CPE 449: ENCRYPTION ASSIGNMENT

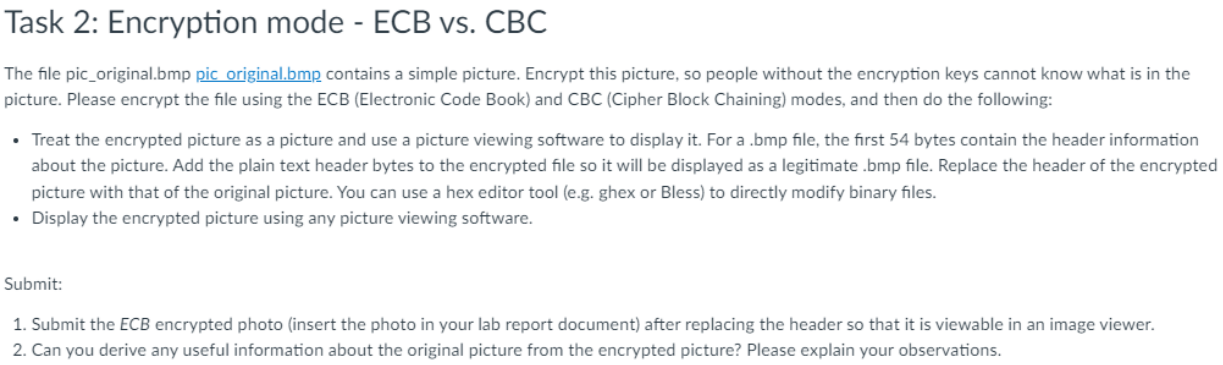
DAN OTIENO

09/01/2023.

Task 1:

No submission required.

Task 2:



Task 2: Encryption mode - ECB vs. CBC

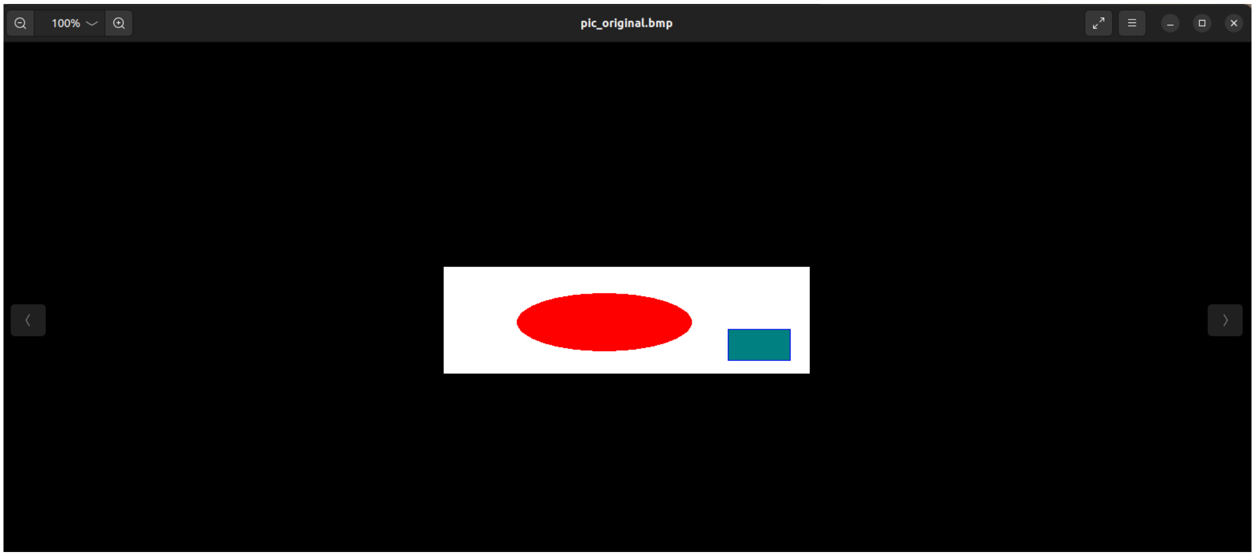
The file pic_original.bmp pic_original.bmp contains a simple picture. Encrypt this picture, so people without the encryption keys cannot know what is in the picture. Please encrypt the file using the ECB (Electronic Code Book) and CBC (Cipher Block Chaining) modes, and then do the following:

- Treat the encrypted picture as a picture and use a picture viewing software to display it. For a .bmp file, the first 54 bytes contain the header information about the picture. Add the plain text header bytes to the encrypted file so it will be displayed as a legitimate .bmp file. Replace the header of the encrypted picture with that of the original picture. You can use a hex editor tool (e.g. ghex or Bless) to directly modify binary files.
- Display the encrypted picture using any picture viewing software.

