Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

Lab 2

Assignment- Secret-Key Encryption

2.1 Task 1: Frequency Analysis

It is well-known that monoalphabetic substitution cipher (also known as monoalphabetic cipher) is not secure, because it can be subjected to frequency analysis.

The Encryption key here is ‘SJMVBOLCKYUNQPTEWZDXFGRHIA’

By using given command in guideline: $ tr 'aet' 'XGE' < in.txt > out.txt

Using frequency analysis I figured out the encryption key and the original plaintext.

Plaintext found is shown in the below screenshot.

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

2.2 Task 2: Encryption using Different Ciphers and Modes

To handle encryption of larger plaintext, block cipher cryptography gets used. It divides plaintext into multiple blocks and applies cryptography. There are different modes of block cipher. Few of them are used in this task.

Cipher Block Chaining (CBC): It chains blocks together. Makes encryption probabilistic and combines encryption of all the blocks. It uses randomness to achieve no-deterministic approach. Uses Initialization Vector (IV) to initialize CBC mode.

Read about Cipher Feedback (CFB) and Output Feedback (OFB) from link provided.

Plaintext file was created using following commands:

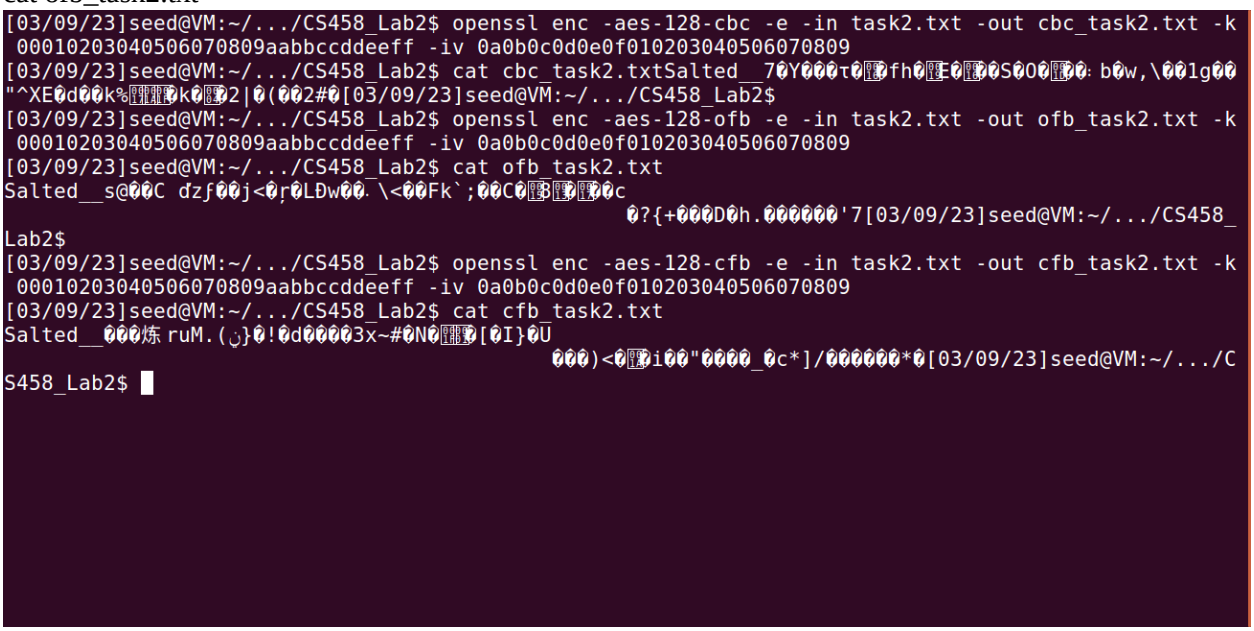
$ gedit plain.txt -→ here plaintext is saved in Task2.txt file

After creation of plaintext file, I used aes-128-cbc, aes-128-ofb, aes-128-cfb cipher types for encryption.

cbc_task2.txt, cfb_task2.txt, ofb_task2.txt are the ciphertext files.

Commands used:

1. openssl enc -aes-128-cbc -e -in task2.txt -out cbc_task2.txt -k 00010203040506070809aabbccddeeff \ -iv 0a0b0c0d0e0f010203040506070809
2. cat cbc_task2.txt\
3. openssl enc -aes-128-cfb -e -in task2.txt -out cfb_task2.txt -k 00010203040506070809aabbccddeeff \ -iv 0a0b0c0d0e0f010203040506070809
4. cat cfb_task2.txt
5. openssl enc -aes-128-ofb -e -in task2.txt -out ofb_task2.txt -k 00010203040506070809aabbccddeeff \ -iv 0a0b0c0d0e0f010203040506070809
6. cat ofb_task2.txt



Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

2.3 Task 3: Encryption Mode – ECB vs. CBC

2.3.1 Using ECB (Electronic Code Book) and CBC (Cipher Block Chaining) to encrypt image provided.

Provide image is saved as pic_original.bmp

I encrypted provide image by ECB mode by using following commands:

openssl enc -aes-128-ecb -e -in pic_original.bmp -out ecb_pic.bmp

I encrypted provide image by CBC mode by using following commands:

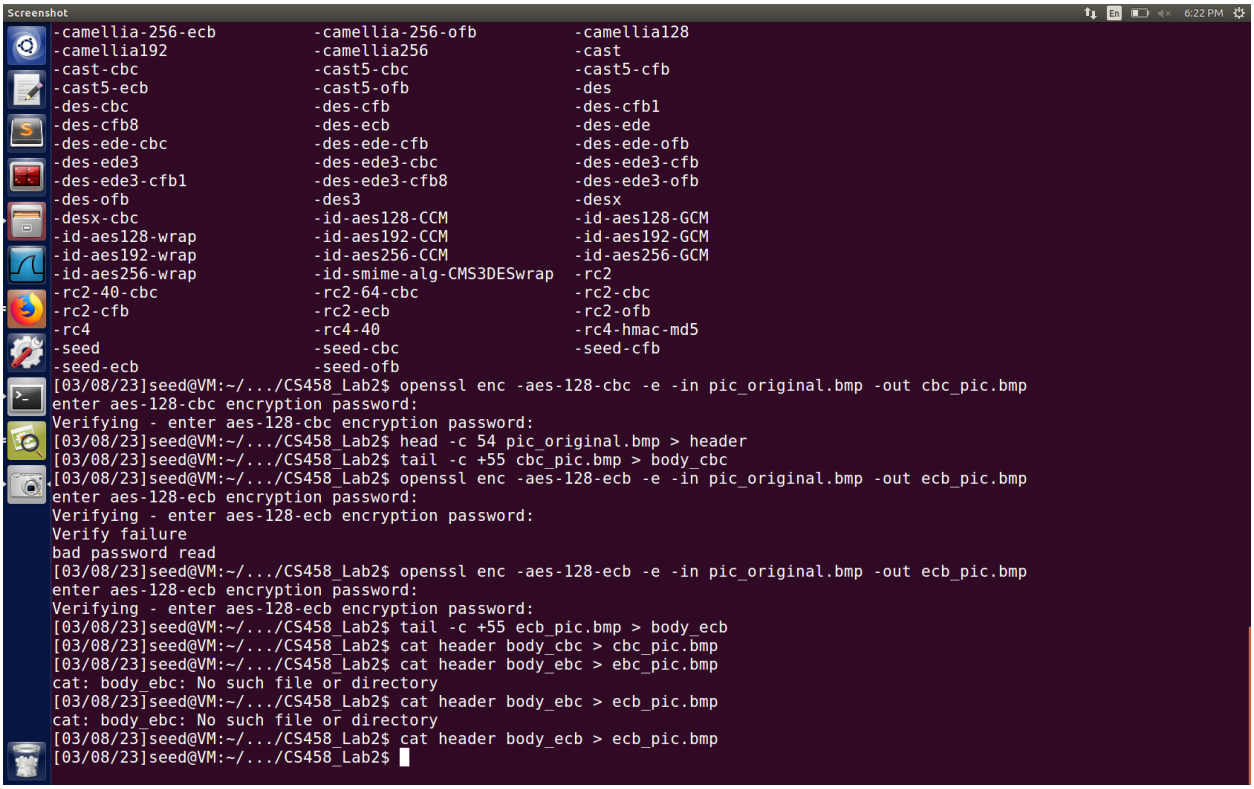
“openssl enc -aes-128-cbc -e -in pic_original.bmp -out cbc_cipher_image.bmp”

As input image is in bitmap format, following procedures were made:

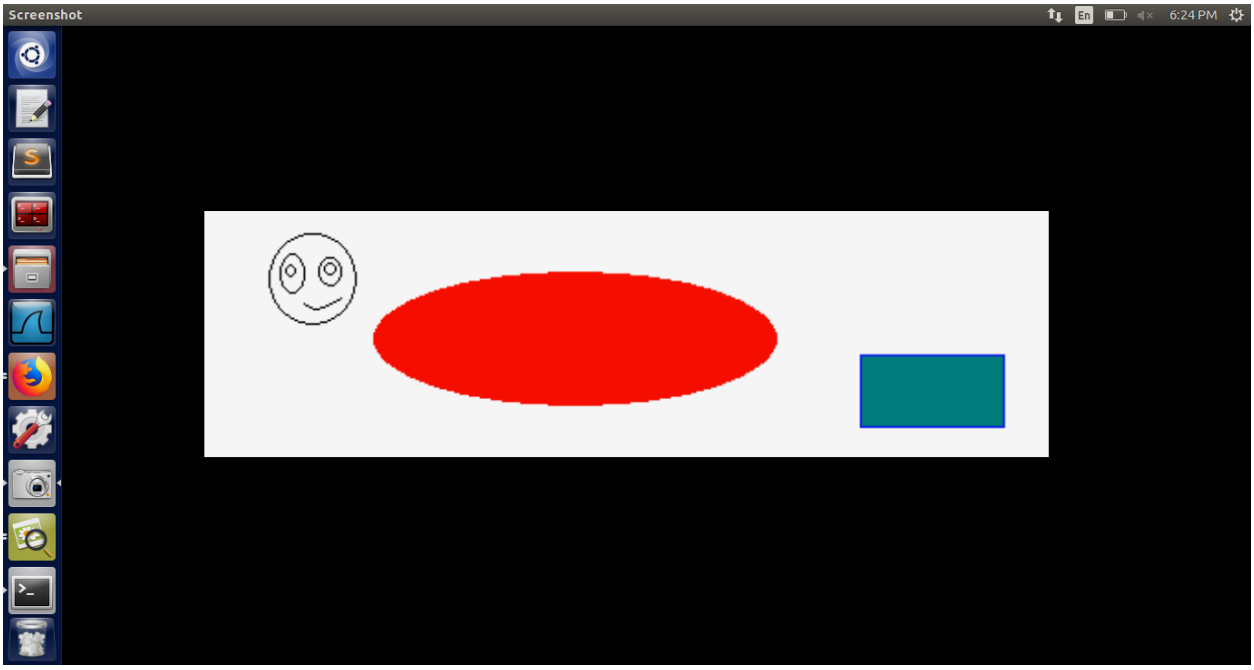
1. In order to treat encrypted file as a legitimate .bmp file, the first 54 bytes were set correctly because they contain the header information about the picture.
2. Replaced the header of the encrypted picture with that of the original picture.
3. Used following commands to get the header from pic_original.bmp, the data (from offset 55 to the end of the file) from ecb_pic.bmp
4. Combined the header and data together into a new file.
5. Following Commands were used:
   head -c 54 pic_original.bmp > header
   tail -c +55 cbc_pic.bmp > body_cbc
   tail -c +55 ecb_pic.bmp> body_ecb
   Cat header body_cbc > cbc_pic.bmp
   Cat header body_ecb > ecb_pic.bmp

