XIFRATGE D'ARXIUS

Mikel Masquefa Ortega

Estudiant de EITG

Consultor: **Antoni Martinez Balleste**

Data lliurament: 15/06/2006

Índex

1. Especificació del problema

L'objectiu d'aquest projecte és dissenyar i implementar una utilitat que permeti xifrar (i posteriorment desxifrar) carpetes i fitxers fent servir claus basades en contrasenyes, la qual haurà de ser informada per l'usuari. Aquesta relació entre clau i contrasenya haurà de garantir que el sistema sigui prou segur davant d'atacs criptogràfics.

L'accés a aquesta utilitat podrà tenir dos opcions: o bé accedir des de el menú contextual de l'explorador de Windows, o bé accedir directament fent una crida al programa. Per tant, el programa haurà de ser resident al sistema.

2. Estat de l'art

L'objectiu d'aquest treball és desenvolupar una utilitat per a **xifrar/desxifrar fitxers** amb claus basades en contrasenya, i garantir la seguretat i integritat del sistema davant d'atacs criptoanalítics, com poden ser la intercepció i la modificació. És a dir, el sistema ha de garantir la **confidencialitat** i la **integritat** del text xifrat.

Per a aconseguir aquest objectiu el primer pas serà estudiar quins tipus de criptosistemes existeixen que ens permetin portar a terme aquesta funcionalitat.

Un cop catalogats els diferents criptosistemes, i escollit el que farem servir, s'haurà de veure quina implementació fer per a garantir la confidencialitat i la integritat de les dades xifrades.

D'altra banda, haurem de definir una interfície de comunicació amb l'usuari prou amigable com per que l'usuari es senti còmode davant del programa. En aquest cas, la opció més raonable és implementar una interfície gràfica on l'usuari només tingui que informar les dades amb les quals executar el procés i que li permeti xifrar/desxifrar recursos del sistema d'una manera molt intuïtiva, sense disposar de cap coneixement tècnic.

Per últim, haurem d'intentar definir un procés d'instal·lació el suficientment pseudoautomàtic per tal de que l'usuari pugui instal·lar el programa amb un mínim de passos, deixant al propi procés d'instal·lació la part més complexa, com pot ser la modificació del registre de Windows.

2.1 Classificació dels criptosistemes

Segons el tipus d'algorismes i claus que es fan servir, tenim diferents classificacions dels criptosistemes:

- Segons el tractament del missatge o text en clar, es classifiquen en:
  - *Xifrat en bloc*: el missatge es divideix en blocs d'una longitud prefixada (64-128 bits), i es xifra bloc a bloc amb la mateixa clau. Per a xifrar es suma bit a bit el text del bloc amb la clau.
  - *Xifrat en flux*: el missatge es xifra bit a bit, sumant bit a bit el text en clar amb la clau.
- Segons el tipus de clau, es classifiquen en:
  - *Xifrat amb clau secreta o sistemes simètrics*: existeix una única clau (secreta) per xifrar i desxifrar, que han de compartir emissor i receptor del missatge xifrat. La seguretat del sistema resideix en mantenir aquesta clau secreta.
  - *Xifrat amb clau pública o sistemes asimètrics*: cada usuari crea un parell de claus, una secreta i una pública, les quals tenen la relació que totes dues són inverses dintre d'un cos finit F_q. En aquest cas, en cadascun dels passos de xifratge/desxifratge es fa servir una de les dues claus en cada cas (l'emissor fa servir la clau pública del receptor per a xifrar un missatge).

    La seguretat del sistema resideix en la dificultat computacional d'esbrinar la clau secreta a partir de la clau pública.

A les següents figures, del llibre "*Curso de Seguridad Informática y Criptografia*", podem veure dos esquemes que representen el funcionament d'aquest dos tipus de criptosistemes:

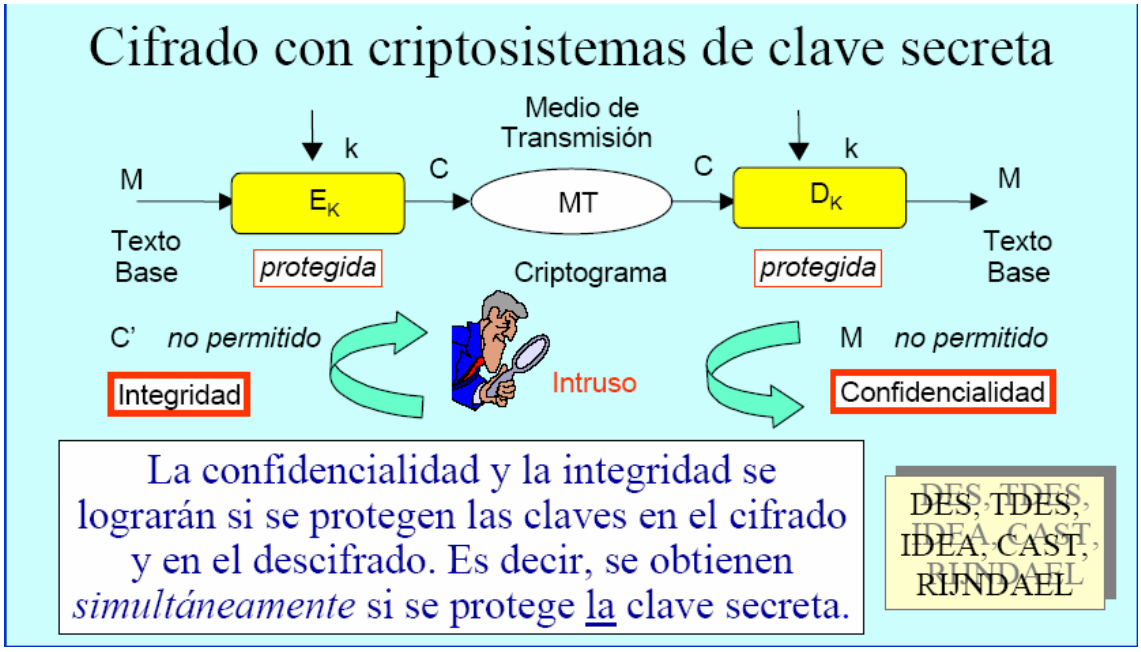

Figura 1: Funcionament dels criptosistemes de clau secreta

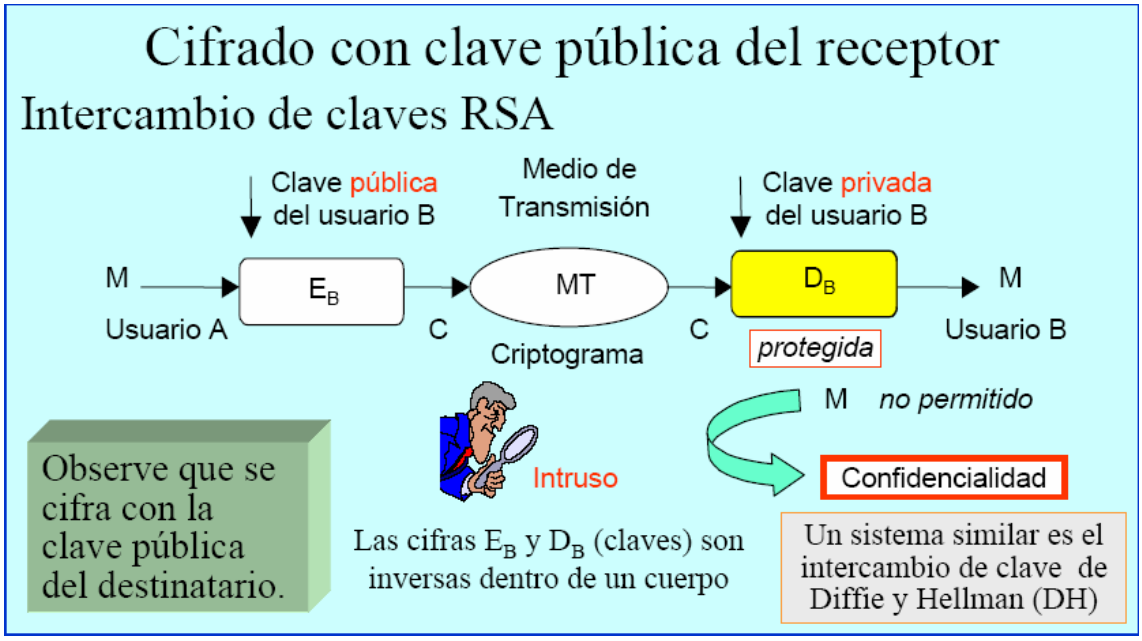

Figura 2: Funcionament dels criptosistemes de clau pública

2.2 Proposta de solució

A partir de les dues classificacions anteriors dels criptosistemes, hem de prendre una decisió alhora de com implementar la nostra eina, de tal manera que es garanteixin els requeriments indicats inicialment.

Segons els requeriments de la utilitat a implementar, la clau per a xifrar i desxifrar els arxius i carpetes ha de ser la mateixa, fet que ens porta a prendre la primera decisió: escollir un **criptosistema de clau secreta o simètric**, ja que sabem que en aquest cas tant l'emissor com el receptor comparteixen la mateixa clau per a xifrar i desxifrar.

Ens queda ara l'elecció entre un *xifrador de bloc* o un *xifrador de flux*. En aquest cas, aquesta elecció és una mica més complexa que l'anterior, ja que tots dos criptosistemes s'adapten a les necessitats del sistema.

Segons *Aguirre J.R.*, si fem una comparativa entre tots dos criptosistemes, tenim les següents avantatges /desavantatges en cada cas:

	Avantatges	**Desavantatges**
Xifrat en bloc	- Alta difusió dels elements en el criptograma. - Immune: impossible introduir blocs estranys sense detectar-ho.	- Baixa velocitat de xifrat al tenir que llegir abans tot el bloc. - Propens a errors de xifra. Un error es propagarà a tot el bloc.
Xifrat en flux	- Alta velocitat de xifrat al no tenir en compte altres elements. - Resistent a errors. La xifra és independent per a cada element.	- Baixa difusió dels elements en el criptograma. - Vulnerable: poden alterar-se els elements per separat.

Considerant els punts anteriors, veiem que la relació entre velocitat de xifrat i difusió és inversament proporcional.

El fet de tenir una bona difusió dels elements ens garanteix una millor protecció del criptograma davant d'atacs del tipus intercepció, els quals, estadísticament, són més habituals en situacions on el criptograma ha de viatjar per un medi no segur.

En el nostre cas, el criptograma no haurà de viatjar per cap medi, sinó que residirà al mateix sistema que el text en clar, és a dir, el criptograma residirà en el mateix ordinador on estigui instal·lat el programa (almenys inicialment).

Aquest fet ens porta a apostar per una major velocitat de xifrat i per tant una pèrdua de difusió del criptograma. És a dir, escollim un **criptosistema de xifratge de flux**. Això ens garanteix una menor necessitat de hardware per al nostre sistema.

Ara bé, per a garantir la integritat del criptograma, podem afegir un grau més de seguretat al nostre sistema que compensi la pèrdua de difusió: cada element xifrat serà acompanyat per un **codi MAC**, el qual ens permetrà garantir que el missatge xifrat no ha sigut modificat per cap intrús.

Per a generar aquest codi MAC, es farà servir la mateixa clau secreta del criptosistema, però en aquest cas es farà servir un algorisme de xifrat de bloc.

Evidentment, la seguretat global del sistema resideix en el fet de que **només l'emissor i el receptor del criptograma coneixen la clau secreta**. En el moment en el qual aquesta condició es trenca, el sistema deixa de ser segur.

2.3 Productes disponibles al mercat

Al mercat podem trobar diferents productes que fan una feina semblant a la del producte Lekim, tant de pagament, com de lliure distribució.

Per als usuaris de *Windows XP* ©, el propi SO incorpora aquesta funcionalitat de xifrar/desxifrar arxius i carpetes, sempre hi quan facin servir una partició de disc NTFS. Ara bé, els arxius xifrats no es poden comprimir amb la mateixa funcionalitat de Windows, fet que implica fer servir un producte extern per a comprimir els fitxers xifrats.

