Xat Segur

Autor: Eloi Andreu Mas
ETIS

Consultor: Antoni Martínez Ballesté

18/06/2004

En aquest Treball de Fi de Carrera s'intentarà resoldre el problema de la seguretat informàtica en les comunicacions entre dues entitats i poder, d'aquesta manera, protegir la informació sensible quan viatgi per la xarxa. Per fer-ho utilitzarem una sèrie de tècniques basades en la **criptografia**.

Així doncs, el primer que farem serà un petit anàlisi de les diverses tipologies d'atacs que ens podem trobar, per després buscar una forma segura de contrarestar-los.

És clar però, que haurem d'arribar a un compromís entre la seguretat i l'eficiència del sistema. Per exemple, no ens seria de gran utilitat dissenyar un programari en el que el temps necessari per protegir certa informació fos superior al temps de vida útil d'aquesta informació.

El programari que implementarem consisteix en un sistema de missatgeria instantània (xat) entre dos usuaris que només disposen d'un canal insegur per comunicar-se. Així doncs, analitzarem i implementarem les tècniques necessàries per trobar una solució a aquest cas concret, tot i que aquestes tècniques es poden fer extensibles a tot tipus de software que requereixi una comunicació segura entre dues parts.

Índex

1.-Introducció

En els últims anys, la seguretat en les comunicacions informàtiques s'ha convertit en una necessitat, tant en l'àmbit professional com en el domèstic. El creixement de les xarxes de computadors juntament amb l'increment de la potència de càlcul del maquinari actual ens obliga a establir una seguretat més forta en els sistemes informàtics.

L'objectiu d'aquest Treball de Final de Carrera consisteix en realitzar un sistema de missatgeria instantània segur. Concretament ens interessa garantir els següents requisits de la comunicació segura: **privacitat, autenticitat, integritat i no repudi**, els quals explicarem més endavant.

El sistema de missatgeria instantània serà un programa client/servidor que ens permetrà mantenir vàries converses punt a punt simultànies, cadascuna d'elles xifrada amb una clau de sessió diferent, que negociaran les dues parts a l'inici de la comunicació.

1.1 Enfocament i mètode seguit:

Donat que hem de construir un programari complex partint de zero, i que a més hi ha limitació en les dates de finalització (com en la majoria de projectes), es fa evident que necessitem elaborar una bona planificació, subdividint-lo amb tasques i subtasques, i assignar fites temporals realistes a cadascuna d'elles.
Gràcies a aquesta planificació podrem detectar de seguida qualsevol retràs que es produeixi i solucionar-lo ràpidament.

1.2 Planificació del Projecte:

Tal i com ja hem comentat, per la realització d'un projecte d'aquesta envergadura és imprescindible una bona planificació.
Hem de tenir en compte que aquesta planificació ha de tenir un grau de flexibilitat que ens permeti introduir-hi petites modificacions al llarg del temps durant tota la realització del projecte. És preferible modificar les dates de finalització previstes d'una o més tasques que no pas anar arrossegant les tasques fins arribar al punt que es faci impossible el seguiment de tota la temporització establerta.

El primer que farem, doncs, serà especificar quines seran aquestes tasques:

- **T1: Cerca d'informació**: Aquesta tasca consisteix en cercar tota la informació necessària per la realització del projecte, si més no per tenir una bona base per on començar. De fet, he comprovat que aquesta tasca no havia finalitzat totalment fins que el projecte ha estat del tot acabat, perquè a mida que avançava anaven sortint nous dubtes (tant de Concepte com d'Implementació) que necessitaven resposta.

Aquesta tasca es subdivideix en les següents subtasques:

- Cerca d'informació sobre l'algorisme Diffie-Hellman
- Cerca d'informació sobre l'algorisme DES
- Cerca d'informació sobre comunicació amb sockets
- Cerca d'informació sobre llibreries que suportin Diffie-Hellman i DES

- **T2: Disseny**: En aquesta tasca es desenvoluparà el disseny del programari, especificant els requisits que haurà de complir i donant una solució viable per poder-los realitzar. És possible que més endavant, en l'etapa d'implementació, s'hagi de revisar aquest disseny i introduir-hi alguna modificació.

  La podem dividir en les següents subtasques:

  - Disseny del programa, processos, comunicació i interacció Client/servidor
  - Disseny de la interfície amb l'usuari
  - Desenvolupar el programari a alt nivell i de forma descriptiva

- **T3: Implementació**: Aquesta tasca consisteix en la codificació del programari seguint el disseny que haurem realitzat prèviament. A mida que es va implementant el programa pot ser necessària una revisió del disseny.

  Es descomposa en les següents subtasques:

  - Implementació de la interfície gràfica
  - Afegiment del cos del programa, amb rutines de comunicació
  - Afegiment de l'intercanvi de claus i del xifratge

- **T4: Redacció de la Memòria**: Aquesta tasca consisteix en l'elaboració d'aquest document, explicant tot el treball realitzat.

  La podem dividir en les següents subtasques:

  - Elaboració de la memòria: objectiu del projecte i explicar tota la informació que hem extret a la tasca T1
  - Elaboració de la memòria: disseny del programa (amb diagrames i descripció a alt nivell)
  - Elaboració de la memòria: proves, incidències i conclusions
  - Elaboració de la memòria: apèndix amb codi font degudament comentat.
  - Presentació del projecte amb PowerPoint

La planificació temporal de totes aquestes tasques s'ha vist lleugerament alterada durant aquest temps, sobretot en les primeres etapes de cerca d'informació en la que es va fer necessari a obtenir una visió general del problema a resoldre, buscant possibles problemes que poden sorgir, solucions i alternatives d'implementació.

Es complica bastant començar de zero el disseny d'un projecte sense tenir abans la idea clara del que es vol aconseguir i del camí a seguir per arribar-hi.

També s'ha produït algun petit retràs en l'etapa d'implementació perquè a mida que l'anava fent m'adonava que havia de modificar i/o afegir alguna de les decisions de disseny.

Finalment la temporització final es pot resumir en el quadre següent:

Tasca	Descripció	Data Inici	Data Fi	Eines Necessàries
PT	**Pla de Treball**	**09/03/2004**	**09/03/2004**	
T1.1	Cerca d'informació sobre l'algorisme Diffie-Hellman	09/03/2004	16/03/2004	Internet
T1.2	Cerca d'informació sobre l'algorisme DES	09/03/2004	16/03/2004	Internet
T1.3	Cerca d'informació sobre comunicació amb sockets i SSL	09/03/2004	16/03/2004	Internet
T1.4	Cerca d'informació sobre llibreries que suportin Diffie-Hellman i/o DES	09/03/2004	16/03/2004	Internet
T2.1	Disseny del programa, processos, comunicació i interacció Client/servidor	16/03/2004	23/03/2004	Microsoft Visio o similar. Resultat de T1.1, T1.2, T1.3 i T1.4
T2.2	Disseny de la interfície amb l'usuari	23/03/2004	30/03/2004	Microsoft Visio o similar. Resultat de T1.1, T1.2, T1.3 i T1.4
T2.3	Desenvolupar el programari a alt nivell i de forma descriptiva	13/04/2004	20/04/2004	Microsoft Visio o similar. Resultat de T1.1, T1.2, T1.3 i T1.4
PAC2	**PAC2**	**13/04/2004**	**13/04/2004**	
T3.1	Implementació de la interfície gràfica	20/04/2004	27/04/2004	Entorn de desenvolupament en Java
T3.2	Afegiment del cos del programa, amb rutines de comunicació.	27/04/2004	03/05/2004	Entorn de desenvolupament en Java. Llibreries sockets
T3.3	Afegiment de l'intercanvi de claus i del xifratge	03/05/2004	17/05/2004	Entorn de desenvolupament en Java, Llibreries sockets i criptografiques
PAC3	**PAC3**	**17/05/2004**	**17/05/2004**	
T4.1	Elaboració de la memòria: objectiu del projecte, explicar T1.1 T1.2 i T1.3	18/05/2004	25/05/2004	Resultat de T1.1, T1.2, T1.3 i T1.4

T4.2	Elaboració de la memòria: disseny del programa (amb diagrames i descripció a alt nivell)	18/05/2004	25/05/2004	Microsoft Visio o similar
T4.3	Elaboració de la memòria: proves, incidències i conclusions	25/05/2004	01/06/2004	
T4.4	Elaboració de la memòria: apèndix amb codi font degudament comentat.	01/06/2004	08/06/2004	
T4.5	Presentació del projecte amb PowerPoint	08/06/2004	15/06/2004	Microsoft PowerPoint
LLIURAMENT	**Lliurament Final**		**18/06/2004**	

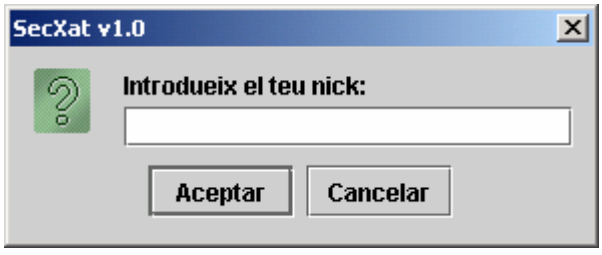
1.3 Producte obtingut i Funcionament del programari:

El resultat final que hem obtingut és la implementació del programari de missatgeria instantània que anomenarem **SecXat** juntament amb el seu disseny i tota la documentació relativa al projecte.

El programa l'implementarem amb el llenguatge de programació Java, tal i com comentarem més endavant, i per tant tindrem un software multiplataforma que constarà d'una sèrie de classes precompilades amb ByteCode en diversos fitxers.
Aquests fitxers seràn frmClient.class, threadClient.class, dimoniServidor.class, Missatge.class, Enciptador.class i SecXat.class.
Per poder executar el programa necessitarem tenir el Java Runtime Environment, que es pot descarregar de la web de Sun Microsystems http://www.sun.com, instal·lat a l'equip.

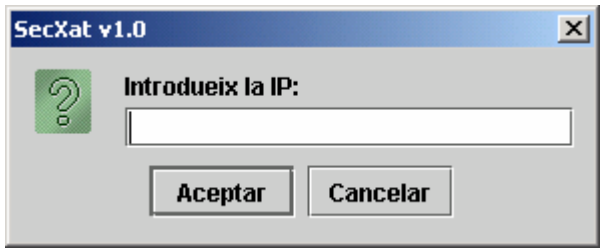
El programa s'inicia executant la classe SecXat.

El primer que ens apareixerà serà una diàleg que ens demanarà amb quin nick ens volem indentificar.



Seguidament ens apareixerà la pantalla principal amb dos botons, un per iniciar el servidor, és a dir, per començar a escoltar les peticions de connexió i poder rebre les trucades. I un altre per iniciar una conversa.

Si premem la opció *Nova Conversa* ens apareixerà un diàleg que ens demanarà l'adreça IP de la màquina amb quin ens volem comunicar.



Una vegada hem introduït l'adreça IP s'obrirà una nova finestra per aquesta conversa.

En aquesta finestra hi haurà un espai on escriurem el missatge que volem enviar i un altre on anirà apareixent tota la conversa, tant els missatges rebuts com els enviats.

Com es pot veure, el funcionament de l'aplicació és senzill i intuïtiu, de manera que serà fàcil d'utilitzar tant per usuaris experts com per usuaris novells.

1.4 Breu explicació dels altres capítols de la memòria:

En els següents capítols entrarem en matèria sobre el disseny i implementació del programari, i també farem una explicació de les tècniques criptogràfiques utilitzades i dels mecanismes de comunicació mitjançant sockets.

- En primer lloc tenim el capítol *Documentació i Tècniques Utilitzades* format pels següents apartats:
    - *Comunicació de processos a través d'Internet (Sockets)* on s'explica com es fa l'intercanvi d'informació entre dos equips remots.
    - *Criptografia* on explicarem les tècniques criptogràfiques que farem servir en la realització del programari, com ara la criptografia de clau simètrica, criptografia de clau pública, funcions de hash, codis d'autenticació MAC, etc...

- *Disseny.* En aquest capítol exposarem l'anàlisi i el disseny del programari. És a dir que volem resoldre i com ho farem

- *Implementació.* Descripció de la implementació del programari, és a dir, amb un llenguatge s'ha implementat, quines llibreries s'han utilitzat i com. També apuntarem les modificacions de disseny que han sigut necessàries.
  En aquest capítol hi inclourem el codi del programari amb alguns comentaris.

- Conclusions.

- Glossari. Descripció i definició dels termes utilitzats en la memòria.

2. Documentació i tècniques utilitzades

Explicarem de forma resumida tota la documentació que hem obtingut a l'etapa de recollida d'informació.

2.1 Comunicació de processos a través d'Internet (Sockets)

Els **sockets** ens ofereixen un mecanisme de comunicació entre processos que resideixen en màquines diferents (o la mateixa). Podem definir els sockets com els extrems finals d'aquesta comunicació i actuen com si fossin **pipes** bidireccionals. La comunicació entre processos mitjançant sockets es basa en la filosofia **client/servidor**, és a dir, hi haurà un procés, que actuarà de servidor, que crearà un socket amb un nom conegut (IP,Port) pels clients i esperarà que aquests s'hi connectin. Els processos clients realitzaran peticions de connexió a aquest servidor.

Els sockets poden tenir una sèrie de propietats que defineixen el tipus de connexió que es realitzarà:

- Lliurament de dades en el mateix ordre en què van ser enviades
- Lliurament de dades sense duplicacions
- Lliurement fiable de dades
- Suport d'entrega de missatges urgents
- Comunicació orientada a la connexió

Així doncs, tenim els següents tipus de sockets segons les seves propietats:

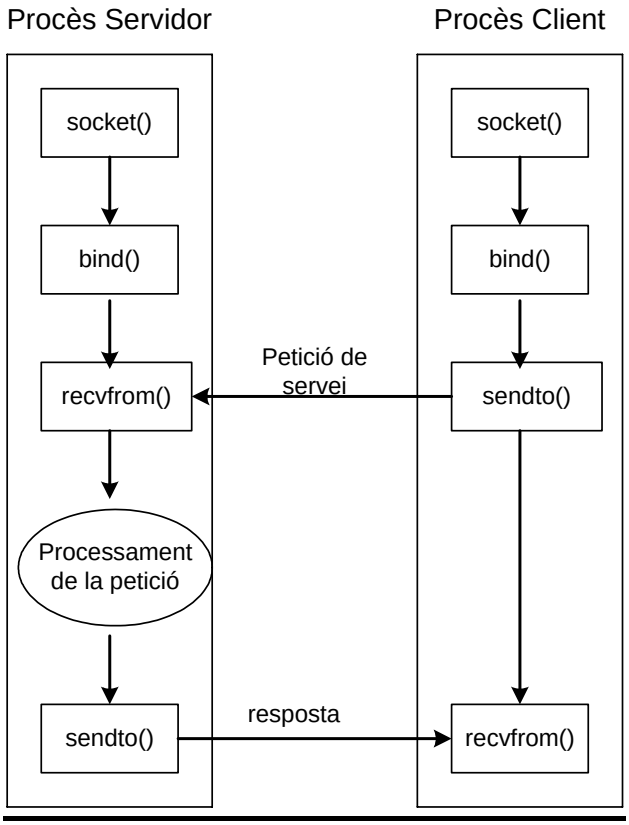
- **Sockets RAW**: Permeten l'accés als protocols de més baix nivell (com ara l'IP).
- **Sockets Stream**: Utilitza el protocol de comunicació TCP (Transfer Control Protocol) i estableix comunicacions bidireccionals orientades a la connexió i amb fiabilitat de dades. Els paquets són de tamany variable.
- **Sockets Datagram**: Utilitza el protocol UDP (User Datagram Protocol) i ens proporciona un servei de transport no orientat a la connexió. La fiabilitat no està garantida, amb la qual cosa els paquets (datagrames) poden no arribar, arribar duplicats o arribar desordenats.

Segons el tipus de sockets tenim dos modes de comunicació:

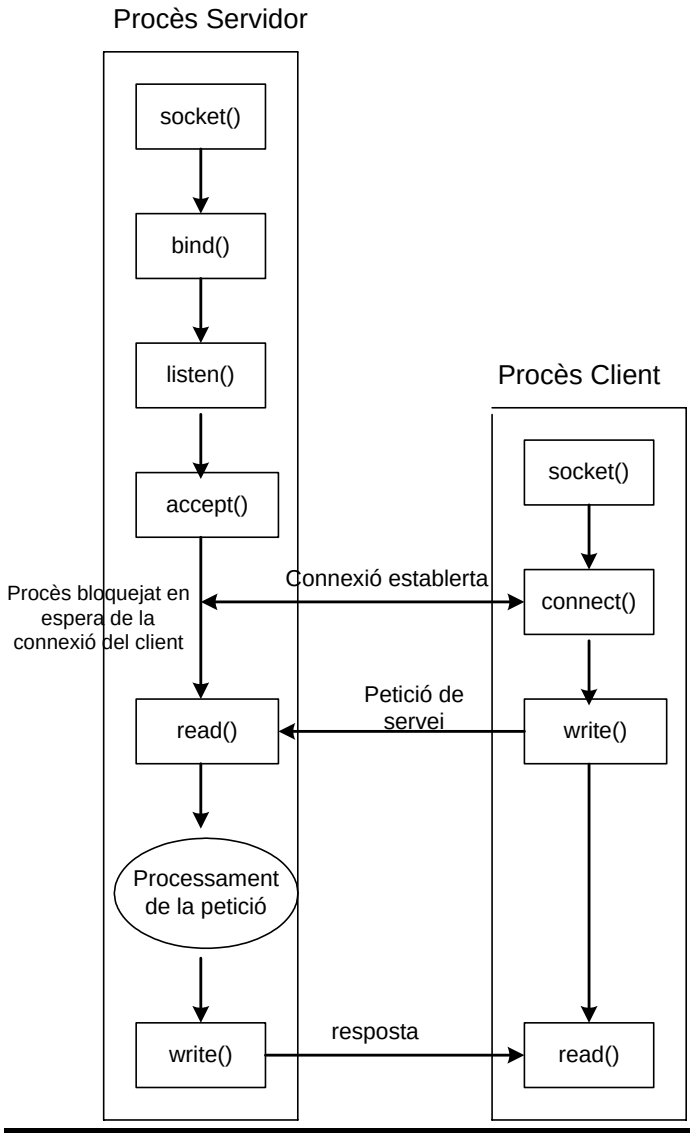
- **Orientat a la connexió**: Primer s'estableix una connexió permanent entre el client i el servidor, després les dades s'envien i es reben seqüencialment per aquesta connexió. Finalment cal desfer la connexió establerta. Com ja hem comentat abans, es garanteix la fiabilitat de les dades incloent l'ordre d'arribada.
- **No Orientada a la connexió**: Quan la comunicació és no orientada a la connexió, no cal establir una connexió permanent, sinó que directament s'envien els paquets. Per això és necessari que en cada paquet hi hagi la informació de les adreces origen i destí. No es garanteix la fiabilitat.

