CSCI368 Network Security
2025 Session 3

Assignment 1 (20 Marks)

Submission Due: 01 Aug 2025, 23:55 (SG Time)

Objectives

On completion of this assignment, you should be able to:

- Understand some basic techniques for building a secure channel.
- Understand network programming.

Write (Java, Python or C/C++) UDP programs allowing two parties to establish a secure communication channel. For simplicity, let us call the programs "Host" and "Client", which are executed by Alice and Bob, respectively.

Alice and Bob share a common password PW, which contains at least 6 alphanumeric characters. Alice/Host stores the password in the hashed form (i.e., H(PW) where H denotes the SHA-1 hash function) and Bob/Client memorizes the password. They want to establish a secure communication channel that can provide data confidentiality and integrity. They aim to achieve this goal via the following steps: (1) use the shared password to establish a shared session key; (2) use the shared session key to secure the communication.

Step 1 is done via the following key exchange protocol:

1: B → A: "Bob"

2: A → B: E(H(PW), p, g, g^a mod p)

3: B → A: E(H(PW), g^b mod p)

4: A → B: E(K, N_A)

5: B → A: E(K, N_A+1, N_B)

6: A → B: E(K, N_B+1) or "Login Failed"

In the above protocol, p and g are the parameters for the Diffie-Hellman key exchange, E denotes the RC4 stream cipher. The shared key K is computed as K = H(g^{ab} mod p) where a and b are random numbers selected by Alice and Bob in each session, and N_A (resp. N_B) denotes a nonce selected by A (resp. B).

After establishing the session key, **step 2** is achieved as follows:

1. whenever Alice wants to send a message M to Bob, Alice first computes hash = H(K||M||K), and then computes C = E(K, M||hash) and sends C to Bob. Here || denotes the string concatenation.
2. upon receiving a ciphertext C, Bob first runs the decryption algorithm to obtain M||hash = D(K, C). After that, Bob computes hash' = H(K||M||K) and checks if hash = hash'. If the equation holds, then Bob accepts M; otherwise, Bob rejects the ciphertext.
3. the same operations are performed when Bob sends a message to Alice.

Implementation guidelines

- Place Host and Client in two separate directories: Alice and Bob.

- Generate the Diffie-Hellman parameters (p, g), choose a password PW for Bob and save (p, g, H(PW)) in a text file under the directory of Alice. This completes the setup of the Host. You can use an individual program to perform the setup.

  Remark: You can use an open-source crypto library or some open-source code to generate the Diffie-Hellman parameters.

- Alice executes Host.

  - Host reads the parameters and the hashed password from the file.

  - Host is running and listening to the opened port (you need to select a port for your code).

- Bob executes Client.

  - Client asks for a password PW from user input (via keyboard).

  - Client sends a connection request "Bob" to Host.

  - Client is ready and listens to the port.

- Host generates a random a, and sends E(H(PW), p, g, g^a mod p) to Client.

- Client generates a random b, computes g^b mod p, and sends E(H(PW), g^b mod p) to Host. Client computes the shared key K.

- Upon receiving the ciphertext from the Client, Host decrypts it using H(PW) to obtain g^b mod p and computes the shared key K. Host picks a nonce N_A and sends E(K, N_A) to Client.

- Client performs the decryption to get N_A, picks a nonce N_B, and sends E(K, N_A+1, N_B) to Host.

- Host performs the decryption and checks the response N_A+1. If the response is correct, Host sends E(K, N_B+1) to the client; otherwise, it sends "Login Failed" to the Client and terminates the current connection.

- Client checks the response N_B+1. If the response is not correct, Client terminates the connection. Otherwise, the handshake is successful and the Client starts the conversation with the Host.

- If the handshake is done successfully

  - Either Alice or Bob can send a message encrypted and authenticated by the key K. They type the message on their own terminal. The message is processed by their code (Host or Client) according to step 2 given above.

  - The received message is printed on the screen if decryption is successful. Otherwise, an appropriate error message is displayed on the screen.

  - To terminate the connection, either party should type "exit".

Coding requirement:

You need to write the codes for implementing Host and Client. Some sample code for UDP will be provided, but you can also use other open-source code as you like. You can use a crypto library or some open-source code to implement the encryption and hashing functions and the Diffie-Hellman key exchange, including the generation of the Diffie-Hellman parameters. **You should cite the source if you use a downloaded code.**

How to run?

Your programs should run according to the protocol. Host and Client should be executed on different windows. For convenience of marking, please use the local IP: 127.0.0.1 for the submitted version. For simplicity, there is no GUI required in this assignment. That is, messages are simply typed on the sender's window and printed on the receiver's window. The looping should continue until the connection is terminated.

Files to be submitted:

All source codes.

A readme file (text/ACSII only): instructions about how to compile and run your code.

Submission

Compress all the files to be submitted into a zip file and submit it via the submission link provided in the Moodle site.

Late Submission: Penalty is **5%** deduction per day (including weekends) unless Academic Consideration is granted.

Marking

Mark distribution:

1. UDP connection: 2 marks
2. System setup: 2 marks
3. Key Establishment (Step 1): 10 marks
   //proper message display in key establishment: 2 marks
4. Data Encryption/Decryption (Step 2): 6 marks

Plagiarism

A plagiarised assignment will receive a zero mark and be penalised according to the university rules. Plagiarism detection software may be used.