Dintre del software de pagament disponible al mercat, un dels de més pes potser és el Security Suite 7 de Steganos (www.steganos.com), el qual permet, entre altres funcionalitats, xifrar arxius i carpetes fent servir l'algoritme AES amb una clau de 256 bits, però no disposa de l'accés des de el menú contextual.

Dintre de les aplicacions gratuïtes, potser la més coneguda és PGP (www.pgp.com) la qual també permet xifrar arxius i carpetes. Ara bé, a diferència del programa Lekim, PGP fa servir un xifratge asimètric.

Dintre dels programes que fan servir xifratge simètric, la més interessant potser és CrypText (www.pcug.org.au/~njpayne) ja que és molt fàcil d'instal·lar i executar, i a més ofereix unes característiques molt semblants a Lekim: accés des de el menú contextual, xifratge simètric fent servir una combinació dels algorismes RC4 i SHA-1 amb una clau de 160 bits,..., l'únic problema que té és que no permet xifrar carpetes, només arxius, i d'altra banda no ofereix cap seguretat davant d'atacs contra la integritat del contingut xifrat.

Per tant, dintre de la gran quantitat de programes semblants que hi ha al mercat, Lekim ofereix els següents fets diferencials, que el fan un bon candidat per a l'ús domèstic:

- Gratuït.
- Fàcil d'instal·lar.
- Xifra carpetes i arxius des de el menú contextual.
- Fa servir l'algorisme RC4, el qual es prou segur si es fan servir contrasenyes diferents a paraules del diccionari.
- Ocupa poc espai de disc.
- Els arxius xifrats es comprimeixen en un arxiu ZIP.
- Ofereix seguretat davant d'atacs contra la integritat del contingut xifrat, fent servir un codi d'autentificació MAC per a cada fitxer xifrat.

3. Disseny del programa

El disseny del programa el podem dividir en tres grans apartats:

- *selecció del llenguatge de programació i llibreries a fer servir*: en aquest primer apartat s'ha de seleccionar quin llenguatge de programació s'ha de fer servir per a implementar el programa, així com que llibreries fer servir.
- *interfície gràfica de l'usuari*: en aquesta part es defineix com ha de ser la interfície de comunicació de l'usuari amb el programa. En particular, aquesta interfície ha de permetre a l'usuari com a mínim indicar quin és el recurs a xifrar/desxifrar i quina és la contrasenya a fer servir.
- *mòdul xifrador/desxifrador*: en aquesta part es defineix quin ha de ser el comportament del procés de xifrat/desxifrat així com quins mecanismes s'han de fer servir per a aconseguir els objectius del programa.

3.1 Selecció del llenguatge de programació

Dintre de les diferents opcions disponibles en llenguatges de programació, l'elecció feta és la de fer servir **Java** com a llenguatge de programació.

Les raons que m'han portat a aquesta elecció les podrien resumir en els següents arguments:

- Java és el llenguatge que més conec.
- És un llenguatge en el qual hi ha moltes llibreries disponibles en el mercat, tant per a temes criptogràfics com per a temes visuals.
- Permet d'una manera molt simple definir un programa que pot ser executat tant des de el menú contextual com directament.
- Els seus requeriments són molt reduïts: disposar d'una màquina virtual de Java (JVM) en el sistema.

En particular, l'aplicació s'ha implementat fent servir la següent versió de Java: JDK1.5.

Relacionat amb l'elecció anterior, un cop seleccionat el llenguatge de programació s'havia de triar quines llibreries del mercat fer servir per a implementar el programa.

Les llibreries seleccionades ens han de permetre implementar tant la part de la interfície de l'usuari així com el mòdul xifrador/desxifrador.

Per a la part de la interfície gràfica, l'elecció ha sigut fer servir l'API de Java Swing per a definir i implementar la interfície de l'usuari. Aquesta API ens permet una gran flexibilitat alhora de definir la interfície.

Per a la part del mòdul xifrador, aquesta havia de verificar els següents requeriments:

- Ser gratuïta, i a poder ser Open Source (OS).
- Permetre la implementació de l'algorisme de xifratge de flux seleccionat.
- Ser compatible amb la versió d la JVM seleccionada.

L'elecció feta en aquest cas ha sigut fer servir la llibreria criptogràfica desenvolupada per l'equip de *bouncy castle*: www.bouncycastle.org, la qual verifica els dos requeriments: és OS i implementa els diferents algorismes necessaris per al mòdul xifrador.

3.2 Disseny de la interfície gràfica de l'usuari

El disseny de la interfície gràfica ha partit de les següents premisses:

- L'usuari ha de poder informar d'una manera simple i entenedora les dades necessàries per a poder executar l'aplicació.
- Ha de poder tenir l'opció de poder saber quin ha sigut el resultat del procés.
- En cas de dubte, ha de disposar d'una ajuda contextual que li permeti aclarir qualsevol dubte relacionat amb l'aplicació.

A partir d'aquests requeriments, la interfície gràfica s'ha dividit en dues pantalles:
- Pantalla d'introducció de dades.
- Pantalla d'ajuda.

3.2.1 Pantalla d'introducció de dades

Els valors que si li demanen informar a l'usuari són els següents:

Ruta del recurs: Localització del recurs que es vol xifrar o desxifrar (només és necessari informar en el cas d'accedir directament a l'aplicació, és a dir, sense accedir-hi pel menú contextual). En cas d'haver de seleccionar un recurs, l'usuari disposa d'un botó, a la dreta, per tal d'obrir una nova finestra per a seleccionar el recurs.

Acció: Indica l'acció que vol realitzar l'usuari. Té dues opcions: xifrar o desxifrar.

Contrasenya: Permet que l'usuari informi la contrasenya a fer servir pel criptosistema.

Repetició de la contrasenya: Simplement permet validar que l'usuari no s'ha equivocat al informar la contrasenya.

Activar Log: Permet indicar a l'usuari si vol veure quin ha sigut el resultat del procés. Per defecte està activat. Les diferents traces del log es mostren per defecte a la mateixa pantalla, a la secció d'aquesta reservada per a aquest fet.

Activar log per consola: Permet indicar si a més de mostrar el log per pantalla volem veure el log per la consola. Útil per al cas de debug o desenvolupament.

El resultat final d'aquest disseny el podem veure a la següent pantalla:

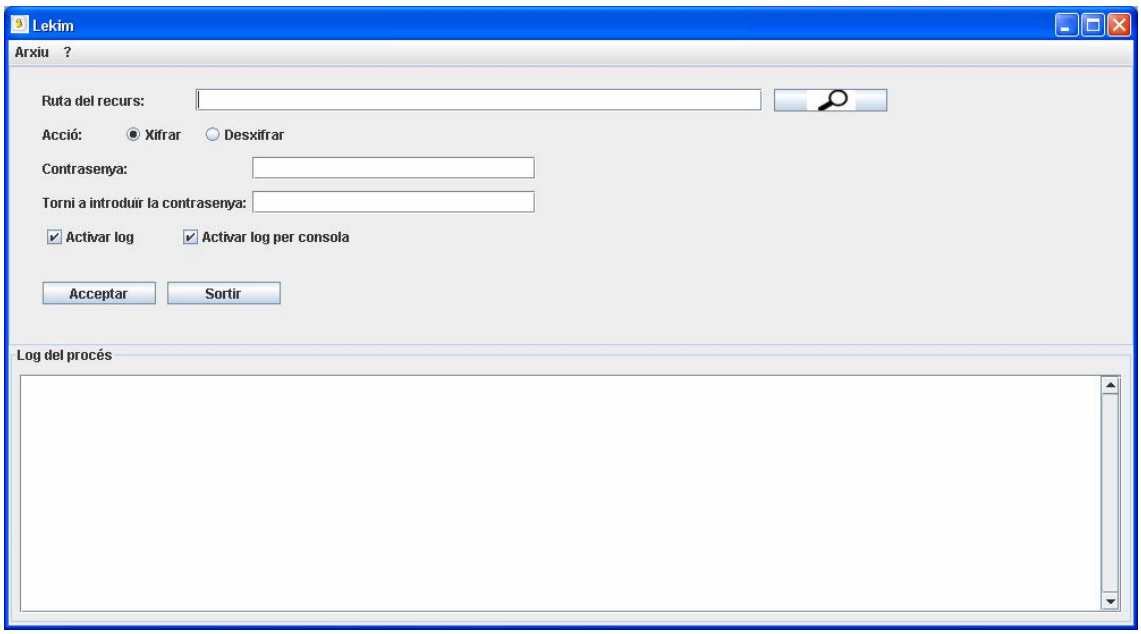

Figura 3: Pantalla principal de la interfície gràfica de l'usuari

3.2.2 Pantalla d'ajuda

Aquesta pantalla, a la qual s'accedeix des de el menú de la pantalla principal, s'obre en una nova finestra i mostra una descripció del funcionament del programa, donant una descripció dels diferents camps de la pantalla principal així com una descripció dels possibles resultats del procés.

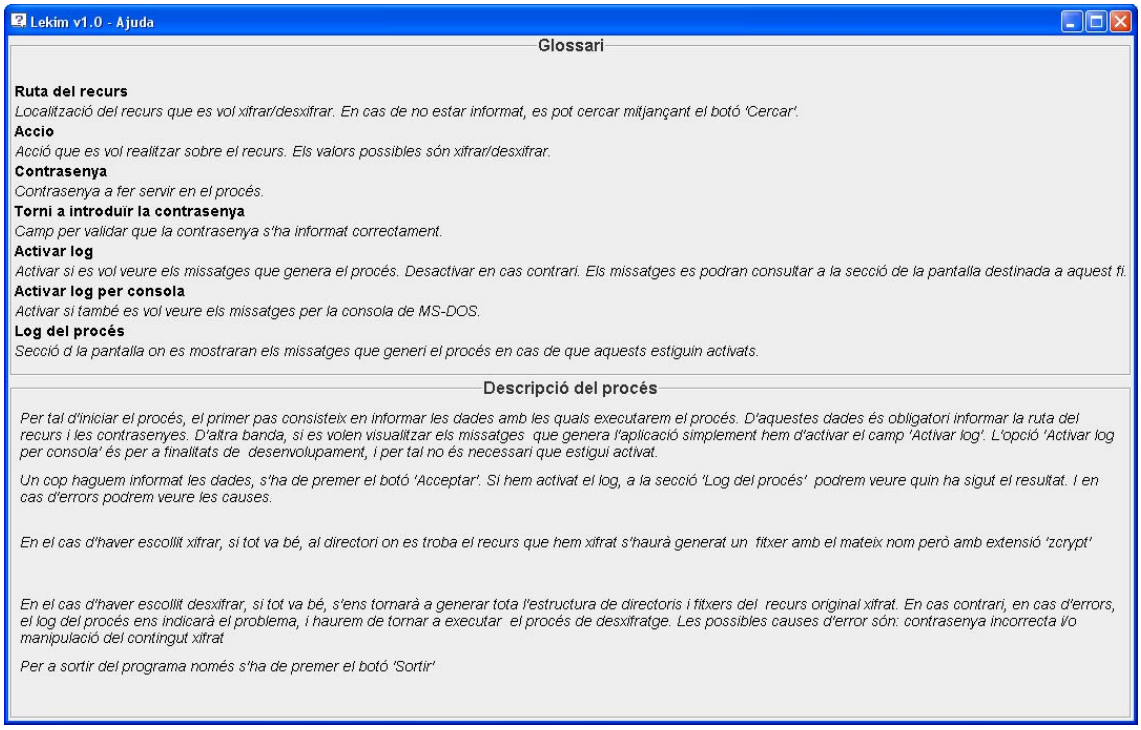

Figura 4: Pantalla d'ajuda de l'aplicació.

