Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Licenciatura en Ingeniería en computación

Sistema de compresión de archivos de texto sin pérdida de datos con encriptación de clave pública

Modalidad: Proyecto Tecnológico

Trimestre 17 Invierno

Franck Cortes Hernandez

2123033261

al2123033261@alumnos.azc.uam.mx

José Alfredo Estrada Soto

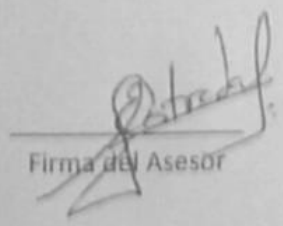
Maestro en Ciencias

Profesor Titular

Departamento de Electrónica

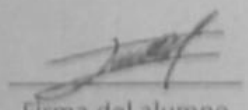
jestrada@correo.azc.uam.mx

Yo, José Alfredo Estrada Soto, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Firma del Asesor

Yo, Franck Cortes Hernandez, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.



Firma del alumno

Resumen

El presente documento presenta el desarrollo e implementación de un sistema de compresión Huffman y encriptado mediante RSA, en la primera parte del documento se describen los antecedentes del sistema, asimismo se plantean los objetivos y la visión general del sistema.

En la segunda parte del documento se establecen las base teóricas de la compresión y encriptación; se describen conceptos tales como: la compresión con perdida/sin pérdida de datos, la encriptación de clave pública y de clave privada, etc. Así mismo se describe el funcionamiento y se dan ejemplos de aplicación de los algoritmos Huffman y RSA.

En la tercera parte se describe detalladamente el desarrollo del proyecto, desde el diseño hasta la implementación del mismo; en esta parte del documento se describe los problemas encontrados al momento del diseño y/o implementación así como las soluciones a dichos problemas, también se describen los cambios requeridos por el proyecto para su culminación.

En la cuarta parte se describen las pruebas, el análisis de resultados y las conclusiones obtenidas; en la quinta y última parte se agrega la bibliografía y el código del sistema.

Tabla de contenido

Índice de figuras

Índice de Tablas

Introducción

La compresión de datos es una rama tecnológica que se ha estado desarrollando desde el inicio de la computación, ya que responde a una necesidad computacional básica: poder guardar la mayor cantidad de datos posibles en el menor espacio posible. Los algoritmos de compresión han ido evolucionando a través de los años, siempre con la idea de aumentar el volumen de datos comprimidos, actualmente logrando algunos ser muy complejos en su implementación, otros muy sencillos pero siempre con la misma filosofía: almacenar la mayor cantidad de datos en el menor espacio informático.

La encriptación hablando en términos de informática y computación, nació de la necesidad de preservar la privacidad en transmisiones/comunicaciones militares, garantizando que una persona (o personas) no autorizada, no pudiera acceder al contenido de los mensajes transmitidos. Actualmente el concepto de encriptación engloba todas aquellas técnicas que se encargan de proteger la información de observadores no autorizados.

En conclusión la compresión de datos y el encrespamiento de estos, son dos ramas de las tecnologías de la información que más interés por su desarrollo han presentado en los últimos años, sobre todo después de la popularización de la red de comunicaciones conocida como internet, y ha llegado a un punto donde se ha vuelto crítico tanto la compresión, como la protección de la información. Es por eso que este documento describe el desarrollo de un software que comprime y encripta archivos de texto plano para permitir visualizar los beneficios obtenidos en cuanto a reducir espacio en disco e inhabilitar el acceso a terceros.

Antecedentes

Tesis

"***Encriptación RSA De Archivos De Texto" de Katia Regina Leon Lomparte,Perú [1]***

A diferencia de mi propuesta donde se hará énfasis en la cantidad de compresión que se pueda realizar, aquí se hace énfasis en demostrar matemáticamente porqué es muy confiable esta encriptación frente a diversos tipos de ataque que podría tener con el fin de tener acceso al mensaje del archivo.

La similitud que encuentro con mi trabajo (aparte del sistema RSA) es el hecho de que la autora realizó su trabajo enfocando su visión a POO trabajando con c++ y java, y que en mi proyecto también se visualizó con POO pero trabajando enteramente en java.

Artículos

"***Matemáticas discretas" de R. Johnsonbaugh[2]***

El capítulo 9 de este libro (que es referente a arboles) es relevante a este proyecto porque se utiliza el algoritmo Huffman con el objetivo de mostrar las aplicaciones de árboles de definición jerárquica al compactar caracteres ASCII como cadenas de bits (0's y 1's) que ocuparan menos espacio que dichos caracteres, sólo que el autor lo define como seudocódigo sin llegar a implementar Huffman.

"***JPEG Image compression: Transformation, Quantization and Encoding" de C. Holloway [3]***

Este artículo se relaciona con esta propuesta porque casi al final la autora propone una versión de huffman aplicado a imágenes y realiza una comparación entre dos imágenes una original en HQ y la versión de esta después de haber sido comprimida con Huffman para concluir que ambas tienen casi la misma calidad aunque entre ellas tengan una diferencia de tamaño en disco.

"***Digital Image Encryption Based on RSA Algorithm" de Ali E. Taki El-Deen, El-Sayed A. El-Badawy, Sameh N. Gobran, Egipto [4]***

Este artículo se relaciona con esta propuesta porque se hace una comparación del algoritmo RSA (sistema asimétrico de encriptación) con los algoritmos DES y BLOWFISH (sistemas simétricos de encriptación) aplicando todos estos a encriptar un misma imagen, para después realizar un análisis en el tiempo en el que le toca a cada sistema encriptar y desencriptar dicha imagen, concluyendo que en general el sistema RSA tiene un mejor tiempo de respuesta y mayor seguridad que los otros 2.

"Text Encryption with Huffman Compression" de Nigam Sangwan [5]

Este artículo se relaciona con esta propuesta porque de igual manera el tema objetivo de estudio y análisis es la encriptación y compresión de archivos de texto, si bien se hace uso también del algoritmo Huffman como el medio de compresión la diferencia radica en que hace uso de sistemas de encriptación simétricos y no asimétricos como es el caso del RSA.

Proyectos Terminales

Implementación en software-hardware de aritmética sobre campos finitos binarios F2m en curvas elípticas para aplicaciones criptográficas de llave pública [6]

Este proyecto se relaciona con esta propuesta porque el tema objeto de estudio es la encriptación asimétrica, aunque aquí se hace énfasis en como el uso de curvas elípticas sobre campos binarios podría ser aplicado para construir criptosistemas asimétricos, la diferencia radica en que el proyecto es enteramente implementado sobre el lenguaje de descripción de circuitos VHDL.

Justificación

Dadas las necesidades tecnológicas actuales, resulta impensable realizar transferencias de información a altas velocidades dentro de una red de computadoras sin utilizar algún tipo de compresión y/o encriptado, e indirectamente esto ha generado 2 grandes necesidades:

- Existen algunos casos donde es requisito el poder garantizar que todo el flujo de datos original se pueda recuperar después de un proceso de compresión/descompresión; como ejemplo: bases de datos, hojas de cálculo y algunos audiovisuales donde la necesidad de toda la información sea crítica, es bajo esta razón que se han desarrollado algoritmos con la filosofía de que cualquier pérdida de datos es inaceptable (ejemplo el algoritmo Huffman).
- También se ha vuelto una necesidad que durante una transmisión se garantice la capacidad de mantener inaccesible la información a terceros, para que cuando el destinatario (o destinatarios) reciba la información se pueda afirmar con certeza la integridad de esta (esto es que no se le haya añadido, modificado o eliminado datos), por esta razón se han desarrollado algoritmo de encriptación tales como el RSA (Rivest, Shamir y Adleman).

En conclusión la popularización del internet ha dejado como consecuencia que estas dos necesidades en la transmisión de información sean consideradas problemáticas de nivel global y no simples problemas aislados en una red local, por lo que el desarrollo e implementación de sistemas de compresión/encriptado de la información se ha vuelto de vital importancia en la actualidad.

Objetivos

Objetivo general

Desarrollar un sistema de software que comprima/encripte y descomprima/desencripte archivos de texto usando el algoritmo Huffman y el criptosistema asimétrico RSA.

Objetivos particulares

- Implementar un menú con interfaz gráfica que reciba o genere (depende del usuario) las claves necesarias en el sistema RSA así como el nombre del archivo de texto a ser comprimido y encriptado.
- Implementar un módulo que comprima el archivo mediante el algoritmo Huffman.
- Implementar un módulo que encripte el archivo mediante el sistema RSA.
- Implementar un módulo que descomprima el archivo con descompresión Huffman.
- Implementar un módulo que desencripte el archivo mediante el sistema RSA.

Descripción técnica

El proyecto que se llevará a cabo se compone de los siguientes módulos para poder realizar la compresión/encriptado y la descompresión/desencriptado:

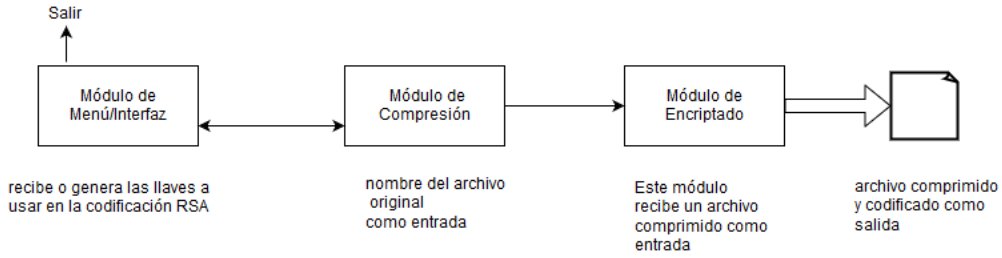
- **Módulo de Menú/Interfaz:** Es el módulo por donde el usuario interactuará con el software, desplegará un menú donde le pedirá al usuario cual acción desea realizar (compresión/encriptado, descompresión/desencriptado o salir) y de acuerdo a la elección del usuario desencadenará alguno de los 2 procesos descritos en la ***figura1*** (o cierra el programa).

  Si es el proceso **a)** entonces le pedirá al usuario si este desea introducir las llaves a usar en el algoritmo RSA o si desea que se generen aleatoriamente por el programa; además le pedirá el nombre del archivo a procesar, posteriormente invocará al *Módulo de Compresión.*

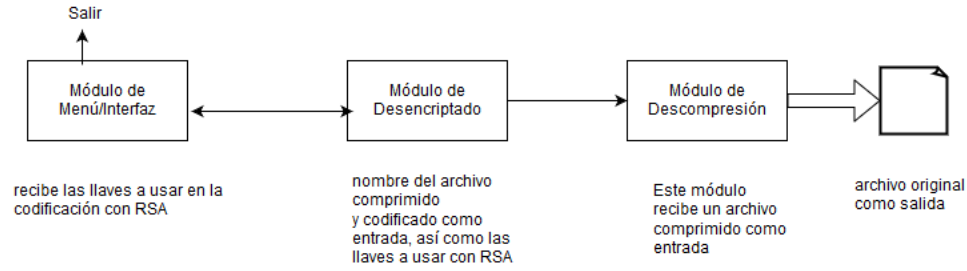
  Si es el proceso **b)** entonces le pedirá al usuario las llaves usadas en la codificación así como el nombre del archivo a procesar para posteriormente llamar al *Módulo de Desencriptado.*
- **Módulo de Compresión**: Es el módulo mediante el cual se llevará a cabo la compresión del archivo, recibe el nombre de un archivo como entrada y lo trata de abrir, si la apertura del archivo es exitosa realiza la compresión del archivo mediante Huffman y manda a llamar al

Módulo de Encriptado (ver figura 1), en caso contrario notifica al usuario que no se pudo abrir el archivo y llama al *Módulo de Menú/Interfaz* (ver figura 1).

- **Módulo de Encriptado:** Es el módulo donde se lleva a cabo la encriptación del archivo, como entrada recibe: un archivo comprimido por el *Módulo de Compresión* (ver figura 1), así como las llaves a usar en el RSA, posteriormente realiza la codificación del archivo con RSA. La salida es un archivo comprimido y encriptado.
- **Módulo de Desencriptado:** es el módulo donde se desencripta el archivo, recibe como entrada las llaves a usar en el RSA y el nombre del archivo a procesar, trata de abrir el archivo y si la apertura fue exitosa realiza la desencriptación del archivo con las llaves recibidas para posteriormente mandar a llamar al *Módulo de Descompresión* (ver figura 1), en caso de que no se pueda abrir notifica al usuario que no se pudo abrir el archivo y llama al *Módulo de Menú/Interfaz*(ver figura 1).
- **Módulo de descompresión:** es el módulo donde se realiza la descompresión del archivo, recibe un archivo comprimido como entrada y realiza la descompresión del mismo. La salida es el archivo de texto original.



a) compresión y encriptado



b) Desencriptación y descompresión

Figura 1: Módulos del sistema

Especificación técnica

Este proyecto se dará por concluido en el momento en que el software sea capaz de comprimir/encriptar y posteriormente descomprimir/desencriptar 10 archivos de texto plano con una extensión de mínimo 2000 palabras de manera exitosa (esto es que se pueda comprobar que el archivo obtenido es idéntico al original).

El archivo comprimido y encriptado obtenido al final del proceso **a)** (ver figura 1) será un archivo de texto plano y el tamaño deberá ser estrictamente menor que el archivo original.

El archivo que comprimido y encriptado que servirá como entrada al proceso **b)** (ver figura 1) será un archivo de texto plano, asimismo el archivo obtenido al final de dicho proceso será otro archivo de texto plano y deberá ser idéntico al original.

Las frecuencias que se usarán para implementar Huffman se basarán en las frecuencias en el idioma Español y el alfabeto a usar serán los 254 caracteres ASCII imprimibles; esto implica que los textos a comprimir deberán estar escritos en idioma español usando como alfabeto los 254 caracteres ASCII imprimibles.

El todos los módulos descritos en la Descripción técnica será enteramente implementado en lenguaje java.

Marco teórico

Compresión y criptografía

La compresión de datos así como la encriptación de estos son dos ramas centrales en las tecnologías de la información que se orienten a la transmisión de datos, ya que por una parte la compresión disminuye la inversión de tiempo y dinero con lo que se obtiene una transmisión de datos más rápida y barata; y por su parte la criptografía permite preservar la privacidad en la transmisión de datos reservando el acceso de la información solo a personas autorizadas.

Compresión sin pérdida de datos (Lossless)

Los algoritmos de compresión del tipo lossless están pensados bajo la premisa de que la totalidad de la información es necesaria, por ende tienen el objetivo de que después de un proceso de compresión/descompresión el flujo de datos obtenido sea igual al original. Sus aplicaciones en el mundo en el que vivimos son variadas, por ejemplo: algunos protocolos de transmisión de información implementan algoritmos lossless para conseguir multiplexar un mayor número de datos por el medio físico, también algunos sistemas de almacenamiento de datos implementan este tipo de algoritmos bajo la idea de que cuando los datos sean descomprimidos conserven toda

la información original, y además que al ser sometidos a un proceso de compresión permitan almacenar la mayor cantidad de datos en un mismo espacio.

