International Institute of Information Technology Hyderabad

System and Network Security (CS5.470)

Lab Assignment 2:
Secure Telemedical Conference using Digital Signature

Hard Deadline: 7 March 2025 (23:59 PM)
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,C++ or Python programming language for implementation. No other programming language implementation will be accepted.***

1 Introduction

With the increasing reliance on Telemedical systems, ensuring the security of one-on-one consultations between doctors and patients is essential. This assignment focuses on designing a secure Telemedical system, where a doctor can interact confidentially with his/her treating patients using cryptographic protocols. The primary objectives include secure authentication, data access control, and encrypted communication.

2 Problem Statement

A doctor operates within a telemedical platform, handling multiple patient consultations. To maintain patient confidentiality and prevent unauthorized access, a robust security mechanism must be employed. The doctor and a patient must authenticate themselves before initiating a secure session, followed by encrypted communication to protect sensitive medical data. The process involves the following components:

- Establishing secure identities through cryptographic authentication.
- Exchanging session keys using the ElGamal cryptosystem.

- Encrypting messages and consultation records using AES-256 encryption.
- Ensuring message integrity using cryptographic signatures.

2.1 Phase 1: Initialization

2.1.1 Key Generation

Each participant A generates a key pair as follows:

- Select a large prime number p and a **generator** g in the finite field $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$ so that the Discrete Logarithm Problem (DLP) becomes intractable. For this reason, you should use the library to generate large prime.
- Choose a **private key** x by randomly selecting from $\mathbb{Z}_{p-1}^*$.
- Compute the **public key** y as:

$$y = g^x \mod p$$

- The **public key** is $KU_a = (p, g, y)$, and the **private key** is $KR_a = x$.

2.1.2 Encryption and Decryption

For encrypting a message m:

- The sender chooses a random **ephemeral key** k from $\mathbb{Z}_{p-1}^*$.
- Compute the ciphertext components as follows:

$$c_1 = g^k \mod p$$

$$c_2 = m \cdot y^k \mod p$$

- The encrypted message is (c_1, c_2).

For decryption:

- The receiver computes:

$$m = c_2 \cdot (c_1^x)^{-1} \mod p$$

where $(c_1^x)^{-1}$ is the modular inverse of $c_1^x \bmod p$, used to retrieve the plaintext message.

2.2 Phase 2: Authentication and Key Exchange

In order to start the authentication, each patient having IoT (mobile) device D_i sends an authentication request to a doctor (acting as the GWN) that contains the following parameters:

- First, the D_i will select a TS_i which is a current timestamp generated by D_i.
- Then, it selects a RN_i (random secret) generated by D_i.
- After that, it will generate $K_{D_i,GWN}$ as a random key generated by D_i.
- D_i will prepare an authentication request as: $\langle TS_i, RN_i, ID_{GWN}, E_{KU_{GWN}}[K_{D_i,\text{GWN}}], Sign_{Data_1}\rangle$, where ID_{GWN} is the identity for the GWN, $E_{KU_{\text{GWN}}}$ denotes the ElGamal encryption of data using doctor's public key (KU_{GWN}) and $Sign_{Data_1}$ is the ElGamal signature on data $\{ TS_i, RN_i, ID_{GWN}, E_{KU_{GWN}}[K_{D_i,GWN}]\}$ using the private key of the D_i, KR_{D_i}.

The patient device will send this authentication request to the doctor device where the doctor device will follow the following steps:

- Upon receiving authentication request at time TS_i^*, the gateway node (doctor device) GWN verifies the timestamp by checking the condition: $|TS_i^* - TS_i| \leq \Delta TS$, where ΔTS denotes the "maximum allowable transmission delay". If it is within the defined time, then the device will allow the request for further verification; otherwise, it will discard the request.
- The doctor device will validate the $Sign_{Data_1}$. If the signature matches and is validated, then proceed; otherwise, discard the request.
- After validating the signature, the doctor device will decrypt the session key by using its private key as $E_{KR_{GWN}}[K_{D_i,GWN}]$.
- After computing the secret key $K_{D_i,GWN}$, the doctor (GWN) device will respond with the following message:

  $$\langle TS_{GWN}, RN_{GWN}, ID_{D_i}, E_{KU_{D_i}}[K_{D_i,GWN}], Sign_{Data_2}\rangle$$

  where

  - TS_{GWN} is the current timestamp generated by GWN
  - RN_{GWN} is the random secret generated by GWN
  - ID_{D_i} is the identity for the patient device
  - $Sign_{Data_2}$ is the ElGamal signature on the data $\{ TS_{GWN}, RN_{GWN}, ID_{D_i}, E_{KU_{D_i}}[K_{D_i,GWN}] \}$ using the private key of GWN.

After receiving the response message from the doctor device (GWN), the patient device (D_i) will perform the following tasks:

- Upon receiving authentication request at time TS_{GWN}^*, the gateway node doctor device D_i verifies the timestamp by checking the condition: $|TS_{GWN}^* - TS_{GWN}| \leq \Delta TS$. If it is within the defined time, the device will allow the request for further verification; otherwise, it will discard the request.

- The doctor device will validate the $Sign_{Data_2}$. If the signature matches and is validated, then proceed; otherwise, discard the request.
- After validating the signature, the patient device (D_i) will calculate the session key as

$$SK_{D_i,GWN} = h(K_{D_i,GWN} \parallel TS_i \parallel TS_{GWN} \parallel RN_i \parallel RN_{GWN} \parallel ID_i \parallel ID_{GWN})$$

- After calculating the shared session key with the GWN, the D_i will select the current time stamp $TS_i^{'}$ and calculate the session key verifier as $SKV_{D_i,GWN} = h(SK_{D_i,GWN} \parallel TS_i^{'})$.
- D_i will now send the following message to GWN: $\langle SKV_{D_i,GWN}, TS_i^{'} \rangle$.

