Final Design of Instant Messaging System

-KETAN KALE

-ABHIRUCHI KARWA

Architecture

- The secure instant messaging system will have one server and multiple clients.
- A client application can have multiple users where a user can log in after another user has finished their session and has logged out.

Assumptions

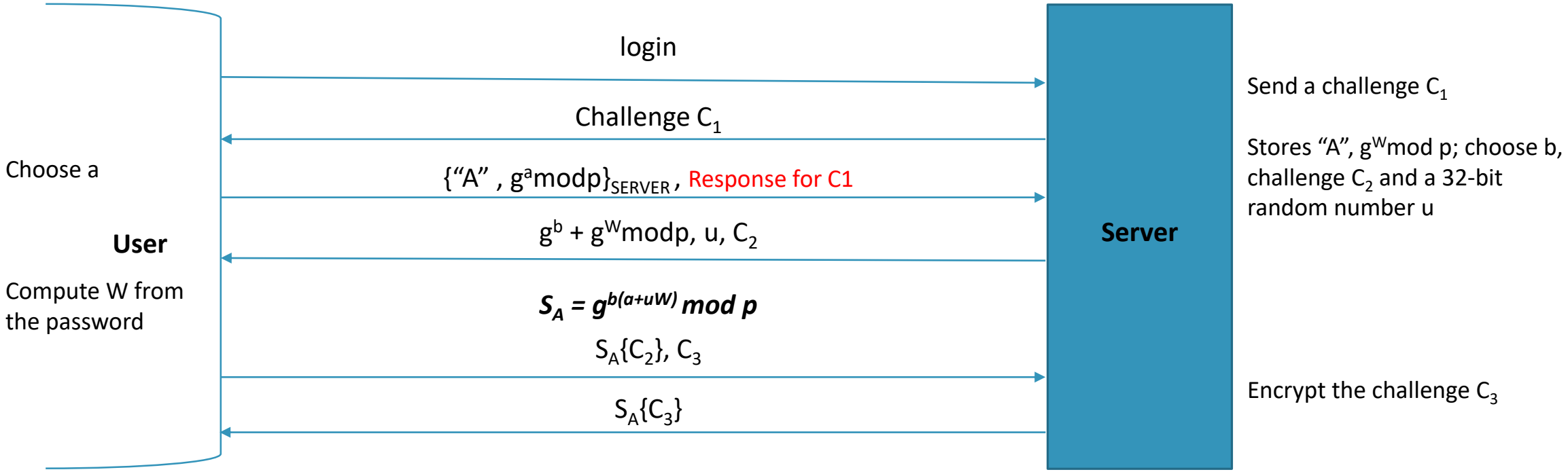
- All the users are pre-registered with the server.
- Every user will have a username and a password which will be used for logging into the system; only the user knows his/her password.
- The username-password pair will be stored on the server, but in a secured manner.
- Every client in the system will know the public key amongst the public-private key pair of the server.
- No other keys or passwords are stored.
- The server is trusted.

Authentication protocol (Login)

- The user can log into the system using the username and password.
- After, the user has logged in successfully, a session will be started and a session key will be generated by the server which will be a secret between the user and the server.
- Also, after the user has logged in, his/her name should be added to the list of users currently connected to the server.
- A DoS attack where server's resources are exhausted is also prevented because unless the client does some proof of work, a server doesn't have to engage its resources.
- Also, in a case of an offline dictionary attack no passwords are compromised as the SRP protocol is used. Thus, the server only knows g^{W}modp where W = f(password).
- Login is triggered after the user enters the username and password

The entities in red are the additions/modifications made in the previously submitted protocol

Authentication protocol (Login)



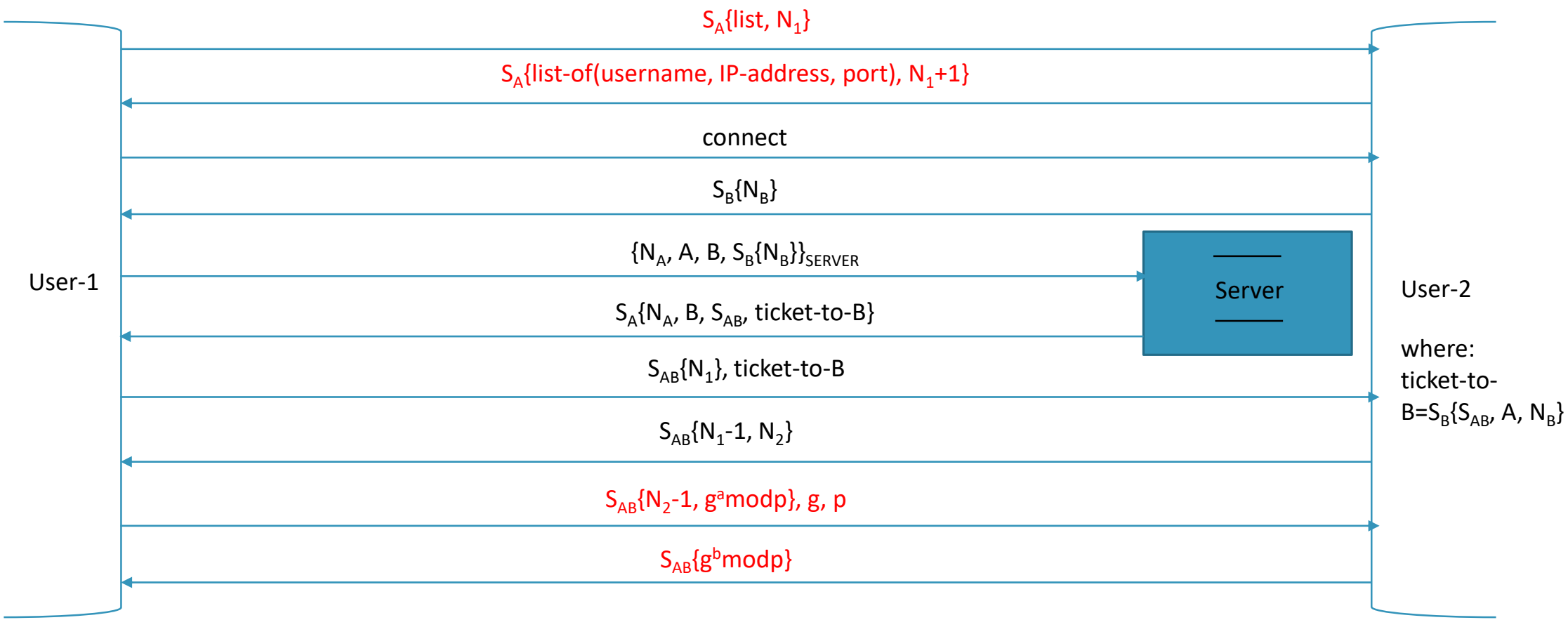
Key establishment protocol (and authentication with peers)

