NIS Project Report

By Anele Dlamini(DLMSIL008), Fezeka Nzama(NZMFEZ001), Kunta-Kinte Ngcobo(NGCTHA030), Lusanda Vilane(VILLUS002)

Communication Implementation

Communication Process

The communication system implemented makes use of a client-server architecture and TCP connections enabled by java sockets.

Server-Side Architecture

- Server Class: The server is implemented using an instance of a Server class. The Server class when instantiated makes use of a java ServerSocket to listen for new connections from clients. When a new connection is established a new instance of a ServerWorker inner class is instantiated on which a client communicates with the server whilst leaving the main Server thread open to listen for new connections.
- ServerWorker Class: The ServerWorker class extends the java Thread class. This class handles all communications between a single client and the server.
- UserClient Class: The UserClient class is used to verify the username and password allowing a client access to the communication environment offered by the server. Passwords for all registered clients are saved in the ServerCoolThings.txt as salted SHA-256 hashes. Saved passwords are compared with entered passwords at login.

Client-Side Architecture

- Client Class: Each client is an instance of the client class. An instance of the client class makes use of a java socket to connect with the Server. The two client instances used in this system have the usernames Alice and Bob, with passwords Apass for Alice and Bpass for Bob. All message encryption and decryption occur in this class.

Certification and Key Exchange Process

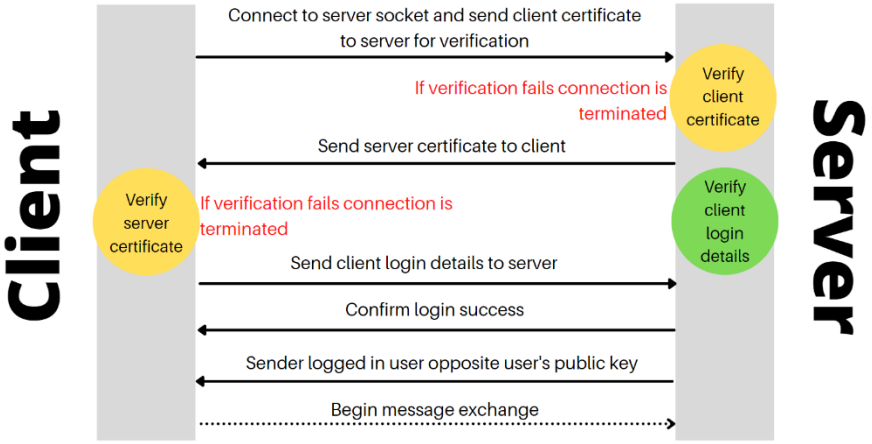


Figure 1: Certification Process

The certificates and keystores of Alice, Bob, and the server as well as the Certificate Authority's (CA's) certificate have been previously generated and signed using the CA's private key. The certificates and keystores of Alice and Bob are saved in the Client package folder, with the CA's certificate. The certificate and keystore of the server, as well as the CA's certificate are saved in the Server package folder. Once a connection between a client and a server is established, the server and client exchange certificates. The client verifies the server's certificate using the CA's public key, which is extracted from the CA's certificate and the server does the same with the client's certificate. If a certificate is found to be unsigned by the CA, the connection is terminated. Only after certificate verification is a client's username and password tested, thus allowing the client to login to the system. The methods used in this process are handleClientCertification() in the server and handleCertification() in the client.

Key Exchange Process

Once a client (e.g., Bob) is logged into the system, the server sends to Bob Alice's public key and vice versa. These public keys are signed by the server prior to sending to ensure the integrity of this public key. Prior to sending a message, the sender generates a new shared key. This key is used to encrypt the message using the AES algorithm. This key is then encrypted using the receiver's public key and the RSA encryption algorithm.

Security Implementation

Message Integrity

Message integrity is about ensuring that the received message is unaltered. Hashing is used to do this.

- Sender Side: The image and caption or message being sent is hashed using the sha256() method. This hash is concatenated onto the original string representation of the message being sent, which is later encrypted and transmitted.
- Receiver Side: After decrypting the message, the receiver calculates its own hash of the received message and compares this hash with the hash received as part of the message. If a discrepancy arises between these hashes, the receiver knows that message has been altered. This comparison happens in the decodeString() and decodeText() methods in the Client class.

Message Authentication

Message authentication is about ensuring that the message received originated from the expected sender. Authentication is done using the public key pair.

- Sender Side: Following the generation of a message hash, a signature is generated on this hash using the sender's private key. This signature is concatenated onto the message string along with the hash. The sign() method in the Client class is used to do this.
- Receiver Side: Once the hash and signature are separated from the main message the signature is verified using the sender's public key, returning true if the signature belongs to the sender. This is implemented using the verify method in the Client class.

