CPSC 452 Cryptography
Final Project - Secure Chat
Spring 2016

Duy Do
Holly Ho
Billy Saysavath
Tevin Vu

Abstract

What's Chat is a chat application in the terminal that allows users to securely connect to a chatroom and send messages to whoever is online. Some of its salient features include authenticating users and user credentials, allowing users to check online users, inviting online users to the chat, sending messages to other online users, checking what members are in the chat session, and quitting the chat.

Introduction

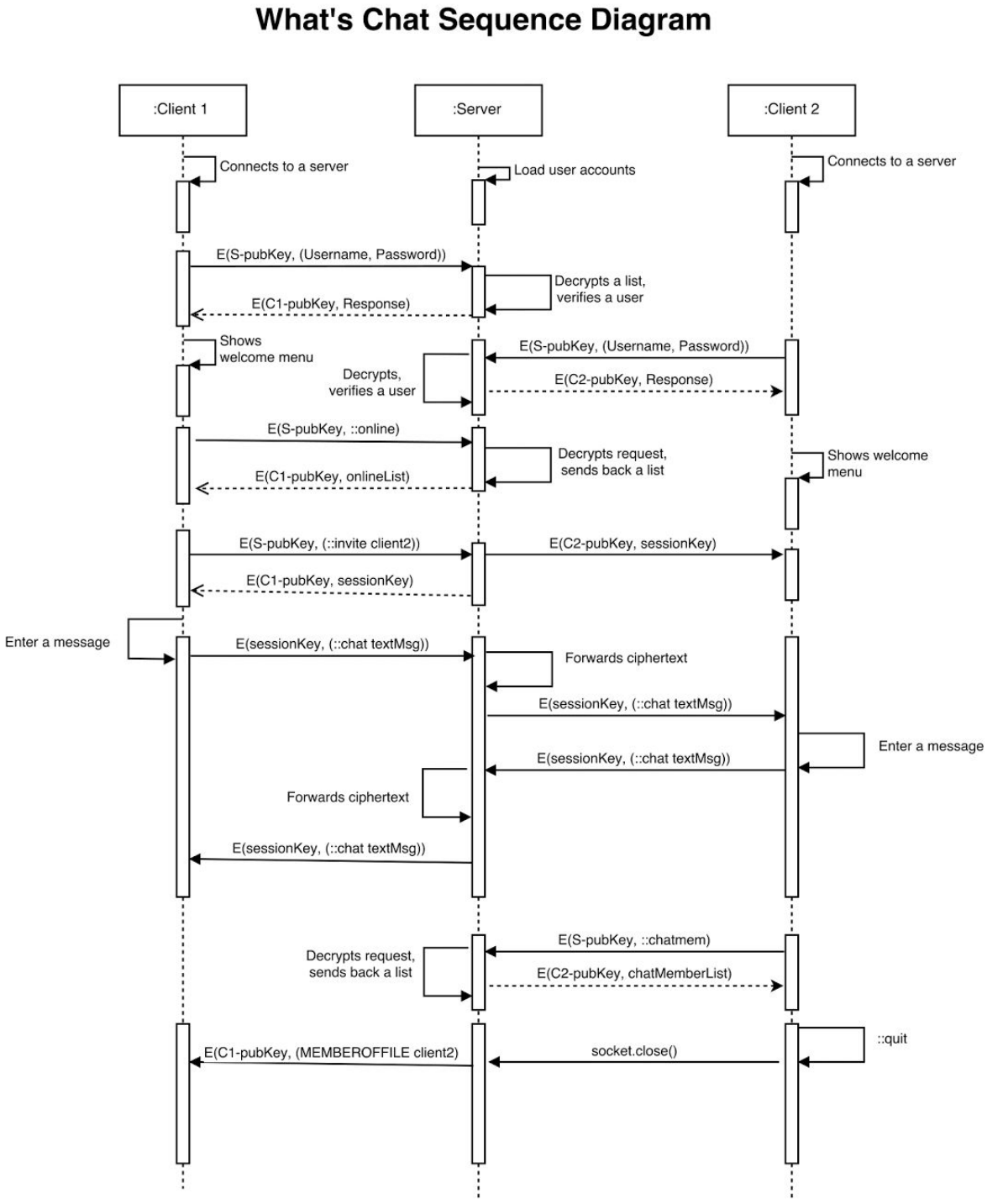
What's Chat has many services to ensure the security of our chat application.

