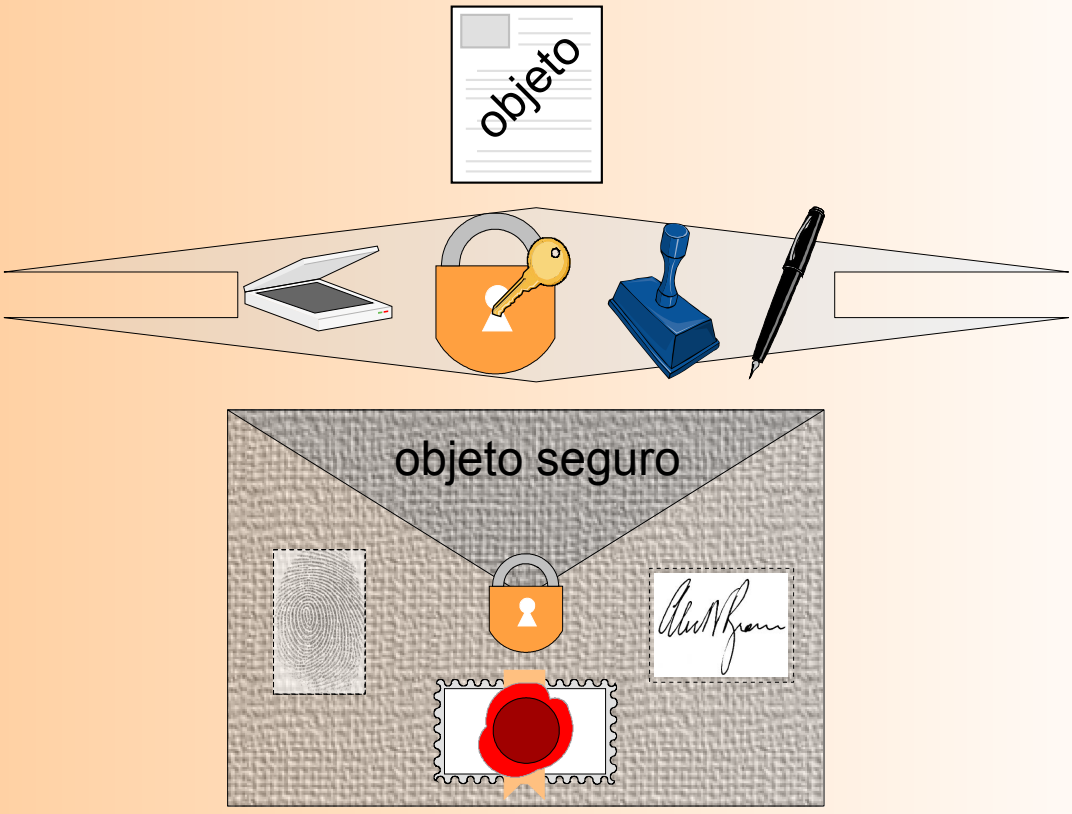
Proyecto Fin de Carrera

Aplicaciones Criptográficas Java



Jesús María Ramos Saky

Miguel Ángel Pérez Aguiar

Aplicaciones Criptográficas Java

Mayo 2006

Jesús María Ramos Saky

Tutor:

Miguel Ángel Pérez Aguiar

Facultad de Informática

Universidad de las Palmas de Gran Canaria

Agradecimientos

Este Proyecto Fin de Carrera ha sido realizado por D. Jesús María Ramos Saky, alumno de este Proyecto Fin de Carrera, y por el tutor del mismo D. Miguel Ángel Pérez Aguiar, profesor de Ingeniería Informática de la Universidad de Las Palmas de Gran Canaria. Agradezco el apoyo de mi familia, en especial, el de mis padres. Por otro lado, en menor medida, han participado los siguientes colaboradores: D. Luis Álvarez León, D. Yeray Zurita López, D. Francisco Mario Hernández Tejera, D. Miguel Ramírez Alemán, D. Carmelo Quintana Hernández y Dª Yessica Hernandez Mendoza.

Colaboradores	Información adicional
D. Luis Álvarez León	Catedrático de ULPGC en Ingeniería Informática
D. Yeray Zurita López	Licenciado en Turismo
D. Francisco Mario Hernández Tejera	Catedrático de ULPGC en Ingeniería Informática
D. Miguel Ramírez Alemán	Graduado en Ingeniería Informática
D. Carmelo Quintana Hernández	Graduado en Ingeniería Informática
Dª. Yessica Hernández Mendoza	Graduada en Ingeniería Informática

Gracias a todos.

Para obtener más información sobre este tema puede consultar la web http://jcef.sourceforge.net y http://jcef.sourceforge.net/doc/gratitudes.pdf.

Contenido

1.Sumario

En estos tiempos en los que las comunicaciones y las tecnologías de la información están siendo cada vez más importantes, combatir a los virus, hackers, escuchas y fraudes electrónicos, provoca que la seguridad tenga que tomarse muy en cuenta. Con el objetivo de proporcionar seguridad, existe la criptografía como herramienta principal y más importante con diferencia. A grandes rasgos, el uso de la criptografía ayuda a evitar el uso fraudulento de sistemas, a proteger información confidencial o importante, permitir comunicaciones seguras y posibilitar el comercio electrónico. Puede obtenerse más información sobre este tema consultando la dirección http://jcef.sourceforge.net/doc/introsecurity.pdf.

El objetivo primordial de este proyecto es aprender a utilizar los principales mecanismos criptográficos. Estos mecanismos son los de protección y autentificación. Como objetivo secundario, inicialmente se había planteado desarrollar un conjunto de programas representativos de las aplicaciones de la criptografía, pero tras el estudio preliminar, dicho objetivo fue eliminado y suplantado por otro mucho más interesante y más novedoso: desarrollar una librería criptográfica potente y sobre todo de fácil uso.

Siendo más precisos, la criptografía permite asegurar un objeto convirtiéndolo en otro objeto incomprensible y/o autentificable (un objeto autentificable es aquel en el que se puede comprobar si su origen es auténtico y/o sus propiedades son auténticas). Además, también permite obtener el objeto asegurado a partir de su versión segura. Para realizar estas transformaciones se utilizan algoritmos y parámetros criptográficos concretos; y es en la seguridad de estos elementos donde se basa la seguridad de la criptografía. Por otro lado, cabe destacar que un objeto puede ser cualquier cosa: información, recursos, datos, mensajes, ficheros, un objeto ya seguro incluso, etc... Más información sobre este tema en http://jcef.sourceforge.net/doc/oocryptography.pdf.

El resultado de este proyecto ha sido un conjunto de librerías Java. Entre ellas destaca la librería llamada JCEF (Java Cryptographic Extension Framework) cuya página web oficial y la de este proyecto se encuentra hospedada en el mayor repositorio de proyectos software de código abierto existente llamado SourceForge.net; la dirección de esta web es http://jcef.sourceforge.net.

Antes de la existencia de este proyecto fin de carrera ya existían unas librerías Java llamadas JCA y JCE encargadas de permitir a sus usuarios utilizar algoritmos criptográficos y algo más; pero su uso resulta y resultaba demasiado complicado. Es por ello por lo que se decidió realizar una librería Java para programar fácilmente sistemas que hagan uso de algoritmos criptográficos; en lugar de desarrollar un conjunto de programas representativo de las aplicaciones de la criptografía.

JCEF se soporta actualmente sobre JCA y JCE aunque no depende de dichas librerías. Eso sí, utilizar JCEF es mucho más sencillo que emplear JCA y JCE.

El cliente de esta nueva librería JCEF sería el desarrollador de software que necesita usar sistemas criptográficos de seguridad con suma facilidad sin las complejidades inherentes a JCA y JCE. JCEF es útil para programadores expertos en JCA y JCE y en especial para los nuevos programadores de sistemas criptográficos en Java.

Entrando en detalles, JCEF es un marco de trabajo para programadores que deseen desarrollar sistemas de seguridad basados en la criptografía a través del empleo de algoritmos criptográficos. Básicamente, JCEF define una especificación que permite definir y utilizar algoritmos criptográficos de protección y autentificación.

Para definir nuevos algoritmos criptográficos que cumplan la especificación JCEF se han implementado cinco especificaciones JCEF:

- Una para definir nuevos algoritmos criptográficos de una forma mucho más fácil de lo que permite JCA y JCE.
- Otra implementación permite adaptar fácilmente los algoritmos existentes con especificación JCA y JCE a la especificación JCEF.
- Una tercera implementación permite que los algoritmos implementados mediante la especificación JCEF sean compatibles con la especificación JCA y JCE.
- Para generar, convertir y comprobar automáticamente parámetros se ha realizado una cuarta implementación JCEF.
- Se ha realizado una última y quinta implementación JCEF para poder definir algoritmos criptográficos cuya implementación soporte las cuatro implementaciones anteriores como si fuera una única implementación, es decir, para definir algoritmos criptográficos de nueva implementación JCEF o adaptados de JCA y JCE y que además cumplan la especificación JCA y JCE y generen, conviertan y comprueben automáticamente parámetros que intervengan en dichos algoritmos.

A continuación puede verse una imagen que representa la posición del proyecto en el mundo Java de la criptografía:

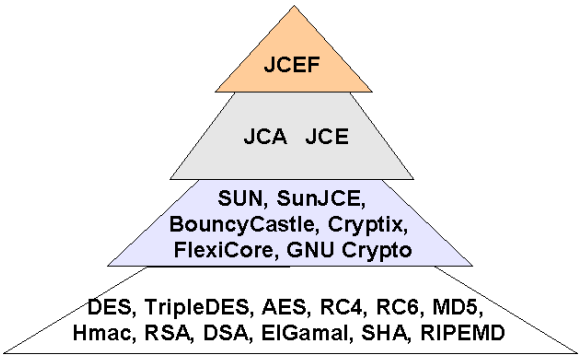


Ilustración 1: JCEF acerca la criptografía al usuario

JCEF mejora enormemente a JCA y JCE. Algunas de las deficiencias de JCA y JCE que han sido mejoradas son: JCE es muy difícil de utilizar, dificultando su uso y aprendizaje al proporcionar mecanismos heterogéneos y difíciles de emplear, requerir al usuario más conocimiento técnico del necesario y más líneas de código de las realmente necesarias para usuarios inexpertos con JCA y JCE. En definitiva, utilizar JCE requiere un gran esfuerzo en tiempo y dedicación al usuario del mismo.

Sin embargo, JCEF posee valores añadidos que hacen que valga la pena su uso. Algunos de estos valores añadidos más importantes son:

- Es amigable y fácil de utilizar al simplificar, eliminar y automatizar muchas operaciones de bajo nivel.
- Se aprende de forma muy sencilla y no requiere demasiado tiempo de aprendizaje ni grandes conocimientos técnicos.
- Y por si fuera poco viene con grandes posibilidades de extensión o mejora para el futuro y pruebas para comprobar el correcto funcionamiento de los algoritmos criptográficos y sus implementaciones.
- Permite además definir algoritmos criptográficos muy fácilmente.
- Además trae consigo un paquete inicial de 64 algoritmos criptográficos de todo tipo (algoritmos de protección asimétrica, protección simétrica de bloques y de flujo, protección basada en contraseña, algoritmos de autentificación mediante huellas digitales, sellos digitales, sellos digitales basados en contraseña, firmas digitales y generadores de fuentes de datos) y unas grandes colecciones de otros algoritmos de las mismas características, aunque estas colecciones no han sido probadas por razones de tiempo, como resultado de la traducción de todos los proveedores de algoritmos criptográficos Java encontrados.
- Y por último y más importante, JCEF permite asegurar objetos mediante la construcción de objetos seguros de una forma muy sencilla utilizando para ello cualquier tipo de algoritmo criptográfico y posibilitando obviamente recuperar estos objetos asegurados a partir de sus versiones seguras.

Por desgracia, JCEF no es perfecto y todavía puede y debe seguir creciendo. Algunas de las posibles mejoras que se le pueden hacer son:

- Implementar flujos de entrada/salida seguros para cualquier algoritmo criptográfico.
- Añadir soporte de almacenes seguros para cualquier tipo de objetos incluyendo claves y certificados digitales.
- Incluir protocolos de intercambio de claves.
- Gestionar certificados digitales.
- Configurar completamente los modos de operación y esquemas de relleno de los algoritmos criptográficos que los tengan.
- Realizar pruebas de implementación a los algoritmos mediante vectores de test con el objetivo de garantizar la correcta implementación de los algoritmos y aumentar el nivel de confianza de los usuarios de este proyecto.

- Corregir aquellos algoritmos que no pasen las pruebas de funcionalidad.
- Implementar algoritmos nuevos no implementados hasta ahora en Java.

Como un ejemplo del valor añadido de este proyecto, observe el código siguiente donde se muestra cómo se asegura un objeto e inmediatamente después se recupera el mismo; y todo ello de una forma super sencilla:

```
Object object = new String("my object");
CryptoAlgorithm secureAlgorithm = new AES_BlockSymmetricProtectionRREXKY();
SecureObject secureObject = new SecureObject(object, secureAlgorithm);
object = (String)secureObject.getObject(secureAlgorithm);
```

Tabla 1: Pequeño ejemplo de uno de los valores añadidos de JCEF

Si lo desea, también puede consultar la página web oficial del proyecto http://jcef.sourceforge.net y el recurso http://jcef.sourceforge.net/doc/project.pdf.

Sobre el diseño de este proyecto se puede decir que ha sido descompuesto en numerosas librerías. Además, cada librería tiene asociada otra que contiene las pruebas para comprobar el correcto funcionamiento de dicha librería. Por si fuera poco, todas las librerías han sido documentadas. Son un total de 38 librerías.

Las librerías de este proyecto son "JCEF" como librería principal, "JCEF Addons" como librería para extensiones futuras y el resto de librerías son proveedores criptográficos que cumplen al menos la especificación JCEF y librerías de pruebas para comprobar el correcto funcionamiento de todas las librerías. Concretamente se prueba el correcto funcionamiento de todos los métodos de todas las clases que componen el proyecto y se realizan pruebas de funcionalidad a cada algoritmo.

El diseño de este proyecto sigue los principios básicos de los Java Beans y otra consideración a tener en cuenta es que se ha intentado al máximo que el número de líneas de código por método fuera el menor posible.

Puede obtenerse más información sobre este tema en la web http://jcef.sourceforge.net y en la dirección http://jcef.sourceforge.net/doc/project.pdf que contiene información detallada del proyecto.

Además, se proporcionan una serie de distribuciones para que todo el mundo tenga acceso a este proyecto. Estas distribuciones se podrán encontrar en http://jcef.sourceforge.net y http://sourceforge.net/projects/jcef. Existen 6 distribuciones: una con lo básico para usuarios, otra para desarrolladores con todo lo necesario, otra con todo pero sólo para usuarios, otra que contiene todo el código fuente, otra distribución con toda la documentación del proyecto y por último una distribución que contiene la página web.

La página web del proyecto http://jcef.sourceforge.net permite el acceso del mundo al proyecto de forma libre y gratuita. En esta web se podrá encontrar todo lo relacionado con el proyecto: documentación, código fuente, presentación, las últimas noticias y muchos otros enlaces.

El manual de usuario de este proyecto, sin considerar este documento, se encuentra en http://jcef.sourceforge.net/api/org/rrexky/security/crypto/jcef/UserGuide.html Para obtener más información es importante consultar la página web de este proyecto http://jcef.sourceforge.net y la página principal online de la documentación de JCEF que se encuentra en http://jcef.sourceforge.net/api/index.html.

La dimensión del proyecto es considerable, ya que de ahí el gran número de librerías, debido principalmente a que se ha querido proporcionar el mayor número de proveedores y algoritmos criptográficos posible realizando adaptaciones de proveedores y algoritmos criptográficos ya existentes que cumplan con la especificación JCA y JCE u otras.

Para que el lector pueda hacerse una idea de la dimensión de este proyecto, basta con decir que este proyecto contiene 3142 clases, 315 campos, 2914 métodos, 73959 líneas de código y 22941 líneas de comentarios.

Otra métrica para imaginarse el coste de este proyecto es el tiempo de desarrollo y realización del mismo; el cual se estima entre un mínimo de 2000 horas de trabajo y un máximo de 3000 horas. Este proyecto ha tenido pocas pausas durante su desarrollo, el cual comenzó a finales de febrero de 2005 y terminó a finales de mayo de 2006, es decir, más de un año de desarrollo.

Sobre el coste económico del proyecto hay que decir que su coste económico es cero debido a que para el desarrollo del mismo se han utilizado únicamente recursos y programas totalmente gratuitos de licencia Open Source o Freeware.

También se ha documentado un conjunto de posibles futuros proyectos fin de carrera que podrán realizarse como continuación de éste para mejorarlo y ampliarlo. La descripción de estos futuros proyectos se ha realizado mediante el formato exigido a una propuesta de proyecto de fin de carrera por la Universidad de Las Palmas de Gran Canaria, con el objetivo de facilitar la elección de estos proyectos por alumnos y tutores de dicha universidad y dar así continuidad a este proyecto. Los 7 futuros proyectos propuestos se titulan: «Ampliaciones de JCEF», «Pruebas sobre algoritmos JCEF», «Certificados Digitales con JCEF», «Archivos seguros con JCEF», «Proveedor Criptográfico JCEF», «Almacén de objetos seguros con JCEF», «Metaimplementación de "Aplicaciones Criptográficas Java"».

Se puede obtener más información sobre el proyecto en http://jcef.sourceforge.net y sobre los futuros proyectos solamente en la dirección web http://jcef.sourceforge.net/doc/futureprojects.pdf.

Para la realización del proyecto se han utilizado numerosos recursos totalmente gratuitos de licencia Open Source o Freeware, tales como bibliografía, recursos web, programas, etc... A continuación se muestra un sumario de los recursos empleados para el desarrollo del proyecto:

- Para obtener información: Bibliografía y numerosas páginas y recursos web.
- Para dar valor al proyecto: Se han utilizado todos los proveedores criptográficos existentes de algoritmos criptográficos para Java que se han encontrado. Un total de 16 proveedores.
- Como apoyo al proyecto: se han reutilizado algunas herramientas y librerías Java.
- Para el desarrollo del proyecto: Se ha utilizado principalmente el entorno de desarrollo para Java *"Eclipse"* y también algunos de sus *plugins*.

- Como herramientas de documentación: se utilizó la suite de ofimática *"OpenOffice"*, *la utilidad de generación de documentación para código Java "Javadoc"*, programa de diseño UML llamado *"ArgoUML"*, y otras llamadas *"PDFCreator" y "Java2HTML"*.
- Para diseñar la página web del proyecto: *"Nvu"* y una plantilla para dicha página web.
- Como navegador web para Internet: *"Mozilla Firefox"*.
- Para publicar el proyecto en Internet: *"WinSCP"*, *"FileZilla"* y *"SourceForge.net"*.
- Como lenguajes de desarrollo: *"Java"*, *"HTML"*, *"XML"*, *"UML"* y *"CSS"*.
- Y otros recursos como *"7-zip"*, *"AlZip"*, "CDBurnerXP Pro", *"JEdit"*, *"Notepad++"*, *"GIMP"*, *"Irfanview"*, *"Paint.NET"*, *"Directrices de desarrollo"*, *"NotesHolder Lite"*, *"JSmooth"* y *"Cobian Backup"*.

Para mayor información sobre este proyecto se puede consultar la web http://jcef.sourceforge.net y también puede encontrar información completa sobre los recursos utilizados en http://jcef.sourceforge.net/doc/resources.pdf.

También se podrá encontrar en este documento una sección llamada «Preguntas frecuentes» que incluye las preguntas más frecuentes que el lector de este proyecto puede hacerse.

Quizás, la pregunta más importante podría ser: "¿Por qué no se ha desarrollado un conjunto de aplicaciones?" Es decir, "¿porque no se ha cumplido uno de los dos objetivos de este proyecto?". Respondiendo de forma corta, se podría podría decir que: "Porque tras realizar el análisis surgió una idea mejor". La respuesta más larga sería: "La razón de existencia del segundo objetivo previsto «Mostrar aplicaciones de las técnicas criptográficas mediante software desarrollado en el proyecto» simplemente era el de probar que realmente se había alcanzado el primer objetivo: «Aprender a utilizar técnicas criptográficas». Tras finalizar el estudio preliminar, se consideró que era mucho mejor desarrollar algo nuevo, antes que realizar un conjunto de aplicaciones representativas de la criptografía, lo cual no es nada novedoso. Además, ya existen herramientas muy buenas como "CrypTool" (http://www.cryptool.com/), TrueCrypt (http://www.truecrypt.org/) y AxCrypt (http://axcrypt.sourceforge.net) de código abierto y otras freeware tales como "EncryptOnClick" y "FingerPrint" (http://www.2brightsparks.com/).".

También son interesantes las siguientes cuestiones que se responderán aquí de forma breve:

- ¿Cuál es el tamaño del proyecto? Su tamaño es bastante grande ya que está compuesto por 3142 clases, 315 campos, 2914 métodos, 73959 líneas de código y 22941 líneas de comentarios.
- ¿Cuál ha sido el coste del proyecto? Económico ninguno al utilizar herramientas gratuitas pero el coste de tiempo se estima entre 2000 y 3000 horas; más de un año de trabajo sin pausa.
- ¿Qué metodología se ha utilizado? Se ha utilizado la clásica (análisis, diseño, implementación y pruebas) con retroalimentaciones.

Para mayor información sobre el proyecto puede consultar la web http://jcef.sourceforge.net y si desea consultar más preguntas frecuentes puede acceder directamente a ellas dirigiéndose a la dirección http://jcef.sourceforge.net/doc/faqs.pdf.

En resumidas cuentas, las conclusiones más importantes sobre el proyecto en sí son los principales aspectos valorables positivamente:

1. Haber desarrollado un trabajo novedoso en lugar de lo que se tenía previsto.
2. El uso 100% de herramientas Open Source o Freeware.
3. Haber aplicado el conocimiento de varias áreas: Ingeniería del software, Gráficos, Programación, Ofimática, Programación web, etc...
4. El enfoque orientado a objetos de la criptografía.
5. Publicación del proyecto en una página web (http://jcef.sourceforge.net).
6. El diseño de imágenes propias originales.
7. Haber sido capaz de trabajar en el proyecto sin prisas y sin pausas durante algo más de un año.
8. Las propuestas realizadas sobre posibles futuros proyectos fin de carrera.
9. La sección de preguntas frecuentes como ayuda adicional.

También es importante conocer algunos puntos que podrían considerarse como negativos. Éstos son los siguientes:

1. El tiempo empleado en el proyecto podría considerarse excesivo.
2. No haber desarrollado exactamente lo que estaba previsto desde un principio.

En las siguientes secciones se podrá ver información más detallada sobre el proyecto, comenzando por un par de capítulos de introducción tanto a la seguridad como a la criptografía, luego otro capítulo en el que se habla del proyecto en sí; seguidamente se continúa con una sección titulada “Futuros proyectos”, para posteriormente continuar con el capítulo donde se detallan los recursos utilizados en este proyecto; y finalmente un capítulo donde se colocan las preguntas frecuentes que puede hacerse el lector del proyecto.

Recuerde el lector que toda la información sobre este proyecto y más se puede encontrar en la dirección web http://jcef.sourceforge.net.

2.Introducción a la seguridad

Las necesidades de seguridad de la información han ido evolucionando al igual que las ciencias de la computación y las tecnologías de la información. De este modo, las herramientas de seguridad empleadas también lo han hecho.

En sus comienzos, la seguridad consistía en el uso de medios físicos, tales como cajas fuertes o armarios con cierre de seguridad. Poco después, aparecieron los medios administrativos, como lo son los contratos de empleados para salvaguardar información confidencial de la empresa.

Con la llegada de la computación, hicieron falta nuevas herramientas de seguridad diseñadas para proteger los datos y evitar la intrusión de los hackers. Estas herramientas se conocen con el nombre de "seguridad informática".

Finalmente, con la aparición de Internet, las redes locales y los sistemas distribuidos, surgió la necesidad de disponer de seguridad durante la transmisión. Las herramientas que satisfacen estas nuevas necesidades se engloban bajo la denominación "seguridad en redes". A continuación se puede observar la evolución en el tiempo tanto de la seguridad como de la informática.

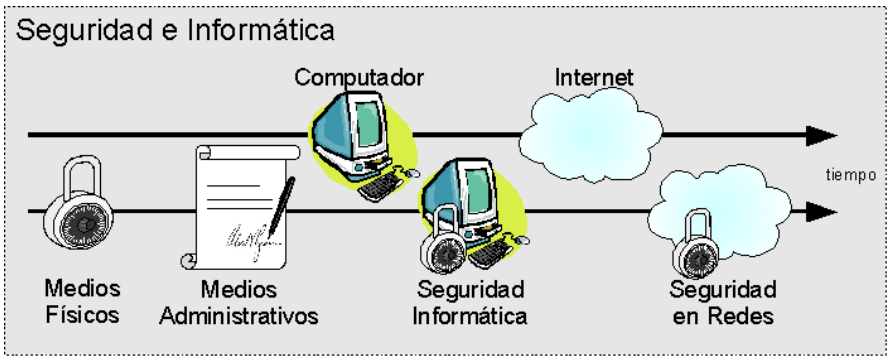


Ilustración 2: Evolución en el tiempo de la seguridad y la informática

Para mayor información puede consultar los recursos bibliográficos utilizados; a los cuales también se puede acceder desde http://jcef.sourceforge.net/doc/resources.pdf.

1 Una arquitectura de seguridad

Para analizar de forma efectiva las necesidades de seguridad de una organización, evaluar y elegir distintos productos y políticas de seguridad, el responsable de la seguridad necesita una forma sistemática de definir los requisitos de seguridad y caracterizar los enfoques para satisfacer dichos requisitos.

Uno de estos posibles enfoques interesantes se centra en los ataques a la seguridad, los mecanismos y los servicios.

Un ataque a la seguridad es cualquier acción que comprometa la seguridad de la información de una organización. Los ataques a la seguridad se detectan, eliminan, previenen y/o se reestablece el sistema de su daño, siendo los encargados de estas funciones los mecanismos de seguridad.

Por el contrario, un servicio de seguridad es un servicio que mejora la seguridad de los sistemas, contrarrestando los ataques a la seguridad, haciendo uso de uno o más mecanismos y políticas de seguridad con el objetivo de proporcionar el servicio.

En resumen, los componentes de una arquitectura de seguridad junto con sus objetivos son:

Componentes	Objetivo
Ataque a la seguridad	Comprometer la seguridad de la información de un sistema
Servicio de seguridad	Contrarrestar los ataques a la seguridad
Mecanismo de seguridad	Ayudar a contrarrestar los ataques a la seguridad

Tabla 2: Componentes de una arquitectura de seguridad y sus objetivos

2 Ataques a la seguridad

Los ataques a la seguridad se pueden clasificar en ataques pasivos y activos. En la siguiente tabla se muestran cada una de ellos junto con sus objetivos primarios y sus soluciones:

Características		**Ataques**	
		Pasivos	**Activos**
Objetivo	**¿Acceden a recursos?**	Sí	Sí
	¿Obtienen información?	Sí	Sí
	¿Alteran el sistema?	No	Sí
	¿Alteran recursos?	No	Sí

Tabla 3: Comparativa entre Ataques Pasivos y Activos (1/2)

Características		Ataques	
		Pasivos	Activos
Solución	¿Se pueden prevenir?	Sí	No
	¿Se pueden detectar?	No	Sí
	¿Se pueden recuperar?	No	Sí

Tabla 4: Comparativa entre Ataques Pasivos y Activos (2/2)

1 Ataques pasivos

De la misma forma, existen varios tipos de ataques pasivos. En la siguiente tabla se muestran cada uno de ellos junto con sus objetivos básicos:

Ataques Pasivos	Objetivos
Obtención de contenidos de mensajes	Copiar información
Análisis del Tráfico	Averiguar la naturaleza de la comunicación
	Desvelar la identidad de los comunicantes

Tabla 5: Ataques Pasivos y sus Objetivos

También es interesante descatar los otros nombres de estos tipos de ataques:

Ataques Pasivos	Alias
Ataques pasivos	Ataques de intercepción
Obtención de contenidos de mensajes	Intercepción de datos
Análisis del Tráfico	Intercepción de entidad

Tabla 6: Otros nombres de los ataques pasivos

En las siguientes ilustraciones se pueden ver cada uno de los diagramas de los ataques pasivos existentes:

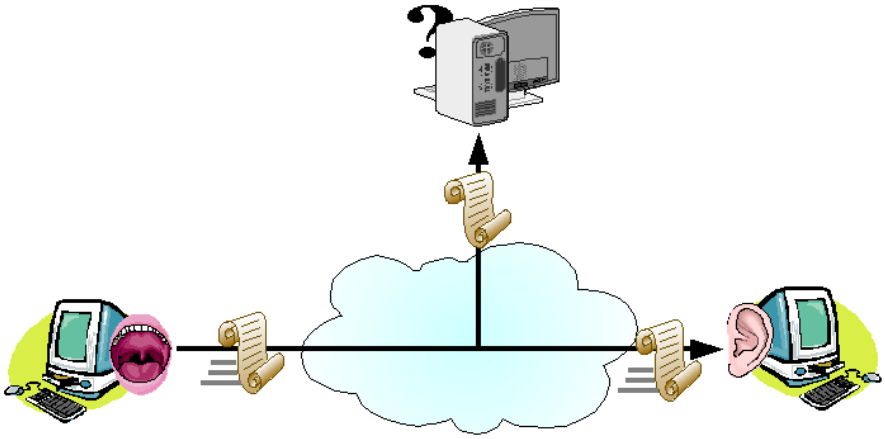


Ilustración 3: Obtención del contenido de mensajes

Un ejemplo de ataque pasivo mediante obtención de contenidos de mensajes podría ser: "Un usuario A envía un archivo a otro usuario B. El archivo contiene información confidencial que debe protegerse, por ejemplo los registros de nóminas. Otro usuario C, que no está autorizado a leer el archivo, observa la transmisión y captura una copia del archivo durante dicha transmisión".

