Note: For this assignment, feel free to use a programming language of your choice. Python is recommended. For Python, the packages PyCryptodome (https://pypi.org/project/pycryptodome/) or PyCryptodomex (https://pypi.org/project/pycryptodomex/) are recommended.

Submission requirements: Your submission must contain:

• The source code, the executable(s) if any, and all other related files.

• The project report, which will contain the instructions for running your program, and the screenshots demonstrating that your program works according to the assignment tasks.

Tasks:

1. Create a text file "pt1.txt" with the following text "This is the test message 1".
   Create a copy of this file with name "pt2.txt".
   Generate a random 128-bit key "k".
   Encrypt the contents of "pt1.txt" using the AES-ECB mode and the key "k", and save the resulting ciphertext to a file "ct1.txt". Before saving, encode the ciphertext using Base64 (so that it can be conveniently printed on the screen).
   Encrypt the contents of "pt2.txt" in the same way as described above, on the same key "k", and save the contents to a file "ct2.txt".
   Print the contents of the files "ct1.txt" and "ct2.txt" to the screen (to make sure that the resulting ciphertexts are the same).

2. Repeat Task 1 but use the AES-CBC mode with randomized IV. Use the same key "k". Print the contents of the files "ct1.txt" and "ct2.txt" to the screen (to make sure that the resulting ciphertexts are different).

3. Generate a random 256-bit key "k1". Compute an HMAC authentication tag for "p1.txt" using "k1", and save it to a file "tag1.txt", using Base64 encoding.

Change the contents of "p1.txt" to "This is the test message 2".

Compute an HMAC authentication tag for the updated "p1.txt" using "k1", and save it to a file "tag2.txt", using Base64 encoding. Print the contents of the files "tag1.txt" and "tag2.txt" to the screen (to make sure that the resulting authentication tags are different).

4. Generate a random 256-bit key "k2". Encrypt the contents of "pt1.txt" using the AES-GCM-256 mode and the key "k2", and save the resulting ciphertext to a file "ct3.txt", using Base64 encoding. Decrypt the contents of "ct3.txt", and print the result to the screen to make sure that the decryption has been performed correctly. Change the first symbol in the file "ct3.txt" to a different alphanumerical character (to be consistent with Base64 encoding). Attempt to decrypt the updated ciphertext "ct3.txt" and display the result (to make sure that the ciphertext is rejected).