Finalment, l'aplicació conté una tercera pantalla, la qual mostra els crèdits de l'aplicació:

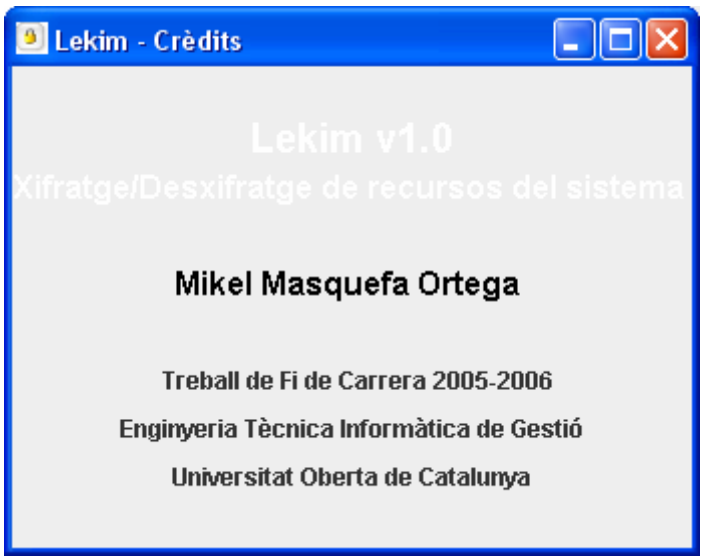

Figura 5: Pantalla de crèdits.

3.3 Disseny del mòdul xifrador/desxifrador

El disseny d'aquest mòdul ha sigut el més complex, ja que aquest mòdul és la part central de l'aplicació. És la part que porta a terme tot el procés de xifrat/desxifrat, així com la part que s'encarrega de garantir la integritat de les dades.

Alhora de fer el disseny d'aquest mòdul, les premisses que s'han de complir són les següents:

- El mòdul ha de permetre xifrar tant carpetes com fitxers.
- La confidencialitat i integritat de les dades xifrades ha de quedar garantida.
- L'autenticitat dels fitxers xifrats ha d'estar garantida.
- El resultat final del procés de xifratge ha de ser indiferent del recurs xifrat, és a dir, tant en el cas d'una carpeta com d'un fitxer, el resultat final ha de ser el mateix.
- El procés de desxifratge ha de recuperar l'estructura complerta de directoris del recurs original, de tal manera que el resultat de desxifrar el recurs xifrat sigui recuperar el recurs original.
- El procés ha de donar missatges de cadascun dels passos que realitza així com del resultat final del procés.
- En cas d'errors en el procés, el sistema ha de ser capaç de fer una sortida controlada i informar a l'usuari del problema.

Considerant aquestes premisses, es prenen les següents decisions respecte al disseny del mòdul:

- Es pren com a algorisme de xifratge en flux l'algorisme **RC4** per ser dels algorismes més ràpids del mercat a nivell de software. Aquest algorisme va ser desenvolupat l'any 1987 per RSA corp. i no és públic. En particular, es fa servir la implementació que d'aquest algorisme fa la llibreria criptogràfica de bouncycastle.
- Per a garantir la integritat de les dades, com a resultat de xifrar un fitxer també es genera un **codi MAC** associat a aquest.
- Per a generar aquest codi MAC es fa servir **l'algorisme de xifratge en bloc CBC en mode AES**. Com a contrasenya de l'algorisme es pren la mateixa contrasenya que per l'algorisme RC4.

- Com a resultat final del procés de xifrat es generen 3 tipus de fitxers, amb extensions:
    - .zcrypt: extensió que representa un fitxer comprimit i que conté el resultat de xifrar una carpeta o un fitxer. Cada fitxer que es fica del fitxer zcrypt es guarda amb la seva ruta complerta, per tal de poder ser després restaurat a la seva posició.
    - .crypt: extensió que representa a un fitxer amb les dades xifrades. És a dir, representa al criptograma.
    - .mac: extensió que representa a un fitxer amb el codi mac del fitxer original

  Per a cada recurs xifrat es genera un fitxer de tipus .zcrypt amb el mateix nom que el recurs però amb extensió zcrypt. Per exemple, si el recurs és un fitxer de nom *mi_document.pdf*, el resultat final serà un fitxer de nom *mi_document.pdf.zcrypt*. I si el recurs és una carpeta de nom *mi_carpeta*, el resultat final és un fitxer de nom *mi_carpeta.zcrypt*.

- El resultat final de xifrar un fitxer de nom *mi_document.pdf* seran dos fitxers de noms
  mi_document.pdf.crypt i *mi_document.pdf.mac*
  els quals estaran continguts en el fitxer
  mi_document.pdf.zcrypt

- El resultat final de xifrar una carpeta serà el mateix que el procés de xifrar cadascun dels elements que integren la carpeta.

- Després de xifrar un recurs satisfactòriament, aquest serà esborrat del sistema, així com també seran esborrats tots aquells fitxers temporals que hagin sigut creats.

- En el procés de desxifratge, per a cada fitxer a desxifrar, per tal de validar la integritat i autenticitat de les dades xifrades, es validarà que el codi MAC generat del fitxer desxifrat sigui idèntic al codi MAC del fitxer original. En cas de coincidència es regenerarà al sistema el fitxer original. En cas contrari, el procés de desxifratge s'aturarà i no es regenerarà el fitxer original.

A continuació es mostra de manera esquematitzada el pseudocodi associat a cadascun dels processos de xifrat i desxifrat de carpetes i fitxers.

Xifrat d'un recurs
entrada: path del recurs *mi_recurs*, contrasenya K. **sortida**: fitxer zcrypt amb el resultat del xifrat del recurs. **Procés**: 1. Inicialitzem l'algorisme RC4 amb la contrasenya K i en mode xifrat. 2. Creem un fitxer ZIP al sistema de nom *mi_recurs.zcrypt* i localitzat al mateix directori que *mi_recurs*. 3. Inicialitzem una llista buida, *wListFitxers*, on anirem deixant els fitxers que volem ficar dintre del ZIP. 4. Si *mi_recurs* és un fitxer passem al punt 6, en cas contrari continuem pel punt 5. 5. Recuperem la llista d'elements del recurs a xifrar, i per a cada element apliquem el punt 4. 6. Un cop llegides les dades del fitxer a xifrar, de nom *mi_recurs*.xxx, fem el següent: a. Xifrem el fitxer, i generem un fitxer al sistema de nom *mi_recurs.xxx.crypt* amb les dades xifrades i localitzat al mateix directori que *mi_recurs.xxx*. b. Generem el codi MAC per al fitxer *mi_recurs.xxx*, fent servir l'algorisme AES-CBC i contrasenya K, i generem al sistema un fitxer de nom *mi_recurs.xxx.mac* amb les dades del codi MAC i localitzat al mateix directori que *mi_recurs.xxx*. c. Esborrem del sistema el fitxer *mi_recurs.xxx*. d. Afegim a *wListFitxers* els .crypt i .mac generats. 7. Per a cada element de la llista *wListfitxers*, l'anem afegint al ZIP i l'esborrem del sistema. 8. Esborrem del sistema el recurs inicial *mi_recurs.* 9. Finalitzem el procés.

Figura 6: Esquema del procés de xifrat d'un recurs

Desxifrat d'un recurs
entrada: path del recurs *mi_recurs.zcrypt*, contrasenya K. **sortida**: fitxer *mi_recurs* original o missatge d'error. **Procés**: 1. Inicialitzem l'algorisme RC4 amb la contrasenya K i en mode desxifrat. 2. Obrim el fitxer ZIP *mi_recurs.zcrypt* i recuperem la llista de fitxers que l'integren. Per a cada fitxer fem: a. Llegim les dades del fitxer i en funció de la seva extensió passem als punts a.1 o a.2. a.1 Si el fitxer té extensió .crypt, desxifrem el fitxer i generem el seu codi .mac. Les dades d'aquest codi .mac les guardarem en un array de bytes temporal, *bMac*, i les dades del fitxer desxifrat les guardarem en un array de bytes temporal *bFile*. a.2 Si el fitxer té extensió .mac, aleshores comparem les seves dades amb l'array de bytes temporal *bMac*. Si les dades coincideixen aleshores generem un nou fitxer al sistema amb les dades de l'array temporal *bFile* i de nom igual al nom del fitxer .mac però sense l'extensió .mac. Si les dades no coincideixen, s'atura el procés de desxifrat i es genera un missatge d'error informant a l'usuari que o bé el fitxer xifrat ha sigut manipulat o bé que la contrasenya informada és incorrecta. 3. Si el procés ha finalitzat sense errors, el recurs inicial *mi_recurs.zcrypt* és esborrat. En cas contrari no s'esborra.

Figura 7: Esquema del procés de desxifrat d'un recurs

4. Aspectes concrets de la implementació

Al següent diagrama de classes podem veure una representació de les relacions entre les entitats que integren el programa:

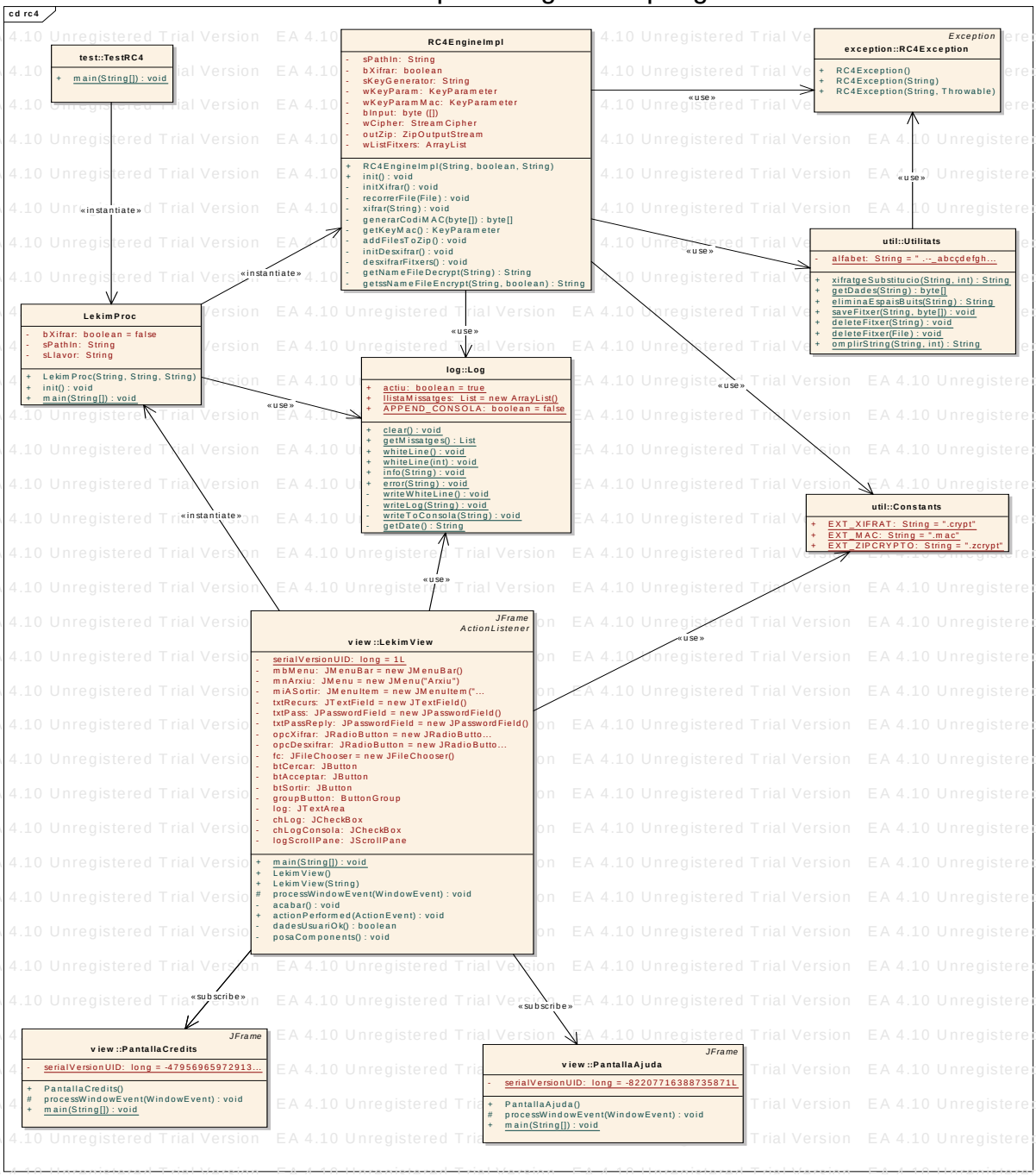

Figura 8: Diagrama de classes

Les classes del projecte es troben agrupades segons la següent estructura de paquets:

Package	Descripció
tfc.rc4	És el directori arrel del projecte. Conté les classes *LekimProc* i *RC4EngineImpl* les quals defineixen el comportament del mòdul xifrador/desxifrador.
tfc.rc4.exceptions	Conté la classe *RC4Exception*, la qual permet centralitzar totes les possibles excepcions del programa.
tfc.rc4.log	Conté la classe *Log*, la qual permet centralitzar totes les traces del sistema en un únic punt.
tfc.rc4.test	Conté la classe *TestRC4* la qual permet validar el funcionament del programa.
tfc.rc4.util	Conté les classes *Utilitats* i *Constants*, les quals donen servei a la resta de classes.
tfc.rc4.view	Conté les diferents classes de l'aplicació que defineixen la part gràfica: *LekimView*, *PantallaAjuda* i *PantallaCredits*.

