Manual de usuario de la File Encoder Application v1.5

# Índice

[0. Índice 1](#_Toc62155240)

[1. Introducción 2](#_Toc62155241)

[2. Inicio rápido 5](#_Toc62155242)

[2.1. Requisitos 5](#_Toc62155243)

[2.2. Inicio rápido 5](#_Toc62155244)

[3. Interfaz de usuario 5](#_Toc62155245)

[3.1. Pantalla principal 5](#_Toc62155246)

[3.2. Menú Archivo 8](#_Toc62155247)

[3.3. Menú Herramientas 8](#_Toc62155248)

[3.4. Menú Vista 9](#_Toc62155249)

[3.5. Menú de Ayuda 9](#_Toc62155250)

[3.6. Lista de configuraciones de encriptación 10](#_Toc62155251)

[3.6.1. Conceptos básicos 10](#_Toc62155252)

[3.6.2. El formulario 10](#_Toc62155253)

[3.7. Configuración de encriptación 12](#_Toc62155254)

[3.7.1. Parámetros de la configuración de encriptación 13](#_Toc62155255)

[3.8. Configuración de la aplicación 17](#_Toc62155256)

[3.8.1. Parámetros configuración de encriptación de la aplicación 17](#_Toc62155257)

[3.8.2. Parámetros configuración de idiomas de la aplicación 19](#_Toc62155258)

[3.8.3. Parámetros configuración de vista de la aplicación 20](#_Toc62155259)

[3.9. Borrado de ficheros 21](#_Toc62155260)

[3.10. Acerca de 22](#_Toc62155261)

[3.11. Licencia 23](#_Toc62155262)

[4. Encriptación por la línea de comandos 24](#_Toc62155263)

[4.1. Scripts para Windows 25](#_Toc62155264)

[4.2. Scripts para Linux y para Mac 25](#_Toc62155265)

[4.3. Parámetros de encriptación como argumentos 26](#_Toc62155266)

[5. Método de encriptación 28](#_Toc62155267)

[5.1. Cabecera del fichero encriptado 28](#_Toc62155268)

[5.2. Algoritmo de encriptación 29](#_Toc62155269)

[5.3. Generador pseudoaleatorio utilizado 30](#_Toc62155270)

[5.4. Utilización de hilos de ejecución (paralelismo) 33](#_Toc62155271)

[6. Tipos de encriptación 35](#_Toc62155272)

[6.1. FileEncoderType-1 35](#_Toc62155273)

[6.2. FileEncoderType-2 36](#_Toc62155274)

[6.3. FileEncoderType-3 37](#_Toc62155275)

[6.4. FileEncoderType-4 38](#_Toc62155276)

[6.5. FileEncoderType-5 38](#_Toc62155277)

[6.6. FileEncoderType-6 39](#_Toc62155278)

[6.7. FileEncoderType-7 40](#_Toc62155279)

[6.8. FileEncoderType-8 41](#_Toc62155280)

# Introducción

La File Encoder Application es una aplicación que permite encriptar ficheros.

Básicamente permite:

* Encriptar fichero.
* Desencriptar fichero (encriptado con esta misma aplicación).
* Abrir fichero encriptado (desencriptándolo previamente)

La aplicación permite operar con ficheros encriptados (con extensión .jfe) y desencriptados con su interfaz gráfica.

Así mismo el motor de la aplicación puede ser invocado a través de la interfaz de comandos obteniendo de esta forma la misma funcionalidad en cuando a encriptación/desencriptación que la aplicación gráfica.

La aplicación está escrita en Java, por lo que es portable a distintas plataformas y debería ser compatible. Es decir, debería ser posible encriptar un fichero en una plataforma y desencriptarlo en otra distinta.

Ha sido probado sólo con procesadores Intel, en Windows 8, Linux y OS-X pero también debería ser compatible con cualquier plataforma que tenga instalada la máquina virtual de Java (JRE).

Existen varios parámetros de configuración de la encriptación que modificados convenientemente pueden permitir:

* incrementar la rapidez de los procesos de encriptación/desencriptación.
* incrementar la robustez de la encriptación.
* minimizar el uso de memoria.

Desgraciadamente no pueden optimizarse todos los puntos a la vez. Así pues, si se quiere incrementar la robustez de la encriptación a través de los valores de los parámetros de encriptación, eso va en contra de incrementar la rapidez o en contra de minimizar el uso de la memoria.

Si no tienes ganas de aprender los parámetros, puedes dejar la configuración básica que incluye configuración de parámetros basados en el tamaño del fichero, para permitir que los ficheros mayores sean encriptados con una configuración más rápida que los pequeños.

Independientemente de los parámetros utilizados para encriptar, una buena contraseña ayuda a que la robustez de la encriptación sea mayor.

Por ello se aconseja utilizar una contraseña muy larga (puede ser una frase), para que un intento de romper la encriptación por fuerza bruta se haga más complicado.

Por ejemplo, una buena contraseña debería tener mayúsculas, minúsculas, números y no estar compuesta únicamente por palabras de diccionario. Si a ello añadimos que la contraseña puede tener una longitud larga, por ejemplo entre 30 y 40 caracteres, el sistema de encriptado se hace más fuerte.

**Novedades en la versión 1.2**

Esta nueva versión de la aplicación ha surgido debido a un error encontrado en el fileEncoderType 2, que ha aparecido al hacer pruebas de encriptación con archivos grandes.

Se ha creado el fileEncoderType 4 para substituir al fileEncoderFile 2 que corrige el error encontrado, y además aumenta el rango de los números hasta 256.

Además se han creado dos fileEncoderTypes adaptados a ficheros grandes y muy grandes.

Los fileEncoderType 5 y 6 son mucho más robustos que sus predecesores para ficheros grandes.

El fileEncoderType 6 incluye una simplificación que lo hace un poco menos robusto que el 5 para archivos grandes, pero a cambio aumenta la velocidad de encriptación en un factor de varias unidades.

Este nuevo tipo (6) no es aconsejado para encriptar archivos pequeños o medianos (menores de 10 MB).

Se han realizado pruebas de encriptación de archivos grandes (mayores de 50 GB) con éxito.

**Novedades en la versión 1.3**

En esta nueva versión de la aplicación, se han creado dos nuevos tipos de encriptación que permiten utilizar múltiples hilos de ejecución (paralelismo), ideal para utilizarlo con archivos grandes.

* fileEncoderType 7.

Este tipo de encriptación tiene dos formas de funcionamiento:

* + Fichero de entrada y fichero de salida en el mismo disco:
		- Se utiliza un hillo separado para el acceso a disco (el mismo hilo para lectura y escritura).
		- Se utilizan varios hilos para la encriptación.
	+ Fichero de entrada y fichero de salida en discos separados:
		- Se utiliza un hillo separado para la lectura del fichero de entrada.
		- Se utiliza otro hilo separado para la escritura del fichero de salida.
		- Se utilizan varios hilos para la encriptación.

Dependiendo de la capacidad de procesamiento de la máquina y de la rapidez de los discos de entrada y de salida, la mejora en la rapidez de encriptación variará entre equipos.

En las pruebas que se han realizado (utilizando el número de hilos óptimo y siendo el único usuario de todos los recursos de mi máquina) se ha llegado a dividir el tiempo total de encriptación en un factor 2 para archivos muy grandes.

Si además hacemos que el fichero de entrada y el fichero de salida estén en discos distintos, la mejora en velocidad se hace más tangible, llegando a factores de mejora de tiempo algo mayores de 3.

Para más detalles puedes ir a los apartados que explican el nuevo tipo de encriptación.

***6.7-FileEncoderType-7***

* + Comparativa de tiempos de encriptación/desencriptación de archivos grandes con el nuevo tipo de encriptación (puede variar en distintos sistemas):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tamaño | FileEncoderType | -numOfThreads | Discos distintos | Duración de la encriptación (s) | Duración de la desencriptación (s) |
| 27,2 GB | 6 | (1) | no | 1631 | 1512 |
| 27,2 GB | 7 (nuevo) | 3 | no | 704 | 886 |
| 27,2 GB | 7 (nuevo) | 6 | sí | 455 | 669 |

* fileEncoderType 8.

Este tipo de encriptación es equivalente al 7, pero permitiendo dividir el archivo encriptado en volúmenes de un determinado tamaño.

Para más información revisar el punto: ***6.8-FileEncoderType-8***

**Novedades en la versión 1.4**

Se ha corregido un efecto negativo que se producía con máquinas con un dpi alto (alta densidad de pixels por centímetro) a partir de Java-9.

A partir de Java-9, se introduce un tratamiento automático teniendo en cuenta la densidad de pixels por centímetro de la pantalla del dispositivo.

Este tratamiento automático interfería de forma negativa con la funcionalidad de ampliación incorporada en mis aplicaciones.

Para evitar este efecto no deseado, se ha desactivado la nueva funcionalidad de Java-9.

**Novedades en la versión 1.5**

Se hace uso de la versión de plataforma v1.4, que añade la consulta de nueva versión durante el arranque de la aplicación.

# Inicio rápido

## Requisitos

Para ejecutar la aplicación es necesario tener el entorno de java instalado (Java Runtime Environment), en la versión 8 o superior.

De esa manera, se podrá abrir el fichero .jar existente en la carpeta ./binary/

## Inicio rápido

Antes de empezar, indicar que la aplicación utiliza el micrófono para recoger bytes aleatorios, por lo que sería necesario tener un micrófono conectado para que la aplicación funcione de forma óptima..

En esta versión de la aplicación se ha mejorado esta parte y ahora si el micrófono está desconectado, se generan números aleatorios utilizando los generadores de semillas que provee el propio Java.

Para comenzar a utilizar la aplicación, sigue los siguientes pasos:

1. Abre la File Encoder Application, haciendo doble click en: el .jar de la aplicación: que está en .../\_binary/jfe.v1.3.jar
2. Abre una ventana nueva de fichero (Menú: Archivo->Nuevo).
3. Fichero. Introduce el fichero que deseas encriptar (cualquier nombre) o desencriptar (fichero con extensión .jfe). Puedes pulsar el botón "..." para abrir el diálogo de selección de fichero.

Ten en cuenta que para encriptar o desencriptar un fichero, no debe existir el archivo encriptado o desencriptado. Si existe este archivo, la operación fallará, ya que la aplicación no va a sobreescribir el archivo.

1. Contraseña. Introduce la contraseña que va a tener el fichero encriptado, o que tiene el fichero a desencriptar.
2. Repetir contraseña. Vuelve a introducir la contraseña. Ambas contraseñas deben coincidir para poder encriptar/desencriptar.