Per tant, en l'esquema client/servidor tenim una seqüència de crides diferent segons el tipus de comunicació:

Comunicació no orientada a la connexió:



Comunicació no orientada a la connexió:



Cada tipus de xarxa està caracteritzada per un conjunt de components com ara són el protocol de comunicacions, el sistema d'adreçament, el sistema de noms,... A aquest conjunt de components se'ls anomena **domini o família**. Els dominis poden ser **AF_INET** i **AF_UNIX**.

2.2 Criptografia

En els sistemes de comunicació sovint hi viatja informació sensible la qual volem protegir. La criptografia ens ofereix una sèrie de tècniques per protegir aquestes dades. Però quan diem "protegir", a què ens referim realment ? Per respondre a aquesta pregunta primer ens hem de plantejar quines són les amenaces a la que està sotmesa la informació, per després trobar la manera d'evitar-les.

Podem classificar les amenaces en els quatre grups següents:

1. **Espionatge:**
   L'espionatge consisteix en que algú no autoritzat vegi una informació confidencial que s'ha de mantenir en secret. L'espionatge és indetectable ja que les dades no sofreixen cap modificació.

2. **Manipulació:**
   És el fet de modificar, és a dir, eliminar, afegir o substituir el contingut de les dades.

3. **Falsificació o suplantació d'autoria:**
   Consisteix en alterar les dades de manera que sembli que l'autor és algú que realment no és.

4. **Repudi d'autoria:**
   Consisteix en la negació de l'autoria de les dades per part de l'emissor.

Per contrarestar aquestes amenaces podem fer ús de diverses eines criptogràfiques que ens ofereixen els següents serveis de seguretat:

1. **Confidencialitat:**
   Aquest servei ens permet mantenir la informació en secret de manera que ningú no autoritzat en pugui obtenir el contingut.

2. **Integritat:**
   Proporciona la garantia que les dades no han estat modificades per ningú, i que estan tal i com l'autor les va crear.

3. **Autenticitat:**
   El destinatari de les dades té la garantia que l'autor és qui realment consta com a tal. És a dir que no hi ha cap tercera part que enviï les dades en nom d'un altre.

4. **No repudi:**
   Aquest servei garanteix al destinatari que l'originador és qui realment consta com a tal, per tant aquest no pot negar-ne l'autoria.

La criptografia ens ofereix les següents eines per cobrir aquests serveis:

- El **xifratge** ens proporciona la **confidencialitat**.
- Les funcions de **Hash** ens proporcionen **integritat**.
- Els codis d'autenticació MAC i la **signatura** ens proporcionen l'**autenticitat** i el **no repudi**.

2.2.1 Xifratge

Tal i com ja hem comentat, el xifratge ens ofereix **confidencialitat** de les dades, és a dir garanteixen el secret de la informació. Això s'aconsegueix aplicant una transformació a les dades de manera que quedin inintel·ligibles.

Aquesta transformació s'anomena **algorisme de xifratge**, i converteix un **text en clar** en un **text xifrat**. Evidentment, hi ha d'haver un algorisme de **xifratge invers** que ens permeti recuperar el text en clar original a partir del text xifrat. Aquesta funció de transformació inversa haurà de ser coneguda només pel destinatari.

Els algorismes de xifratge poden ser de dos tipus: **secrets** i **públics**.
Els **secrets** són aquells que només coneixen l'originador i el destinatari. El problema que tenen és que si algú aconsegueix deduir-lo ja no es pot fer servir més.
Els **públics** són aquells que són coneguts i que depenen d'un paràmetre secret anomenat **clau de xifratge**. Si un atacant aconseguís la clau només caldria agafar-ne una altra sense haver de canviar l'algorisme.

Una premissa fonamental de la criptografia moderna és l'anomenada **suposició de Kerckhoffs**, d'acord amb la qual els algorismes han de ser coneguts públicament i la seva seguretat només ha de dependre de la clau.
Així doncs, un algorisme es considera segur si a un atacant li és impossible obtenir el text en clar encara que conegui l'algorisme i el text xifrat.

Hi ha dos tipus d'atacs que es poden portar a terme per intentar descobrir el missatge sense conèixer-ne la clau:

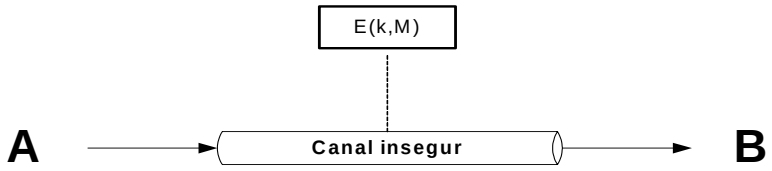
- **Criptoanàlisi**: És una tècnica basada les matemàtiques amb la que s'intenta deduir el text en clar a partir del text xifrat. L'estudi de la criptoanàlisi ens pot servir de gran ajuda per buscar debilitats als algorismes de xifratge i poder millorar-los per tal de fer-los més segurs. La criptoanàlisi juntament amb la criptografia formen la **criptologia**.

- **Força Bruta**: Consisteix en anar provant totes les claus possibles una a una fins a trobar la correcta i poder, llavors, desxifrat el missatge original.

2.2.2 Criptografia de Clau Simètrica

Els sistemes de criptografia de clau simètrica es caracteritzen perquè la clau de xifratge és idèntica a la clau de desxifratge, o bé es pot deduir directament d'aquesta. Així doncs, tant l'emissor com el receptor han de conèixer la mateixa clau.

X = e(k,M) --> X és el text M xifrat amb la clau k.
M = d(k,X) = d(k,e(k,M))



La seguretat de la clau és el factor més important en els algorismes simètrics, i això ens introdueix el problema de la distribució de claus, ja que en principi no disposem d'un canal segur per on compartir-les. Per solucionar això utilitzarem la **criptografia de clau asimètrica**, de la qual parlarem més endavant.

Els algorismes de clau simètrica els podem agrupar en dos grans grups:

2.2.2.1. Algorismes de xifratge de flux:

El xifratge de flux consisteix en combinar el text en clar M amb un missatge de xifratge S que s'obté a partir de la clau simètrica k. Per desxifrar només cal fer l'operació inversa amb el text xifrat i el mateix text de xifratge S.
La característica d'aquest tipus de xifratge és que el text M pot ser de mida variable, i el text de xifratge S ha de ser com a mínim igual de llarg. A més no és necessari tenir tot el text xifrat abans de començar a desxifrar-lo.

Hi ha diversos mètodes per obtenir un text de xifratge:

- Podríem fer que el text de xifratge fos igual que la clau: S(k) = k. Això significa que la clau ha de ser tant llarga com el missatge a xifrar. Aquest és el principi de la **xifra de Vernam**. Aquest és un algorisme molt segur, però no és gens pràctic donada la dificultat de distribuir les claus.

- Una altra possibilitat és escollir una seqüència de xifratge més curta que el missatge i anar-la repetint cíclicament tantes vegades com sigui necessari. Aquest mètode però es pot trencar fàcilment, sobretot si la clau és molt curta.

- A la pràctica el que es fa és utilitzar funcions que generen **seqüències pseudoaleatòries** a partir d'una **llavor**. La clau secreta actua com a llavor.

Un dels algorismes de xifratge de flux més utilitzat en l'actualitat, per la seva simplicitat i rapidesa, és l'**RC4 (Rivest Cipher 4)**. Va ser dissenyat per Ronald Rivest l'any 1987. La característica més destacable és que permet claus de mida variable. S'utilitza sobretot en aplicacions comercials.

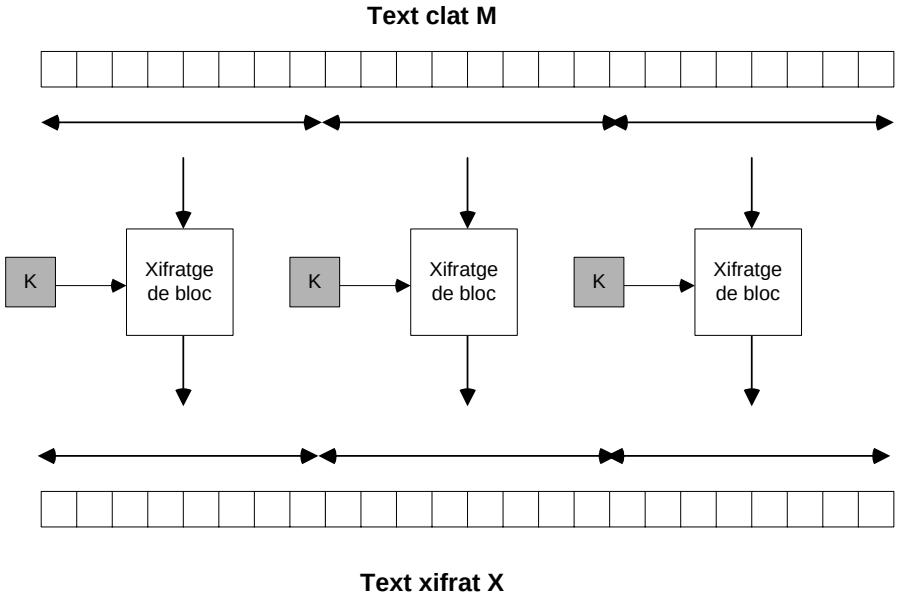
2.2.2.2. Algorismes de xifratge de bloc:

En el xifratge de bloc, l'algorisme de xifratge/desxifratge s'aplica separadament a blocs d'entrada de longitud fixa, i per cadascun d'ells el resultat és un bloc de la mateixa longitud.
Si la longitud del text no és múltiple a la mida del bloc, llavors s'afegeixen bits addicionals fins a arribar a un nombre sencer de blocs. Aquests bits s'anomenen **bits de padding**.

Com que el desxifratge també s'aplica de bloc en bloc, no poden començar a desxifrar el missatge fins que hem rebut un bloc sencer.

Generalment els algoritmes de xifratge de bloc estan basats en operacions de **substitució** i **transposició** (permutació).

L'esquema bàsic del xifratge de bloc és el següent:



Alguns exemples d'algorismes de xifratge simètric de bloc són **DES**, **Triple DES**, **AES**, **IDEA**, **Blowfish** ...

Els algorismes de xifratge de bloc tenen 4 modes de funcionament : **ECB**, **CBC**, **CFB** i **OFB**. Aquests modes de funcionament els veurem amb més detall una mica més endavant, quan parlem de l'algorisme DES.

2.2.2 Criptografia de Clau Pública

Com ja hem comentat anteriorment, un dels principals inconvenients de la criptografia simètrica és la distribució de claus, és a dir, que totes les parts es posin d'acord en les claus simètriques que faran servir per comunicar-se, sense que una tercera part que estigui escoltant el diàleg pugui deduir quines són aquestes claus.

Aquest problema el podem solucionar amb la criptografia de **clau pública**, també anomenat de **clau asimètrica**.
En un algorisme de clau pública es fan servir dues claus diferents, una pel xifratge i l'altre pel desxifratge. La característica principal que han de complir aquestes claus és que una d'elles, **anomenada clau púlica**, es pot obtenir fàcilment a partir de l'altra, anomenada **clau privada**. Però en canvi, és molt difícil, computacionalment parlant, obtenir la clau privada a partir de la pública.

Típicament els algorismes de clau pública permeten xifrar amb la clau pública i desxifrar amb la clau privada:

$$X = e(k_{pub}, M)$$
$$M = d(k_{pr}, X)$$

El funcionament és el següent: cada una de les parts genera el **seu parell** de claus (pública i privada), llavors publica la seva clau pública i manté en secret la privada. El fet de publicar la clau pública no compromet de cap manera la privacitat de la clau privada, ja que és inviable trobar la segona a partir de la primera.

Quan A vol enviar un missatge a B l'encripta amb la clau pública de B. Quan B rep el missatge el desencripta amb la seva clau privada. Així ens assegurem que només B podrà desxifrar el missatge, ja que serà l'únic que tindrà la seva clau privada. Anàlogament, quan B vol enviar un missatge a A el xifra amb la clau pública d'aquest, i A el desxifra amb la seva clau privada.

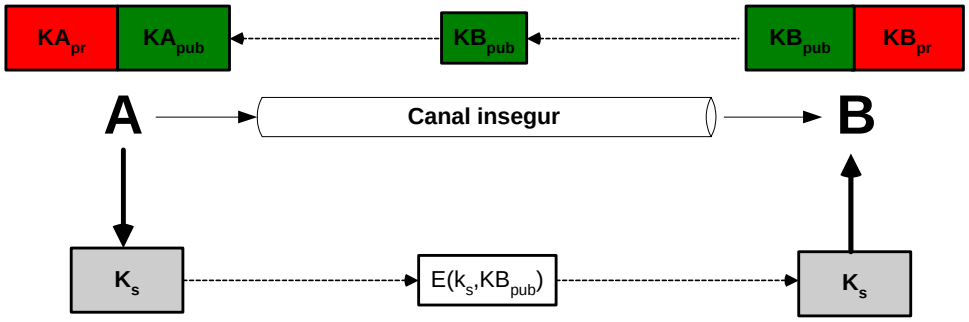
Alguns dels algorismes de clau pública ens permeten, també, xifrar amb la clau privada i desxifrar amb la clau pública. Aquesta situació, que en principi podria semblar absurda perquè tothom podrà desxifrar el missatge, ens proporciona el servei d'autenticació. Això és així perquè només l'originador del missatge el pot haver xifrat amb la seva clau privada. A aquest procés se li dóna el nom de **signatura digital**.

Per tant, amb la criptografia de clau pública també tenim un sistema que ens permet xifrar, però el problema que te és que és considerablement més lent (uns quants ordres de magnitud més lent per la mateixa quantitat de bytes) que els algoritmes de clau simètrica. A més, perquè aquest tipus d'algorismes siguin segurs hem de tenir unes claus molt més grans.
És per això que utilitzarem la criptografia de clau pública i la criptografia de clau simètrica de forma combinada per tal de millorar les prestacions de velocitat sense perdre cap de les virtuts que ens ofereixen.

De la criptografia de clau pública ens centrarem, doncs, en el seu ús per intercanviar una clau simètrica, anomenada **clau de sessió**, amb la que xifrar/desxifrar la resta de la conversa.

El funcionament seria el següent:

- B envia la seva clau pública a A
- A genera una clau simètrica k_s
- A xifra k_s amb la clau pública de B (KB_{pub}) i ho envia a B
- B desxifra el missatge amb la seva clau privada KB_{priv} i n'obté k_s
- A partir d'aquí A i B xifren/desxifren els missatges amb la clau secreta k_s



Els atacs contra la criptografia asimètrica depenen principalment dels següents factors:

- **La longitud de les claus**: Cada algorisme té uns valors de mida de la clau que es consideren segurs, però a mesura que la tecnologia avança es fa necessària una actualització d'aquests valors.
  Per exemple, per l'algorisme RSA es consideren segures les claus de 800 bits, però tot i així s'acostuma a fer servir claus de 1025 bits.

- **Qualitat de les claus**: Si la implementació de la generació "aleatòria" de les claus no és capaç de crear prou claus diferents, és a dir que no avarca tot l'espai possible, ens podem trobar que dos usuaris tinguin la mateixa clau o que algú que conegui l'algorisme generador de claus pugui trobar la nostra, ja que l'espai de búsqueda serà molt més petit que el teòric.

- **Secret de la clau privada**: Tota la seguretat de la criptografia de clau pública rau en la privacitat real de la clau privada. Si un atacant aconseguís fer-se amb aquesta clau, seria devastador.

2.2.4 Funcions de Hash

Hi ha una sèrie d'algorismes basats en tècniques criptogràfiques que ens permeten obtenir una seqüència de longitud fixa de bits, relativament curta, que en principi és única per cada missatge M. Aquestes funcions, anomenades funcions de **hash** o de **resum**, ens serveixen per garantir la **integritat** dels missatges.

$$H = h(M)$$

Dos missatges M iguals tindran un hash H igual, però hi ha la possibilitat que dos missatges diferents M i M' donin el mateix resum. D'això se'n diu **col·lisió** i és degut a que només hi ha un conjunt limitat de valors H (perquè la seva longitud és fixa) en canvi de missatges M n'hi pot haver infinits.

Per poder garantir la integritat, la funció hash ha de ser una funció **hash segura** complint les condicions següents:

- Ha de ser **unidireccional**: Això vol dir que ha de ser computacionalment inviable trobar el missatge M a partir del hash H.

- **Resistent a col·lisions**: Donat un missatge qualsevol M ha de ser computacionalment inviable trobar un altre missatge M' tal que:

  h(M) = h(M'). És a dir, que tinguin el mateix hash.

2.2.5 Codis d'autenticació MAC

Els codis MAC ens ofereixen el servei d'**autenticació** mitjançant un algorisme de dues entrades: el missatge M i una clau secreta k compartida entre l'originador i el destinatari del missatge. Com a resultat dóna un codi $C_{MAC} = f(k,M)$ que serà de longitud fixa i relativament curta. L'algorisme ha de garantir que sigui computacionalment inviable trobar un missatge M' diferent a M que doni el mateix codi que M. A més també ha de ser impossible obtenir el codi d'un missatge qualsevol sense conèixer la clau secreta.

