Programming Assignment 1
CS6903: Network Security
2024-25 - Semester II
Department of Computer Science and Engineering
Indian Institute of Technology Hyderabad
Due Date: 3rd February 2025, 11:55pm [*No extension]

This programming assignment makes you acquainted with the working principle of Cryptography. You need to establish a communication channel between two processes (Client/Server) to achieve different functionalities as described in each task. You can use any programming language you are comfortable with. BUT CANNOT USE IN-BUILD LIBRARY TO IMPLEMENT ANY OF THE TASK.

Please note that your submission will be evaluated on a Linux machine. Therefore, ensure that your programming platform is compatible with the Linux environment. Submissions that fail to execute on Linux will receive zero marks.

Task 1:Lets Do Some Calculation !!!! Client-Server Calculator [10 Points]

In this task, you need to implement a client-server based calculator. The client can submit any number of queries (i.e. mathematical expressions) to the server. The server will evaluate client's queries one by one and send back the answer as a response.

Requirement specifications for this task are as follows:

1. Input expression is always space separated, i.e. blank space between operand and operator.
2. Input expression always starts with an operand.
3. Input expression can have at most 20 operands (i.e., it can have at most 19 operators).
4. Operands in input expression may be an integer or float/double number.
5. Input expression can contain any combination of these five operators viz. (i) Addition (+), (ii) Subtraction (–), (iii) Multiplication (*), (iv) Division (/), and (V) modulo (%).
6. When server performs calculation, it needs to follow **BODMAS** rule where division, multiplication and modulo operators are at same precedence (say precedence-1) and addition and subtraction are at precedence-2. According to BODMAS, server should first perform precedence-1 operations followed by precedence-2 operations. Among same precedence operations, server should perform operation from left to right.
7. When client receives **END** word from STDIN, it should send a termination signal to the server to terminate it properly.
8. On receiving the result of mathematical expressions from the server, client should print "**RESULT:** " followed by result. As an example, if client receive **5** from the server, then it should display "**RESULT: 5**" on STDOUT.

Example of Input & Output:

```
[INPUT]  1 + 5 * 2 - 1 + 10
[OUTPUT] RESULT: 20
[INPUT]  5 + 1 - 2 * 5 + 10
[OUTPUT] RESULT: 6
[INPUT]  END
```

Task 2: Lets Chit-Chat !!! Chat Between Two Users [20 points]

Time to move beyond the client-server model and enable direct communication between two users.

What to do:

- **Client and Server Programs:**
  - Create one server program and one client program.
  - The client program can be run multiple times to create multiple users. For this task, run the client program twice to simulate two users (User-1 and User-2).
- **Fetching Connected User Details:** A user (e.g. User-1) can contact the server to retrieve a list of all connected users, including their IP addresses and port numbers (if needed).
- **Establishing a P2P Connection:** After receiving the details from the server, User-1 can initiate a peer-to-peer (P2P) connection with any other user (Say User-2) including himself, if he/she wishes to communicate.
- **Chat Functionality:** Once connected, both users can exchange messages in a chat session (without involving the server) until one of them terminates the connection by sending the message **EOM** (End of message).
- **Printing the messages:**
  - In the terminal, display both the sent and received chat data at each end.
  - Format the output as follows, with each message pair separated by a new line:
    [Send]: <send_data>
    [Received]: <received_dat>

Task 3: Establishing a Shared Secret!!!! Generating a Shared Key Using Diffie-Hellman [30 points]

Now, we need to share the secret key between two Users. Once User-1 establishes a connection with User-2 (as described in **Task 2**), they will securely exchange a shared secret using the Diffie-Hellman protocol.

What to do:

- **Reading Parameters:** The client program reads the values of p and g from environment variables named **P** and **Q**, respectively.
- **Generating Secrets:**
  - Both User-1 and User-2 independently generate their own private secret numbers (a and b).

  - Using the Diffie-Hellman key exchange protocol, they communicate to establish a shared secret (K_{SC}).

- **Initializing the Key:**
  - Use the shared secret (K_{SC}) to initialize the seed for the generation of random numbers.
  - Based on the generated random number, drive a key for AES or DES encryption algorithm (the choice of AES or DES is on you).
  - The derived key must be identical for both parties.

- **Printing the messages:**
  - In the terminal, display both the sent and received information at each end related to the key exchange steps.
  - Format the output as follows, with each message pair separated by a new line:
    [Send]: <send_data>
    [Received]: <received_dat>

- **Printing the Key:** Both users will print the generated encryption key to verify consistency.

Task 4: Encrypt the Messages!!!! Securing the Chat Communication Suign AES [40 points]

Now that you have established a key exchange protocol, it's time to secure the chat using encryption.

What to do:

- **Encryption and Decryption:** Both users will use the shared key derived in **Task 3** to encrypt outgoing messages and decrypt incoming messages using AES.

- **Printing the messages:**
  - On the terminal, display both the ciphertext and the corresponding plaintext received at each end.
  - Format the output as follows, with each message pair separated by a new line:
    [Cipher]: <ciphertext>
    [Plain]: <plaintext>

Deliverable:

Source Code:

- Submit the source code for **Task 1**, **Task 2**, **Task 3**, and **Task 4.**

- Document your code properly with comments and meaningful variable names.

- Place each task's source code in a separate folder named with the Task number.

README.txt:

Create a README.txt file containing the following:

- A summary of your implementation for **each task.**
- Add the instruction for execution.
- If you used any shared code/scripts or have a discussion with others, clearly mention all in this file.
- **Contribution of the group member, What has been done by whom.**

Network traffic capture (.pcap files):

- For **Task 2**, **Task 3** , and **Task 4**, capture the network traffic using *Wireshark* or *tcpdump* and save it as a **.pcap** file.
- Show the **encrypted traffic** to demonstrate that encrypted messages are sent and received.
- Place the **.pcap** files for each task (including earlier tasks) inside each task folder to provide a clear record of network communication.

Submission Format:

- Submit a **tar-ball/zip** file of your assignment named as your **roll number** (e.g., 12345678.tar.gz).
- The zip file should contain the Source code, README.txt file, and network traffic captured files.
- Upload your submission to the Classroom. Submissions via email or any other medium are strictly prohibited.

All the best. Looking forward to the submissions !!.....PS: Start Early.

ANTI-PLAGIARISM STATEMENT <Include it in your report>

We certify that this assignment/report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether books, articles, packages, datasets, reports, lecture notes, or any other document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand my responsibility to report honor violations by other students if we become aware of it.
Names:
Date:
Signature: *<keep your initials here>*