Memòria TFC – Xat Segur

Estudiant : **Jordi Coll Vera, ETIS**

Consultor : **Antoni Martínez Ballesté**

***Data Lliurament : 18* de Juny de 2004**

Taula de continguts :

Taula d'il·lustracions :

Definició del treball : XatSegur

Es tracta de realitzar un sistema de xat o de missatgeria instantània (tipus Messenger o ICQ, però punt-a-punt en comptes de centralitzat en un servidor) on la informació vagi xifrada. Es demanen les següents funcionalitats:

- Cadascun dels usuaris té un programa client/servidor que permet efectuar "trucades" a una determinada IP. El programa ha de contestar i ha d'intercanviar una clau de sessió.
- Els missatges han d'anar xifrats usant aquesta clau de sessió, i aquesta ha d'ésser diferent per a cadascuna de les comunicacions.
- Es podran mantenir vàries converses simultàniament.
- L'aplicació es realitzarà en llenguatge Java, sobre Windows XP.
- L'aplicació disposarà d'una interfície gràfica.

Planificació i fases del desenvolupament

El projecte complert serà dividit en quatre fases: cerca d'informació (F1), fase de disseny (F2), fase d'implementació (F3) fase de documentació i redacció de la memòria (F4). Aquestes fases es subdividiran en les següents tasques :

- **F1.1** Cerca d'informació sobre l'algorisme Diffie-Hellman. Saber per a què serveix i una mica com funciona.
- **F1.2** Cerca d'informació sobre l'algorisme DES (*Data Encryption Standard*). Saber per a què serveix i una mica com funciona.
- **F1.3** Cerca d'informació sobre comunicació de processos a través d'Internet a nivell de transport OSI (sockets).
- **F1.4** Cerca d'informació sobre llibreries de programació que permetin treballar amb Diffie-Hellman i/o amb DES.
- **F2.1** Dissenyar el programa, a nivell de processos, comunicació i interacció client-servidor.
- **F2.2** Dissenyar la interfície amb l'usuari (definir opcions de programa, botons, missatges, etc.)
- **F2.3** Desenvolupar el programari, a alt nivell, amb comentaris i de forma descriptiva. Realitzant aquesta tasca ens podem adonar d'errors de disseny o millores, havent doncs de modificar els resultats de F2.1 i F2.2.
- **F3.1** Implementació de la interfície gràfica.
- **F3.2** Afegiment del cos del programa, amb rutines de comunicació.
- **F3.3** Afegiment de l'intercanvi de claus i del xifratge.
- **F4.1** Elaboració de la memòria: objectiu del projecte, estat de l'art (explicar F1.1, F1.2 i F1.3) i eines de programació usades.
- **F4.2** Elaboració de la memòria: disseny del programa (amb diagrames i descripció a alt nivell).
- **F4.3** Elaboració de la memòria: proves, incidències i conclusions.
- **F4.4** Elaboració de la memòria: apèndix amb codi font, degudament comentat.
- **F4.5** Fer presentació del projecte, amb PowerPoint o altres aplicacions.

Cadascuna de les fases anteriors té una data fixada a la planificació inicial i uns terminis de presentació dels resultats de les diferents fases establerts pel consultor:

Informació prèvia, *Estat de l'art*

Els criptosistemes de clau pública

Els criptosistemes de clau secreta compartida presenten tres inconvenients :

1 - **Distribució de claus** : cada parella d'usuaris ha d'acordar una clau abans de començar a comunicar-se; personalment o a través d'un canal insegur.

2 - **Gestió de claus** : en una xarxa de n usuaris calen $\binom{n}{2} = \frac{n(n-1)}{2} = \mathrm{O}(n^2)$ claus, cada parella d'usuaris ha de tenir la seva clau compartida en particular.

3 - **Sense signatura digital** : no donen mecanismes d'autenticació (identificació, data/hora, etc.) La signatura digital és l'equivalent de les signatures manuals en el cas de la informació electrònica; amb un criptosistema de clau compartida, generalment no hi ha la possibilitat de signat els missatges, pel mateix fet que totes les claus són compartides almenys per dos usuaris.

En canvi presenten l'avantatge de que són sistemes molt més ràpids :

- En software, DES és com a mínim 100 vegades més ràpid que RSA
- En hardware, DES és entre 1000 i 10000 vegades més ràpid que RSA

El concepte de criptosistema de clau pública, que permet superar els inconvenients anteriors, va ser proposat per W. Diffie i M.E. Hellman en l'article "*New directions in cryptography*" aparegut l'any 1976. La seva idea permet l'intercanvi segur de missatges entre emissor i receptor sense ni tan sols haver-se de trobar prèviament per acordar una clau secreta comuna.

El concepte de *clau pública*

Com a filosofia, la regla d'encriptació es pot fer pública sense comprometre la seguretat del sistema. Els sistemes de clau pública permeten establir comunicació xifrada sense intercanvi previ de claus. A més, fan possible la signatura de documents i l'autenticació de signatures.

En un criptosistema de clau pública cada usuari, *u*, té associada una parella de claus < P_u, S_u > : la clau pública, P_u , que es publica amb el nom de l'usuari en un directori públic que tothom pot llegir, mentre que la clau privada, S_u , només la coneix *u*. Els parells de claus es generen mitjançant un algorisme de generació de claus.

Per enviar un missatge secret m a u, tothom de la xarxa fa servir el mateix mètode :

1 - Cercar P_u

2 - Calcular *c = E(Pu, m)*, en que *E* és un **algorisme públic de xifratge**

3 - Enviar *c* a l'usuari *u*

En rebre el text xifrat *c*, l'usuari *u* el pot desxifrar de la manera següent:

1 - Cercar la seva clau privada Su

2 - Calcular m = D(Su,c), en que D és un algorisme públic de desxifratge

De l'esquema anterior es dedueix que per tal de que el procediment funcioni cal que *D(Su, E(Pu, m))=m*. D'aquesta manera la gestió de claus queda simplificada, perquè ara només hi ha n parelles de claus per a n usuaris, en comptes de les n(n-1)/2 claus que calien amb la criptografia de clau compartida.

Funció unidireccional i funció unidireccional amb trapa (porta trampa)

Els criptosistemes de clau pública es basen en l'existència d'alguns tipus de funcions que són difícils (= computacionalment intractable) d'invertir.

La seguretat depèn del secret de la clau privada. Ha d'ésser impossible obtenir-la a partir de la pública. Tothom sap quina és la fórmula matemàtica per obtenir-la però no existeix (o no es coneix ?) un bon algorisme per calcular-la ràpidament. Per contra, els criptosistemes de clau secreta basen la seguretat en la confusió i difusió de la informació.

Una **funció unidireccional** f : M→ C és invertible tal que és *fàcil* (tractable) calcular f(m)=c, mentre que és *difícil* (intractable) calcular $f^{-1}(c)=m$.

> Entenent fàcil i difícil des del punt de vista de la complexitat algorísmica : fàcil = existeix un algorisme que el resol en temps polinòmic. Difícil vol dir que es resol en temps exponencial.

Una **funció unidireccional amb porta trampa** és una funció unidireccional que pot ser invertida *fàcilment* quan es coneix certa informació addicional. Aquesta informació addicional s'anomena *trapa*.

Intercanvi de claus de Diffie-Hellman

L'algorisme d'encriptació de Whitfield Diffie i Martin Hellman va suposar una vertadera revolució en el campo de la criptografia, ja que va ser el punt de partida pels sistemes asimètrics, basats en dues claus diferents, la clau pública i la clau privada. L'algorisme va ser descrit per primer cop l'any 1976, com il·lustració de l'article "*New directions in Cryptography*" citat anteriorment. En aquest article els autors varen descriure un protocol d'intercanvi de claus que evita l'inconvenient de la distribució de claus. El fet innovador és que dos usuaris poden pactar una clau secreta compartida sobre un canal insegur.

Representa l'inici dels sistemes asimètrics, ja que en la pràctica només és vàlid per l'intercanvi de claus simètriques, i amb aquesta funcionalitat és molt usat en els diferents sistemes segurs implementats en Internet, com SSL (Secure Socket Layer) i VPN (Virtual Private Network).

Matemàticament es basa en les potències dels nombres i en la funció *mod* (mòdul discret). Unint els dos conceptes es defineix la potència discreta d'un número com $\mathbf{Y = X^a \bmod q}$. Si bé el càlcul de potències discretes és fàcil, la obtenció de la seva funció inversa, el logaritme discret, no té una solució analítica per a nombres grans.

Per a dur a terme correctament el protocol d'intercanvi de Diffie-Hellman, s'han de seguir els passos següents :

1 - Es busca un nombre primer molt gran, **q**.

2 - S'obté el número **ß**, arrel primitiva de q, es a dir, que compleix que ß mod q, $ß^2$ mod q,...., $ß^{q-1}$ mod q son nombres diferents.

3 - **ß** i **q** son les claus públiques.

Figura 1 – Algorisme de Diffie-Hellman

Per generar una clau simètrica compartida entre dos usuaris, *A* i *B*, ambdós parteixen d'un generador de nombres pseudoaleatoris, que subministra un nombre d'aquest tipus diferent a cadascun, X_a i X_b. Aquestes són les claus privades de *A* i *B*. Amb aquests números i les claus públiques ß i q que ambdós coneixen, cadascun genera un número intermedi, Y_a i Y_b, segons la fórmula:

$$Y_a = ß^{Xa} \bmod q$$
$$Y_b = ß^{Xb} \bmod q$$

Aquests números són intercanviats entre ambdós, i després cadascun opera amb el que rep de l'altre, obtenint en el procés els dos el mateix número:

$$K = Y_b{}^{Xa} \bmod q$$
$$K = Y_a{}^{Xb} \bmod q$$

Aquest número K és la clau simètrica que a partir d'aquest moment ambdós comparteixen, i que poden usar per establir una comunicació xifrada utilitzant qualsevol dels sistemes simètrics.

És a dir :

1 - Els dos usuaris *A* i *B*, trien públicament un grup multiplicatiu finit *G* d'ordre *n* i un generador $\propto \in G$

2 - *A* genera un nombre aleatori *a*, calcula $\propto^a$dins *G* i transmet el resultat a *B*

3 - *B* genera un nombre aleatori *b*, calcula $\propto^b$dins *G* i transmet el resultat a *A*

4 - *A* rep $\propto^b$ i calcula $(\propto^b)^a$ dins *G*

5 - *B* rep $\propto^a$ i calcula $(\propto^a)^b$ dins *G*

Amb aquest esquema, si es desitja compartir una clau privada amb un usuari qualsevol, basta amb accedir a la seva clau pública Y_u i que ell accedeixi a la nostra. Per tal de facilitar aquest procés es solen publicar las Y_u de tots els usuaris interessats en un directori d'accés públic (o comú).

Diffie i Hellman van formular la idea d'un criptosistema de clau pública, però no en van donar cap exemple pràctic. En aquest sentit els dos criptosistemes més emprats són l'RSA[1] i l'ElGamal, que fonamenten la seguretat en els problemes de la factorització i del logaritme discret, respectivament.

Els criptosistemes de clau compartida (criptografia de clau secreta)

Anomenem **mètodes simètrics** als que utilitzen la mateixa clau per xifrar i desxifrar el missatge (*text en clar*).

Distingim dos mètodes bàsics de xifrat simètric :

1 - **Xifrat en flux** : La transformació de xifrat varia de símbol a símbol. Per exemple el mètode RC4.

2 - **Xifrat en bloc :** el missatge es divideix en blocs de la mateixa longitud als quals s'aplica la mateixa transformació de xifrat.. Per exemple els mètodes DES, IDEA, AES.

El xifrat en flux

[1] Rivest Shamir i Adleman, "*A method for obtaining digital signatures and public-key cryptosystems*", 1978

Es una aproximació al xifrat de Vernam: emissor i receptor acorden una clau (curta) i un algorisme (determinístic) generador de seqüència xifrant (cadena de bits pseudoaleatòria) :

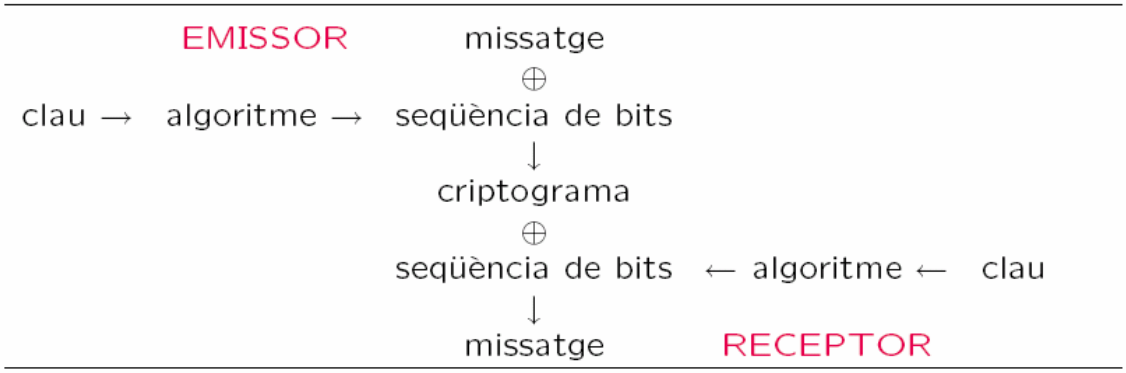


Figura 2 – El xifrat en flux

Els generadors binaris pseudoaleatoris han de satisfer :

- Període > 10^{38} (superior a la longitud de qualsevol text a xifrar)
- Distribució uniforme de zeros i uns. Segons els Postulats de Golomb:
  - Mateix nombre de zeros i uns
  - Digrames 00, 01, 10, 11 equirepartits, trigrames,. . . ,n−grames
  - Les coincidències entre la seqüència i la seva desplaçada no proporcionen informació sobre el període
- Imprevisibilitat: donada una porció, no es pot predir el següent dígit amb probabilitat d'encert superior a ½
- Facilitat d'implementació

El xifrat en bloc

Propietats :

- el missatge es xifra agrupant els símbols en blocs d'una longitud fixada.
- cada símbol es xifra de manera dependent dels adjacents
- cada bloc es xifra sempre d'igual manera, independentment de la seva posició en el missatge
- un missatge es pot desxifrar parcialment, a partir del bloc que interessi

Arquitectura :

- Transformació inicial. Introdueix aleatorietat
- Funció iterada *r* vegades ("tombs"). Funció no lineal (complicada) de les dades i la clau, que pot definir-se mitjançant diverses transformacions simples. Els tombs es connecten amb un XOR bit a bit.
- Transformació final. Simetritza xifrat i desxifrat.
- Algorisme d'expansió de clau. La clau de l'usuari (curta) es transforma en un conjunt de subclaus (molts bits)

Un algorisme d'encriptació simètrica àmpliament utilitzat en el passat (i encara avui en dia) és el DES (*Data Encryption Standard*), que xifra blocs de 64 bits amb una clau de 56 bits.

El criptosistema DES

Història

1973 El NBS (*National Bureau of Standards*) demana públicament propostes de sistemes criptogràfics, que:

- Proporcionin un alt nivell de seguretat i siguin eficients.
- La seguretat depengui només del secret de la clau (no del secret de l'algoritme).
- Puguin adaptar-se a diverses aplicacions.
- La implementació en dispositius electrònics sigui barata.

1974 Segona crida: IBM presenta LUCIFER. La NSA (*National Security Agency*) proposa canvis i són acceptats.

1975 El 17 de març el NBS publica els detalls del DES.

1976 El 23 de novembre és adoptat pel govern USA per a la transmissió i emmagatzematge d'informació no classificada. Es revisarà cada cinc anys.

1981 Diversos organismes privats l'adopten com a estàndard.

1983 Es ratifica com a estàndard sense problemes.

1987 La NSA s'oposa a una nova ratificació però, per motius econòmics, finalment es renova.

1992 Per manca d'alternatives es torna a renovar.

1997 El 17 de juny el DES es trenca. A començaments d'any RSA llença el repte i en quatre mesos es troba la solució, havent examinat aproximadament el 25% de les claus.

1998 El 26 de febrer el DES es torna a trencar. Només han calgut 39 dies, per examinar aproximadament el 85% de les claus. El 17 de juliol la *Electronic Frontier Foundation* (EFF) presenta el seu DES craker, que pot trencar el DES utilitzant la cerca exhaustiva en un temps mitjà de 4.5 dies.

1999 El 19 de gener la EFF trenca el DES en menys de 23 hores (amb un poder de còmput que efectuava aproximadament 250 mil milions de proves per segon).

2001 Substituït pel AES (Advanced Encryption Standard). Deixa de ser un estàndard.

La versió de DES implementada per hardware forma part dels estàndards de la ISO amb el nom de **DEA**

Descripció de l'algorisme

DES es basa en un sistema monoalfabètic, amb un algorisme de xifrat consistent en la aplicació successiva de varies permutacions i substitucions. Inicialment el text en clar a xifrar es sotmet a una permutació, amb bloc d'entrada de 64 bits (o múltiple de 64), per posteriorment ser sotmès a la acció de dos funcions principals, una funció de permutació amb entrada de 8 bits i una altra de substitució amb entrada de 5 bits, en un procés que consta de 16 etapes de xifrat.

En general, DES utilitza una clau simètrica de 64 bits, dels quals 56 són usats per a l' encriptació, mentre que els altres 8 són de paritat, i s'utilitzen per la detecció d'errors en el procés.

Com la clau efectiva és de 56 bits, són possibles un total de 2^{56} = 72.057.594.037.927.936 claus possibles, es a dir, uns 72.000 bilions de claus, això fa que la ruptura del sistema per força bruta o diccionari sigui improbable, però no impossible (com s'ha vist anteriorment) si es disposa de sort o d'una gran potència de càlcul.

Els principals inconvenients que presenta DES són:

- Es considera un secret nacional de EEUU, pel que està protegit per lleis específiques i no es pot comercialitzar ni en hardware ni en software fora d'aquest país sense permís específic del Departament d'Estat.
- La clau és curta, tant que no assegura una fortalesa adequada. Havia resultat suficient fins l'any 1997 (encara que en un termini de temps no viable per a la informació xifrada). Amb la potencia de càlcul actual i futura i amb el treball en equip per Internet es creu que es pot violar l'algorisme en un temps factible.
- No permet longitud de clau variable, amb la qual cosa les seves possibilitats de configuració són molt limitades, a més de permetre's amb això la creació de limitacions legals.
- La seguretat del sistema es veu reduïda considerablement si es coneix un nombre suficient de textos elegits, ja que existeix un sistema matemàtic (el Criptoanàlisi Diferencial) que pot en aquest cas trencar el sistema en 2^{47} iteracions.

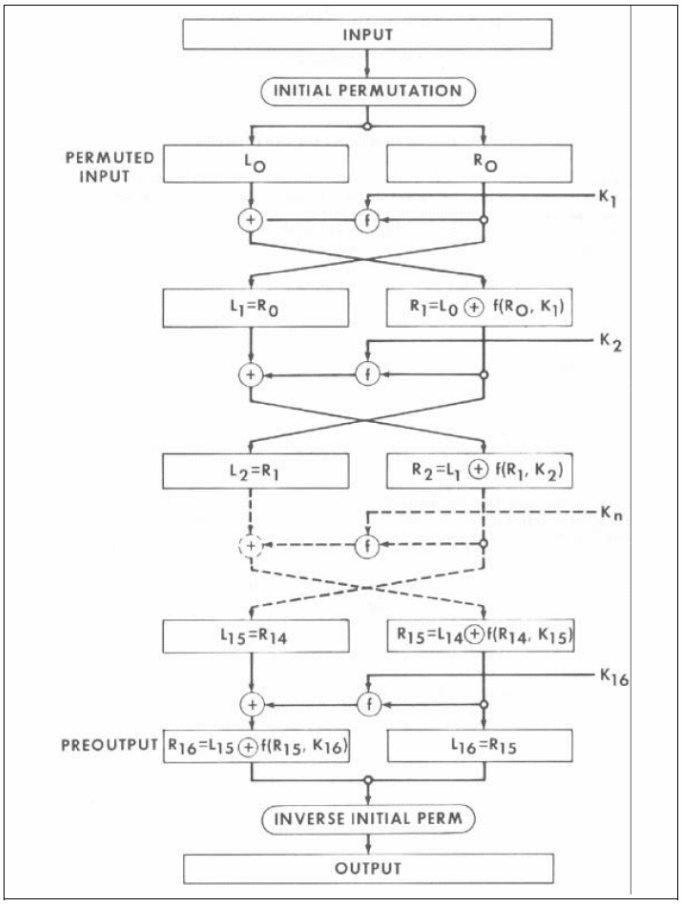
Entre els seus avantatges podem citar:

- És el sistema més estès del mon, el que més màquines usen, el més econòmic i el més provat.
- És molt ràpid i fàcil d'implementar.
- Des de la seva aparició mai ha estat trencat amb un sistema pràctic.

Implementació

Seguirem el següent conveni: Si agafem els dos blocs de bits ***L*** *i* ***R***, ***LR*** denota el bloc format pels bits de ***L*** *seguits* dels bits de ***R***. És a dir : la concatenació és associativa, ***B1B2...B8***, per exemple, denota el bloc format pels bits de ***B1 seguits pels bits*** de ***B2***...seguits pels bits de ***B8***.

Figura 3 – Esquema de xifrat



El xifrat :

Es correspon a l'esquema de la figura 3.

Els 64 bits del bloc d'entrada que ha de ser xifrat són sotmesos a una permutació inicial anomenada ***permutació IP*** :

IP

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

Després d'aquesta permutació el bit 58 del bloc d'entrada ocuparà la posició 1, el bit 50 la posició 2, i així successivament fins al bit 7 que ocuparà la última posició. El bloc permutat serà a continuació l'entrada d'un complex algorisme depenent de clau, que es descriurà a continuació. La sortida d'aquest procés, anomenada *preoutput*, és sotmesa a la següent permutació què és la inversa de la permutació inicial :

IP^{-1}

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

L'algorisme que ha generat el bloc *preoutput* consisteix en 16 iteracions de l'anomenada *funció f*, que es descriu a continuació. Aquest algorisme opera amb dos blocs, un de 32 bits i un de 48 bits i produeix un bloc de 32 bits.

Agafant els 64 bits del bloc d'entrada, una iteració consisteix en generar un bloc *L* de 32 bits seguit del bloc *R*, també de 32 bits. Aquest bloc, seguint el conveni descrit al inici, s'anomena *LR*.

Sigui *K* el bloc de 48 bits agafats dels 64-bits de la clau. Aleshores la sortida *L'R'* d'una iteració amb entrada *LR* es defineix com:

(1) $L' = R$

$R' = L \oplus f(R,K)$

On $\oplus$ indica la suma bit-a-bit mòdul 2.

Com s'ha remarcat abans, l'entrada de la primera iteració de l'algorisme és el bloc permutat inicial.

Si **L'R'** és la sortida de la 16 iteració, aleshores **R'L'** és el bloc *preoutput*. A cada iteració s'agafa un bloc diferent *K* de bits de la clau de 64 bits designada com a *KEY*.

Necessitem una mica més de notació per descriure les iteracions de l'algorisme amb més detall. Sigui *KS* la funció que pren un enter *n* del 1 al 16 i un bloc *KEY* de 64 bits com a entrada i produeix la sortida d'un bloc de 48 bits K_n, permutació de bits de *KEY*. És a dir :

(2) $\mathbf{K_n = KS(n,KEY)}$

Amb *Kn* determinat pels bits de les 48 posicions diferents de *KEY*. *KS* s'anomena *key schedule* perquè el bloc *K*, utilitzat en la iteració *n* de (1) és el bloc K_n determinat per (2).

Com abans, sigui *LR* el bloc permutat inicial i, finalment, L_0 i R_0 respectivament *L* i *R* i siguin L_n i R_n respectivament *L'* i *R*' de (1) quan *L* i *R* són respectivament L_{n-1} i R_{n-1} i *K* és K_n; és a dir, quan *n* és en el rang 1-16,

(3) $\mathbf{L_n = R_{n-1}}$

$\mathbf{R_n = L_{n-1} \oplus f(R_{n-1},K_n)}$

El *preoutput* bloc és aleshores $R_{16}L_{16}$.

KS es descriu més endavant. Produeix el 16 K_n requerit per l'algorisme.

El desxifrat :

La permutació IP^{-1} aplicada al bloc *preoutput* és la permutació inversa de la permutació inicial *IP* aplicada a la entrada. A més, de (1) segueix que:

(4) $\mathbf{R = L'}$

$\mathbf{L = R' \oplus f(L',K)}$

En conseqüència, per desxifrar és únicament necessari aplicar exactament el mateix algorisme al bloc xifrat del missatge, tenint compte de que a cada iteració s'utilitzen els mateixos *K* bits de clau usats durant el xifrat del bloc.

Amb la nostra notació, podem expressar-ho amb les equacions:

(5) $\mathbf{R_{n-1} = L_n}$

$\mathbf{L_{n-1} = R_n \oplus f(L_n,K_n)}$

On $R_{16}L_{16}$ is és el bloc permutat inicial del desxifratge i L_0R_0 és el bloc *preoutput*. És a dir, pel càlcul del desxiframent amb $R_{16}L_{16}$ com a bloc permutat d'entrada, utilitzem K_{16} en la primera iteració, K_{15} en la segona, i així succesivament, amb K_1 utilitzat a la 16 iteració.

La funció f de xifrat

La Figura 4 ens mostra un esquema del càlcul de *f(R,K)*.

