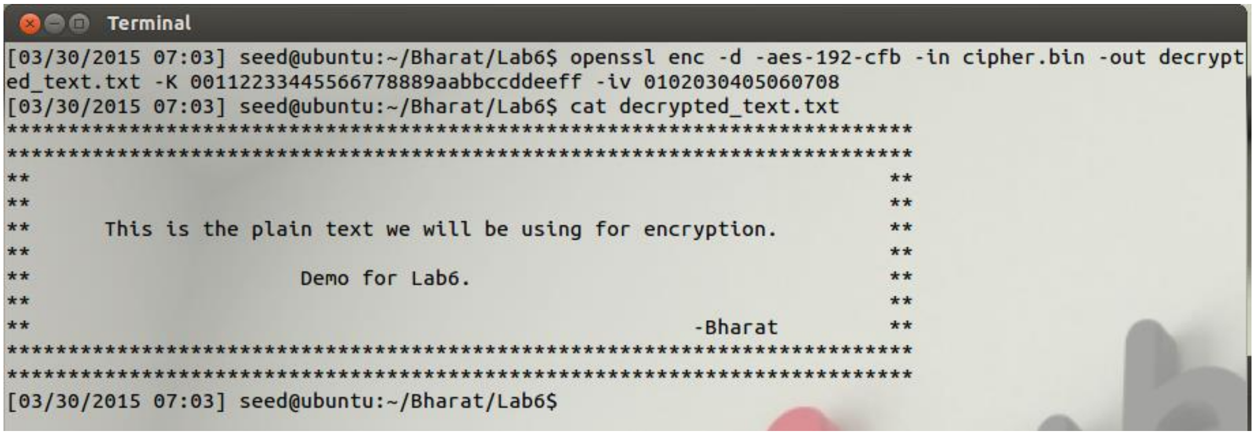
For this task we will be using the following command: ‘openssl enc -e -aes-192-cfb -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708’ and the encrypted text is also shown:



```
[03/30/2015 07:02] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-192-cfb -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:02] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
R�{3��%���.:g����□�8�;�ʐ�□���;����Kc��wMf��□�P209_-_]□□��w�9�����{W�I�4�F7��>��□�JÀ��(�r넕��"$
�����y!���SY�□D����□8|������%z
D QG□��j�M����6.Ṽ"~:B�$�=� .��EE
                                          □+�□杵      �xg��1�]�U□�□eϒb�l��)u\�8>�□�\J      □����R�kn��
�XN�����B@�-�□�R���9:ṫU�&��h���/]l���k�c��r[��F
                                                        ��R^r��ݜ��B��□
                                                                       $V□P�M�\�Z□S'�����b�□��w6�□�□"f
                                                                                                    Rs
�11□_��□\/□bo2
/�y����Q���7A�����@�f
                      ��F□�5�Kx.'tKo�,�:Yt�?���□�aU#dty�□��P�7n�U��3� t���x���[03/30/2015 07:02] se
ed@ubuntu:~/Bharat/Lab6$
```

AES-192-CFB encryption

WE will use the following command to decrypt and then see the decrypted text:
‘openssl enc -d -aes-192-cfb -in cipher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708’

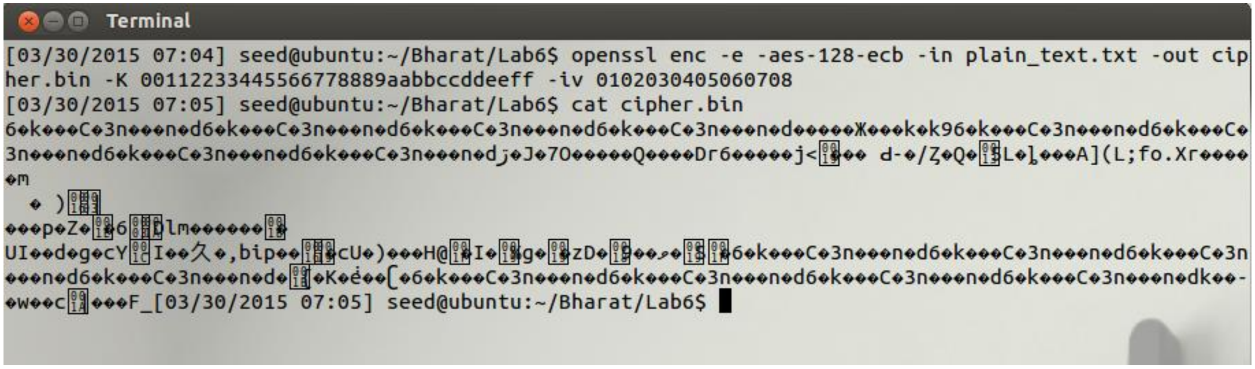


```
[03/30/2015 07:03] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -aes-192-cfb -in cipher.bin -out decrypt
ed_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:03] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
**************************************************************************
**************************************************************************
**                                                                      **
**                                                                      **
**      This is the plain text we will be using for encryption.         **
**                                                                      **
**                      Demo for Lab6.                                  **
**                                                                      **
**                                                      -Bharat         **
**************************************************************************
**************************************************************************
[03/30/2015 07:03] seed@ubuntu:~/Bharat/Lab6$
```

AES-192-CFB decryption

3.1.7 Encryption Scheme: AES -128, Mode: ECB

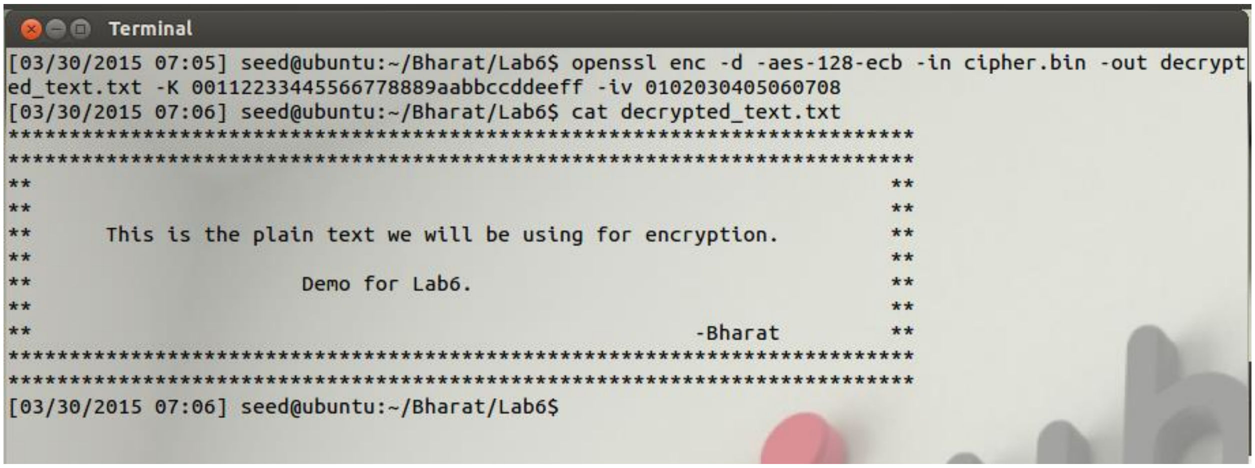
For this task we will be using the following command: ‘openssl enc -e -aes-128-ecb -in plain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708’ and the encrypted text is also shown:



```
[03/30/2015 07:04] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-ecb -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:05] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n���n�d�����Ж���k�k96�k���C�3n���n�d6�k���C�
3n���n�d6�k���C�3n���n�d6�k���C�3n���n�dȷ�J�7O�����Q����Dr6�����j<□�� d-�/Ȥ�Q�□L�ȴ���A](L;fo.Xr����
�m
  � )□
���p�Z�□6□Dlm������□
UI��d�g�cY□I��久�,bip��□cU�)���H@□I�□g�□zD�□��ـ�□□6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n
���n�d6�k���C�3n���n�d�□�Kėǿ�[�6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n���n�dk��-
�w��c□���F_[03/30/2015 07:05] seed@ubuntu:~/Bharat/Lab6$ █
```

AES-128-ECB encryption

WE will use the following command to decrypt and then see the decrypted text:
‘openssl enc -d -aes-128-ecb -in cipher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708’

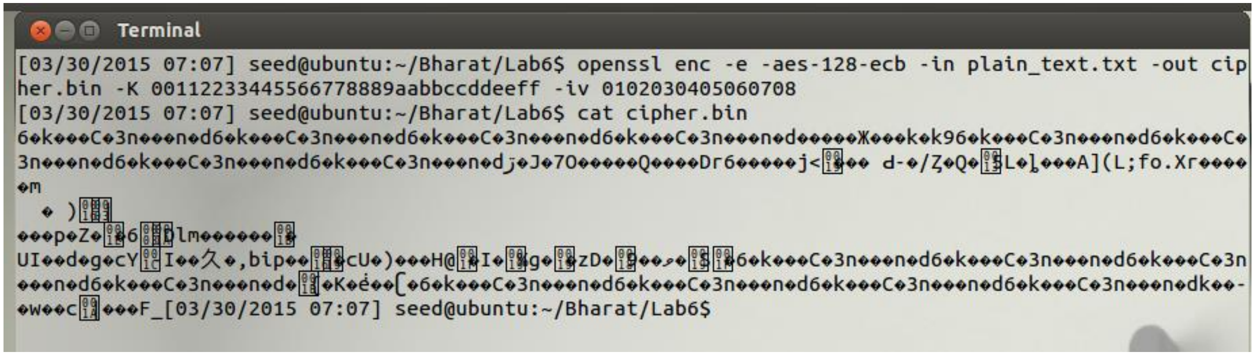


```
[03/30/2015 07:05] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -aes-128-ecb -in cipher.bin -out decrypt
ed_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:06] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
**************************************************************************
**************************************************************************
**                                                                      **
**                                                                      **
**      This is the plain text we will be using for encryption.         **
**                                                                      **
**                      Demo for Lab6.                                  **
**                                                                      **
**                                                      -Bharat         **
**************************************************************************
**************************************************************************
[03/30/2015 07:06] seed@ubuntu:~/Bharat/Lab6$
```

AES-192-CFB decryption

Now just out of curiosity let us see what happens when we encrypt with one key size and try to decrypt with another key size.

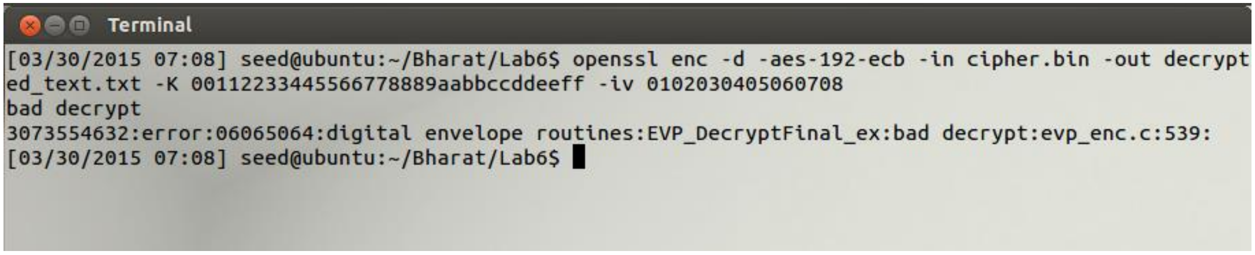
Encrypt with 128 bit:



```
[03/30/2015 07:07] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-ecb -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:07] seed@ubuntu:~/Bharat/Lab6$ cat cipher.bin
6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n���n�d�����Ж���k�k96�k���C�3n���n�d6�k���C�
3n���n�d6�k���C�3n���n�d6�k���C�3n���n�dȷ�J�7O�����Q����Dr6�����j<□�� d-�/Ȥ�Q�□L�ȴ���A](L;fo.Xr����
�m
  � )□□
���p�Z�□6□□lm������□
UI��d�g�cY□I��久�,bip��□□cU�)���H@□I�□g�□zD�□���,�□□6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n
���n�d6�k���C�3n���n�d�□�Kėė�[�6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n���n�d6�k���C�3n���n�dk��-
�w��c□���F_[03/30/2015 07:07] seed@ubuntu:~/Bharat/Lab6$
```

encrypt with AES 128 bit key size and mode as ECB

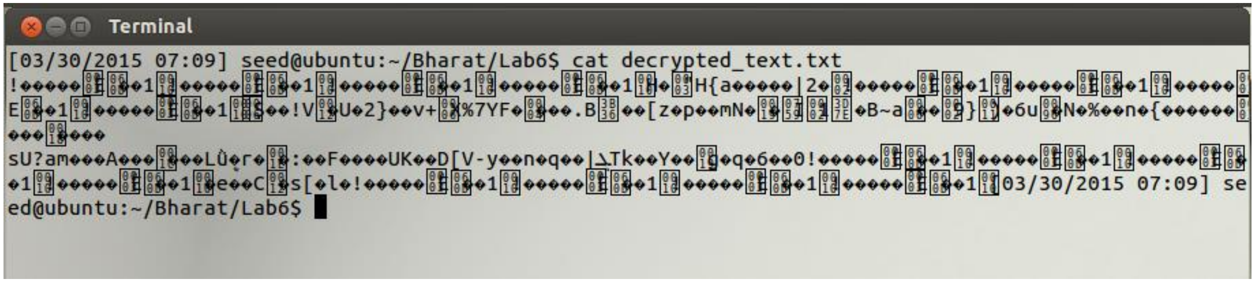
Now let us decrypt with 192 bit key size. WE get the following error.



```
[03/30/2015 07:08] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -aes-192-ecb -in cipher.bin -out decrypt
ed_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
bad decrypt
3073554632:error:06065064:digital envelope routines:EVP_DecryptFinal_ex:bad decrypt:evp_enc.c:539:
[03/30/2015 07:08] seed@ubuntu:~/Bharat/Lab6$ █
```

decryption error

And when we see the decrypted file:



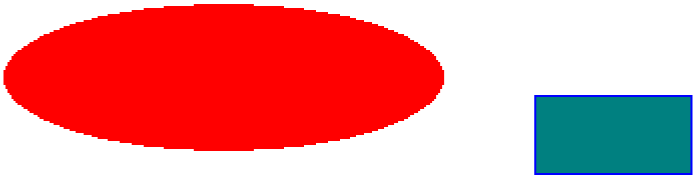
```
[03/30/2015 07:09] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
!�����□□�1□�����□□�1□�����□□�1□�����□□�1□�□H{a�����|2�□�����□□�1□�����□□�1□�����□
E□�1□�����□□�1□□��!V□U�2}��v+□%7YF�□��.B□��[z�p��mN�□□□□�B~a□�□}□�6u□N�%��n�{������□
���□���
sU?am���A���□��LÛ�r�□:��F����UK��D[V-y��n�q��|ↂTk��Y��□�q�6��0!�����□□�1□�����□□�1□�����□□
�1□�����□□�1□e��C□s[�l�!�����□□�1□�����□□�1□�����□□�1□�����□□�1□03/30/2015 07:09] se
ed@ubuntu:~/Bharat/Lab6$ █
```

contents when decrypted with wrong key size.

3.2 Task 2: Encryption Mode – ECB vs. CBC

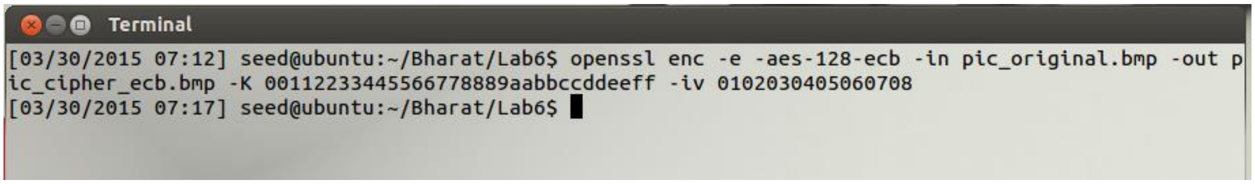
In this task we shall try to encrypt a picture rather than a text file and check which mode gives which results. The image we will be using is the one provided in the lab (pic_original.bmp).