Los métodos o algoritmos de compresión lossless suelen ser subdivididos de acuerdo al tipo de dato a comprimir para el que fueron diseñados, por ejemplo:

- Texto
- Imagen
- Sonido

Aunque existen algunos métodos que fueron diseñados con un propósito de compresión más general, por ejemplo el conocido como algoritmo Huffman.

Huffman

El algoritmo Huffman aplicado a archivos de texto tiene su premisa en asignar "códigos" de distinta longitud de bits a cada uno de los caracteres que aparecen en un archivo, asignándole los "códigos" cortos a aquellos caracteres que tengan una mayor presencia en el archivo; ya que si sustituimos los caracteres por dichos "códigos" obtendremos una compresión Huffman. El algoritmo se apoya en la creación de árboles binarios (aunque existen variaciones propuestas de árboles huffman ternarios y cuaternarios).

El algoritmo Huffman funcionalmente depende de obtener los códigos del árbol Huffman, lo cual se logra de la siguiente manera:

1. Crear una cola de prioridad de los caracteres presentes en el archivo, donde la prioridad estará dada por la frecuencia de aparición de cada carácter
2. Crear un nodo árbol/huffman a partir de los 2 caracteres con la menor frecuencia, estos serán las hojas del nodo y este (el nodo) tendrá como identificador un carácter auxiliar o nulo (un carácter que no aparece en el archivo), asimismo la frecuencia de aparición será dada por la suma de las frecuencias de sus hijos.
3. Se retiran de la cola de prioridad las dos estructuras usadas en el paso anterior y se inserta el nodo árbol/huffman en la cola de acuerdo a su frecuencia.
4. Se repite iterativamente el paso 2 y 3 hasta que la cola solo tenga un elemento que representara la raíz de nuestro árbol Huffman
5. El siguiente paso consiste en etiquetar con '0's o '1's cada nodo del árbol asociando por ejemplo los '0' a todos los nodos izquierdos y '1's a todos los derechos. El código asociado a cada hoja del árbol estará dado por el recorrido de la raíz hasta dicha hoja.
6. Para comprimir cada carácter en el archivo se sustituirá por el código (o los bits) asociado a la hoja que represente a dicho carácter.

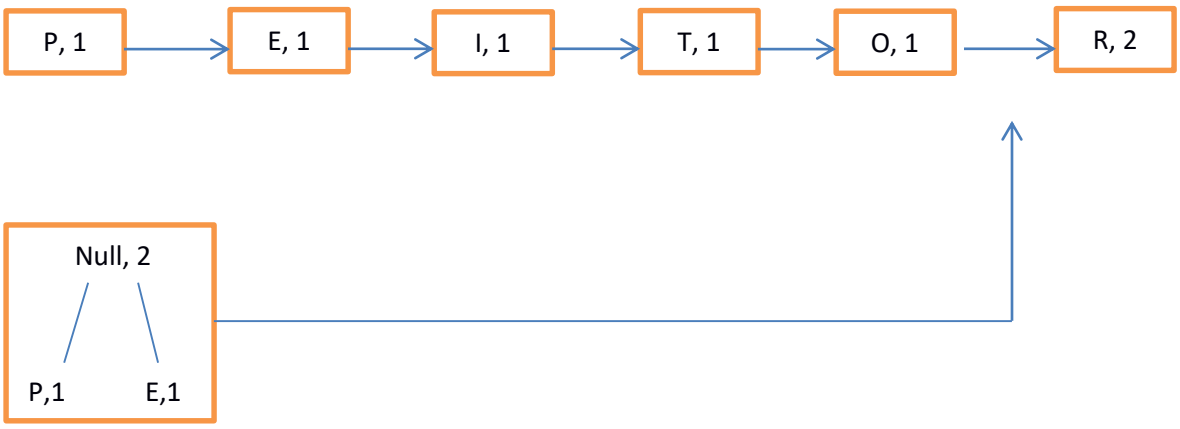
7. Para descomprimir se recorre el árbol (empezando desde la raíz) de acuerdo al flujo de bits hasta llegar una hoja que tendrá el carácter deseado, y así sucesivamente hasta acabar con todo el flujo de bits codificados.

Ejemplo:

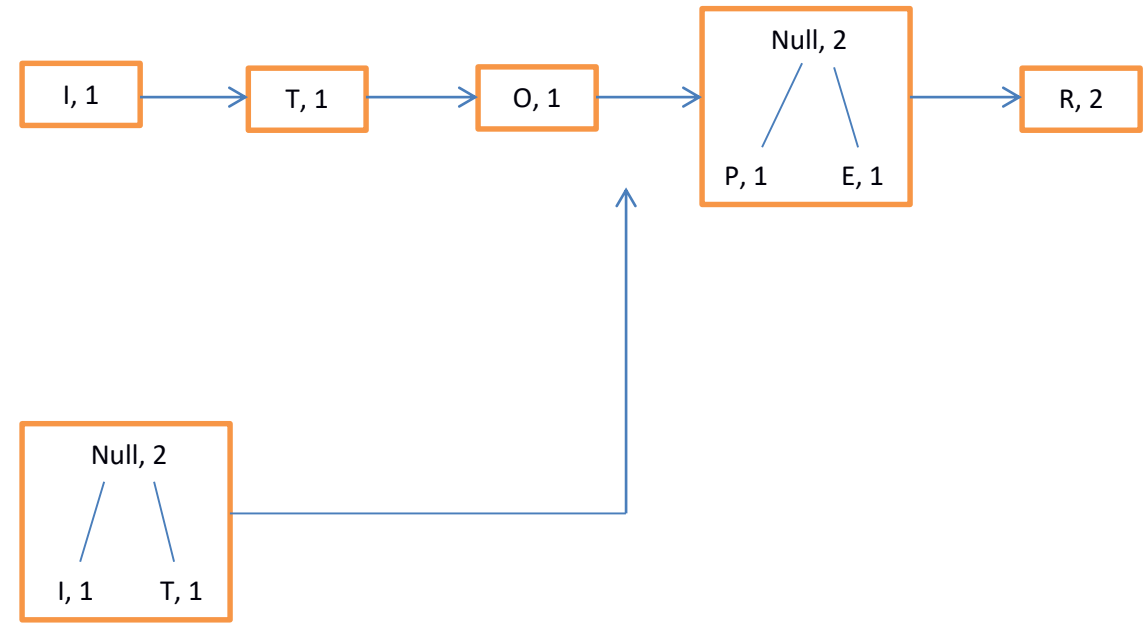
Palabra a comprimir: PERRITO

Carácter	Frecuencia
P	1
E	1
R	2
I	1
T	1
O	1

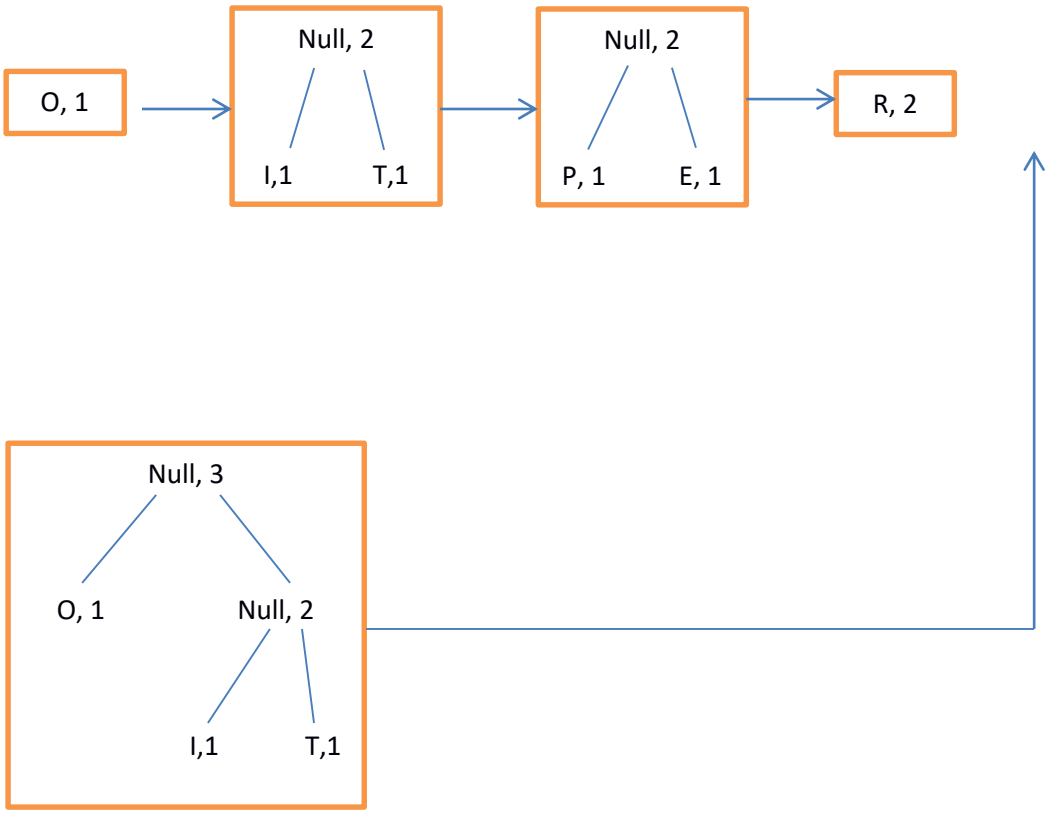
Primera Iteración



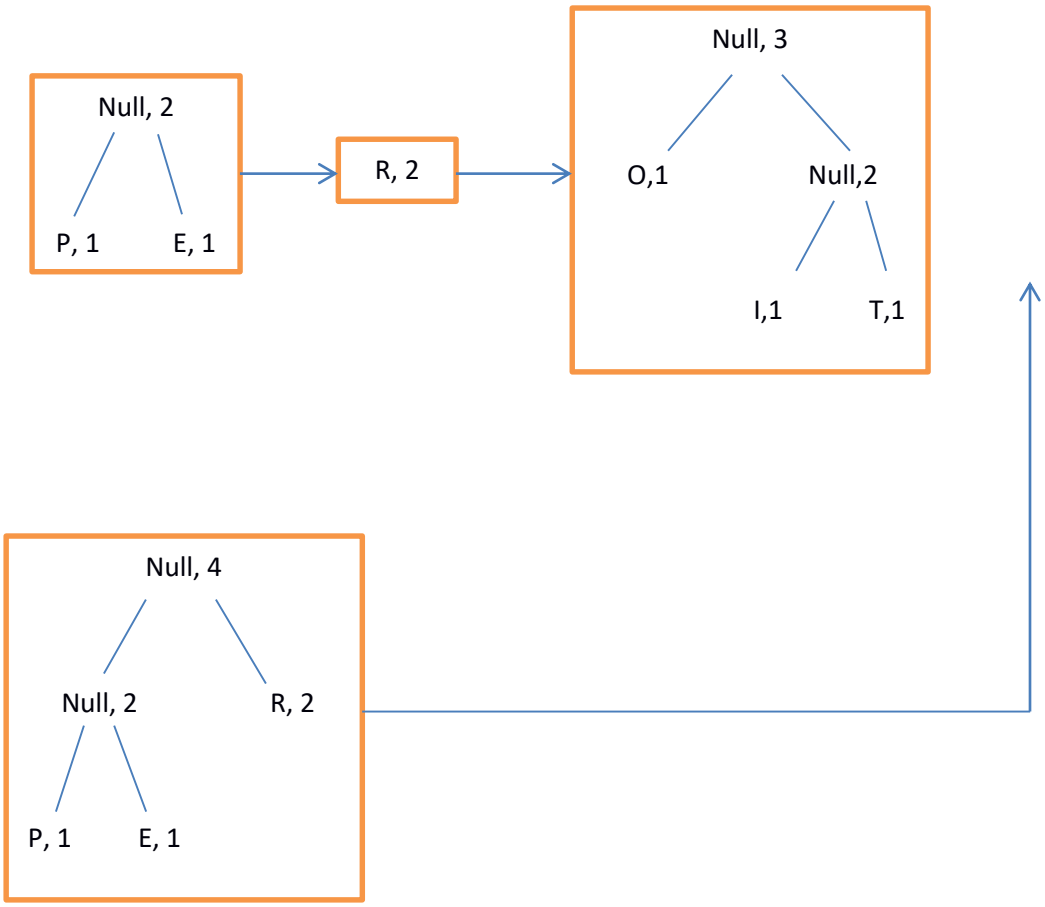
Segunda Iteración



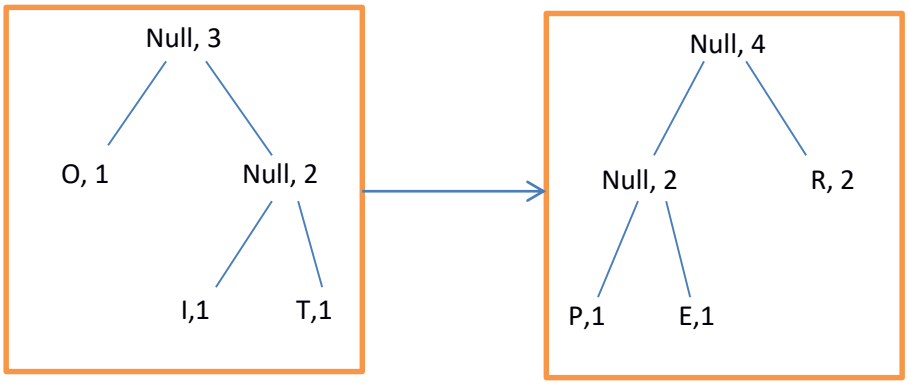
Tercera Iteración

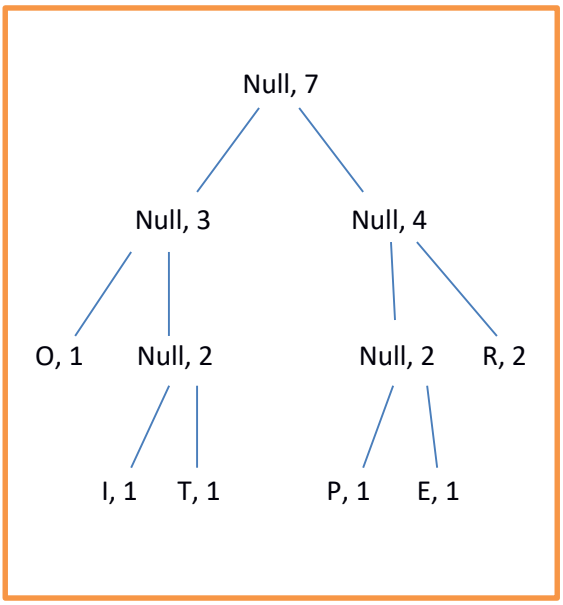


Cuarta Iteración

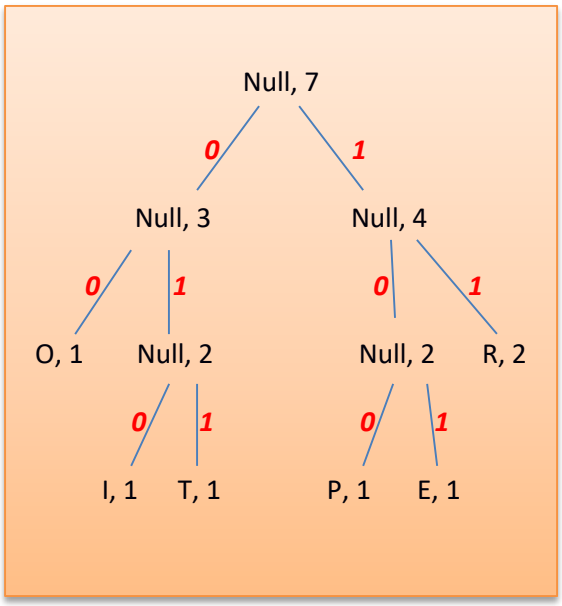


Quinta Iteración





Árbol Huffman



Carácter	Código Huffman Asociado
P	100
E	101
R	11
I	010
T	011
O	00

Palabra PERRITO comprimida con Huffman

P	E	R	R	I	T	O
100	**101**	**11**	**11**	**010**	**011**	**00**

Por lo que la palabra PERRITO se puede comprimir en 18 bits con Huffman

Criptografía

La palabra criptografía etimológicamente viene del griego *Kryptos* que significa esconder y de *graphein* que significa escribir, significando escritura escondida. Pero el significado contextual de la palabra a través de los años ha sido asociado a aquellas técnicas cuyo propósito es el de transmitir mensajes sobre líneas de comunicación inseguras garantizando que solo las personas autorizadas sean capaces de entender el contenido del mensaje.