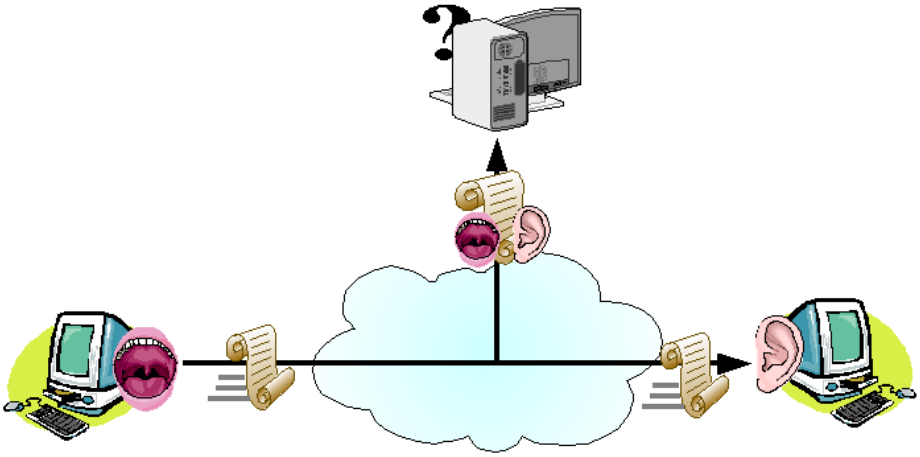


Ilustración 4: Análisis del tráfico

Para el caso de ataques basado en el análisis del tráfico, el ejemplo sería equivalente al anterior, excepto que en lugar de analizar el contenido de los paquetes que se transmiten a través de la red, se observan las cabeceras de cada uno de ellos.

2 Ataques activos

Existen varios tipos de ataques activos, que junto con sus objetivos, se mencionan a continuación:

Ataques Activos	Objetivos
Suplantación de identidad	Fingir ser otra entidad
Repetición	Retransmitir mensajes
Modificación de mensajes	Alterar, retrasar y/o reordenar mensajes
Interrupción del servicio	Dejar fuera de servicio algún recurso del sistema

Tabla 7: Ataques Activos y sus objetivos

Igualmente es interesante destacar los otros nombres de estos tipos de ataques:

Ataques activos	Alias
Suplantación de identidad	Falsificación de identidad
Repetición	Reactuación
Modificación de mensajes	Alteración de mensajes
Interrupción del servicio	Degradación fraudulenta del servicio

Tabla 8: Otros nombres de los ataques activos

A continuación, se pueden observar cada uno de los diagramas para cada uno de los ataques activos existentes:

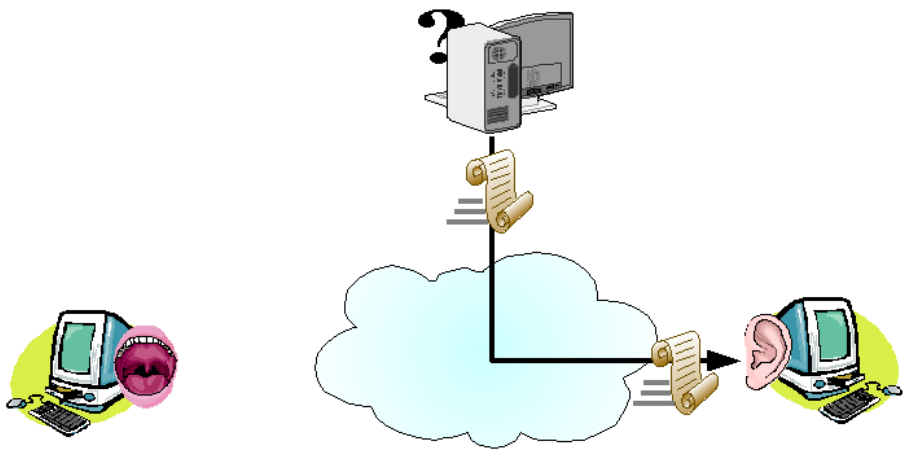


Ilustración 5: Suplantación de identidad

Un ejemplo de ataque por suplantación de identidad podría ser: "Más allá de interceptar un mensaje, un usuario podría construir un mensaje con las entradas deseadas, transmitiéndolo posteriormente a otro usuario como si procediera de la entidad suplantada, como por ejemplo la entidad de un administrador. Por consiguiente, el usuario receptor acepta el mensaje y reacciona ante el mensaje como si del administrador procediera. El mensaje podría ser cualquier tipo de información, como un fichero de los usuarios que están autorizados o no".

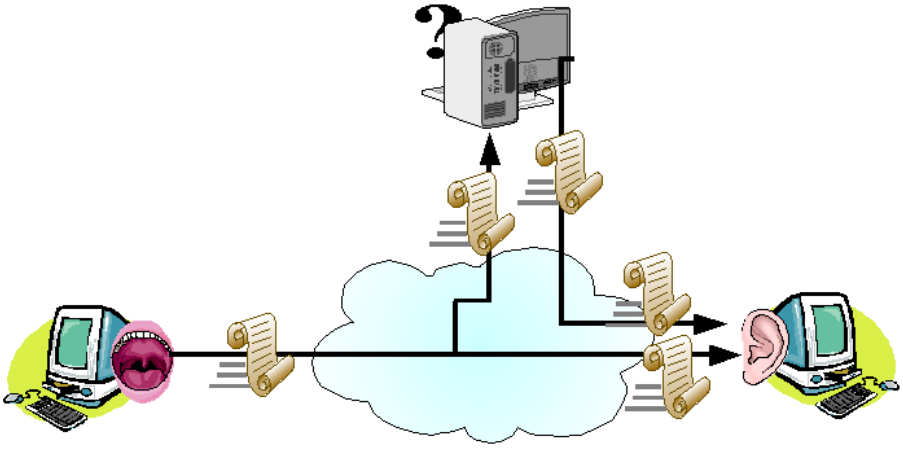


Ilustración 6: Repetición de mensajes

Por otro lado, un ejemplo de una posible consecuencia de un ataque mediante repetición de mensajes sería: "Haber ingresado dinero repetidas veces en una cuenta dada".

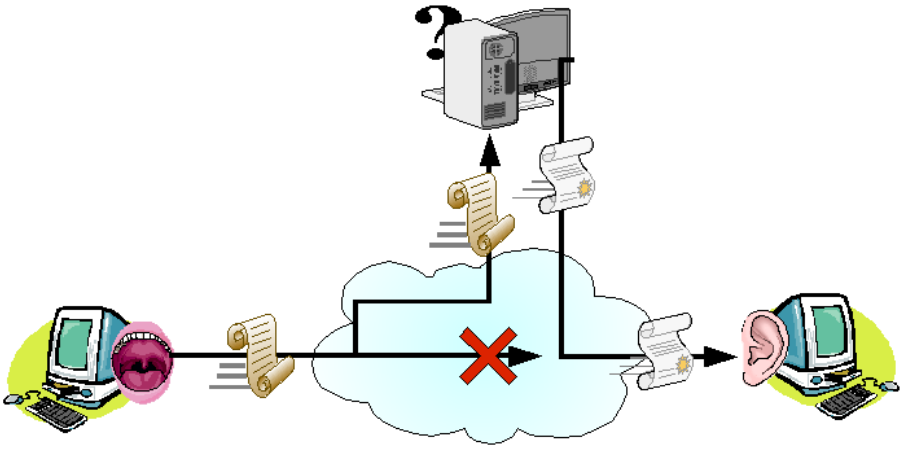


Ilustración 7: Modificación de mensajes

Para el caso de ataques mediante modificación de mensajes, algunos de sus objetivos podrían ser: "alterar un programa para que funcione de forma diferente" o "modificar una orden", por ejemplo, en lugar del mensaje "Ingresa un millón de pesetas en la cuenta A", podría ser modificado para decir "Ingresa un millón de pesetas en la cuenta B".

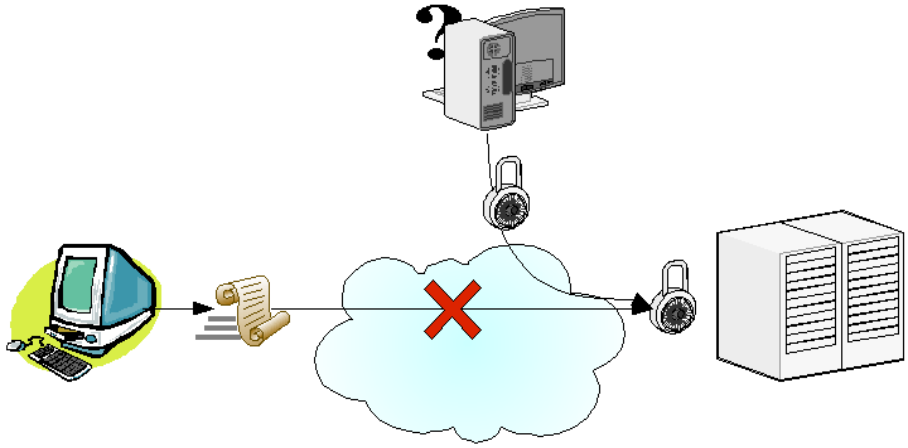


Ilustración 8: Interrupción de servicio

Finalmente, ejemplos de ataques mediante interrupción de servicio son: "Suprimir todos los mensajes dirigidos a una determinada entidad", "Interrumpir el servicio de una red inundándola con mensajes espurios" o "Deshabilitar el sistema de gestión de ficheros".

3 Servicios de seguridad

Un servicio de seguridad es un servicio que mejora la seguridad de los sistemas, contrarrestando los ataques a la seguridad, haciendo uso de uno o más mecanismos y políticas de seguridad con el objetivo de proporcionar el servicio. También pueden definirse como atributos deseables para que un sistema pueda considerarse seguro. Los principales servicios de seguridad, junto con sus objetivos son:

Servicios de Seguridad	**Objetivos**
Autentificación	Garantizar la procedencia de los objetos
Control de Acceso	Prevenir el uso no autorizado de los objetos
Confidencialidad	Proteger los objetos ante entidades no autorizadas
Integridad	Garantizar que no se puedan modificar los objetos sin ser detectado dicho hecho
No repudio	Asegurar que el receptor recibió los objetos
	Asegurar que el emisor envió los objetos
Disponibilidad	Asegurar que los objetos estén disponibles, evitando que entidades no autorizadas afecten a dicha disponibilidad

Tabla 9: Servicios de Seguridad y sus Objetivos

Un símil entre los servicios de seguridad y la vida cotidiana se establece en la siguiente tabla:

Servicios de Seguridad	**Ejemplos de la vida cotidiana**
Autentificación	DNI
Control de acceso	Llaves y cerrojos
Confidencialidad	Tinta invisible
Integridad	Tinta indeleble
No repudio	Firma notarizada

Tabla 10: Los Servicios de Seguridad y la vida cotidiana

Cada uno de los servicios de seguridad está pensado para hacer frente a uno o varios ataques. La siguiente tabla muestra dichas relaciones:

Servicios de Seguridad	**Ataques defendidos**
Autentificación	Suplantación de identidad
Control de Acceso	Suplantación de identidad
Confidencialidad	Obtención del contenido de mensajes
	Análisis del tráfico
Integridad	Repetición
	Modificación de mensajes
No repudio	——
Disponibilidad	Interrupción de servicio

Tabla 11: Ataques defendidos por los Servicios de Seguridad

4 Mecanismos de seguridad

Los mecanismos de seguridad son utilizados por los servicios de seguridad con el objetivo de contrarrestar los ataques a la seguridad. Es decir, para que los servicios de seguridad hagan frente a los ataques, es necesario que utilicen una serie de mecanismos de seguridad:

Servicios de Seguridad	Mecanismos utilizados
Autentificación	Criptografía
Control de Acceso	
Confidencialidad	
Integridad	
No repudio	
Disponibilidad	

Tabla 12: Mecanismos utilizados por los servicios

La criptografía es el mecanismo de seguridad primordial de todo sistema de seguridad, pero no es el único. Se pueden utilizar otros mecanismos tales como: el control de acceso, el intercambio de autentificación, la notarización, el control de enrutamiento, las auditorías de seguridad, la funcionalidad fiable mediante políticas de seguridad, las etiquetas de seguridad, la detección de acciones, los mecanismos de recuperación, etc.

El único mecanismo de seguridad que se tratará será la criptografía, cuyo único objetivo es construir objetos seguros.

3.Criptografía Orientada a Objetos

La criptografía, vista en términos sociales, es la ciencia que trata de que el coste de adquirir o alterar objetos de modo impropio sea mayor que el posible valor obtenido al hacerlo. Desde un punto de vista más formal, la criptografía es la práctica y el estudio de técnicas para asegurar objetos haciéndolos incomprensibles y/o autentificables y recuperarlos a partir de dicha versión incomprensible y/o autentificable.

Un objeto puede ser cualquier cosa: información, recursos, datos, mensajes, ficheros, un objeto seguro, etc... Desde el punto de vista de la criptografía, se distinguen los siguientes tipos de objetos:

Un objeto es ...	Normal	... sólo si ...	Puede ser inseguro
	Seguro		Está protegido y/o es autentificable
	Protegido		Sus propiedades son incomprensibles
	Auténtico		Su origen es auténtico y/o si es íntegro
	Íntegro		Sus propiedades son auténticas
	Autentificable		Puede comprobarse su autenticidad

Tabla 13: Tipos de objetos criptográficos

Para alcanzar un grado mayor de comprensión, véase la siguiente tabla donde aparecen cada una de las características de dichos objetos:

Tipos de Objeto	**Características**					
	Propiedades				**Origen**	
	Accesibles	**Entendibles**	**Auténticas**	**Autentificables**	**Auténtico**	**Autentificable**
Normal	Siempre	Siempre	A veces	Nunca	A veces	Nunca
Seguro	Siempre	A veces	A veces	A veces	A veces	A veces
Protegido	Siempre	Nunca	A veces	Nunca	A veces	Nunca
Auténtico	Siempre	Siempre	A veces	Nunca	A veces	Nunca
Autentificable	Siempre	Siempre	A veces	A veces	A veces	A veces

Tabla 14: Características de los objetos

Para mayor información puede consultar las referencias bibliográficas sobre este tema; a las cuales también se puede acceder desde http://jcef.sourceforge.net/doc/resources.pdf.

1 Operaciones criptográficas

En general, la criptografía proporciona una serie de operaciones con el único objetivo de construir objetos seguros y obtener con posterioridad el objeto asegurado a partir de su versión segura. Estas operaciones junto con sus funciones son:

Operaciones Criptográficas	Función
Protección	Construir la versión protegida de un objeto dado
Desprotección	Obtener el objeto original a partir de su versión protegida
Autentificación	Construir la versión autentificable de un objeto dado
Verificación de autenticidad	Obtener el objeto original a partir de su versión autentificable sólo si es auténtico

Tabla 15: Operaciones Criptográficas y sus funciones

1 Protección

La protección genera objetos protegidos, los cuales se pueden generar por dos procesos ligeramente distintos: la protección simétrica o basada en el uso de claves secretas y la protección asimétrica o basada en el uso de claves públicas y privadas.

En resumen, estas operaciones utilizan una serie de claves tanto para la protección como la desprotección, tal y como se observa en la siguiente tabla.

Claves de ...	Protección	
	Simétrica	**Asimétrica**
Protección	Secreta	Pública del receptor
Desprotección	Secreta	Privada del receptor

Tabla 16: Categorías de protección y sus claves

Para dar mayor luz al asunto, véanse las siguientes ilustraciones.

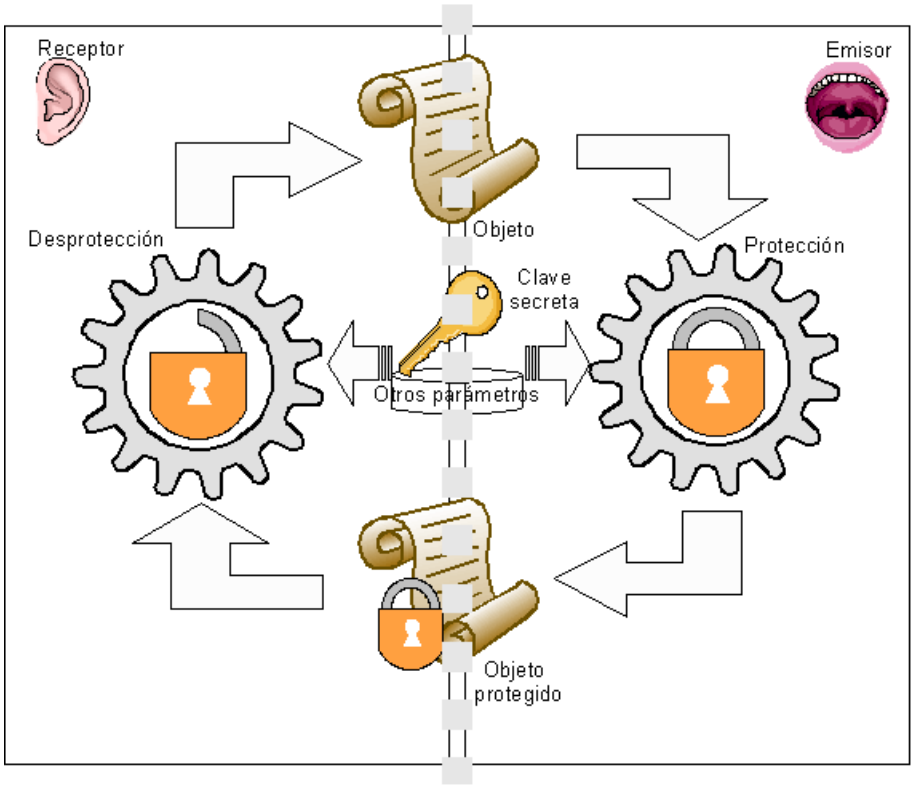


Ilustración 9: Protección y Desprotección Simétricas

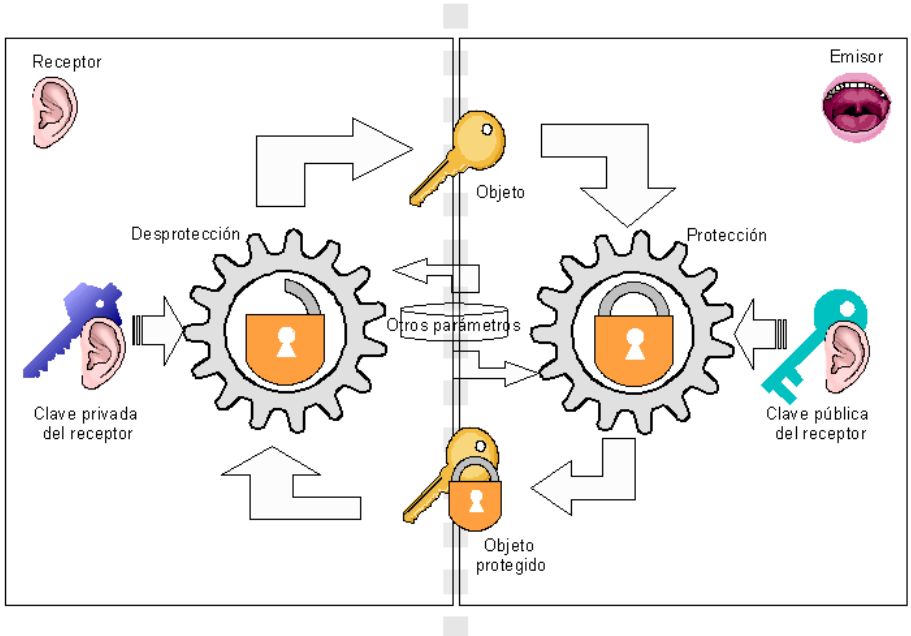


Ilustración 10: Protección y Desprotección Asimétricas

En el caso de la protección simétrica, existen a su vez dos categorías dignas de mención, la simétrica de bloques y la de flujo.

La protección simétrica de bloques, divide el objeto en partes iguales, protegiendo cada una de ellas. Además, este tipo de protección utiliza en exclusividad modos de operación y esquemas de relleno. El modo de operación define el tamaño que deben tener cada una de las partes y la forma en la que se protege/desprotege cada una de ellas. Existen numerosos modos de operación. Algunos son rápidos, eficientes pero poco seguros y otros son menos eficientes pero más seguros. El esquema de relleno es utilizado para rellenar la última parte con el objetivo de que todas las partes del objeto posean el mismo tamaño. Esto suele ser necesario ya que generalmente el tamaño del objeto no suele ser múltiplo del tamaño predefinido por el modo de operación. Existen numerosos esquemas de relleno cada uno de ellos con distintas características.

Por otro lado, la protección simétrica de flujo no divide al objeto en partes, sino que lo trata como si fuera una única entidad.

2 Autentificación

La autentificación genera objetos autentificables, los cuales están formados por el objeto a autentificar y otro objeto llamado autentificador, cuyas propiedades son las siguientes:

¿Es un código representativo del objeto?	Sí
¿Exclusivo del objeto correspondiente?	Sí
¿Debe asociarse al objeto al que pertenece?	Sí
¿Por sí solo sirve para algo?	No
¿Sirve para otros objetos?	No
¿Sólo se utiliza para la verificación de la autenticidad?	Sí

Tabla 17: Propiedades de todo autentificador

Además, existen varios tipos de autentificadores, los cuales se describen a continuación:

Tipos de Autentificadores	**Descripción**
Huella digital de un objeto	Código de tamaño fijo y representativo del objeto
Sello digital de un objeto	Huella digital protegida simétricamente
Firma digital de un objeto	Huella digital protegida asimétricamente

Tabla 18: Tipos de autentificadores y sus descripciones

Tanto para la autentificación como la verificación de la autenticidad, estas operaciones utilizan una serie de claves tal y como figura a continuación.

Claves de ...	Autentificación mediante ...		
	Huella	**Sello**	**Firma**
Autentificación	Ninguna	Secreta	Privada del emisor
Verificación	Ninguna	Secreta	Pública del emisor

Tabla 19: Categorías de Autentificación y sus claves

En las siguientes ilustraciones pueden verse cada uno de los objetos autentificables en base al tipo de autentificador utilizado para su generación y posterior verificación de la autenticidad.

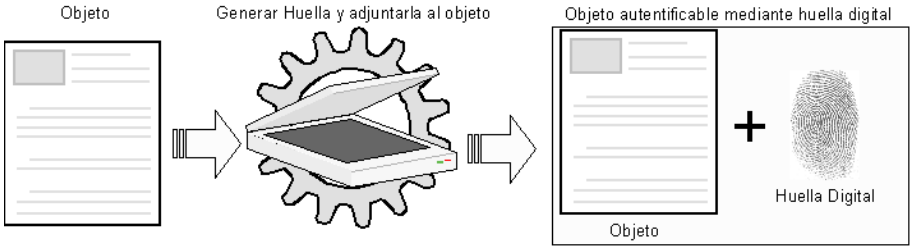


Ilustración 11: Objeto Autentificable mediante Huella Digital

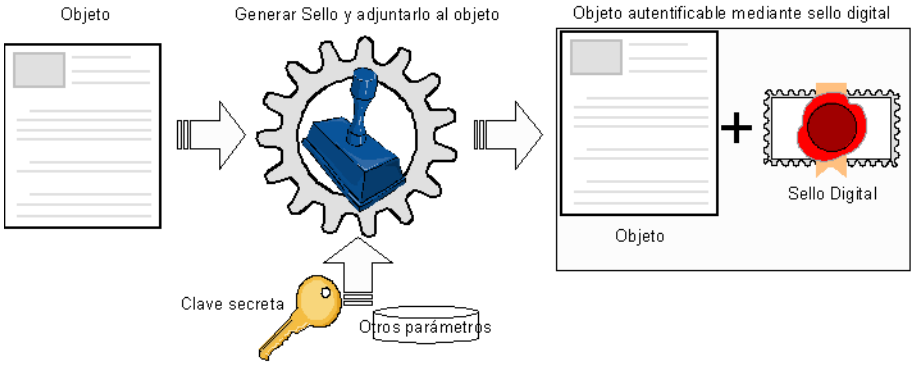


Ilustración 12: Objeto Autentificable mediante Sello Digital

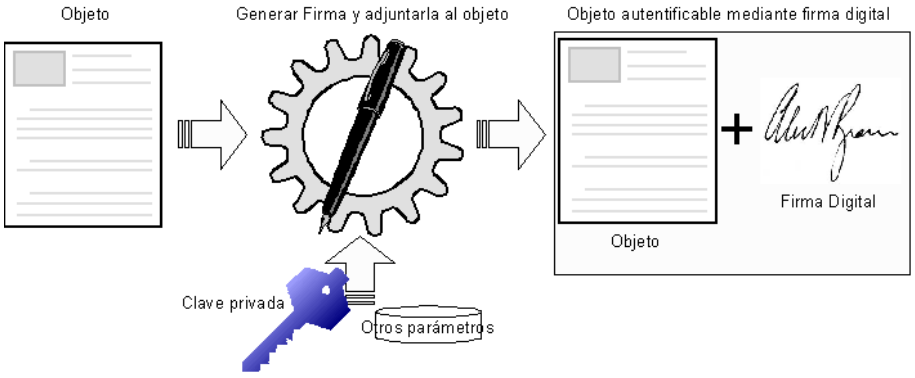


Ilustración 13: Objeto Autentificable mediante Firma Digital

3 Aclaraciones sobre las claves

Los tipos de claves requieren una serie de aclaraciones adicionales, siendo éstas las siguientes:

Características	Claves		
	Secreta	**Pública**	**Privada**
¿Compartida?	Siempre	Siempre	Nunca
¿Publicada?	Nunca	Siempre	Nunca
¿Perteneciente a un único propietario?	No	Sí	Sí
¿Exclusiva de su propietario?	Nunca	No	Siempre
¿Utilizada por el propietario?	Sí	No	Sí

Tabla 20: Aclaraciones adicionales sobre los distintos tipos de claves (1/2)

Otras aclaraciones adicionales son:

Claves	Características	
	¿Qué clave invierte su efecto?	**¿Cómo se genera?**
Secreta	Sólo la misma clave secreta	Individualmente
Pública	Sólo su clave privada asociada	En pareja, junto con su clave privada
Privada	Sólo su clave pública asociada	En pareja, junto con su clave pública

Tabla 21: Aclaraciones adicionales sobre los distintos tipos de claves (2/2)

4 Sus problemas y sus soluciones

En la siguiente tabla se muestran los problemas de los que adolecen las operaciones criptográficas:

Operaciones Criptográficas		Problema
Protección	**Simétrica**	Privacidad/Distribución de la clave secreta
	Asimétrica	Falsificación de la clave pública
Autentificación	**Huella**	No asegura su propia integridad
	Sello	Privacidad/Distribución de la clave secreta
	Firma	Falsificación de la clave pública

Tabla 22: Problemas de las operaciones criptográficas

A continuación se mencionan las soluciones a cada uno de los problemas de las operaciones criptográficas:

Operaciones		Soluciones
Protección	**Simétrica**	Protección asimétrica de las claves secretas
	Asimétrica	Notarización basada en certificados
Autentificación	**Huella**	Asegurar la huella mediante protección simétrica
		Asegurar la huella mediante protección asimétrica
		Sin protección. Generar la huella digital no del objeto, sino de la unión de éste con un secreto compartido.
	Sello	Protección asimétrica de las claves secretas
	Firma	Notarización basada en certificados

Tabla 23: Soluciones a los problemas de las operaciones criptográficas

5 Sus características

Las principales características de cada una de las operaciones criptográficas son:

Características	Protección		Autentificadores		
	Simétrica	**Asimétrica**	**Huella**	**Sello**	**Firma**
Nivel de Seguridad	Alto	Muy alto	Bajo	Medio	Muy alto
Velocidad de Procesamiento	Alta	Muy baja	Alta	Media	Muy baja
Longitud de las claves	Pequeña	Grande	Ninguna	Pequeña	Grande
Vida aconsejable de las claves	Corta	Larga	Ninguna	Corta	Larga
Número de claves	Elevado	Reducido	Ninguna	Elevado	Reducido
¿Adecuada para objetos grandes?	Sí	No	Sí	Sí	Sí
¿Adecuada para objetos pequeños?	Sí	Sí	Sí	Sí	Sí

Tabla 24: Características principales de las operaciones criptográficas

6 Sus aplicaciones

Las principales aplicaciones de cada una de las operaciones criptográficas son:

Operaciones		Aplicaciones
Protección	**Simétrica**	Protección de objetos grandes
	Asimétrica	Distribución de claves secretas
Autentificación	**Huella**	Asegurar la autenticidad de los objetos
	Sello	Asegurar la autenticidad de los objetos
	Firma	Asegurar la autenticidad de los objetos

Tabla 25: Aplicaciones principales de las operaciones criptográficas

7 Servicios de seguridad

Las operaciones criptográficas son mecanismos utilizados por algunos servicios de seguridad tal y como se muestra en la siguiente tabla:

Servicios de Seguridad	Protección		Autentificación		
	Simétrica	**Asimétrica**	**Huella**	**Sello**	**Firma**
Confidencialidad	Siempre	Siempre	Nunca	Nunca	Nunca
Autentificación	A veces	Siempre	Nunca	A veces	Siempre
Integridad	Nunca	Nunca	Siempre	Siempre	Siempre
No repudio	Nunca	Siempre	Nunca	Nunca	Siempre
Disponibilidad	A veces	A veces	A veces	A veces	A veces
Control de Acceso	A veces	A veces	A veces	A veces	A veces

Tabla 26: Servicios de seguridad y las operaciones criptográficas

2 La Seguridad de la Criptografía

También es muy importante conocer el grado de seguridad de los algoritmos criptográficos utilizados por las operaciones criptográficas y para ello se emplea el criptoanálisis como conjunto de procedimientos, procesos y métodos empleados para romper un algoritmo criptográfico.