Let us open the file and see the image before encryption:



pic_original before encryption.

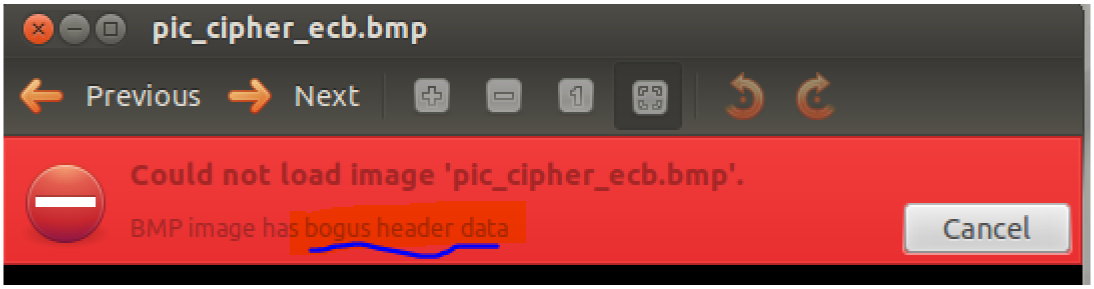
Let us use AES-128 for the encryption as follows:



```
[03/30/2015 07:12] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-ecb -in pic_original.bmp -out p
ic_cipher_ecb.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:17] seed@ubuntu:~/Bharat/Lab6$ 
```

encryption using aes 128 ecb mode

However since the header information is changed the file will not be recognized as an image file and cannot be opened. The error we get is:



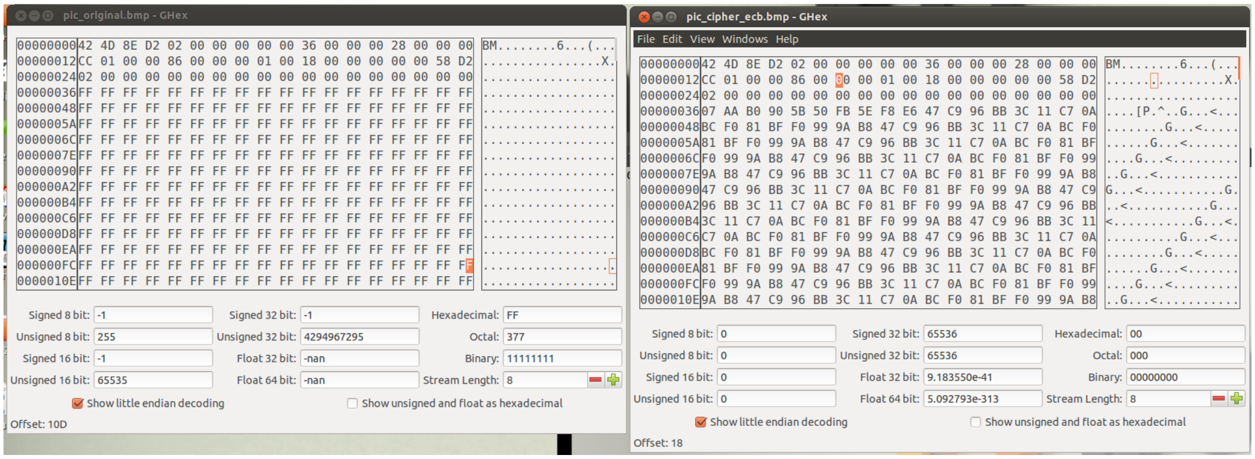
Could not load image 'pic_cipher_ecb.bmp'.

BMP image has bogus header data

ciper_image cannot be opened error

Let us copy the header from the original file and change that of the encrypted file.

The first 54 bits (36 in hex) will be the header file.



```
0000000042 4D 8E D2 02 00 00 00 00 00 36 00 00 00 28 00 00 00 BM........6...(...
00000012CC 01 00 00 86 00 00 00 01 00 18 00 00 00 00 00 58 D2 ................X.
0000002402 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
00000036FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
00000048FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
0000005AFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
0000006CFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
0000007EFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
00000090FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
000000A2FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
000000B4FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
000000C6FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
000000D8FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
000000EAFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
000000FCFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
0000010EFF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ..................
```

Signed 8 bit: -1 | Signed 32 bit: -1 | Hexadecimal: FF
Unsigned 8 bit: 255 | Unsigned 32 bit: 4294967295 | Octal: 377
Signed 16 bit: -1 | Float 32 bit: -nan | Binary: 11111111
Unsigned 16 bit: 65535 | Float 64 bit: -nan | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 10D

```
0000000042 4D 8E D2 02 00 00 00 00 00 36 00 00 00 28 00 00 00 BM........6...(...
00000012CC 01 00 00 86 00 00 00 01 00 18 00 00 00 00 00 58 D2 ................X.
0000002402 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ..................
0000003607 AA B0 90 5B 50 FB 5E F8 E6 47 C9 96 BB 3C 11 C7 0A ....[P.^..G...<...
00000048BC F0 81 BF F0 99 9A B8 47 C9 96 BB 3C 11 C7 0A BC F0 ........G...<.....
0000005A81 BF F0 99 9A B8 47 C9 96 BB 3C 11 C7 0A BC F0 81 BF ......G...<.......
0000006CF0 99 9A B8 47 C9 96 BB 3C 11 C7 0A BC F0 81 BF F0 99 ....G...<.........
0000007E9A B8 47 C9 96 BB 3C 11 C7 0A BC F0 81 BF F0 99 9A B8 ..G...<...........
0000009047 C9 96 BB 3C 11 C7 0A BC F0 81 BF F0 99 9A B8 47 C9 G...<...........G.
000000A296 BB 3C 11 C7 0A BC F0 81 BF F0 99 9A B8 47 C9 96 BB ..<...........G...
000000B43C 11 C7 0A BC F0 81 BF F0 99 9A B8 47 C9 96 BB 3C 11 <...........G...<.
000000C6C7 0A BC F0 81 BF F0 99 9A B8 47 C9 96 BB 3C 11 C7 0A ..........G...<...
000000D8BC F0 81 BF F0 99 9A B8 47 C9 96 BB 3C 11 C7 0A BC F0 ........G...<.....
000000EA81 BF F0 99 9A B8 47 C9 96 BB 3C 11 C7 0A BC F0 81 BF ......G...<.......
000000FCF0 99 9A B8 47 C9 96 BB 3C 11 C7 0A BC F0 81 BF F0 99 ....G...<.........
0000010E9A B8 47 C9 96 BB 3C 11 C7 0A BC F0 81 BF F0 99 9A B8 ..G...<...........
```

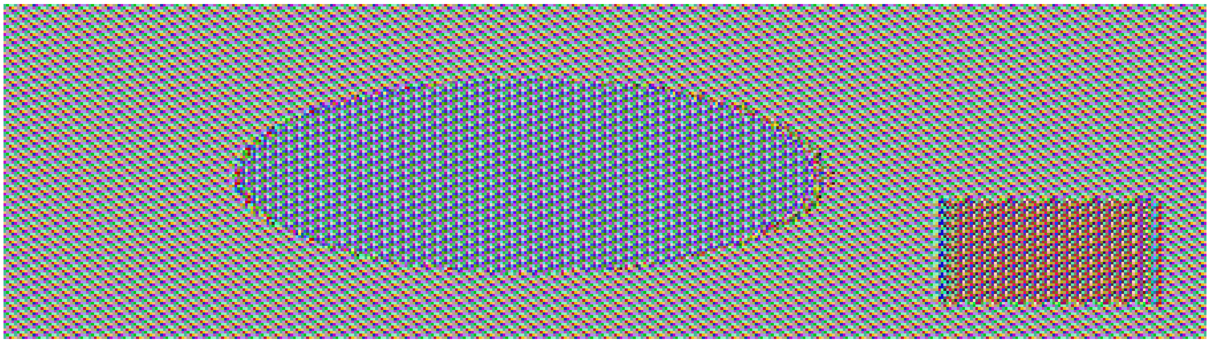
Signed 8 bit: 0 | Signed 32 bit: 65536 | Hexadecimal: 00
Unsigned 8 bit: 0 | Unsigned 32 bit: 65536 | Octal: 000
Signed 16 bit: 0 | Float 32 bit: 9.183550e-41 | Binary: 00000000
Unsigned 16 bit: 0 | Float 64 bit: 5.092793e-313 | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 18

The first 54 bits (hex format) are copied from the original file to the encrypted file.

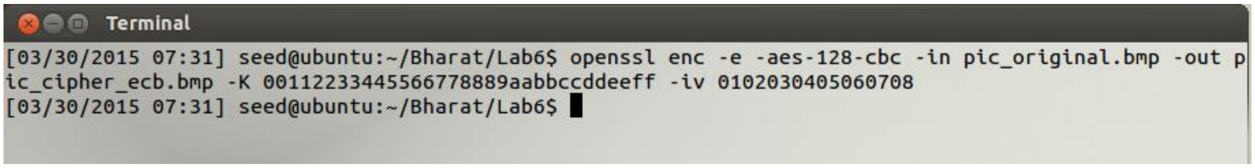
Now when we open the ciphered image file we can see the image as:



ciphered image

As we can see though the image file is encrypted the essential elements can still be found i.e. the image is not completely scrambled.

Now let us try the same image using CBC mode:



```
[03/30/2015 07:31] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-cbc -in pic_original.bmp -out p
ic_cipher_ecb.bmp -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:31] seed@ubuntu:~/Bharat/Lab6$ █
```

pic_original encrypted using aes-128 and mode used is CBC

The same happens here and we cannot open the file.
So same as above we change the first 54 bits using ghex. After that the file can be seen as follows:

ciphered image

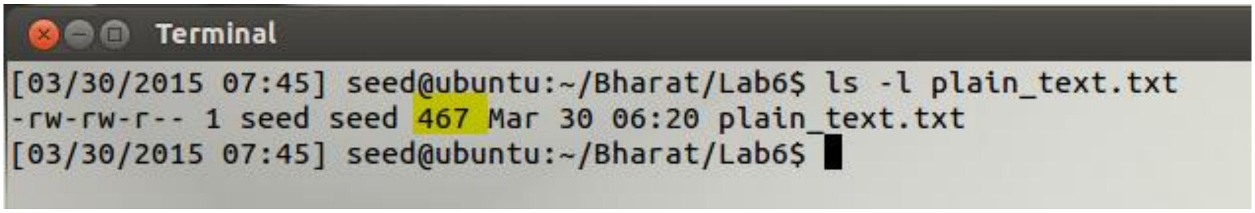
It is now completely scrambled.

ECB vs CBC: Observation:

We can say that ECB mode is quite simple and if there is data repetition it will be encrypted in the same way and thus we can derive some conclusion about the original file. This happens because the message is divided into blocks and each block is encrypted separately. However in CBC mode everything is completely scrambled and is not possible to derive any information about the original file. In CBC mode each block of plain text is XORed with the previous cipher text block before being encrypted.

3.3 Task 3: Encryption Mode – Corrupted Cipher Text

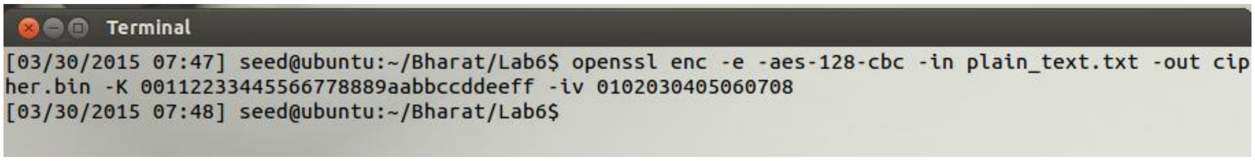
For this we need a file which is at least 64 bytes long. The file which we creates above is about 467 bytes long so we can safely use that:



```
[03/30/2015 07:45] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text.txt
-rw-rw-r-- 1 seed seed 467 Mar 30 06:20 plain_text.txt
[03/30/2015 07:45] seed@ubuntu:~/Bharat/Lab6$ 
```

file size is 467 bytes.

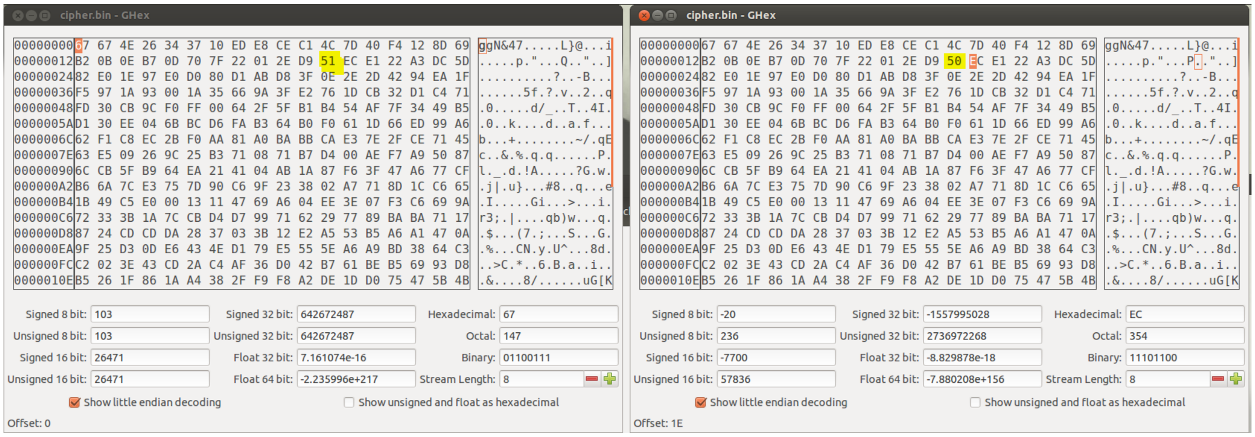
Next we encrypt the file using AES-128 and CBC mode.



```
[03/30/2015 07:47] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-cbc -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 07:48] seed@ubuntu:~/Bharat/Lab6$
```