Encriptación y desencriptación son dos términos asociados a la criptografía; el primero engloba el proceso de "esconder" el mensaje antes de transmitirlo por la línea de comunicación insegura y el segundo refiere al proceso de "mostrar" el contenido del mensaje a la persona autorizada después de recibirlo a través de la línea de comunicación insegura.

La criptografía tiene dos grandes vertientes de estudio la criptografía de clave privada o simétrica y la criptografía de clave pública o asimétrica.

- Criptografía Simétrica: La criptografía simétrica es aquella en la cual los procesos de encriptar y desencriptar se realizan por medio de una misma clave (o llave) que los usuarios emisor y receptor comparten. Ejemplo de esta el Algoritmo DES
- Criptografía Asimétrica**:** Por definición la criptografía asimétrica es aquella que hace uso de dos claves (o llaves) una con el propósito de encriptar llamada clave pública y otra con el propósito de desencriptar llamada clave privada. Ejemplo de esta el algoritmo RSA

RSA

El algoritmo RSA es un algoritmo de encriptación asimétrica desarrollado en 1977 en el MIT por Ronald Rivest, Adi Shamir y Leonard Adelman. La idea detrás del algoritmo es aprovechar la complejidad computacional que representa factorizar un número entero ***n*** (complejidad conocida como Problema de Factorización Entera o PFE) del cual se sabe es producto de dos números primos ***p, q*** (***n=p*q***); con los cuales se obtienen las claves: la clave pública es el numero ***n*** y la clave privada es la pareja ***p***, ***q***.

El algoritmo opera de la siguiente manera es:

1) A cada usuario se le asigna un número entero ***n*** como clave pública.
2) Solamente el usuario conoce la factorización ***(p, q)*** de su respectiva ***n***, que mantendrá secreta y será su respectiva clave privada.
3) existe un directorio o repositorio de claves públicas.
4) si alguien desea mandar un mensaje ***m*** a algún usuario entonces obtiene su clave pública ***n*** del directorio y le manda el mensaje cifrado con la formula ***c=m^e mod(n)***.
5) el mensaje cifrado c viaja por el medio de comunicación inseguro.
6) cuando el mensaje arriba al destinatario este procede a descifrarlo con la formula ***m= c^d mod(n)***, ***d*** es un número equivalente a la pareja ***(p, q).***

La forma de obtener las claves públicas es:

1) Se generan dos números primos grandes a los que llamaremos ***p*** y ***q***.
2) Se calcula ***n*** como producto de ***p*** y ***q***, esto es ***n=p*q***.
3) Se calcula un numero ***fi(n)= (p-1) (q-1)***.
4) Se calcula un número natural ***e*** tal que el *Máximo Común Divisor* de ***e*** y ***fi(n)*** sea igual a 1; ***MCD (e, fi(n))=1***, es decir e debe ser primo relativo de fi(n).
5) Se calcula ***d= ((i*fi(n))+1)/e*** para i=1, 2,3,… hasta que ***((i*fi(n))+1)/e*** de como residuo 0.
6) El par ***(e, n)*** es la clave publica
7) El par ***(d, n)*** es la clave privada
8) Para cifrar se usa la formula ***c=m^e mod(n)***.
9) Para descifrar se usa ***m= c^d mod(n)***.

El algoritmo funciona porque ***e*** y ***d*** son inversos multiplicativo modulo ***fi(n),*** esto significa que:

c^d= (m^e) ^d mod(n) = => c^d=m^ed mod(n)

De acuerdo a la forma en que hemos elegido ***d*** y ***e*** se tiene que ***ed=1+ Kfi(n)***, por lo que:

m^ed=m^1+kfi(n) ==> m^ed=m (m^fi(n)) ^k=m mod(n)

Lo último se obtiene directamente del teorema de Euler cuando ***m*** es coprimo con ***n*** y en general puede demostrarse que las ecuaciones se cumplen para todo ***m*** usando congruencias y el teorema chino del resto. Por lo que finalmente tenemos que:

m= c^d mod(n)

Ejemplo:

- Escogemos los números primos ***p=3*** y ***q=11*** para simplificar las operaciones
- Obtenemos ***n=pq***
  n=3*11=33
- ***fi(n)*** se obtiene como ***fi(n)=(p-1)(q-1)***
  fi(n)= (3-1) (11-1)=20
- ***e*** se obtiene de tal manera que cumpla que el ***MCD(e,fi(n))=1*** por lo que en este caso ***e=3***
- Se calcula ***d= ((i*fi(n))+1)/e*** para i=1, 2,3,… hasta que ***((i*fi(n))+1)/e*** de como residuo 0.
  d = ((i* fi(n)) + 1) / e = (i * 20 + 1) / 3 = 21 / 3 = 7
- La pareja ***(3, 33)*** es la clave pública
- La pareja ***(7, 33)*** es la clave privada
- ***Encriptando: Mensaje = 5, C = M^e mod n = 5^3 mod 33 = 26***
- ***Desencriptando: M = C^d mod n = 26^7 mod 33 = 8031810176 mod 33 = 5***

Desarrollo del proyecto

El proyecto se realizó en una computadora con sistema operativo Lubuntu (una distribución ligera de Ubuntu) con 4G de RAM, 1T de disco duro y un procesador AMD x64, además como entorno de desarrollo java se usó Netbeans 8.1 con jdk8.

La primera etapa del proyecto fue el de diseñar e implementar los módulos de compresión y descompresión, para ello se consultó los capítulos referentes a compresión con Huffman de *Data compression the complete reference [7]* y *The data compression book [8]*, una vez asimilada la forma en que trabaja el algoritmo Huffman, procedí a diseñar e implementar el módulo de compresión.

Módulo de compresión (apéndice a): Lo primero de este módulo fue diseñar el nodo de frecuencias (a.1) que trabaja en la cola de frecuencias y el árbol de codificación Huffman, lo segundo fue diseñar la cola, el árbol y la forma en que estos se creaban; para posteriormente diseñar e implementar la parte de obtener las frecuencias del archivo nombre.txt original, hacer que estas se guardaran en el árbol y finalmente te creara el archivo comprimido nombre_com.txt con huffman (a.2), esto se puede ver en la imagen 1 e imagen 2.

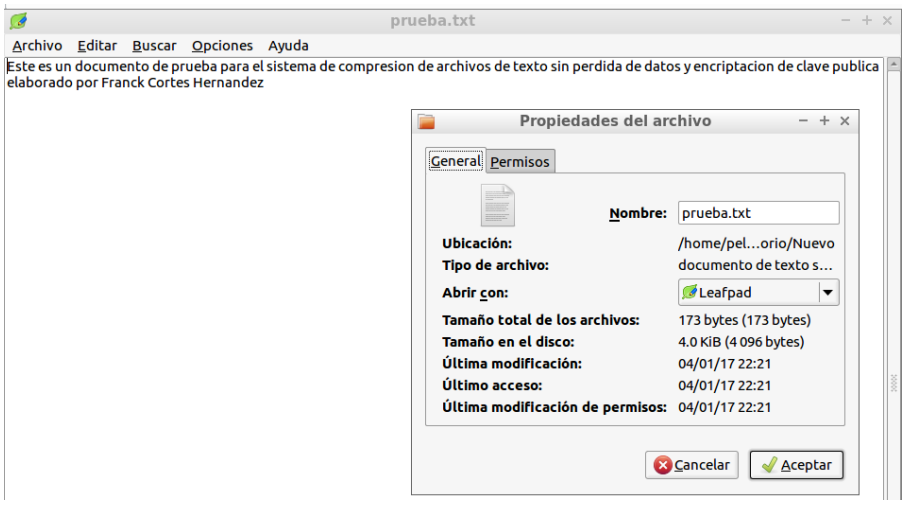


Figura 2: archivo original

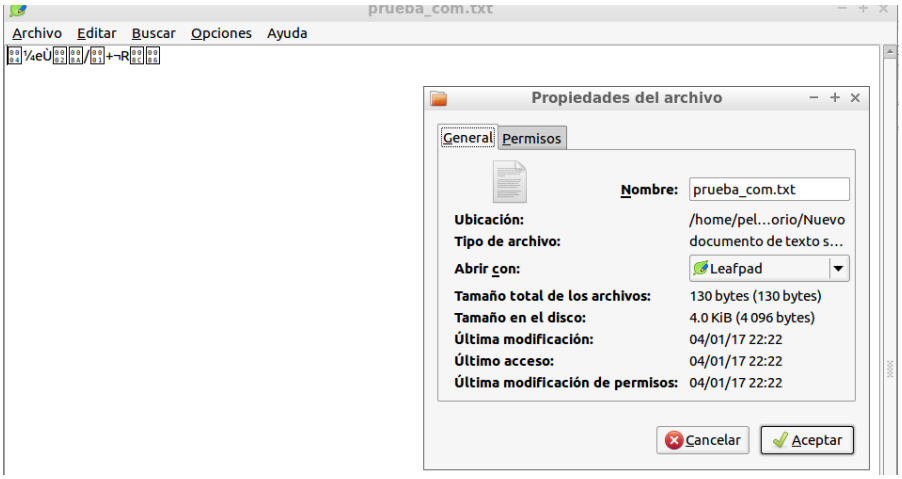


Figura 3: archivo comprimido mediante Huffman

Módulo de descompresión (apéndice b): Una vez que el módulo de compresión estuvo implementado me dispuse a diseñar e implementar el módulo de descompresión, primero diseñe la parte que recibiendo una cadena comprimida te devolviera la cadena descomprimida, después diseñe e implemente la parte que recibiendo un archivo nombre_com.txt y la raíz de un árbol Huffman te obtuviera la cadena comprimida de dicho archivo y por medio del árbol te devolviera la cadena original, para posteriormente guardarla en el archivo nombre_desc.txt que tendría que ser igual al archivo original (b.1), esto se puede observar en Image3 e Imagen4.

Llegado a este punto tuve problemas ya que algunos archivos nombre.txt y nombre_desc.txt no coincidan en algunas letras, las cuales me di cuenta correspondían (en el archivo original) a ciertos elementos específicos del ASCII extendido (áàäéèëíìïóòöúùuñÁÀÄÉÈËÍÌÏÓÒÖÚÙÜÑçÇ), esto descubrí se debió a mi sistema operativo y al hecho de que los archivos .txt el sistema trabaje bajo Unicode. Resolví esto diseñando e implementando un código que me quitara estos “caracteres especiales” y los sustituyera por caracteres ASCII equivalentes (apéndice a.3) aceptados por Lubuntu.

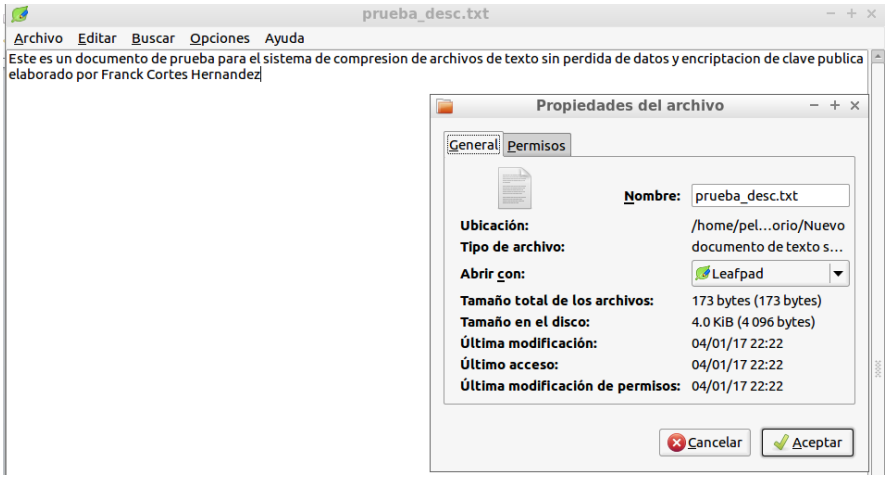


Figura 4: archivo descomprimido

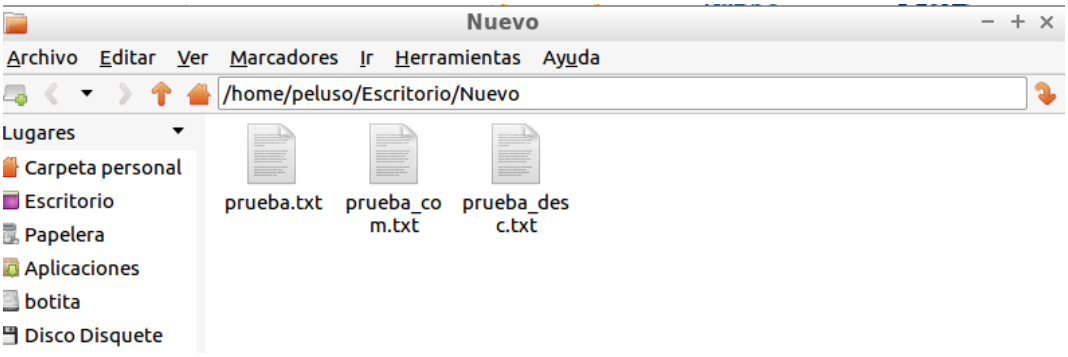


Figura 5 : Archivo obtenidos al final de comprimir/descomprimir con Huffman

La siguiente etapa del proyecto consistió en desarrollar los módulos de encriptado y desencriptado; primero consulte información referente al sistema de encriptación asimétrico RSA en *Introduction to cryptography [9]*, una vez asimilada la información de cómo funcionaba el sistema, consulte en *The first book of compression and encryption [10]* información de cómo un sistema de encriptado podía integrarse a un sistema de compresión.