1 Algoritmos seguros

En general, para que la criptografía sea un mecanismo de seguridad seguro, el algoritmo empleado debe cumplir las siguientes propiedades:

1	La clave empleada debe mantener su privacidad
2	Debe ser robusto
	En el caso de un algoritmo de autenficación, además debe cumplirse:
	No es posible averiguar el objeto basándose sólo en su autentificador
	Dado un autentificador debe ser imposible encontrar un objeto que lo genere
	Debe ser computacionalmente imposible encontrar dos objetos con el mismo autentificador
3	Debe poseer un nivel de confianza alto
4	Debe pasar por una revisión completa
5	Debe ser seguro computacionalmente, es decir, cumplir al menos uno de los siguientes criterios:
	El coste de romper el algoritmo excede el valor de la información obtenida
	El tiempo necesario para romper el algoritmo excede el tiempo de vida útil de la información
6	Debe disponer de un número muy elevado de claves posibles
7	Debe producir objetos seguros que parezcan aleatorios a un test estadístico estándar
8	Debe ser resistente al criptoanálisis

Tabla 27: Propiedades generales de un algoritmo criptográficamente seguro

Actualmente, no existe ningún algoritmo seguro incondicionalmente, es decir, no existe ningún algoritmo que resista ataques disponiéndose de todos los recursos y gran cantidad de objetos seguros.

Además, es necesario saber que existen una serie de criterios para seleccionar el algoritmo criptográfico más adecuado:

1	A mayor tamaño de bloque, mayor nivel de seguridad
2	A mayor longitud de clave, mayor seguridad
3	Generalmente, a mayor seguridad, menor velocidad de procesamiento
4	La longitud del autentificador es directamente proporcional a la seguridad
5	Si se desea asegurar el contenido de los objetos, elegir algoritmos de protección
6	Si se desea asegurar la autenticidad de los objetos, elegir algoritmos de autentificación: huellas, sellos o firmas digitales

Tabla 28: Criterios de selección de algoritmos criptográficos

2 Ataques

Los algoritmos criptográficos se someten a una serie de ataques con el objetivo de romperlos. Algunos de los más importantes sobre algoritmos criptográficos son: «Criptoanálisis diferencial», «Criptoanálisis lineal», «Explotación de claves débiles», «Ataques algebraicos», «Complejidad lineal», «Ataques de correlación», «Ataque del cumpleaños», «Explotación de las pseudocolisiones», «Fuerza Bruta», «Fuerza Bruta basada en diccionario», «Ataque cíclico», «Factorización de la clave pública», «Ataque de Merkle-Hellman», «Ataque basado en oráculos», «Ataque por control de tiempos», «Ataque por introducción de faltas», «Ataque por introducción de canales subliminales en las claves», etc...

3 Aplicaciones de la criptografía

La criptografía es una disciplina con multitud de aplicaciones; muchas de las cuales están en uso hoy en día. Todas ellas simplemente lo que hacen es manejar objetos seguros, en lugar de simplemente objetos. Entre las más importantes se destacan las siguientes: Seguridad de las comunicaciones, Identificacion y Autentificación, Protección de software y Correo Electrónico.

1 Seguridad de las comunicaciones

Seguridad de las comunicaciones	Aplicaciones
Entre ordenadores	Protocolos SSL, TLS y SET
Telefónicas	Comunicaciones móviles seguras AT&T, chip criptográfico Clipper
Correo electrónico	Protocolos PGP y PEM

Tabla 29: Aplicaciones sobre "Seguridad de las comunicaciones"

2 Identificación y Autentificación

Identificación y Autentificación	Aplicaciones
Autentificación de Usuarios	Kerberos
Certificación	Certificación X.509

Tabla 30: Aplicaciones sobre "Identificación y Autentificación"

3 Protección de software

Protección de software	Aplicaciones
Evitar la ingeniería inversa	Cargador de clases protegidas
Evitar usos malintencionados	Paquetes firmados digitalmente

Tabla 31: Aplicaciones sobre "Protección de software"

4 Comercio Electrónico

Comercio Electrónico	Aplicaciones
Aplicaciones bancarias	Cajeros automáticos, Webs de bancos
Transacciones electrónicas	Interbancarias, Interpersonales, Entre individuos y bancos, Entre individuos y comercios, Entre comercios
Sistemas de pago electrónicos	Dinero electrónico, Cheques electrónicos, Pagos con tarjeta de crédito, Micropagos
Intercambio Electrónico de Documentos	Pedidos, Avisos de pagos, Facturas, Informes de ventas, etc.

Tabla 32: Aplicaciones sobre "Comercio Electrónico"

4 Conceptos técnicos

En este capítulo y en general en todo el proyecto se ha evitado al máximo el uso de conceptos o términos técnicos ya que son totalmente innecesarios. Sin embargo, para aquel lector que quiera profundizar en los detalles de la criptografía, bien le valdría saber las siguientes correspondencias entre los conceptos utilizados y sus términos técnicos asociados. Esta correspondencia se muestra a continuación:

Concepto	Términos técnicos
Operación criptográfica	Algoritmo criptográfico
Protección	Cifrado, Encriptado
Desprotección	Descrifrado, Desencriptado
Protección simétrica	Cifrado simétrico, Cifrado de clave secreta, Criptografía de clave secreta, Criptografía simétrica, Algoritmo de clave secreta, Algoritmo simétrico
Protección asimétrica	Cifrado asimétrico, Cifrado de clave pública, Criptografía de clave pública, Criptografía asimétrica, Algoritmo de clave pública, Algoritmo asimétrico

Tabla 33: Conceptos y sus términos técnicos (1/2)

Concepto	Términos técnicos
Autentificación	Cifrado irreversible
Firma digital	Digital Signature, Algoritmo de firma digital, Signature
Huella digital	Resumen de mensaje, Algoritmo de resumen de mensajes, Función de dispersión criptográfica, Message digest, Digest, Hash, Función hash unidireccional, FingerPrint
Sello digital	Código de Autentificación de Mensajes, Message Authentication Code, MAC, Digital Seal
Objeto	Texto claro, Plain text, Mensaje
Objeto seguro	Texto cifrado, Cipher text

Tabla 34: Conceptos y sus términos técnicos (2/2)

4.El proyecto

Desde el punto de vista de la seguridad, el conjunto de clases de seguridad distribuidas con el Java 2 SDK pueden dividirse en dos subconjuntos: clases relacionadas con el control de acceso, la gestión de permisos y las clases relacionadas con la criptografía.

Java proporciona funciones criptográficas de propósito general, conocidas colectivamente como la *Arquitectura Criptográfica de Java (JCA)* y la *Extension Criptográfica de Java (JCE)*. El JCA está formado por las clases básicas relacionadas con la criptografía y el soporte para la protección lo proporciona el paquete de extensión JCE.

La librería JCA es un marco de trabajo para acceder y desarrollar funciones criptográficas en la plataforma Java. Se diseñó alrededor de dos principios básicos, la independencia e interoperabilidad de las implementaciones y la independencia y extensibilidad de los algoritmos.

Las principales funciones criptográficas implementadas son: encriptar datos, desencriptar datos, encriptar claves, desencriptar claves, generar claves, generar otros parámetros, realizar conversiones entre distintas representaciones de los parámetros y de las claves, generar y verificar resúmenes de mensajes, códigos de autentificación de mensajes y firmas digitales.

Sin embargo, utilizar la arquitectura criptográfica de Java (JCA) y su extensión (JCE) resulta demasiado complicado. Es por ello por lo que se ha decidido desarrollar una nueva librería criptográfica llamada JCEF (Java Cryptographic Extension Framework), en lugar de mostrar un conjunto de aplicaciones concretas de la criptografía. JCEF se soporta sobre JCA y JCE facilitando enormemente su uso. Esta nueva librería sería mayormente una fachada de las librerías JCA y JCE. El cliente de esta nueva librería sería el desarrollador de software que necesita usar sistemas criptográficos de seguridad con suma facilidad sin las complejidades inherentes a JCA y JCE.

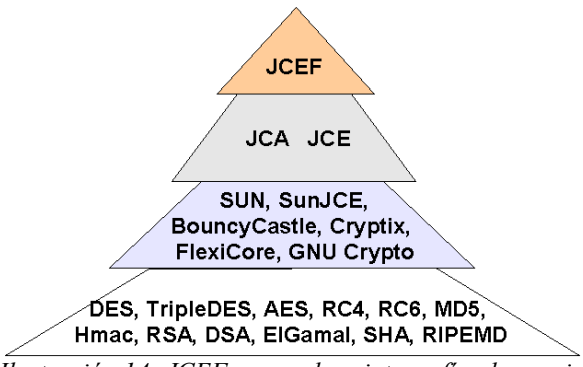


Ilustración 14: JCEF acerca la criptografía al usuario

A continuación se describirán las deficiencias encontradas y mejoradas en JCE (y JCA) y las mejoras realizadas por JCEF al mundo de la criptografía mediante Java. Y finalmente se muestran las mejoras que están pendientes de realizarse en futuras versiones de este proyecto. De esta forma, un usuario podrá decidir convenientemente cuál de las dos librerías debería usar.

JCEF mejora enormemente a JCE. Las deficiencias de JCE que han sido mejoradas son:

- JCE deja mucho que desear, puesto que la calidad de su código y diseño es baja.
- Dificulta su uso y aprendizaje al utilizar términos poco comprensibles.
- No proporciona al usuario un uso sencillo de sus funciones.
- Generar autentificadores es complicado al igual que su verificación, principalmente de huellas y sellos digitales al exigir que el usuario conozca cómo se verifican autentificadores como las huellas, sellos o firmas digitales.
- La protección y desprotección es muy complicada de utilizar.
- Los algoritmos hay que inicializarlos de una forma anticuada y engorrosa.
- Los parámetros se generan y gestionan de una forma nada sencilla.
- JCE no permite construir objetos seguros para cualquier algoritmo criptográfico.
- No reconstruye automáticamente las operaciones tras un cambio.
- Proporciona un modo de operación muy particular para cada algoritmo criptográfico.
- Finalmente, para terminar con la paciencia de cualquier usuario, la gestión de los proveedores es complicada y desgraciadamente necesaria.
- Los algoritmos JCE no se pueden configurar en tiempo de ejecución ni tampoco se puede probar su funcionamiento con facilidad.
- Para definir algoritmos se utilizan clases diferentes a las que se usan en la utilización de los algoritmos.

En definitiva, utilizar JCE requiere un gran esfuerzo en tiempo y dedicación al usuario del mismo. Sin embargo, JCEF proporciona los siguientes valores añadidos:

- Proporciona una serie de mejoras muy significativas.
- Vale la pena usarlo, es amigable y fácil de utilizar.
- Se aprende de forma intuitiva y no requiere demasiado tiempo de aprendizaje.
- No se cometen errores fácilmente y no requiere grandes conocimientos técnicos.
- Evita el uso de conceptos técnicos, y además, su diseño y código son de mayor calidad.
- Simplifica enormemente la autentificación, haciéndola sencilla y homogénea, tanto la generación como verificación de huellas, sellos y firmas digitales; por lo que la autentificación se aprende de forma más sencilla y rápida.
- Proporciona un uso sencillo de los algoritmos, además de hacer que éstos sean reutilizables.

- Suministra una jerarquía de objetos muy completa.
- Permite seleccionar los algoritmos de una forma muy sencilla.
- La inicialización de los algoritmos es infinitamente más sencilla.
- Proporciona adaptaciones de proveedores criptográficos ya existentes que cumplen con la especificación JCE o no. Incluso permite definir fácilmente nuevos algoritmos con especificación JCEF.
- Permite de forma sencilla obtener información completa de inicialización y de configuración.
- Proporciona gestión adeacuada de los parámetros de inicialización y configuración.
- Permite cambiar fácilmente de algoritmo y de implementación.
- Para reducir la dificultad, las operaciones de bajo nivel son automáticas, transparentes y sencillas, de esta forma es innecesario realizar un sinfín de operaciones como por ejemplo la encapsulación y desencapsulación de claves.
- Los algoritmos aceptan los parámetros en cualquiera de sus formas con el objetivo de mejorar la usabilidad.
- Aún hay más, JCEF potencia la creación de proveedores criptográficos personalizados.
- Se generan de forma automática todos los parámetros de los algoritmos.
- Para continuar simplificando, es posible evitar la realización de ciertas operaciones.
- Es innecesario conocer los algoritmos secundarios de los algoritmos criptográficos.
- Con JCEF, no hace falta tener conocimientos importantes.
- Proporciona mecanismos para la definición de algoritmos criptográficos compuestos.
- Permite construir objetos seguros de forma sencilla.
- La implementación y el algoritmo poseen la misma especificación.
- Proporciona un marco sencillo para pruebas de funcionalidad e implementación.
- Recopila una gran cantidad de algoritmos criptográficos y proveedores.

Por desgracia, JCEF no es perfecto y todavía puede y debe seguir creciendo. Algunas de las posibles mejoras que se le pueden hacer son:

- Proporcionar todas aquellas características que JCE proporciona y JCEF no, tales como los flujos de entrada/salida seguros, gestión de certificados digitales, almacenes seguros, protocolos de intercambio de claves, etc...
- Implementar flujos de entrada/salida seguros de forma sencilla para cualquier algoritmo criptográfico.
- Proporcionar algunos aspectos avanzados, tales como almacenes seguros para cualquier tipo de objetos incluyendo claves y certificados.
- Permitir gestionar certificados de una manera muy sencilla.

- Proporcionar sencillos mecanismos para los protocolos de intercambio de claves.
- Mejorar la gestión de los modos de operación y los esquemas de relleno en los algoritmos criptográficos de protección ya existentes.
- Verificar el correcto funcionamiento de todos los algoritmos criptográficos a través del uso de vectores de tests y corregir aquellos incorrectos.
- Comprobar el correcto funcionamiento de las pruebas de implementación.
- Terminar de implementar los algoritmos compuestos, probarlos y añadir a todos los proveedores JCEF existentes nuevos algoritmos compuestos reutilizando sus algoritmos criptográficos.
- Comprobar el correcto funcionamiento de algunos detalles de las implementaciones JCEF.
- Reducir la dependencia con otros proyectos.
- Aumentar el nivel de confianza de este proyecto.
- Implementar algoritmos nuevos no implementados hasta ahora en Java.

Como un ejemplo del valor añadido de este proyecto, observe el código siguiente donde se muestra cómo se asegura un objeto e inmediatamente después se recupera el mismo; y todo ello de una forma super sencilla:

```
Object object = new String("my object");
CryptoAlgorithm secureAlgorithm = new AES_BlockSymmetricProtectionRREXKY();
SecureObject secureObject = new SecureObject(object, secureAlgorithm);
object = (String)secureObject.getObject(secureAlgorithm);
```

Tabla 35: Pequeño ejemplo de uno de los valores añadidos de JCEF

En las próximas secciones podrán encontrarse más detalles sobre las deficiencias, mejoras realizadas y pendientes y otros detalles del proyecto como parte de su valor añadido, comparativa con lo que había antes (JCA-JCE), detalles de implementación, presentación, distribuciones, etc...

Si lo desea, también puede consultar la página web oficial del proyecto http://jcef.sourceforge.net y los recursos utilizados http://jcef.sourceforge.net/doc/resources.pdf.

1 Deficiencias mejoradas

JCE deja mucho que desear puesto que:

- La calidad de su código es baja.
- Dificulta su uso y aprendizaje.
- Se utilizan términos poco comprensibles.
- Su uso es muy complejo.
- La generación y verificación de autentificadores es complicada.

La calidad del código es baja, ya que:

- Complica el mantenimiento del código.
- Dificulta su reutilización.
- Complica el código volviéndolo incomprensible.
- Complica la accesibilidad del código volviéndolo cerrado.
- Se desconoce si su funcionamiento es totalmente correcto.
- No proporciona ningún conjunto de pruebas que se puedan utilizar.
- Carece de una documentación útil y completa.

Dificulta su uso y aprendizaje. Esto se debe principalmente a que proporciona muchísimos métodos no fundamentales, lo que hace que el usuario se pierda y acabe por no utilizar esta herramienta al haber demasiadas funciones donde elegir. Por dar algunos detalles, JCE proporciona múltiples formas de indicar los datos, ya sea para proteger o autentificarlos. Es decir, suministra gestión de arrays de bytes, por lo que rompe el patrón experto, o sea, realiza funciones que no le corresponde. Además, esto dificulta el aprendizaje ya que hay más funciones que aprender.

Se utilizan términos poco comprensibles para el gran público, tales como resumen de mensaje, código de autentificación de mensaje, encriptar, desencriptar, etc... Es decir, no evita el uso de conceptos técnicos al utilizar nombres complejos e incomprensibles para la mayoría del público.

Su uso es muy complejo. Esto se debe principalmente a que tanto la autentificación como la protección de datos son difíciles de utilizar porque requiere tener muchísimos aspectos en cuenta. JCE no realiza de forma automática muchas operaciones de bajo nivel, es decir, le deja mucha responsabilidad al usuario. Y todas sus operaciones son extremadamente difíciles de utilizar, aprender y mantener.

La generación de autentificadores es complicada, tanto la generación de huellas, sellos y firmas digitales, ya que cada uno de ellos se generan de forma muy distinta, proporcionando heterogeneidad y complejidad. JCE proporciona formas inapropiadas para generar cualquier tipo de autentificadores. Además, tampoco proporciona ninguna base común para los autentificadores ni los algoritmos que los gestionan. Con JCE se pueden utilizar más de 15 formas distintas para generar simplemente un autentificador.

La verificación de autentificadores es complicada, debido a que complica la verificación tanto de huellas, sellos y firmas digitales. Esto se debe principalmente al uso de multitud de métodos inapropiados para la verificación de los autentificadores ya que existen muchísimas formas de verificar un autentificador con JCE, cuando en realidad sólo una es la realmente útil. Además, tampoco proporciona ninguna base común para la verificación de autentificadores, es decir, las huellas, sellos y firmas digitales no son tratados de forma homogénea.

La verificación de huellas y sellos digitales es complicada. Esto sucede principalmente porque no se proporciona ninguna base común para los algoritmos de autentificación y porque es necesario conocer que para verificar la autenticidad de una huella digital hay que generar una nueva huella digital y compararla con la original; si son iguales entonces es auténtica, si no, no lo es. Sin embargo, las firmas digitales son relativamente sencillas de verificar, lo que añade heterogeneidad y complejidad a la verificación de autentificadores.

El usuario necesita saber cómo verificar autentificadores. Se requiere conocer que para comprobar la autenticidad de unos datos mediante huella o sello digital, es preciso volver a generar un nuevo autentificador de los mismos datos y comporarlo con el autentificador original. Sin embargo, el usuario no necesita saber cómo verificar firmas digitales. Esto es un aspecto positivo, pero también lo es negativo al poner de manifiesto que se rompe la homogeneidad debido a los dos puntos anteriores.

La protección y desprotección es compleja por muchos motivos entre los que destacan los siguientes: el modo de operación se indica de forma dificultosa, la inicialización de estos algoritmos no es nada sencilla y se proporcionan múltiples formas para obtener el mismo resultado.

La calidad de su diseño es baja, ya que no es homogéneo, ni actual y es complicado de entender y reutilizar.

Los algoritmos se inicializan de forma compleja debido a que los métodos de inicialización son heterogéneos e inapropiados; y sobre todo al suministrar múltiples formas complejas de realizar la inicialización. Además, para algoritmos muy similares se utilizan mecanismos de inicialización distintos. De esta forma se dificulta el aprendizaje y se complica su utilización. Por otro lado, además es necesario indicar los parámetros de inicialización en un orden concreto y por si esto no fuera poco, es tal la complejidad, que en el caso de los algoritmos de sellos digitales, se hace creer que estos algoritmos aceptar claves asimétricas cuando en realidad sólo aceptan claves simétricas.

Genera parámetros de forma muy compleja y oculta. Algunos algoritmos generan automáticamente algunos parámetros si éstos no son proporcionandos, pero desgraciadamente estas circunstancias no están documentadas y son muy difíciles de detectar. Esto pone de manifiesto lo complejo que puede resultar utilizar JCE.

No permite construir objetos seguros para cualquier algoritmo criptográfico, sólo acepta algoritmos de firmas digitales y protección. Es decir, no permite construir objetos seguros autentificables mediante huella o sello digital.

JCE no reconstruye automáticamente las operaciones tras un cambio, es necesario reconstruir la operación para realizar un cambio.

JCE proporciona un modo de operación particular para cada algoritmo criptográfico. Por ejemplo, para los algoritmos de protección es necesario indicar el modo de operación (protección o desprotección) mediante un entero a modo de tipo enumerado.

La gestión de los proveedores es complicada y necesaria y además no potencia la creación de proveedores criptográficos personalizados. Por otro lado, uno de los principales defectos de esta característica es que no es posible manejar todos los algoritmos criptográficos de la misma forma. Y además, algoritmos idénticos de proveedores distintos poseen nombres distintos.

Los algoritmos JCE no se pueden configuran en tiempo de ejecución. Además, **tampoco se puede probar su funcionamiento con facilidad** y por si no fuera poco, **para definir algoritmos se utilizan clases diferentes a las que se usan en la utilización de los algoritmos**.

2 Mejoras realizadas

JCEF proporciona una serie de mejoras muy significativas tales como:

- Muchos proveedores totalmente reutilizables.
- Además vale la pena su uso para usuarios que sepan muy poco o nada de criptografía y también para aquellos, que como yo, se hayan sentido decepcionados por una herramienta tan compleja de utilizar como JCE.

Vale la pena usar JCEF, debido a que es amigable, fácil de utilizar, se aprende de forma intuitiva, no se cometen errores fácilmente, no requiere grandes conocimientos técnicos y no requiere demasiado tiempo de aprendizaje y se ajusta a las necesidades de la mayoría de los usuarios.

Es amigable debido a que es fácil de utilizar en las situaciones habituales. Aunque también es cierto que puede ser complejo en algunas situaciones excepcionales.

Fácil de utilizar porque se aprende de forma intuitiva, no se cometen errores fácilmente, no requiere grandes conocimientos técnicos, evita el uso de conceptos técnicos y no requiere demasiado tiempo de aprendizaje.

Se aprende de forma intuitiva debido a la calidad del código, su diseño y documentación.

No requiere demasiado tiempo de aprendizaje ya que un aprendiz necesita como mucho una hora para su aprendizaje completo. Sin embargo, aprender por completo a utilizar JCE requiere entre una y cuatro semanas de dedicación. Esta reducción del tiempo de aprendizaje se debe primordialmente a que no se cometen errores fácilmente, al ahorro de operaciones, a la simplificación de muchas operaciones, a que no requiere grandes conocimientos técnicos y se ajusta más a las necesidades del usuario.

No se cometen errores fácilmente debido a la calidad de su código, su diseño, su documentación, al ahorro de operaciones, ...

No requiere grandes conocimientos técnicos. JCEF no requiere que el usuario tenga grandes conocimientos técnicos sobre criptografía, algoritmos criptográficos y su utilización mediante Java. Es decir, exige menor cantidad de conocimiento al usuario. Esto se consigue gracias a que no se usan conceptos técnicos, se utilizan nombres sencillos y comprensibles, se ahorran operaciones, se simplifica el uso de muchas operaciones y se ajusta mayormente a las necesidades del usuario.

Evita el uso de conceptos técnicos. JCEF no utiliza conceptos técnicos como hace JCE. Usa conceptos y nombres muy sencillos y comprensibles para el gran público tales como: autentificadores, protección, desprotección, huella digital, sello digital, firma digital, traductores de claves y parámetros, generadores de autentificadores, verificadores de autentificadores, generador de claves asimétricas, generador de claves simétricas, etc...

Su código es de calidad ya que fácilmente es mantenible, reutilizable y comprensible. Además, también es "Open Source" (Código Abierto) y está totalmente documentado y probado.

Su diseño es de mejor calidad debido a que es homogéneo, actual, simple, está enfocado a las necesidades del usuario, está basado en los principios de diseño contrastados como el de los JavaBeans y fomenta la reutilización.

Simplifica la autentificación, simplificando el uso de los algoritmos de autentificación al proporcionar una base común para todos ellos. Esto provoca además que usar este tipo de algoritmos sea muy sencillo.

La autentificación es sencilla y homogénea. Esto es debido a que la generación y verificación de autentificadores es sencilla, ya que tanto las huellas digitales, como los sellos y las firmas digitales se generan y verifican de forma sencilla; es decir, no es necesario distinguir entre huellas, sellos o firmas digitales.

La generación de huellas, sellos y firmas digitales es sencilla y homogénea porque todos estos autentificadores se generan igual, de manera homogénea y de una única forma.

La verificación de autentificadores es sencilla es sencilla y homogénea, principalmente debido a que tanto las huellas, sellos y firmas digitales se verifican de la misma forma, que además es homogénea y única, tal y como sucede con la generación.

La autentificación se aprende de forma más sencilla y rápida debido a su simplificación, sencillez y homogeneidad. En resumen, si se aprende a generar un autentificador, automáticamente se aprende a generar el resto de tipos de autentificadores. En general, sólo basta con aprender a generar y verificar autentificadores.

Proporciona un uso sencillo de los algoritmos además de hacer que éstos sean reutilizables, ya que proporciona bases comunes para todos los tipos de algoritmos, se utilizan principios de diseño adecuados y se proporciona un modo común homogéneo de uso. JCEF hace que sea sencillo realizar las siguientes funciones:

- Seleccionar, configurar, inicializar y utilizar algoritmos para asegurar y desasegurar datos.
- Autentificar, generar y verificar autentificadores como huellas, sellos o firmas digitales.
- Proteger y desproteger datos.
- Crear objetos seguros mediante algoritmos criptográficos con independencia de que se trate de algoritmos de protección, ya sean simétricos, asimétricos, de bloques, de flujo o basados en contraseña, o algoritmos de autentificación, se traten de huellas, sellos o firmas digitales.
- Obtener los objetos asegurados a partir de su versión segura.

JCEF proporciona una jerarquía de objetos muy completa. JCEF proporciona las siguientes categorías de objetos: objetos JCEF, proveedores JCEF, algoritmos, generadores, generadores de claves, generadores de claves asimétricas, generadores de claves simétricas, generadores de parámetros, generadores de datos aleatorios, traductores, traductores de claves, traductores de claves asimétricas, traductores de claves simétricas, traductores de parámetros, algoritmos criptográficos, algoritmos criptográficos de autentificación, algoritmos criptográficos de autentificación mediante huellas, sellos y firmas digitales tanto para generación como verificación, algoritmos de sellos digitales mediante contraseña, algoritmos de protección basada en contraseña, algoritmos de protección de datos, algoritmos criptográficos de protección/desprotección asimétrica o simétrica, simétrica de bloques y simétrica de flujo.

JCEF permite seleccionar los algoritmos de una forma muy sencilla, ya que basta simplemente con utilizar el constructor del algoritmo.

La inicialización de los algoritmos es infinitamente más sencilla. JCEF emplea los principios de los JavaBeans para acceder a los objetos, simplemente accediendo a sus propiedades utilizando métodos sencillos de lectura y escritura de las propiedades.

Proporciona un nuevo proveedor completo de algoritmos criptográficos llamado RREXKY.

Proporciona adaptaciones de proveedores criptográficos ya existentes que no cumplen con la especificación JCE. Tales proveedores son Mindbright, Jacksum y LogiCrypto.

Proporciona adaptaciones de proveedores criptográficos ya existentes que sí cumplen con la especificación JCE. Tales proveedores son BouncyCastle, Cryptix, CryptixCrypto, IAIK, GNU Crypto, FlexiCore, FlexiEC, FlexiNF, JHBCI, SUN, SunJCE, SunJSSE y SunRsaSign.

Permite, de forma sencilla, obtener información completa de inicialización y de configuración de los algoritmos, simplemente accediendo a través de los métodos de lectura de dichas propiedades. O dicho de otro modo, proporciona una única forma de inicialización y configuración. También permite distinguir fácilmente entre los distintos tipos de algoritmos.

Permite definir fácilmente nuevos algoritmos con especificación JCEF, al proporcionar un soporte sencillo de ampliación para adaptar los algoritmos existentes desde la especificación JCE y cualquier otra que no sea JCE a la especificación JCEF, además de permitir la definición de algoritmos de nueva implementación.

Proporciona gestión adecuada de los parámetros de inicialización y configuración, ya que:

- Permite recuperar los parámetros de inicialización y configuración.
- Reconstruir la operación automáticamente tras un cambio.
- No complica la reutilización de las operaciones.
- Facilita la reutilización de los algoritmos.
- Permite indicar los parámetros en cualquier orden sin obligar a proporcionarlos en un orden concreto.
- Además, los algoritmos JCEF se pueden reconfigurar en tiempo de ejecución.

Permite cambiar fácilmente de algoritmo y de implementación, ya que los algoritmos se seleccionan de una forma natural, tienen una base común y no es necesario gestionar los proveedores criptográficos.

Las operaciones de bajo nivel son automáticas, transparentes y sencillas. Estas operaciones son: traducción de claves, conversión entre las distintas representaciones de los parámetros y claves y la generación de claves y parámetros.

Es innecesario realizar un sinfín de operaciones. El usuario se ahorra realizar un sinfín de operaciones de forma manual. Tales operaciones son:

- Las operaciones de bajo nivel.
- Las operaciones relacionadas con el uso de los algoritmos.
- Encapsular claves, ya que encapsular una clave es protegerla como si fuera un objeto.
- Desencapsular claves, ya que desencapsular una clave es desprotegerla como si fuera un objeto.
- Generar parámetros y claves tanto simétricas como asimétricas.
- Realizar conversiones entre representaciones de parámetros y claves tanto simétricas como asimétricas.
- Indicar el modo de inicialización para los algoritmos de protección.
- Realizar traducciones de parámetros y claves tanto simétricas como asimétricas.

Es innecesaria la encapsulación/desencapsulación de claves. Estas operaciones son innecesarias ya que una clave es como cualquier otro objeto, por lo que "inventarse" nuevas operaciones tan particulares, dificulta su uso. La operación de encapsular una clave consiste en proteger dicha clave. Por otro lado, la operación de desencapsular una clave se trata de desproteger dicha clave.

Los algoritmos aceptan los parámetros en cualquiera de sus formas. No distingue entre claves conocidas, claves desconocidas, especificadas de forma trasparente, opaca o persistente. Igualmente, no distingue entre parámetros especificados de forma transparente o persistente.

Se generan de forma automática todos los parámetros de los algoritmos, siempre y cuando sean necesarios y no hayan sido especificados.

Se potencia la creación de proveedores criptográficos personalizados. La gestión de proveedores es sencilla, ya que se podría decir que es inexistente a diferencia de JCE, es decir, simplemente cada proveedor contiene un conjunto de algoritmos que puede que hayan implementado los desarrolladores de dicho proveedor o no; por lo que se potencia la creación de proveedores criptográficos personalizados, fomentando a su vez la reutilización.

