1. Asymmetric encryption exercises

Note for both exercises in this part:
When coding, it is recommended that you generate the private-public key pair once, save the generated public-private key pairs to file, and you read the keys from the file in following tries. The reason is it takes some time for the algorithm to find large primes. Generating a large prime once and saving the key for later use can help you save some time in debugging.
Also, you need to generate different private-public keys for different algorithms, and for different users (users are who hold the keys).
Coding exercises use the file week5.py, and ElGamalsupport.py. You should download both file to a folder you can run python scripts. You do not need to change anything in ElGamalsupport.py. You should write code in week5.py.

1.1. RSA

Try to use the RSA functions implemented in PyCryptodome with the correct API.

What you should do / TASKS

1. Read the following materials: RSA key, example code of encryption
   After you read these materials, you should be able to follow the #TODO instructions in the provided code. The examples show you what API to use to complete each task.
2. Write code to complete the following tasks:
   a. Alice wants to send Bob a message in secret (confidentiality). Use public key encryption to do so, and explain how the goal is achieved.
   b. Alice wants to send Bob a message with data origin authentication. Use public key encryption to do so, and explain how the goal is achieved.
   c. Compare the computational costs (time used for encryption and decryption) between the solution you have done in week 4 workshop, and these two solutions.
   d. Compare the security properties you achieve.

1.2. ElGamal

PyCryptodome has an ElGamal implementation; however, it is redacted at the moment. However, for understanding the ElGamal cryptosystem, we will implement the encryption and decryption algorithms by ourselves.
It is still useful to look through the ElGamal code implemented in PyCryptodome. We still use the key generation function as is provided in it. It may be helpful for you to finish your encryption and decryption code too. The encryption and decryption code is still written in the ElGamal code; the current version has only closed the API to use them.

What you should do / TASKS

1. Look at the ElGamal code; understand the key generation, encryption and decryption functions.
2. Look at the helper file, ElGamalsupport.py, provided in the course github repository. It contains function you need to support ElGamal encryption and decryption. Understand how to use them.
3. Write code to perform encryption and decryption. Follow the #TODO instructions in the provided code file.

2. Symmetric key establishment using asymmetric cryptography

We are going to establish a new shared secret, a symmetric key for a new session, for two entities, A and B. We will accomplish it in two ways: Diffie-Hellman Key Exchange, and key encapsulation.

a. Diffie-Hellman Key Exchange (DHKE)
   DHKE allows exchanging key materials in a public channel, and establish a secret using the publicly accessible key materials. Review the DHKE process (from slides and from the Stallings book) and code the process to share the secret.

b. Key encapsulation
   In key encapsulation, a new symmetric key is generated by one of the two key establishing entities. Let's name it as A. A generates the new symmetric key, and sends it to the other entity, B, with confidentiality protection provided by public-key encryption. Complete the steps of key encapsulation in the code.

c. Key confirmation
   Design a way to verify the key has been established between the two entities, i.e., both A and B have the same key.

What you should do / TASKS

1. Review the Diffie-Hellman Key Exchange steps.
2. Write code to implement the steps. Follow the #TODO instructions in the code file.
   - You can start with two small p and g, to test that the code works correctly
   - Then you use a large p and g to get a secure key
3. Write the code to use key encapsulation to establish a symmetric key. In the code file, the public-key encryption is based on RSA. Follow the #TODO instructions to complete the process.
4. Design a way for key confirmation.
   - First, describe the ways you can do the verification, without coding.

- Second, try coding it with functions available to you.