- Using the session key received from the server, the user will establish a key that will be common between two users that want to communicate.
- Expanded Needham Schroder protocol along with the Diffie Hellman key exchange will be used to authenticate both the users and also generate shared secret between the two users.
- The shared secret will only last as long as the two users are communicating. If they want to end the communication, both of the users must forget the shared secret; thus ensuring forward secrecy.

- Command : disconnect <sender_username> <receiver_username>

The entities in red are the additions/modifications made in the previously submitted protocol

Key-establishment protocol



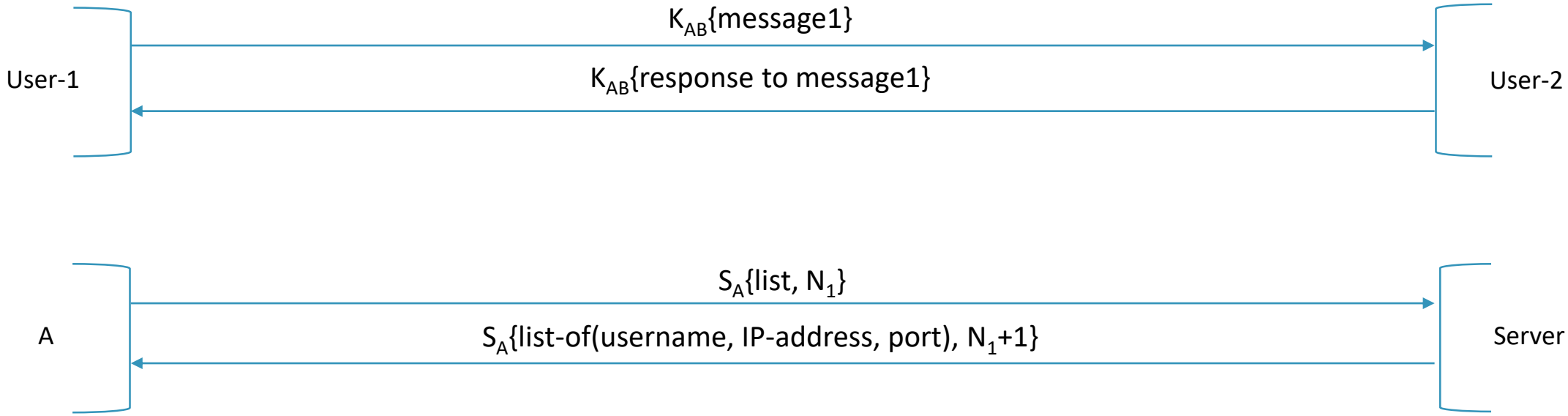
The shared secret between A and b now becomes -> $K_{AB} = g^{ab} modp$

Messaging Protocol

- To encrypt the data that is exchanged between the two users, AES algorithm with the Galois Counter Mode is used. The key size will be 256 bits.
- Now the users A and B share a key K_{AB} , so they use it in the messaging protocol.
- The message is encrypted with this shared secret by AES-GCM and sent to the other user.
- A -> B : K_{AB}{message}

- Command : send <receiver> <message>
- Command : list

Messaging Protocol

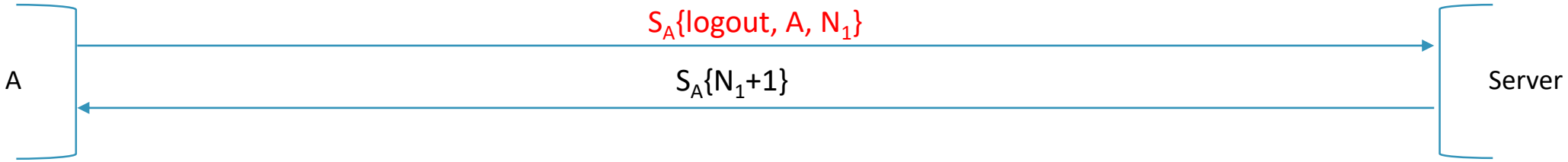


K_{AB} => Shared secret between A and B
S_A => Session key of A known to A and Server

The entities in red are the additions/modifications made in the previously submitted protocol

Logout protocol

- If the user has to log out of the session, it can do so by a logout command.
- When a user logs out of the session it must forget the session key received from the user.
- When any user logs out from the system, the server should remove the user's name from the list of all the active users and forget the session key generated for that user.
- Command : logout <username>



Services

- Mutual authentication
- Message integrity
- Confidentiality
- Perfect forward secrecy
- DoS Protection
- Online/offline dictionary attack protection
- Protection against weak passwords
- End-point hiding