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722



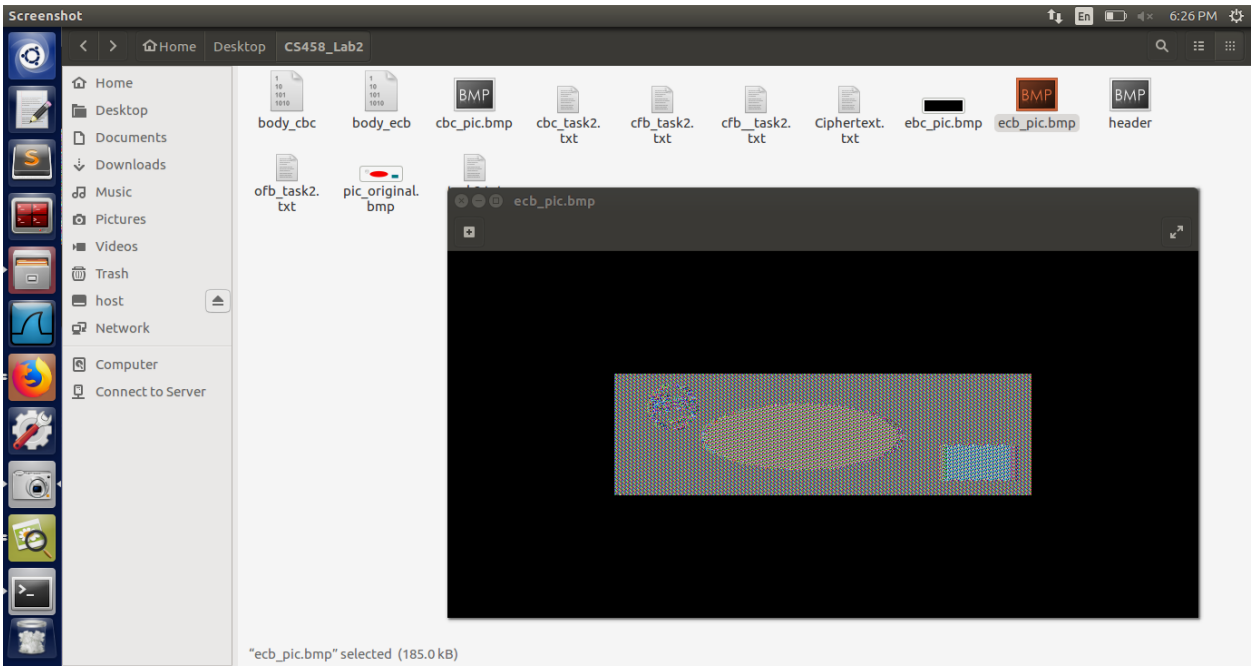
Original Image provide for encryption:



Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

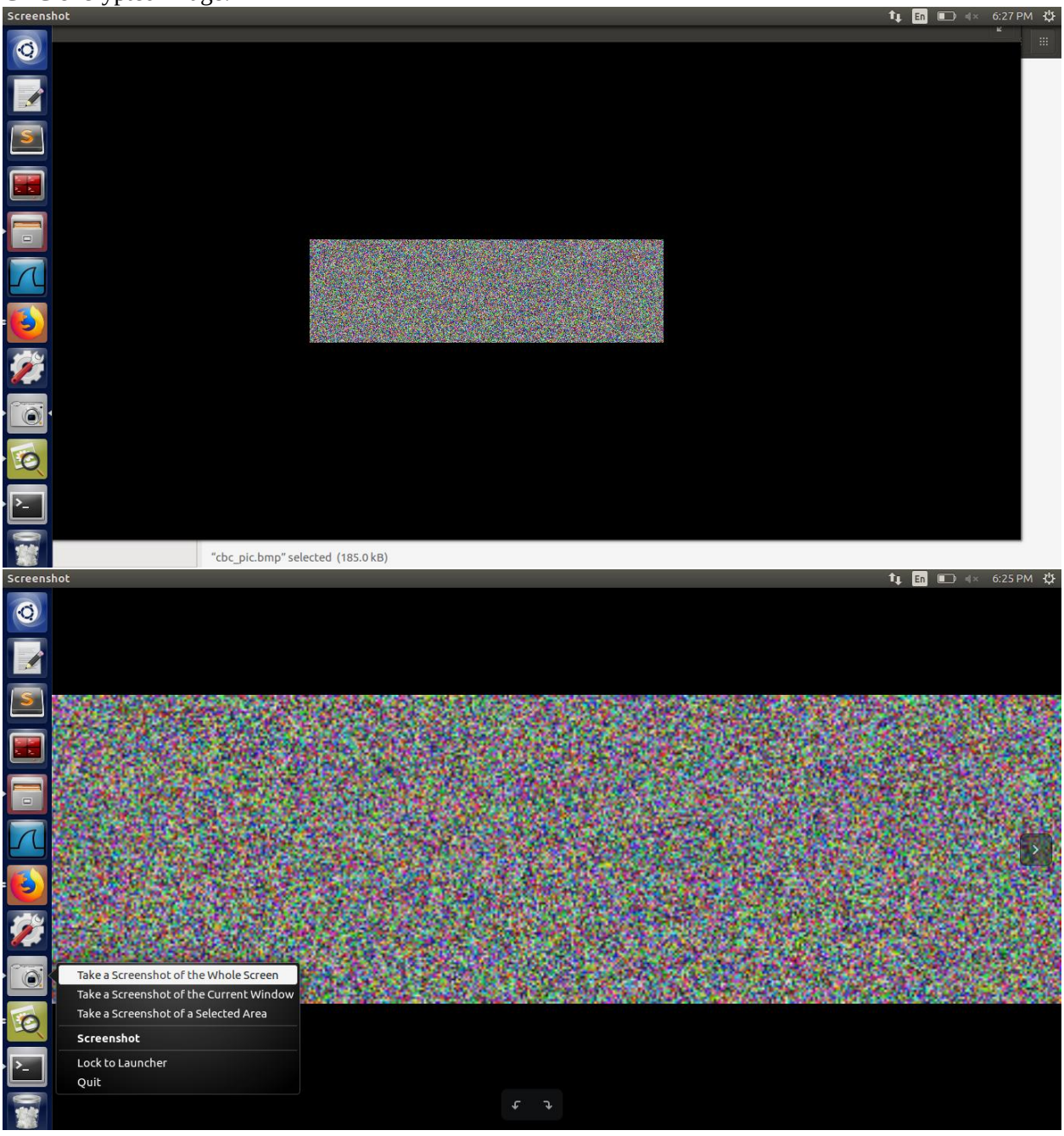
ECB encrypted image:



Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

CBC encrypted image:

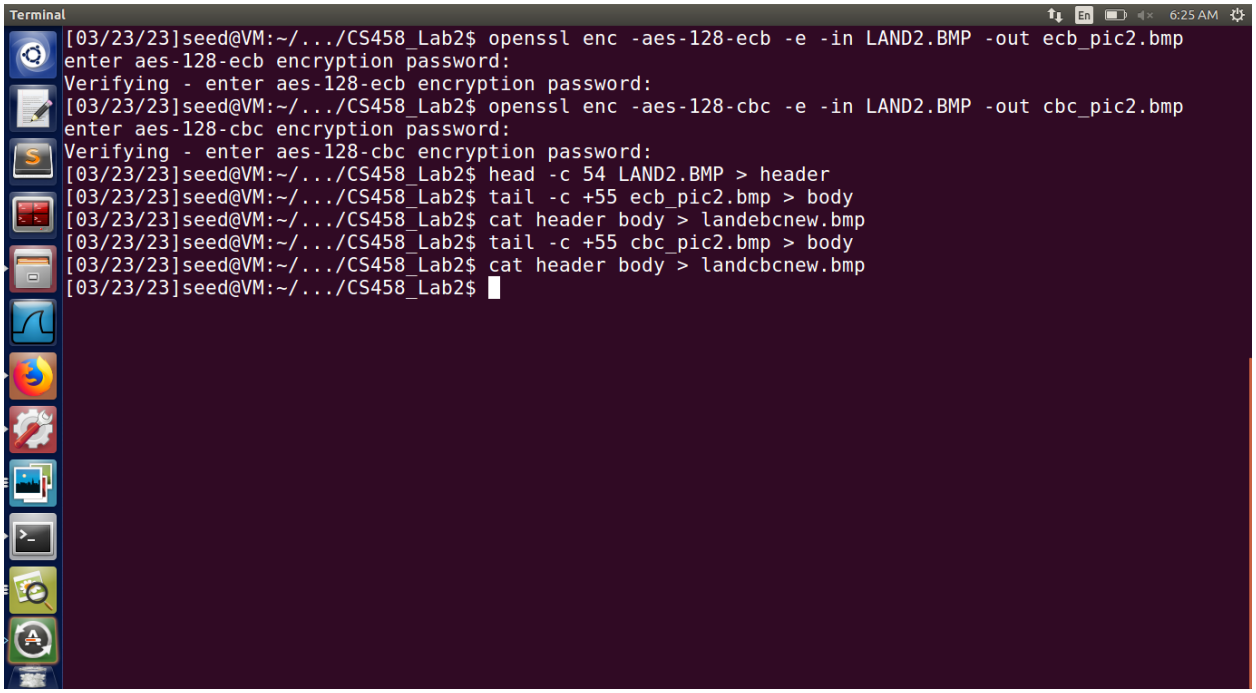


After evaluating result of both CBC and ECB encryption, we can clearly see that :

1. Each block of pixels is independently encrypted using the same key when encrypting an image in ECB (Electronic Codebook) mode. Regardless of where they are located within the image, identical blocks of pixels will result in identical blocks of ciphertext when this happens. As a

result, some patterns might be recognizable in the encrypted image, which might reveal something about the original image.

2. In contrast, each block of pixels in CBC (Cipher Block Chaining) mode is encrypted using the ciphertext from the block before it as an additional input. This increases the complexity of the encryption process and the reliance on the preceding ciphertext, making it more challenging to spot patterns in the encrypted image. Also, CBC mode can offer more security than ECB mode. Since the same block of pixels encrypted with the same key will produce different ciphertexts when encrypted at different times, due to the different previous ciphertexts used as input.
3. As a result, following the comparison of ECB and CBC for picture encryption, the following conclusions may be drawn:
    - While the encrypted image created in CBC mode is less likely to do so, the encrypted image created in ECB mode may expose certain patterns or information about the original image.
    - The CBC mode's encrypted picture might be more secure than the ECB mode's because it is less vulnerable to certain kinds of attacks that take advantage of ciphertext pattern-exploiting techniques.
    - As each block can be encrypted separately, the encrypted image created in ECB mode may require less processing power than the one created in CBC form. The compromise is a less robust security guarantee.



