Network and Internet Security Assignment

Gabriel Marcus
University of Cape Town
Cape Town, South Africa
MRCGAB004@myuct.ac.za

Owethu Novuka
University of Cape Town
Cape Town, South Africa
NVKOWE001@myuct.ac.za

Shivaskar Naidoo
University of Cape Town
Cape Town, South Africa
NDXSHI026@myuct.ac.za

1 SECURITY MEASURES

Data security forms an integral part of the design of our cryptosystem. In the development of the cryptosystem, our objective is to achieve the following security measures to ensure Pretty Good Privacy:

(1) Confidentiality
(2) Integrity
(3) Authenticity
(4) Non-repudiation

The measures taken to ensure the security of messages transmitted within our cryptosystem are outlined below.

1.1 Confidentiality

Confidentiality is achieved through the implementation of assymetric and symmetric encryption we made use of two cryptographic algorithms to do this. AES and RSA encryption. The generate_confidentiality() achieves message confidentiality by implementing these cryptographic algorithms.

1.1.1 Asymmetric Encryption. RSA algorithm: RSA/ECB/PKCS1Padding is used. The RSA alogorithm encrypts the secret key with the receivers public key. which ensures there is confidentiality of the messages exchanged between the two parties as only the receiver can decrypt the secret key using their private key. The rsa_encrypt_message() function implements this. This ensures that only the holder of the recipient's private key can decrypt the secret key.

1.1.2 Symmetric encryption. AES Algorithm: AES/CBC/PKCS5Padding is used. we use PKCS7Padding which is equivalent to PKCS5Padding but allows chunks to be greater than 64 bits. The AES algorithm encrypts the data (i.e the photo) being sent with the session key, ensuring only the individual we want to receive the message can decrypt it. Symmetric encryption using the aes_encrypt_message() function encrypts the actual message exchanged between Alice and Bob using a secret key. The choice for AES encryption for the encryption of the message was chosen for its computational efficiency in generating the cipher text compared to using RSA.

1.2 Integrity and Authentication

The generate_signature() function takes the sender's private key, message, and the recipient's public key as arguments. To ensure the integrity of the messages, a message digest of the message is generated using the SHA-256 hash function. The message digest is signed using the sender's private key to ensure the integrity and authenticity of the message. This is used in confirming the user who sent the message is actually who they say are as the receiver will need to decrypt the signed message digest using the senders public key.It also allows us to know if the message was tamopered with. These measures allow the receiver to verify that the messages sent have not been altered during transmission and the authenticity of the sender can be verified.

1.3 Non-repudiation

Non-repudiation is achieved through digital signatures. The recipient verifies that the message was signed by the sender by using the sender's public key to decrypt the signed message digest.

The verify_signature() function which takes the sender's public key, signed digest and the message received as arguments verifies that the sender signed the message. If the function returns true, then the receiver can verify that it was that specific person who sent the message and that the message was not tampered with. It checks for tampering by comparing the a hash of the message to the digest.

2 KEY MANAGEMENT

Our key Management involved the generation and exchange of private/public key pairs, the use of a trusted Certification Authority (CA) for key validation, and the generation of shared secret keys. Both Alice(Client 1) and Bob(Client 2) generate their RSA key pairs, consisting of a private key (kept to themselves) and a public key (shared with others). To prevent impersonation, our system relies on a trusted CA to validate the public keys by issuing digital certificates that bind the public key to the owner's identity and are digitally signed by the CA. Alice and Bob exchange certificates, and using the CA's public key they verify the authenticity of the received certificate and individual. Once Alice and Bob have exchanged and validated each other's public keys, they can securely generate shared secret keys for symmetric encryption, which is more efficient for encrypting large amounts of data. The shared secret key is encrypted with the recipient's public key before being transmitted, and the recipient decrypts it using their private key, ensuring that only the intended recipient can access the shared secret key.