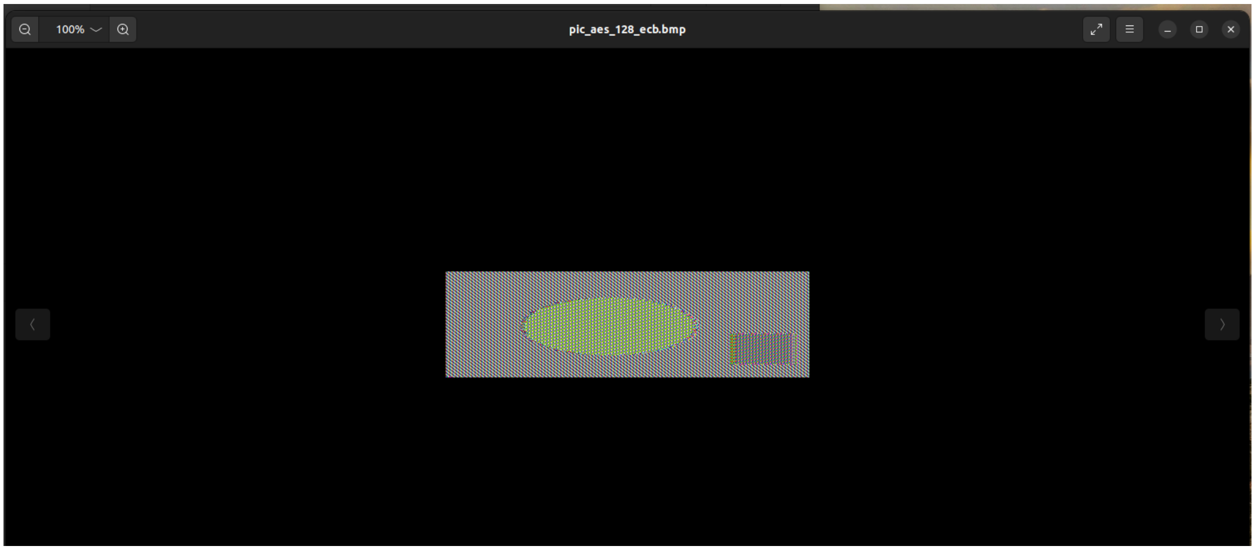
Submit:

1. Submit the *ECB* encrypted photo (insert the photo in your lab report document) after replacing the header so that it is viewable in an image viewer.
2. Can you derive any useful information about the original picture from the encrypted picture? Please explain your observations.

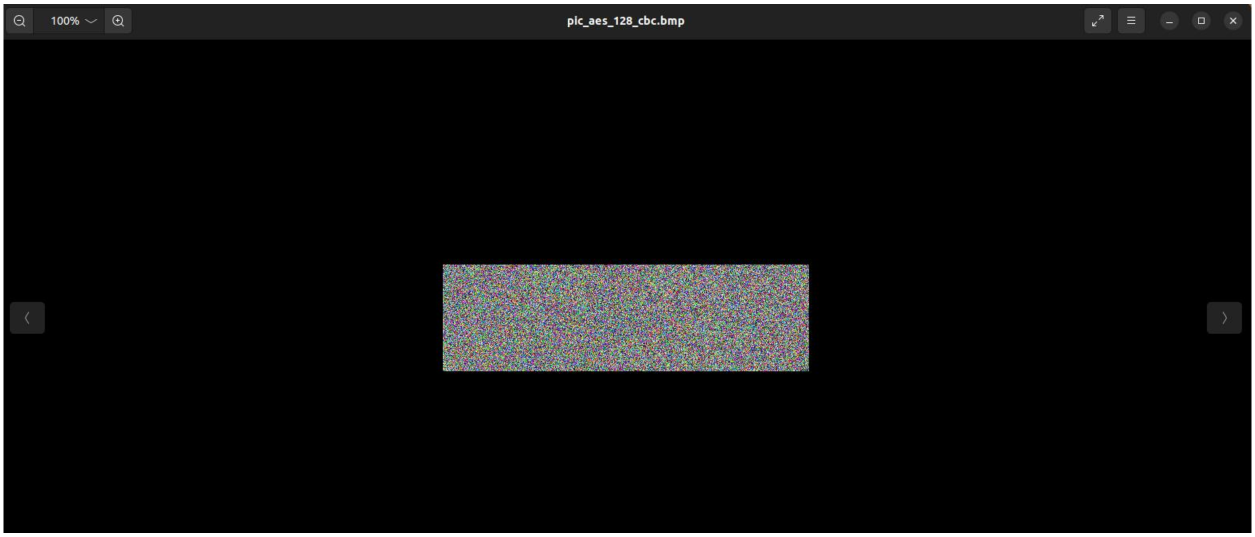
1. **Original photo:**



Encrypted photo (aes-128-ecb):

