Team:

- Sergey Frolov
- Ian Martiny
- Shirly Montero Quesada

Title: OTR-Messenger

Project Summary:
A chat client that implements an Off-The-Record (OTR) protocol of communication. OTR allows clients to talk with each other in an encrypted fashion with repudiation -- meaning a client can later deny having sent something.

Project Requirements:

This is a chat service, it can be privatized for in-house use of a company, which it will impose other business requirements –e.g., use company email as the username–. However, in a general public use the following was the only foreseen necessary requirement.

Table 1. Business requirements

Requirements	Specifications	Topic Area	Actor	Priority
BR-001	Password at least 8 characters and consists of, at least, and one uppercase, one lowercase char, one special char, and one number	Sign up	All	Medium

Users could be computer illiterate as well as a little savy. The requirements were written to complete the phrase: “As a <user> I need to <task> to <accomplish my goal>”.

Table 2. User requirements. The stretch functionalities are shown with red background.

Requirement	<user>	<task>	<accomplish my goal>	priority
UR-001	client	sign up	to create an account and access chat service	High
UR-002	client	log in	to access the chat service	High

UR-003	client	send messages	to communicate with others	Critical
UR-004	client	receive messages	to communicate with others	Critical
UR-005	client	view friends list	to see who is online	High
UR-006	client	modify friends list	to update the list	Medium
UR-007	client	organize friends list	to modify groups in the list	High
UR-008	client	view keys (public/private)	so I can verify them	Medium
UR-009	client	request change key	to communicate securely	Medium
UR-010	admin	change server status (launch/reset/terminate)	provide, temporarily stop or terminate service	Critical
UR-011	admin	view/log list of all users	part of documentation to monitor the system	High
UR-012	admin	view logged in users	manage the system	Medium
UR-013	admin	view keys (have access the database)	security analysis on keys (not repeating)	Medium
UR-014	client	manually change key	to have direct control over my security	Nice-to-have
UR-015	client	access password	to view it, modify it	Low
UR-016	client	import contact list (select/deselect)	to add many friends at once	Nice-to-have
UR-017	client	upload/download files	to send/receive more data	Low
UR-018	client	read old messages (memento)	to review what was said	Low
UR-019	client	reject/accept invitiations to be added to a list	to decide who I want to talk with	Nice-to-have
UR-020	client	black lists other clients	to block others from bothering me	Low

Table 3. Functional or Non-functional requirements

Requirement	Area: specifications
FNFR-001	Security: On account creation a public and a private key will be generated
FNFR-002	Security: Create a shared encryption key as two users connect with each other (exchange each others public key)
FNFR-003	Security: After some time out 60s, the users will publish their private signing keys and new ones will be generated.
FNFR-004	Legal: Once a new shared/private signing key is generated, the user will be alerted of the change
FNFR-005	Legal: If user wants to override a key change time setting, an advice will be generated (if longer time then alert of the consequences)
FNFR-006	Performance: after log in it takes 7 s to show friend's list
FNFR-007	Performance: 2 s after showing friends list window, show list of friends who are online

Use Cases:

