Títol del Treball Fi de Carrera

Roger López Reixach
ETIS

Antoni Martinez Ballesté.

Data Lliurament: 20 de Juny 2004

Resum:

Aquest projecte de fi de carrera tracte de assentar un cicle formatiu iniciat fa més de 4 anys. Com és de suposar, és ben difícil de mostrar tots els coneixements adquirits únicament en un projecte però si donar una petita visió del fet fins ara.

En concret, degut a la meva vocació per el àmbit de les comunicacions he decidit, dintre dels projectes oferts, el de realitzar un xat segur.

Dintre d'aquest projecte s'engloben coneixements relacionats amb moltes de les assignatures estudiades en la carrera com criptografia, sistemes de comunicació, programació orientada a objectes i un llarg etcetera.

A la vegada, vull ressaltar el àmbit d'aquest projecte, igualment que totes les assignatures, és l'aprenentatge i per tant el propòsit d'aquest projecte s'allunya de un projecte "comercial". Amb tot això vull referir-me, que no s'ha tingut en compte molts aspectes i si que hi ha hagut un especial interès en aspectes concrets com poden ser els relacionats amb els processos de comunicació.

Per concloure aquest petit resum inicial, només dir que el projecte no queda del tot acabat (si el demanat per l'assignatura) degut a les moltes altres possibilitats que ja he començat i que comentaré mes endavant.

Índex.

1.Introducció:

Àmbit del TFC:

Segons l'enunciat del projecte, es tracte de realitzar un aplicatiu que permeti la comunicació (xat) entre un grup de persones mitjançant internet. Per tant, aquest és el primer punt a remarcar, ha de ser una aplicació multi-usuari. Cada usuari del xat es connecta a un servidor qualsevol (el servidor no està centralitzat) indicant la seva direcció. Aquest servidor donarà suport per tal de que els clients connectats puguin comunicar-se en el xat. El client serà l'encarregat de demanar connexió al servidor i donar l'entorn gràfic a l'usuari per tal de realitzar les tasques xat.

Per tal de realitzar una comunicació segura, hem de realitzar l'encriptació de les dades transmeses. Aquest punt és una de les finalitzats principal del projecte :l'encriptació i desencriptació de dades. Per tal de realitzar aquesta encriptació el client i el servidor intercanviaran una clau de sessió al inici de la comunicació que servirà per a mantenir la confidencialitat de les dades transmeses.

Aquest aplicatiu respon a les exigències del enunciat. No obstant he iniciat el desenvolupament d'un Xat caracteritzat per la utilització de Certificats.(veure apartat de Certificació). De totes formes no he tingut temps de finalitzar aquesta ampliació juntament amb altres que comentaré. L'objectiu d'aquesta idea és obtenir les claus pertinents per a autentificació i codificació a partir d'un certificat autoritzat per una organització de certificació.

Queden fora del àmbit del programari moltes altres funcionalitats, que no obstant poden enriquir la qualitat del programa. De idees hi ha varies, com per exemple servidor de traducció d'adreces IP, que tradueixi a partir de noms (com pot ser el nom d'un amic amb el que volem realitzar la conversa) direccions IP. Una altra ampliació molt interessant és incloure la possibilitat de transmetre fitxers mitjançant el nostre xat.

Objectius del TFC:

Com a resposta a l'enunciat d'aquest projecte, els objectius del mateix es centren en :

- Establir una connexió client/servidor.
    i. Connexió mitjançant sockets.
    ii. Intercanvi de claus de sessió.
- Xifratge de dades.

Enfocament i mètode seguit:

L'enfocament del projecte radica en la seva mòdulilitat. Així dons he enfocat el projecte per mòduls que una vegada provats formen part del mateix programari. Així dons, primerament he donat solució als processos de comunicació sense incloure cap tractament de xifratge ni intercanvi de claus.

Com que estem utilitzant processos que es comuniquen mitjançant una xarxa IP em de fer ús dels sockets per tal de poder donar sortida de la nostra aplicació al punt de la xarxa desitjat. Així dons en primer lloc hem d'utilitzar la llibreria de sockets.

Per tal de resoldre l'intercanvi de claus he utilitzat la generació de la clau secreta (protocol keyagreement) Diffie-Helman.

Per a xifrar les dades he utilitzat l'implementació del xifratge simètric DES (DataEncrytionStandard).

Per acabar he donat solució a un petit entorn gràfic per tal de facilitar el programa a l'usuari final.

Com a resum podem concloure la mòdulitat del programa:

- Connectivitat entre processos client/servidor.

- Intercanvi de claus.

- Xifratge de dades.

- Entorn gràfic.

2. Descripció del funcionament del programa

Com ja he comentat, podem dir que es tracte de dos programes completament independents entre si. Un d'ells fa la funció de servidor mentre que l'altre respon a les tasques de client.

Servidor:

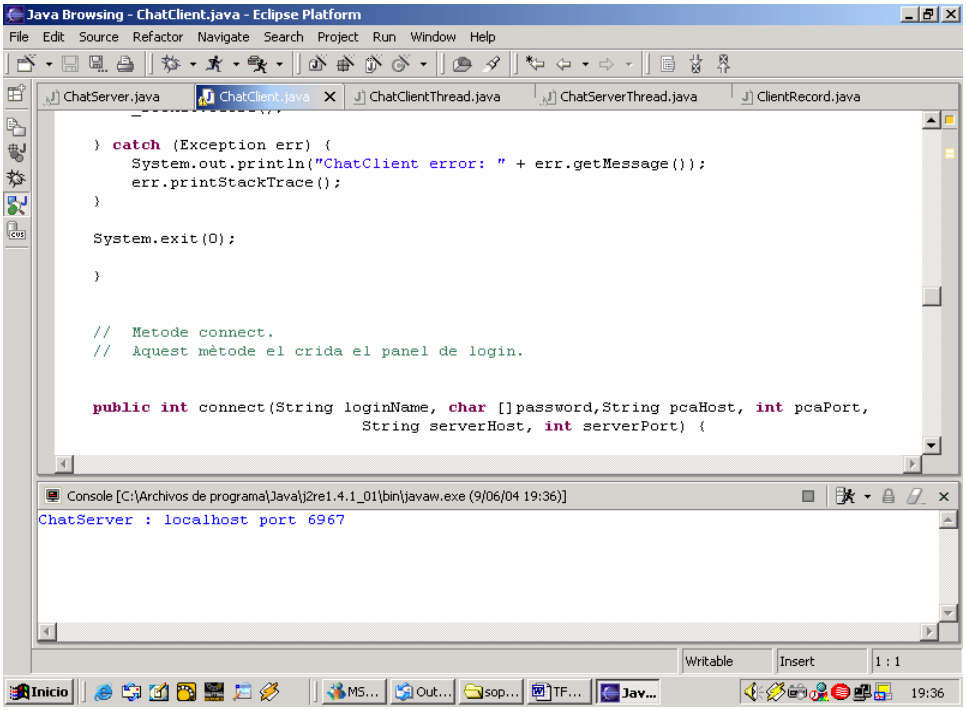
En primer lloc, hem de tenir un servidor a l'escolta. Cap client no pot iniciar la seva execució sense ser acceptat per un servidor. Per tant, aquest ha de ser el primer en ser executat. Tan el client com el servidor poden tenir múltiples Threads independents. Això es tradueix bàsicament en la possibilitat de tenir amb el mateix programa, múltiples servidors escoltant ports diferents (escoltar el mateix està tractat com a excepció).

El funcionament del Servidor és el següent.

Java ChatServer <port> on port com a argument que descriu el nº de port.

En el cas de no introduir cap argument hi haurà un error tractat amb la descripció dels arguments vàlids. En el cas de que el port no es pugi escoltar també obtenim per la sortida estandard el missatge que descriu l'error.

Una vegada llançat el programa, obtenim una visualització del seu funcionament mitjançant la consola de sortida



* En aquest cas hem executat l'aplicació amb l'entorn Eclipse (eina de treball utilitzada en tot el projecte).

