College of Computer and Information Science
Northeastern University
CS4740/6740: Network Security

Secure Instant Messaging System

Architecture

The secure IM system consist one server and multiple clients.

1) A server keeps records of registered users, authenticate users to login (authentication), discover online clients (client discovery), acts as the key distribution center for clients to help establish session keys between clients (key distribution).
2) Multiple clients to use the secure instant messaging system.

Assumptions

- Users only need to remember his/her username and password
- Client workstation knows server's public key which is built in the configuration file
- All users are already registered and server keeps a list of all usernames encrypted with its public key and hashed password

Services

- DoS resiliency
- Perfect forward secrecy
- Identity hiding
- Mutual authentication
- Confidentiality
- Integrity
- Non repudiation
- Replay resistance
- Break in tolerance
- MITM resistance
- Dictionary attack
- Offline attack
- Impersonation

Authentication protocol (login)

a) $A \rightarrow S$: Request Login

b) $S \rightarrow A: \text{Cookie} - \text{Hash}(IP_A,\ \text{timestamp})$

c) $A \rightarrow S: \text{Cookie}, \{A\}_{PK_S}, g^a \bmod p$

d) $S \rightarrow A: g^b + g^W \bmod p, u, C_1$

The shared key between client and server is $K = g^{b(a+uW)} \bmod p$

e) $A \rightarrow S: K\{C_1\}, C_2$

f) $S \rightarrow A: K\{C_2\}$

1. User A inputs the username and password into the workstation
2. The workstation sends a request login message to server
3. After receiving authentication request message from client, the server first hash IP address of A with timestamp as a cookie to client. Since IP address information is included, DoS attack from this certain IP address can be avoided.
4. Then client randomly generate 'a', and sends back server the cookie, A's username encrypted with server's public key and $g^a \bmod p$. Then both client and server can compute the session key which is $K_{AS} = g^{b(a+uW)} \bmod p$
5. Client sends server $K_{AS}\{C_1\}$ and a random number C_2
6. Server sends client $K_{AS}\{C_2\}$ to fulfill the authentication that both client and server know the session key

Key Establishment Protocol (and authentication with peers)

a) $A \rightarrow S: K_{AS}\{A, B, N_1\}$

b) $S \rightarrow A: K_{AS}\{B, IP_B, port_B, N_1, K_{AB}, t-to-B\}$
$$t-to-B_1 = K_B\{A, K_{AB}, TTL\}$$

c) $A \rightarrow B: t-to-B, K_{AB}\{N_2\}$

d) $B \rightarrow A: K_{AB}\{N_2 - 1, N_3, g^b \bmod p\}$

e) $A \rightarrow B: K_{AB}\{N_3 - 1, g^a \bmod p\}$
Key used between A and B is $K = g^{ab} \bmod p$

1. Client A sends to the server a request message to talk to B with a random number N_1 and the whole message is encrypted by session key between client A and server S.
2. The server then sends back a message which contains who A is going to talk to, B's IP address, port number, N_1 which is sent from A, K_{AB} which is used to establish communication between A and B and a ticket to B. The ticket contains A, K_{AB}, TTL and is encrypted using B's secret key
3. Then A sends B the ticket to B and $K_{AB}\{N_2\}$ as a challenge
4. If B admits the chat request from A, B will sends A $N_2 - 1$ which is an answer to the previous challenge, number N_3 as a challenge to A and $g^b \bmod p$ to try to establish D-H
5. A sends $N_3 - 1$ to answer B's challenge and $g^a \bmod p$ to complete D-H

We should note that session key between A and B is $K_{AB'} = g^{ab} \bmod p$ instead of K_{AB}. This can work perfectly if A or B doesn't trust the server. Even if the server acts like man in the middle, he can only know $g^a \bmod p$ and $g^b \bmod p$ and knows nothing about the real session key. And once $K_{AB'}$ is established, A forgets a and B forgets b.

Messaging Protocol

a) $A \rightarrow B: K_{AB'}\{M_1, HMAC(M_1)\}$
b) $B \rightarrow A: K_{AB'}\{M_2, HMAC(M_2)\}$

1. When A wants to send a message to B, A will use $K_{AB'}$ to encrypt the message and HMAC of that message, where the key in HMAC is derived from $K_{AB'}$, hash of $K_{AB'}$, for example.
2. B does the same while B wants to send message to A

Logout Protocol

a) $A \rightarrow S: K_{AS}\{FIN,\ N_1\}$
b) $S \rightarrow A: K_{AS}\{FIN - ACK,\ N_1 - 1, N_2\}$
c) $A \rightarrow S: K_{AS}\{ACK, N_2 - 1\}$

1. Once A wants to logout, A just send a message include FIN and challenge $N_1\}$ encrypted using K_{AS}
2. Server then sends back $FIN - ACK,\ N_1 - 1$ and another challenge N_2 encrypted using K_{AS}
3. A sends back a ACK message as well as $N_2 - 1$

Discuss the following issues

a. Does your system protect against the use of weak passwords? Discuss both online and offline dictionary attacks.

Yes. First, password or hash of password is not transmitted in clear text. Second, protocol we used in login authentication is based on SRP (Secure Remote Password) which is a strong password protocol. It prevents password verification.

Online attack: After step 4, server already computes out the session key. A only has limited attempt to send K{C1}, which can be verified by server.

Offline attack: W is a derivation of the real password, so the password never appears in any part of the conversation. What's more, in order to do an offline attack, intruder need to solve a D-H problem to get the session, which is proved to e very hard.

b. Is your design resistant to denial of service attacks?

Yes, because we have use cookies. According to our protocol, when there is a connection initiation request, server will send a timestamp to the IP address of request. Server will not compute anything until it receives the cookie sent from the source IP address. If

What's more, because of timestamp, it will time out in a short time.

c. To what level does your system provide end-points hiding, or perfect forward secrecy?

Perfect forward secrecy: Each A and B use a Diffie-Hellman session key to encrypt their conversation and both of them forget the session key and random number a and

b. Even though bot of A and B's servers get compromised, attacker still need to solve D-H problem to retrieve the session key.

End-points hiding: We can't hiding IP address for each message from the lower level, but we username is encrypted during the login authentication.

d. If the users do not trust the server can you devise a scheme that prevents the server from decrypting the communication between the users without requiring the users to remember more than a password? Discuss the cases when the user trusts (vs. does not trust) the application running on his workstation.

Users do not trust the server: During the key establishment process, when client A communicates with client B, they first use key K_{AB} got from the server to establish Diffie-Hellman session key. Even though server can eavesdrop to have $g^a \bmod p$ and $g^b \bmod p$, he'll still need to solve D-H problem to retrieve session key.

User does not trust the application running on his workstation: If user doesn't trust application running on his workstation, password still suffer the possibility of being stolen.