Per a una descripció més detallada de la funcionalitat de cada classe es recomana llegir el *javadoc* annex a la documentació lliurada amb aquesta memòria.

Les classes principals del programa són la *LekimView*, que controla tot el procés de la interfície gràfica de l'usuari, controlant que els camps obligatoris estiguin informats i mostrant a l'usuari el resultat de l'execució; i la classe RC4EngineImpl que és la que s'encarrega d'implementar el mòdul xifrador/desxifrador.

Per a portar a terme aquesta implementació del mòdul xifrador/desxifrador, la classe fa servir les classes de la llibreria *bcprov-jdk15-132.jar* per a implementar els diferents algorismes i conceptes criptogràfics que fa servir: RC4, KeyStream, AES, CBC, ...

5. Manual d'instal·lació

La instal·lació del programa és molt simple. El programa es distribueix amb un fitxer auto instal·lable que s'encarrega de fer la major part de la instal·lació.

Com a requisit del programa, **cal tenir instal·lada una versió de la JVM de java 1.5**. Si no disposem d'aquesta màquina virtual, la podem descarregar de següent pàgina de SUN:

http://java.sun.com/javaee/index.jsp

Un cop instal·lada la JVM, hem de tenir una variable d'entorn de nom JAVA_HOME apuntant al directori d'instal·lació de la màquina virtual.

Per exemple, si el directori en el qual hem instal·lat la JVM és C:\java\jre1.5.0, per a donar d'alta la variable haurem de fer el següent (instruccions per a un sistema operatiu Windows XP):

1. Anar a Panell de Control > Sistema > Opcions Avançades

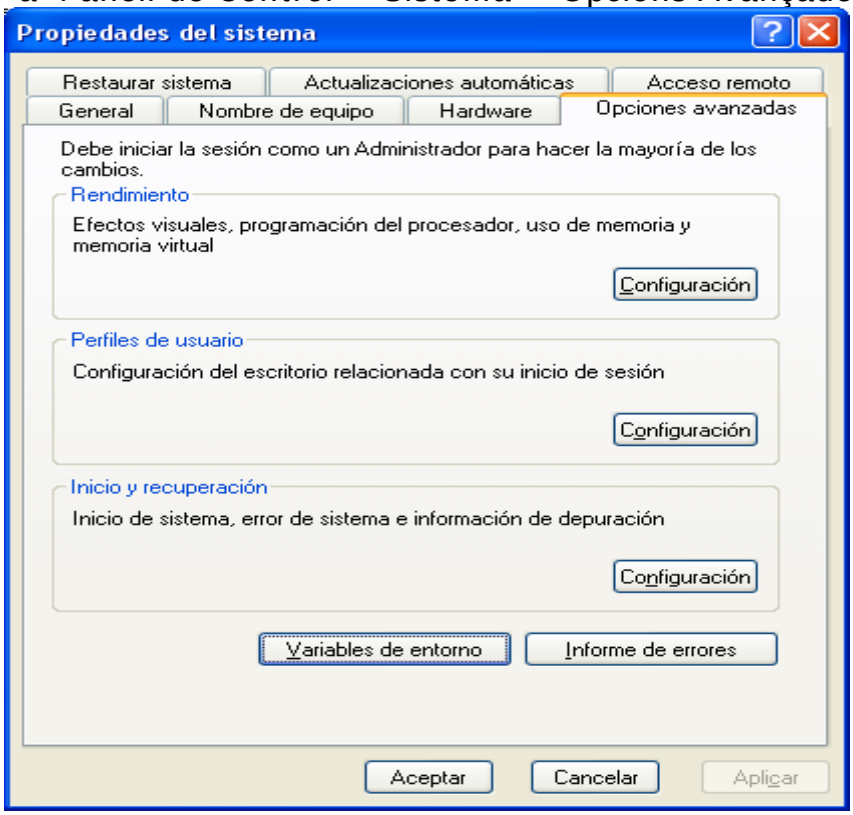

Figura 9: Opcions avançades del sistema

2. Anar a variables d'entorn i donar d'alta una nova variable de sistema amb el nom JAVA_HOME i valor la ruta on està instal·lada la JVM 1.5.

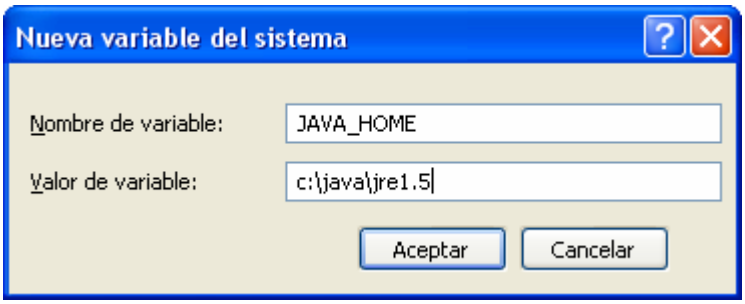

Figura 10: Alta de la variable dels sistema JAVA_HOME

Un cop fet el pas anterior, passem a executar el fitxer setup.exe, el qual s'encarregarà d'instal·lar al nostre sistema el programa Lekim. Durant el procés d'instal·lació, el procés ens demanarà en quin directori volem instal·lar l'aplicació.

A continuació es mostren les diferents pantalles del procés d'instal·lació de l'aplicació Lekim:

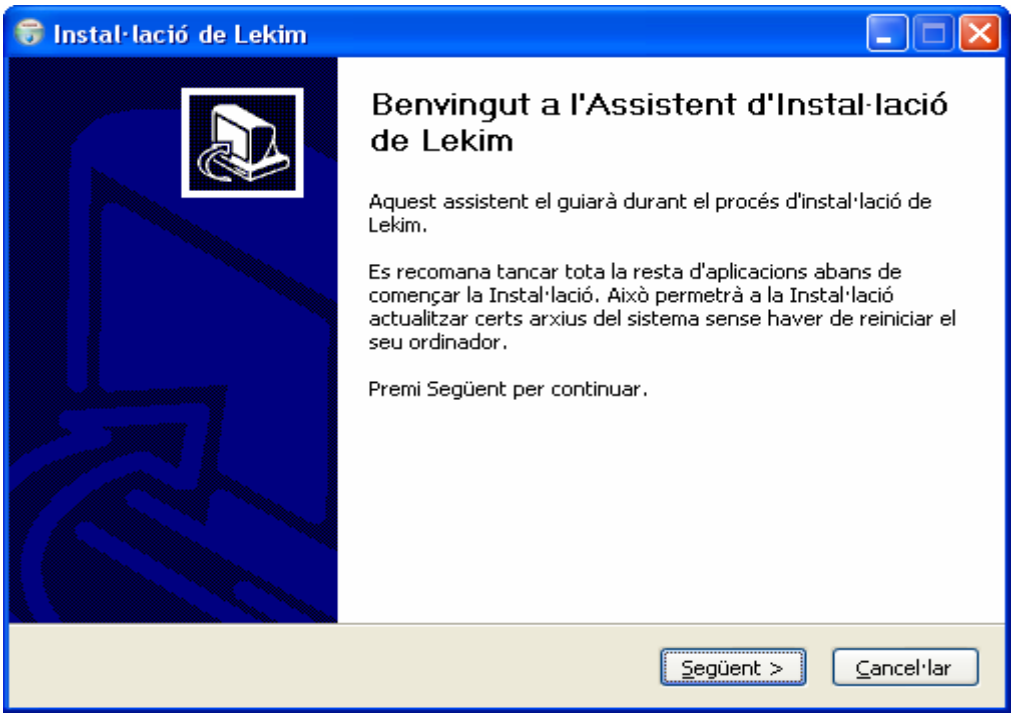

Figura 11: Benvinguda al procés d'instal·lació.

