APPLIED CRYPTOGRAPHY

**Lab 4 : Secret Key Encryption**

Name: Siri N Shetty SRN: PES2UG22CS556

(use SEEDVM)

**Task 1: Frequency Analysis**

Step 1 : Generation of key from python code.

Step 2 : Generation of plaintextcustom.txt from below article.txt

There are two main types of cryptosystems: symmetric and asymmetric. In symmetric systems, the only ones known until the 1970s, the same secret key encrypts and decrypts a message. Data manipulation in symmetric systems is significantly faster than in asymmetric systems. Asymmetric systems use a "public key" to encrypt a message and a related "private key" to decrypt it. The advantage of asymmetric systems is that the public key can be freely published, allowing parties to establish secure communication without having a shared secret key. In practice, asymmetric systems are used to first exchange a secret key, and then secure communication proceeds via a more efficient symmetric system using that key.

Step 3 : Generate ciphertextcuston.txt from plaintextcustom.txt

Step 4 : Frequency Analysis on given ciphertext.txt (run freq.py)

Step 5 : Decrypted text from given ciphertext.txt

Expected Deliverables -   
i) Output Screenshot (should have SRN) for step 1



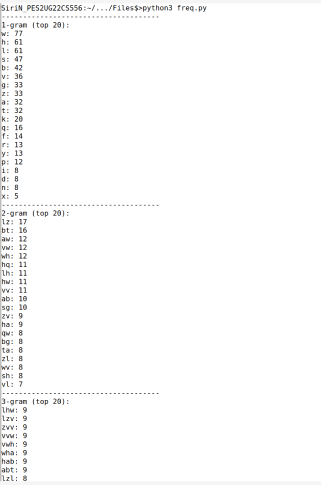
ii) Output Screenshot (should have SRN) for step 2



iii) Output Screenshot (should have SRN) for step 3



iv) Output Screenshot (should have SRN) for step 4



v) Complete Decrypted text Output (should have SRN) for step 5



**Task 2: Encryption using Different Ciphers and Modes**

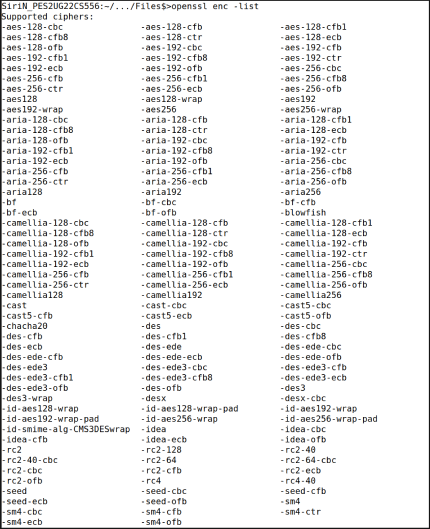
Step 1 : Various ciphers supported using “ openssl enc -list ”

Step 2 : Encrypt (-e flag) and Decrypt (-d flag) for ciphertype1

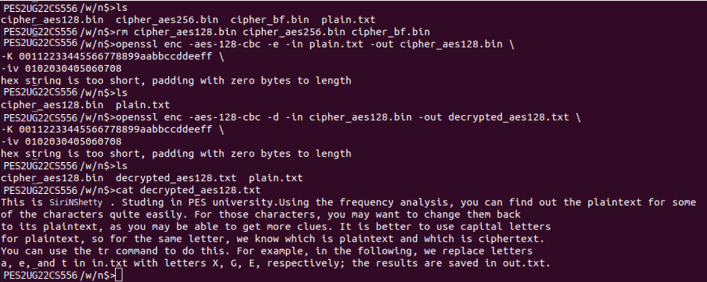
Step 3 : Encrypt (-e flag) and Decrypt (-d flag) for ciphertype2