Tras estos pasos, ya puedes utilizar los botones de Encriptar, Desencriptar o abrir fichero encriptado.

# Interfaz de usuario

En este capítulo se verá como encriptar, desencriptar, abrir ficheros encriptados y borrar ficheros existentes y también se verán las opciones más comunes a la hora de ejecutar estas acciones.

## Pantalla principal

Al arrancar la aplicación gráfica, aparece una pantalla como la siguiente:



En ella puede observarse que hay un menú principal y un área grande principal donde estarán las ventanas MDI de la aplicación.

En el menú Archivo aparece una opción (Nuevo).

Seleccionando esa opción aparecerá una nueva ventana de fichero.

Se pueden abrir tantas ventanas de fichero como sea necesario.

El aspecto de la ventana de fichero es este:



En ella hay varios parámetros que deben rellenarse antes de acceder a la funcionalidad.

* "Fichero desencriptado". En este cuadro de texto debe aparecer el nombre del fichero desencriptado.
* "Fichero encriptado". En este cuadro de texto debe aparecer el nombre del fichero encriptado (con extensión .jfe).

Para rellenar los dos cuadros de texto anteriores hay dos opciones:

* + Pulsando el botón de su derecha "...". Cuando se pulse este botón, se abrirá un diálogo para navegar por los discos del equipo y seleccionar el fichero.
	+ editándolo con el teclado (esto puede ser útil cuando ya tienes en el cuadro de texto el nombre del fichero y quieres editar el nombre de fichero).
* "Contraseña" y "Repetir Contraseña". En estos dos cuadros de texto debes escribir la contraseña que se va a aplicar al encriptar o desencriptar el fichero.

Los dos cuadros de texto deben contener la misma contraseña. Esto es así para que, en caso de haber escrito mal la contraseña (sobre todo al encriptar), puedas darte cuenta. De otra forma se encriptaría con una contraseña desconocida y luego no podría desencriptarse el fichero.

Obviamente la contraseña utilizada para encriptar y para desencriptar debe ser la misma para que se realice el proceso de desencriptación correctamente.

* El checkbox de “Entrada y salida se encuentran en discos diferentes”, indica si se utilizará el parámetro de encriptación “-encDecOnDifferentDrives” que indica que el fichero encriptado y el desencriptado se encuentran en discos diferentes, y eso hace que para los fileEncoderType 7 y 8, se utilicen hilos separados para la lectura y escritura de disco.
* El checkbox “auto”, indica si el valor del checkbox anterior se calcula automáticamente (la aplicación considera que los ficheros están en discos diferentes si se encuentran el particiones diferentes, criterio que no siempre es válido).

Una vez rellenos los parámetros anteriores, ya puede utilizarse la funcionalidad.

Con la nueva ventana de fichero se puede acceder a las funcionalidades pulsando los siguientes botones:

* "Encriptar". Encripta el fichero
* "Desencriptar". Desencripta el fichero
* "Abrir un fichero encriptado". Desencripta y abre el fichero con la aplicación predefinida en el sistema para ese tipo de ficheros.
* Modificar la "Configuración de encriptación". Sirve para modificar los parámetros de encriptación. Ver punto: ***3.7-Configuración de encriptación***
* "Borrar fichero". Al lado de los componentes de texto que contienen los nombres de fichero encriptados y desencriptados, existen unos botones con una cruz, que permiten borrar el fichero del cuadro de texto de su izquierda. Ver punto: ***3.9-Borrado de ficheros***
* "Limpiar historia". Borra la historia del cuadro de texto Historia.
* "Cancelar". Cancela la encriptación, desencriptación o borrado de fichero.

Hay un cuadro de texto (Historia), en el que aparece el estado y resultado de las acciones que vamos tomando sobre los ficheros.

También existe una barra que indica el progreso de la acción en curso y cuando finaliza la encriptación/desencriptación de un fichero, existe una etiqueta a la derecha de la barra de progreso que indica el tiempo transcurrido en milisegundos desde el inicio de la acción.

## Menú Archivo

El menú Archivo tiene este aspecto:



Este menú tiene dos opciones:

* **Nuevo**. Al pulsar sobre este elemento del menú crearemos una nueva ventana de fichero, vista en el punto anterior.
* **Salir**. Permite salir de la aplicación.

## Menú Herramientas

El menú Herramientas tiene este aspecto:



Este menú tiene tres opciones:

* **Idioma**. Al abrir este submenú aparecen los idiomas disponibles. Sirve para poder cambiar el idioma de la aplicación.

Por defecto hay disponibles los siguientes idiomas:

* "EN". Inglés
* "ES". Castellano
* "CAT". Catalán
* **Configuraciones de encriptación**. Cuando se pulsa este elemento de menú, aparece un formulario en el que se permite modificar las configuraciones de encriptación utilizadas según los tamaños de fichero a encriptar. Ver punto: ***3.6-Lista de configuraciones de encriptación***
* **Configuración de la aplicación**. Cuando se pulsa este elemento de menú, aparece un formulario en el que se permite modificar los parámetros de configuración generales de la aplicación. Ver punto: ***3.8-Configuración de la aplicación***

## Menú Vista

El menú Vista tiene este aspecto:



Este menú sólo tiene una opción:

* **Zoom**. Permite escoger el tamaño de la parte visual de la aplicación, con un porcentaje que indica el tamaño de los componentes en comparación con la visualización normal de las ventanas (100%).

## Menú de Ayuda

El menú de ayuda tiene este aspecto:



Este menú sólo tiene dos opciones:

* **Ayuda**. Abre este archivo de ayuda.
* **Buscar nueva versión**. Esta opción realiza una consulta al servidor, para ver si hay disponible una nueva versión de la aplicación.
* **Qué es nuevo**. Esta opción muestra el documento que explica lo que es nuevo en esta versión.
* **Licencia**. Muestra la licencia que se aceptó la primera vez que se ejecutó la aplicación. Ver apartado: ***3.11-Licencia***
* **Acerca de**. Cuando se pulsa este elemento de menú, aparece un formulario con los datos de la aplicación, los agradecimientos y los datos de contacto. Ver apartado: ***3.10-Acerca de***

## Lista de configuraciones de encriptación

Este formulario permite crear, modificar y borrar configuraciones de encriptación o revertir a la lista de configuraciones de encriptación por defecto.

### Conceptos básicos

Como se comentó en la introducción, existen varios parámetros para la encriptación que inciden sobre la rapidez, el uso de memoria y la robustez de la encriptación. Ver punto: ***3.7.1-Parámetros de la configuración de encriptación***

Dado que los parámetros de encriptación inciden sobre todo en la rapidez de la encriptación/desencriptación parece lógico pensar que en función del tamaño del fichero debería utilizarse una configuración de encriptación u otra.

Para facilitar las cosas al usuario final, la File Encoder Application permite tener configuraciones de encriptación predefinidas para los distintos rangos de tamaño de fichero, pudiendo escoger automáticamente una configuración de encriptación en función del tamaño del fichero a encriptar.

Este formulario permite mantener esas configuraciones de encriptación definidas para cada rango de tamaño de fichero.

### El formulario

El formulario tiene el siguiente aspecto:



En él existe una tabla resumen de las configuraciones de encriptación ordenadas de menor a mayor tamaño de fichero, de la que sólo puede seleccionarse una fila a la vez.

Cada fila se corresponde con una configuración de encriptación, asociada a un rango de tamaños de fichero.

La configuración de encriptación aplicaría al rango de tamaños de fichero desde el campo SIZE\_FROM de la propia configuración de encriptación en octetos, hasta el campo SIZE\_FROM de la siguiente configuración de encriptación (la siguiente fila), o bien ilimitado si no existe ninguna fila posterior.

También existen una serie de botones que aportan la funcionalidad del formulario:

* Botón revertir. Restablecer configuraciones de encriptación por defecto. Revierte todos los cambios que se hayan realizado sobre las configuraciones de encriptación por rango de tamaño de fichero, dejando la lista de configuraciones de encriptación original (la que venía inicialmente en la aplicación).
* Botón aceptar (aplicar). Salir salvando configuraciones. Permite salir del formulario salvando los cambios realizados en las configuraciones de encriptación.
* Botón cancelar. Salir sin salvar configuraciones. Permite salir del formulario sin salvar ni aplicar los cambios realizados en las configuraciones de encriptación por rango de tamaño de fichero.

Además, cuando se pulsa el botón derecho del ratón dentro de la tabla de configuraciones, aparece este menú emergente que permite realizar operaciones sobre los elementos de esa tabla:



* Modificar configuración seleccionada. Abre el formulario que permite modificar los parámetros de encriptación de la configuración de encriptación seleccionada de la tabla. Ver punto: ***3.7-Configuración de encriptación***
* Modificar nueva configuración. Abre el formulario que permite crear una configuración de encriptación desde cero, teniendo que incluir el parámetro SIZE\_FROM que permitirá definir el rango de tamaños de fichero para los cuales aplica esa configuración.
* Borrar configuración. Borra la configuración de encriptación seleccionada de la tabla.

## Configuración de encriptación

Este formulario permite modificar los parámetros de encriptación para una configuración de encriptación determinada, ya sea para aplicarla directamente sobre un fichero a encriptar (desde un formulario de fichero) o bien para modificar los parámetros de configuración de encriptación de un rango de tamaño de fichero determinado.

El formulario tiene el siguiente aspecto:



A este formulario puede llegarse o bien desde la ventana de fichero, vista en el punto: ***3.1-Pantalla principal*** o bien desde el formulario de lista de configuraciones de encriptación en el punto: ***3.6.2-El formulario***.

Existe un botón para cargar los parámetros por defecto en función del tamaño del fichero. Este botón sólo está habilitado en caso de que el acceso al formulario haya sido a través de la ventana de fichero. En caso de haber accedido a este formulario a través del formulario de lista de configuraciones de encriptación, este botón estará deshabilitado.

Este botón sólo ofrece funcionalidad en caso de que esté habilitado y además tiene que existir el fichero configurado en la ventana de fichero. En otro caso al pulsarlo la aplicación no hará nada.

### Parámetros de la configuración de encriptación

Se pueden configurar los siguientes parámetros:

* Tamaño de fichero (desde). Este valor no es un parámetro de configuración propiamente dicho y sólo está habilitado en caso de que el acceso al formulario haya sido a través del formulario de lista de configuraciones de encriptación.

Esto es así, porque en ese caso se pretende crear o editar una configuración de encriptación que aplicará a un rango de tamaños de fichero. El tamaño de fichero (desde) indicará el inicio del rango de tamaño de fichero en bytes. El final del rango al que aplicará la configuración de este formulario, dependerá del parámetro tamaño de fichero (desde) de la siguiente configuración de encriptación en la lista de configuraciones de encriptación.

