Final project report

Multicast Chat

by Nurmurat Smagulov and Akmaral Tokbergenova

**Features of our implementation:**

There are some differences in our implementation from the required one. Now all the messages on TCP socket are encrypted by the XOR algorithm as well as on UDP socket. The asymmetric encryption is used once for TCP connection encryption. So, the difference is that the asymmetric encryption is no longer used for group key transmit. Also, there are some new protocol commands. The server is running on localhost.

**Key generation for message encryption:**

First of all, the key of required length is randomly generated with simple class Random from the allowed symbols array. This array includes all the digits and case-sensitive letters. This initial key is shared between the client and the server, however the final key is obtained from this initial key when the message encryption happens.

**Message encryption:**

The message is encrypted by a simple XOR algorithm with an obtained dynamic key. The obtained key depends on the message length and its own length. If the sum of lengths is even, then the obtained key is formed from the even elements of the main key, otherwise it is formed from the even respectively.

The length of the obtained key can be less than the message length. In this case the key is extended by repeated flipping of this key. For instance, the initial shared key is "aB1c23e" and the message is "encrypted text". The length sum is 7 + 14 = 21, so the odd elements are selected for the new key (remember that the counting starts from 0) - "Bc3". The length is 3 and smaller than the length of the message, so the extended key is "Bc33cBBc33cBBc", which matches the message length.

The final stage is an encryption by XOR algorithm, which is taken and modified from <https://github.com/KyleBanks/XOREncryption/blob/master/Java%20(Android%20compatible)/XOREncryption.java>

**XOR key transmission:**

The transmission is handled by RSA encryption algorithm ([http://www.reindel.com/asymmetric-public-key-encryption-using-rsa-java-openssl/),](http://www.reindel.com/asymmetric-public-key-encryption-using-rsa-java-openssl/),%20) where the client generates a XOR key and the server generates key pair. After that the server sends the public key. This public key is restored from the encoded bytes by the client and the client sends the XOR key encrypted by the restored public key to the server. Finally, the server decrypts the XOR key by the its own private key. So, all the messages on TCP socket are encrypted by the shared XOR key.

**Group message transmission:**

Every group has a shared key as was required, but this key is sent directly to the client on TCP socket with previously mentioned XOR algorithm. It looks like an exposed group XOR key is encrypted by the client-server XOR encryption. After the client knows the key, group members can communicate using this group key.

**TCP socket protocol:**

*From client:*

**NAME 'name'** - informs about new client name. The names are not unique.

**JOIN 'group id'** - client asks to join group with this id. The new group is added to the end of the group list

**LEAVE** - client asks to leave the current group

**CREATE 'group name'** - create group with this name. The group name is unique

**DESTROY 'group id' -** removes group from the group list and kicks all the clients inside the group.

*From server:*

**GROUPS 'group amount/group name/client amount/group owner(0 or 1)/group name/client amount/group owner/...** - group list, where 'group owner' means is this client is an owner of this group

**GROUP 'group id/group ip/group port/client amount/client name/client ip/client port/client amount/client name/client ip/client port/.../group XOR key' -** full group information

**LEAVE** - forces the client to leave from the current group

**UDP socket protocol:**

There is no any message from the server

*From client:*

**MSG 'msg/client ip/client port'** - member group says. The member group is identified by TCP socket IP and port (not Multicast socket IP and port).

**JOIN 'client name/client ip/client port'** - event when a client enters a group

**LEAVE 'client ip/client port'** - a client left the group

**Other:**

The uniqueness of the client name and group name is checked on the client side, when the client enters them. If it is not unique, then the dialog box appears again.

When the server is down, the client receives a notification box and the application stops executing.

There are log messages in the console to debug the execution.

**Responsibilities of team members:**

Team manager and programmer - Akmaral

Worked on Server logic and implementation, managed the work of the team and was responsible for design of the project. Also she came up with the encryption algorithm.

Team lead programmer - Nurmurat

Implemented the Client part and base classes, also he made a revision of the code and corrections of this code.