Step 4 : Encrypt (-e flag) and Decrypt (-d flag) for ciphertype3

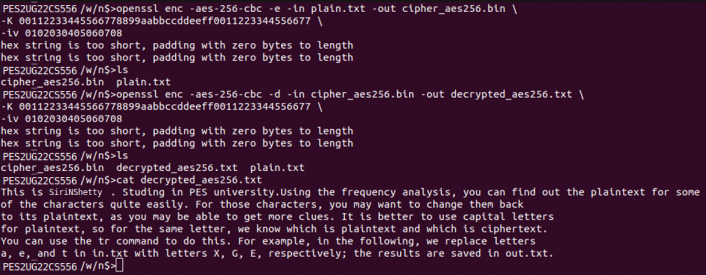
Expected Deliverables -   
i) Output Screenshot (should have SRN) for step 1



ii) Output Screenshot (should have SRN) for step 2 with Brief description of ciphertype-1



iii) Output Screenshot (should have SRN) for step 3 with Brief description of ciphertype-2



iv) Output Screenshot (should have SRN) for step 4 with Brief description of ciphertype-3



**Task 3 : Encryption Mode – ECB vs. CBC**

Step 1 : Encrypt picture using ecb

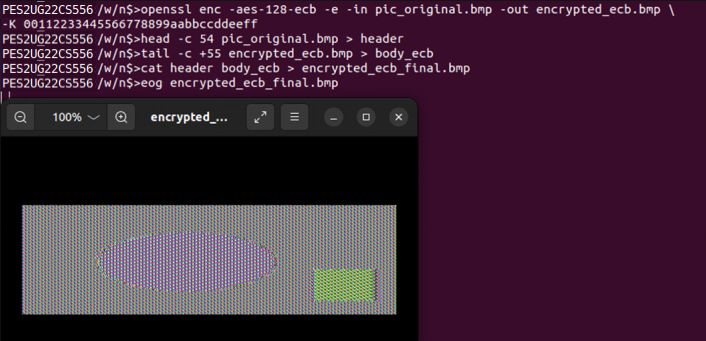
Step 2 : Encrypt picture using cbc

Step 3 : Attach headers to both the encrypted pictures  
(p1 refers to original\_pic ; p2 refers to encrypted\_pic ;   
new refers to header\_attached\_encrypted\_pic)

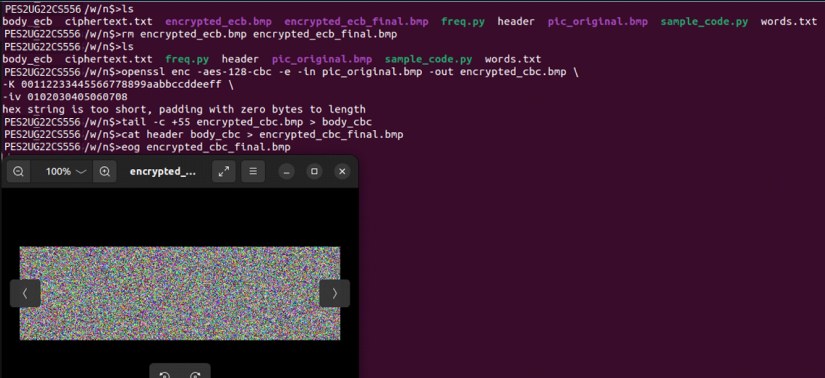
Step 4 : Display the encrypted picture using a picture viewing program

Step 5 : Repeat for custom selected picture.

