Assignment 3 (CS 558)

Due: 11:59pm November 10 (Sunday)

This assignment is done individually. You can use C/C++/Java/Python in this assignment.

In this assignment, you will implement a secure remote key-value store system using the **Secure Socket Layer (SSL),** which enables clients to add and retrieve key-value pairs over the network. SSL is used to establish a secure connection between the server and the client.

Your will implement an SSL server and an SSL client. The server is an iterative server (i.e., the server serves one client at a time). The server has at least one argument <server_port>, which specifies the port number at which the server accepts the connection request from the client. The port number should be a user-defined number between 1024 and 65535. As multiple students will test their programs on remote.cs.binghamton.edu, please use a unique port number (e.g., the last 4 digits of your B number) so that it is different from other students' port numbers.

The SSL client has at least two arguments: <server_domain> and <server_port>. <server_domain> is the domain name of the machine on which the server is running. There are seven remote.cs machines: remote01-07.cs.binghamton.edu. For example, If the server runs on remote01.cs.binghamton.edu, then the domain name of the server is remote01.cs.binghamton.edu. When we test your program, we may execute your server and your client on the same or different remote.cs machines. <server_port> is the port number of the server. You can add other arguments to the client and the server if needed. If you use C/C++, your program needs to convert the domain name to the corresponding 32-bit IP address using **gethostbyname** function (https://man7.org/linux/man-pages/man3/gethostbyname.3.html).

If you use C, your Makefile should generate two executables **serv** and **cli**. If you use Java, your Makefile should generate files **Serv.**class and C**li.class**. If you use python, your python files should have name **serv.py** and **cli.py**.

For example, if you use C/C++, the server can be invoked as:

./serv ***3479***

The client can be invoked as (if the server is running on remote01):

./cli ***remote01.cs.binghamton.edu 3479***

The key-value store has the following format:

$<key_1>$ $<value_1>$
……
$<key_n>$ $<value_n>$

$<key_i>$ is a key, which is a sequence of lower-case letters (e.g., lego, phone). $<value_i>$ is a value, which is an integer (e.g., 20, 40). In the key-value store, the key and value are separated using a white space.

Upon connection, the client prints "srs **>"**, which allows the user to execute the following commands:

srs > add <key> <value> <filename>

- The client sends the command to the server.
- If <filename> does not exist, the server creates a file with that name in the directory where the server executes and adds the key-value pair "<key> <value>" to the file. The server then sends "the key-value pair has been added successfully" to the client and the client displays the message to the user.
- If <filename> exists and the key <key> is not already in file <filename>, then the server adds the key-value pair "<key> <value>" to <filename> and sends "the key-value pair has been added successfully" to the client. The client displays the message to the user.
- If the key <key> already exists in file <filename>, the server sends "the key already exists in the file" to the client, which then displays the message to the user.
- Example:
    - src > add abc 100 store // add the key-value pair "abc 100" to the file "store".
    - src > add cde 3000 store //add the key-value pair "cde, 3000" to the file "store".
    - After executing the above commands, the file "store" will contain:

      abc 100
      cde 3000

srs > retrieve <key> <filename>

- The client sends the command to the server.
- If <filename> does not exist, or if the key <key> is not found in file <filename>, the server sends "the file does not exist" or "the key is not in the file" to the client, respectively. Otherwise, the server retrieves the value associated with <key> from file <filename> and sends "the value is:" followed by the value to the client. The client then displays the message received from the server.
- Example:
  Assume the file "store" contain the following information:

      abc 100
      cde 3000

  After executing the command "retrieve cde store", the client prints "3000".

srs> exit

- The client sends the command to the server and terminates the connection.
- The server continues listening for new connections.

You will need to generate a **public key certificate for the server** to establish the SSL connection. The certificate can be a self-signed certificate. You do NOT have to generate a certificate for the client. You can use commands or programs to generate the certificate. The certificate generated should be stored as a file, which will be included in the assignment submission. For example, if you use C, you can use the following command to generate the certificate file "cert.pem".

```
openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365
```

If you use java, you can use keytool to generate the certificate.

For ease of grading, please try not to use passphrase when generating the certificate. If this is impossible, please use passphrase 1234.

If you use C, please use the following commands to compile your C program:

```
gcc -Wall -w -o sslcli sslcli.c -I/usr/local/ssl/include/ -L/usr/local/ssl/lib -lssl -lcrypto
gcc -Wall -w -o sslserv sslserv.c -I/usr/local/ssl/include/ -L/usr/local/ssl/lib -lssl -lcrypto
```

Please note that, you may utilize any publicly available code for SSL connection. However, you must write your own code to implement the commands. You should also generate the certificate for the server by yourself.

Grading guideline

- Readme: 4'
- Makefile (C/C++/Java): 6'
- SSL connection (C/C++/Java): 41'
- SSL connection (python): 47'
- Add: 22'
- Retrieve: 22'
- exit: 5'

Submission guideline

- Create a directory with a unique name (e.g. p3-[userid]), which contains
  - A makefile
  - A README file
  - A sub-directory "server", which contains the source code of the server and the server's certificate
  - A sub-directory "client", which contains the source code of the client
- **README** file (text file, please do not submit a .doc file) contains
  - Your name and email address.
  - The URL of the website where you get the SSL code.
  - Whether your code was tested on remote.cs.
  - How to compile and execute your program.
  - (Optional) Briefly describe your algorithm or anything special about your submission.
- Tar the contents of this directory using the following command.

  tar –cvf p3-[userid].tar p3-[userid]

  E.g. tar -cvf p3-pyang.tar p3-pyang/
- Upload the tared file you create above to brightspace.

Academic Honesty:

All students should follow Student Academic Honesty Code (**if you have not already read it, please read it carefully**). All forms of cheating will be treated with utmost seriousness. You may discuss the problems with other students, however, you must write your OWN codes and solutions. Discussing algorithms and code is NOT acceptable. Copying an assignment from another student or allowing another student to copy your work may lead to the following:

- **Report to the School of Computing and Watson College.**
- **0 in the assignment or F in this course.**

Moss will be used to detect plagiarism in programming assignments. You need ensure that your code and documentation are protected and not accessible to other students. Use **chmod 700** command to change the permissions of your working directories before you start working on the assignments. **The use of ChatGPT or other AI tools is prohibited in this assignment.**

If you have any questions about whether an act of collaboration may be treated as academic dishonesty, please consult the instructor before you collaborate.