

Protocol for Secure Instant Messenger

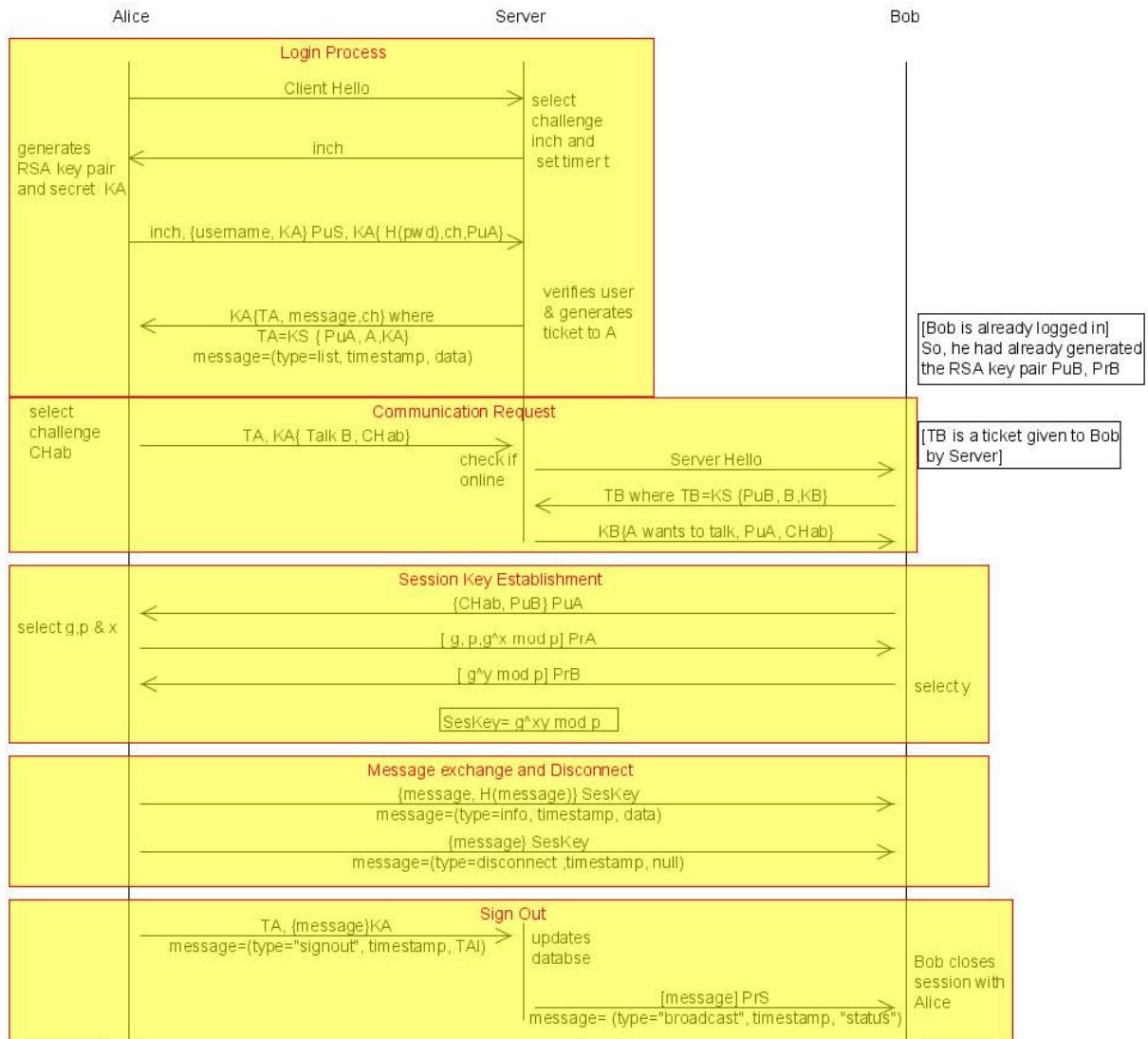
Team Members

Ameya Karkhanis
Ashish Musale
Bhavin Khivesara
Clive D'souza

Introduction

The secure instant messenger (IM) will have two primary entities, a centralized server and users. The protocol for this IM will run over TCP protocol. This ensures reliability of packet delivery. The server maintains a database of the user information. The Clients log in to the service using their username & password. The server provides the user with list of currently online users as soon as the user logs in. The user requests the server to initiate a conversation with another online user. The server is responsible for connecting the two users. After the connection, the message exchange process is based on peer to peer communication. The user can disconnect from the conversation when he wishes to end the conversation. Users inform the server with the sign out message as he disconnects from the service.