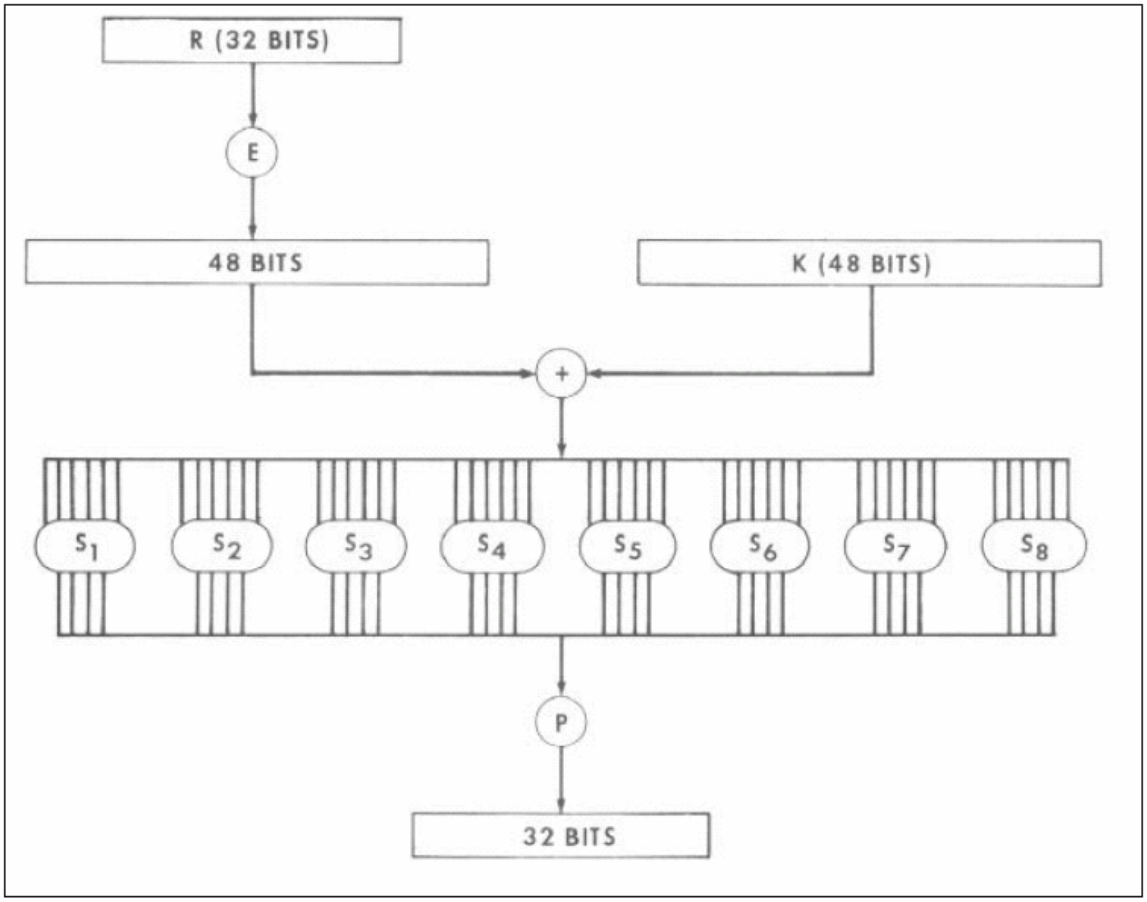


Figura 4 – Càlcul de *f(R,K)*.

Sigui *E* la funció que agafa un bloc de 32 bits com entrada i proporciona un bloc de 48 bits com a sortida. Sigui *E* tal que els 48 bits de sortida, escrits com a 8 blocs de 6 bits cadascun, s'han obtingut seleccionant els bits de l'entrada segons la taula següent :

Així els tres primers bits de *E(R)* són els bits en posicions 32, 1 i 2 de R mentres que els dos últims de *E(R)* són els bits en les posicions 32 i 1 (... el primer bit de sortida correspon al bit 32 d'entrada; el segon bit de sortida, a l'1 d'entrada, etc.)

E BIT-SELECTION					
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Cadascuna de les seleccions úniques de les funcions $S_1,S_2,...,S_8$, agafa un bloc de 6 bits com a *input* i proporciona un bloc de 4 bits com a *output*, segons la taula següent, S_1:

S_1

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Si S_1 és la funció definida a la taula anterior i *B* és el bloc de 6 bits, aleshores $S_1(B)$ queda determinat de la manera següent: El primer i l'últim bit de *B* representa en base 2 un número entre 0 i 3. Sigui *i* aquest número.. Els quatre bits del centre de *B* representen en base 2 un número entre 0 i 15. Sigui *j* aquest número.. Mirem a la taula en la fila i columna j i aquest és el número entre 0 i 15 que representa de manera única per un bloc de 4 bits. Aquest bloc és la sortida $S_1(B)$ de S_1 per l'entrada *B*. Per exemple, per l'entrada 011011 la fila és 01, la fila 1, i la columna és la 1101, la columna 13. A la fila 1 columna 13 hi tenim un 5, aleshores la sortida serà 0101.

Més endavant incloc les taules S_1, ..., S_8.

La permutació *P* genera una sortida de 32 bits a partir d'una entrada de 32 bits, segons una permutació definida a la taula següent :

La sortida *P(L)* de la funció *P* definida per aquesta taula s'obté de l'entrada *L* prenen el bit 16 de *L* com a primer bit de *P(L)*, el bit 7 com a segon bit de *P(L)*, i així succesivament fins al bit 25 de *L* que s'agafa com el bit 32 de *P(L)*.

La funció de permutació *P* de l'algorisme es detalla més endavant.

Ara siguin $S_1,...,S_8$ les vuit funcions de selecció diferents, *P* la funció permutació i *E*

La funció definida prèviament.

Per definir *f(R,K)* definirem primer $B_1,...,B_8$ els blocs de 6 bits cadascún, on :

P

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

(6) $\mathbf{B_1B_2...B_8 = K \oplus E(R)}$

El bloc *f(R,K)* és aleshores definit com

(7) $\mathbf{P(S_1(B_1)S_2(B_2)...S_8(B_8))}$

Així $K \oplus E(R)$ és dividit primer en els 8 blocs indicats a (6). Després cada B_i es pren com a entrada de S_i i 8 blocs $S_1(B_1),S_2(B_2),...,S_8(B_8)$ de 4 bits cadascún són consolidats en un únic bloc de 32 bits que constitueix l'entrada de *P*. La sortida (7) és aleshores la sortida de la funció *f* pels inputs *R* i *K*.

Funcions primitives de l'algorisme de xifrat :

1 - Les funcions (*caixes*) $S_1,S_2,...,S_8$, :

Tal i com s'ha explicat abans, cada bloc B_i s'usa com a entrada del que s'anomena una caixa S. Cada caixa, S_i rep el seu bloc corresponent de 6 bits i en retorna un de 4 bits, d'acord amb les taules següents.

S_1

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

	Columna															
Fila.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Figura 5 – Generació de subclaus DES

2 - Generació de subclaus :

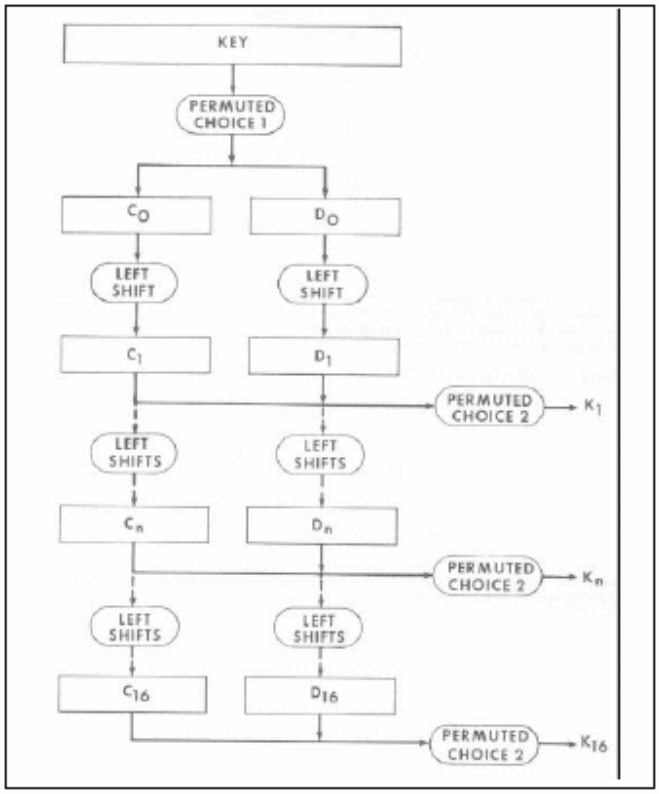
Recordem que el criptosistema DES opera inicialment amb una clau *K* de 56 bits de llargada. En canvi l'algorisme descrit utilitza 16 subclaus K_i de 48 bits. Vegem doncs com es poden obtenir aquestes subclaus a partir de la clau principal. L'algorisme de generació de subclaus s'il·lustra a la figura 5.

A la clau inicial *K* de 56 bits s'hi ha afegit 8 bits més en les posicions 8, 16, ..., 64 que corresponen a les paritats. La funció de permutació *PC-1* descarta els bits de paritat par i trasposa els 56 bits restants.

El resultat de la funció de permutació PC-1(K) és dividir la clau KEY en dues meitats, C0 i D0, de 28 bits cadascuna. Els blocs D0 i C0 es desplacen cap a l'esquerra per a obtenircada subclau Ki. Si representem per Ci i Di, els valors emprats per a obtenir Ki, tenim :

$\mathbf{C_i = DC_i(C_{i-1})}$

$\mathbf{D_i = DC_i(D_{i-1})}$



On DC_i representa el desplaçament circular cap a l'esquerra de tantes posicions com determina la taula de desplaçament *DC* següent :

Taula de desplaçament *DC*	
Iteració número	Desplaçament A l'esquerra
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Finalment obtenim la subclau K_i per mitjà de l'expressió :

$$\mathbf{K_i = PC\text{-}2(C_i,D_i)}$$

On *PC-2* representa una funció de permutació més.

***PC*-1**

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

***PC*-2**

14	17	11	24	1	5	14
3	28	15	6	21	10	3
23	19	12	4	26	8	23
16	7	27	20	13	2	16
41	52	31	37	47	55	41
30	40	51	45	33	48	30
44	49	39	56	34	53	44
46	42	50	36	29	32	46

El criptosistema TDEA (Triple DES)

Descripció i implementació

Tal com s'ha dit abans, el sistema DES es considera avui en dia poc pràctic com a conseqüència de la seva curta clau. Per millorar-lo i continuar utilitzant DES es va crear el sistema *Triple-DES* (ANSI X9.52) basat en tres iteracions successives de l'algorisme DES, aconseguint-se d'aquesta manera una longitud de clau de 128 bits, compatible amb el DES simple.

Això és possible donat que DES té la característica matemàtica de no ser un grup, la qual cosa implica que es xifra el mateix bloc dos cops amb claus diferents s'augmenta efectivament el tamany de la clau.

Per implementar-lo, es pren una clau de 128 bits i es divideix en 2 diferents de 64 bits, aplicant-se el procés representat a la figura 6 al document del text en clar.

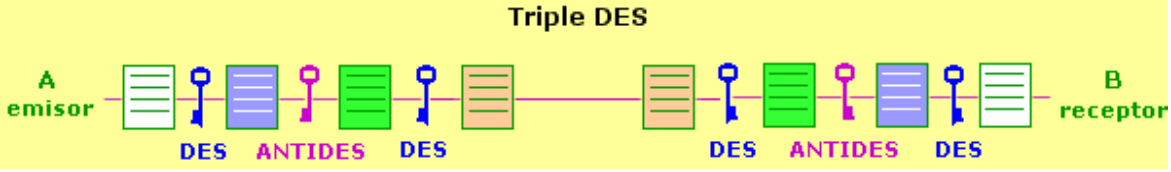


Figura 6 – L'algorisme Triple DES

1 - S'aplica al document a xifrar un primer xifrat amb la primera clau, K_1.

2 - Al resultat (denominat ANTIDES) se li aplica un segon xifrat amb la segona clau, K_2.

3 - Finalment, al resultat se li torna a aplicar un tercer xifrat amb la primera clau, K_1.

Si la clau de 128 bits està formada per dues claus iguals de 64 bits (K_1= K_2), aleshores el sistema es comporta com un DES simple.

Actualment TDES usa 3 claus diferents, el que el fa un sistema molt més robust, al aconseguir-se longituds de clau de 192 bits (dels quals son efectius 168), mentre que l'ús de DES simple no és aconsellat.

Comunicació de processos a nivell de transport OSI : els sockets

Introducció. Els sockets de Java.

URLs i URLConnections proveeixen en Java un mecanisme de relatiu alt nivell per accedir als recursos en Internet. Hi ha ocasions en que es necessita una comunicació de més baix nivell, per exemple quan volem disenyar una aplicació *client/servidor*.

A una aplicació *client/servidor*, la part *servidor* proporciona un servei concret, com el procés d'una base de dades, o l'enviament de determinada informació sol·licitada. La part *client* utilitza el servei proporcionat per la part *servidor*. La comunicació ha de ser fiable , és a dir que cap dada es pot perdre i que la informació ha d'arrivar a la part *client* en el mateix ordre en que la part *servidor* la va enviar.

El protocol TCP proporciona un canal de comunicació punt-a-punt (P2P) fiable que les aplicacions client/servidor utilitzen per a comunicar-se a través d'Internet. Per fer-ho, el programa client i el programa servidor estableixen una comunicació mútua. Cada programa *lliga* (*bind*) un *socket* a la seva part final de la connexió. Cadascún, client i servidor, llegeix i escriu el socket lligat a la connexió.

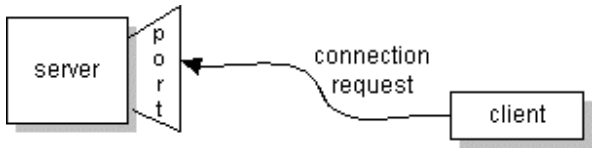
Simplificant podriem dir que un socket és una interfície d'E/S de dades que permet la intercomunicació entre processos. Aquests processos poden estar executant-se en el mateix o diferent sistemes, comunicats a través d'una xarxa. Fent una analogia amb el telèfon, els sockets permeten la comunicació entre processos de la mateixa manera en que els telèfons permeten la comunicació entre persones.

El socket és el destí final de la comunicació bidireccional que lliga dos programes corrent per la xarxa. En Java les classes *Socket* s'usen per representar la connexió entre un programa client i un programa servidor. El paquet java.net proporciona dues classes diferents : *Socket* i *ServerSocket* que implementen la connexió de la part del client i de la part del servidor respectivament.

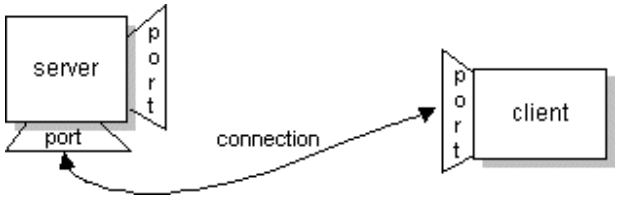
Funcionament bàsic

Normalment, el servidor corre en un ordinador en concret i té un socket lligat a un determinat número de port. El servidor està únicament a l'espera, *escoltant* el port, de que un client faci una sol·licitud de connexió.

A la part del client: el client coneix el *hostname* de la màquina on és la part servidor en marxa i coneix també el número de port al qual el servidor és connectat. Per tal de sol·licitar una connexió (*connection request*), el client intenta localitzar al servidor al port en concret de la màquina on és funcionant.



Si tot va bé, el servidor accepta la connexió i ofereix un nou socket lligat a un número de port diferent. Això és necessari per tal de que pugui continuar escoltant del port original per tal d'atendre noves sol·licituds mentre atén la connexió amb el client.



A la part del servidor, si la connexió ha estat acceptada, es crea un socket per comunicar-se amb el servidor. S'ha d'observar que aquest socket, de la part del client, no està lligat al port inicial (*connection request*) del servidor sinó que el client ha estat assignat a un número de port local a la màquina on el client és en funcionament.

Client i servidor poden ara comunicar-se llegint/escrivint dels/als seus respectius sockets.

Dominis de comunicació.

Els sockets es creen dins un espai de noms o domini de comunicació, igual que un arxiu es crea dins d'un *filesystem*.

El domini de comunicació ens diu on es troben els processos que es van a intercomunicar. Si els processos són units per una xarxa TCP/IP (Internet), el domini de comunicació serà AF_INET. Hi ha altres dominis de comunicació (AF_UNIX, etc.)

Els sockets han estat dissenyats per poder utilitzar la mateixa interfície amb diferents famílies de protocols (= dominis de comunicació)

Tipus de sockets al domini AF_INET.

- Sockets Stream.
- Sockets Datagram.
- Sockets Raw.

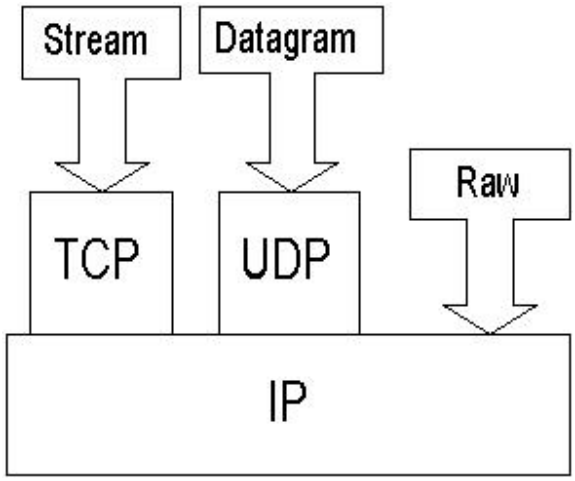


Figura 7 – Sockets del dominiAF_INET

Sockets Stream (estil seqüència de bytes)

Són els més utilitzats, fan us del protocol TCP (figura 7), el qual ens proporciona un flux de dades bidireccional, seqüenciat, sense duplicació de paquets i lliure d'errors. La especificació del protocol TCP es pot llegir a la RFC-793 .

En els sockets que utilitzen l'estil seqüència de bytes, les dades es transmeten d'extrem a extrem com un corrent o flux ordenat de bytes.

Aquest estil també es pot anomenar orientat a connexió, perquè la comunicació consisteix a establir prèviament una connexió amb el socket remot i després utilitzar-la per a intercanviar les dades. Quan s'utilitza l'espai de noms Internet, aquest estil es correspon amb el protocol TCP.

És possible que el protocol permeti la transmissió de dades fora de banda. Aquestes dades s'usen normalment per a enviar informació urgent, ja que se suposa que el destinatari, quan les rebi, les processarà abans que les dades normals que tingui pendents de llegir.

Sockets Datagram (estil datagrama)

Fan us del protocol UDP, el qual ens proporciona un flux de dades bidireccional, però els paquets poden arribar fora de seqüència (tots els datagrames contenen l'adreça del socket al qual van dirigits, de manera
que es poden encaminar independentment). poden no arribar o contenir errors. Per tant el procés que rep les dades ha de realitzar-ne el reseqüenciament, eliminar duplicats i assegurar la confiabilitat.
S'anomenen també sockets sense connexió, perquè no s'ha de mantenir una connexió activa, com en el cas dels sockets stream.

Son utilitzats per a la transferència d'informació paquet per paquet. Exemple: dns, tftp, bootp, etc. Ens podríem preguntar com fan aquests programes per no perdre dades. La resposta és que implementen un protocol per sobre de UDP que realitza el control d'errors.

L'avantatge d'aquest estil de comunicació rau en que requereix molts menys recursos que l'estil seqüència de bytes (no es necessiten números de seqüència en les capçaleres, buffers i finestres de transmissió i recepció, confirmacions –acknowledgements–, retransmissions, etc.).

Els datagrames són apropiats quan no és indispensable la fiabilitat total en la recepció de les dades, per exemple quan n'hi ha prou amb un mecanisme senzill de recuperació dels errors (com pot ser reenviar un paquet si no s'ha rebut cap resposta després d'un cert temps). En aquest cas, els mecanismes de correcció, si n'hi ha, s'implementen en el nivell d'aplicació.

En canvi, quan es necessita assegurar la recepció ordenada de les dades sense pèrdues ni repeticions, no és convenient que l'aplicació utilitzi datagrames i s'intenti recuperar dels errors per si mateixa, sinó que és més eficient encarregar aquesta tasca als nivells inferiors (és a dir, utilitzar un protocol de transport que garanteixi la fiabilitat).

L'especificació del protocol UDP es pot llegir a la RFC-768.

Sockets raw

No són per l'usuari més comú, son destinats principalment per aquells interessats en desenvolupar nous protocols de comunicació o per a fer us de facilitats ocultes d'un protocol existent.

La comunicació entre sockets

Per a poder establir una comunicació entre dos *socket*s, les dues parts involucrades, client i servidor, han d'efectuar una sèrie d'operacions, abans i després de fer l'intercanvi d'informació.

Les operacions que ha d'efectuar el servidor són les següents:

- Crear un *socket* en l'espai de noms i amb el protocol volguts.
- Assignar una adreça al *socke*t.
- Deixar el *socket* preparat per a rebre connexions i crear una cua en què s'aniran guardant les peticions que li arribin (només en els protocols orientats a connexió). Això s'anomena *establir una connexió passiv*a.
- Acceptar una petició de connexió de la cua o esperar fins que n'arribi una si no n'hi ha cap (només en els protocols orientats a connexió).
- Intercanviar dades amb el client que ha sol·licitat la comunicació llegint les dades i escrivint-les en el *socke*t.
- Tancar el *socket* quan hagi acabat l'intercanvi.

D'altra banda, les operacions que ha d'efectuar el client són les següents:

- Crear un socket en l'espai de noms i amb el protocol volguts.
- Assignar una adreça al socket (si la família de protocols no ho requereix, com en el cas dels protocols Internet, aquest pas és opcional).

- Enviar una petició de connexió al servidor (només en els protocols orientats a connexió). Això s'anomena establir una connexió activa.
- Intercanviar dades amb el servidor llegint-les i escrivint-les en el socket.
- Tancar el socket quan hagi acabat l'intercanvi

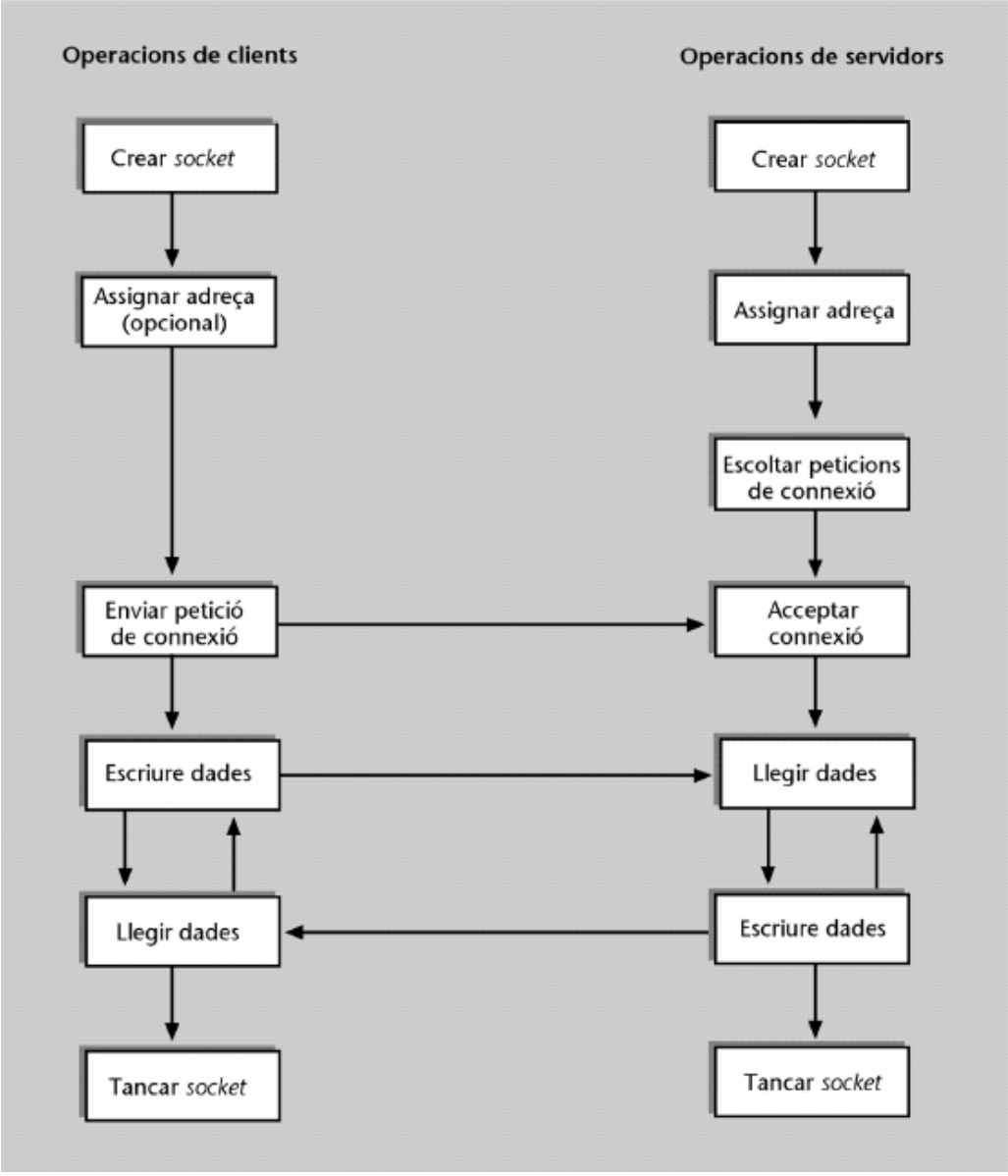


Figura 8 – Connexió entre sockets

Estat de l'art

Els criptosistemes *Simètrics* (clau privada) i *asimètrics* (clau pública) són els dos grans grups d'algorismes de xifrat utilitzats en les comunicacions P2P.