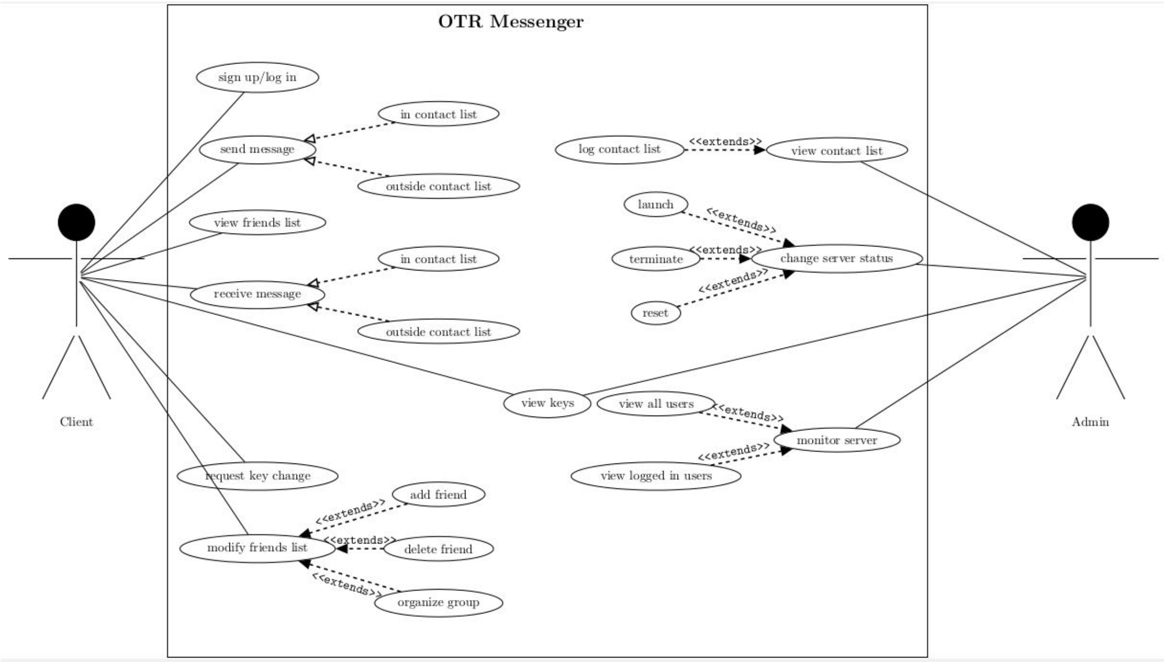


Figure 1. OTRMessenger Use Case Diagram

UI Mockups:

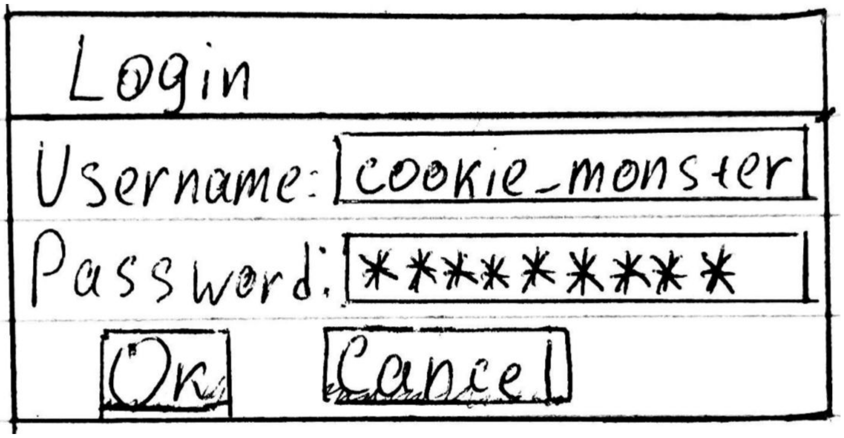


Figure 2. Login GUI.

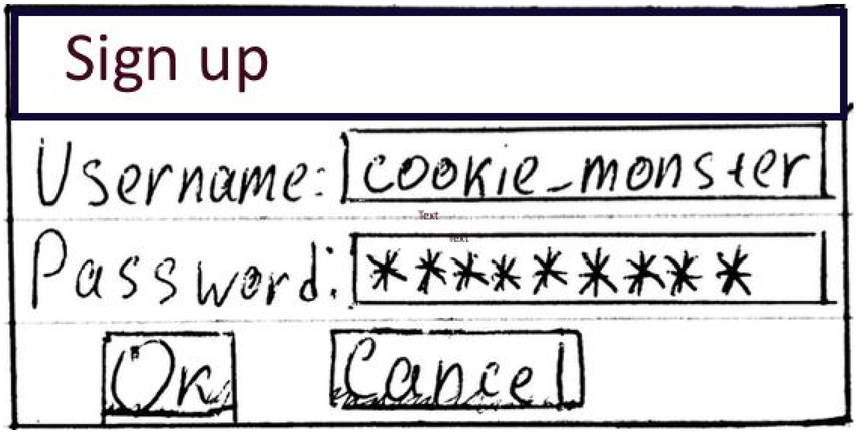


Figure 3. Signup GUI.