De acorde a lo asimilado se desarrollaron e implementaron los dos siguientes módulos:

Módulo de encriptado (apéndice c): Lo primero en este módulo fue desarrollar en seudocódigo los métodos o cálculos pertinentes que nos permitieran obtener los valores de las claves pública y privada (estos son el cálculo de MCD, obtención de números primos, el de la exponenciación modular rápida y el de checar que los valores obtenidos en las claves no permitieran encriptar cada uno de los caracteres ASCII imprimibles) y se implementaron a través del método obtener_claves de la clase Encriptar(c.1). Posteriormente se implementó la función que teniendo como entrada una cadena llamara a obtener_claves para inicializar las claves públicas, después procediera a usar la función de encriptado (en este caso *C = M^e mod n)* y devolver la cadena encriptada.

Módulo de desencriptado (apéndice d): En el desarrollo de este módulo lo primero fue diseñar e implementar la función de desencriptado que teniendo la clave privada y la cadena encriptada, devolviera la original a través de la función de desencriptado (*M = C^d mod n*).

Una vez con los dos módulos (encriptado y desencriptado) implementados y funcionando correctamente llego el momento de integrarlos con el sistema de compresión/descompresión, la manera de implementarlo fue la siguiente: en el módulo de encriptado se tenía que la cadena a encriptar se obtenía a través del archivo nombre_com.txt (que es el archivo con la cadena comprimida) y la cadena encriptada se guardaba en el archivo nombre_com_encrip.txt (en este archivo la cadena esta comprimida y encriptada), a su vez el módulo de desencriptado obtenía la cadena encriptada de nombre_com_encrip.txt y devolvía la cadena desencriptada en nombre_com_desencrip.txt (guarda la cadena comprimida y desencriptada) y finalmente el

módulo de descompresión obtenía su entrada de nombre_com_desencrip.txt y su salida la guardaba en nombre_desc.txt (Imagen5).

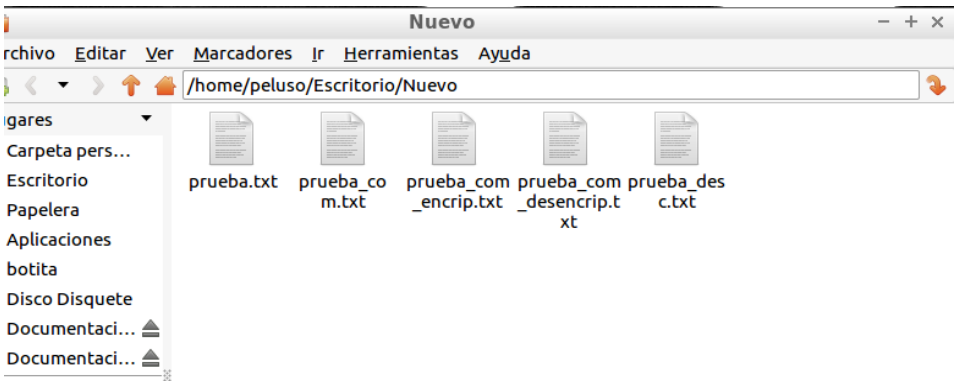


Figura 6: archivos obtenidos de comprimir/encriptar desencriptar/descomprimir

Llegado a este punto había que implementar el módulo de menú/interfaz gráfica, se implementó una primera versión pero sentí que esta estaba muy sobrecargada (todas las instanciaciones las realizaba desde la misma clase principal, además que esta también implementaba al menú e instanciaba los elementos de la misma interfaz); por lo que decidí optar por una nueva forma de trabajo entre los módulos, que consistió en aplicarle pequeños cambios a los módulos para acoplarlos al patrón de arquitectura de software conocido como MVC (modelo-vista-controlador).

El MVC a grandes rasgos es un modelo de arquitectura de software que tiene tres componentes

- **Modelo**: este contiene y encapsula el núcleo de la aplicación, todo esto independientemente de la vista y controlador
- **Vista:** es la presentación de la aplicación y puede acceder al modelo pero solo a modo consulta.
- **Controlador:** es la parte del sistema que reaccionando a las peticiones del cliente modifica el estado del modelo.

Los cambios fueron mínimos; básicamente consistieron en tomar los módulos de compresión, encriptado, desencriptado y descompresión como elementos de modelo, el módulo de menú/interfaz como parte de la vista y gestionar la comunicación entre ellas a través de dos clases almacenadas en controlador que son las siguientes:

Módulo de compresión/encriptado (apéndice e) En esta clase lo que realizo es la instanciación e inicializaciones o actualizaciones de las variables de los módulos comprimir y encriptar.

Módulo de desencriptado/descompresión (apéndice f) En esta clase lo que realizo es la instanciación e inicializaciones o actualizaciones de las variables de los módulos descomprimir y desencriptar.

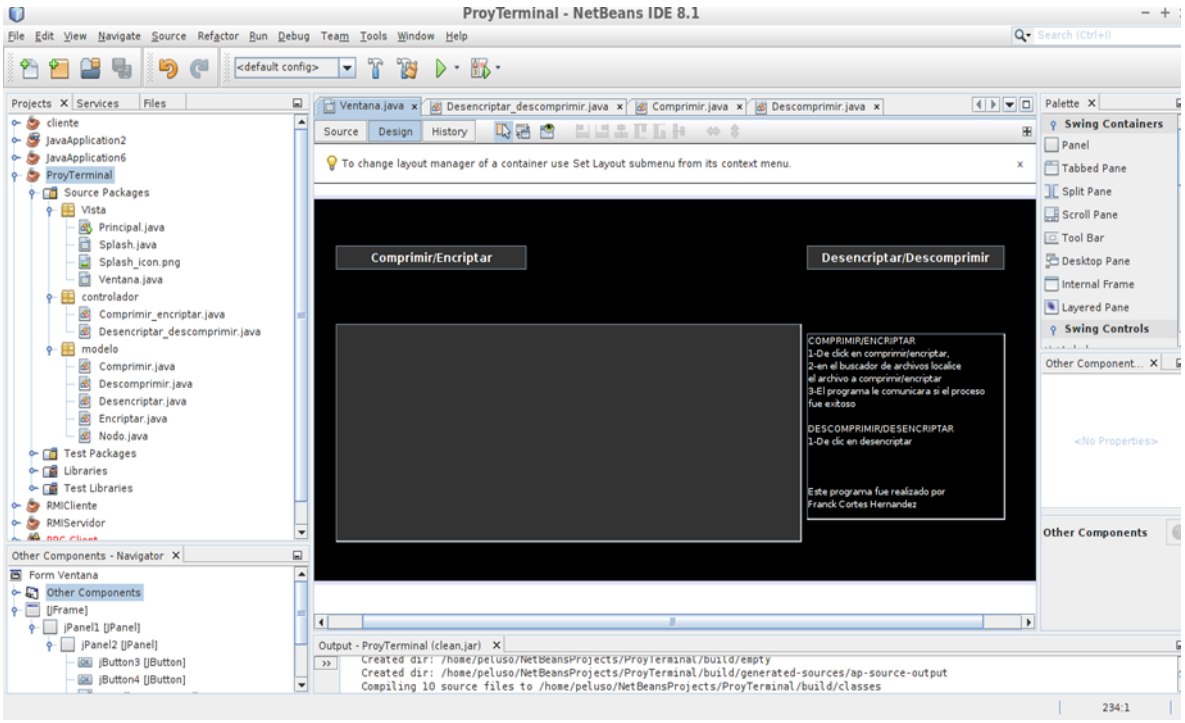


Figura 7: MVC aplicado al sistema

Antes de empezar con las pruebas de compresión/descompresión y encriptado/desencriptado había que definir los textos en los que se realizarían dichas pruebas, ya que estos tendrían que ser de libre distribución para no infringir leyes relacionadas derechos de autor o algo por el estilo; para evitar esto utilice los siguientes textos de libre distribución alojados en la página www.loscuentos.net:

Nombre del archivo: texto01.txt
Lo que le paso a la tía Erika [11]
Palabras: 2316

Nombre del archivo: texto02.txt
Simona [12]
Palabras: 2121

Nombre del archivo: texto03.txt
Se Me Olvido Decirte. . . [13]
Palabras: 3027

Nombre del archivo: texto04.txt
Su Majestad, La Reina Virginia [14]
Palabras: 2607

Nombre del archivo: texto05.txt
El Cielo Y El Infierno Me Sostienen. [15]
Palabras: 2090

Nombre del archivo: texto06.txt
Yin Yang [16]
Palabras: 2571

Nombre del archivo: texto07.txt
La Casa De Los Méndez [17]
Palabras: 3708

Nombre del archivo: texto08.txt
Pánico [18]
Palabras: 2245

Nombre del archivo: texto09.txt
NDEE - Cap 4 [19]
Palabras: 2420

Nombre del archivo: texto10.txt
Fatum [20]
Palabras: 3016

Ya como último realice las pruebas (Imagen7) y los resultados y conclusiones se describen a continuación.

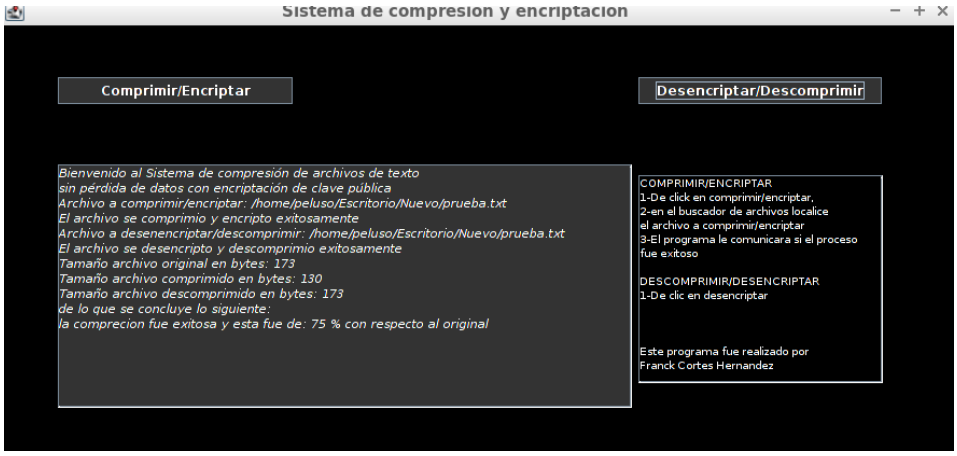


Figura 8: Prueba del Sistema

Resultados

Primeros resultados emitidos por el sistema con una frecuencia promedio

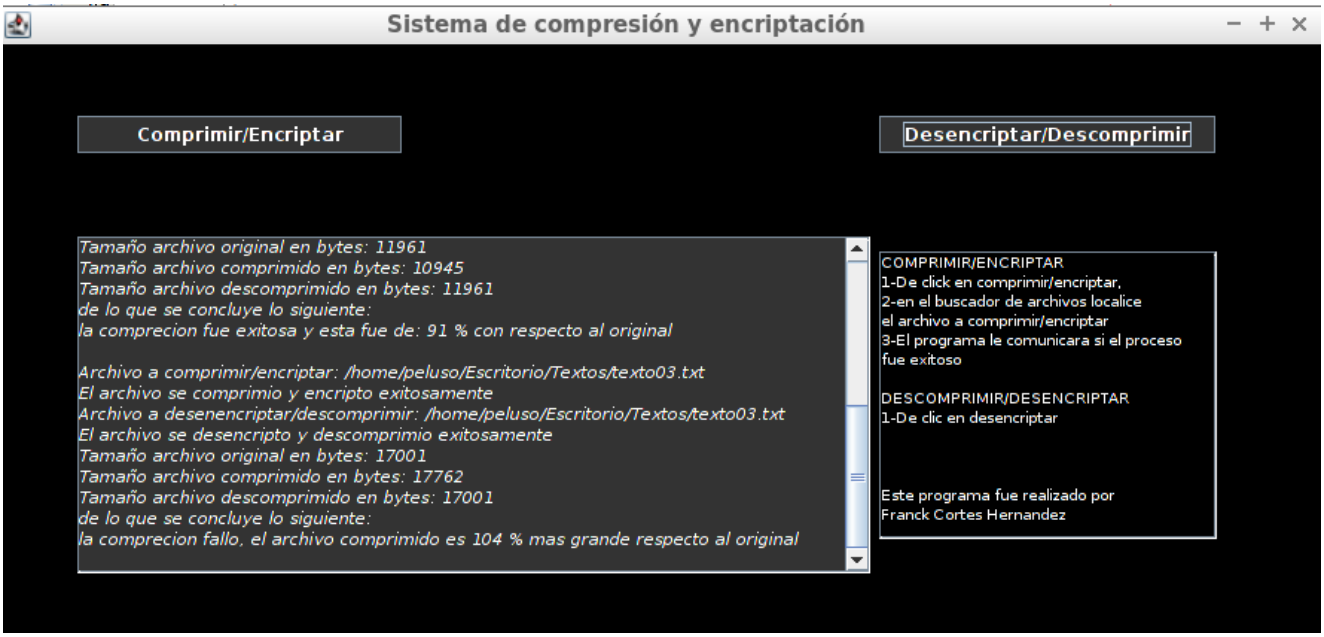


Figura 9: primera prueba del sistema

Bienvenido al Sistema de compresión de archivos de texto
Sin pérdida de datos con encriptación de clave pública
Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 13216
Tamaño archivo comprimido en bytes: 10845
Tamaño archivo descomprimido en bytes: 13216
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 82 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto02.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto02.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 11961
Tamaño archivo comprimido en bytes: 10945
Tamaño archivo descomprimido en bytes: 11961
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 91 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto03.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto03.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 17001
Tamaño archivo comprimido en bytes: 17762
Tamaño archivo descomprimido en bytes: 17001
De lo que se concluye lo siguiente:
La compresión fallo, el archivo comprimido es 104 % más grande respecto al original

Nombre archivo	tamaño del archivo original	tamaño del archivo comprimido	tamaño del archivo descomprimido	% compresión
texto01.txt	13216	10845	13216	82.0596247
texto02.txt	11961	10945	11961	91.50572695
texto03.txt	17001	17772	17001	***104.5350274***

Tabla 1: Datos de la primera prueba

El promedio de compresión con los datos obtenidos de la primera prueba es de 92.7001263 %

