CS6349 - Network Security - Project Report

Varsha Hosamath
Kannan Prasshanth Srinivasan

The aim of this project is to design and implement a chat application that has various security features that preserve authentication, confidentiality and integrity. The application consists of a single server and multiple clients, which can communicate with each other. The details of the architecture and protocols involved are given below.

Design:

Each of the functionalities of the application were designed with authentication, confidentiality and integrity in mind. They are described below:

Login:

The below protocol describes the process of the authentication of the user by the server:

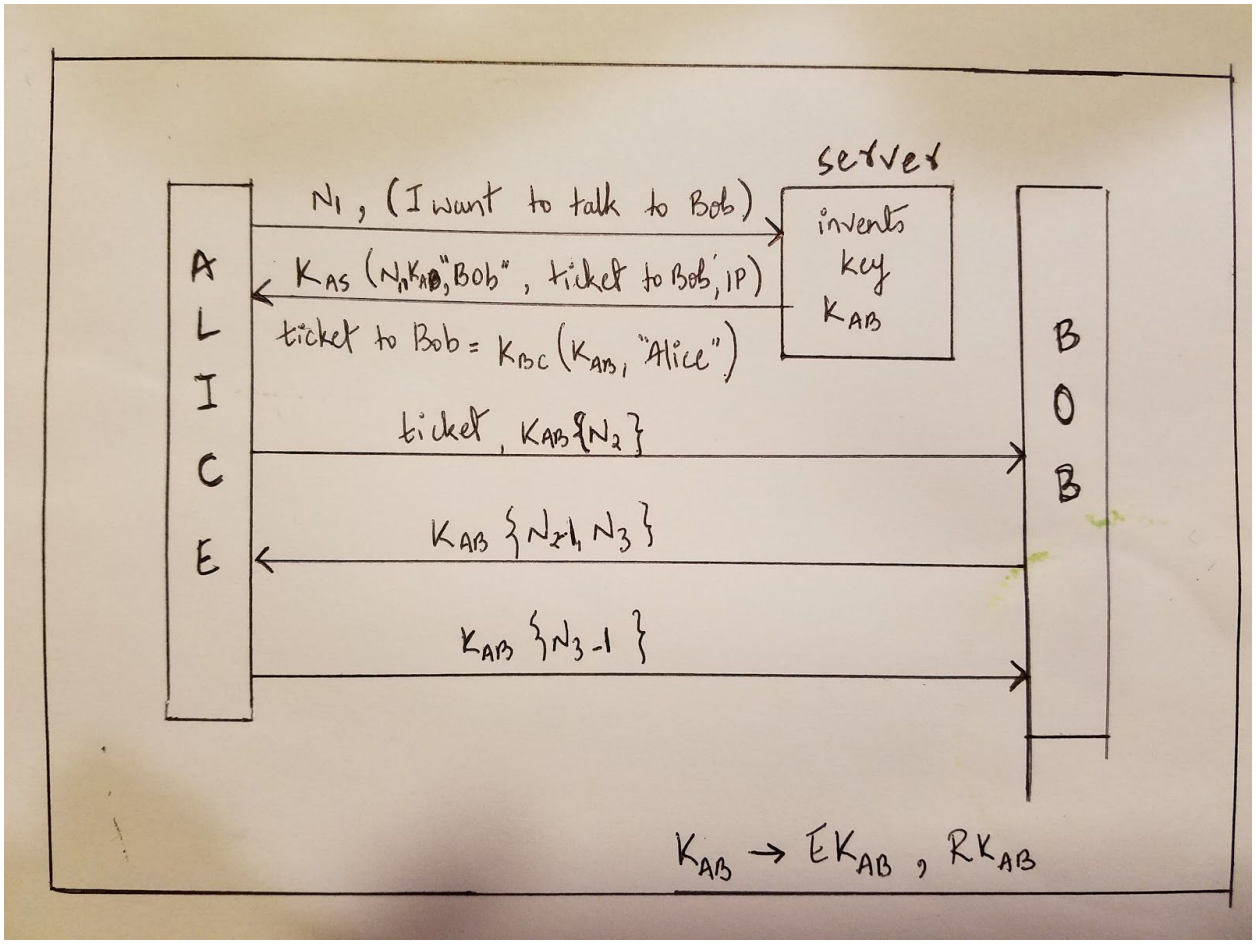
1. The user creates a secret key and sends his or her username, password, IP address and the key to the server, encrypted with the server's public key.
2. The server decrypts the user's message using its private key, stores the user. Thus, at the end of the login authentication protocol, the user and the server have a unique shared secret key.

This exchange is secure because the message sent by the user is encrypted using the server's public key, which cannot be decrypted by anyone but the server, since it is the only one in possession of the corresponding private key. The private and public RSA key pairs were generated in advance, and made accessible to the clients and the server.

Session creation:

The diagram that follows describes the protocol for session creation between users:

1. Alice sends a random nonce N1 along with a request for a ticket to talk to Bob to the server, this would be encrypted using the public key of the server.
2. The server generates a shared secret key for Alice and Bob, and sends it back along with the ticket to Bob, the nonce N1 and Bob's IP address back to Alice, encrypted with the shared key that it had established with Alice during login. The ticket to Bob, contains the phrase "Alice", along with the shared key that the server generated with Bob during Bob's login.
3. Alice then sends to Bob the ticket along with a random nonce N2 encrypted with the shared secret key between Alice and Bob.
4. Bob sends back the decremented nonce N2 - 1, and another nonce N3, encrypted with their shared key.
5. Alice then sends back to Bob, the decremented nonce N3 - 1, encrypted with the shared key.



This is basically a modified Needham-Shroeder protocol, to suit the requirement of the users not owning a public/private key pair. Thus, it utilizes the shared key established with the server by the users during login. During the final implementation of the project, the last two messages were discarded, as authentication of Alice was deemed sufficient due to possession of the ticket.

Heartbeat:

It is possible for users to go offline without sending a logout message to the server, such as when their network goes down. Thus, it is necessary to have another mechanism to detect the activity of the users. This was done by having the individual clients send periodic heartbeat messages to the server, the server decides that a particular client is offline if it doesn't receive heartbeat messages for a set number of intervals in a row. For the purpose of implementation, this was 90 seconds, but could be set to any time. The messages were encrypted using the shared key, and have increasing timestamps, implemented by getting the current system time, which is the number of milliseconds from the unix epoch.

Integrity:

In order to preserve both privacy and integrity, all messages between the clients were doubly encrypted as follows: a strong symmetric cipher, 128 bit AES was used for encryption for privacy, while a weaker cipher, 56 bit DES was used for the message digest, thus ensuring integrity. The message digest used was SHA256.

Implementation:

The language used for the implementation is Java, for generating the secret keys, the javax.crypto library was used. The sockets library was used for establishing connections between the clients and the servers.

Challenges faced:

Some of the challenges faced are mentioned below:

1. Research was required to generate the proper bytestreams from the cryptographic functions in order for them to be compatible with sockets to be sent across the network.
2. Ensuring that all facets of the protocol run simultaneously without interfering in other functionality. For example, clients have to listen for new requests and send out heartbeats while simultaneously providing the users the opportunity to connect to other clients via the designed protocol. This required careful consideration of concurrency and network connection issues.

Project Contributions:

The majority of the work was divided into two main categories and divided as follows:

1. Server and Cryptographic functionality - Kannan Prasshanth Srinivasan
2. Client functionality - Varsha Hosamath