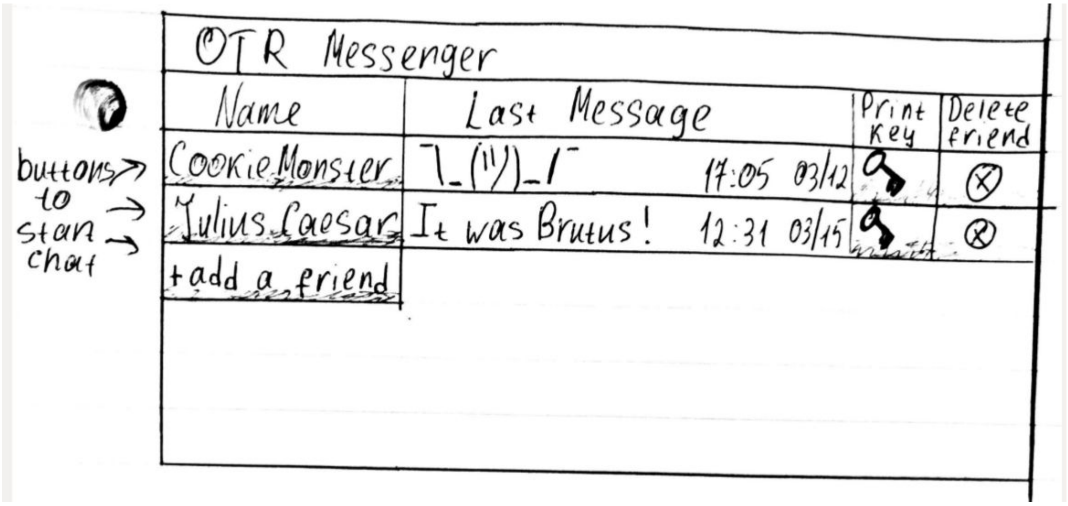
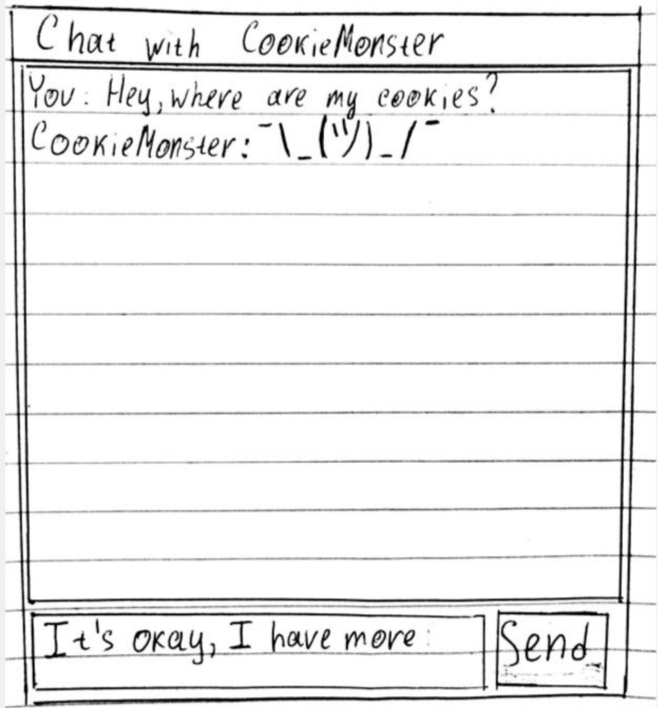


Figure 4. Friends list GUI



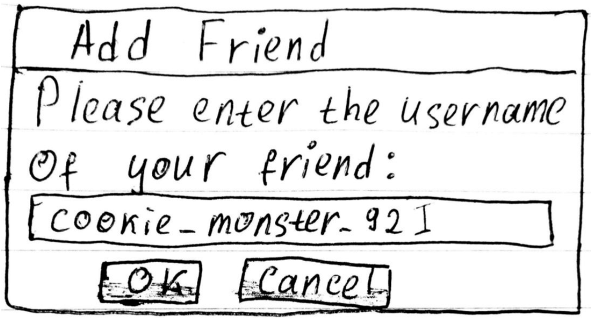
Chat with CookieMonster

You: Hey, where are my cookies?