En caso de que se haya accedido a este formulario desde la ventana de fichero, la configuración de encriptación que se está editando, en caso de ser aceptada, podrá ser aplicada directamente en los parámetros de la próxima encriptación, dependiendo de la configuración de prioridades de configuración de encriptación explicada en el punto: ***3.8-Configuración de la aplicación***

* FileEncoderType. Este parámetro indica qué clase de FileEncoder se va a utilizar para encriptar. Es un valor numérico que, para la v1.0, puede tomar los valores 1 o 2.
* 1 (NO RECOMENDADO). Clase de FileEncoder que utiliza un generador pseudoaleatorio caótico basado en la case BigDecimal de Java para hacer los cálculos.

Como para hacer las operaciones de encriptación se utiliza esta clase a bajo nivel, es posible que algún cambio de implementación en esta clase en alguna actualización de Java, pudiera romper la compatibilidad hacia atrás, por lo que se decidió crear una clase propia numérica larga menos dependiente de Java.

* 2 (OBSOLETO). Este FileEncoderType contiene un bug que hace que no sea aconsejable su uso. Indica que se utilice la clase de FileEncoder que utiliza un generador pseudoaleatorio caótico basado en una clase numérica propia para hacer los cálculos.

Esta clase trabaja con números de precisión variable y no es genérica. Está muy enfocada al rango numérico alcanzado con el generdor pseudoaleatorio caótico.

Contiene la información del número en un array de long de los que por razones de optimización sólo se utilizan los 31 bits inferiores de cada elemento del array.

Cuando se hable de "bloques" en los siguientes parámetros de encriptación aplicables a este FileEncoderType, nos referimos a un elemento del array de long, es decir un long (31 bits).

El encriptador de este tipo, utiliza encriptación XOR combinada con reordenación pseudoaleatoria de octetos en cada una de las rodajas en las que se divide el archivo a encriptar.

* 3 (OBSOLETO). FileEncoderType idéntico al anterior, pero con la particularidad de que el tamaño de los números aumenta su rango. Con este FileEncoderType, se puede trabajar con tamaños de hasta 256.

Al igual que el FileEncoderType 2, tiene un bug que hace que no sea aconsejable su uso.

* 4. Este fileEncoderType, es idéntico al anterior, con la diferencia de que el bug ha sido corregido.
* 5. Este fileEncoderType es idéntico al anterior, pero, además de la encriptación XOR combinada con reordenación de octetos (como todos los anteriores), añade la particularidad de que las rodajas en las que se divide el archivo a encriptar, son reordenadas. Esto lo hace mucho más fuerte frente a intentos de romper la encriptación por fuerza bruta.

Este tipo de encriptación es particularmente fuerte para archivos medianamente grandes.

* 6. (NO RECOMENDADO) Este fileEncoderType es idéntico al anterior, pero para acelerar la velocidad de encriptación/desencriptación, se ha eliminado la reordenación de bytes en cada una de las rodajas. Es mucho más rápido que el fileEncoderType 5, y para archivos grandes, debería ser bastante difícil de romper (mucho más que el fileEncoderType 4).

No se aconseja utilizar en archivos pequeños y medianos (menores de 10 MB).

Se aconseja utilizar el fileEncoderType 7 en lugar de este, ya que puede ser notablemente más rápido.

* 7. Este fileEncoderType es muy parecido al anterior, pero añadiendo la posibilidad de utilizar múltiples hilos de ejecución.

Especialmente aconsejado para archivos grandes (más de 100 MB), donde puede apreciarse claramente como baja el tiempo de ejecución al utilizar procesamiento en paralelo.

* 8. Este fileEncoderType es idéntico al anterior, pero añadiendo la posibilidad de escribir el fichero encriptado en múltiples volúmenes, siendo configurable el tamaño máximo de cada volumen.
* NumBytesFileSlice. Este parámetro de configuración de encriptación indica el número de bytes de rodaja que se utilizarán para encriptar.

Para ficheros grandes no es viable cargar todo el fichero en memoria y hacer los cálculos para la encriptación, ya que para hacer esos cálculos se utiliza una memoria entre 100 y 200 veces mayor que la del tamaño en bytes que se está encriptando, por lo que para, por ejemplo, un fichero de 50 MiB, tendríamos que la cota superior de la memoria utilizada serían 10 GiB, cantidad de memoria que no está disponible en muchas máquinas actuales.

Este parámetro hace que el contenido del fichero se divida en rodajas, y para cada una de esas rodajas, primero se lea el contenido de la rodaja en un array de bytes, se aplique la encriptación, y se escriba en el fichero encriptado el contenido de la rodaja encriptada. Para encriptar la siguiente rodaja ya no será necesario tener en memoria las rodajas anteriores, por lo que no sería necesario utilizar tanta memoria como si se encriptara de una vez el fichero completo.

Entonces bajar este parámetro hace que baje el consumo de memoria de la aplicación.

Sin embargo no sería bueno bajar mucho este parámetro, ya que la encriptación completa del fichero puede romperse por fuerza bruta si se rompe la encriptación de la primera rodaja, por lo que para poner las cosas más costosas al que quiera romper esta encriptación, este parámetro debería estar lo más alto posible en función de la memoria de que dispongamos.

Habrá que tener en cuenta que si ponemos este parámetro muy alto, el archivo podría no poder ser desencriptado en una máquina con menos memoria.

Si a este parámetro se le da el valor 0, hace que al encriptar el archivo no se divida en rodajas (utilizando una única rodaja que incluye todos los bytes del archivo).

* SizeOfNumbersSimpleEncoder. Este parámetro de configuración de encriptación aplica al tamaño numérico aplicado en el generador pseudoaleatorio caótico.

Ver punto: ***5.2-Algoritmo de encriptación***

El parámetro SizeOfNumbersSimpleEncoder hace referencia al tamaño de los números en el generador pseudoaleatorio utilizado para el primer paso de la encriptación (XOR).

Las unidades de este parámetro es ligeramente diferente para los FileEncoderType 1 y los demás.

* Para el FileEncoderType 1, este parámetro está expresado en bytes.
* Para el FileEncoderType 2, 3, 4, 5, 6, 7 y 8 este parámetro está expresado en bloques de 31 bits. En este caso sería igual al número de elementos long que hay en el array que tiene los datos del número. (En cada elemento long del array se almacenan 31 bits).
* SizeOfNumbersReordererEncoder. Este parámetro de configuración de encriptación aplica al tamaño numérico aplicado en el generador pseudoaleatorio caótico del segundo paso de la encriptación (reordenación).

Las unidades de este parámetro es ligeramente diferente para los FileEncoderType 1 y los demás.

* Para el FileEncoderType 1, este parámetro está expresado en bytes.
* Para el FileEncoderType 2, 3, 4 y 5 este parámetro está expresado en bloques de 31 bits. En este caso sería igual al número de elementos long que hay en el array que tiene los datos del número. (En cada elemento long del array se almacenan 31 bits).
* Los fileEncoderType 6, 7 y 8 no usan este parámetro.
* NumBitsPerIterationSimpleEncoder. Este parámetro de configuración de encriptación aplica al número de bits que devuelve el generador pseudoaleatorio en cada iteración.

Este parámetro aplica al generador pseudoaleatorio que se utiliza en el primer paso (XOR).

Este parámetro influye drásticamente en el tiempo de computación de la encriptación.

Para ficheros pequeños a encriptar, es aconsejable utilizar un valor muy bajo de este parámetro, para que la fortaleza de la encriptación sea mayor.

Para ficheros grandes o muy grandes, será mejor utilizar valores grandes para este parámetro, ya que de lo contrario el tiempo para encriptar/desencriptar sería muy grande y se haría pesado tener que esperar cada vez que se necesite encriptar/desencriptar el fichero.

Los valores aceptados por el FileEncoderType 1 y los demás son ligeramente distintos:

* Para el FileEncoderType 1, este parámetro puede aceptar los valores 1, 2 o 4. El valor 4 es el que produce encriptaciones más rápidas, sin embargo no se aconseja utilizar con SizeOfNumbersSimpleEncoder menores a 16 bytes, ya que en ese caso se obtendrían números entre 0 y 15 en cada iteración que no serían equiprobables. A partir de tamaños de números de 16 bytes sí que podrían utilizarse aunque mejor a partir de números de 32 bytes.
* Para el FileEncoderType 2, 3, 4, 5,6, 7 y 8 este parámetro puede aceptar los valores 1, 2, 4 o múltiplos de 8 hasta 64 (64 es el valor máximo). No todos los valores están permitidos.

Para el valor de 4 es aconsejable no utilizar tamaños de números (configurados en SizeOfNumbersSimpleEncoder) menores a 5 bloques. Mejor aún a partir de 8 bloques.

Para valores mayores que 4 (múltiplos de 8 hasta 64), el valor configurado debe cumplir esta condición:

NumBitsPerIterationSimpleEncoder <= 3 x SizeOfNumbersSimpleEncoder

* NumBitsPerIterationReordererEncoder. Este parámetro de configuración de encriptación aplica al número de bits que devuelve el generador pseudoaleatorio en cada iteración.

Este parámetro aplica al generador pseudoaleatorio que se utiliza en el segundo paso (reordenación).

Este parámetro influye drásticamente en el tiempo de computación de la encriptación.

Para ficheros pequeños a encriptar, es aconsejable utilizar un valor muy bajo de este parámetro, para que la fortaleza de la encriptación sea mayor.

Para ficheros grandes o muy grandes, será mejor utilizar valores grandes para este parámetro, ya que de lo contrario el tiempo para encriptar/desencriptar sería muy grande y se haría pesado tener que esperar cada vez que se necesite encriptar/desencriptar el fichero.

Los valores aceptados por el FileEncoderType 1 y los demás son ligeramente distintos:

* Para el FileEncoderType 1, este parámetro puede aceptar los valores 1, 2 o 4. El valor 4 es el que produce encriptaciones más rápidas, sin embargo no se aconseja utilizar con SizeOfNumbersReordererEncoder menores a 16 bytes, ya que en ese caso se obtendrían números entre 0 y 15 en cada iteración que no serían equiprobables. A partir de tamaños de números de 16 bytes sí que podrían utilizarse aunque mejor a partir de números de 32 bytes.
* Para el FileEncoderType 2, 3, 4 y 5 este parámetro puede aceptar los valores 1, 2, 4 o múltiplos de 8 hasta 64 (64 es el valor máximo). No todos los valores están permitidos.