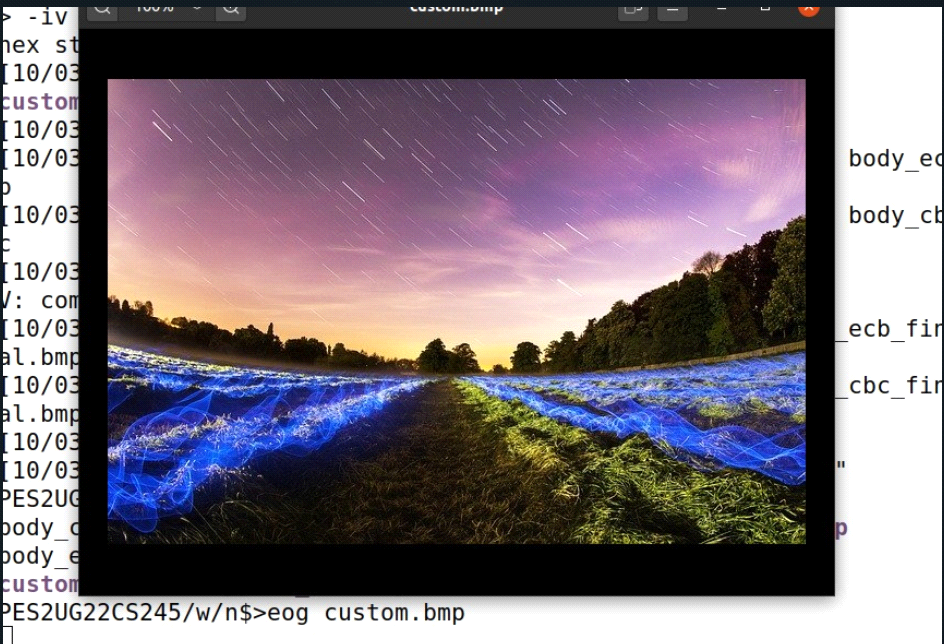
Expected Deliverables -   
i) Output Screenshot (should have SRN) for step 1



ii) Output Screenshot (should have SRN) for step 2



v) Output Screenshots for step 5 (include deliverables i to iv)







vi) Image encrypted by which mode of encryption has greater similarity to the original? Give reason. Explain your observations.

In ECB mode, identical plaintext blocks are transformed into identical ciphertext blocks, thereby exposing patterns within the data. ECB exhibits a closer resemblance to the original data. This absence of diffusion renders ECB inappropriate for images, as repeated patterns are maintained in the encrypted output.

**Task 4 : Error Propagation – Corrupted Cipher Text**

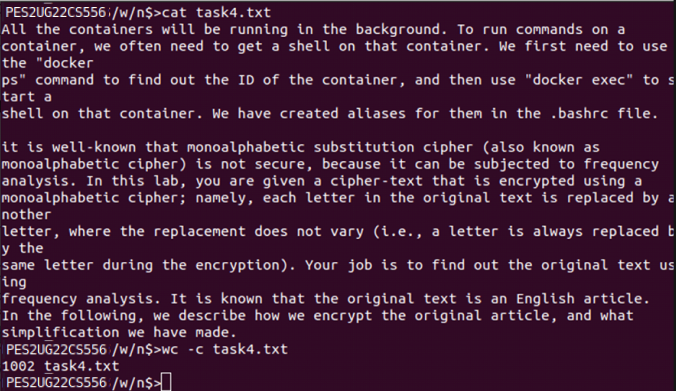
Step 1 : Create task4.txt, atleast 1000 bytes long.   
 Run “ wc -c task4.txt ” to show size.

Step 2 : Encrypt the file using the AES-128 (ECB, CBC, CFB, OFB)

Step 3 : Corrupt the 55th bit in the generated .bin using “ bless <file.bin> ”

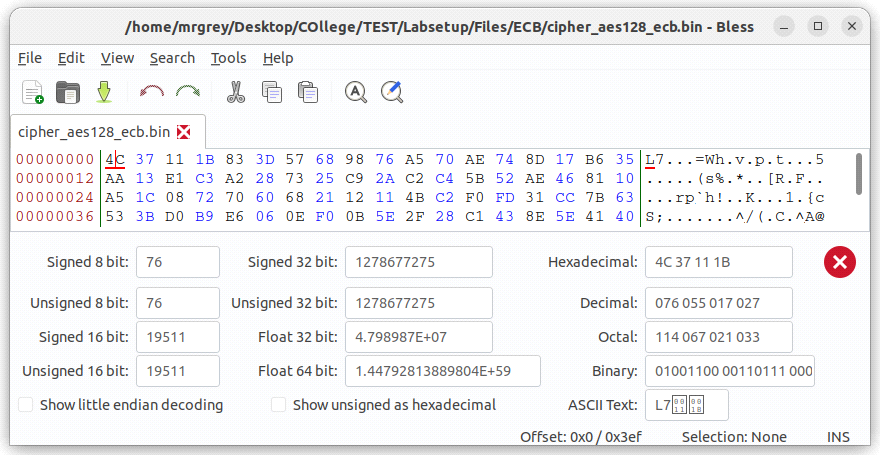
Step 4 : Decrypt all 4 modes files using openssl command.