Els algorismes simètrics utilitzen la mateixa clau secreta per xifrar i desxifrar els missatges i la seva característica més destriable és potser la seva velocitat. Algorismes d'aquest tipus, molt coneguts, són : DES, triple-DES (algorisme utilitzat a XatSegur), IDEA, RC5, Blowfish, i Rijndael AES (192- i 256-bits).

Els algorismes asimètrics per altra banda, utilitzen una clau pública coneguda pel xifrat i una clau privada secreta pel desxifrat, amb les dues claus matemàticament relacionades. Donat que les claus han de tenir un nombre elevat de bits (1024- i 2048-bits) per tal d'assegurar un nivell similar de seguretat que els algorismes simètrics, es requereix un temps de processament. L'estàndard més àmpliament utilitzat en aquest sentit és el protocol RSA, aplicat a protocols d'Internet com SSL, TLS, 802.11, S/MIME, PEM, S/WAN, S-HTTP.

Altres sistemes de clau pública àmpliament coneguts són DSA, ElGamal, i Diffie-Hellman (DH). Aquest últim ha estat l'utilitzat a XatSegur per tal d'intercanviar de manera segura les claus privades a partir de les claus públiques.

Des de la introducció de la criptografia de les corbes el·líptiques a mitjans de la dècada dels 80, la recerca en ECC (*elliptic curve cryptography)* ha vist un gran increment respecte al seu rival més immediat (els sistemes RSA). Per un nivell equivalent de seguretat, ECC ofereix més velocitat, menys consum i més estalvi de recursos (memòria i ample de banda) amb parelles de claus més curtes. Per exemple, un sistema triple-DES, amb una clau simètrica de 112 bits té un nivell de seguretat comparable (cadascun requereix uns 10^{24} MIPS anys per trencar el codi – uns 30 anys de protecció amb la tecnologia actual -) a un sistema ECC de 224 bits i a un sistema asimètric RSA/DSA/DH de 2048 bits.

Els algorismes ECC resolen típicament corbes de tipus $y^2 = x^3 + ax + b$ sobre un conjunt finit de valors primers (mòdul p, amb p un nombre primer). L'ECDH (Elliptic Curve Diffie Hellman) seria l'algorisme amb corbes el·líptiques comparable a l'algorisme de Diffie-Hellmann. Així com l'ECDSA (Elliptic Curve Digital Signature Algorithm) és comparable a l'algorisme DSA.

Podem resumir-ho en la següent taula :

Sistema de clau pública	Exemples	Problema matemàtic plantejat	Millor mètode conegut (en velocitat de càlcul)
Factorització entera	RSA, Rabin-Williams	Donat un nombre n, troba els seus factors primers	exp[1.923(log n)1/3(log log n)2/3] (Sub-exponencial)
Logaritme discret	DH, DSA, ElGamal	Donat un nombre primer n, i dos nombres g i h, troben x de manera que $h = g^x \bmod n$	exp[1.923(log n)1/3(log log n)2/3] (Sub-exponencial)
Corba el·líptica del logaritme discret	DCDH, ECDSA	Donada un corba el·líptica E i dos punts P i Q sobre E, calculen x tal que Q = xP	Polar-rho algorisme: $\sqrt{n}$ (totalment exponencial)

Disseny i Implementació

Segons recull el pla de treball inicial, aquesta fase contempla :

- El disseny del programa, a nivell de processos, comunicació i interacció client-servidor.
- El disseny de la interfície amb l'usuari (definir opcions de programa, botons, missatges, etc.)
- El desenvolupament del programari, a alt nivell, amb comentaris i de forma descriptiva. Realitzant aquesta tasca ens podem adonar d'errors de disseny o millores, havent doncs de modificar els resultats de les fases anteriors.

La programació s'efectuarà en llenguatge Java (versió 1.4.1_01).

Segons s'especifica al mateix pla de treball , l'aplicació consta, a molt alt nivell, dels següents components :

Component Client / Servidor. La part Client es basa en sockets asíncrons (controlats per events) i la part Servidor serà Multi-Fil. Cada usuari disposarà d'una part Servidor a la que accedeixen un o més usuaris clients. El nombre de clients dependrà del nombre de connexions simultànies que s'originin a la màquina del usuari Servidor.

Un model de sockets simple podria ser:

El servidor estableix un port i espera durant un cert temps (timeout segons), a que el client estableixi la connexió. Quan el client sol·liciti una connexió, el servidor obrirà la connexió socket amb el mètode accept(). El client estableix una connexió amb la màquina host a través del port que es designa en port# . El client i el servidor es comuniquen amb manegadors InputStream i OutputStream.

L'encriptació es basa en un sistema de Clau Pública per les comunicacions amb el servidor de Xat. Es genera un parell de claus pública/privada a cada costat de la connexió i s'intercanvien les claus públiques. S'utilitza la llibreria criptogràfica IAIK-JCE3.02eval (*http://jce.iaik.tugraz.at/*), concretament el JCE Toolkit (http://jce.iaik.tugraz.at/products/01_jce/index.php) - la versió actual és la 3.03 -.

A cada connexió intervenen els següents components d'encriptació: l'encarregat del mecanisme de Clau Pública/Clau Privada pel que s' utilitza l'algorisme de Diffie-Hellman, la part del xifrat/dexifrat basada en l'algorisme Triple DES de 128 bits.

La comunicació. El xat entre usuaris es realitza utilitzant un port TCP. La part servidora instància un objecte de la classe ServerSocket amb un número de port rebut com a paràmetre : *server = new ServerSocket (port)*, així com un objecte de la classe Socket que queda *escoltant* a l'espera de la connexió d'un client : *client = server.accept ()*. Un cop s'ha establert la connexió, el servidor queda a la espera de nous clients al port inicial, els quals es podran enviar missatges mitjançant *broadcasting* amb un sistema *MultiThread*.

En el moment d'establir la connexió, la part servidora i la part client s'intercanvien les claus públiques i es genera a cada costat de la connexió la clau privada.

Tant la part Client com la part Servidor disposen d'una **interfície gràfica** per la interacció amb l'usuari. També es disposa de la part client en forma d'**Applet**.

Fases del disseny i implementació

Intercanvi de claus

En una primera fase vaig dissenyar els processos de comunicació, intercanvi de claus i intercanvi dels missatges xifrats, sense interfície gràfica.

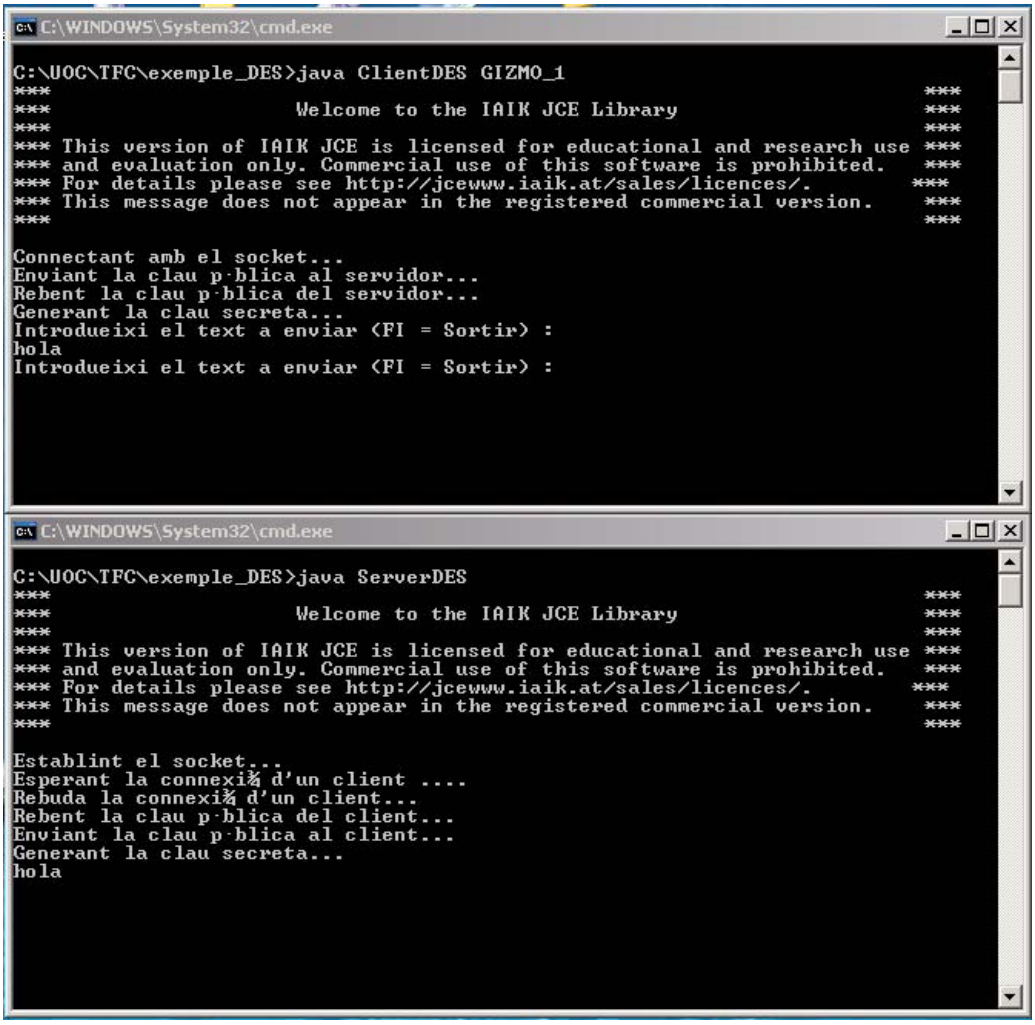


Figura 9 – Intercanvi de claus

En aquesta fase vaig verificar que l'intercanvi de claus i la generació de la clau privada fos correcte. També vaig verificar que l'intercanvi de missatges xifrats entre client i servidor (el servidor retornava el missatge al client) fos correcte i intel·ligible.

En aquesta fase em vaig trobar que la classe CipherOutputStream tallava els missatges en trossos múltiple de vuit. La solució adoptada va ser concatenar espais en blanc al final del missatge fins que la longitud fos múltiple de vuit.

L'intercanvi de claus es basa en una de les classes d'exemple de la llibreria IAIK : la classe DHKeyAgreement. La implementació de la part client en aquesta fase va ser :

```
import java.net.*;
import java.io.*;
import java.security.*;
import javax.crypto.*;
import iaik.security.provider.IAIK;

class ClientDES {

  protected static DHKeyAgreement DHKey = new DHKeyAgreement();

  public static void main(String argv[]) {

    try {
```

```
      System.out.println("Connectant amb el socket...");
      Socket s=new Socket(argv[0],4096);

      ObjectOutputStream OOout = new ObjectOutputStream(s.getOutputStream());
      ObjectInputStream OOin = new ObjectInputStream(s.getInputStream());

      System.out.println("Enviant la clau pública al servidor...");

      // envio la clau pública al servidor...
      OOout.writeObject(DHKey.getPublicKey());

      System.out.println("Rebent la clau pública del servidor...");

      //espero la clau pública del servidor...
      Key keyServer = (PublicKey) OOin.readObject();

      System.out.println("Generant la clau secreta...");
      DHKey.setSecretKey(keyServer);

      BufferedReader teclado = new BufferedReader(new InputStreamReader(System.in));

      String cadena;

      // creemo un objecte Cipher amb l'algorisme DES
      Cipher cipher = Cipher.getInstance("3DES");

      cipher.init(javax.crypto.Cipher.ENCRYPT_MODE, DHKey.getSecretKey());
      CipherOutputStream cos = new CipherOutputStream(s.getOutputStream(), cipher);

      while (true) {
        System.out.println("Introdueixi el text a enviar (FI = Sortir) : ");
        cadena = teclado.readLine();

        while (((cadena.length()) % 8) != 0) cadena = cadena.concat(" ");
        cos.write(cadena.getBytes(), 0, cadena.length());
        if (cadena.compareTo("FI      ")==0) break;
      }

      teclado.close();
      cos.close();
      s.close();

    } catch (Exception e) { e.printStackTrace(); }
  }
```

La implementació de la part servidora va ser :

```
import java.net.*;
import java.io.*;
import java.security.*;
import javax.crypto.*;
import iaik.security.provider.IAIK;

class ServerDES {

  protected static DHKeyAgreement DHKey = new DHKeyAgreement();

  public static void main(String argv[]) {

    try {

      Security.insertProviderAt(new IAIK(), 2);

      System.out.println("Establint el socket...");
      ServerSocket servidor=new ServerSocket(4096);

      // Esperem que es connecti un client.
      System.out.println("Esperant la connexió d'un client ....");
      Socket s=servidor.accept(); // mètode bloquejant
      System.out.println("Rebuda la connexió d'un client...");

      ObjectOutputStream OOout = new ObjectOutputStream(s.getOutputStream());
      ObjectInputStream OOin = new ObjectInputStream(s.getInputStream());
```

```
      System.out.println("Rebent la clau pública del client...");

      //espero la clau pública del servidor...
      Key keyClient = (PublicKey) OOin.readObject();

      System.out.println("Enviant la clau pública al client...");

      // envio la clau pública al client...
      OOout.writeObject(DHKey.getPublicKey());

      System.out.println("Generant la clau secreta...");
      DHKey.setSecretKey(keyClient);

      // instanciem un objete Cipher amb l'algorisme 3DES
      Cipher cipher = Cipher.getInstance("3DES");
      cipher.init(javax.crypto.Cipher.DECRYPT_MODE, DHKey.getSecretKey());

      byte[] in = new byte[1000];
      byte[] out= new byte[1000];

      int length;
      String cadena;

      CipherInputStream cis = new CipherInputStream(s.getInputStream(), cipher);

     while (true) {
         length = cis.read(in);
         cadena = new String(in,0,length);
         if (cadena.compareTo("FI      ")==0) break;
         System.out.println(cadena);

         //
         // debugging :
         //
         //cipher.init(javax.crypto.Cipher.ENCRYPT_MODE, DHKey.getSecretKey());
         //length = cipher.doFinal(in,0,length,out);
         //System.out.println(new String(out,0,length));
         //cipher.init(javax.crypto.Cipher.DECRYPT_MODE, DHKey.getSecretKey());
         //length = cipher.doFinal(out,0,length,in);
         //System.out.println(new String(in,0,length));
     }
     cis.close();
     s.close();

    } catch (Exception e) { e.printStackTrace(); }
  }
}}
```

Interfície gràfica senzilla a la part client

En aquesta fase vaig dissenyar i implementar una interfície senzilla per la part client :

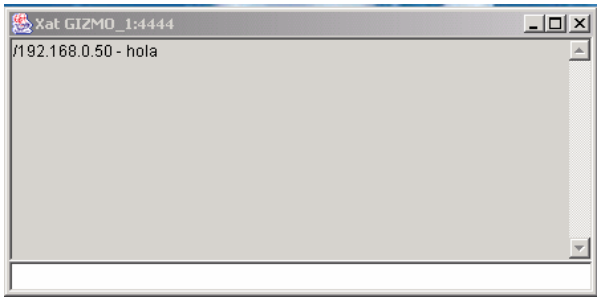


Figura 10 – Interfície gràfica senzilla de la part client

Interfície gràfica senzilla a la part servidor

En aquesta fase vaig dissenyar i implementar interfícies gràfiques senzilles a la part servidor i a la part client. Les finestres eren basades en la classe *Frame* de la llibreria *awt* de java.

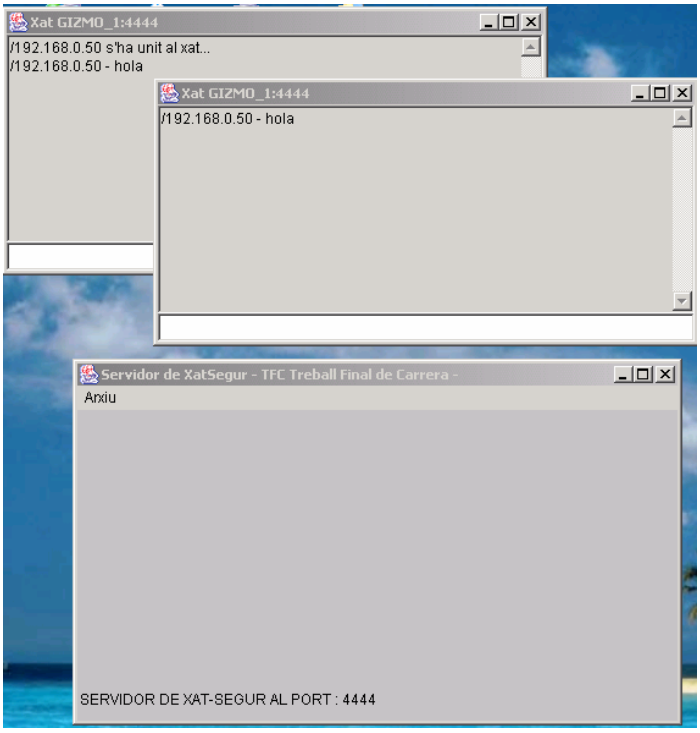


Figura 11 – Interfície gràfica senzilla de la part servidor

També vaig dotar al servidor d'un menú amb la opció de visualitzar l'estat de les connexions actives :

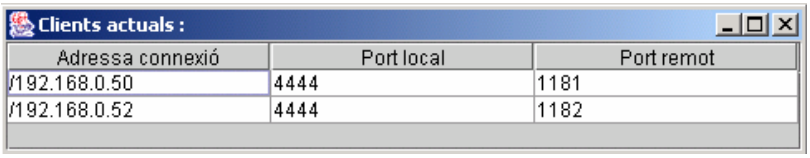


Figura 12 – Clients actuals

Gestió de múltiples servidors i múltiples clients

En aquesta fase dissenyo i implemento els gestors MDI de servidors i de clients. Les finestres passen a ser basades en les classes de la llibreria *swing* de java.

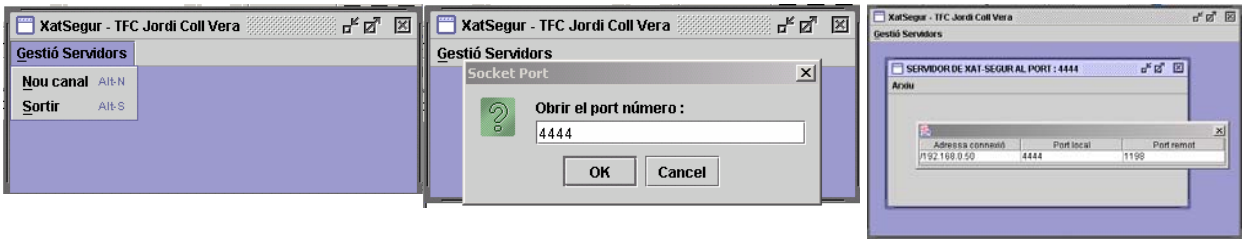


Figura 13 – Gestió MDI de múltiples servidors

Es poden obrir diferents ports (servidors) als quals es poden unir (canals de xat) diferents clients.

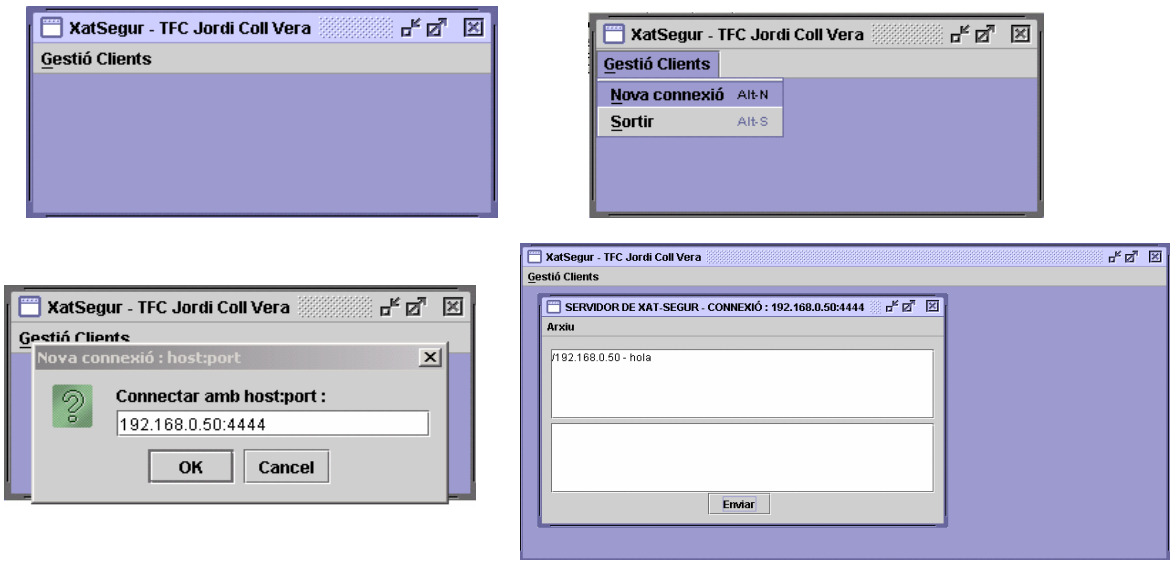
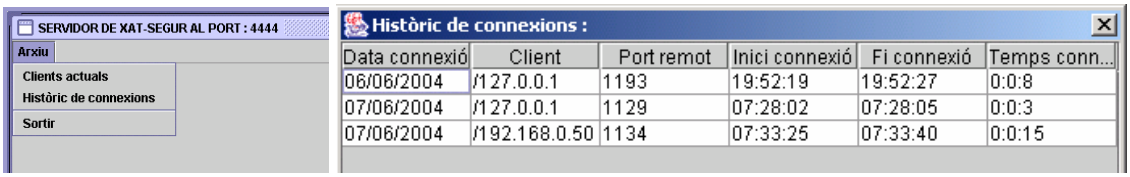


Figura 14 – Gestió MDI de múltiples clients

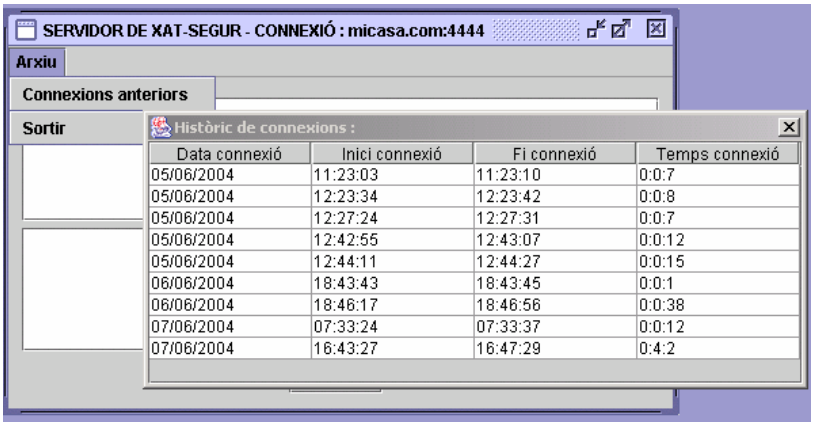
Històric de connexions a la part servidor

S'afegeix la opció de consultar informació històrica sobre la connexió a un determinat port local del servidor.



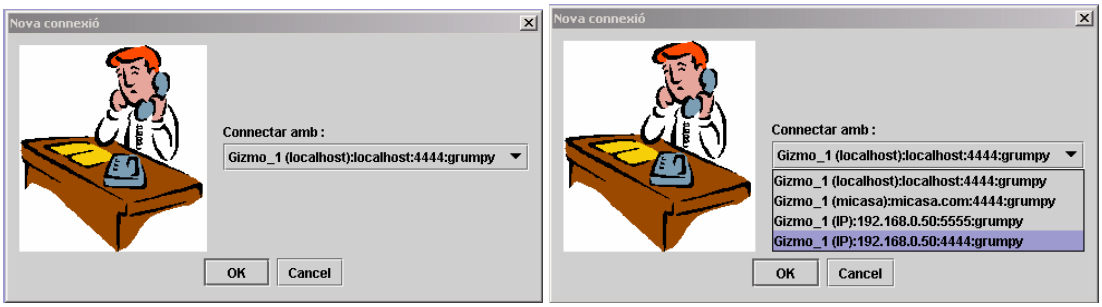
Històric de connexions a la part client

Es manté una base de dades en mode text amb les connexions efectuades a la part client. S'afegeix la opció de consultar informació històrica sobre la connexió a un determinat port remot del servidor.



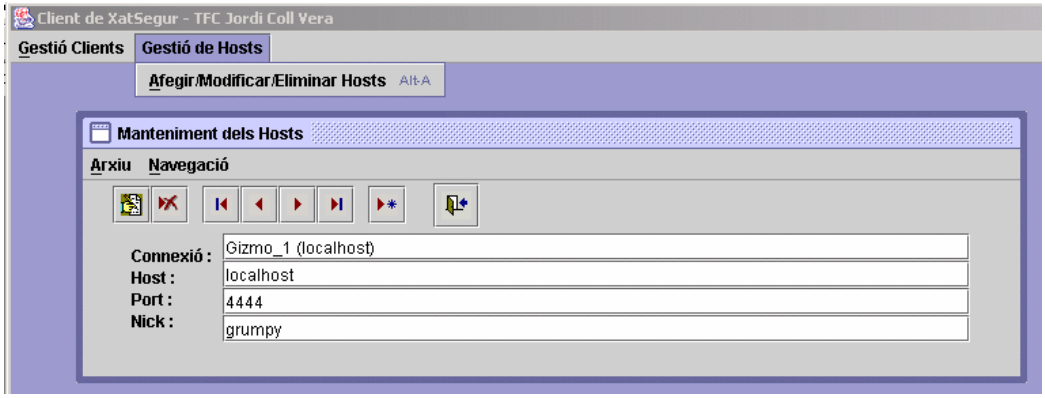
Millora del diàleg de connexió a la part client

Es manté una base de dades en mode text amb la configuració de les connexions a la part client. S'afegeix un quadre de diàleg per efectuar la connexió, seleccionant-la d'una llista.