Es posible evitar la realización de ciertas operaciones, tales como proteger y desproteger datos, generar autentificadores (huellas, sellos y firmas digitales), verificar autentificadores (huellas, sellos y firmas digitales). Para ello se pueden utilizar los objetos seguros.

Es innecesario conocer los algoritmos secundarios de los algoritmos criptográficos, tales como los generadores de claves (simétricas y asimétricas), generadores de parámetros, conversores y traductores entre las distintas representaciones de los parámetros y conversores y traductores entre las distintas representaciones de las claves (simétricas y asimétricas).

Es innecesario tener conocimientos importantes. JCEF reduce el conocimiento necesario para el usuario, ya que es innecesario:

- Conocer los detalles técnicos sobre las distintas claves.
- Tener conocimientos importantes para utilizar un algoritmo, ya que sólo basta con conocer el algoritmo que se desea utilizar.
- Diferenciar una clave de un objeto.
- Conocer cómo se verifican los autentificadores.
- Conocer las distintas representaciones de los parámetros.
- Conocer las distintas representaciones de las claves tanto simétricas como asimétricas.
- Conocer las clases de cada una de las representaciones de los parámetros.
- Conocer las clases de cada una de las representaciones de las claves tanto simétricas como asimétricas.
- Conocer los parámetros de configuración de los algoritmos.
- Conocer los parámetros para los generadores de claves tanto simétricas como asimétricas.
- Conocer los parámetros para los generadores de parámetros.

- Conocer los parámetros para los traductores de claves tanto simétricas como asimétricas.
- Conocer los parámetros para los conversores entre las distintas representaciones de las claves tanto simétricas como asimétricas.
- Conocer los parámetros para los conversores entre las distintas representaciones de los parámetros
- Conocer los parámetros para los traductores de claves tanto simétricas como asimétricas.
- Conocer los parámetros para los algoritmos criptográficos de protección, ya sea protección asimétrica, protección simétrica, simétrica de bloques o de flujo, o basada en contraseña.
- Conocer los parámetros para los algoritmos criptográficos de autentificación, ya sean de huellas, sellos, sellos basados en contraseña o firmas digitales.

Proporciona funciones especiales, tales como los algoritmos criptográficos compuestos y el marco para la realización de pruebas con el objetivo de comprobar el correcto funcionamiento de los algoritmos.

Permite construir objetos seguros de forma sencilla, ya sean objetos protegidos, o autentificables mediante huella, sello o firma digital.

La implementación y el algoritmo poseen la misma especificación. A diferencia de JCE, JCEF emplea los algoritmos directamente con la posibilidad de no perder la abstracción de la implementación de los algoritmos. Y además, utilizar e implementar un algoritmo se realiza del mismo modo.

Proporciona un marco sencillo para pruebas, basado en la verificación de un conjunto de pruebas más sencillas con el objetivo de comprobar el correcto funcionamiento del algoritmo y de su implementación. Se utiliza la reflexión para simplificar el código de las pruebas. Además, realiza pruebas generales para todos los algoritmos.

JCEF recopila una gran cantidad de algoritmos criptográficos:

- Huellas digitales: MD2, MD4, MD5, SHA1, SHA224, SHA256, SHA384, SHA512, RIPEMD128, RIPEMD160, RIPEMD256, RIPEMD320, Tiger, Whirlpool, HashGOST.
- Sellos digitales: HmacMD2, HmacMD4, HmacMD5, HmacSHA1, HmacSHA224, HmacSHA256, HmacSHA384, HmacSHA512, HmacRIPEMD128, HmacRIPEMD160, HmacTiger, MacDES, MacTripleDES, MacRC2, MacRC5, MacIDEA, MacSkipjack.
- Sellos digitales basados en contraseña: PBEwithHmacSHA1, PBEwithHmacRIPEMD160.
- Firmas digitales (incluyen variantes): MD2withRSA, MD4withRSA, MD5withRSA, RIPEMD128withRSA, RIPEMD160withRSA, RIPEMD256withRSA, SHA1withRSA, SHA224withRSA, SHA256withRSA, SHA384withRSA, SHA512withRSA, SHA1withDSA, NONEwithDSA, SignatureGOST, NONEwithRSA, MD5andSHA1withRSA.
- Protección simétricas de bloques: DES, TripleDES, Blowfish, RC2, AES, Skipjack, CAST5, CAST6, Square, RC6, MARS, IDEA, Serpent, Twofish, GOST, RC5, RC5-64.

- Modos de operación: ECB, CBC, PCBC, CTR, CFB<n>, OFB<n>, OpenPGPCFB, OpenPGPCFBwithIv, GOFB.
- Esquemas de relleno: ISO10126Padding, X9.23Padding, withCTS, TBCPadding, ZeroBytePadding, PKCS1Padding, PKCS5Padding, PKCS7Padding, SSL3Padding, ISO78164Padding.
- Protección simétrica de flujo: RC4.
- Protección asimétrica (incluyen variantes): RSA, RSA con huellas digitales (MD2, MD4, MD5, SHA1, SHA224, SHA256, SHA384, SHA512, RIPEMD128, RIPEMD160, Tiger), ElGamal, ECIES.
- Protección basada en contraseña: PBE con algoritmos de huellas digitales (MD5,SHA1, SHA256) y protección simétrica (DES, TripleDES, RC2, IDEA, RC4, AES).

3 Mejoras pendientes

JCE proporciona unas características que de momento no posee JCEF, como son los protocolos de intercambio de claves, almacenes de claves y certificados, la gestión de certificados y flujos de entrada/salida seguros.

Sería importante implementar flujos de entrada/salida seguros de forma sencilla para cualquier algoritmo criptográfico, ya sea de protección (simétrica, asimétrica, de bloques, de flujo, o basada en contraseña) o autentificación (huellas digitales, sellos digitales, firmas digitales o sellos digitales basados en contraseña). Por lo tanto, una futura versión de JCEF, debería incluir, flujos de entrada/salida seguros mediante cualquier algoritmo criptográfico. Cabe destacar que JCE sólo proporciona y de manera compleja, flujos de entrada/salida para algoritmos criptográficos de protección y de autentificación sólo mediante huellas digitales.

En futuras versiones también deberá proporcionar algunos aspectos avanzados, tales como:

- Almacenes seguros para cualquier tipo de objetos, incluyendo claves y certificados.
- Sencilla gestión de certificados.
- Uso sencillo de protocolos de intercambio de claves.

Mejorar la gestión de los modos de operación y los esquemas de relleno en los algoritmos criptográficos de protección ya existentes también será importante en futuras versiones.

Verificar el correcto funcionamiento de todos los algoritmos criptográficos a través del uso de vectores de tests y corregir aquellos incorrectos. De momento se puede confiar en los algoritmos proporcionados por el proveedor JCEF "RREXKY". Decir que esta tarea debería realizarse empezando por los proveedores más importantes y terminando por los menos importantes. La lista de proveedores JCEF de mayor a menor importancia sería: "RREXKY", "GNU Crypto", "BouncyCastle", "FlexiCore", "FlexiEC", "FlexiNF", "Jacksum", "Cryptix", "CryptixCrypto", "SUN", "SunJCE", "SunJSSE", "SunRsaSign", "JHBCI", "IAIK", "Mindbright" y "LogiCrypto".

Terminar de comprobar el correcto funcionamiento de las pruebas de implementación y de algunos detalles de las implementaciones JCEF, como bien pudiera ser la verificación de que todos los algoritmos JCEF se pueden reutilizar con JCE.

Terminar de implementar los algoritmos compuestos, probarlos y añadir a todos los proveedores JCEF existentes nuevos algoritmos compuestos reutilizando sus algoritmos criptográficos ya existentes.

Importante mejorar la dependencia con otras librerías. Sería interesante independizar al máximo a JCEF de librerías de terceros.

Se ve indispensable aumentar el nivel de confianza de esta librería, ya que hay que tener en cuenta que un trabajo desarrollado por un alumno de una universidad no proporciona el mismo nivel de confianza que el trabajo desarrollado por una empresa conocida y establecida desde hace bastantes años. Para ello se podrá, por ejemplo, mejorar las pruebas sobre los algoritmos.

Implementar algoritmos nuevos no implementados hasta ahora en Java de huellas digitales, sellos digitales, firmas digitales, protección asimétrica, protección simétrica de flujo, protección simétrica de bloques, y aumentar la cantidad de modos de operación y esquemas de relleno, tales como:

- Huellas digitales: Snefru128 (Tamaño de la huella en bits = 128), Snefru256 (256), Tiger2 (192), HAS-160 (160), N-Hash (128), SaphireII128 (128), SaphireII160 (160), SaphireII192 (192), SaphireII224 (224), SaphireII256 (256), SaphireII288 (288), SaphireII320 (320), SquareHash, XOR16 (16), XOR32 (32), PMD128 (128), PMD160 (160), PMD192 (192), PMD224 (224), PMD256 (256), CRC16CCITT (16), CRC32 (32), Panama (256).
- Sellos digitales:OMAC y PMAC.
- Firmas digitales: GMR, KCDSA, Schnorr, Desmedt, Lamport-Diffie, Rabin, Matyas-Meyer.
- Protección asimétrica: Rabin, Merkle-Hellman, Paillier, McEliece, Cayley-Purser, Williams, Goldwasser-Micali.
- Protección simétrica de flujo: ORYX, SEAL, Chameleon, Fish, A5/1, A5/2, Helix, Pike, Panama (Tamaño de clave en bits = 256), Sober (128), Wake (128), SCOP, Snow (128), SN3, Turing, Leviathan, MUGI, Sapphire2 ([8, 8, 2040]), Scream (128), SEAL3 (128), Cloak (992) y Solitare.
- Protección simétrica de bloques: 3-Way (Tamaño de bloque en bits = 96, Tamaños de clave en bits = 96), SHACAL (160, [128, 512]), SHACAL2 (256, [128, 512]), Magenta (128, {128, 192, 256}), DEAL (128, {128, 192, 256}), FEAL (64, {64, 128}), KHAFRE (64, {64, 128}), ICE (64, [64, 64, ...)), Madryga, LOKI89 (64, 64), LOKI91 (128, {128, 192, 256}), LOKI97 (128, {128, 192, 256}), Akelarre (128, [64, 64, ...)), KASUMI (64, 128), MISTY1 (64, 128), MISTY2 (64, 128), TEA (64, 128), XTEA (64, 128), NewDES (64, 120), Red Pike (64, 64), CMEA (64, -), GDES (64, 64), GOST768 (64, 768), IBC (256, 160), S-1, Diamond2 (128, [8, 8, 65536]), Diamond2Lite (128, [8, 8, 65536]), REDOC2 (80, 160), REDOC3 (80, [0, 20480)), KHUFU (64, 512), DESX (64, {128, 192}), Lucifer (128, 128), MMB (128, 128), SEED (128, 128), Shark (64, 128), MacGuffin (64, 128), Bear, Lion, Lioness, LadderDES, HPC, Noekeon (128, 128), SPEED64 (64, [48, 16, 256]),

SPEED128 (128, [48, 16, 256]), SPEED256 (128, [48, 16, 256]), FROG ([64, 1024], [40, 1000]), Q128 (128, 128), SaferK40 (64, 40), SaferSK40 (64, 40), SaferK64 (64, 64), SaferSK64 (64, 64), SaferK128 (64, 128), SaferSK128 (64, 128), Safer+ (128, {128, 192, 256}), Safer++ (128, {128, 256}), Crypton (128), DFC (128, {128, 192, 256}), E2 (128, {128, 192, 256}), CS-Cipher (64, [0, 8, 128]), MDC (?, [64, 8, 640]), Rainbow (128, [128, 32, 256]), NSEA (128, 128), MPJ (128, 128), Cobra (128, [1, 1, 1152])].

- Modos de operación [PBC, PFB, CBCPD, OFBNLF, BCF, BOF]

4 Detalles

El núcleo principal del proyecto es una librería llamada "JCEF", la cual proporciona la interfaz necesaria para poder manejar algoritmos criptográficos y objetos seguros. El diagrama general de clases de dicha librería es el siguiente:

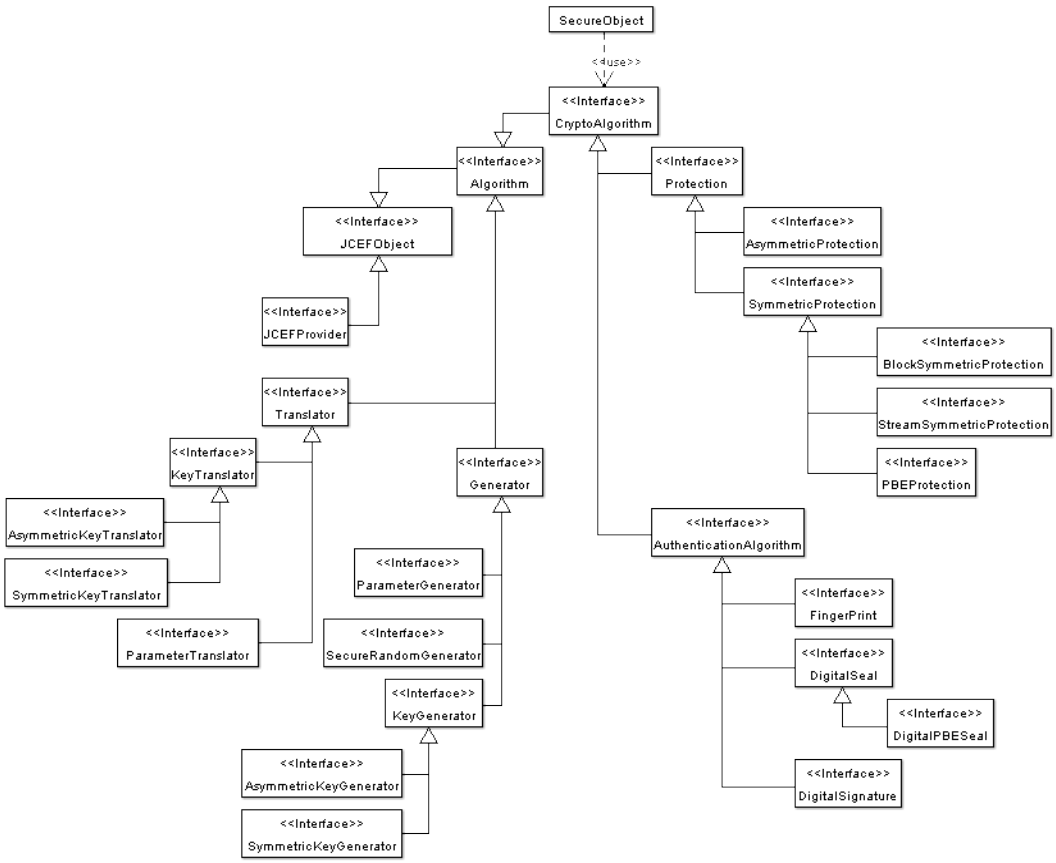


Ilustración 15: Diagrama general de clases de JCEF

La clase "SecureObject" es la clase que representa a los objetos seguros y éstos son generados utilizando algoritmos criptográficos que son representados mediante la interfaz "CryptoAlgorithm". Para obtener nuevamente los objetos asegurados es necesario volver a utilizar los algoritmos criptográficos.

Es tal la dimensión del proyecto, que de ahí el gran número de librerías debido principalmente a que se ha querido proporcionar el mayor número de proveedores y algoritmos criptográficos posible realizando adaptaciones de proveedores y algoritmos criptográficos ya existentes que cumplan con la especificación JCE u otra.

Para que el lector puede hacerse una idea de la dimensión de este proyecto, basta con decir que este proyecto contiene 3142 clases, 315 campos, 2914 métodos, 73959 líneas de código y 22941 líneas de comentarios. Dentro de estos datos ya se encuentran incluidos las métricas para las pruebas, cuyas dimensiones concretas son 1724 clases, 73 campos, 958 métodos, 30936 líneas de código y 8119 líneas de comentarios. Además, también se incluyen las métricas para el manual de usuario de JCEF, cuyos datos son 11 clases, 3 campos, 412 métodos, 4572 líneas de código y 5740 líneas de comentarios.

Otra métrica para imaginarse el coste de este proyecto es el tiempo de desarrollo y realización del mismo; el cual se estima entre un mínimo de 2000 horas de trabajo y un máximo de 3000 horas. Este proyecto ha tenido pocas pausas durante su desarrollo, el cual comenzó a finales de febrero de 2005 y terminó a finales de mayo de 2006, es decir, más de un año de desarrollo.

El manual de usuario de este proyecto, sin considerar este documento, se encuentra en una clase llamada "UserGuide". Se encuentra en una clase ya que no hay mejor manual para una librería de programación que una clase compuesta por numerosos ejemplos escritos en el mismo lenguaje de programación (Java) y explicaciones escritas con el lenguaje de documentación para código (Javadoc). Este manual de usuario podrá encontrarlo fácilmente en la dirección web http://jcef.sourceforge.net/api/org/rrexky/security/crypto/jcef/UserGuide.html. Para obtener más información es importante consultar la página web de este proyecto http://jcef.sourceforge.net y la página principal online de la documentación de JCEF que se encuentra en http://jcef.sourceforge.net/api/index.html.

A continuación se mostrarán los siguientes apartados:

- Visión general del diseño, donde se hablará del diseño de este proyecto.
- Componentes. Dicha sección enumerará los partes de las que se compone el proyecto.
- Dependencias, donde se mostrarán las dependencias del proyecto y de sus componentes.
- Distribuciones que son conjuntos de ficheros que contienen diferentes construcciones del proyecto. Unas destinadas para el uso general, otra para desarrolladores, etc...
- También se describirán las pruebas realizadas al proyecto, el contenido del CD-ROM que se adjunta con la memoria de este proyecto y una pequeña guía de instalación y uso del proyecto.

1 Visión general del diseño

El proyecto "Aplicaciones Criptográficas Java" ha sido descompuesto en numerosas librerías. Además, cada librería tiene asociada otra que contiene las pruebas para comprobar el correcto funcionamiento de dicha librería. Por si fuera poco, todas las librerías han sido documentadas.

Las librerías de este proyecto son "JCEF" como librería principal, "JCEF Addons" como librería para extensiones futuras y el resto de librerías son proveedores criptográficos que cumplen al menos la especificación JCEF y librerías de pruebas para comprobar el correcto funcionamiento de todas las librerías. Concretamente se prueba el correcto funcionamiento de todos los métodos de todas las clases que componen el proyecto y se realizan pruebas de funcionalidad a cada algoritmo.

El diseño de este proyecto sigue los principios básicos de los Java Beans. Es decir, todas las clases se componen de atributos a los cuales se accede mediante al menos un método de lectura (llamado "get + nombre del atributo") y al menos otro de escritura (llamado "set + nombre del atributo").

Otra consideración a tener en cuenta es que se ha intentado al máximo que el número de líneas de código por método fuera el menor posible y así de esta forma conseguir que:

- No sea necesario escribir grandes cantidades de documentación ya que los métodos casi se explican por sí solos mediante el código, necesitando tan solo la ayuda de unas pocas líneas de comentarios para la documentación del mismo.
- La realización de las pruebas y su ejecución fuera lo más sencillo posible.

2 Componentes

Este proyecto se compone de las siguientes librerías:

1. JCEF: Esta es la librería principal del proyecto, donde se encuentra la especificación JCEF, las interfaces para manipular algoritmos criptográficos y otros algoritmos de apoyo relacionados con la criptografía. Además también tiene la posibilidad de utilizar objetos seguros. Por si no fuera poco, también contiene varias implementaciones de la especificación o interfaz JCEF para implementar rápidamente nuevos algoritmos criptográficos, adaptar otros ya existentes y además ser compatible con la anterior especificación JCE. Su página web oficial y la de este proyecto es http://jcef.sourceforge.net.
2. JCEF Tests: Esta librería es la encargada de realizar pruebas a la librería "JCEF" para comprobar su correcto funcionamiento.
3. RREXKY JCEF Provider: Principal proveedor de algoritmos criptográficos que cumplen la especificación JCEF. Todos estos algoritmos han sido probados y funcionan correctamente. Se trata de un proveedor personalizado con algoritmos criptográficos extraidos de otros proveedores tales como "BouncyCastle JCEF Provider", "Jacksum JCEF

Provider" y "FlexiCore JCEF Provider". Este proveedor contiene 64 algoritmos de todos los tipos funcionando correctamente y son:

- *Algoritmos criptográficos de protección asimétrica*: RSA
- *Algoritmos criptográficos de protección simétrica de bloques*: AES, Blowfish, Camellia, CAST5, CAST6, GOST, RC2, RC6, Serpent, Skipjack, TripleDES, Twofish.
- *Algoritmos criptográficos de protección simétrica de flujo*: RC4
- *Algoritmos criptográficos de protección basada en contraseña*: PBEwithSHA1andTwofish, PBEwithSHA256andAES256
- *Algoritmos generadores de fuentes de datos aleatorios*: BBS
- *Algoritmos criptográficos de autentificación mediante huellas digitales*: GOST, Haval128, Haval160, Haval192, Haval224, Haval256, MD2, MD4, MD5, RIPEMD128, RIPEMD160, RIPEMD256, RIPEMD320, SHA1, SHA224, SHA256, SHA384, SHA512, Tiger, Whirlpool
- *Algoritmos criptográficos de autentificación mediante sellos digitales*: HmacMD2, HmacMD4, HmacMD5, HmacRIPEMD128, HmacRIPEMD160, HmacSHA1, HmacSHA224, HmacSHA256, HmacSHA384, HmacSHA512, HmacTiger, MacDES, MacRC2, MacSkipjack, MacTripleDES
- *Algoritmos criptográficos de autentificación mediante sellos basados en contraseña*: PBEHmacRIPEMD160, PBEHmacSHA1
- *Algoritmos criptográficos de autentificación mediante firmas digitales*: MD2withRSA, MD4withRSA, MD5withRSA, RIPEMD128withRSA, RIPEMD160withRSA, RIPEMD256withRSA, SHA1withRSA, SHA224withRSA, SHA256withRSA, SHA384withRSA, SHA512withRSA

4. RREXKY JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "RREXKY JCEF Provider" para comprobar su correcto funcionamiento.

5. JCEF Addons: Extensiones para JCEF. Actualmente incluye, aunque no se han probado correctamente, clases que permiten definir nuevos algoritmos criptográficos en base a otros algoritmos criptográficos ya existentes. También contiene las pruebas para poder realizarlas con el objetivo de comprobar que las implementaciones de los algoritmos criptográficos son correctas.

6. JCEF Addons Tests: Esta librería es la encargada de realizar pruebas a la librería "JCEF Addons" para comprobar su correcto funcionamiento.

7. BouncyCastle JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://www.bouncycastle.org. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

8. BouncyCastle JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "BouncyCastle JCEF Provider" para comprobar su correcto funcionamiento.

9. Jacksum JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente que no cumple ni la especificación JCE ni la JCEF cuya

web es http://www.jonelo.de/java/jacksum/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

10. Jacksum JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "Jacksum JCEF Provider" para comprobar su correcto funcionamiento.

11. FlexiCore JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://www.flexiprovider.de/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

12. FlexiCore JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "FlexiCore JCEF Provider" para comprobar su correcto funcionamiento.

13. FlexiEC JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://www.flexiprovider.de/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

14. FlexiEC JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "FlexiEC JCEF Provider" para comprobar su correcto funcionamiento.

15. FlexiNF JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://www.flexiprovider.de/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

16. FlexiNF JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "FlexiNF JCEF Provider" para comprobar su correcto funcionamiento.

17. CryptixCrypto JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://www.cryptix.org/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

18. CryptixCrypto JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "CryptixCrypto JCEF Provider" para comprobar su correcto funcionamiento.

19. Cryptix JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://www.cryptix.org/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

20. Cryptix JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "Cryptix JCEF Provider" para comprobar su correcto funcionamiento.

21. GNUCrypto JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://www.gnu.org/software/gnu-crypto/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

22. GNUCrypto JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "GNUCrypto JCEF Provider" para comprobar su correcto funcionamiento.

23. IAIK JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://jce.iaik.tugraz.at/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

24. IAIK JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "IAIK JCEF Provider" para comprobar su correcto funcionamiento.

25. JHBCI JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://www.jhbci.de/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

26. JHBCI JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "JHBCI JCEF Provider" para comprobar su correcto funcionamiento.

27. Logi Crypto JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero una especificación desconocida que no es JCE cuya web es http://www.logi.org/logi.crypto/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

28. Logi Crypto JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "Logi Crypto JCEF Provider" para comprobar su correcto funcionamiento.

29. Mindbright JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de una especificación desconocida que no es JCE cuya web es http://www.appgate.com/products/80_MindTerm/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

30. Mindbright JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "Mindbright JCEF Provider" para comprobar su correcto funcionamiento.

31. SUN JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://java.sun.com/j2se/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

32. SUN JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "SUN JCEF Provider" para comprobar su correcto funcionamiento.

33. SunJCE JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://java.sun.com/j2se/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

34. SunJCE JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "SunJCE JCEF Provider" para comprobar su correcto funcionamiento.

35. SunJSSE JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://java.sun.com/j2se/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

36. SunJSSE JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "SunJSSE JCEF Provider" para comprobar su correcto funcionamiento.

37. SunRsaSign JCEF Provider: Se trata de un proveedor JCEF que ha sido adaptado de otro proveedor criptográfico ya existente pero de especificación JCE cuya web es http://java.sun.com/j2se/. Este proveedor ha sido casi completamente desarrollado pero no del todo probado. Funcionan muchos algoritmos y el resto necesitan configurarse mejor.

38. SunRsaSign JCEF Provider Tests: Esta librería es la encargada de realizar pruebas a la librería "SunRsaSign JCEF Provider" para comprobar su correcto funcionamiento.

3 Dependencias

En esta sección se indican las librerías de las cuales este proyecto depende. En la siguiente lista se indicarán las dependencias de cada una de las librerías de este proyecto:

- JCEF: Depende de "API de J2SE" y "Apache Jakarta Commons Lang".
- JCEF Tests: Depende de "API de J2SE", "JUnit", "JCEF" y "JCEF Addons".
- RREXKY JCEF Provider: .Depende de "API de J2SE", "BouncyCastle JCEF Provider", "Jacksum JCEF Provider" y "FlexiCore JCEF Provider".
- RREXKY JCEF Provider Tests: Depende de "API de J2SE", "JUnit", "JCEF Tests" y "RREXKY JCEF Provider".
- JCEF Addons: Depende de "API de J2SE", "JCEF" y "JCEF Tests".
- JCEF Addons Tests: Depende de "API de J2SE", "JCEF Addons" y "JCEF Tests".
- BouncyCastle JCEF Provider: Depende de "API de J2SE", "JCEF" y "BouncyCastle JCE Provider".
- BouncyCastle JCEF Provider Tests: Depende de "API de J2SE", "JUnit", "JCEF Tests" y "BouncyCastle JCEF Provider".
- Jacksum JCEF Provider: Depende de "API de J2SE", "JCEF" y "Jacksum NonJCE Provider".
- Jacksum JCEF Provider Tests: Depende de "API de J2SE", "JUnit", "JCEF Tests" y "Jacksum JCEF Provider".
- FlexiCore JCEF Provider: Depende de "API de J2SE", "JCEF" y "FlexiCore JCE Provider".
- FlexiCore JCEF Provider Tests: Depende de "API de J2SE", "JUnit", "JCEF Tests" y "FlexiCore JCEF Provider".
- FlexiEC JCEF Provider: Depende de "API de J2SE", "JCEF" y "FlexiEC JCE Provider".
- FlexiEC JCEF Provider Tests: Depende de "API de J2SE", "JUnit", "JCEF Tests" y "FlexiEC JCEF Provider".

- FlexiNF JCEF Provider: Depende de “API de J2SE”, “JCEF” y “FlexiNF JCE Provider”.
- FlexiNF JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “FlexiNF JCEF Provider”.
- CryptixCrypto JCEF Provider: Depende de “API de J2SE”, “JCEF” y “CryptixCrypto JCE Provider”.
- CryptixCrypto JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “CryptixCrypto JCEF Provider”.
- Cryptix JCEF Provider: Depende de “API de J2SE”, “JCEF” y “Cryptix JCE Provider”.
- Cryptix JCEF Provider Tests: Depende de “J2SE”, “JUnit”, “JCEF Tests” y “Cryptix JCEF Provider”.
- GNUCrypto JCEF Provider: Depende de “API de J2SE”, “JCEF” y “GNUCrypto JCE Provider”.
- GNUCrypto JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “GNUCrypto JCEF Provider”.
- IAIK JCEF Provider: Depende de “API de J2SE”, “JCEF” y “IAIK JCE Provider”.
- IAIK JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “IAIK JCEF Provider”.
- JHBCI JCEF Provider: Depende de “API de J2SE”, “JCEF” y “JHBCI JCE Provider”.
- JHBCI JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “JHBCI JCEF Provider”.
- Logi Crypto JCEF Provider: Depende de “API de J2SE”, “JCEF” y “Logi Crypto NonJCE Provider”.
- Logi Crypto JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “Logi Crypto JCEF Provider”.
- Mindbright JCEF Provider: Depende de “API de J2SE”, “JCEF” y “Mindbright NonJCE Provider”.
- Mindbright JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “Mindbright JCEF Provider”.
- SUN JCEF Provider: Depende de “API de J2SE” y “JCEF”.
- SUN JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “SUN JCEF Provider”.
- SunJCE JCEF Provider: Depende de “API de J2SE” y “JCEF”.
- SunJCE JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “SunJCE JCEF Provider”.

- SunJSSE JCEF Provider: Depende de “API de J2SE” y “JCEF”.
- SunJSSE JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “SunJSSE JCEF Provider”.
- SunRsaSign JCEF Provider: Depende de “API de J2SE” y “JCEF”.
- SunRsaSign JCEF Provider Tests: Depende de “API de J2SE”, “JUnit”, “JCEF Tests” y “SunRsaSign JCEF Provider”.