Resultados obtenidos con frecuencias individuales

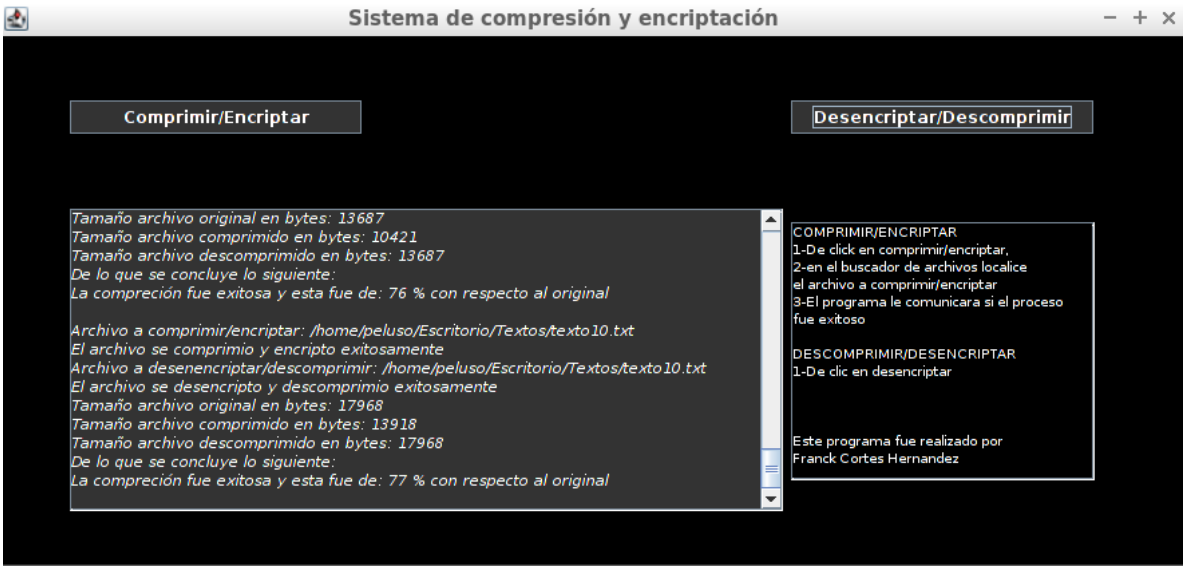


Figura 10: pruebas con frecuencias individuales

Bienvenido al Sistema de compresión de archivos de texto
sin pérdida de datos con encriptación de clave pública
Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 13216
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo descomprimido en bytes: 13216
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 77 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto02.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto02.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 11961
Tamaño archivo comprimido en bytes: 9201
Tamaño archivo descomprimido en bytes: 11961
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 76 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto03.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto03.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 17001
Tamaño archivo comprimido en bytes: 13215
Tamaño archivo descomprimido en bytes: 17001
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 77 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto04.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto04.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 15089
Tamaño archivo comprimido en bytes: 12244
Tamaño archivo descomprimido en bytes: 15089
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 81 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto05.txt
El archivo se comprimió y encripto exitosamente

Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto05.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 12464
Tamaño archivo comprimido en bytes: 9750
Tamaño archivo descomprimido en bytes: 12464
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 78 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto06.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto06.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 14234
Tamaño archivo comprimido en bytes: 11009
Tamaño archivo descomprimido en bytes: 14234
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 77 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto07.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto07.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 20710
Tamaño archivo comprimido en bytes: 15789
Tamaño archivo descomprimido en bytes: 20710
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 76 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto08.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto08.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 11704
Tamaño archivo comprimido en bytes: 8905
Tamaño archivo descomprimido en bytes: 11704
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 76 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto09.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto09.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 13687
Tamaño archivo comprimido en bytes: 10421
Tamaño archivo descomprimido en bytes: 13687
De lo que se concluye lo siguiente:

La compresión fue exitosa y esta fue de: 76 % con respecto al original

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto10.txt
El archivo se comprimió y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto10.txt
El archivo se desencripto y descomprimió exitosamente
Tamaño archivo original en bytes: 17968
Tamaño archivo comprimido en bytes: 13918
Tamaño archivo descomprimido en bytes: 17968
De lo que se concluye lo siguiente:
La compresión fue exitosa y esta fue de: 77 % con respecto al original

Nombre archivo	tamaño del archivo original	tamaño del archivo comprimido	tamaño del archivo descomprimido	% compresión
texto01.txt	13216	10298	13216	77.92070218
texto02.txt	11961	9201	11961	76.92500627
texto03.txt	17001	13215	17001	77.73072172
texto04.txt	15089	12224	15089	81.01265823
texto05.txt	12464	9750	12464	78.22528883
texto06.txt	14234	11009	14234	77.34298159
texto07.txt	20710	15789	20710	76.23853211
texto08.txt	11704	8905	11704	76.08509911
texto09.txt	13687	10421	13687	76.13794111
texto10.txt	17968	13918	17968	77.45992876

Tabla 2: Datos de la segunda prueba

El promedio de compresión de la segunda prueba es de 77.507886 %

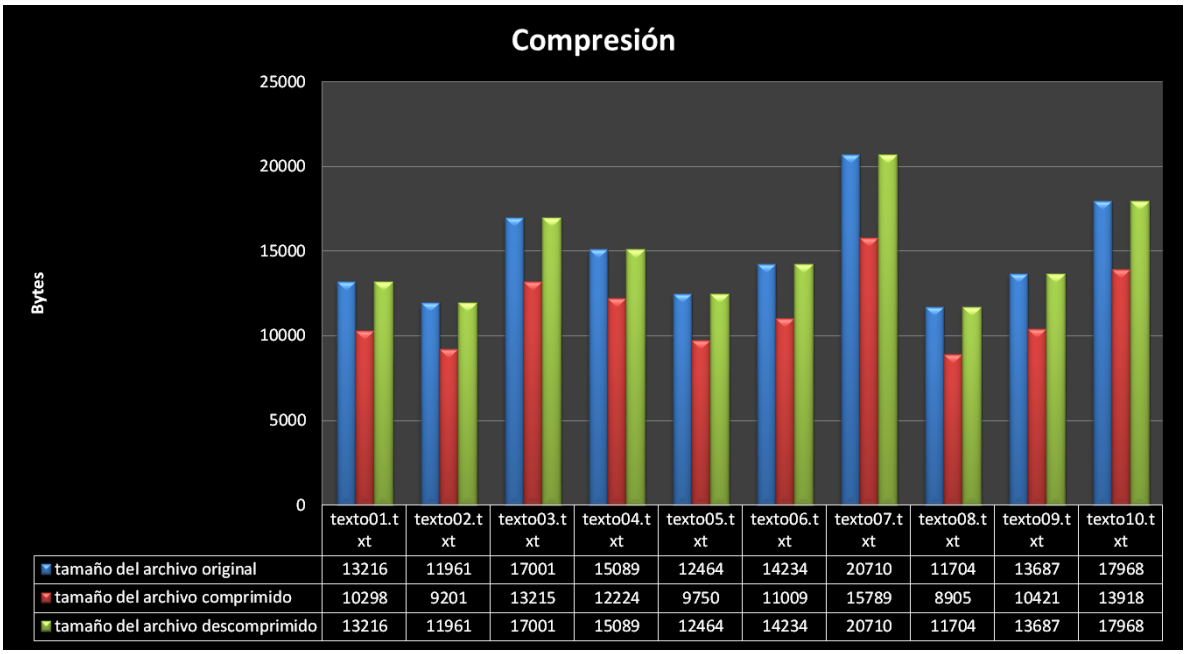


Figura 11: grafica de compresión obtenida

Resultados de redundancia obtenida por la encriptación

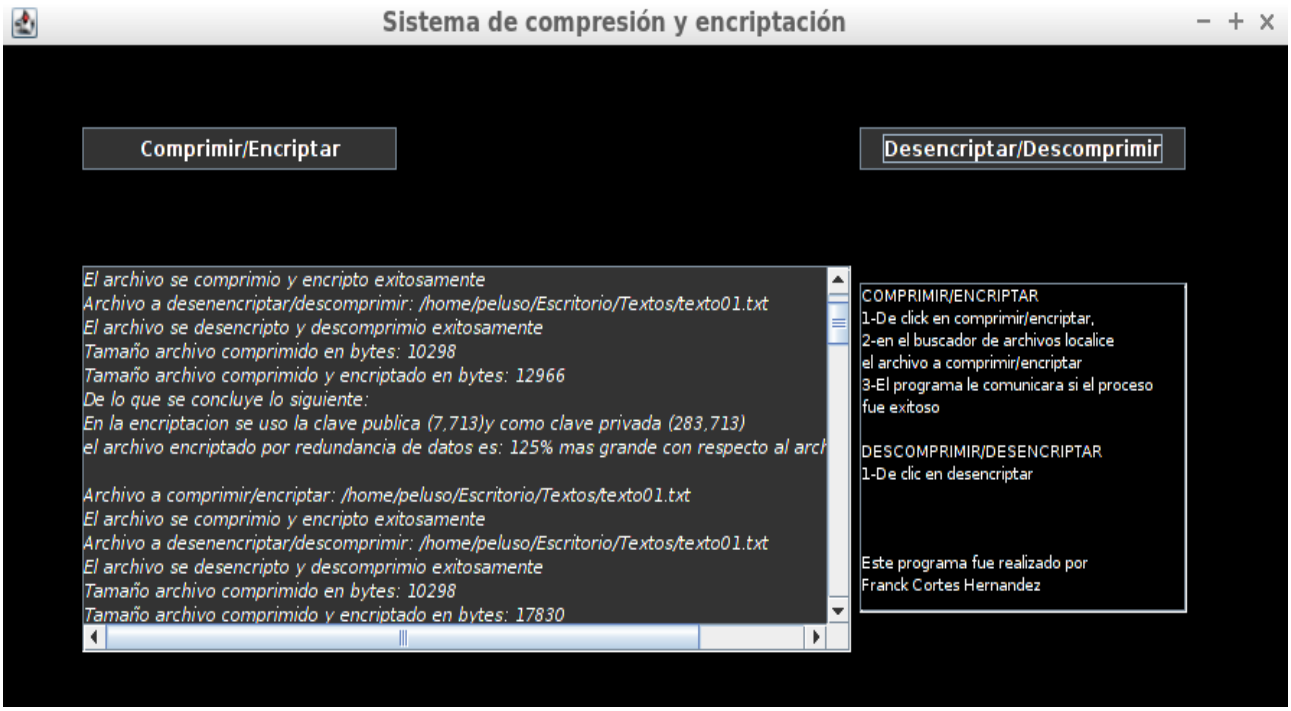


Figura 12: Prueba de redundancia obtenida

Bienvenido al Sistema de compresión de archivos de texto
sin pérdida de datos con encriptación de clave pública
Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 12966
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (7,713)y como clave privada (283,713)
el archivo encriptado por redundancia de datos es: 125% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 17830
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (5,4891)y como clave privada (1901,4891)
el archivo encriptado por redundancia de datos es: 173% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 17991
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (5,5293)y como clave privada (3089,5293)
el archivo encriptado por redundancia de datos es: 174% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 17830
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (5,4891)y como clave privada (1901,4891)
el archivo encriptado por redundancia de datos es: 173% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente

Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 12966
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (7,713)y como clave privada (283,713)
el archivo encriptado por redundancia de datos es: 125% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 12966
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (7,713)y como clave privada (283,713)
el archivo encriptado por redundancia de datos es: 125% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 15452
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (3,2773)y como clave privada (1779,2773)
el archivo encriptado por redundancia de datos es: 150% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 17173
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (5,3953)y como clave privada (2297,3953)
el archivo encriptado por redundancia de datos es: 166% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 18155
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (11,5609)y como clave privada (3971,5609)
el archivo encriptado por redundancia de datos es: 176% mas grande con respecto al archivo comprimido

Archivo a comprimir/encriptar: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se comprimio y encripto exitosamente
Archivo a desenencriptar/descomprimir: /home/peluso/Escritorio/Textos/texto01.txt
El archivo se desencripto y descomprimio exitosamente
Tamaño archivo comprimido en bytes: 10298
Tamaño archivo comprimido y encriptado en bytes: 12367
De lo que se concluye lo siguiente:
En la encriptacion se uso la clave publica (3,493)y como clave privada (299,493)
el archivo encriptado por redundancia de datos es: 120% mas grande con respecto al archivo comprimido

# prueba sobre texto_01.txt	Clave publica usada	Clave privada usada	Aumento de tamaño por redundancia en %
1	(7,713)	(283,713)	125
2	(5,4891)	(1901,4891)	173
3	(5,5293)	(3089,5293)	174
4	(5,4891)	(1901,4891)	173
5	(7,713)	(283,713)	125
6	(7,713)	(283,713)	125
7	(3,2773)	(1779,2773)	150
8	(5,3953)	(2297,3953)	166
9	(11,5609)	(3971,5609)	176
10	(3,493)	(299,493)	120

Tabla 3: Datos de la tercera prueba

El promedio de redundancia en la tercera prueba es de 150.7 %

Análisis de resultados

En la primera prueba usando una frecuencia general para Huffman resulto que el texto_03.txt aumentaba su tamaño 4% respecto al archivo original y por ende fue una prueba fallida ya que el menos un texto no se puede comprimir con esta frecuencia general.

En la segunda prueba se utilizó las frecuencias propias de cada texto para aplicar Huffman, con los siguientes resultados:

- El promedio de compresión en los textos fue del 77.5 %
- El mayor porcentaje de compresión obtenido fue en el texto08.txt comprimiéndolo a 76.08 % de su tamaño original
- El menor porcentaje de compresión obtenido fue en el texto04.txt comprimiéndolo a 81.01 % de su tamaño original
- Tres textos (texto07, texto08, texto09) lograron una compresión cercana al 76% con respecto a sus tamaños de archivo originales
- Tres textos (texto02, texto06, texto10) lograron una compresión cercana al 77% con respecto a sus tamaños de archivo originales
- Tres textos (texto01, texto03, texto05) lograron una compresión cercana al 78% con respecto a sus tamaños de archivo originales
- Un texto (texto04) obtuvo una compresión cercana al 81 % con respecto a su tamaño de archivo original

En la tercera prueba se midió el % de redundancia obtenido al encriptar, para esto se sometió archivo *texto_01.txt* (se escogió este archivo porque su compresión obtenida de 77.92% se asemeja al promedio de compresión obtenido con los diez textos de 77.5%) a los procesos de compresión/encriptado y descompresión/desencriptado durante diez veces para tener un parámetro con el que medir dicha redundancia de datos. De dicha prueba se obtuvo lo siguiente:

- La menor redundancia obtenida del archivo compreso/encriptado respecto al archivo comprimido fue del 120% (decima prueba sobre *texto_01.txt*)
- La mayor redundancia obtenida del archivo compreso/encriptado respecto al archivo comprimido fue del 176 % (novena prueba sobre *texto_01.txt*)
- La redundancia promedio fue del 150.7%
- Durante la segunda prueba el archivo *texto_01.txt* obtuvo una compresión del 77.92%, además la redundancia promedio del mismo archivo es de 150.7% por lo que esto significa que globalmente el tamaño del archivo (texto_01.txt) comprimido y encriptado estaría cercana al 117.42544% con respecto al archivo original

Conclusiones

Durante la primera prueba utilice como frecuencia para Huffman el promedio de las frecuencias de todos los textos, pero con los resultados obtenidos me di cuenta que no podía usar una frecuencia general del español (tal y como propuse en la especificación técnica), ya que la frecuencia con la que aparecen algunas letras en el español latinoamericano no son las mismas que en las que aparecen en el español de España por ejemplo, y esto causaba que al final no pudiera apreciarse las ventajas de la compresión mediante Huffman todos los textos; ya que como se observa durante la prueba con el archivo llamado *"texto03.txt"* la compresión fue negativa ya que el archivo comprimido era 4% más grande que el archivo original, por lo cual tenemos al menos un archivo que no puede ser comprimido usando la frecuencia promedio de todos los archivos. Por lo que tome la decisión de que en la versión final de este sistema las frecuencias a utilizar en cada texto son las propias de cada texto.