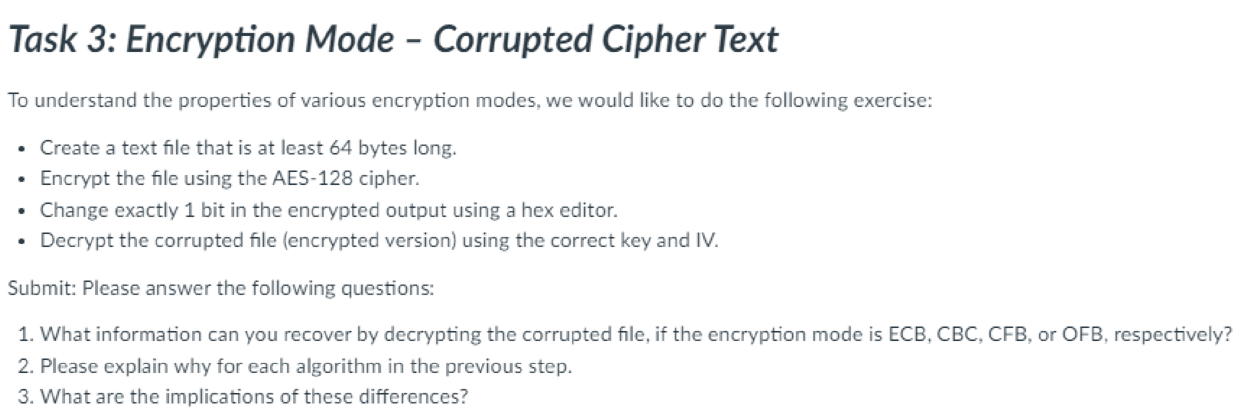


Encrypted photo (aes-128-cbc):



2. There’s not much information that can be derived when it comes to the colors from the original image. However, whether anything can be determined from the original image depends on the encryption method. When the ECB method is used, we can see the exact shapes from the original image, whereas, in the second encryption method, CBC, the original image is fully encrypted such that there is no visible data from the original. Because the CBC method carries over encryption information from the one block and uses it to encrypt the next block, whereas ECB encrypts each block independently, CBC is considered a more secure method between the two, and that is demonstrated in the exercise above.

Task 3:



Task 3: Encryption Mode – Corrupted Cipher Text

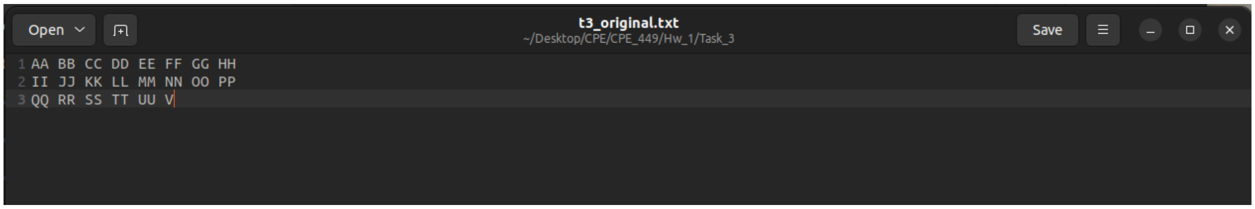
To understand the properties of various encryption modes, we would like to do the following exercise:

- Create a text file that is at least 64 bytes long.
- Encrypt the file using the AES-128 cipher.
- Change exactly 1 bit in the encrypted output using a hex editor.
- Decrypt the corrupted file (encrypted version) using the correct key and IV.

Submit: Please answer the following questions:

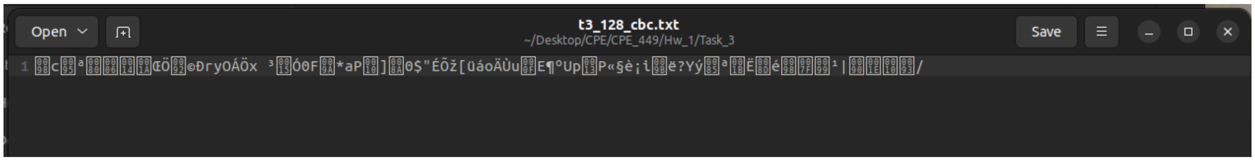
1. What information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively?
2. Please explain why for each algorithm in the previous step.
3. What are the implications of these differences?

Original text:

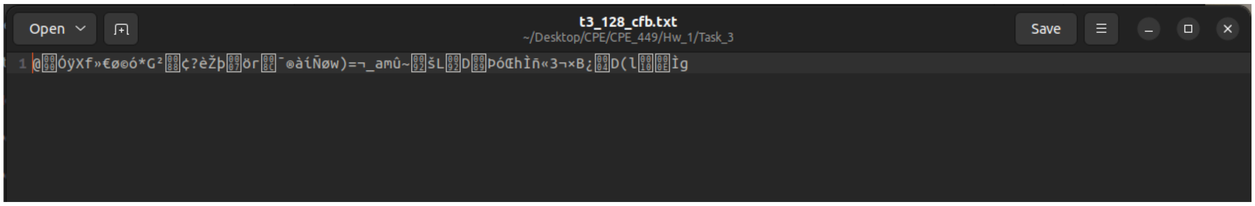


```
AA BB CC DD EE FF GG HH
II JJ KK LL MM NN OO PP
QQ RR SS TT UU V
```

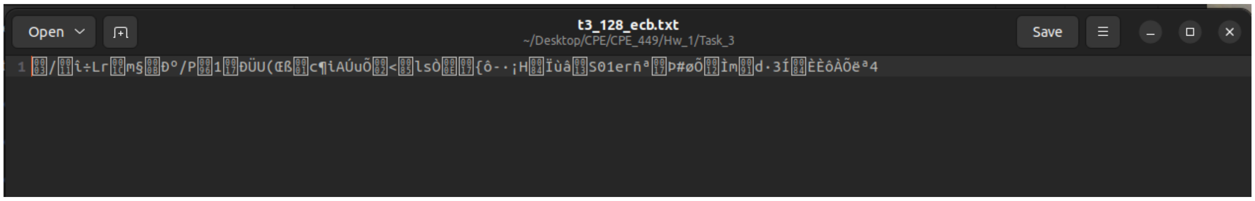
Aes-128 Encryptions (CBC, CFB, ECB, OFB respectively):