** Observem el missatge per la consola que descriu el funcionament del programa sobre el port 6967.

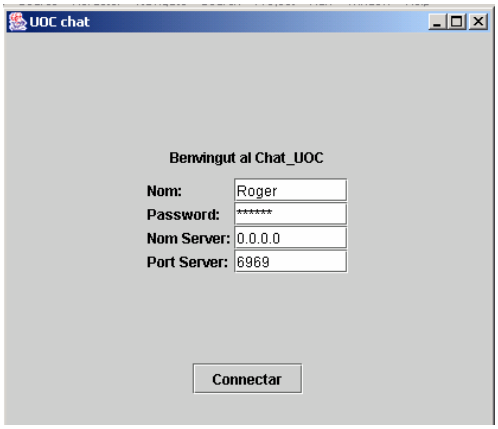
Client:

Una vegada executat el Servidor, aquest queda escoltant un port. Ara qualsevol client que es connecti a aquest port i a la adreça on resideix el programa servidor, iniciarà la sessió amb aquest servidor.

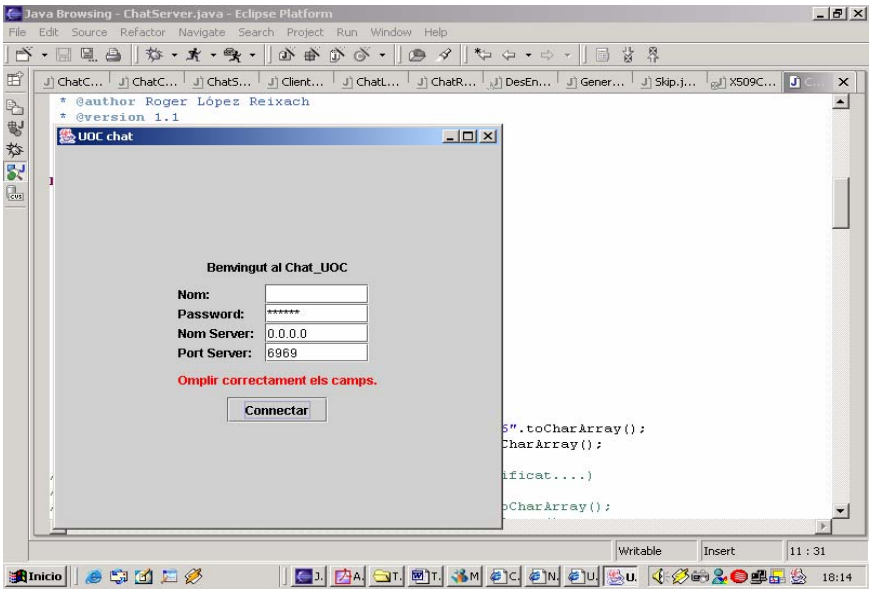
Per tal de iniciar el Client farem:

Java Client.

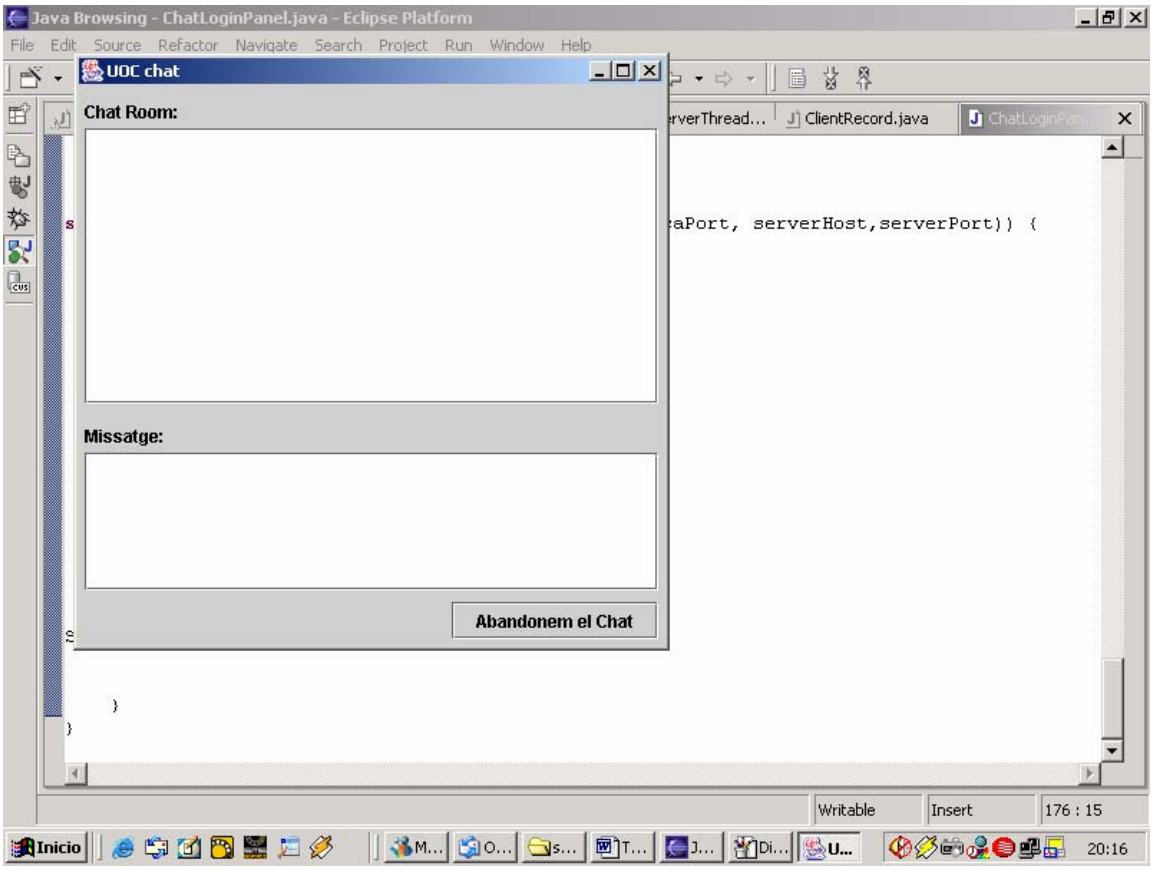
Per començar tindrem un LoginPanel.



Apareix un Applet on ens permet introduir els atributs per a la nostra aplicació client.
En primer lloc tenim el Nom (nom d'usuari) i el seu password. Aquest password l'utilitzarem com un petit control de seguretat al programa.(hem de contemplar que no tots els fraus provenen de l'exterior).
Tenim el nom del server que respon a la direcció IP que ens volem connectar, el seu port al que volem accedir.
Per tal de connectar al servidor fem un click al botó Connectar.
En cas d'introduir un atribut erroni o buit es genera l'error pertinent:



Una vegada connectats correctament al servidor ens apareix el ChatRoomPanel.



On Tenim dues finestres de text.
La finestra superior ens mostra els diàlegs de la Room mentre que la finestra inferior ens permet escriure els nostres missatges.
Per finalitzar el xat prenem el botó Abandonem el xat.

Joc de proves realitzat:

Servidor:

Per tal d'assegurar el funcionament correcte del ChatServer, el servidor no pot :

Connectar-se a un port ocupat ni reservat. En el cas de intentar-ho dona el següent error:

```
Error escoltan el port: 6969
```

Així dons, dos servidors no poden conviure en el mateix port i en un mateix host ,però si en diferents ports dins d'un mateix host.

Client:

Respecte al client no es pot connectar a un port on no hi hagi cap servidor escoltant-ho.

En cas de intentar-ho dona el següent error:

```
Error en els Streams E/S conectant al server: 0.0.0.0
java.net.ConnectException: Connection refused: connect
```

3.Classes implicades.

Breu descripció de les classes:

Anem ara a descriure la utilitat de les classes que formen el nostre programari.