Message Confidentiality

Message confidentiality is about ensuring that only the authorized receiver can read a message sent to them. The RSA and AES encryption algorithms are used to do this.

- Sender Side: The message, including the hash and signature are encrypted using a session key and the AES algorithm. This session key is then encrypted using the receiver's public key and the RSA encryption algorithm. The AES encrypted message and RSA encrypted key are combined and sent to the receiver. This is done using the encryptAES() and encryptRSA() methods in the Client class.
- Receiver Side: On receipt of the message the receiver separates the encrypted AES session key from the encrypted message. The encrypted AES session key is then decrypted using the receiver's private key. It is this session key that is then used to decrypt the actual message.
- Server side: To ensure transmit level confidentiality when messages are sent from the sender to the server, the message meta data including the receiver's name, as well as the encrypted message are combined and encrypted using the RSA algorithm and the server's public key as well. This method is repeated when transmitting from the server to the receiver.

Overall System Design

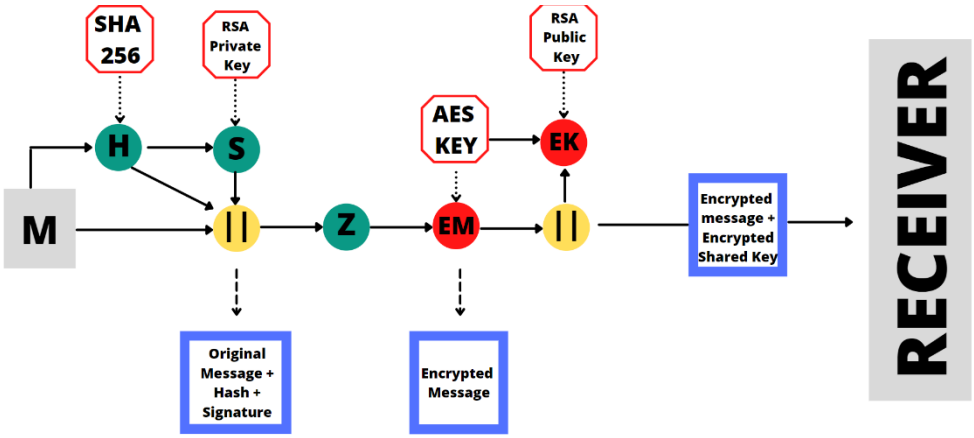


Figure 2: Sender Side Encryption System

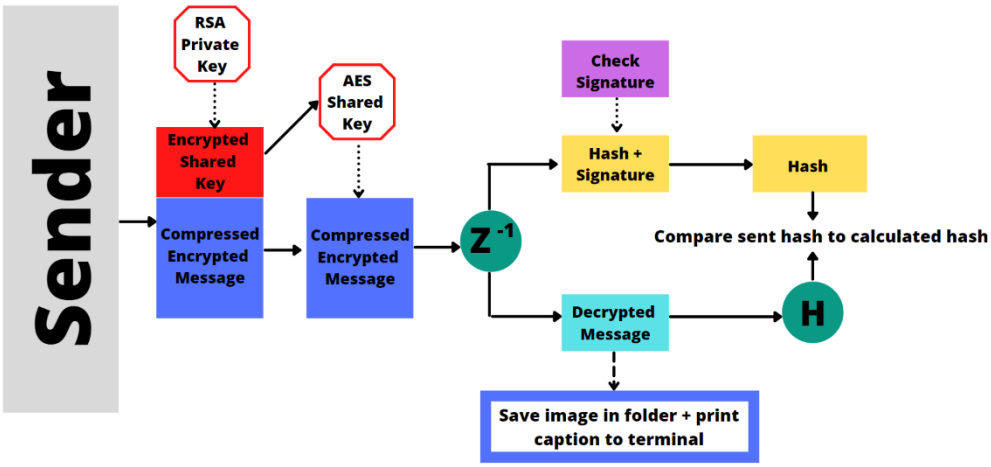


Figure 3: Receiver Side Decryption System

Compression

The messages are compressed before they are encrypted in the compress() function. The compress function compresses messages using GZIP class provided by Java. The compression helps the chat application save transmission time, it also makes the messages smaller and therefore limit size on storage and size issues that may occur during encryption (images too are compressed). These two benefits of compression further enforce the cryptographic services we provided in the chat application. On the receiver side, they decompress (using the decompress() method) incoming data before decrypting the messages.

Shared Key Usage

AES encryption is used to encrypt every message shared between clients. New shared keys are generated with every message and image sent to prevent any interception to access previously sent messages since those keys are discarded as soon as the receiver decrypts the incoming message.