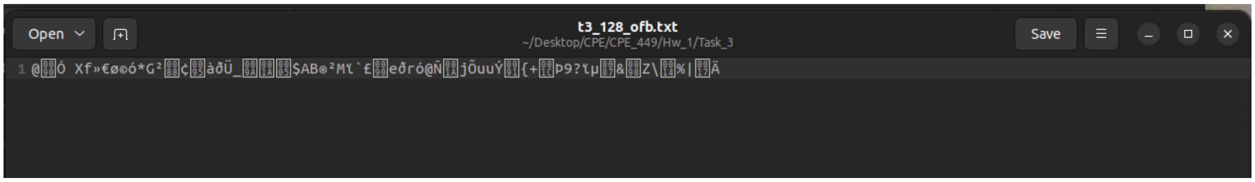
```
□c□ª□□□□ŒÖ□©ÐryOÁÖx ³□Ó0F□*aP□]□0$"ÉÕž[üáoÄÙu□E¶ºUp□P«§è¡ĭ□ë?Yý□ª□Ë□é□□□¹|□□□□/
```



```
@□ÓÿXf»€ø©ó*G²□¢?èŽþ□ör□¯®àíÑøw)=¬_amû~□šL□D□ÞóŒhÌñ«3¬×B¿□D(l□□Ìg
```

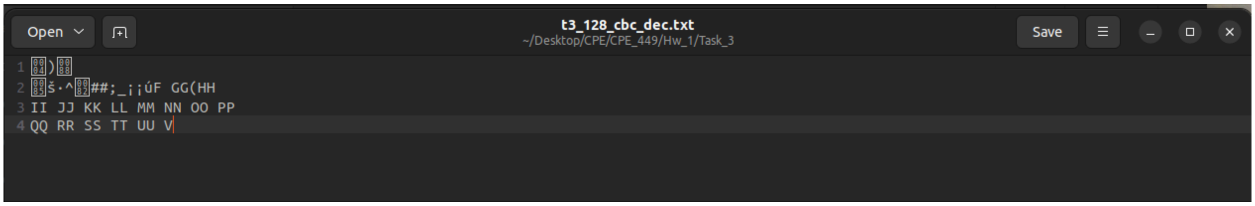


```
□/□ĩ÷Lr□m§□Ð°/P□1□ÐÜU(Œß□c¶iAÚuÕ□<□lsÒ□□{ô-·¡H□Ïùâ□S01erñª□Þ#øÕ□Ìm□d·3Í□ÈÈôÀÕëª4
```

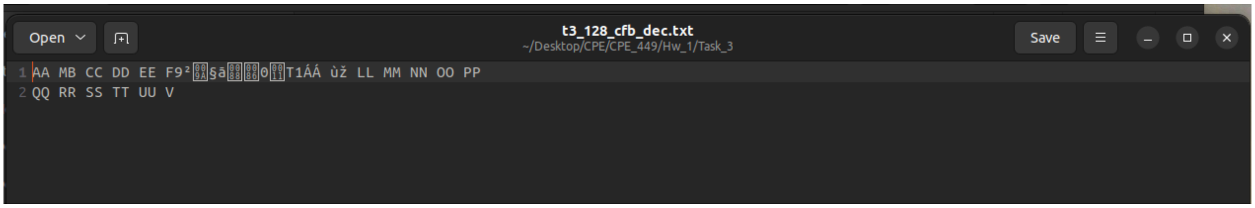


```
@□Ó Xf»€ø©ó*G²□¢□àðÜ_□□□$AB®²Mĭ`£□eðró@Ñ□jÕuuÝ□{+□Þ9?ĭµ□&□Z\□%|□Ä
```

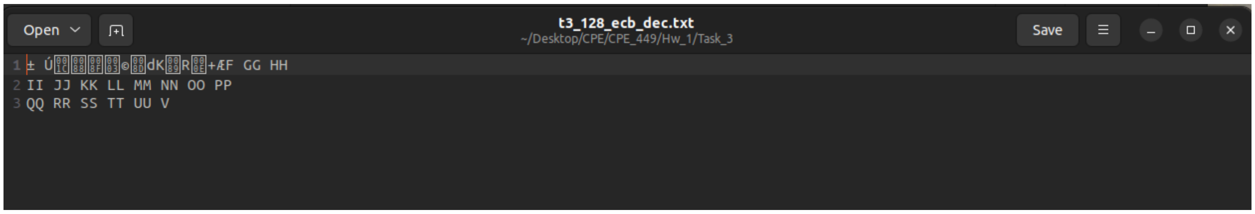
Aes-128 Decryptions after modifying 1 bit per file (CBC, CFB, ECB, OFB respectively):