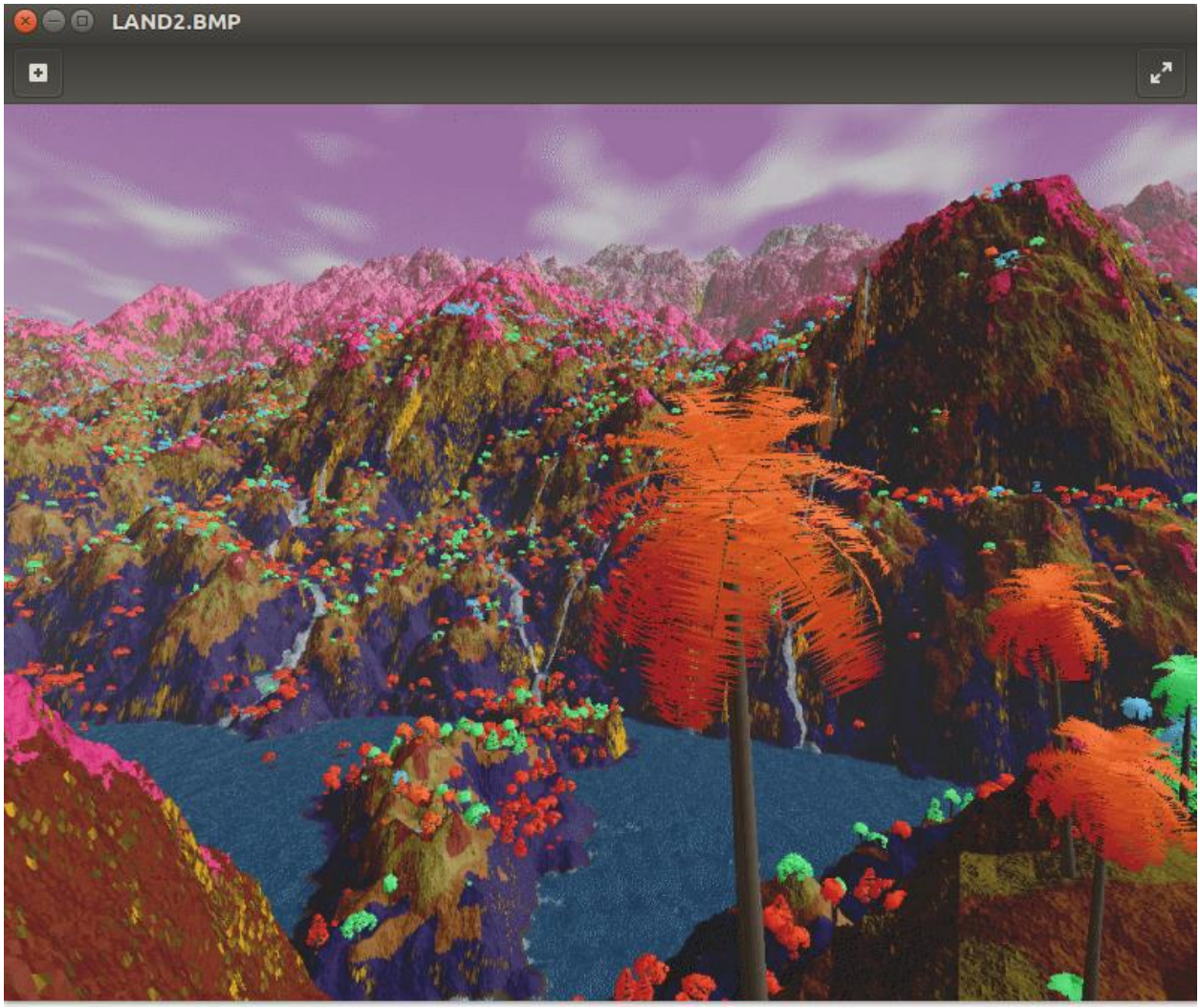
2.3.1 Select a picture of your choice, repeat the experiment above, and report your observations.

Repeating all the steps again and following are the screenshots of results:

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

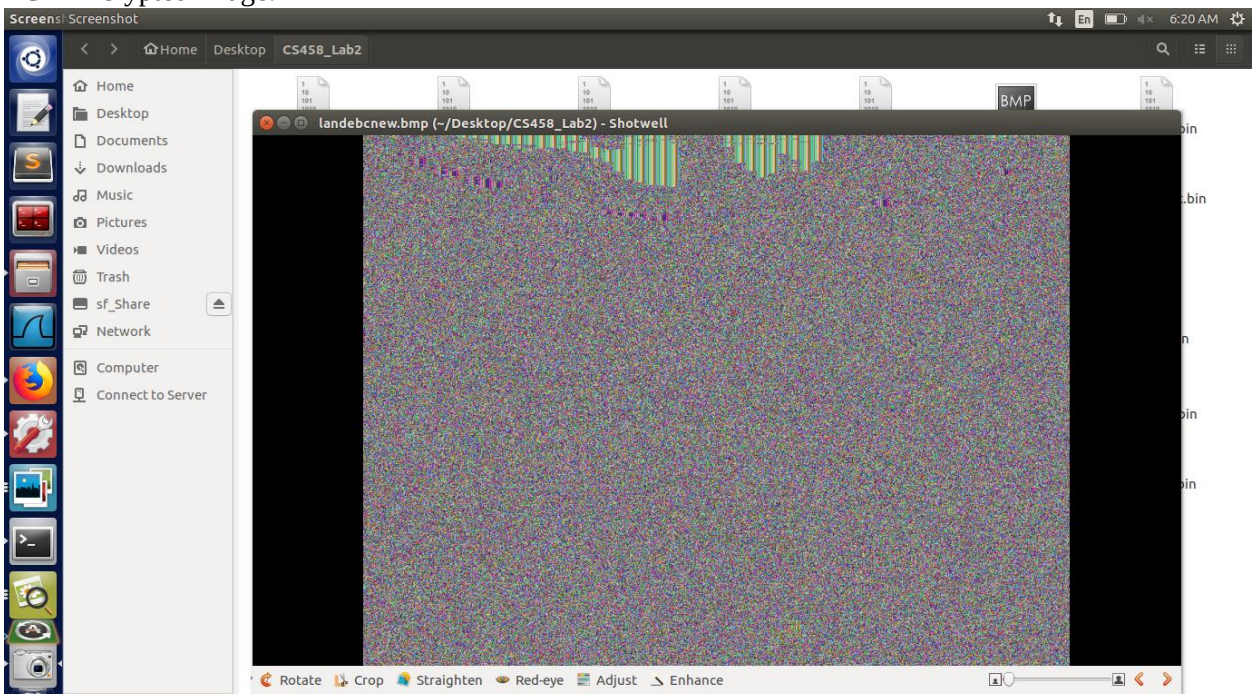
Original Image:



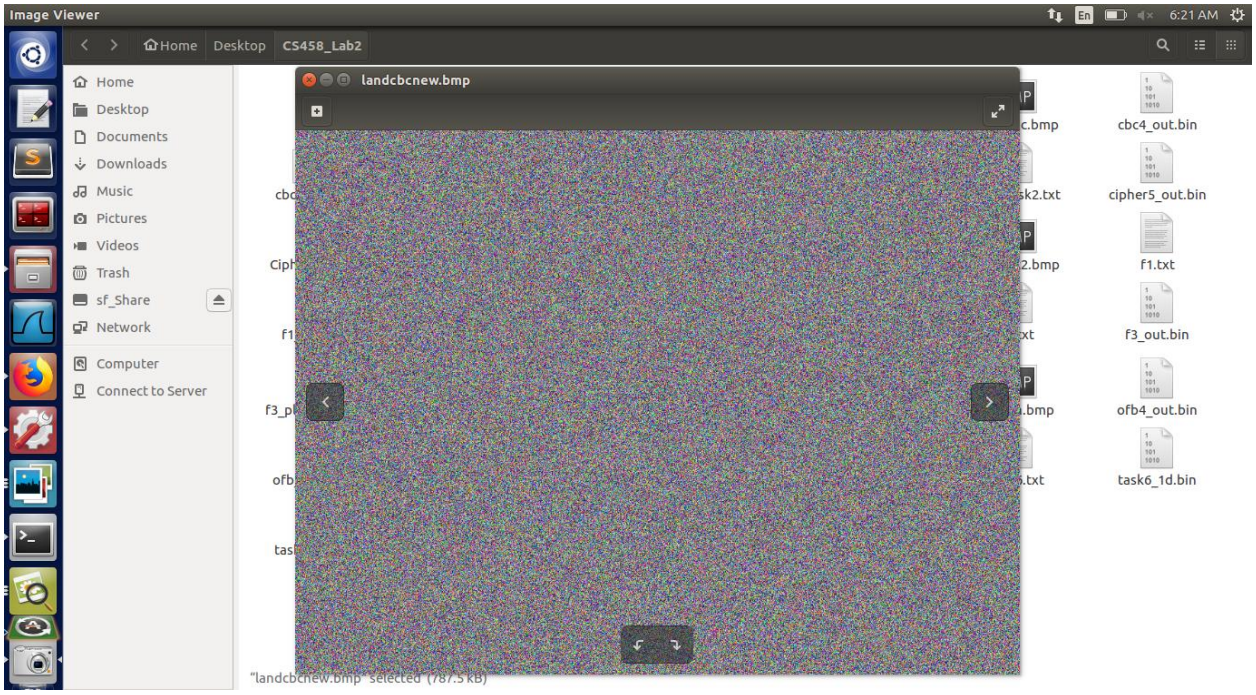
Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

ECB Encrypted image:



CBC encrypted image:



Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

2.4 Task

4 : Padding

Q. Use ECB, CBC, CFB, and OFB modes to encrypt a file (you can pick any cipher). Please report which modes have paddings and which ones do not. For those that do not need paddings, please explain why?

ECB: Electronic Code Book mode does not require padding, as the plaintext is simply divided into blocks of a set size and each is separately encrypted. Due to the fact that identical plaintext blocks would be converted to the same ciphertext in this mode, it is vulnerable to plaintext attacks.

CBC: Cipher Block Chaining mode necessitates padding because the plaintext is divided into blocks of a predetermined size, but each block is first concatenated with the preceding ciphertext block via an XOR operation before being encrypted. For this technique, uniformly sized plaintext blocks are necessary.

CFB: The cipher feedback mode necessitates padding since it divides the plaintext into blocks of a certain size, with each block being encrypted and merged with the one before it via an XOR operation.

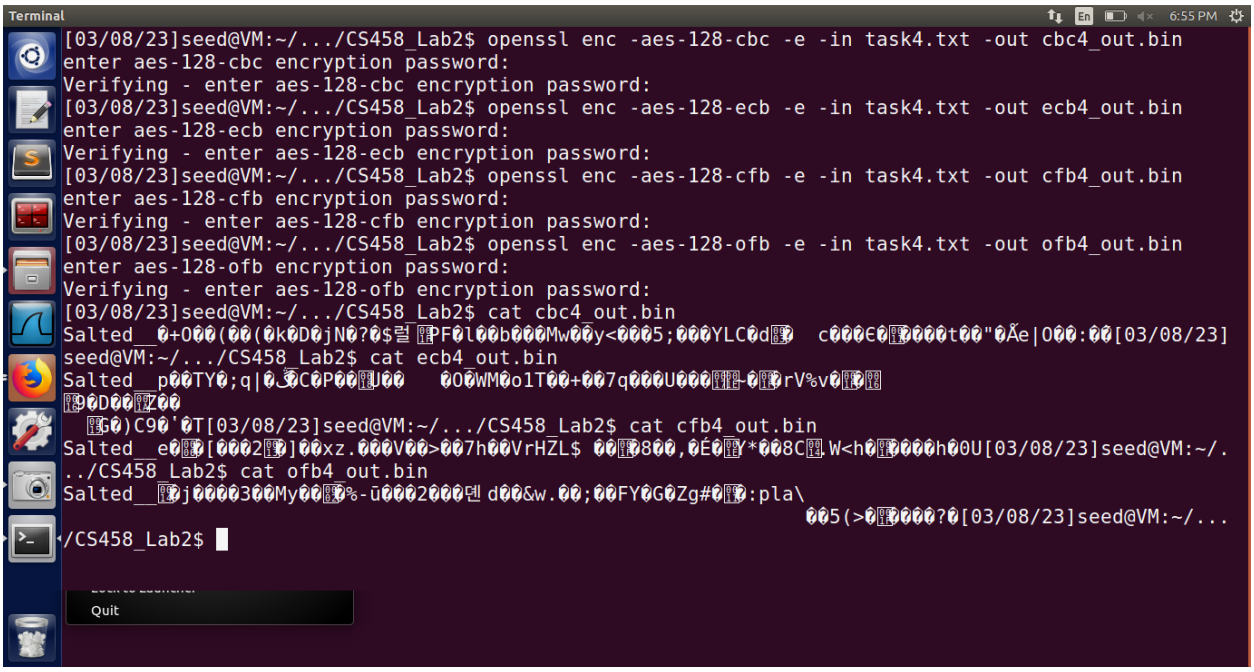
OFB: Output Feedback mode does not require padding, as the plaintext is broken into blocks of a set size, and each block is encrypted and combined with the previous ciphertext block using an XOR operation. This mode does not require the plaintext blocks to be the same size, as it only uses the output of the encryption operation as the input for the next encryption operation.