Tenim un total de 10 classes desenvolupades

Tenim les classes:

ChatClient: Representa la classe que inicia l'aplicatiu per al client del Xat
ChatClientThread: Representa un fil d'execució (Thread) del client
ChatServer: Representa la classe que inicia l'aplicatiu servidor.
ChatServerThread: Representa un fil d'execució del servidor.
ClientRecord: record dels clients en execució
ChatLoginPanel: Login panel (panel d'entrada)
ChatRoomPanel: Room de conversa del nostre Xat
Skip: Classe representativa dels paràmetres Diffie-Hellman.
DesEncrypter: Classe que encapsula la encryptació/desencryptació de les dades

Genera_clau: Classe (només amb caràcter il·lustratiu)del procés Diffie-Hellman

Analitzem amb més profunditat les classes més representatives de l'aplicatiu.

Classe ChatClient

Aquesta classe, com ja hem comentat, representa el client del servei de xat. En primer lloc, aquesta classe és la responsable de llençar l'entorn gràfic del login que ens permetrà introduir les dades de direcció del servidor, password (mètode initComponents).

```
private void initComponents() throws Exception {
```

Aquest mètode és l'encarregat de d'instanciar el ChatLoginPanel i el ChatRoomPanel.
Una vegada inicialitzat les classes esmentades i introduït les dades necessàries per a la connexió, el mètode connect es cridat.
Aquest és el mètode que realment representa tot el procés de connexió. En primer lloc realitza una comprovació de l'atribut password i comprova la seva validesa.

Per una banda establim connexió amb el servidor desitjat:

```
_socket = new Socket(serverHost, serverPort);
DataOutputStream out = new
DataOutputStream(_socket.getOutputStream());
DataInputStream in = new DataInputStream(_socket.getInputStream());
```

Una vegada establert connexió amb el servidor comencem la fase d'intercanvi de claus (Diffie-Hellman):

En primer lloc generem les claus:

```
KeyPairGenerator kpg = KeyPairGenerator.getInstance("DH");
kpg.initialize(Skip.sDHParameterSpec);
KeyPair keyPair = kpg.genKeyPair();
```

La classe KeyPairGenerator ens crea el parell de claus (pública i privada) i les retorna en format de parella de claus KeyPair.
Una vegada generat la parella de claus les intercanviem la clau pública amb el servidor per el canal obert:

```
byte[] keyBytes = keyPair.getPublic().getEncoded();
out.writeInt(keyBytes.length);
out.write(keyBytes);
```

I rebem la clau corresponent al servidor.

```
keyBytes = new byte[in.readInt()];
in.readFully(keyBytes);
```

Generem una instància de la classe KeyFactory (java.security.KeyFactory) amb l'algorisme Diffie-Hellman com a paràmetre:

```
KeyFactory kf = KeyFactory.getInstance("DH");
X509EncodedKeySpec x509Spec = new X509EncodedKeySpec(keyBytes);
PublicKey tPublicKey = kf.generatePublic(x509Spec);
```

Ara ve quan realitzem el KeyAgreement. KeyAgreement és un protocol on varis integrants d'una comunicació és posen d'acord amb una clau secreta. Hi ha molts tipus d'aquest tipus de protocols. Hem utilitzat el protocol de Diffie-Hellman que està encapsulat dintre de javax.crypto.KeyAgreement:

```
KeyAgreement ka = KeyAgreement.getInstance("DH");
ka.init(keyPair.getPrivate());
ka.doPhase(tPublicKey, true);
claus = ka.generateSecret(algorithm);
```

Encriptació dins la classe ChatClient:

Una vegada explicat el procés de generació de claus, ja podem donar una idea les classes implicades en la encriptació dels textos xifrats que enviem al servidor.

```
DesEncrypter encrypter = new DesEncrypter(claus);
```

DesEncrypter és la classe encarregada donar els mètodes necessaris per tal d'encriptar un text en clar. Només en cal un Cipher encarregat d'encriptar les dades.

Crearem una instància del Cipher per tal d'encriptar amb l'algorisme de xifratge DES:

```
ecipher = Cipher.getInstance("DES");
ecipher.init(Cipher.ENCRYPT_MODE, key);
```

Encriptarem/Desencriptarem amb aquest xifrador.

Una vegada fet el procés explicat, el client es dona per acceptat (Per tant actualitzem la classe ClientRecord. Aquesta classe s'encarrega d'emmagatzemar les dades dels clients connectats.
Molt important és, una vegada realitzat el procés, llençar el thread client que continuarà la seva execució independent i persistent.

```
_thread = new ChatClientThread(this);
_thread.start();
```

Així dons, he resumit les classes més importants utilitzades dins de ChatClient.

ChatServidor:

La classe ChatServidor és la responsable de donar funcionalitat al servidor del nostre Xat. Així dons, podem considerar el chatServidor com un servei o aplicació resident. Una vegada el ChatServidor es llançat, resta a l'escolta de totes les connexions que es realitzen al Host i al port corresponent. Anem amb més detall a explicar més detalladament el funcionament bàsic d'aquesta classe i les classes més importants implicades en aquest procés.
Es pot observar que hi ha molta similitud entre la especificació del servidor i la del client. En primer lloc el servidor obre un punt (ServerSocket) de connexió. A diferència del ChatClient on s'utilitzava la classe Socket, en el servidor utilitzarem la classe ServerSocket que dona funcionalitat de servidor desitjada:

Instanciem el socket del server:

```
while (true) {
_serverSocket = new ServerSocket(_port);
Socket socket = _serverSocket.accept();
DataOutputStream out=new DataOutputStream socket.getOutputStream();
DataInputStream in= new DataInputStream(socket.getInputStream());
```

En aquest moment, el socket s'obre en escolta del port permanentment (dins d'un blucle while).
Una vegada acceptat el socket del client, capturem la seva clau publica (procés descrit en chatclient intercanvi de claus). Realitzem el mateix procés de generació de claus descrit en ChatClient. Una vegada realitzat el procés d'intercanvi de claus, s'accepta el client registrant-lo:

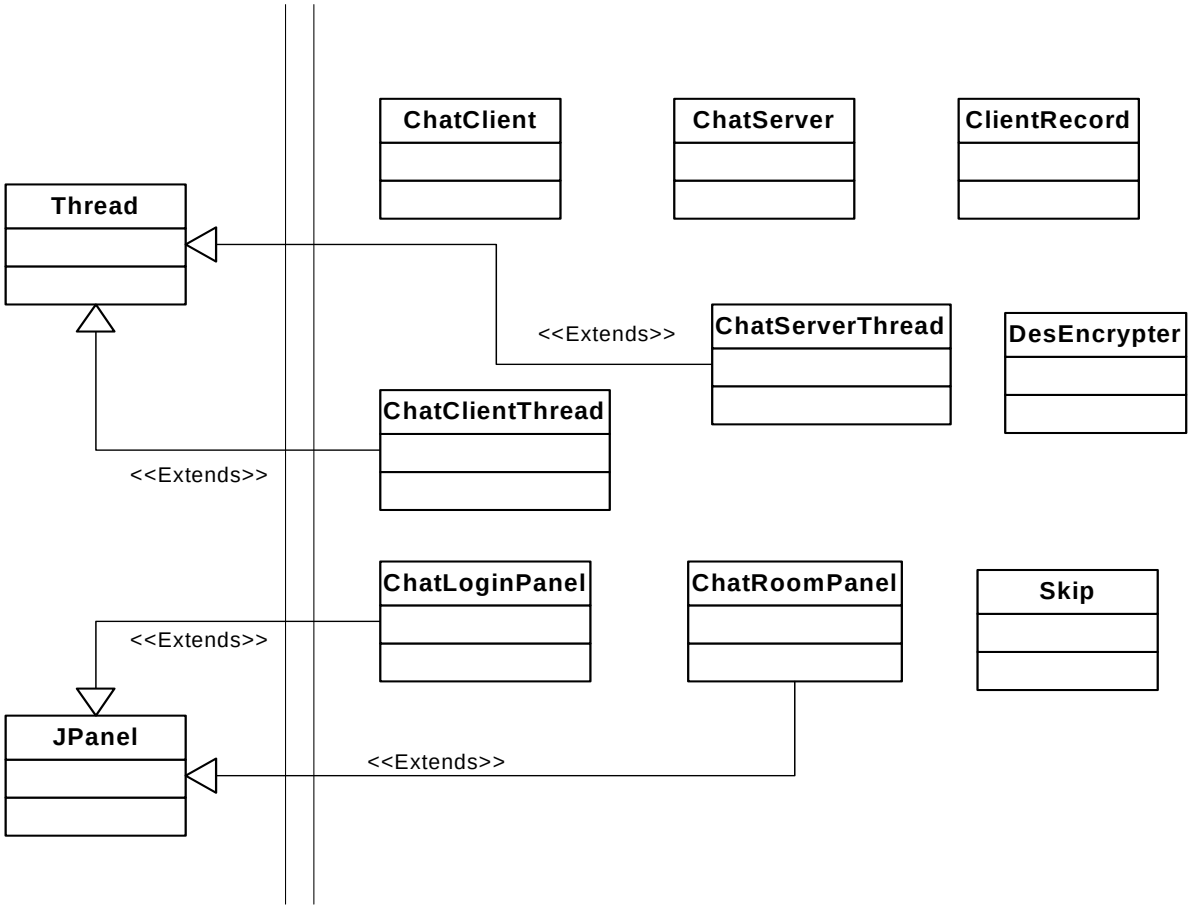
```
ClientRecord clientRecord = new ClientRecord(socket,claus);
_clients.put(new Integer(_clientID++), clientRecord);
```

