Exercise 4: Making Communication faster between parties

Now that you deterimined that AES is lot faster. Update the encryption and decryption program of Exercise-2 to make it faster.

Encrypt function:
Input: a random symmetric key K, Message M and public key of Agent X
Program: encrypt symmetric key with RSA (call is cipher C1) and encrypt message with symmetric key and a random IV (Call it cipher C2).
Output: C1, C2, IV.

Decrypt function:
Input C1 and C2, IV; private key of Agent X
Program: decrypt the symmetric key with RSA (K) and use K and IV to decrypt message (M).
Output: K and M

Instructions for the programming task -

- The tar contains 2 directories named test_decrypt and test_encrypt to test your decrypt and encrypt function respectively.
- Write your hybrid_encrypt function in test_encrypt/hybrid_test_encrypt.c
- Write your hybrid_decrypt function in test_decrypt/hybrid_test_decrypt.c
- Main function in rsa.c takes 3 command line arguments -
  - arg1: message_file (.txt file)
  - arg2: private_key_file (.pem file)
  - arg3: public_key_file (.pem file)
  - arg4: symmetric_key_file
- Main function reads the message file and calls hybrid_encrypt function to encrypt the message according to the scheme and then calls hybrid_decrypt function to decrypt the encrypted message
- Your task is to write these two functions (hybrid_encrypt and hybrid_decrypt) in two separate files (hybrid_encrypt.c and hybrid_decrypt.c respectively)
- Specification of the functions -
  - int hybrid_encrypt(char* symmetric_key_file, char* message_file, char* public_key_file)
    where,
    - symmetric_key_file - file name of symmetric key (for eg. “symmetric.pem”)
    - message_file - message to be encrypted
    - public_key_file - file name of public key (for eg. “public.pem”)
    - Return type is int, (1 if successful, else 0)

    The function should implement following things -
    - Use the mentioned scheme to calculate C1,C2 and generate random IV (16 bytes)
    - Write C1, C2, IV in files named C1, C2 and IV respectively

  - int hybrid_decrypt(char* private_key_file, char* decrypted_file)
    where,
    - private_key_file - file name of public key (for eg. “public.pem”)
    - decrypted_file - Write the decrypted message in this file
    - Read C1, C2 and IV from the files (named C1, C2 and IV respectively)
    - Assume all the above files are present in the same directory
    - Return type is int, (1 if successful, else 0)

At this point, Agent X has a question (he learnt the lessons the hard way in symmetric key). When using RSA as above would identical messages leak information i.e. accidentally if an agent chose the same symmetric key again would an enemy know this, he may not know the key value but know it has been reused?. Essentially he wants a robust code from you.

Guidance:

1. Assume that the message size (number of characters in message.txt) is small.(less than 64 characters).
2. To execute your encrypt and decrypt function, go to the respective directory and execute following commands :
   1. make
   2. make test
3. To handle Agent X's concern, run your code twice, but in both cases, provide the same input i.e. same symmetric key and message as well. Note message is encrypted via symmetric key algorithm, but test this robustness as well once again. Neither C1 nor C2 should leak info. If your code is leaking info, go back and fix it.
4. Use RSA_PKCS1_OAEP_PADDING as the padding argument in RSA_public_encypt(..) and RSA_private_decrypt(..) functions.
5. The file names should be exactly the same as described in the problem statement.

NOTE : please refer to reference.txt for help in coding assignment.