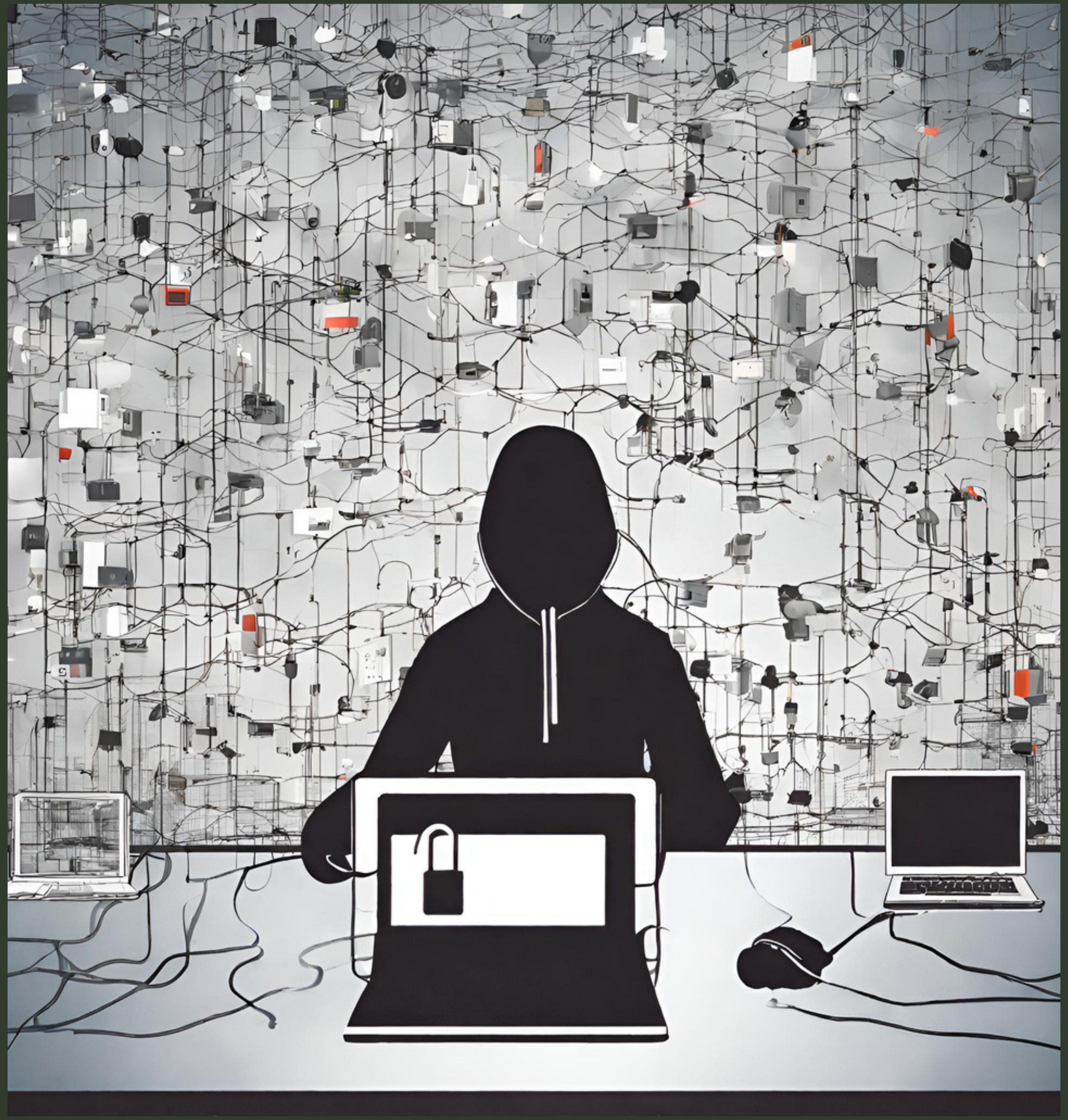
March 31, 2024

Assignment 3
Project - 1

Presented by Sarvagya Kaushik & Kunal Sharma

Winter 2024



Network Security

Prof. B N Jain

We have 2 files named ‘client.py’ and ‘pkda.py’. Below is the explanation of each file one by one.

