**Chat Application Report**

By {Karan Tanwar (2017CS50411), Harsh Yadav(2017CS50409)}

Implementation specifics:

1. If the recipient states an error “Header incomplete” to the server and server give an error to the sender immediately. So, the sender can send next message after ACK from server.
2. Signature is sent the header itself so that we don’t need to send signature first and then the encrypted message.
3. All instances of public key, private key and signature are sent in string format only.
4. Three hash tables are maintained at the sever side as of now. One for storing Input Streams specific to every user’s socket, other for Output Stream, and another for storing public keys of every registered user. When someone unregisters, the its entry gets deleted from all the three hash tables so that the buckets (in hashtable) don’t get full and time complexity for searching remains O(1). Also, there is no overflow of registration in case of malware/script attacking the server, so that memory gets full.
5. Handled the cases when same username is being registered.

Extensions:

Ctrl-C vs Unregister:

For handling cases such as user disconnecting arbitrarily by pressing Ctrl-C, we can try to catch error on the in.readLine() function in receiving message from client. Because by pressing ctrl-c, there is a NoLineException displayed and after catching the error, we can close the sending thread of client and send server a message to UNREGISTER. When server reads UNREGISTER message, then it also sends the same signal to client’s receiving socket and then client’s receiving thread is also closed.

Also, when a client is successfully unregistered, we can update in the hash table when the threads of this client were closed. This is another Hash Table which is not implemented in our design but can be done. [This can be used as to implement “Last Seen” for every user].

Sending Messages to Offline users:

Whenever a client comes online, then it can send an ack to server itself and server can update the previously mentioned Hash Table by changing Last Seen and change the ‘status’ field of user as ‘online’. At that point server will send an Ack to every online user that this user is now online.

Now, if a client wants to send a message to the recipient, first it checked that the recipient is online by searching the Hash Table. If yes, then it goes conventionally. If no, then the message is stored in the server side in a buffer [Single Tick] and whenever the recipient is server gets Ack and now buffered message can be sent. [Blue Double Tick] Now, the sender is notified that the buffered message is sent to the recipient.

Encrypted message/public keys not sent in binary format:

Encrypted messages and public keys are padded accordingly in the byte format to add data to the beginning, middle, or end of a message prior to encryption. This is done so that no one can see patterns in the message. So if we send in binary format then one can detect this padding and remove them accordingly thus checking common phrases like *sincerely yours*.