One first service that we provide is that we encrypted all of our list of accounts in our accounts.json file. This way, our user accounts information is never in plaintext so that nobody can easily know our users' credentials. It is only when the server is running that we decrypt the accounts.json file to get this information.

Another service that What's Chat delivers is the secure exchange of requests from the client and the server. We use RSA for this, utilizing the server's public and private key along with the user's public and private key.

The last service we provide is that we encrypt all the chat messages sent by the user using AES. A symmetric key is provided by the user and all messages that are sent are encrypted by this symmetric key. This is done so that the server will never have the plaintext messages that is sent by the user.

Design


What's Chat Sequence Diagram
:Client 1
:Server
:Client 2
Connects to a server
Load user accounts
Connects to a server
E(S-pubKey, (Username, Password))
Decrypts a list, verifies a user
E(C1-pubKey, Response)
Shows welcome menu
E(S-pubKey, (Username, Password))
Decrypts, verifies a user
E(C2-pubKey, Response)
E(S-pubKey, ::online)
Decrypts request, sends back a list
Shows welcome menu
E(C1-pubKey, onlineList)
E(S-pubKey, (::invite client2))
E(C2-pubKey, sessionKey)
E(C1-pubKey, sessionKey)
Enter a message
E(sessionKey, (::chat textMsg))
Forwards ciphertext
E(sessionKey, (::chat textMsg))
Enter a message
E(sessionKey, (::chat textMsg))
Forwards ciphertext
E(sessionKey, (::chat textMsg))
E(S-pubKey, ::chatmem)
Decrypts request, sends back a list
E(C2-pubKey, chatMemberList)
::quit
E(C1-pubKey, (MEMBEROFFILE client2)
socket.close()

Security Protocols

To ensure that the user has the correct credentials to log into the chat application, we use RSA by encrypting the request and the account information with the server's public key. Once the server gets this request, it will decrypt the request and the account using the server's private key. After processing whether the user's credentials are correct, the server will encrypt the response using the user's public key. The client will receive the response and decrypt it using the user's private key to see whether the user's credentials are correct to authorize the user to use the chat application. This achieves authenticity because we verify whether a user has the correct credentials to use our chat application.

We used the RSA protocol for the secure exchange of session keys. We do this when a user requests to invite another user into a chat session. When this is done, the server sends both users the session key encrypted with the user's public key. This is important for confidentiality because the user must have the correct private key to decrypt and get the correct session key. If they do not have the correct private key, they will not have the correct session key for to read the messages send by the other users.

We also use the RSA protocol to decrypt and encrypt requests from the user. We do this by encrypting the request using the server's public key on the client side. When the server gets this request, it will decrypt the request using the server's private key to see what the request is. The server will send back the response by encrypting with the user's public key. When the client gets the response, it will decrypt the response using the user's private key. This ensure confidentiality of the requests because the clients must have the correct private key in order to decrypt the responses that are sent from the server.

We use AES to encrypt and decrypt the chat messages that the user sends on the client side. All the messages exchanged during the chat are encrypted using the symmetric key and are delivered to all users participating in the chat. This is important for confidentiality because only those that have the correct symmetric key will be able to decrypt the messages. This is important because only those that should be able to read the message should have the correct symmetric key to decrypt the message.

Implementation

We decided to use python as our language because we were familiar with using sockets from python because of our experience from our CPSC 471 class. Our platform was the terminal because we had made something very similar in our CPSC 471 involving the client and server side code and thought we could do something similar for a chat.

We used the rsa library for the encryption of the request and response messages. This is because the example that was given in class of rsa used the rsa library.

We used the Crypto.Cipher library of AES for the encryption of the chat messages. This is because the DES example that was given in class used the Crypto.Cipher library and this library also had a module for AES.

We also used the cPickle library because you can only send a string through a socket. The cPickle library helps to encode an array into a string so that we can send it through the socket.

Conclusion

What's Chat is a terminal chat application that allows user to securely send messages to one another. After a user was successfully logged in, users would be able to check who was online, invite other online members to the chat session, check who was in the chat, send messages to the chat, and quit the chat. We were able to use both AES and DES in our chat application to ensure the security of our requests, responses, and chat messages.

Video

Video link to demo:
https://www.youtube.com/watch?v=ESbOMOxFDgg