Q2. Let us create three files, which contain 5 bytes, 10 bytes, and 16 bytes, respectively. We can use the following echo -n command to create such files. The following example creates a file f1.txt with length 5 (without the -n option, the length will be 6, because a newline character will be added by echo):

```
$ echo -n "12345" > f1.txt
```

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722



Created 3 files of lengths 5,10,16 bytes and named them f1.txt, f2.txt, f3.txt respectively.

Commands used:

$ gedit f1.txt

$ gedit f2.txt

$ gedit f3.txt

After creation of text files, encrypted by using CBC mode.

Commads used:

Openssl enc -aes-128-cbc -e -in f1.txt -out f1_out.bin

Openssl enc -aes-128-cbc -e -in f2.txt -out f2_out.bin

Openssl enc -aes-128-cbc -e -in f3.txt -out f3_out.bin

Once encryption is done, all cipher files were decrypted using following commands:

Openssl enc -aes-128-cbc -d -in f1_out.bin -out f1_plaintext.txt

Openssl enc -aes-128-cbc -d -in f2_out.bin -out f2_plaintext.txt

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

Openssl enc -aes-128-cbc -d -in f3_out.bin -out f3_plaintext.txt

To open and read decrypted text following commands were used:

Xxd -g 1 f1_plaintext.txt

Xxd -g 1 f2_plaintext.txt

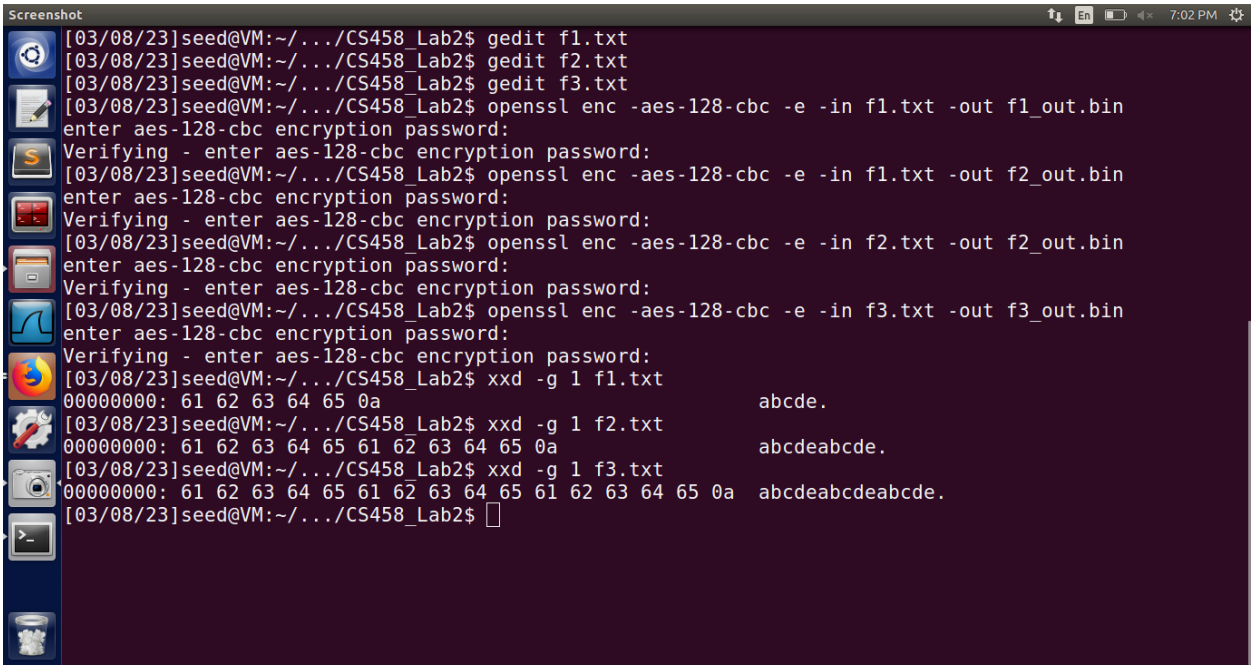
Xxd -g 1 f3_plaintext.txt

As you can see in following screenshot decrypted data is:

61 62 63 64 65 0a

61 62 63 64 65 61 62 63 64 65 0a

61 62 63 64 65 61 62 63 64 65 61 62 63 64 65 0a



Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

Padding data may not be printable, so we are using a hex tool to display the content.

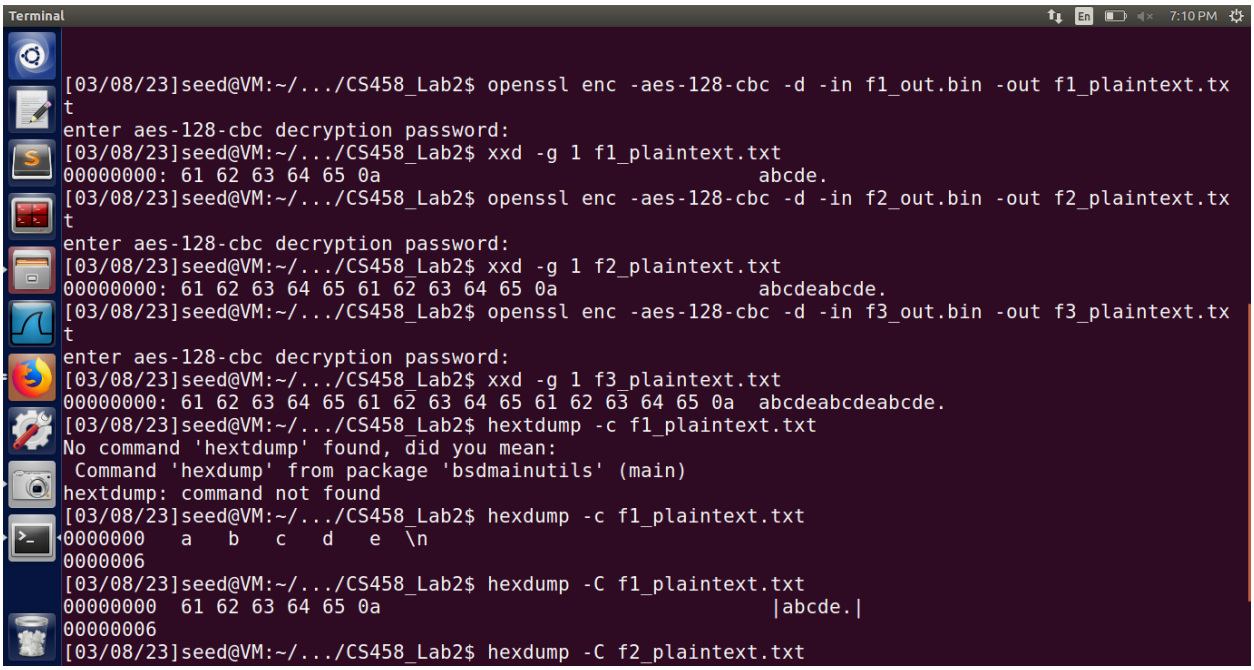
The following screenshot shows a file in the hex format.

Following commands were used:

Hexdump -C f1_plaintext.txt

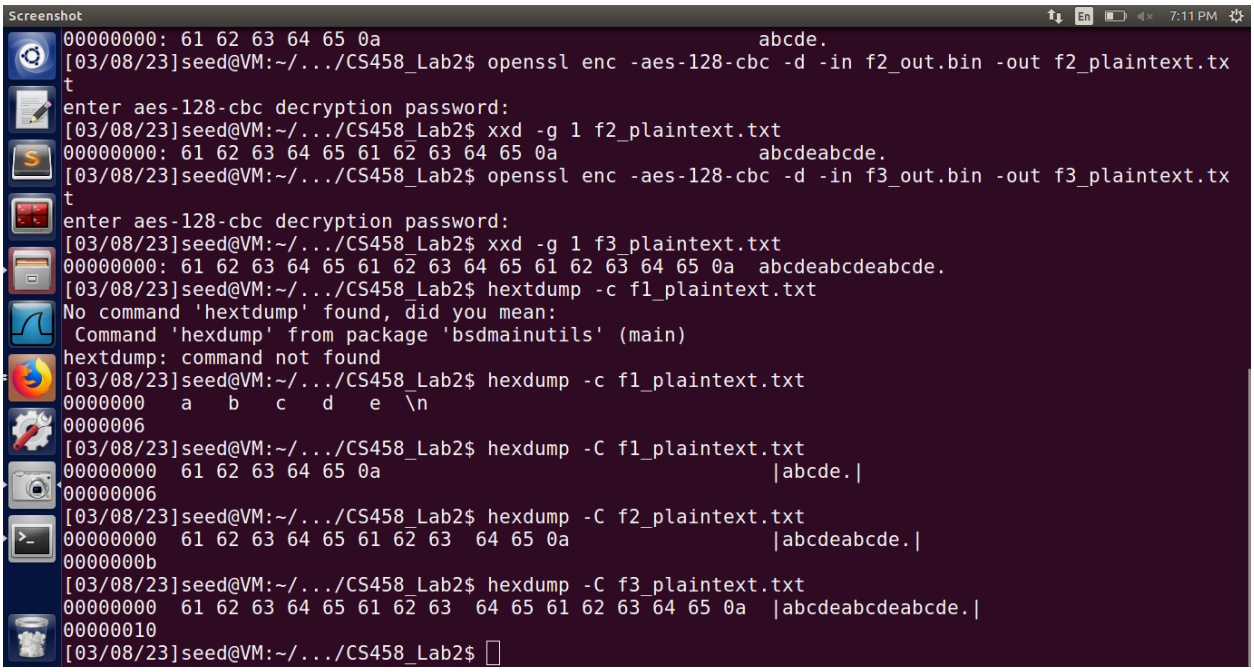
Hexdump -C f2_plaintext.txt

Hexdump -C f3_plaintext.txt



Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722



2.5 Task 5: Error Propagation – Corrupted Cipher Text

How much information can you recover by decrypting the corrupted file, if the encryption mode is ECB, CBC, CFB, or OFB, respectively?

Decrypting a corrupted file may recover some information, but how much depends on the type of encryption technique employed, the encryption key, the degree of corruption, and the mode of operation.

ECB: Each block of the encrypted file is encrypted separately from the other blocks if the encryption method is ECB (Electronic Codebook). As a result, if a block of the file becomes corrupt, it just affects that one particular block, and the remaining blocks can still be correctly decrypted. The quantity of data that may be recovered from a corrupted file in ECB mode will thus be based on the quantity and size of corrupted blocks.

CBC: Each block of the encrypted file is XORed with the preceding block before encryption if the encryption method is CBC (Cipher Block Chaining). Because the XOR method propagates errors, if a block of the file is faulty, it will influence the decryption of all succeeding blocks. As a result, the degree of corruption and the location of the infected block within the file will determine how much information can be recovered from a corrupted file in CBC mode.