Diem que ens dóna autenticació perquè ningú que no conegui la clau podrà calcular el codi MAC, i se suposa que la clau només la coneixeran l'emissor i el destinatari, per tant només ells poden haver generat el missatge.

Normalment s'utilitzen funcions de hash segures (gràcies a les seves propietats), com ara MD5 i SHA-1, juntament amb una clau simètrica per generar els codis MAC. Aquests codis s'anomenen **HMAC**.

Les dues tècniques més habituals són :

- L'originador A calcular el resum a partir de la concatenació del missatge original M i de la clau k:

  $M_k = M + k$
  $C_{MAC} = h(M_k)$

  Llavors envia el missatge M i el codi C_{MAC} al destinatari B.

  B verifica que el resum correspon realment a M_k. Si és així vol dir que l'ha generat algú que coneix la clau secreta k, per tant ha de ser A.

- L'altre possibilitat consisteix en calcular el resum H del missatge M i després xifrar-lo amb la clau secreta k.

  $H = h(M)$
  $C_{MAC} = e(k,H)$

  A envia el missatge M i el codi $C_{MAC.}$ Per verificar-ho el destinatari haurà de recuperar el resum enviat, desxifrant-lo amb la clau k i comparant-lo amb el resum del missatge M que ell mateix calcularà.

2.2.6 Data Encryption Standard

2.2.6.1 Història:

L'any 1973 el National Insitute of Standards and Technology (NIST), anomenat National Bureau of Standards (NBS) en aquell moment, es va adonar de la necessitat de disposar d'un algorisme criptogràfic fort per protegir les dades en format digital, tant les de caràcter governamental, industrial com en els sectors privats. Els requisits d'aquest algorisme eren que havia de ser barat, fàcil d'implementar, accessible a tothom i molt segur.
No va ser fins l'any 1974 que IBM va proposar un algorisme anomenat Lucifer, que cumplia tots els requisits. El NIST amb l'ajuda de la National Security Agency (NSA) van avaluar la seguretat de Lucifer fins que finalment, el 1977 van adoptar una modificació de Lucifer com a estàndard amb el nom de **Data Encryption Standard** (DES).
En molt poc temps es va convertir en l'algorisme més utilitzat en tots els àmbits, incloent les transaccions bancàries.

2.2.6.2 Funcionament :

DES és un algorisme de clau simètrica, és a dir que la mateixa clau serveix tant per encriptar com per desencriptar, i treballa encriptant les dades en blocs de 64 bits amb una clau de 64 bits. El bit de menys pes de cada byte de la clau és un bit de paritat, que no s'utilitza en el xifratge. Així doncs, tenim una clau efectiva de 56 bits.
Per cada bloc de 64 bits obtenim un altre bloc de 64 bits que representen les dades encriptades. Si les dades que volem xifrar no són múltiples de 64 bits s'afegeixen bits de padding, és a dir, omplim el bloc amb dades supèrflues.

2.2.6.2.1 Xifratge:

El primer pas consisteix en permutar cada bit de la clau de 64 bits segons la taula PC-1 per obtenir la clau de 56 bits.

PC-1: Permuted Choice 1

Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**
1	57	49	41	33	25	17	9
8	1	58	50	42	34	26	18
15	10	2	59	51	43	35	27
22	19	11	3	60	52	44	36
29	63	55	47	39	31	23	15
36	7	62	54	46	38	30	22
43	14	6	61	53	45	37	29
50	21	13	5	28	20	12	4

La permutació comença posant el 57é bit de la clau original al primer bit de la nova clau, el bit 49é al segon bit, i així successivament fins a posar el 4rt de la clau original a la última posició de la nova clau.

Per exemple, per la clau de 64 bits K:

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

obtenim la clau K' de 56 bits:

K' = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

El següent pas consisteix en generar 16 subclaus de 48 bits. El procediment a seguir per fer-ho és el següent:

1. Posar el número d'iteracions R a 1
2. Dividir la clau de 56 bits en dos blocs de 28 bits. El dret (R) i l'esquerre (L)
3. Rotar els bits de L i de R cap a l'esquerra segons la taula següent:

Subkey Rotation Table

Round Number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Number of bits to rotate	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

4. Concatenar L' i R' per obtenir una nova K
5. Aplicar la taula PC-2 a la nova K per permutar-la i obtenir la subclau de 48 bits K[R].

PC-2: Permuted Choice 2

Bit	**0**	**1**	**2**	**3**	**4**	**5**
1	14	17	11	24	1	5
7	3	28	15	6	21	10
13	23	19	12	4	26	8
19	16	7	27	20	13	2
25	41	52	31	37	47	55
31	30	40	51	45	33	48
37	44	49	39	56	34	53
43	46	42	50	36	29	32

6. Incrementar el número d'iteracions R i repetir el procediment fins a obtenir les 16 subclaus.

Ara que ja tenim les subclaus, hem de preparar el text que volem xifrar per poder-lo tractar. El que s'ha de fer és permutar el missatge (en blocs de 64 bits) segons la taula Initial Permutation (IP).

IP: Initial Permutation

Bit	0	1	2	3	4	5	6	7
1	58	50	42	34	26	18	10	2
9	60	52	44	36	28	20	12	4
17	62	54	46	38	30	22	14	6
25	64	56	48	40	32	24	16	8
33	57	49	41	33	25	17	9	1
41	59	51	43	35	27	19	11	3
49	61	53	45	37	29	21	13	5
57	63	55	47	39	31	23	15	7

El funcionament d'aquesta taula de permutació és idèntic al de les taules PC-1 i PC-2, és a dir el bit 58 del missatge original estarà a la posició 1 del nou missatge, el bit 50 a la segona, etc... i el bit 7 a la última.

Amb aquesta permutació obtenim un nou bloc de 64 bits, que hem de dividir en dos blocs de 32 bits, L són els 32 bits de més pes i R són els 32 bits de menys pes.

Repetirem 16 vegades el procés següent, on I és el numero d'iteració i K[I] són les subclaus:

1. Agafem R[I-1] i li apliquem la la taula E-Bit Selection Table, que té un funcionament similar al de les taules de permutació però amb la peculiaritat que alguns dels bits s'utilitzen més d'una vegada. Amb això obtenim un bloc R[I-1] de 48 bits.

E-Bit Selection Table

Bit	0	1	2	3	4	5
1	32	1	2	3	4	5
7	4	5	6	7	8	9
13	8	9	10	11	12	13
19	12	13	14	15	16	17
25	16	17	18	19	20	21
31	20	21	22	23	24	25
37	24	25	26	27	28	29
43	28	29	30	31	32	1

2. Agafem el bloc R[I-1] de 48 bits obtingut al pas anterior i li fem una XOR amb la subclau K[I]. El resultat el posem en un búffer temporal per no modificar R[I-1].

3. El resultat del pas anterior el dividim en 8 segments de 6 bits cadascun. B[1] són els 6 bits de més a l'esquerra i B[6] els de més a la dreta. Aquests blocs formen un índex per les taules de Substitució S-Boxes, que són un conjunt de 8 matrius S[1]-S[8].

S-Box 1: Substitution Box 1

Row / Column	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-Box 2: Substitution Box 2

Row / Column	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-Box 3: Substitution Box 3

Row / Column	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-Box 4: Substitution Box 4

Row / Column	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-Box 5: Substitution Box 5

Row / Column	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-Box 6: Substitution Box 6

Row / Column	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-Box 7: Substitution Box 7

Row / Column	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-Box 8: Substitution Box 8

Row / Column	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

4. Agafem el primer i últim bit de B[1] i el fem servir d'índex per la fila de la taula S[1]. Els 4 bits del mig els fem servir d'índex per les columnes de la taula S[1]. Guardem el numero obtingut de la taula i repetim el procés pels altres blocs B[2] i S[2]... B[8] i S[8]. Tenim doncs, 8 números de 4 bits que concatenats formes un bloc de 32 bits.

5. Aquest bloc de 32 bits del pas anterior el passem per una taula de permutació P.

P Permutation				
Bit	**0**	**1**	**2**	**3**
1	16	7	20	21
5	29	12	28	17
9	1	15	23	26
13	5	18	31	10
17	2	8	24	14
21	32	27	3	9
25	19	13	30	6
29	22	11	4	25

6. Al resultat li fem una XOR amb L[I-1] i el posem a R[I]. Després posem R[I-1] a L[I].
7. En aquest moment tenim uns nous L[I] i R[I]. Incrementem I i repetim el procés fins que I=17, és a dir, fins que l'haguem processat 16 vegades utilitzant les 16 subclaus K[I].

Una vegada haguem acabat, concatenem L[16] i R[16] obtenint així un bloc de 64 bits.

L'últim pas consisteix en passar aquest bloc per la taula de permutació inversa de IP, l'anomenada IP^{-1}. Obtenint així el bloc final de 64 bits.

IP^(-1): Inverse Initial Permutation								
Bit	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
1	40	8	48	16	56	24	64	32
9	39	7	47	15	55	23	63	31
17	38	6	46	14	54	22	62	30
25	37	5	45	13	53	21	61	29
33	36	4	44	12	52	20	60	28
41	35	3	43	11	51	19	59	27
49	34	2	42	10	50	18	58	26
57	33	1	41	9	49	17	57	25

2.2.6.2.2 Desxifratge:

El procés a seguir per desencriptar és el mateix descrit anteriorment, però aplicant les 16 subclaus en ordre invers.

2.2.6.2.3 Modes de funcionament:

L'algorisme DES té 4 modes de funcionament segons el tractament previ que fa de cada bloc de 64 bits.

- **Electronic Code Book (ECB)**

  Aquest mode és exactament el descrit anteriorment, on cada bloc de 64 bits de dades és tractat independentment dels altres. Això significa que si les dades viatgen per la xarxa, els errors de transmissió només afecten al bloc que conté l'error. És el mode més senzill d'implementar i un dels més utilitzat tant en aplicacions comercials com privades.

- **Cipher Block Chaining (CBC)**

  En aquest mode de funcionament, a cada bloc de 64 bits encriptats se li fa una XOR amb el següent bloc de 64 bits de text clar que haguem d'encriptar. D'aquesta manera cada bloc depèn de tots els anteriors. Per conèixer el text clar d'un bloc encriptat necessitem el bloc encriptat, la clau i l'anterior bloc encriptat.
  Això significa que un error de transmissió d'un bloc afectarà a tots els blocs següents.
  El primer bloc no disposa de cap bloc anterior, per això s'utilitza un número de 64 bits anomenat Vector d'Inicialització (IV).
  Aquest mode ens ofereix més seguretat que ECB.

- **Cipher FeedBack (CFB)**

  Aquest mode ens permet encriptar blocs de menys de 64 bits. Així doncs, no ens cal fer un tractament especial (padding) als textos d'entrada que no són múltiples de 64 bits.
  El procediment consisteix en passar un bloc de 64 bits anomenat Shift Register com a entrada a l'algorisme DES. El Shift Register està inicialitzat amb un numero aleatòri i s'encripta amb DES. El resultat es passa a un nou component anomenat M-Box, que simplement agafa els M bits de més pes del bloc xifrat, on M és el número de bits que té el text clar original. A aquest valor li fem una XOR amb el text que volem encriptar i el resultat és el bloc encriptat. Finalment, aquest bloc xifrat el posem al Shift Register per tractar el següent bloc. Així doncs, igual que en el mode CBC, cada bloc depèn dels anteriors.
  És un mode molt segur, però és més lent degut a la complexitat afegida.

- **Output FeedBack (OFB)**

  Aquest mode és similar al CFB, excepte en que el resultat del DES no es torna a posar al Shift Register, de manera que els diferents blocs no depenen entre sí.

2.2.7 Diffie-Hellman

2.2.7.1 Introducció:

Mitjançant un algorisme de clau simètrica podem crear un canal de comunicació segur entre dues parts per tal que aquestes puguin intercanviar informació sensible. Però el requeriment que tenen aquests tipus d'algorismes és que les dues parts han de conèixer una mateixa clau secreta amb la qual encriptar i desencriptar les dades transmeses. Ens trobem amb el problema, doncs, de fer que les dues entitats coneguin aquesta clau secreta, tenint en compte que encara no disposem d'un canal segur per on enviar-la. Per tant, hem de trobar un sistema amb el que puguem transmetre la clau per un canal insegur amb la màxima seguretat.
L'algorisme de Diffie-Hellman ens ofereix una solució a aquest problema permetent que les dues parts es posin d'acord amb una mateixa clau secreta, sense que aquesta hagi de viatjar en cap moment pel canal insegur.

2.2.7.2 Història:

L'algorisme Diffie-Hellman fou introduït per Whitfield Diffie i Martin Hellman el 1976 i publicat en l'article "*New Directions in Cryptography*". Va establir les bases de la criptografia de clau asimètrica també anomenat de clau pública.
Des d'aquell moment, s'ha utilitzat extensament en una gran quantitat de protocols incloent Secure Sockets Layer (SSL), Secure Shell (SSH) i Internet Protocol Security (IPSec).

2.2.7.3 Funcionament :

La seguretat de l'algorisme rau en la dificultat del càlcul de logaritmes discrets.
Els passos a seguir són els següents:

1. Primer de tot hem de disposar de dos paràmetres **p** i **g** anomenats paràmetres DH. Aquests paràmetres són públics i els han de conèixer les dues parts.
   El paràmetre **p** és un número primer gran (més gran que 2). **g** és un enter més petit que **p** i que sigui arrel primitiva de **p**. O sigui que **g** ha de poder generar tots els números entre 1 i p-1 multiplicant-lo per ell mateix un cert nombre de vegades i fent el mòdul amb **p**.

   $$q \bmod p,\ q^2 \bmod p,\ q^3 \bmod p,\ ...,\ q^{(p-1)} \bmod p \text{ són números diferents.}$$

2. Una vegada tenim p i g, cada node genera, de forma privada, un numero aleatori **x** que és més petit que **(p-1)**.

3. Després, cada part genera una clau pública **y** calculada a partir de **p**, **g** i **x**:

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

4. Els dos nodes intercanvien les seves respectives claus publiques **y** i calculen la clau secreta **K**.

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

Matemàticament, es pot demostrar que les dues parts hauran calculat la mateixa clau secreta **K,** per la propietat distributiva:

$$K = (g^x \bmod p)^{x'} \bmod p = (g^{x'} \bmod p)^x \bmod p$$

Una vegada les dues parts ja disposen de la mateixa clau secreta ja poden iniciar una comunicació segura encriptant les dades amb algun algorisme de clau simètrica com per exemple DES, TripleDES, CAST, IDEA, Blowfish, etc...

Un atacant que volgués obtenir la clau secreta només disposaria dels valors **p**, **g** i les dues $\mathbf{y_i}$. Per tant, necessitaria conèixer almenys una de les k_i. ja que sinó és computacionalment molt difícil calcular **k** amb un número **p** suficientment gran.

2.2.7.4 Vulnerabilitats:

Una vulnerabilitat del protocol DH és l'anomenat **Atac de l'Home al Mig** (Man in the Middle MIM). Aquest atac consisteix en que algú (C) intercepta les claus públiques que envien les dues parts (A i B) i les substitueix per les seva pròpia clau pública. Llavors es creen dues claus secretes, una entre C i A, i l'altre entre C i B. D'aquesta manera, C pot llegir i modificar els missatges entre A i B sense que aquests se n'adonin de res.

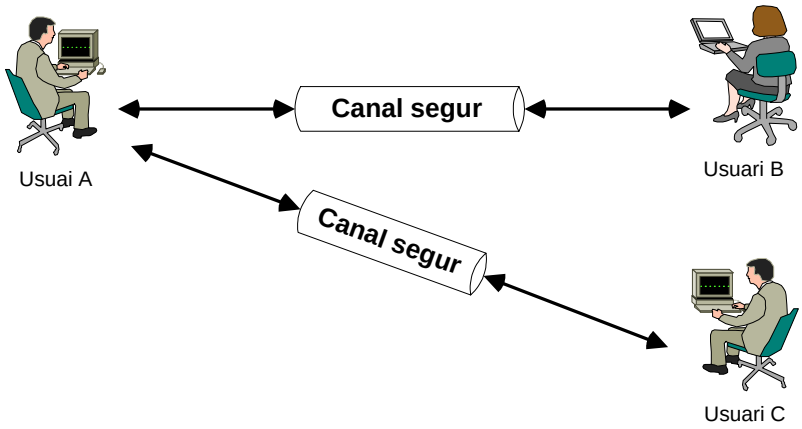
Aquest atac és molt difícil de portar a terme fora del laboratori, però tot i així és possible. Per això l'any 1992 Diffie, van Oorschot, i Wiener van desenvolupar el protocol **Authenticated Diffie-Hellman Key Agreement Protocol**. Aquesta variant del HD clàssic consisteix en que cadascuna de les parts s'ha d'autenticar amb les altres mitjançant **signatures digitals** i **certificats de clau pública**.
El funcionament és el següent: abans de començar el protocol, cadascuna de les parts ha de tenir un parell de claus publica/privada i un certificat digital de clau pública.
Durant el protocol, les parts s'envien les respectives y^x mod p també hi adjunten la signatura i el certificat de clau pública. Així algú que interceptés les comunicacions hauria de disposar de les claus privades per poder fer el MIM.

3.- Disseny

3.1 Visió general:

El programari que volem desenvolupar consisteix en un sistema de missatgeria instantània segur. Més concretament, ens interessa tenir un programa que permeti a dos usuaris mantenir una conversa segura, és a dir, que ens ofereixi els serveis de seguretat de **confidencialitat**, **integritat**, **autenticitat** i **no repudi**. A més cada usuari haurà de poder conversar simultàniament amb altres usuaris, mantenint un entorn segur independent per cadascun d'ells.

A grans trets, doncs, podem il·lustrar la funcionalitat del programari amb el següent exemple:



L'usuari A manté una comunicació segura amb els usuaris B i C. Mentre que els usuaris B i C només en mantenen una amb A. Els dos canals són totalment independents, tot i que físicament les dades viatjaran a través del mateix canal insegur. Així doncs, l'usuari C no podrà veure la informació que intercanvien A i B. I anàlogament B no veurà el que intercanvien A i C.

