Task #1.

1. For prevent plaintext password used Bcrypt, it means on the server database will saved only hash of the client password.
    a. Client requests server to salt
    b. Server response salt
    c. Client re-generate password hash using salt and plaintext password
    d. Send password hash to the server
    e. Server verify hash and uid
2. For bound chat history to the current machine used unique MAC address as identifier.

Task #2.

1. For it used Bcrypt library and we store on the server db only hash of the password, it means if server db will leak, it means attackers can't get clients password.
2. For encryption and decryption history log was used `javax.crypto.*` package and symmetric-key algorithm. As key symmetric-key was used sha1hash of the string H(H(salt) + "|" + login + "|" + password). What is it mean?

2.1. Client can't decrypt history log without servers salt

2.2. If attackers get history log, they can't decrypt it, because, they need also salt, which stored only on the server, if attackers get server side info, like H(salt) and H(password), it is also give nothing, because server not stores password in plaintext.

Task #3.

Note. *For this task additional used Apache Commons Codec library, it can be found in attached project SecureChat3/src and standard Java 8 package java.security.**

1. Create Root certificate

openssl req -x509 -sha256 -newkey rsa:4096 -keyout MyRoot_prv.pem -nodes -out MyRoot.pem -days 365

2. Generate private key and certificate of Alice *(this step needed to be done for bob.csr and server.csr)*

openssl genrsa -out alice_prv.pem 2048

openssl req -new -key alice_prv.pem -out alice.csr

3. Get it signed by the root CA *(this step needed to be done for bob.csr and server.csr)*

openssl x509 -req -days 365 -in alice.csr -CA MyRoot.crt -CAkey MyRoot_prv.pem -set_serial 02 -out alice.crt

4. Extract CA root public key

openssl x509 -pubkey -noout -in MyRoot.crt > MyRoot_pub.pem

5. Extract Alice's public key

openssl x509 -pubkey -noout -in alice.crt > alice_pub.pem

6 Extract Bob's public key

openssl x509 -pubkey -noout -in bob.crt > bob_pub.pem

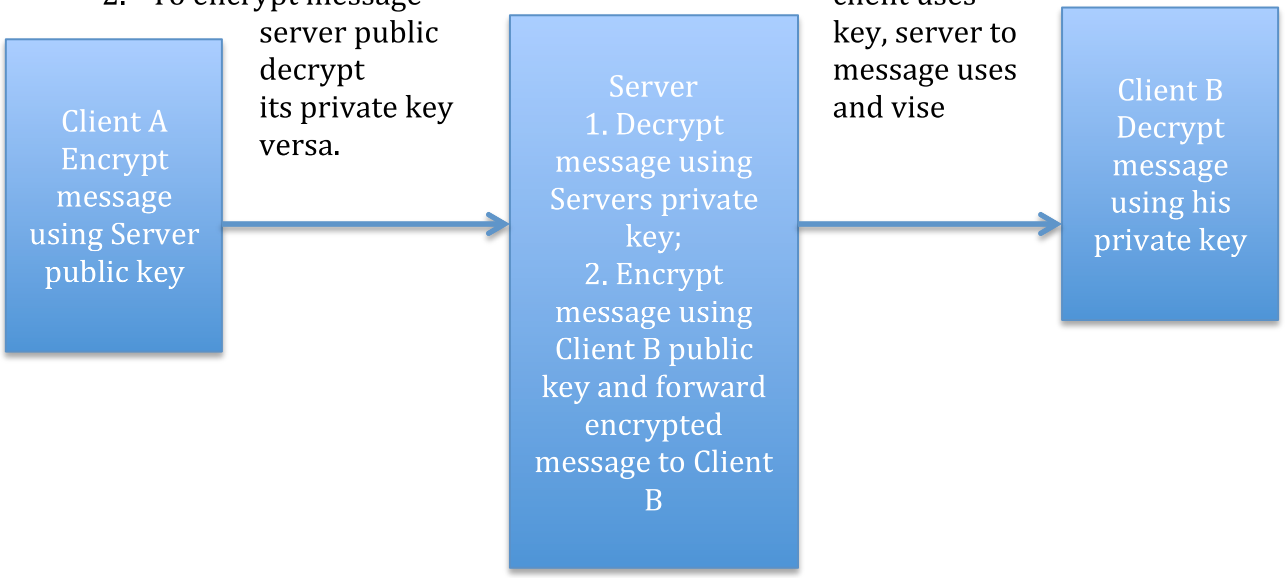
7. openssl pkcs8 -topk8 -inform PEM -outform DER -in alice_prv.pem -nocrypt > alice_prv.pk8 *(this step needed to be done for bob.csr and server.csr)*

8. For login client need to select certificate .crt file, for chatting private key *_prv.pk8.

How encryption/decryption and login using certificate woks:

1. When client select certificate, MyChatClient.java verify selected certificate against Root CA, and check validity and check is subject name of the certificate (CN) match to current client login
    a. if selected certificate verification success, client requests (in request also sends client public key) servers public key.

2. To encrypt message client uses server public key, server to decrypt message uses its private key and vise versa.


Client A Encrypt message using Server public key
Server 1. Decrypt message using Servers private key; 2. Encrypt message using Client B public key and forward encrypted message to Client B
Client B Decrypt message using his private key

3. If attackers will get servers private key, it gives them nothing, because, to decrypt chat history log uses clients private key.