Només resta, llençà el fil d'execució que es mantindrà a l'escolta.

```
ChatServerThread thread = new ChatServerThread(this, socket);
thread.start();
```

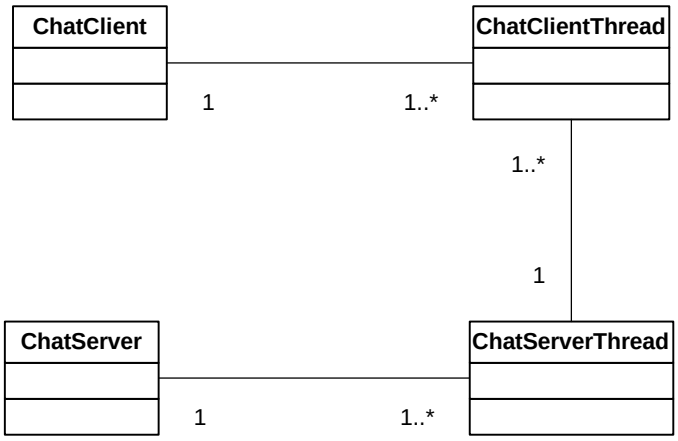
3.1 Diagrama general de classe.

Primerament, mostrem el diagrama general bàsic de les classes i les seves herències:



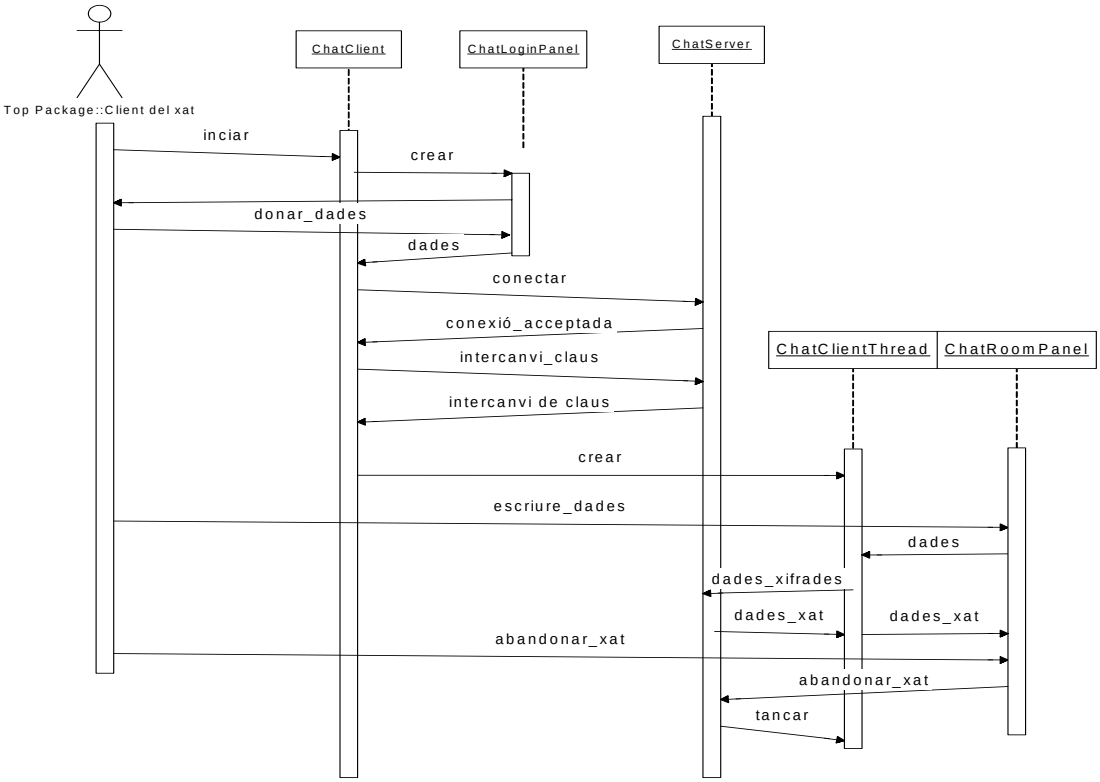
El més remarcable de la representació de les classes de l'aplicatiu és el fet de com ChatRoomPanel, ChatLoginPanel, així com ChatClientThread són fruit d'una herència per extensió.

Anem a analitzar amb un diagrama les associacions entre client/servidor :



Un servei de servidor (ChatServer) pot mantenir una associació amb infinitat de Clients, mentre que un client només pot pertànyer a un únic servidor. Aquest és el fet més remarcable del diagrama anterior.

Per acabar amb els diagrames UML mostraré un diagrama de seqüències de la classe ChatClient.
En aquest diagrama podrem observar el diàleg existent entre les diferents classes al establir la connexió, així com resta la connexió una vegada establerta.
Finalment indicarem la finalització del Client amb les missatges pertinents per tal de concloure el Xat.


Top Package::Client del xat
ChatClient
ChatLoginPanel
ChatServer
ChatClientThread
ChatRoomPanel
inciar
crear
donar_dades
dades
conectar
conexió_acceptada
intercanvi_claus
intercanvi de claus
crear
escriure_dades
dades
dades_xifrades
dades_xat
dades_xat
abandonar_xat
abandonar_xat
tancar

4. Anàlisis conectivitat processos client/servidor.

Estudi conectivitat.

La classe Socket: Descripció:

Com ja he dit avanç, per tal de realitzar la comunicació entre client/servidor he utilitzats els sockets. Els sockets són els punts finals de la comunicació entre processos. La gran avantatge del disseny amb sockets és la seva gran facilitat d'utilització.
La nostra aplicació client/servidor funciona com la majoria d'aplicacions d'aquest estil, és a dir, el servidor escolta un port determinat a la espera de que un client sol·liciti un establiment de comunicació. Faig un breu resum dels dos costats de la comunicació. La classe Socket forma part del paquet java.net de les llibreries SDK i dins del client l'instanciem definint el Host servidor i el Port del server on ens volem connectar.

```
_socket = new Socket(serverHost, serverPort);
```

D'altra banda el servidor generem un ServerSocket que tindrà com a argument el número de port a escoltar.

```
_serverSocket = new ServerSocket(_port);r
```

Així dons, tant en el client com en el servidor s'ha establert la connectivitat. Una vegada tenim els punts de connexió podem iniciar la comunicació utilitzant els Streams d'entrada i de sortida del socket:

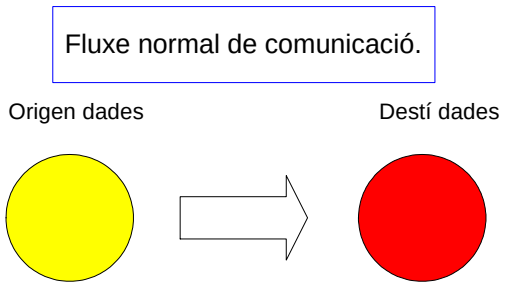
```
out = new DataOutputStream(_socket.getOutputStream());
in = new DataInputStream(_socket.getInputStream());
```

Estudi de seguretat:

Introducció a la seguretat en processos de comunicació:

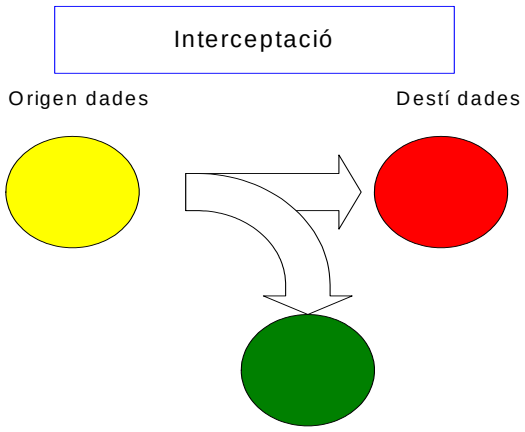
Els sockets ens donen l'instrument per tal de realitzar comunicació entre processos. Ara be, aquesta comunicació no és ni de bon tros segura. Hem de conèixer l'existència dels possibles "atacs" que pot rebre les nostres dades.