És clar, que a priori no disposem de cap canal segur, sinó que només tenim un canal insegur, per exemple Internet. Per això haurem d'utilitzar algunes tècniques criptogràfiques per crear un canal segur virtual entre cada parell d'usuaris que hagin de mantenir una conversa.

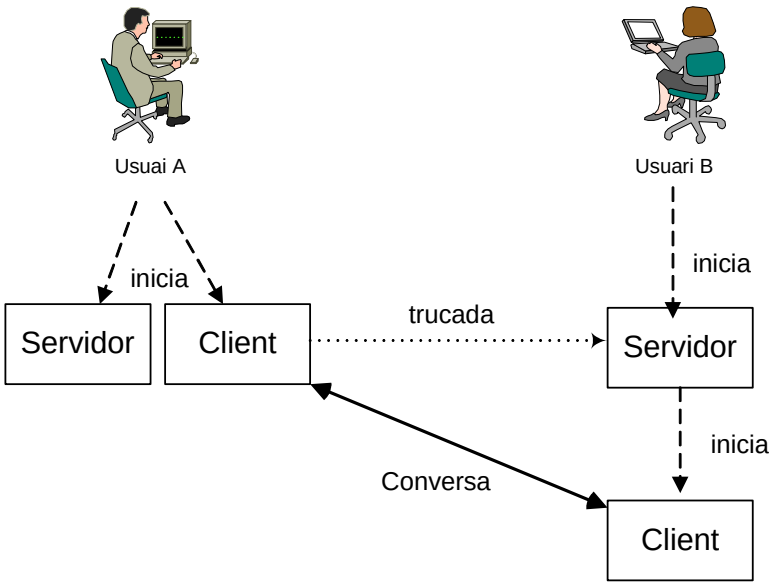
Un altre dels requisits que té el programa és que haurà de tenir una interfície gràfica amb la que interactuaran els usuaris. Així podran mantenir converses d'una manera molt més amigable que amb una simple finestra de text.

El funcionament general serà el següent:

Cada usuari tindrà una part Servidora i una part Client. La part servidora quedarà a l'espera que algú faci una "trucada", és a dir, que inici una conversa. La part client tindrà diverses instàncies i serà l'encarregada de mantenir la conversa. Quan un usuari

inicia una conversa crea una instància de la part client. Quan un servidor rebi una trucada crearà una instància de la part client.

Veiem-ne un exemple:



Els dos usuaris A i B inicien els seus respectius servidors, que queden a l'espera de que algú els faci una trucada.
En aquest exemple, l'usuari A és qui inicia la conversa i fa una trucada a B. La part servidora de B rep la trucada i crea una instància client amb la informació necessària perquè es pugui comunicar amb la part client d'A. A partir d'aquí els clients d'A i de B poden intercanviar informació directament (sense haver de passar pel servidor).
El servidor de B, una vegada ha creat un client per aquesta connexió, ja pot continuar escoltant les noves peticions.

3.2 Disseny de la Interfície gràfica:

Des del punt de vista de l'usuari, la interfície gràfica ha de disposar de les opcions necessàries que li permetin:

1. **Iniciar el programari**. L'usuari iniciarà el programari del Xat Segur, i aquest li demanarà el nom (nick) amb el que es vol identificar. Llavors el programari quedarà a l'espera que l'usuari iniciï un client o el servidor.

2. **Iniciar el servidor**. En principi la part servidora no s'iniciarà automàticament perquè es pot donar el cas que a l'usuari li convingui iniciar una o més converses sense estar disponible per altres usuaris.

3. **Iniciar un client**. Quan l'usuari vulgui establir una connexió, engegarà un client, i el sistema li demanarà l'adreça IP de la màquina amb qui es vol

comunicar. A continuació el sistema mostrarà un missatge a l'usuari remot preguntant-li si accepta o no la nova connexió. En cas afirmatiu se li iniciarà un client.

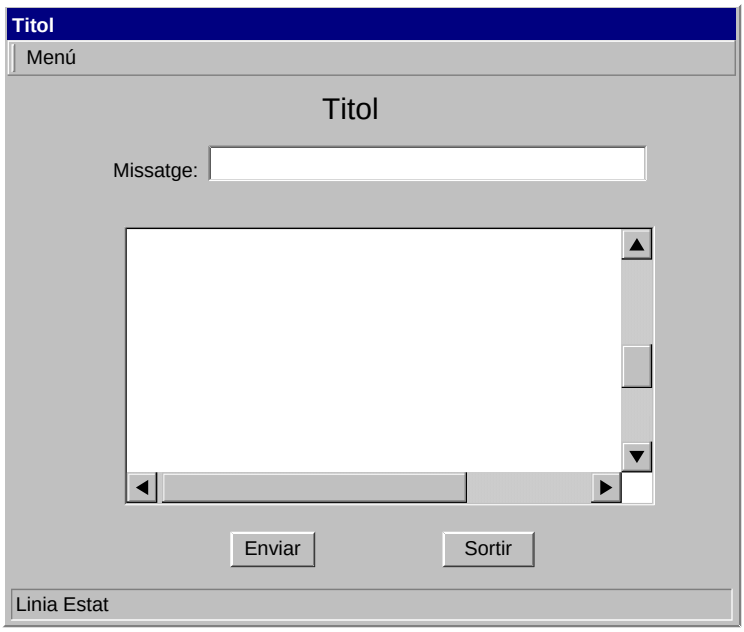
4. **Mantenir una conversa**. Una vegada s'han iniciat els clients en els dos extrems i s'ha establert la connexió, els usuaris han de poder enviar missatges i també visualitzar tota la conversa en temps real, és a dir, tant els missatges rebuts com els enviats. Quan algun dels usuaris vulgui finalitzar la conversa, podrà desconnectar la sessió i finalitzar el client. En aquest cas l'usuari oposat rebrà una notificació. A més els usuaris podran, en tot moment, emmagatzemar la conversa en un fitxer de text.

Tindrem doncs, dues interfícies gràfiques diferenciades: la primera que servirà per iniciar el servidor i els clients (les converses) i la segona que la que permetrà a l'usuari mantenir la conversa.
Per la primera en tindrem prou amb una petita finestra amb dos botons i una línia d'estat que tingui informat a l'usuari. Els botons serviran per iniciar el servidor o els clients.
L'esquema general podria ser el següent:

Com que la part servidora no té interfície gràfica, utilitzarem la línia d'estat per indicar si el servidor està actiu o no.

La segona interfície gràfica, i la més important, haurà de tenir un camp de text on l'usuari podrà introduir el missatge que vol enviar i un altre camp de text on apareixerà tota la conversa mantinguda fins al moment. També tindrem botons per enviar el missatge escrit i per sortir. A més dels botons també hi haurà un menú amb totes les opcions del programa. La interfície ha de mostrar també l'estat de la connexió.

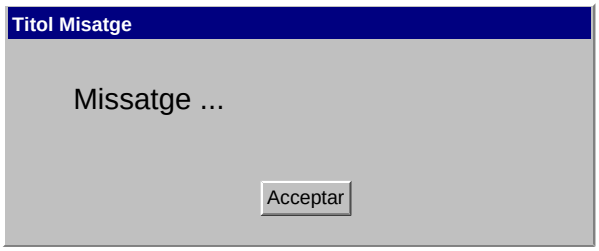
L'esquema general podria ser el següent:

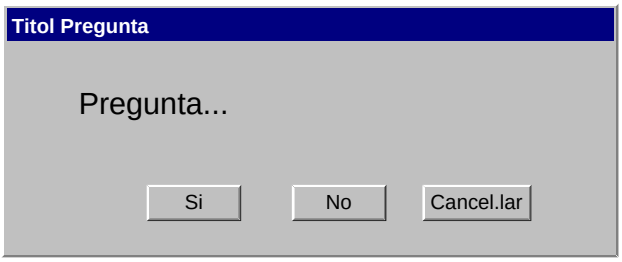


Alguns dels requisits que haurà de tenir seran:

- L'àrea de text on hi haurà la conversa ha de disposar de barres de desplaçament (scroll) tant vertical com horitzontal i el desplaçament serà automàtic.
- Quan s'introdueixi un missatge no caldrà prémer el botó d'enviar, simplement prement la tecla Return ja s'enviarà el missatge.
- La finestra es podrà minimitzar en tot moment per tal d'ocultar-la.
- Si l'usuari selecciona les opcions de *desconnectar* o de *sortir* el sistema haurà de mostrar un missatge de confirmació.
- En tot moment es mostrarà l'estat de forma actualitzada.

Pel que fa als missatges d'informació i de confirmació, tindran el següent aspecte:





Evidentment el sistema mantindrà informat a l'usuari de tots els events i condicions d'error que es produeixin.

3.3 Disseny dels processos i fils d'execució:

A nivell de processos i fils d'execució (threads) tindrem els següents:

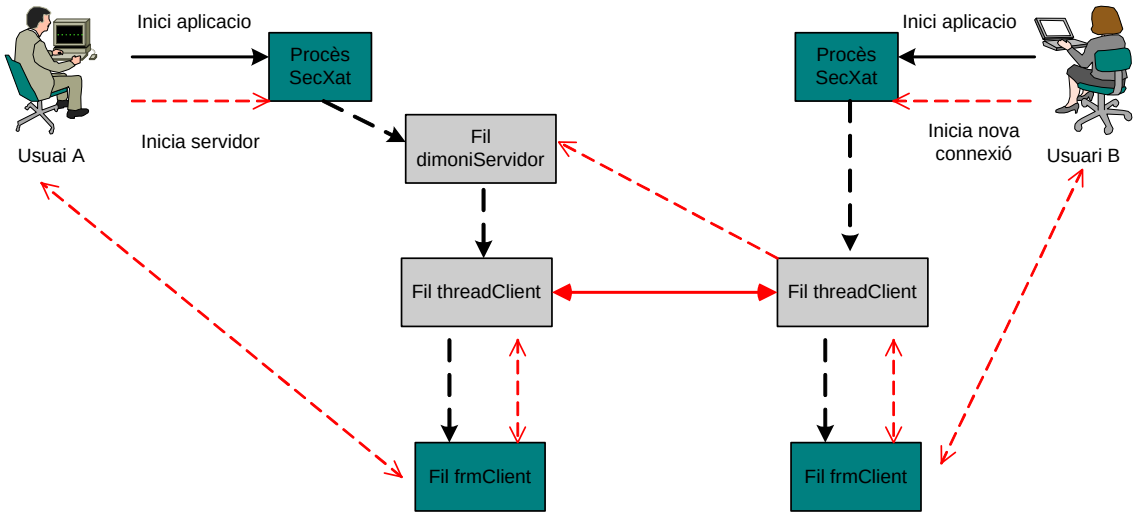
- **SecXat**: Petita interfície gràfica que permetrà a l'usuari executar el servidor i tants clients com converses vulgui iniciar. Podríem dir que és el punt de partida a partir del qual s'engegaran tots els altres processos.

- **dimoniServidor**: Aquest serà un fil d'execució que serà iniciat pel procés principal SecXat i portarà a terme les tasques de la part servidora de l'aplicatiu, és a dir, obrirà un port TCP determinat i esperarà a que algun client iniciï una connexió. Quan això es produeixi, serà l'encarregat de crear un nou thread el qual gestionarà la part client per aquella connexió.
  Aquest fil s'executarà en mode **dimoni**, o sigui que s'executarà en background. Només n'hi pot haver un d'actiu en cada moment.

- **threadClient**: Aquest fil pot ser iniciat tant pel fil dimoniServidor com pel procés principal SecXat. Serà l'encarregat de mantenir la conversa amb el sistema remot. També gestionarà tot el relacionat amb la seguretat de la connexió creant un contexte de seguretat únic. Hi haurà tants fils threadClient com converses simultànies s'hagin establert. Cadascun d'aquests threads engegarà un altre fil encarregat de recollir i mostrar tota la informació a l'usuari.

- **frmClient**: Aquest fil és una interfície gràfica que constitueix la capa de presentació de l'aplicatiu i serà amb el que interactuarà l'usuari. Serà l'encarregat de recollit els missatges que es vulguin enviar i també de mostrar la conversa en tot moment. També mantindrà informat a l'usuari de l'estat de la connexió i dels diversos events que es poden produir. Sempre l'iniciarà el fil threadClient i n'hi haurà un per conversa.

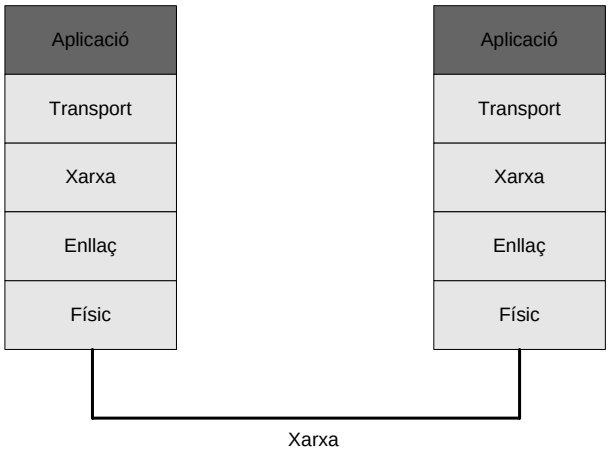
Podem representar la interacció entre els processos i els usuaris així:



- En aquest cas, els dos usuaris A i B inicien les seves respectives aplicacions, és a dir, el procés principal SecXat.
- L'usuari A prem la opció d'engegar la part servidora i llavors el procés SecXat inicia el fil d'execució dimoniServidor.
- L'usuari B prem l'opció d'iniciar una nova conversa i llavors el procés SecXat inicia un fil threadClient. Aquest crea un fil frmClient per interactuar amb l'usuari B. A més el threadClient connecta amb el fil dimoniServidor de l'usuari A.
- El thread dimoniServidor de l'usuari A quan rep la petició de connexió del threadClient de B l'accepta, crea un fil threadClient i li passa tota la informació necessària de B. Acte seguit, el threadClient d'A crea un fil frmClient per interactuar amb l'usuari A.
- A partir d'aquí els fils threadClient d'A i de B ja poden començar a intercanviar la informació que els seus respectius fils frmClient recullen dels usuaris.

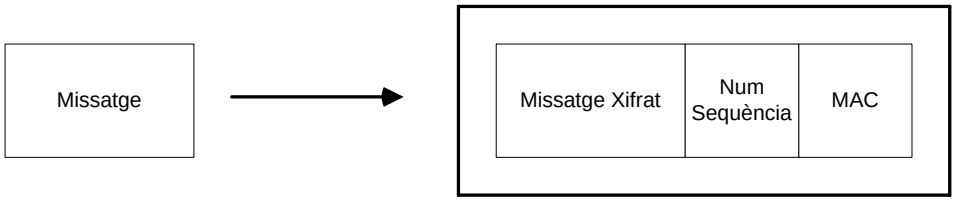
3.4 Disseny Seguretat:

Pel que fa a la seguretat, de la qual s'encarregaran els fils d'execució threadClient, la implementarem a nivell de la capa d'aplicació. Recordem les cinc capes de la pila de protocols TCP/IP:



Per tant, serà la pròpia aplicació l'encarregada de gestionar la seguretat i ho farà utilitzant tècniques criptogràfiques per xifrar/desxifrar els missatges i també per verificar-ne la integritat i l'autenticitat.

Quan un usuari vol enviar un missatge, l'aplicació el transforma i l'encapsula de manera que estigui protegit quan viatgi per la xarxa. Quan l'aplicació del destinatari rebi el missatge el desencapsularà, farà les comprovacions oportunes i el desxifrarà abans de mostrar-lo a l'usuari



Missatge Xifrat: El missatge xifrat serà el texte que haurà introduït l'usuari originador encriptat amb una clau simètrica mitjançant l'algorisme **DES**. El destinatari farà servir la mateixa clau per desxifrar el missatge. Amb aquest xifratge obtenim el servei de **Confidencialitat**.

Num Seqüència: Un possible atac que podem patir consisteix en que una tercera persona intercepti algun dels missatges que s'envien A i B i el dupliqui o be l'elimini. Una manera de protegir-nos és afegint a cada missatge un número de seqüència (també xifrat). Així si un missatge arriba duplicat o no arriba el destinatari ho podrà detectar.

MAC: També afegirem un codi MAC, calculat a partir del missatge, el número de seqüència i la clau simètrica mitjançant l'algorisme **SHA-1**. El codi MAC, tal i com hem comentat en capítols anteriors, ens proporcionarà els serveis d'**Integritat** i **Autenticitat**.

El primer que hem d'aconseguir, però, és que els dos extrems es posin d'acord en la clau simètrica que faran servir per xifrar/desxifrar la resta de la conversa. Això ho farem mitjançant l'ús de la criptografia asimètrica, concretament amb l'algorisme **Diffie-Hellman**.