Expected Deliverables -   
i) Output Screenshot (should have SRN) for step 1

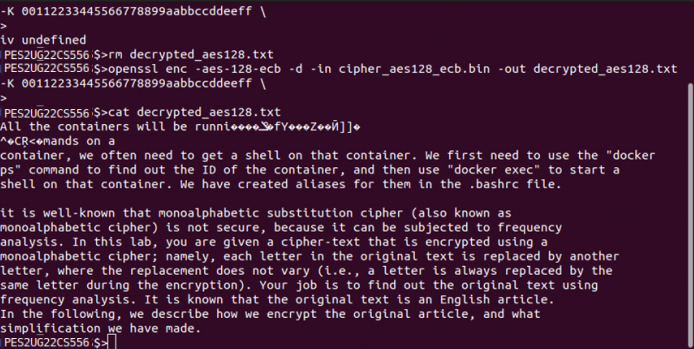


ii) Output Screenshot (should have SRN) for step 2,3,4 – ecb

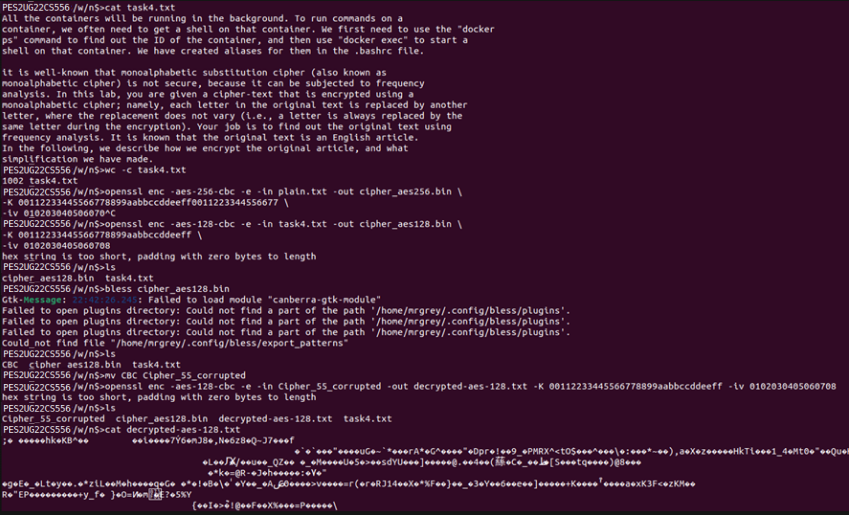
Before Corrupting 55th Bit



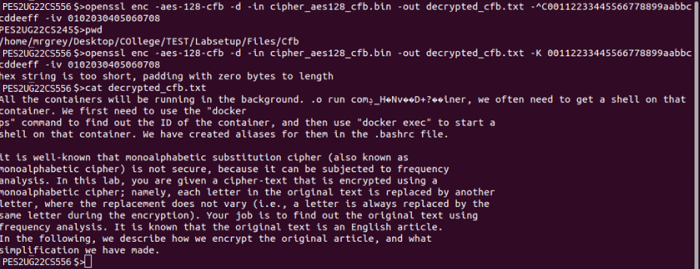
After Corrupting && Decrypting the bit we get