3 MESSAGE COMPOSITION

The structure of the message is constructed within the constuct_pgp_message method found in the message_utils.py program. This method encapsulates the steps involved in constructing a message that includes the name of the file, timestamp of when the message was created as well as the caption and image data.

The selected image is encoded using base64 to convert it to a string format. This string conversion enhances the ability to transfer and store within a JSON object. The file name and timestamp are added to the header of the message to identify image file by name, as well as to capture time of the message creation. Information about the caption and image data is stored in metadata.

The message structure is illustrated as follows:[1]:

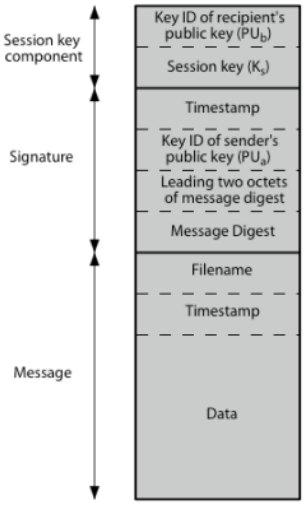

Figure 1: PGP message composition

4 CRYPTOSYSTEM DESIGN

The application makes use of the Pretty Good Privacy(PGP) cryptosystem. The PGP makes use of the aforementioned security features.

The flow of the encryption of PGP is:

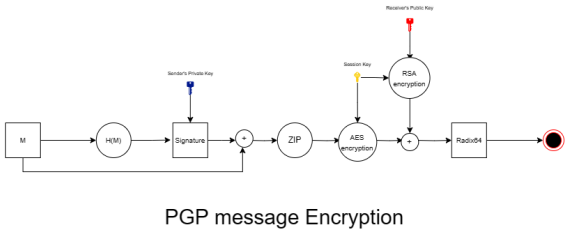

Figure 2: PGP message encryption

and the decryption procedure used was:

[1]Image taken from CSC4026Z email security lecture slides.

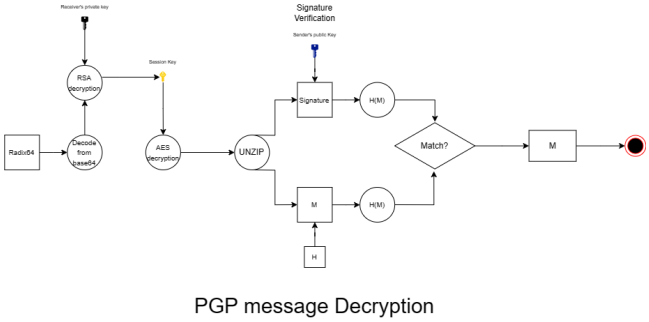

Figure 3: PGP message decryption

5 NETWORK EXPLANATION

This Network makes use of a Client-Server architecture. All encryption and decryption is done on the clients side. The Server is an intermediary who just passes the data to the other client. The network's security relies on a mix of symmetric (AES) and asymmetric (RSA) encryption, making use of the functionality of Pretty Good Privacy (PGP).

Here are the steps we follow to establish a secure communication channel:

1. Key Generation and Certificate Exchange:
Both Alice and Bob generate their RSA key pairs and receive a certificate from a trusted Certification Authority (CA). This certificate contains their public key and is signed by the CA, ensuring its authenticity.

2.Key Exchange:
Both Alice and Bob start with their own certificates and the CA Certificate. They join the network, and Alice sends their certificate to Bob. Bob receives the certificate, uses the CA's public key from the CA Certificate to verify Alice's certificate, and extracts Alice's public key. Automatically, Bob sends their certificate to Alice upon receiving Alice's certificate. Alice then verifies Bob's certificate using the CA's public key and extracts Bob's public key. Finally, both parties store each other's public keys, completing the key sharing process.

Communication:
Alice and Bob can exchange messages securely. Their communication involves sending data (like text and images) that are encrypted using the shared secret key (AES) and signed using their private RSA keys to ensure both confidentiality and integrity.