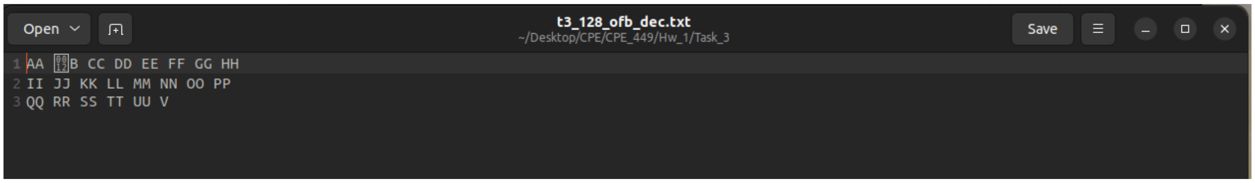
```
)
š·^##;_¡¡úF GG(HH
II JJ KK LL MM NN OO PP
QQ RR SS TT UU V
```



```
AA MB CC DD EE F9²§āθT1ÁÁ ùž LL MM NN OO PP
QQ RR SS TT UU V
```



```
± Ú©dKR+ÆF GG HH
II JJ KK LL MM NN OO PP
QQ RR SS TT UU V
```

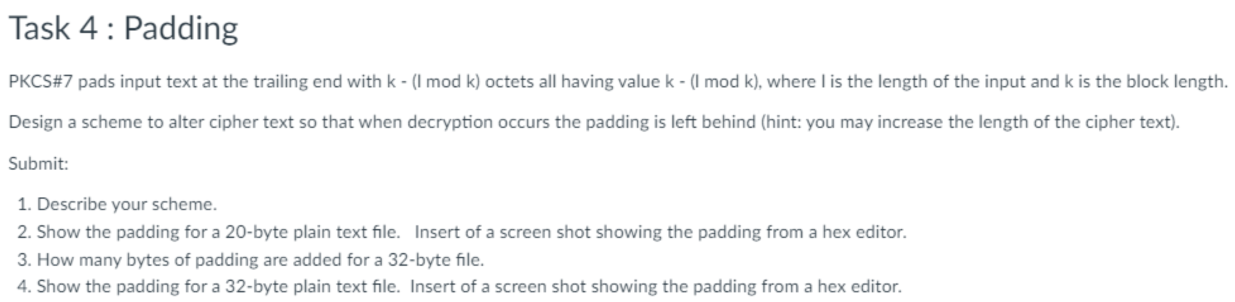


```
AA B CC DD EE FF GG HH
II JJ KK LL MM NN OO PP
QQ RR SS TT UU V
```

1. **All the cypher files are modified in some way when decrypted. We get varying amounts of data recovered from the corrupted cypher files. 48 bytes of data is recovered from the CBC cypher, 46 bytes recovered from CFB cypher, 47 bytes recovered from the ECB cypher, and finally, 63 bytes recovered from the OFB cypher.**
2. **The most data recovered from the corrupted cypher is from the OFB-encrypted file. This is because the algorithm uses a stream of bits to encrypt subsequent data blocks, the decryption is completed one bit at a time, such that only the text corresponding to the corrupted bit is affected. The CBC and CFB modes are similar in that during decrypting of a ciphertext block, one should add XOR the output data received from the decryption algorithm to the previous ciphertext block. Because the receiver knows all the ciphertext blocks just after obtaining the encrypted message, he can decrypt the message using many threads simultaneously. So, if one bit is corrupted, then all the corresponding bits are affected when the file is decrypted. For the ECB method, each block of text/cyphertext is encrypted or decrypted separately, so all corresponding bits are affected if one is corrupted as well. (Source: https://www.crypto-it.net/eng/theory/modes-of-block-ciphers.html)**
3. **Because of the differences in how much of the original data can be retrieved from a corrupted cypher text file, the implications depend on the circumstances, in the case where bad actors are involved in decrypting the file, the method of encryption/decryption is important because less data can be recovered to prevent information ending up in the**

wrong hands. However, in the case where extremely important encrypted information ends up with the rightful receiver, the loss of data can lead to misinterpretation or misrepresentation of the original message, leading to detrimental outcomes.

Task 4:



Task 4 : Padding

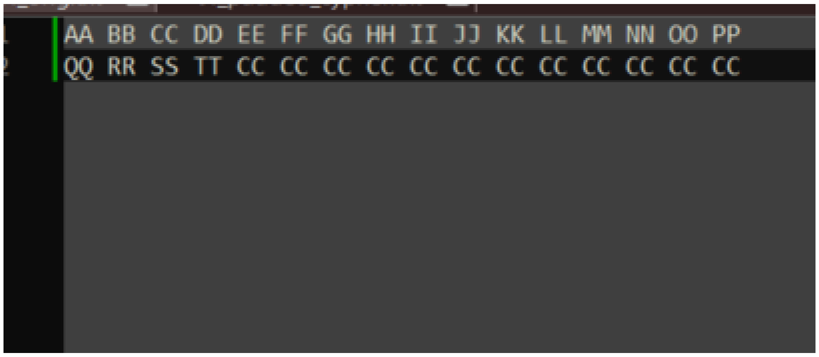
PKCS#7 pads input text at the trailing end with k - (l mod k) octets all having value k - (l mod k), where l is the length of the input and k is the block length.