Para el valor de 4 es aconsejable no utilizar tamaños de números (configurados en SizeOfNumbersReordererEncoder) menores a 5 bloques. Mejor aún a partir de 8 bloques.

Para valores mayores que 4 (múltiplos de 8 hasta 64), el valor configurado debe cumplir esta condición:

NumBitsPerIterationReordererEncoder <= 3 x SizeOfNumbersReordererEncoder

* Los fileEncoderType 6, 7 y 8 no utilizan este parámetro.
* NumOfThreads. Indica el número de hilos de encriptación que se van a intentar utilizar en caso de que el fichero encriptado y el fichero desencriptado estén en el mismo disco (para los fileEncoderType 7 y 8).

Se puede configurar con el valor “optimum”, que indica que previamente a la encriptación se estimará automáticamente el valor óptimo. Esta característica es experimental, y lo mejor sería indicar directamente el número óptimo de hilos con un valor numérico.

El número total de hilos a utilizar en este caso, sería el valor de este parámetro más un hilo adicional que se utilizará para la lectura/escritura en disco.

* NumOfThreadsForDifferentDrives. Indica el número de hilos de encriptación que se van a intentar utilizar en caso de que el fichero encriptado y el fichero desencriptado estén en discos distintos (para los fileEncoderType 7 y 8).

Se puede configurar con el valor “optimum”, que indica que previamente a la encriptación se estimará automáticamente el valor óptimo. Esta característica es experimental, y lo mejor sería indicar directamente el número óptimo de hilos con un valor numérico.

El número total de hilos a utilizar en este caso, sería el valor de este parámetro más dos (un hilo adicional que se utilizará para la lectura del fichero de entrada y otro hilo adicional que se utilizará para la escritura del fichero de salida).

* MaxTotalNumOfThreads. Indica el límite máximo de hilos totales a utilizar en una encriptación (para los fileEncoderTypes 7 y 8). Si el número total de hilos es mayor que el valor de este parámetro, entonces se limitará el número de hilos de encriptación al valor adecuado para que no se supere este valor.

Se puede utilizar el valor “max”, que indicará que como número máximo de hilos, se toma el número máximo de hilos de procesamiento que soporta el procesador del PC sobre el que se ejecuta la aplicación (relacionado con el número de cores que tiene el procesador).

* SizeInBytesForVolumes. Indica el número máximo de octetos que podrá tener cada volumen del fichero encriptado (para el fileEncoderType 8).

## Configuración de la aplicación

Este formulario permite modificar los parámetros de configuración generales de la aplicación.

El formulario tiene varias pestañas que detallaremos a continuación

### Parámetros configuración de encriptación de la aplicación

La pestaña tiene el siguiente aspecto:



Los parámetros que pueden configurarse son los siguientes:

* **Borrar fichero encriptado después de encriptar**. Este check box permite configurar precisamente eso, si tras encriptar un fichero, deseas borrar automáticamente el fichero sin encriptar.

El borrado que se realiza no es un borrado normal, sino que se borra poniendo ceros en cada uno de los bytes del fichero, y tras esto, se hace un borrado normal.

De esta forma, es imposible recuperar el fichero sin encriptar, a menos que desencriptes el fichero encriptado de la forma habitual, utilizando la contraseña.

* **Renombrar fichero encriptado a .old después de desencriptar**. Este check box permite configurar si tras una desencriptación deseas guardar el fichero encriptado que acaba de ser desencriptado como .old. De esta forma te aseguras tener almacenada una versión encriptada del fichero, por si acaso ocurre algo inesperado.
* **Preguntar si hay que sobreescribir fichero encriptado .old**. En caso de que la opción anterior esté activa, esta opción permite configurar si deseas ser preguntado para confirmar el sobreescribir un fichero encriptado .old con el nuevo fichero encriptado pendiente de ser almacenado como .old.
* **Prioridad de configuración de encriptación**. En esta tabla se puede configurar el orden de prioridad de la configuración de encriptación.

Por defecto las opciones tienen esta prioridad:

* (1) - Manual. Se refiere a cuando aplicas una configuración de encriptación manualmente.

Lo lógico es ponerla como primera opción, ya que si no la pones como primera opción, es posible que hagas la modificación de la configuración de encriptación y la aceptes, pero esta configuración no sea aplicada cuando realizas la encriptación.

* (2) - Basado en tamaño de fichero. Esta opción se refiere a que la configuración de encriptación se obtiene de la lista de configuraciones de encriptación que hay configuradas dependiendo del tamaño del fichero. Ver punto: 3.6-Lista de configuraciones de encriptación
* (3) - Basados en los parámetros del fichero encriptado después de desencriptar. Se refiere a recoger la configuración de encriptación basada en la configuración de encriptación existente en el fichero encriptado que acaba de ser desencriptado.

La manera de configurar este orden de prioridades, es seleccionar en la tabla la fila donde está la opción a la que deseas cambiarle la prioridad, y pulsar los botones Arriba y Abajo para subir o bajar la prioridad.

* **Preguntar siempre antes de cambiar fileEncoderTypes obsoletos**.

En esta versión de la aplicación, se han encontrado unos fileEncoderTypes que son obsoletos, debido a un bug encontrado en ellos.

Este checkbox permite configurar si cuando se encuentre uno de esos fileEncoderTypes a la hora de encriptar un archivo, se preguntará al usuario qué hacer.

En caso de que este checkbox esté desmarcado, se habilitará el siguiente checkbox.

* **Cambiar siempre fileEncoderTypes obsoletos**.

En caso de que el checkbox anterior esté marcado, este checkbox no podrá configurarse, ya que se preguntará siempre al usuario qué hacer mediante una ventana.

En caso de que el checkbox anterior esté desmarcado, este checkbox permitirá configurar el comportamiento automático cuando se encuentren fileEncoderTypes obsoletos.

Si se marca este checkbox (cambiar siempre fileEncoderTypes obsoletos), cuando se encuentre o bien el fileEncoderType 2 o el 3, se cambiará por el fileEncoderType 4.

Si se desmarca este checkbox, se dejará el fileEncoderType configurado, ignorando que sea obsoleto (esto puede ser útil si se desea desencriptar con una versión anterior de la aplicación, que sólo tenía los fileEncoderTypes 1 y 2).

### Parámetros configuración de idiomas de la aplicación

La pestaña tiene el siguiente aspecto:



Los parámetros que pueden configurarse son los siguientes:

* **Idioma**. Es el idioma de los textos que escribe la aplicación.

Los idiomas que pueden escogerse son:

* EN. Inglés
* ES. Español
* CAT. Catalán
* **Locale de idioma**. Es el locale de Java que va a utilizar la aplicación para este idioma.

La aplicación lo utiliza para convertir números a cadenas numéricas formateadas.

* **Idioma web**. Indica el idioma que se va a utilizar en las consultas de nueva versión al servidor.
* **Idioma adicional**. Es un nuevo idioma adicional que aparece a la hora de poder cambiar de idioma.

Para el nuevo idioma, tendrás que seleccionar el locale de Java a utilizar.

Si quieres poner un idioma del que no existen textos en la aplicación, puedes añadir tu idioma traduciendo los ficheros que hay dentro del directorio que se crea cuando pulsas el botón de Aceptar.

Los ficheros a traducir se copian al directorio indicado en: "Directorio de idioma adicional"

El formato de estos ficheros de textos es el de properties de java.

Para los que no conozcáis este formato, os diré que un fichero properties, tiene un título, y tras este, un número variable de etiquetas con su valor, parecido a lo que sigue:

# TITULO

# xxxxxxxxxx

ETIQUETA1=texto 1

ETIQUETA2=texto 2

...

Las etiquetas deben permanecer sin modificar, y los textos deben modificarse dependiendo de la traducción en el idioma escogido.

Además, también hay algunos archivos en formato RTF, que tendrás que traducir con un editor de RTF (un editor típico para ese formato es el Word de Office).

Si creas la traducción para un idioma adicional distinto a los que hay disponibles en la aplicación, si quieres puedes enviármelo (frojasg1@hotmail.com) y lo incluiré en las siguientes versiones de la aplicación.

### Parámetros configuración de vista de la aplicación

La pestaña tiene el siguiente aspecto:



* Tamaño de las ventanas de la aplicación. Este parámetro permite controlar el aspecto de las ventanas que aparecen en la aplicación, existiendo la posibilidad de escoger el tamaño pequeño, el normal, o bien el tamaño grande.

## Borrado de ficheros

La aplicación no permite encriptar/desencriptar si el fichero de salida ya existe, es decir, no permite sobreescribir un fichero.

Es por ello que se han habilitado opciones para el borrado de ficheros de manera cómoda y flexible.

Para abrir el diálogo de borrado de fichero, sólo hay que pulsar el botón con un aspa asociado al fichero que deseas borrar:



Si el fichero que has escogido para ser borrado no existe, entonces aparecerá una indicación en el componente de texto (Historia) de la parte inferior.

Si el fichero existía, entonces aparecerá el diálogo que permite escoger el método de borrado que deseas utilizar:



En este punto se permiten tres opciones:

* Borrado seguro: Es un borrado que no permite que el fichero que se borra se pueda recuperar con ninguna herramienta de recuperación de ficheros.

Se utiliza el método de escribir con ceros todos los octetos del fichero, para luego, finalmente, realizar un borrado normal.

Este tipo de borrado, al ser más costoso, permite mostrar el progreso en la barra de progreso e incluso puede ser cancelado por el usuario, mediante el botón de cancelar que hay al lado de la barra de progreso.

* Borrado normal: Se realiza un borrado normal del archivo
* Cancelar: Permite cancelar la operación, en caso de que te hubieras equivocado de botón.

Una vez que se ha escogido el tipo de borrado, la aplicación comprueba si existe el fichero pareja (es decir, si se trata de borrar el fichero encriptado, se comprueba que exista el fichero desencriptado, y con el fichero desencriptado, exactamente igual).

Esto es así, para evitar un borrado accidental de un archivo, en caso de que no exista su pareja, ya que en teoría no podría volver a crearse a partir de ella el fichero que deseamos borrar.

En ese caso, se muestra un aviso como este:



Que básicamente pide la confirmación de que se desea borrar el fichero.

Una vez que la aplicación se ha asegurado que se quiere borrar el fichero, entonces se ejecutan las acciones sobre todos los ficheros asociados (es decir, si se trata de un fichero encriptado del fileEncoderType-8, que permite múltiples volúmenes, entonces se borrarían todos los volúmenes de una vez).

## Acerca de

La opción Acerca de …, muestra una ventana con un resumen de las novedades en esta versión. También se incluyen los agradecimientos.