Secure Instant Messenger Protocol



Legend :

=====

- KS = Server's secret key
- SesKey = Key established for message between peers
- TA, TB = Ticket from Server to the users
- PrA, PuA = RSA key pair for user A (Pr = Private, Pu = Public)
- PrB, PuB = RSA key pair for user B (Pr = Private, Pu = Public)
- Prs, PuS = RSA key pair for Server (Pr = Private, Pu = Public)
- ch, CHab, inch = challenge
- H = Hashing function (e.g. SHA1)
- A, B = IP address & port of A and B respectively
- $\{m\}n$ = Asymmetric encryption of message m with key n
- $n\{m\}$ = Symmetric encryption of message m with key n
- $\{msg\}k$ = message (msg) signed by private key (k)

Authentication Messages:

1. Alice sends a message to the Server to log into the machine.
2. Server responds by sending an initial challenge (inch). This challenge is obtained by applying a secret function on source IP of received first message packet. This is used to prevent DOS attack.
3. Alice generates a RSA key pair and a secret session key KA. In this message Alice sends:
Received Inch
Username and Session Key KA encrypted with server's Public Key
Hash of Password, challenge for server and her public key encrypted by this session key
4. Server checks the challenge. If the IP was spoofed, then the user will not be able to respond with the challenge. Server verifies if username exist in the database and he is not online and its password.
Server then generates a Ticket to Alice (TA) containing the public key of alice, her identity and the session key encrypted by servers secret key KS. It also sends the list of all online users and returns the alice's challenge.
The entire response is encrypted with session key KA.

Communication Request:

The following exchange will be done when Alice has already finished his login procedure and wishes to start communication with any of the online members (e.g. Bob).

5. Alice requests the server to talk to Bob by sending the Ticket (T_A) server gave her along with identity of Bob (B) and a challenge for Bob (CH_{ab}). Alice encrypts the Identity of Bob and challenge by the session key KA.
On receiving this message the server checks if it is the valid ticket by decrypting it with his secret key K_S . If valid the server extracts the session key KA from ticket and decrypts the remaining message. The server then checks if the user's identity is same as the one from where the packet is received and the user is online user.
6. On successful validation the server sends a Server Hello message to Bob.
7. On receiving this message Bob will send the Ticket (T_B), issued to him by server when he was logged in.
Server will then decrypt the Ticket and find the session key used with bob KB. It also verifies if the packet has originated from the same address to prevent against replay attacks.
8. Server then sends the following message encrypted with session key of Bob (KB):
 - "Alice wants to talk"
 - Public key of Alice (PuA) extracted from the ticket received from Alice.
 - Challenge given by Alice for Bob (CH_{ab}).

After reception of this message Bob will start the Session Key Establishment phase.

Session Key Establishment:

9. Bob sends Alice the challenge given by Alice to the server for Bob and his public key (PuB), encrypted by public key of Alice (PuA). This ensures that only Alice knows the public key of Bob.
10. Alice gives Bob the information to derive the diffie-hellman key i.e. g, p and $g^x \text{ mod } p$ and signs it with his private key (PrA). This will assure that the message was generated by Alice. Alice will reuse the same g and p used during the login procedure.
11. Bob selects y and provides $g^y \text{ mod } p$ to Alice signed by Bob.
Now, both Alice and Bob share the secret Key $Seskey = g^{xy} \text{ mod } p$. Rest of the conversation between Alice and Bob takes place by encrypting all messages with Seskey. Seskey is only known to Alice and Bob. Server is not involved when both peers talk to each other.

Message Exchange & Log out Process:

12. Now messages are exchanged between Alice & Bob consists of two parts ie message & hash of the message. The message further consists of message type, timestamp & data (actual message information to be sent). We use HMAC-SHA1 to compute hash of the message. This entire message is then encrypted using the session key Seskey. Message type field is basically used to differentiate between messages like update messages, disconnect message & information message.

Disconnect: This is used to inform the other user that he wants to end the conversation.

Signout: This is used to inform the server that is wants to log out.

Information: This type is used to exchange information between two users

Broadcast: Used by server to inform all the active users if any user signs out or Signs in.

