Computer Engineering 4DN4
Laboratory 2
Online Grade Retrieval Application

The objective of this lab is to develop a client/server network application that can be used by a course instructor to distribute grades. The code will be written in Python 3 using the Berkeley/POSIX socket API discussed in class. The server software would be run by the course instructor and is given access to a database of student grade records. The client software would be run by a student and communicates with the server in order to retrieve this information. A client can also issue commands that will return grade averages for particular grade categories. The server encrypts the data sent so that only the requesting client can decrypt the server response. The working system has to be demonstrated by each group in one of the online laboratory sign-up sessions.

1 Description

The software to be developed consists of separate server and client applications written using Python 3. The working software must be demonstrated by each group in one of the online laboratory sign-up sessions. In the Marking Scheme section there is a detailed description of output that the software must generate when doing the demonstration.

In order to perform the encryption/decryption you will need to install the Python cryptography module. You should be able to do this by going to a terminal or PowerShell window and typing `pip3 install cryptography` (or maybe pip instead of pip3). Try running the script `Howto_encrypt_and_decrypt.py` to see if this is working.

The required functionality is described as follows.

1.1 Server

The server code is run from the command line on a server host and awaits TCP connections from clients. If the right conditions are satisfied, the server responds with student grade information, which is then displayed on the client console. The details are as follows.

1. A mark database is provided to the server in the form of a CSV (comma separated values) file. A CSV file can be exported from a standard Microsoft Excel spreadsheet file by using the `File` menu and selecting `Save As > CSV`. An example of a CSV file named

   `course_grades_2023.csv`

   is posted with the lab including the Excel spreadsheet, `course_grades_2023.xlsx`, from which it was obtained.

2. When the server is started, it reads in the CSV file and listens for client TCP connections. An example of reading and writing a CSV file was given in the "Python review files" code posted on the course web site during the first week of class (see `company.py`). An example of part of the CSV file considered is shown below. The first row in the file defines the meaning of the columns. Note that the second and third entries in each row are the student's ID number and the encryption key to use for sending messages to that student. The key for a student is a secret that is shared between the student and the server. The rest of the columns are the grades obtained by each student.

```
Name,ID Number,Key,Lab 1,Lab 2,Lab 3,Lab 4,Midterm,Exam 1,Exam 2,Exam 3,Exam 4
Kacie Stephenson,1803933,M7E8erO15CIh902P8DQsHxKbOADTgEPGHdiY0MplTuY=,3,9,9,0,7,4,5,8,10
Yassin Jordan,1884159,PWMKkdXW4VJ3pXBpr9UwjefmlIxYwPzk11Aw9TQ2wZQ=,9,2,10,3,8,3,9,5,7
Lowri Mathews,1853847,UVpoR9emIZDrpQ6pCLYopzE2Qm8bCrVyGEzdOOo2wXw=,9,0,0,2,17,6,10,7,4
Tiya Sheridan,1810192,bHdhydsHzwKdb0RF4wG72yGm2a2L-CNzDl7vaWOu9KA=,10,1,0,6,15,8,7,6,6
```

3. When a client connection occurs, the server reads what has been sent from the client. The request from the client always consists of the student ID number (i.e., 7 bytes) followed by one of the following commands: i.e., GMA, GL1A, GL2A, GL3A, GL4A, GEA and GG. GMA/GEA are "get midterm/exam average" and the others are "get lab average" commands, e.g., GL3A is "get lab 3 average", and so on. The GG command means to "get grades" for the requesting student rather than class averages. When one of the commands is received, the server returns the requested data, where it is displayed on the client command line. Any time that the server responds to a client request, it will encrypt the message using the student's encryption key that was read from the CSV file. Please see the file:

   `Howto_encrypt_and_decrypt.py`

   for a simple example of using the keys for encryption and decryption.

1.2 Client

The client code is run on a client host and creates a TCP connection(s) to access the server mark database. The details are as follows.

1. The client runs as a command line application and first prompts the user for an input command. If the user enters one of the commands, then the client sends the (unencrypted) command through a TCP connection to the server. As discussed above, all client requests consist of the student's 7 byte ID number followed by the command. The encrypted response from the server is then decrypted and printed on the command line.

2. After the printout, the client should close the connection and return to the command prompt state. At that point, the client should be able to reconnect to the server without restarting either application.

2 Requirements

Groups: You can work in a group of up to 4 students.

Demonstration: Please read the following instructions carefully.

1. The working system must be demonstrated on one of the group's computers, i.e., the server and client can be run on the same PC or laptop. The demonstrations will be done online.
2. All group members must be in attendance for the demonstration.
3. Form your group (maximum of 4) as soon as possible and join one of the Lab 2 groups on Avenue to Learn. This will be used for your report submission.
4. Each group then needs to reserve a 20 minute demo time slot in one of the Lab 2 sessions. Time slots will be assigned on a first-come-first-served basis using the Doodle scheduler. Make sure you are certain of the group's schedule before you book a time slot. The deadline for doing this is Sunday, February 19. Read the following document about doing the demo registration:

   `COMPENG4DN4_Lab_2_demo_signup.pdf`

Marking Scheme: The assigned mark consists of two parts, i.e., 80% for demonstrating the system and a 20% discretionary component for questions answered, based on your Python code, during the demonstration. The demonstration consists of the following, where each step below is weighted equally in the demonstration component of the mark.

1. Start the server in a shell window. The server prints out the data read from the CSV file, e.g., "Data read from CSV file: ", followed by each row of the file. The server must print output indicating that it is then listening on the host computer for incoming connections on a particular TCP port, e.g., "Listening for connections on port `<port number>`."
2. Start the client in a second window. The client will prompt the student for their ID number and then a command. When a command is entered, the client echos what has been entered, e.g., "Command entered: `<cmd>`".
3. If the command entered is a "get average" command, then the client will output a message such as "Fetching Lab 1 average:". If the command entered is "GG", the client will output a message such as"Getting Grades:"
4. When the server receives the TCP connection, it should print output, e.g., "Connection received from `<IP address>` on port `<port>`."
5. The server should print out what it has received, e.g., "Received GL3A command from client.".
6. If an ID was sent and matches a database entry, the server outputs a confirmation message, e.g., "User found." Otherwise, the server prints out an error message, e.g., "User not found." and the server closes the connection. Otherwise, it encrypts and returns the requested information. The server outputs the encrypted message that is sent.

7. The client will then retrieve the server response, decrypt and output it on the terminal window.
8. Show that steps 2 to 7, above, are repeatable without restarting the server.

Writeup: You must upload the following two documents to your Avenue To Learn Lab 2 group dropbox by Sunday, March 12:

1. A PDF report that briefly describes how you implemented the client and server applications. Make sure that the names and student ID numbers are on the front page of the report. In a few sentences explain how each member of the group contributed to the lab.
2. A single Python source code (py) file that includes both the client and server classes that you created.