Network Security

Assignment 4 - Project 2

Prof. B N Jain

Submitted by:

Sarvagya Kaushik - 2021350

Kunal Sharma - 2021331

The programming language used in the assignment is **Python**. Below is the explanation of each file.

1. policeman.py: It simulates a secure **authentication** and **communication** process between a client, a Ticket Granting Server (TGS), and a Service Provider (SP) over a network.

a) Libraries Used: 'socket' for networking, **'Crypto'** for cryptographic operations, **'pickle'** for serialization, **'time'** for time-related operations, **'datetime'** for handling dates and times, and **'base64'** for base64 encoding and decoding.

b) AES Encryption and Decryption Functions: The two functions that are for AES encryption and decryption are aes_encrypt() and aes_decrypt() respectively. These functions use AES encryption in **Cipher Block Chaining (CBC) mode**. The encryption function pads the data before encryption, and the decryption function reverses this process.

c) Signature Verification Functions: Two functions are defined for verifying signatures. First is **verify_signature()** it verifies a signature using **PKCS1_v1_5** signature scheme and second is **verify_time_signature()** which verifies a signature for a given message using the public key read earlier.

d) Public Key Decryption Function: decrypt_with_public_key() decrypts ciphertext using **RSA** public key encryption.

e) Getting Time from Server: The **get_time_from_server()** function connects to a server specified by a host and port, receives encrypted time and signature from the server, decrypts the time using the public key, and verifies the signature.

f) Main Function: The main() function starts by asking a **username** and **password** of the policeman. It then connects to an authentication server **(auth_socket)**, sends the username and password, and receives the authentication result. It reads a key **(K_C)** from a file, decrypts the received authentication result using **AES**, and sends an authenticator to the **TGS.** It then connects to the TGS, sends encrypted data containing a ticket and authenticator, receives encrypted data from the TGS, decrypts it, and obtains a session key **(K_C_v)** and a ticket for the service provider **(T_v)**. Next, it connects to the service provider, sends encrypted data containing the ticket and authenticator, receives encrypted personal data from the service provider, and decrypts it using the session key. It verifies the received data's integrity and authenticity using a signature and compares the received timestamp with the current time to ensure the data's freshness. If **authentication** and **integrity** checks pass, it prints the received personal data (name and date of birth) and proceeds to receive an image file from the service provider. It decrypts the image data using a **Fernet cipher** and saves it to a file named **'received_image.jpg'.** If authentication fails at any step, it prints an error message.

2. authentication_server.py: It is an authentication Server that authenticates policemen based on their username and password.

a) Libraries Used: 'socket' for networking, **'hashlib'** for hashing passwords, **'Crypto'** for cryptographic operations, and **'pickle'** for serialization.

b) AuthenticationServer Class: It represents the authentication server. It has **3 methods** to add policemen with their hashed passwords and to authenticate policemen based on their provided username and password.
__init__(): Initializes the policeman dictionary to store usernames and hashed passwords.
add_user(): Adds a policeman to the policemen dictionary with their username and hashed password.
authenticate(): Authenticates a policeman by checking if the provided username and password exists.
c) Main Function: It initializes an instance of the **AuthenticationServer class** and It asks for a policeman username and password. Then, it creates a server socket and binds it to localhost on port 8000.

Then it starts listening for incoming connections. When a client connects, it accepts the connection and receives a **username** and **password** from the client. It then attempts to authenticate the policeman using the authenticate method of the AuthenticationServer class. If authentication succeeds, it generates three random keys (**K_C**, **K_C_tgs**, **K_tgs**) and saves them to separate files. It then **encrypts K_C_tgs using K_tgs** and sends it along with **T_tgs** to the client after encrypting them using **K_C**. If authentication fails, it sends an "Authentication failed" message to the client. At last, we closed the client socket.