Flux normal de les dades:

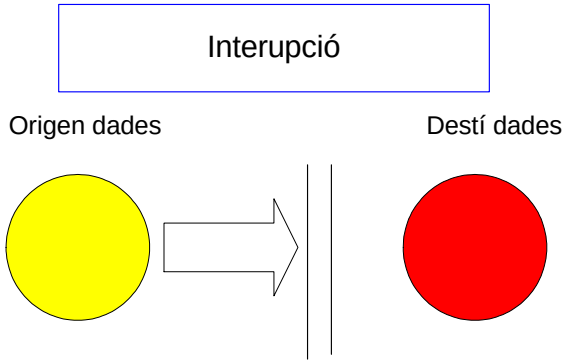


Hi ha molts tipus d'atacs que pot rebre la nostra comunicació:

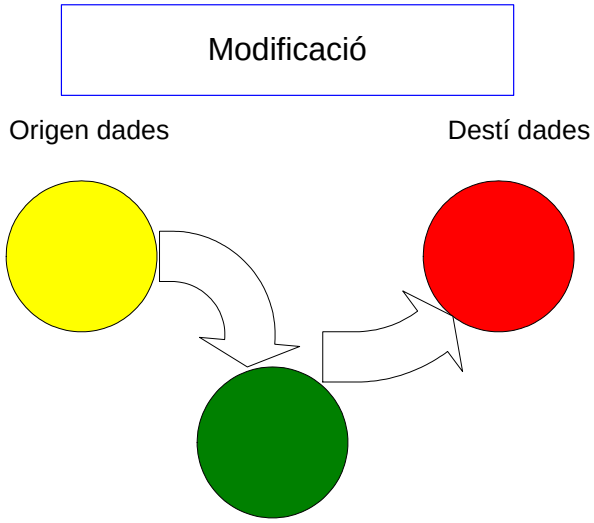
Intercepció de les dades.



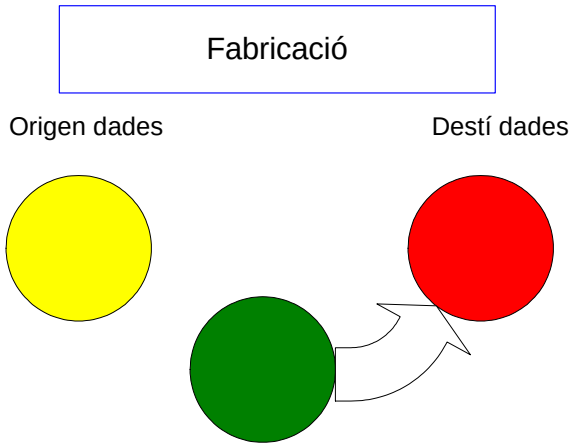
Interrupció de les dades.



Modificació.



Fabricació de dades falses.



Seguretat en el nostre Xat:

Hem de donar una confidencialitat ,autentificació a les dades (a més a més he donat un petit control d'accés "password" a la aplicació).

Analitzem dons els mecanismes de seguretat que s'han aplicat al aplicatiu:

Autentificació del client/servidor mitjançant un intercanvi de claus al inici de la connexió.

Xifratge de dades mitjançant una clau secreta (xifratge DES).

Per tal de comprovar la seguretat del nostre xat cal realitzar unes proves que simulin possibles atacs anteriorment descrits. Per tal de fer-ho, podem instal·lar un sniffer que ens permet escoltar ports de comunicació. Per tal de donar una comunicació segura entre client i servidor hem de donar una sèrie de solucions a aquestes amenaces.

Atac home enmig: De les proves realitzades cal destacar la feblesa de la nostra implementació respecte aquest atac. (la solució del qual pot ser la iniciada, implementar el nostre xat amb suport de la certificació.)

DES: També he comentat , actualment la feblesa del algorisme de codificació DES degut a la longitud de la seva clau. Actualment es considera un algorisme dèbil degut a les possibilitats de computació actuals.

Intercanvi de claus client/servidor:

Els dos participants en la comunicació (client/servidor) utilitzen el protocol d'acceptació de claus per generar una clau secreta (per encriptar les dades) idèntica sense transmetre en cap moment la clau secreta (SecretKey). El protocol està basat en l'algorisme de Diffie-Hellman. Per tant, gràcies a aquest protocol podem establir en un medi insegur (internet per exemple) una clau secreta.

Diffie-Hellman:

Aquest protocol d'acceptació de claus data de 1976 i es amb diferència el més popular dins de la critografia.

-Primerament es generen els valors (base, mòdul primer, exponent).

```
public static String genDhParams() {
      try {
            AlgorithmParameterGenerator paramGen =
            AlgorithmParameterGenerator.getInstance("DH");
            paramGen.init(1024);

             // Generem els  parametres
            AlgorithmParameters params =
            paramGen.generateParameters();

            DHParameterSpec dhSpec
                  =
            (DHParameterSpec)params.getParameterSpec(DHParameterSpe
            c.class);
            return
            ""+dhSpec.getP()+","+dhSpec.getG()+","+dhSpec.getL();

      } catch (NoSuchAlgorithmException e) {
      } catch (InvalidParameterSpecException e) {
      }
      return null;
  }
```

*Aquesta no ha sigut l'implementació final. Es de caire explicatiu.

- Una vegada hem obtingut els valors (ambdos coneixen la base i el mòdul), s'escull un valor aleatori x i es computa :

$$y = g^x \bmod p$$

- Aquest valor es s'envia a ambdues bandes.

- Ambdues bandes calculen: $k_m = y_r^{x_m} \bmod p$ que correspon al mateix valor que $g^{x_r x_m} \bmod p$.

Amb tot aquest procés, les dues bandes han calculat per separat el valor Km que correspondrà al valor secret.

A nivell de seguretat, aquest protocol ens garanteix una clau secreta per tal de poder encriptar les dades a transmetre, només coneixen els valors g i p.

De totes formes, aquest protocol només és vàlid per construir una clau de sessió (no és vàlid per autentificar-se mútuament).