3.5 Disseny de la comunicació:

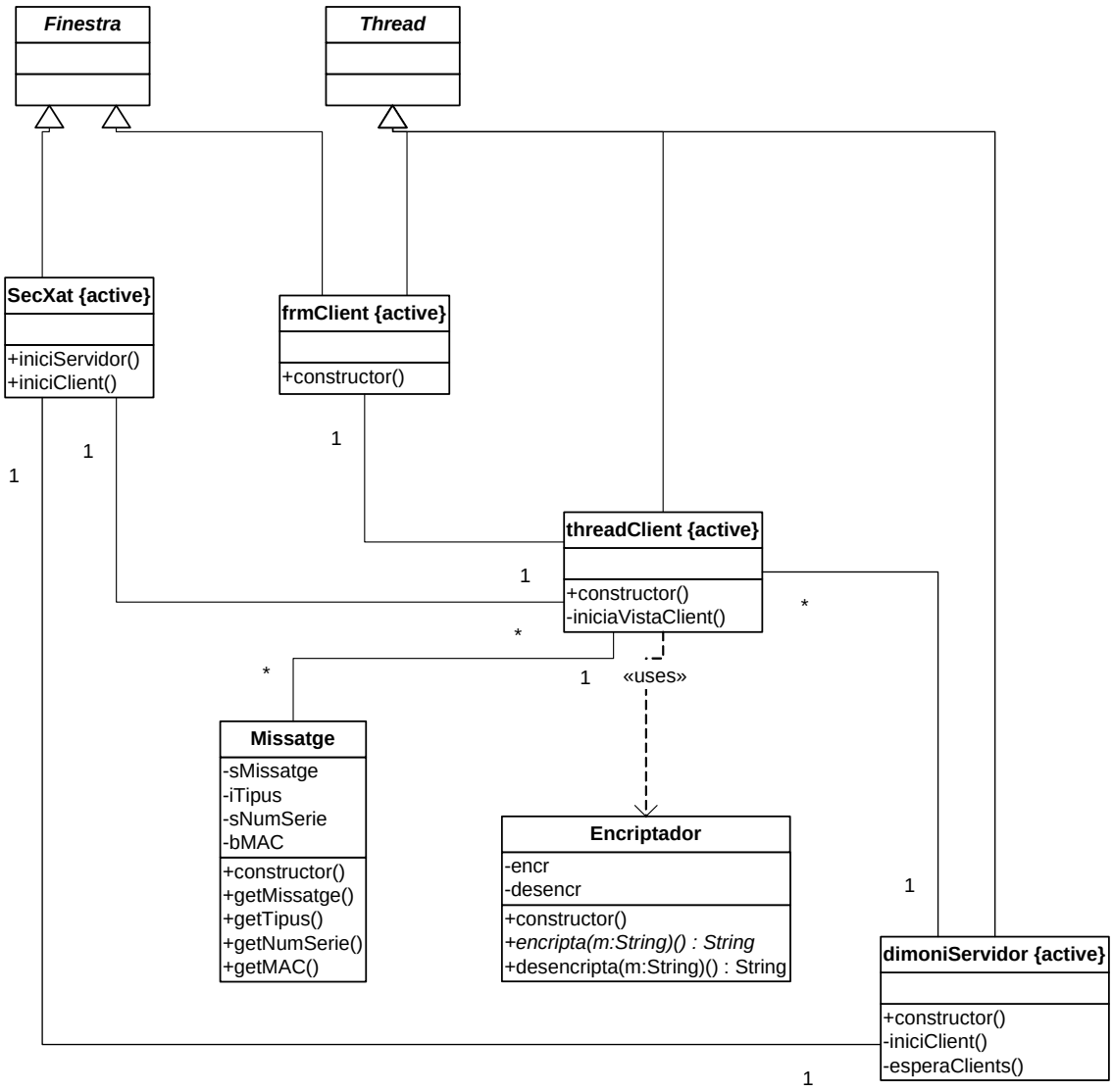
El que ens interessa de l'aplicació de missatgeria és poder comunicar dos usuaris situats en màquines remotes que estan connectades per una xarxa (habitualment Internet). Per portar a terme aquesta comunicació entre els dos extrems farem servir el mecanisme de **sockets** seguint la filosofia **client/servidor** amb la comunicació **orientada a la connexió**. És a dir, treballarem sobre el protocol TCP

El procés servidor escoltarà peticions pel port 9999 i quan algun client estableixi una connexió crearà un nou socket per comunicar-s'hi. Després el servidor continuarà escoltant per aquest port per acceptar noves connexions.

Quan un client vulgui iniciar una nova connexió obrirà un socket i el connectarà amb l'adreça IP del servidor en el port 9999.

3.6 Disseny UML:

Podem fer una primera representació de l'aplicació amb el següent diagrama estàtic en UML:



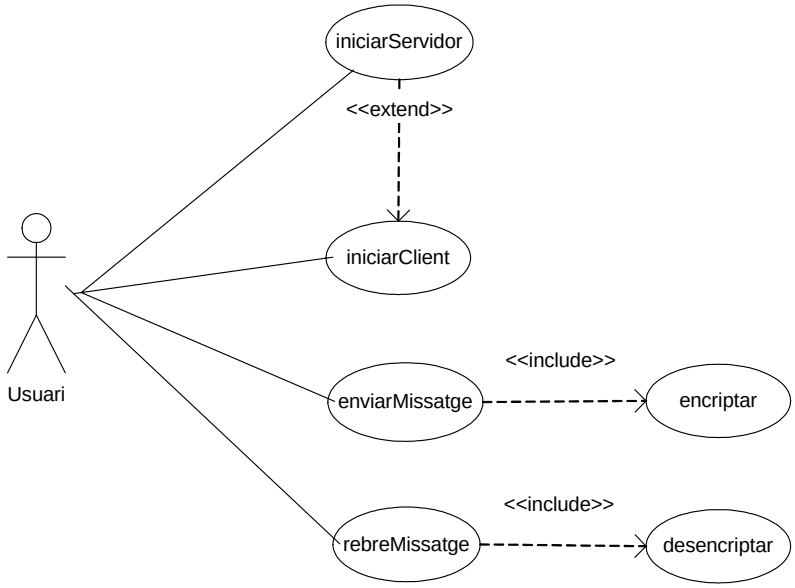
Tindrem una classe abstracta "Finestra", que s'encarregarà de la gestió gràfica, i de la que hereteran les classes "SecXat" i "frmClient". A més tindrem la classe "Thread" de la que heretaran les classes "frmClient", "threadClient", "dimoniServidor". Notem que la classe "frmClient" té una herència múltiple (de "Finestra" i "Thread"). En l'etapa de disseny ens podem permetre l'herència múltiple, però en l'etapa d'implementació haurem de buscar una solució alternativa ja que el llenguatge de programació que farem servir (Java) no suporta aquest tipus d'herència.

La classe “SecXat”, tindrà dues operacions: “iniciServidor” que crearà una instància de la classe “dimoniServidor” i “iniciClient” que crearà una instància de “threadClient”.

La classe “dimoniServidor”, que implementarà el fil d’execució de la part servidora de l’aplicació, tindrà dues operacions bàsiques: “esperaClients” que esperarà les peticions de connexió dels clients, i “iniciClient” que iniciarà una instància de la classe “thradClient” quan s’hagi establert una nova connexió.

La classe “threadClient” iniciarà una instància de la classe “frmClient”, que li servirà per recollir/mostrar les dades a l’usuari. “threadClient” utilitzarà la classe “Encriptador” per fer el xifratge/desxifratge dels missatges. A més utilitzarà la classe “Missatge” per comunicar-se amb la instància de “threadClient” de l’altre extrem de la connexió.

Amb el següent diagrama de **casos d’ús** en **UML** podem il·lustrar la interacció de l’usuari amb el sistema:



Descripció textual dels casos d’ús:

IniciarServidor:

L’actor Usuari crida la funció iniciarServidor de l’objecte SecXat, i aquest crea un objecte dimoniServidor (a menys que ja n’existeixi una instància). Aquesta instància de dimoniServidor, que serà un fil d’execució en mode dimoni, obrirà un socket de servidor pel port 9999 i quedarà a l’espera d’alguna petició. Quan rebi una petició d’un client es crearà un socket per aquella connexió. Després el thread dimoniServidor crearà una instància de la classe threadClient i li passarà com a paràmetre el socket que s’ha creat. És a dir executarà el cas d’ús iniciarClient.
dimoniServidor continuarà esperant i creant tants fils d’execució threadClient com peticions rebi.

IniciarClient: Aquest cas d'ús el pot iniciar tant el fil dimoniServidor com l'actor Usuari quan crida la funció iniciarClient de l'objecte SecXat, En aquest últim cas,
el sistema demana a l'usuari l'adreça IP remota amb qui es vol comunicar. Després obre un socket i el connecta amb el fil dimoniServidor que estarà escoltant pel port 9999 d'aquella IP. Una vegada ha creat i connectat el socket, crea una nova instància de la classe threadClient i li passa com a paràmetre aquest socket.
Cadascun dels fils threadClient que es creïn es posarà d'acord amb els seus respectius threadClients remots (mitjançant el socket que rep com a paràmetre) en una clau secreta amb la qual xifraran i desxifraran tots els missatges.
A més cada fil threadClient crearà una instància de la interfície gràfica frmClient, amb la que interactuarà l'usuari.

EnviarMissatge: L'Usuari introduirà el text que vol enviar utilitzant la interfície gràfica frmClient. Llavors la instància frmClient cridarà la operació enviarMissatge de la seva respectiva instància de threadClient, passant-l'hi com a paràmetre el text.
ThreadClient executarà el cas d'ús encriptar i després enviarà el missatge encriptat a través del socket que té obert.

RebreMissatge: Quan una instància de threadClient rebi, a través del socket obert, un missatge executarà el cas d'ús desencriptar. Si el missatge s'ha desencriptat correctament cridarà la operació rebreMissatge del seu objecte frmClient, passant-l'hi el missatge desxifrat. Aquest s'encarregarà de mostrar el missatge a l'usuari.

EncriptarMissatge: ThreadClient rebrà el text de frmClient, el xifrarà amb la clau secreta, també incrementarà en un el número de missatges rebuts i xifrarà aquest valor. Després calcularà el codi MAC i retornarà tot el que ha calculat.

DesencriptarMissatge: Per obtenir el missatge original, threadClient desxifrarà el missatge i el número de sèrie. Comprovarà que el número de sèrie sigui correcte. Després calcularà la MAC del missatge i comprovarà si correspon amb la rebuda.
Si tot és correcte retornarà el text original.

4.- Implementació

4.1 Consideracions generals:

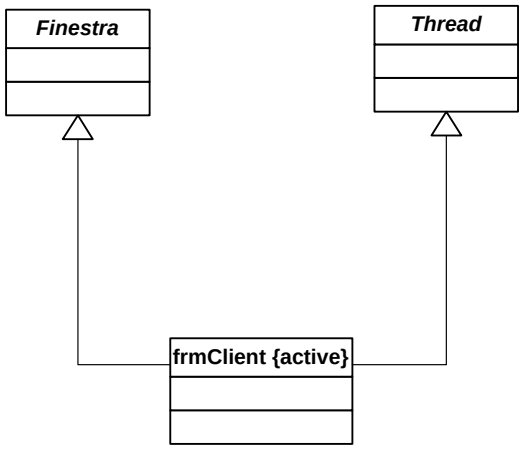
Per implementar el programari SecXat utilitzarem el llenguatge de programació orientat a objectes **Java**, aprofitant les llibreries gràfiques, de threads, de comunicació amb sockets i criptogràfiques que ens ofereix.
Farem servir l'entorn de desenvolupament **Java 2 SDK Standard Edition versió 1.4.2**, que es pot descarregar gratuïtament de la pàgina web de Sun Microsystems http://www.sun.com

També utilitzarem l'entorn gràfic de desenvolupament anomenat **Java NetBeans IDE** en la seva versió 3.5.1 per implementar les interfícies gràfiques.

4.2 Revisió del Disseny:

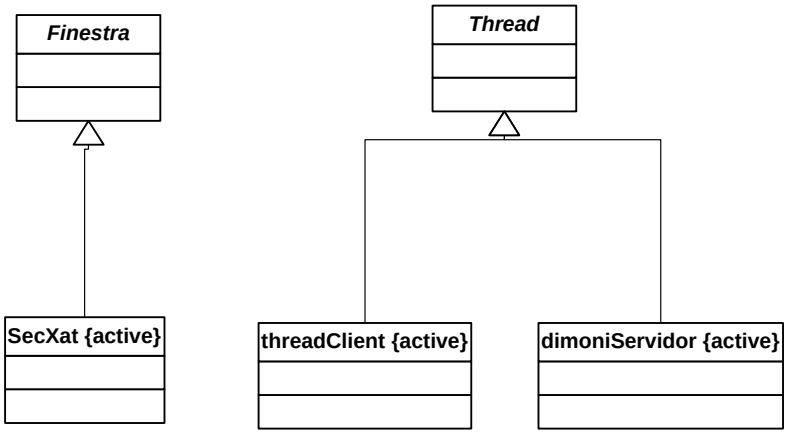
En el moment de la implementació haurem de replantejar-nos algunes de les decisions preses en l'etapa de disseny per tal de compatibilitzar-l'ho amb el llenguatge escollit.
El primer que hem de tenir en compte és que Java no suporta l'herència múltiple, per tant haurem d'adaptar les classes del diagrama UML de disseny per solventar aquesta restricció.

Recordem la secció del diagrama de classes on tenim herència múltiple:



Veiem que la classe frmClient hereta de la classe Finestra i de la classe Thread.

Les herències simples són les següents:



Les classes threadClient i dimoniClient hereten de la classe Thread, i la classe SecXat hereta de la classe Finestra.

Un altre fet a tenir en compte és que el comportament de la classe threadClient dependrà de sí és ell el qui inicia la comunicació o no. Per exemple, el que en l'etapa de disseny hem anomenat de forma general "posar-se d'acord o intercanviar la clau secreta amb el threadClient remot" No serà una tasca totalment simètrica.
El fil d'execució que inicia la conversa haurà de generar els paràmetres, de l'algorisme Diffie-Hellman, necessaris per començar l'intercanvi. Un cop generats enviarà la seva clau pública per la xarxa, i després esperarà la clau pública de l'altra part. Amb aquestes dades crearà la clau secreta.
En canvi, el fil que no inicií la conversa, és a dir el que haurà estat creat pel fil d'execució dimoniServidor, el primer que farà serà esperar la clau pública de l'"iniciador", crearà les seves claus i enviarà la pública. Després crearà la clau secreta.

Cada parell de clients que tinguin una connexió oberta entre ells necessitaran poder intercanviar-se informació relativa al seu estat i a l'estat de la comunicació, a part dels propis missatges introduïts pels usuaris. Per exemple, si un dels clients detecta que s'ha violat la integritat d'un missatge o la seqüencialitat haurà d'informar a l'altre client, perquè aquest avisi a l'usuari i es prenguin les mesures oportunes. El mateix passarà quan un usuari vulgui finalitzar la comunicació, s'haurà d'informar a l'altre usuari.
Evidentment aquests tipus de missatges, que anomenarem missatges de sistema, també hauran de viatjar per la xarxa xifrats. Si això no fos així ens podríem trobar amb atacs de **Denegació de Servei** (DOS). Per exemple, si un atacant aconseguís crear un missatge de finalització de connexió podria "enganyar" als clients i aquests finalitzarien la comunicació.

Per solucionar això adaptarem la classe Missatge que hem definit a l'etapa de disseny per tal que accepti aquest tipus de missatges. Això ho farem afegint un camp "tipus" que ens indicarà si el missatge és d'usuari o de sistema.

Quan un client rebi un missatge haurà de mirar de quin tipus es tracta. Si és un missatge d'usuari simplement el mostrarà per pantalla, i si és de sistema farà les accions necessàries.

4.3 Llibreries utilitzades:

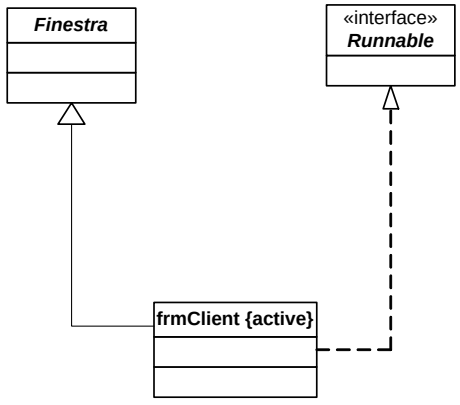
A continuació és descriuen les llibreries, que ens ofereix el llenguatge Java, que farem servir per evitar l'herència múltiple i per implementar els fils d'execució, la comunicació per la xarxa, les interfícies gràfiques i la criptografia.

4.3.1 Solució a l'herència múltiple i els fils d'execució:

Java permet implementar els threads de dues formes diferents:

- Heretant directament de la classe **Thread** de la llibreria **java.lang** i sobrecarregant el mètode run().
- Implementant la interfície **Runnable** de la llibreria **java.lang** i declarant el mètode run(). Posteriorment es crearà un objecte de la classe Thread passant-li al constructor l'objecte de la classe que implementa la interfície Runnable, que haurem creat prèviament.

Així doncs farem servir el primer mètode (heretant de la classe Thread) per les classes threadClient i dimoniServidor.
I el segon mètode (implementant la interfície Runnable) per la classe frmClient.
Així eliminem l'herència múltiple de frmClient, quedant com segueix:



4.3.2 Interfícies gràfiques:

Pel que fa a les interfícies gràfiques utilitzarem el paquet **javax.swing**, que ens ofereix totes les classes necessàries per implementar les finestres, quadres de text, botons, missatges, etc...
Així doncs, la classe que fins ara hem anomenat Finestra serà la classe **JFrame** d'aquest paquet. Aquesta classe serà contenidora de tots els altres objectes que necessitem com ara botons i quadres de text.

4.3.3 Comunicacions:

Per implementar les comunicacions per la xarxa farem servir les classes **Socket** i **ServerSocket** del paquet **java.net**.

La classe ServerSocket implementa els anomenats **sockets de servidor**, els quals escolten per un port determinat i esperen a rebre peticions de connexió a través de la xarxa. Quan reben una connexió creen un objecte Socket i continuen escoltant.

La classe Socket implementa els que s'anomenen sockets de client, i són el punt final de la connexió entre dues màquines.

Per tal de poder enviar els objectes de la classe Missatge per la xarxa, a través dels sockets, aquesta classe implementarà la interfície **Serializable** del paquet **java.io**. Aquesta interfície ens permet llegir i escriure objectes en forma de cadena d'strings i ens facilita així l'intercanvi d'objectes a través dels sockets.

4.3.4 Funcions criptografiques:

Per les funcions criptogràfiques utilitzarem classes dels paquets **java.security**, **javax.crypto** i **java.math**.
Algunes d'aquestes classes seran :

- **AlgorithmParameterGenerator**, **AlgorithmParameters**, **KeyFactory**, **KeyPair**, **KeyPairGenerator**, **PublicKey**, **PrivateKey**, **SecretKey**, **KeyAgreement** i **BigInteger**... per generar i intercanviar la clau simètrica per l'algorisme **DES** amb **Diffie-Hellman**.
- La classe **Cipher** per xifrar i desxifrar amb l'algorisme **DES**.
- La classe **Mac** per calcular el codi MAC amb l'algorisme **SHA-1**.

