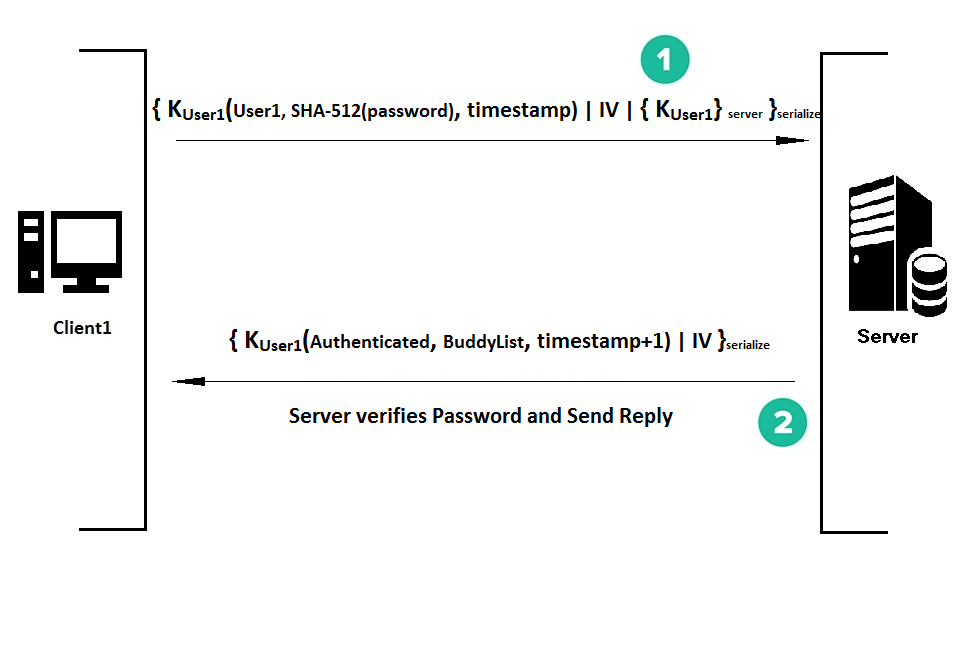
**Secure Instant Messenger Report**

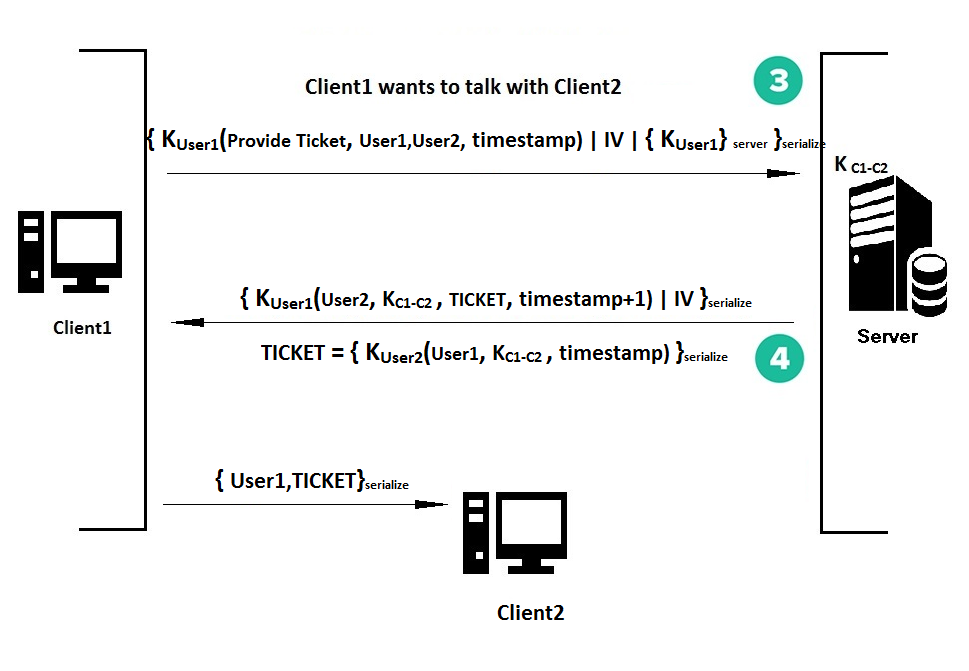
**Introduction**

This is project report for secure instant messaging application. The main component of this application is client and server. Server authenticates the clients and hence, the clients can then communicate with each other in a peer to peer environment.

