WSU ID: Q486k227
First name: ROHITH BANGARI
Last name: BENAKAHALLI SIDDESHAPPA

CS 766-Symmetric Key Encryption Lab

4.1 Task 1: Frequency Analysis Against Monoalphabetic Substitution Cipher.

1. Provide the substitution key that you found. Partial credit will be given for partial substitution keys. The substitution should be a string of uppercase letters. The first letter in the key indicates the character that represents 'a' in the ciphertext, the second one corresponds to 'b', etc.

Ans:

```
[03/12/21]seed@SEED:~$ tr 'abcdefghijklmnopqrstuvwxyz' 'kmgjszbnvwplhcteoxfaquriyd' <ciphertext.txt> plaintext1.txt
[03/12/21]seed@SEED:~$
```

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Substitution keys	K	M	G	J	S	Z	B	N	V	W	P	L	H	C	T	E	O	X	F	A	Q	U	R	I	Y	D

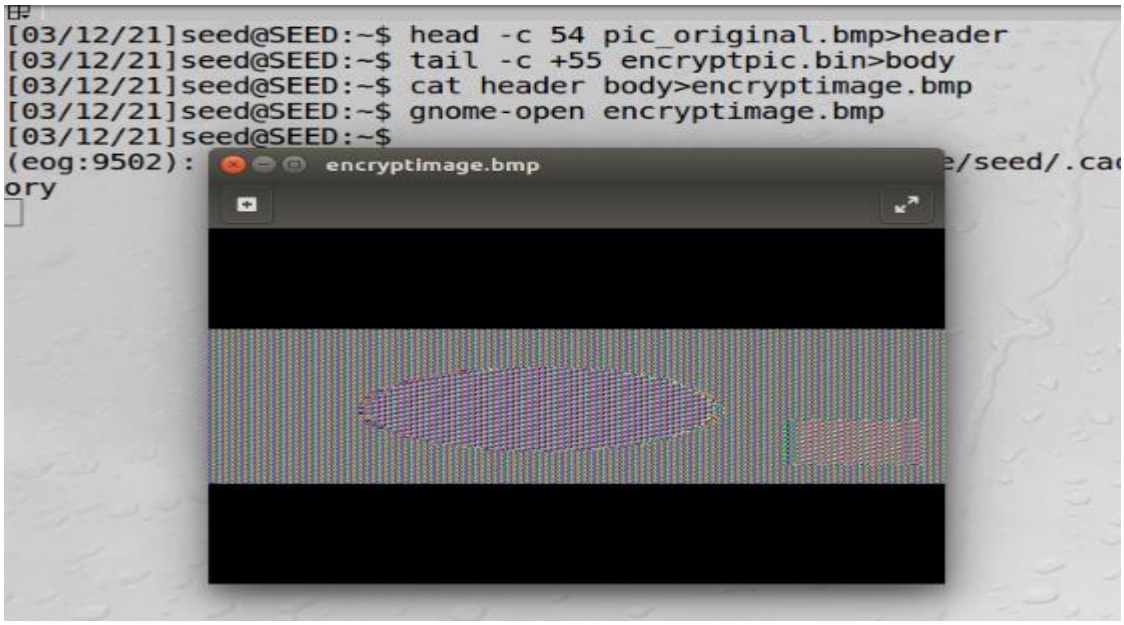
4.3 Task 1.1: Encryption Mode – ECB vs. CBC)

1. Using EBC mode perform the following tasks:

(a) Encrypt the picture using EBC mode. Attach a screen shot of the terminal showing your encryption command.

```
[03/12/21]seed@SEED:~$ openssl enc -aes-128-ecb -e -in pic_original.bmp -out encryptpic.bin -k 00112233445566778889aabbccddeeff
[03/12/21]seed@SEED:~$
```

(b) Display the encrypted picture using any picture viewing software. Attack a screen shot of the displayed encrypted image.



(c) Answer the following questions:

* Can you derive any useful information about the original picture from the encrypted picture?

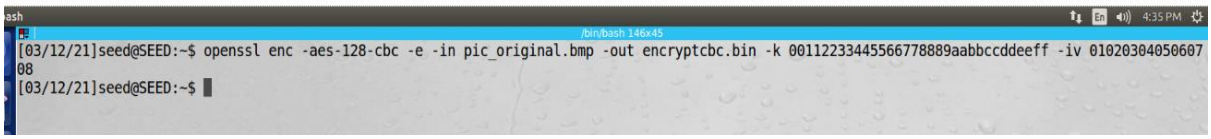
Yes, we can drive information about original picture from encrypted picture.

* What characteristic of the EBC mode causes the displayed picture to display (or protect) useful information from the original picture? (Write 1-2 sentences on your report).