Pantalla de manteniment de la base de dades de connexions, a la part client

Es manté una base de dades en mode text amb la configuració de les connexions a la part client. S'afegeix una classe per mantenir la base de dades de les connexions.



Applet de client

També s'ha implementat la part client en forma d'Applet d'Internet.

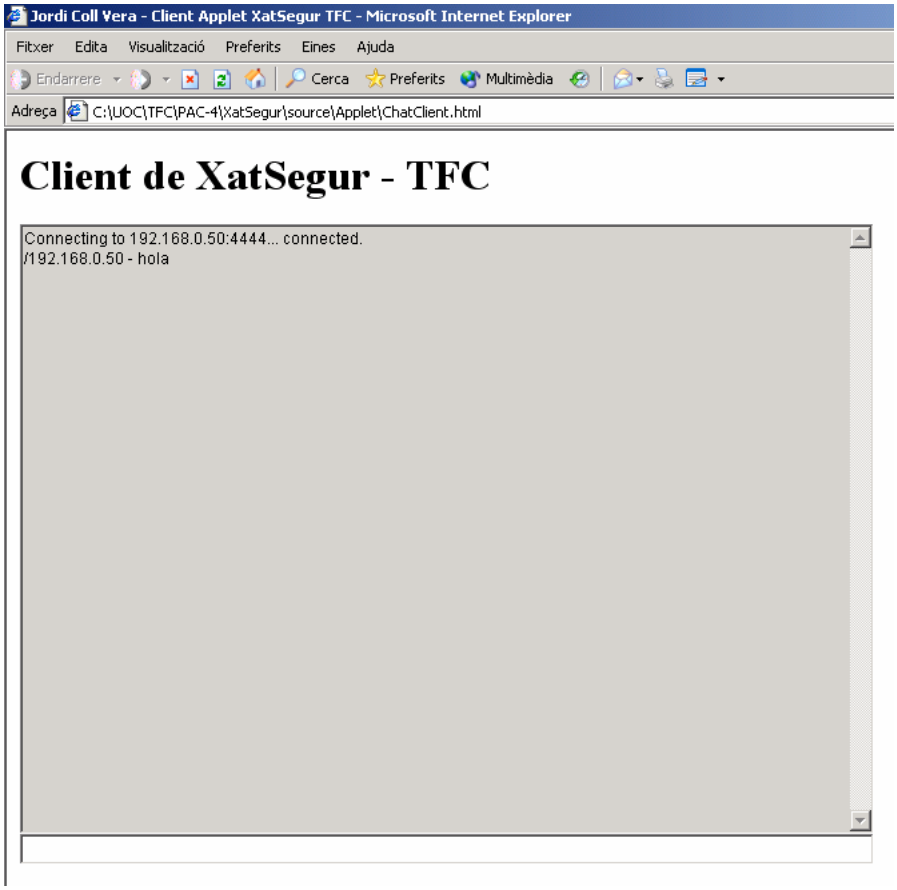


Figura 15 – Applet de client

Diagrames UML

Establiment de connexió i flux de dades xifrades

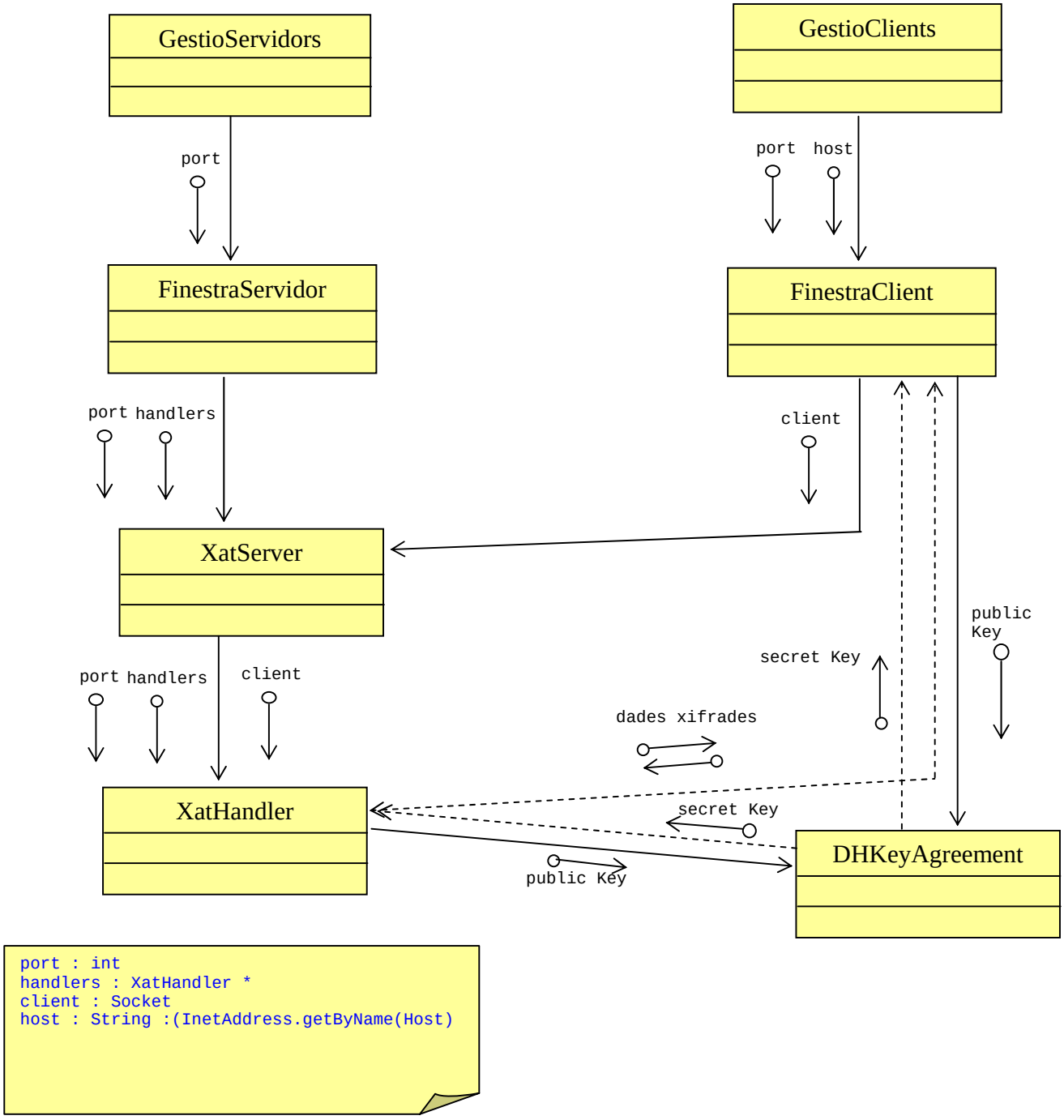


Figura 16 – Diagrama UML d'establiment de connexió i flux de dades xifrades

Llistat de Clients actualment connectats al port del servidor

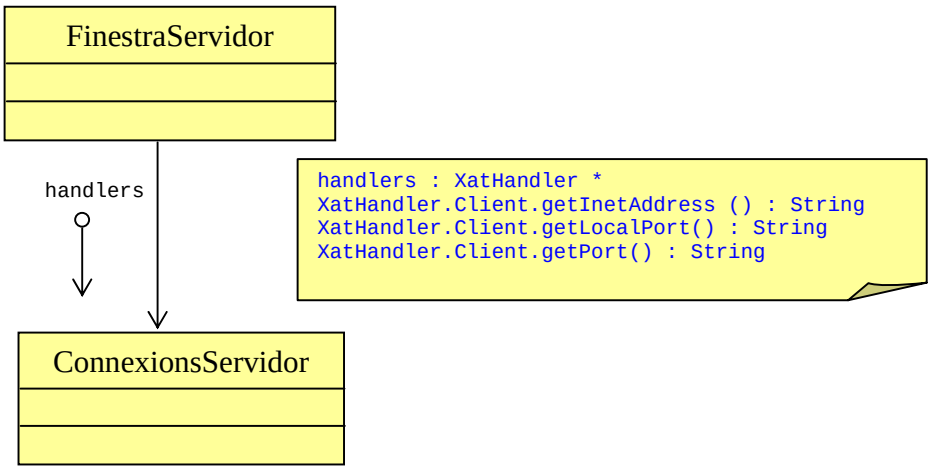


Figura 17 – Diagrama UML del llistat de clients actuals

Llistat històric de Clients connectats al port del servidor

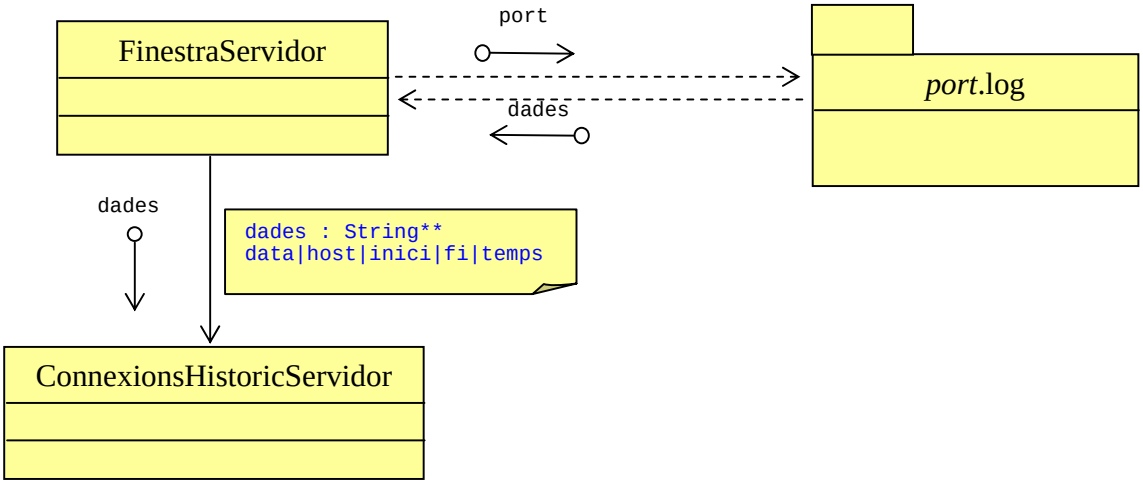


Figura 18 – Diagrama UML del llistat històric de clients/servidor

Llistat històric de connexions del client al port del servidor

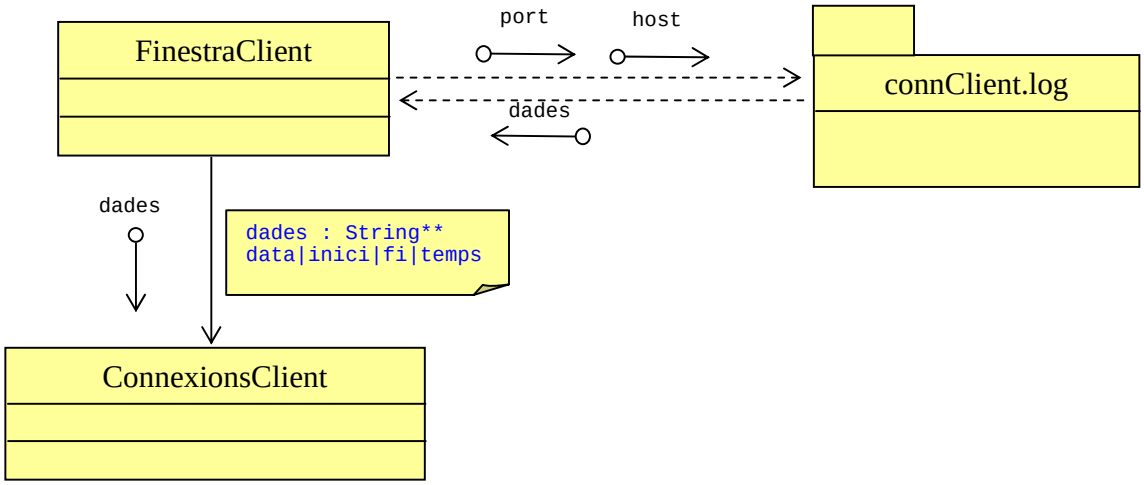


Figura 19 – Diagrama UML del llistat històric de servidor/clients

Manteniment de la base de dades de connexions

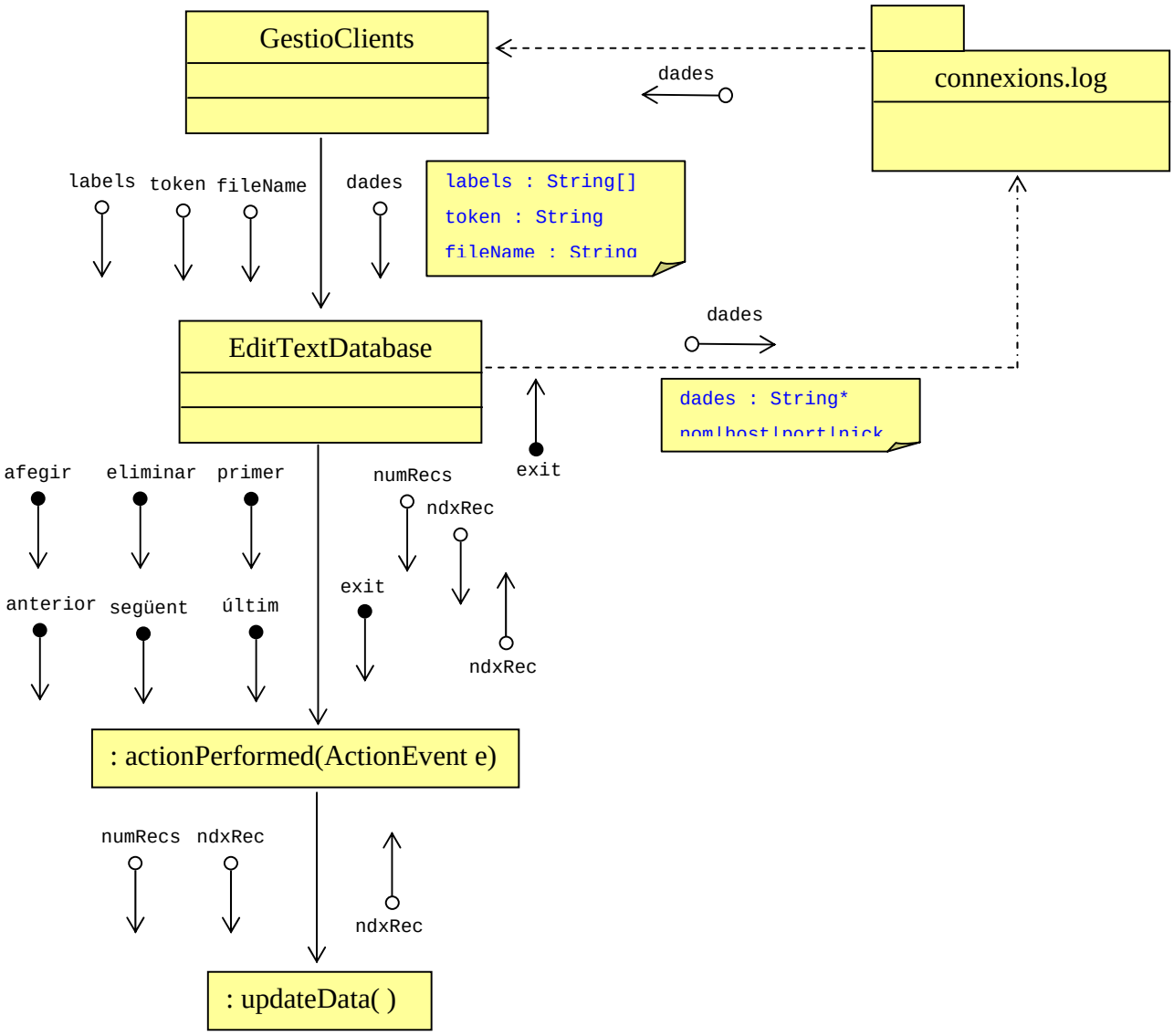


Figura 20 – Diagrama UML del manteniment de la base de dades de connexions

Manual d'usuari

Instal·lació de l'aplicatiu

L'aplicatiu consta d'un únic arxiu anomenat **jcollv_TFC_XatSegur**, comprimit en format zip, que està inclòs en el lliurament final del projecte XatSegur.

Aquest arxiu conté tot el necessari per poder executar l'aplicació XatSegur, a excepció de la plataforma Java que cal tenir prèviament correctament instal·lada (Java es pot descarregar de http://java.sun.com/j2se/1.4.2/download.html). També es subministren les llibreries criptogràfiques IAIK (en unes llibreries JAR) i en aquest aspecte no hauria d'haver problemes d'execució. En cas d'haver-lo s'hauria de descarregar i instal·lar la última versió de la llibreria criptogràfica (la IAIK-JCE3.02eval de *http://jce.iaik.tugraz.at/download/)*

Per instal·lar l'aplicació s'han de seguir els següents passos:

1. crear un directori on instal·lar els arxius de l'aplicació XatSegur.
2. descomprimir **jcollv_TFC_XatSegur.zip** en el directori creat
3. es genera el següent arbre de sots-directoris :

Amb el següent contingut :

- directori **bin** : conté els arxius binaris de l'aplicatiu
- directori **doc** : conté una còpia del manual d'usuari
- directori **javadoc** : conté els arxius de documentació *javadoc*
- directori **run** : conté els arxius per iniciar l'aplicatiu i els arxius binaris a la llibreria XatSegur.jar :
  - part **servidor** : *servidors.bat* , inicia la part servidor
  - part **client** : *clients.bat* , inicia la part client

  També es disposa en aquest directori dels arxius java.security i java.policy que és possible sigui necessari instal·lar al directori C:\j2sdk1.#.#_##\jre\lib\security de la instal·lació de Java
  - directori **conf** : conté la base de dades de tipus text de les connexions del client (*connexions.dat*)
  - directori **images** : conté les imatges necessàries pel funcionament de l'aplicatiu
  - directori **logs** : conté la base de dades de tipus text de l'històric de connexions del client (*connClient.log*) i dels ports del servidor (#####.log)

- directori source : conté els arxius font de l'aplicatiu i els arxius necessaris per compilar-los (*compila.bat*) i per generar la documentació a javadoc (*genera_java_doc.bat*)
- directori **applet** : conté la versió client en format Applet de java
- directori **iaik** : conté les llibreries necessàries de la llibreria criptogràfica IAIK.

Execució de l'aplicació

Un cop finalitzada la instal·lació iniciem la part **servidor** executant l'arxiu *servidors.bat*. Aquest arxiu té el següent contingut :

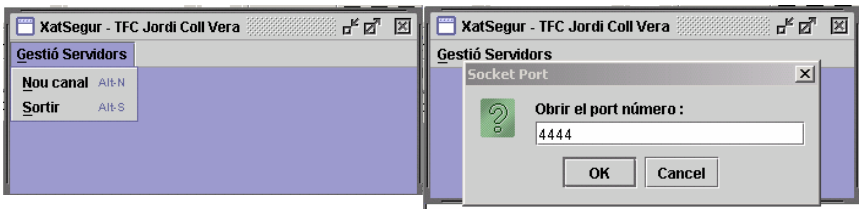
```
java-classpath XatSegur.jar;../source/iaik/iaik_javax_crypto.jar;../source/iaik/iaik_jce.jar
GestioServidors
```

Iniciem la part **client** executant l'arxiu *clients.bat*. Aquest arxiu té el següent contingut :

```
java -classpath
XatSegur.jar;../source/iaik/iaik_javax_crypto.jar;../source/iaik/iaik_jce.jar GestioClients
```

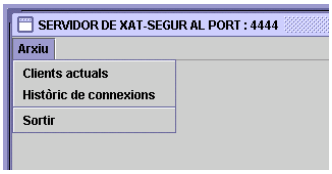
Finestra de servidor

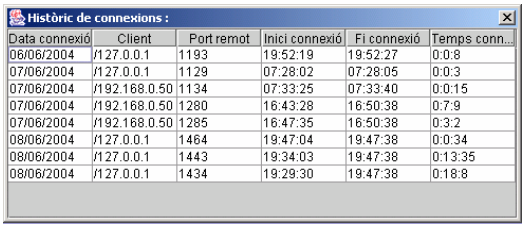
Des de la pantalla principal del servidor iniciem una finestra de servidor (a la que es connectaran els clients), amb la opció Nou canal del menú.

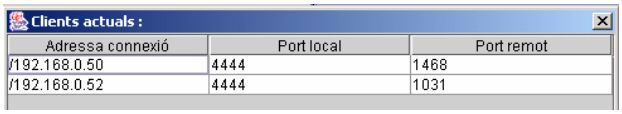


Consultes de la part servidor

Des de la pantalla de la finestra de servidor podem executar dues consultes (llistats per pantalla) : la consulta dels clients actualment connectats al port del servidor, i la de l'històric de connexions a aquest mateix port.

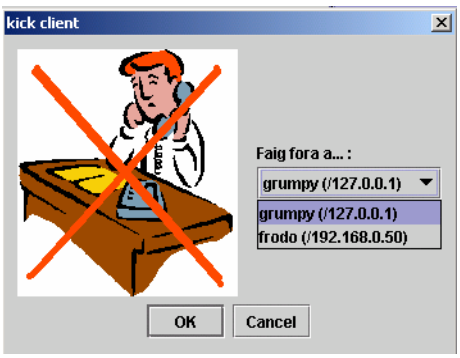






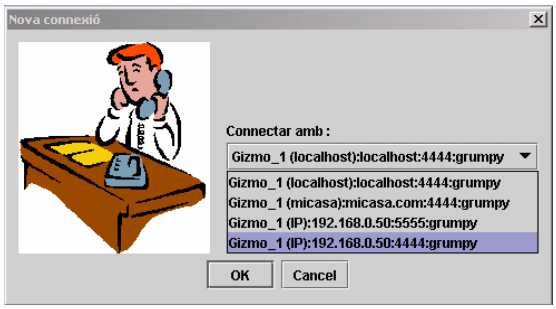
Fer fora un client (*kick*)

Des de la pantalla de la finestra de servidor podem fer fora un client determinat, seleccionant-lo de la llista de clients actuals.



Finestra de client

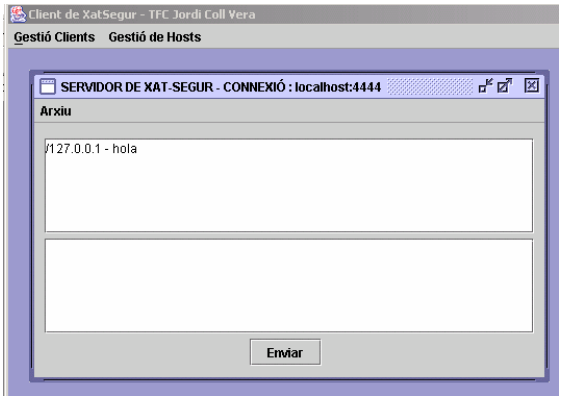
Des de la pantalla principal del client iniciem una finestra de client (amb la que gestionarem la comunicació amb el port del servidor que està escoltant), amb la opció Nova Connexió del menú Gestió Clients.



Comunicació amb la resta de clients (a través del servidor)

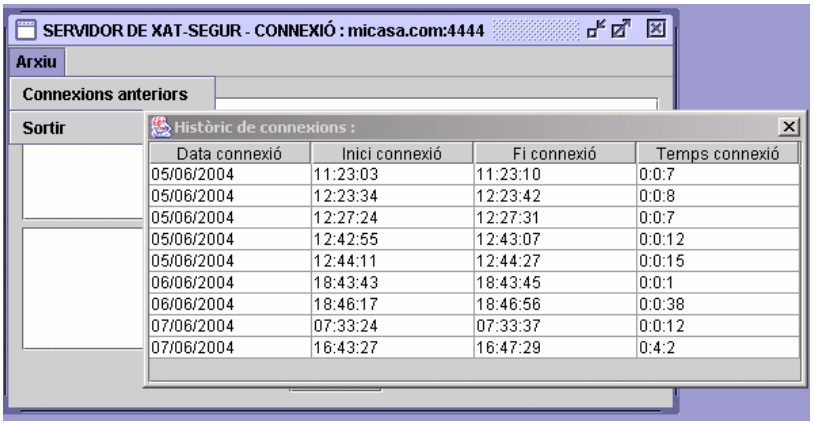
La comunicació s'efectua a través d'una finestra molt senzilla. S'escriu a la part inferior i al pitjar el botó *enviar* s'envia el missatge xifrat al servidor i aquest fa el broadcast xifrat a la resta de clients connectats i al propi client que envia el missatge.

Al obrir la finestra i al tancar-la, s'envien un missatge de benvinguda i de comiat a la resta de clients connectats.