Tiene un aspecto como este:



## Licencia

La opción Licencia, dentro del menú Acerca de, permite mostrar la licencia que se aceptó la primera vez que se ejecutó la aplicación.

Tiene un aspecto como este:



# Encriptación por la línea de comandos

Además de poder encriptar y desencriptar desde la interfaz gráfica de la aplicación, también es posible invocar a las funciones de encriptar y desencriptar mediante la línea de comandos.

Se puede pasar la password de encriptación mediante un parámetro de los scripts. Esto no es muy recomendable, ya que para encriptar/desencriptar hay que poner la password en claro, y es posible que alguien pueda verla mirando la historia de comandos, o incluso viendo los procesos que están ejecutándose en el sistema. También es posible que tengas que teclear la contraseña y que esta se vea en tu pantalla, por lo que esa no es una forma muy segura de usar los scripts.

Es preferible no pasar el parámetro password (que es opcional) a los scripts. De esta forma la aplicación preguntará por teclado el valor de la password y su confirmación.

Existe un directorio (\_scripts) en el que se han generado unos scripts para facilitar la tarea.

Hay dos versiones de los scripts una para Windows (scripts \*.bat) y otra que sirve tanto para Linux como para Mac (scripts \*.sh)

## Scripts para Windows

La versión para Windows de los scripts se encuentra en el directorio:

...\\_scripts\windows

Los scripts disponibles son estos:

* command.interface.FileEncoder.bat
* decodeFile.bat
* encodeFile.bat
* example.decodeFile.bat
* example.encodeFile.bat

Para invocar a estos scripts, hay que hacerlo desde el mismo directorio en el que se encuentran ubicados.

El primer script contiene la llamada a la aplicación de Java, recogiendo los argumentos que le son pasados y añadiéndolos a los argumentos para la aplicación de Java.

El segundo script contiene los comandos para desencriptar un fichero encriptado. Toma como parámetros el nombre del fichero encriptado y la contraseña.

El tercer script contiene los comandos para encriptar un fichero. Toma como parámetros el nombre del fichero y la contraseña.

El cuarto script es un ejemplo de cómo invocar al script que encripta.

El quinto script es un ejemplo de cómo invocar al script que desencripta.

En todos los scripts, el parámetro password se ha hecho opcional, de manera que si no se incluye la password en la invocación a la aplicación, esta preguntará la password por teclado.

## Scripts para Linux y para Mac

La versión para Linux y Mac de los scripts se encuentra en el directorio:

.../\_scripts/Mac.or.Linux

En un principio los scripts están comprimidos en el fichero: scripts.tar.gz

Hay que descomprimir este archivo para poder trabajar con los scripts. Para ello, desde el directorio donde se encuentra el archivo scripts.tar.gz, teclear el siguiente comando:

tar -xvzf scripts.tar.gz

Una vez descomprimidos en el directorio tendremos los siguientes scripts:

* command.interface.FileEncoder.sh
* decodeFile.sh
* encodeFile.sh
* example.decodeFile.sh
* example.encodeFile.sh

Para invocar a estos scripts, hay que hacerlo desde el mismo directorio en el que se encuentran ubicados.

El primer script contiene la llamada a la aplicación de Java, recogiendo los argumentos que le son pasados y añadiéndolos a los argumentos para la aplicación de Java.

El segundo script contiene los comandos para desencriptar un fichero encriptado. Toma como parámetros el nombre del fichero encriptado y la contraseña.

El tercer script contiene los comandos para encriptar un fichero. Toma como parámetros el nombre del fichero y la contraseña.

El cuarto script es un ejemplo de cómo invocar al script que encripta.

El quinto script es un ejemplo de cómo invocar al script que desencripta.

En todos los scripts, el parámetro password se ha hecho opcional, de manera que si no se incluye la password en la invocación a la aplicación, esta preguntará la password por teclado.

## Parámetros de encriptación como argumentos

De los scripts anteriores el principal es este:

command.interface.FileEncoder

Desde él se invoca a la aplicación Java y como parámetros toma principalmente los que ya se han descrito en el punto: 3.7.1-Parámetros de la configuración de encriptación

Los parámetros disponibles para encriptar/desencriptar son estos:

* -password Seguido por un argumento con la password. Es opcional. Si no se suministra la password, la aplicación la preguntará por teclado al usuario (password y confirmación).
* -encodedFileName Seguido por un argumento con el nombre de fichero encriptado.
* -decodedFileName Seguido por un argumento con el nombre de fichero desencriptado.
* -encode o -decode Para seleccionar entre encriptación y desencriptación.
* -fileEncoderType Seguido por un argumento con el identificador del fileEncoderType (1 obsoleto or 2) a utilizar en la encriptación (sólo para encriptación)
* -sizeOfNumbersSimpleEncoder Seguido por el tamaño de los números para el paso de encriptación 1 (XOR) (sólo para encriptación).
* -sizeOfNumbersReordererEncoder Seguido por el tamaño de los números para el paso 2 (reordenación) (sólo para encriptación).
* -numberOfBitsPerIterationSimpleEncoder Seguido por el número de bits devueltos por el generador pseudoaleatorio en cada iteración para el paso 1 (XOR) (sólo para encriptación).
* -numberOfBitsPerIterationReordererEncoder Seguido por el número de bits devueltos por el generador pseudoaleatorio en cada iteración para el paso 2 (reordenación). (sólo para encriptación).
* -numBytesFileSlice Seguido por el número de bytes de la rodaja. Se recomienda utilizar un valor alto para este paramétro. La memoria utilizada por la aplicación está limitado aproximadamente por 100 veces el tamaño de rodaja. (sólo para encriptación).
* -useFileSizeForEncryptingParams Si este parámetro está presente, los parámetros de encriptación (definidos por los 5 parámetros anteriores) serán calculados basándose en el tamaño del fichero a encriptar y los 5 parámetros anteriores serán sobreescritos.
* -encDecOnDifferentDrives Sólo aplica para los nuevos tipos de encriptación (FileEncoder-7 y 8).

Si este parámetro está presente (no tiene valor asociado), querría decir que los ficheros de entrada y salida se encuentran en discos distintos y que la aplicación va a utilizar hilos separados para la lectura del fichero de entrada y para la escritura del fichero de salida (uno para cada uno).

Si este parámetro no está presente entonces se utilizará un solo hilo separado conjunto para la lectura del fichero de entrada y para la escritura del fichero de salida.

* -numOfThreads Este parámetro sólo es válido para los nuevos tipos de encriptación (fileEncoderType-7 y 8), e indica el número de hilos de ejecución de encriptación/desencriptación que se van a utilizar (en el caso de que no aparezca el parámetro: -encDecOnDifferentDrives).

 (ver apartado: ***5.4-Utilización de hilos de ejecución (paralelismo)*** sobre el uso de hilos ).

Valor *optimum*:

Este parámetro puede configurarse con un valor no concreto “optimum” (que está en fase experimental) que indicará a la aplicación que este parámetro se estimará justo antes de la encriptación mediante la fórmula indicada en el apartado ***5.4-Utilización de hilos de ejecución (paralelismo)*** ).

* -numOfThreadsForDifferentDrives. Este parámetro de configuración sirve para configurar el número de hilos de encriptación a utilizar en el caso de que los ficheros de entrada y de salida se encuentren en discos distintos (este parámetro sería el que aplicaría para configurar el número de hilos en el caso de que también esté presente el parámetro: -encDecOnDifferentDrives

Este parámetro toma valores mayores que el parámetro homólogo (el anterior), cuando los archivos de entrada y salida se encuentran en el mismo disco.

( Ver apartado ***5.4-Utilización de hilos de ejecución (paralelismo)*** para más detalle).

Valor *optimum*:

Este parámetro puede configurarse con un valor no concreto “optimum” (que está en fase experimental) que indicará a la aplicación que este parámetro se estimará justo antes de la encriptación mediante la fórmula indicada en el apartado ***5.4-Utilización de hilos de ejecución (paralelismo)*** ).

* -maxTotalNumOfThreads. Este parámetro sirve para limitar el número máximo total de hilos a utilizar.

El número total de hilos a utilizar, es el número de hilos de encriptación (que en caso de limitación, sería el valor que se modificaría para cumplir con el límite de este parámetro), más el número de hilos fijos de entrada/salida (1 o 2).

( Ver apartado ***5.4-Utilización de hilos de ejecución (paralelismo)*** para mayor detalle).

Este parámetro se puede configurar con el valor “max”, que indica el número máximo de hilos que permite utilizar el procesador donde se ejecuta la aplicación.

* -sizeInBytesForVolumes. Este parámetro se utiliza para configurar el tamaño máximo de los volúmenes en los que se dividirá el fichero encriptado en el tipo de encriptación 8.

# Método de encriptación

Los algoritmos de encriptación y desencriptación son muy similares, e idénticos en tiempo de computación (encriptación simétrica).

En los siguientes puntos se verá un resumen del método de encriptación utilizado en la aplicación.

## Cabecera del fichero encriptado

Se calcula una hash de la contraseña de encriptación (SHA-256) que devuelve una clave de 256 bits que llamaremos clave-1.

El tamaño de clave-1 es siempre el mismo.

Si utilizamos el generador pseudoaleatorio con diferentes parámetros de configuración, es posible que una clave de 256 bits no sea óptima.

El tamaño óptimo de la clave a ser utilizada para inicializar los generadores pseudoaleatorios para la encriptación es calculado y lo llamamos tamaño-2

El micrófono es utilizado como fuente de bytes aleatorios, y se cogen de él tamaño-2 bytes que serán utilizados para formar la clave-2 que será utilizada para encriptar.

Si el micrófono no está disponible, se obtendrán bytes aleatorios con la función generateSeed de la clase SecureRandom.

Este método de obtención de octetos aleatorios es posible que sea más lento que la obtención de octetos aleatorios a través del micrófono, por lo que es aconsejable tener el micrófono conectado.

La clave-2 se utiliza para encriptar el fichero.

La clave-1 se utiliza para encriptar la clave-2

La clave-2 encriptada, los parámetros de configuración de la encriptación (sin encriptar) y una hash (MD5) (encriptada) del fichero original serán escritos en la cabecera del fichero encriptado. (Eso quiere decir que además de la encriptación/desencriptación, hay que leer el fichero desencriptado para calcular la hash MD5).

Esta información será utilizada para desencriptar el fichero. La hash MD5 se utilizará para comprobar que el proceso de desencriptación ha ido correctamente.

En caso de los fileEncoderType 5, 6, 7 y 8, será necesario guardar también la longitud del archivo desencriptado, para poder recuperarlo correctamente con la desencriptación.