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

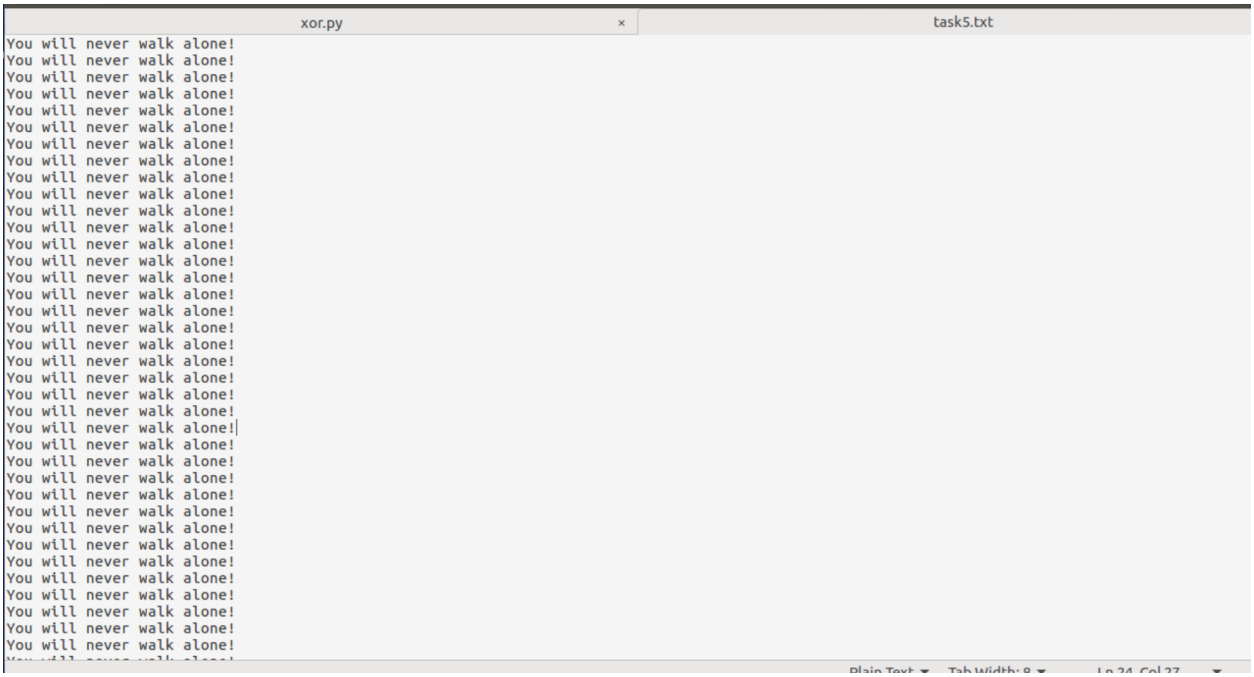
CFB and OFB: If CFB (Cipher Feedback) or OFB (Output Feedback) is selected as the encryption mode, the encryption is carried out on a feedback basis, where the output of one encryption serves as the input for the subsequent encryption. In either mode, the encryption operation doesn't use the plaintext; it just uses the previous output and the key. Because of this, if a block of the file is corrupted, it will only effect that particular block's decryption; the remaining blocks can still be correctly decrypted. The number and size of the corrupted blocks will therefore determine how much data may be recovered from a corrupted file in CFB or OFB mode.

In conclusion, the degree and location of the contamination, as well as the encryption mechanism employed, will all affect how much data may be recovered from a corrupted file. The least resilient against corruption is often the ECB mode, followed by the more resilient CFB and OFB modes, and the intermediate CBC mode.

1. Create a text file that is at least 1000 bytes long.

```
03/08/23]seed@VM:~/.../CS458_Lab2$ fallocate -l 1k task5.txt
```

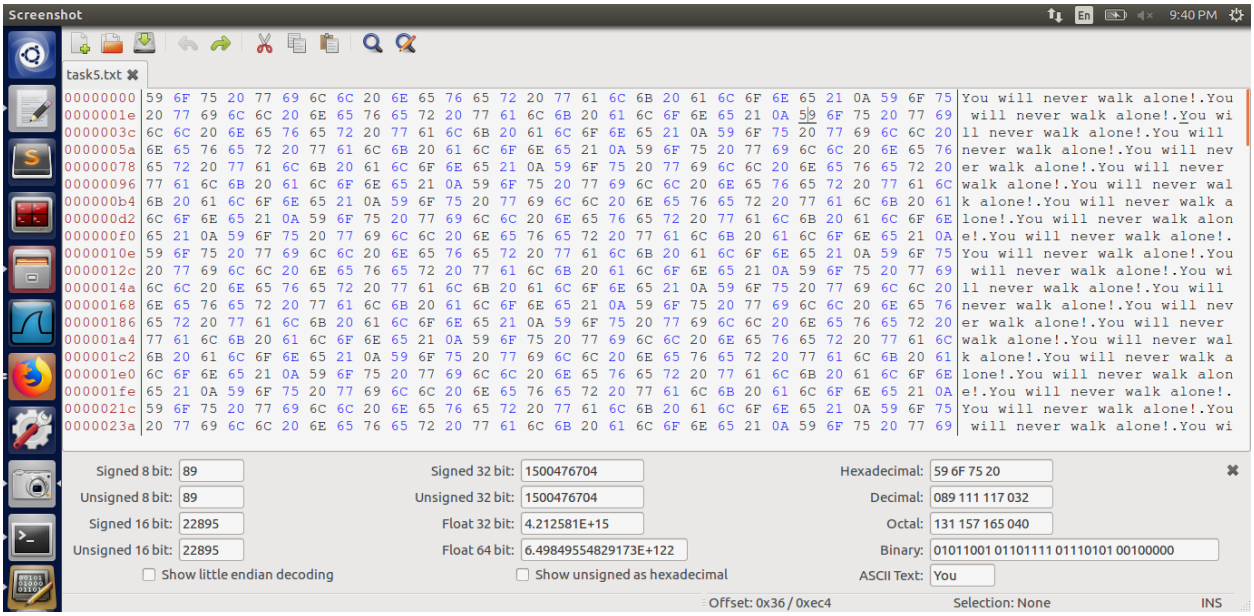
Created txt file



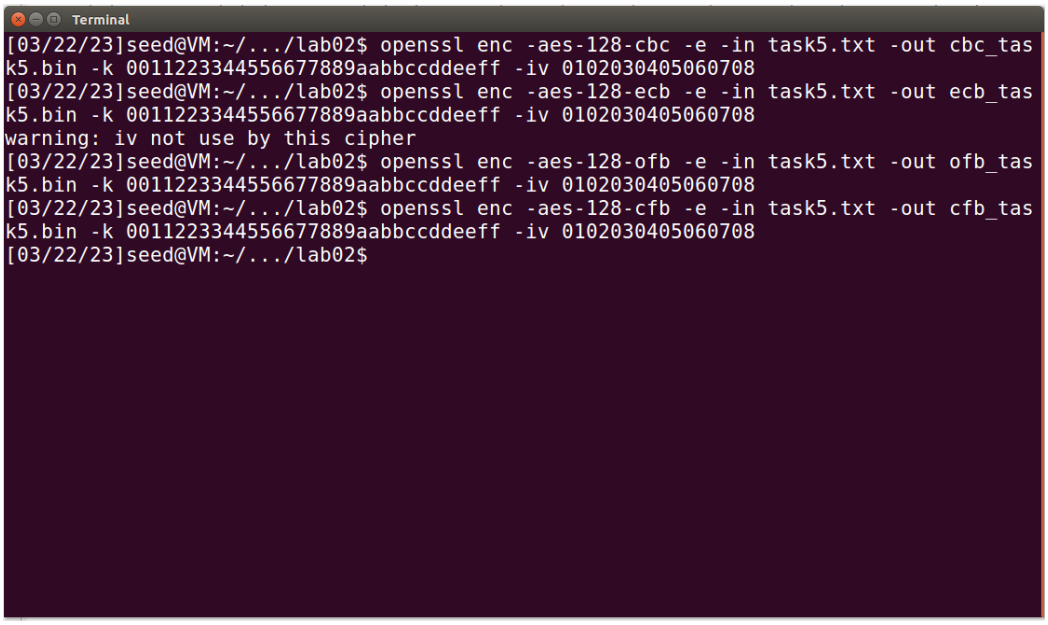
Before encryption

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722



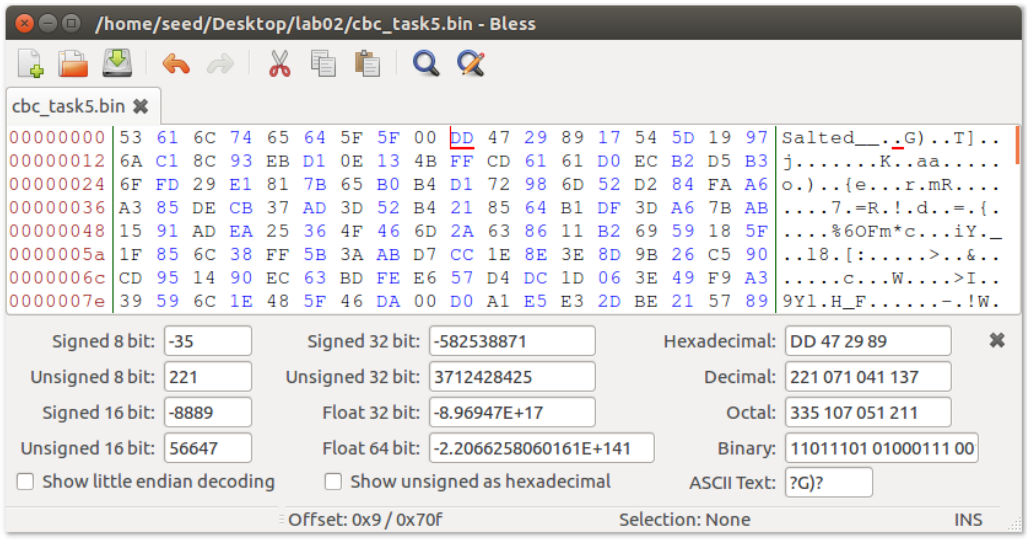
2. Encrypt the file using the AES-128 cipher.

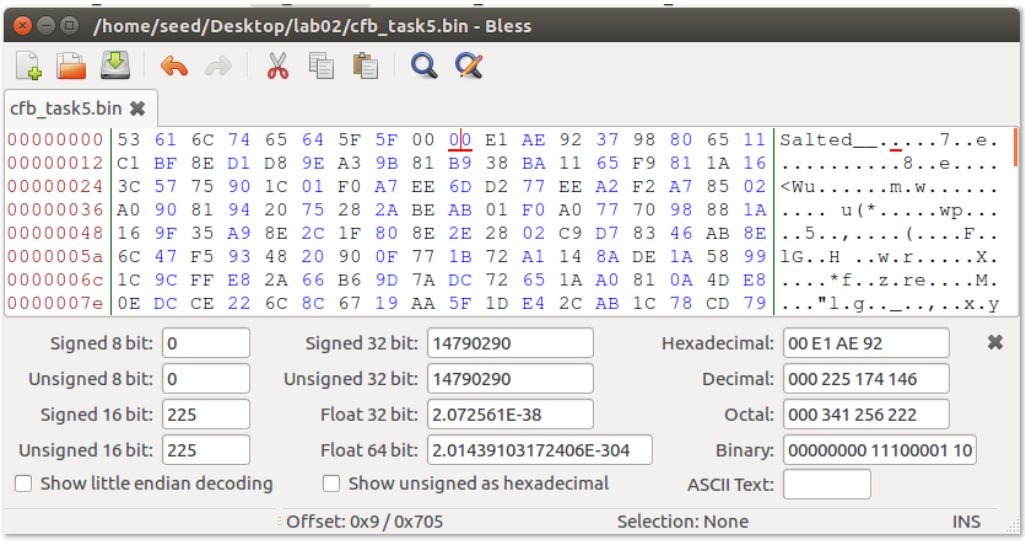


4. Unfortunately, a single bit of the 55th byte in the encrypted file got corrupted. You can achieve this corruption using the bless hex editor.
   :After encryption I changed 55th bit to zero to represent error.