After receiving the message from the D_i, the GWN will do the following steps:

- Upon receiving authentication request at time $TS_i^{*'}$, the gateway node doctor device GWN verifies the timestamp by checking the condition: $|TS_{i'}^{*} - TS_i^{'}| \leq \Delta TS$. If it is within the defined time, the device D_i will allow the request for further verification; otherwise, it will discard the request.
- The doctor device will calculate the session key as:

$$SK_{GWN,D_i} = h(K_{D_i,GWN} \parallel TS_i \parallel TS_{GWN} \parallel RN_i \parallel RN_{GWN} \parallel ID_i \parallel ID_{GWN})$$

  and calculate the session key verifier as $SKV_{GWN,D_i} = h(SK_{GWN,D_i} \parallel TS_i^{'})$.
- If the condition $SKV_{GWN,D_i} \stackrel{?}{=} SKV_{D_i,GWN}$ is valid; both the devices are on the same session key. They can start communicating securely; otherwise, they terminate the session request and block the patient's device for 24 hours.

Figure 1 illustrates the secure communication among the doctor and the patients.

2.3 Phase 3: Secure Message Broadcasting to Patients

After establishing the session keys with all the n number of patients, the doctor's device will broadcast a message so that every authenticated patient device will receive the message and securely decrypt and read it. In order to perform this process GWN will perform the following steps:

- First, the GWN will compute the shared group key as:

$$GK = h(SK_{D_{1,GWN}} \parallel SK_{D_{2,GWN}} \parallel SK_{D_{3,GWN}} \parallel \cdots \parallel SK_{D_{n,GWN}} \parallel KR_{GWN})$$

- After computing the group key, the GWN encrypt this key using the established session key for each patient as: $E_{SK_{D_i,GWN}}[GK]$
- GWN sends this group to each user specifically, and then when the doctor wants to send a broadcast message $\mathcal{M}$ (in this case unavailability schedule), it encrypted that $\mathcal{M}$ by using the group as: $E_{GK}[\mathcal{M}]$

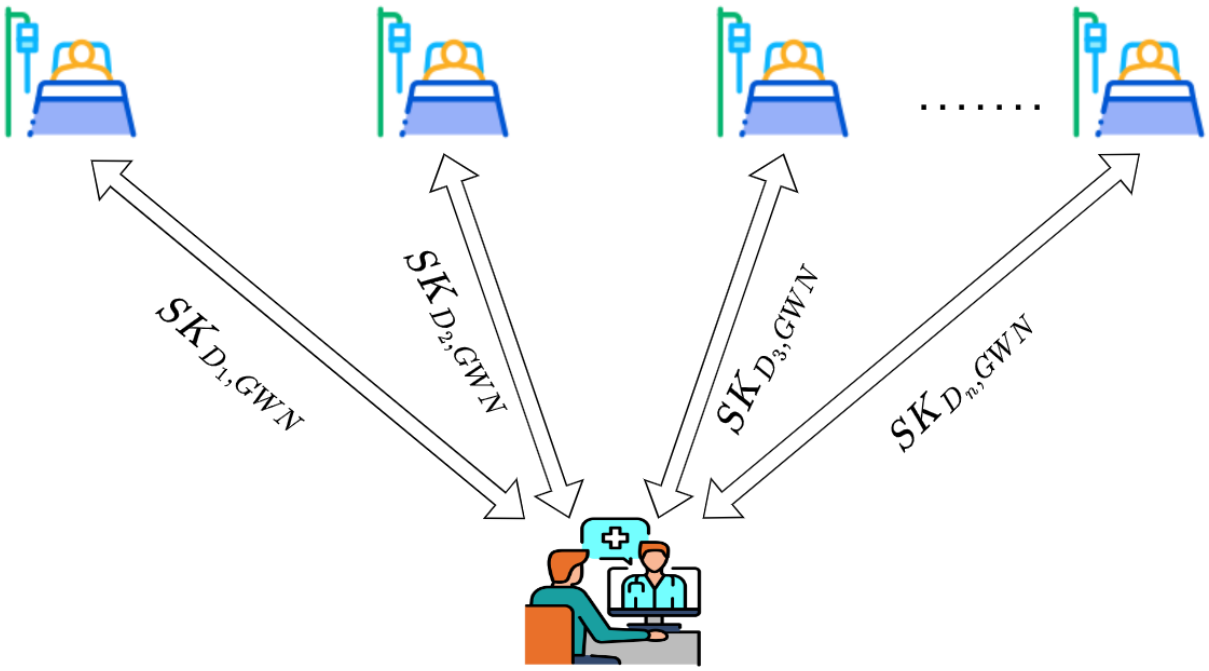

Figure 1: Secure authentication of patient and doctor

3 Performance Analysis

To test the designed authentication, every team needs to evaluate the execution time of various cryptographic primitives. This includes measuring how long it takes to generate private-public key pairs, sign data, and compute hashes. Give a table in your read-me file explaining how much time is taken by which cryptographic primitive.

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 device and GWN verify the established keys
20	SESSION_TOKEN	GWN sends an encrypted session token
30	GROUP_KEY	Encrypted Groupkey established by the server
40	ENC_MSG	Emergency message broadcasted by GWN
50	DEC_MSG	Emergency message decrypted at clients using group key
60	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:
   - `patient_1.py`: Device #1 implementation.
   - `patient_2.py`: Device #2 implementation.
     ⋮
   - `patient_n.py`: Device #n implementation.
   - `doctor.py`: Gateway Node 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
- Code quality and documentation