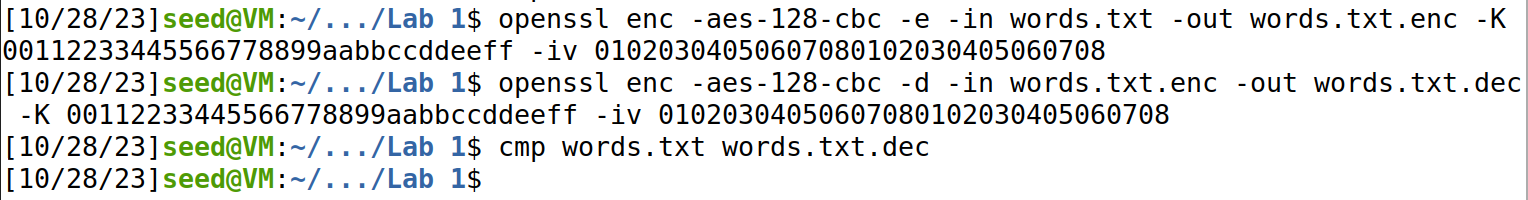
Secret Key Encryption (CS 915) Post-Lab Assignment Report

Hanzhi Zhang - 5525549

Task 1: 1) Briefly explain the results of this task with screenshots.



Decrypting the ciphertext can recover the same plaintext.

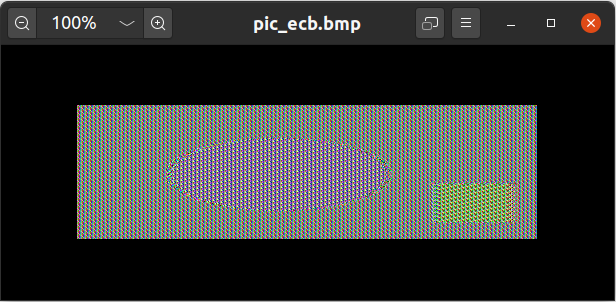
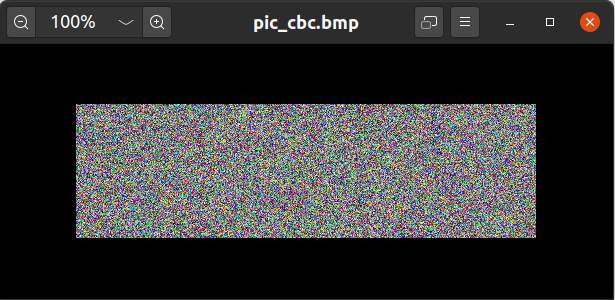
2) When you encrypt data using a password, you observe a warning “deprecated key derivation used”. Explain why you have such a warning.

In this version of OpenSSL using a key derivation method without PBKDF2 algorithm is considered “deprecated” (too weak or not secure enough), it suggests that we should use PBKDF2 algorithm to derive the key for higher security (and maybe also to specify a given number of iterations on the password, according to the manual page for ***enc***, a larger iteration count increases the time required to brute-force the file).

3) Explain what is -pbkdf2 and why it is needed.

PBKDF2 is short for Password-Based Key Derivation Function 2, it generally applies a pseudorandom function to an input password along with a salt value, then iterates to derive a cryptographic key. Passwords, if directly used to encrypt data, are vulnerable to brute force or dictionary attacks, etc. PBKDF2 can perform key stretching, helping strengthen passwords and making them more resistant to such attacks.