4 Distribuciones

Aquí se describen las distribuciones que proporciona este proyecto desde su web http://jcef.sourceforge.net y http://sourceforge.net/projects/jcef:

- “jcef_v1.0_basicruntime.zip”: Es la distribución básica. Contiene las librerías para el uso de JCEF y del proveedor RREXKY tan solo. Se trata de una distribución sólo para usuarios que necesitan lo básico. Ocupa unos 2'54 MB y se distribuye en formato ZIP. Esta distribución viene acompañada de las librerías de las que depende excepto de la “API de J2SE” que es la estándar y no necesita incluirse.
- “jcef_v1.0_development.zip”: Es la distribución destinada a los desarrolladores que quieran mejorar este proyecto. Contiene el código fuente, las pruebas y la documentación tanto de “JCEF”, “JCEF Addons” y todos los proveedores criptográficos “JCEF”. Ocupa unos 24'3 MB y se distribuye en formato ZIP. Esta distribución viene acompañada de las librerías de las que depende excepto de la “API de J2SE” que es la estándar y no necesita incluirse.
- “jcef_v1.0_runtime.zip”: Es la distribución completa para usuarios, no desarrolladores. Contiene únicamente todo el código compilado de “JCEF”, “JCEF Addons”, las pruebas y los proveedores. Ocupa unos 6'64 MB y se distribuye en formato ZIP. Esta distribución viene acompañada de las librerías de las que depende excepto de la “API de J2SE” que es la estándar y no necesita incluirse.
- “jcef_v1.0_all_src.zip”: Contiene todo el código fuente Java del proyecto, incluyendo obviamente los comentarios de documentación Javadoc. Ocupa unos 7'18 MB y se distribuye en formato ZIP. Esta distribución viene acompañada de las librerías de las que depende excepto de la “API de J2SE” que es la estándar y no necesita incluirse.
- “jcef_v1.0_doc.zip”: Contiene toda la documentación de la memoria y la documentación de todas las librerías del proyecto. Ocupa unos 22'4 MB y se distribuye en formato ZIP.
- “jcef_v1.0_web.zip”: Distribución compuesta por la página web del proyecto, pero sin incluir el resto de distribuciones. Ocupa unos 9'51 MB y se distribuye en formato ZIP.

5 Página web

Desde la página web del proyecto http://jcef.sourceforge.net (o también http://sourceforge.net/projects/jcef/) el visitante podrá acceder a los siguientes recursos:

- Documentación del proyecto, desde un sumario hasta la documentación completa. Podrá acceder además a los capítulos por separado de la documentación completa. Estos capítulos son: "Introducción a la seguridad", "Criptografía orientada a objetos", "El proyecto", "Futuros proyectos", "Recursos utilizados" y "Preguntas frecuentes".
- Documentación de la API JCEF. Tan solo se ha incluido la documentación de la librería "JCEF" y se han dejado fuera el resto de librerías por cuestiones de espacio en el servidor de alojamiento de la página web del proyecto ya que éste sólo permite un máximo de 100 MB y la documentación completa de todas las librerías del proyecto ocupa más de 150 MB.
- Zona de descarga de las distribuciones.
- Presentación. Se podrá acceder a una presentación de diapositivas para ver lo más interesante del proyecto y aquello que está sujeto de ser visionado en la lectura de este proyecto ante el tribunal evaluador del mismo.
- Multitud de enlaces de herramientas utilizadas para el desarrollo del proyecto.
- Últimas noticias sobre el proyecto.

6 Pruebas realizadas

Las librerías de pruebas se encargan de someter al proyecto a una serie de tests con el objetivo de comprobar el correcto funcionamiento del mismo.

Se han realizado pruebas para comprobar el correcto funcionamiento de todos los métodos de todas las clases que componen el proyecto.

La mayoría de las pruebas se basan en comprobar que las propiedades de los objetos de una clase se leen y escriben correctamente.

Existen otras pruebas que verifican la funcionalidad de los algoritmos. Esta prueba genera párametros y claves, realiza traducciones de parámetros y claves y especialmente trata de generar objetos seguros para posteriormente obtener el objeto asegurado con varios tipos de parámetros y claves. Si el objeto asegurado es el mismo que el original o si es auténtico, entonces el algoritmo supera la prueba de funcionalidad. Un algoritmo no supera esta prueba de funcionalidad en caso contrario o cuando se produce alguna excepción provocada principalmente por una mala configuración del algoritmo.

Hay otras pruebas llamadas de implementación. Su finalidad es la de verificar la correcta implementación de los algoritmos, no se trata de que funcionen, si no de que funcionen como tienen que hacerlo. Pero hay que decir que estas pruebas de implementación están implementadas pero no se ha realizado ninguna ya que esto se ha considerado parte de un proyecto futuro.

Otra curiosidad es que son tantas pruebas que se tarda bastante en poder comprobar si éstas son superadas. También es conveniente saber que si se lanzan muchas pruebas a la vez, se corre el riesgo de quedarse sin memoria y no poder ver si son superadas o no. Lo mejor que se puede hacer es lanzar las pruebas sobre los paquetes hoja (aquellos que no contienen otros paquetes).

Recuerde también que las únicas librerías que han superado completamente las pruebas, son "JCEF" y "RREXKY JCEF Provider". El resto de librerías requieren una revisión "clase por clase", "algoritmo por algoritmo". Esto último no se ha realizado por falta de tiempo y se ha dejado la responsabilidad de ello a un proyecto futuro.

7 Contenido del CD-ROM

El CD-ROM que acompaña a la memoria de este proyecto contendrá las siguientes carpetas:

- "Source Code": Todo el código fuente del proyecto.
- "Libs": Todas las librerías de las que depende este proyecto exceptuando la librería estándar J2SE.
- "Distributions": Almacenará las distribuciones del proyecto.
- "Papers": La memoria del proyecto en formato OpenOffice y PDF junto con las imágenes utilizadas. Además se incluye la documentación API sobre todas las librerías.
- "Web": Página web del proyecto, a la que se podrá acceder de forma offline u online si se desea.
- "Software": Contiene el software mínimo necesario para poder consultar el código fuente del proyecto y poder lanzar las pruebas para comprobar su correcto funcionamiento:
  - "JRE.exe": Archivo de instalación de Java 2 Runtime Standard Edition (J2SE).
  - "jce policy files": Carpeta que contiene los archivos que permiten desbloquear JRE/J2SE.
  - "Eclipse": Carpeta que contiene el IDE para Java llamado "Eclipse". Para instalarlo, basta con copiar esta carpeta al disco duro.
  - "7-Zip.exe": Archivo de instalación del programa "7-Zip", el cual se utiliza para comprimir/descomprimir archivos.
  - "AlZip.exe": Archivo de instalación del programa "AlZip" utilizado para comprimir/descomprimir archivos.
  - "Adobe Reader ES.exe": Archivo de instalación del programa "Adobe Reader" en español para Windows XP. Permite leer archivos PDF.

- "OpenOffice 2.exe": Archivo de instalación del programa "OpenOffice" que permitirá ver y sobre todo editar la memoria del proyecto.
- "OpenOffice 2 ES pack.exe": Archivo de instalación para traducir OpenOffice a español.

8 Guía de instalación y uso

Si lo que desea es ver qué se ha hecho exactamente, examinar el código fuente y comprobar que las pruebas son superadas, sólo es necesario que siga los siguientes pasos:

1. Instalar la máquina virtual de Java: "J2SE Runtime Environment" (JRE) en un directorio cualquiera que llamaremos simbólicamente <java-home>.
2. Desproteger JRE copiando los ficheros de políticas de seguridad llamados "US_export_policy.jar" y "local_policy.jar" al directorio "<java-home>/lib/security".
3. Copiar todo el código fuente del proyecto en una carpeta llamada <source-home>.
4. Copiar todas las librerías de las que depende este proyecto en una carpeta llamada <libs-home>.
5. Instalar el IDE "Eclipse", abrirlo, definirle un directorio cualquiera para el espacio de trabajo y cerrar la pestaña "Welcome" si le apareciera.
6. Crear un nuevo proyecto Java llamado "Aplicaciones Criptográficas Java" a partir de la carpeta <source-home> y a continuación pulsar en siguiente.
7. Añadir el JRE desprotegido como librería que está ubicado en <java-home>.
8. Añadir las librerías ubicadas en <libs-home> bajo el nombre de una única librería que podría llamarse "Dependences". Es muy importante asegurarse de que la librería JRE se encuentra encima de la librería "Dependences".
9. Esperar un tiempo a que se construya el espacio de trabajo.
10. Y ya está en disposición de ver el código fuente y verificar que las pruebas se superan al menos para las librerías "JCEF" y "RREXKY JCEF Provider".
11. Para ver el código, simplemente haga doble click en el nodo raíz del proyecto, en nodos librería, nodos carpeta y nodos archivo Java.
12. Para lanzar las pruebas sobre "JCEF", debe dirigirse a la librería "JCEF Tests" e ir ejecutando cada una de las pruebas, desde pruebas sencillas (ficheros cuyo nombre termina en "Test") hasta múltiples pruebas (ficheros cuyos nombres comienzan con "AllTests"). Para lanzar pruebas basta con pulsar el botón derecho sobre ella y hacer click en la opción "Run As --> JUnit Test". Recuerde que es importante que no ejecute muchas pruebas a la vez puesto que puede que se quede sin memoria y por ello no terminen correctamente las pruebas. Lo mejor es lanzar las pruebas de los paquetes hoja por cada librería. También puede comprobar el correcto funcionamiento de la librería "RREXKY JCEF Provider".

5 El uso habitual

En esta sección se muestra el uso más habitual de la criptografía, es decir, se muestra cómo se crean objetos seguros y se obtienen sus objetos asegurados a partir de sus versiones seguras utilizando parámetros nuevos y reutilizando unos ya existentes generados con anteriodad.

En los siguientes puntos se mostrará el mencionado uso habitual mediante la utilización de este proyecto. Se verá que es muy fácil construir objetos seguros con la librería JCEF. En resumen, este uso habitual se compone de los siguientes pasos:

1. Asegurar un objeto con nuevos parámetros criptográficos.
2. Almacenar parámetros criptográficos para un uso posterior.
3. Obtener el objeto asegurado utilizando los nuevos parámetros criptográficos.
4. Asegurar otro objeto reutilizando parámetros criptográficos ya existentes.
5. Obtener el objeto asegurado reutilizando parámetros criptográficos ya existentes.

1 Asegurar un objeto con nuevos parámetros criptográficos

```
/**
 * Mediante JCEF, crea un objeto seguro y obtiene su versión insegura
nuevamente. Posteriormente crea y deshace otro objeto seguro reutilizando
parámetros generados con anterioridad
 */
public static void example1_JCEF () {
 try {
  // 1. Se desea asegurar un objeto
  // 1.1. Definimos un objeto que se desea asegurar
  String object = "my object";
  // 1.2. Se selecciona un algoritmo de seguridad
          SymmetricProtection         secureAlgorithm         =         new
AES_BlockSymmetricProtectionRREXKY();
   // 1.3. Se inicializa el algoritmo con parámetros concretos si fuera
necesario (esta inicialización es opcional)
   // Esta inicialización es inexistente ya que se desea utilizar unos
parámetros nuevos (clave y parámetro generado automáticamente por el
algoritmo)
  // 1.4. Se asegura el objeto y se guarda en algún lugar
  SecureObject secureObject = new SecureObject(object, secureAlgorithm);
  saveObject(secureObject);
  // ...
```

Tabla 36: JCEF – 1. Asegurar un objeto con nuevos parámetros criptográficos

2 Almacenar parámetros criptográficos para un uso posterior

```
  // 2. Se guardan los parámetros para su posterior uso, ya sea para generar nuevos objetos seguros u obtener objetos asegurados (objetos inseguros) que es lo más habitual
  byte[] encodedKey = secureAlgorithm.getSymmetricKey().getEncoded();
  byte[] encodedParameter = secureAlgorithm.getEncodedParameter();
  saveKey(encodedKey); saveParameter(encodedParameter);
  // ...
```

Tabla 37: JCEF – 2. Almacenar parámetros criptográficos para un uso posterior

3 Obtener el objeto asegurado utilizando los nuevos parámetros criptográficos

```
  // ... tiempo más tarde en algún otro lugar del código ...
  // 3. Se desea recuperar el objeto de forma segura
  // 3.1. Cargamos los datos que se necesitan
  encodedKey = loadKey(); encodedParameter = loadParameter();
  // 3.2. Inicializamos el algoritmo con los parámetros apropiados para poder obtener el objeto asegurado
  secureAlgorithm = new AES_BlockSymmetricProtectionRREXKY();
  secureAlgorithm.setSymmetricKey(encodedKey);
  secureAlgorithm.setParameter(encodedParameter);
  // 3.3. Se obtiene el objeto asegurado a partir de su versión en forma de objeto seguro
  secureObject = (SecureObject)loadObject();
  object = (String)secureObject.getObject(secureAlgorithm);
  // ...
```

Tabla 38: JCEF – 3. Obtener el objeto asegurado utilizando los nuevos parámetros criptográficos

4 Asegurar otro objeto reutilizando parámetros criptográficos ya existentes

```
  // ... tiempo más tarde en algún otro lugar del código ...
  // 4. Se desea asegurar un nuevo objeto con la misma clave y parámetro en
un instante de tiempo posterior
  // 4.1. Se define el nuevo objeto
  String otherObject = "other object";
  // 4.2. Se cargan los parámetros que se necesitan
  encodedKey = loadKey(); encodedParameter = loadParameter();
  // 4.3. Se inicializa el algoritmo
  secureAlgorithm = new AES_BlockSymmetricProtectionRREXKY();
  secureAlgorithm.setSymmetricKey(encodedKey);
  secureAlgorithm.setParameter(encodedParameter);
  // 4.4. Se asegura el nuevo objeto
    SecureObject otherSecureObject = new SecureObject(otherObject,
secureAlgorithm);
  saveObject(otherSecureObject);
  // ...
```

Tabla 39: JCEF – 4. Asegurar otro objeto reutilizando parámetros criptográficos ya existentes

5 Obtener el objeto asegurado reutilizando parámetros criptográficos ya existentes

```
  // ... tiempo más tarde en algún otro lugar del código ...
  // 5. Se desea recuperar el nuevo objeto asegurado
  // 5.1. Cargamos los datos que se necesitan
  encodedKey = loadKey(); encodedParameter = loadParameter();
  // 5.2. Inicializamos el algoritmo con los parámetros apropiados para
poder obtener el objeto asegurado
  secureAlgorithm = new AES_BlockSymmetricProtectionRREXKY();
  secureAlgorithm.setSymmetricKey(encodedKey);
  secureAlgorithm.setParameter(encodedParameter);
  // 5.3. Se obtiene el objeto asegurado a partir de su versión en forma de
objeto seguro
  otherSecureObject = (SecureObject)loadObject();
  object = (String)otherSecureObject.getObject(secureAlgorithm);
 }
 catch (Throwable throwable) { throwable.printStackTrace(); return; }
}
```

Tabla 40: JCEF – 5. Obtener el objeto asegurado reutilizando parámetros criptográficos ya existentes

6 Comparativa con JCA y JCE

A continuación se podrá ver el uso más habitual de la criptografía, es decir, se mostrará cómo se crean unos objetos seguros y se obtienen sus objetos asegurados a partir de sus versiones seguras utilizando parámetros nuevos y reutilizando unos ya existentes generados con anteriodad. Pero en esta ocasión, no se utilizará el proyecto; se empleará la arquitectura criptográfica de Java (JCA) y su extensión (JCE). De esta forma, podrá verse una parte importante del valor añadido de este proyecto al mundo de la criptografía y Java.

1 Asegurar un objeto con nuevos parámetros criptográficos

- *Definición del objeto a asegurar y carga del proveedor*

```
/**
 * Mediante JCE, crea un objeto seguro y obtiene su versión insegura nuevamente. Posteriormente crea y deshace otro objeto seguro reutilizando parámetros generados con anterioridad
 */
public static void example1_JCE () {
 // 1. Se desea asegurar un objeto
 // 1.1. Definimos un objeto que se desea asegurar
 String object = "my object";
 // 1.2. Se selecciona un algoritmo de seguridad
 // 1.2.1. Es muy posible que sea necesario cargar el proveedor de los algoritmos que se desean utilizar. Esto hay que hacerlo una sola vez
 Provider provider = new BouncyCastleProvider();
 Security.addProvider(provider);
 String providerName = provider.getName();
 // ...
```

Tabla 41: JCE – 1.1. Definición del objeto a asegurar y carga del proveedor

- *Definición del generador de claves simétricas*

```
 // ...
// 1.2.2. Se deben generar los parámetros del algoritmo antes de seleccionar el mismo
// 1.2.2.1. Se genera la fuente de datos aleatorios si fuera necesario (no es el caso)
SecureRandom random = null;
// 1.2.2.2. Se genera la clave
Key key = null;
// 1.2.2.2.1. Se selecciona el algoritmo generador de claves simétricas para el algoritmo deseado
javax.crypto.KeyGenerator keyGenerator = null;
try { keyGenerator = javax.crypto.KeyGenerator.getInstance("AES", providerName); }
catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
catch (NoSuchProviderException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 42: JCE – 1.2. Definición del generador de claves simétricas

- *Inicialización del generador de claves simétricas y generación de la clave*

```
 // ...
 // 1.2.2.2.2. Se inicializa el algoritmo de generación de claves simétricas para el algoritmo deseado
int keySize = 256;
AlgorithmParameterSpec genParameter = null;
if (genParameter != null && random == null) {
 try { keyGenerator.init(genParameter); }
catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
} else if (genParameter != null && random != null) {
 try { keyGenerator.init(genParameter, random); }
catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
} else if (genParameter == null && keySize > 0 && random == null) {
 keyGenerator.init(keySize);
} else if (genParameter == null && keySize > 0 && random != null) {
 keyGenerator.init(keySize, random);
} else if (genParameter == null && keySize <= 0 && random != null) {
 keyGenerator.init(random);
}
// 1.2.2.2.3. Se utiliza el generador ya seleccionado e inicializado para generar la clave simétrica
key = keyGenerator.generateKey();
 // ...
```

Tabla 43: JCE – 1.3. Inicialización del generador de claves simétricas y generación de la clave

- *Definición del generador de parámetros*

```
 // ...
 // 1.2.2.3. Se necesita generar además un parámetro para el algoritmo además de la clave generada con anterioridad
 AlgorithmParameterSpec parameter = null;
 // 1.2.2.3.1. Se indica el tipo de parámetro del que se trata
 Class parameterType = IvParameterSpec.class;
 // 1.2.2.3.2. Se selecciona el algoritmo generador de parámetros para el algoritmo deseado
 AlgorithmParameterGenerator parameterGenerator = null;
 try { parameterGenerator = AlgorithmParameterGenerator.getInstance("AES", providerName); }
 catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
 catch (NoSuchProviderException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 44: JCE – 1.4. Definición del generador de parámetros

- *Inicialización del generador de parámetros*

```
 // ...
 // 1.2.2.3.3. Se inicializa el algoritmo de generación de parámetros
 genParameter = null; int parameterSize = 16;
 if (genParameter != null && random == null) {
  try { parameterGenerator.init(genParameter); }
 catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
 } else if (genParameter != null && random != null) {
  try { parameterGenerator.init(genParameter, random); }
 catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
 } else if (genParameter == null && random == null) {
  parameterGenerator.init(parameterSize);
 } else if (genParameter == null && random != null) {
  parameterGenerator.init(parameterSize, random);
 }
 // ...
```

Tabla 45: JCE – 1.5. Inicialización del generador de parámetros

- *Generación del parámetro*

```
 // ...
// 1.2.2.3.4. Se utiliza el generador para obtener el parámetro necesario para el algoritmo
AlgorithmParameters algorithmParameters = parameterGenerator.generateParameters();
try { parameter = algorithmParameters.getParameterSpec(parameterType); }
catch (InvalidParameterSpecException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 46: JCE – 1.6. Generación del parámetro

- *Definición del algoritmo de seguridad*

```
 // ...
// 1.2.3. Se selecciona el algoritmo de seguridad
Cipher secureAlgorithm = null;
try { secureAlgorithm = Cipher.getInstance("AES/CBC/PKCS7Padding", providerName); }
catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
catch (NoSuchProviderException e) { e.printStackTrace(); return; }
catch (NoSuchPaddingException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 47: JCE – 1.7. Definición del algoritmo de seguridad

- *Inicialización del algoritmo de seguridad*

```
 // ...
 // 1.3. Se inicializa el algoritmo con parámetros concretos si fuera necesario (esta inicialización NO es opcional)
 // 1.3.1. Se inicializa el algoritmo
 int mode = Cipher.ENCRYPT_MODE;
 if (parameter != null && random == null) {
  try { secureAlgorithm.init(mode, key, parameter); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
 catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
 } else if (parameter != null && random != null) {
  try { secureAlgorithm.init(mode, key, parameter, random); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
 catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
 } else if (parameter == null && random == null) {
  try { secureAlgorithm.init(mode, key); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
 } else if (parameter == null && random != null) {
  try { secureAlgorithm.init(mode, key, random); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
 }
  // ...
```

Tabla 48: JCE – 1.8. Inicialización del algoritmo de seguridad

- *Obtención del parámetro que se haya podido generar automáticamente*

```
 // ...
 // 1.3.2. Se capturan los parámetros si se han generado automáticamente. Esto es necesario si no se especifican los algoritmos, e incluso es necesario aunque se especifique puesto que el algoritmo puede ignorar el parámetro indicado por el usuario y generar otro de forma automática
 try { algorithmParameters = secureAlgorithm.getParameters(); }
 catch (Throwable throwable) {}
 if (parameter == null && algorithmParameters != null) {
  if (parameterType == null) {
   parameterType = AlgorithmParameterSpec.class;
  }
  try { parameter = algorithmParameters.getParameterSpec(parameterType); }
  catch (InvalidParameterSpecException e) { e.printStackTrace(); return; }
 }
  // ...
```

Tabla 49: JCE – 1.9. Obtención del parámetro que se haya podido generar automáticamente

- *Creación del objeto seguro*

```
 // ...
// 1.4. Se asegura el objeto y se guarda en algún lugar
SealedObject secureObject = null;
try { secureObject = new SealedObject(object, secureAlgorithm); }
catch (IllegalBlockSizeException e) { e.printStackTrace(); return; }
catch (IOException e) { e.printStackTrace(); return; }
saveObject(secureObject);
 // ...
```

Tabla 50: JCE – 1.10. Creación del objeto seguro

2 Almacenar parámetros criptográficos para un uso posterior

- *Traducción de la clave*

```
 // ...
 // 2. Se guardan los parámetros para su posterior uso, ya sea para generar nuevos objetos seguros u obtener objetos asegurados (objetos inseguros) que es lo más habitual
 // 2.1. Se traducen los parámetros a su versión persistente
 // 2.1.1. Se traduce la clave a su versión persistente. En este caso es sencillo. Pero en otras ocasiones es distinto y en otras es necesario conocer el algoritmo de traducción dificultando el proceso mucho más al no ser homogéneo ya que también se utiliza para traducciones un algoritmo de tipo SecretKeyFactory tal y como se ve en los comentarios siguientes.
 // 2.1.1.1. Esta es la forma básica suministrada. Sin embargo, para los parámetros se proporcionan mecanismos más avanzados al permitir formatos de codificación pero son más complicados
 encodedKey = key.getEncoded();
 // ...
```

Tabla 51: JCE – 2.1. Traducción de la clave

- *Traducción del parámetro*

```
 // ...
 // 2.1.2. Se traduce el parámetro a su versión persistente. En este caso es sencillo pero no
es más que la traducción manual de un caso particular. En general se necesita manejar un
algoritmo del tipo AlgorithmParameters dificultando mucho más el proceso tal y como se
muestra a continuación. En total se muestra la forma manual pero particular y la forma
"automática" general. Observando en esta última que se utiliza un nombre distinto para el
algoritmo de traducción que para el algoritmo de protección/desprotección aunque sean nombres
"sinónimos"/alias o equivalentes
 // 2.1.2.1. Forma manual pero particular de traducir este parámetro
 if (parameter instanceof IvParameterSpec) {
  IvParameterSpec iv = (IvParameterSpec)parameter;
  encodedParameter = iv.getIV();
 }
  // ...
```

Tabla 52: JCE – 2.2. Traducción del parámetro

- *Otra traducción del parámetro*

```
 // ...
// 2.1.2.2. Forma "automática" pero general de traducir cualquier parámetro
AlgorithmParameters parameterTranslator = null;
try { parameterTranslator = AlgorithmParameters.getInstance("RIJNDAEL", providerName); }
catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
catch (NoSuchProviderException e) { e.printStackTrace(); return; }
try { parameterTranslator.init(parameter); }
catch (InvalidParameterSpecException e) { e.printStackTrace(); return; }
try { encodedParameter = parameterTranslator.getEncoded(); }
catch (IOException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 53: JCE – 2.3. Otra traducción del parámetro

- *Almacenamiento de los parámetros*

```
 // ...
// 2.2. Se guardan los parámetros
saveKey(encodedKey);
saveParameter(encodedParameter);
 // ...
```

Tabla 54: JCE – 2.4. Almacenamiento de los parámetros

3 Obtener el objeto asegurado utilizando los nuevos parámetros criptográficos

- *Carga de los parámetros*

```
 // ...
// ... tiempo más tarde en algún otro lugar del código ...
// 3. Se desea recuperar el objeto de forma segura
// 3.1. Cargamos los datos que se necesitan
encodedKey = loadKey();
encodedParameter = loadParameter();
 // ...
```

Tabla 55: JCE – 3.1. Carga de los parámetros

- *Definición del algoritmo de seguridad*

```
 // ...
 // 3.2. Inicializamos el algoritmo con los parámetros apropiados para poder obtener el
objeto asegurado
 // 3.2.1. Se selecciona el algoritmo para desproteger el objeto seguro
 try { secureAlgorithm = Cipher.getInstance("AES/CBC/PKCS7Padding", providerName); }
 catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
 catch (NoSuchProviderException e) { e.printStackTrace(); return; }
 catch (NoSuchPaddingException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 56: JCE – 3.2. Definición del algoritmo de seguridad

- *Traducción de los parámetros a la forma adecuada*

```
 // ...
// 3.2.2. Se inicializa el algoritmo para desprotección
// 3.2.2.1. Es necesario convertir los parámetros en su forma adecuada
// 3.2.2.1.1. Se reconvierte la clave a la forma adecuada
key = new SecretKeySpec(encodedKey, "AES");
// 3.2.2.1.2. Se reconvierte el parámetro a la forma adecuada
parameterType = IvParameterSpec.class; parameterTranslator = null;
try { parameterTranslator = AlgorithmParameters.getInstance("RIJNDAEL", providerName); }
catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
catch (NoSuchProviderException e) { e.printStackTrace(); return; }
try { parameterTranslator.init(encodedParameter); }
catch (IOException e) { e.printStackTrace(); return; }
try { parameter = parameterTranslator.getParameterSpec(parameterType); }
catch (InvalidParameterSpecException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 57: JCE – 3.3. Traducción de los parámetros a la forma adecuada

- *Inicialización del algoritmo de seguridad para desprotección*

```
 // ...
// 3.2.2.2. Se inicializa el algoritmo para desprotección
mode = Cipher.DECRYPT_MODE;
if (parameter != null && random == null) {
 try { secureAlgorithm.init(mode, key, parameter); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
 catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
} else if (parameter != null && random != null) {
 try { secureAlgorithm.init(mode, key, parameter, random); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
 catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
} else if (parameter == null && random == null) {
 try { secureAlgorithm.init(mode, key); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
} else if (parameter == null && random != null) {
 try { secureAlgorithm.init(mode, key, random); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
}
 // ...
```

Tabla 58: JCE – 3.4. Inicialización del algoritmo de seguridad para desprotección

- *Obtención del objeto asegurado*

```
 // ...
// 3.3. Se obtiene el objeto asegurado a partir de su versión en forma de objeto seguro
try {
 secureObject = (SealedObject)loadObject();
 object = (String)secureObject.getObject(secureAlgorithm); }
catch (IllegalBlockSizeException e) { e.printStackTrace(); return; }
catch (BadPaddingException e) { e.printStackTrace(); return; }
catch (IOException e) { e.printStackTrace(); return; }
catch (ClassNotFoundException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 59: JCE – 3.5. Obtención del objeto asegurado

4 Asegurar otro objeto reutilizando parámetros criptográficos ya existentes

- *Definición del objeto y carga de los parámetros*

```
 // ...
// ... tiempo más tarde en algún otro lugar del código ...
 // 4. Se desea asegurar un nuevo objeto con la misma clave y parámetro en un instante de tiempo posterior
 // 4.1. Se define el nuevo objeto
 String otherObject = "other object";
 // 4.2. Se cargan los parámetros que se necesitan
 encodedKey = loadKey();
 encodedParameter = loadParameter();
 // ...
```

Tabla 60: JCE – 4.1. Definición del objeto y carga de los parámetros

- *Traducción de la clave a su forma adecuada*

```
 // ...
// 4.3. Se inicializa el algoritmo
// 4.3.1. Se reconvierten los parámetros a la forma adecuada
// 4.3.1.1. Se reconvierte la clave a la forma adecuada
key = new SecretKeySpec(encodedKey, "AES");
 // ...
```

Tabla 61: JCE – 4.2. Traducción de la clave a su forma adecuada

- *Traducción del parámetro a su forma adecuada*

```
 // ...
// 4.3.1.2. Se reconvierte el parámetro a la forma adecuada
parameterType = IvParameterSpec.class; parameterTranslator = null;
try { parameterTranslator = AlgorithmParameters.getInstance("RIJNDAEL", providerName); }
catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
catch (NoSuchProviderException e) { e.printStackTrace(); return; }
try { parameterTranslator.init(encodedParameter); }
catch (IOException e) { e.printStackTrace(); return; }
try { parameter = parameterTranslator.getParameterSpec(parameterType); }
catch (InvalidParameterSpecException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 62: JCE – 4.3. Traducción del parámetro a su forma adecuada

- *Definición del algoritmo de seguridad*

```
 // ...
// 4.3.2. Se selecciona el algoritmo de seguridad
secureAlgorithm = null;
try { secureAlgorithm = Cipher.getInstance("AES/CBC/PKCS7Padding", providerName); }
catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
catch (NoSuchProviderException e) { e.printStackTrace(); return; }
catch (NoSuchPaddingException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 63: JCE – 4.4. Definición del algoritmo de seguridad