Design a scheme to alter cipher text so that when decryption occurs the padding is left behind (hint: you may increase the length of the cipher text).

Submit:

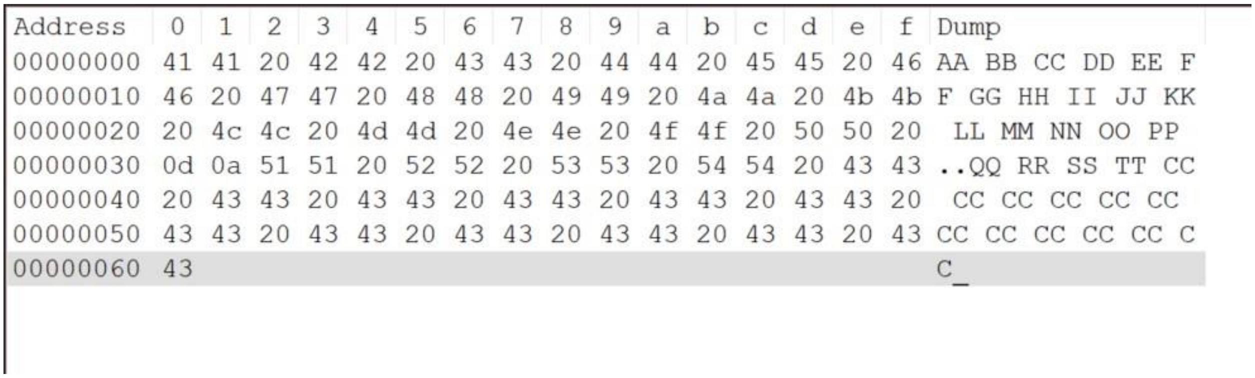
1. Describe your scheme.
2. Show the padding for a 20-byte plain text file.   Insert of a screen shot showing the padding from a hex editor.
3. How many bytes of padding are added for a 32-byte file.
4. Show the padding for a 32-byte plain text file. Insert of a screen shot showing the padding from a hex editor.

Original text (with padding):



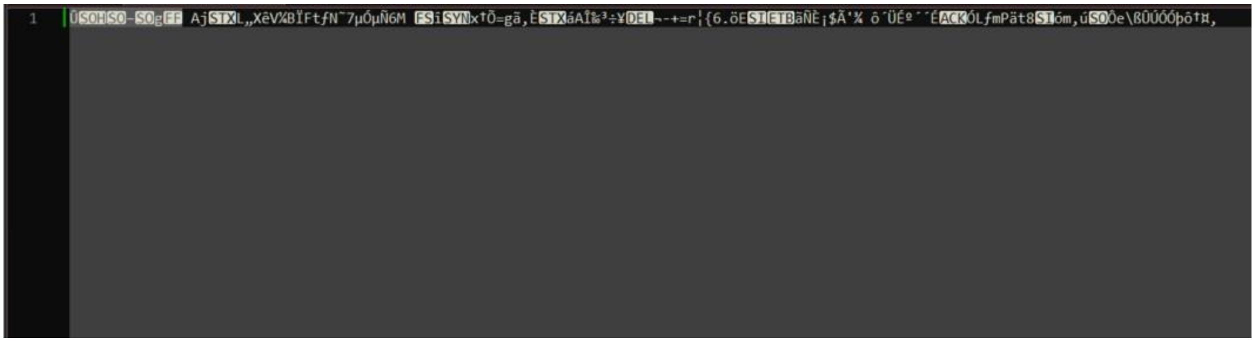
```
AA BB CC DD EE FF GG HH II JJ KK LL MM NN OO PP
QQ RR SS TT CC CC CC CC CC CC CC CC CC CC CC CC
```

Hex view of original text:



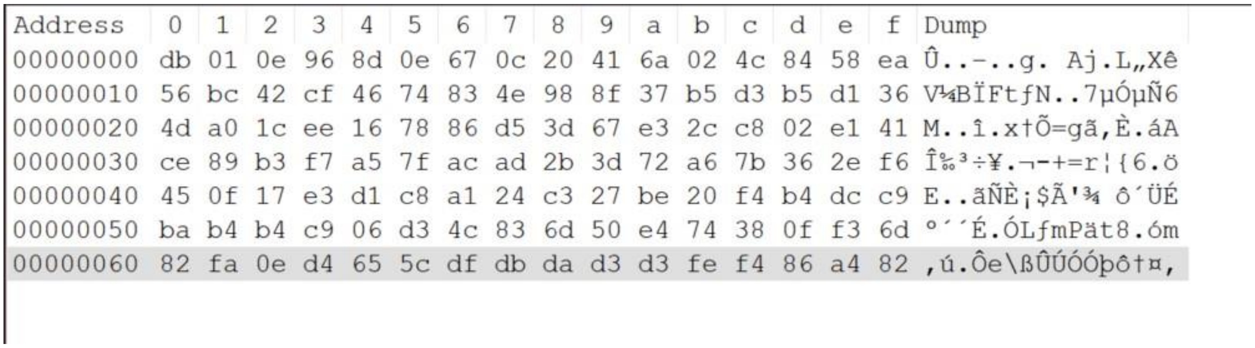
```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 41 41 20 42 42 20 43 43 20 44 44 20 45 45 20 46 AA BB CC DD EE F
00000010 46 20 47 47 20 48 48 20 49 49 20 4a 4a 20 4b 4b F GG HH II JJ KK
00000020 20 4c 4c 20 4d 4d 20 4e 4e 20 4f 4f 20 50 50 20  LL MM NN OO PP
00000030 0d 0a 51 51 20 52 52 20 53 53 20 54 54 20 43 43 ..QQ RR SS TT CC
00000040 20 43 43 20 43 43 20 43 43 20 43 43 20 43 43 20  CC CC CC CC CC
00000050 43 43 20 43 43 20 43 43 20 43 43 20 43 43 20 43 CC CC CC CC CC C
00000060 43                                              C
```