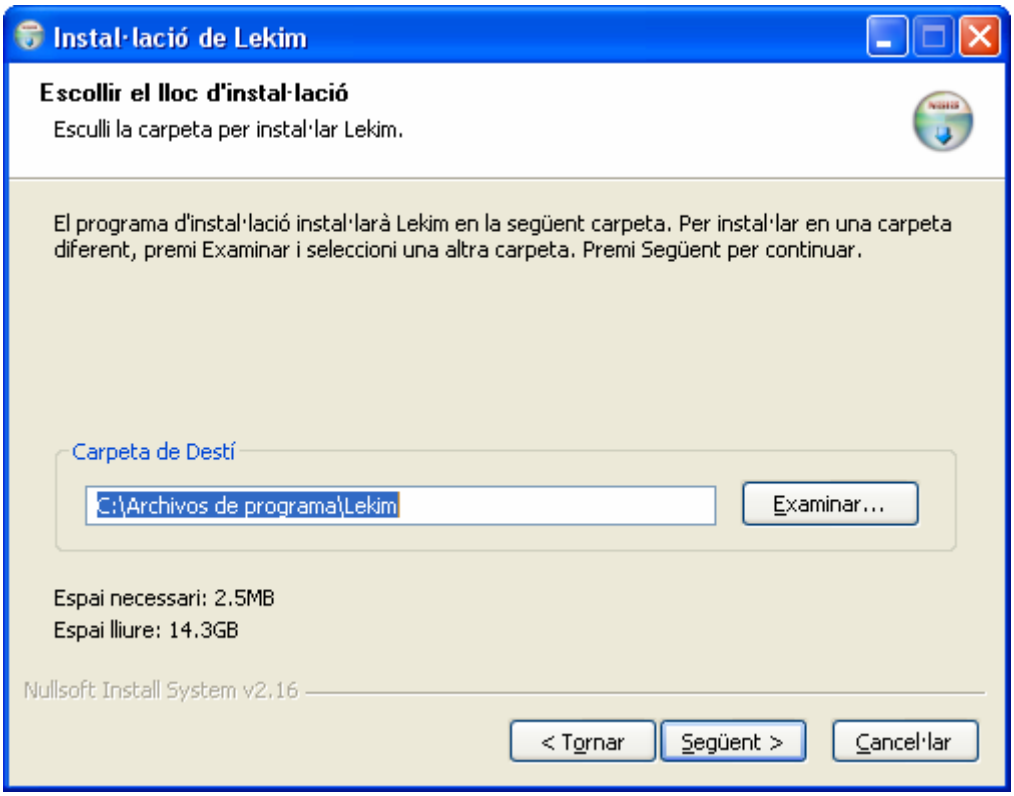

Figura 12: Selecció del directori on instal·lar Lekim

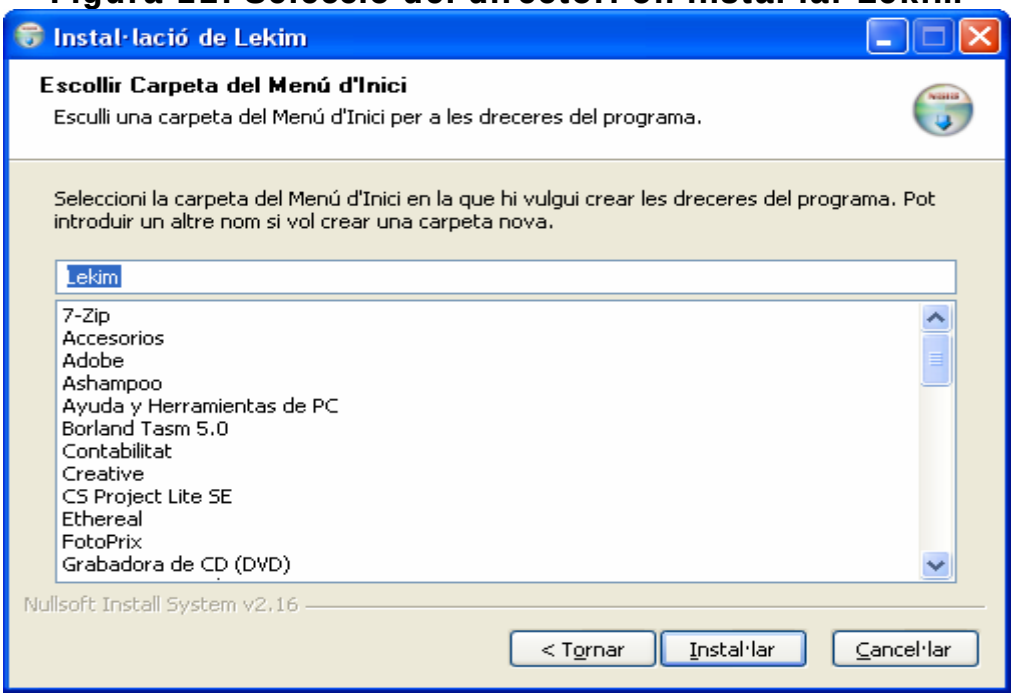

Figura 13: Seleccionar lloc del menú d'inici

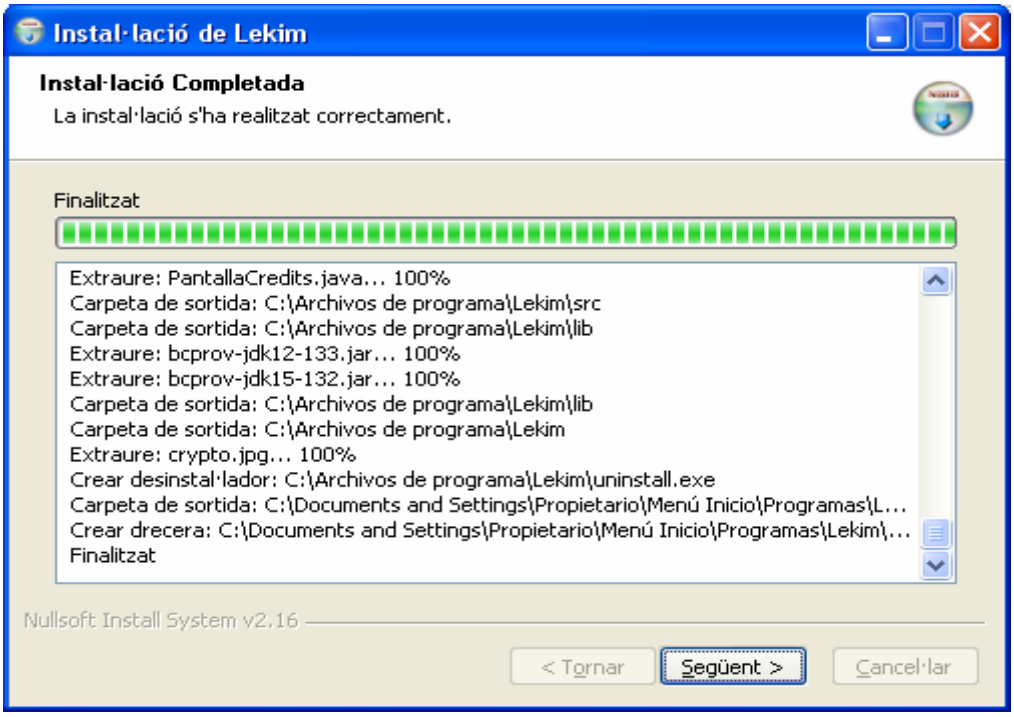

Figura 14: Log amb el resultat de la instal·lació

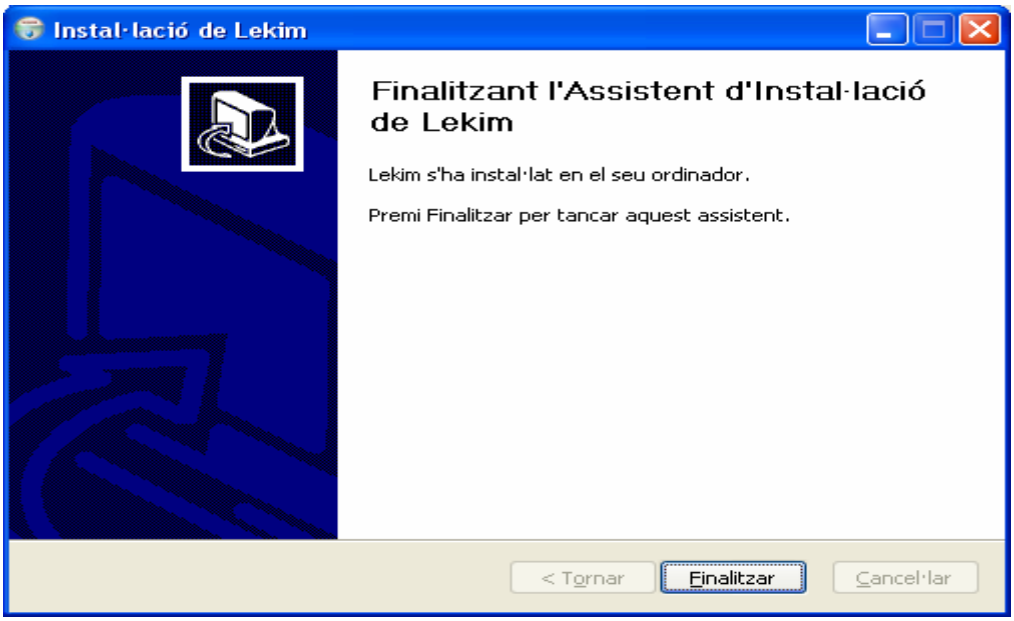

Figura 15: Final del procés d'instal·lació

Si tot a anat bé, haurem de tenir al directori indicat la següent estructura de directoris:

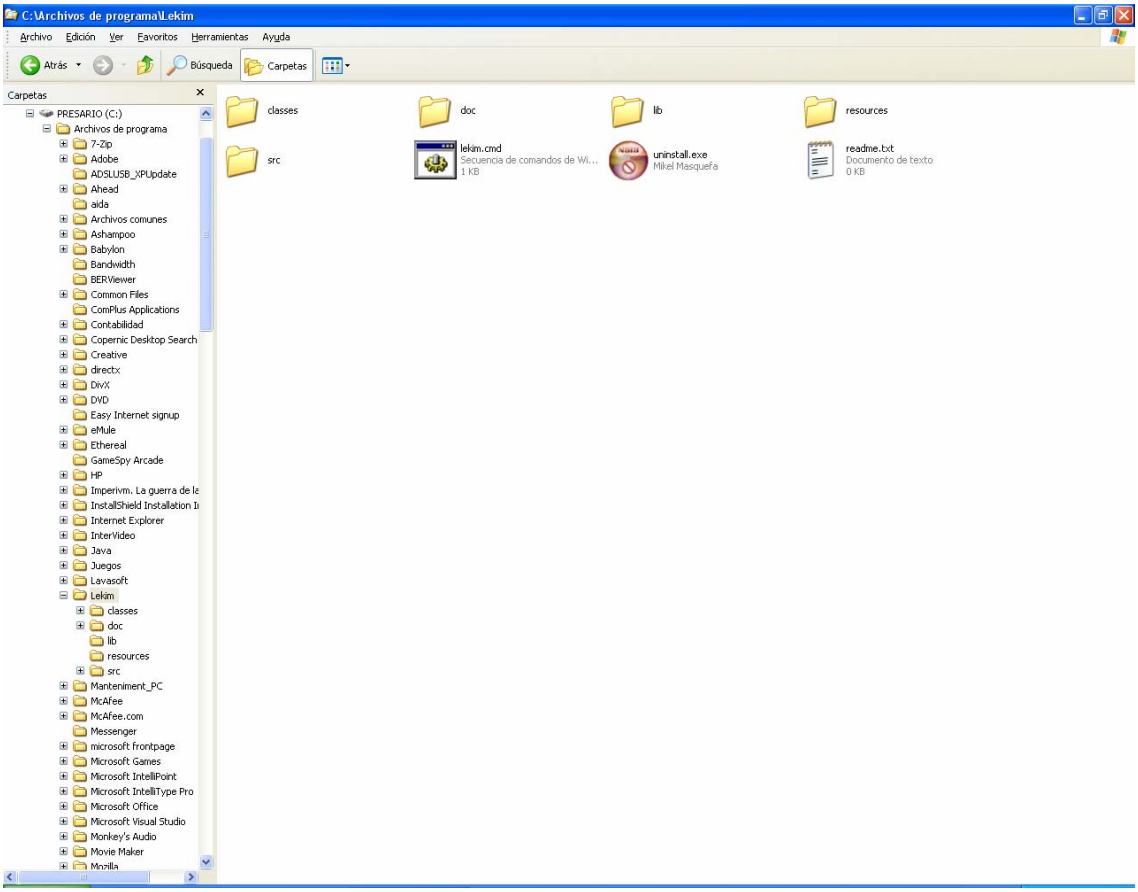

Figura 16: Contingut de l'aplicació Lekim

On

- classes conté els fitxers compilats de l'aplicació.
- doc conté la documentació de les classes de l'aplicació.
- lib conté les llibreries externes necessàries.
- src conté els fitxers font de l'aplicació.
- lekim.cmd és el fitxer que posa en marxa l'aplicació.
- uninstaller.exe és el desinstal·lador de l'aplicació.
- Readme.txt conté informació d'interès respecte a la llicència del programa, com informació per al procés d'execució.

I altra característica del procés d'instal·lació és que al menú contextual dels fitxers i directori del sistema ens ha de sortir una nova opció, de nom *Lekim*, que és la que ens permetrà xifrar/desxifrar aquests recursos.

Un cop finalitzat el procés anterior, l'únic que caldrà fer és donar d'alta una nova variable d'entorn de nom LEKIM_HOME i valor igual al directori on hem instal·lat l'aplicació. Per exemple:

LEKIM_HOME=c:\archivos de programa\lekim

Com a complement a l'aplicació, però no necessari per al seu funcionament, es pot associar a les extensions .zcrypt i .crypt una icona del sistema, fet que permetrà localitzar d'una manera mes ràpida visualment aquests tipus de fitxers.

A la següent pantalla es mostra una recomanació de quina hauria de ser la icona seleccionada:

Figura 17: Icona recomanada per a associar als fitxers de tipus zcrypt i crypt

Un cop tenim instal·lada l'aplicació, tenim dues opcions de posar-la en marxa:

- O bé des de el menú contextual de l'explorador.
- O bé executant directament el fitxer lekim.cmd.

En tots dos casos, s'obrirà la interfície gràfica de l'usuari. La diferència entre totes dues opcions és que en el primer cas no caldrà informar el recurs que es vol xifrar/desxifrar i en el segon si.

El resultat de l'execució en cadascun dels casos el podem observar en les següents figures: la figura 15 mostra el resultat de seleccionar l'opció Lekim del menú contextual de la carpeta C:\Archivos de programa\Lekim, i la figura 16 mostra el resultat d'executar directament el fitxer lekim.cmd. La diferència entre totes dues opcions és l'opció de poder seleccionar el recurs a xifrar/desxifrar.