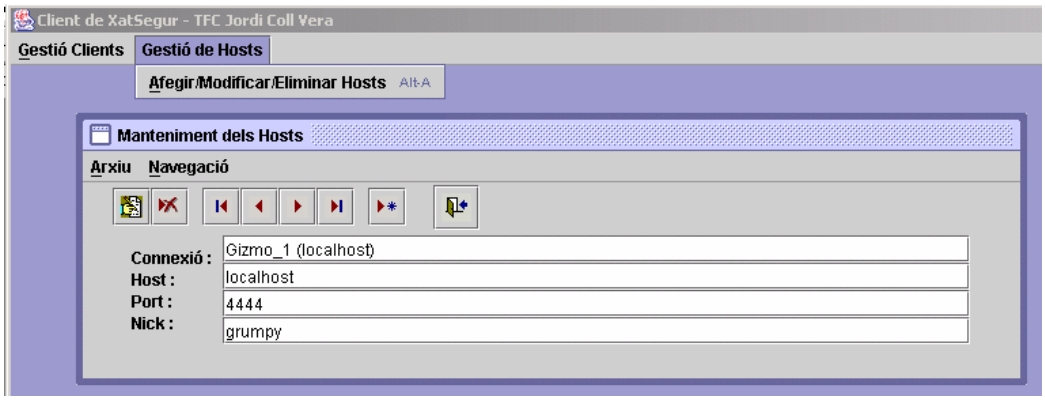
Consultes de la part client

Des de la pantalla de la finestra de client podem executar la consulta del històric de connexions d'aquest client al mateix port del servidor.



Manteniment de la base de dades de connexions

Les dades que apareixen a la finestra de clients, a la opció Nova Connexió, són a la base de dades ./conf/connexions.dat i es mantenen amb la opció Afegir/Modificar... del menú Gestió de Hosts

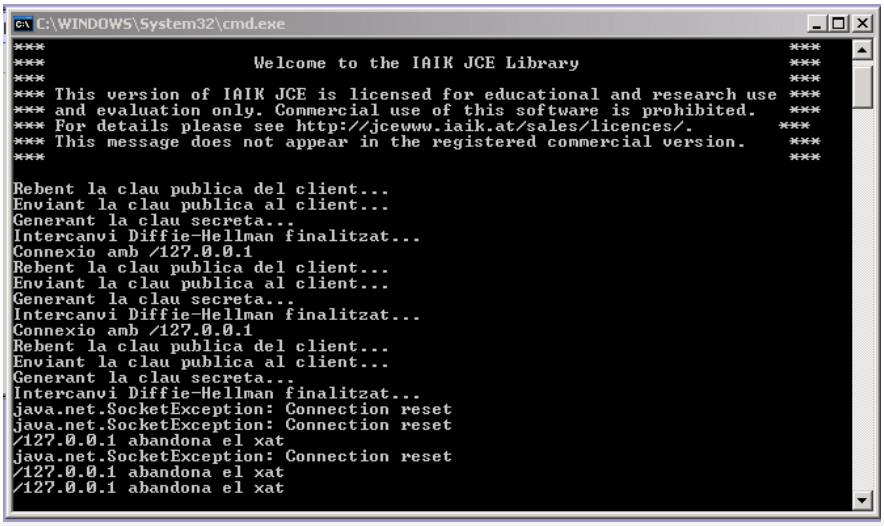


Disposem de les opcions :Modificar el registre actual, Eliminar registre , Posicionar-se al primer registre, Posicionar-se al registre anterior, Posicionar-se al registre següent, Posicionar-se a l'últim registre, Afegir un nou registre i Sortir de la pantalla. Aquestes opcions són disposades en el mateix ordre a la barra de botons i al menú Navegació.

Cada cop que es modifica alguna dada i canviem de registre (posicionant-nos en un altre) les dades queden modificades en memòria però no al disc. Les dades són realment actualitzades al disc si responem positivament a la pregunta que se'ns fa al sortir de la pantalla.

Missatges informatius a les finestres del *runtime* de Java

L'aplicació genera missatges informatius a la sortida estàndard (pantalles *runtime* de Java)



Conclussions

Aquest treball representa una versió simplificada però força operativa del que seria un sistema comercial de xat segur. L'objectiu del treball no era en tot cas aquest, que queda fora de l'abast d'una persona i seria pròpia d'un equip de programadors, sinó el posar en pràctica el que he après al llarg dels estudis d'ETIS.

Nogensmenys, penso que el treball assoleix tots els objectius proposats al pla de treball inicial , pla que he seguit estrictament.

En una versió posterior podria afegir noves funcionalitats :

- Canals privats entre usuaris
- Descàrregues d'arxius
- Canals de vídeoconferència
- Canals de veu
- algorismes de xifrat més ràpids i robustos (comentats a l'apartat *estat de l'art*)
- es pot millorar l'establiment de la connexió i intercanvi de claus, amb un sistema de tipus *Hello / Challenge*.

El treball m'ha semblat ben planificat i dirigit, amb una relació correcta hores de treball / crèdits de la matèria.

Bibliografia i Recursos de Documentació

- Anna Rio, *CRIPTOGRAFIA-FIB, Tema 5*, Departament de Matemàtica Aplicada II Universitat Politècnica de Catalunya, March 15, 2004
- FIPS PUB 46-3, *DATA ENCRYPTION STANDARD (DES),* U.S. DEPARTMENT OF COMMERCE/National Institute of Standards and Technology, Reaffirmed 1999 October 25, CATEGORY: COMPUTER SECURITY, SUBCATEGORY: CRYPTOGRAPHY
- B. Schneier, *Applied Cryptography*. (2nd Edition). New York: Wiley, 1996.
- Alfred J. Menezes, Paul C. van Oorschot i Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1999. http://cacr.math.uwaterloo.ca/hac/
- Oppliger, R. *Security technologies for the Word Wide Web*. Artech House. 2000
- Carlisle Adams, Steve Lloyd. *Understanding PKI: Concepts, Standards, and Deployment Considerations*. (2nd Edition). Addison Wesley Professional. 2002.
- "RSA Data Security, Inc. Public-Key Cryptography Standards (PKCS)" #3: Diffie-Hellman Key Agreement Standard. Which is based upon W. Diffie and M.E. Hellman's *New directions in cryptography* from IEEE transactions on Information Theory, IT 22:644-654, 1976.
- Loosemore, S.; Stallman, R.M.; McGrath, R.; Oram, A.; Drepper, U. (1992). *The GNU C Library Reference Manua*l. Boston: Free Software Foundation.
- Márquez García, F.M. (1996). *UNIX. Programación avanzada*. Madrid: Ra-ma.
- Orfali, R. (1998). *Client/server programming with Java and CORB*A. Nova York: Wiley & sons.
- Rifflet, J.M. (1992). *Comunicaciones en UNIX*. Madrid: McGraw-Hill.
- Enric Peig Olivé, Xavier Perramon Tornil, *Xarxes* (Apunts de l'assignatura – *Programació de sockets* -), Universitat Oberta de Catalunya 2003
- Josep Domingo Ferrer, Jordi Herrera Joancomartí, Ricardo X. Sánchez del Castillo, *Criptografia* (Apunts de l'assignatura), Universitat Oberta de Catalunya 1999
- Michael D. Thomas, Pratik R. Patel, Alan D. Hudson, Donald A. BAll JR., *Programación en Java para Internet*, Paraninfo 1997
- Jamie Jaworski, *Java 1.2 Al descubierto*, Sams Prentice Hall 2000

Pàgines WEB visitades :

- http://www.htmlweb.net/seguridad/seguridad.html
- http://www.undersec.com/sjfproject/prog/sockets/sockets.htm
- http://www.monografias.com/trabajos12/chtjava/chtjava.shtml
- http://www.rsasecurity.com/rsalabs/faq/
- http://java.sun.com/docs/books/tutorial/networking/TOC.html#sockets
- http://java.sun.com/products/jce/doc/apidoc/index.html
- http://www.logi.org/logi.crypto/
- http://www.thefreecountry.com/sourcecode/encryption.shtml

Recursos de Maquinari i Programari

Per poder portar a bon terme el present projecte s'ha necessitat disposar d'un ordinador connectat a Internet a més a més de tenir instal·lat el software següent:

- Microsoft® *Windows XP* com a Sistema Operatiu

- Microsoft® *Word 2000*, com a eina per redactar els documents de la documentació del projecte.
- Microsoft® *PowerPoint 2000*, com a eina per redactar la presentació del projecte.
- Microsoft® *Excel 2000* com a eina per desenvolupar l'schedule del projecte.
- La llibreria criptogràfica IAIK-JCE3.02eval (*http://jce.iaik.tugraz.at/download/*)
- Java™ 2 SDK, Standard Edition Version 1.4.1 de Sun Microsystems®
- Helios Software Solutions © T*extPad*, editor de mode texte per programació en Java
- David Mold © *RealJ*, entorn de desenvolupament en Java
- Microsoft® *Visio* pel disseny

ANNEX : Classes implementades al projecte XatSegur

Classe XatServer

XatServer
#client : Socket #handlers : XatHandler* #port : int #server : ServerSocket #c : XatHandler #actiu : boolean
+XatServer (pPort : int, pHandlers : XatHandler*) +getServerSocket() : ServerSocket +run() +close()

Codi de la classe XatServer

```
/*
*
* Classe que gestiona la connexió servidora multifil a un port de XatSegur
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*
*/
import java.net.*;
import java.io.*;
import java.util.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.JOptionPane;

/**
*
* Aquesta classe gestiona la connexió del servidor a un port de XatSegur.
* <p>
* Cada cop que hi accedeix un client nou es genera un nou fil de connexió
* @author Jordi Coll Vera.
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
* @extends Thread
*
**/
public class XatServer extends Thread {

    /** Atribut que representa el socket del client*/
    protected Socket client = null;

    /** Vector amb les connexions actives*/
    protected Vector handlers;

    /** Atribut que representa el port servidor de la connexió*/
    protected int port;

    /** Atribut que representa el socket servidor de la connexió*/
    protected ServerSocket server;

    /** Atribut que representa un objecte de la classe XatHandler.
        Classe que gestiona la connexió del client*/
    protected XatHandler c;

    /** Atribut que representa un flag que indica que el servidor és actiu*/
    protected boolean actiu = true;

    /**
    * Constructor de la classe XatServer.
```

```
    * <p>
    * @param port int que representa el port de la connexió.
    * @param handlers Vector per emmagatzemar les connexions dels clients
    * @exception throws IOException
    **/
    public XatServer (int port, Vector handlers) throws IOException {
        this.port = port;
        this.handlers = handlers;
    }

    /**
    *
    * Accessor de lectura del socket servidor de la connexió.
    * <p>
    * {<b>Pre:</b> x és una instància de la classe XatServer i server
    * l'atribut que conté el ServerSocket de x}
    * <p>
    * {<b>Post:</b> Es retorna server de x}
    **/
    public ServerSocket getServerSocket() {
        return server;
    }

    // run ...
    public void run () {
        try {
            server = new ServerSocket (port);
            while (actiu) {
                if (! server.isClosed() & actiu){
                    try {

                        // escolta una nova connexió...
                        client = server.accept ();
                        System.out.println ("Connexio amb " + client.getInetAddress ());

                        // genera un nou fil...
                        c = new XatHandler (client, handlers, port);
                        c.start ();

                    } catch(Exception e) {}
                }
            }

        } catch(Exception e) {
            JOptionPane.showMessageDialog(null, e.toString(),"Error XatServer
ServerSocket",JOptionPane.ERROR_MESSAGE);
        }
    }

    // tanca...
    public void close() {
        try {
            //c.destroy();
            actiu = false;
            c = null;
            if (client != null) client.close();
            server.close();

        } catch(Exception e) {
            JOptionPane.showMessageDialog(null, e.toString(),"Error XatServer tancat
socket",JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

Comentaris de la classe XatServer

La classe gestiona els fils de connexions al servidor. La classe instancia un socket TCP (`server`) que escolta el port del servidor (`port`) la connexió d'un client. Cada cop que un nou client s'uneix a *XatSegur* es genera un nou fil amb la classe *XatHandler*. Aquesta classe rep com a paràmetres

el socket del client que s'ha unit al xat (`client`) , el vector per manegar les connexions (`handlers`) i el número de port local del servidor (`port`).

La classe es crida des de la classe *FinestraServidor* cada cop que es crea una nova finestra des de la classe *GestioServidors*, generant-se així un nou fil que manega la connexió del servidor a un port determinat.

Classe XatHandler

XatHandler
#s : Socket #handlers : XatHandler* #port : int #keyClient: Key #cis : CipherInputStream #cos : CipherOutputStream #DHKey: DHKeyAgreement #dataConnexio: Date #nick: String
+XatHandler (pSocket : Socket, pHandlers : XatHandler*, pPort : int) +getCipherOutputStream () : CipherOutputStream +getSocket() : Socket +getPort() : int +run() -escriuLogHistoric(pFileName : String) #broadcast(pMessage : String)

Codi de la classe XatHandler

```
/*
*
* Classe que gestiona els fils de la connexió dels clients a XatSegur
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*
*/
import java.net.*;
import java.io.*;
import java.util.*;
import java.lang.String;
import java.security.Security;
import iaik.security.provider.IAIK;
import java.security.Key;
import java.security.PublicKey;
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import java.text.DateFormat;


/**
*
* Aquesta classe gestiona la connexió dels clients de XatSegur per un sistema
* multifil..
* <p>
* Cada cop que hi accedeix un client nou es genera un nou fil de connexió i aquesta classe
* el manega.
* @author Jordi Coll Vera.
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
* @extends Thread
*
**/
public class  XatHandler  extends Thread {

    /** Atribut que representa el socket del client*/
```

```
protected Socket s;

/** Atribut que representa el port local de la connexió*/
protected int port;

/** Vector amb les connexions actives*/
protected Vector handlers;

/** Atribut que representa la clau pública del client*/
protected Key keyClient = null;

/** Atribut que representa el flux de dades (xifrades) d'entrada*/
protected CipherInputStream cis;

/** Atribut que representa el flux de dades (xifrades) de sortida*/
protected CipherOutputStream cos;

/** Atribut que representa un objecte de la classe DHKeyAgreement*/
protected DHKeyAgreement DHKey = new DHKeyAgreement();

/** Atribut que representa la data d'inici de la connexió*/
protected Date dataConnexio;

/** Atribut que representa al nick del client*/
protected String nick;

/**
* Constructor de la classe XatHandler.
* <p>
* @param s socket de la connexió del client.
* @param handlers Vector per emmagatzemar la connexió del client
* @exception throws IOException
**/
public XatHandler (Socket s, Vector handlers, int port) throws IOException {
    this.s = s;
    this.handlers = handlers;
    this.port = port;
}

/**
*
* Accessor de lectura del flux de dades xifrades de sortida
* <p>
* {<b>Pre:</b> x és una instància de la classe XatHandler i cos
* l'atribut que representa el flux de dades xifrades de sortida de x}
* <p>
* {<b>Post:</b> Es retorna cos de x}
**/
public CipherOutputStream getCipherOutputStream() {
    return cos;
}

/**
*
* Accessor de lectura del socket del client
* <p>
* {<b>Pre:</b> x és una instància de la classe XatHandler i s
* l'atribut que representa el socket del client de x}
* <p>
* {<b>Post:</b> Es retorna s de x}
**/
public Socket getSocket() {
    return s;
}

/**
*
* Accessor de lectura del port local
* <p>
* {<b>Pre:</b> x és una instància de la classe XatHandler i port
* l'atribut que representa el port local del servidor de x}
* <p>
* {<b>Post:</b> Es retorna port de x}
```

```
**/
public int getPort() {
    return port;
}


// run ...
public void run () {

    // capturo l'adreça Inet de l socket del client
    String name = s.getInetAddress ().toString ();

    // inicialitzo la data d'inici de la connexió
    dataConnexio = new Date();

    try {

        Security.insertProviderAt(new IAIK(), 2);

        // instancio un flux de dades (objecte) de sortida i un d'entrada
        // per rebre i enviar les claus públiques
        ObjectOutputStream OOout = new ObjectOutputStream(s.getOutputStream());
        ObjectInputStream OOin = new ObjectInputStream(s.getInputStream());

        System.out.println("Rebent la clau publica del client...");

        //espero la clau pública del client...
        while (keyClient == null)
            keyClient = (PublicKey) OOin.readObject();

        System.out.println("Enviant la clau publica al client...");

        // envio la clau pública al client...
        OOout.writeObject(DHKey.getPublicKey());

        System.out.println("Generant la clau secreta...");
        DHKey.setSecretKey(keyClient);

        System.out.println("Intercanvi Diffie-Hellman finalitzat...");

        // instanciem un objete Cipher amb l'algorisme 3DES per desxifrar
        // els missatges entrants
        Cipher cipherIn = Cipher.getInstance("3DES/ECB/NoPadding");
        cipherIn.init(javax.crypto.Cipher.DECRYPT_MODE, DHKey.getSecretKey());

        cis = new CipherInputStream(s.getInputStream(), cipherIn);

        // instanciem un objete Cipher amb l'algorisme 3DES
        // per xifrar els missatges sortints
        Cipher cipherOut = Cipher.getInstance("3DES/ECB/NoPadding");
        cipherOut.init(javax.crypto.Cipher.ENCRYPT_MODE, DHKey.getSecretKey());

        cos = new CipherOutputStream(s.getOutputStream(), cipherOut);

        byte[] in = new byte[1000];
        byte[] out= new byte[1000];

        int length;
        String cadena;

        // espero el Nick
        length = cis.read(in);
        nick = (new String(in,0,length + 1)).trim();
        broadcast (nick + " s'ha unit al xat...");
        handlers.addElement (this);

        while (true) {
            length = cis.read(in);
            cadena = new String(in,0,length + 1);
            broadcast (nick + " - " + cadena);
        }

    } catch (Exception ex) {
```

```
            System.out.println(ex.toString());

        } finally {
            handlers.removeElement (this);
            broadcast (name + " ha abandonat el xat.");
            try {
                System.out.println(nick + " (" + name + ") abandona el xat");
                escriuLogHistoric(nick + " (" + name + ")");

                cis.close();
                cos.close();
                s.close();

            } catch (IOException ex) {
                ex.printStackTrace();
            }
        }
    }

    // escriu l'arxiu log de les dades històriques
    private void escriuLogHistoric(String name) {
        try {
                // escric al fitxer de log de connexions al servidor, en mode append
                // si el fitxer no existeix el crea de nou
                // al final s'afegeix un salt de línia
                PrintStream out = new PrintStream( new FileOutputStream("./logs/" +
                                    (new Integer(getPort())).toString() + ".log", true));
                out.print( DateFormat.getDateInstance().format(dataConnexio) + "|" +
                           nick + " (" + name + ")|" +
                           (new Integer(s.getPort())).toString() + "|" +
                           DateFormat.getTimeInstance().format(dataConnexio));

                Date fiConnexio = new Date();
                String tempsConnexio = (new TempsConnexio(fiConnexio.getTime() -
                                     dataConnexio.getTime()+10)).toString();

                out.println("|" + DateFormat.getTimeInstance().format(fiConnexio) + "|" +
                                tempsConnexio );
                out.close();

            } catch(FileNotFoundException  fnfExc) {System.err.println("El fitxer log no
                                                    existeix");
            }
    }

    // broadcast dels missatges a tots els clients del xat...
    protected void broadcast (String message) {
        synchronized (handlers) {
            // recorro el vector dels clients...
            Enumeration e = handlers.elements ();
            while (e.hasMoreElements ()) {
                XatHandler c = (XatHandler) e.nextElement ();
                try {
                    synchronized (c.cos) {

                        // allargo el missatge amb blancs fins a 512 bytes
                        while (((message.length()) % 512) != 0)
                            message = message.concat(" ");

                        // escric el missatge al flux de sortida xifrada
                        c.cos.write(message.getBytes());
                        c.cos.flush();
                    }

                } catch (IOException ex) {
                    c = null;
                }
            }
        }
    }
}
```

Comentaris de la classe XatHandler

La classe gestiona el fil de la connexió d'un determinat client de XatSegur al socket d'un port determinat del servidor a través de la classe XatServer. La classe genera en el servidor una còpia local del socket TCP de la connexió (`s`), del seu port (`port`) i del vector manegador de les connexions actives (`handlers`).

Un cop s'ha establert la connexió. La classe s'encarrega de rebre del client la seva clau pública, enviar al client la clau pública de la part servidor i generar a la part del servidor la clau privada que permetrà establir la connexió xifrada amb l'algorisme 3DES.

Un cop client i servidor s'han entès (amb l'intercanvi de claus de *Diffie-Hellmann*), la comunicació xifrada s'estableix a través dels objectes *CipherOutputStream* (`cos`) i *CipherInputStream* (`cis`) de la pròpia classe.

Cada cop que el client escriu al *stream* de sortida s'envia el missatge xifrat al servidor, que s'encarrega de fer el broadcast a la resta de clients connectats al mateix port a través de la classe *XatHandler*.

Quan el client abandona *XatSegur* s'executa el mètode *escriuLogHistoric* que genera una línia de dades persistents a l'arxiu *./logs/#####.log*, on ***#####*** representa el número del port local del servidor.

La classe es crida des de la classe *FinestraServidor* cada cop que un nou client s'incorpora a *XatSegur*.

Classes DHKeyAgreement i User

DHKeyAgreement

`-p : BigInteger`
`-g : BigInteger`
`-params : DHParameterSpec`
`-random : SecureRandom`
`-user : User`
`-publicKey : Key`
`-secretKey : Key`

`+DHKeyAgreement()`
`#setPublicKey()`
`+setSecretKey(pPublicKey : Key)`
`+getPublicKey() : Key`
`+getSecretKey() : Key`

User
`-key_pair : KeyPair`

`+User()`
`+getPublicKey() : Key`
`+agreeSecret(pPublicKey : Key) : Key`

Codi de les classes DHKEYAgreement i User

```
/*
*
* Classe que gestiona les claus públiques i privades a XatSegur
* segons el protocol de Diffie-Hellman
* @author Jordi Coll Vera
* Basada en la classe del mateix nom de la llibreria IAIK
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*
*/
```

```
import java.math.BigInteger;
import java.security.*;
import java.io.IOException;
import javax.crypto.KeyAgreement;
import javax.crypto.spec.DHParameterSpec;
import iaik.security.dh.DHKeyPairGenerator;
import iaik.security.random.SecRandom;
import iaik.security.provider.IAIK;

/**
*
* Aquesta classe gestiona les claus sota el protocol Diffie-Hellman.
* <p>
* @author Jordi Coll Vera.
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*
**/
public class DHKeyAgreement {

    /** Atribut que representa un nombre primer precalculat*/
    BigInteger p = new
BigInteger("da583c16d9852289d0e4af756f4cca92dd4be533b804fb0fed94ef9c8a4403ed574650d36999db29
d776276ba2d3d412e218f4dd1e084cf6d8003e7c4774e833", 16);

    /** Atribut que representa un nombre primer precalculat*/
    BigInteger g = BigInteger.valueOf(2);

    /** Atribut que representa els paràmetres per l'intercanvi de Diffie-Hellman */
    DHParameterSpec params;

    /** Llavor aleatòria de la clau */
    SecureRandom random;

    /** Atribut que representa un usuari */
    User user;

    /** Atribut que representa la clau pública de l'usuari */
    Key publicKey;

    /** Atribut que representa la clau privada de l'usuari */
    Key secretKey;

    /**
    * Constructor per defecte de la classe DHKeyAgreement.
    **/
    public DHKeyAgreement() {
        random = SecRandom.getDefault();
        params = new DHParameterSpec(p, g);
        try {
            user = new User();
            setPublicKey();
        } catch(GeneralSecurityException ex) {
          System.out.println("Excepcio: "+ex.toString());
        }
    }

    /**
    *
    * Accessor d'escriptura a la clau pública Diffie-Hellman.
    * <p>
    * {<b>Pre:</b> x és una instància de la classe DHKeyAgreement i pb
    * la clau pública del parell de claus de l'usuari - user.getPublicKey() - }
    * <p>
    * {<b>Post:</b> S'ha assignat pb a l'atribut publicKey de x}
    * <p>
    **/
    protected void setPublicKey() {
        publicKey = user.getPublicKey();
    }

    /**
    *
    * Accessor d'escriptura a la clau privada Diffie-Hellman.
```

```
* <p>
* {<b>Pre:</b> x és una instància de la classe DHKeyAgreement i pv
* la clau privada del parell de claus de l'usuari - user.agreeSecret(pubKey) - }
* <p>
* {<b>Post:</b> S'ha assignat pv a l'atribut secretKey de x}
* <p>
* @param pubKey objecte de la classe Key que representa la clau pública de x
**/
public void setSecretKey(Key pubKey) {
    try {
        secretKey = user.agreeSecret(pubKey);
    } catch(GeneralSecurityException ex) {
        System.out.println("Excepcio: " + ex.toString());
    }
}

/**
*
* Accessor de lectura de la clau pública Diffie-Hellman.
* <p>
* {<b>Pre:</b> x és una instància de la classe DHKeyAgreement i publicKey
* l'atribut que conté la clau pública de x}
* <p>
* {<b>Post:</b> Es retorna publicKey de x}
**/
public Key getPublicKey() {
    return publicKey;
}

/**
*
* Accessor de lectura de la clau privada Diffie-Hellman.
* <p>
* {<b>Pre:</b> x és una instància de la classe DHKeyAgreement i secretKey
* l'atribut que conté la clau privada de x}
* <p>
* {<b>Post:</b> Es retorna secretKey de x}
**/
public Key getSecretKey() {
    return secretKey;
}


/**
* Classe que implementa un usuari propietari d'un parell de claus Diffie-Hellman.
* Aquest parell de claus s'utilitza per efectuar l'intercanvi.
*/
class User {

/**
 * Atribut que representa el parell de claus de l'usuari.
 */
KeyPair key_pair;

    /**
     * Constructor per defecte. Genera un nou parell de claus DH per l'usuari.
     *
     * @exception GeneralSecurityException
     */
    public User() throws GeneralSecurityException {
        try {
            // Cast necessari només amb JDK 1.1.x.
            // La classe KeyPairGenerator de JDK 1.1.2 inclou aquest constructor.
            DHKeyPairGenerator generator = (DHKeyPairGenerator)
                                           KeyPairGenerator.getInstance("DH", "IAIK");
            generator.initialize(params, random);
            key_pair = generator.generateKeyPair();