CookieMonster: ¯_(ツ)_/¯

It's okay, I have more :

Figure 5. Chat GUI

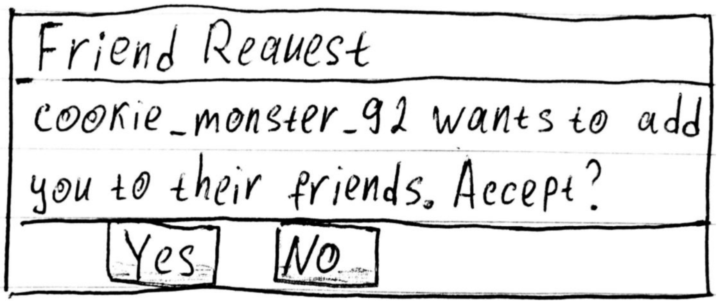


Add Friend

Please enter the username Of your friend:

cookie_monster_92

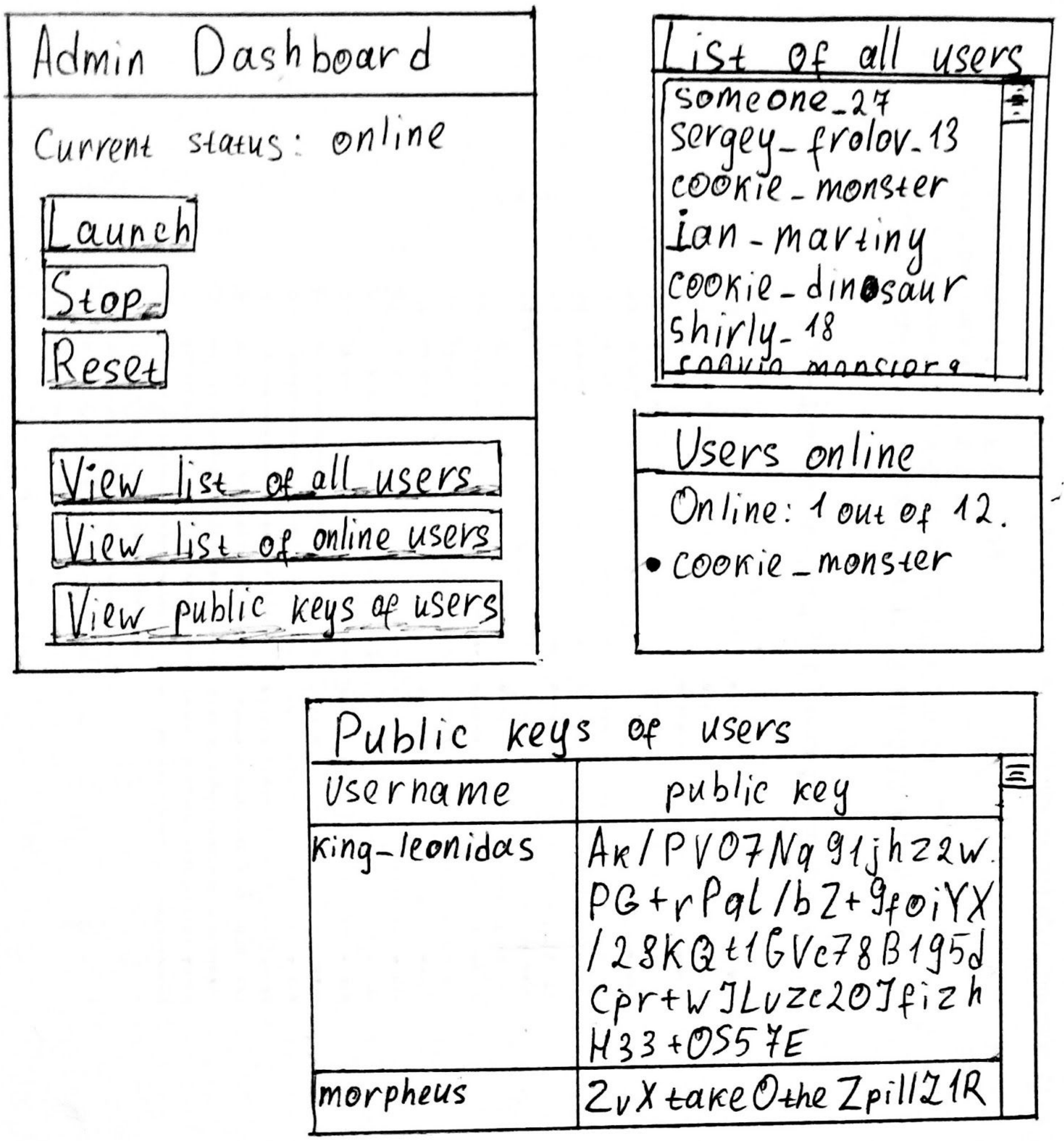
Figure 6. Add friend GUI



Friend Request

cookie_monster_92 wants to add you to their friends. Accept?