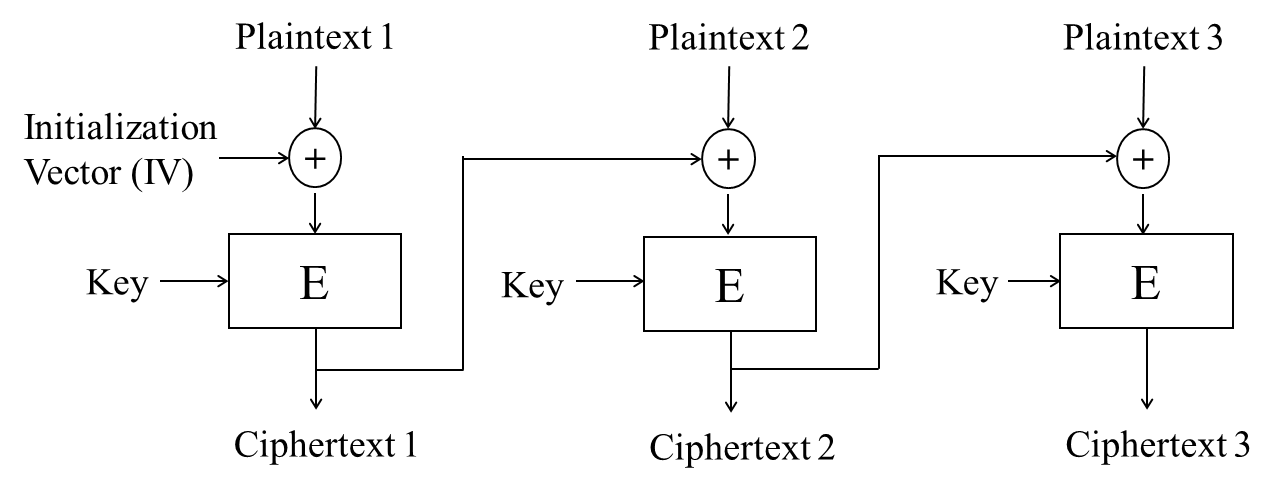
Task 2: 1) Briefly explain the results of this task with screenshots.



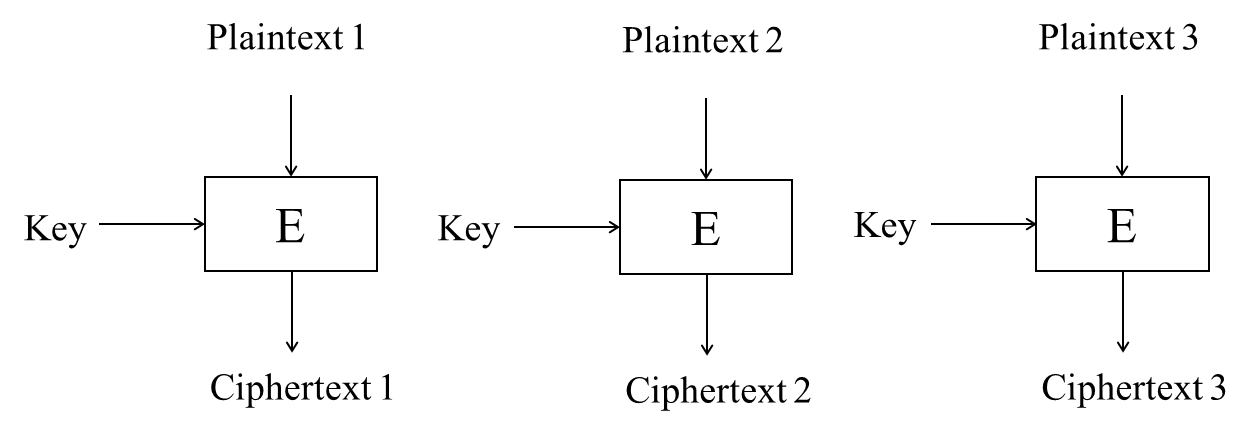
The output using AES-128-CBC looks “better” encrypted, for we cannot see any pattern from the picture, the colour of pixels seems randomly and evenly distributed, but in the one generated by AES-128-ECB, we can observe the shapes of an oval and a rectangle as in the original picture, which is not a very satisfying result.

2) You encrypt pic\_original.bmp in two methods: AES-128-CBC and AES-128-ECB. Use diagrams to explain how these two modes work.