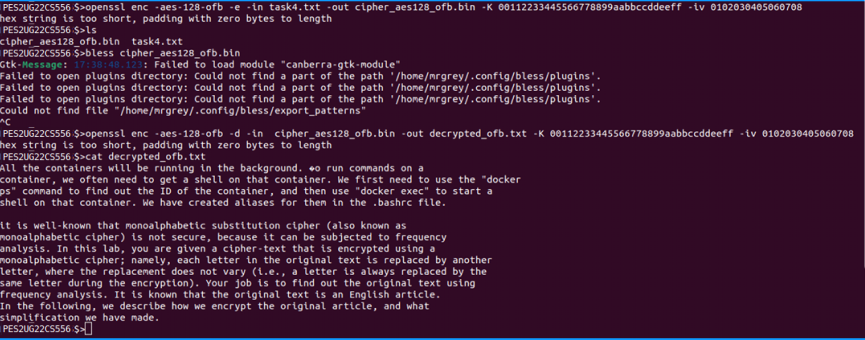
iii) Output Screenshot (should have SRN) for step 2,3,4 – cbc



iv) Output Screenshot (should have SRN) for step 2,3,4 – cfb



v) Output Screenshot (should have SRN) for step 2,3,4 - ofb



vi) In which mode is the information least recoverable? Give reason. Explain your observations.

In CBC and CFB the information is least recoverable because successive cipher blocks are dependent on the previous cipher blocks which maximally scramble the output on subtle changes of the cipher text.

**Task 5 : Initial Vector (IV) and Common Mistakes**

**Task 5.1. IV Experiment**

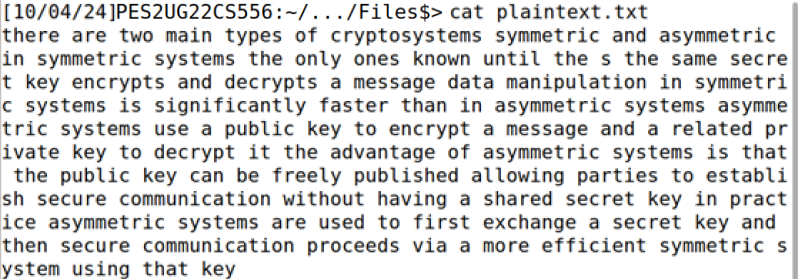
Step 1 : Create a plaintext.txt file

Step 2 : Encrypt using a key and IV.

Step 3 : Encrypt using same key and same IV as in step 2.

Step 4 : Encrypt using same key and different IV as in step 2.

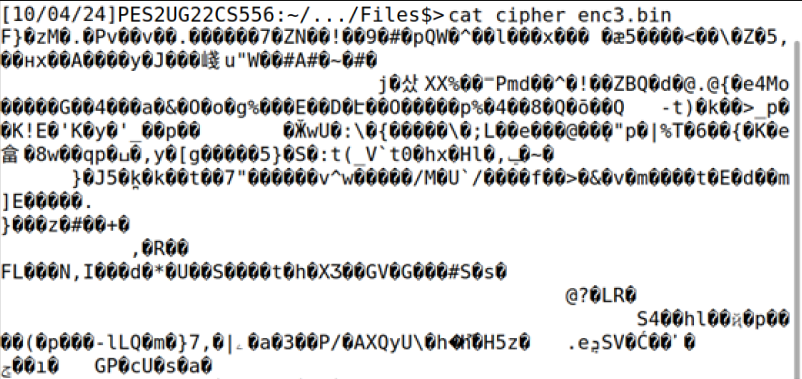
Expected Deliverables -   
i) Output Screenshot (should have SRN) for step 1



