Global General Assembly

Summary

A secure, web-based, encrypted text chat system for organizing political movements. The chat system will integrate encryption of text through OTR. Global GA will also incorporate a system for voting and 'temperature checks', as seen in offline general assemblies.

Benefits to the Community

In the past year, revolutions swept the world. A key part of these was the general assembly, a forum for anyone to share their opinions on issues facing the movement. I believe social movements worldwide could benefit from an online version of the general assembly to share ideas with people all over the world, including those who might not be comfortable with the risk associated with attending in-person meetings. Global General Assembly would provide that forum for secure online discussions.

Overall Structure

- Chat Client: A web-based client with basic chat functionality. It allows users to send messages back and forth through the server and directly to other users. The client is primarily in the file Client.java. ClientApplet.java grabs some data from the webpage and initializes the main client.
- Server Software: The chat server software will be deployed on a server. It is responsible tor transmitting messages between connected clients. The server is in Server.java. ServerThread.java also handles specific connected threads for each client.
- Chat Encryption (Client-Side): A Java applet that allows integration of OTR in the browser without additional plugin installs. It encrypts and decrypts messages sent to or from the server. All of this is done automatically, including key generation. Users do not need to understand public key encryption. Keys are stored in HTML5 LocalStorage.
- Chat Encryption (Server-Side): The server receives encrypted messages, decrypts them, and re-encrypts them for users. The big disadvantage here is the server must be trusted and users need to authenticate with the server.
- Voting System: A poll system on the side of the chat that allows users to vote on proposals. This is used as the equivalent of a temperature check in a normal general assembly. It will also include the ability to block proposals the voter believes detrimental to the movement.

Server Structure

Server.java:

public Server (int port)
Data- int port
Purpose- This calls listen(int port).

private void listen (int port)

Data- int port, ServerSocket ss, Socket socket, DataOutputStream dout, String name, int numuser, Hashtable outputStreams, Hashtable users
Purpose- This method listens for incoming connections on the appropriate port with server socket ss, accepts the connections, and gets the output stream. It then saves the output stream dout and the socket s to the hashtable outputStreams. Finally, it assigns new users a temporary nickname by concatenating the string name with an int numuser which represents the number user they are to join. It then adds this name the the hashtable of users.

Enumeration getOutputStreams()
Data- Hashtable outputStreams
Purpose- This helper method returns an enumeration of the output streams.

void sendToAll (String message)
Data- String message, Hashtable outputStreams, Enumeration e, DataOutputStream dout
Purpose- This method takes a message as input and calls getOutputStreams for each client. In this process, it sends the message to each client. This sends a message to all connected users.

void removeConnection (Socket socket)
Data- Socket socket, Hashtable outputStreams
Purpose- removeConnection takes a socket and removes it from the hashtable outputStreams. It then closes the socket.

Public String addUser (String message, String oldname, Socket socket)
Data- String message, String oldname, Socket socket, Hashtable users, String name, Hashtable regnames, DataOutputStream dout
Purpose- This method takes a message that starts with /nick and extracts the new desired name from it and saves it in the string name. It then checks in the hashtables users and regnames to see if the name is already in use or registered. If it is, it sends an error to the user and returns the original name, oldname, to the caller. If the proposed name starts with nick, it also sends an error to the user and returns oldname. All names that start with nick are restricted and automatically assigned. If the name is not in use, is not registered, and does not start with nick, it removes the oldname from the users hashtable and adds the string name. Finally, it notifies all users of the name change and returns the new name to the caller.

Public String registerUser (String message, String oldname, Socket socket)
Data- String message, String oldname, Socket socket, String userpass, String[] userpassarray, Hashtable regnames, String name
Purpose- This method handles registration of users. It takes input of the form /register name password. It saves the name and password in the string userpass and then places them in an array. If the new name is the same as the current name, it adds the name and password regnames. If it is different, it first changes the name by passing the data through addUser and, if successful, adds the name to regnames. It returns the registered name, be it oldname or name.

Public String identUser (String message, String oldname, Socket socket)
Data- String message, String oldname, Socket socket, String userpass, String[] userpassarray, Hashtable regnames, String name, Hashtable users, DataOutputStream dout
Purpose- This method handles identification of registered users. It takes input of the form /identify name password. It saves the name and password in the string userpass and then places them in an array. If regnames contains the name, it checks the password, removes the oldname from the user list, adds the new name, and notifies everyone of the change. It then returns the new name. If regnames contains the name but the password is wrong, it prints an error and returns the old name. If regnames does not contain the new name, it also sends an error and returns the old name.