Introduction to Information Security - CS 458 - Spring 2022

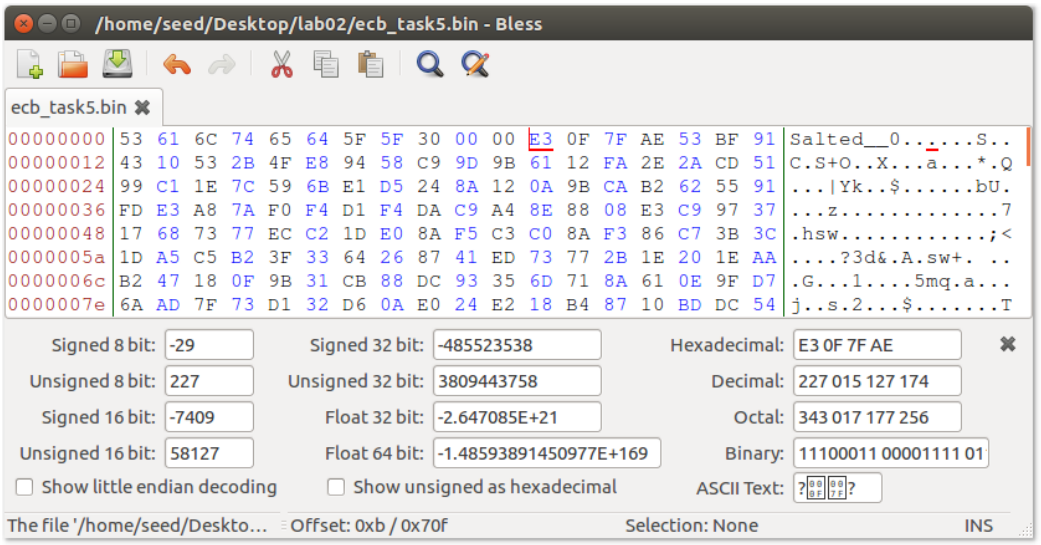
Neha Ramesh Gawali A20523722

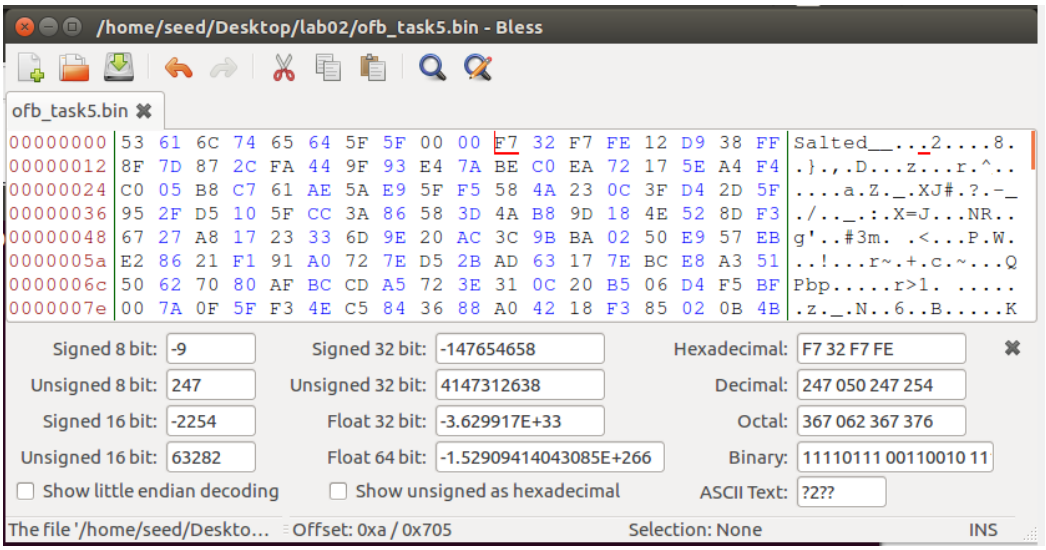




Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

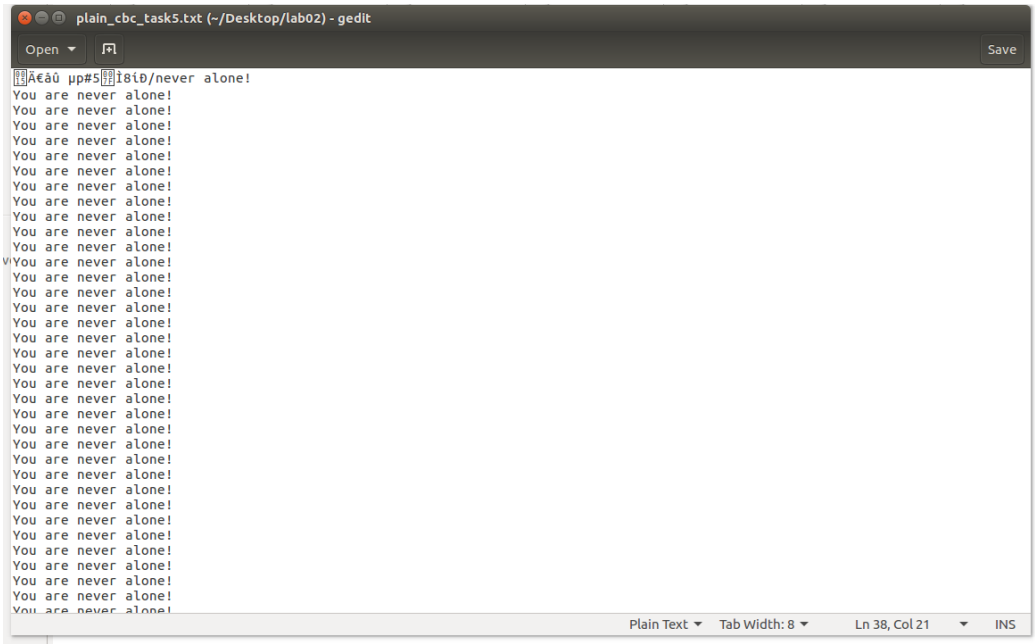




4. Decrypt the corrupted ciphertext file using the correct key and IV.

```
Terminal
[03/22/23]seed@VM:~/.../lab02$ openssl enc -aes-128-cbc -e -in task5.txt -out cbc_task5.bin -k 00112233
44556677889aabbccddeeff -iv 0102030405060708
[03/22/23]seed@VM:~/.../lab02$ openssl enc -aes-128-ecb -e -in task5.txt -out ecb_task5.bin -k 00112233
44556677889aabbccddeeff -iv 0102030405060708
warning: iv not use by this cipher
[03/22/23]seed@VM:~/.../lab02$ openssl enc -aes-128-ofb -e -in task5.txt -out ofb_task5.bin -k 00112233
44556677889aabbccddeeff -iv 0102030405060708
[03/22/23]seed@VM:~/.../lab02$ openssl enc -aes-128-cfb -e -in task5.txt -out cfb_task5.bin -k 00112233
44556677889aabbccddeeff -iv 0102030405060708
[03/22/23]seed@VM:~/.../lab02$ openssl enc -aes-128-cfb -d -in cfb_task5.bin -out plain_cfb_task5.txt -
k 00112233445566778899aabbccddeeff -iv 0102030405060708
[03/22/23]seed@VM:~/.../lab02$ openssl enc -aes-128-ofb -d -in ofb_task5.bin -out plain_ofb_task5.txt -
k 00112233445566778899aabbccddeeff -iv 0102030405060708
[03/22/23]seed@VM:~/.../lab02$ openssl enc -aes-128-cfb -d -in ecb_task5.bin -out plain_ecb_task5.txt -
k 00112233445566778899aabbccddeeff -iv 0102030405060708
[03/22/23]seed@VM:~/.../lab02$ openssl enc -aes-128-ofb -d -in cbc_task5.bin -out plain_cbc_task5.txt -
k 00112233445566778899aabbccddeeff -iv 0102030405060708
[03/22/23]seed@VM:~/.../lab02$
```

Neha Ramesh Gawali A20523722



Conclusion:

Each block of the plaintext is separately encrypted in ECB (Electronic Code Book) mode, thus an error only affects the corresponding block in the decoded plaintext. The other bytes of the decrypted file will therefore be correct, and just the 55th byte will be erroneous.

The damaged byte will have an impact on the decryption of both the current and all following blocks in CBC (Cipher Block Chaining) mode. As a result, the decrypted file's subsequent bytes will all be erroneous.

The erroneous byte will only impact the matching byte in the decrypted file and will not spread to additional bytes in CFB (Cipher Feedback) and OFB (Output Feedback) modes. The erroneous byte will only impact the matching byte in the decrypted file and will not spread to additional bytes in CFB (Cipher Feedback) and OFB (Output Feedback) modes. In the decrypted file, just the 55th byte will be inaccurate; all other bytes will be accurate.

The task's outcomes will support the aforementioned assertions.

2.6 Task 6: Initial Vector (IV)

2.6.1 Task 6.1. Uniqueness of the IV

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

To guarantee the secrecy and integrity of the data being sent, Initialization Vector (IV) is a crucial element utilized in symmetric key encryption techniques. An IV's primary function is to introduce randomness and unpredictable elements to the encryption process, which makes it more difficult for an attacker to anticipate or duplicate the encrypted data.

Uniqueness is a fundamental condition for an IV, which means that no IV may be used more than once under the same key. This is due to the fact that if the same IV and key are used, the ciphertext that results would be identical, leaving it susceptible to certain attacks such replay assaults, in which an attacker can intercept and replay the same message.

Let's look at an instance where the same IV is utilized for two distinct messages that use the same key in order to better appreciate why IV must be unique. An attacker in this situation could learn something about the plaintext by comparing the two ciphertexts and quickly spotting any patterns or similarities.

Consider a situation where an attacker is aware that two messages were encrypted with the same key but different IVs. The blocks of ciphertext that are identical between the two communications can be identified by comparison if the attacker manages to intercept both messages' ciphertexts. This could offer some hints about the plaintext, such as the message's length or format.

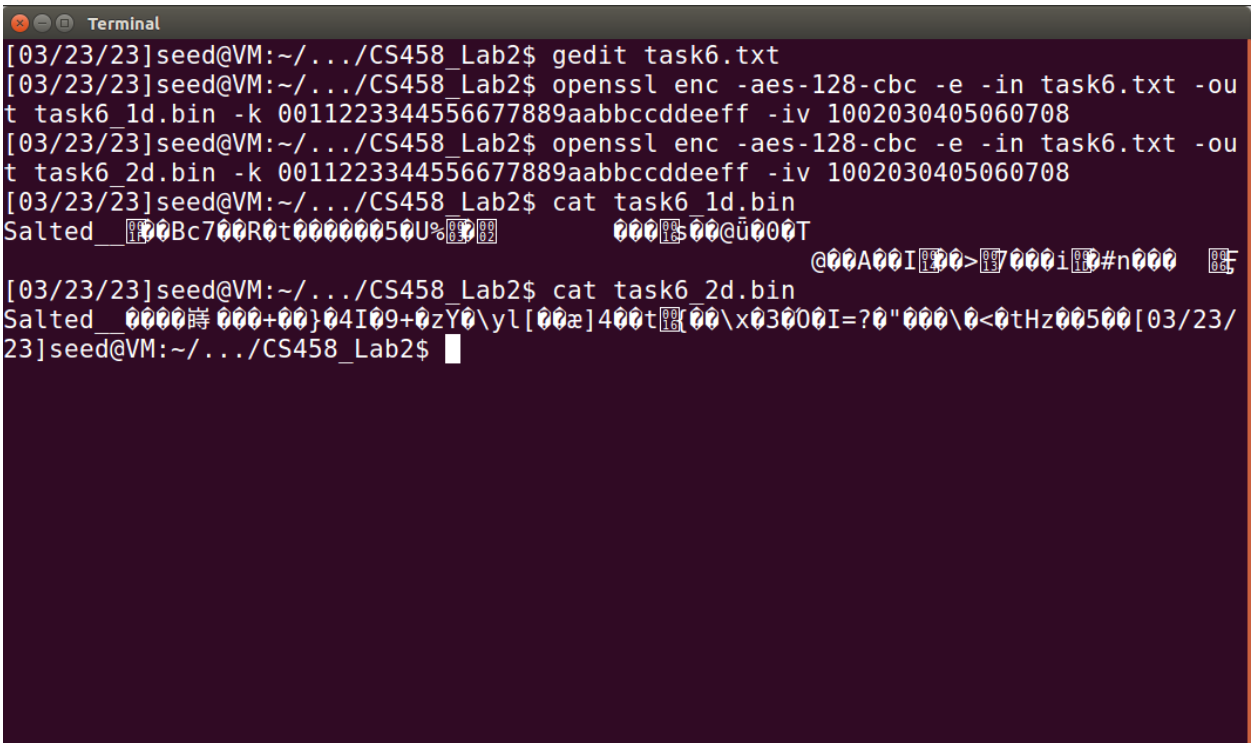
The ciphertexts, on the other hand, would seem utterly random and unrelated if the messages were encrypted with different IVs, even though the key is the same. This would prevent the attacker from finding any patterns or parallels.