AES-128 encryption

Now using a hex editor we shall change just one bit in the cipher.bin file

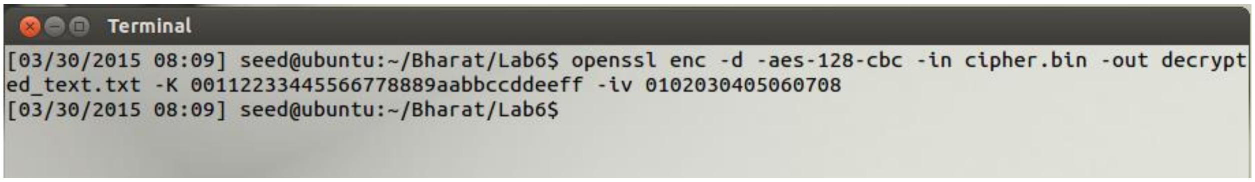


```
cipher.bin - GHex

00000000 67 67 4E 26 34 37 10 ED E8 CE C1 4C 7D 40 F4 12 8D 69  ggN&47.....L}@...i
00000012 B2 0B 0E B7 0D 70 7F 22 01 2E D9 51 EC E1 22 A3 DC 5D  .....p."...Q.."..]
00000024 82 E0 1E 97 E0 D0 80 D1 AB D8 3F 0E 2E 2D 42 94 EA 1F  ..........?..-B...
00000036 F5 97 1A 93 00 1A 35 66 9A 3F E2 76 1D CB 32 D1 C4 71  ......5f.?.v..2..q
00000048 FD 30 CB 9C F0 FF 00 64 2F 5F B1 B4 54 AF 7F 34 49 B5  .0.....d/_..T..4I.
0000005A D1 30 EE 04 6B BC D6 FA B3 64 B0 F0 61 1D 66 ED 99 A6  .0..k....d..a.f...
0000006C 62 F1 C8 EC 2B F0 AA 81 A0 BA BB CA E3 7E 2F CE 71 45  b...+........~/.qE
0000007E 63 E5 09 26 9C 25 B3 71 08 71 B7 D4 00 AE F7 A9 50 87  c..&.%.q.q......P.
00000090 6C CB 5F B9 64 EA 21 41 04 AB 1A 87 F6 3F 47 A6 77 CF  l._.d.!A.....?G.w.
000000A2 B6 6A 7C E3 75 7D 90 C6 9F 23 38 02 A7 71 8D 1C C6 65  .j|.u}...#8..q...e
000000B4 1B 49 C5 E0 00 13 11 47 69 A6 04 EE 3E 07 F3 C6 69 9A  .I.....Gi...>...i.
000000C6 72 33 3B 1A 7C CB D4 D7 99 71 62 29 77 89 BA BA 71 17  r3;.|....qb)w...q.
000000D8 87 24 CD CD DA 28 37 03 3B 12 E2 A5 53 B5 A6 A1 47 0A  .$...(7.;...S...G.
000000EA 9F 25 D3 0D E6 43 4E D1 79 E5 55 5E A6 A9 BD 38 64 C3  .%...CN.y.U^...8d.
000000FC C2 02 3E 43 CD 2A C4 AF 36 D0 42 B7 61 BE B5 69 93 D8  ..>C.*..6.B.a..i..
0000010E B5 26 1F 86 1A A4 38 2F F9 F8 A2 DE 1D D0 75 47 5B 4B  .&....8/......uG[K

  Signed 8 bit: 103          Signed 32 bit: 642672487       Hexadecimal: 67
Unsigned 8 bit: 103        Unsigned 32 bit: 642672487             Octal: 147
 Signed 16 bit: 26471         Float 32 bit: 7.161074e-16         Binary: 01100111
Unsigned 16 bit: 26471        Float 64 bit: -2.235996e+217  Stream Length: 8

☑ Show little endian decoding        ☐ Show unsigned and float as hexadecimal

Offset: 0
```

```
cipher.bin - GHex

00000000 67 67 4E 26 34 37 10 ED E8 CE C1 4C 7D 40 F4 12 8D 69  ggN&47.....L}@...i
00000012 B2 0B 0E B7 0D 70 7F 22 01 2E D9 50 EC E1 22 A3 DC 5D  .....p."...P.."..]
00000024 82 E0 1E 97 E0 D0 80 D1 AB D8 3F 0E 2E 2D 42 94 EA 1F  ..........?..-B...
00000036 F5 97 1A 93 00 1A 35 66 9A 3F E2 76 1D CB 32 D1 C4 71  ......5f.?.v..2..q
00000048 FD 30 CB 9C F0 FF 00 64 2F 5F B1 B4 54 AF 7F 34 49 B5  .0.....d/_..T..4I.
0000005A D1 30 EE 04 6B BC D6 FA B3 64 B0 F0 61 1D 66 ED 99 A6  .0..k....d..a.f...
0000006C 62 F1 C8 EC 2B F0 AA 81 A0 BA BB CA E3 7E 2F CE 71 45  b...+........~/.qE
0000007E 63 E5 09 26 9C 25 B3 71 08 71 B7 D4 00 AE F7 A9 50 87  c..&.%.q.q......P.
00000090 6C CB 5F B9 64 EA 21 41 04 AB 1A 87 F6 3F 47 A6 77 CF  l._.d.!A.....?G.w.
000000A2 B6 6A 7C E3 75 7D 90 C6 9F 23 38 02 A7 71 8D 1C C6 65  .j|.u}...#8..q...e
000000B4 1B 49 C5 E0 00 13 11 47 69 A6 04 EE 3E 07 F3 C6 69 9A  .I.....Gi...>...i.
000000C6 72 33 3B 1A 7C CB D4 D7 99 71 62 29 77 89 BA BA 71 17  r3;.|....qb)w...q.
000000D8 87 24 CD CD DA 28 37 03 3B 12 E2 A5 53 B5 A6 A1 47 0A  .$...(7.;...S...G.
000000EA 9F 25 D3 0D E6 43 4E D1 79 E5 55 5E A6 A9 BD 38 64 C3  .%...CN.y.U^...8d.
000000FC C2 02 3E 43 CD 2A C4 AF 36 D0 42 B7 61 BE B5 69 93 D8  ..>C.*..6.B.a..i..
0000010E B5 26 1F 86 1A A4 38 2F F9 F8 A2 DE 1D D0 75 47 5B 4B  .&....8/......uG[K

  Signed 8 bit: -20          Signed 32 bit: -1557995028     Hexadecimal: EC
Unsigned 8 bit: 236        Unsigned 32 bit: 2736972268            Octal: 354
 Signed 16 bit: -7700         Float 32 bit: -8.829878e-18        Binary: 11101100
Unsigned 16 bit: 57836        Float 64 bit: -7.880208e+156  Stream Length: 8

☑ Show little endian decoding        ☐ Show unsigned and float as hexadecimal

Offset: 1E
```

30th bit (highlighted) is changed to simulate the condition as corrupted.

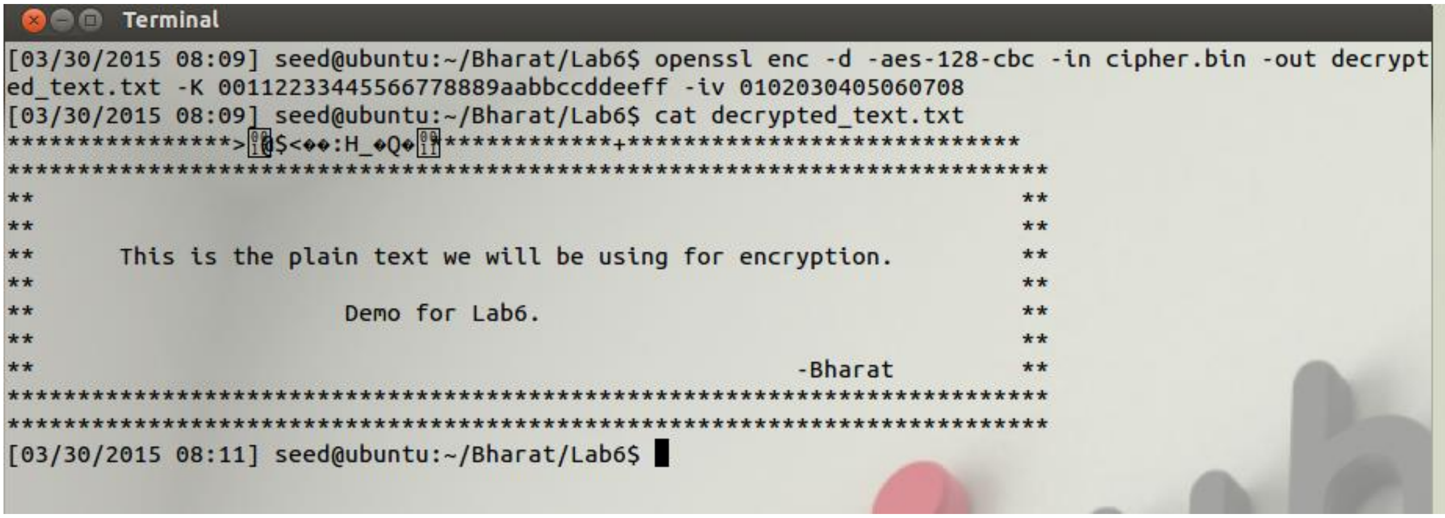
Now we try to decrypt the corrupted file using AES-128 bit, the original iv and the original key.



```
[03/30/2015 08:09] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -aes-128-cbc -in cipher.bin -out decrypt
ed_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:09] seed@ubuntu:~/Bharat/Lab6$
```

decrypt using the correct key and IV

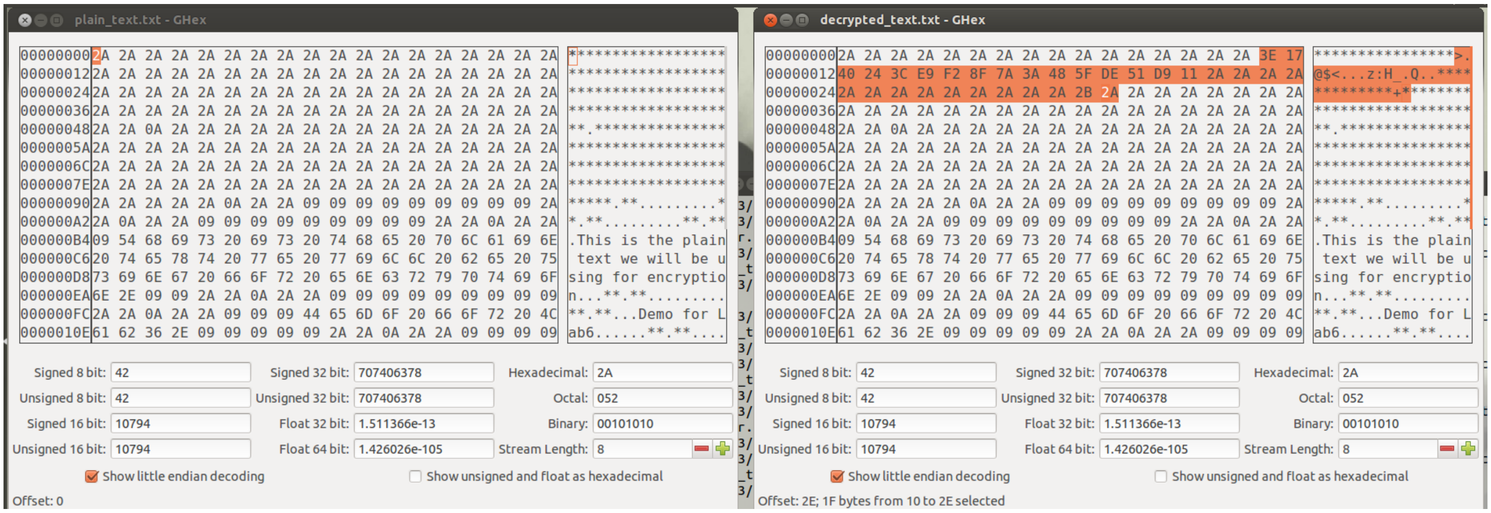
Not lets open the decrypted text and see what’s inside.



```
[03/30/2015 08:09] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -aes-128-cbc -in cipher.bin -out decrypt
ed_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:09] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
****************>▯@$<��:H_�Q�▯*************+****************************
**************************************************************************
**                                                                      **
**                                                                      **
**      This is the plain text we will be using for encryption.         **
**                                                                      **
**                      Demo for Lab6.                                  **
**                                                                      **
**                                                      -Bharat         **
**************************************************************************
**************************************************************************
[03/30/2015 08:11] seed@ubuntu:~/Bharat/Lab6$ █
```

decrypted text

As we can see the first line has some scrambled text but other than that the entire file is decrypted successfully.



```
000000002A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000122A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000242A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000362A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000482A 2A 0A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  **.***************
0000005A2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
0000006C2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
0000007E2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000902A 2A 2A 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A  *****.**.........*
000000A22A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A 2A 0A 2A 2A  *.**.........**.**
000000B409 54 68 69 73 20 69 73 20 74 68 65 20 70 6C 61 69 6E  .This is the plain
000000C620 74 65 78 74 20 77 65 20 77 69 6C 6C 20 62 65 20 75   text we will be u
000000D873 69 6E 67 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F  sing for encryptio
000000EA6E 2E 09 09 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09  n...**.**.........
000000FC2A 2A 0A 2A 2A 09 09 09 44 65 6D 6F 20 66 6F 72 20 4C  **.**...Demo for L
0000010E61 62 36 2E 09 09 09 09 09 2A 2A 0A 2A 2A 09 09 09 09  ab6......**.**....
```

Signed 8 bit: 42 | Signed 32 bit: 707406378 | Hexadecimal: 2A
Unsigned 8 bit: 42 | Unsigned 32 bit: 707406378 | Octal: 052
Signed 16 bit: 10794 | Float 32 bit: 1.511366e-13 | Binary: 00101010
Unsigned 16 bit: 10794 | Float 64 bit: 1.426026e-105 | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0

```
000000002A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 3E 17  ****************>.
0000001240 24 3C E9 F2 8F 7A 3A 48 5F DE 51 D9 11 2A 2A 2A 2A  @$<...z:H_.Q..****
000000242A 2A 2A 2A 2A 2A 2A 2A 2A 2B 2A 2A 2A 2A 2A 2A 2A 2A  *********+********
000000362A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000482A 2A 0A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  **.***************
0000005A2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
0000006C2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
0000007E2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000902A 2A 2A 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A  *****.**.........*
000000A22A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A 2A 0A 2A 2A  *.**.........**.**
000000B409 54 68 69 73 20 69 73 20 74 68 65 20 70 6C 61 69 6E  .This is the plain
000000C620 74 65 78 74 20 77 65 20 77 69 6C 6C 20 62 65 20 75   text we will be u
000000D873 69 6E 67 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F  sing for encryptio
000000EA6E 2E 09 09 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09  n...**.**.........
000000FC2A 2A 0A 2A 2A 09 09 09 44 65 6D 6F 20 66 6F 72 20 4C  **.**...Demo for L
0000010E61 62 36 2E 09 09 09 09 09 2A 2A 0A 2A 2A 09 09 09 09  ab6......**.**....
```

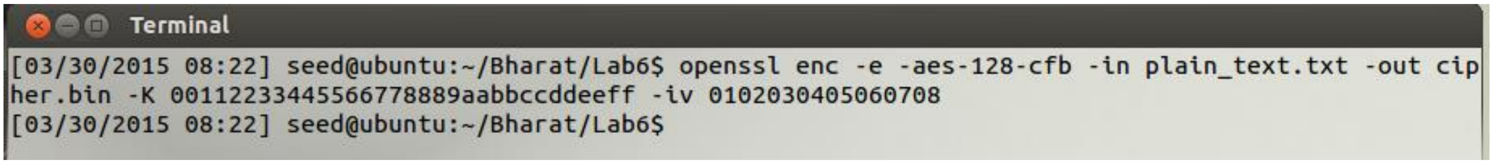
Signed 8 bit: 42 | Signed 32 bit: 707406378 | Hexadecimal: 2A
Unsigned 8 bit: 42 | Unsigned 32 bit: 707406378 | Octal: 052
Signed 16 bit: 10794 | Float 32 bit: 1.511366e-13 | Binary: 00101010
Unsigned 16 bit: 10794 | Float 64 bit: 1.426026e-105 | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 2E; 1F bytes from 10 to 2E selected

The highlighted blocks were not decrypted as expected.