Por otro lado durante la segunda prueba se puede observar que el algoritmo Huffman aplicándolo a textos escritos en el idioma español es una herramienta informática bastante poderosa; ya que con su ratio general de compresión cercano al 77% aunado a una dificultad no tan alta que se requiere para implementar en algún lenguaje de alto nivel la hacen excelente para integrase en otros sistemas informáticos más complejos, como por ejemplo en sistemas de comunicación, de protección de información, de almacenamiento de datos, etc. Pero así mismo observe que el algoritmo no es perfecto y tiene contras, por ejemplo que requiere un costo computacional y de tiempo, ya que para realizar la compresión necesita obtener el árbol de frecuencias primero; lo cual quiere decir que requiere leer el archivo completamente al menos una vez antes de comprimir y esto en máquinas con pocos recursos y que requieran comprimir archivos muy grandes significa un gasto computacional y de tiempo; además para descomprimir requiere tener acceso al árbol Huffman, esto significa que forzosamente tenemos que almacenar el árbol en alguna estructura, base de datos o archivo; esto en sistemas donde el tiempo de respuesta sea importante podría suponer un problema por los tiempos de acceso a memoria, o por el tiempo de respuesta de una base de datos externa, etc.

Por ultimo en la última prueba se midió el % que aumentaba el tamaño del archivo por la redundancia de datos, en esta se obtuvo que el promedio de redundancia generada por la encriptación es del 150.7%, esto a mi parecer es un buen ratio de redundancia ya que recordemos que esta redundancia es sobre el archivo comprimido (no sobre el archivo original) por lo que si tomamos en cuenta que el texto_01 obtuvo una compresión de 77.92% esto implica que en promedio el texto_01 al ser comprimido mediante Huffman y encriptado con RSA genera un archivo 17% más grande que el archivo original lo cual yo considero como redundancia más que aceptable para un sistema de encriptación.

Finalmente concluyo que el sistema RSA y el algoritmo de compresión Huffman son dos herramientas informáticas que pueden trabajar en conjunto como un sistema de Compresión y Encriptación de archivos o datos, generando una buena entidad informática de seguridad; ya que como vimos el promedio de redundancia de datos generado por el sistema Huffman/RSA es cercano al 117% respecto al archivo original hacen al sistema deficiente como sistema de compresión , pero sumamente eficiente como sistema de encriptación; esto sumado a que la implementación del sistema no es necesariamente complicada en lenguajes de alto nivel hacen que como herramienta de seguridad el sistema Huffman/RSA pueda ser etiquetado como aceptable en entornos donde se requiera una seguridad baja/media.

Bibliografía

[1] K. Leon Lomparte, "ENCRIPTACION RSA DE ARCHIVOS DE TEXTO", Doctorado, PONTIFICIA UNIVERSIDAD CATOLICA DEL PERU, 2005.

[2]R. Johnsonbaugh, "Capitulo 9: Arboles", en Matemáticas Johnsonbaugh, Ed. 2005. discretas, 6th ed., R.

[3] C. Holloway, JPEG Image compression: Transformation, Quantization and Encoding , 2008.

[4] http://www.iosrjournals.org, "Digital Image Encryption Based on RSA Algorithm", 2016. [Online]. Available: http://www.iosrjournals.org/iosr-jece/papers/Vol9-Issue1/Version-4/N09146973.pdf. [Accessed: 19- nov- 2016].

[5]N. Sangwan, "Text Encryption with Huffman Compression", International Journal of Computer Applications, vol. 54, no. 6, pp. 29-32, 2012.

[6] A. Álvarez Gaona, Implementación en software-hardware de aritmética sobre campos finitos binarios F2m en curvas elípticas para aplicaciones criptográficas de llave pública , proyecto terminal, División de Ciencias Básicas e Ingeniería, Universidad Autónoma Metropolitana Azcapotzalco, México, 2012.

[7]D. Salomon, Data compression the complete reference, 1st ed. New York: Springer, 2004.

[8]M. Nelson and J. Gailly, The data compression book, 1st ed. New York: M & T Books, 1996.

[9]H. Delfs and H. Knebl, Introduction to cryptography, 1st ed. Berlin: Springer, 2007.

[10]HiFn Inc., The first book of compression and encryption. Hifn Whitepaper, January 2001.

[11]"La Página de los Cuentos - Texto 'Lo que le pasó a la tía Erika' de Ariel_Negod", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/554/554155/. [Accessed: 23- Dec- 2016].

[12]"La Página de los Cuentos - Texto 'Simona' de alipuso", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/463/463002/. [Accessed: 23- Dec- 2016].

[13]"La Página de los Cuentos - Texto 'Se me Olvido DecIrTe. . .' de ArianaEscobar", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/447/44769/. [Accessed: 23- Dec- 2016].

[14]"La Página de los Cuentos - Texto 'Su majestad, la reina Virginia' de CJVR", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/531/531223/. [Accessed: 23- Dec- 2016].

[15]"La Página de los Cuentos - Texto 'El cielo y el infierno me sostienen.' de Pato-Guacalas", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/564/564744/. [Accessed: 23- Dec- 2016].

[16]"La Página de los Cuentos - Texto 'Yin Yang' de El_Quinto_jinete", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/564/564672/. [Accessed: 23- Dec- 2016].

[17]"La Página de los Cuentos - Texto 'LA CASA DE LOS MENDEZ' de ome", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/564/564685/. [Accessed: 23- Dec- 2016].

[18]"La Página de los Cuentos - Texto 'Pánico' de ome", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/564/564839/. [Accessed: 23- Dec- 2016].

[19]"La Página de los Cuentos - Texto 'NDEE - Cap 4' de Darkyharry", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/570/570580/. [Accessed: 23- Dec- 2016].

[20]"La Página de los Cuentos - Texto 'Fatum' de Julia_flora", Loscuentos.net, 2016. [Online]. Available: http://www.loscuentos.net/cuentos/link/570/570174/. [Accessed: 23- Dec- 2016].

Apéndice

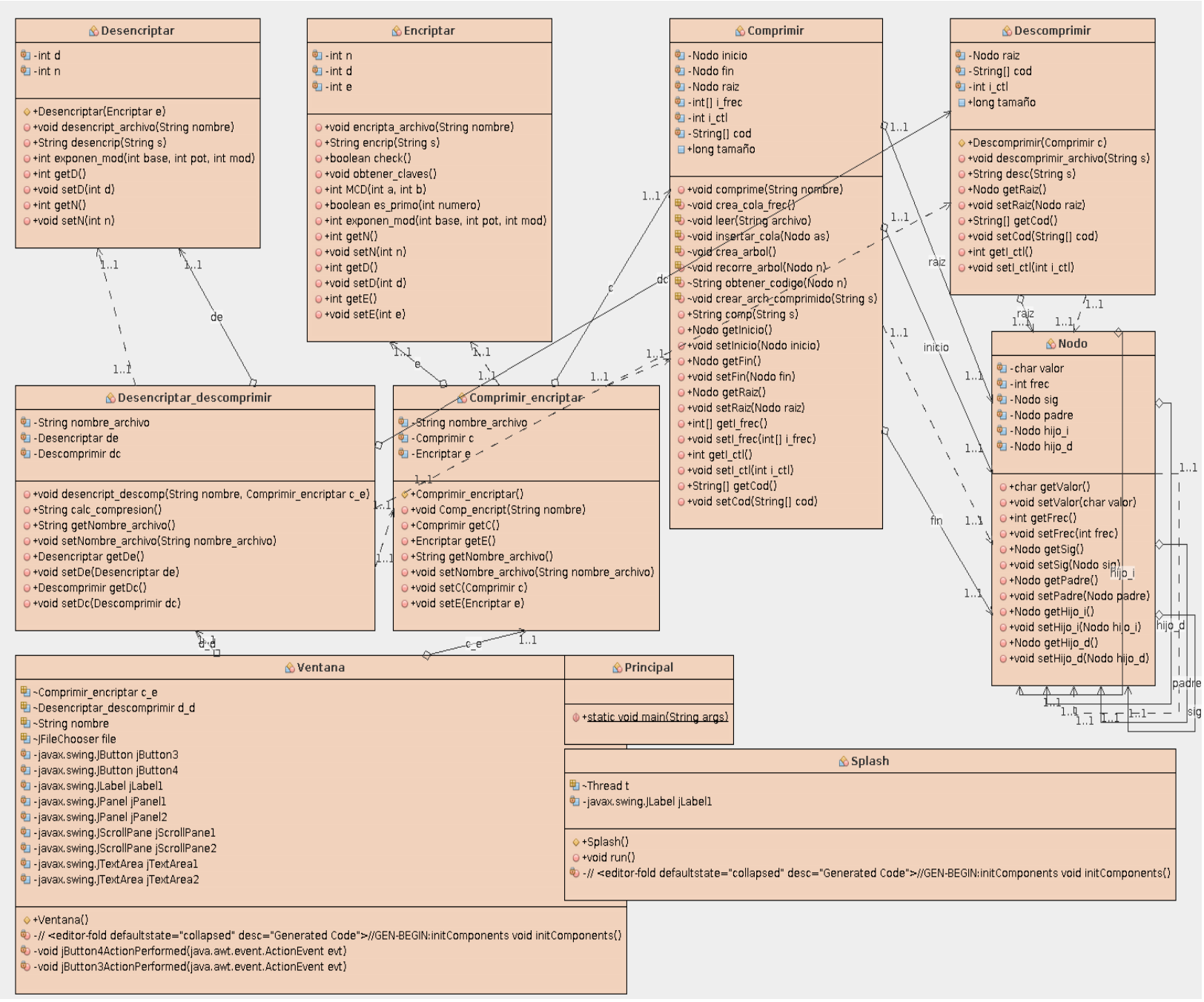


Figura 13 : Diagrama UML del sistema

Apéndice a “Módulo de compresión”

a.1 código fuente del nodo

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package modelo;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Nodo {
    private char valor;
    private int frec;
    private Nodo sig;
    private Nodo padre;
    private Nodo hijo_i;
    private Nodo hijo_d;

    /**
     * @return the valor
     */
    public char getValor() {
        return valor;
    }

    /**
     * @param valor the valor to set
     */
    public void setValor(char valor) {
        this.valor = valor;
    }

    /**
     * @return the frec
     */
    public int getFrec() {
        return frec;
    }

    /**
     * @param frec the frec to set
     */
    public void setFrec(int frec) {
        this.frec = frec;
    }

    /**
```

```
 * @return the sig
 */
public Nodo getSig() {
   return sig;
}

/**
 * @param sig the sig to set
 */
public void setSig(Nodo sig) {
   this.sig = sig;
}

/**
 * @return the padre
 */
public Nodo getPadre() {
   return padre;
}

/**
 * @param padre the padre to set
 */
public void setPadre(Nodo padre) {
   this.padre = padre;
}

/**
 * @return the hijo_i
 */
public Nodo getHijo_i() {
   return hijo_i;
}

/**
 * @param hijo_i the hijo_i to set
 */
public void setHijo_i(Nodo hijo_i) {
   this.hijo_i = hijo_i;
}

/**
 * @return the hijo_d
 */
public Nodo getHijo_d() {
   return hijo_d;
}

/**
 * @param hijo_d the hijo_d to set
 */
public void setHijo_d(Nodo hijo_d) {
   this.hijo_d = hijo_d;
```

```
    }
}
```

a.2 código fuente de compresión

```
package modelo;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Franck Corte Hernandez
 */

public class Comprimir {
    private Nodo inicio;
    private Nodo fin;
    private Nodo raiz;
    private int[] i_frec;
    private int i_ctl;
    private String[] cod = new String [256];
    public long tamaño;

    public void comprime(String nombre) throws IOException{
      //lee el texto y obtiene las frecuencias de cada carácter
      leer(nombre);
      //crea la cola de prioridad de acuerdo a las frecuencias
      crea_cola_frec();
      //crea el árbol que auxiliara en la compresión
      crea_arbol();
      //recorre el árbol para acceder a cada hoja y obtener los códigos de cada carácter
      recorre_arbol(getRaiz());
      crear_arch_comprimido(nombre);
    }

    void crea_cola_frec(){
       //crea la cola de frecuencia que posteriormente sería usada para crear el árbol Huffman
       int i=0;

       for(i=0;i<256;i++){
         if(getI_frec()[i]>0){
           Nodo n=new Nodo();
           n.setValor((char)i);
           n.setFrec(getI_frec()[i]);
           insertar_cola(n);
         }
```

```
    }
}

void leer(String archivo) throws IOException {
    //lee el archivo carácter por carácter y obtiene las frecuencias
    FileReader f=null;
    try {
        setI_frec(new int[256]);
        int i=0;
        for(i=0;i<256;i++)
            getI_frec()[i]=0;
        String cadena="";
        //abrimos el archivo
        f = new FileReader(archivo);
        tamaño=new File(archivo).length();
        BufferedReader b = new BufferedReader(f);
        //leemos el archivo
        int ia=b.read();
        while(ia!=-1){
            cadena=cadena+(char)ia;
            ia=b.read();
        }
        b.close();
        //con la cadena obtenida de leer el archivo
        //obtenemos las frecuencias
        for(i=0;i<cadena.length();i++){
                char ca=cadena.charAt(i);
                int aux=(int)ca;
                if(aux<256)
                    getI_frec()[aux]++;
            }

    } catch (FileNotFoundException ex) {
        Logger.getLogger(Comprimir.class.getName()).log(Level.SEVERE, null, ex);
    } finally {
        try {
            f.close();
        } catch (IOException ex) {
            Logger.getLogger(Comprimir.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

void insertar_cola(Nodo as){
    //insertamos el nodo as en la cola de acuerdo a su frecuencia
    Nodo a=new Nodo();
    a=as;

    if(getInicio()==null){
        setInicio(new Nodo());
        setFin(new Nodo());
        setInicio(a);
        setFin(a);
```

```
    }

    else{
       if(a.getFrec()<=getInicio().getFrec())
       {
          a.setSig(new Nodo());
          a.setSig(getInicio());
          setInicio(a);
       }
       if(a.getFrec()>getFin().getFrec()){
          getFin().setSig(new Nodo());
          getFin().setSig(a);
          setFin(a);
       }
       if(a.getFrec()>getInicio().getFrec()&&a.getFrec()<=getFin().getFrec()){
          Nodo aux=new Nodo();
          aux=getInicio();
          while(getInicio().getSig().getFrec()<a.getFrec()){
             setInicio(getInicio().getSig());
          }
          a.setSig(new Nodo());
          a.setSig(getInicio().getSig());
          getInicio().setSig(a);

          setInicio(aux);
       }
    }
    getFin().setSig(null);
}

void crea_arbol(){
  //creamos el árbol correspondiente a la cola de frecuencias

  while(getInicio().getSig()!=getFin()){
  Nodo nuevo=new Nodo();
  nuevo.setHijo_i(new Nodo());
  nuevo.setHijo_d(new Nodo());
        getInicio().setPadre(new Nodo());
        getInicio().getSig().setPadre(new Nodo());

  nuevo.setHijo_i(getInicio());
  nuevo.setHijo_d(getInicio().getSig());
        getInicio().setPadre(nuevo);
        getInicio().getSig().setPadre(nuevo);
        setInicio(getInicio().getSig().getSig());
  nuevo.setFrec(nuevo.getHijo_i().getFrec() + nuevo.getHijo_d().getFrec());
  nuevo.setValor((char)1000);
  insertar_cola(nuevo);
}