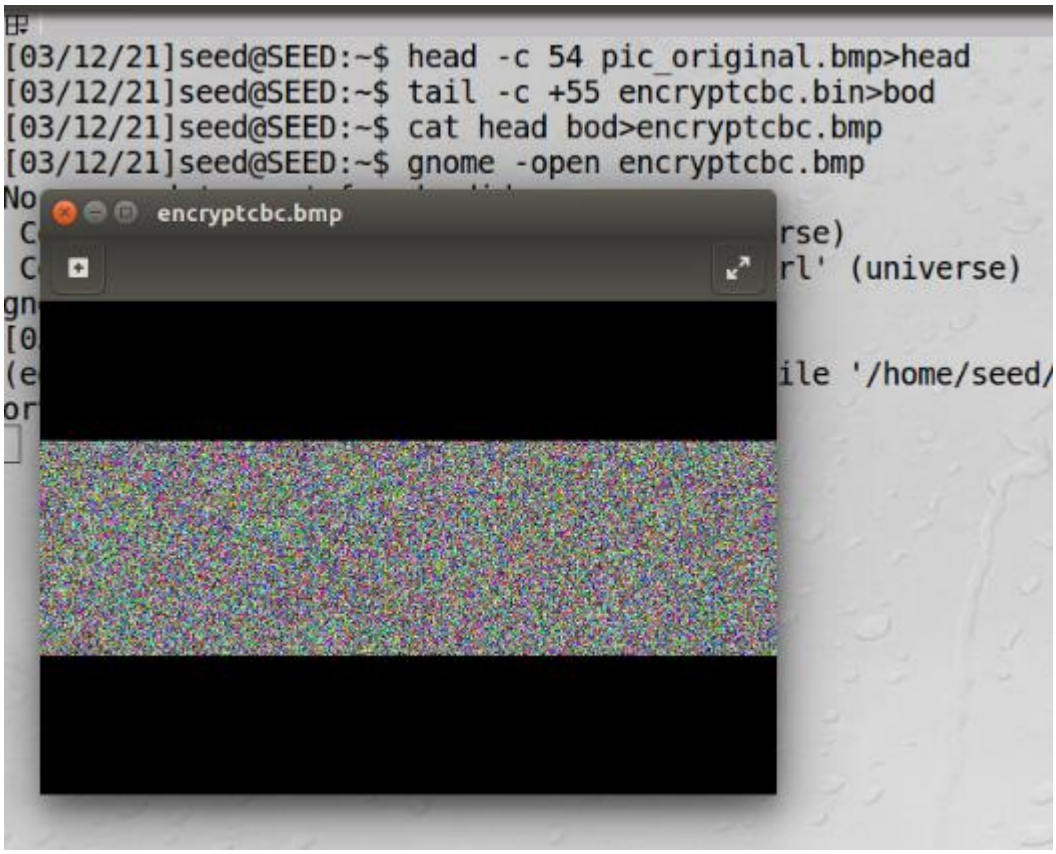
ECB mode generates same cipher text for repeating plain text. Even though the cipher image is different from the original image, most of the original image's information can be obtained from the cipher image.

2. Using CBC mode perform the following tasks:

(a) Encrypt the picture using CBC mode. Attach a screen shot of the terminal showing your encryption command.



(b) Display the encrypted picture using any picture viewing software. Attack a screen shot of the displayed encrypted image.



(c) Answer the following questions:

* Can you derive any useful information about the original picture from the encrypted picture?

No, we cannot extract any information in the encrypted picture.

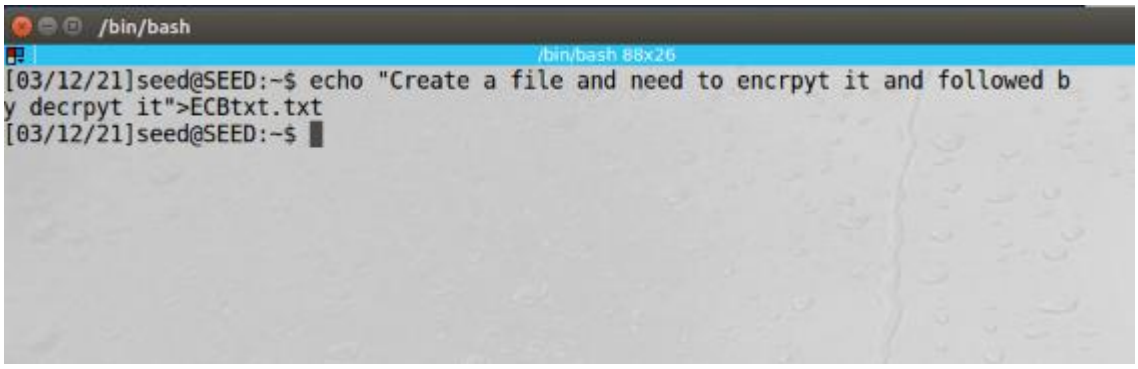
*What characteristic of the CBC mode causes the displayed picture to display (or protect) useful information from the original picture? (Write 1-2 sentences on your report)

CBC mode creates different cipher text for repeating plain text.