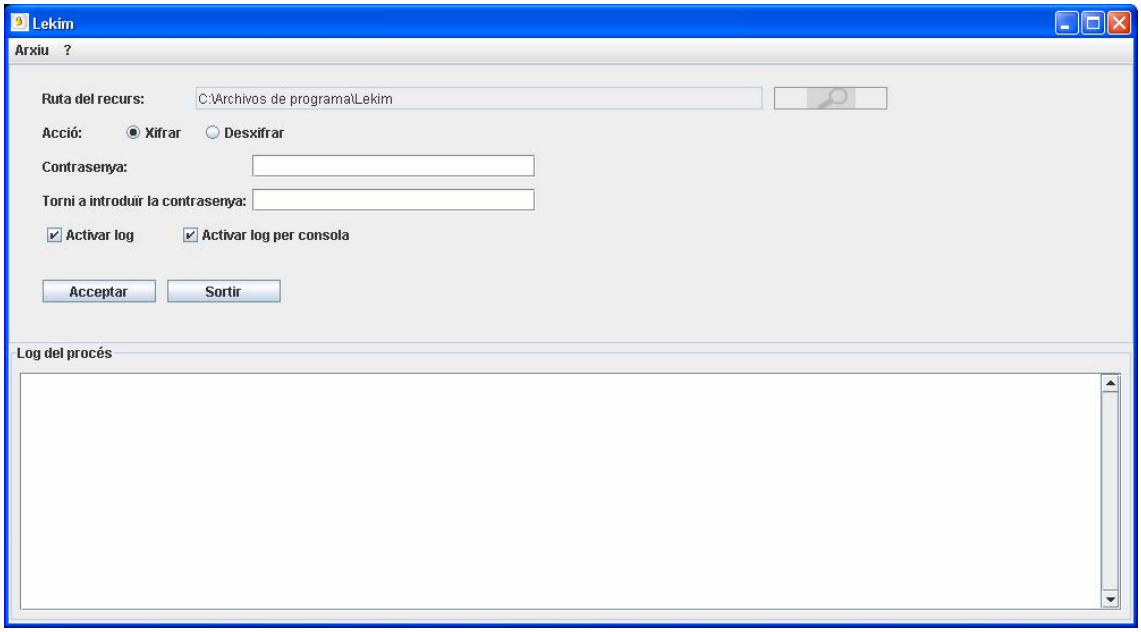

Figura 18: Execució des de el menú contextual

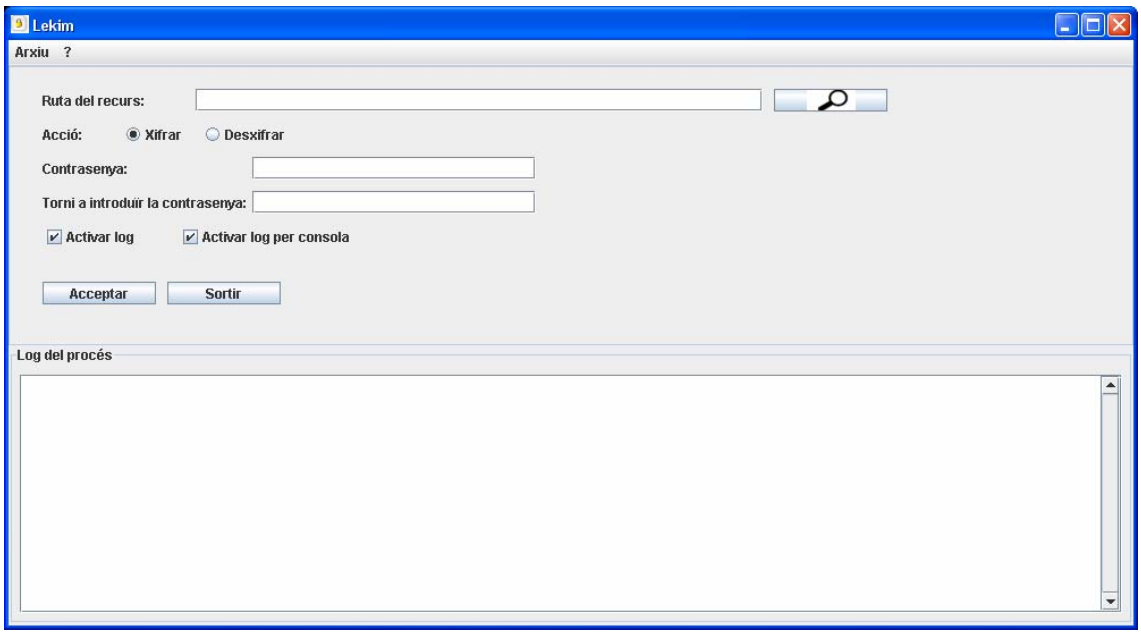

Figura 19: Execució directa del programa Lekim

Per tal de desinstal·lar l'aplicació l'únic que cal fer és executar la comanda *uninstaller.exe* i esborrar del sistema les dues variables d'entorn donades d'alta.

6. Proves de test

Per a validar el funcionament del programa, anem a fer diferents proves en les quals validarem el funcionament en diferents situacions.

La situació inicial de les proves és un directori *G:\temp\proves*, en el qual tenim: un fitxer word de nom *document_word.doc* i un subdirectori de nom *temp*. Dintre d'aquest subdirectori tenim dos fitxers més: un de nom *document_acrobat.pdf* i *document_postscript.ps*

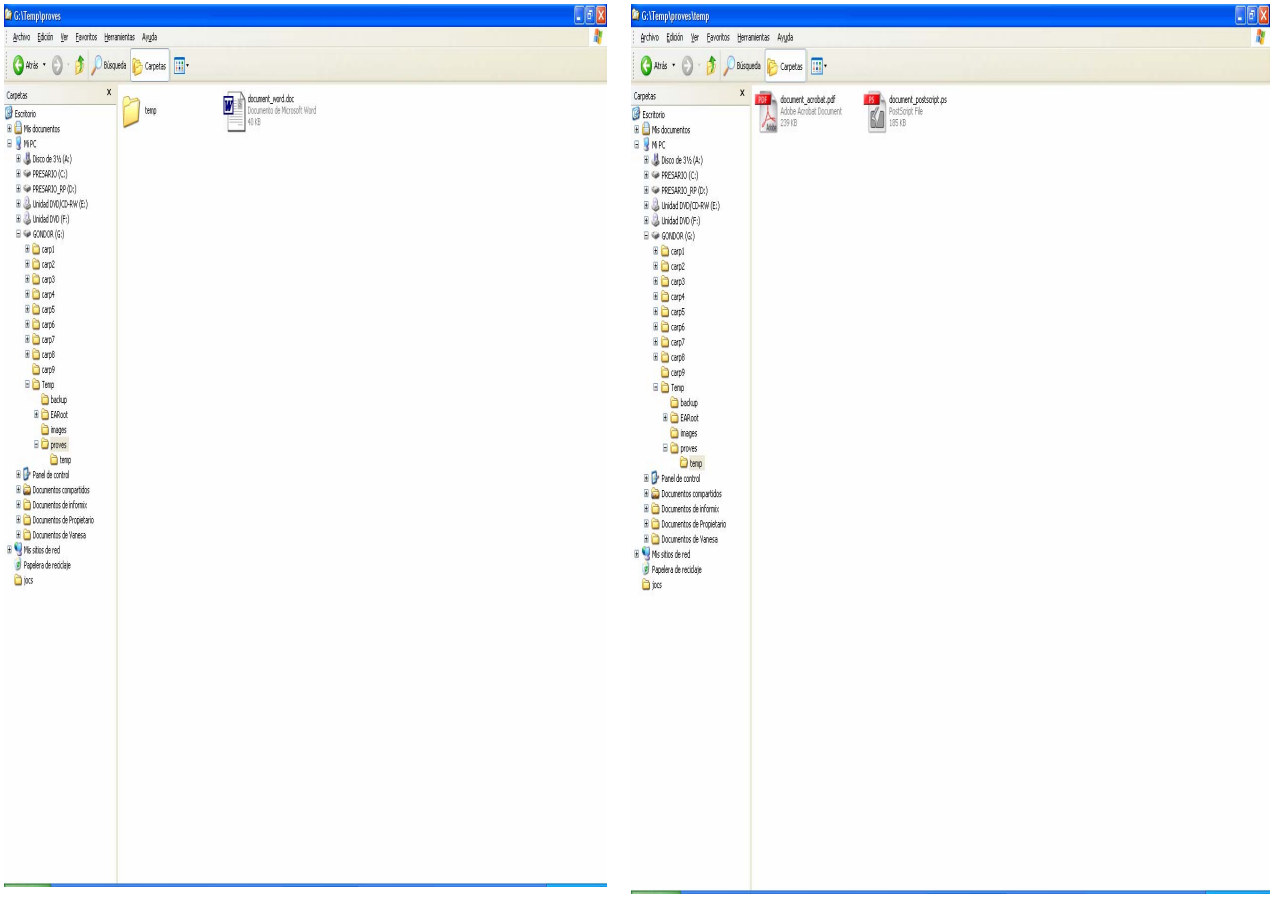

Figura 20: Estructura del directori de proves

Al seleccionar l'opció del menú contextual *Lekim* del directori G:\temp\proves, se'ns obre la següent finestra:

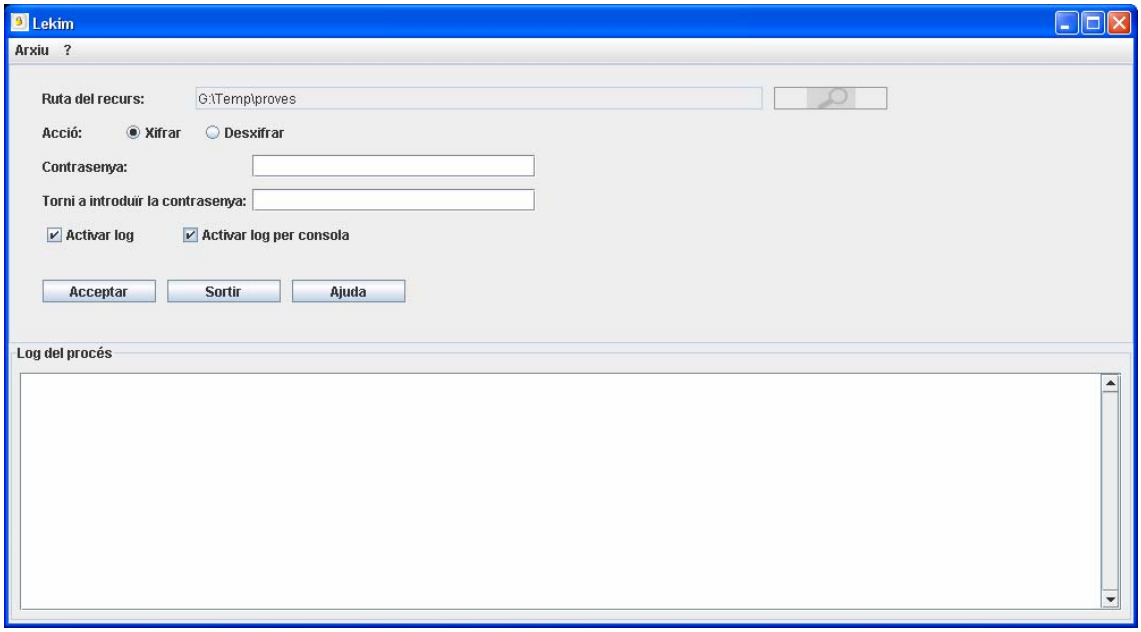

Figura 21: Inici joc de proves

Informem com a valor de la contrasenya la paraula *criptografia*, i polsem sobre executar. El resultat el podem observar a les següents tres figures:

- La figura 22 ens mostra el resultat per pantalla després de l'execució. A la secció de logs se'ns diu tots els passos que ha fet així com el resultat final del procés.
- La figura 23 ens mostra el resultat final sobre el directori G:\temp. La carpeta G:\temp\proves ens ha quedat buida i en el seu lloc ens ha aparegut el fitxer G:\temp\proves.zcrypt
- La figura 24 ens mostra el contingut del fitxer proves.zcrypt (el qual es pot obrir amb WinZip, per exemple), en el qual podem veure els fitxers .crypt i .mac, així com la ruta a la qual es trobaven.