  Nodo nuevo=new Nodo();
  nuevo.setHijo_i(new Nodo());
  nuevo.setHijo_d(new Nodo());
```

```
      getInicio().setPadre(new Nodo());
      getInicio().getSig().setPadre(new Nodo());

   nuevo.setHijo_i(getInicio());
   nuevo.setHijo_d(getInicio().getSig());
      getInicio().setPadre(nuevo);
      getInicio().getSig().setPadre(nuevo);
   nuevo.setValor((char)1000);
   nuevo.setFrec(nuevo.getHijo_i().getFrec() + nuevo.getHijo_d().getFrec());

   setRaiz(new Nodo());
   setRaiz(nuevo);
}

void recorre_arbol(Nodo n){
   //recorremos el árbol con el objetivo de obtener el código a sustituir los 8 bits del ASCII
   if(n!=null){
      if((int)n.getValor()<256){
          getCod()[(int)n.getValor()]=obtener_codigo(n);
      }
      recorre_arbol(n.getHijo_i());
      recorre_arbol(n.getHijo_d());
   }
}

String obtener_codigo(Nodo n){
   //obtenemos el código a sustituir los 8 bits del ASCII
   String s="";
   Nodo aux=new Nodo();
   Nodo aux2=new Nodo();
   aux=n;
   while(aux.getPadre()!=getRaiz()){
      aux2=aux;
      aux=aux.getPadre();
      if(aux2==aux.getHijo_i()){
      s=s+"1";
      }
      if(aux2==aux.getHijo_d()){
      s=s+"0";
      }
   }
   aux2=aux;
   aux=aux.getPadre();
   if(aux2==aux.getHijo_i()){
      s=s+"1";
   }
   if(aux2==aux.getHijo_d()){
      s=s+"0";
   }
   StringBuilder builder=new StringBuilder(s);
   s=builder.reverse().toString();
   return s;
}
```

```
void crear_arch_comprimido(String s){
   //creamos el nuevo archivo comprimido
   try {

      String cadena="";
      FileReader fr=new FileReader(s);
      String nombre_com=s.substring(0 ,s.length()-4)+"_com.txt";
      FileWriter fw=new FileWriter(nombre_com);
      BufferedReader b = new BufferedReader(fr);

      PrintWriter p=new PrintWriter(fw);
      String aux="";
      while((cadena = b.readLine())!=null) {

      if(b.ready()){
         aux=aux+cadena+(char)10;
      }
      else{
         aux=aux+cadena;
      }
      }
      cadena=aux;
      String cadena2="";
      for(int i=0;i<cadena.length();i++){
            char ca=cadena.charAt(i);
            int x=(int)ca;
            if(x<256){
               cadena2=cadena2+getCod()[(int)ca];
            }
         }
       cadena2=comp(cadena2);
       p.print(cadena2);

      fw.close();
      fr.close();
   } catch (IOException ex) {
      Logger.getLogger(Comprimir.class.getName()).log(Level.SEVERE, null, ex);
   }
}
public String comp(String s){
   //se comprime la cadena obtenida de '0's y '1's
   String aux="";
   int i_aux=0;
   setI_ctl(s.length()%8);
   int j=1,i;//1,2,4,8,16,32,64,128
   for( i=0;i<s.length()-getI_ctl();i++){
      if(j==1){i_aux=i_aux+Integer.parseInt(""+s.charAt(i))*128;}
      if(j==2){i_aux=i_aux+Integer.parseInt(""+s.charAt(i))*64;}
      if(j==3){i_aux=i_aux+Integer.parseInt(""+s.charAt(i))*32; }
      if(j==4){i_aux=i_aux+Integer.parseInt(""+s.charAt(i))*16; }
      if(j==5){i_aux=i_aux+Integer.parseInt(""+s.charAt(i))*8; }
      if(j==6){i_aux=i_aux+Integer.parseInt(""+s.charAt(i))*4; }
```

```
      if(j==7){i_aux=i_aux+Integer.parseInt(""+s.charAt(i))*2; }
      if(j==8){
         i_aux=i_aux+Integer.parseInt(""+s.charAt(i))*1;
         aux=aux+(char)i_aux;
         j=0;
         i_aux=0;
      }
      j++;
   }
   aux=aux+s.substring(i);
   s=aux;
   return s;
}

/**
 * @return the inicio
 */
public Nodo getInicio() {
   return inicio;
}

/**
 * @param inicio the inicio to set
 */
public void setInicio(Nodo inicio) {
   this.inicio = inicio;
}

/**
 * @return the fin
 */
public Nodo getFin() {
   return fin;
}

/**
 * @param fin the fin to set
 */
public void setFin(Nodo fin) {
   this.fin = fin;
}

/**
 * @return the raiz
 */
public Nodo getRaiz() {
   return raiz;
}

/**
 * @param raiz the raiz to set
 */
public void setRaiz(Nodo raiz) {
```

```
        this.raiz = raiz;
    }

    /**
     * @return the i_frec
     */
    public int[] getI_frec() {
        return i_frec;
    }

    /**
     * @param i_frec the i_frec to set
     */
    public void setI_frec(int[] i_frec) {
        this.i_frec = i_frec;
    }

    /**
     * @return the i_ctl
     */
    public int getI_ctl() {
        return i_ctl;
    }

    /**
     * @param i_ctl the i_ctl to set
     */
    public void setI_ctl(int i_ctl) {
        this.i_ctl = i_ctl;
    }

    /**
     * @return the cod
     */
    public String[] getCod() {
        return cod;
    }

    /**
     * @param cod the cod to set
     */
    public void setCod(String[] cod) {
        this.cod = cod;
    }
}
```

a.3 código fuente para quitar caracteres especiales

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
```

```
import java.util.StringTokenizer;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author darkry
 */
public class Quitar_caracteres_especiales {
   //clase que quita los caracteres especiales
   public static void main(String[] args) throws FileNotFoundException, IOException{
      String s="/home/peluso/Escritorio/Textos/texto02.txt";
      FileReader fr=fr=new FileReader(s);
      BufferedReader b = new BufferedReader(fr);
      String cadena;
      String aux="";
      while((cadena = b.readLine())!=null) {

         if(b.ready()){
            aux=aux+cadena+(char)10;
         }
         else{
            aux=aux+cadena;
         }
      }
      b.close();
      fr.close();

      cadena=remover(aux);
      FileWriter fw=new FileWriter(s);
      PrintWriter p=new PrintWriter(fw);
      p.print(cadena);
      fw.close();
      p.close();
   }
   public static String remover(String entrada) {
   // Cadena de caracteres original a sustituir.
   String original = "áàäéèëíïóòöúùuñÁÀÄÉÈËÍÏÓÒÖÚÙÜÑçÇ";
   // Cadena de caracteres ASCII que reemplazarán los originales.
   String ascii = "aaaeeeiiiooouuunAAAEEEIIIOOOUUUNcC";
   String salida = entrada;
   for (int i=0; i<original.length(); i++) {
      // Reemplazamos los caracteres especiales.
      salida = salida.replace(original.charAt(i), ascii.charAt(i));
   }
   return salida;
}

}
```

Apéndice b “Módulo de descompresión”

b.1 código fuente de descompresión

```
package modelo;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Descomprimir {
   private Nodo raiz;
   private String [] cod=new String [256];
   private int i_ctl;
   public long tamaño;

  public Descomprimir(Comprimir c){
     this.raiz=c.getRaiz();
     this.cod=c.getCod();
     this.i_ctl=c.getI_ctl();
  }
  public void descomprimir_archivo(String s) throws IOException{
    if(getRaiz()!=null){
       FileReader fr=null;
       try {
          String cadena="";
          //abre el archivo comprimido y desencriptado
          String nombre=s.substring(0 ,s.length()-4)+"_com_desencrip.txt";
          fr = new FileReader(nombre);
          //crea el archivo  donde guardara el archivo descomprimido
          nombre=s.substring(0 ,s.length()-4)+"_desc.txt";
          //nombre=s.substring(0 ,s.length()-4)+"_descom_desc.txt";
          FileWriter fw=new FileWriter(nombre);

          BufferedReader b = new BufferedReader(fr);
          PrintWriter p=new PrintWriter(fw);

          String saux="";
//lee el archivo comprimido carácter por carácter
          int ia=b.read();
          while(ia!=-1){
             saux=saux+(char)ia;
             ia=b.read();
```

```
        }
//
         cadena=saux;
         //obtenemos la cadena de '0's y '1's
         cadena=desc(cadena);
         //obtenemos el texto original
            String cadena2="";
            Nodo aux=getRaiz();
            for(int i=0;i<cadena.length();i++){
                char ca=cadena.charAt(i);
                if(ca=='1'){
                  aux=aux.getHijo_i();
                  if((int)aux.getValor()<256){
                     cadena2=cadena2+aux.getValor();
                     aux=getRaiz();
                  }
                }
                if(ca=='0'){
                  aux=aux.getHijo_d();
                  if((int)aux.getValor()<256){
                     cadena2=cadena2+aux.getValor();
                     aux=getRaiz();
                  }
                }
             }
             p.print(cadena2);
         //cierra los archivos
         fr.close();
         fw.close();
         tamaño= new File(nombre).length();
      } catch (FileNotFoundException ex) {
         Logger.getLogger(Descomprimir.class.getName()).log(Level.SEVERE, null, ex);
      } finally {
         try {
            fr.close();
         } catch (IOException ex) {
            Logger.getLogger(Descomprimir.class.getName()).log(Level.SEVERE, null, ex);
         }
      }
   }
 }
    public String desc(String s){
     //descomprime la cadena
     String aux="";
     int i,j=1;
     for(i=0;i<s.length()-getI_ctl();i++){
        j=1;
        int i_aux=s.charAt(i);
        while(j<9){
        if(j==1){
           if(i_aux>=128)
           {
              aux=aux+1;
```

```
      i_aux=i_aux-128;
   }
   else{
      aux=aux+0;
   }
}
if(j==2){
   if(i_aux>=64)
   {
      aux=aux+1;
      i_aux=i_aux-64;
   }
   else{
      aux=aux+0;
   }
}
if(j==3){
   if(i_aux>=32)
   {
      aux=aux+1;
      i_aux=i_aux-32;
   }
   else{
      aux=aux+0;
   }
}
if(j==4){
   if(i_aux>=16)
   {
      aux=aux+1;
      i_aux=i_aux-16;
   }
   else{
      aux=aux+0;
   }
}
if(j==5){
   if(i_aux>=8)
   {
      aux=aux+1;
      i_aux=i_aux-8;
   }
   else{
      aux=aux+0;
   }
}
if(j==6){
   if(i_aux>=4)
   {
      aux=aux+1;
      i_aux=i_aux-4;
   }
   else{
```

```
            aux=aux+0;
         }
      }
      if(j==7){
         if(i_aux>=2)
         {
            aux=aux+1;
            i_aux=i_aux-2;
         }
         else{
            aux=aux+0;
         }}
      if(j==8){
         if(i_aux>=1)
         {
            aux=aux+1;
            i_aux=i_aux-1;
         }
         else{
            aux=aux+0;
         }
      }
      j++;
      }
   }
   aux=aux+s.substring(i);
   return s=aux;
}

/**
 * @return the raiz
 */
public Nodo getRaiz() {
   return raiz;
}

/**
 * @param raiz the raiz to set
 */
public void setRaiz(Nodo raiz) {
   this.raiz = raiz;
}

/**
 * @return the cod
 */
public String[] getCod() {
   return cod;
}

/**
 * @param cod the cod to set
 */
```

```
    public void setCod(String[] cod) {
        this.cod = cod;
    }

    /**
     * @return the i_ctl
     */
    public int getI_ctl() {
        return i_ctl;
    }

    /**
     * @param i_ctl the i_ctl to set
     */
    public void setI_ctl(int i_ctl) {
        this.i_ctl = i_ctl;
    }

}
```

Apéndice c “Módulo de encriptado”

c.1 código fuente de encriptación

```
package modelo;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Random;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Encriptar {

    private int n;
    private int d;
    private int e;

    public void encripta_archivo(String nombre) throws IOException{
        FileReader fr=null;
        fr=new FileReader(nombre.substring(0 ,nombre.length()-4)+"_com.txt");
        BufferedReader b = new BufferedReader(fr);
        String cadena;
        String aux="";

        int ia=b.read();
```

```
        while(ia!=-1){
           aux=aux+(char)ia;
           ia=b.read();
        }
     cadena=aux;
     b.close();
     fr.close();
     cadena=encrip(cadena);
     FileWriter fw=new FileWriter(nombre.substring(0 ,nombre.length()-4)+"_com_encrip.txt");
     PrintWriter p=new PrintWriter(fw);
     p.print(cadena);
     p.close();
     fw.close();
  }
  public String encrip(String s){
     //La función de cifrado RSA es C = M^e mod n
     String aux="";
     obtener_claves();
     while(!check()){
        obtener_claves();
     }

     for(int i=0;i<s.length();i++){
        char c=s.charAt(i);
       int j=exponen_mod(c, getE(), getN());
        aux=aux+(char)j;
     }
     return aux;
  }
  public boolean check(){
     //checa si los valores obtenidos para n,d,e
     //se pueden usar para encriptar correctamente los 256 caracteres ASCII
     //correctamente

     String s="",aux="";
     char c;
     int i;
     for( i=0;i<256;i++){
         c=(char)i;
        s=s+c;
     }

     for(i=0;i<s.length();i++){
        c=s.charAt(i);
        int j=exponen_mod(c, getE(), getN());
        aux=aux+(char)j;
     }
     s=aux;

     for( i=0;i<s.length();i++){
        c=s.charAt(i);
        int j=exponen_mod(c, getD(), getN());
        aux=aux+(char)j;
```

```
    }
    for( i=0;i<256;i++){
        c=(char)i;
       s=s+c;
    }
    if(s.equals(aux)){
       return true;
    }
    else
    return false;
}

public void obtener_claves(){
    //obtenemos los valores de n,d,e a partir
    //de los cálculos siguientes
    setN(0);
    setD(1);
    setE(3);
    int p,q,fi,i=1;
    Random rnd = new Random();
    p = (int)(rnd.nextDouble() * 70 +10);
    q = p - 10;
    while(es_primo(p)==false){
    p = p+ 1;
    }
    while(es_primo(q)==false){
    q = q+ 1;
    }
    //calculo n
    setN(p*q);
    fi=(p-1)*(q-1);
    //calculo e
    while(MCD(getE(),fi)!=1){
       setE(getE() + 2);
    }
    //calculo de d
    while(((i*fi)+1)%getE()!=0){
    i++;
    }
    setD(((i*fi)+1) / getE());
    //e y n son la clave pública
    //d y n son la clave privada
}

public int MCD(int a,int b){
    //calculamos el MCD de a y b
    int mcd=1;
    //a es el menor
    for(int i=1;i<=a;i++){
        if(a % i == 0 && b % i == 0){
           mcd = i;
        }
     }
```

```
   return mcd;
}

public boolean es_primo(int numero){
 //comprobamos si el valor numero es primo
 if (numero%2==0)
   return false;

 int contador = 2;
 boolean primo=true;
 while ((primo) && (contador!=numero)){
     if (numero % contador == 0){
       primo = false;
       return primo;
     }
     contador++;
 }
 return primo;
}

public int exponen_mod(int base,int pot,int mod){
   //calculamos lo siguiente
   //exp=base^pot(modulo mod)
  //por medio de  exponenciación modular rápida
   int exp=1,aux=base%mod;
   while (pot>0){
      if(pot%2!=0){
         exp=(exp*aux)%mod;
      }
      aux=(aux*aux)%mod;
      pot=pot/2;
   }

   return exp;
}

/**
 * @return the n
 */
public int getN() {
   return n;
}

/**
 * @param n the n to set
 */
public void setN(int n) {
   this.n = n;
}

/**
 * @return the d
 */
```

```
    public int getD() {
        return d;
    }

    /**
     * @param d the d to set
     */
    public void setD(int d) {
        this.d = d;
    }

    /**
     * @return the e
     */
    public int getE() {
        return e;
    }

    /**
     * @param e the e to set
     */
    public void setE(int e) {
        this.e = e;
    }

}
```