Consequently, it is crucial to use a distinct IV for each message encrypted with the same key in order to ensure the security of the encryption process.

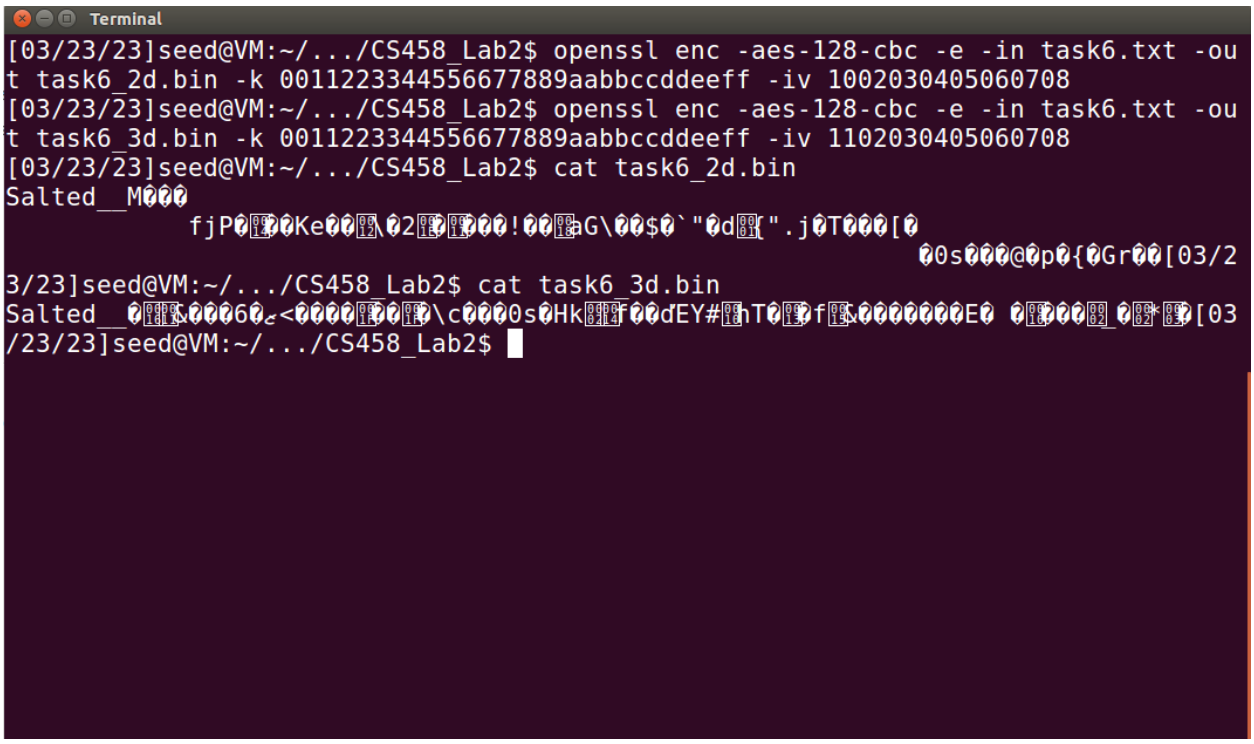
Similar IV:

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722



Different IV



Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

2.6.2 Task 6.2. Common Mistake: Use the Same IV

The resulting ciphertexts will vary when the identical plaintext is encrypted with two distinct IVs. This is so that the encryption process can produce a different sequence of ciphertext blocks if the IV changes. The IV serves as the beginning point for the encryption process. The resulting ciphertexts, however, will be the same when employing the same IV to encrypt the same plaintext. This is so that the encryption process can produce the identical sequence of ciphertext blocks every time because the IV serves as the beginning point for encryption.

If the same IV is used to encrypt two separate messages, an attacker with access to both ciphertexts can utilize the knowledge that the same IV was used to conduct a chosen-plaintext attack. Thus, IV must be unique. In this kind of attack, the attacker can utilize his or her understanding of the two ciphertexts to learn the encryption method or the key, which can then be used to decrypt further ciphertexts encrypted with the same key. As each encryption will start from a different position thanks to the usage of a distinct IV, the attacker is prevented from utilizing the same approach.

An attacker who has access to a known plaintext and its accompanying ciphertext can utilize the IV to decrypt other encrypted messages if the IV is constant in OFB mode. In the example, an attacker can use P1 and C1 to acquire the keystream used to encrypt P1 if the IV is the same. The recovery of P2 is therefore possible by using this ciphertext to decrypt C2.

If CFB mode is used instead of OFB mode and the IV never changes, an attacker can still perform a known-plaintext attack to learn more about the plaintext. The ciphertext will soon become dependent on the plaintext due to feedback from the preceding block, therefore the amount of information that can be gleaned will be restricted to the first few blocks of the plaintext. Consequently, only a piece of the plaintext can be recovered by the attacker, not the complete thing.

Here,

Plaintext (P1): This is a known message!

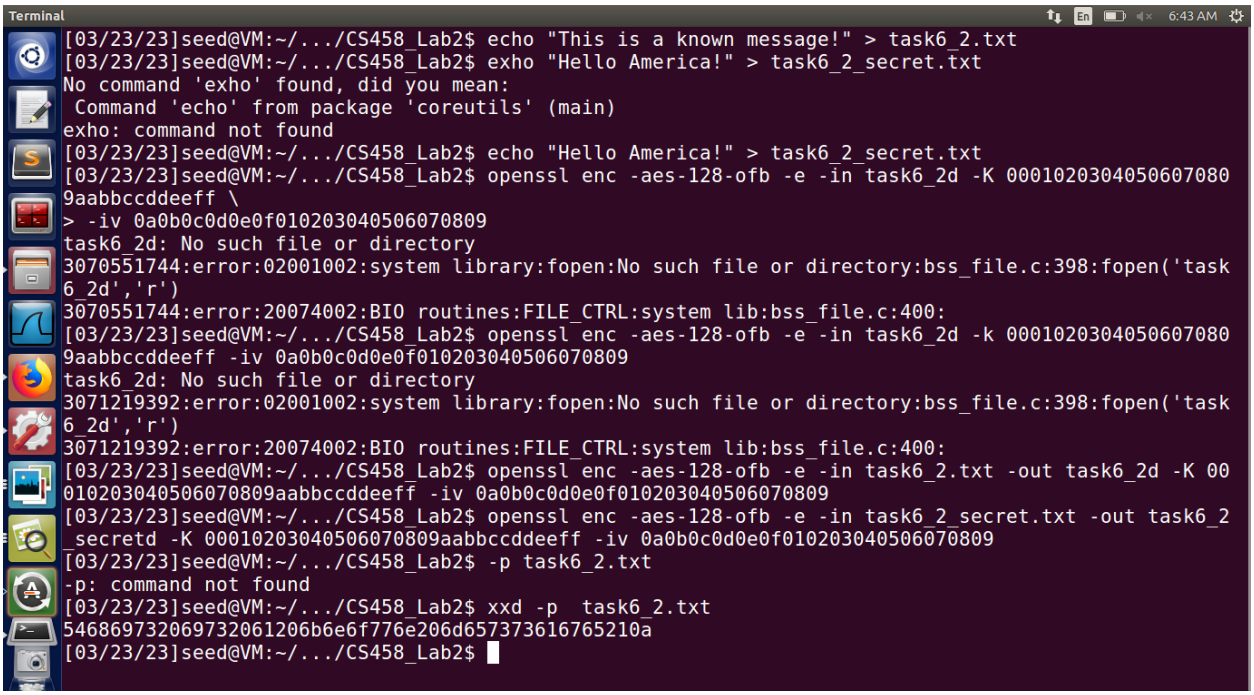
Ciphertext (C1): 546869732069732061206b6e6f776e206d657373616765210a

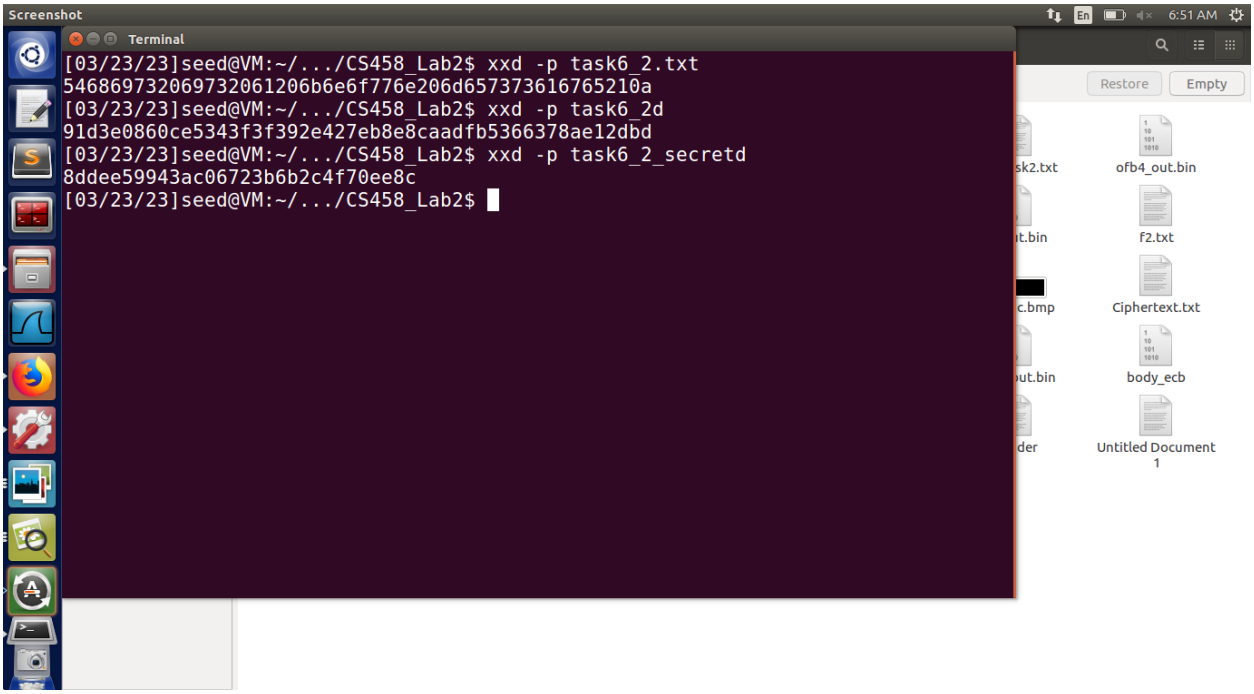
Plaintext (P2): Unknown

Ciphertext (C2): 91d3e0860ce5343f3f392e427eb8e8caadfb5366378ae12dbd

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722





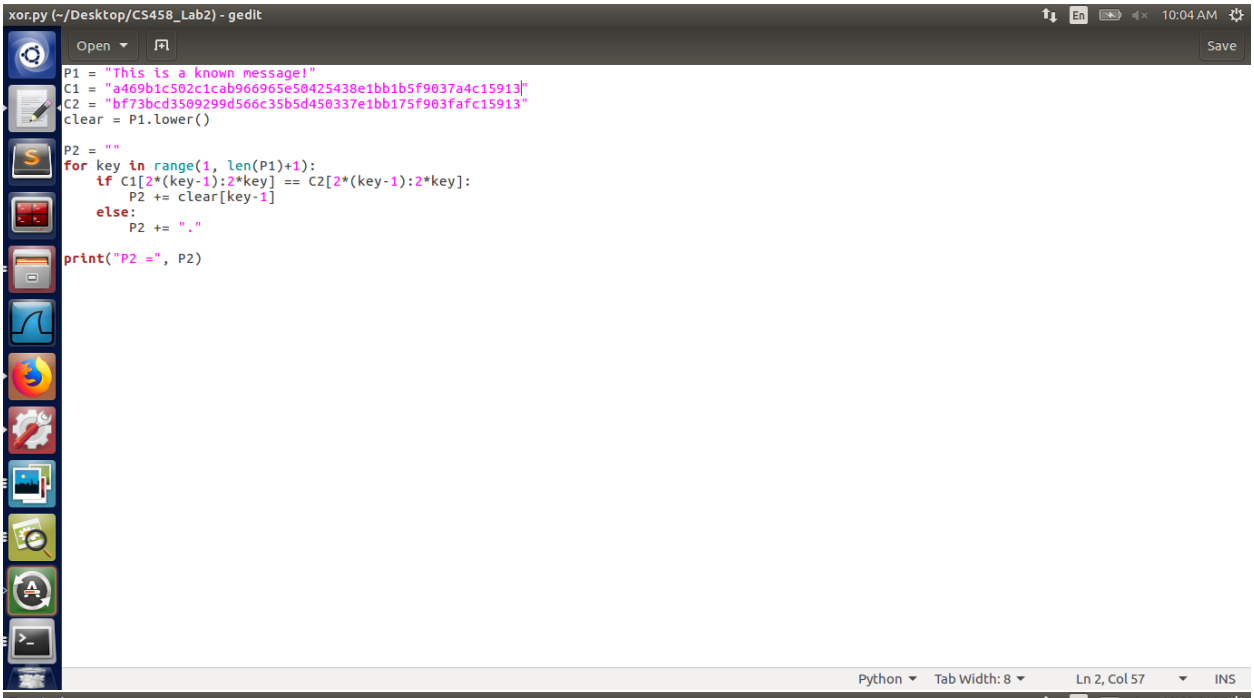
If the IV is always the same in the OFB mode and the attacker gets hold of a plaintext (P1) and a ciphertext (C1), they cannot decrypt other encrypted messages because OFB mode is a stream cipher that