4.4 Codi Font:

```
// ********************************************************************************
// * Nom Classe: Encriptador                                                      *
// * Descripcio: Encripta/Desencripta amb l'algoritme DES en mode ECB             *
// * Autor     : Eloi Andreu Mas                                                  *
// ********************************************************************************

import javax.crypto.*;
import java.io.*;
import java.lang.*;
import java.security.*;

public class Encriptador
{
	public Encriptador(SecretKey clau)
	{
		// constructor de la classe
		// reb la clau secreta com a paràmetre
		try
		{
			//creem i inicialitzem els objectes Chipher per treballar
			 // amb DES
			encr = Cipher.getInstance("DES");
			desencr = Cipher.getInstance("DES");
			encr.init (Cipher.ENCRYPT_MODE,clau);
			desencr.init (Cipher.DECRYPT_MODE,clau);
		}
		catch (NoSuchAlgorithmException e1){}
		catch (NoSuchPaddingException e2){}
		catch (InvalidKeyException e3){}
	}

	public String encripta(String str)
	{
		// Funció que encripta l'String que rep com a paràmetre
		// i retorna el resultat xifrat
		try
		{
		  // Codifiquem l'String en una cadena de bytes en format UTF8
		   byte[] utf8 = str.getBytes("UTF8");

		  // Encriptem la cadena de bytes
		  byte[] enc = encr.doFinal(utf8);

		   // Codifiquem la cadena de bytes e un String i retornem el
resultat
		  // és a dir, el text encriptat
		   return new sun.misc.BASE64Encoder().encode(enc);
		}
		catch (BadPaddingException e1){}
		catch (IllegalBlockSizeException e2){}
		catch (UnsupportedEncodingException e3){}
		catch (IOException e4){}
		// si hi ha algun error retornem null
		return(null);
	}

	public String desencripta(String str)
	{
		// Funció que desencripta l'String que rep com a paràmetre
		// i retorna el resultat desxifrat
		try
		{
			// Codifiquem l'Sring en una cadena de bytes
```

```
                    byte[] dec = new
sun.misc.BASE64Decoder().decodeBuffer(str);

                    // Desencriptem la cadena de bytes
                    byte[] utf8 = desencr.doFinal(dec);

                     //Codifiquem la cadena de bytes e un String i retornem el
resultat
                     // és a dir, el text desencriptat en fornat UTF8
                    return new String(utf8, "UTF8");
             }
             catch (BadPaddingException e1) {}
             catch (IllegalBlockSizeException e2) {}
             catch (UnsupportedEncodingException e3){}
             catch (IOException e4) {}
             // si hi ha algun error retornem null
             return null;
      }

private Cipher encr;
private Cipher desencr;
}


/**************************************************************************
*******/

// ************************************************************************
// * Nom Classe: Missatge                                                 *
// * Descripcio: Classe que representa els missatges que s'intercanviaran *
// *             els dos clients a través de la xarxa.                    *
// *             implementa la interfície "Serialitzable" per tal de poder*
// *             transmetre'l per la xarxa mitjançant sockets.            *
// * Autor     : Eloi Andreu Mas                                          *
// ************************************************************************

import java.lang.*;
import java.io.*;

class Missatge implements Serializable
{

      public Missatge(String s,int t,String n,byte[] m)
      {
             // constructor de la classe.
             sMissatge=s;
             sTipus=t;
             sNumSerie=n;
             bMAC=m;
      }

      public String getMissatge()
      {
             // retorna el text del missatge
             return sMissatge;
      }

      public int getTipus()
      {
             // retorna el tipus de missatge (sistema o usuari)
             return sTipus;
      }

      public String getNumSerie()
      {
             // retorna el numero de serie
             return sNumSerie;
      }
```

```
	public byte[] getMAC()
	{
		// retorna el codi MAC
		return bMAC;
	}

	static int TIPUS_SISTEMA=0;
	static int TIPUS_USUARI=1;

	private String sMissatge;
	private int sTipus;
	private String sNumSerie;
	private byte[] bMAC;

}


/*****************************************************************************/

// *****************************************************************************
// * Nom Classe: SecXat                                                        *
// * Descripcio: Interficie grafica que permet a l'usuari iniciar el servidor*
// *             i diversos clients                                            *
// * Autor     : Eloi Andreu Mas                                               *
// *****************************************************************************

import javax.swing.*;
import java.net.*;
import java.lang.*;
import java.io.*;

public class SecXat extends javax.swing.JFrame {


    public SecXat()
    {
        // Constructor de la classe SecXat
        initComponents();
    }
    private void initComponents()
   {
        // inicialitza tots els objectes gràfics
        panCentre = new javax.swing.JPanel();
        btnIniciServidor = new javax.swing.JButton();
        btnIniciClient = new javax.swing.JButton();
        jPanel1 = new javax.swing.JPanel();
        txtEstat = new javax.swing.JLabel();

        setTitle("SecXat v1.0");
        setLocationRelativeTo(this);
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
        public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
        }
        });

        panCentre.setBorder(new javax.swing.border.TitledBorder(""));
        btnIniciServidor.setText("Iniciar Servidor");
        btnIniciServidor.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        btnIniciServidor.setPreferredSize(new java.awt.Dimension(100, 25));
        btnIniciServidor.addActionListener(new java.awt.event.ActionListener()
{
        public void actionPerformed(java.awt.event.ActionEvent evt) {
                accioIniciarServidor(evt);
        }
```

```
        });

        panCentre.add(btnIniciServidor);

        btnIniciClient.setText("Nova Connexi\u00f3");
        btnIniciClient.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        btnIniciClient.setPreferredSize(new java.awt.Dimension(100, 25));
        btnIniciClient.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
                accioIniciClient(evt);
        }
        });

        panCentre.add(btnIniciClient);
        getContentPane().add(panCentre, java.awt.BorderLayout.CENTER);
        jPanel1.setBorder(new javax.swing.border.EtchedBorder());
        txtEstat.setFont(new java.awt.Font("Dialog", 0, 10));
        txtEstat.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        txtEstat.setText("Servidor No Iniciat.");
        txtEstat.setVerticalAlignment(javax.swing.SwingConstants.BOTTOM);
        txtEstat.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
        jPanel1.add(txtEstat);

        getContentPane().add(jPanel1, java.awt.BorderLayout.SOUTH);

        pack();
    }

    private void accioIniciarServidor(java.awt.event.ActionEvent evt)
   {
        // Funció que inicia la part servidora. S'executa quan es produeix
       // l'event que l'usuari ha premut el boto corresponent.

         // Comprovem que no hi hagi cap altra instància del servidor
         // executant-se
        if (dimoniRunning==0)
        {
            // Si no s'està executant, creem un fil d'execució dimoniServidor.
            threadServidor = new dimoniServidor(nick);
            threadServidor.start();
            dimoniRunning=1;
            canviaEstat();
        }
        else
            JOptionPane.showMessageDialog(this,"El Servidor ja està
iniciat!","Atenció!",JOptionPane.WARNING_MESSAGE);
    }

    private void accioIniciClient(java.awt.event.ActionEvent evt)
   {
     // Funció que inicia un client. S'executa quan es produeix l'event
     // que l'usuari ha premut el boto corresponent.

        // Demanem a l'usuari amb quina IP vol connectar
        sIP=JOptionPane.showInputDialog(this,"Introdueix la IP:","SecXat
v1.0",JOptionPane.QUESTION_MESSAGE);
        if (sIP!=null && sIP.length()!=0)
        {
            try
            {
                // Creem un socket i el connectem amb la IP
                sock = new Socket(sIP,9999);
            }
            catch (UnknownHostException e1)
            {
                JOptionPane.showMessageDialog(this,"Error: No s'ha pogut
establir la connexió amb: " + sIP,"Error!",JOptionPane.ERROR_MESSAGE);
```

```
            }
            catch (IOException e2)
            {
                JOptionPane.showMessageDialog(this,"Error: No s'ha pogut
establir la connexió amb: " + sIP,"Error!",JOptionPane.ERROR_MESSAGE);
            }

            if (sock!=null)
            {
               // Es crea i s'inicia un fil d'execució threadClient
               threadClient unClient = new threadClient(sock,nick,this);
               Thread thClient = new Thread(unClient);
               thClient.start();
            }
            else
            {
               canviaEstat();
            }
         }
    }

    public void canviaEstat(String s)
    {
        this.txtEstat.setText(s);
    }

    public void canviaEstat()
    {
      String s;
      if (dimoniRunning==1)
      {
         s="Servidor Iniciat.";
      }
      else
      {
         s="Servidor No Iniciat.";
      }
      this.txtEstat.setText(s);
    }


    private void exitForm(java.awt.event.WindowEvent evt)
    {
        System.exit(0);
    }

    public static void main(String args[])
   {
      // funció principal de la classe.

      new SecXat().show();
      // demanem el nick que es farà servir per aquesta sessió.
      nick=JOptionPane.showInputDialog(null,"Introdueix el teu nick:","SecXat
v1.0",JOptionPane.QUESTION_MESSAGE);
      if ((nick==null) || nick.length()==0)
      {
         try
         {
            // Agafem la IP de la màquina local
            nick=InetAddress.getLocalHost().getHostAddress();
         }
         catch (IOException e)
         {
                JOptionPane.showMessageDialog(null,"Error: No s'ha pogut
obtenir l'adreça IP","Error!",JOptionPane.ERROR_MESSAGE);
         }
      }
   }
```

```
    // Declaració de Variables
    private javax.swing.JButton btnIniciClient;
    private javax.swing.JButton btnIniciServidor;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel panCentre;
    private javax.swing.JLabel txtEstat;
    private String sIP= new String("");
    private ServerSocket srvSocket;
    private Socket sock;
    private int dimoniRunning=0;
    private dimoniServidor threadServidor;
    private static String nick;
}



/**********************************************************************/

// ************************************************************************
// * Nom Classe: dimoniServidor                                           *
// * Descripcio: Fil d'execució que implementa la part servidora de       *
// *             l'aplicació. Espera connexions de clients i per cada una *
// *             crea una instància de threadClient                       *
// *             Hereta de la classe Thread                               *
// * Autor     : Eloi Andreu Mas                                          *
// ************************************************************************

import java.lang.*;
import java.io.*;
import java.net.*;
import javax.swing.*;

public class dimoniServidor extends Thread
{

   public dimoniServidor(String n)
   {
      // Constructor de la classe. Rep el nick de l'usuari com a paràmetre
      nick=n;
      setDaemon(true);
   }

   public void run()
   {
      EsperaClients();
   }

   private void Aturar()
   {
      try
      {
         // tanquem el socket
         srvSocket.close();
      }
      catch (IOException e)
      {
         JOptionPane.showMessageDialog(null,"Error desconnectant el Servidor.
" +
e.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);
      }
   }

   private void EsperaClients()
   {
      // funció que espera les connexions dels clients
      try
      {
```

```
         // es crea el socket de servidor
         srvSocket = new ServerSocket(9999);
         Socket sock=null;
         while (true)
         {
            // esperem que algun client inicii una connexió. Quan això passa
            // es crea un socket de client.
            sock=srvSocket.accept();

            // es crea i inicia un fil d'execució threadClient i se li passa
           // el socket de client
            // que s'ha creat i el nick com a paràmetres.
            threadClient SClient = new threadClient(sock,nick);
            Thread thClient = new Thread(SClient);
            thClient.start();
         }
      }
      catch (IOException e)
      {
         JOptionPane.showMessageDialog(null,"Error en el servidor. " +
e.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);
      }
   }

private ServerSocket srvSocket;
private BufferedReader buffIn;
private PrintWriter buffOut;
private String nick;
}


/*********************************************************************************/


// ********************************************************************************
// * Nom Classe: frmClient                                                        *
// * Descripcio: Implementa la interfície gràfica del client del xat.             *
// *             Esperarà que l'usuari introdueixi algun missatge i li            *
// *             passarà al threadClient que l'ha creat perquè aquest l'envii*
// *             per la xarxa. També mostrarà a l'usuari els missatges que        *
// *             rebi de threadClient                                             *
// * Autor     : Eloi Andreu Mas                                                  *
// ********************************************************************************

import javax.swing.*;
import java.awt.*;
import java.lang.*;
import java.io.*;
import java.util.*;
import java.text.*;

public class frmClient extends javax.swing.JFrame implements Runnable{
// hereta de la classe JFrame i implementa la interfície "Runnable"
// per tal que es pugui executar com un thread.

   public frmClient(threadClient c)
   {
      // Constructor de la classe.
      thCli=c;
      initComponents();
   }
   private void initComponents()
   {
        // s'inicialitzen els components gràfics
        panCapcalera = new javax.swing.JPanel();
        lTitol = new javax.swing.JLabel();
        panCentre = new javax.swing.JPanel();
        lIntrodueix = new javax.swing.JLabel();
```

```
        txtMissatgeEnv = new javax.swing.JTextField();
        spScroll = new javax.swing.JScrollPane();
        txtMissatgeRev = new javax.swing.JTextArea();
        txtEstat = new javax.swing.JLabel();
        panPeu = new javax.swing.JPanel();
        btnEnviar = new javax.swing.JButton();
        btnSortir = new javax.swing.JButton();
        mnuMenu1 = new javax.swing.JMenuBar();
        mnuArxiu = new javax.swing.JMenu();
        mnuGuardar = new javax.swing.JMenuItem();
        jSeparator1 = new javax.swing.JSeparator();
        Sortir = new javax.swing.JMenuItem();
        mnuConnexió = new javax.swing.JMenu();
        Desconnectar = new javax.swing.JMenuItem();

        setTitle("SecXat v1.0");
        setLocationRelativeTo(this);
        setMaximizedBounds(new java.awt.Rectangle(0, 0, 5000, 5000));
        setName("frmClient");
        setResizable(false);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowClosing(java.awt.event.WindowEvent evt) {
                exitForm(evt);
            }
        });

        panCapcalera.setLayout(new
java.awt.FlowLayout(java.awt.FlowLayout.CENTER, 2, 2));

        panCapcalera.setBackground(new java.awt.Color(142, 182, 200));
        panCapcalera.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        lTitol.setFont(new java.awt.Font("Century Gothic", 1, 20));
        lTitol.setForeground(new java.awt.Color(0, 0, 255));
        lTitol.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        lTitol.setLabelFor(panCapcalera);
        lTitol.setText("SecXat v1.0");
        lTitol.setVerticalAlignment(javax.swing.SwingConstants.TOP);
        lTitol.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        panCapcalera.add(lTitol);

        getContentPane().add(panCapcalera, java.awt.BorderLayout.NORTH);

        panCentre.setLayout(new
java.awt.FlowLayout(java.awt.FlowLayout.CENTER, 5, 20));

        panCentre.setBackground(new java.awt.Color(218, 218, 230));
        panCentre.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        panCentre.setMaximumSize(new java.awt.Dimension(30000, 30000));
        panCentre.setAutoscrolls(true);
        lIntrodueix.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        lIntrodueix.setText("Introdueix el missatge");
        lIntrodueix.setVerticalAlignment(javax.swing.SwingConstants.TOP);
        panCentre.add(lIntrodueix);

        txtMissatgeEnv.setColumns(25);
        txtMissatgeEnv.setFont(new java.awt.Font("Dialog", 1, 12));
        txtMissatgeEnv.setHorizontalAlignment(javax.swing.JTextField.LEFT);
        txtMissatgeEnv.setBorder(new javax.swing.border.LineBorder(new
java.awt.Color(51, 0, 204)));
        txtMissatgeEnv.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                enviarMissatge(evt);
            }
        });

        panCentre.add(txtMissatgeEnv);
```

```
        spScroll.setPreferredSize(new java.awt.Dimension(405, 200));
        spScroll.setAutoscrolls(true);
        txtMissatgeRev.setBackground(new java.awt.Color(204, 204, 204));
        txtMissatgeRev.setColumns(37);
        txtMissatgeRev.setEditable(false);
        txtMissatgeRev.setLineWrap(true);
        txtMissatgeRev.setRows(10);
        txtMissatgeRev.setWrapStyleWord(true);
        txtMissatgeRev.setAlignmentX(1.0F);
        txtMissatgeRev.setBorder(new javax.swing.border.LineBorder(new
java.awt.Color(0, 0, 0)));
        spScroll.setViewportView(txtMissatgeRev);

        panCentre.add(spScroll);

        txtEstat.setHorizontalAlignment(javax.swing.SwingConstants.LEFT);
        txtEstat.setText("Estat: Connectat.");
        txtEstat.setVerticalAlignment(javax.swing.SwingConstants.BOTTOM);
        txtEstat.setPreferredSize(new java.awt.Dimension(425, 25));
        txtEstat.setHorizontalTextPosition(javax.swing.SwingConstants.LEFT);
        txtEstat.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        panCentre.add(txtEstat);

        getContentPane().add(panCentre, java.awt.BorderLayout.CENTER);

        panPeu.setLayout(new java.awt.FlowLayout(java.awt.FlowLayout.CENTER,
10, 5));

        panPeu.setBackground(new java.awt.Color(218, 218, 230));
        panPeu.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        btnEnviar.setText("Enviar");
        btnEnviar.setVerticalAlignment(javax.swing.SwingConstants.TOP);
        btnEnviar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                enviarMissatge(evt);
            }
        });

        panPeu.add(btnEnviar);

        btnSortir.setMnemonic('s');
        btnSortir.setText("Sortir");
        btnSortir.setVerticalAlignment(javax.swing.SwingConstants.TOP);
        btnSortir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                preguntaSortir(evt);
            }
        });

        panPeu.add(btnSortir);

        getContentPane().add(panPeu, java.awt.BorderLayout.SOUTH);

        mnuMenu1.setBackground(new java.awt.Color(204, 204, 255));
        mnuMenu1.setBorder(new
javax.swing.border.BevelBorder(javax.swing.border.BevelBorder.RAISED));
        mnuArxiu.setBackground(new java.awt.Color(204, 204, 255));
        mnuArxiu.setMnemonic('c');
        mnuArxiu.setText("Arxiu");
        mnuArxiu.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                mnuArxiuActionPerformed(evt);
            }
        });

        mnuGuardar.setBackground(new java.awt.Color(204, 204, 255));
```

```
        mnuGuardar.setText("Guardar Conversa...");
        mnuGuardar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                accioGuardar(evt);
            }
        });

        mnuArxiu.add(mnuGuardar);

        mnuArxiu.add(jSeparator1);

        Sortir.setBackground(new java.awt.Color(204, 204, 255));
        Sortir.setText("Sortir");
        Sortir.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                preguntaSortir(evt);
            }
        });

        mnuArxiu.add(Sortir);

        mnuMenu1.add(mnuArxiu);

        mnuConnexió.setBackground(new java.awt.Color(204, 204, 255));
        mnuConnexió.setText("Connexi\u00f3");
        Desconnectar.setBackground(new java.awt.Color(204, 204, 255));
        Desconnectar.setText("Desconnectar");
        Desconnectar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                accioDesconnectar(evt);
            }
        });

        mnuConnexió.add(Desconnectar);

        mnuMenu1.add(mnuConnexió);

        setJMenuBar(mnuMenu1);

        java.awt.Dimension screenSize =
java.awt.Toolkit.getDefaultToolkit().getScreenSize();
        setBounds((screenSize.width-440)/2, (screenSize.height-436)/2, 440,
436);
    }

    private void accioDesconnectar(java.awt.event.ActionEvent evt)
   {
      // Operació que envia un missatge de sistema de Desconnexió, i
      // invoca la operació tancaSocket() del seu thread pare threadClient.
         enviarMissatgeSist("F");
         thCli.tancaSocket();
    }

    private void accioGuardar(java.awt.event.ActionEvent evt)
   {
      // Funció que mostra a l'usuari un diàleg demanant un nom de fitxer.
      // Després guarda la conversa mantinguda fins al moment en aquest
      // fitxer.
        try
        {
            FileDialog fd = new FileDialog(this,"Guardar
com...",FileDialog.SAVE);
            fd.show();
            String sFitx= fd.getDirectory() + fd.getFile();
            if (sFitx.compareTo("nullnull")!=0)
            {
                FileWriter fitx = new FileWriter(sFitx);
                fitx.write(txtMissatgeRev.getText());
```

```
                fitx.close();
                JOptionPane.showMessageDialog(null,"Conversa guardada
correctament.","Ok",JOptionPane.INFORMATION_MESSAGE);
            }
        }
        catch (IOException e)
        {
            JOptionPane.showMessageDialog(null,"Error guardant la
conversa.","Error!",JOptionPane.ERROR_MESSAGE);
        }

    }

    private void mnuArxiuActionPerformed(java.awt.event.ActionEvent evt)
    {
    }


   public void rebreMissatge(String miss)
   {
      // reb un missatge i el mostra a l'usuari
      this.txtMissatgeRev.append("\n" + miss);
      this.txtMissatgeRev.setText(this.txtMissatgeRev.getText());
      this.setState(JFrame.NORMAL);
   }


    private void enviarMissatge(java.awt.event.ActionEvent evt)