6 DIAGRAM EXPLAINING HOW KEYS ARE SHARED

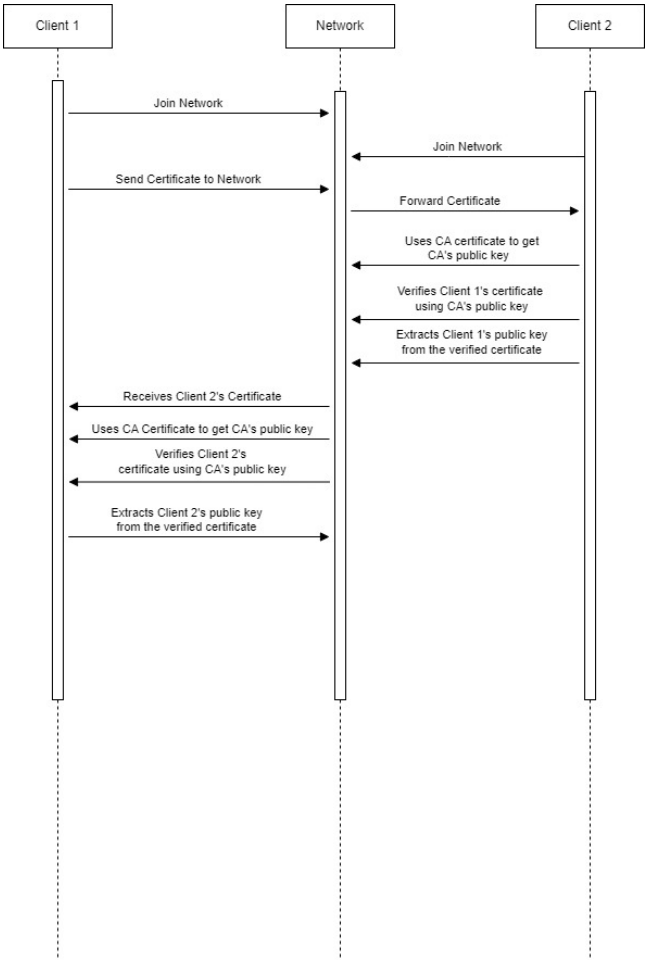

Figure 4: Sequence Diagram explaining how keys are shared

Process:

1. Initial Setup:
Both Clients have their own certificates. Both Clients have the CA Certificate.

2. Joining the Network:
Both Clients join the network.

3. Sending Certificate:
Step 1: Click "Send Certificate" button.
Step 2: Client 1 sends their certificate to Client 2.
Step 3: Client 2 receives Client 1's certificate.

4. Verifying Certificate:
Step 1: Client 2 uses the CA Certificate to get the CA's public key.
Step 2: Client 2 verifies Client 1's certificate using the CA's public key.
Step 3: Client 2 extracts Client 1's public key from the verified certificate.

5. Automated Return Certificate:
Step 1: Upon receiving Client 1's certificate, Client 2 automatically sends their certificate to Client 1.
Step 2: Client 1 receives Client 2's certificate.

6. Verifying Return Certificate:
Step 1: Client 1 uses the CA Certificate to get the CA's public key.
Step 2: Client 1 verifies Client 2's certificate using the CA's public key.
Step 3: Client 1 extracts Client 2's public key from the verified certificate.

7. Completion:
Both Clients have each other's public keys stored.

7 GRAPHICAL USER INTERFACE

This Asssignment uses interactive graphical user interface. The interface allows users to send both messages and images with captions