3. service_provider.py: It is a Service Provider that receives license verification requests from authenticated clients, **verifies the license numbers**, and provides personal data and encrypted images accordingly.

a) Libraries Used: 'socket' for networking, **'pickle'** for serialization, **'datetime'** for handling dates and times, **'Crypto'** for cryptographic operations. **'fernet'** for Fernet symmetric encryption, and **'base64'** for base64 encoding and decoding.

b) Signature Functions: The function named **sign_data()** is used to sign data using **PKCS1_v1_5** signature scheme.

c) Time Signature Verification Function: The function named **verify_time_signature()** is used to verify a signature for a given message using the public key read earlier.

d) Public Key Decryption Function: The function named **decrypt_with_public_key()** is used to decrypt ciphertext using RSA public key encryption.

e) Getting Time from Server: The **get_time_from_server()** function connects to a server specified by a host and port, receives encrypted time and signature from the server, decrypts the time using the public key, and verifies the signature.

f) ServiceProvider Class: It represents the **service provider**. It has a method named verify_license_number() to verify **license numbers**.

g) Main Function: It first initializes an instance of the ServiceProvider class. Then, it creates a server socket and binds it to localhost on port 8002. Then it starts listening for incoming connections. When a client connects, it accepts the connection, receives encrypted data containing **T_v** and **authenticator_c**, decrypts them using a key read from a file, and checks if the decrypted username is the same as entered by the policeman. If the client is authorized, it sends an **'Authenticated'** message to the client and receives encrypted license number and date/time from the client. It verifies the received date/time and if it's within a certain threshold, it verifies the license number using the **verify_license_number()** method of the ServiceProvider class. Based on the verification result, it sends personal data(Name and DOB) and encrypted image to the client. If the client is not authorized, it prints **'Client not authorized'** and closes the client socket.

4. time_server.py: It is a time server that provides the current time to clients over a network securely using RSA encryption and digital signatures.

a) Libraries Used: 'socket' for networking, **'datetime'** for handling dates and times, **'base64'** for base64 encoding and decoding, and classes from the **'Crypto'** library for cryptographic operations.

b) Generating RSA Key Pair: It generates an RSA key pair (**private_key** and **public_key**) with a key size of **2048 bits**. It writes the private key to a file named **'public_key_time.txt'.**

c) Signing and Encryption Functions:

sign_message(): This function takes a message, hashes it using **SHA256**, signs the hash using the private key, and returns the signature.
encrypt_with_private_key(): This function encrypts a message using **RSA-OAEP** encryption with the private key and returns the base64-encoded ciphertext.

d) Function to Get Current Time: The **get_current_time()** function returns the current time in the format '**YYYY-MM-DD HH:MM:SS**'.

e) serve() Function: It initializes a server socket, binds it to localhost on **port 8003**, and starts listening for incoming connections. When a client connects, it accepts the connection and retrieves the client's address. It then gets the **current time**, encrypts it using the private key, and signs it. The encrypted time and signature are sent to the client. Any exceptions that occur during this process are caught and printed.

5. ticket_granting_server.py: It is a Ticket Granting Server (TGS) that **generates session keys** and provides them to authenticated clients.

a) Libraries Used: 'socket' for networking, **'random'** for generating random session keys, **'pickle'** for serialization, and classes from the **'Crypto'** library for cryptographic operations.

b) AES Encryption and Decryption Functions: Same as explained in Point 1.

c) TicketGrantingServer Class: It represents the Ticket Granting Server. It has an attribute session to store session keys.
__init__(): Initializes the sessions dictionary.
generate_session_key(): Generates a random session key.

d) Main Function: It initializes an instance of the **TicketGrantingServer class.** It creates a server socket and binds it to localhost on **port 8001**. When a client connects, it accepts the connection, receives serialized data containing **T_tgs** and **authenticator_c**, and decrypts them using a key read from a file. If the decrypted username is the same as the input username, it generates a session key **K_v** and a client-to-service session key **K_C_v**. It then encrypts **K_C_v** with **K_v** to form **T_v** and sends it back to the client after encrypting it with **K_C_tgs**. If the client is not authorized, it prints "Client not authorized".