   {
        // agafa el missatge introduit per l'usuari i crida la operació
        // enviarMissatge(m) de l'objecte threadClient, passant-li el missatge
        // com a paràmetre.
        if (estat==ESTAT_CONNECTAT)
        {
            String m = txtMissatgeEnv.getText();
            if (m.length()!=0)
            {
                txtMissatgeEnv.setText("");
                txtMissatgeRev.append("\n" + thCli.getNick() + ": " + m);
                this.txtMissatgeRev.setText(this.txtMissatgeRev.getText());
                thCli.enviarMissatge(m);
             }
        }
    }

    private void enviarMissatgeSist(String m)
   {
        // envia un missatge de sistema al threadClient corresponent
        if (estat==ESTAT_CONNECTAT)
        {
            if (m!=null && m.length()!=0)
            {
                thCli.enviarMissatgeSist(m);
            }
        }
    }

    private void preguntaSortir(java.awt.event.ActionEvent evt)
   {
        // Demana confirmació a l'usuari per tancar el client
        int res=JOptionPane.showConfirmDialog(null,"Segur que vols sortir
?","Sortir",JOptionPane.YES_NO_OPTION);
        if (res==JOptionPane.YES_OPTION)
        {
            accioSortir(null);
        }

    }
```

```
    public void canviaEstat(int e)
    {
        // funció que ens permet actualitzar la linia d'estat del formulari
        estat=e;
        if (estat==ESTAT_CONNECTAT)
        {
            txtEstat.setText("Estat: Connectat");
        }
        else
        {
            txtEstat.setText("Estat: Desconnectat");
        }
    }

    private  void sortir()
    {
        this.dispose();
    }

   private  void accioSortir(java.awt.event.WindowEvent evt)
    {
        thCli.tancaSocket();
        sortir();
    }


    private void exitForm(java.awt.event.WindowEvent evt)
    {
        accioSortir(evt);
    }

    public void run()
    {
        setTitle("Client SecXat v1.0");
        show();
        Date d = new Date();
        SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");
        txtMissatgeRev.append("Conversa iniciada el " + df.format(d) +"\n-----
----------------------------------------------\n");
    }


    private javax.swing.JMenuItem Desconnectar;
    private javax.swing.JMenuItem Sortir;
    private javax.swing.JButton btnEnviar;
    private javax.swing.JButton btnSortir;
    private javax.swing.JSeparator jSeparator1;
    private javax.swing.JLabel lIntrodueix;
    private javax.swing.JLabel lTitol;
    private javax.swing.JMenu mnuArxiu;
    private javax.swing.JMenu mnuConnexió;
    private javax.swing.JMenuItem mnuGuardar;
    private javax.swing.JMenuBar mnuMenu1;
    private javax.swing.JPanel panCapcalera;
    private javax.swing.JPanel panCentre;
    private javax.swing.JPanel panPeu;
    private javax.swing.JScrollPane spScroll;
    private javax.swing.JLabel txtEstat;
    private javax.swing.JTextField txtMissatgeEnv;
    private javax.swing.JTextArea txtMissatgeRev;

    private threadClient thCli;
    private static int ESTAT_CONNECTAT=1;
    private static int ESTAT_DESCONNECTAT=0;
    private int estat;
}
```

```
/*****************************************************************************/


// ****************************************************************************
// * Nom Classe: threadClient                                                 *
// * Descripcio: Thread que s'encarrega de la comunicació entre clients.      *
// *             Es pot executar en mode "iniciador" o "no iniciador"         *
// *             També s'encarrega de fer tot el control relacionat amb       *
// *             la seguretat: xifrar/desxifrar, calcular MAC...              *
// * Autor     : Eloi Andreu Mas                                              *
// ****************************************************************************

import java.lang.*;
import java.io.*;
import javax.swing.*;
import java.net.*;
import javax.crypto.*;
import javax.crypto.spec.*;
import javax.crypto.interfaces.*;
import java.security.*;
import java.security.spec.*;
import javax.crypto.interfaces.*;
import java.math.*;
import java.text.*;
import java.util.*;

public class threadClient extends Thread
{
   // La classe threadClient hereta de la classe Thread

   public threadClient(Socket s, String n)
   {
      // Constructor de la classe. Crea un threadClient dels que no
      // inicien la conversa, o sigui, l'executa dimoniServidor.
      // Rep un socket i el nick (string) com a paràmetres

      sTipus="NO_INICIA_CONVERSA";
      sock=s;
      nick=n;
      //setDaemon(true);
   }

   public threadClient(Socket s, String n, SecXat fSrv)
   {
      // Constructor de la classe. Crea un threadClient dels que
      // inicien la conversa, o sigui, l'executa SecXat.
      // Rep un socket, el nick (string) i l'objecte SecXat que l'a creat
      // com a paràmetres

      sTipus="INICIA_CONVERSA";
      sock=s;
      nick=n;
      frmSrv=fSrv;
   }

   private void generaParametresDH()
   {
      // Funció que genera els paràmetres necessàris per realitzar
      // l'intercanvi de claus amb l'algorisme Diffie-Hellman
      try
      {
         // Creem un generador de parametres DH
         AlgorithmParameterGenerator paramGen =
AlgorithmParameterGenerator.getInstance("DH");
         paramGen.init(1024);

         // Generem els parametres DH
```

```
         AlgorithmParameters params = paramGen.generateParameters();
         DHParameterSpec DHParamSpec =
(DHParameterSpec)params.getParameterSpec(DHParameterSpec.class);

         p=DHParamSpec.getP();
         g=DHParamSpec.getG();
         l=DHParamSpec.getL();
      }
      catch (NoSuchAlgorithmException e1){}
      catch (InvalidParameterSpecException e2){}
   }


   private void generaClau()
   {
      // Funció que genera la clau secreta que es farà servir per
      // xifrar i desxifrar amb l'algorisme DES.
      try
      {
         byte[] bytesClauPublica2=new byte[1024];
         PrivateKey clauPrivada;
         PublicKey clauPublica2;
         PublicKey clauPublica;

         if (!sTipus.equals("INICIA_CONVERSA"))
         {
            // Si no hem iniciat la conversa, esperem la clau publica de
            // l'altre part
            sock.getInputStream().read(bytesClauPublica2);

            // Generem un parell de claus DH
            X509EncodedKeySpec x509KeySpec = new
X509EncodedKeySpec(bytesClauPublica2);
            KeyFactory keyFac = KeyFactory.getInstance("DH");
            clauPublica2=keyFac.generatePublic(x509KeySpec);

            KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DH");
            DHParameterSpec DHParamSpec =
((DHPublicKey)clauPublica2).getParams();
            keyGen.initialize(DHParamSpec);
            KeyPair keypair  = keyGen.generateKeyPair();

            clauPrivada = keypair.getPrivate();
            clauPublica = keypair.getPublic();

            // enviem la nostra clau publica
            byte[] bytesClauPublica = clauPublica.getEncoded();
            sock.getOutputStream().write(bytesClauPublica);
         }
         else
         {
            // Si hem iniciat la conversa, generem el parell de claus amb els
            // paràmetres DH calculats abans.
            KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DH");
            DHParameterSpec DHParamSpec = new DHParameterSpec(p,g,l);
            keyGen.initialize(DHParamSpec);
            KeyPair keypair  = keyGen.generateKeyPair();

            clauPrivada = keypair.getPrivate();
            clauPublica = keypair.getPublic();
            byte[] bytesClauPublica = clauPublica.getEncoded();
            frmSrv.canviaEstat();

            // enviem la nostra clau publica
            sock.getOutputStream().write(bytesClauPublica);

            // esperem la clau publica de l'altre part
            sock.getInputStream().read(bytesClauPublica2);
```

```
         }

         X509EncodedKeySpec x509KeySpec = new
X509EncodedKeySpec(bytesClauPublica2);
         KeyFactory keyFac = KeyFactory.getInstance("DH");
         clauPublica2=keyFac.generatePublic(x509KeySpec);

         // generem la clau secreta (simètrica) DES amb la nostra clau privada
         // i la clau publica de l'altre part
         KeyAgreement ka = KeyAgreement.getInstance("DH");
         ka.init(clauPrivada);
         ka.doPhase(clauPublica2,true);
         clau = ka.generateSecret("DES");
      }
      catch (InvalidKeyException e1)
{JOptionPane.showMessageDialog(null,"Error generant la clau secreta. " +
e1.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);}
      catch (InvalidKeySpecException e2)
{JOptionPane.showMessageDialog(null,"Error generant la clau secreta. " +
e2.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);}
      catch (InvalidAlgorithmParameterException e3)
{JOptionPane.showMessageDialog(null,"Error generant la clau secreta. " +
e3.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);}
      catch (NoSuchAlgorithmException e4)
{JOptionPane.showMessageDialog(null,"Error generant la clau secreta. " +
e4.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);}
      catch (IOException e5){JOptionPane.showMessageDialog(null,"Error
generant la clau secreta. " +
e5.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);}
   }

   public void enviarMissatge(String miss)
   {
      // Envia un missatge d'usuari xifrat (que es rep com a paràmetre),
      // amb el número de sèrie corresponent i el codi MAC

      if (estat==ESTAT_CONNECTAT)
      {
         try
         {
            // s'actualitza i es xifra el número de sèrie
            DecimalFormat df = new DecimalFormat("000");
            numEnviats=(numEnviats+1) % 999;
            String seq = encripta.encripta(df.format(numEnviats));

            // es xifra el text del missatge
            String m=encripta.encripta(nick + ": " + miss);

            // Calculem el codi MAC amb SHA-1
            Mac mac;
            mac = Mac.getInstance("HmacSHA1");
            mac.init(clau);
            String tmp = nick + ": " + miss + Missatge.TIPUS_USUARI +
df.format(numEnviats);
            byte[] bMAC = mac.doFinal(tmp.getBytes());

            // es crea un objecte Missatge amb els valors calculats
            Missatge missatge = new
Missatge(m,Missatge.TIPUS_USUARI,seq,bMAC);

            // s'envia l'objecte missatge, és a dir, s'escriu al sócket
            buffOut2.writeObject(missatge);
         }
         catch (NoSuchAlgorithmException e1){}
         catch (InvalidKeyException e2){}
         catch (Exception e)
         {
```

```
            JOptionPane.showMessageDialog(null,"Error enviant el missatge. " +
e.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);
         }
      }
   }

   public void enviarMissatgeSist(String miss)
   {
      // Envia un missatge de sistema xifrat (que es rep com a paràmetre),
      // amb el número de sèrie corresponent i el codi MAC

      if (estat==ESTAT_CONNECTAT)
      {
         try
         {
            // s'actualitza i es xifra el número de sèrie
            DecimalFormat df = new DecimalFormat("000");
            numEnviats=(numEnviats+1) % 999;
            String seq = encripta.encripta(df.format(numEnviats));

            // es xifra el text del missatge
            String m=encripta.encripta(miss);

            // Calculem el codi MAC amb SHA-1
            Mac mac;
            mac = Mac.getInstance("HmacSHA1");
            mac.init(clau);
            String tmp = miss + Missatge.TIPUS_SISTEMA +
df.format(numEnviats);
            byte[] bMAC = mac.doFinal(tmp.getBytes());

            // es crea un objecte Missatge amb els valors adequats
            Missatge missatge = new
Missatge(m,Missatge.TIPUS_SISTEMA,seq,bMAC);

            // s'envia l'objecte missatge, és a dir, s'escriu al sócket
            buffOut2.writeObject(missatge);
         }
         catch (NoSuchAlgorithmException e1){}
         catch (InvalidKeyException e2){}
         catch (Exception e)
         {
            JOptionPane.showMessageDialog(null,"Error enviant el missatge. " +
e.getMessage(),"Error!",JOptionPane.ERROR_MESSAGE);
         }
      }
   }


   private boolean comprovaNumSeq(String s)
   {
      // Funcio que comprova si el número de sèrie que reb com a paràmetre
      // és correcte. És a dir, si els missatges han arribat amb l'ordre
      // en que van ser enviats

      numRebuts=(numRebuts+1) % 999;
      DecimalFormat df = new DecimalFormat("000");

      if (s.compareTo(df.format(numRebuts))==0)
      {
         return(true);
      }
      else
      {
         return(false);
      }
   }
```

```
   private boolean comprovaMAC (String m, int t,String s,byte[] bMAC_remot)
   {
      // Funció que comprova si el codi MAC que s'ha rebut amb el missatge és
      // correcte
      Mac MAC_local;
      byte[] bMAC_local;
      boolean b;
      try
      {
         // Calculem el MAC del missatge en funció del text, el tipus i el num
         // de seq.
         MAC_local = Mac.getInstance("HmacSHA1");
         MAC_local.init(clau);
         String tmp = m + t + s;
         bMAC_local = MAC_local.doFinal(tmp.getBytes());

         // Comparem el MAC calculat amb el rebut i retornem el resultat
         b=Arrays.equals(bMAC_local, bMAC_remot);
         return(b);

      }
      catch (NoSuchAlgorithmException e){}
      catch (InvalidKeyException e2){}

      return(false);
   }


   public void tancaSocket()
   {
      // Enviem un missatge de sistema de "Fi de Connexió"
      enviarMissatgeSist("F");
      fi=1;
      estat=ESTAT_DESCONNECTAT;
      SClient.canviaEstat(ESTAT_DESCONNECTAT);
      try
      {
         //tanquem el socket
         sock.close();
      }
      catch (IOException e){}
   }

   public void run()
   {
      try
      {
         buffOut2 = new ObjectOutputStream (sock.getOutputStream());
         buffIn2 = new ObjectInputStream (sock.getInputStream());
      }
      catch (IOException e)
      {
         JOptionPane.showMessageDialog(SClient,"Error establint la
connexió.","Error!",JOptionPane.ERROR_MESSAGE);
         fi=1;
      }

      if (sTipus.equals("INICIA_CONVERSA"))
      {
         frmSrv.canviaEstat("Iniciant Client. Esperi...");
         // Si estem en mode "inicia_conversa" es generen els paràmetres DH
         generaParametresDH();
      }

      // cridem la operació que genera la clau secreta
      generaClau();
```

```
      // creem un objecte Encriptador per xifrar i desxifrar
      encripta = new Encriptador(clau);

      estat=ESTAT_CONNECTAT;
      if (sTipus.equals("INICIA_CONVERSA")) {enviarMissatgeSist("I");}

      // Creem un thread pel formulari del client
      SClient = new frmClient(this);
      thfrmClient = new Thread(SClient);
      thfrmClient.setDaemon(true);
      thfrmClient.start();
      SClient.canviaEstat(ESTAT_CONNECTAT);

      String missatge_des="";

      // creem un objecte Missatge pels missatges rebuts
      Missatge miss;

      while (fi==0)
      {
         try
         {
            // espera que arribi algun missatge pel socket
            miss = (Missatge)buffIn2.readObject();
         }
         catch (IOException e1)
         {
            if (fi==0)
            {
               JOptionPane.showMessageDialog(SClient,"Error en la connexió.
S'ha perdut la comunicació","Error!",JOptionPane.ERROR_MESSAGE);
               estat=ESTAT_DESCONNECTAT;
               SClient.canviaEstat(ESTAT_DESCONNECTAT);
               miss=null;
               fi=1;
            }
            break;
         }
         catch (ClassNotFoundException e2)
         {
            JOptionPane.showMessageDialog(SClient,"Error en la connexió. Les
dades rebudes són incorrectes","Error!",JOptionPane.ERROR_MESSAGE);
            enviarMissatgeSist("E");
            miss=null;
            break;
         }

         // quan arriba un missatge es desencripta i se'n extreu cada una de
         // les parts
         missatge_des=encripta.desencripta(miss.getMissatge());
         String s=encripta.desencripta(miss.getNumSerie());
         if (missatge_des==null || s==null)
         {
            // Si el missatge no s'ha pogut desxifrar correctament
            JOptionPane.showMessageDialog(SClient,"Error en la connexió. No
s'ha pogut desxifrar l'ultim missatge
rebut","Error!",JOptionPane.ERROR_MESSAGE);
            numRebuts=(numRebuts+1) % 999;
            enviarMissatgeSist("E");
         }
         else
         {
            // Comprovem el número de seqüència
            if (comprovaNumSeq(s))
            {
               // Comprovem el codi MAC
               if (comprovaMAC(missatge_des,miss.getTipus(),s,miss.getMAC()))
               {
```

```
                    // Mirem quin tipus de missatge ha arribat (usuari o
                    // sistema)
                    if (miss.getTipus()==Missatge.TIPUS_USUARI)
                    {
                       // Si es un missatge de sistema, cridem la operació
                       // rebreMissatge
                       // de SClient (la instància de frmClient) el qual
                       // mostrarà el miss a l'usuari
                       SClient.rebreMissatge(missatge_des);
                    }
                    else
                    {
                       // Si es tracta d'un missatge de sistema l'analitzem
                       switch (missatge_des.charAt(0))
                       {
                       case 'S' :   // Si hi ha hagut un error en la seqüència
                                    // d'enciament dels missatges
                                 JOptionPane.showMessageDialog(SClient,"Error.
L'ordre d'enviament dels missatges ha estat alterat. Es desconnectarà la
sessió.","Error!",JOptionPane.ERROR_MESSAGE);
                       case 'F' :  // Si s'ha finalitzat la connexió
                                 JOptionPane.showMessageDialog(SClient,"S'ha
finalitzat la connexió.","SecXat v1.0",JOptionPane.INFORMATION_MESSAGE);
                                 estat=ESTAT_DESCONNECTAT;
                                 SClient.canviaEstat(ESTAT_DESCONNECTAT);
                                 fi=1;
                                 try
                                 {
                                    sock.close();
                                 }
                                 catch (IOException e4){}
                                 break;
                       case 'N' :  // Si l'usuari remot no ha acceptat la
                                    // petició de connexió
                                 JOptionPane.showMessageDialog(SClient,"L'usuari
remot no ha acceptat la connexió","SecXat
v1.0",JOptionPane.INFORMATION_MESSAGE);
                                 estat=ESTAT_DESCONNECTAT;
                                 SClient.canviaEstat(ESTAT_DESCONNECTAT);
                                 fi=1;
                                 try
                                 {
                                    sock.close();
                                 }
                                 catch (IOException e4){}
                                 break;

                       case 'I' :  // Si s'ha rebut una petició de connexió
                                 int
res=JOptionPane.showConfirmDialog(SClient,"S'ha rebut una petició de connexió
de la IP: " + sock.getInetAddress().getHostAddress() + " Acceptar ?","Petició
de Connexió",JOptionPane.YES_NO_OPTION);
                                 if (res==JOptionPane.NO_OPTION)
                                 {
                                    enviarMissatgeSist("N");
                                    estat=ESTAT_DESCONNECTAT;
                                    SClient.canviaEstat(ESTAT_DESCONNECTAT);
                                    fi=1;
                                    try
                                    {
                                       sock.close();
                                    }
                                    catch (IOException e4){}
                                 }
                                 break;
                       case 'E' :   // Si hi ha hagut algun error en l'ultim
missatge enviat
```

```
                                JOptionPane.showMessageDialog(SClient,"Error:
L'ultim missatge enviat no ha arribat al seu
destí.","Error!",JOptionPane.ERROR_MESSAGE);
                                break;

                        case 'M' :   // Si el destinatari de l'ultim missatge
                                     // enviat no ha pogut verificar-ne la
                                     // integritat
                                JOptionPane.showMessageDialog(SClient,"Error: El
destinatari no ha pogut verificar la integritat de l'ultim missatge
enviat.","Error",JOptionPane.ERROR_MESSAGE);
                                break;
                        default :   JOptionPane.showMessageDialog(SClient,"Error:
S'ha rebut un missatge de sistema no
esperat.","Error",JOptionPane.ERROR_MESSAGE);
                        }
                    }
                }
                else
                {
                    // Si no es pot verificar la integritat del missatge rebut.
                    JOptionPane.showMessageDialog(SClient,"Error. No s'ha pogut
verificar la integritat del missatge
rebut.","Error!",JOptionPane.ERROR_MESSAGE);
                    // enviem missatge de sistema (Error MAC)
                    enviarMissatgeSist("M");
                }
            }
            else
            {
                // Si l'ordre d'arribada s'ha alterat
                JOptionPane.showMessageDialog(SClient,"Error. L'ordre
d'arribada dels missatges ha estat alterat. Es desconnectarà la
sessió.","Error!",JOptionPane.ERROR_MESSAGE);
                // enviem missatge de sistema (Error SEQ)
                enviarMissatgeSist("S");
                estat=ESTAT_DESCONNECTAT;
                SClient.canviaEstat(ESTAT_DESCONNECTAT);
                fi=1;
                try
                {
                    sock.close();
                }
                catch (IOException e5){}
                break;
            }
        }
      }
      try
      {
         // esperem a que el fil d'execució frmClient acabi. És a dir, que
         // l'usuari finalitzi el client.
         thfrmClient.join();

      }
      catch (InterruptedException e6){}
   }

   public String getNick()
   {
      return (nick);
   }


static int ESTAT_CONNECTAT=1;
static int ESTAT_DESCONNECTAT=0;

private Socket sock;
```

```
private ObjectInputStream buffIn2;
private ObjectOutputStream buffOut2;
private Thread thfrmClient;
private String sTipus;
private SecretKey clau;
private BigInteger p;
private BigInteger g;
private int l;
private int fi=0;
private String nick;
private frmClient SClient;
private Encriptador encripta;
private SecXat frmSrv;
private int estat;
private int numEnviats=0;
private int numRebuts=0;

}
```