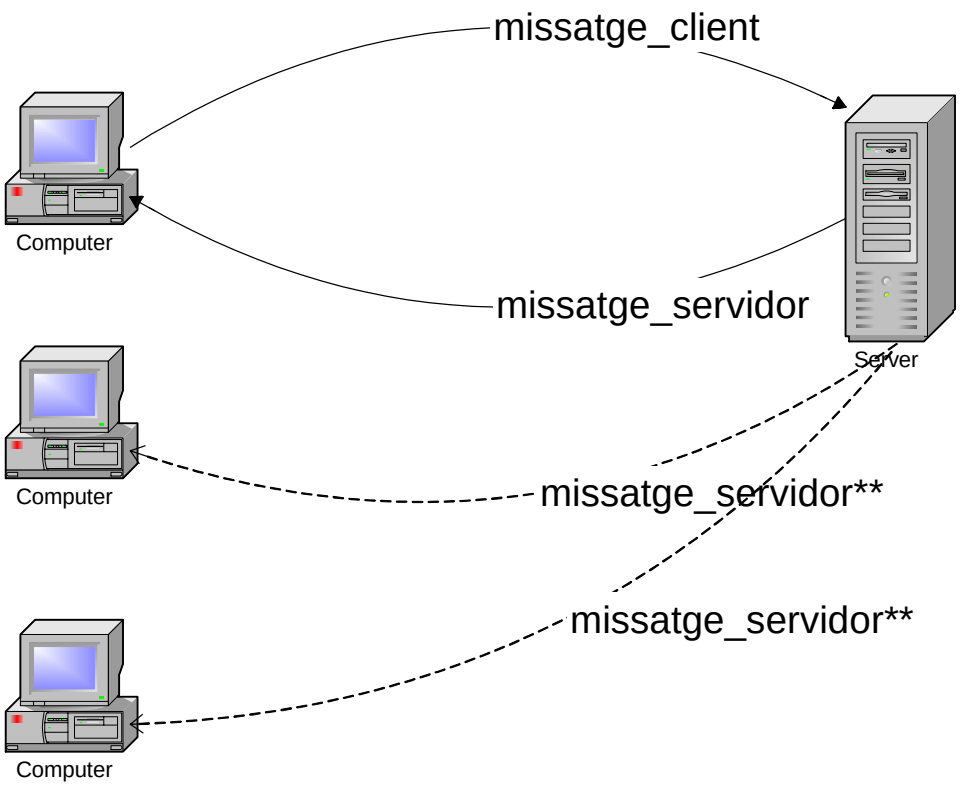
Aquest protocol és aplicable a una sessió multi-client (únicament és necessari generar Threads que escoltin les claus dins de la classe Skip).

En l'aplicatiu desenvolupat hi ha una generació de clau secreta entre (client-servidor), és a dir, un client negocia la clau individualment amb el servidor.

Es pot observar com la comunicació entre amb dos es xifra amb aquesta clau. Es responsabilitat del servidor la necessitat d'encriptar els missatges als altres participants.

En concret s'ha implementat (per veure ambdues possibilitats) com els missatges entre client-servidor són encriptats mentre que el broadcast a la resta de clients del xat es fa amb missatges en clar.

Figura comunicació:



**Aquests missatges poden ser encriptats o no segons les nostres necessitats.

Xifratge de dades transmeses:

Una vegada hem realitzat l'intercanvi de la clau de sessió, podem realitzar l'encriptació/desencriptació de les dades.

Introducció encryptació:

L'encryptació és una eina que podem utilitzar per protegir les nostres dades secretes. Podem encryptar fitxer del nostre ordinador per preveure perdues o robatoris que posin en perill la integritat dels nostres interessos.

Per tal d'encryptar, utilitzarem l'anomenat Cyphers (Xifradors). Hi ha de tres tipus:

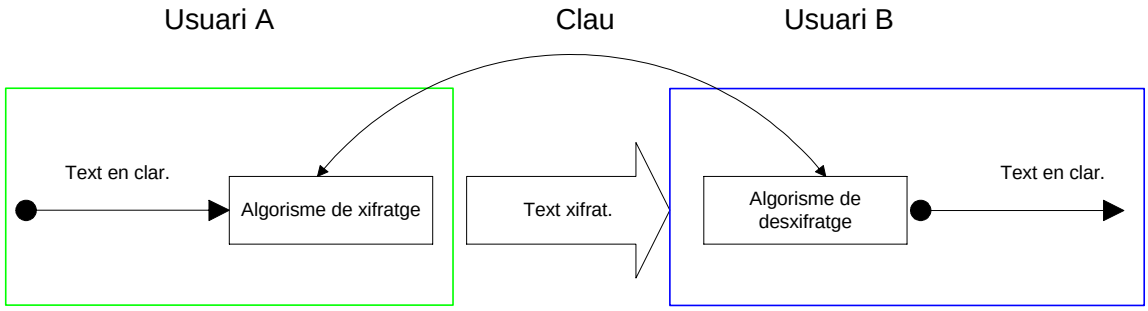
Xifrador simètric: Utilitza una única clau secreta per encrytar/desencryptar les dades.

Xifrador asimètric: El xifrador utilitza un parell de claus (keypair) per tal de codificar les dades. Una clau és pública i pot ser lliurada en clar mentre que la clau privada no es enviada.

Xifrador híbrid: Aquest tipus de xifrador resulta de la combinació dels dos xifratges esmentats anteriorment.

Utilitzem un dels anomenats algorismes d'encriptació de xifratge simètric. En concret escollim el DES (Data Encryption Standard).
Aquest algorisme va néixer (va ser adoptat)al 1077 per el NIST (National Institute of Standard and Technology). Es caracteritza per el xifratge de les dades en blocs de 65 bits i per tindre una longitud de clau de 56 bits. Aquesta darrera dada cobra molta importància a l'hora d'avaluar la seguretat del algorisme. En concret, avui en dia es considera un algorisme no del tot segur degut al petit volum relatiu de la seva clau. Per donar solució a aquest problema han sorgit variants d'aquest algorisme com pot ser el Triple-Des on la clau té una longitud de 168 bits.

Diagrama de codificació de la informació.



Descripció del algorisme DES:

El algorisme DES xifra la informació per blocs (xifratge de blocs). El text en clar s'introdueix en divisions de 64 bits que seran encriptats un darrera d'un altre. Com avanç he mencionat, DES utilitza claus de 56 bits, tot i que solen distribuir-se en forma de números de 64 bits. D'aquest 64 bits, un de cada buit s'utilitza de bit de paritat.

L'algorisme DES segueix el següent esquema:

```
                    Bloc d'entrada 64 bits
*                           |
*                        -----
*                        | A |
*                         -----
*                           |
*                  ------------------
*                  | 16 iteracions |
*                  ------------------
*                    |
*                   -----
*                  | B |
*                   -----
*                  Bloc de sortida 64 bits
```

Permutacions:

Després de rebre cada bloc d'entrada de 64 bits, primerament s'aplica a cada un una permutació mitjançant la taula següent:

```
*
*                    --------------------------
*                    | Taula de permutació A |
*                    --------------------------
*                    | 58 50 42 34 26 18 10 2 |
*                    | 60 52 44 36 28 20 12 4 |
*                    | 62 54 46 38 30 22 14 6 |
*                    | 64 56 48 40 32 24 16 8 |
*                    | 57 49 41 33 25 17  9 1 |
*                    | 59 51 43 35 27 19 11 3 |
*                    | 61 53 45 37 29 21 13 5 |
*                    | 63 55 47 39 31 23 15 7 |
*                    --------------------------
```

El primer bit de la entrada serà ficat en la posició 58, el segon en la posició 50....

Segona permutació:

Després de les permutacions avanç descrites, es realitza una altre permutació mitjançant la taula següent:

```
*
*                    --------------------------
*                    | Taula de permutació B |
*                    --------------------------
*                    | 40 8 48 16 56 24 64 32 |
*                    | 39 7 47 15 55 23 63 31 |
*                    | 38 6 46 14 54 22 62 30 |
*                    | 37 5 45 13 53 21 61 29 |
*                    | 36 4 44 12 52 20 60 28 |
*                    | 35 3 43 11 51 19 59 27 |
*                    | 34 2 42 10 50 18 58 26 |
*                    | 33 1 41  9 49 17 57 25 |
*                    --------------------------
*
```

El primer bit de l'entrada serà de nou ficat en la posició 40, el segon a la posició 8.....

16 iteracions:

Després de la permutació A y avanç de la permutació B, el algorisme DES realitza 16 iteracions. Cada iteració realitza:

2. Els 64 bits d'entrada es divideixen en dos parts de 32 bits.
3. La meitat dreta s'introdueix en la funció F (descrit més endavant).
4. Es realitza una funció XOR entre la sortida de la funció i la part esquerra.

El resultat (32 bits) passarà a ser la part dreta de la següent iteració , mentre que la part dreta actual passarà a ser la part esquerra.