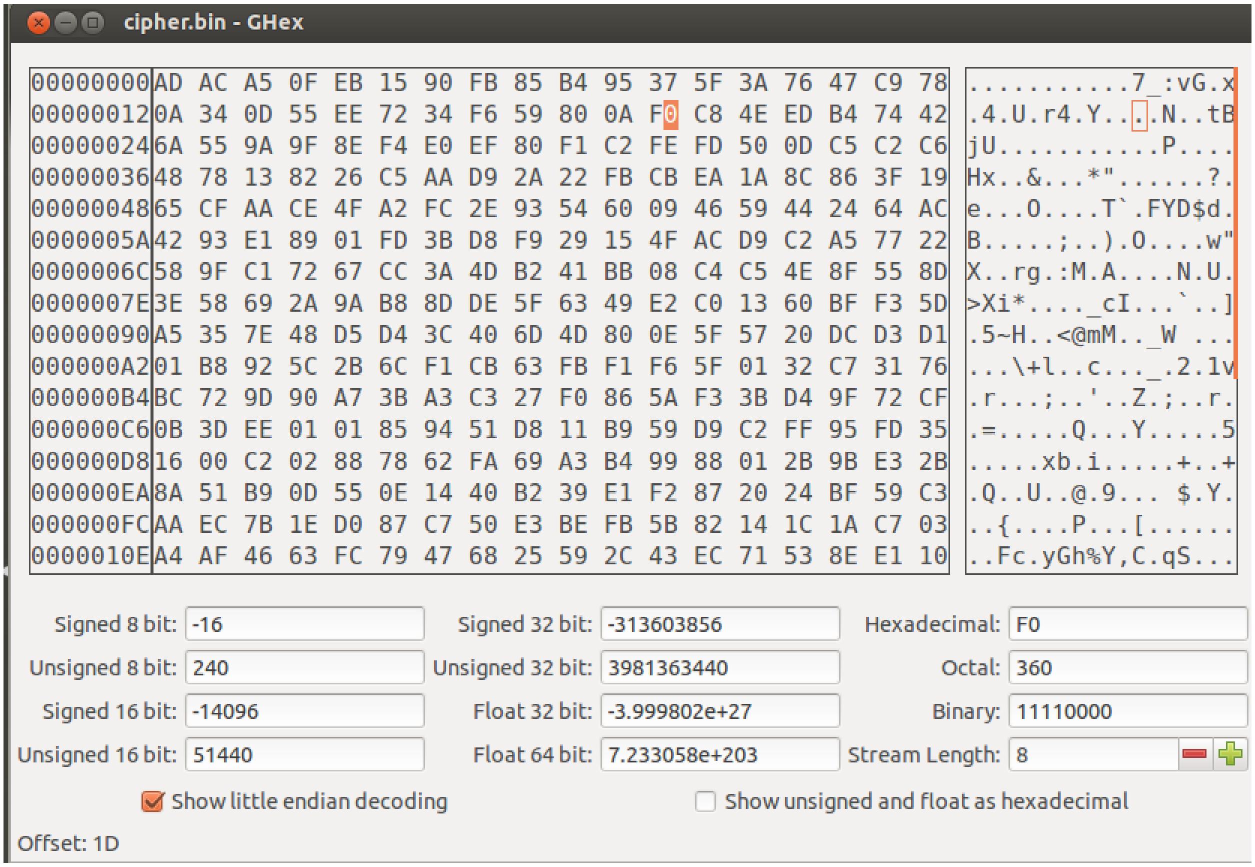
Now let us try to repeat the same using **‘CFB’** mode.



```
[03/30/2015 08:22] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-cfb -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:22] seed@ubuntu:~/Bharat/Lab6$
```

encrypted using CFB mode

Now same as above, we corrupt the 30th bit.



```
00000000 AD AC A5 0F EB 15 90 FB 85 B4 95 37 5F 3A 76 47 C9 78  ...........7_:vG.x
00000012 0A 34 0D 55 EE 72 34 F6 59 80 0A F0 C8 4E ED B4 74 42  .4.U.r4.Y...N..tB
00000024 6A 55 9A 9F 8E F4 E0 EF 80 F1 C2 FE FD 50 0D C5 C2 C6  jU...........P....
00000036 48 78 13 82 26 C5 AA D9 2A 22 FB CB EA 1A 8C 86 3F 19  Hx..&...*"......?.
00000048 65 CF AA CE 4F A2 FC 2E 93 54 60 09 46 59 44 24 64 AC  e...O....T`.FYD$d.
0000005A 42 93 E1 89 01 FD 3B D8 F9 29 15 4F AC D9 C2 A5 77 22  B.....;..).O....w"
0000006C 58 9F C1 72 67 CC 3A 4D B2 41 BB 08 C4 C5 4E 8F 55 8D  X..rg.:M.A....N.U.
0000007E 3E 58 69 2A 9A B8 8D DE 5F 63 49 E2 C0 13 60 BF F3 5D  >Xi*...._cI...`..]
00000090 A5 35 7E 48 D5 D4 3C 40 6D 4D 80 0E 5F 57 20 DC D3 D1  .5~H..<@mM.._W ...
000000A2 01 B8 92 5C 2B 6C F1 CB 63 FB F1 F6 5F 01 32 C7 31 76  ...\+l..c..._.2.1v
000000B4 BC 72 9D 90 A7 3B A3 C3 27 F0 86 5A F3 3B D4 9F 72 CF  .r...;..'..Z.;..r.
000000C6 0B 3D EE 01 01 85 94 51 D8 11 B9 59 D9 C2 FF 95 FD 35  .=.....Q...Y.....5
000000D8 16 00 C2 02 88 78 62 FA 69 A3 B4 99 88 01 2B 9B E3 2B  .....xb.i.....+..+
000000EA 8A 51 B9 0D 55 0E 14 40 B2 39 E1 F2 87 20 24 BF 59 C3  .Q..U..@.9... $.Y.
000000FC AA EC 7B 1E D0 87 C7 50 E3 BE FB 5B 82 14 1C 1A C7 03  ..{....P...[......
0000010E A4 AF 46 63 FC 79 47 68 25 59 2C 43 EC 71 53 8E E1 10  ..Fc.yGh%Y,C.qS...
```

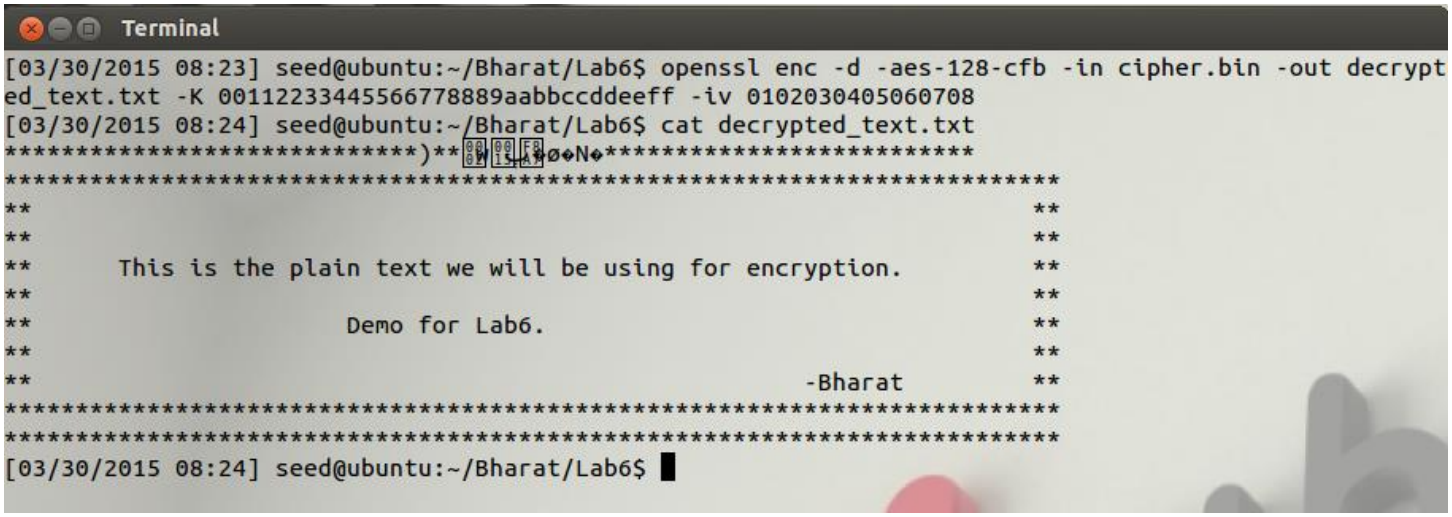
Signed 8 bit:	-16	Signed 32 bit:	-313603856	Hexadecimal:	F0
Unsigned 8 bit:	240	Unsigned 32 bit:	3981363440	Octal:	360
Signed 16 bit:	-14096	Float 32 bit:	-3.999802e+27	Binary:	11110000
Unsigned 16 bit:	51440	Float 64 bit:	7.233058e+203	Stream Length:	8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 1D

30th bit is corrupted from F4 to F0

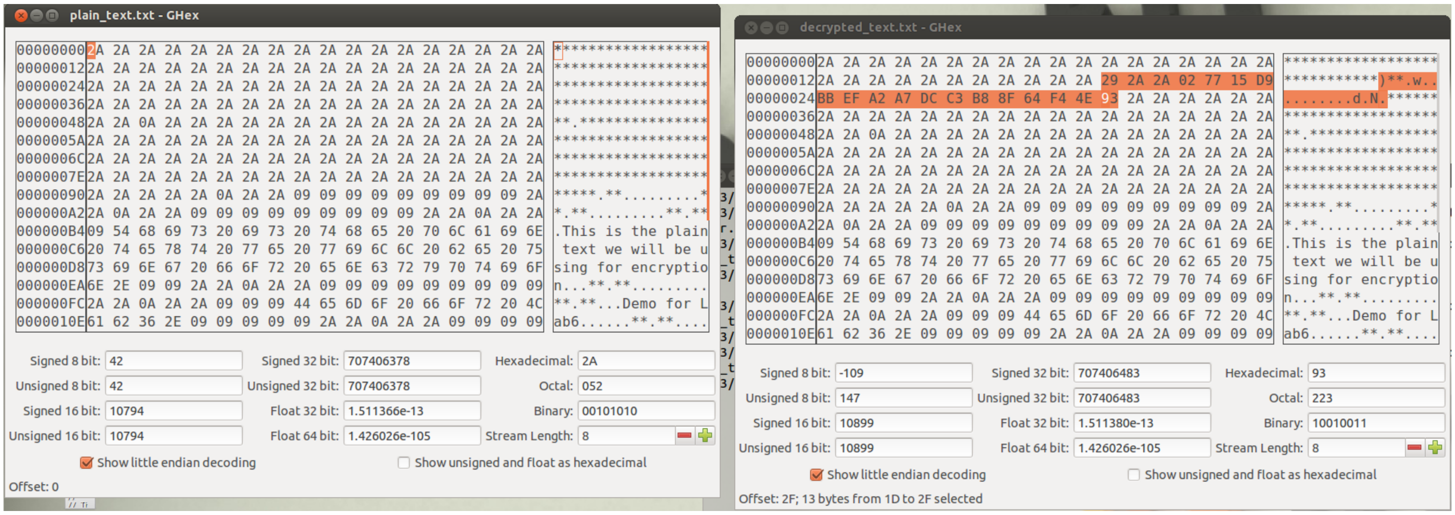
Next we decrypt the cipher text using the same key and iv.



```
[03/30/2015 08:23] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -aes-128-cfb -in cipher.bin -out decrypt
ed_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:24] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
*****************************)**WiSA�ø�N�**************************
**************************************************************************
**                                                                      **
**                                                                      **
**      This is the plain text we will be using for encryption.         **
**                                                                      **
**                      Demo for Lab6.                                  **
**                                                                      **
**                                                      -Bharat         **
**************************************************************************
**************************************************************************
[03/30/2015 08:24] seed@ubuntu:~/Bharat/Lab6$ 
```

decrypted text

As we can see when there is even a single bit corrupted the whole block becomes useless



plain_text.txt - GHex

```
000000002A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000122A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000242A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000362A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000482A 2A 0A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A **.***************
0000005A2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
0000006C2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
0000007E2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000902A 2A 2A 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A *****.**.........*
000000A22A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A 2A 0A 2A 2A *.**.........**.**
000000B409 54 68 69 73 20 69 73 20 74 68 65 20 70 6C 61 69 6E .This is the plain
000000C620 74 65 78 74 20 77 65 20 77 69 6C 6C 20 62 65 20 75  text we will be u
000000D873 69 6E 67 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F sing for encryptio
000000EA6E 2E 09 09 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 n...**.**.........
000000FC2A 2A 0A 2A 2A 09 09 09 44 65 6D 6F 20 66 6F 72 20 4C **.**...Demo for L
0000010E61 62 36 2E 09 09 09 09 09 2A 2A 0A 2A 2A 09 09 09 09 ab6......**.**....
```

Signed 8 bit: 42 | Signed 32 bit: 707406378 | Hexadecimal: 2A
Unsigned 8 bit: 42 | Unsigned 32 bit: 707406378 | Octal: 052
Signed 16 bit: 10794 | Float 32 bit: 1.511366e-13 | Binary: 00101010
Unsigned 16 bit: 10794 | Float 64 bit: 1.426026e-105 | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0

decrypted_text.txt - GHex

```
000000002A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000122A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 29 2A 2A 02 77 15 D9 ***********)**.w..
00000024BB EF A2 A7 DC C3 B8 8F 64 F4 4E 93 2A 2A 2A 2A 2A 2A ........d.N.******
000000362A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000482A 2A 0A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A **.***************
0000005A2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
0000006C2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
0000007E2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000902A 2A 2A 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A *****.**.........*
000000A22A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A 2A 0A 2A 2A *.**.........**.**
000000B409 54 68 69 73 20 69 73 20 74 68 65 20 70 6C 61 69 6E .This is the plain
000000C620 74 65 78 74 20 77 65 20 77 69 6C 6C 20 62 65 20 75  text we will be u
000000D873 69 6E 67 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F sing for encryptio
000000EA6E 2E 09 09 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 n...**.**.........
000000FC2A 2A 0A 2A 2A 09 09 09 44 65 6D 6F 20 66 6F 72 20 4C **.**...Demo for L
0000010E61 62 36 2E 09 09 09 09 09 2A 2A 0A 2A 2A 09 09 09 09 ab6......**.**....
```

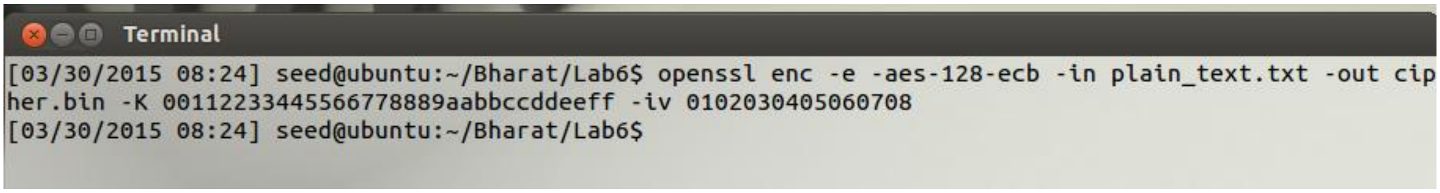
Signed 8 bit: -109 | Signed 32 bit: 707406483 | Hexadecimal: 93
Unsigned 8 bit: 147 | Unsigned 32 bit: 707406483 | Octal: 223
Signed 16 bit: 10899 | Float 32 bit: 1.511380e-13 | Binary: 10010011
Unsigned 16 bit: 10899 | Float 64 bit: 1.426026e-105 | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 2F; 13 bytes from 1D to 2F selected