5.- Conclusions

Durant el procés d'elaboració d'aquest projecte hem fet un estudi de les tècniques criptogràfiques disponibles, avaluant-ne els pros i contres, i hem aplicat les més adequades per aconseguir l'objectiu establert: un sistema de missatgeria instantània, entre dos usuaris remots, amb comunicacions segures.

Per fer-ho hem analitzat a fons el problema de la seguretat, determinant en primer lloc els possibles atacs a què es poden veure sotmeses les dades. I després hem analitzat les possibles solucions.

Per dur a terme aquest treball, s'ha passat per diverses fases, des de la planificació fins a la implementació, passant per una etapa de disseny; totes elles de gran importància.

Un pla de treball o planificació del projecte ben plantejada i realista ajuda molt en totes les etapes posteriors, fent notar ràpidament qualsevol desviació imprevista, de manera que es pot buscar una solució abans que sigui massa tard. A més, una bona planificació és imprescindible si s'ha de valorar el cost del projecte, tant econòmicament com en qüestió de recursos necessaris o de fites temporals que s'han de complir.

Pel que fa al disseny i a la Implementació, hem après a construir un programari partint de zero, la qual cosa significa que primer hem hagut d'analitzar els requisits que ha de complir i dissenyar una solució que els satisfaci. Una vegada tenim el disseny complet passem a la fase d'implementació, on hem après el funcionament d'algunes de les eines (llenguatges de programació, llibreries...) de que disposem actualment.

Així doncs, aquest projecte, que a priori pot semblar que només consta d'una "simple" aplicació de missatgeria, ens dóna els coneixements necessaris per poder realitzar projectes molt més complexos, ja que la metodologia emprada serveix per tot tipus de desenvolupaments.
A part de la pràctica en la planificació i el disseny, també hem assolit uns grans coneixements tècnics pel que fa a les comunicacions, i més concretament a les comunicacions segures. Per tant, podrem aplicar aquests coneixements en properes ocasions.

6.- Glossari

- **Atac:** Acció realitzada per una tercera part, diferent de l'emissor i el receptor d'informació protegida, per intentar contrarestar aquesta protecció.

- **Atac d'aniversari:** Atac contra les funcions *hash*, consistent en trobar dos missatges que donin el mateix resum, en lloc de trobar un missatge que doni el mateix resum que un altre determinat, que requereix moltes més operacions.

- **Atac de força bruta:** Atac contra les funcions criptogràfiques, consistent en provar tots els possibles valors de la clau fins trobar el correcte.

- **Atac Man in the Middle:** Atac d'Home al Mig en català. Atac contra l'autenticació en els protocols de comunicació segurs, en què l'atacant intercepta els missatges d'autenticació i els substitueix per altres amb les claus públiques canviades, de manera que es pot produir una suplantació si no es comprova l'autenticitat d'aquestes claus.

- **Autenticitat:** Servei de seguretat gràcies al qual el destinatari de les dades té la garantia que l'autor és qui realment consta com a tal. És a dir que no hi ha cap tercera part que enviï les dades en nom d'un altre.

- **Clau Privada:** Una de les dues claus que s'utilitzen en la criptografia de clau pública. Només la pot conèixer el seu propietari. S'utilitza principalment per desxifrar els missatges rebuts (que hauran estat xifrats amb la clau pública).

- **Clau Pública:** Una de les dues claus que s'utilitzen en la criptografia de clau pública. Aquesta clau és coneguda per tothom. Serveix per xifrar els missatges, que s'hauran de desxifrar amb la clau privada. Ha de ser computacionalment inviable calcular el valor de la clau privada a partir de la pública.

- **Clau Simétrica:** Clau única usada en algorismes simètrics tant per xifrar com per desxifrar un missatge.

- **Confidencialitat:** Servei de seguretat que ens permet mantenir la informació en secret de manera que ningú no autoritzat en pugui obtenir el contingut.

- **Criptoanàlisi:** Estudi de les tècniques matemàtiques per anul·lar la protecció que proporciona la criptografia.

- **Criptografia de clau Asimétrica:** Tècnica d'encriptació basat en una funció matemàtica irreversible. Aquesta funció i la seva funció inversa depenen de dues claus diferents anomenades clau pública i clau privada. S'utilitza la clau pública per xifrar i la clau simètrica per desxifrar.

- **Criptografia de clau Simétrica:** Tècnica d'encriptació basat en una funció matemàtica reversible de manera que tant la funció com la seva inversa depenen

d'un mateix paràmetre anomenat clau secreta. Així doncs, utilitzarem la mateixa clau per xifrar que per desxifrar.

- **Criptografia:** Estudi de les tècniques matemàtiques per protegir la informació, de manera que no pugui ser llegida o modificada per parts no autoritzades.

- **Denegació de Servei (DOS):** Atac que fa un apropiació exclusiva d'un recurs o servei amb la intenció d'evitar qualsevol accés a terceres parts. En anglès, *deny of service*.

- **DES:** Data Encryption Standard. Algorisme de xifratge de clau simètrica adoptat pel NIST l'any 1977. Funciona amb claus de 64 bits dels quals 56 s'utilitzen pel xifratge/desxifratge.

- **Diffie-Hellman:** Algorisme de clau pública publicat per Whitfield Diffie i Martin Hellman el 1976. Aquest algorisme permet que dues parts es posin d'acord en mateixa una clau secreta. La seguretat de l'algorisme rau en la dificultat del càlcul de logaritmes discrets.

- **Hash, funció de:** És una funció matemàtica unidireccional que transforma una cadena de dades de qualsevol longitud en una cadena de longitud més curta i fixa, també anomenada *resum*.

- **HMAC:** Tècnica per calcular codis d'autenticació de missatge (MAC) basada en funcions *hash*.

- **Integritat:** Servei de seguretat que ens proporciona la garantia que les dades no han estat modificades per ningú, i que estan tal i com l'autor les va crear.

- **IPsec:** Conjunt de protocols a nivell de xarxa (AH, ESP, etc.) que afegeixen seguretat al protocol IP.

- **MAC:** Message Authentication Code. Valor calculat a partir d'un text amb una clau secreta, i que pot ser utilitzat per qui conegui la clau per comprovar l'autenticitat del missatge.

- **NIST:** National Insitute of Standards and Technology. Organisme fundat pel Congrés dels Estats Units que s'encarrega de desenvolupar noves tecnologies i estàndards.

- **No repudi:** Servei de seguretat que garanteix al destinatari que l'originador és qui realment consta com a tal, per tant aquest no pot negar-ne l'autoria.

- **NSA:** Agència de Seguretat Nacional dels Estats Units.

- **Signatura Digital:** Valor calculat a partir d'un text amb una clau privada, i que pot ser comprovat amb la corresponent clau pública, la qual cosa permet confirmar que només el pot haver generat el posseïdor de la clau privada.

- **SSH:** Socket Secure Shel*l*. Aplicació que proporciona un servei anàleg al del programa *Remote Shell* dels sistemes Unix, però amb la comunicació protegida mitjançant autenticació i xifratge, i amb funcionalitats afegides, com la redirecció de ports TCP a través de connexions segures, etc. També és el nom que rep el protocol utilitzat per aquesta aplicació per a la comunicació segura.

- **SSL:** Secure Sockets Layer**.** Protocol per protegir les comunicacions a nivell de transport, que ofereix uns serveis de comunicació segura anàlegs als que ofereix la interfície dels sockets.

- **Suposició de Kerckhoffs:** Premissa fonamental de la criptografia moderna d'acord amb la qual els algorismes han de ser coneguts públicament i la seva seguretat només ha de dependre de la clau.

- **Text en clar:** Dades intel·ligibles que poden ser llegides i processades sense la intervenció de cap algorisme de desxifratge.

- **Text xifrat:** Dades inintel·ligibles produïdes mitjançant algun algorisme de xifratge.

- **Xifra de Vernam, principi de:** Consisteix xifrar usant una clau secreta la longitud de la qual ha de ser igual a la longitud del text a xifrar. És una tècnica molt segura però poc pràctica perquè les claus són massa llargues.

- **Xifratge:** Transformació d'un text en clar, mitjançant un algorisme que té com a paràmetre una clau, en un text xifrat inintel·ligible per a qui no conegui la clau de desxifratge.

- **Xifratge de bloc:** Tipus de xifratge utilitzat per alguns algorismes simètrics que consisteix en aplicar el xifratge/desxifratge separadament a blocs d'entrada de longitud fixa. El resultat també és un bloc de longitud fixa.

- **Xifratge de flux:** Tipus de xifratge utilitzat per alguns algorismes simètrics que consisteix en combinar el text en clar amb un missatge de xifratge que s'obté a partir de la clau simètrica. El text a xifrar pot ser de mida variable.

7.- Bibliografia

Criptografia:

- Criptografía y Seguridad en Computadores 3 Edición. Manuel José Lucena López, 2004. http://wwwdi.ujaen.es/~mlucena/lcripto.html
- Handbook of Applied Cryptography 2001. Menezes, van Oorschot, Vanstone. CRC Press, 1996. http://www.cacr.math.uwaterloo.ca/hac/
- http://www.hack.gr/users/dij/crypto/overview/index.html
- CryptographyWorld. http://www.cryptographyworld.com
- CriptoRed. http://www.criptored.upm.es
- Apunts de l'Assignatura Seguretat en Xarxes de Computadors de la UOC

Diffie-Hellman:

- RFC 2631 - Diffie-Hellman Key Agreement Method. http://www.faqs.org/rfcs/rfc2631.html
- A Review of the Diffie-Hellman Algorithm and its Use in Secure Internet Protocols. David A. Carts. http://www.sans.org/rr/papers/20/751.pdf
- Diffie-Hellman Method For Key Agreement. http://postdiluvian.org/~seven/diffie.html
- Diffie-Hellman Key Exchange – A Non-Mathematician's Explanation. Keith Palmgren, CISSP. http://www.netip.com/articles/keith/diffie-helman.htm

DES:

- DES Encryption Overview. Tropical Software. http://www.tropsoft.com/strongenc/des.htm
- The DES Algorithm Illustrated. J. Orlin Grabbe. http://www.aci.net/kalliste/des.htm

Java:

- Sun MicroSystems. http://www.sun.com
- Thinking in Java,3rd Edition, Bruce Eckel, President,MindView, Inc.
- Java™ 2 Platform, Standard Edition, v 1.4.2 API Specification. http://java.sun.com/j2se/1.4.2/docs/api/index.html
- Aprenda Java como si estuviera en primero. Escuela Superior de Ingenieros Industriales de San Sebastián.
- The Java Developers Almanac 1.4. Patrick Chan http://javaalmanac.com
- Java Cryptography Extension (JCE) Reference Guide. http://java.sun.com/j2se/1.5.0/docs/guide/security/jce/JCERefGuide.html
- Experts-Exchange Forum http://www.experts-exchange.com/Programming/Programming_Languages/Java/Q_20854255.html
- Swing y JFC (Java Foundation Classes). http://www.programacion.com/java/tutorial/swing/
- WebZip Tutorial de Java Swing http://www.programacion.com/java/tutorial/swing/
- Web Taller. Diversos Tutorials de Java: http://www.webtaller.com/construccion/lenguajes/info/manuales/java/
- Java a través de Ejemplos, 2003. Jesús Bobadilla. Ed. Ra-Ma

Altres:

- El Lenguage Unificado de Modelado. Grady Booch, James Rumbaugh, Ivar Jacobson. Ed. Addison Wesley
- Apunts de l'assignatura Enginyeria del Programari I de la UOC.