13. If the user wish to disconnect the conversation it sends a message to User B, with message type disconnect, timestamp & data field is empty for this message. It uses the session key Seskey to encrypt the message. At this point user A & user B forgets the Session Key used.
14. Now finally if the user decides to signs out from the service it just sends to the server, a ticket given by server in plain and a message with type as signout, timestamp and data field as Ticket encrypted with the session key established with server during login process.
15. Now the serverfirst decrypts the received ticket and finds the session key. Using the key it decrypts the rest of the message and finds out that the user wish to signout. It closes his connection with that user and sends a broadcast message to all active users that A has left the service and is offline. The message consists of message type as broadcast, timestamp & its mentions in the data field that A is offline. (Similarly a Broadcast message is sent when a new user logs in the service)

Features of the proposed protocol:

Perfect Forward Secrecy:

During the login procedure of the protocol the hash of password is encrypted with a randomly generated session key. After signout the client forgets the session key. So even if someone has recorded the past communication it cannot find the session key that was used.

Only when the server's private key gets compromised, attacker can find the session key used in past communication for login.

Perfect forward secrecy is also ensured during actual message exchange because of the use of key established by Diffie-Hellman Exchange.

So even though attacker might be successful in getting login credentials he cannot decrypt actual communication between Alice and Bob.

End-Point Hiding:

End Point hiding is achieved to certain extent in the proposed protocol.

Initially the client sends a Client Hello message to the server. So the attacker will come to know that the client wants to talk to the server. But the client hello message does not contain the identity of the client. The attacker will just have the knowledge of the source of the packet looking at the IP header of the packet.

The request for initiating the chat with any online user is sent to server by encrypting it with the session key. Only server will have the knowledge that "A is talking to B". Any attacker won't come to know about this communication.

So end-point hiding is achieved from external users.

Denial of Service:

When the client sends "client hello" message to the server it replies with a random initial challenge (inch). Now in reply client sends the 2nd message in the login process. Here the server expects the client has sent same initial challenge (inch).

This initial challenge is passed on both sides in open without encryption. This is included to detect any Denial of Service attack performed on server.

Possible attack:

- Attacker has spoofed the IP address and sent a client hello message. The reply from the server would be sent to the actual workstation having that IP address. So the attacker can't proceed with the 3rd message and try to load the server by making the server decrypt a random message. This is because attacker will have to send the same initial challenge in reply. If the server receives some random reply for the attacker, before proceeding with the decryption, it will check the received initial challenge in the message. If this value is not equal to the one sent then it has come from some random user. So the server will simply drop the connection.
- If the attacker has not spoofed the IP address then he can proceed with 3rd message in the login process and make the server decrypt the message. He can again open a new connection and perform the same 3 steps again and thus load the server. In such case if the server is receiving frequent client hello request from the same IP address it can simply ignore the 1st message. This will prevent from possible Denial of Service attack and the source of this attack could also be traced.

So the system is resilient to denial of service.

Protection against Weak Passwords:

A weak password is one which can be easily obtainable by performing a dictionary attack or being easily detected by a password cracking device or software. Our system sends the hash of the password.

Also the 3rd message in the login process is encrypted with an established session key. Getting this key is difficult for the attacker. So he cannot find hash value of the password. Also in the third message a random challenge is sent by the user.

So even if the attacker is recorded 3rd message it is very difficult to find the correct hash value of password by performing dictionary attack. The random challenge appended with the hash value makes the password strong.

For an online dictionary attack, the intruder will have to try all the possible passwords. Even if the attacker tries this, it would be of no use, since, after he tries the password the third time and still it fails, he will not be allowed to enter the password the fourth time as our system will block him and prevent him from entering the password. This protects our system from an online dictionary attack.

Thus the protocol protects against weak password.

Prevention of decryption between users by server:

In the proposed scheme, we see that initially user A and user B undergo a login process with server. After the login is complete, user A uses the ticket which he receives from server to initiate a conversation with user B. Then A & B exchange signed (with their respective private keys) Diffie Helman messages and calculate the session key K. The server cannot calculate this session key as it has no knowledge of the random x and y value used in Diffie Helman scheme. Hence the server cannot get the session key which is used to encrypt the messages when A & B begin communicating. Hence this prevents server from decrypting the communication between the users.

Also since the diffie-hellman exchange is signed by client server cannot perform an MITM attack and establish 2 different keys with the users.

Assumptions

- 1) Communication is done through TCP
- 2) Username and password are stored in the server's database
- 3) Users only know their username and the password.
- 4) The workstation has the knowledge of the server's public key.