AES-128-CBC: For block No. 1 we have Ciphertext 1 = E(k, IV ⨁ Plaintext 1), for the rest of the blocks, Ciphertext i+1 = E(k, Ciphertext i ⨁ Plaintext i+1)



AES-128-ECB: For all blocks there is Ciphertext i = E(k, Plaintext i)

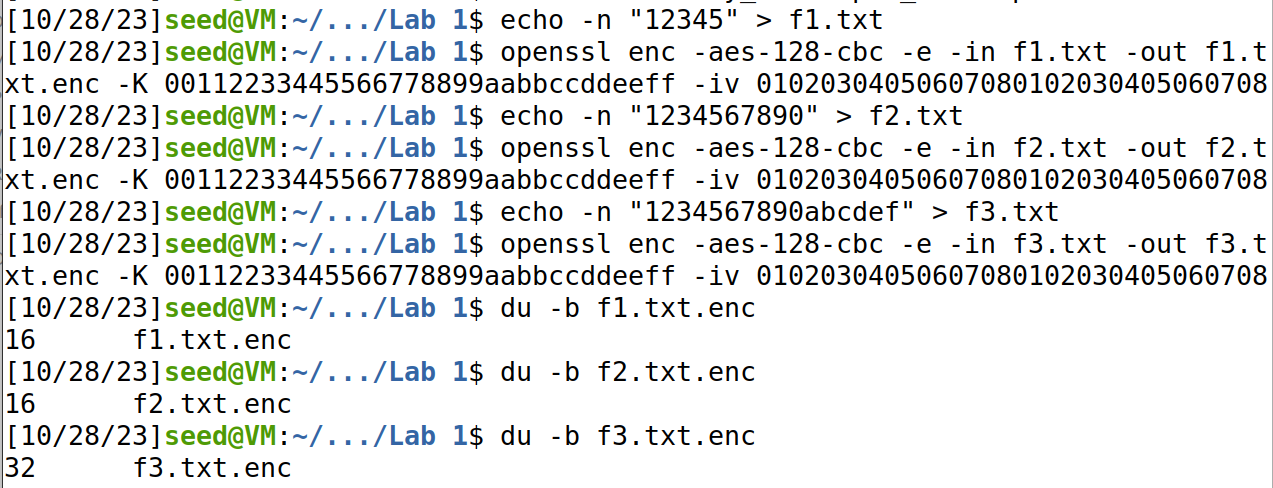


3) Describe any difference in the output between the two methods and explain why.

With ECB, the confusion and diffusion are only done within each block, but not across different blocks, it is a deterministic operation so that same plaintext-block input always produce the same output. If two small areas have the same (very similar) colour in the original picture, they will also look similar in the encrypted picture, such correlations can leak information, for example the shapes or the outline of the pattern.

But with CBC, the output of a block is influenced either by a random IV or the output of the former block, so same plaintext-block will produce different outputs. Confusion and diffusion are done across blocks, the output of the entire file will be more randomly distributed, and a lot more difficult to spot any pattern from the original picture.

Task 3: 1) Briefly explain the results of this task with screenshots.