1. client.py: This file contains in total 2 classes and a main() function. Which are explained below:

a) Class RSA: This class contain 5 functions which are as follow:

- **Encryption(message: tuple, key: tuple):** This function takes a message tuple and a key tuple as input and encrypts the message using the RSA encryption algorithm with the provided key.
- **Decryption(message: tuple, key: tuple):** This function takes an encrypted message tuple and a key tuple as input and decrypts the message using the RSA decryption algorithm with the provided key.
- **RSA_Operation(m, x, n):** This is a helper function for RSA encryption and decryption. It performs the core RSA operation (m^x) mod n.
- **RSA_Encode(msg):** This function encodes a given string message into a tuple of ASCII values, which is useful for encryption.
- **RSA_Decode(tup):** This function decodes a tuple of ASCII values into a string message, which is useful for decryption.

b) Class Client: This class contain 8 functions and a constructor which are as follow:

- **__init__(self, client_id, pr_key, pu_key, pkda_pu_key):** Constructor initializes a client instance with its ID, private key, public key, and PKDA's public key.
- **Generate_msg_for_pkda(self, client_id: int):** Generates a message tuple to request public keys from the PKDA, including the client's ID, current time, and a nonce.
- **Msg_from_pkda(self, message):** Processes the message received from the PKDA, extracting necessary information like public keys and other metadata.
- **Msg_for_client(self, client_id: int, msg: str, nonce=None):** Generates a message tuple to send to another client, including the current time, a nonce, sender's ID, and the encrypted message.
- **Msg_from_client(self, message):** Processes the message received from another client, decrypting it and extracting relevant information.
- **Gen_Nonce(self):** Generates a random nonce (number used once) for secure communication.
- **Res_Nonce(n):** Generates a response nonce based on the received nonce, useful for confirming nonce integrity and we have **Time**(): Returns the current time, used for timestamping messages.
- **Req_pu_k_from_pkda(self, pkda_address, m):** Requests the public key of another client from the PKDA by establishing a connection, sending a message, and receiving the response.

C) main() function:

- The main() function starts by reading key information from text files for **client A's public key** (**A_pu_k.txt**), **client B's public key** (**B_pu_k.txt**), **client A's private key** (**A_pr_k.txt**), **client B's private key** (**B_pr_k.txt**), and the **PKDA's public key** (**pkda_pu_k.txt**) and initializing client instances with their respective keys.
- Depending on the client ID entered by the user, it executes different logic for client A and client B. **For client A** (**ID 1**), the instance is created with its private key (A_pr_k), public key (global_mappings[1]), and the PKDA's public key (pkda_pu_k). **For client B** (**ID 2**), similar initialization is done with its private key, public key, and the PKDA's public key.
- **For client A** (**ID 1**), it initiates a connection with client B (ID 2) and sends encrypted messages using socket communication. It also requests **client B's public key** from the PKDA and establishes a connection with client B to exchange encrypted messages.
- **For client B** (**ID 2**), it listens for incoming connections from client A, responds to messages, and exchanges messages using socket communication. It also r**equests client A's public key** from the PKDA and establishes a connection with client A to exchange encrypted messages.

2. pkda.py: This file contains in total 2 classes and a main() function. Which are explained below:

a) Class RSA: This class contains 6 functions out of which 5 functions are the same as explained above. The sixth function is as follow:

- **gcd(self,x, y, a, b, c, d):** This recursive function **calculates the greatest common divisor (GCD)** of two integers x and y using the Euclidean algorithm. It updates variables a, b, and c in each iteration and stores the result in d if x % y equals 1. It uses the Extended Euclidean algorithm to calculate the decryption key d. The function returns the GCD when x % y becomes 0, otherwise, it recursively calls itself with updated values.

b) Class PKDA: This class contains 3 functions and a constructor which are as follow:

- **__init__(self, mappings, pr_key, pu_key):** Initializes PKDA with mappings, private key, and public key.
- **Msg_from_client(self, message):** Decrypts the message from the client, extracts client ID, generates a nonce, and encrypts the response.

- **Res_Nonce(n):** Nonces are used to prevent replay attacks by ensuring that each communication session is unique. This function generates a new nonce for each session.
- **Time():** This function provides a timestamp for messages exchanged between clients and the PKDA, helping to ensure message freshness and integrity.

C) main() function:

- **Generates RSA public and private keys for multiple entities** (A, B, PKDA), writes them to files, and stores them in a global mapping dictionary. Using the number.getPrime() function from the Crypto.Util module, **prime numbers are generated** to serve as the basis for RSA key pairs. These prime numbers are combined to calculate the modulus (n) and Euler's totient function (phi). Then, suitable values for the private exponent (e) and public exponent (d) are determined. Finally, public and private key pairs are formed for each entity.
- **Initializes PKDA** the PKDA object is instantiated with the global mappings of client IDs to their public keys, along with its own private and public key pairs. These mappings and keys are necessary for the PKDA to facilitate secure communication between clients.

- Using the socket.socket() function, a **TCP socket is created** and bound to a specified localhost address and port number. The server then enters a listening state, awaiting incoming connections from clients.
- The server enters a loop where it **continuously accepts incoming connections** from clients. Upon accepting a connection, it receives data from the client, deserializes it using the pickle.loads() function, processes the message using the Msg_from_client() method of the PKDA, and sends back a response encrypted with PKDA's private key.

Thank you !

Name	Sarvagya Kaushik
Roll No.	2021350
Branch	CSD
E-mail	Sarvagya21350@iiitd.ac.in

Name	Kunal Sharma
Roll No.	2021331
Branch	CSD
E-mail	Kunal21331@iiitd.ac.in

Winter 2024