Figure 7. Friend Request GUI



someone_27
sergey_frolov_13
cookie_monster
ian-martiny
cookie-dinosaur
shirly-18

Online: 1 out of 12.

- cookie_monster

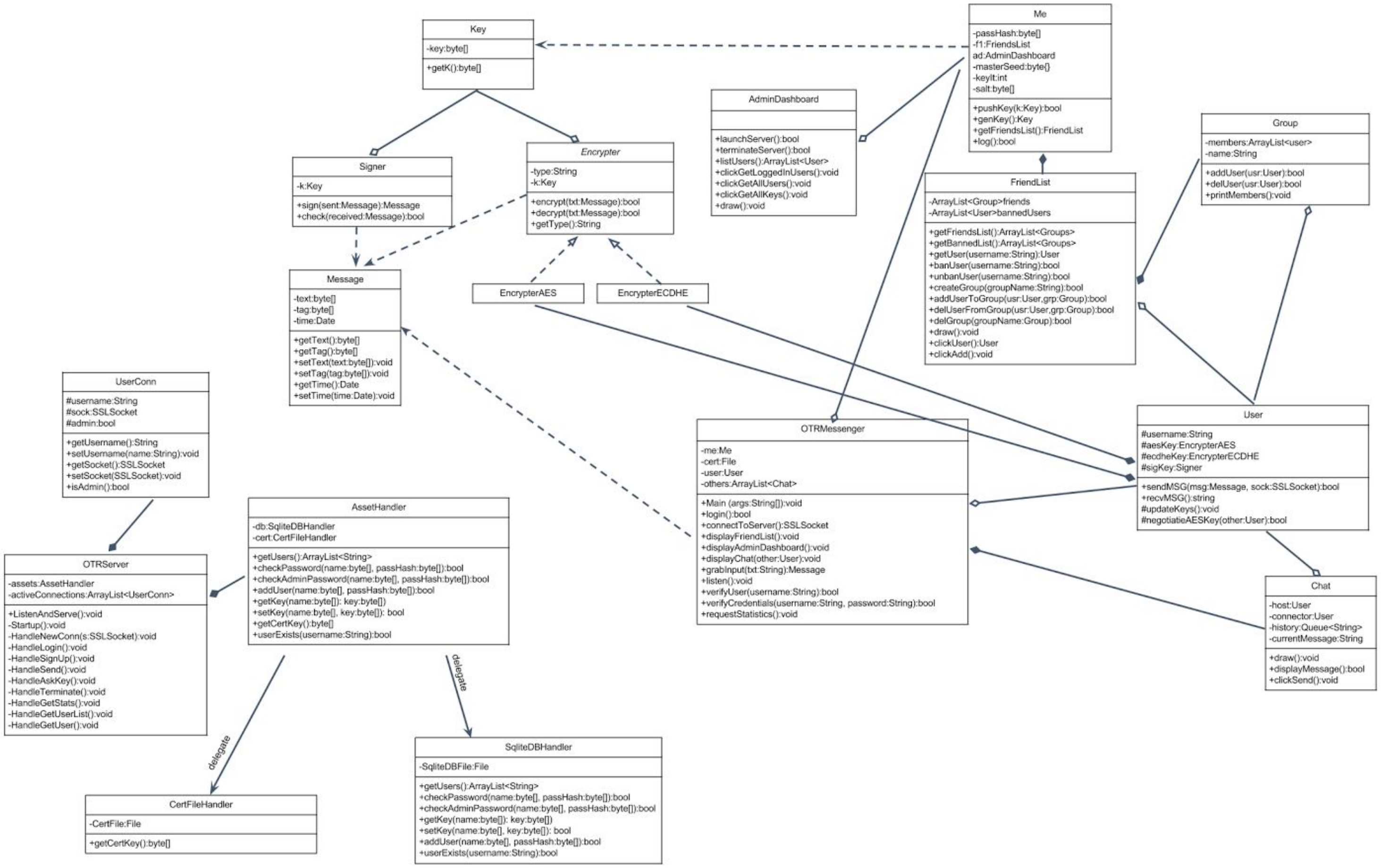
Username	public key
King-leonidas	Ak/PV07Nq91jhz2w.PG+rPql/bZ+9foiYX/28KQt1GVc78B195dCpr+wJLvzc20JfizhH33+0S57E
morpheus	2vXtake0theZpillZ1R

Figure 8. Admin Dashboard and all windows popped from it.

