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 taste. For simple handling of large numbers Python is recommended (not necessary).

Simple reminder of the protocol:

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

Communication: Write a program that does the following:

1. (25 pt) Takes as input the user name and generates the Diffie-Hellman secret (a for Alice).

2. (25 pt) 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 there exist the string “000000000000000”, i.e. fifteen 0’s so that the end of message will be understood. With initiation command “init”, 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 SHA-256 algorithm. There already exists implementations available on the internet for many programming languages, just use them.

3. (25 pt) 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. There already exists implementations available on the internet for many programming languages, just use them. 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 intervals. The first message will be sent by the first one that writes her Diffie-Hellman public into the file.

4. (25 pt) **Man-in-the-middle:** Write an man-in-the-middle attacker’s code that takes as input two different file names 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.