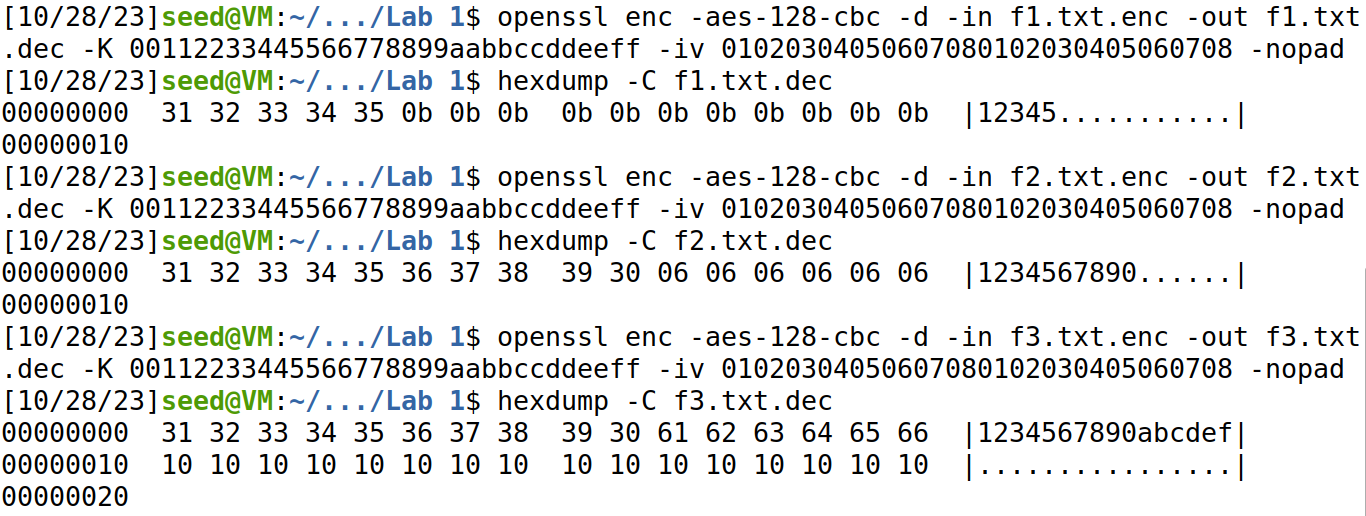


The size of encrypted f1.txt (5 bytes) is 16 bytes, the size of encrypted f2.txt (10 bytes) is 16 bytes, and for f3.txt (16 bytes) the encrypted file is 32 bytes.

2) You encrypt the three files using 128-bit AES with CBC mode. Describe the sizes of the encrypted files and explain why you have such sizes.

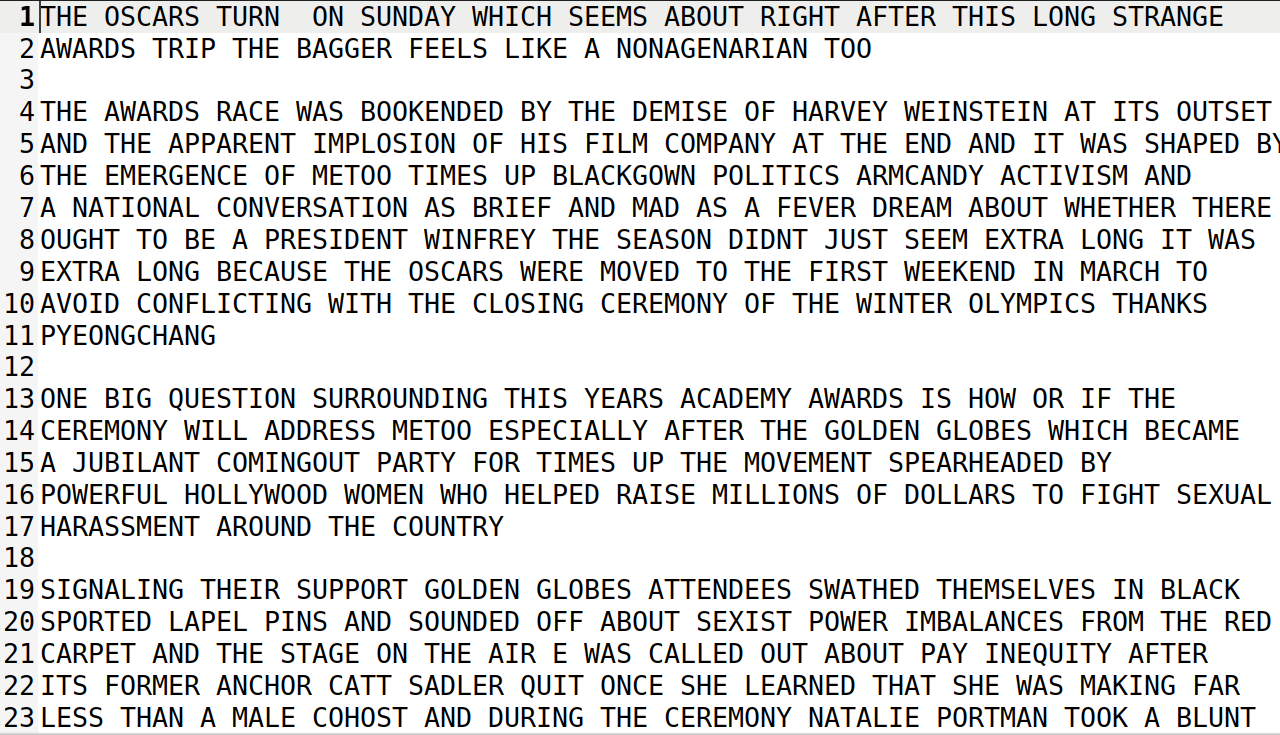
The block size of 128-bit AES, that is 16 bytes. For the files respectively containing 5 and 10 bytes, both smaller than a block, so that padding is added at the end to make up to the exact size of the block. As for the file with 16 bytes, the same as a block, an entire block of padding is added to ensure unambiguous distinction of data and padding.