- *Inicialización del algoritmo de seguridad para protección*

```
 // ...
 // 4.3.3. Se inicializa el algoritmo con parámetros concretos (esta inicialización NO es
opcional)
 // 4.3.3.1. Se inicializa el algoritmo
 mode = Cipher.ENCRYPT_MODE;
 if (parameter != null && random == null) {
  try { secureAlgorithm.init(mode, key, parameter); }
  catch (InvalidKeyException e) { e.printStackTrace(); return; }
  catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
 } else if (parameter != null && random != null) {
  try { secureAlgorithm.init(mode, key, parameter, random); }
  catch (InvalidKeyException e) { e.printStackTrace(); return; }
  catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
 } else if (parameter == null && random == null) {
  try { secureAlgorithm.init(mode, key); }
  catch (InvalidKeyException e) { e.printStackTrace(); return; }
 } else if (parameter == null && random != null) {
  try { secureAlgorithm.init(mode, key, random); }
  catch (InvalidKeyException e) { e.printStackTrace(); return; }
 }
  // ...
```

Tabla 64: JCE – 4.5. Inicialización del algoritmo de seguridad para protección

- *Obtención del parámetro si fuera generado automáticamente*

```
  // ...
 // 4.3.3.2. Se capturan los parámetros si se han generado automáticamente. Esto es necesario
si no se especifican los algoritmos, e incluso es necesario aunque se especifique puesto que
el algoritmo puede ignorar el parámetro indicado por el usuario y generar otro de forma
automática
 try { algorithmParameters = secureAlgorithm.getParameters(); }
 catch (Throwable throwable) {}
 if (parameter == null && algorithmParameters != null) {
  if (parameterType == null) { parameterType = AlgorithmParameterSpec.class; }
  try { parameter = algorithmParameters.getParameterSpec(parameterType); }
  catch (InvalidParameterSpecException e) { e.printStackTrace(); return; }
 }
  // ...
```

Tabla 65: JCE – 4.6. Obtención del parámetro si fuera generado automáticamente

- *Creación del objeto seguro*

```
 // ...
// 4.4. Se asegura el nuevo objeto y se guarda en algún lugar
SealedObject otherSecureObject = null;
try { otherSecureObject = new SealedObject(otherObject, secureAlgorithm); }
catch (IllegalBlockSizeException e) { e.printStackTrace(); return; }
catch (IOException e) { e.printStackTrace(); return; }
saveObject(otherSecureObject);
 // ...
```

Tabla 66: JCE – 4.7. Creación del objeto seguro

5 Obtener el objeto asegurado reutilizando parámetros criptográficos ya existentes

- *Carga de los parámetros*

```
 // ...
// ... tiempo más tarde en algún otro lugar del código ...
// 5. Se desea recuperar el nuevo objeto asegurado
// 5.1. Cargamos los datos que se necesitan
encodedKey = loadKey();
encodedParameter = loadParameter();
 // ...
```

Tabla 67: JCE – 5.1. Carga de los parámetros

- *Definición del algoritmo de seguridad*

```
 // ...
 // 5.2. Inicializamos el algoritmo con los parámetros apropiados para poder obtener el
objeto asegurado
 // 5.2.1. Se selecciona el algoritmo para desproteger el objeto seguro
 try { secureAlgorithm = Cipher.getInstance("AES/CBC/PKCS7Padding", providerName); }
 catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
 catch (NoSuchProviderException e) { e.printStackTrace(); return; }
 catch (NoSuchPaddingException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 68: JCE – 5.2. Definición del algoritmo de seguridad

- *Traducción de la clave a su forma adecuada*

```
 // ...
// 5.2.2. Se inicializa el algoritmo para desprotección
// 5.2.2.1. Es necesario convertir los parámetros en su forma adecuada
// 5.2.2.1.1. Se reconvierte la clave a la forma adecuada
key = new SecretKeySpec(encodedKey, "AES");
 // ...
```

Tabla 69: JCE – 5.3. Traducción de la clave a su forma adecuada

- *Traducción del parámetro a su forma adecuada*

```
 // ...
// 5.2.2.1.2. Se reconvierte el parámetro a la forma adecuada
parameterType = IvParameterSpec.class; parameterTranslator = null;
try { parameterTranslator = AlgorithmParameters.getInstance("RIJNDAEL", providerName); }
catch (NoSuchAlgorithmException e) { e.printStackTrace(); return; }
catch (NoSuchProviderException e) { e.printStackTrace(); return; }
try { parameterTranslator.init(encodedParameter); }
catch (IOException e) { e.printStackTrace(); return; }
try { parameter = parameterTranslator.getParameterSpec(parameterType); }
catch (InvalidParameterSpecException e) { e.printStackTrace(); return; }
 // ...
```

Tabla 70: JCE – 5.4. Traducción del parámetro a su forma adecuada

- *Inicialización del algoritmo de seguridad para desprotección*

```
 // ...
// 5.2.2.2. Se inicializa el algoritmo para desprotección
mode = Cipher.DECRYPT_MODE;
if (parameter != null && random == null) {
 try { secureAlgorithm.init(mode, key, parameter); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
 catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
} else if (parameter != null && random != null) {
 try { secureAlgorithm.init(mode, key, parameter, random); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
 catch (InvalidAlgorithmParameterException e) { e.printStackTrace(); return; }
} else if (parameter == null && random == null) {
 try { secureAlgorithm.init(mode, key); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
} else if (parameter == null && random != null) {
 try { secureAlgorithm.init(mode, key, random); }
 catch (InvalidKeyException e) { e.printStackTrace(); return; }
}
 // ...
```

Tabla 71: JCE – 5.5. Inicialización del algoritmo de seguridad para desprotección

- *Obtención del objeto asegurado*