        } catch (InvalidAlgorithmParameterException ex) {
            throw new GeneralSecurityException(ex.toString());

        } catch (NoSuchAlgorithmException ex) {
            throw new GeneralSecurityException(ex.toString());
```

```
            } catch (NoSuchProviderException ex) {
                throw new GeneralSecurityException(ex.toString());
            }
        }

        /**
         * Retorna la clau pública de l'usuari. Això és similar a enviar
         * la clau a un directori públic.
         */
        public Key getPublicKey() {
            return key_pair.getPublic();
        }

        /**
         * Aquest mètode usa la clau pública d'un altre usuari per tal de generar
         * la clau privada comuna.
         *
         * @param publicKey la clau pública de l'altre usuari
         * @return la clau privada compartida
         *
         * @exception GeneralSecurityException
         */
        public Key agreeSecret(Key publicKey) throws GeneralSecurityException {

            try {
                // instancia un nou objecte de la classe KeyAgreement
                KeyAgreement key_agreement = KeyAgreement.getInstance("DH", "IAIK");

                // l'inicialitza amb la clau privada de l'usuari
                key_agreement.init(key_pair.getPrivate());

                // en una única fase (només hi ha dues entitats...)
                key_agreement.doPhase(publicKey, true);

                // genera la clau privada amb l'algorisme 3DES
                Key secret = key_agreement.generateSecret("3DES");

                // retorna la clau privada de l'usuari
                return secret;

            } catch (InvalidKeyException ex) {
                throw new GeneralSecurityException(ex.toString());
            } catch (NoSuchProviderException ex) {
                throw new GeneralSecurityException(ex.toString());
            } catch (NoSuchAlgorithmException ex) {
                throw new GeneralSecurityException(ex.toString());
            }
        }
    }
}
```

Comentaris de la classe DHKeyAgreement

La classe gestiona l'intercanvi de claus de *Diffie-Hellmann*. Gestiona la parella de claus pública/privada d'un usuari (representat amb la classe `User`). El constructor utilitza dues instàncies de *BigInteger* precalculades.

La classe es crida des de les classes *XatHandler* (part *servidor*) i des de *FinestraClient* cada cop que es genera el nou fil de la connexió des de la part *client*.

Classe TempsConnexio

TempsConnexio
#segons : long #minuts : long #hores : long
+TempsConnexio (pTimeMillis : long)

```
+toString () : String
```

Codi de la classe TempsConnexio

```
/**
*
* Aquesta classe retorna la informació del temps de connexió (un número long que representa
* milisegons ) en un format inteligible : hores:minuts:segons
* <p>
* @author Jordi Coll Vera.
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*/
public class TempsConnexio {
    long segons;
    long minuts;
    long hores;

    /**
    * Constructor de la classe TempsConnexio.
    * <p>
    * Genera una instància de TempsConnexio que conté les hores,
    * minuts i segons que representen els milisegons rebuts com a paràmetre.
    * <p>
    * Tenint en compte que :
    *
    * 1 hora    = 3600 segons
    * 1 minut   = 60 segons
    * <p>
    * @param time long que representa el nombre de milisegons
    **/
    public TempsConnexio(long time){
        segons   = (time / 1000 ) % 60;
        minuts   = (time / 1000 ) / 60;
        hores    = minuts / 60;
        minuts   = minuts - 60 * hores;
    }

    /**
    *
    * Mètode que retorna la informació formatada en un String
    * <p>
    * {<b>Pre:</b> x és una instància de la classe TempsConnexio i hores,
    * minuts i segons són atributs de x que representen el que el seu nom indica}
    * <p>
    * {<b>Post:</b> Es retornen els valors formatats en un String :
    * hores:minuts:segons}
    *
    **/
    public String toString(){
        return(hores + ":" + minuts + ":" + segons );
    }

    /**
    *
    * Mètode main de la classe TempsConnexio
    *
    * permet passar-hi un seguit de números com a paràmetres
    * es retorna el camp TempsConnexio formatat
    **/
    public static void main(String[] args){
        for (int j = 0; j < args.length; j++) {
            System.out.println(new TempsConnexio(Long.parseLong(args[j])));
        }
    }
}
```

Comentaris de la classe TempsConnexio

La classe retorna el temps de la connexió, rebut en mil·lisegons, en un format intel·ligible i apropiat per registrar-ho en l'arxiu *log* de dades persistents.

La classe es crida des de *XatHandler* (logs de la part *servidor*) i des de *FinestraClient* (logs de la part *client*)

Classe EditTextDatabase

```
EditTextDatabase
#ICONFILE : String[] = {  "edit.gif", "delete.gif",
                          "first.gif", "previous.gif",
                          "next.gif", "last.gif",
                          "append.gif", "sortir.gif"}
#ACCEPT : String = "Modificar Registre"
#DELETE : String = "Eliminar Registre"
#FIRST : String = "Primer Registre"
#PREVIOUS : String = "Registre Anterior"
#NEXT : String = " Registre Següent"
#LAST : String = "Últim Registre"
#APPEND : String = "Afegir Registre"
#EXIT : String = "Tancar Finestra"
#buttonLabels : String[] = { ACCEPT, DELETE, FIRST, PREVIOUS, NEXT,
                             LAST, APPEND, EXIT }
#txtFields : JTextField[]
#txtLabels : JLabel []
#numFields : int
#numRec : int
#strToken : String
#fileName : String
#ndxRec : int
#toolBar : JToolBar
#icons : ImageIcon[]
#buttons : JButton []
#menuBar : JMenuBar
#menu : JMenu
#menuItem : JMenuItem
#dades : String*
+EditTextDatabase ( pVectorDades : String*, pLabels : String[],
                    pStrToken : String, pFileName : String )
+internalFrameClosing ( pEvent : InternalFrameEvent )
+internalFrameClosed ( pEvent : InternalFrameEvent )
+internalFrameOpened ( pEvent : InternalFrameEvent )
+internalFrameIconified ( pEvent : InternalFrameEvent )
+internalFrameDeIconified ( pEvent : InternalFrameEvent )
+internalFrameActivated ( pEvent : InternalFrameEvent )
+internalFrameDeActivated ( pEvent : InternalFrameEvent )
-updateData ( pIndex : int )
-getNumFields () : int
-getDadesPantalla () : String
+actionPerformed ( pActionEvent : ActionEvent)
```

Codi de la classe EditTextDatabase

```
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import java.util.Vector;
import java.util.Enumeration;
import java.util.StringTokenizer;
```

```
public class EditTextDatabase     extends JInternalFrame
                                  implements InternalFrameListener,
                                             ActionListener {

/** imatges de la barra de buttons*/
static final String[] iconFiles = {   "edit.gif",
                                      "delete.gif",
                                      "first.gif",
                                      "previous.gif",
                                      "next.gif",
                                      "last.gif",
                                      "append.gif",
                                      "sortir.gif"};

/** Atribut del botó acceptar*/
static final String ACCEPT = "Modificar Registre";

/** Atribut del botó delete*/
static final String DELETE = "Eliminar Registre";

/** Atribut del botó first*/
static final String FIRST = "Primer Registre";

/** Atribut del botó previous*/
static final String PREVIOUS = "Registre Anterior";

 /** Atribut del botó next*/
static final String NEXT = "Registre Següent";

/** Atribut del botó last*/
static final String LAST = "Últim Registre";

/** Atribut del botó append*/
static final String APPEND = "Afegir Registre";

/** Atribut del botó append*/
static final String EXIT = "Tancar la finestra";

/** Etiquetes dels buttons*/
static final String[] buttonLabels = {   ACCEPT,
                                         DELETE,
                                         FIRST,
                                         PREVIOUS,
                                         NEXT,
                                         LAST,
                                         APPEND,
                                         EXIT};

/**Array de JTextField que conté els camps a editar*/
protected JTextField[] txtFields;

/**Array de JLabels que conté les etiquetes dels camps a editar*/
protected JLabel[] txtLabels;

/**int que conté el nombre de camps a editar*/
protected int numFields;

/**int que conté el nombre d'elements del vector dels camps a editar*/
protected int numRec;

/**String que conté el token de separació dels camps*/
protected String strToken;

/**String que conté el nom de l'arxiu al disc*/
protected String fileName;

/**int que conté l'índex de l'element actual del vector*/
protected int ndxRec;

/** Barra de buttons*/
protected JToolBar toolBar;

/** Icons de la barra de buttons*/
```

```
protected ImageIcon[] icons;

/** Buttons de la barra de buttons*/
protected JButton[] buttons;

/** Menú de l'aplicació*/
protected JMenuBar menuBar;

/** Opció de la barra del menú*/
protected JMenu menu;

/** Menú ítem*/
protected JMenuItem menuItem;

/** Vector amb les dades a editar*/
protected Vector dades;

public EditTextDatabase(  Vector dades,
                          String[] aLabels,
                          String strToken,
                          String fileName) {

        // instanciem un JInternalFrame - super - amb els següents atributs
        super("Manteniment dels Hosts",
                false, //resizable
                false, //closable
                false, //maximizable
                false);//iconifiable

        //inicialitzem el vector de dades de la classe
        this.dades = dades;

        //inicialitzem el token de separació dels camps
        this.strToken = strToken;

        //inicialitzem l'atribut que conté el nom de l'arxiu al disc
        this.fileName = fileName;

        //inicialitzem el nombre de camps a editar
        numFields = getNumFields();

        //inicialitzem el array amb els noms dels camps a mostrar
        txtLabels = new JLabel[numFields];

        //inicialitzem el array amb els camps a mostrar
        txtFields = new JTextField[numFields];

        // definim la manera de tancar la finestra
        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

        //posicionem la finestra
        setLocation(100, 100);

        //instanciem un listener per la finestra
        addInternalFrameListener(this);

        //assignem el tamany de la finestra
        setSize(700,200);

        //instanciem la barra de buttons
        toolBar = new JToolBar();

        //instanciem l'array d'imatges de la barra de buttons
        icons = new ImageIcon[iconFiles.length];

        //instanciem els buttons barra de buttons
        buttons = new JButton[buttonLabels.length];

        //instanciem la barra de menú
        menuBar = new JMenuBar();

        //menú Arxiu
        menu = new JMenu("Arxiu");
```

```
menu.setMnemonic(KeyEvent.VK_A);
menuBar.add(menu);

// ítems del menú Arxiu...
menuItem = new JMenuItem("Sortir");
menuItem.setMnemonic(KeyEvent.VK_X);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
         KeyEvent.VK_X, ActionEvent.ALT_MASK));
menuItem.setActionCommand(EXIT);
menuItem.addActionListener(this);
menu.add(menuItem);

//menú Navegació
menu = new JMenu("Navegació");
menu.setMnemonic(KeyEvent.VK_N);
menuBar.add(menu);

// ítems del menú Navegació...
menuItem = new JMenuItem(ACCEPT);
menuItem.setMnemonic(KeyEvent.VK_M);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
           KeyEvent.VK_M, ActionEvent.ALT_MASK));
menuItem.setActionCommand(ACCEPT);
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem(DELETE);
menuItem.setMnemonic(KeyEvent.VK_E);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
           KeyEvent.VK_E, ActionEvent.ALT_MASK));
menuItem.setActionCommand(DELETE);
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem(FIRST);
menuItem.setMnemonic(KeyEvent.VK_P);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
           KeyEvent.VK_P, ActionEvent.ALT_MASK));
menuItem.setActionCommand(FIRST);
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem(PREVIOUS);
menuItem.setMnemonic(KeyEvent.VK_P);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
        KeyEvent.VK_P, ActionEvent.ALT_MASK));
menuItem.setActionCommand(PREVIOUS);
menuItem.addActionListener(this);
menu.add(menuItem);
setJMenuBar(menuBar);

menuItem = new JMenuItem(NEXT);
menuItem.setMnemonic(KeyEvent.VK_N);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
           KeyEvent.VK_N, ActionEvent.ALT_MASK));
menuItem.setActionCommand(NEXT);
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem(LAST);
menuItem.setMnemonic(KeyEvent.VK_U);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
           KeyEvent.VK_U, ActionEvent.ALT_MASK));
menuItem.setActionCommand(LAST);
menuItem.addActionListener(this);
menu.add(menuItem);

menuItem = new JMenuItem(APPEND);
menuItem.setMnemonic(KeyEvent.VK_N);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
           KeyEvent.VK_N, ActionEvent.ALT_MASK));
menuItem.setActionCommand(APPEND);
menuItem.addActionListener(this);
```

```
        menu.add(menuItem);

        //instanciem un layout pels camps i etiquetes...
        JPanel labelPane = new JPanel();
        labelPane.setLayout(new GridLayout(0, 1));
        JPanel fieldPane = new JPanel();
        fieldPane.setLayout(new GridLayout(0, 1));

        // omplim els arrays d'etiquetes i camps...
        for (int i = 0; i < numFields; i++) {
            // camp i ...
            txtLabels[i] = new JLabel(aLabels[i]);
            txtFields[i] = new JTextField(50);
            labelPane.add(txtLabels[i]);
            fieldPane.add(txtFields[i]);
        }

         // inicialitzem els buttons de la barra de buttons
         for (int i = 0; i < buttonLabels.length; i++) {
            icons[i] = new ImageIcon("./images/" + iconFiles[i]);
            buttons[i] = new JButton(icons[i]);
            buttons[i].setToolTipText(buttonLabels[i]);
            if (i == 2 || i == 6) toolBar.addSeparator();
            if (i == 7 ) { toolBar.addSeparator();toolBar.addSeparator();}

            toolBar.add(buttons[i]);
        }
        toolBar.setBorderPainted(true);
        toolBar.setFloatable(false);

        //buttons actions
        buttons[0].setActionCommand(ACCEPT);
        buttons[1].setActionCommand(DELETE);
        buttons[2].setActionCommand(FIRST);
        buttons[3].setActionCommand(PREVIOUS);
        buttons[4].setActionCommand(NEXT);
        buttons[5].setActionCommand(LAST);
        buttons[6].setActionCommand(APPEND);
        buttons[7].setActionCommand(EXIT);

        //buttons listener
        buttons[0].addActionListener(this);
        buttons[1].addActionListener(this);
        buttons[2].addActionListener(this);
        buttons[3].addActionListener(this);
        buttons[4].addActionListener(this);
        buttons[5].addActionListener(this);
        buttons[6].addActionListener(this);
        buttons[7].addActionListener(this);

        //col·loquem bé buttons, etiquetes i camps de texte...
        JPanel contentPane = new JPanel();
        contentPane.setBorder(BorderFactory.createEmptyBorder(0, 20, 20, 20));
        contentPane.setLayout(new BorderLayout());
        contentPane.add(toolBar, BorderLayout.NORTH);
        contentPane.add(Box.createRigidArea(new Dimension(50, 660)),BorderLayout.CENTER);

        JPanel contentFieldsPane = new JPanel();
        contentFieldsPane.add(labelPane, BorderLayout.WEST);
        contentFieldsPane.add(fieldPane, BorderLayout.EAST);

        contentPane.add(contentFieldsPane, BorderLayout.SOUTH);
        setContentPane(contentPane);

        // inicialitzo el nombre de registres i la posició actual
        numRec = dades.size();
        ndxRec = 0;

        // actualitzem les dades amb la posició inicial del vector
        updateData(ndxRec);

    }
```

```
/**
 * Listener de la interfície InternalFrameListener
 */
 public void internalFrameClosing(InternalFrameEvent e) {
}

/**
 * Listener de la interfície InternalFrameListener
 * Tanquem la finestra de connexió del client
 */
public void internalFrameClosed(InternalFrameEvent e) {
     try {
         // Demano confirmació...
         if (JOptionPane.showConfirmDialog(null, "Escriure l'arxiu de dades ?",
                                                  "Escriure canvis",
                                                  JOptionPane.YES_NO_OPTION) ==
                                                  JOptionPane.YES_OPTION) {
             try {
                 // escric al fitxer de dades
                 // el fitxer es crea de nou
                 // al final s'afegeix un salt de línia
                 PrintStream out = new PrintStream( new FileOutputStream(fileName));

                 // recorro el vector de dades...
                 Enumeration enum = dades.elements ();
                 while (enum.hasMoreElements ()) {
                     out.println(enum.nextElement ());
                 }

                 out.close();

             } catch(FileNotFoundException  fnfExc) {System.err.println(
                                                  "El fitxer log no existeix");
             }
         }

     } catch (Exception ex) {
         ex.printStackTrace();
     }
 }

/**
 * Listener de la interfície InternalFrameListener
 */
 public void internalFrameOpened(InternalFrameEvent e) {
 }

/**
 * Listener de la interfície InternalFrameListener
 */
 public void internalFrameIconified(InternalFrameEvent e) {
 }

/**
 * Listener de la interfície InternalFrameListener
 */
 public void internalFrameDeiconified(InternalFrameEvent e) {
 }

/**
 * Listener de la interfície InternalFrameListener
 */
 public void internalFrameActivated(InternalFrameEvent e) {
 }

/**
 * Listener de la interfície InternalFrameListener
 */
 public void internalFrameDeactivated(InternalFrameEvent e) {
 }

/**
 * Actualitza la pantalla amb les dades actuals del vector
```

```
 */
 private void updateData(int index) {
     StringTokenizer tokenizer;    //tokenizer

     if (numRec > 0) {
         // tokenitzo el contingut del vector
         tokenizer = new StringTokenizer(  dades.elementAt(index).toString(),
                                           strToken);
         txtFields[0].setText(tokenizer.nextToken(strToken));
         txtFields[1].setText(tokenizer.nextToken(strToken));
         txtFields[2].setText(tokenizer.nextToken(strToken));
         txtFields[3].setText(tokenizer.nextToken(strToken));

     } else {
         txtFields[0].setText("");
         txtFields[1].setText("");
         txtFields[2].setText("");
         txtFields[3].setText("");
     }

     // em posiciono sobre el primer camp
     txtFields[0].requestFocus();

 }

/**
 * Retorna el nombre de camps a editar
 */
 private int getNumFields() {
     StringTokenizer tokenizer;    //tokenizer

     // tokenitzo el contingut del vector
     tokenizer = new StringTokenizer(dades.firstElement().toString(), strToken);
     return ( tokenizer.countTokens() );
 }


/**
 * Retorna les dades de la posició actual del vector
 */
 private String getDadesPantalla() {
     String dadesPantalla = "";
     for (int i = 0; i < numFields - 1; ++i) {
         dadesPantalla += txtFields[i].getText() + strToken;
     }
     dadesPantalla += txtFields[numFields - 1].getText();
     return ( dadesPantalla );
 }

/**
 * Gestor dels events de la finestra
 */
 public void actionPerformed(ActionEvent e) {

      // s'ha pitjat el botó modificar
      if (ACCEPT.equals(e.getActionCommand())) {
         if (numRec > 0) {
             try {
                 // actualitzo la posició del vector amb les dades actuals
                 dades.setElementAt(    getDadesPantalla(), ndxRec);
                 updateData(ndxRec);

             } catch (Exception ex) {
                 ex.printStackTrace();

             }

         } else {
             // Pantalla informativa...
             JOptionPane.showMessageDialog(null, "El fitxer és buït",
                                                 "Navegació",
                                                 JOptionPane.ERROR_MESSAGE);
         }
```

```
// s'ha pitjat el botó eliminar
} else if (DELETE.equals(e.getActionCommand())) {
    if (numRec > 0) {
        try {
            // Demano confirmació...
            if (JOptionPane.showConfirmDialog(null, "Eliminar el registre ?",
                                                    "Delete",
                                                    JOptionPane.YES_NO_OPTION) ==
                                                    JOptionPane.YES_OPTION) {

                // elimino la posició actual
                dades.remove(ndxRec);

                // disminueixo el nombre de registres
                numRec--;

                // m'asseguro de que la posició és correcta...
                if (ndxRec >= numRec) ndxRec = numRec - 1;
                updateData(ndxRec);
            }

        } catch (Exception ex) {
            ex.printStackTrace();

        }
    } else {
        // Pantalla informativa...
        JOptionPane.showMessageDialog(null, "El fitxer és buït",
                                            "Navegació",
                                            JOptionPane.ERROR_MESSAGE);
    }

// s'ha pitjat el botó first
} else if (FIRST.equals(e.getActionCommand())) {
    if (numRec > 0) {
        try {
            // actualitzo la posició del vector amb les dades actuals
            dades.setElementAt(    getDadesPantalla(), ndxRec);

            // em posiciono a l'inici
            ndxRec = 0;
            updateData(ndxRec);

        } catch (Exception ex) {
            ex.printStackTrace();
        }

    } else {
        // Pantalla informativa...
        JOptionPane.showMessageDialog(null, "El fitxer és buït",
                                            "Navegació",
                                            JOptionPane.ERROR_MESSAGE);
    }

// s'ha pitjat el botó previous
} else if (PREVIOUS.equals(e.getActionCommand())) {
    if (numRec > 0) {
        try {
            // actualitzo la posició del vector amb les dades actuals
            dades.setElementAt(    getDadesPantalla(), ndxRec);

            //em posiciono...
            if (ndxRec > 0) {
                ndxRec--;
                updateData(ndxRec);

            } else {
                // Pantalla informativa...
                JOptionPane.showMessageDialog(null, "Som a l'inici del fitxer",
                                                    "Registre anterior",
                                                    JOptionPane.ERROR_MESSAGE);
```

```
                }

            } catch (Exception ex) {
                ex.printStackTrace();

            }
        } else {
            // Pantalla informativa...
            JOptionPane.showMessageDialog(null, "El fitxer és buït",
                                                "Navegació",
                                                JOptionPane.ERROR_MESSAGE);
        }

    // s'ha pitjat el botó next
    } else if (NEXT.equals(e.getActionCommand())) {
        if (numRec > 0) {
            try {
                // actualitzo la posició del vector amb les dades actuals
                dades.setElementAt(    getDadesPantalla(), ndxRec);

                //em posiciono...
                if (ndxRec < numRec - 1) {
                    ndxRec++;
                    updateData(ndxRec);

                } else {
                    // Pantalla informativa...
                    JOptionPane.showMessageDialog(null, "Som al final del fitxer",
                                                        "Registre següent",
                                                        JOptionPane.ERROR_MESSAGE);

                }
            } catch (Exception ex) {
                ex.printStackTrace();

            }

        } else {
            // Pantalla informativa...
            JOptionPane.showMessageDialog(null, "El fitxer és buït",
                                                "Navegació",
                                                JOptionPane.ERROR_MESSAGE);
        }

    // s'ha pitjat el botó last
    } else if (LAST.equals(e.getActionCommand())) {
        if (numRec > 0) {
            try {
                // actualitzo la posició del vector amb les dades actuals
                dades.setElementAt(    getDadesPantalla(), ndxRec);

                // em posiciono ...
                ndxRec = numRec - 1;
                updateData(ndxRec);

            } catch (Exception ex) {
                ex.printStackTrace();

            }

        } else {
            // Pantalla informativa...
            JOptionPane.showMessageDialog(null, "El fitxer és buït",
                                                "Navegació",
                                                JOptionPane.ERROR_MESSAGE);
        }

    // s'ha pitjat el botó append
    } else if (APPEND.equals(e.getActionCommand())) {
        try {
            // Demano confirmació...
            if (JOptionPane.showConfirmDialog(null,
                                        "Afegir el registre amb les dades actuals ?",
```

```
                                                "Afegir",
                                                JOptionPane.YES_NO_OPTION) ==
                                                JOptionPane.YES_OPTION) {

                    // elimino la posició actual
                    dades.insertElementAt(getDadesPantalla(), ndxRec);

                    // augmento el nombre de registres
                    numRec++;

                    updateData(ndxRec);
                }

            } catch (Exception ex) {
                ex.printStackTrace();

            }

        // s'ha sel·leccionat la opció del menú
        } else if (EXIT.equals(e.getActionCommand())) {
            dispose();

        // es tanca la finestra
        }  else if ((e.getSource() == this) && (e.getID()== Event.WINDOW_DESTROY)) {
                hide();
                setVisible(false);
        }
    }
}
```

Comentaris de la classe EditTextDatabase

La classe gestiona les dades persistents (altes, baixes i modificacions) d'un arxiu de text.

A l'arxiu de text les dades han d'estar separades amb el token passat com a paràmetre al constructor de la classe. Les dades del fitxer es reben en un vector (String *) que conté els registres de la base de dades separats pel token (`strToken`) rebut com a paràmetre al constructor de la classe.

La classe manté les dades del vector i al final es demana si es vol actualitzar la informació al disc.

Classe ConnexionsClient

ConnexionsClient
+ConnexionsClient (pOwner : JInternalFrame, pTitol : String, pVectorDades : String**)

Codi de la classe ConnexionsClient

```
/*
*
* Classe per mostrar les connexions d'un client a XatSegur
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004*
*/

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;

class ConnexionsClient extends JDialog {

    /**
    * Constructor de la classe ConnexionsClient.<p>
    * Crea un dialeg amb una taula per mostrar les connexions efectuades
    * anteriorment pel client amb el host actual.
```

```
    * <p>
    * @param owner JInternalFrame des del qual és cridat el dialeg.
    * @param titol String que conté el títol de la finestra.
    * @param dades Vector de vectors que conté les dades a mostrar.
    */
    public ConnexionsClient(JInternalFrame owner, String titol, Vector dades) {
        super((Frame)SwingUtilities.getAncestorOfClass(Frame.class, owner));
        setLocationRelativeTo(owner);

        // variables locals
        JTable taula;
        JScrollPane panell;

        //assigno el títol
        this.setTitle(titol);

        // vector que conté els títols de les columnes
        Vector nomCols = new Vector();
        nomCols.add("Data connexió");
        nomCols.add("Inici connexió");
        nomCols.add("Fi connexió");
        nomCols.add("Temps connexió");