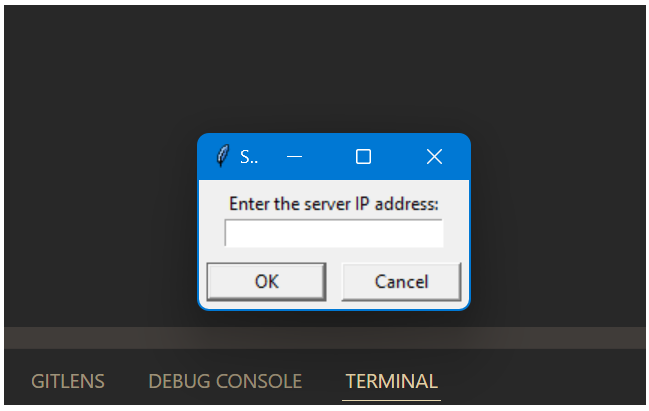

Figure 5: Prompts the client to enter the IP address of the server

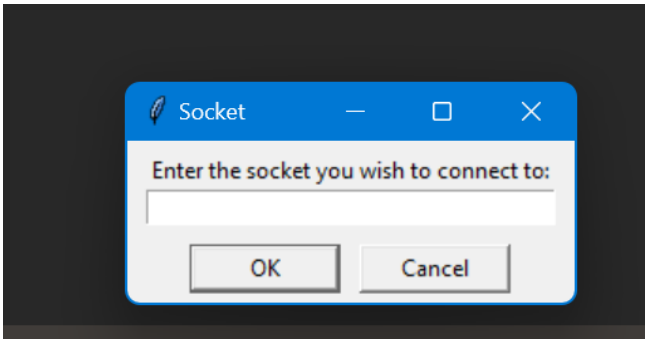

Figure 6: Prompts the client to enter the port to which they wish to connect

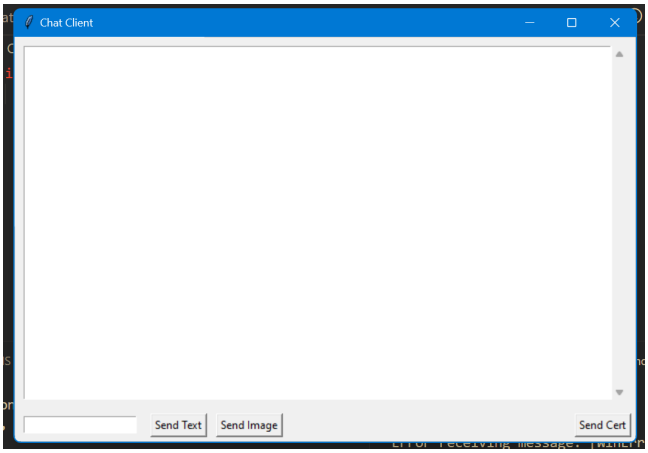

Figure 7: The Graphical display of the program

The Program Can send both messages and images(with captions).

To run the program:

(1) start up the server enter the socket you would like to use.
(2) start the client and enter the IP and port
(3) start another client and enter the IP and port
(4) exchange certificates
(5) Use the porgram

8 TESTING

Various testing methods were made use of:

(1) Unit Testing
(2) Functional Testing
(3) Black Box Testing (with help from Gabriels friends)

All Unit tests can be found in src folder with the files labelled with the prefix z. Proof of Functional testing can be given(A video was recorded).

Tests for Certificate Handling was implemented which includes:

(1) Generation of RSA key pairs.
(2) Creation of certificates.
(3) Sending and receiving certificates between clients.
(4) Verification of received certificates using the CA's public key.

Tests for GUI Functionality includes:

(1) Prompting the client to enter the IP address of the server.
(2) Prompting the client to enter the port to connect.
(3) Displaying the graphical interface of the program.
(4) Ensuring the correct functionality of buttons and user inputs within the GUI.

Unit Tests for RSA Encryption includes:

(1) RSA encryption of messages using the recipient's public key.
(2) RSA decryption of messages using the recipient's private key.
(3) Verification of the integrity and authenticity of encrypted messages.

Functional testing involves testing the program to ensure that it behaves as expected with its specified functionality. This type of testing verifies that the system performs correctly according to the functional requirements.

Black box testing is a method of testing that examines the functionality of an application without peering into its internal structures or workings. This testing method is typically performed from the user's perspective. In this project, black box testing was conducted with the help of Gabriel's friends, who tested the system without any knowledge of its internal code.