```
 // ...
// 5.3. Se obtiene el objeto asegurado a partir de su versión en forma de objeto seguro
try {
 otherSecureObject = (SealedObject)loadObject();
 object = (String)otherSecureObject.getObject(secureAlgorithm); }
catch (IllegalBlockSizeException e) { e.printStackTrace(); return; }
catch (BadPaddingException e) {e.printStackTrace(); return; }
catch (IOException e) { e.printStackTrace(); return; }
catch (ClassNotFoundException e) { e.printStackTrace(); return; }
}
 // ...
```

Tabla 72: JCE – 5.6. Obtención del objeto asegurado

Tras observar el proceso anterior, se ve claramente que JCEF simplifica enormemente el uso de la criptografía ya que se obtiene el mismo resultado pero con menos líneas de código, menos conocimiento y además de una forma más fácil y sencilla. Por consiguiente, JCE complica el uso de la criptografía al obligar al usuario a escribir más líneas de código y tener más conocimiento para poder usar JCE.

7 Conclusiones

En resumidas cuentas, las conclusiones más importantes sobre el proyecto en sí son los principales aspectos valorables positivamente y también los puntos negativos.

Comencemos por algunos puntos que podrían considerarse como negativos. Éstos son los siguientes:

1. El tiempo empleado en el proyecto podría considerarse excesivo.
2. No haber desarrollado exactamente lo que estaba previsto desde un principio.

Los principales aspectos valorables positivamente son:

1. Haber desarrollado un trabajo novedoso en lugar de lo que se tenía previsto.
2. El uso 100% de herramientas Open Source o Freeware.
3. Haber aplicado el conocimiento de varias áreas: Ingeniería del software, Gráficos, Programación, Ofimática, Programación web, etc...
4. El enfoque orientado a objetos de la criptografía.
5. Publicación del proyecto en una página web (http://jcef.sourceforge.net).
6. El diseño de imágenes propias originales.
7. Haber sido capaz de trabajar en el proyecto sin prisas y sin pausas durante algo más de un año.
8. Las propuestas realizadas sobre posibles futuros proyectos fin de carrera.
9. La sección de preguntas frecuentes como ayuda adicional.

5.Futuros Proyectos

Este capítulo detalla los posibles proyectos de fin de carrera fuertemente relacionados con este proyecto. La descripción de estos futuros proyectos se ha realizado mediante el formato exigido a una propuesta de proyecto de fin de carrera por la Universidad de Las Palmas de Gran Canaria, con el objetivo de facilitar la elección de estos proyectos por alumnos y tutores y dar así continuidad a este proyecto.

Los nuevos proyectos propuestos son:

1. Ampliaciones de JCEF: Consiste en realizar mejoras significativas al proyecto "Aplicaciones Criptográficas Java".
2. Pruebas sobre algoritmos JCEF: Garantizar que el mayor número de algoritmos criptográficos de todos los proveedores JCEF funcionan correctamente. Para ello es necesario realizar pruebas de implementación mediante vectores de datos y corregir las configuraciones que sean necesarias.
3. Certificados Digitales con JCEF: Este proyecto consiste en añadir funcionalidades sobre el campo de la gestión de certificados digitales al proyecto "Aplicaciones Criptográficas Java".
4. Archivos seguros con JCEF: Programa que construye ficheros seguros utilizando el proyecto "Aplicaciones Criptográficas Java".
5. Proveedor Criptográfico JCEF: Se trata de diseñar un proveedor de algoritmos criptográficos no implementados hasta la fecha por ningún proveedor conocido en Java utilizando para ello el proyecto "Aplicaciones Criptográficas Java".
6. Almacén de objetos seguros con JCEF: Consiste en construir una librería que permita almacenar objetos de forma segura en almacenes persistentes y posteriormente recuperarlos utilizando para ello el proyecto "Aplicaciones Criptográficas Java".
7. Metaimplementación de "Aplicaciones Criptográficas Java": Se trata de volver a implementar el código Java del proyecto "Aplicaciones Criptográficas Java" pero utilizando herramientas de plantillas de código.

Para mayor información puede consultar la página web del proyecto http://jcef.sourceforge.net.

1 Ampliaciones de JCEF

<table>
<tr><td>Título</td><td>Ampliaciones de JCEF</td></tr>
<tr><td>Tutor</td><td>—</td></tr>
<tr><td>Descripción General</td><td>En la actualidad, la seguridad es importantísima. Sin servicios de seguridad, el desarrollo de ciertos sistemas se vería frenado; entre los que destacan: el comercio electrónico y las comunicaciones seguras a través de la red. Por lo que la criptografía se vuelve indispensable</td></tr>
<tr><td>Objetivos</td><td>Realizar las mejoras siguientes al proyecto “Aplicaciones Criptográficas Java”: flujos de entrada/salida, protocolos de intercambio de claves, gestión de modos de operación y esquemas de relleno, algoritmos compuestos y adición de algoritmos de este tipo a todos los proveedores JCEF existentes</td></tr>
<tr><td>Metodología</td><td>En un principio se utilizará un enfoque clásico de desarrollo pero que podrá adaptarse a la naturaleza del proyecto y de las personas que trabajan en el mismo</td></tr>
<tr><td>Recursos necesarios</td><td>PC, Internet, Linux, Java, Proyecto “Aplicaciones Criptográficas Java” (http://jcef.sourceforge.net), Eclipse IDE, OpenOffice y CamStudio</td></tr>
<tr><td>Perfil del alumno</td><td>Soltura con Java, UML y tener conocimientos de Criptografía y Diseño de software</td></tr>
<tr><td rowspan="3">Etapas</td><td>Etapa 1 Preliminares: Analizar el proyecto “Aplicaciones Criptográficas Java”</td></tr>
<tr><td>Etapa 2 Desarrollo: Mejorar el proyecto “Aplicaciones Criptográficas Java” añadiéndole las mejoras indicadas en la sección “Objetivos”</td></tr>
<tr><td>Etapa 3 Presentación: Elaborar y ensayar la presentación con el objetivo de obtener el máximo interés del público, resaltando por encima de todo los resultados y conclusiones más importantes del proyecto. Publicar el proyecto en https//sourceforge.net</td></tr>
<tr><td>Planificación temporal estimada</td><td>
<table>
<tr><td>Etapas</td><td>Duración estimada</td></tr>
<tr><td>Etapa 1 Preliminares</td><td>80 horas</td></tr>
<tr><td>Etapa 2 Desarrollo</td><td>160 horas</td></tr>
<tr><td>Etapa 3 Presentación</td><td>60 horas</td></tr>
</table>
<table>
<tr><td>Duración total</td><td>300 horas</td></tr>
</table>
</td></tr>
</table>

Tabla 73: Futuro Proyecto: Ampliaciones de JCEF

2 Pruebas sobre algoritmos JCEF

<table>
<tr><td>Título</td><td>Pruebas sobre algoritmos JCEF</td></tr>
<tr><td>Tutor</td><td>——</td></tr>
<tr><td>Descripción General</td><td>En la actualidad, la seguridad es importantísima. Sin servicios de seguridad, el desarrollo de ciertos sistemas se vería frenado; entre los que destacan: el comercio electrónico y las comunicaciones seguras a través de la red. Por lo que la criptografía se vuelve indispensable</td></tr>
<tr><td>Objetivos</td><td>Garantizar que el mayor número de algoritmos criptográficos de todos los proveedores JCEF funcionan correctamente. Para ello es necesario realizar pruebas de implementación mediante vectores de datos y corregir las configuraciones que sean necesarias.</td></tr>
<tr><td>Metodología</td><td>En un principio se utilizará un enfoque clásico de desarrollo pero que podrá adaptarse a la naturaleza del proyecto y de las personas que trabajan en el mismo</td></tr>
<tr><td>Recursos necesarios</td><td>PC, Internet, Linux, Java, Proyecto “Aplicaciones Criptográficas Java” (http://jcef.sourceforge.net), Eclipse IDE, OpenOffice y CamStudio</td></tr>
<tr><td>Perfil del alumno</td><td>Soltura con Java, UML y tener conocimientos de Criptografía y Diseño de software</td></tr>
<tr><td rowspan="3">Etapas</td><td>Etapa 1 Preliminares: Analizar el proyecto “Aplicaciones Criptográficas Java” y recopilar el mayor número de vectores de datos para las pruebas de implementación</td></tr>
<tr><td>Etapa 2 Desarrollo: Realizar el conjunto de pruebas y correcciones a los algoritmos</td></tr>
<tr><td>Etapa 3 Presentación: Elaborar y ensayar la presentación con el objetivo de obtener el máximo interés del público, resaltando por encima de todo los resultados y conclusiones más importantes del proyecto. Publicar el proyecto en https//sourceforge.net</td></tr>
<tr><td>Planificación temporal estimada</td><td>
<table>
<tr><td>Etapas</td><td>Duración estimada</td></tr>
<tr><td>Etapa 1 Preliminares</td><td>80 horas</td></tr>
<tr><td>Etapa 2 Desarrollo</td><td>160 horas</td></tr>
<tr><td>Etapa 3 Presentación</td><td>60 horas</td></tr>
</table>
<table>
<tr><td>Duración total</td><td>300 horas</td></tr>
</table>
</td></tr>
</table>

Tabla 74: Futuro Proyecto: Pruebas sobre algoritmos JCEF

3 Certificados Digitales con JCEF

<table>
<tr><td>Título</td><td>Certificados Digitales con JCEF</td></tr>
<tr><td>Tutor</td><td>——</td></tr>
<tr><td>Descripción General</td><td>En la actualidad, la seguridad es importantísima. Sin servicios de seguridad, el desarrollo de ciertos sistemas se vería frenado; entre los que destacan: el comercio electrónico y las comunicaciones seguras a través de la red. Por lo que la criptografía se vuelve indispensable</td></tr>
<tr><td>Objetivos</td><td>Realizar mejoras al proyecto “Aplicaciones Criptográficas Java”. Mejoras tales como la gestión de certificados digitales, generación automática de certificados, listas de anulación, autentificación mediante certificados, cadenas de certificación, ...</td></tr>
<tr><td>Metodología</td><td>En un principio se utilizará un enfoque clásico de desarrollo pero que podrá adaptarse a la naturaleza del proyecto y de las personas que trabajan en el mismo</td></tr>
<tr><td>Recursos necesarios</td><td>PC, Internet, Linux, Java, Proyecto “Aplicaciones Criptográficas Java” (http://jcef.sourceforge.net), Eclipse IDE, OpenOffice y CamStudio</td></tr>
<tr><td>Perfil del alumno</td><td>Soltura con Java, UML y tener conocimientos de Criptografía y Diseño de software</td></tr>
<tr><td rowspan="3">Etapas</td><td>Etapa 1 Preliminares: Analizar el proyecto “Aplicaciones Criptográficas Java”, la gestión de certificados y las herramientas disponibles para ello</td></tr>
<tr><td>Etapa 2 Desarrollo: Mejorar el proyecto “Aplicaciones Criptográficas Java” añadiéndole gestión de certificados digitales</td></tr>
<tr><td>Etapa 3 Presentación: Elaborar y ensayar la presentación con el objetivo de obtener el máximo interés del público, resaltando por encima de todo los resultados y conclusiones más importantes del proyecto. Publicar el proyecto en https//sourceforge.net</td></tr>
<tr><td>Planificación temporal estimada</td><td>
<table>
<tr><td>Etapas</td><td>Duración estimada</td></tr>
<tr><td>Etapa 1 Preliminares</td><td>80 horas</td></tr>
<tr><td>Etapa 2 Desarrollo</td><td>160 horas</td></tr>
<tr><td>Etapa 3 Presentación</td><td>60 horas</td></tr>
</table>
<table>
<tr><td>Duración total</td><td>300 horas</td></tr>
</table>
</td></tr>
</table>

Tabla 75: Futuro Proyecto: Certificados Digitales con JCEF

4 Archivos Seguros con JCEF

<table>
<tr><td>Título</td><td>Archivos Seguros con JCEF</td></tr>
<tr><td>Tutor</td><td>——</td></tr>
<tr><td>Descripción General</td><td>En la actualidad, la seguridad es importantísima. Sin servicios de seguridad, el desarrollo de ciertos sistemas se vería frenado; entre los que destacan: el comercio electrónico y las comunicaciones seguras a través de la red. Por lo que la criptografía se vuelve indispensable</td></tr>
<tr><td>Objetivos</td><td>Desarrollar un programa para generar archivos seguros utilizando algoritmos criptográficos y que además debe tener la posibilidad de añadir nuevos algoritmos en tiempo de ejecución. Debe ser una seria alternativa a “Encrypt Easy” (http://www.baltsoft.com), “EasyCrypto” (http://www.handybits.com/), “EncryptOnClick” (http://www.2brightsparks.com) y “AxCrypt” (http://axcrypt.sourceforge.net)</td></tr>
<tr><td>Metodología</td><td>En un principio se utilizará un enfoque clásico de desarrollo pero que podrá adaptarse a la naturaleza del proyecto y de las personas que trabajan en el mismo</td></tr>
<tr><td>Recursos necesarios</td><td>PC, Internet, Linux, Java, Proyecto “Aplicaciones Criptográficas Java” (http://jcef.sourceforge.net), Eclipse IDE, OpenOffice, CamStudio y un diseñador de GUIs como Visual Editor (http://ww.eclipse.org/vep/).</td></tr>
<tr><td>Perfil del alumno</td><td>Soltura con Java, UML y tener conocimientos de Criptografía y Diseño de software</td></tr>
<tr><td rowspan="3">Etapas</td><td>Etapa 1 Preliminares: Analizar el proyecto “Aplicaciones Criptográficas Java” y los programas “Encrypt Easy”, “EasyCrypto”, “EncryptOnClick” y “AxCrypt”</td></tr>
<tr><td>Etapa 2 Desarrollo: Construir la aplicación “Archivos Seguros con JCEF”</td></tr>
<tr><td>Etapa 3 Presentación: Elaborar y ensayar la presentación con el objetivo de obtener el máximo interés del público, resaltando por encima de todo, los resultados y conclusiones más importantes del proyecto. Publicar el proyecto en https//sourceforge.net</td></tr>
<tr><td>Planificación temporal estimada</td><td>
<table>
<tr><td>Etapas</td><td>Duración estimada</td></tr>
<tr><td>Etapa 1 Preliminares</td><td>80 horas</td></tr>
<tr><td>Etapa 2 Desarrollo</td><td>160 horas</td></tr>
<tr><td>Etapa 3 Presentación</td><td>60 horas</td></tr>
</table>
<table>
<tr><td>Duración total</td><td>300 horas</td></tr>
</table>
</td></tr>
</table>

Tabla 76: Futuro Proyecto: Archivos Seguros con JCEF

5 Proveedor Criptográfico JCEF

<table>
<tr><td>Título</td><td>Proveedor Criptográfico JCEF</td></tr>
<tr><td>Autor</td><td>——</td></tr>
<tr><td>Descripción general</td><td>En la actualidad, la seguridad es importantísima. Sin servicios de seguridad, el desarrollo de ciertos sistemas se vería frenado; entre los que destacan: el comercio electrónico y las comunicaciones seguras a través de la red. Por lo que la criptografía se vuelve indispensable</td></tr>
<tr><td>Objetivos</td><td>Diseñar un proveedor de algoritmos criptográficos JCEF totalmente nuevo, es decir, algoritmos criptográficos no implementados hasta ahora en Java</td></tr>
<tr><td>Metodología</td><td>En un principio se utilizará un enfoque clásico de desarrollo pero que podrá adaptarse a la naturaleza del proyecto y de las personas que trabajan en el mismo</td></tr>
<tr><td>Recursos necesarios</td><td>PC, Internet, Linux, Java, Proyecto “Aplicaciones Criptográficas Java” (http://jcef.sourceforge.net), Eclipse IDE, OpenOffice y CamStudio</td></tr>
<tr><td>Perfil del alumno</td><td>Soltura con Java, UML y tener conocimientos de Criptografía y Diseño de software</td></tr>
<tr><td rowspan="3">Etapas</td><td>Etapa 1 Preliminares: Analizar el proyecto “Aplicaciones Criptográficas Java” y sus proveedores JCEF</td></tr>
<tr><td>Etapa 2 Desarrollo: Desarrollar un proveedor JCEF que suministre algoritmos actualmente no disponibles probando el correcto funcionamiento de cada uno de ellos. Es posible ahorrarse las implementaciones desde cero, se podría adaptar implementaciones ya existentes en otros lenguajes. Una fuente de información sería Crypto++ (http://www.eskimo.com/~weidai/cryptlib.html)</td></tr>
<tr><td>Etapa 3 Presentación: Elaborar y ensayar la presentación con el objetivo de obtener el máximo interés del público, resaltando por encima de todo, los resultados y conclusiones más importantes del proyecto. Publicar el proyecto en https//sourceforge.net</td></tr>
<tr><td>Planificación temporal estimada</td><td>
<table>
<tr><td>Etapas</td><td>Duración estimada</td></tr>
<tr><td>Etapa 1 Preliminares</td><td>50 horas</td></tr>
<tr><td>Etapa 2 Desarrollo</td><td>210 horas</td></tr>
<tr><td>Etapa 3 Presentación</td><td>60 horas</td></tr>
</table>
<table>
<tr><td>Duración total</td><td>300 horas</td></tr>
</table>
</td></tr>
</table>

Tabla 77: Futuro Proyecto: Proveedor Criptográfico JCEF

6 Almacén de objetos seguros con JCEF

<table>
<tr><td>Título</td><td>Almacén de objetos seguros con JCEF</td></tr>
<tr><td>Tutor</td><td>——</td></tr>
<tr><td>Descripción general</td><td>En la actualidad, la seguridad es importantísima. Sin servicios de seguridad, el desarrollo de ciertos sistemas se vería frenado; entre los que destacan: el comercio electrónico y las comunicaciones seguras a través de la red. Por lo que la criptografía se vuelve indispensable</td></tr>
<tr><td>Objetivos</td><td>Mejorar el proyecto “Aplicaciones Criptográficas Java” (http://jcef.sourceforge.net), añadiéndole una solución más amplia que el KeyStore de JCE, es decir, que permite almacenar claves y cualquier otro tipo de objetos. Además los almacenes se deben poder gestionar directamente desde su almacenamiento en ficheros empaquetados como por ejemplo del tipo JAR</td></tr>
<tr><td>Metodología</td><td>En un principio se utilizará un enfoque clásico de desarrollo pero que podrá adaptarse a la naturaleza del proyecto y de las personas que trabajan en el mismo</td></tr>
<tr><td>Recursos necesarios</td><td>PC, Internet, Linux, Java, Proyecto “Aplicaciones Criptográficas Java”, Eclipse IDE, OpenOffice, CamStudio y http://jcef.sourceforge.net</td></tr>
<tr><td>Perfil del alumno</td><td>Soltura con Java, UML y tener conocimientos de Criptografía y Diseño de software</td></tr>
<tr><td rowspan="3">Etapas</td><td>Etapa 1 Preliminares: Consistirá en analizar la funcionalidad de KeyStore de la especificación criptográfica Java: JCA y JCE.</td></tr>
<tr><td>Etapa 2 Desarrollo: Desarrollar el almacén de objetos seguros, permitiendo su persistencia y acceso directo en ficheros con formato JAR.</td></tr>
<tr><td>Etapa 3 Presentación: Elaborar y ensayar la presentación con el objetivo de obtener el máximo interés del público, resaltando por encima de todo los resultados y conclusiones más importantes del proyecto. Publicar el proyecto en https//sourceforge.net</td></tr>
<tr><td>Planificación temporal estimada</td><td>
<table>
<tr><td>Etapas</td><td>Duración estimada</td></tr>
<tr><td>Etapa 1 Preliminares</td><td>70 horas</td></tr>
<tr><td>Etapa 2 Desarrollo</td><td>150 horas</td></tr>
<tr><td>Etapa 3 Presentación</td><td>60 horas</td></tr>
</table>
<table>
<tr><td>Duración total</td><td>280 horas</td></tr>
</table>
</td></tr>
</table>

Tabla 78: Futuro Proyecto: Almacén de objetos seguros con JCEF

7 Metaimplementación de "Aplicaciones Criptográficas Java"

<table>
<tr><td>Título</td><td>Metaimplementación de "Aplicaciones Criptográficas Java"</td></tr>
<tr><td>Tutor</td><td>——</td></tr>
<tr><td>Descripción General</td><td>Hoy día el tiempo es muy importante, y también lo es para el desarrollo y mantenimiento de software. Actualmente, las plantillas de código son unas herramientas excepcionales para desarrollar y mantener software, reduciendo considerablemente el tiempo empleado para ello. Por lo tanto, sería interesante realizar la metaimplementación del Proyecto Fin de Carrera "Aplicaciones Criptográficas Java" utilizando dichas herramientas</td></tr>
<tr><td>Objetivos</td><td>Implementar el Proyecto Fin de Carrera "Aplicaciones Criptográficas Java" mediante herramientas de plantillas de código</td></tr>
<tr><td>Metodología</td><td>En un principio se utilizará un enfoque clásico de desarrollo pero que podrá adaptarse a la naturaleza del proyecto y de las personas que trabajan en el mismo</td></tr>
<tr><td>Recursos necesarios</td><td>PC, Internet, herramientas de plantillas de código, Linux, OpenOffice y CamStudio</td></tr>
<tr><td>Perfil del alumno</td><td>Soltura con Java, Linux, OpenOffice y Diseño de software</td></tr>
<tr><td rowspan="3">Etapas</td><td>Etapa 1 Análisis: Buscar, seleccionar y aprender a utilizar la herramienta para plantillas de código Java más adecuada. Un buen punto de partida es el enlace "Template Engines" en http://www.java-source.net/</td></tr>
<tr><td>Etapa 2 Desarrollo: Se trata de analizar brevemente la implementación del proyecto "Aplicaciones Criptográficas Java" y reimplementarlo utilizando para ello la herramienta para plantillas de código Java seleccionada con anterioridad</td></tr>
<tr><td>Etapa 3 Presentación: Elaborar y ensayar la presentación con el objetivo de obtener el máximo interés del público, resaltando por encima de todo, los resultados y conclusiones más importantes del proyecto. Publicar el proyecto en https//sourceforge.net</td></tr>
<tr><td>Planificación temporal estimada</td><td>
<table>
<tr><td>Etapas</td><td>Duración estimada</td></tr>
<tr><td>Etapa 1 Búsqueda</td><td>50 horas</td></tr>
<tr><td>Etapa 2 Análisis</td><td>200 horas</td></tr>
<tr><td>Etapa 3 Presentación</td><td>50 horas</td></tr>
</table>
<table>
<tr><td>Duración total</td><td>300 horas</td></tr>
</table>
</td></tr>
</table>

Tabla 79: Futuro Proyecto: Metaimplementación de "Aplicaciones Criptográficas Java"

6.Recursos Utilizados

A continuación se muestra un resumen de los recursos empleados para el desarrollo del proyecto:

- Para obtener información: Bibliografía sobre *seguridad*, *UML*, *Java* y *Criptografía*. También se han usado páginas web sobre *Java*, *Javadoc*, *Curso de técnicas de venta*, *Criptografía*, *Algoritmos criptográficos*, *JUnit*, *Eclipse* y *HTML*. Y por si no fuera poco, además se han empleado diccionarios o enciclopedias online tales como *"Real Academia Española"*, *"Wikipedia"* y *"Wordreference"*. *"MailxMail"*

- Para trabajar en el proyecto: Obviamente se ha empleado un *ordenador personal* con un *sistema operativo* instalado y listo para su uso.

- Para dar valor al proyecto: Se han utilizado todos los proveedores criptográficos existentes de algoritmos criptográficos para Java que se han encontrado: *"BouncyCastle"*, *"GNU Crypto"*, *"IAIK"*, *"Cryptix"*, *"CryptixCrypto"*, *"JHBCI"*, *"LogiCrypto"*, *"Mindbright"*, *"Jacksum"*, *"FlexiCore"*, *"FlexiEC"*, *"FlexiNF"*, *"SUN"*, *"SunJCE"*, *"SunJSSE"* y *"SunRsaSign"*.

- Como apoyo al proyecto: De alguna u otra forma, también se han empleado las librerías de desarrollo: *"OsterMiller Utils"*, *"JODE"*, *"Apache Jakarta Commons Lang", "Junit" y "API de J2SE"*.

- Para el desarrollo del proyecto: Se ha utilizado principalmente el entorno de desarrollo para Java *"Eclipse"* y también algunos de sus *plugins* para HTML, JUnit, JODE y Java2HTML.

- Para gestionar archivos comprimidos: *"7-zip"* y *"AlZip"* han sido utilizados.

- Como herramientas de documentación: *"OpenOffice"*, *"Javadoc"*, *"ArgoUML"*, *"PDFCreator" y "Java2HTML"*.

- Como navegador web para Internet: *"Mozilla Firefox"*.

- Para diseñar la página web del proyecto: *"Nvu"* y una plantilla para dicha página web.

- Para publicar el proyecto en Internet: *"WinSCP", "FileZilla"* y *"SourceForge.net"*.

- Como lenguajes de desarrollo, se han utilizado *"Java"*, *"HTML"*, *"XML"*, *"UML"* y *"CSS"*.

- Para grabar CDs: "CDBurnerXP Pro".

- Para la eventual edición de archivos de texto: *"JEdit"* y *"Notepad++"*.

- Como programas de edición y gestión de imágenes, se han empleado *"GIMP"*, *"Irfanview"* y *"Paint.NET"*.

- También se han empleado directrices de desarrollo, es decir, consejos, normas y/o cuestiones a tener en cuenta sobre *"cualquier tipo de producto"*, *"la importancia del cliente"*, *"el desarrollo de un producto"*, *"la presentación de un producto"* y *"la documentación de un producto"*.

- Además se han usado otras herramientas para gestionar notas como *"NotesHolder Lite"*, para ejecutar fácilmente en Windows aplicaciones Java como *"JSmooth"* y para realizar copias de seguridad se ha empleado *"Cobian Backup"*.

En las siguientes secciones se podrá encontrar más información en detalle sobre todos los recursos utilizados para la realización del proyecto. Y también en las direcciones http://jcef.sourceforge.net/doc/resources.pdf y http://jcef.sourceforge.net y http://sourceforge.net/projects/jcef/.

1 Bibliografía

1 Bibliografía sobre Seguridad

- *Fundamentos de Seguridad en Redes. Aplicaciones y Estándares*
  - Resumen: Criptografía, Aplicaciones y Estándares de seguridad de redes y Seguridad de los sistemas
  - Página oficial: http://www.librosite.net/stallings4
  - Otros datos: Edición: 2ª; ISBN: 84-205-4002-1; Autor: William Stallings

- *Técnicas Criptográficas de protección de datos*
  - Resumen: Criptografía, Aplicaciones y Fundamentos Matemáticos
  - Página oficial: http://www.ra-ma.es (http://www.ra-ma.es/libros/0001689.htm)
  - Otros datos: Edición: 2ª; Editorial: Ra-Ma; ISBN: 84-7897-421-0; Autor: Amparo Fúster Sabater y otros tantos más

- *Criptografía Digital*
  - Resumen: Fundamentos y Aplicaciones de la Criptografía
  - Otros datos: Edición: 2ª; Editorial: Prensas Universitarias de Zaragoza; ISBN: 84-7733-558-3; Autor: José Pastor Franco y otros tantos más

- *Hackers de Java y J2EE*
  - Resumen: Aprender a desarrollar aplicaciones Java seguras
  - Página oficial: http://www.mcgraw-hill.es (http://www.mcgraw-hill.es/html/8448121856.html)
  - Otros datos: Editorial: McGraw-Hill; ISBN: 84-481-2185-6; Autores: Art Taylor, ...;

2 Bibliografía sobre UML

- *El Lenguaje Unificado de Modelado. Manual de Referencia*
  - Resumen: Aprender UML y sus detalles
  - Otros datos: Editorial: Pearson Educación; Autores: James Rumbaugh, Ivar Jacobson, Grady Booch; ISBN: 84-7829-037-0

- *El Proceso Unificado de Desarrollo de Software*
  - Resumen: Aprender a desarrollar software con UML
  - Otros datos: Editorial: Pearson Educación; Autores: James Rumbaugh, Ivar Jacobson, Grady Booch; ISBN: 84-7829-036-2

- *El Lenguaje Unificado de Modelado*
  - Resumen: Aprender UML
  - Otros datos: Editorial: Pearson Educación; Autores: James Rumbaugh, Ivar Jacobson, Grady Booch; ISBN: 84-7829-028-1

3 Bibliografía sobre Java

- *Programación en Java 2 J2SE 1.4*
  - Resumen: Completísimo libro para aprender Java y sus librerías J2SE 1.4
  - Otros datos: Editorial: Anaya; Autor: John Zukowski; ISBN: 84-415-1559-X

2 Webs

1 Webs secundarias sobre Java

- *Extensa colección de ejemplos Java organizados por paquetes*
  - Página oficial: http://javaalmanac.com/
  - Idioma: Inglés
- *Importante punto de inicio para aprender Java*
  - Página oficial:
    - http://directory.google.com/Top/Computers/Programming/Languages/Java/
  - Idioma: Inglés
- *Espléndidos tutoriales sobre Java*
  - Página oficial: http://java.sun.com/docs/books/tutorial/index.html
  - Idioma: Inglés
- *Grandes cantidades de recursos sobre Java*
  - Página oficial: http://www.javareference.com
  - Idioma: Inglés
- *Importante portal sobre programación en las principales tecnologías, entre ellas Java*
  - Página oficial: http://www.programacion.net, http://www.programación.com
  - Idioma: Castellano
  - Contenido: Noticias, Blogs, Tutoriales, Artículos, Cursos, Foros, Código Fuente, Direcciones, Formación, HTML, Java, PHP, ASP, Bases de Datos
- *Página web en castellano con interesante información sobre Java*
  - Página oficial: http://www.javahispano.org
  - Idioma: Castellano
  - Contenido: Noticias, Tutoriales, Artículos, Cursos, Código útil, Java FAQ, Tips, Eventos, Monográficos, Enlaces, Entrevistas

2 Webs principales sobre Java

- *Guía de referencia sobre la seguridad en Java*
  - Página oficial: http://java.sun.com/j2se/1.5.0/docs/guide/security/index.html
  - Idioma: Inglés
  - Contenido: Mejoras de Seguridad en JDK 5.0, Guía de Seguridad y Guía de Caminos de Certificación, Servicio de Autentificación y Autorización (JAAS), Servicios Genéricos de Seguridad (JGSS), Extensión Criptográfica de Java (JCE), Extensión de Sockets Seguros de Java (JSSE), Capa Simple de Autentificación y Seguridad (SASL), Documentación, Herramientas y Tutoriales de Seguridad
- *Página principal de Sun sobre la seguridad con Java*
  - Página oficial: http://java.sun.com/security/index.jsp
  - Idioma: Inglés
  - Contenido: Java Cryptographic Extension (JCE), Java Authentication and Authorization Service (JAAS), Java Secure Socket Extension (JSSE), Especificaciones, Documentación, FAQs, Artículos, ...
- *Documentación sobre la API J2SE 5.0*
  - Página oficial: http://java.sun.com/j2se/1.5.0/docs/api/
  - Idioma: Inglés
- *Tutorial introductorio sobre los JavaBeans*
  - Página oficial: http://www.programacion.net/java/tutorial/beans/
  - Idioma: Castellano
  - Contenido: JavaBeans, Componentes de la Plataforma, Java Conceptos sobre los JavaBeans y contenido del BDK, BeanBox: Uso, Arranque, Menús, Applets, Eventos, Beans: Ejemplo sencillo, Propiedades Beans: Sencillas, Compartidas, Restringidas e Indexadas, BeanInfo y la Introspección, Personalización y Persistencia de los Beans, Nuevas Características de JavaBeans
- *Tutorial introductorio sobre los JavaBeans*
  - Página oficial:
    - http://java.sun.com/developer/onlineTraining/Beans/JBeansAPI/shortcourse.html
  - Idioma: Inglés

- *Documentación de referencia sobre JCE*
  - Página oficial: http://java.sun.com/products/jce/index-14.html
  - Idioma: Inglés
  - Contenido: Introducción y ¿Por qué usar JCE?, Características importantes y Guía de Referencia de JCE
- *Guía de referencia sobre JCE*
  - Página oficial:
    - http://java.sun.com/j2se/1.5.0/docs/guide/security/jce/JCERefGuide.html
  - Idioma: Inglés
  - Contenido: Introducción y Lo nuevo de JCE en el J2SE 5.0, Conceptos criptográficos y Las clases principales, Instalación de Proveedores y Códigos de ejemplo, Apéndice: Información sobre los algoritmos, tamaños de clave, programas de ejemplo, etc.
- *Guía de referencia sobre la librería criptográfica JCA*
  - Página oficial: http://java.sun.com/j2se/1.5.0/docs/guide/security/CryptoSpec.html
  - Idioma: Inglés
  - Contenido: Introducción, Las clases principales y Códigos de ejemplo, Apéndice: Información sobre los algoritmos, tamaños de clave, programas de ejemplo, etc.
- *Lista de proveedores criptográficos reconocidos por Sun Microsystems*
  - Página oficial: http://java.sun.com/products/jce/jce122_providers.html
  - Idioma: Inglés

3 Curso de técnicas de venta

- Título: Entiendo, luego vendo: técnicas de venta
- Página oficial: http://www.mailxmail.com/curso/empresa/vender
- Idioma: Español

4 Webs sobre JUnit

- *Tutorial sobre Junit*
  - Página oficial: http://www.programacion.com/java/articulo/jap_junit/
  - Idioma: Castellano
  - Contenido: ¿Por qué utilizar Junit?. Diseño de Junit. Ejemplos de utilización. Prácticas recomendables para su uso
- *Tutorial "Pruebas de programas Java mediante Junit"*
  - Página oficial: http://www.inf-cr.uclm.es/www/mpolo/tutorial/
  - Idioma: Castellano
  - Contenido: Características del diseño Junit. Ejemplos reales de su uso junto con su código fuente

5 Webs sobre Eclipse

- *Almacén de plugins para Eclipse*
  - Página oficial:http://eclipse-plugins.2y.net/eclipse/plugins.jsp
  - Idioma: Inglés
  - Contenido: Conjunto de módulos/plugins Eclipse organizados por categorías, tales como: generación de código, bases de datos, documentación, electrónica, lenguajes, modelado, redes, patrones, SCM, team, testing, UML, UI, web, etc.

6 Webs sobre HTML

- *Tutorial sobre HTML*
  - Página oficial: http://www.um.es/~psibm/tutorial/
  - Idioma: Castellano
  - Contenido: Comandos básicos, Listas, Imágenes, Enlaces, Tablas, Funciones especiales y Recomendaciones

7 Webs sobre Javadoc

- *Tutorial introductorio para Javadoc*
  - Página oficial: http://ict.udlap.mx/people/carlos/tutoriales/javadoc/javadoctutorial.pdf
- *Guía de referencia sobre Javadoc con ejemplos*
  - Página oficial: http://java.sun.com/j2se/1.5.0/docs/tooldocs/windows/javadoc.html
- *Importante tutorial para aprender a manejar Javadoc de la mejor forma posible*
  - Página oficial: http://java.sun.com/j2se/javadoc/writingdoccomments/

8 Webs principales sobre Criptografía

- *Importante curso sobre la seguridad, la criptografía y Java*
  - Página oficial: http://www.uv.es/~sto/cursos/seguridad.java/html/sjava.html
  - Idioma: Castellano
  - Contenido: Criptología, Cifrado, Autentificación hash y mac, Firmas digitales. Seguridad de los algoritmos, Ataques más importantes. Aplicaciones, Técnicas criptográficas, Certificados Digitales. Protocolos SSL y TLS. La seguridad en Java, su arquitectura criptográfica JCA y su extensión JCE. JSSE: Extensión de Sockets Seguros de Java. JAAS: Servicio de Autentificación y Autorización de Java. Referencias
- *Tutorial muy interesante y resumido sobre la criptografía y sus aplicaciones*
  - Página oficial: http://www.cert.fnmt.es/tuto1.htm
  - Idioma: Castellano
  - Contenido: Criptografía, Cifrado, Firma digital, Certificados digitales, Terceras partes de confianza, Infraestructura de clave pública
- *Página web de un completísimo libro y curso electrónico sobre seguridad y criptografía en español*
  - Página oficial: http://www.criptored.upm.es/guiateoria/gt_m001a.htm
  - Idioma: Castellano
  - Contenido: Criptografía. Seguridad Informática. Cifrados. Autentificación hash. Firmas digitales. Certificados digitales. Aplicaciones de correo seguro. Protocolos criptográficos. Curvas elípticas. Bibliografía. Enlaces. Tablas. Software. Documentos

9 Webs secundarias sobre Criptografía

- *Página oficial del libro gratuito: "Handbook of Applied Cryptography"*
  - Página oficial: http://www.cacr.math.uwaterloo.ca/hac/
  - Idioma: Inglés
  - Contenido: Criptografía. Matemáticas y Teoría de Números. Parámetros. Cifrados | Autentificación Hash. Firmas digitales. Protocolos de intercambio de claves. Técnicas de gestión de claves. Implementaciones eficientes. Patentes y Standards
- *Página personal muy interesante sobre el mundo de la Criptología*
  - Página oficial: http://webs.ono.com/usr005/jsuarez/diffiehellman.html
  - Idioma: Castellano
  - Contenido: Criptografía. Criptoanálisis. Otras técnicas. Noticias. Colaboraciones. Varios
- *Web personal con interesante información sobre criptografía, seguridad y otras cosas*
  - Página oficial: http://jo.morales0002.eresmas.net/fsumario.html
  - Idioma: Castellano
  - Contenido: Criptografía. Seguridad. Interesantes enlaces.
- *Seguridad en agentes móviles basados en la criptografía*
  - Página oficial: http://grasia.fdi.ucm.es/jpavon/agentes/
  - Idioma: Castellano
  - Contenido: Criptografía. Firmas digitales. Análisis del protocolo SET. Información sobre agentes de seguridad
- *Introducción a la seguridad, sus servicios y Java*
  - Página oficial: http://java.sun.com/security/javaone97-whitepaper.html
  - Idioma: Inglés
  - Contenido: Introducción a la seguridad y sus servicios. La seguridad en Java

10 Webs sobre Algoritmos Criptográficos

- *Información de referencia con independencia de la implementación sobre muchos algoritmos criptográficos*
  - Página oficial: http://www.users.zetnet.co.uk/hopwood/crypto/scan/
  - Idioma: Inglés
- *Extensa enciclopedia sobre la criptografía y muchos otros conceptos*
  - Página oficial:
    - http://www.absoluteastronomy.com/encyclopedia/C/Cr/Cryptography.htm
  - Idioma: Inglés
  - Contenido: Introducción a la Criptografía. Algoritmos de cifrado de bloques, de flujo y de resumen de mensajes
- *Conjunto de algoritmos criptográficos muy populares*
  - Página oficial: http://kremlinencrypt.com/algorithms.htm
  - Idioma: Inglés
  - Contenido: Algoritmos de cifrado de bloques, de flujo y de resumen de mensajes
- *Otro conjunto de algoritmos criptográficos muy populares*
  - Página oficial: http://www.baltsoft.com/files/ee/Cryptographic_Algorithms.htm
  - Idioma: Inglés
  - Contenido: Algoritmos de cifrado de bloques, de flujo y de resumen de mensajes
- *Otra extensa enciclopedia sobre la criptografía y muchos otros conceptos*
  - Página oficial: http://www.answers.com/topic/cryptography-1
  - Idioma: Inglés
  - Contenido: Introducción a la Criptografía. Multitud de enlaces muy interesantes. Algoritmos de Cifrado de Bloques, de Flujo y de Resumen de Mensajes

3 Eclipse

- Descripción: La plataforma Eclipse es un entorno de desarrollo integrado (IDE) abierto y extensible que proporciona los componentes y la metodología para la creación y utilización de herramientas de desarrollo de software integradas. La arquitectura de plugins de Eclipse permite, además de integrar diversos lenguajes sobre un mismo IDE, introducir otras aplicaciones accesorias que pueden resultar útiles durante el proceso de desarrollo como: herramientas UML, editores visuales de interfaces, ayuda en línea para librerías, etc...
- Características: Editor de texto y resaltado de sintaxis | Adición de módulos con suma facilidad. Compilación en tiempo real e integración con Ant. Tests unitarios con Junit y control de versiones con CVS. Asistentes: para creación de proyectos, clases, tests, etc... Etc...
- Categoría: Software de desarrollo
- Licencia: CPL; ¿Open Source?: Sí; ¿Gratuito?: Sí;Versión: 3.1
- Página oficial: http://www.eclipse.org/
- Alternativas: Borland JBuilder Foundation 2005

4 API de J2SE

- Descripción: API de Java (J2SE) es un conjunto de clases que es utilizado para generar programas básicos en el lenguaje; utilizando una analogía, estas clases tienen la misma funcionalidad que las funciones clases estándar utilizadas en otros lenguajes como C o C++.
- Características: Contiene funcionalidades generales, para cálculos matemáticos, beans, comunicaciones, interfaces gráficas, seguridad, criptografía, bases de datos, XML, CORBA, multimedia, impresión, imágenes, utilidades, RMI, etc.
- Categoría: Librería de desarrollo estándar para Java
- ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.5
- Página oficial: http://java.sun.com/j2se/index.html
- Alternativas: GNU Classpath (http://www.gnu.org/software/classpath/classpath.html)

5 Plugins de Eclipse

- *Eclipse JUnit Plugin*
  - Descripción: Módulo de Eclipse que facilita la realización de pruebas mediante JUnit
  - Características: Asistentes para la creación de unidades de prueba. Control de las pruebas superadas y fracasadas. Integrado en el Eclipse base
- *Eclipse HTML Editor Plugin*
  - Descripción: Módulo para Eclipse utilizado para la edición de los lenguajes de marcado HTML, JSP y XML
  - Características: Resaltado HTML, JSP, XML y CSS. Previsualización HTML y JSP. Validación JSP y XML. Asistentes para crear archivos HTML, JSP o XML. Editor de preferencias. Navegador web embebido en Eclipse. Visor de imágenes. Paleta de etiquetas. Y otras muchas características
  - Página oficial:
  - http://amateras.sourceforge.jp/cgi-bin/fswiki_en/wiki.cgi?page=EclipseHTMLEditor
  - Licencia: Open Source; ¿Open Source?: Sí
- *JODE Plug-in Eclipse*
  - Descripción: Integra el decompilador JODE en Eclipse
  - Utilizado para: Consultar el código cuando éste no está disponible, sólo su versión bytecode (versión compilada)
  - Características: Funciona de manera automática y transparente para el usuario
  - Página oficial:
  - Licencia: GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.0.5
- *Java2HTML Plug-in Eclipse*
  - Descripción: Convierte el código java en texto formateado a color en HTML, RTF, TeX y XHTML.
  - Página oficial: http://www.java2html.de/
  - Licencia: GPL o CPL1.0; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.5.0

6 Proveedores Criptográficos JCE

1 BouncyCastle

- Descripción: La librería Bouncy Castle Crypto es una implementación Java de algoritmos criptográficos. Es uno de los mejores proveedores de servicios criptográficos que hay para Java
- Características: Proveedor JCE y JCA. Generación de certificados y listas de anulación de certificados. Otras muchas características interesantes tales como S/MIME, OCSP, TSP, OpenPGP, etc... Incluye código de ejemplo para aprender a utilizar, además del conjunto de pruebas realizadas para verificar su funcionamiento
- Página oficial: http://www.bouncycastle.org
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.29

2 IAIK

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://jce.iaik.tugraz.at/
- Licencia: Freeware; ¿Open Source?: No; Versión: 3.13
- ¿Gratuito?: Sí, pero sólo para usos no comerciales

3 Cryptix

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://www.cryptix.org/
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 28/03/2005

4 CryptixCrypto

- Descripción: Proveedor de servicios criptográficos para JCE
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 28/03/2005
- Página oficial: http://www.cryptix.org/

5 FlexiCore

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://www.flexiprovider.de/
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.1.5

6 FlexiEC

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://www.flexiprovider.de/
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.1.5

7 FlexiNF

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://www.flexiprovider.de/
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.1.5

8 GNU Crypto

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://www.gnu.org/software/gnu-crypto/
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 2.0.1

9 JHBCI

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://www.jhbci.de/
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 0.0.6

10 SUN

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://java.sun.com/j2se/
- ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 1.5

11 SunJCE