*
*
•

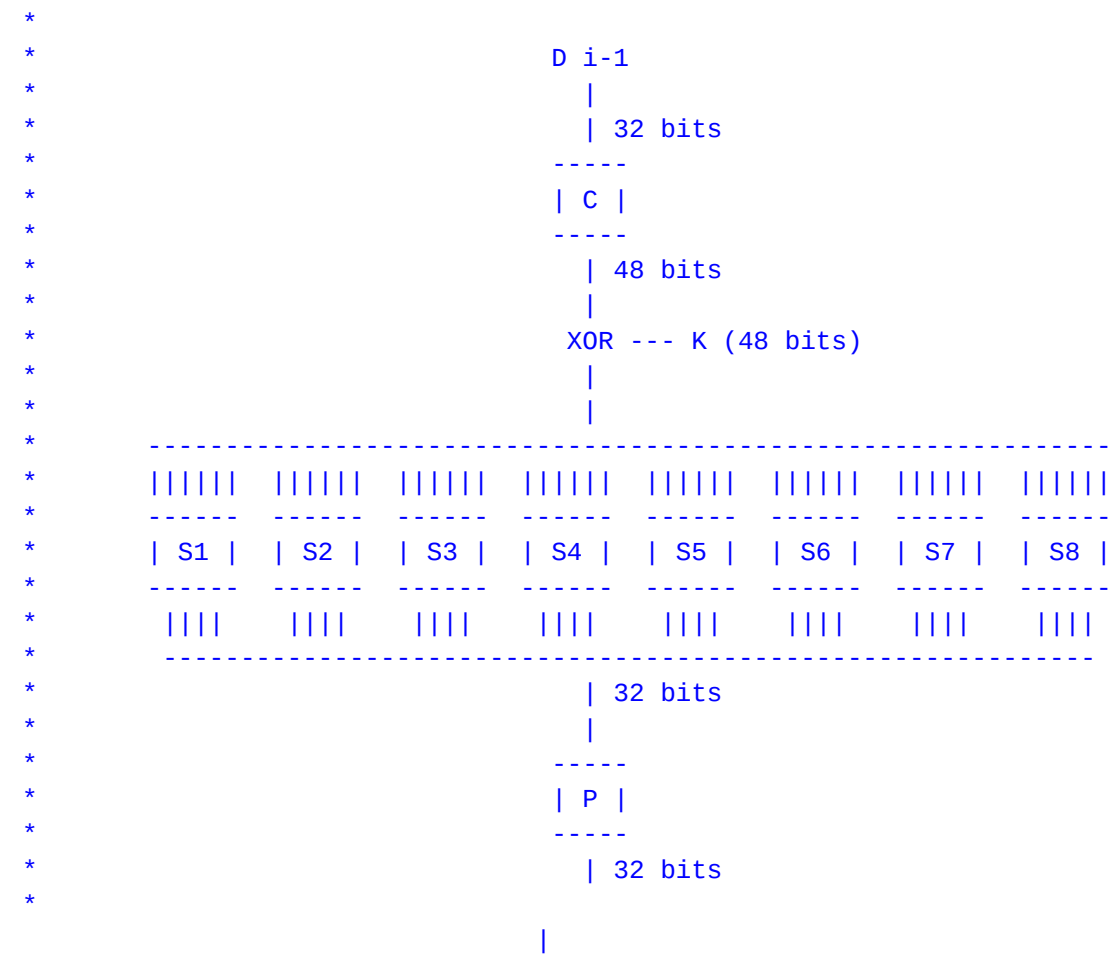
$$L_{i+1} = R_i$$

$$R_{i+1} = L_i \text{ XOR } f(R_i, K)$$

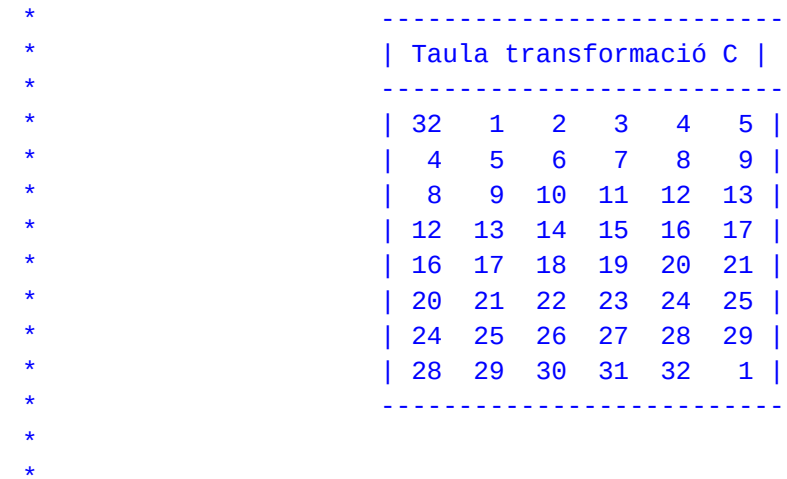
En la última iteració (i=16) no es realitza aquest últim pas descrit.

Funció F

Descrivim la funció avanç esmentada:



Els 32 bits de la part dreta entra en la funció f. El primer pas consisteix en transformar l'entrada de 32 bits a 48 bits mitjançant la taula C.



A continuació es realitza una funció XOR amb una subclau de 48 bits.

Donat que la clau inicial era de 56 bits (representada com a 64 bits) es necessari generar, a partir d'aquesta, subclaus de 48 bits.

Tot seguit, els 48 bits es divideixen en blocs de 6 bits, cada un dels quals es utilitzat com a entrada de l'anomentat com una caixa S.

Els 6 bits que formen cada bloc b1,b2,b3,b4,b5 i b6 son utilitzats per llegir cada caixa S. B1 y b6 indica la fila, mentre que b2,b3,b4 y b5 indiquen la columna.

A continuació es mostra les caixes S descrites :

```
*
*
*               S-1
*             ------------------------------------------------
*             | 14 04 13 01 02 15 11 08 03 10 06 12 05 09 00 07 |
*             | 00 15 07 04 14 02 13 01 10 06 12 11 09 05 03 08 |
*             | 04 01 14 08 13 06 02 11 15 12 09 07 03 10 05 00 |
*             | 15 12 08 02 04 09 01 07 05 11 03 14 10 00 06 13 |
*             ------------------------------------------------
*
*               S-2
*             ------------------------------------------------
*             | 15 01 08 14 06 11 03 04 09 07 02 13 12 00 05 10 |
*             | 03 13 04 07 15 02 08 14 12 00 01 10 06 09 11 05 |
*             | 00 14 07 11 10 04 13 01 04 08 12 06 09 03 02 15 |
*             | 13 08 10 01 03 15 04 02 11 06 07 12 00 05 14 09 |
*             ------------------------------------------------
*
*               S-3
*             ------------------------------------------------
*             | 10 00 09 14 06 03 15 05 01 13 12 07 11 04 02 08 |
*             | 13 07 00 09 03 04 06 10 02 08 05 14 12 11 15 01 |
*             | 13 06 04 09 08 15 03 00 11 01 02 12 05 10 14 07 |
*             | 01 10 13 00 06 09 08 07 04 15 14 03 11 05 02 12 |
*             ------------------------------------------------
*
*               S-4
*             ------------------------------------------------
*             | 07 13 14 03 00 06 09 10 01 02 08 05 11 12 04 15 |
*             | 13 08 11 05 06 15 00 03 04 07 02 12 01 10 14 09 |
*             | 10 06 09 00 12 11 07 13 15 01 03 14 05 02 08 04 |
*             | 03 15 00 06 10 01 13 08 09 04 05 11 12 07 02 14 |
*             ------------------------------------------------
```