En el caso de los fileEncoderType 7 y 8, también se almacena en la cabecera el número de hilos de ejecución de encriptación a utilizar.

En el caso del fileEncoderType 8 también se almacena el tamaño máximo de los volúmenes en los que se divide el fichero encriptado.

Debido a la particularidad de los fileEncoderType 5, 6, 7 y 8, previamente a la acción de desencriptar, se tiene que crear el fichero completo repleto de ceros. Es por eso que para estos tipos de encriptación, la desencriptación se hace ligeramente más lenta que la encriptación.

## Algoritmo de encriptación

Esta aplicación realiza la encriptación de ficheros en rodajas, esto es dividiendo el fichero en partes más pequeñas. Para cada encriptar cada una de las rodajas se siguen dos pasos:

* Encriptación de la rodaja estilo XOR. En esta parte de la encriptación, generando bits pseudoaleatorios y haciendo un XOR con los datos originales, se obtiene una rodaja encriptada cuyos bytes son bastante aleatorios.

Si dejáramos aquí la encriptación, sería relativamente rápido intentar romper esta encriptación si se conocen algunos bytes del inicio del fichero sin encriptar, como es posible que se conozcan en el caso de ficheros que puedan abrirse con alguna aplicación (por ejemplo Word o Excel), ya que generalmente esos ficheros tendrán algún tipo de cabecera que hace la encriptación de este tipo más vulnerable.

Debido a ello la aplicación genera un segundo paso en la encriptación:

* Reordenación pseudoaleatoria de los bytes de la rodaja. En esta parte de la encriptación, se generan posiciones pseudoaleatorias, donde se van a ir moviendo uno a uno los bytes de la rodaja, reordenando el conjunto de los bytes de la rodaja.

Para obtener los bytes iniciales del fichero desencriptado, es necesario desencriptar una rodaja completa, por lo que eso hace que se haga más difícil romper la encriptación, ya que es mucho más costoso en térmimos de tiempo de computación desencriptar una rodaja completa (cuyo tamaño es configurable, y por defecto está a 256.000 bytes), que desencriptar los 100 o 1000 primeros bytes y compararlos con una cabecera válida de word o excel , por ejemplo.

Los fileEncoderType 6, 7 y 8 eliminan este paso, por lo que son mucho más rápidos que los otros fileEncoderTypes.

* Reordenación pseudoaleatoria de las rodajas. Esta parte de la encriptación sólo está presente en los fileEncoderTypes 5, 6, 7 y 8.

Este paso permite que la encriptación de archivos muy grandes sea muy difícil de romper, ya que al ser aleatorio el orden de las rodajas, para desencriptar la primera rodaja (que contiene la cabecera del archivo desencriptado que puede tener un formato conocido, y permitir determinar si una password generada por fuerza bruta es buena o no), habrá que desencriptar en media, la mitad de las rodajas del archivo.

Si el archivo es muy grande, un ataque por fuerza bruta se hace inviable, debido al coste computacional (se tardaría mucho en encontrar la password buena, incluso con superordenadores).

Esta característica hace que los fileEncoderTypes 5, 6, 7 y 8 sean los más difíciles de romper para ficheros grandes.

En los pasos anteriores, para realizar su parte de la encriptación se utiliza un generador pseudoaleatorio caótico, que en la implementación actual están caracterizados básicamente por un tamaño de los números a utilizar en el generador y el número de bits pseudoaleatorios a devolver en cada iteración. Dependiendo de estos dos parámetros el generador pseudoaleatorio calcula el número de bytes óptimo que debe tener la contraseña.

Cuanto mayor sea el tamaño de los números, mayor será el tamaño óptimo de la contraseña con la que hay que inicializar el generador pseudoaleatorio, y mayor será el tiempo de computación de cada iteración. Así pues, un mayor tamaño de los números hace que el tiempo de encriptación/desencriptación sea mayor. El efecto sobre el tiempo de computación de este parámetro es mayor cuanto menor sea el número de bits por iteración del generador pseudoaleatorio.

La configuración ideal sería una que produjera una encriptación robusta y que fuera rápida.

Para conseguir este objetivo, podríamos utilizar tamaños de número grandes (más robusto) con número alto de bits devueltos por iteración (más rápido) y combinado con tamaños de rodaja lo más grandes posible (mucho más robusto, pero cuanto mayor es este número, más memoria se utiliza).

Para ficheros pequeños se debería utilizar un número muy pequeño de bits devueltos por iteración (por ejemplo: 1) y un tamaño de números grande. El tamaño de la rodaja debería ser como mínimo igual al tamaño del fichero a encriptar.

Como ejemplos de configuraciones de encriptación óptimas:

* Para archivos muy pequeños:
* fileEncoderType 4
* Tamaño de números: 256
* Bits por iteración: 1
* Tamaño de rodaja: 0
* Para archivos grandes (mayores de 100 MB):
* fileEncoderType: 7
* Tamaño de números: 10
* Bits por iteración: 64
* Tamaño de rodaja: (como mucho el tamaño del fichero / 10, con un máximo de 16 MB).
* Número de hilos: 3 (para ficheros de entrada y de salida en el mismo disco)
* Número de hilos: 5 (para ficheros de entrada y de salida en discos diferentes)

## Generador pseudoaleatorio utilizado

El generador pseudoaleatorio utilizado en esta aplicación, es un generador pseudoaleatorio basado en secuencias caóticas.

Es un generador pseudoaleatorio muy sencillo y de muy fácil implementación.

La descripción de este generador pseudoaleatorio fue obtenida del libro:

"Secuencias pseudoaleatorias para telecomunicaciones" Edicions UPC (1996) Ernesto J. Forner Cruselles y José L. Melús Moreno

Un agradecimiento para ellos por escribir un libro tan interesante.

Citando del propio libro:

"

*Este generador, propuesto por M. Romera, I. Jiménez y J. Negrillo [ROM90], está basado*

*también en la generación de secuencias pseudoaleatorias mediante el empleo de funciones caóticas.*

*[ROM90] ROMERA, M.; JIMENEZ, I. NEGRILLO, J. Generación de Secuencias Cifrantes Mediante*

*Funciones Caóticas. I Reunión Española de Criptología. Mallorca, 1990.*

*...*

*La secuencia caótica de este generador (de números reales) está definida por:*

*Xo = 0*

*Xi+1 = Xi2 + K*

 *Si -2 <= K <= 0,25, entonces los números de esta secuencia están acotados: -2 <= Xi <= 2*

*Unas secuencias son cíclicamente convergentes (periodos 1, 2, 3 ...) y otras son caóticas.*

*Cuando K es muy próximo a -2, por ejemplo K = -1,99999XXXXXXXXXXX, las series que se obtienen a partir de la iteración real de Mandelbrot son casi siempre caóticas.*

*...*

*La secuencia real obtenida puede convertirse fácilmente en una secuencia pseudoaleatoria binaria, tomando el signo de cada número de la serie (la distribución es simétrica respecto a cero), o aplicando un criterio de paridad a los dígitos de cada número de la serie.*

"

Basándonos en este generador de secuencias caóticas, hemos creado nuestro generador muy similar a este, que a grandes rasgos cumple con las siguientes condiciones:

-1 <= Xo  <= 1

K = -1,99999999xxxxxxxxxxxxxxxxxxxxxxxx

Para inicializar el generador pseudoaleatorio necesitaremos una semilla (un número de bytes), que utilizaremos para inicializar Xo y la parte variable de K

Tanto X como K deben ser números de precisión configurable que permita una precisión muy grande, ya que si utilizamos los tipos básicos de números de mayor precisión que existen en Java (double), el rango de la semilla sería muy limitado y por fuerza bruta, podría romperse el sistema pseudoaleatorio.

Las opciones que tenemos son:

* Utilizar la clase BigDecimal de Java (Utilizado en el FileEncoderType=1, no recomendado)
* Programar una clase nueva que maneje números de precisión variable (Utilizado en el FileEncoderType=2 y 3 (OBSOLETOS) y FileEncoderType=4 (recomendado para archivos pequeños), 5 (recomendado para archivos medianos) 6 (no recomendado) y 7 y 8 (recomendados para archivos grandes ).

Los bits pseudoaleatorios que se derivan de la secuencia caótica son obtenidos de estas maneras:

* Cuando el número de bits producidos por iteración es 1, 2 o 4 (para todos los fileEncoderTypes):

Se cuentan los bits "1" del Xi y se hace el módulo (2 si el número de bits por iteración era 1, 4 si era 2 o 16 si era 4).

Esta es la versión más robusta del generador pseudoaleatorio (más costosa en tiempo de computación, y menos dependiente del valor concreto de Xi)

De esa forma se obtienen los bits pseudoaleatorios de cada iteración de la secuencia caótica.

* Si se utiliza 1 bit por iteración, no hay limitación de tamaño de número, ya que los posibles valores producidos (0 o 1) son siempre equiprobables independientemente del número de bits utilizado.
* Si se utilizan 2 bits por iteración, es mejor utilizar números de tamaños superiores o iguales a 16 bits, ya que por ejemplo haciendo el módulo 4 de la cuenta de bits 1 de un número caótico 8 bits (resultando 0, 1, 2 o 3), esos valores no serían equiprobables, pero con tamaños de número de 16 bits, la diferencia de probabilidad ya es despreciable.
* Si se utilizan 4 bits por iteración, es mejor utilizar números de tamaños superiores o iguales a 256 bits.