3) Describe what padding has been used in the encryption and explain how you verify that through the decryption process.



We can see from the files decrypted using the "-nopad" option, that f1.txt was padded with "0b" \* 11, f2.txt with "06" \* 6, and f3.txt with "10" \* 16, so the rule or pattern used for padding here is to count how many bytes of padding is needed and use that number as the “pad”, so that 0bhex = 16 – 5, 06hex = 16 – 10, and 10hex = 16.

Task 4: 1) Briefly explain the results of this task with screenshots.



The ciphertext is decrypted using the substitution cipher and reads fine.

2) Write down the substitution letters in the key (the top row being the plaintext letters) and explain how you have obtained them.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |
| V | G | A | P | N | B | R | T | M | O | S | I | C | U | X | E | J | H | Q | Y | Z | F | L | K | D | W |

Look at each letter separately first, the top three in ciphertext are N (12.41%), Y(9.49%) and V(8.85%), quite distinguishable from each other and corresponding to top three in English texts, E(12.7%), T(9.1%) and A(8.2%), so E --> N, T --> Y and A --> V.

Then we look at bigrams, top two YT(2.7%), TN(2.24%) in ciphertext may correspond to TH(3.56%) and HE(3.07%) in English text, so we guess H --> T, we can also guess NH and HN correspond to ER and RE, R --> H. Also recall in separate letters we would roughly guess XUQM in ciphertext map to OINS in plaintext, now MU ranks 3rd place in bigram frequency for ciphertext, while IN for English comes in the same rank, so we decide that I --> M and N --> U (can also be supported by the pair VU and AN).

Then we look for plaintext ON and OR in ciphertext, which should be \_U and \_H, we find XU and XH, so O --> X, for ES (N\_) we find NQ, S --> Q (which is also the only letter left in OINS ~~> XUQM, so it very likely would make sense).

For bigram ciphertext pair NP, UP or plaintext E\_, N\_, we find ED and ND, considering P(3.97%, No. 11) and D(4.3%, No. 10) we might guess D --> P for now.

Now for the trigram, search for plaintext ING, ciphertext MU\_, found MUR, G --> R; search for plaintext FOR(0.34%), ciphertext \_XH , found BXH(0.31%), F --> B.

Continue looking for plaintext OU in bigram, X\_ found XZ, compare Z(2.42%, No. 14) and U(2.8%, No. 13), U --> Z. For AL and LE, V\_ and \_N found VI and IN, I(4.22%, No. 10) and L(4.0%, No. 11). For CO, CE and \_X, \_N found CX, CN, C --> C. For VE (\_N) and ME (\_N) found GN FN, match G(2.11%, No. 16) to M(2.4%, No. 14) and F(1.25%, No. 21) to V(0.98%, No. 21). These are all just guesses!

The rest are done by looking at the partial decrypted file and manually filling in for the remaining letters and making adjustments until the article reads fluently. For example, when we see “SUNDAd”, “Md”, we know we shall replace “d” with “Y”, and when we see “HOLLYlOOD”, “lILL”, “lITH”, we know “l” should be substituted by “W”, and for “jUESTION”, “jUIT”, “INEjUITY”, “j” is then replaced with “Q”, etc.