Padded cypher:



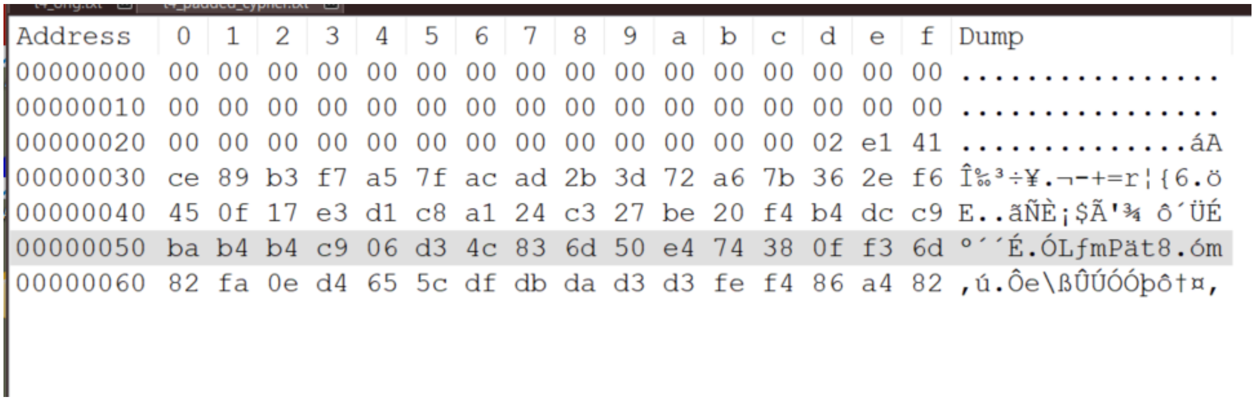
```
ÛSOHSO-SOgFF AjSTXL„XêV¼BÏFtƒN˜7µÓµÑ6M FSiSYNx†Õ=gã,ÈSTXáAÎ‰³÷¥DEL¬-+=r¦{6.öESIETBãÑÈ¡$Ã'¾ ô´ÜÉº´´ÉACKÓLƒmPät8SIóm,úSOÔe\ßÛÚÓÓþô†¤,
```

Hex view of padded cypher:



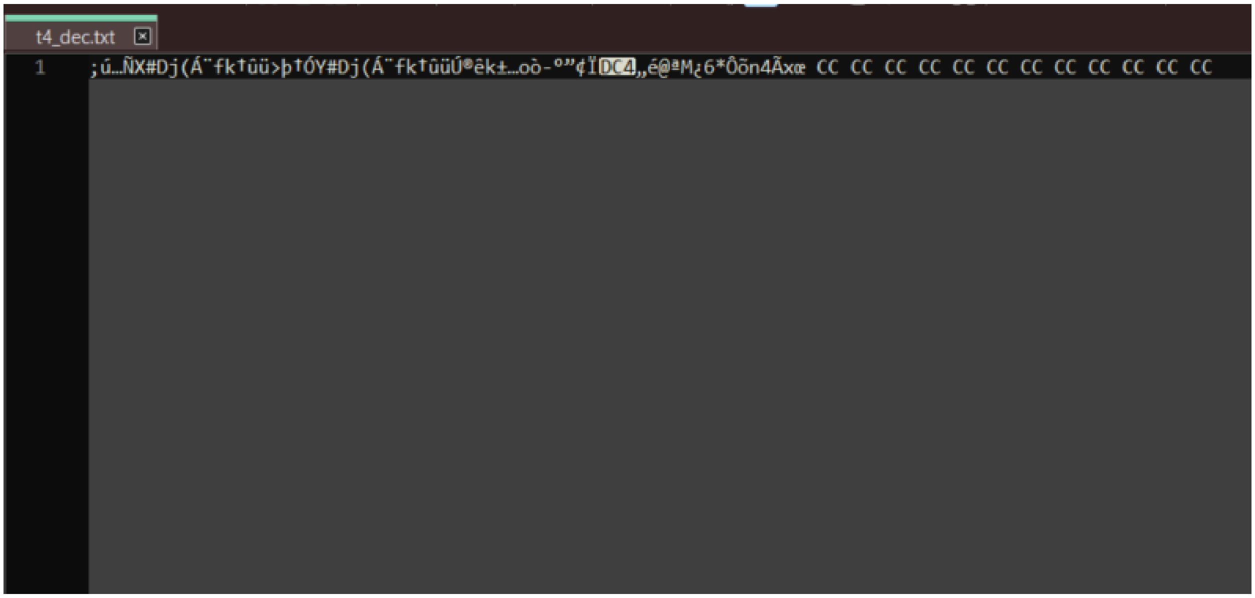
```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 db 01 0e 96 8d 0e 67 0c 20 41 6a 02 4c 84 58 ea Û..–..g. Aj.L„Xê
00000010 56 bc 42 cf 46 74 83 4e 98 8f 37 b5 d3 b5 d1 36 V¼BÏFtƒN..7µÓµÑ6
00000020 4d a0 1c ee 16 78 86 d5 3d 67 e3 2c c8 02 e1 41 M..î.x†Õ=gã,È.áA
00000030 ce 89 b3 f7 a5 7f ac ad 2b 3d 72 a6 7b 36 2e f6 Î‰³÷¥.¬-+=r¦{6.ö
00000040 45 0f 17 e3 d1 c8 a1 24 c3 27 be 20 f4 b4 dc c9 E..ãÑÈ¡$Ã'¾ ô´ÜÉ
00000050 ba b4 b4 c9 06 d3 4c 83 6d 50 e4 74 38 0f f3 6d º´´É.ÓLƒmPät8.óm
00000060 82 fa 0e d4 65 5c df db da d3 d3 fe f4 86 a4 82 ‚ú.Ôe\ßÛÚÓÓþô†¤‚
```

Hex view of modified bytes in padded cypher:



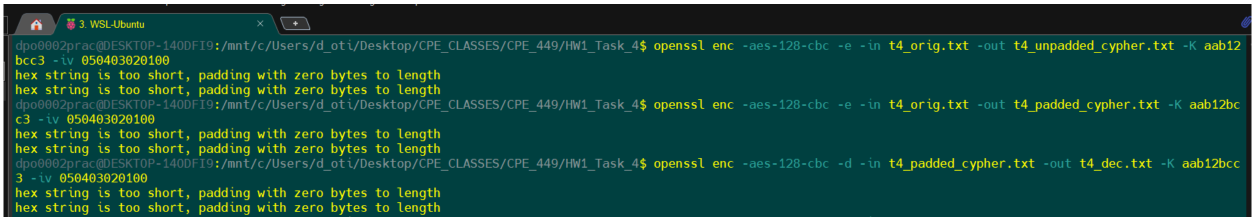
```
Address   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f  Dump
00000000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 02 e1 41 ..............áA
00000030 ce 89 b3 f7 a5 7f ac ad 2b 3d 72 a6 7b 36 2e f6 Î‰³÷¥.¬-+=r¦{6.ö
00000040 45 0f 17 e3 d1 c8 a1 24 c3 27 be 20 f4 b4 dc c9 E..ãÑÈ¡$Ã'¾ ô´ÜÉ
00000050 ba b4 b4 c9 06 d3 4c 83 6d 50 e4 74 38 0f f3 6d º´´É.ÓLƒmPät8.óm
00000060 82 fa 0e d4 65 5c df db da d3 d3 fe f4 86 a4 82 ‚ú.Ôe\ßÛÚÓÓþô†¤‚
```

Decrypted cypher file after altering bytes such that only the padding is viewable:



```
1   ;ú…ÑX#Dj(Á¨fk†ûü>þ†ÓY#Dj(Á¨fk†ûüÚ®êk±…oò-º”¢ÏDC4„é@ªM¿6*Ôõn4Ãxœ CC CC CC CC CC CC CC CC CC CC CC CC
```

Command line arguments for Task 4.



```
dpo0002prac@DESKTOP-140DFI9:/mnt/c/Users/d_oti/Desktop/CPE_CLASSES/CPE_449/HW1_Task_4$ openssl enc -aes-128-cbc -e -in t4_orig.txt -out t4_unpadded_cypher.txt -K aab12
bcc3 -iv 050403020100
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
dpo0002prac@DESKTOP-140DFI9:/mnt/c/Users/d_oti/Desktop/CPE_CLASSES/CPE_449/HW1_Task_4$ openssl enc -aes-128-cbc -e -in t4_orig.txt -out t4_padded_cypher.txt -K aab12bc
c3 -iv 050403020100
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
dpo0002prac@DESKTOP-140DFI9:/mnt/c/Users/d_oti/Desktop/CPE_CLASSES/CPE_449/HW1_Task_4$ openssl enc -aes-128-cbc -d -in t4_padded_cypher.txt -out t4_dec.txt -K aab12bcc
3 -iv 050403020100
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
```

1. **The Padding scheme used is PKCS#7.**
2. **See screenshot above.**
3. **12 Bytes are added for a 32-byte file.**
4. **See screenshot above.**