The highlighted blocks were corrupted

Now let us repeat using ‘ECB’ mode

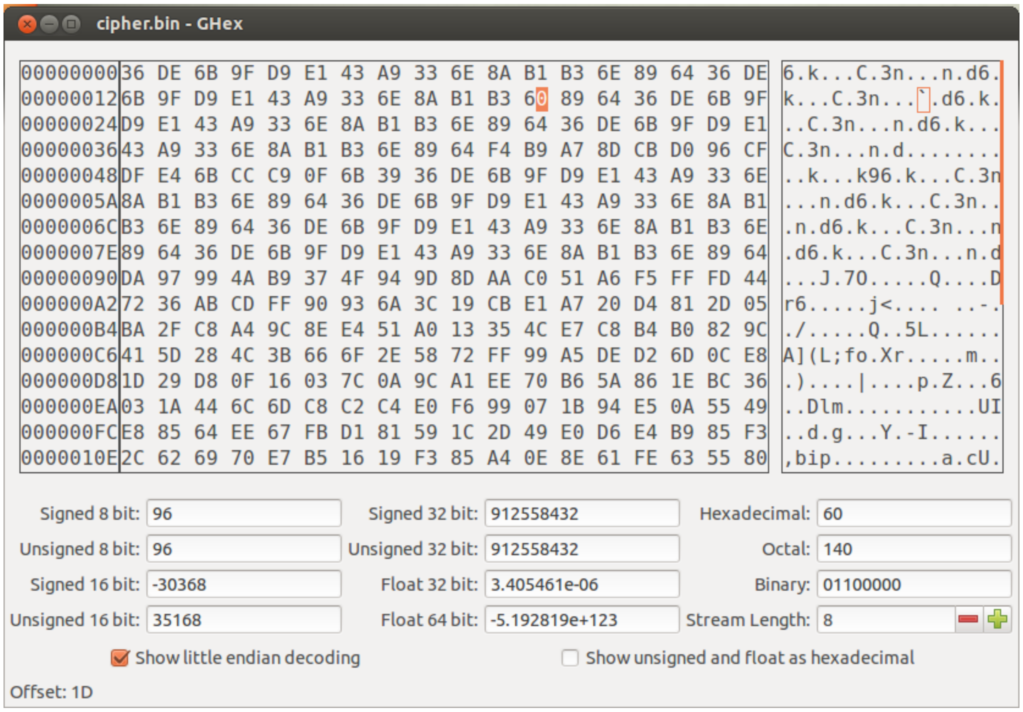


Terminal

```
[03/30/2015 08:24] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-ecb -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:24] seed@ubuntu:~/Bharat/Lab6$
```

encryption using ecb mode

Next we corrupt the 30th bit:



cipher.bin - GHex

```
0000000036 DE 6B 9F D9 E1 43 A9 33 6E 8A B1 B3 6E 89 64 36 DE 6.k...C.3n...n.d6.
000000126B 9F D9 E1 43 A9 33 6E 8A B1 B3 60 89 64 36 DE 6B 9F k...C.3n...`.d6.k.
00000024D9 E1 43 A9 33 6E 8A B1 B3 6E 89 64 36 DE 6B 9F D9 E1 ..C.3n...n.d6.k...
0000003643 A9 33 6E 8A B1 B3 6E 89 64 F4 B9 A7 8D CB D0 96 CF C.3n...n.d........
00000048DF E4 6B CC C9 0F 6B 39 36 DE 6B 9F D9 E1 43 A9 33 6E ..k...k96.k...C.3n
0000005A8A B1 B3 6E 89 64 36 DE 6B 9F D9 E1 43 A9 33 6E 8A B1 ...n.d6.k...C.3n..
0000006CB3 6E 89 64 36 DE 6B 9F D9 E1 43 A9 33 6E 8A B1 B3 6E .n.d6.k...C.3n...n
0000007E89 64 36 DE 6B 9F D9 E1 43 A9 33 6E 8A B1 B3 6E 89 64 .d6.k...C.3n...n.d
00000090DA 97 99 4A B9 37 4F 94 9D 8D AA C0 51 A6 F5 FF FD 44 ...J.7O.....Q....D
000000A272 36 AB CD FF 90 93 6A 3C 19 CB E1 A7 20 D4 81 2D 05 r6.....j<.... ..-.
000000B4BA 2F C8 A4 9C 8E E4 51 A0 13 35 4C E7 C8 B4 B0 82 9C ./.....Q..5L......
000000C641 5D 28 4C 3B 66 6F 2E 58 72 FF 99 A5 DE D2 6D 0C E8 A](L;fo.Xr.....m..
000000D81D 29 D8 0F 16 03 7C 0A 9C A1 EE 70 B6 5A 86 1E BC 36 .)....|....p.Z...6
000000EA03 1A 44 6C 6D C8 C2 C4 E0 F6 99 07 1B 94 E5 0A 55 49 ..Dlm...........UI
000000FCE8 85 64 EE 67 FB D1 81 59 1C 2D 49 E0 D6 E4 B9 85 F3 ..d.g...Y.-I......
0000010E2C 62 69 70 E7 B5 16 19 F3 85 A4 0E 8E 61 FE 63 55 80 ,bip.........a.cU.
```

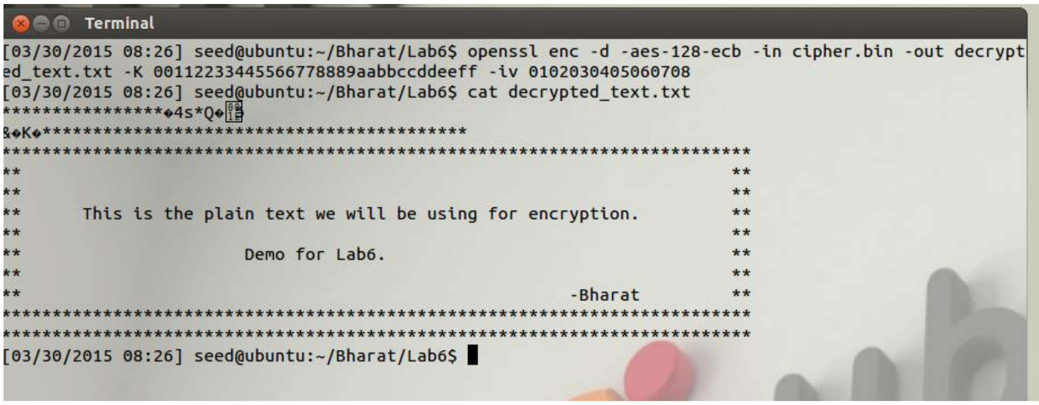
Signed 8 bit: 96 | Signed 32 bit: 912558432 | Hexadecimal: 60
Unsigned 8 bit: 96 | Unsigned 32 bit: 912558432 | Octal: 140
Signed 16 bit: -30368 | Float 32 bit: 3.405461e-06 | Binary: 01100000
Unsigned 16 bit: 35168 | Float 64 bit: -5.192819e+123 | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 1D

30th bit corrupted from 6E to 60

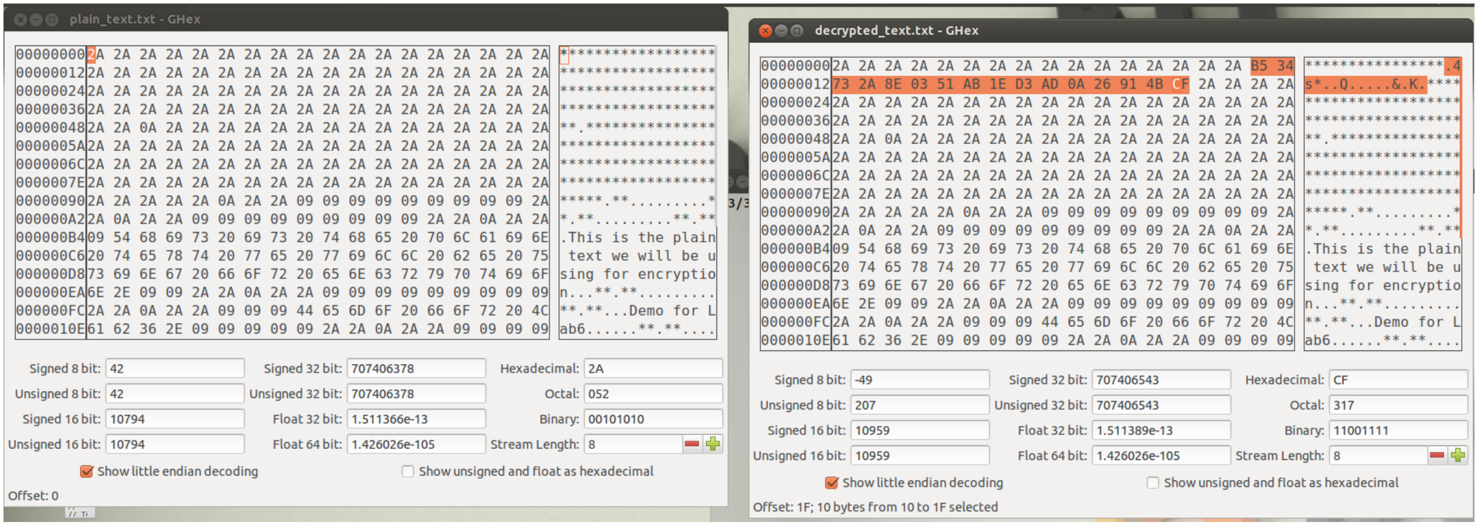
Next we decrypt using the same key and IV



```
[03/30/2015 08:26] seed@ubuntu:~/Bharat/Lab6$ openssl enc -d -aes-128-ecb -in cipher.bin -out decrypt
ed_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:26] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
****************◆4s*Q◆
&◆K◆****************************************
**************************************************************************
**                                                                      **
**                                                                      **
**      This is the plain text we will be using for encryption.         **
**                                                                      **
**                      Demo for Lab6.                                  **
**                                                                      **
**                                                      -Bharat         **
**************************************************************************
**************************************************************************
[03/30/2015 08:26] seed@ubuntu:~/Bharat/Lab6$ █
```

decrypted text

The block change can be seen as follows:

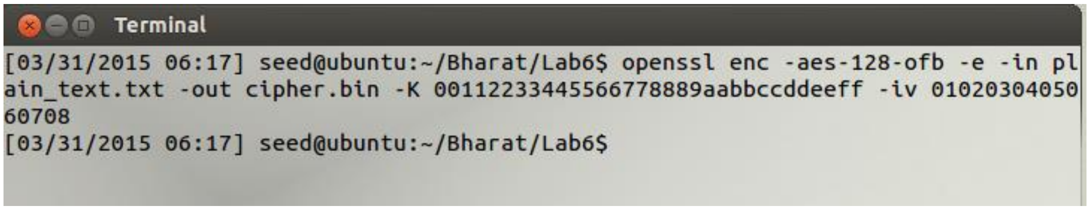


```
plain_text.txt - GHex

000000002A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000122A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000242A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000362A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000482A 2A 0A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  **.***************
0000005A2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
0000006C2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
0000007E2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000902A 2A 2A 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A  *****.**.........*
000000A22A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A 2A 0A 2A 2A  *.**.........**.**
000000B409 54 68 69 73 20 69 73 20 74 68 65 20 70 6C 61 69 6E  .This is the plain
000000C620 74 65 78 74 20 77 65 20 77 69 6C 6C 20 62 65 20 75   text we will be u
000000D873 69 6E 67 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F  sing for encryptio
000000EA6E 2E 09 09 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09  n...**.**.........
000000FC2A 2A 0A 2A 2A 09 09 09 44 65 6D 6F 20 66 6F 72 20 4C  **.**...Demo for L
0000010E61 62 36 2E 09 09 09 09 09 2A 2A 0A 2A 2A 09 09 09 09  ab6......**.**....

    Signed 8 bit: 42          Signed 32 bit: 707406378       Hexadecimal: 2A
  Unsigned 8 bit: 42        Unsigned 32 bit: 707406378             Octal: 052
   Signed 16 bit: 10794        Float 32 bit: 1.511366e-13         Binary: 00101010
 Unsigned 16 bit: 10794        Float 64 bit: 1.426026e-105 Stream Length: 8

      ☑ Show little endian decoding          ☐ Show unsigned and float as hexadecimal
Offset: 0
```

```
decrypted_text.txt - GHex

000000002A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A B5 34  ****************.4
0000001273 2A 8E 03 51 AB 1E D3 AD 0A 26 91 4B CF 2A 2A 2A 2A  s*..Q.....&.K.****
000000242A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000362A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000482A 2A 0A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  **.***************
0000005A2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
0000006C2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
0000007E2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A  ******************
000000902A 2A 2A 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A  *****.**.........*
000000A22A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A 2A 0A 2A 2A  *.**.........**.**
000000B409 54 68 69 73 20 69 73 20 74 68 65 20 70 6C 61 69 6E  .This is the plain
000000C620 74 65 78 74 20 77 65 20 77 69 6C 6C 20 62 65 20 75   text we will be u
000000D873 69 6E 67 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F  sing for encryptio
000000EA6E 2E 09 09 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09  n...**.**.........
000000FC2A 2A 0A 2A 2A 09 09 09 44 65 6D 6F 20 66 6F 72 20 4C  **.**...Demo for L
0000010E61 62 36 2E 09 09 09 09 09 2A 2A 0A 2A 2A 09 09 09 09  ab6......**.**....

    Signed 8 bit: -49         Signed 32 bit: 707406543       Hexadecimal: CF
  Unsigned 8 bit: 207       Unsigned 32 bit: 707406543             Octal: 317
   Signed 16 bit: 10959        Float 32 bit: 1.511389e-13         Binary: 11001111
 Unsigned 16 bit: 10959        Float 64 bit: 1.426026e-105 Stream Length: 8

      ☑ Show little endian decoding          ☐ Show unsigned and float as hexadecimal
Offset: 1F; 10 bytes from 10 to 1F selected
```

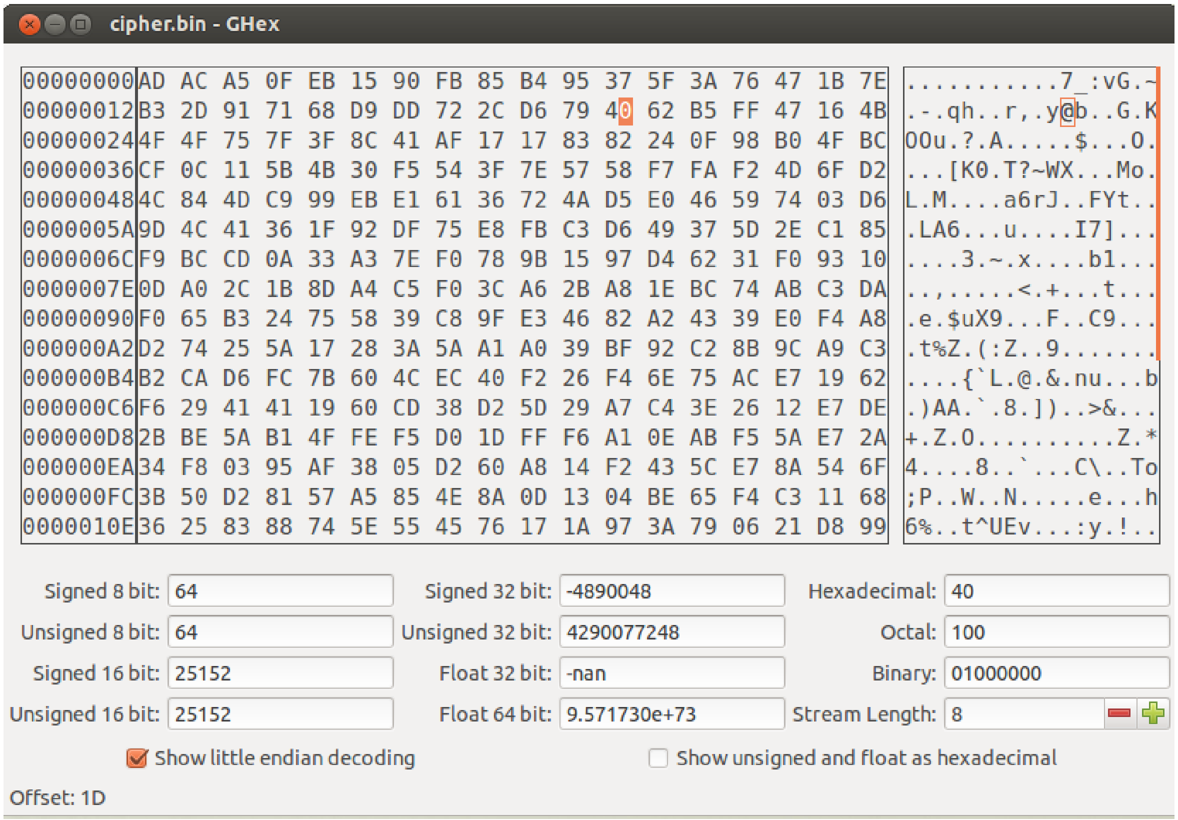
highlighted blocks were corrupted in ECB mode.

