Team Number - 6

Saurabh Kumar - 2024202029

Atharva Keskar - 2024202017

Kuldeep Lakhe - 2024202025

System and Network Security (CS5.470) - Lab Assignment 1

Our Assumptions:

- Due to limited computation we used smaller prime number 137 instead of the recommended 2048 bits in the generator
- We are printing both the aggregate of the numbers and the average of them in the client console so our output format is aggregate: xxx , avg : xx.xx, count: xx
- We used Pycryptodome library for the implementation

Summary

We implemented a secure multi-client-server communication channel using Distributed Double DES (D-DDES) encryption. The implementation includes key security features such as Diffie-Hellman key exchange, session token management, HMAC verification, and double DES encryption/decryption processes.

Architecture Overview

The system consists of two main components:

- A multi-threaded server that can handle multiple client connections
- Client implementation with secure communication capabilities

1. Key Exchange Implementation

- **Diffie-Hellman Key Exchange**
  - Uses parameters: p = 137 (prime modulus), g = 2 (generator)
  - Generates two DES keys (Key1 and Key2) from shared secret
  - Implementation concern: Small prime modulus (137) makes it vulnerable to brute force attacks

2. Session Management

- **Session Token Generation**
  - Uses `secrets.token_hex(16)` for generating random 16-byte session tokens
  - Tokens are encrypted using Key1 before transmission
  - Server maintains session token mapping per client address

3. Encryption Implementation

- **Double DES Implementation**
  - Uses DES in ECB mode for both encryption layers
  - Data flow: Plaintext -> DES(Key1) -> DES(Key2) -> Ciphertext

4. Message Authentication

- **HMAC Implementation**
  - Uses SHA-256 for HMAC generation
  - Key2 used as HMAC key
  - Properly implemented verification on both sides

Protocol Analysis

Communication Flow

1. **Key Verification (Opcode 10)**
   - A client and server verify the established keys Key1 and Key2 through handshake mechanism

2. **Session Token Distribution (Opcode 20)**
   - Server sends a session token encrypted with Key1

3. **Data Transmission (Opcode 30)**
    - DoubleDESencrypteddata sent to the server

4. **Result Distribution (Opcode 40)**
    - Encrypted aggregated result sent to clients

5. **Disconnect (Opcode 50)**
    - End the session for all participants

Error Handling and Security Controls

Implemented Security Controls

1. **Session Token Validation**
    - Immediate disconnection on invalid tokens
    - Proper cleanup of client resources
2. **HMAC Verification**
    - Rejects messages with invalid HMACs
    - Implements proper error responses
3. **Data Validation**
    - Validates message format and structure
    - Implements proper error handling for malformed data
    - Implement session timeout mechanisms

Conclusion

Our implementation successfully meets the basic requirements of the assignment, implementing all required security features.