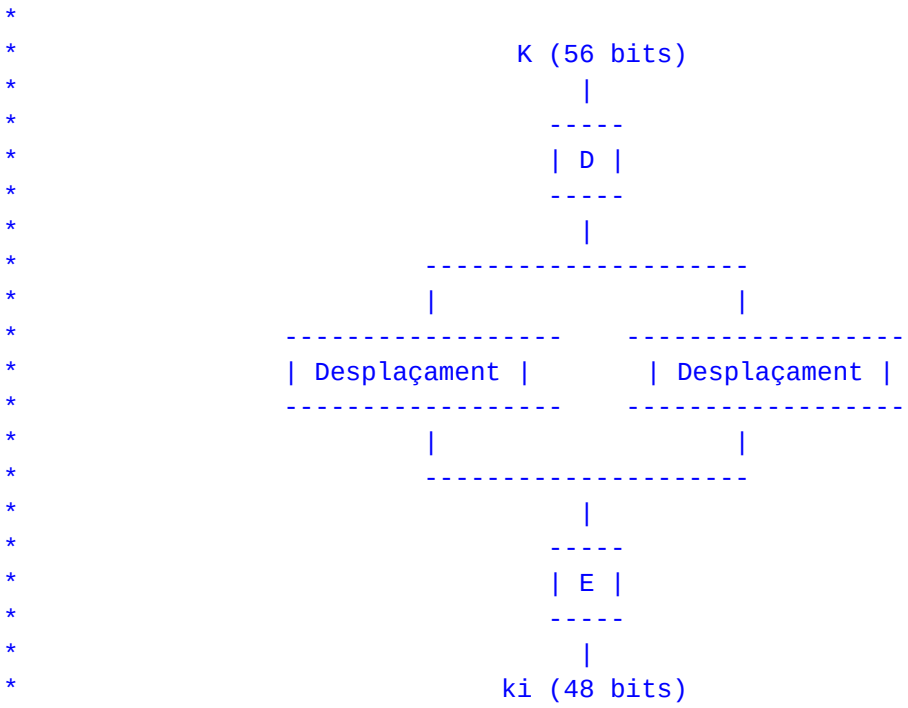
```
*
*               S-5
*            ------------------------------------------------
*            | 02 12 04 01 07 10 11 06 08 05 03 15 13 00 14 09 |
*            | 14 11 02 12 04 07 13 01 05 00 15 10 03 09 08 06 |
*            | 04 02 01 11 10 13 07 08 15 09 12 05 06 03 00 14 |
*            | 11 08 12 07 01 14 02 13 06 15 00 09 10 04 05 03 |
*            ------------------------------------------------
*

*               S-6
*            ------------------------------------------------
*            | 12 01 10 15 09 02 06 08 00 13 03 04 14 07 05 11 |
*            | 10 15 04 02 07 12 09 05 06 01 13 14 00 11 03 08 |
*            | 09 14 15 05 02 08 12 03 07 00 04 10 01 13 11 06 |
*            | 04 03 02 12 09 05 15 10 11 14 01 07 06 00 08 13 |
*            ------------------------------------------------
*

*               S-7
*            ------------------------------------------------
*            | 04 11 02 14 15 00 08 13 03 12 09 07 05 10 06 01 |
*            | 13 00 11 07 04 09 01 10 14 03 05 12 02 15 08 06 |
*            | 01 04 11 13 12 03 07 14 10 15 06 08 00 05 09 02 |
*            | 06 11 13 08 01 04 10 07 09 05 00 15 14 02 03 12 |
*            ------------------------------------------------
*
*               S-8
*            ------------------------------------------------
*            | 13 02 08 04 06 15 11 01 10 09 03 14 05 00 12 07 |
*            | 01 15 13 08 10 03 07 04 12 05 06 11 00 14 09 02 |
*            | 07 11 04 01 09 12 14 02 00 06 10 13 15 03 05 08 |
*            | 02 01 14 07 04 10 08 13 15 12 09 00 03 05 06 11 |
*            ------------------------------------------------
*
```

Generació de subclaus K:

Aquest esquema es realitza per a cada una de les iteracions:



El primer pas consisteix en una permutació mitjançant la taula D;

Taula permutació D						
57	49	41	33	25	17	09
01	58	50	42	34	26	18
10	02	59	51	43	35	27
19	11	03	60	52	44	36
63	55	47	39	31	23	15
07	62	54	46	38	30	22
14	06	61	53	45	37	29
21	13	05	28	20	12	04

Els 56 bits es divideixen en dos parts de 28 bits. Cada una d'aquestes parts rota cap a l'esquerra un número X de bits en cada iteració, segons la taula descrita a continuació:

Iteració	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
Desplaçament	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	1

Finalment, mitjançant la taula de compressió E, els 56 bits del resultat es comprimeixen formant una subclau de 48 bits que es utilitzada per la funció F.

Taula de compressió E					
14	17	11	24	01	05
03	28	15	06	21	10
23	19	12	04	26	08
16	07	27	20	13	02
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Introducció als certificats:

Com ja he esmentat, una vegada desenvolupat el programari amb intercanvi de claus i analitzat els possibles atacs que pot rebre, he iniciat el que podem dir com a següent pas per tal de millorar la seguretat del nostre Xat. Per tal de millor la seguretat i evitar per exemple la amenaça d'home enmig podem realitzar un xat on la seguretat es bases en Certificats.

Autentificació:

El primer repte d'una aplicació segura és l'autentificació. La criptografia de clau pública ens ofereix eines molt potents per tal de verificar l'autor d'una comunicació.
Els certificats són continents de les claus de xifratge. Un certificat associa una identitat amb una clau pública. Aquest certificat conté les dades relatives al signant de les dades a enviar.
Per tal de millorar la seguretat del nostre xat podem implementar (ja iniciat) la solució següent:

Obtindrem la signatura per al ChatServer.

La autoritat certificador CA té un parell de claus (privada/pública) que les generarem utilitzant la eina keytool. La CA generaria una clau pública auto-signada.

Cada entitat que integri el nostre Xat (el servidor i tots els usuaris del xat es generaran les parelles de claus.

Exportarem el certificat de la CA a un fitxer i importarem la resta de claus.

Tindrem, una vegada realitzat els passos anteriors, el ChatServer amb una clau pública signada per la CA.

Obtenir certificació clients.

Al iniciar el client, es connecta a la autoritat certificadora i aquesta verifica la seva identitat. Una vegada verificat la identitat la CA crea i signa el certificat per al client i el retorna per un canal segur (SSL) al client.

Una vegada realitzat aquest passos tenim els certificats per als clients i per al servidor i podem realitzar un control més segur per al nostre xat.

Aquesta proposta intenta donar solució a les deficiències en seguretat existents en la implementació realitzada. Cal recordar que el nostre programari és vulnerable al atac anomenat home enmig descrit anteriorment.

5. Conclusions

Una vegada presentat el projecte cal remarcar la feblesa trobada en l'intercanvi de claus i la següent aplicació de les claus obtingudes. Una de les solucions possibles és la descrita anteriorment, la utilització de certificats.
El programari esdevé inestable en alguns casos. Considero que per donar per acabat el programa, si més no per considerar la implementació com un producte comercial, cal una depuració del codi. Un dels punts febles que cal repassar és els Streams de comunicació.

No obstant la realització és molt satisfactòria a nivell personal, sobretot pel costat educatiu. He vist la resolució de problemes de comunicació entre processos, intercanvi de claus d'una forma segura, encriptació de dades, controlar diferents fils d'execució....

Espero tenir la possibilitat d'ampliar aquest projecte continuant els meus estudis a la UOC realitzant els estudis superiors, ja que les possibilitats, alguna de les quals ja he esmentat, de millorar l'aplicatiu són molt interessants.

També m'agradaria comentar positivament la recerca necessària per tal de realitzar el projecte. Actualment, i en particular en el món informàtic, hi ha molta feina feta. Les fonts de documentació són molt bones i gràcies a Internet ràpidament accessibles. Jo diria, que sense aquesta ajuda el projecte hagués sigut molt més costós d'implementar.

6 Bibliografia

- El lenguaje de Programación JAVA (Ed Addison Wesley).

Comunicaciones y Redes de Computadores (Stallings Ed Prentice Hall).

OReilly Java Cryptography (molt recomanat)

-

Adreces d'Internet consultades:(entre d'altres)

Programming Networked Applications
http://math.hws.edu/eck/cs124/javanotes3/c10/s5.html
SOCKETS EN JAVA & JAVA
http://www.geocities.com/chuidiang/java/sockets/socket.html.
The Java Developers Almanac (molt interessant)
http://javaalmanac.com

7. Annexos

Codi font classes.