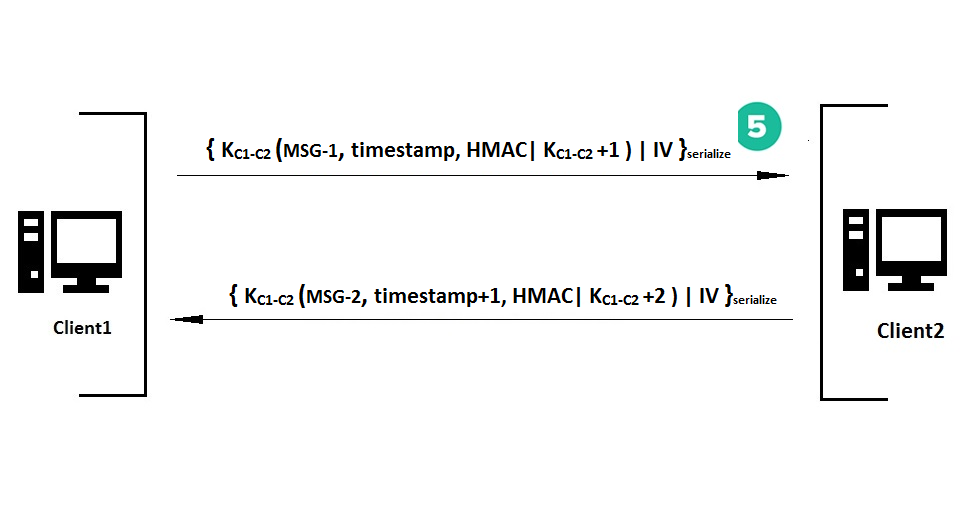
1. **Protocol Design and High Level Implementation**
   1. **Authentication between Client and Server (Login)**

****

* 1. **Getting Ticket from Server for communication with another client**

****

* 1. **Secure Communication between two clients**
     1. Common Session Key **K C1-C2**



1. **Contribution**

We have applied distributed approach for various API development and verification. First, we developed encryptutil APIs and verified them on a standalone machine. Then we developed simple socket application for communication between client and server. Our final step was to integrate these APIs to Application. Varun contributed significant part in verification of APIs and development of protocol design in application. Jay contributed significant part in development of security APIs and verification of protocol design and integration.

1. **Security Features**
2. Authentication: Client is authenticated using password based authentication. When client enters the username and password into the workstation, the workstation generates SHA512 hash of the password.
3. Communication: Communication between clients is session based. That is, for each session a session key is assigned by the server for client to client or peer to peer communication. This session key remains valid only for that particular session.
4. Confidentiality: As the session keys are known only to the clients, the message to be sent remains confidential.
5. Integrity: Integrity is maintained by sending HMAC of the message along with the message. If the message is changed by the attacker, its integrity won’t be the same when calculated at the receiver end. Hence, a change in message won’t go undetected.
6. Attacks considered

1.      Eavesdropping

2.      Man in the Middle

3.      Replay attack

4.      Message modification(Integrity)

5.      Active attacker posing as Alice to server or other clients

1. **Challenges Faced**

* **Key strength of AES:**

The default java APIs allow us to use only 128 bit key for AES Algorithm. But our AIM was to use 256 bit key for more secure communication. For resolution of this problem we added JCE unlimited strength APIs provided by Java.

* **RSA long data encryption problem:**

When we were encrypting data larger than 245 bits then Java APIs for RSA threw exception of length limitation for RSA. On realizing that, we changed the architecture and encrypt symmetric key instead of encryption of whole message.

* **Integration of JavaFX and Listener Component of Socket:**

While we run listener for TCP socket on client side, it restricts front application from running. We have used multi-threaded approach for this implementation. From this our listener run in another thread except daemon thread.