In this project, students are expected to implement Diffie-Hellman key exchange protocol. The programming language can be picked by each student according to his/her preference. For simple handling of large numbers Python is recommended (not necessary).

- Employing cryptographic random generators from the built-in libraries such as 'secrets' library of the python is recommended.
- Use key size of 1024 bits.
- You can generate the group first and save it so that you do not generate each try.
- You will run 2 separate instance of your code to simulate a server like communication.

1 Diffie-Hellman protocol

In this scope of this project, a group of prime order with generator g is agreed before protocol execution.

Alice	Bob
Picks secret a at random	Picks secret b at random
Sends public g^a to Bob	Sends public g^b to Alice
Computes $g^{ab} = (g^a)^b$	Computes $g^{ab} = (g^b)^a$
Uses $H(g^{ab})$ as private key	Uses $H(g^{ab})$ as private key

Table 1: Key exchange protocol between Alice and Bob.

2 Deliverables

Students are expected to deliver following task in order.

2.1 Generating secrets (30 pts)

Program Takes as input the user name and generates the Diffie-Hellman secret (a for Alice, and b for Bob)

2.2 Communication (35 pts)

Communicates with another instance of the program via a file "Communication.txt" inside your computer storage. Everything that will be written to the file will be as ASCII characters. And after each message, apply padding so that the end of message will be understood (size of the padding does not matter). After any command given by the user, the program writes to that file its Diffie-Hellman public (g^a for Alice).

If there already exists a written Diffie-Hellman public in that file (different than it has written to that file itself), the program reads it. Otherwise, it keeps re-reading the file until it finds one with 10 seconds intervals. Then both sides compute the private key ($H(g^{ab})$ in Alice and Bob's case). For hash function, you may use any built-in SHA-256 algorithm in existing libraries.

2.3 Encrypted Communications (35 pts)

When both programs get the private keys. The encrypted communication phase start. Namely, each side sends the message by encrypting it with the private key they computed in the previous step. For encryption algorithm use AES-128-CTR. Once again you can use the existing libraries for the AES. If CTR is not available you can use any other mode. The phase continues as one message by each side basis, i.e., after sending a message the other side will send one. This way only after sending a message the program just need to screen the file for other side's reply with 10 seconds (tentative) intervals. The first message will be sent by the first one that writes her Diffie-Hellman public into the file. For the next part, you need to save the communications for Alice and Bob in a a separate and respective text files.

2.4 Man-in-the-middle (BONUS) (30 pts)

Write an man-in-the-middle attacker's code that takes as input two different file names (for example, alice.txt, bob.txt) for communicating with Alice and Bob. In this setting, Alice and Bob will run as before but the communication files will not be the same for them, they will be the ones given to the attacker. The attacker will trick both sides for generating private keys only with the attacker. After Alice or Bob tries to send a message attacker will print it on the screen and ask what to send to other side. This way user of the attacker program will always see what is being sent and can change it when conveying to the other side.

3 Helpful links

Practical implementation of Diffie-Hellman