International Institute of Information Technology Hyderabad

System and Network Security (CS5.470)

Lab Assignment 1:
Secure Client-Server Connection
Using a Distributed Double DES (D-DDES)
Hard Deadline: 30 January 2025 (Thursday, 23:59)
Total Marks: 100

Note:- *It is strongly recommended that no group is allowed to copy programs from others. Hence, if there is any duplicate in the assignment, simply both parties will be given zero marks without any compromise. The rest of the assignments will not be evaluated further, and assignment marks will not be considered towards final grading in the course. No assignment will be taken after the deadline.* ***You can use C or python programming language for implementation. No other programming language implementation will be accepted.***

Objective

The goal of this assignment is to design and implement a secure multi-client-server communication system using the Distributed Double DES (D-DDES) cryptographic scheme. Students will simulate a networked environment where multiple clients exchange data with a server, utilizing D-DDES for enhanced security.

Problem Description

- Suppose there is a one server and multiple clients present in a networked environment.
- Each client and the server generate a pair of DES keys (Key1 and Key2) using the Diffie-Hellman key exchange protocol. The server stores these keys securely for subsequent communication rounds with the clients. Note that DDES uses two 56-bit keys $Key1$ and $Key2$, and 64-bit plaintext block, and produces 64-bit ciphertext block as shown in Figure 1.
- At the start of each communication round, the server generates a unique session token (nonce) and sends it to all connected clients. The session token is encrypted using Key1 of each client.

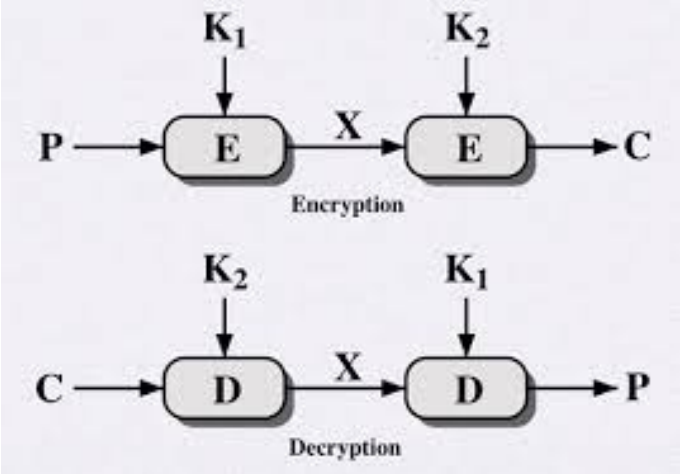

Figure 1: Double DES operation

- Each client performs the following steps to exchange data securely:
  1. Encrypts the data twice using Double DES with Key1 and Key2.
  2. Appends the session token to the encrypted data for validation.
  3. Sends the encrypted message and a HMAC (Hashed Message Authentication Code) (for instance, hash value of the message plus the shared key Key2) to the server.
- The server:
  1. Verifies the session token for authenticity.
  2. Decrypts the Double DES-encrypted data using the client-specific Key1 and Key2.
  3. Verifies the HMAC for message integrity.
  4. Aggregates the data received from all clients (e.g., computes the average or sum of numeric data).
  5. Encrypts the aggregated result using the client-specific DES keys and sends it back to the clients with a new MAC.
- To handle the error add following error scenario:
  - Incorrect HMAC: The server rejects the message and logs the incident.
  - Invalid Session Token: The server disconnects the client and sends an appropriate error message.
  - Data Tampering: If the decrypted data does not match the expected format, the server discards it and sends a warning.

Main Tasks

- **Task 1**: Develop the pair of DES encryption keys and implement the secure data transmission protocol.
- **Task 2**: Implement the session token validation and HMAC generation/verification process.

- **Task 3**: Design the server to handle multiple clients, manage Double DES decryption, session token validation, and data aggregation.

Communication Protocol

Here are the opcodes you can utilize in your code to perform the specific tasks outlined in this assignment. These opcodes help distinguish the purpose of each message in the communication protocol.

Opcode	Message Type	Description
10	KEY_VERIFICATION	A client and server verify the established keys Key1 and Key2 through handshake mechanism
20	SESSION_TOKEN	Server sends a session token encrypted with Key1
30	CLIENT_ENC_DATA	Double DES encrypted data sent to the server
40	ENC_AGGR_RESULT	Encrypted aggregated result sent to clients
50	DISCONNECT	End the session for all participants

Table 1: Communication Protocol Opcodes

Submission Guidelines

1. Submit a zipped file named `_<group_number>_lab1.zip` containing:
   - `client1.py`: Client 1 implementation.
   - `client2.py`: Client 2 implementation.
   - ⋮
   - `clientn.py`: Client n implementation.
   - `server.py`: Server implementation.
   - `README`: Explanation of implementation and execution steps.
2. Each group must demo their solution.
3. Late submissions will not be accepted.

Evaluation Criteria

- Correct implementation of double DES keys encryption and decryption.
- Proper handling of session tokens and HMACs.
- Robustness against tampered or invalid messages.
- Code quality and documentation.