Public void createRoom (String message, Socket socket)
Data- String message, Socket socket, String room, Hashtable roomlist, int roomnum, ArrayList rooms, DataOutputStream dout
Purpose- This handles the creation of rooms or channels. It takes commands of the form /create roomname. It extracts the room name from the message and checks if it already exists. If the room does not exist, roomnum increments by one and the room and room number are added to the hashtable roomlist. Finally, a new hashtable in the array list rooms is created for the purpose of tracking users in the room. If the room already exists, an error is printed and send to the user.

Public void joinRoom (String message, String currentname, Socket socket)
Data- String message, String currentname, Socket socket, String room, Hashtable roomlist, int roomnum, ArrayList rooms, DataOutputStream dout
Purpose- This handles users joining a channel and takes commands of the form /join channelname. It extracts the channel name from the message, saves it in the string room, and checks if the room exists. If the room exists, it extracts the roomnum from roomlist and checks if the user is already in the room. If the user is already in the room, it sends an error back. If the user is not in the room, it adds the user to the list of users in the room's hashtable in the ArrayList of rooms and notifies all others in the channel. It also sends a message to the client so it can track the rooms the user is in. If the channel does not exist already, it calls createRoom to create the room and then joinRoom to have the user join the room they created.

Public void partRoom (String message, String currentname, Socket socket)
Data- String message, String currentname, Socket socket, String room, int roomnum, Hashtable roomlist, ArrayList rooms, DataOutputStream dout
Purpose- This handles users parting a room and takes commands of the form /part roomname. It extracts the name of the room from the command and checks if the room exists. If the channel exists, it gets the number of the channel's hashtable in the ArrayList rooms then checks if the user is in the room. If the user is in the room, it notifies all users in the channel that the user has left and removes the user from the list of those in the room. If the user is not in the room, it sends an error to the user. If the room does not exist, it also sends an error to the user.

Enumeration getChannelStreams (int roomnumber)
Data- int roomnumber, ArrayList rooms
Purpose- This is similar to getOutputStreams but only gets the output streams of those in a specific channel by accessing the hashtable of users in the arraylist of rooms.

Public void sendToChannel (String message, int roomnumber, String currentname, Socket socket, String roomname)
Data- String message, int roomnumber, String currentname, Socket socket, String roomname, ArrayList rooms, Enumeration e, DataOutputStream dout, DataOutputStream dout2
Purpose- This sends messages to a specific channel. It checks if the sender is in the room first. If the sender is in the room, it calls getChannelStreams for each user in the room and sends the message to each user after adding the channel name and a space to the beginning of the message. If the user is not in the room, it sends an error.

Public void removeUser (String name)
Data- String name, Hashtable users
Purpose- This removes the user name from the hashtable of users. It is used when the user leaves the chat completely.

Public boolean containsKey (String name)
Data- String name, Hashtable users
Purpose- This checks if the hashtable of users includes the user name.

Public static void main (String[] args)
Data- int port, String [] args
Purpose- This saves the port number and starts a new instance of Server.

ServerThread.java:

public ServerThread (Server server, Socket socket)
Data- Server server, Socket socket, String name
Purpose- This saves the server, socket, and name and starts.

Public void run()
Data- Server server, Socket socket, String name, DataInputStream din, String message, int roomnum, int messagelen, String[] chaninfo, String tempmessage
Purpose- This processes messages and normally just passes them on unless they are a command. If the message is a command, it calls the appropriate method in the Server class and passes the appropriate arguments. It does a little extra work for messaging specific channels, where it extracts the command, splits the message into an array, extracts the room number and message length, reconstructs the message, and calls sendToChannel. This method also handles removing users and connections.

Client Structure
Client.java:

public Client (String host, int port)
Data- String host, int port, TextField tf, Socket socket, DataOutputStream dout, DataInputStream din, GridBagLayout gbl, GridBagConstraints gbc, ArrayList<TextArea> taarray, JTabbedPane tabbedPane, JTabbedPane tabbar, Button send, int i
Purpose- This constructs a basic GUI using the GridBagLayout and two TabbedPanes. It adds text areas to one TabbedPane. It also adds a text field and a send button, both with listeners which call processMethod when the button is clicked or the user hits enter in the text field. Finally, this method tries to connect to the server and starts a new thread.

Public void processMessage (String message)
Data- String message, DataOutputStream dout, TextField tf
Purpose- This sends outgoing messages.