        // inicialitza la taula amb el vector de vectors i el vector
        // amb els títols de les columnes
        taula = new JTable(dades, nomCols);

        // afegeixo la taula al dialeg i la mostro en pantalla
        panell = new JScrollPane(taula);
        taula.setPreferredScrollableViewportSize(new Dimension(480, 160));
        this.setContentPane(panell);
    }

}
```

Comentaris de la classe ConnexionsClient

La classe mostra en una taula l'històric de connexions efectuades a *XatSegur* pel client actual al port actual del servidor.

La classe rep com a paràmetre les dades a mostrar, en un vector de vectors (`pDades : String**`). Aquest vector ha estat omplert a la classe des d'on es truca al llistat (la classe *FinestraClient*). Aquestes dades són persistents i es troben a l'arxiu *./logs/connClient.log*

Classe ConnexionsServidor

ConnexionsServidor
+ConnexionsServidor (pOwner : JInternalFrame, pTitol : String, pDades : Object[][])

Codi de la classe ConnexionsServidor

```
/*
*
* Classe per mostrar les connexions actuals del servidor
* de XatSegur al port actual
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004*
*/
import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;

class ConnexionsServidor extends JDialog {
    // constants
    final String[] nomCols = {"Adressa connexió", "Port local", "Port remot"};
```

```
    /**
    * Constructor de la classe ConnexionsServidor.<p>
    * Crea un dialeg amb una taula per mostrar les connexions actuals
    * al port actiu.
    * <p>
    * @param owner JInternalFrame des del qual és cridat el dialeg.
    * @param titol String que conté el títol de la finestra.
    * @param dades Object[][] que conté les dades a mostrar.
    */
    public ConnexionsServidor(JInternalFrame owner, String titol, Object[][] dades) {
        super((Frame)SwingUtilities.getAncestorOfClass(Frame.class, owner));
        setLocationRelativeTo(owner);

        // variables locals
        JTable taula;
        JScrollPane panell;

        //assigno el títol
        this.setTitle(titol);

        // afegeixo la taula al dialeg i la mostro en pantalla
        taula = new JTable(dades, nomCols);
        panell = new JScrollPane(taula);
        taula.setPreferredScrollableViewportSize(new Dimension(480, 160));
        this.setContentPane(panell);
    }

}
```

Comentaris de la classe ConnexionsServidor

La classe mostra en una taula el llistat dels clients actualment connectats a *XatSegur* pel port actual del servidor.

La classe rep com a paràmetre les dades a mostrar, en un array multidimensional (`pDades : Object[][]`). Aquest array ha estat omplert a la classe des d'on es truca al llistat (la classe *FinestraServidor*). Aquestes dades són no persistents.

Classe ConnexionsHistoricServidor

ConnexionsHistoricServidor
+ConnexionsHistoricServidor (pOwner : JInternalFrame, pTitol : String, pVectorDades : String**)

Codi de la classe ConnexionsHistoricServidor

```
/*
*
* Classe per mostrar les connexions dels clients a un port del servidor XatSegur
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004*
*/

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;
import javax.swing.table.*;


class ConnexionsHistoricServidor extends JDialog {

    /**
    * Constructor de la classe ConnexionsHistoricServidor.<p>
    * Crea un dialeg amb una taula per mostrar les connexions efectuades
    * anteriorment pels clients al port actual del servidor.
    * <p>
```

```
    * @param owner JInternalFrame des del qual és cridat el dialeg.
    * @param titol String que conté el títol de la finestra.
    * @param dades Vector de vectors que conté les dades a mostrar.
    */
    public ConnexionsHistoricServidor(JInternalFrame owner, String titol, Vector dades) {
        super((Frame)SwingUtilities.getAncestorOfClass(Frame.class, owner));
        setLocationRelativeTo(owner);

        // variables locals
        JTable taula;
        JScrollPane panell;

        //assigno el títol
        this.setTitle(titol);

        // vector que conté els títols de les columnes
        Vector nomCols = new Vector();
        nomCols.add("Data connexió");
        nomCols.add("Client");
        nomCols.add("Port remot");
        nomCols.add("Inici connexió");
        nomCols.add("Fi connexió");
        nomCols.add("Temps connexió");

        // inicialitza la taula amb el vector de vectors i el vector
        // amb els títols de les columnes
        taula = new JTable(dades, nomCols);

        // afegeixo la taula al dialeg i la mostro en pantalla
        panell = new JScrollPane(taula);
        taula.setPreferredScrollableViewportSize(new Dimension(480, 160));
        this.setContentPane(panell);
    }

}
```

Comentaris de la classe ConnexionsHistoricServidor

La classe mostra en una taula l'històric de connexions efectuades a *XatSegur* pels clients anteriors al port actual del servidor.

La classe rep com a paràmetre les dades a mostrar, en un vector de vectors (`pDades : String**`). Aquest vector ha estat omplert a la classe des d'on es truca al llistat (la classe *FinestraServidor*). Aquestes dades són persistents i es troben a l'arxiu *./logs/#####.log*, on ##### representa el número de port del servidor.

Classe GestioServidors

GestioServidors #desktop : JDeskTopPane
+GestioServidors() #createMenuBar() : JMenuBar +actionPerformed(pActionEvent : ActionEvent) #creaServidor() #quit() #quit() -createAndShowGui(pLookAndFeel : String) +main(pArgs : STring[])

Codi de la classe GestioServidors

```
/*
*
* Classe que gestiona la connexió de servidors de XatSegur
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*
```

```
*/
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.util.Vector;
import com.sun.java.swing.plaf.windows.WindowsLookAndFeel;
import com.sun.java.swing.plaf.motif.MotifLookAndFeel;
import javax.swing.plaf.metal.MetalLookAndFeel;

/**
*
* Aquesta classe gestiona la connexió de servidors de XatSegur.
* <p>
* @author Jordi Coll Vera.
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
* @extends JFrame
* @implements ActionListener
*
**/
public class GestioServidors extends JFrame
                             implements ActionListener {

    /** Atribut que representa el desktop*/
    JDesktopPane desktop;


    /**
    * Constructor de la classe GestioServidors.
    **/
    public GestioServidors() {

        // instanciem un JFrame - super - amb el següent títol
        super("Servidor de XatSegur - TFC Jordi Coll Vera");

        // la finestra principal és indentada 50 pixels des de cada cantonada
        // de la pantalla.
        int inset = 50;

        SwingUtilities.updateComponentTreeUI(this);
        this.pack();

        // tamany de la finestra en funció de la ressolució de la pantalla
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        setBounds(inset, inset,
                  screenSize.width  - inset*2,
                  screenSize.height - inset*2);


        // genero el desktop
        desktop = new JDesktopPane();
        setContentPane(desktop);

        // afegim el menú
        setJMenuBar(createMenuBar());

        // dragmode del desktop
        desktop.setDragMode(JDesktopPane.OUTLINE_DRAG_MODE);


    }


    /**
     * crea els menús de la finestra
     */
    protected JMenuBar createMenuBar() {
        JMenuBar menuBar = new JMenuBar();

        // menú principal
        JMenu menu = new JMenu("Gestió Servidors");
        menu.setMnemonic(KeyEvent.VK_G);
        menuBar.add(menu);
```

```
        // item del menu
        JMenuItem menuItem = new JMenuItem("Nou canal");
        menuItem.setMnemonic(KeyEvent.VK_N);
        menuItem.setAccelerator(KeyStroke.getKeyStroke(
                KeyEvent.VK_N, ActionEvent.ALT_MASK));
        menuItem.setActionCommand("nou");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        // item del menu
        menuItem = new JMenuItem("Sortir");
        menuItem.setMnemonic(KeyEvent.VK_S);
        menuItem.setAccelerator(KeyStroke.getKeyStroke(
                KeyEvent.VK_S, ActionEvent.ALT_MASK));
        menuItem.setActionCommand("sortir");
        menuItem.addActionListener(this);
        menu.add(menuItem);

        // retorno el menú
        return menuBar;
    }

    // accions corresponents als itmes del menú
    public void actionPerformed(ActionEvent e) {
        if ("nou".equals(e.getActionCommand())) { // nou canal
            creaServidor();
        } else { //sortir
            quit();
        }
    }

    // crea un nou servidor i la seva finestra.
    protected void creaServidor() {
        String port = JOptionPane.showInputDialog(desktop, "Obrir el port número : ","Socket
Port",JOptionPane.QUESTION_MESSAGE);

        // verifico el port solicitat
        if ( port != null) {

            try {
                int nPort = (new Integer(port)).intValue();

                if (nPort > 0 & nPort < 65565) {

                    // si el port és correcte...
                    try {

                        // obro una nova finestra de servidor
                        FinestraServidor frame = new FinestraServidor(nPort);
                        frame.setVisible(true);
                        desktop.add(frame);
                        frame.setSelected(true);

                    } catch (java.beans.PropertyVetoException e) {}

                } else JOptionPane.showMessageDialog(null, "Número de port
incorrecte","Error",JOptionPane.ERROR_MESSAGE);

            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null, e.toString(),"Error
creaServidor",JOptionPane.ERROR_MESSAGE);
            }
        }
    }

    // abandona l'aplicació.
    protected void quit() {
        System.exit(0);
    }

    /**
     * crea el GUI i el mostra.
     */
```

```
    private static void createAndShowGUI(String lf) {

        try {
           if (lf.equals("1")) {
               // L&F windows
               UIManager.setLookAndFeel(new WindowsLookAndFeel());

           } else if (lf.equals("2")) {
               // L&F motif
               UIManager.setLookAndFeel(new MotifLookAndFeel());

           } else if (lf.equals("3")) {
               // L&F motif
               UIManager.setLookAndFeel(new MetalLookAndFeel());

           } else {
               UIManager.getSystemLookAndFeelClassName();

           }

        } catch(UnsupportedLookAndFeelException ulf) {
           // decoració per defecte
           JFrame.setDefaultLookAndFeelDecorated(true);
        }

        // crea la finestra i la configura
        GestioServidors frame = new GestioServidors();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // mostra la finestra
        frame.setVisible(true);
    }

    public static void main(final String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                if (args.length == 0) {
                   createAndShowGUI("0");

                } else {
                    // L&F sel·leccionat
                    createAndShowGUI(args[0]);
                }

            }
        });
    }
}
```

Comentaris de la classe GestioServidors

La classe crea la pantalla principal de la part servidor. Des de ella es crea l'aplicació MDI que gestiona la creació de finestres de servidor a ports concrets. Cada cop que es sol·licita la creació d'una nova finestra de servidor és genera una instància de *FinestraServidor* (de la classe *swing JInternalFrame*) al port seleccionat.

La classe rep com a paràmetre un número que indica el *Look&Feel* de l'aplicació.

Classe GestioClients

GestioClients #desktop : JDeskTopPane
+GestioServidors() #createMenuBar() : JMenuBar +actionPerformed(pActionEvent : ActionEvent) #creaClient() #editaHosts() #quit() -createAndShowGui(pLookAndFeel : String) +main(pArgs : STring[])

Codi de la classe GestioClients

```
/*
*
* Classe que gestiona la connexió de clients de XatSegur
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*
*/

import java.io.*;
import java.util.*;
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;
import java.util.Vector;
import java.util.StringTokenizer;
import com.sun.java.swing.plaf.windows.WindowsLookAndFeel;
import com.sun.java.swing.plaf.motif.MotifLookAndFeel;
import javax.swing.plaf.metal.MetalLookAndFeel;

/**
*
* Aquesta classe gestiona la connexió de clients de XatSegur.
* <p>
* @author Jordi Coll Vera.
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
* @extends JFrame
* @implements ActionListener
*
**/
public class GestioClients extends JFrame implements ActionListener {

    /** Atribut que representa el desktop*/
    JDesktopPane desktop;

    /**
    * Constructor de la classe GestioClients.
    **/
    public GestioClients() {

        // instanciem un JFrame - super - amb el següent títol
        super("Client de XatSegur - TFC Jordi Coll Vera");

        SwingUtilities.updateComponentTreeUI(this);
        this.pack();

        // la finestra principal és indentada 50 pixels des de cada cantonada
        // de la pantalla.
        int inset = 50;
        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        setBounds(inset, inset,
                  screenSize.width  - inset*2,
                  screenSize.height - inset*2);
```

```
        // genero el desktop
        desktop = new JDesktopPane(); //a specialized layered pane
        setContentPane(desktop);

        // afegim el menú
        setJMenuBar(createMenuBar());

        // drag mode del desktop
        desktop.setDragMode(JDesktopPane.OUTLINE_DRAG_MODE);
    }

    /**
         * crea els menús de la finestra
    */
    protected JMenuBar createMenuBar() {
        JMenuBar menuBar = new JMenuBar();

        // menú principal 1
        JMenu menu1 = new JMenu("Gestió Clients");
        menu1.setMnemonic(KeyEvent.VK_G);
        menuBar.add(menu1);

        // item del menu
        JMenuItem menuItem = new JMenuItem("Nova connexió");
        menuItem.setMnemonic(KeyEvent.VK_N);
        menuItem.setAccelerator(KeyStroke.getKeyStroke(
                KeyEvent.VK_N, ActionEvent.ALT_MASK));
        menuItem.setActionCommand("nou");
        menuItem.addActionListener(this);
        menu1.add(menuItem);

        // item del menu
        menuItem = new JMenuItem("Sortir");
        menuItem.setMnemonic(KeyEvent.VK_S);
        menuItem.setAccelerator(KeyStroke.getKeyStroke(
                KeyEvent.VK_S, ActionEvent.ALT_MASK));
        menuItem.setActionCommand("sortir");
        menuItem.addActionListener(this);
        menu1.add(menuItem);

        // menú principal 2
        JMenu menu2 = new JMenu("Gestió de Hosts");
        menu2.setMnemonic(KeyEvent.VK_C);
        menuBar.add(menu2);

        // item del menu
        menuItem = new JMenuItem("Afegir/Modificar/Eliminar Hosts");
        menuItem.setMnemonic(KeyEvent.VK_A);
        menuItem.setAccelerator(KeyStroke.getKeyStroke(
                KeyEvent.VK_A, ActionEvent.ALT_MASK));
        menuItem.setActionCommand("edit");
        menuItem.addActionListener(this);
        menu2.add(menuItem);

        // retorno el menú
        return menuBar;
    }

    // accions corresponents als items del menú
    public void actionPerformed(ActionEvent e) {
        if ("nou".equals(e.getActionCommand())) { //nova connexió
            creaClient();

        } else if ("edit".equals(e.getActionCommand())) { //editar hosts
           editaHosts();

        } else { //sortir
            quit();
        }
    }

    //Crea un nou client i la seva finestra.
```

```
    protected void creaClient() {
        // variables
        Vector possibilities = new Vector();                    //vector de connexions
        ImageIcon icon = new ImageIcon("./images/conversa.gif");//imatge del diàleg
        BufferedReader in = null;                               //buffer de lectura del
                                                                //fitxer log
        StringTokenizer tokenizer;                              //tokenizer
        String line;                                            //línia llegida
        String data = "";                                       //data de connexió
        String host = "";                                       //host de connexió (adreça
                                                                //IP)
        String port = "";                                       //port remot de la connexió
        String nick = "";                                       //nick name de la connexió

        try {

            // obro el fitxer de connexions
            in = new BufferedReader( new FileReader("./conf/connexions.dat"));

        } catch(FileNotFoundException  fnfExc) {System.err.println("No trobo
./conf/connexions.dat");}

        try {
            // llegeixo el fitxer dat mentre hi hagi dades
            while ((line = in.readLine()) != null) {

                if (line != null) {
                    // tokenitzo la línia
                    tokenizer = new StringTokenizer(line, "|");

                    // omplo les variables
                    data = tokenizer.nextToken("|");
                    host = tokenizer.nextToken("|");
                    port = tokenizer.nextToken("|");
                    nick = tokenizer.nextToken("|");

                    possibilities.add(data + ":" + host + ":" + port + ":" + nick);
                }
            }

            in.close();

        } catch(IOException IOExc) {System.err.println("Error llegint o tancant el fitxer
./conf/connexions.dat");}


        String connexio = (String)JOptionPane.showInputDialog(desktop,
                                              "Connectar amb :\n","Nova connexió",
                                              JOptionPane.INFORMATION_MESSAGE,
                                              icon,
                                              possibilities.toArray(),
                                              "");

        host = null;
        port = null;
        if (connexio != null) {
            StringTokenizer st = new StringTokenizer(connexio, ":");
            data = st.nextToken(":");
            host = st.nextToken(":");
            port = st.nextToken(":");
            nick = st.nextToken(":");
        }

// Podem ser més informatius...
// JOptionPane.showMessageDialog(null, "host = " + host + "\n port = " + port,"Connexió
client",JOptionPane.INFORMATION_MESSAGE);

        // verifico el host:port solicitat
        if ( host != null & port != null) {
            try {
                int nPort = (new Integer(port)).intValue();

                if (nPort > 0 & nPort < 65565) {
```

```
                    // si tot és correcte...
                    try {

                        // obro una nova finestra de client
                        FinestraClient frame = new FinestraClient(host, nPort, nick);
                        frame.setVisible(true);
                        desktop.add(frame);
                        frame.setSelected(true);

                    } catch (java.beans.PropertyVetoException e) {}

                } else JOptionPane.showMessageDialog( null,
                                                      "Número de port incorrecte",
                                                      "Error",
                                                      JOptionPane.ERROR_MESSAGE);

            } catch (NumberFormatException e) {
                JOptionPane.showMessageDialog(null, e.toString(),"Error connecta
client",JOptionPane.ERROR_MESSAGE);
            }
        }
    }

     //Manteniment dels hosts en una nova finestra.
    protected void editaHosts() {
        // variables
        Vector dades = new Vector();          //vector amb les dades del fitxer
        String[] txtLabels = { "Connexió : ",
                               "Host : ",
                               "Port : ",
                               "Nick : "};    //array amb els noms dels camps a mostrar
        BufferedReader in = null;             //buffer de lectura del fitxer dat
        String line;                          //línia llegida


        try {

            // obro el fitxer de connexions
            in = new BufferedReader( new FileReader("./conf/connexions.dat"));

        } catch(FileNotFoundException  fnfExc) {System.err.println("No trobo
./conf/connexions.dat");}

        try {
            // llegeixo el fitxer dat mentre hi hagi dades
            while ((line = in.readLine()) != null) {

                if (line != null) {
                    // afegeixo la línia al vector de dades
                    dades.add(line);
                }
            }

            in.close();

        } catch(IOException IOExc) {System.err.println("Error llegint o tancant el fitxer
./conf/connexions.dat");}

        try {
            // obro una finestra de manteniment dels hosts
            EditTextDatabase frame = new EditTextDatabase(dades, txtLabels, "|",
"./conf/connexions.dat");
            frame.setVisible(true);
            desktop.add(frame);
            frame.setSelected(true);

        } catch (java.beans.PropertyVetoException e) {}
    }

    // abandona l'aplicació.
    protected void quit() {
        System.exit(0);
```

```
    }

    /**
     * crea el GUI i el mostra.
     */
    private static void createAndShowGUI(String lf) {

         try {
           if (lf.equals("1")) {
               // L&F windows
               UIManager.setLookAndFeel(new WindowsLookAndFeel());

           } else if (lf.equals("2")) {
               // L&F motif
               UIManager.setLookAndFeel(new MotifLookAndFeel());

           } else if (lf.equals("3")) {
               // L&F motif
               UIManager.setLookAndFeel(new MetalLookAndFeel());

           } else {
               UIManager.getSystemLookAndFeelClassName();

           }

         } catch(UnsupportedLookAndFeelException ulf) {
           // decoració per defecte
           JFrame.setDefaultLookAndFeelDecorated(true);
        }

         // crea la finestra i la configura
        GestioClients frame = new GestioClients();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // mostra la finestra
        frame.setVisible(true);
    }

    public static void main(final String[] args) {
        javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                if (args.length == 0) {
                   createAndShowGUI("0");

                } else {
                    // L&F sel·leccionat
                    createAndShowGUI(args[0]);
                }
            }
        });
    }
}
```

Comentaris de la classe GestioClients

La classe crea la pantalla principal de la part client. Des de ella es crea l'aplicació MDI que gestiona la creació de finestres de client a un host:port concrets. Cada cop que es sol·licita la creació d'una nova finestra de client és genera una instància de *FinestraClient* (de la classe *swing JInternalFrame*) al port i host seleccionats.

Des d'aquí es gestiona també la base de dades persistent ./conf/connexions.dat que conté la informació per efectuar la connexió al servidor.

La classe rep com a paràmetre un número que indica el *Look&Feel* de l'aplicació.

Classe FinestraServidor

FinestraServidor
#openFrameCount : int = 0 #XOFFSET : int = 20 #YOFFSET : int = 20 #handlers : XatHandler* #port : int #nouServidor : XatServer
+FinestraServidor(pPort : int) -creaMenuBar() : JMenuBar -sortirListener { actionPerformed(pActionEvent : ActionEvent) } +getPort() : int +internalFrameClosing (pEvent : InternalFrameEvent) +internalFrameClosed (pEvent : InternalFrameEvent) +internalFrameOpened (pEvent : InternalFrameEvent) +internalFrameIconified (pEvent : InternalFrameEvent) +internalFrameDeIconified (pEvent : InternalFrameEvent) +internalFrameActivated (pEvent : InternalFrameEvent) +internalFrameDeActivated (pEvent : InternalFrameEvent) +actionPerformed(pActionEvent : ActionEvent) -connexionsActualsListener{actionPerformed(pActionEvent : ActionEvent)} -connexionsHistoricListener{actionPerformed(pActionEvent : ActionEvent)}

Codi de la classe FinestraServidor

```
/*
*
* Classe que gestiona les connexions a ports servidor de XatSegur
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*
*/
import java.net.*;
import java.io.*;
import java.util.*;
import java.awt.event.*;
import java.awt.*;
import javax.swing.JInternalFrame;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.event.*;
import javax.swing.JOptionPane;
import javax.swing.WindowConstants;
import java.util.Vector;
import java.util.Enumeration;

/**
*
* Aquesta classe gestiona la connexió d'un servidor de XatSegur a un port.
* <p>
* cridat per GestioServidors
* @author Jordi Coll Vera.
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
* @extends JInternalFrame
* @implements InternalFrameListener, ActionListener
*
**/
public class FinestraServidor
               extends JInternalFrame
               implements InternalFrameListener, ActionListener {

    /** Atribut que compta els frames oberts per simular les finestres en cascade*/
```

```
    static int openFrameCount = 0;

    /** Atributs per simular les finestres en cascade*/
    static final int xOffset = 20, yOffset = 20;

    /** vector que conté les connexions a clients d'aquest servidor */
    protected Vector handlers = new Vector ();

    /** atribut que conté el port local de la connexió */
    protected int port;

    // gestor multi Thread de les connexions del servidor a clients*/
    protected XatServer nouServidor = null;

    /**
    * Constructor de la classe FinestraServidor.
    * <p>
    * @param port int que representa el port a gestionar.
    **/
    public FinestraServidor(int port)  {

        // instanciem un JInternalFrame - super - amb els següents atributs
        super("SERVIDOR DE XAT-SEGUR AL PORT : " + (new Integer(port)).toString(),
              true, //resizable
              true, //closable
              true, //maximizable
              true);//iconifiable

        //incrementem el comptador de finestres obertes
        openFrameCount += 1;

        // creem la barra de menu de la que penjaran tots els components del mateix
        JMenuBar menuBar = creaMenuBar();

        // definim la manera de tancar la finestra
        this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

        // instanciem el port local de la connexió
        this.port = port;

        //instanciem un listener per la finestra
        addInternalFrameListener(this);

        //assignem el tamany de la finestra
        setSize(500,300);

        //posicionem la finestra
        setLocation(xOffset*openFrameCount, yOffset*openFrameCount);

        // instanciem un nou gestor de connexió al port solicitat
        try {
            nouServidor = new XatServer(port, handlers);
            nouServidor.start();

        } catch(Exception e) {
                    JOptionPane.showMessageDialog(null, e.toString(),"Error
creaServidor",JOptionPane.ERROR_MESSAGE);
        }
    }

    /**
     * crea els menús de la finestra
     */
    private JMenuBar creaMenuBar() {

        // creem un objecte barra de menu
        JMenuBar menuBar = new JMenuBar();

        // creem el primer objecte menu, que serà arxiu
        JMenu menu = new JMenu("Arxiu");

        // creem el component del menu Arxiu i li associem un event
        JMenuItem menuItem = new JMenuItem("Clients actuals");
```

```
        menu.add(menuItem);
        menuItem.addActionListener(new ConnexionsActualsListener());

        // creem el component del menu Arxiu i li associem un event
        menuItem = new JMenuItem("Històric de connexions");
        menu.add(menuItem);
        menuItem.addActionListener(new ConnexionsHistoricListener());

        // creem el component del menu Arxiu i li associem un event
        menu.addSeparator();
        menuItem = new JMenuItem("Sortir");
        menu.add(menuItem);
        menuItem.addActionListener(new SortirListener());

        // assigno la barra de menú a la finestra
        setJMenuBar(menuBar);

        //afegeixo les opcions a la barra de menú
        menuBar.add(menu);

        // creem un altre menú principal
        menu = new JMenu("Gestió");

        // creem el component del menu Arxiu i li associem un event
        menuItem = new JMenuItem("Kick client");
        menu.add(menuItem);
        menuItem.addActionListener(new KickClientListener());

        //afegeixo les opcions a la barra de menú
        menuBar.add(menu);

        //retorno el menú
        return menuBar;
    }

    /**
     * Listener pel menú opció sortir
     */
    private class SortirListener implements ActionListener {
        public void actionPerformed(ActionEvent event) {

            // tanco tot i crido de manera automàtica a internalFrameClosed
            dispose();
        }
    }