En la siguiente tabla pueden verse las probabilidades de cada valor posible en función del tamaño del número cuando se producen 4 bits pseudoaleatorios por iteración:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Valor | númerosde 16 bits | númerosde 32 bits | númerosde 64 bits | númerosde 128 bits | númerosde 192 bits | númerosde 256 bits |
| 0 | 0,00305176% | 13,99499346% | 9,93997199% | 7,29371224% | 6,55137753% | 6,33706399% |
| 1 | 0,02441406% | 13,17175925% | 9,64196694% | 7,21415681% | 6,52843586% | 6,33043664% |
| 2 | 0,18310547% | 10,97647697% | 8,80338313% | 6,98766512% | 6,46310389% | 6,31156353% |
| 3 | 0,85449219% | 8,08803737% | 7,57614083% | 6,64887061% | 6,36532878% | 6,28331793% |
| 4 | 2,77709961% | 5,25798649% | 6,17124246% | 6,24950378% | 6,24999687% | 6,24999998% |
| 5 | 6,66503906% | 3,00877392% | 4,81247919% | 5,85042763% | 6,13466680% | 6,21668205% |
| 6 | 12,21923828% | 1,52314156% | 3,69661687% | 5,51233488% | 6,03689611% | 6,18843647% |
| 7 | 17,45605469% | 0,73142946% | 2,96941304% | 5,28654495% | 5,97156857% | 6,16956339% |
| 8 | 19,63806152% | 0,48979651% | 2,71754309% | 5,20728020% | 5,94862872% | 6,16293605% |
| 9 | 17,45605469% | 0,73142946% | 2,96941304% | 5,28654495% | 5,97156857% | 6,16956339% |
| 10 | 12,21923828% | 1,52314156% | 3,69661687% | 5,51233488% | 6,03689611% | 6,18843647% |
| 11 | 6,66503906% | 3,00877392% | 4,81247919% | 5,85042763% | 6,13466680% | 6,21668205% |
| 12 | 2,77709961% | 5,25798649% | 6,17124246% | 6,24950378% | 6,24999687% | 6,24999998% |
| 13 | 0,85449219% | 8,08803737% | 7,57614083% | 6,64887061% | 6,36532878% | 6,28331793% |
| 14 | 0,18310547% | 10,97647697% | 8,80338313% | 6,98766512% | 6,46310389% | 6,31156353% |
| 15 | 0,02441406% | 13,17175925% | 9,64196694% | 7,21415681% | 6,52843586% | 6,33043664% |

Como se puede observar en la tabla, si el tamaño de los números es pequeño no obtendremos unos bits retornados lo suficiente equiprobables como para encriptar.

* Cuando el número de bits producidos por iteración es un múltiplo de 8 entre 8 y 64 (ambos incluidos) (FileEncoderType 2, 3, 4, 5, 6, 7 y 8):

En ese caso se toman bytes completos que forman parte de Xi para utilizarlos como si fueran números pseudoaleatorios. Esto no es exacto, pero para que vaya más rápido hay que sacar los bits pseudoaleatorios de algún sitio. Seguramente no es tan robusto como contar el número de bits "1" del número Xi, pero creo que puede servir.

Obtener los bits pseudoaleatorios por este método es muy rápido, sólo hay que coger los bytes pseudoaleatorios directamente. Además, permite coger un número grande de bits por iteración.

La condición que debe cumplirse entre el número de bits producidos por iteración y el tamaño en bloques (bloques de 31 bits, cada bloque se almacena en un long, que se almacena en un array de longs de longitud variable) es esta:

NumeroBitsPorIteracion / 8 <= 3 x NumeroBloquesPorNumero

Ya que se cogen como mucho 3 bytes de cada bloque.

## Utilización de hilos de ejecución (paralelismo)

Esta característica sólo está presente en los fileEncoderType 7 y 8.

Estos tipos de encriptación añaden soporte para múltiples hilos de ejecución.

* El uso de los hilos:
	+ Cuando el fichero encriptado y el desencriptado están en el mismo disco:
		- Un hilo fijo que se utiliza para centralizar la entrada/salida desde/hacia ficheros. Ese hilo, alimenta a los hilos de encriptación leyendo rodajas del fichero de entrada y también se encarga de ir escribiendo el resultado en el fichero de salida.
		- Un conjunto de hilos de encriptación.
	+ Cuando el fichero encriptado y el desencriptado están en discos separados:
		- Un hilo fijo que se utiliza para centralizar la lectura del fichero de entrada. Este hilo, alimenta a los hilos de encriptación leyendo rodajas del fichero de entrada.
		- Un hilo fijo que se utiliza para centralizar la escritura en el fichero de salida.

Este hilo recibe rodajas a escribir, y las va escribiendo en paralelo a la lectura del fichero de entrada.

* + - Un conjunto de hilos de encriptación.
* Características de los hilos de encriptación:

Cada hilo de encriptación utiliza un encriptador/desencriptador propio y cada uno de estos hilos se caracteriza por un índice que va desde 0 al número total de hilos de encriptación utilizados.

Cada hilo de encriptación, recibirá por orden todas las rodajas con una posición de rodaja congruente con el índice del hilo módulo el número total de hilos de encriptación. (es decir, las rodajas se van repartiendo en orden una para cada hilo de ejecución).

Cada hilo de encriptación utiliza el encriptador/desencriptador para encriptar/desencriptar en orden las rodajas que le corresponden del fichero de entrada.

* Valor óptimo del número de hilos de encriptación
	+ Caso de un solo hilo fijo de entrada/salida

El número mínimo de hilos de encriptación que minimiza el tiempo total de procesamiento puede calcularse teóricamente con esta fórmula:

$$optimum=EnteroSuperior\left( \frac{T\_{p}}{\left(T\_{i}+T\_{o}\right)}\right)$$

Donde: $optimum$ es el número óptimo de hilos de encriptación a utilizar

 $T\_{p}$ es el tiempo de procesamiento (encriptación pura) de una rodaja

 $T\_{i}$ es el tiempo de lectura de una rodaja

 $T\_{o}$ es el tiempo de escritura de una rodaja

* + - Tenemos que $T\_{p}$ depende de los parámetros de encriptación (el tamaño de los números a utilizar por el generador pseudoaleatorio y también el número de bits pseudoaleatorios obtenidos por iteración). Como es obvio, $T\_{p}$ también depende de la potencia de la CPU.
		- Y tenemos que $T\_{i}$ y $T\_{o}$ dependen de la unidad de almacenamiento donde se está el fichero de entrada y donde se escribirá el fichero de salida. También dependerá de si el disco está siendo utilizado en esos momentos por otra aplicación, de si está fragmentado, etc.
	+ Caso de hilos de entrada y salida separados (en el caso de que los ficheros estén en discos separados)

El número óptimo de hilos puede ser calculado como:

$$optimum=EnteroSuperior\left( \frac{T\_{p}}{max\left(T\_{i}, T\_{o}\right)}\right)$$

Ya que los hilos de lectura y de escritura en disco están separados (trabajando en paralelo), y el tiempo total que se tarda en leer y escribir una rodaja, puede calcularse como el máximo entre el tiempo de lectura y el de escritura (que en cualquier caso será menor que la suma de ellos).

Eso quiere decir que el número óptimo de hilos en este caso será mayor o igual al del caso de entrada y salida conjunta (en el mismo disco).

* Limitaciones teóricas:

El número de hilos de encriptación no debería ser mayor que el número de hilos en paralelo que soporta el procesador menos el número de hilos dedicados a lectura/escritura de disco. (para ello se puede utilizar el parámetro: -maxTotalNumOfThreads, que limita automáticamente el máximo número total de hilos a usar).

El número de hilos que soporta el procesador suele ser un múltiplo del número de cores que tiene el procesador.

Por ejemplo, el procesador de mi PC es un i7, (4 cores), pero debido al hyper-threading el número de hilos simultáneos que maneja el procesador es de 8.

Tras el cálculo del número óptimo de hilos de encriptación, debe limitarse ese resultado al límite teórico que acabamos de ver.

Esto es así, porque podría ser que tuviéramos un sistema con el disco muy rápido (tiempo de entrada y salida pequeños), y un tiempo de procesamiento grande (procesador muy lento).

En ese caso tendríamos un valor óptimum grande, que podría ser superior al número máximo de hilos que es recomendable utilizar (relacionado con las capacidades de nuestro procesador).

En este caso, a pesar de tener el disco muy rápido, tendríamos el cuello de botella en el procesador, y en este caso no podríamos utilizar el número óptimo de hilos.

* La opción más rápida

Si los ficheros están en discos separados, se puede conseguir una mayor eficiencia y ocupación de CPU, traduciéndose en una mayor rapidez de encriptación. (Siempre que tengamos una CPU suficientemente potente).

Lo ideal si se busca rapidez en la encriptación, es hacer que los ficheros de entrada y de salida se encuentren en discos separados.

* Memoria utilizada

Otro punto a tener en cuenta es la memoria utilizada por la aplicación.

Cuanto mayor sea el número de hilos, mayor será la cantidad de memoria utilizada por la aplicación.

Es por ello que se ha permitido crear un límite, para evitar valores muy grandes sin sentido, que utilizarían mucha memoria del sistema.

# Tipos de encriptación

La aplicación FileEncoderApplication permite encriptar en base a un conjunto de tipos de encriptación que se va ampliando en cada versión.

Ello permite mantener la compatibilidad hacia atrás, permitiendo aumentar su funcionalidad añadiendo nuevos tipos de encriptación.

En este capítulo vamos a detallar los distintos tipos de encriptación disponibles en esta versión de la aplicación.

## FileEncoderType-1

FileEncoderType **no recomendado**, que funciona con un generador pseudoaleatorio caótico que internamente utiliza números en formato BigDecimal.

La desventaja de este generador pseudoaleatorio es que está íntimamente ligado a la implementación en Java, siendo más complicada su traducción a otros posibles lenguajes de programación.

Además, utilizando ese tipo de números se perdía el control y pudiera haber pasado que la implementación de esa clase cambiara entre versiones de Java, rompiendo la compatibilidad de la aplicación ejecutada sobre diferentes versiones de JRE.

Es por ello que la utilización de este tipo de encriptación no está recomendada.

Sin embargo, por compatibilidad hacia atrás todavía existe.

Las características de este tipo de encriptación son estas:

* Basado en rodajas.
* Cada rodaja se codifica con una encriptación basada en XOR.
* Cada rodaja reordena sus octetos internamente, para hacer más costosa su desencriptación.
* Generador pseudoaleatorio caótico basado en la clase de Java BigDecimal.
* Parámetros:
	+ NumBytesFileSlice. Tamaño de rodaja en bytes.
	+ SizeOfNumbersSimpleEncoder. Tamaño de los números BigDecimal (en octetos) que conforman el generador pseudoaleatorio XOR (rango-> 1-127)
	+ SizeOfNumbersReordererEncoder. Tamaño de los números BigDecimal (en octetos) que conforman el generador pseudoaleatorio utilizado para la reordenación. (rango-> 1-127)
	+ NumBitsPerIterationSimpleEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio XOR.
	+ NumBitsPerIterationReordererEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio para la reordenación.

## FileEncoderType-2

FileEncoderType **obsoleto**, que funcionaba con un generador pseudoaleatorio caótico que internamente utilizaba números en formato BigLong que pasó a ser BigLong\_deprecated tras haber localizado un ERROR, que aparecía de vez en cuando, pero que aparecía siempre al encriptar archivos medianamente grandes.

Es un tipo que se mantiene por compatibilidad hacia atrás, para poder decodificar ficheros encriptados con este tipo de encriptación de versiones anteriores de la aplicación.