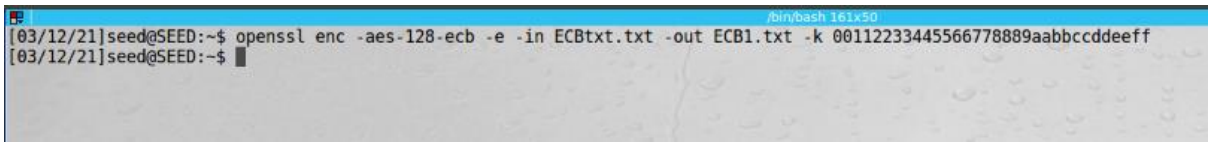
4.4 Task 2: Encryption Mode – Corrupted Cipher Text

1. EBC mode.

(a) Create a file that is 64 bytes long. Attach a screen shot of the file.



(b) Encrypt the file using EBC. Attach a screen shot of the terminal showing your encryption command.



(c) Corrupt the file. Attach a screen shot of the corrupted file (the corrupted file should be displayed with the same software/command used to display the original file).

```
[03/12/21]seed@SEED:~$ openssl enc -aes-128-ecb -e -in ECBtxt.txt -out ECB1.txt -k 00112233445566778889aabbccddeeff
[03/12/21]seed@SEED:~$ cat ECB1.txt
Salted__yM:□□7KD□□□K□2□□□□□□0□T5□U□n□^□L□□A□□1U+□a□□,□□□
                                                        □3Y□G□□□n=□□e□e□□1Ýy□ □□□□jjT□□]□□+□[03/12/21]seed@SEED:~$
[03/12/21]seed@SEED:~$ bless ECB1.txt
Unexpected end of file has occurred. The following elements are not closed: pref, preferences. Line 22, position 36.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Directory '/home/seed/.config/bless/plugins' not found.
Could not find file "/home/seed/.config/bless/export_patterns".
Could not find file "/home/seed/.config/bless/history.xml".
Document does not have a root element.
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Sharing violation on path /home/seed/.config/bless/preferences.xml
Document does not have a root element.
[03/12/21]seed@SEED:~$ cat ECB1.txt
Salted__yM:□□7KD□□□K□2□□□□□□0□55□U□n□^□L□□A□□1U+□a□□,□□□
                                                        □3Y□G□□□n=□□e□e□□1Ýy□ □□□□jjT□□]□□+□[03/12/21]seed@SEED:~$
[03/12/21]seed@SEED:~$ openssl enc -aes-128-ecb -d -in ECB1.txt -out ECB2.txt -k 00112233445566778889aabbccddeeff
[03/12/21]seed@SEED:~$ cat ECB2.txt
□□!□DkD□□□□□□I□od need to encrpyt it and followed by decrpyt it
[03/12/21]seed@SEED:~$
```

(d) How much information can you recover by decrypting the corrupted file?

During decrypting, out of 64 bytes except starting 13 bytes got corrupted and rest are recovered.

(e) Explain why in terms of the decryption procedures of both EBC.

During decryption we can recover everything except single bit of the cipher block is corrupted then only the corresponding plain test block will be corrupted. The rest of the plain text will not get corrupted. However, the corrupted bit of the single byte in cipher text block 8 bytes might spread to all n bits in plaintext block 8 bytes since we do the decryption one block at a time.

(2) CBC mode

(a) Create a file that is 64 bytes long. Attach a screen shot of the file.

```
[03/12/21]seed@SEED:~$ echo "Create a file and need to Encrypt it and followed by decrpyt it">cbctxt.txt
[03/12/21]seed@SEED:~$
```

(b) Encrypt the file using CBC. Attach a screen shot of the terminal showing your encryption command.

```
[03/12/21]seed@SEED:~$ openssl enc -aes-128-cbc -e -in cbctxt.txt -out cbc1.txt -k 00112233445566778889aabbccddeeff -iv 0102030405060708
[03/12/21]seed@SEED:~$
```

(c) Corrupt the file. Attach a screen shot of the corrupted file (the corrupted file should be displayed with the same software/command used to display the original file).

(d) How much information can you recover by decrypting the corrupted file?

Decrypting the corrupted file, we can recover around 45 bytes out of 62 bytes.

(e) Explain why in terms of the decryption procedures of both CBC.

The corrupted bytes of the cipher text were decrypted using the key it will affect the corresponding plain text formed and when the same corrupted cipher block was XOR with the decrypted cipher block only the corresponding byte in the decrypted block would be corrupted.