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://java.sun.com/j2se/
- ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 1.5

12 SunJSSE

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://java.sun.com/j2se/
- ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 1.5

13 SunRsaSign

- Descripción: Proveedor de servicios criptográficos para JCE
- Página oficial: http://java.sun.com/j2se/
- ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 1.5

7 Proveedores Criptográficos no JCE

1 Jacksum

- Descripción: Proveedor de servicios criptográficos. No cumple la especificación JCE
- Página oficial: http://www.jonelo.de/java/jacksum/
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.5.1

2 Logi Crypto

- Descripción: Proveedor de servicios criptográficos. No cumple la especificación JCE
- Página oficial: http://www.logi.org/logi.crypto/
- Licencia: Open Source; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.1.2

3 Mindbright

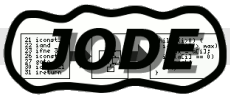
- Descripción: Proveedor de servicios criptográficos. No cumple la especificación JCE
- Página oficial: http://www.appgate.com/products/80_MindTerm/
- Licencia: Open Source; ¿Open Source?: Sí; Versión: 2.4.2
- ¿Gratuito?: Sí, excepto para usos comerciales

8 Librerías de Desarrollo (APIs)

1 Ostermiller Utils

- Descripción: Varias utilidades Java, entre las que destaca un generador de contraseñas
- Página oficial: http://ostermiller.org/
- Licencia: GNU GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.04.03

2 JODE



- Descripción: Decompilador, optimizador y ofuscador para el código bytecode Java
- Página oficial: http://jode.sourceforge.net/
- Licencia: GNU LGPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.1.1

3 Apache Jakarta Commons Lang

- Descripción: Utilidades generales para Java
- Página oficial: http://jakarta.apache.org/commons/lang/
- Licencia: Apache; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 2.1

4 JUnit

- Descripción: Librería para automatizar la ejecución de pruebas unitarias para software orientado a objetos
- Características: Particular para las pruebas de programas Java. Es adecuado para el desarrollo dirigido por las pruebas
- Página oficial: http://www.junit.org
- Licencia: CPL 1.0; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 3.8.1

9 Utilidades de compresión

1 7-zip

- Descripción: 7-Zip es un programa de compresión de datos muy bueno
- Características: Alto porcentaje de compresión con el nuevo formato 7z (el mejor de momento). Además de 7z, acepta: ZIP, CAB, RAR, ARJ, GZIP, BZIP2, Z, TAR, CPIO, RPM y DEB. Capacidad de autoextracción para el formato 7z. Integración con el intérprete de Windows, potente administrador de ficheros y línea de comandos. Utilizable a través de una interfaz gráfica de usuario o una línea de comandos. Microsoft Windows y Linux. Extensión para FAR Manager y Traducción a 53 idiomas
- Licencia: GNU LGPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Version:
- Página oficial: http://www.7-zip.org
- Alternativa Freeware: IZArc (http://www.izarc.org/)
- Alternativas Comerciales:
  - WinZip (http://www.winzip.com)
  - WinRar (http://winrar.com.es)

2 AlZip

- Descripción: Es una utilidad de compresión con multitud de formatos
- Utilizado para: Acceder de forma eventual a ficheros ".jar"
- Características: Soporta 36 formatos de compresión (ZIP, ACE, RAR, LZH, ...). Proporciona soporte "Drag & Drop". Interfaz gráfica amigable y acceso desde la línea de comandos. Completo manual de usuario. Múltiples idiomas, incluyendo español. Integrable con el explorador de Windows
- Página oficial: http://www.altools.net
- Alternativas Open Source: 7-zip (http://www.7-zip.org)
- Alternativas Comerciales:
  - WinZip (http://www.winzip.com)
  - WinRar (http://winrar.com.es)
- Licencia: Freeware; ¿Gratuito?: Sí; Versión: 6.13

10 Herramientas de documentación

1 OpenOffice

- Descripción: Es un suite de oficina de software libre, gratis para todo el mundo, y que tiene módulos.
- Módulos: Procesador de textos: OpenOffice.org Writer. Hoja de cálculo: OpenOffice.org Calc. Gráficos vectoriales: OpenOffice.org Draw. Presentaciones: OpenOffice.org Impress. Bases de datos: OpenOffice.org Base. Fórmulas matemáticas: OpenOffice.org Math. Y otras características como edición HTML y generación PDF
- Licencia: GNU LGPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 2.0
- Página oficial: http://www.openoffice.org
- Alternativas Comerciales: Microsoft Office: Word, PowerPoint, Excel, Access. Microsoft Visio. Smartdraw

2 Javadoc

- Descripción: Herramienta de documentación enfocada a programas realizados en el lenguaje Java, la cual es semiautomática al facilitar la generación de un grupo de archivos HTML en base a una sintáxis de comentarios insertados en el código fuente.
- Página oficial: http://java.sun.com/j2se/javadoc/index.jsp

3 Java2HTML

- Descripción: Convierte el código java en texto formateado a color en HTML, RTF, TeX y XHTML.
- Página oficial: http://www.java2html.de/
- Licencia: GPL o CPL1.0; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 5.0

4 Metrics Analisys Tool

- Descripción: Permite obtener métricas sobre un conjunto de ficheros código fuente de Java.
- Características: Obtiene métricas tales como número de clases, interfaces, métodos, campos, ficheros, líneas de código, líneas de comentarios, etc...
- Página oficial: http://www.cs.appstate.edu/~jrc/metricsTool.html
- Versión: 1.1; Licencia: GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí

5 ArgoUML

- Descripción: Herramienta de modelado para realizar diseños en UML
- Página oficial: http://argouml.tigris.org
- Licencia: BSD; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 0.16.1
- Alternativas: Umbrello UML Modeller (http://uml.sourceforge.net/)

6 PDFCreator

- Descripción: Sencilla aplicación para crear documentos PDF
- Características: Permite crear documentos PDF a partir de cualquier otra aplicación que permita imprimir. Utiliza una impresora virtual para la comunicación con el resto de aplicaciones. Permite configurar el grado de compresión y muchas características de los PDF. Crea documentos en los siguientes formatos: Portable Document Format (PDF), Postscript (PS), Encapsulated Postscript (EPS), PNG, JPEG, BMP, PCX y TIFF. Combina múltiples documentos en un único PDF. Permite proteger los documentos PDF con contraseñas de 128 bits. Permite autoguardado y mucho más ...
- Página oficial:
  - http://www.pdfcreator.de.vu/
  - http://www.sourceforge.net/projects/pdfcreator
- Versión: 0.8.1 RC10; Licencia: GPL y AFPL; ¿Open Source?: Sí; ¿Gratuito?: Sí
- Idioma: Español

11 Utilidades de Internet

1 Mozilla Firefox

- Descripción: Navegador web
- Página oficial: http://www.mozilla.org/products/firefox/
- ¿Open Source?: Sí; Versión: 1.5
- Alternativas: Microsoft Internet Explorer

2 Nvu

- Descripción: Programa para la creación de páginas web HTML
- Características: Sistemas operativos soportados: Windows, Macintosh, Linux. Fácil de usar, potente e ideal para no programadores
- Página oficial: http://www.nvu.com/
- Licencia: MPL 1.1; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 1.0
- Alternativas: FrontPage (http://www.microsoft.com/frontpage/). Dreamweaver (http://www.macromedia.com/software/dreamweaver/)

3 WinSCP

- Descripción: Cliente SFTP gráfico para Windows que emplea SSH y SCP. Su función principal es facilitar la transferencia segura de archivos entre dos sistemas informáticos, el local y uno remoto que ofrezca servicios SSH.
- Utilizado para: Acceder al servidor de SourceForge.net para publicar la página web utilizando el protocolo SFTP para acceso al host shell.sourceforge.net y a la carpeta "/home/groups/j/jc/jcef/htdocs".
- Características: Interfaz gráfica, disponible en varios idiomas, integración con Windows, soporte de las operaciones comunes de archivo, soporte de protocolos SCP y SFTP sobre SSH-1 y SSH-2, soporte de operaciones programadas-batch, sincronización de directorios, soporte de autentificación, interfaces similares a gestores de archivos populares, soporte para guardar la información de la sesión, gestión portátil de la configuración del programa, etc...
- Página oficial: http://winscp.net/
- Licencia: GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 3.8

4 FileZilla

- Descripción: Cliente rápido FTP y SFTP para Windows con muchas características.
- Utilizado para: Subir las distribuciones del proyecto de forma anónima y en modo binario al servidor FTP upload.sourceforge.net y a la carpeta "/incoming" para posterior definir los "file release".
- Página oficial: http://filezilla.sourceforge.net/
- Licencia: GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 2.2.22

12 Lenguajes de desarrollo

1 Java

- Descripción: Java es una plataforma de software, de tal manera que los programas creados en ella puedan ejecutarse de la misma forma en diferentes tipos de arquitecturas y dispositivos computacionales
- Página oficial: http://java.sun.com
- Componentes: El lenguaje de programación totalmente orientado a objetos. La máquina virtual de Java (Java Virtual Machine, JVM), que permite la portabilidad en ejecución. El API Java, una biblioteca estándar para el lenguaje. El entorno de ejecución (Java Runtime Environment, JRE)
- Creador: Sun Microsystems

2 HTML

- Descripción: El HTML, acrónimo inglés de Hyper Text Markup Language (lenguaje de marcación de hipertexto), es un lenguaje de marcas diseñado para estructurar textos y presentarlos en forma de hipertexto, que es el formato estándar de las páginas web
- Página oficial: http://www.w3.org/MarkUp/

3 XML

- Descripción: XML es un metalenguaje para lenguajes de marcado como HTML basado en documentos de texto plano, en los que se utilizan etiquetas para delimitar los elementos del documento
- Aplicaciones: Conseguir una página web más semántica y suceder al HTML. Separar la estructura del contenido. Estándar para el intercambio de datos entre diversas aplicaciones. Y muchísimas otras aplicaciones
- Página oficial: http://www.w3.org/XML/
- Creador: World Wide Web Consortium (W3C)

4 UML

- Descripción: Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, Unified Modelling Language) es el lenguaje de modelado de software más conocido en la actualidad. Utiliza modelos funcionales, de objetos y dinámicos.
- Página oficial: http://www.uml.org/

5 CSS

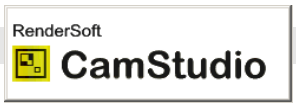
- Descripción: Las hojas de estilo en cascada (Cascading Style Sheets, CSS) son un lenguaje formal usado para definir la presentación de un documento estructurado en HTML o XML (y por extensión en XHTML). La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación.
- Página oficial: http://www.w3.org/Style/CSS/

13 Utilidades de presentación

1 Macromedia Flash Player

- Descripción: Reproductor de películas Flash SWF
- Página oficial: http://www.macromedia.com/es/software/flashplayer/
- Licencia: Freeware; ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 7.0.19.0
- Idioma: Español; Fecha: 10/12/2004

2 CamStudio



- Descripción: Permite grabar la actividad de la pantalla en una película
- Características: Útil para demostraciones de utilización de software. Grabar secuencias de una película en reproducción. Formatos soportados: AVI vídeo y además en SWF (Flash). Genera películas Flash SWF a partir de vídeos AVI. Graba eventos tales como los movimientos del cursor del ratón, hacer clicks y la escritura de caracteres. Además, es posible grabar sonido desde un micrófono, los altavoces o ambas
- Página oficial: http://sourceforge.net/projects/camstudio/ y http://www.camstudio.org/
- Licencia: Freeware; ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 2.00
- Alternativas Comerciales: Camtasia Studio 2.0 (http://www.techsmith.com)
- Alternativas Open Source: xvidcap (http://xvidcap.sourceforge.net). SWFtools (http://www.quiss.org/swftools)

3 Plantilla para página web

- Descripción: Almacén de plantillas para páginas web
- Página oficial: http://www.templatesbox.com
- ¿Gratuito?: Sí; Idioma: ¿Español?

14 Utilidades de Edición de Archivos de Texto

1 JEdit

- Descripción: Editor de ficheros de texto plano que soporta múltiples lenguajes, principalmente de programación
- Características: Extensible mediante módulos (plugins). Escrito en Java. Lenguaje de macros. Sistemas Operativos soportados: Mac OS X, OS/2, Unix, VMS y Windows. Disponibilidad de un repositorio de macros y módulos en Internet. Gestor de módulos para actualizaciones sencillas y más de 130 lenguajes. Resaltado de la sintaxis y autoidentación. Vistas avanzadas del texto y "word wrap". Soporta un gran número de codificación de caracteres, tales como UTF8 y Unicode
- Página oficial: http://www.jedit.org
- Licencia: GNU GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 4.2
- Alternativas: Editplus (http://www.editplus.com)

2 Notepad++

- Descripción: Edita archivos de texto
- Utilizado para: Editar archivos de texto rápidamente
- Características: Soporta el resaltado de más de 36 lenguajes: Java, HTML, Assembler, PHP, VHDL, C, ... Permite definir nuevos lenguajes. Y muchas otras características más
- Página oficial: http://notepad-plus.sourceforge.net
- Versión: 3.3; Licencia: GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí

15 Utilidades de Edición de Imágenes

1 GIMP

- Descripción: Programa de manipulación y retoque de imágenes
- Página oficial: http://www.gimp.org
- Versión Windows: http://gimp-win.sourceforge.net/
- Licencia: GNU GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 2.2
- Alternativas comerciales: Adobe Photoshop

2 Irfanview

- Descripción: Visor de gráficos y archivos multimedia
- Características: Soporta infinidad de formatos de imágenes, audio y vídeo. Soporta multitud de idiomas. Álbum de imágenes. Soporta filtros de Adobe Photoshop. "Arrastrar y Soltar". Reproductor multimedia y conversiones programadas. Opciones de impresión y efectos (sharpen, blur, ...). Posibilidad de añadir nuevos módulos/plugins. Y muchas otras características más
- Página oficial: http://www.irfanview.com/
- Licencia: Freeware; ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 3.97
- Alternativas comerciales: ACDSee (http://www.acdsystems.com)

3 Paint.NET

- Descripción: Aplicación de manipulación de imágenes y fotos para Windows
- Utilizado para: Añadir fácilmente transparencias a imágenes
- Características: Soporta capas, deshacer sin límites, efectos especiales y una amplia variedad de herramientas útiles y poderosas. Ampliación mediante plugins. Permite formatos PNG, JPEG, BMP, GIF, TGA y TIFF. Interfaz gráfica de usuario simple e intuitiva
- Página oficial: http://www.eecs.wsu.edu/paint.net/
- Licencia: MIT; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 2.5

16 SourceForge.net

- Descripción: Web de desarrollo de software Open Source más grande del mundo.
- Características: Proporciona acceso al almacén más grande de Internet sobre código y aplicaciones Open Source. Además, también proporciona servicios gratuitos a los desarrolladores de software Open Source
- Página oficial: https//sourceforge.net
- Alternativas: http://www.tigris.org, http://www.berlios.de/ y http://savannah.gnu.org/.

17 Otros recursos

1 Ordenador Personal

- Características: Procesador AMD y Conexión a Internet
- Sistema Operativo: Windows 2000
- Alternativas Open Source: Fedora Project Core (http://fedora.redhat.com/)

2 CDBurnerXP Pro

- Descripción: Solución para grabar CDs y DVDs
- Características: Formatos: CD-R, CD-RW, DVD+R/RW, DVD-R/RW, DVD doble capa. Graba CDs de audio con y sin pausas entre pistas. 100% libre, sin publicidad ni restricciones. Soporta la mayoría de dispositivos IDE, USB, Firewire y SCSI. Copia CDs de audio al disco duro y obtiene información musical desde Internet. Graba y crea imágenes ISO. Verificación de datos tras grabación. Crea discos de inicio disponibles en múltiples idiomas. Conversor entre imágenes ISO, BIN y NRG. Sistemas operativos soportados: Windows 98/ME/2000/XP/2003 Server
- Página oficial: http://www.cdburnerxp.se/
- Licencia: Freeware; ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 3.0.116
- Alternativas: Nero (http://wwww.nero.com)

3 Real Academia Española

- Descripción: Diccionario online de la Real Academia Española de la lengua
- Página oficial: http://www.rae.es; ¿Gratuito?: Sí

4 Wikipedia

- Descripción: Enciclopedia online libre en más de cien idiomas que todos podemos modificar. Está organizado en artículos y categorías.
- Página oficial: http://www.wikipedia.org; ¿Gratuito?: Sí

5 Wordreference

- Descripción: Diccionarios online de traducción entre idiomas
- Características: Diccionarios Español-Inglés, Francés-Inglés, Italiano-Inglés | Diccionario Inglés
- Página oficial: http://www.wordreference.com
- ¿Gratuito?: Sí; Alternativas: http://www.diccionarios.com/

6 www.mailxmail.com

- Descripción: Iniciativa de "Open E-learning" consistente en ofrecer formación gratuita por Internet destinada al gran público
- Áreas: Informática, Idiomas, Empresa y Calidad de vida
- Página oficial: http://www.mailxmail.com
- ¿Gratuito?: Sí; Idioma: Español; Alternativas: http://www.emagister.com

7 NotesHolder Lite

- Descripción: Sencillo gestor de notas
- Características: Permite programar recordatorios para las notas. Exportar las notas a documentos de texto. Dispone de múltiples idiomas, entre ellos el español. Permite personalizar la interfaz. Se añade a la bandeja del sistema
- Página oficial:
  - http://www.notesholder.com
  - http://notes.aklabs.com
- Licencia: Freeware; ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 1.33

8 JSmooth

- Descripción: Construye ejecutables ".exe" para Windows con el objetivo de ejecutar aplicacions Java de una forma más sencilla
- Utilizado para: Construir la versión ejecutable de ArgoUML
- Características: Sencillo de usar. Interfaz de usuario amigable y bastante completa. Múltiples opciones de construcción: icono, máquina virtual de java requerida, gestión de memoria, ... Completos manuales de usuario
- Página oficial: http://jsmooth.sourceforge.net/
- Licencia: GPL; ¿Open Source?: Sí; ¿Gratuito?: Sí; Versión: 0.9.7

9 Cobian Backup

- Descripción: Permite realizar copias de seguridad
- Utilizado para: Realizar, de forma automática, copias de seguridad del proyecto
- Características: Programador de copias de seguridad. Múltiples idiomas, entre ellos el español. Modo de aplicación, servicio y línea de comandos. Capacidades de compresión y cifrado. Copias incrementales y diferenciales. Interfaz gráfica de usuario sencilla e intuitiva
- Página oficial: http://www.cobian.se/
- Licencia: Freeware; ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 7.4.5.335

10 Adobe Reader

- Descripción: Permite la visualización y la impresión de archivos PDF (Portable Document Format).
- Página oficial: http://www.adobe.com/es/products/acrobat/readermain.html
- Licencia: Freeware; ¿Open Source?: No; ¿Gratuito?: Sí; Versión: 7.0

18 Directrices de Desarrollo

A lo largo del desarrollo del proyecto se han considerado o seguido una serie de directrices relacionadas con el producto, la importancia del cliente, el trabajo que conlleva el desarrollo de un producto, la presentación del producto y su documentación.

1 Sobre el producto

- *La introducción, conclusiones y resultados de un producto son muy importantes*

En la introducción de un producto es donde se atrae al lector sobre la importancia de lo que se va a hacer.

En la presentación de las conclusiones y resultados es donde se transmite el mensaje de todo lo bueno que hay en el producto.

Una deficiente redacción de la introducción o una mala presentación de las conclusiones y resultados, pone en entredicho la calidad global del producto.

- *Todo producto debe poseer características importantes*

Todo producto, desarrollo y/o presentación debe poseer un conjunto de características, tales como: Buena estructuración, Claridad, Convergencia hacia un objetivo, Coherencia, Resaltar lo importante y Elegancia.

- *La sencillez es muy importante*

La homogeneidad es fácil de entender, manejar, asimilar, etc... Sin embargo, la heterogeneidad provoca todo lo contrario.

Las ciencias técnicas, y en particular la ciencia de la computación, se dan a conocer al mundo mediante términos muy técnicos y muy poco familiares para el gran público.

- *Todo Proyecto Fin de Carrera debe resaltar ciertos aspectos*

Existe un conjunto de aspectos que son interesantes a resaltar de un proyecto fin de carrera. Estos aspectos son: Sus resultados, Sus conclusiones, El interés del proyecto, La utilidad del proyecto en el ámbito social, La calidad del trabajo realizado, La dificultad del trabajo realizado, La justificación del tiempo empleado, Facilidad de utilización de los resultados del proyecto por terceras personas, Publicidad del proyecto a través de páginas web, etc... y Carácter integrador de conocimientos técnicos.

2 La importancia del cliente

- *Es esencial entender la verdadera necesidad del cliente*

Cuando hablamos de vender un producto, es esencial entender la verdadera necesidad o deseo del cliente, estableciendo prioridades.

Los principales objetivos de este principio son: Realizar una presentación efectiva, Diferenciar lo que es importante y lo que no lo es, Diferenciarse de la competencia, A mayor grado de confianza, menor grado de resistencia.

Está basado en la Ley de Pareto que dice: "Si se atiende al 20% de las principales necesidades o deseos de un cliente, éste quedará impactado positivamente en un 80%". Algunos ejemplos son:

- El cliente no desea conocer los detalles de la criptografía, simplemente desea crear objetos seguros.
- Es como aquel cliente que va a una ferretería y pide una broca. Él no necesita una broca, lo que realmente necesita es un agujero en la pared.

- *Es muy importante identificar a los clientes potenciales del producto*

Para ello es importante realizar las siguientes preguntas: ¿Quién es el cliente del producto?, ¿A quién va dirigido?

Un ejemplo podría ser: En el caso de una redacción, es importante ponerse en el lugar del potencial lector, preguntarse "¿El lector captará la idea?" y no presuponer nada.

3 Sobre el desarrollo de un producto

- *Descansar de forma inteligente es importante*

Es importante ya que así: Se adquiere perspectiva, Se aumenta la objetividad, Se aumenta el sentido crítico.

Un ejemplo sería: Redactar bien tiene su dificultad y no todos los días tenemos la inspiración adecuada, para esos días negros, que no nos viene nada a la cabeza, lo mejor es dedicarse a cosas más mecánicas que no requieren tanta concentración, como puede ser completar la bibliografía, ir haciendo un manual de usuario o un anexo técnico, etc...

4 Sobre la presentación de un producto

- *Los primeros minutos de una presentación son los más importantes*

Los primeros minutos de una presentación son los más importantes ya que su motivo es: "Ganarse al público si se hace bien, o perderlos para siempre en caso contrario".

- *Los últimos minutos de una presentación también son muy importantes*

Los últimos minutos de una presentación también son muy importantes con el objetivo de "Dejar un buen sabor de boca".

- *La presentación del producto es igual de importante que el propio producto*

La presentación de un producto es como mínimo igual de importante que el propio producto

- *La presentación oral debe satisfacer unos criterios concretos muy importantes*

La presentación oral de un producto debe satisfacer unos criterios concretos muy importantes: Transmitir correctamente la idea principal del producto, Exponer el producto de forma que resulte interesante, Ajustarse al guión escrito, Ajustarse al tiempo, Ritmo de presentación adecuado, Fluidez en la presentación y Evitar los silencios improcedentes.

- *El lenguaje utilizado en una presentación oral debe ser adecuado*

El lenguaje utilizado en una exposición oral debe satisfacer unos criterios concretos e indispensables que son: Lenguaje adecuado, ni demasiado formal ni coloquial, Evitar utilizar muletillas, Hablar alto y claro, Vocalizar correctamente y Modular correctamente el tono de voz para captar la atención.

- *La imagen del presentador debe ser adecuada*

La imagen del presentador en una exposición oral debe cumplir una serie de características deseables: Mostrar seguridad en lo que se dice, Transmitir entusiasmo, Gesticulación con las manos, Habilidades en la comunicación no oral y Mirar al público correctamente.

5 Sobre la documentación de un producto

- *La documentación debe ser entretenida*

Aprender leyendo está bien, pero está mucho mejor aprender divirtiéndose.

- *La cantidad documentación no es importante*

La cantidad de documentación no es importante, sólo su contenido lo es.

7.Preguntas frecuentes

Esta sección también es conocida como FAQs e incluye las preguntas más frecuentes que el lector de este proyecto puede hacerse. Pues bien, aquí se resolverán algunas de ellas.

Estas preguntas se han clasificado por categorías y sus respuestas se muestran en dos versiones, una corta y otra larga. Estas categorías son: "Sobre el proyecto en cuestión", "Sobre la documentación y su estructura", "Sobre los resultados utilizados", "Sobre el proyecto en sí", "Sobre las dificultades y errores", "Sobre la metodología y la temporización" y "¿Qué opina el mundo de este proyecto?".

Quizás, la pregunta más importante podría ser: "¿Por qué no se ha desarrollado un conjunto de aplicaciones?" Es decir, "¿porque no se ha cumplido uno de los dos objetivos de este proyecto?". Respondiendo de forma corta, se podría podría decir que: "Porque tras realizar el análisis surgió una idea mejor". La respuesta más larga sería: "La razón de existencia del segundo objetivo previsto «Mostrar aplicaciones de las técnicas criptográficas mediante software desarrollado en el proyecto» simplemente era el de probar que realmente se había alcanzado el primer objetivo: «Aprender a utilizar técnicas criptográficas». Tras finalizar el estudio preliminar, se consideró que era mucho mejor desarrollar algo nuevo, antes que realizar un conjunto de aplicaciones representativas de la criptografía, lo cual no es nada novedoso. Además, ya existen herramientas muy buenas como "CrypTool" (http://www.cryptool.com/), TrueCrypt (http://www.truecrypt.org/) y AxCrypt (http://axcrypt.sourceforge.net) de código abierto y otras freeware tales como "EncryptOnClick" y "FingerPrint" (http://www.2brightsparks.com/).".

También son interesantes las siguientes cuestiones que se responderán aquí de forma breve:

- ¿Cuál es el tamaño del proyecto? Su tamaño es bastante grande ya que está compuesto por 3142 clases, 315 campos, 2914 métodos, 73959 líneas de código y 22941 líneas de comentarios.
- ¿Cuál ha sido el coste del proyecto? Económico ninguno al utilizar herramientas gratuitas pero el coste de tiempo de estima entre 2000 y 3000 horas; más de un año de trabajo sin pausa.
- ¿Qué metodología se ha utilizado? Se ha utilizada la clásica (análisis, diseño, implementación y pruebas) con retroalimentaciones.

En las siguientes secciones podrán encontrar más preguntas frecuentes con respuestas cortas y largas.

Para mayor información puede consultar la web http://jcef.sourceforge.net.

1 Sobre el proyecto en cuestión

1 ¿Por qué no se ha desarrollado un conjunto de aplicaciones?

Respuesta corta: Porque tras realizar el análisis surgió una idea mejor.

Respuesta larga: La razón de existencia del segundo objetivo previsto "Mostrar aplicaciones de las técnicas criptográficas mediante software desarrollado en el proyecto" simplemente era el de probar que realmente se había alcanzado el primer objetivo: "Aprender a utilizar técnicas criptográficas".

Tras finalizar el estudio preliminar, se consideró que era mucho mejor desarrollar algo nuevo, antes que realizar un conjunto de aplicaciones representativas de la criptografía, lo cual no es nada novedoso. Además, ya existen herramientas muy buenas como "CrypTool" (http://www.cryptool.com/) y TrueCrypt (http://www.truecrypt.org/) de código abierto y otras freeware tales como "EncryptOnClick" y "FingerPrint" (http://www.2brightsparks.com/).

2 ¿Por qué se ha intentado evitar en la medida de lo posible usar términos técnicos?

Respuesta corta: Porque son totalmente innecesarios y perjudiciales.

Respuesta larga: Los términos técnicos sólo son entendibles por un grupo reducido de personas, no por el público en general. Además, esto ha permitido obtener la verdadera funcionalidad de la criptografía, asegurar objetos.

3 ¿Por qué no se ha entrado en detalle en el tema de la criptografía?

Respuesta corta: Porque no ha sido necesario.

Respuesta larga: La criptografía no sirve para nada sino se le saca provecho y este proyecto trata principalmente de sacarle el máximo partido a la criptografía, no en entrar en detalles.

4 ¿De dónde surge la idea del proyecto?

Respuesta corta: D. Jesús María Ramos Saky (Ingeniero Informático)

Respuesta larga: En especial surge del interés en desarrollar sistemas seguros, concretamente, sistemas de gestión de bases de datos orientados a objetos.

5 ¿Quién es el autor de la propuesta de Proyecto Fin de Carrera?

D. Jesús María Ramos Saky (Ingeniero Informático)

2 Sobre la documentación y su estructura

1 ¿Por qué no se ha seguido la estructura general de un proyecto fin de carrera?

Respuesta corta: Se consideró que la estructura tradicional es antigua y muy general.

Respuesta larga: En general, no está pensada para una fácil y rápida comprensión del proyecto.

2 ¿Todas las fuentes de información expuestas han sido utilizadas?

Respuesta corta: Sí, todas las que se han puesto es porque han sido utilizadas para la elaboración de la documentación.

Respuesta larga: Por otro lado, también es cierto que se han consultado muchas otras fuentes de información que finalmente no han sido utilizadas de una forma claramente identificable en la documentación, pero sí han servido para adquirir mayor conocimiento sobre los temas.

3 ¿Por qué las referencias a bibliografía no han sido colocadas en los capítulos?

Respuesta corta: Pues porque se consideran que donde están, siempre estarán más a mano y además no se interrumpe al lector con los accesos directos a las referencias.

Respuesta larga: Por lo tanto, se considera que no hay mejor sitio para colocar las referencias que en el capítulo de "Recursos utilizados".

4 ¿Por qué la documentación es escasa en cuestión de volumen?

Respuesta corta: No es escasa, es justa y suficiente.

Respuesta larga: El volumen de la documentación no es importante, lo importante es el contenido y que esté bien referenciado. También hay que decir que toda la documentación es totalmente original, salvo algunos pequeños fragmentos; es decir, nunca se ha copiado y pegado información tal cual venía de su fuente de información y también se ha colocado sólo lo importante. Si se quieren más detalles, para ello existen los recursos utilizados. Hubiera resultado muy fácil hacer que el volumen de la documentación fuera mayor, bastaría con hacer la letra más grande y copiar un montón de información de las fuentes consultadas y no por ello, el proyecto estaría mejor documentado.

5 ¿Por qué se ha generado la documentación de forma muy esquemática?

Respuesta corta: Es una apuesta personal. La documentación narrativa deja mucho que desear.

Respuesta larga: Lo único que se hubiera ganado documentando de forma narrativa, sería el ahorro de páginas, pero por el contrario, se dificultaría la comprensión de la información expuesta, ya que el lector tendría que construirse sus propios esquemas mentales.

6 ¿El enfoque de la criptografía orientada a objetos es entendible?

Respuesta corta: Sí.

Respuesta larga: Ha sido corroborado por varias personas con y sin conocimientos previos de criptografía.

3 Sobre los recursos utilizados

1 ¿Ha valido la pena utilizar recursos 100% Open Source o Freeware?

Respuesta corta: Sí.

Respuesta larga: Indudablemente, ya que se podría decir que desarrollar el proyecto no ha costado dinero, sólo tiempo.

2 ¿Por qué se decidió utilizar herramientas exclusivamente Open Source o Freeware?

Respuesta corta: Por principios morales, éticos, profesionales, legales y comerciales.

Respuesta larga: Debido a que utilizar software comercial pirata hacía que sintiera remordimientos. Pienso que el día de mañana a mí tampoco me gustaría que piratearan mi trabajo.

3 ¿Por qué se ha utilizado Windows en lugar de un sistema operativo Open Source?

Respuesta corta: Por problemas de compatibilidad de hardware.

Respuesta larga: Durante mucho tiempo se intentó utilizar sistemas operativos Linux de código abierto y gratuitos, pero por problemas de compatibilidad de hardware tuve que usar Windows.

4 Sobre el proyecto en sí

1 El diseño es sumamente simple, ¿esto es bueno o malo?

Respuesta corta: Totalmente bueno.

Respuesta larga: Llegar a este diseño no ha sido fruto de la casualidad ni resultado de un día. Para llegar hasta aquí se han pasado por un montón de diseños previos.

2 ¿Qué tamaño y coste tiene el proyecto?

Respuesta corta: Su tamaño es bastante grande ya que está compuesto por 3142 clases, 315 campos, 2914 métodos, 73959 líneas de código y 22941 líneas de comentarios.

Respuesta larga: Para hacerse una mejor idea, se puede comparar el proyecto con otros ya existentes escritos en Java tales como "Dr. Java" (un entorno de desarrollo para Java, http://drjava.sourceforge.net), "Gantt Project" (herramienta de gestión, http://ganttproject.sourceforge.net), "HTML Unit" (Pruebas para aplicaciones web, http://htmlunit.sourceforge.net), "iReport" (Diseñador visual de informes, http://ireport.sourceforge.net), "iText" (Generador de ficheros PDF, http://www.lowagie.com/iText/), "JAxe" (Editor XML, http://jaxe.sourceforge.net) y "JEdit" (Editor de texto para programadores, http://www.jedit.org).

Proyecto	Clases	Campos	Métodos	Líneas de Código	Líneas de Comentarios
Dr. Java	355	680	1739	50358	24563
Gantt Project	303	1241	1492	31947	26433
HTML Unit	274	449	2271	56463	24743
iReport	298	2475	3516	77760	14805
iText	415	4180	4216	124405	47511
JAxe	135	497	887	23315	3385
JEdit	765	3263	4789	139900	49867
Aplicaciones Criptográficas Java	**3142**	**315**	**2914**	**73959**	**22941**

Tabla 80: Tamaños de este proyecto y otros

Este proyecto contiene 3142 clases, 315 campos, 2914 métodos, 73959 líneas de código y 22941 líneas de comentarios. Dentro de estos datos ya se encuentran incluidos las métricas para las pruebas, cuyas dimensiones concretas son 1724 clases, 73 campos, 958 métodos, 30936 líneas de código y 8119 líneas de comentarios. Además, también se incluyen las métricas para el manual de usuario de JCEF, cuyos datos son 11 clases, 3 campos, 412 métodos, 4572 líneas de código y 5740 líneas de comentarios.

Otra métrica para imaginarse el coste de este proyecto es el tiempo de desarrollo y realización del mismo; el cual se estima entre un mínimo de 2000 horas de trabajo y un máximo de 3000 horas. Este proyecto ha tenido pocas pausas durante su desarrollo, el cual comenzó a finales de febrero de 2005 y terminó a finales de mayo de 2006, es decir, más de un año de desarrollo.

3 ¿Por qué no hay registro de los anteriores diseños?

Respuesta corta: Porque siempre fueron considerados pasos previos hacia el diseño final.

Respuesta larga: Todo diseño anterior es en todos los sentidos inferior al diseño final, por lo que no hay características positivas de diseños anteriores que hicieran razonable la supervivencia de dicho diseño aunque sólo fuera documentado.

4 ¿Se implementan algoritmos criptográficos?

Respuesta corta: No.

Respuesta larga: Se podría decir que no, aunque en realidad implementa algún que otro algoritmo criptográfico para completar los proveedores criptográficos ya existentes. Pero en general, simplemente facilita enormemente el uso de los mismos abstrayendo un montón de detalles.

5 Sobre las dificultades y errores

1 ¿Qué es lo que más ha costado?

Respuesta corta: No abandonar de manera casi permanente el proyecto.

Respuesta larga:

- "Vivir" con las ideas: ¿le gustará al tribunal?, ¿se habrán tomado las decisiones correctas?.
- Mantener el ritmo.
- La realización de las tareas no creativas.
- La búsqueda de una idea nueva.
- El nuevo enfoque de la Criptografía Orientada a Objetos.
- Rediseñar y rediseñar ciertos aspectos.
- Utilizar algunos recursos resultaba frustrante. Por ejemplo, el OpenOffice se colgaba y perdía información, y el ArgoUML no era nada amigable de usar.
- Trabajar en el proyecto durante más de un año.

6 Sobre la metodología y la temporización

1 ¿Cuál ha sido la metodología empleada en el desarrollo del proyecto?

Respuesta corta: La clásica.

Respuesta larga: La basada en el ciclo de vida clásico del software: Análisis, Diseño, Implementación y Pruebas con retroalimentaciones.

2 ¿Se han cumplido los tiempos previstos?

Respuesta corta: No.

Respuesta larga: El tiempo real de dedicación al proyecto está muy alejado del estimado inicialmente, y es que hay que tener en cuenta que es la primera vez que estimo tiempos.

3 ¿Cuál ha sido la temporización de cada una de las etapas del proyecto?

Etapas	**Estimación**	**Real**	**Diferencia**
Estudio Preliminar	80 horas	450 horas	370 horas
Desarrollo	140 horas	1400 horas	1260 horas
Presentación	80 horas	550 horas	470 horas
Total	300 horas	2500 horas	2200 horas

Tabla 81: Temporización del proyecto

4 ¿Por qué no se han estimado los tiempos con mayor detalle?

Respuesta corta: No se consideró útil.

Respuesta larga: Desde un principio se sabía que las estimaciones de tiempo iban a resultar totalmente una pérdida de tiempo, ya que se carece de experiencia al respecto. Predecir la cantidad de tiempo que se tardará en realizar una tarea es demasiado complejo y requiere de suficientes experiencias anteriores para obtener una estimación cercana a los tiempos reales.

5 ¿El tiempo empleado es razonable para lo conseguido?

Respuesta corta: Depende de con quien y/o qué se compare.

Respuesta larga: Se es consciente de que un ingeniero más experimentado en el desarrollo de proyectos hubiera desarrollado este mismo trabajo en mucho menos tiempo.

6 ¿Vale la pena realizar el control de tiempos?

Respuesta corta: Depende.

Respuesta larga: Sólo si se va a sacar provecho de él en un futuro próximo.

7 ¿Qué opina el mundo de este proyecto?

"El contenido está bien, no veo fallos, está bastante completo, cualquiera que sepa algo de criptografía no tendría problemas en la lectura del contenido y para alguien que no tenga ni idea de criptografía creo que podría seguir sin problemas la lectura, pero para mi gusto, y encima siendo la memoria de un proyecto, lo veo muy esquemático, muchas tablas, lenguaje muy poco formal, etc... lo cual, por lo menos a mí, me molesta un poco al leerlo, pero por lo demás, está correcto o al menos no veo fallos a simple vista".

"En general me pareció que era bastante entendible, el formato es sencillo, claro y poco tedioso".

8.Índice de tablas

Tablas

9.Índice de ilustraciones

Ilustraciones

Aplicaciones Criptográficas Java

Mayo 2006

Jesús María Ramos Saky

Tutor:

Miguel Ángel Pérez Aguiar

Facultad de Informática

Universidad de las Palmas de Gran Canaria

Proyecto Fin de Carrera

Aplicaciones Criptográficas Java

A grandes rasgos, el uso de la criptografía ayuda a evitar el uso fraudulento de sistemas, a proteger información confidencial o importante, permitir comunicaciones seguras y posibilitar el comercio electrónico.

Siendo más precisos, la criptografía permite asegurar un objeto convirtiéndolo en otro objeto incomprensible y/o autentificable y posteriormente obtener el objeto asegurado a partir de su versión segura.

El objetivo primordial de este proyecto es aprender a utilizar los principales mecanismos criptográficos: protección y autentificación.

El resultado ha sido un conjunto de librerías Java sobre criptografía, destacando entre ellas JCEF (Java Cryptographic Extension Framework). Esta librería es potente y sobre todo de fácil uso que suplanta a las librerías ya existentes JCA y JCE proporcionadas por Sun Microsystems al ser éstas muy complicadas de utilizar.

Como un ejemplo del valor añadido de este proyecto, observe el código siguiente donde se muestra cómo se asegura un objeto e inmediatamente después se recupera el mismo; y todo ello de una forma super sencilla:

```
Object object = new String("my object");
CryptoAlgorithm secureAlgorithm = new AES_BlockSymmetricProtectionRREXKY();
SecureObject secureObject = new SecureObject(object, secureAlgorithm);
object = (String)secureObject.getObject(secureAlgorithm);
```

Este proyecto se encuentra hospedado en el mayor repositorio de proyectos software de código abierto existente llamado SourceForge.net. Para mayor información puede consultar la dirección web http://jcef.sourceforge.net.

Jesús María Ramos Saky

www.ulpgc.es

Miguel Ángel Pérez Aguiar