Now let us try to repeat the same using **‘OFB’** mode.



```
[03/31/2015 06:17] seed@ubuntu:~/Bharat/Lab6$ openssl enc -aes-128-ofb -e -in pl
ain_text.txt -out cipher.bin -K 00112233445566778889aabbccddeeff -iv 01020304050
60708
[03/31/2015 06:17] seed@ubuntu:~/Bharat/Lab6$
```

aes-128-ofb mode encryption



cipher.bin - GHex

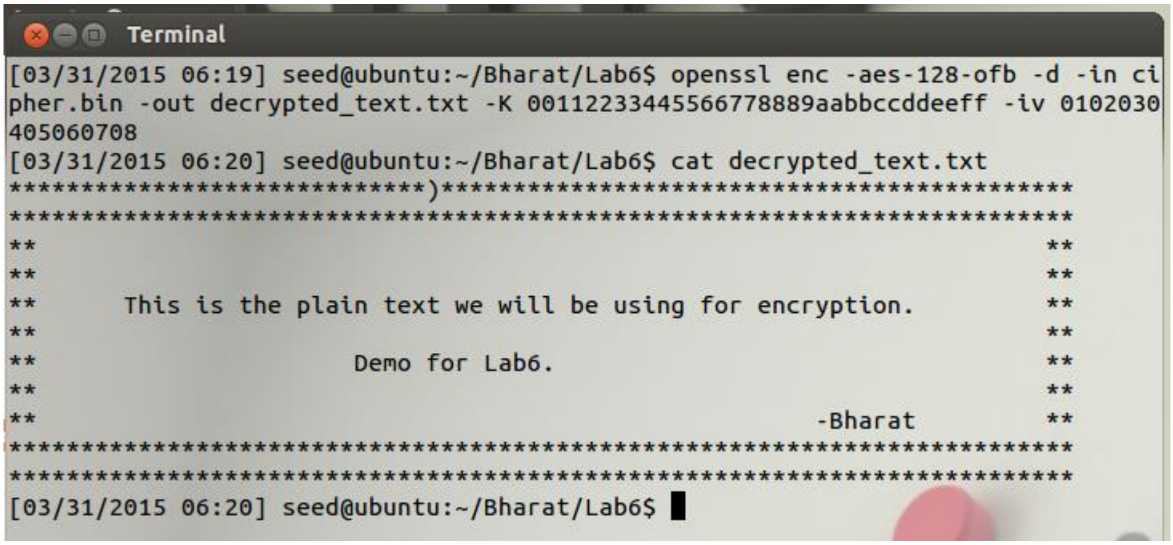
```
00000000AD AC A5 0F EB 15 90 FB 85 B4 95 37 5F 3A 76 47 1B 7E ...........7_:vG.~
00000012B3 2D 91 71 68 D9 DD 72 2C D6 79 40 62 B5 FF 47 16 4B .-.qh..r,.y@b..G.K
000000244F 4F 75 7F 3F 8C 41 AF 17 17 83 82 24 0F 98 B0 4F BC OOu.?.A.....$...O.
00000036CF 0C 11 5B 4B 30 F5 54 3F 7E 57 58 F7 FA F2 4D 6F D2 ...[K0.T?~WX...Mo.
000000484C 84 4D C9 99 EB E1 61 36 72 4A D5 E0 46 59 74 03 D6 L.M....a6rJ..FYt..
0000005A9D 4C 41 36 1F 92 DF 75 E8 FB C3 D6 49 37 5D 2E C1 85 .LA6...u....I7]...
0000006CF9 BC CD 0A 33 A3 7E F0 78 9B 15 97 D4 62 31 F0 93 10 ....3.~.x....b1...
0000007E0D A0 2C 1B 8D A4 C5 F0 3C A6 2B A8 1E BC 74 AB C3 DA ..,.....<.+...t...
00000090F0 65 B3 24 75 58 39 C8 9F E3 46 82 A2 43 39 E0 F4 A8 .e.$uX9...F..C9...
000000A2D2 74 25 5A 17 28 3A 5A A1 A0 39 BF 92 C2 8B 9C A9 C3 .t%Z.(:Z..9.......
000000B4B2 CA D6 FC 7B 60 4C EC 40 F2 26 F4 6E 75 AC E7 19 62 ....{`L.@.&.nu...b
000000C6F6 29 41 41 19 60 CD 38 D2 5D 29 A7 C4 3E 26 12 E7 DE .)AA.`.8.])..>&...
000000D82B BE 5A B1 4F FE F5 D0 1D FF F6 A1 0E AB F5 5A E7 2A +.Z.O..........Z.*
000000EA34 F8 03 95 AF 38 05 D2 60 A8 14 F2 43 5C E7 8A 54 6F 4....8..`...C\..To
000000FC3B 50 D2 81 57 A5 85 4E 8A 0D 13 04 BE 65 F4 C3 11 68 ;P..W..N.....e...h
0000010E36 25 83 88 74 5E 55 45 76 17 1A 97 3A 79 06 21 D8 99 6%..t^UEv...:y.!..
```

Signed 8 bit: 64
Signed 32 bit: -4890048
Hexadecimal: 40
Unsigned 8 bit: 64
Unsigned 32 bit: 4290077248
Octal: 100
Signed 16 bit: 25152
Float 32 bit: -nan
Binary: 01000000
Unsigned 16 bit: 25152
Float 64 bit: 9.571730e+73
Stream Length: 8

☑ Show little endian decoding
☐ Show unsigned and float as hexadecimal

Offset: 1D

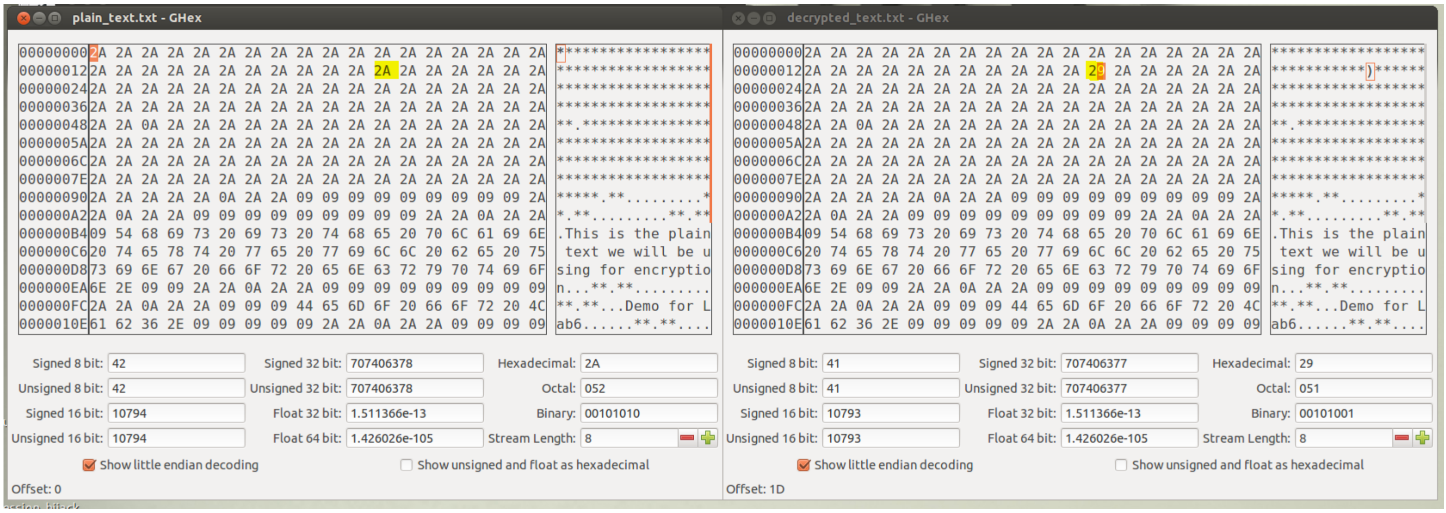
corrupted the 30th bit from 43 to 40



Terminal

```
[03/31/2015 06:19] seed@ubuntu:~/Bharat/Lab6$ openssl enc -aes-128-ofb -d -in ci
pher.bin -out decrypted_text.txt -K 00112233445566778889aabbccddeeff -iv 0102030
405060708
[03/31/2015 06:20] seed@ubuntu:~/Bharat/Lab6$ cat decrypted_text.txt
*****************************)*****************************************
***********************************************************************
**                                                                   **
**                                                                   **
**      This is the plain text we will be using for encryption.      **
**                                                                   **
**                      Demo for Lab6.                               **
**                                                                   **
**                                                      -Bharat      **
***********************************************************************
***********************************************************************
[03/31/2015 06:20] seed@ubuntu:~/Bharat/Lab6$ 
```

aes-128-ofb mode after decryption



plain_text.txt - GHex

```
000000002A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000122A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000242A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000362A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000482A 2A 0A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A **.***************
0000005A2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
0000006C2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
0000007E2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000902A 2A 2A 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A *****.**.........*
000000A22A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A 2A 0A 2A 2A *.**.........**.**
000000B409 54 68 69 73 20 69 73 20 74 68 65 20 70 6C 61 69 6E .This is the plain
000000C620 74 65 78 74 20 77 65 20 77 69 6C 6C 20 62 65 20 75  text we will be u
000000D873 69 6E 67 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F sing for encryptio
000000EA6E 2E 09 09 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 n...**.**.........
000000FC2A 2A 0A 2A 2A 09 09 09 44 65 6D 6F 20 66 6F 72 20 4C **.**...Demo for L
0000010E61 62 36 2E 09 09 09 09 09 2A 2A 0A 2A 2A 09 09 09 09 ab6......**.**....
```

Signed 8 bit: 42 | Signed 32 bit: 707406378 | Hexadecimal: 2A
Unsigned 8 bit: 42 | Unsigned 32 bit: 707406378 | Octal: 052
Signed 16 bit: 10794 | Float 32 bit: 1.511366e-13 | Binary: 00101010
Unsigned 16 bit: 10794 | Float 64 bit: 1.426026e-105 | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 0

decrypted_text.txt - GHex

```
000000002A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000122A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 29 2A 2A 2A 2A 2A 2A ***********)******
000000242A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000362A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000482A 2A 0A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A **.***************
0000005A2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
0000006C2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
0000007E2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A 2A ******************
000000902A 2A 2A 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A *****.**.........*
000000A22A 0A 2A 2A 09 09 09 09 09 09 09 09 09 2A 2A 0A 2A 2A *.**.........**.**
000000B409 54 68 69 73 20 69 73 20 74 68 65 20 70 6C 61 69 6E .This is the plain
000000C620 74 65 78 74 20 77 65 20 77 69 6C 6C 20 62 65 20 75  text we will be u
000000D873 69 6E 67 20 66 6F 72 20 65 6E 63 72 79 70 74 69 6F sing for encryptio
000000EA6E 2E 09 09 2A 2A 0A 2A 2A 09 09 09 09 09 09 09 09 09 n...**.**.........
000000FC2A 2A 0A 2A 2A 09 09 09 44 65 6D 6F 20 66 6F 72 20 4C **.**...Demo for L
0000010E61 62 36 2E 09 09 09 09 09 2A 2A 0A 2A 2A 09 09 09 09 ab6......**.**....
```

Signed 8 bit: 41 | Signed 32 bit: 707406377 | Hexadecimal: 29
Unsigned 8 bit: 41 | Unsigned 32 bit: 707406377 | Octal: 051
Signed 16 bit: 10793 | Float 32 bit: 1.511366e-13 | Binary: 00101001
Unsigned 16 bit: 10793 | Float 64 bit: 1.426026e-105 | Stream Length: 8

☑ Show little endian decoding ☐ Show unsigned and float as hexadecimal

Offset: 1D

The 30th bit that was corrupted was the only bit not recovered.

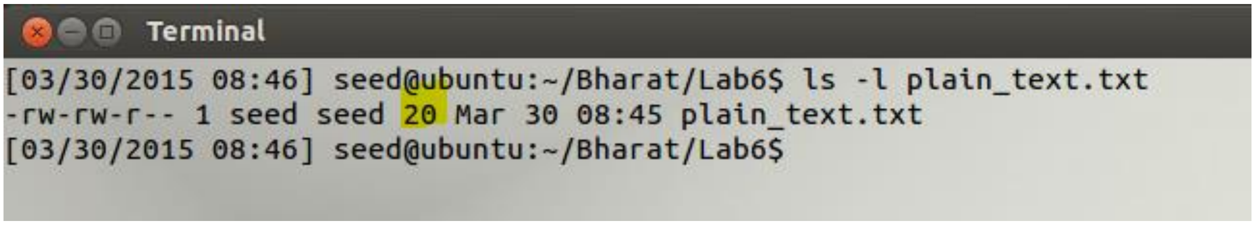
Observation:
CBC: all the data except the 2nd block (16 bytes) is recovered.
CFB: all the data except the 2nd and 3rd blocks is recovered.
ECB: all the data except the 2nd block is recovered.
OFB: all the data except the 30th byte in the 2nd block is recovered.

In CBC, CFB and ECB mode, if one bit is corrupted while transmitting the cipher text then it causes the complete corruption of a block. This happens because the block is XORed with the previously generated cipher text and then encrypted. However in OFB mode if one bit is corrupted the corresponding bit is only not recovered.

This makes us come to the conclusion that if more than one block is to be encrypted then ECB should not be used. Also CBC, CFB, ECB cannot be parallelized and OFB can be.

3.4 Task 4: Padding

To better understand this we shall create a small file. Something around 20 octets.

