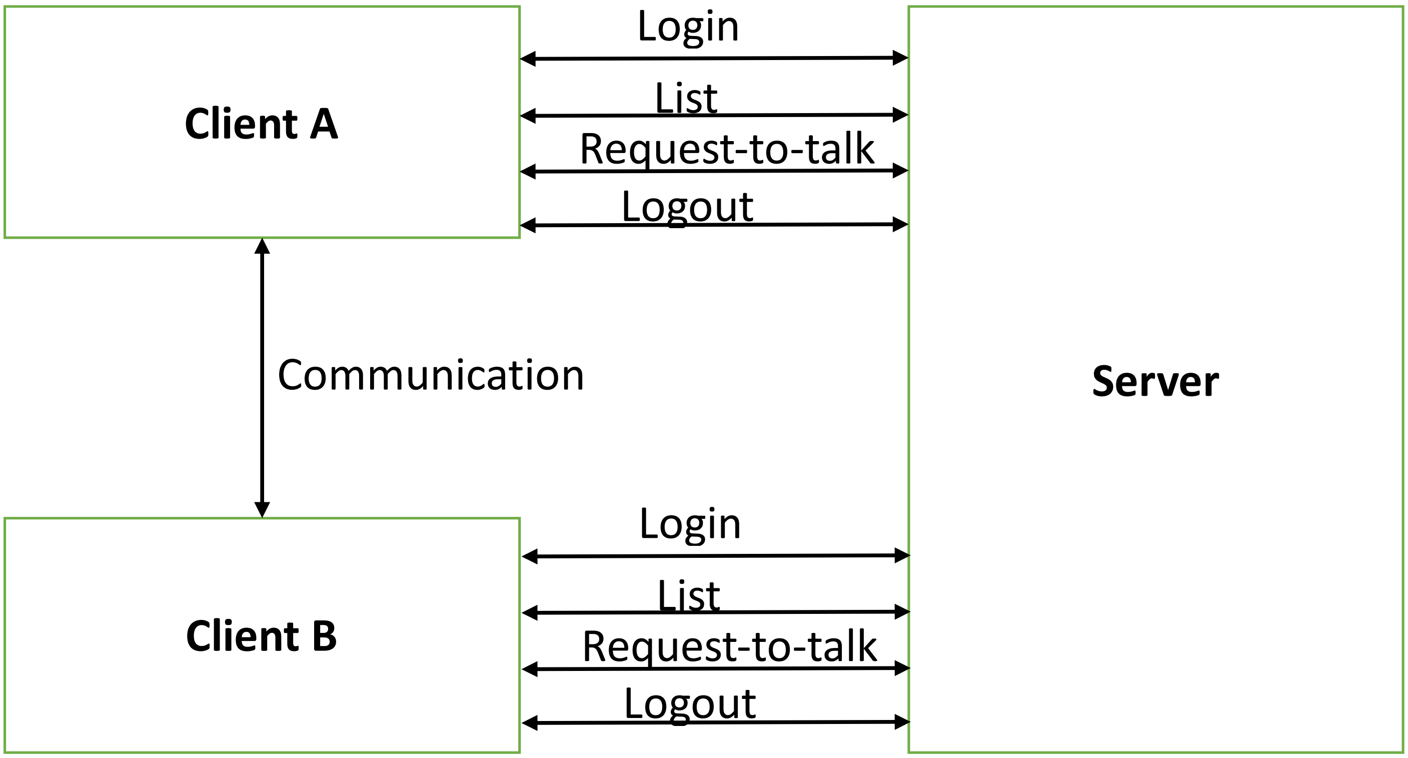
PS4 Report

Rohit Kumar & Sushant Mimani

Architecture:

We have a client-server architecture. The client interacts with the server to login, logout, get a list of online users and to requests for tickers to talk to other users. However, no inter-client communication happens via the server.