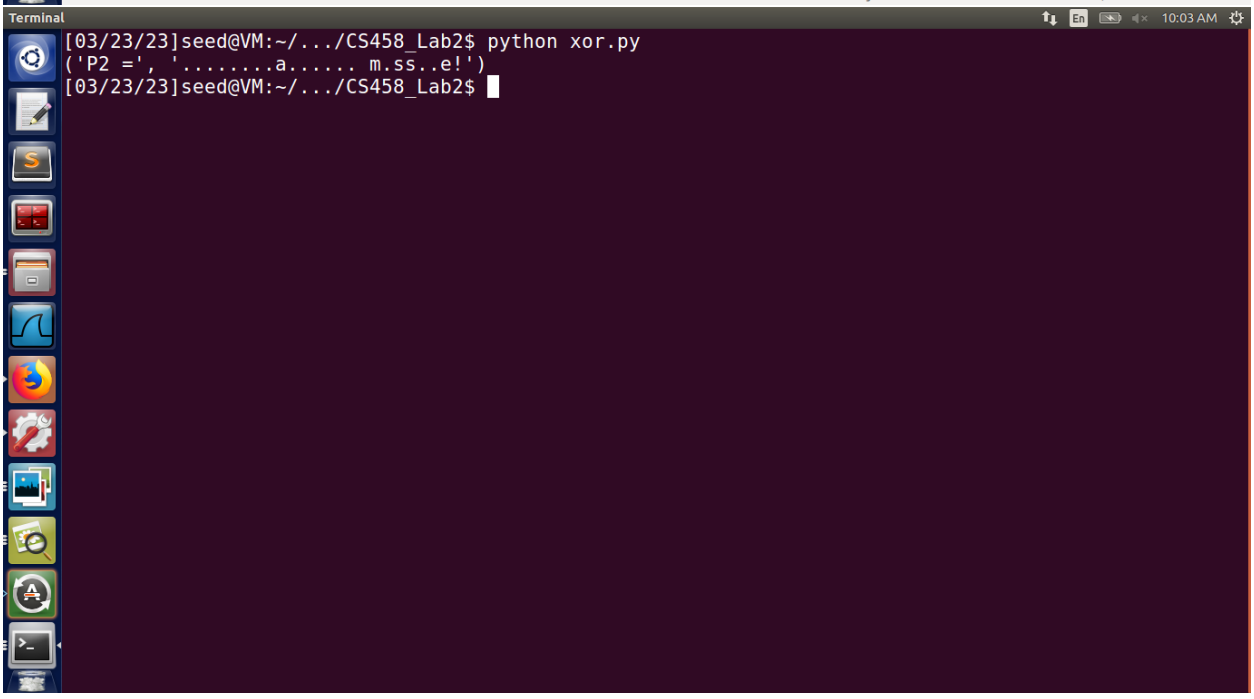
generates a key stream based on the IV and the key. If the IV is always the same, the key stream generated will also be the same for each message, which means that the same plaintext will be encrypted to the same ciphertext. However, other messages will be encrypted with a different key stream, which is not related to the IV used for encrypting P1.

Given the plaintext (P1), ciphertext (C1), and a new ciphertext (C2), we can use the OFB mode to decrypt P2 as follows:

Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722





If we replace OFB mode with CFB mode, knowing the plaintext (P1) and ciphertext (C1) will allow the attacker to decrypt only the first block of P2, because CFB mode uses the ciphertext feedback as the input for the encryption of the next block. Therefore, knowing the ciphertext for the first block of P2, which is C2, will allow the attacker to decrypt the second block, but not the subsequent blocks.

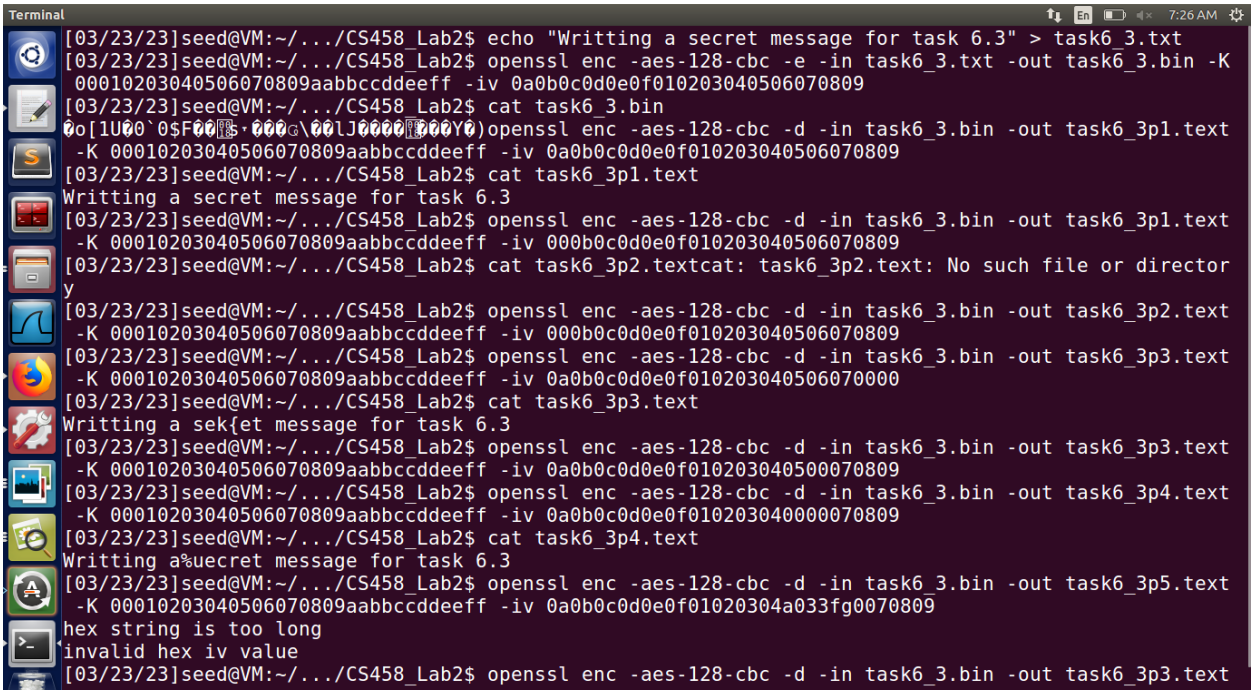
Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722

To summarize, if the attacker knows both the plaintext and ciphertext for a message encrypted using a specific encryption scheme and can use this knowledge to decrypt further messages encrypted using the same scheme, the encryption scheme is considered insecure. This is known as a known-plaintext attack.

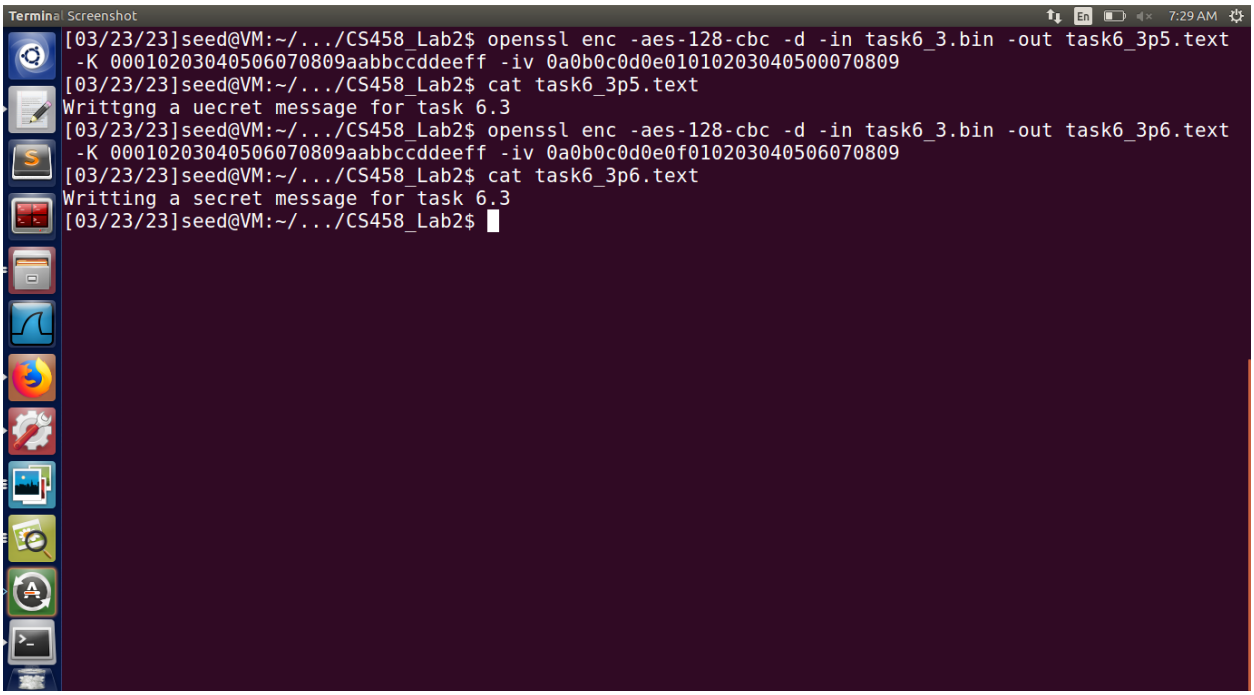
2.6.3 Task 6.3. Common Mistake: Use a Predictable IV

- Created a txt file “Writing a secret message for task 6.3”
- Encrypted the file by aes-cbc.
- And tried multiple decryptions using IV (small changed) and IV(vastly chnaged)



Introduction to Information Security - CS 458 - Spring 2022

Neha Ramesh Gawali A20523722



Eve can see the ciphertext and the IV used to encrypt the message, but since the encryption algorithm AES is quite strong, Eve has no idea what the actual content is. However, since Bob uses predictable IVs, Eve knows exactly what IV Bob is going to use next.

As we can see above, when we used predictable IV and tried decrypting ciphertext :

1. When there was a slight change in the IV, we could decrypt the message almost exactly.

2. Since IV underwent a lot of modifications, we could only decrypt a portion of the message.

3. If IV is completely altered, we won't receive genuine plaintext.

So Yes, if Bob uses predictable IV and Eve guessed it, then Eve can decrypt a message which can be very much similar to original plaintext. Or can get message from which Eve can guess the actual plaintext.