Data Storage: We will use an SQLite database. We will have the following tables:

- username and a hash of their passwords (for verifying logins)
- username and public keys (so other users can get them)
- username and list of published private keys

Class Diagram:


Key
-key:byte[]
+getK():byte[]
Me
-passHash:byte[]
-f1:FriendsList
ad:AdminDashboard
-masterSeed:byte{}
-keyIt:int
-salt:byte[]
+pushKey(k:Key):bool
+genKey():Key
+getFriendsList():FriendList
+log():bool
AdminDashboard
+launchServer():bool
+terminateServer():bool
+listUsers():ArrayList<User>
+clickGetLoggedInUsers():void
+clickGetAllUsers():void
+clickGetAllKeys():void
+draw():void
Group
-members:ArrayList<user>
-name:String
+addUser(usr:User):bool
+delUser(usr:User):bool
+printMembers():void
Signer
-k:Key
+sign(sent:Message):Message
+check(received:Message):bool
Encrypter
-type:String
-k:Key
+encrypt(txt:Message):bool
+decrypt(txt:Message):bool
+getType():String
FriendList
-ArrayList<Group>friends
-ArrayList<User>bannedUsers
+getFriendsList():ArrayList<Groups>
+getBannedList():ArrayList<Groups>
+getUser(username:String):User
+banUser(username:String):bool
+unbanUser(username:String):bool
+createGroup(groupName:String):bool
+addUserToGroup(usr:User,grp:Group):bool
+delUserFromGroup(usr:User,grp:Group):bool
+delGroup(groupName:Group):bool
+draw():void
+clickUser():User
+clickAdd():void
Message
-text:byte[]
-tag:byte[]
-time:Date
+getText():byte[]
+getTag():byte[]
+setText(text:byte[]):void
+setTag(tag:byte[]):void
+getTime():Date
+setTime(time:Date):void
EncrypterAES
EncrypterECDHE
UserConn
#username:String
#sock:SSLSocket
#admin:bool
+getUsername():String
+setUsername(name:String):void
+getSocket():SSLSocket
+setSocket(SSLSocket):void
+isAdmin():bool
User
#username:String
#aesKey:EncrypterAES
#ecdheKey:EncrypterECDHE
#sigKey:Signer
+sendMSG(msg:Message, sock:SSLSocket):bool
+recvMSG():string
#updateKeys():void
#negotiatieAESKey(other:User):bool
OTRMessenger
-me:Me
-cert:File
-user:User
-others:ArrayList<Chat>
+Main (args:String[]):void
+login():bool
+connectToServer():SSLSocket
+displayFriendList():void
+displayAdminDashboard():void
+displayChat(other:User):void
+grabInput(txt:String):Message
+listen():void
+verifyUser(username:String):bool
+verifyCredentials(username:String, password:String):bool
+requestStatistics():void
AssetHandler
-db:SqliteDBHandler
-cert:CertFileHandler
+getUsers():ArrayList<String>
+checkPassword(name:byte[], passHash:byte[]):bool
+checkAdminPassword(name:byte[], passHash:byte[]):bool
+addUser(name:byte[], passHash:byte[]):bool
+getKey(name:byte[]): key:byte[])
+setKey(name:byte[], key:byte[]): bool
+getCertKey():byte[]
+userExists(username:String):bool
OTRServer
-assets:AssetHandler
-activeConnections:ArrayList<UserConn>
+ListenAndServe():void
-Startup():void
-HandleNewConn(s:SSLSocket):void
-HandleLogin():void
-HandleSignUp():void
-HandleSend():void
-HandleAskKey():void
-HandleTerminate():void
-HandleGetStats():void
-HandleGetUserList():void
-HandleGetUser():void
Chat
-host:User
-connector:User
-history:Queue<String>
-currentMessage:String
+draw():void
+displayMessage():bool
+clickSend():void
delegate
delegate
CertFileHandler
-CertFile:File
+getCertKey():byte[]
SqliteDBHandler
-SqliteDBFile:File
+getUsers():ArrayList<String>
+checkPassword(name:byte[], passHash:byte[]):bool
+checkAdminPassword(name:byte[], passHash:byte[]):bool
+getKey(name:byte[]): key:byte[])
+setKey(name:byte[], key:byte[]): bool
+addUser(name:byte[], passHash:byte[]):bool
+userExists(username:String):bool