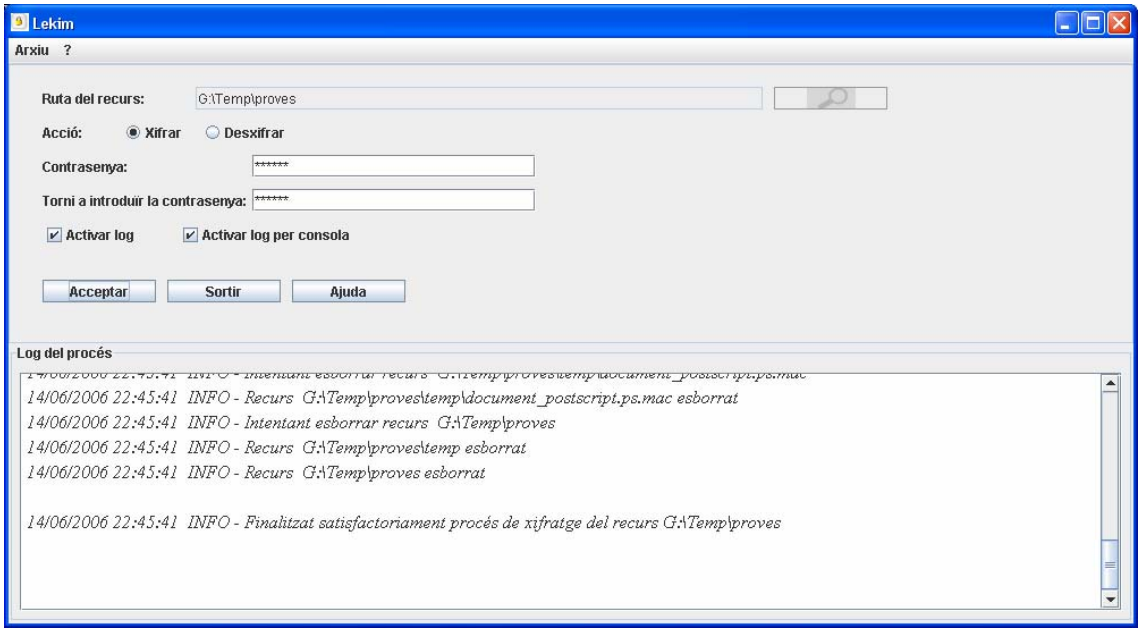

Figura 22: Resultat del procés de xifrat

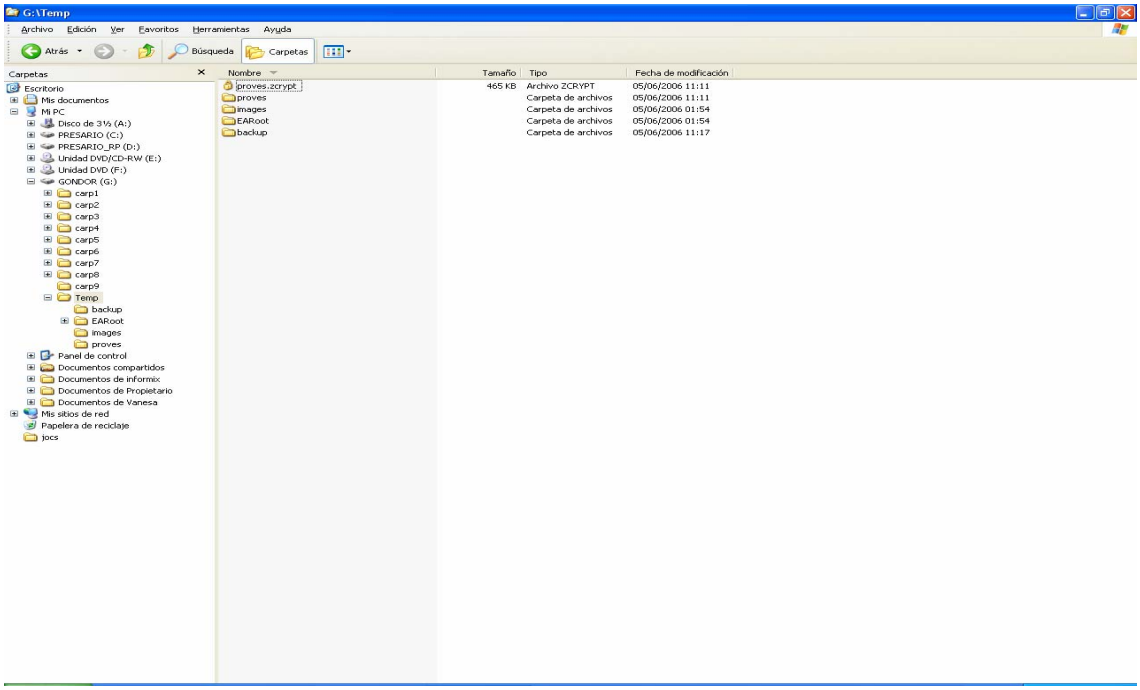

Figura 23: Resultat del procés de xifrat al directori

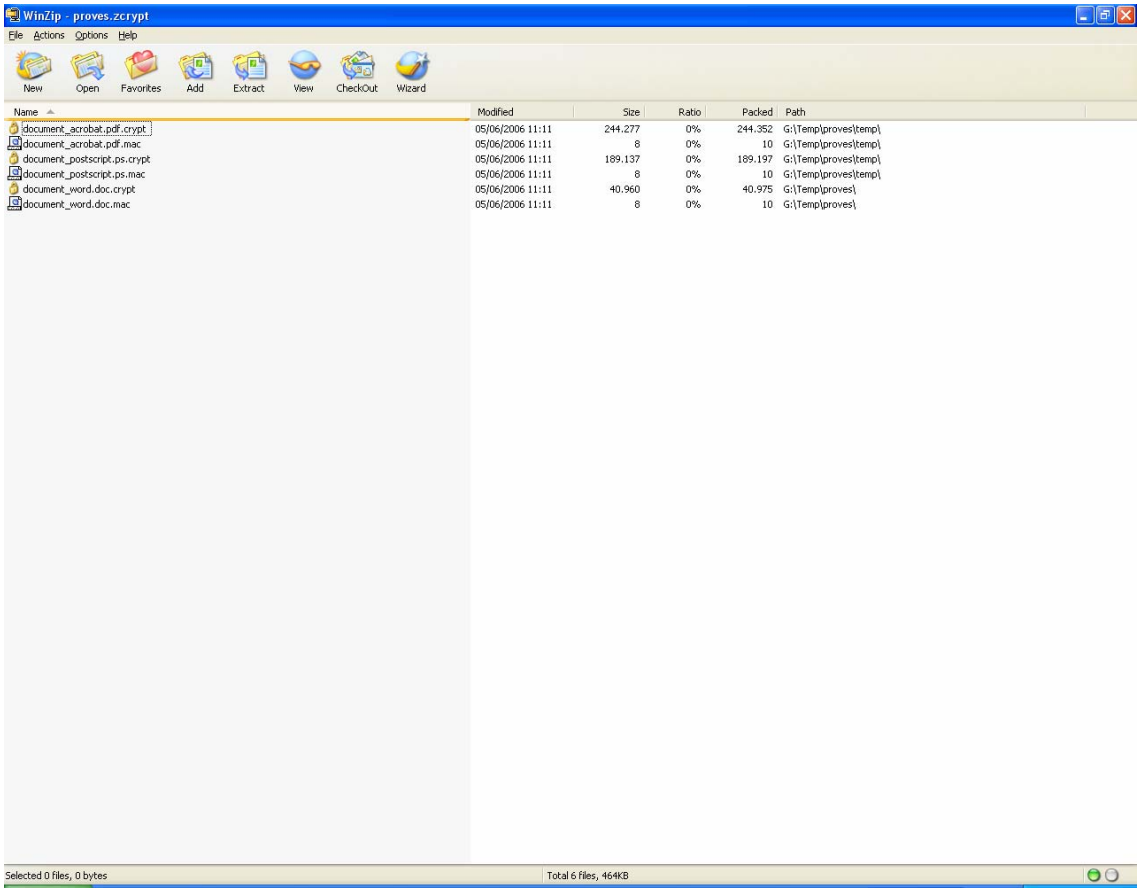

Figura 24: Contingut del fitxer proves.zcrypt

Anem ara a desxifrar el fitxer proves.zcrypt.

En el primer intent introduirem de manera incorrecta la contrasenya, fet que implica que el procés ens ha de donar un missatge d'error i no haurà de desxifrar cap contingut del fitxer (figura 26).

Al segon intent introduirem bé la contrasenya i comprovarem que el procés torna a generar l'estructura de directoris original amb el seu contingut (figura 27 i 28).

Per tant, anem al menú contextual del fitxer proves.zcrypt i seleccionem l'opció Lekim. Se'ns obrirà la finestra que es mostra a la figura 25.

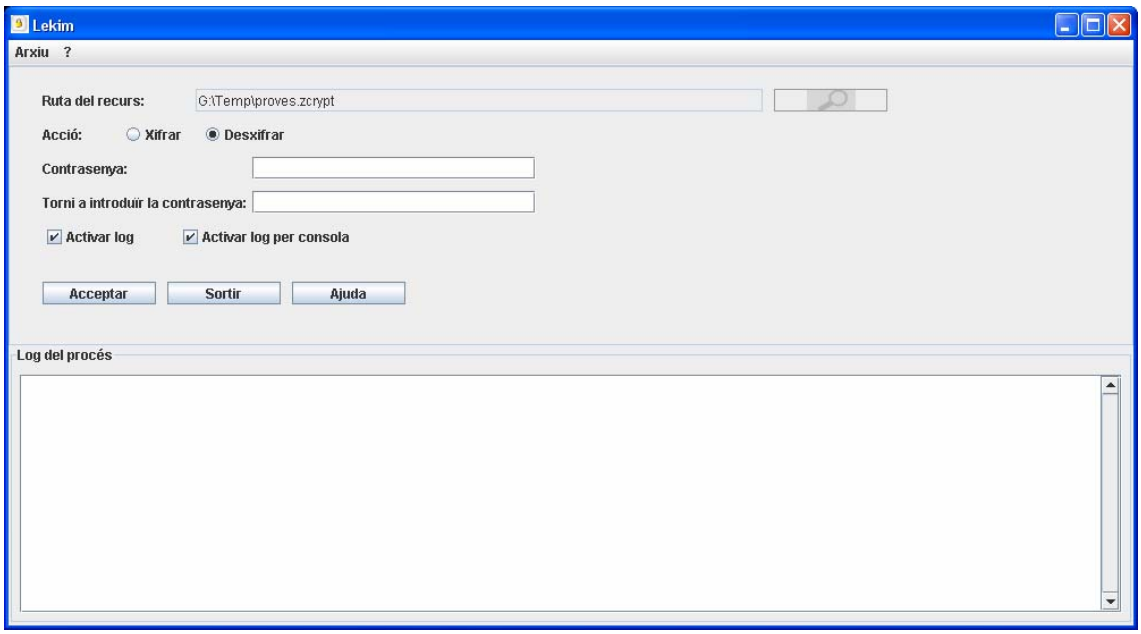

Figura 25: Inici procés de desxifrat de proves.zcrypt

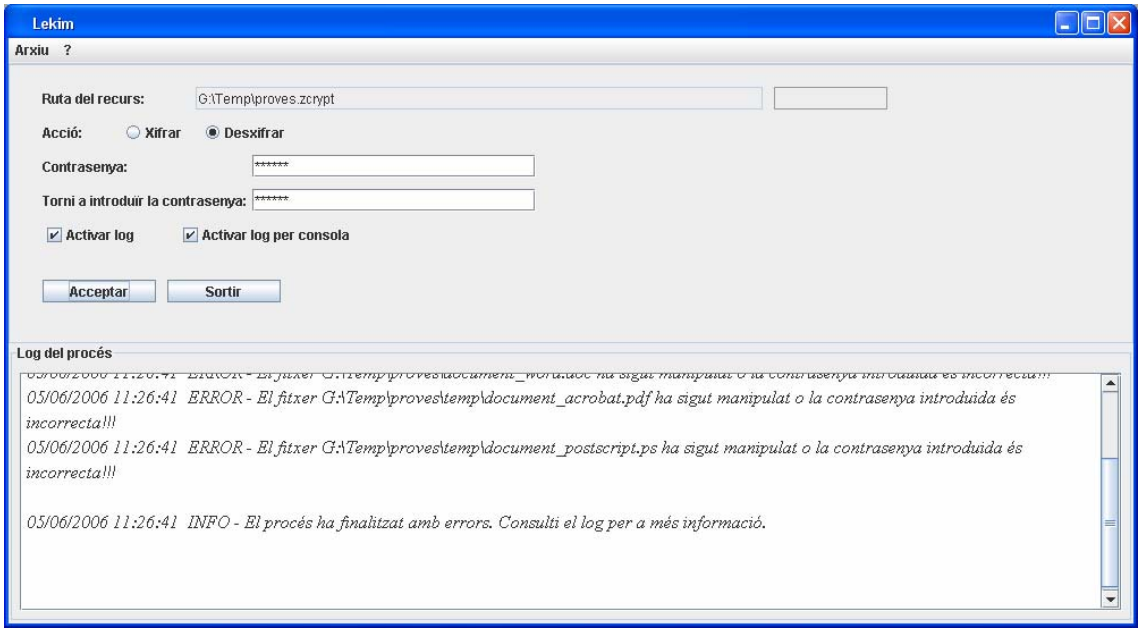

Figura 26: Finalització incorrecta del procés de desxifrat.