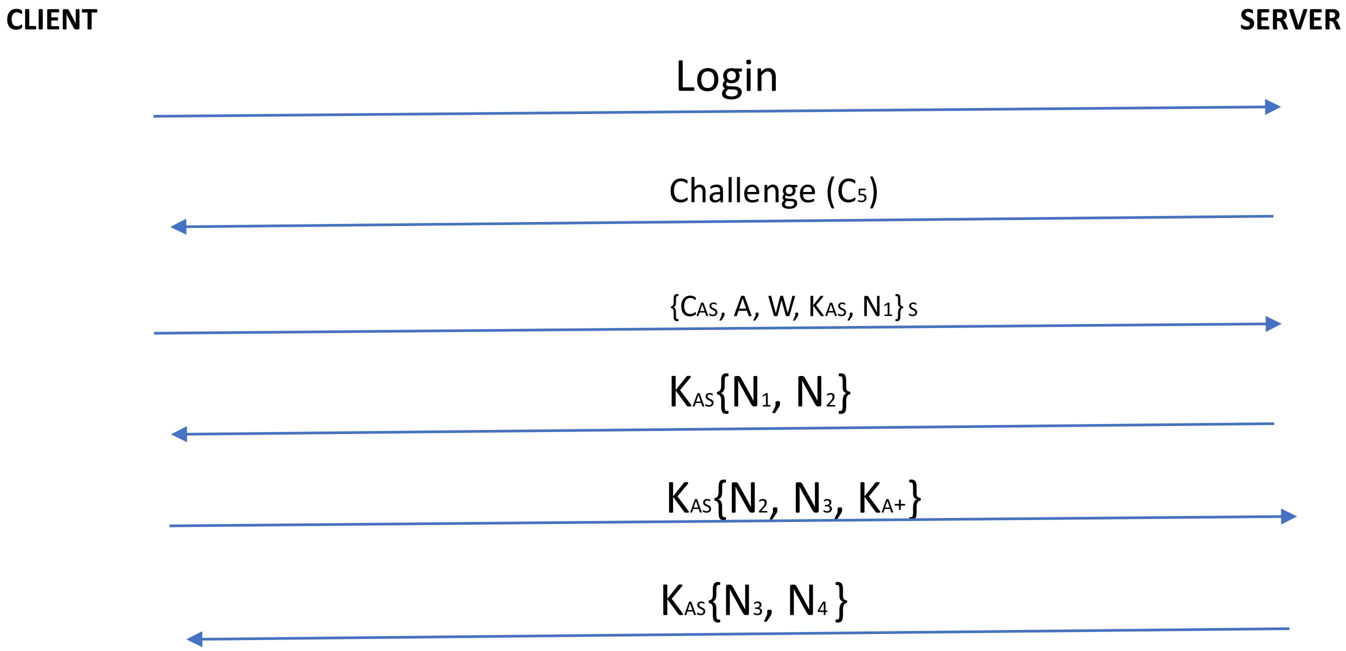
Assumptions:

- Server has list of registered users and their salted passwords
- The client knows the public key of server
- User only knows his password

Security Protocols:

- Symmetric encryption use AES with GCM mode.
- Server session key is generated using PBKDF2
- Timestamp as nonce to prevent replay attacks
- Proof of work at login to prevent DoS

Login:

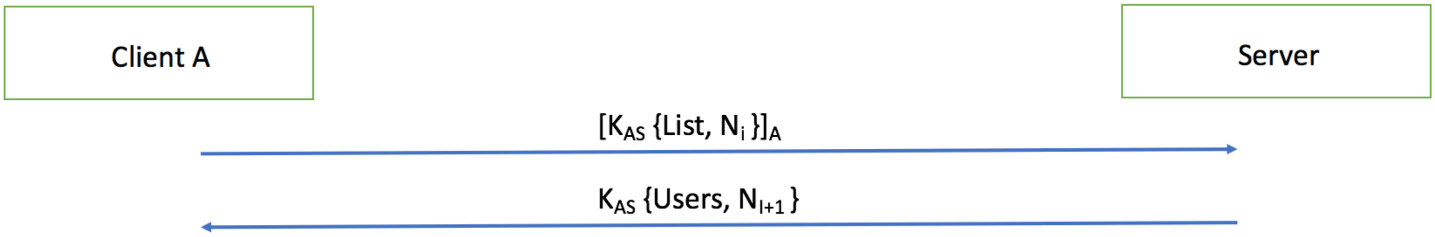


W – A’s password
N1, N2 Nounces

The login protocol we are using is **Needham- Schroeder.** The *Needham–Schroeder Symmetric Key Protocol* is based on a symmetric encryption algorithm. It forms the basis for the Kerberos protocol. This protocol aims to establish a session key between two parties on a network, typically to protect further communication. The client and Server both mutually authenticate each other, on authentication completion a valid client is logged in.

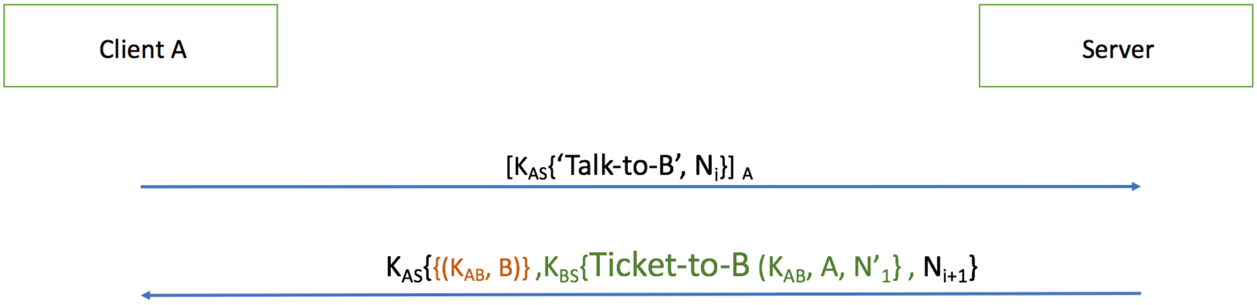
In this case the User initiates the connection by sending a login request to the server. The server requires a challenge to be solved by the client before authenticating it. The client sends its identity password, and key KAS encrypted by server‘s public key. Nounces are used to make sure that the communication is coming from the expected source and man-in-the-middle attack is not hacking the communication.

List users:



Client A sends a LIST request to the server. This message is encrypted using A’s server session key and signed by A. Since the server has A’s public key, it can verify the source of the request and then respond back with the list of online users. The response is also encrypted using A’s server session key.

Client-to-Client communication:

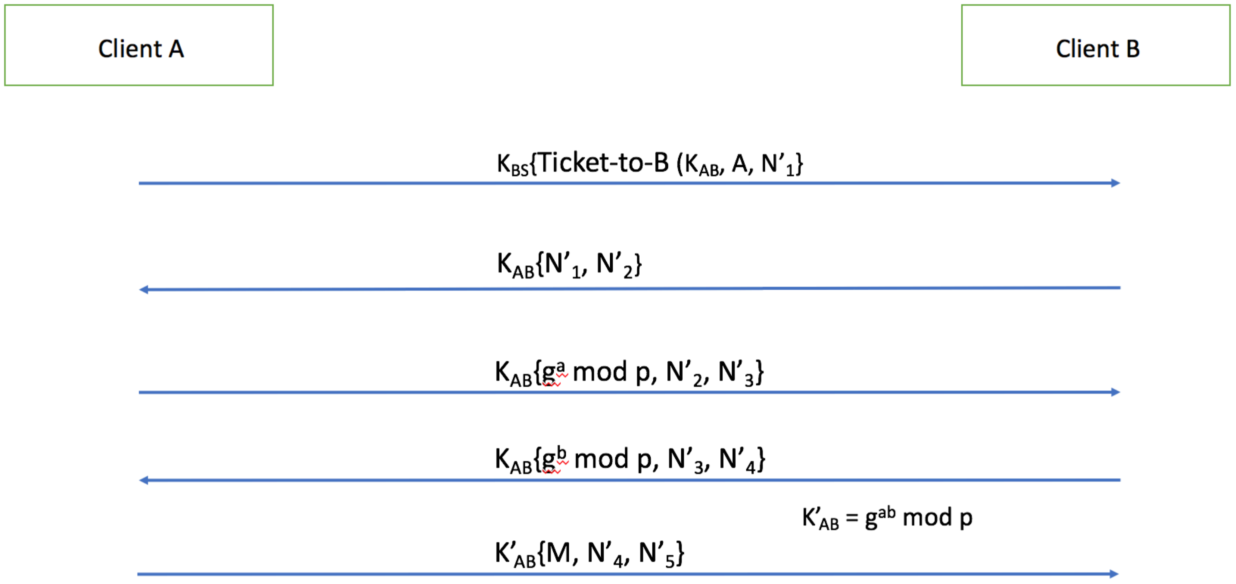

Client A
Server
$[K_{AS}\{\text{‘Talk-to-B’}, N_i\}]_A$
$K_{AS}\{\{(K_{AB}, B)\}, K_{BS}\{\text{Ticket-to-B } (K_{AB}, A, N'_1\}, N_{i+1}\}$

Client A sends a request for key establishment to the server. This message is signed by the private key of A. Since the server, at this point, has the public key for A (sent during login authentication) it can verify that the incoming message has indeed been sent by a legitimate user.

Once the source is verified, the server generates a session key K_{AB} for client A and B and send back a response, which is encrypted using key K_{AS} to client A. The response contains the following.

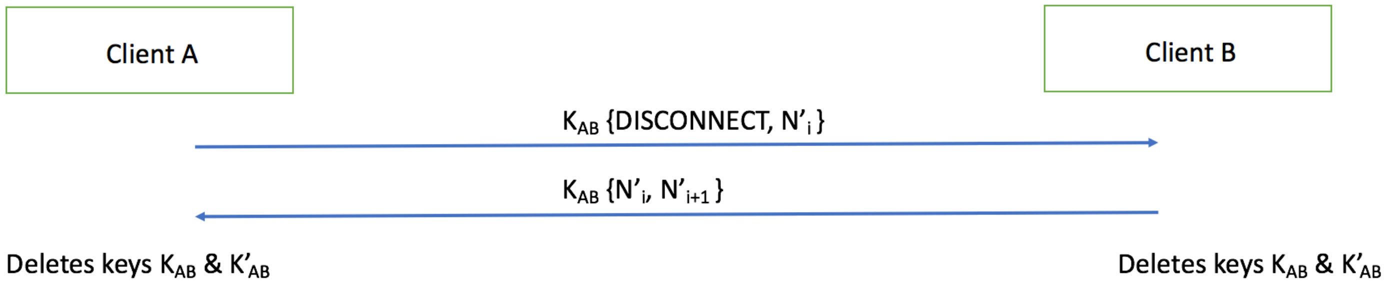
K_{AB} → This is encrypted using A’s server session key
Ticket-to-B → This is encrypted using B’s server session key. So only B can decrypt this message. Ticket-to-B also contains a server generated nonce, that will be used by B to verify its identity to A.