Las características de este tipo de encriptación son estas:

* Basado en rodajas.
* Cada rodaja se codifica con una encriptación basada en XOR.
* Cada rodaja reordena sus octetos internamente, para hacer más costosa su desencriptación.
* Generador pseudoaleatorio caótico basado en la clase propia BigLong\_deprecated, que **tiene un error grave**.
* Parámetros:
	+ NumBytesFileSlice. Tamaño de rodaja en bytes.
	+ SizeOfNumbersSimpleEncoder. Tamaño de los números BigLong\_deprecated (en bloques de 31 bits) que conforman el generador pseudoaleatorio XOR (rango-> 1-127)
	+ SizeOfNumbersReordererEncoder. Tamaño de los números BigLong\_deprecated (en bloques de 31 bits) que conforman el generador pseudoaleatorio utilizado para la reordenación. (rango-> 1-127)
	+ NumBitsPerIterationSimpleEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio XOR.
	+ NumBitsPerIterationReordererEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio para la reordenación.

## FileEncoderType-3

FileEncoderType **obsoleto**, que es idéntico al anterior, pero con el tamaño de los números extendidos para aprovechar también el bit de signo del byte destinado a almacenarlo.

Es un tipo que se mantiene por compatibilidad hacia atrás, para poder decodificar ficheros encriptados con este tipo de encriptación de versiones anteriores.

Las características de este tipo de encriptación son estas:

* Basado en rodajas.
* Cada rodaja se codifica con una encriptación basada en XOR.
* Cada rodaja reordena sus octetos internamente, para hacer más costosa su desencriptación.
* Generador pseudoaleatorio caótico basado en la clase propia BigLong\_deprecated, que **tiene un error grave**.
* Parámetros:
	+ NumBytesFileSlice. Tamaño de rodaja en bytes.
	+ SizeOfNumbersSimpleEncoder. Tamaño de los números BigLong\_deprecated (en bloques de 31 bits) que conforman el generador pseudoaleatorio XOR (rango-> 1-256)
	+ SizeOfNumbersReordererEncoder. Tamaño de los números BigLong\_deprecated (en bloques de 31 bits) que conforman el generador pseudoaleatorio utilizado para la reordenación. (rango-> 1-256)
	+ NumBitsPerIterationSimpleEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio XOR.
	+ NumBitsPerIterationReordererEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio para la reordenación.

## FileEncoderType-4

FileEncoderType **especialmente recomendado para archivos pequeños**.

Es idéntico al anterior, pero con el error de la clase BigLong solucionado.

Las características de este tipo de encriptación son estas:

* Basado en rodajas.
* Cada rodaja se codifica con una encriptación basada en XOR.
* Cada rodaja reordena sus octetos internamente, para hacer más costosa su desencriptación.
* Generador pseudoaleatorio caótico basado en la clase propia BigLong. De momento no se han encontrado errores.
* Parámetros:
	+ NumBytesFileSlice. Tamaño de rodaja en bytes.
	+ SizeOfNumbersSimpleEncoder. Tamaño de los números BigLong (en bloques de 31 bits) que conforman el generador pseudoaleatorio XOR (rango-> 1-256)
	+ SizeOfNumbersReordererEncoder. Tamaño de los números BigLong (en bloques de 31 bits) que conforman el generador pseudoaleatorio utilizado para la reordenación. (rango-> 1-256)
	+ NumBitsPerIterationSimpleEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio XOR.
	+ NumBitsPerIterationReordererEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio para la reordenación.

## FileEncoderType-5

FileEncoderType **no recomendado**. (ya que se creó para archivos grandes, pero con tamaño de rodaja pequeño. Eso no es recomendable, ya que **el cabezal del disco duro** va como un loco, y **podría estropearse**).

Las características de este tipo de encriptación son estas:

* Basado en rodajas.
* Cada rodaja se codifica con una encriptación basada en XOR.
* Cada rodaja reordena sus octetos internamente, para hacer más costosa su desencriptación.
* Las rodajas reordenan su posición pseudoaleatoriamente.
* Generador pseudoaleatorio caótico basado en la clase propia BigLong. De momento no se han encontrado errores.
* Parámetros:
	+ NumBytesFileSlice. Tamaño de rodaja en bytes.
	+ SizeOfNumbersSimpleEncoder. Tamaño de los números BigLong (en bloques de 31 bits) que conforman el generador pseudoaleatorio XOR (rango-> 1-256)
	+ SizeOfNumbersReordererEncoder. Tamaño de los números BigLong (en bloques de 31 bits) que conforman el generador pseudoaleatorio utilizado para la reordenación. (rango-> 1-256)
	+ NumBitsPerIterationSimpleEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio XOR.
	+ NumBitsPerIterationReordererEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio para la reordenación.

## FileEncoderType-6

FileEncoderType (**recomendado para archivos muy grandes**, en ordenadores que dispongan de **poca memoria RAM** libre).

Es idéntico al del punto anterior, pero eliminando la reordenación interna de los octetos de cada rodaja.

Se utiliza para ficheros grandes, ya que al eliminar la reordenación interna de los octetos de cada rodaja, se incrementa la velocidad muchísimo.

Además, para hacer este tipo de encriptación más difícil de romper, se realiza una reordenación pseudoaleatoria del orden de las rodajas del fichero.

Esto hace que (en media), se tengan que desencriptar la mitad de las rodajas del fichero encriptado, para obtener la rodaja inicial del fichero desencriptado.

Además eso hace que las rodajas puedan ser mayores ( ya no se necesita tanta memoria para la reordenación de los octetos de cada rodaja), evitando el problema del cabezal explicado en el tipo anterior.

Sin embargo, para esta versión de la aplicación se han creado tipos de encriptación más rápidos (que utilizan múltiples hilos), especialmente diseñados para archivos grandes, y que hacen que este tipo de encriptación sólo sea recomendable cuando se dispone de poca memoria RAM libre en el sistema.

Las características de este tipo de encriptación son estas:

* Basado en rodajas.
* Cada rodaja se codifica con una encriptación basada en XOR.
* Las rodajas reordenan su posición pseudoaleatoriamente.
* Generador pseudoaleatorio caótico basado en la clase propia BigLong. De momento no se han encontrado errores.
* Parámetros:
	+ NumBytesFileSlice. Tamaño de rodaja en bytes.
	+ SizeOfNumbersSimpleEncoder. Tamaño de los números BigLong (en bloques de 31 bits) que conforman el generador pseudoaleatorio XOR (rango-> 1-256)
	+ NumBitsPerIterationSimpleEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio XOR.

## FileEncoderType-7

FileEncoderType (**recomendado para archivos muy grandes**).

Si tu ordenador tiene problemas de memoria RAM libre, habría que configurar el número de hilos de encriptación a 1.

Es idéntico al del punto anterior, pero utilizando múltiples hilos, cosa que bien configurado, puede darle mucha más rapidez a la encriptación.

Las características de este tipo de encriptación son estas:

* Basado en rodajas.
* Cada rodaja se codifica con una encriptación basada en XOR.
* Las rodajas reordenan su posición pseudoaleatoriamente.
* Generador pseudoaleatorio caótico basado en la clase propia BigLong. De momento no se han encontrado errores.
* Se utilizan varios hilos en paralelo:
	+ Cuando el fichero encriptado y el desencriptado están en el mismo disco:
		- Un hilo separado común para la lectura y escritura del disco.
		- Uno o varios hilos para encriptar.
	+ Cuando el fichero encriptado y el desencriptado están en discos diferentes:
		- Un hilo separado para la lectura del disco del fichero de entrada.
		- Un hilo separado para la escritura en disco del fichero de salida.
		- Uno o varios hilos para encriptar.
* Parámetros:
	+ NumBytesFileSlice. Tamaño de rodaja en bytes.
	+ SizeOfNumbersSimpleEncoder. Tamaño de los números BigLong (en bloques de 31 bits) que conforman el generador pseudoaleatorio XOR (rango-> 1-256)
	+ NumBitsPerIterationSimpleEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio XOR.
	+ NumOfThreads. Número de hilos de encriptación para el caso de fichero encriptado y desencriptado en el mismo disco.
	+ NumOfThreadsForDifferentDrives. Número de hilos de encriptación para el caso de fichero encriptado y desencriptado en discos diferentes.
	+ MaxTotalNumberOfThreads. Límite para el número total de hilos a utilizar (contando los hilos de encriptación y los de entrada y salida).

## FileEncoderType-8

FileEncoderType (recomendado **para archivos muy grandes**). Especialmente recomendado **cuando el disco** donde se escribirá el fichero encriptado **tiene limitación de tamaño máximo de fichero**.

Si tu ordenador tiene problemas de memoria RAM libre, habría que configurar el número de hilos de encriptación a 1.

Es idéntico al del punto anterior, pero añadiendo la posibilidad de definir un tamaño máximo de volumen (se dividirá el fichero encriptado en volúmenes, respetando ese tamaño máximo).

Las características de este tipo de encriptación son estas:

* Basado en rodajas.
* Cada rodaja se codifica con una encriptación basada en XOR.
* Las rodajas reordenan su posición pseudoaleatoriamente.
* Generador pseudoaleatorio caótico en la clase propia BigLong. De momento no se han encontrado errores.
* Se utilizan varios hilos en paralelo:
	+ Cuando el fichero encriptado y el desencriptado están en el mismo disco:
		- Un hilo separado común para la lectura y escritura del disco.
		- Uno o varios hilos para encriptar.
	+ Cuando el fichero encriptado y el desencriptado están en discos diferentes:
		- Un hilo separado para la lectura del disco del fichero de entrada.
		- Un hilo separado para la escritura en disco del fichero de salida.
		- Uno o varios hilos para encriptar.
* Con posibilidad de dividir el fichero encriptado en volúmenes.
* Parámetros:
	+ NumBytesFileSlice. Tamaño de rodaja en bytes.
	+ SizeOfNumbersSimpleEncoder. Tamaño de los números BigLong (en bloques de 31 bits) que conforman el generador pseudoaleatorio XOR (rango-> 1-256)
	+ NumBitsPerIterationSimpleEncoder. Número de bits que se obtienen del generador pseudoaleatorio en cada iteración del generador pseudoaleatorio XOR.
	+ NumOfThreads. Número de hilos de encriptación para el caso de fichero encriptado y desencriptado en el mismo disco.
	+ NumOfThreadsForDifferentDrives. Número de hilos de encriptación para el caso de fichero encriptado y desencriptado en discos diferentes.
	+ MaxTotalNumberOfThreads. Límite para el número total de hilos a utilizar (contando los hilos de encriptación y los de entrada y salida).
	+ SizeInBytesForVolumes. Tamaño máximo en octetos para cada uno de los volúmenes en los que se dividirá el archivo encriptado a la hora de encriptar.