Public void run()
Data- String message, ArrayList<TextArea> taarray, DataInputStream din, String[] messarray, int flag, String[] roomlist, JTabbedPane tabbedpane, int i, int j
Purpose- This receives messages from the server. The server sends join;channelname when the user joins a channel. This method adds that name to the roomlist and creates a tabbedpane for messages for that room. Whenever a new message is received, if it starts with channelname followed by a space (this is inserted by the server when it processes messages to send to a channel), it is sent to the tabbedpane associated with that channel. Otherwise, it is sent to the main channel.

ClientApplet.java:

public void init()
Data- String host, int port
Purpose- This extracts the port and host from the html, makes a basic layout, and initializes an instance of the Client class.

globalga.html:

Data- host, port
Purpose- This is simple html code that inserts the client applet into the page and defines the host and port numbers.

Features Completed
Chat:

- The ability for a user to send a message typed into a text box.
- The ability for the server to receive the message.
- The ability for the server to transmit the message to other users.
- The ability for a server to listen for connections.
- The ability for a server to use a port specified in the command line.
- The ability for a client to connect to a server.
- The ability for the server to track connections (add and remove).

Users:

- The ability for the chat server to assign a user a name.
- The ability for the user to select a name.
- The ability for a user to register a name.
- The ability for a name to be associated with a password.
- The ability for a user to authenticate with a password.
- The ability for the server to block registration of nick#.
- The ability for the chat client to not let people use registered names.
- The ability for the chat client to not let people register already registered names.
- The ability for the chat server to check if the name is in use.
- The ability for the server to notify people when someone changes their name.
- The ability for the server to track names.
- The ability for the user to change names.
- The ability for the server to remove names.
- The ability for the chat to display names selected by users.
- The ability to track what rooms each user is in.

Channels:

- The ability for a server to keep track of multiple rooms.
- The ability to create a room.
- The ability to send messages to a specific room.
- The ability for the server to not let people recreate a room.
- The ability to track the users in a room.
- The ability for the server to notify a room when someone leaves.
- The ability for the server to notify a room when someone enters.
- The ability for the server to determine which room messages go to.
- The ability to leave a single room.
- The ability for the server to not allow users to message rooms they aren’t in.
- The ability for a server to not let a user part a room that does not exist.
- The ability to join a room.
- The ability for the server to create the room if a user tries to join a nonexistent room.
- The ability for the server to not let people re-join a room.
- The ability for the server to not let people leave a room they aren’t in.

GUI:

- The ability for the client to display a gridbaglayout.
- The ability for the client to display a text area on the page.

- The ability for the client to display a text field on the page.
- The ability for the client to display received messages in the text area.
- The ability for the client to display text typed in the text field.
- The ability for the client to send the text in the text field to the server when the user presses enter.
- The ability for a user to click a send button and the server to send a message.
- The ability for the client to create a new tab when the user joins a room.
- The ability for messages sent to a room to appear in the appropriate tab.
- The ability for the client to display a tabbed sidebar.
- The client has the ability to display a main tab when the user connects.

Features Needed/In Progress

Chat:

- The ability for one user to send a message to another.

Channels:

- The ability for a user to change their name and people in the rooms they are in to know.
- The server should have the ability to remove an empty room.

GUI:

- The channel tabs should close when the user exits the channel.
- The user should be able to type text in a textbox when in a certain room and have it sent to that room.
- The channel tabs should close when the users clicks an x on them.
- The user has the ability to use a box on the ‘User’ tab to change their name.
- Long lines should wrap in the text area and text field.
- The user tab in the sidebar has the ability to list users.
- The user tab in the sidebar should give the user the ability to go to an authentication interface.
- The user should be able to authenticate.
- The user tab in the sidebar should give the user the ability to go to a registration interface.
- The user should be able to register through the GUI.
- The channel tab on the sidebar has the ability to display a list of rooms.
- The channel tab on the sidebar should have a button that allows the user to leave the room.
- The channel tab on the sidebar should have a text field and button to allow users to search the room list.
- The channel list should allow users to join a room by clicking the room name.

OTR:

- The user will have the ability to authenticate the key with the server.
- The server will have the ability to generate its own key.
- The user will have the ability to generate their own key.
- The client will store its own key.

- The client will have the ability to encrypt messages.
- The server will have the ability to decrypt messages.
- The server will have the ability to encrypt messages before sending them to clients.
- The clients will have the ability to decrypt messages before displaying them.
- Users will have the ability to authenticate keys with other users.

Voting:

- The ability to initiate a vote.
- The ability to select an option when voting.
- The ability to count and display votes.
- The ability to block a vote and provide a reason. Blocks are used for results of votes that could cause someone to leave the movement.

Outstanding Issues

Is there a better way to store passwords?
Where is the best place to store keys?
For private messages, should users communicate through the server or directly?
Anything else I should add?