For each message encrypted, the shared key used is encrypted using RSA encryption and the sender's private key. The AES ciphertext and RSA ciphertext are sent together to the receiver to decrypt the AES key using RSA decryption and the sender's public key, they then use the key to decrypt the AES ciphertext and get the message in plain text.

Encryption ordering and justification

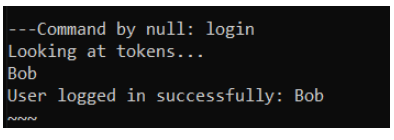
The encryption and decryption processes followed are detailed in Figure 2 and 3 above, with the AES and RSA algorithms both used in the process. The AES encryption algorithm can encrypt files of a larger size than the RSA algorithm, whilst also being the current standard for shared key encryption. As such AES is ideal for the encryption of message. This encryption is then followed by the encryption of the shared key using the RSA algorithm, as asymmetric encryption algorithms provide more security than shared key encryption.

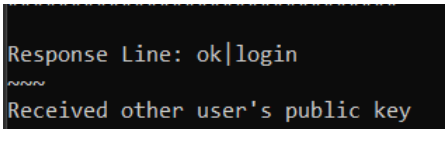
This order of encryption, which starts with AES encryption followed by RSA encryption, allows for two levels of security, with the more robust form of security being used as the outermost defense level. Simultaneously, this order of execution leverages the capacity advantages provided by AES, in a way that would not have been possible using the alternative order of execution.

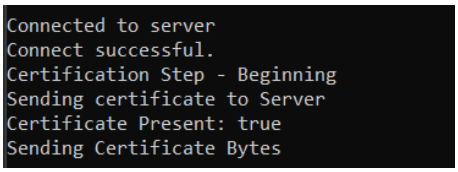
Evidence of Testing

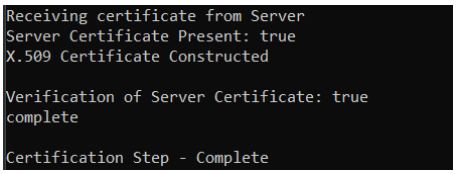
Connection

For testing we are printing the tracing statements required of us from the assignment brief. When the server is running and clients connect to it, we print the certificates exchanged between the server and each client that connects to the server. Both the server and client terminal print these certificates. The server also prints the client (s) certificates upon connection. When clients receive each other's public keys from the server, they print them out on their terminals. See the screenshots of our testing and the tracing steps below.



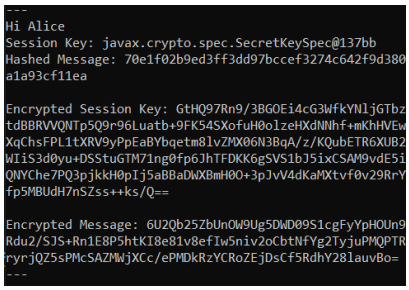


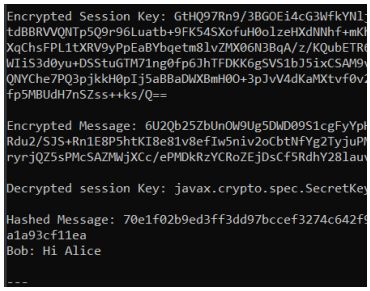




Text messaging

When one client, for instance Bob, sends a text message to another client, for instance Alice, Bob encrypts the text in the encryption order we specified earlier. The shared key used when encrypting with AES is generated by Bob, encrypted using RSA and then sent to Alice along with the AES ciphertext, hash and signature of the hash. For tracing steps, we printed the encrypted message, session key, hashed message on the sender's (Bob) terminal. On the receiver's (Alice) terminal we print out decrypted message and decrypted session key. When a client sends a message, the server prints out the command (message) by the sender, the encrypted message and encrypted session key. See the screenshots of the two clients and the server terminals below.





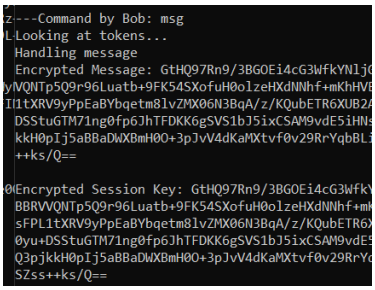


Image and Caption messaging

When a client sends an image to another client, the image and caption are processed (encoded, compressed, etc.) as highlighted previously. On the sender terminal we print the hashed image details, the session key and the encrypted message. On the receiver side we print the encrypted message, encrypted session key, decrypted session key, the hashed image details, who the sender is, the filename and finally the image captions. When a client sends an image, the server prints out the command (image) by the sender, the encrypted message and encrypted session key. See the screenshots of the two clients and the server terminals below.