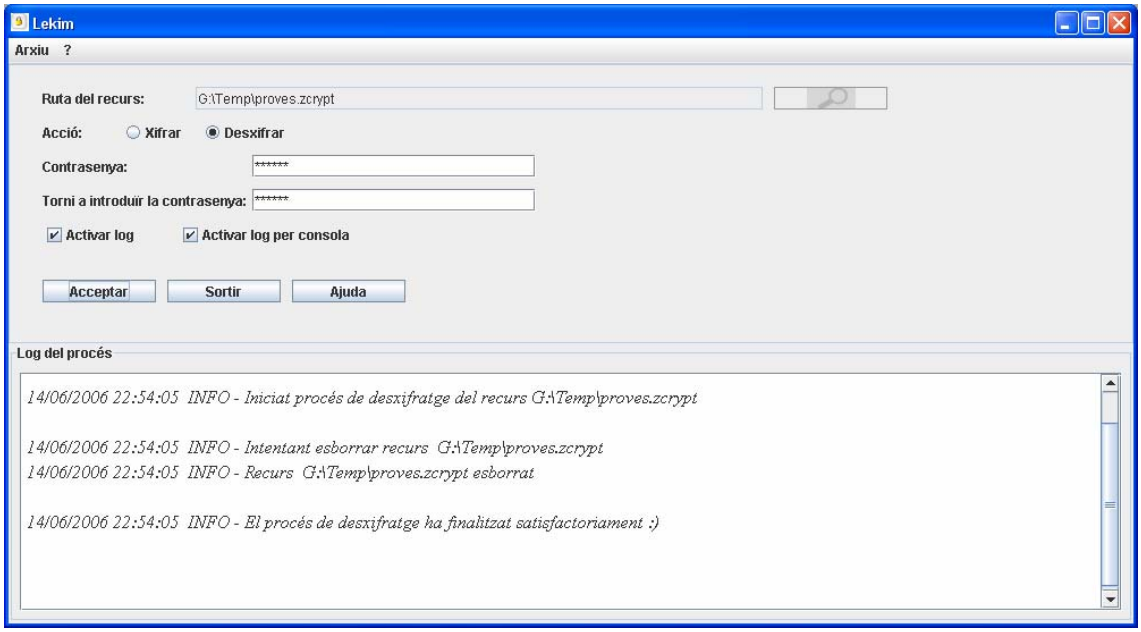

Figura 27: Finalització procés desxifrat satisfactòriament

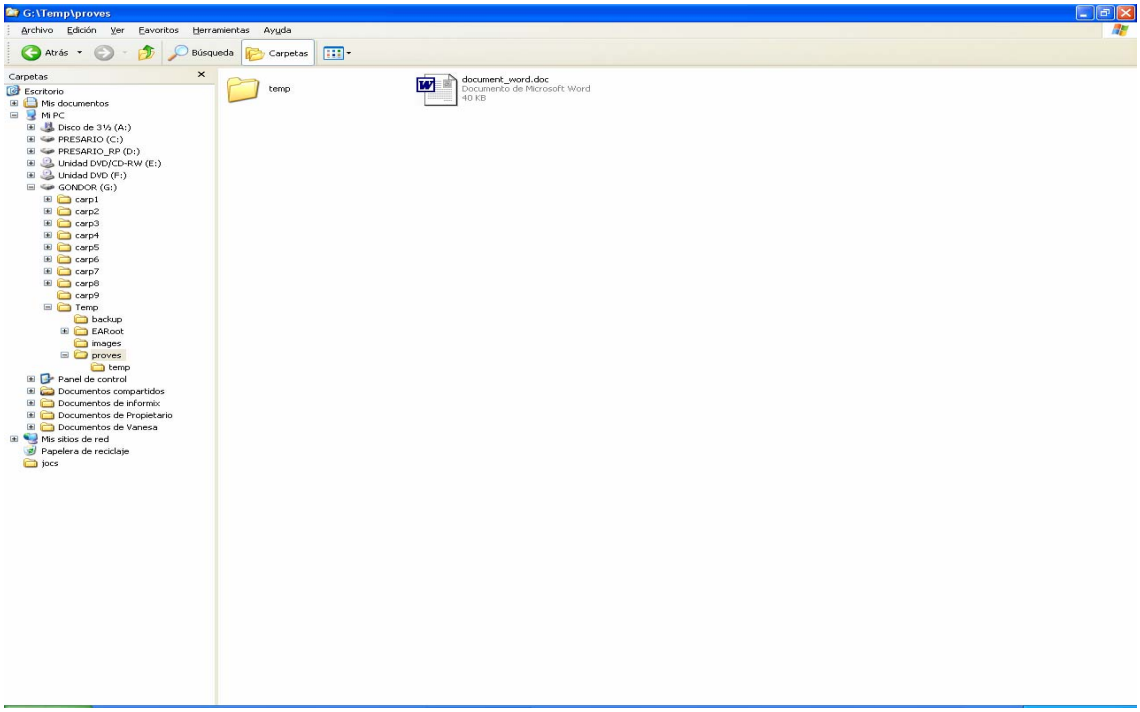

Figura 28: Pantalla amb el resultat del procés de desxifrat

Com a complement a aquestes proves es recomana manipular els fitxers .mac o .crypt del fitxer *proves.zcrypt* i veure quin és el resultat del procés de desxifrat.

7. Comentaris i conclusions

El programa *Lekim* al final compleix amb tots els requeriments especificats inicialment i afegeix un grau més de seguretat davant de possibles intents de modificar la integritat del text xifrat, com és la incorporació d'un MAC per a cada fitxer xifrat.

El fet d'implementar com a algorisme de xifratge en flux l'algorisme RC4, garanteix que la ambigüitat de la clau sigui prou alta com per a garantir una complexitat computacional prou alta per a trencar-la, almenys prou alta en front d'altres algorismes de xifratge en flux com poden ser LFSR, Geffe, ...

Una alternativa a aquest algorisme era fer servir l'algorisme Seal, el qual també té unes prestacions prou bones. La dificultat en aquest cas es que no vaig trobar al mercat cap distribució gratuïta d'una llibreria criptogràfica que implementés aquest algorisme. Vaig intentar implementar jo mateix l'algorisme, però al final vaig desestimar l'opció ja que vaig creure més convenient apostar per una implementació ja contrastada i validada com la RC4 de bouncy castle, en front d'una implementació pròpia de Seal que encara que intel·lectualment era més satisfactòria, a nivell de producte podria tenir certes mancances.

D'altra banda, per tal d'evitar la pèrdua de la informació davant de situacions en les quals la contrasenya de desxifratge s'informes de manera incorrecta, s'ha cregut oportú no eliminar el fitxer xifrat i donar a l'usuari una segona oportunitat. L'opció d'esborrar el fitxer xifrat en cas d'errors s'havia considerat inicialment com una opció més davant d'atacs criptoanalítics, però al final es va desestimar considerant els pros i contres que una opció com aquesta podia aportar a l'usuari final.

Altre punt a favor d'aquest programa es que per a la seva instal·lació no requereix que l'usuari faci moltes coses: simplement ha d'executar el fitxer d'instal·lació i donar d'alta dues variables de sistema com a molt. Això si, requereix la instal·lació de la JVM 1.5 que en cas de no tenir-ho. El propi procés d'instal·lació ja s'encarrega de modificar el registre de Windows per a donar d'alta al menú contextual dels fitxers i directoris el programa Lekim.

Finalment, s'ha cregut convenient no afegir codi font del programa en aquesta memòria per tal de fer més entenedor el seu contingut, i deixar que tots aquells lectors amb interès sobre aquesta part el puguin visualitzar directament a partir de la distribució que es fa del codi font amb el programa. Un altre aspecte interessant consisteix a consultar la documentació de cada classe que s'annexa amb aquest programa, i la qual e pot trobar dintre del directori doc.

"*La Matemàtica és la reina de les ciències i la Teoria dels nombres és la reina de les Matemàtiques*" Gauss (1777-1855)

8. Glossari

AES: *Advanced Encryption Standard*. Criptosistema que xifra blocs de 128 bits mitjançant una clau que pot variar de longitud entre 128, 192 o 256 bits.

Atac: estratègia o mètode que té per objectiu descobrir la clau de xifratge o bé el text en clar. Els atacs criptoanalítics exploten les febleses dels algorismes de xifra.

Ambigüitat de la clau: incertesa que queda sobre el valor de la clau un cop es coneix un criptograma.

CBC: *Cipher Bloc Chaining***.** Criptosistema de xifratge en bloc, on cada bloc de xifratge té una dependència amb l'immediat anterior.

Clau: paràmetre, normalment secret, que controla els processos de xifratge i/o desxifratge.

Confidencialitat: garantir que els components del sistema només seran accessibles per aquells usuaris autoritzats.

Criptograma: sinònim de text xifrat.

Criptosistema: sinònim de xifra.

Criptosistema de flux: sistema de xifratge que fa servir un generador pseudoaleatori per a xifrar un missatge, sumant bit a bit el text en clar amb la seqüència pseudoaleatòria que resulta del generador, la qual com a mínim a de tenir la mateixa longitud que el text en clar.

Desxifratge: procés de transformació del text xifrat a text en clar. Pas contrari a xifratge.

Difusió: transformació sobre el text en clar amb l'objectiu de dispersar les propietats estadístiques del llenguatge sobre tot el criptograma. Aquest objectiu s'aconsegueix mitjançant transposicions.

Funció hash: funció que dóna com a sortida un resum de longitud fixa a partir d'una entrada consistent en un missatge arbitràriament llarg.

Generador pseudoaleatori: procés determinista capaç de generar una seqüència pseudoaleatòria a partir d'una llavor.

Integritat: garantir que els components del sistema només podran ser alterats per aquells usuaris autoritzats. Sinònim d'autenticitat.

MAC: *Message Authentication Code*. Definim el codi MAC d'un conjunt de dades de longitud variable, a un resum de longitud fixa que s'aconsegueix aplicant una funció hash i una clau secreta a les dades. El codi obtingut permet garantir l'autenticitat de les dades.

RC4: criptosistema de xifratge en flux àmpliament utilitzat actualment en molts dispositius (SSL, WEP, WAP, TLA, ...). Entre les seves característiques destaquen una gran velocitat i una gran simplicitat alhora de ser implementat, tant a nivell de software com de hardware.

Xifra: mètode secret d'escriptura, mitjançant el qual un text en clar es transforma en text xifrat.

Xifratge: procés d'agafar un text en clar i transformar-lo en un text xifrat, o criptograma, per mitjà d'un algorisme, i fent ús d'una o més claus.

9. Bibliografia i recursos usats

Llibres – Material escrit

Aguirre, J.R. *"Curso de Seguridad Informática y Criptografia" (2003)*

Bishop, D. "*Introduction to cryptography with java applets*" *(2003)*

Ferrer, J.D.; Joancomartí, J.H.; Pons, H.R. *"Criptografia – Estudis d'Informàtica i Multimèdia" (2005)*

Menezes, A.J.; van Oorschot, P.C. ; Vanstone, S.A. "*Handbook of applied cryptography*" *(1996)*

Rueppel, R.A., "*Analysis and design of stream ciphers*" *(1986)*

Pagines Web

www.bouncycastle.org, Pàgina web del projecte OS criptogràfic *Bouncy Castle*

www.wikipedia.org, Enciclopèdia lliure on-line

http://nsis.sourceforge.net/Main_Page, pàgina principal de l'eina OS NSIS, que permet el desenvolupament de instal·ladors per a Windows.

Articles

Rogaway P.; Coppersmith D. *"A software-optimized encryption algorithm (SEAL)" (1997)*