Sample Inputs/Outputs

Test Case 1:

Input:	**Output:**
Enter Username: police	Name: Kunal Sharma
Enter Password: password123	Date of Birth: 20/02/2002
Enter the licence number you want to verify: XYZ456	

Test Case 2:

Input:	**Output:**
Enter Username: police	Name: Sarvagya Kaushik
Enter Password: password123	Date of Birth: 10/02/2001
Enter the licence number you want to verify: ABC123	

Questions and Answers

Question 1: What is the information to be supplied by the driver to the police officer? And what information is sought and obtained by the police officer from the server in the transport authority?
Answer: The driver should provide their driver license(which contains driver information and license number) to the police officer. The police officer will send the license number to the service provider server for the verification. After verification the police officer receives the personal information of the

driver which includes driver's Name, Date of Birth and the photo of the driver.

Question 2: Would you need a central server that has the correct and complete information on all drivers and the licenses issued to them?
Answer: Yes, a central server is essential for managing and providing access to correct and complete information on driver licenses, ensuring the integrity and efficiency of license verification processes.

Question 3: In what way are digital signatures relevant?
Answer: Digital signatures are relevant in many ways which are as follows:
1. Authentication: Digital signatures provide a way to authenticate the origin and integrity of data. In the scenario described, digital signatures are used to verify that the data received from the server (such as license information) has not been tampered with and indeed originated from the trusted source.
2. Data Integrity: Digital signatures ensure that the data has not been altered or modified during transmission. By signing the data with a private key and verifying it with a corresponding public key, the recipient can be confident that the data has not been tampered with.
3. Non-repudiation: Digital signatures provide a way to prove the authenticity of a message and the identity of the sender. Once a digital signature is applied to a message, the sender cannot deny sending it. This property is important in legal and transactional contexts, such as verifying the authenticity of license verification requests and responses.
4 Secure Communication: Digital signatures enable secure communication between parties over untrusted networks. By using asymmetric cryptography, parties can securely exchange messages without the need for a pre-shared secret key.

Question 4: Does one need to ensure that information is kept confidential? Or not altered during 2-way communication?
Answer: Yes, it is crucial to ensure that information is kept confidential and not altered during 2-way communication because sensitive information should be accessible only to authorized parties and should be protected from unauthorized access.

Question 5: Which of these, viz. confidentiality, authentication, integrity and non-repudiation is/are relevant?
Answer: Confidentiality, Authentication, Integrity, and Non-repudiation are all relevant security properties in the interaction between the police officer and the transport authority server.
1. Confidentiality: It is relevant because the personal data exchanged between the police officer and the transport authority server, such as name, date of birth, and images of license holders, are sensitive and should be kept confidential to prevent unauthorized access or interception by malicious actors.
2. Authentication: It is relevant to ensure that both parties (the police officer and the transport authority server) can verify each other's identities and confirm that they are communicating with legitimate entities. This helps prevent unauthorized access and ensures that the information exchanged is between trusted parties.
3. Integrity: It is relevant to ensure that the data exchanged between the police officer and the transport authority server has not been altered or tampered with during transmission. This ensures that the information received by both parties is accurate and trustworthy, preventing unauthorized modifications that could compromise the validity of the data.
4. Non-repudiation: It is also relevant to ensure that neither party can deny their actions or the authenticity of the information exchanged. Digital signatures are used to provide non-repudiation by ensuring that messages are signed by the sender and can be verified by the recipient.

Bonus Question: Is date and time of communication important? If so, how can that be obtained from a well-known server in a secure manner?
Answer: Yes, date and time of communication is important to avoid replay attack. Date and time can be obtained from a well-known server in a secure manner by encrypting the messages using a well known asymmetric cryptographic algorithm(like RSA) and the server should also digitally sign the encrypted message to ensure authenticity and message integrity.