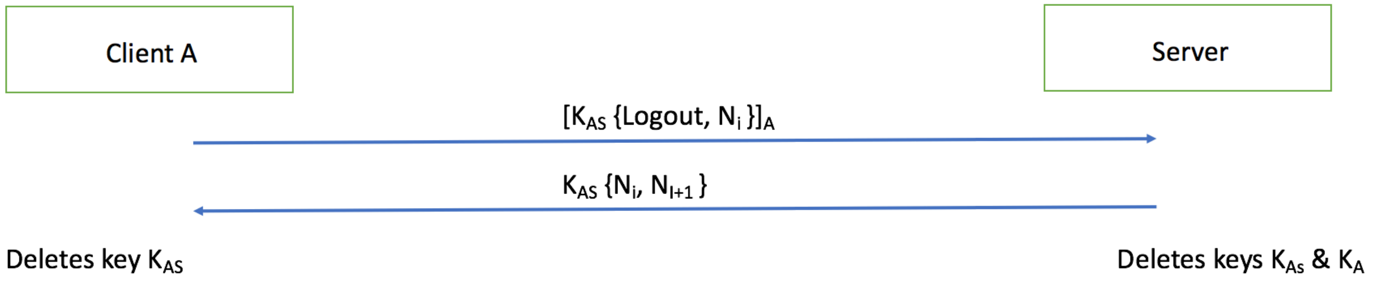
Once A gets a ticket-to-B, it initiates a communication with B. Since the ticket is encrypted using K_{BS}, B can decrypt it. It replies with the server generated nonce to prove its identity to A. Once identity is established, both clients establish a session key using Diffie-Hellman. Once the session key is established, all communication between A & B is encrypted using K'_{AB}


Client A
Client B
K_{BS}\{Ticket-to-B (K_{AB}, A, N'_1\}
$K_{AB}\{N'_1, N'_2\}$
$K_{AB}\{g^a \bmod p, N'_2, N'_3\}$
$K_{AB}\{g^b \bmod p, N'_3, N'_4\}$
$K'_{AB} = g^{ab} \bmod p$
$K'_{AB}\{M, N'_4, N'_5\}$

Update Note: N’2 = N1+1, N’3 = N’2 +1, N’4 = N’3+1
To protect from replay attack and dictionary attack, we check the nonces on either side which arrive after 5 sec interval from the sender.

Session termination and Logout:


Client A
Client B
K_{AB} \{DISCONNECT, N'_i \}
K_{AB} \{N'_i, N'_{i+1} \}
Deletes keys K_{AB} & K'_{AB}
Deletes keys K_{AB} & K'_{AB}


Client A
Server
[K_{AS} \{Logout, N_i \}]$_A$
K_{AS} \{N_i, N_{I+1} \}
Deletes key K_{AS}
Deletes keys K_{As} & K_A

On session termination and logout, all keys relevant to the client logging out are deleted.

1. Does your system protect against the use of weak passwords? Discuss both online and offline dictionary attacks.
    - The system protects against the use of weak passwords by requiring the client to solve a challenge before each login attempt.

    - The system protects against online dictionary attacks because we have the client required to solve a challenge before it can be authenticated by the server. Hence for each attach a challenge has to be solved which, is not practical for the attacker to perform an online dictionary attack that way.
    - The system also protects against offline dictionary attacks because we are salting the password and using key stretching hashing algorithms to generate the hash. This way of generating hash should take some time, that prevents the attacker from doing a dictionary attack.

2. Is your design resistant to denial of service attacks?
    - Yes, the system is resistant to DoS attacks. The server requires every client to provide a proof of work before letting them login. We also plan to implement blocking users for 15 minutes after 3 unsuccessful login attempts.

3. To what level does your system provide end-points hiding, or perfect forward secrecy?
    - Inter-client communication uses Diffie-Hellman. Hence, there is complete PFS. No messages are sent in any communication without being encrypted. Hence, the system provides complete end-point hiding.

4. 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.

    - If the users do not trust the server, only information that server has is users' IP address, ports, public key and their passwords. However, during client-to-client communication, A and B establish perfect-forward-secrecy using Diffie-Hellman, even if it can encrypt messages using B/A's public key it cannot compute the Diffie-Hellman portion of the key establishment phase and hence cannot decrypt any messages.
    - If the user does not trust his workstation, then it is advisable to not save the password, public keys, session keys, symmetric on workstation.

      Also additional authentication devices such as fingerprint scanner access card etc can be deployed to provide additional protection. If the user trusts his workstation, then he can store keys and additional parameters needed for the session for ease of use.