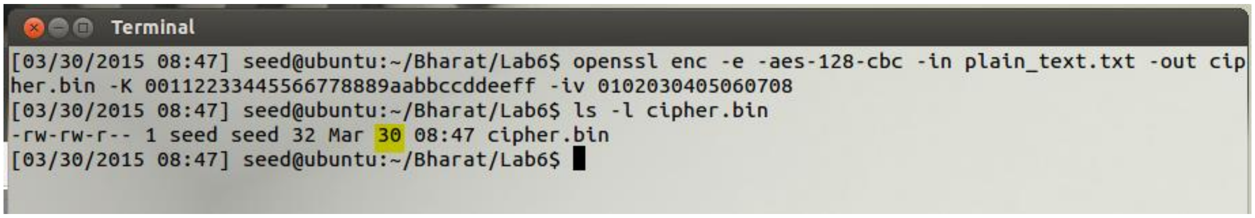


Terminal

```
[03/30/2015 08:46] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text.txt
-rw-rw-r-- 1 seed seed 20 Mar 30 08:45 plain_text.txt
[03/30/2015 08:46] seed@ubuntu:~/Bharat/Lab6$
```

file size 20 octets

Now lets encrypt the file using AES-CBC mode and check the cipher file size



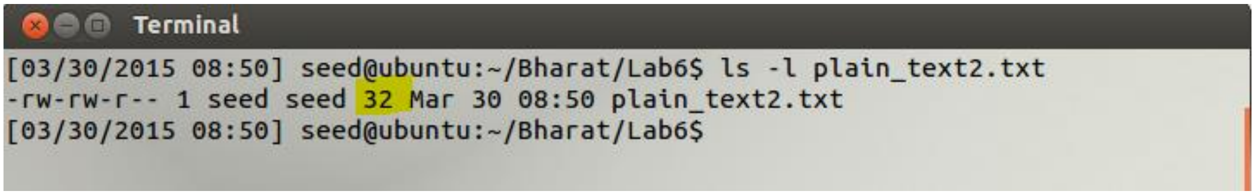
Terminal

```
[03/30/2015 08:47] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-cbc -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:47] seed@ubuntu:~/Bharat/Lab6$ ls -l cipher.bin
-rw-rw-r-- 1 seed seed 32 Mar 30 08:47 cipher.bin
[03/30/2015 08:47] seed@ubuntu:~/Bharat/Lab6$ █
```

cipher file size as 30

We can see that the file size has increased from 20 octets to 30 octets

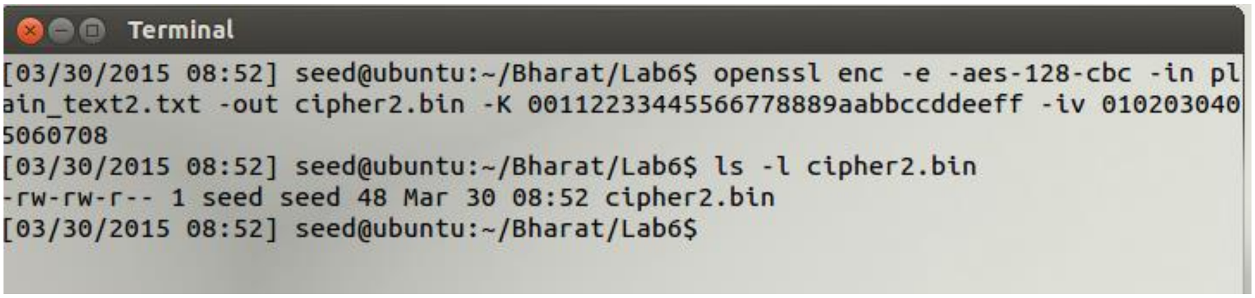
Next we repeat the procedure for a file with 32 octets.



```
[03/30/2015 08:50] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text2.txt
-rw-rw-r-- 1 seed seed 32 Mar 30 08:50 plain_text2.txt
[03/30/2015 08:50] seed@ubuntu:~/Bharat/Lab6$
```

file size 32 octets

Encrypting using AES-**CBC** block mode.



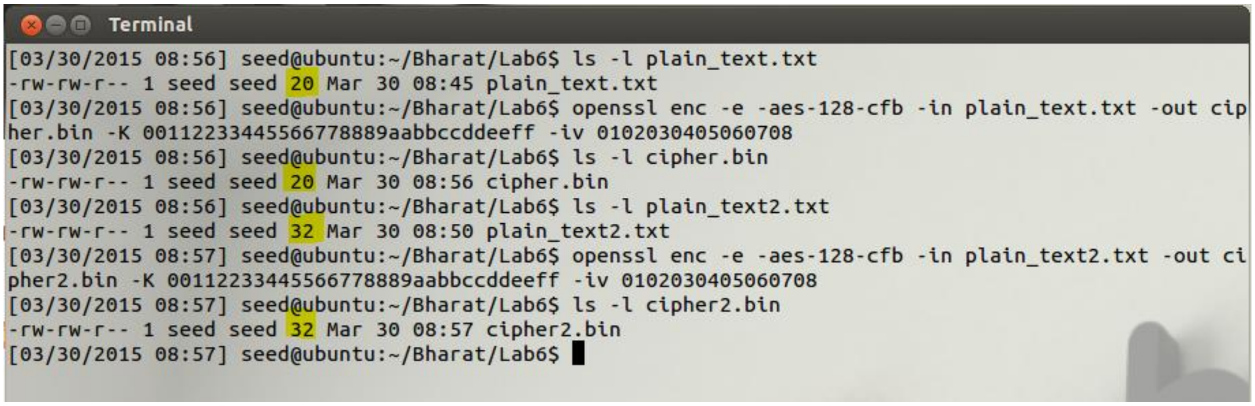
```
[03/30/2015 08:52] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-cbc -in pl
ain_text2.txt -out cipher2.bin -K 00112233445566778889aabbccddeeff -iv 010203040
5060708
[03/30/2015 08:52] seed@ubuntu:~/Bharat/Lab6$ ls -l cipher2.bin
-rw-rw-r-- 1 seed seed 48 Mar 30 08:52 cipher2.bin
[03/30/2015 08:52] seed@ubuntu:~/Bharat/Lab6$
```

encryption using AES-CBC

AS we can see the cipher file size has increased to 32 bytes.

Thus, we come to the conclusion that CBC requires an exact multiple of the block size. If the plain text does not meet this requirement then CBC will add the padding.

Let us repeat the experiment using ‘**CFB’** mode.

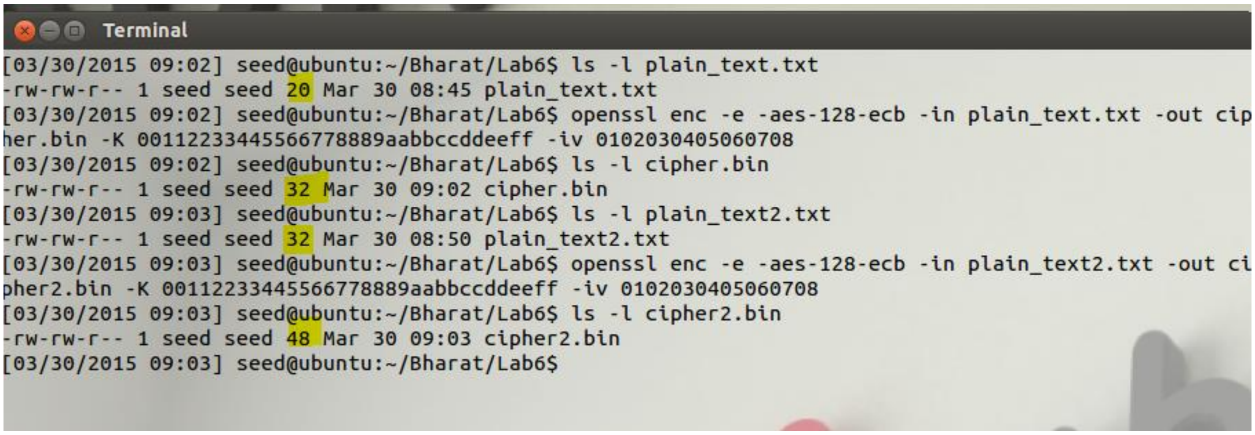


```
[03/30/2015 08:56] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text.txt
-rw-rw-r-- 1 seed seed 20 Mar 30 08:45 plain_text.txt
[03/30/2015 08:56] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-cfb -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:56] seed@ubuntu:~/Bharat/Lab6$ ls -l cipher.bin
-rw-rw-r-- 1 seed seed 20 Mar 30 08:56 cipher.bin
[03/30/2015 08:56] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text2.txt
-rw-rw-r-- 1 seed seed 32 Mar 30 08:50 plain_text2.txt
[03/30/2015 08:57] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-cfb -in plain_text2.txt -out ci
pher2.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 08:57] seed@ubuntu:~/Bharat/Lab6$ ls -l cipher2.bin
-rw-rw-r-- 1 seed seed 32 Mar 30 08:57 cipher2.bin
[03/30/2015 08:57] seed@ubuntu:~/Bharat/Lab6$ █
```

Padding cfb mode

We can see from the above screenshot that the plain text size and the cipher size are the same for both the 20 octet file and 32 octet file. Thus we can come to the conclusion that CFB does not require padding. CFB mode works by XORing the last few bytes with the first few bytes and thus does not require any additional padding. This is characteristic of the stream cipher.

Lets repeat the experiment using ‘**ECB**’ mode

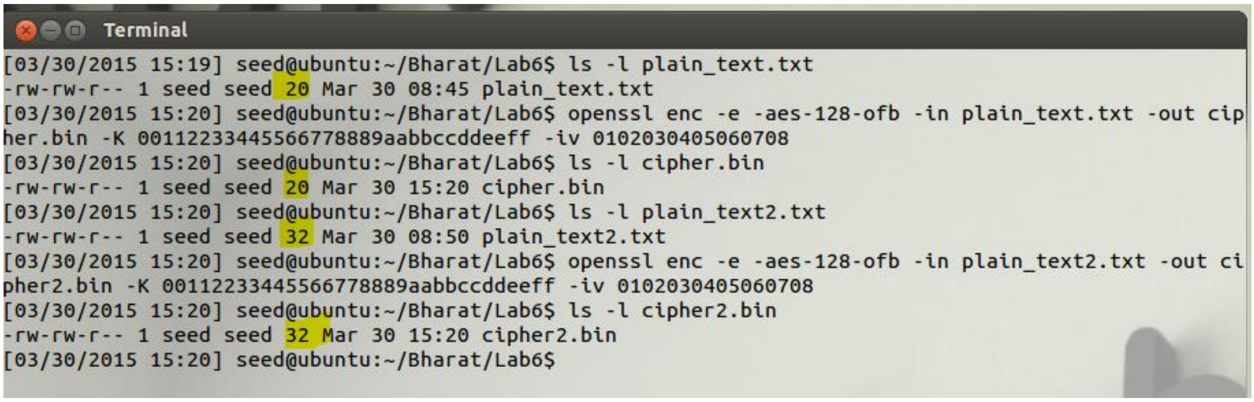


```
[03/30/2015 09:02] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text.txt
-rw-rw-r-- 1 seed seed 20 Mar 30 08:45 plain_text.txt
[03/30/2015 09:02] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-ecb -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 09:02] seed@ubuntu:~/Bharat/Lab6$ ls -l cipher.bin
-rw-rw-r-- 1 seed seed 32 Mar 30 09:02 cipher.bin
[03/30/2015 09:03] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text2.txt
-rw-rw-r-- 1 seed seed 32 Mar 30 08:50 plain_text2.txt
[03/30/2015 09:03] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-ecb -in plain_text2.txt -out ci
pher2.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 09:03] seed@ubuntu:~/Bharat/Lab6$ ls -l cipher2.bin
-rw-rw-r-- 1 seed seed 48 Mar 30 09:03 cipher2.bin
[03/30/2015 09:03] seed@ubuntu:~/Bharat/Lab6$
```

encryption using ECB mode

Here as we can see that the original file sizes and the cipher file sizes are different. We come to the conclusion that ECB mode needs padding. The block size is 16. And to 20 bytes 12 more bytes are padded to make it a multiple of the block size. However 32 is a multiple of 16. But an entire extra block of 16 bytes is added at the end. Thus ECB requires the final block to be padded for encryption.

Finally, let us do the encryption using **OFB** mode.



```
[03/30/2015 15:19] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text.txt
-rw-rw-r-- 1 seed seed 20 Mar 30 08:45 plain_text.txt
[03/30/2015 15:20] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-ofb -in plain_text.txt -out cip
her.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 15:20] seed@ubuntu:~/Bharat/Lab6$ ls -l cipher.bin
-rw-rw-r-- 1 seed seed 20 Mar 30 15:20 cipher.bin
[03/30/2015 15:20] seed@ubuntu:~/Bharat/Lab6$ ls -l plain_text2.txt
-rw-rw-r-- 1 seed seed 32 Mar 30 08:50 plain_text2.txt
[03/30/2015 15:20] seed@ubuntu:~/Bharat/Lab6$ openssl enc -e -aes-128-ofb -in plain_text2.txt -out ci
pher2.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/30/2015 15:20] seed@ubuntu:~/Bharat/Lab6$ ls -l cipher2.bin
-rw-rw-r-- 1 seed seed 32 Mar 30 15:20 cipher2.bin
[03/30/2015 15:20] seed@ubuntu:~/Bharat/Lab6$
```

As we can see in both the files the size of the plain text and the cipher text is the same proving that OFB mode does not require padding. This is because as with others such as cfb, this works like a stream cipher and XORs the plain text with the output. The last block(incomplete) is XORed with the first few bytes of the last block

3.5 Task 5: Programming using the Crypto Library

In this task we are asked to write a program to invoke the crypto library. We will be using the plain text and cipher text given to us and the dictionary list provided in the lab. Also we know that the initial vector (IV) is all zeros.

Our plain text buffer (plain_text_buffer) will contain the following data:
This is a top secret.

Our cipher text buffer (cipher_text_buffer) in hex format is: (64 bytes long)
8d20e5056a8d24d0462ce74e4904c1b5
13e10d1df4a2ef2ad4540fae1ca0aaf9

The **program** was written as follows:

```
/*Including the standard libraries*/
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
```

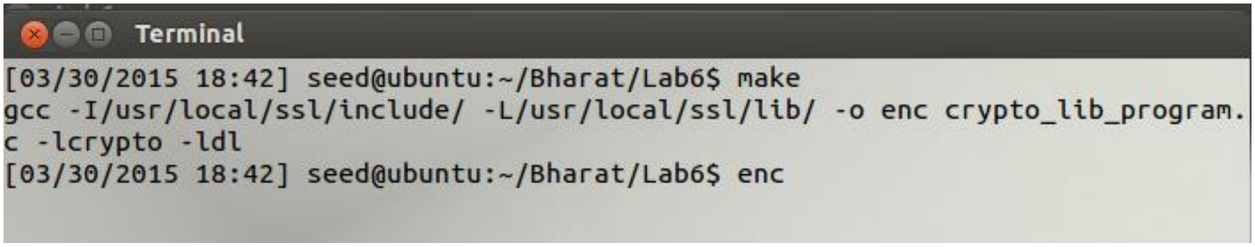
```
/*including openssl header files*/
#include <openssl/evp.h>

int main()
{
//function to load the dictionar file
printf("Loading dictionary into the program.\n");
loadDictionary("words.txt");
return 0;
}

int loadDictionary(char * dictionary_name)
{
FILE *dictionary_file;
dictionary_file = fopen (dictionary_name, "rt");
char dictionary_word[16];
int key_found_flag =0;

while(fgets(dictionary_word, 16, dictionary_file) != NULL)
{
key_found_flag = findKey(dictionary_word);
if (key_found_flag == 0 )
{
printf("\nKey found sucessfully and is: %s\n", dictionary_word);
break;
}
}

if(key_found_flag !=0)
printf("Sorry key was not found. Please try with a different dictionary.\n\n");
fclose(dictionary_file);
return 0;
}

int findKey(char *potential_key)
{
unsigned char cipher_text_buffer[]=
            {0x8d,0x20,0xe5,0x05,0x6a,0x8d,0x24,0xd0,0x46,0x2c,0xe7,0x4e,0x49,0x04,0xc1,0xb5
             ,0x13,0xe1,0x0d,0x1d,0xf4,0xa2,0xef,0x2a,0xd4,0x54,0x0f,0xae,0x1c,0xa0,0xaa,0xf9};

char plain_text_buffer[] = "This is a top secret.";
unsigned char final_key[16];
unsigned char iv[] = {0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00};
unsigned char outbuf[1024];
int outlen, tmplen,i;
strncpy((char *)final_key, potential_key, 16);

//padding is required if key size is not 16
int k = strlen(final_key);
for(i=15;i>=k-1;i--)
final_key[i]=' ';
printf("The key currently being tested is: %s\n",final_key);
EVP_CIPHER_CTX ctx;
FILE *out;
EVP_CIPHER_CTX_init(&ctx);
EVP_EncryptInit_ex(&ctx, EVP_aes_128_cbc(), NULL, final_key, iv);
if(!EVP_EncryptUpdate(&ctx, outbuf, &outlen, plain_text_buffer, strlen(plain_text_buffer)))
{
/* Error */
```

```
 return 0;
}

 /* Buffer passed to EVP_EncryptFinal() must be after data just encrypted to avoid overwriting it. */
if(!EVP_EncryptFinal_ex(&ctx, outbuf + outlen, &tmplen))
{/* Error */
return 0;
}

outlen += tmplen;
EVP_CIPHER_CTX_cleanup(&ctx);
out = fopen("output_ctx.txt", "wt");
fwrite(outbuf, 1, outlen, out);
fclose(out);
return memcmp(outbuf,cipher_text_buffer,32);
}
```