    /**
     *
     * Accessor de lectura del port local
     * <p>
     * {<b>Pre:</b> x és una instància de la classe FinestraServidor i port
     * l'atribut que representa el port local del servidor de x}
     * <p>
     * {<b>Post:</b> Es retorna port de x}
     **/
    public int getPort() {
        return port;
    }

    /**
     * Listener de la interfície InternalFrameListener
     */
    public void internalFrameClosing(InternalFrameEvent e) {
    }

    /**
     * Listener de la interfície InternalFrameListener
     */
    public void internalFrameClosed(InternalFrameEvent e) {
        synchronized (handlers) {

            // recorro el vector de les connexions i vaig tancant els sockets
            // envio un missatge de broadcast als clients avisant de que tanco la
```

```
            // connexió
            Enumeration en = handlers.elements ();
            String message = "El servidor s'està tancant...";
            while (en.hasMoreElements ()) {
                XatHandler c = (XatHandler) en.nextElement ();
                try {
                    synchronized (c.cos) {
                        while (((message.length()) % 512) != 0) message = message.concat("
");
                        c.getCipherOutputStream().write(message.getBytes());
                        c.getCipherOutputStream().flush();
                    }

                    c.getSocket().close();

                } catch (Exception ex) {
                    c = null;

                }
            }

            try {

                // tanco el gestor de la connexió
                if (nouServidor != null) nouServidor.close();
                nouServidor = null;

            } catch (Exception exf) {
                JOptionPane.showMessageDialog(null, exf.toString(),"Error tancant
servidor",JOptionPane.ERROR_MESSAGE);

            }

            // podriem ser més educats...
            //JOptionPane.showMessageDialog(null, "Tanco el canal de xat","Tanca
canal",JOptionPane.ERROR_MESSAGE);

        }
    }

   /**
    * Listener de la interfície InternalFrameListener
    */
    public void internalFrameOpened(InternalFrameEvent e) {
    }

   /**
    * Listener de la interfície InternalFrameListener
    */
    public void internalFrameIconified(InternalFrameEvent e) {
    }

   /**
    * Listener de la interfície InternalFrameListener
    */
    public void internalFrameDeiconified(InternalFrameEvent e) {
    }

   /**
    * Listener de la interfície InternalFrameListener
    */
    public void internalFrameActivated(InternalFrameEvent e) {
    }

   /**
    * Listener de la interfície InternalFrameListener
    */
    public void internalFrameDeactivated(InternalFrameEvent e) {
    }

   /**
    * Mètode de la interfície InternalFrameListener
    */
```

```
    public void actionPerformed(ActionEvent e) {
    }

    /**
     * Listener per a mostrar les connexions actuals
     */
    private class ConnexionsActualsListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {

            // variables
            int i = 0;
            Object[][] dades = new Object[handlers.size()][3];
            try {
                synchronized (handlers) {

                    // recorro el vector de les connexions i recullo la informació pertinent
                    Enumeration en = handlers.elements ();
                    while (en.hasMoreElements ()) {
                        XatHandler c = (XatHandler) en.nextElement ();
                        try {
                            dades[i][0] = new String (c.s.getInetAddress ().toString ());
                            dades[i][1] = new String (
                                              (new Integer(c.s.getLocalPort())).toString ());
                            dades[i][2] = new String (
                                              (new Integer(c.s.getPort())).toString ());
                            i++;

                        } catch (Exception ex) {
                            c = null;
                        }
                    }
                }

                // obro la finestra de diàleg amb les dades recollides
                ConnexionsServidor finestra = new ConnexionsServidor(FinestraServidor.this,
                                                  "Clients actuals : ", dades);
                finestra.pack();
                finestra.setVisible(true);

            } catch (Exception er) {
                System.out.println(er);
            }
        }
    }

    /**
    * Listener per expulsar un client
    */
    private class KickClientListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {

            // variables
            Vector possibilities = new Vector();                //vector de clients actuals
            ImageIcon icon = new ImageIcon("./images/kick_conversa.gif");
                                                                //imatge del diàleg
            try {
                synchronized (handlers) {

                    // recorro el vector de les connexions i recullo la informació pertinent
                    Enumeration en = handlers.elements ();
                    while (en.hasMoreElements ()) {
                        XatHandler c = (XatHandler) en.nextElement ();
                        try {
                            possibilities.add(c.nick + " (" +
                                        new String (c.s.getInetAddress ().toString ()) + ")");

                        } catch (Exception ex) {
                            c = null;
                        }
                    }
                }

                if (possibilities.size() > 0 ) {
```

```
                // obro la finestra de diàleg amb les dades recollides
                String kick = (String)JOptionPane.showInputDialog(null,"Faig fora a...
                                                :\n","kick client",
                                                JOptionPane.INFORMATION_MESSAGE,
                                                icon,
                                                possibilities.toArray(),
                                            "");
                if (kick != null) {
                    System.out.println("Faig fora al client: " + kick );
                    ((XatHandler) handlers.elementAt(possibilities.indexOf((String) kick,
                              0))).s.close();
                }
            }

        } catch (Exception er) {
            System.out.println(er);
        }
    }
}

/**
 * Listener per a mostrar les connexions anteriors del client al servidor
 */
private class ConnexionsHistoricListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {

        // variables
        BufferedReader in = null;     //buffer de lectura del fitxer log
        StringTokenizer tokenizer;    //tokenizer
        String line;                  //línia llegida
        String data = "";             //data de connexió
        String host = "";             //host de connexió (adreça IP)
        String port = "";             //port remot de la connexió
        String inici = "";            //H:M:S inici de la connexió
        String fi = "";               //H:M:S fi de la connexió
        String temps = "";            //H:M:S temps total de connexió
        Vector rowData = new Vector();//vector de vectors amb les dades a mostrar

        try {

            // obro el fitxer de log
            in = new BufferedReader( new FileReader("./logs/" + (new
Integer(getPort())).toString() + ".log"));

        } catch(FileNotFoundException  fnfExc) {System.err.println("El fitxer log no
existeix");}

        try {
            // llegeixo el fitxer log mentre hi hagi dades
            while ((line = in.readLine()) != null) {

                if (line != null) {
                    // tokenitzo la línia
                    tokenizer = new StringTokenizer(line, "|");

                    // omplo les variables
                    data = tokenizer.nextToken("|");
                    host = tokenizer.nextToken("|");
                    port = tokenizer.nextToken("|");
                    inici = tokenizer.nextToken("|");
                    fi = tokenizer.nextToken("|");
                    temps = tokenizer.nextToken("|");

                    // afegeixo les dades a un vector
                    Vector vector = new Vector();
                    vector.add(data);
                    vector.add(host);
                    vector.add(port);
                    vector.add(inici);
                    vector.add(fi);
                    vector.add(temps);

                    //afegeixo el vector amb les dades al vector de vectors
```

```
                    rowData.add(vector);

                }
            }

            // tanco el fitxer log
            in.close();

        } catch(IOException IOExc) {System.err.println("Error llegint o tancant el
fitxer log");}


        // obro la finestra de diàleg amb les dades recollides
        ConnexionsHistoricServidor finestra = new
ConnexionsHistoricServidor(FinestraServidor.this, "Històric de connexions : ", rowData);
        finestra.pack();
        finestra.setVisible(true);
    }
  }
}
```

Comentaris de la classe FinestraServidor

La classe gestiona la connexió dels clients a un port concret del servidor a partir de la classe *GestioServidors*.

Quan es crea la finestra s'instancia un objecte de la classe *XatServer* (`nouServidor`) que gestiona els fils de connexió als clients que es connectin en el futur a través d'instàncies successives de la classe *XatHandler*.

Des d'aquesta classe es gestionen els llistats de *ConnexionsHistoricServidor* i *ConnexionsServidor*.

També es controla la connexió d'un client en concret, podent fer-lo fora (*kick*).

Quan es tanca la connexió (al tancar la finestra) s'envia un missatge de broadcast a tots els clients actualment connectats avisant de que la connexió es tanca.

Classe FinestraClient

FinestraClient
#openFrameCount : int = 0 #XOFFSET : int = 20 #YOFFSET : int = 20 #SHOW : String = “Show” #sortir : JMenuItem #connexionsAnteriors : JMenuItem #display : JTextArea #input : JTextArea #s : Socket #listener : Thread #keyServer : Key #cipherIn : Cipher #cipherOut : Cipher #oosOut : ObjectOutputStream #oosIn : ObjectInputStream #cis : CipherInputStream #cos : CipherOutputStream #DHKey : DHKeyAgreement #dataConnexio: Date #host : String #nick : String
+FinestraClient(pHost : String, pPort : int , pNick : String) -creaMenuBar() : JMenuBar -sortirListener { actionPerformed(pActionEvent : ActionEvent) } -connexionsAnteriorsListener{actionPerformed(pActionEvent : ActionEvent)} +internalFrameClosing (pEvent : InternalFrameEvent) +internalFrameClosed (pEvent : InternalFrameEvent) +internalFrameOpened (pEvent : InternalFrameEvent) +internalFrameIconified (pEvent : InternalFrameEvent) +internalFrameDeIconified (pEvent : InternalFrameEvent) +internalFrameActivated (pEvent : InternalFrameEvent) +internalFrameDeActivated (pEvent : InternalFrameEvent) +actionPerformed(pActionEvent : ActionEvent) -getHost : String +run()

Codi de la classe FinestraClient

```
/*
*
* Classe que gestiona la connexió d'un client de XatSegur
* @author Jordi Coll Vera
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
*
*/
import java.net.*;
import java.io.*;
import java.awt.*;
import javax.swing.*;
import javax.swing.event.*;
import java.awt.event.*;
import java.util.*;
import java.lang.String;
import java.security.Security;
import iaik.security.provider.IAIK;
import java.security.Key;
import java.security.PublicKey;
```

```
import javax.crypto.Cipher;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import java.text.DateFormat;

/**
*
* Aquesta classe gestiona la connexió d'un client de XatSegur.
* <p>
* cridat per GestioClients
* @author Jordi Coll Vera.
* @version TFC XatSegur, Consultor: Antoni Martínez Ballesté, Primer semestre 2004
* @extends JInternalFrame
* @implements Runnable, InternalFrameListener, ActionListener
*
**/
public class   FinestraClient
               extends JInternalFrame
               implements Runnable, InternalFrameListener, ActionListener {

    /** Atribut que compta els frames oberts per simular les finestres en cascade*/
    static int openFrameCount = 0;

    /** Atributs per simular les finestres en cascade*/
    static final int xOffset = 20, yOffset = 20;

    /** Atribut del botó enviar*/
    static final String SHOW = "show";

    /** Atribut opció del menú*/
    protected JMenuItem sortir;

    /** Atribut opció del menú*/
    protected JMenuItem connexionsAnteriors;

    /** Atribut que representa l'area on s'escriu el missatge rebut*/
    protected JTextArea display;

    /** Atribut que representa l'area on s'escriu el missatge a enviar*/
    protected JTextArea input;

    /** Atribut que representa al socket de la connexió del client*/
    protected Socket s;

    /** Atribut que representa el Thread de la finestra (connexió del client)*/
    protected Thread listener;

    /** Atribut que representa la clau pública del servidor*/
    protected Key keyServer;

    /** Objecte de la classe Cipher (dades entrada) a instanciar amb l'algorisme DES*/
    protected Cipher cipherIn;

    /** Objecte de la classe Cipher (dades sortida) a instanciar amb l'algorisme DES*/
    protected Cipher cipherOut;

    /** Atribut que representa el flux de dades (tipus objecte) de sortida*/
    protected ObjectOutputStream oosOut;

    /** Atribut que representa el flux de dades (tipus objecte) d'entrada*/
    protected ObjectInputStream oosIn;

    /** Atribut que representa el flux de dades (xifrades) d'entrada*/
    protected CipherInputStream cis;

    /** Atribut que representa el flux de dades (xifrades) de sortida*/
    protected CipherOutputStream cos;

    /** Atribut que representa un objecte de la classe DHKeyAgreement*/
    protected DHKeyAgreement DHKey = new DHKeyAgreement();

    /** Atribut que representa la data d'inici de la connexió*/
    protected Date dataConnexio;
```

```
/** Atribut que representa al host de la connexió*/
protected String host;

/** Atribut que representa al nick name del client*/
protected String nick;

/**
* Constructor de la classe FinestraClient.
* <p>
* @param host string que representa el host de la connexió.
* @param port int que representa el número de port de la connexió.
**/
public FinestraClient (String host, int port, String nick) {

    // instanciem un JInternalFrame - super - amb els següents atributs
    super("SERVIDOR DE XAT-SEGUR - CONNEXIÓ : " +
           host +
           ":" +
           (new Integer(port)).toString(),
           true, //resizable
           true, //closable
           true, //maximizable
           true);//iconifiable

    //incrementem el comptador de finestres obertes
    openFrameCount += 1;

    // instanciem el nick name
    this.nick = nick;

    //instanciem un socket per la connexió
    try {
        this.host = host;
        s = new Socket (InetAddress.getByName(host), port);

    } catch(Exception e) {
              JOptionPane.showMessageDialog(null,
                                            e.toString(),
                                            "Error socket client",
                                            JOptionPane.ERROR_MESSAGE);
    }

    keyServer = null;

    // creem la barra de menu de la que penjaran tots els components del mateix
    JMenuBar menuBar = creaMenuBar();

    // definim la manera de tancar la finestra
    this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

    //assignem el tamany de la finestra
    setSize(500,300);

    //posicionem la finestra
    setLocation(xOffset*openFrameCount, yOffset*openFrameCount);


    //àrea dels missatges rebuts
    display = new JTextArea(3, 30);
    display.setEditable(false);

    //area dels missatges a enviar
    input = new JTextArea(1, 30);
    input.setEditable(true);

    //afegeixo l'àrea de display a un JScrollPane
    JScrollPane textScroller = new JScrollPane(display);
    textScroller.setPreferredSize(new Dimension(200, 75));
    textScroller.setMinimumSize(new Dimension(10, 10));

    //afegeixo l'àrea d'escriptura a un JScrollPane
    JScrollPane  inputScroller = new JScrollPane(input);
```

```
        inputScroller.setPreferredSize(new Dimension(1, 75));
        inputScroller.setMinimumSize(new Dimension(10, 10));

        //genero el botó d'enviar
        JButton b1 = new JButton("Enviar");
        b1.setActionCommand(SHOW);
        b1.addActionListener(this);
        b1.setAlignmentX(CENTER_ALIGNMENT);

        //creo el panell pel contingut de la finestra i afegeixo els controls
        JPanel contentPane = new JPanel();
        contentPane.setBorder(BorderFactory.createEmptyBorder(10,10,10,10));
        contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.PAGE_AXIS));
        contentPane.add(Box.createRigidArea(new Dimension(0, 5)));
        contentPane.add(textScroller);
        contentPane.add(Box.createRigidArea(new Dimension(0, 5)));
        contentPane.add(inputScroller);
        contentPane.add(Box.createRigidArea(new Dimension(0, 5)));
        contentPane.add(b1);

        //afegeixo a la finestra el panell creat
        setContentPane(contentPane);

        //col·loco el cursor a l'area d'escriptura de missatges
        input.requestFocus ();

        //listener de la finestra
        addInternalFrameListener(this);

        //instancio un Thread per la gestió de la connexió
        listener = new Thread (this);

        // inicialitzo la data d'inici de la connexió
        dataConnexio = new Date();

        listener.start ();
    }

    /**
     * crea els menús de la finestra
     */
    private JMenuBar creaMenuBar() {

        // creem un objecte barra de menu
        JMenuBar mManteniment = new JMenuBar();

        // creem el primer objecte menu, que serà arxiu
        JMenu arxiu = new JMenu("Arxiu");

        // creem un component del menu Arxiu i li associem un event
        connexionsAnteriors = new JMenuItem("Connexions anteriors");
        arxiu.add(connexionsAnteriors);
        connexionsAnteriors.addActionListener(new ConnexionsAnteriorsListener());

        // creem un component del menu Arxiu i li associem un event
        arxiu.addSeparator();
        sortir = new JMenuItem("Sortir");
        arxiu.add(sortir);
        sortir.addActionListener(new SortirListener());

        // assigno la barra de menú a la finestra
        setJMenuBar(mManteniment);

        //afegeixo les opcions a la barra de menú
        mManteniment.add(arxiu);

        // retorno el menú
        return mManteniment;
    }

    /**
     * Listener pel menú opció sortir
     */
```

```
    private class SortirListener implements ActionListener {
        public void actionPerformed(ActionEvent event) {
            // tanco tot i crido de manera automàtica a internalFrameClosed
            dispose();
        }
    }

    /**
     * Listener per a mostrar les connexions anteriors del client al servidor
     */
    private class ConnexionsAnteriorsListener implements ActionListener {
        public void actionPerformed(ActionEvent e) {

            // variables
            BufferedReader in = null;     //buffer de lectura del fitxer log
            StringTokenizer tokenizer;    //tokenizer
            String line;                  //línia llegida
            String data = "";             //data de connexió
            String host = "";             //host de connexió (adreça IP)
            String port = "";             //port remot de la connexió
            String inici = "";            //H:M:S inici de la connexió
            String fi = "";                   //H:M:S fi de la connexió
            String temps = "";            //H:M:S temps total de connexió
            Vector rowData = new Vector();    //vector de vectors amb les dades a mostrar

            try {

                // obro el fitxer de log
                in = new BufferedReader( new FileReader("./logs/connClient.log"));

            } catch(FileNotFoundException  fnfExc) {
                    System.err.println("El fitxer log no existeix");}

            try {
                // llegeixo el fitxer log mentre hi hagi dades
                while ((line = in.readLine()) != null) {

                    if (line != null) {
                        // tokenitzo la línia
                        tokenizer = new StringTokenizer(line, "|");

                        // si no és una línia inacabada (la afegida a la connexió actual...)
                        if (tokenizer.countTokens() > 4 ) {

                            // omplo les variables
                            data = tokenizer.nextToken("|");
                            host = tokenizer.nextToken("|");
                            port = tokenizer.nextToken("|");
                            inici = tokenizer.nextToken("|");
                            fi = tokenizer.nextToken("|");
                            temps = tokenizer.nextToken("|");

                            // si la connexió és al host i port remot actuals...
                            if ( port.equals((new Integer(s.getPort())).toString()) &&
                                 host.equals(InetAddress.getByName(getHost()).toString())) {

                                // afegeixo les dades a un vector
                                Vector vector = new Vector();
                                vector.add(data);
                                vector.add(inici);
                                vector.add(fi);
                                vector.add(temps);

                                //afegeixo el vector amb les dades al vector de vectors
                                rowData.add(vector);

                            }
                        }
                    }
                }

                // tanco el fitxer log
                in.close();
```

```
            } catch(IOException IOExc) {System.err.println("Error llegint o tancant el
fitxer log");}


            // obro la finestra de diàleg amb les dades recollides
            ConnexionsClient finestra = new ConnexionsClient(FinestraClient.this,
                                        "Històric de connexions : ", rowData);
            finestra.pack();
            finestra.setVisible(true);
        }
    }

    /**
     * Listener de la interfície InternalFrameListener
     */
     public void internalFrameClosing(InternalFrameEvent e) {
     }

    /**
     * Listener de la interfície InternalFrameListener
     * Tanquem la finestra de connexió del client
     */
     public void internalFrameClosed(InternalFrameEvent e) {

        if (listener != null) {

            try {
                // escric al fitxer de log de connexions del client, en mode append
                // si el fitxer no existeix el crea de nou
                // al final s'afegeix un salt de línia
                PrintStream out = new PrintStream(
                                  new FileOutputStream("./logs/connClient.log", true));
                out.print(  DateFormat.getDateInstance().format(dataConnexio) + "|" +
                            InetAddress.getByName(host).toString() + "|" +
                            (new Integer(s.getPort())).toString() + "|" +
                            DateFormat.getTimeInstance().format(dataConnexio));


                Date fiConnexio = new Date();
                String tempsConnexio = (
                                    new TempsConnexio(fiConnexio.getTime() –
                                        dataConnexio.getTime()+10)).toString();

                out.println("|" + DateFormat.getTimeInstance().format(fiConnexio) +
                            "|" + tempsConnexio );
                out.close();

            } catch(FileNotFoundException  fnfExc) {
                                    System.err.println("El fitxer log no existeix");

            } catch(UnknownHostException  unknw) {System.err.println("No reconec el host");}

// podem ser més informatius encara ...
// JOptionPane.showMessageDialog(null, "Tanco la connexió","Tanca
connexió",JOptionPane.ERROR_MESSAGE);

            try {
                synchronized (listener) {
                    listener.interrupt();
                    listener = null;

                }

            } catch(Exception exc) {
            }
        }
    }

    /**
     * Listener de la interfície InternalFrameListener
     */
     public void internalFrameOpened(InternalFrameEvent e) {
```

```
    }

    /**
     * Listener de la interfície InternalFrameListener
     */
    public void internalFrameIconified(InternalFrameEvent e) {
    }

    /**
     * Listener de la interfície InternalFrameListener
     */
    public void internalFrameDeiconified(InternalFrameEvent e) {
    }

    /**
     * Listener de la interfície InternalFrameListener
     */
    public void internalFrameActivated(InternalFrameEvent e) {
    }

    /**
     * Listener de la interfície InternalFrameListener
     */
    public void internalFrameDeactivated(InternalFrameEvent e) {
    }

    /**
     * Gestor dels events de la finestra
     */
    public void actionPerformed(ActionEvent e) {

        // s'ha pitjat el botó enviar
        if (SHOW.equals(e.getActionCommand())) {
            try {
                String cadena = (String) input.getText();
                while (((cadena.length()) % 512) != 0) cadena = cadena.concat(" ");
                cos.write(cadena.getBytes());
                cos.flush();

            } catch (Exception ex) {
                ex.printStackTrace();
                listener = null;

            }

            input.setText ("");

        // es tanca la finestra
        }  else if ((e.getSource() == this) && (e.getID()== Event.WINDOW_DESTROY)) {
                 listener = null;
                 hide();
                 setVisible(false);
        }
    }

    /**
     * Accessor de lectura de l'atribut host
     */
    private String getHost() {
        return host;
    }

    /**
     * Gestor de la connexió (Thread)
     */
    public void run () {
        try {

            oosOut = new ObjectOutputStream(s.getOutputStream());
            oosIn = new ObjectInputStream(s.getInputStream());

            System.out.println("Enviant la clau pública al servidor...");
```

```
            // envio la clau pública al servidor...
            oosOut.writeObject(DHKey.getPublicKey());

            System.out.println("Rebent la clau pública del servidor...");

            //espero la clau pública del servidor...
            while (keyServer == null) keyServer = (PublicKey) oosIn.readObject();

            System.out.println("Generant la clau secreta...");
            DHKey.setSecretKey(keyServer);

            System.out.println("Intercanvi Diffie-Hellman finalitzat...");

            // creem un objecte Cipher amb l'algorisme DES per xifrar els missatges
            cipherOut = Cipher.getInstance("3DES/ECB/NoPadding", "IAIK");
            cipherOut.init(javax.crypto.Cipher.ENCRYPT_MODE, DHKey.getSecretKey());
            cos = new CipherOutputStream(s.getOutputStream(), cipherOut);

            // creem un objecte Cipher amb l'algorisme DES per desxifrar els missatges
            cipherIn = Cipher.getInstance("3DES/ECB/NoPadding", "IAIK");
            cipherIn.init(javax.crypto.Cipher.DECRYPT_MODE, DHKey.getSecretKey());
            cis = new CipherInputStream(s.getInputStream(), cipherIn);

            // envio el Nick al servidor
            String cadena = this.nick;
            while (((cadena.length()) % 512) != 0) cadena = cadena.concat(" ");
            cos.write(cadena.getBytes());
            cos.flush();

            byte[] in = new byte[1000];
            byte[] out= new byte[1000];

            int length;

            while (listener != null) {

                length = cis.read(in);
                cadena = new String(in,0,length);
                if (cadena.trim().length() > 0) display.append (cadena.trim() + "\n");

            }

        } catch (Exception ex) {
            ex.printStackTrace ();

        } finally {

            try {
                if (listener != null) {
                    synchronized (listener) {
                        input.setVisible(false);
                        validate ();

                        oosOut.close ();
                        oosIn.close ();
                        cis.close ();
                        s.close ();

                        listener = null;
                    }
                }

            } catch (IOException ex) {
                ex.printStackTrace ();

            }

        }
    }
}
```

Comentaris de la classe FinestraClient

La classe gestiona la connexió dels clients a un port concret del servidor a partir de la classe *GestioClients*.

Quan es crea la finestra es genera un fil que inicia una conversa amb la instancia de *XatServer* generada a *FInestraServidor*, aquesta instancia crea una instancia de *XatHandler* que gestionarà la connexió al client actual.

Un cop s'ha establert la connexió. La classe s'encarrega de rebre del servidor la seva clau pública, enviar al servidor la pròpia clau pública de la part client i generar a la part del client la clau privada que permetrà establir la connexió xifrada amb l'algorisme 3DES.

Un cop client i servidor s'han entès (amb l'intercanvi de claus de *Diffie-Hellmann*), la comunicació xifrada s'estableix a través dels objectes *CipherOutputStream* (`cos`) i *CipherInputStream* (`cis`) de la pròpia classe.

Cada cop que el client escriu al *stream* de sortida s'envia el missatge xifrat al servidor, que s'encarrega de fer el broadcast a la resta de clients connectats al mateix port a través de la classe *XatHandler*.

Des d'aquesta classe es gestiona el llistat de *ConnexionsClient.*

Quan es tanca la connexió (al tancar la finestra) s'escriu a l'arxiu log de dades persistents ./logs/connClient.log la informació del temps í data de connexió del client actual al port i host actuals..