ii) Output Screenshot (should have SRN) for step 2



iii) Output Screenshot (should have SRN) for step 3

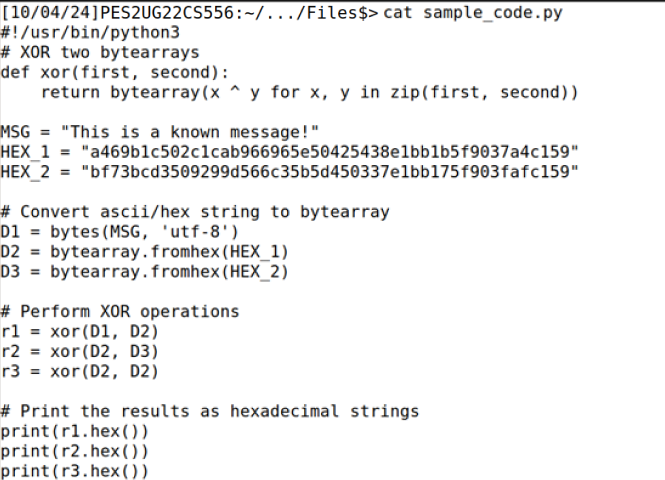


**Task 5.2. Common Mistake: Use the Same IV**

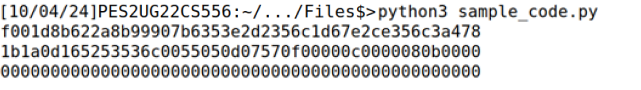
Step 1 : Modify the sample\_code.py file msg,hex1,hex2 with given values of plaintext, ciphertext1, ciphertext2.

Step 2 : Run sample\_code.py

Expected Deliverables -   
i) Output Screenshot for step 1



ii) Output Screenshot (should have SRN) for step 2



iii) Explain your observations.

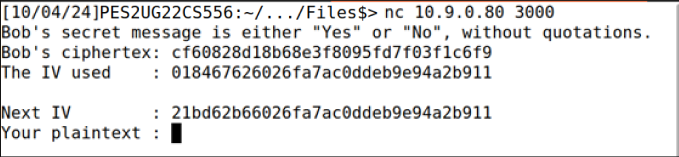
Since the IV's are same across both the cipher texts, we can xor the given plaintext with the known cipher text in-order to obtain the IV and then xor the IV with the ciphertext2 to obtain the message

**Task 5.3. Common Mistake: Use a Predictable IV**

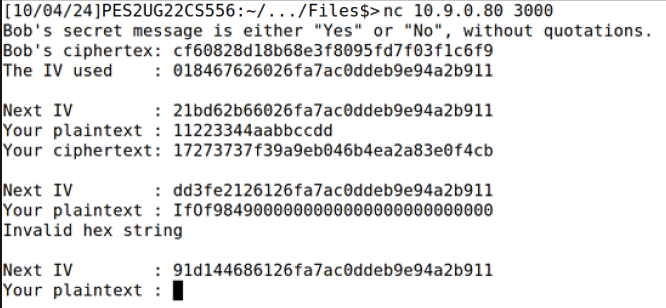
Step 1 : Run the nc cmd

Step 2 : Find the plaintext to enter (hint use sample\_code.py as reference to construct script for this ; (plaintext is generated from the message where message can be ‘yes’/’no’))  
(to know plaintext entered is right match , check that the ciphertext generated from it should be same as bob’s ciphertext given.

Expected Deliverables -   
i) Output Screenshot (should have SRN) for step 1



ii) Output Screenshot (should have SRN) for step 2



iii) What is Bob’s message ? (yes/no)

YES

iv) Explain your observations for deciphering above.

Given the initialization vector (kIV), plaintext (kP), and ciphertext (kC), we know that the ciphertext was generated using AES:

C=AES(kIV⊕kP)C = AES(kIV \oplus kP)C=AES(kIV⊕kP)

Now, when we have a specific IV (mIV) for encrypting a plaintext (mP), the ciphertext is given by:

mC=AES(mIV⊕mP)mC = AES(mIV \oplus mP)mC=AES(mIV⊕mP)

If we select mPmPmP as mP=kIV⊕mIV⊕mPmP = kIV \oplus mIV \oplus mPmP=kIV⊕mIV⊕mP, then mCmCmC simplifies to:

mC=AES(kIV⊕mP)mC = AES(kIV \oplus mP)mC=AES(kIV⊕mP)

Thus, mCmCmC will equal kCkCkC if and only if mPmPmP matches kPkPkP, enabling us to successfully decipher the plaintext.

**Overall Submission in**  SRN\_Lab4\_AC : pdf