The contents of the make file look like:

```
INC=/usr/local/ssl/include/
LIB=/usr/local/ssl/lib/
all:
	gcc -I$(INC) -L$(LIB) -o enc crypto_lib_program.c -lcrypto –ldl
```

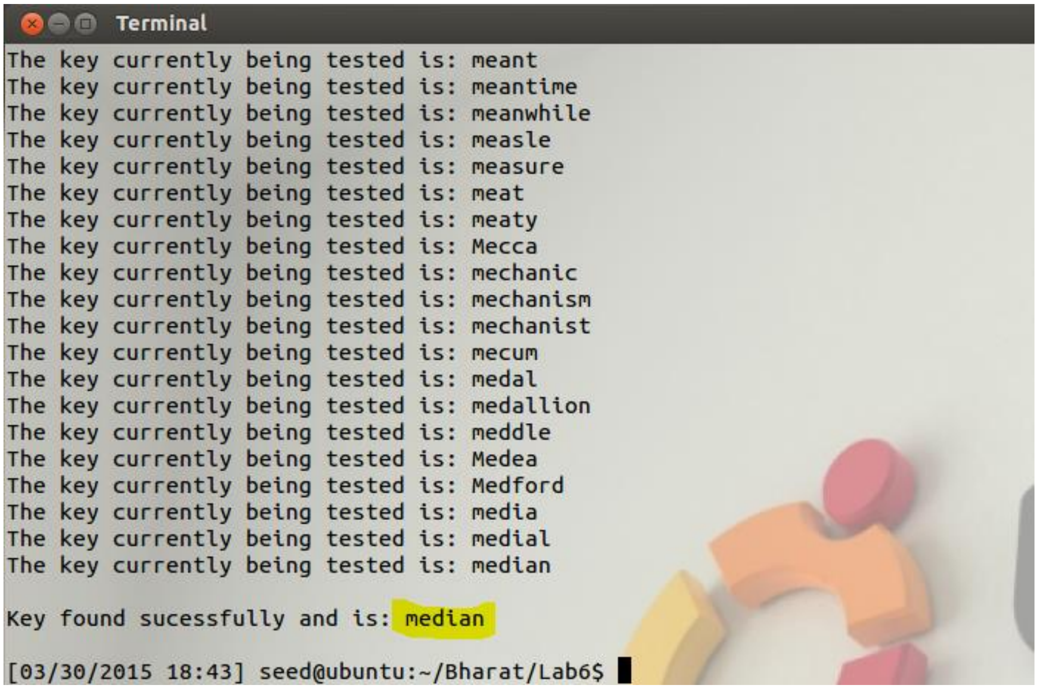
Now let us compile the program:



```
[03/30/2015 18:42] seed@ubuntu:~/Bharat/Lab6$ make
gcc -I/usr/local/ssl/include/ -L/usr/local/ssl/lib/ -o enc crypto_lib_program.
c -lcrypto -ldl
[03/30/2015 18:42] seed@ubuntu:~/Bharat/Lab6$ enc
```

compilation was successful

After we run the program (enc):



```
The key currently being tested is: meant
The key currently being tested is: meantime
The key currently being tested is: meanwhile
The key currently being tested is: measle
The key currently being tested is: measure
The key currently being tested is: meat
The key currently being tested is: meaty
The key currently being tested is: Mecca
The key currently being tested is: mechanic
The key currently being tested is: mechanism
The key currently being tested is: mechanist
The key currently being tested is: mecum
The key currently being tested is: medal
The key currently being tested is: medallion
The key currently being tested is: meddle
The key currently being tested is: Medea
The key currently being tested is: Medford
The key currently being tested is: media
The key currently being tested is: medial
The key currently being tested is: median

Key found sucessfully and is: median

[03/30/2015 18:43] seed@ubuntu:~/Bharat/Lab6$ 
```

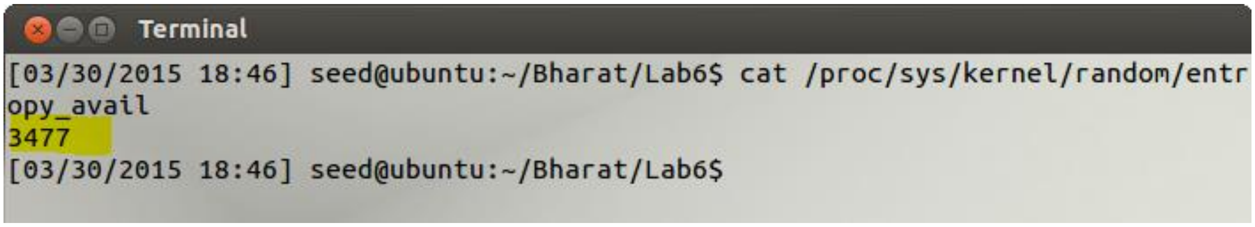
key found as ‘median’

After we run the program with plain text and cipher specified as given in the program. We get the key as ‘median’.

3.6 Task 6: Pseudo Random Number Generation

Task 6.A: Measure the Entropy of Kernel

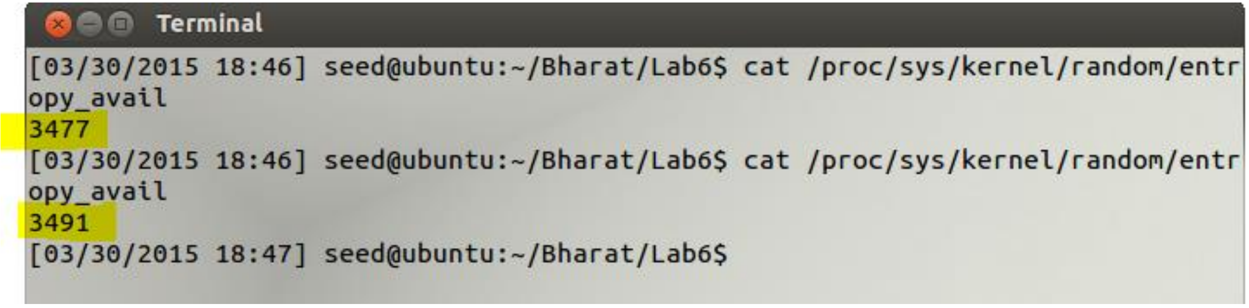
To get the randomness we execute the following command:
'cat /proc/sys/kernel/random/entropy_avail'



```
[03/30/2015 18:46] seed@ubuntu:~/Bharat/Lab6$ cat /proc/sys/kernel/random/entr
opy_avail
3477
[03/30/2015 18:46] seed@ubuntu:~/Bharat/Lab6$
```

current entropy value: 3447

Now we shall move our mouse and after a few press we can see the entropy as:



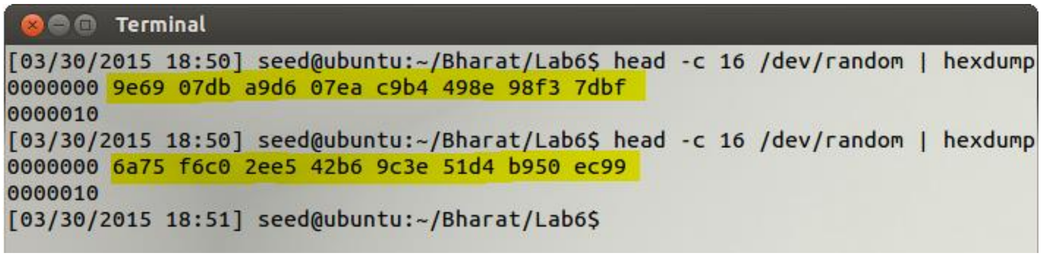
```
[03/30/2015 18:46] seed@ubuntu:~/Bharat/Lab6$ cat /proc/sys/kernel/random/entr
opy_avail
3477
[03/30/2015 18:46] seed@ubuntu:~/Bharat/Lab6$ cat /proc/sys/kernel/random/entr
opy_avail
3491
[03/30/2015 18:47] seed@ubuntu:~/Bharat/Lab6$
```

entropy value increased.

Randomness was accounted for when we moved our mouse and had a few keys pressed. And so there was an increase in the entropy value.

Task 6.B: Get Pseudo Random Numbers from /dev/random

We can use the following command to get 16 bytes of pseudo random numbers from /dev/random and then pipe the data to hexdump to print them out.

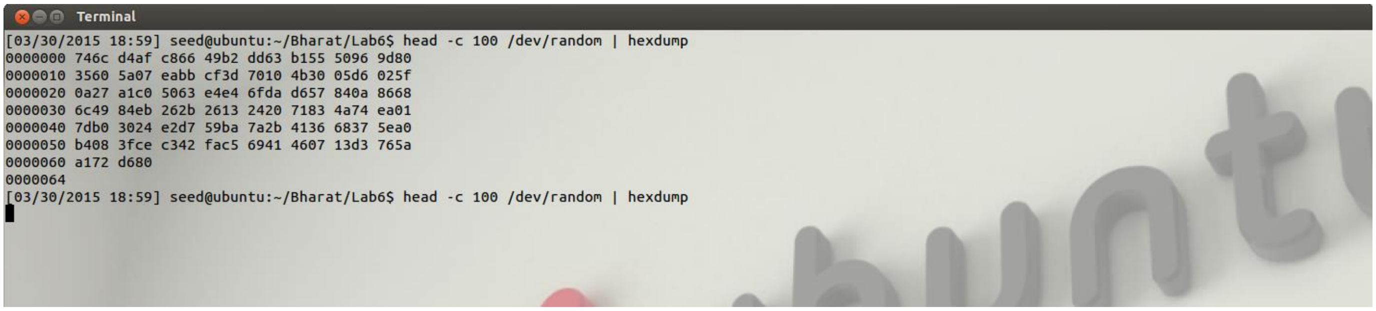


```
[03/30/2015 18:50] seed@ubuntu:~/Bharat/Lab6$ head -c 16 /dev/random | hexdump
0000000 9e69 07db a9d6 07ea c9b4 498e 98f3 7dbf
0000010
[03/30/2015 18:50] seed@ubuntu:~/Bharat/Lab6$ head -c 16 /dev/random | hexdump
0000000 6a75 f6c0 2ee5 42b6 9c3e 51d4 b950 ec99
0000010
[03/30/2015 18:51] seed@ubuntu:~/Bharat/Lab6$
```

random 16 bytes generated

AS we can see when we make the call twice, 2 entirely different random numbers were generated.

16 bytes is small so lets run the above program with 100 bytes for several times and see what happens:



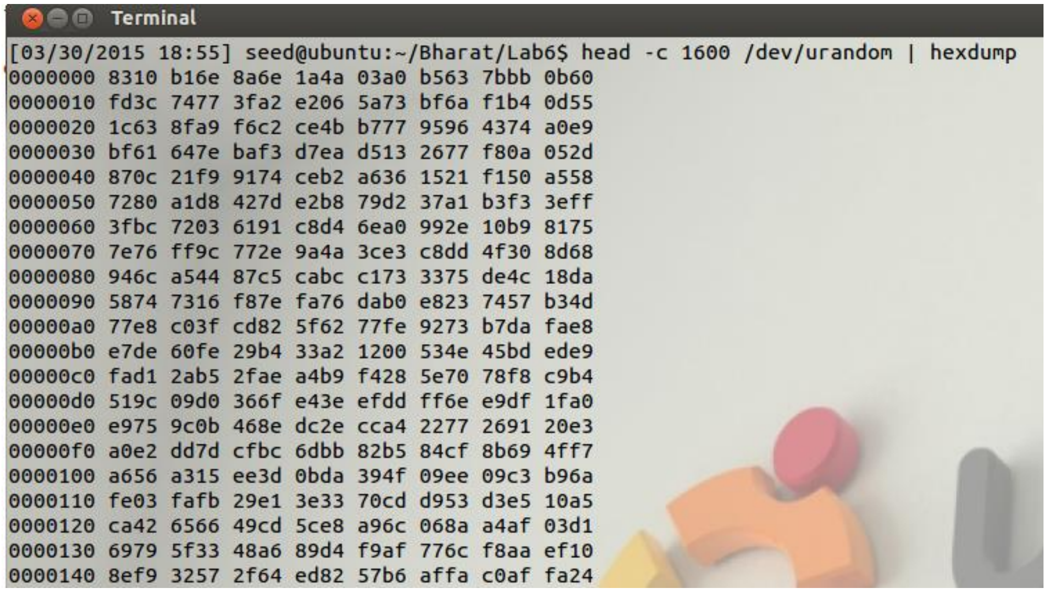
```
[03/30/2015 18:59] seed@ubuntu:~/Bharat/Lab6$ head -c 100 /dev/random | hexdump
0000000 746c d4af c866 49b2 dd63 b155 5096 9d80
0000010 3560 5a07 eabb cf3d 7010 4b30 05d6 025f
0000020 0a27 a1c0 5063 e4e4 6fda d657 840a 8668
0000030 6c49 84eb 262b 2613 2420 7183 4a74 ea01
0000040 7db0 3024 e2d7 59ba 7a2b 4136 6837 5ea0
0000050 b408 3fce c342 fac5 6941 4607 13d3 765a
0000060 a172 d680
0000064
[03/30/2015 18:59] seed@ubuntu:~/Bharat/Lab6$ head -c 100 /dev/random | hexdump
```

/dev/random blocked.

However when we keep moving our mouse pointer or press a few keys the entropy is generated and the /dev/random is unblocked.

Task 6.C: Get Random Numbers from /dev/urandom

We can use the following command to get 1600 bytes of pseudo random numbers:



```
[03/30/2015 18:55] seed@ubuntu:~/Bharat/Lab6$ head -c 1600 /dev/urandom | hexdump
0000000 8310 b16e 8a6e 1a4a 03a0 b563 7bbb 0b60
0000010 fd3c 7477 3fa2 e206 5a73 bf6a f1b4 0d55
0000020 1c63 8fa9 f6c2 ce4b b777 9596 4374 a0e9
0000030 bf61 647e baf3 d7ea d513 2677 f80a 052d
0000040 870c 21f9 9174 ceb2 a636 1521 f150 a558
0000050 7280 a1d8 427d e2b8 79d2 37a1 b3f3 3eff
0000060 3fbc 7203 6191 c8d4 6ea0 992e 10b9 8175
0000070 7e76 ff9c 772e 9a4a 3ce3 c8dd 4f30 8d68
0000080 946c a544 87c5 cabc c173 3375 de4c 18da
0000090 5874 7316 f87e fa76 dab0 e823 7457 b34d
00000a0 77e8 c03f cd82 5f62 77fe 9273 b7da fae8
00000b0 e7de 60fe 29b4 33a2 1200 534e 45bd ede9
00000c0 fad1 2ab5 2fae a4b9 f428 5e70 78f8 c9b4
00000d0 519c 09d0 366f e43e efdd ff6e e9df 1fa0
00000e0 e975 9c0b 468e dc2e cca4 2277 2691 20e3
00000f0 a0e2 dd7d cfbc 6dbb 82b5 84cf 8b69 4ff7
0000100 a656 a315 ee3d 0bda 394f 09ee 09c3 b96a
0000110 fe03 fafb 29e1 3e33 70cd d953 d3e5 10a5
0000120 ca42 6566 49cd 5ce8 a96c 068a a4af 03d1
0000130 6979 5f33 48a6 89d4 f9af 776c f8aa ef10
0000140 8ef9 3257 2f64 ed82 57b6 affa c0af fa24
```

1600 random numbers generated.

As we keep running the same command again and again, however the number of times you repeat the /dec/urandom is never blocked.