Apéndice d "Módulo de desencriptado"

d.1 código fuente de desencriptado

```
package modelo;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Desencriptar {

    private int d;
    private int n;
```

```
public Desencriptar(Encriptar e){
   this.d=e.getD();
   this.n=e.getN();
}

public void desencript_archivo(String nombre) throws FileNotFoundException{
   try {
      FileReader fr=null;
      fr=new FileReader(nombre.substring(0 ,nombre.length()-4)+"_com_encrip.txt");
      BufferedReader b = new BufferedReader(fr);
      String cadena;
      String aux="";

      int ia=b.read();
      while(ia!=-1){
         aux=aux+(char)ia;
         ia=b.read();
      }
      cadena=aux;
      b.close();
      fr.close();
      cadena=desencrip(cadena);
      FileWriter fw=new FileWriter(nombre.substring(0 ,nombre.length()-4)+"_com_desencrip.txt");
      PrintWriter p=new PrintWriter(fw);
      p.print(cadena);
      p.close();
      fw.close();

   } catch (IOException ex) {
      Logger.getLogger(Desencriptar.class.getName()).log(Level.SEVERE, null, ex);
   }
}
public String desencrip(String s){
   // La función de descifrado RSA es M = C^d mod n
   String aux="";
   for(int i=0;i<s.length();i++){
      char c=s.charAt(i);
      int j=exponen_mod(c, getD(), getN());
      aux=aux+(char)j;
   }
   return aux;
}

public int exponen_mod(int base,int pot,int mod){
   //exp=base^pot(modulo mod)
   int exp=1,aux=base%mod;
```

```
        while (pot>0){
            if(pot%2!=0){
                exp=(exp*aux)%mod;
            }
            aux=(aux*aux)%mod;
            pot=pot/2;
        }

        return exp;
    }

    /**
     * @return the d
     */
    public int getD() {
        return d;
    }

    /**
     * @param d the d to set
     */
    public void setD(int d) {
        this.d = d;
    }

    /**
     * @return the n
     */
    public int getN() {
        return n;
    }

    /**
     * @param n the n to set
     */
    public void setN(int n) {
        this.n = n;
    }
}
```

Apéndice e "Módulo de compresión/encriptado"

e.1 código fuente para comprimir/encriptar

```
package controlador;

import java.io.IOException;
```

```
import java.util.logging.Level;
import java.util.logging.Logger;
import modelo.Comprimir;
import modelo.Encriptar;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Comprimir_encriptar {
   private String nombre_archivo;
   private Comprimir c;
   private Encriptar e;


   public Comprimir_encriptar(){
      this.c=new Comprimir();
      this.e=new Encriptar();
   }
   public void Comp_encript(String nombre){
      try {
         setNombre_archivo(nombre);
         //modulo comprimir
         getC().comprime(getNombre_archivo());
         //modulo encriptar
         getE().encripta_archivo(getNombre_archivo());
      } catch (IOException ex) {
         Logger.getLogger(Comprimir_encriptar.class.getName()).log(Level.SEVERE, null, ex);
      }
   }

   /**
    * @return the c
    */
   public Comprimir getC() {
      return c;
   }

   /**
    * @return the e
    */
   public Encriptar getE() {
      return e;
   }

   /**
```

```
    * @return the nombre_archivo
    */
   public String getNombre_archivo() {
      return nombre_archivo;
   }

   /**
    * @param nombre_archivo the nombre_archivo to set
    */
   public void setNombre_archivo(String nombre_archivo) {
      this.nombre_archivo = nombre_archivo;
   }

   /**
    * @param c the c to set
    */
   public void setC(Comprimir c) {
      this.c = c;
   }

   /**
    * @param e the e to set
    */
   public void setE(Encriptar e) {
      this.e = e;
   }
}
```

Apéndice f "Módulo de desencriptado/descompresión"

f.1 código fuente para desencriptar/descomprimir

```
package controlador;

import java.io.FileNotFoundException;
import java.io.IOException;
import modelo.Descomprimir;
import modelo.Desencriptar;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Desencriptar_descomprimir {
   private String nombre_archivo;
   private Desencriptar de;
   private Descomprimir dc;
```

```
  public void desencript_descomp(String nombre,Comprimir_encriptar c_e) throws FileNotFoundException,
IOException{
      setNombre_archivo(nombre);
      setDc(new Descomprimir(c_e.getC()));
      setDe(new Desencriptar(c_e.getE()));
     //modulo desencriptar
      getDe().desencript_archivo(getNombre_archivo());
     //modulo descomprimir
      getDc().descomprimir_archivo(getNombre_archivo());
  }
  public String calc_compresion(){
     String s="";
     long l_original,l_comp=0,l_desc;
     //obteniendo el peso del archivo original
     File f=new File(getNombre_archivo());
     l_original=f.length();
     //obteniendo el peso del archivo comprimido
     f=new File(getNombre_archivo().substring(0, getNombre_archivo().length()-4)+"_com.txt");
     l_comp=f.length();
     //obteniendo el peso del archivo descomprimido
     f=new File(getNombre_archivo().substring(0, getNombre_archivo().length()-4)+"_desc.txt");
     l_desc=f.length();

     s=(char)10+"Tamaño archivo original en bytes: "+l_original+(char)10+"Tamaño archivo comprimido en
bytes: "+l_comp+(char)10+"Tamaño archivo descomprimido en bytes: "+l_desc+(char)10;
     s=s+"De lo que se concluye lo siguiente:"+(char)10;

     if(l_original==l_desc&&l_comp<=l_original){
        //calculo del % de comprecion
        s=s+"La compreción fue exitosa y esta fue de: "+l_comp*100/l_original+" % con respecto al
original"+(char)10;
     }
     else{
        s=s+"La compreción fallo, el archivo comprimido es "+l_comp*100/l_original+" % mas grande
respecto al original"+(char)10;
     }
     return s;
  }

  /**
   * @return the nombre_archivo
   */
  public String getNombre_archivo() {
     return nombre_archivo;
  }
```

```
    /**
     * @param nombre_archivo the nombre_archivo to set
     */
    public void setNombre_archivo(String nombre_archivo) {
        this.nombre_archivo = nombre_archivo;
    }

    /**
     * @return the de
     */
    public Desencriptar getDe() {
        return de;
    }

    /**
     * @param de the de to set
     */
    public void setDe(Desencriptar de) {
        this.de = de;
    }

    /**
     * @return the dc
     */
    public Descomprimir getDc() {
        return dc;
    }

    /**
     * @param dc the dc to set
     */
    public void setDc(Descomprimir dc) {
        this.dc = dc;
    }
}
```

Apéndice g “Módulo de menú/interfaz gráfica”

g.1 código fuente de la ventana

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package VIsta;
```

```
import controlador.Comprimir_encriptar;
import controlador.Desencriptar_descomprimir;
import java.io.File;
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileNameExtensionFilter;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Ventana extends javax.swing.JFrame {

    /**
     * Creates new form ventana
     */
      Comprimir_encriptar c_e;
      Desencriptar_descomprimir d_d;
      String nombre;
      JFileChooser file;
    public Ventana() {
        initComponents();
        this.setResizable(false);
        this.setTitle("Sistema de compresión y encriptación");
        this.setLocationRelativeTo(null);
        this.jTextArea1.setFocusable(false);
        c_e=new Comprimir_encriptar();
        d_d= new Desencriptar_descomprimir();
        nombre="";
        file=new JFileChooser();
        FileNameExtensionFilter filtro = new FileNameExtensionFilter("Archivos .txt", "txt", "text");
        file.setFileFilter(filtro);
        jTextArea1.setText("Bienvenido al Sistema de compresión de archivos de texto "+(char)10+"sin pérdida
de datos con encriptación de clave pública");
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
    private void initComponents() {

        jPanel1 = new javax.swing.JPanel();
        jPanel2 = new javax.swing.JPanel();
        jButton3 = new javax.swing.JButton();
        jButton4 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();
        jLabel1 = new javax.swing.JLabel();
        jScrollPane2 = new javax.swing.JScrollPane();
        jTextArea2 = new javax.swing.JTextArea();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        jPanel2.setBackground(new java.awt.Color(0, 0, 0));

jPanel2.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.border.BevelBorder.RAISED));

        jButton3.setBackground(new java.awt.Color(51, 51, 51));
        jButton3.setForeground(new java.awt.Color(255, 255, 255));
        jButton3.setText("Comprimir/Encriptar");
        jButton3.setToolTipText("De click para comprimir y encriptar un archivo ");
        jButton3.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton3ActionPerformed(evt);
            }
        });

        jButton4.setBackground(new java.awt.Color(51, 51, 51));
        jButton4.setForeground(new java.awt.Color(255, 255, 255));
        jButton4.setText("Desencriptar/Descomprimir");
        jButton4.setToolTipText("De click para desencriptar/descomprimir un archivo");
        jButton4.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
        jButton4.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton4ActionPerformed(evt);
            }
        });

        jTextArea1.setBackground(new java.awt.Color(51, 51, 51));
        jTextArea1.setColumns(20);
        jTextArea1.setFont(new java.awt.Font("Dialog", 2, 11)); // NOI18N
        jTextArea1.setForeground(new java.awt.Color(255, 255, 255));
        jTextArea1.setRows(5);
        jTextArea1.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
```

```
        jTextArea1.setFocusable(false);
        jScrollPane1.setViewportView(jTextArea1);

        jLabel1.setText("Seleccione comprimir/encriptar o Desencriptar/descomprimir, posteriormente seleccione el archivo");

        jTextArea2.setBackground(new java.awt.Color(0, 0, 0));
        jTextArea2.setColumns(20);
        jTextArea2.setFont(new java.awt.Font("Dialog", 0, 10)); // NOI18N
        jTextArea2.setForeground(new java.awt.Color(255, 255, 255));
        jTextArea2.setRows(5);
        jTextArea2.setText("COMPRIMIR/ENCRIPTAR\n1-De click en comprimir/encriptar,\n2-en el buscador de archivos localice \nel archivo a comprimir/encriptar\n3-El programa le comunicara si el proceso \nfue exitoso  \n\nDESCOMPRIMIR/DESENCRIPTAR\n1-De clic en desencriptar\n\n\n\nEste programa fue realizado por\nFranck Cortes Hernandez");
        jTextArea2.setBorder(null);
        jTextArea2.setCaretColor(new java.awt.Color(0, 0, 0));
        jTextArea2.setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        jTextArea2.setFocusable(false);
        jScrollPane2.setViewportView(jTextArea2);

        javax.swing.GroupLayout jPanel2Layout = new javax.swing.GroupLayout(jPanel2);
        jPanel2.setLayout(jPanel2Layout);
        jPanel2Layout.setHorizontalGroup(
            jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(jPanel2Layout.createSequentialGroup()
                .addGap(53, 53, 53)
                .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE, 218, javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                        .addComponent(jButton4)
                        .addGap(69, 69, 69))
                    .addGroup(jPanel2Layout.createSequentialGroup()
                        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addGap(29, 29, 29)
                                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 0, javax.swing.GroupLayout.PREFERRED_SIZE))
                            .addGroup(jPanel2Layout.createSequentialGroup()
                                .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 533, javax.swing.GroupLayout.PREFERRED_SIZE)
                                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
```

```
                    .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 227,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(68, Short.MAX_VALUE))))
    );
    jPanel2Layout.setVerticalGroup(
      jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel2Layout.createSequentialGroup()
        .addGap(46, 46, 46)
        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
          .addComponent(jButton3)
          .addComponent(jButton4))
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 26,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(18, 18, 18)
        .addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
          .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE, 226,
javax.swing.GroupLayout.PREFERRED_SIZE)
          .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup()
            .addComponent(jScrollPane2, javax.swing.GroupLayout.PREFERRED_SIZE, 193,
javax.swing.GroupLayout.PREFERRED_SIZE)
            .addGap(23, 23, 23)))
        .addContainerGap(38, Short.MAX_VALUE))
    );

    javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
    jPanel1.setLayout(jPanel1Layout);
    jPanel1Layout.setHorizontalGroup(
      jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel1Layout.createSequentialGroup()
        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 0, Short.MAX_VALUE))
    );
    jPanel1Layout.setVerticalGroup(
      jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
      .addGroup(jPanel1Layout.createSequentialGroup()
        .addComponent(jPanel2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(0, 0, Short.MAX_VALUE))
    );

    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
    getContentPane().setLayout(layout);
    layout.setHorizontalGroup(
```

```
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
          .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
          .addGap(0, 0, Short.MAX_VALUE))
      );
      layout.setVerticalGroup(
        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
          .addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
          .addGap(0, 0, Short.MAX_VALUE))
      );

      pack();
    }// </editor-fold>

    private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
          // TODO add your handling code here:
          jTextArea1.setText(jTextArea1.getText()+(char)10+"Archivo a desenencriptar/descomprimir:
"+nombre);
          d_d.desencript_descomp(nombre,c_e);
          jTextArea1.setText(jTextArea1.getText()+(char)10+"El archivo se desencripto y descomprimio
exitosamente"+d_d.calc_compresion());
        } catch (IOException ex) {
          Logger.getLogger(Ventana.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
        try {
          // TODO add your handling code here:

          file.showOpenDialog(this);
          File abre=file.getSelectedFile();
          nombre=abre.getAbsolutePath();
          jTextArea1.setText(jTextArea1.getText()+(char)10+"Archivo a comprimir/encriptar: "+nombre);
          c_e=new Comprimir_encriptar();
          c_e.Comp_encript(nombre);
          jTextArea1.setText(jTextArea1.getText()+(char)10+"El archivo se comprimio y encripto
exitosamente");
        } catch (IOException ex) {
          Logger.getLogger(Ventana.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
```

```
    /**
     * @param args the command line arguments
     */


    // Variables declaration - do not modify
    private javax.swing.JButton jButton3;
    private javax.swing.JButton jButton4;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JPanel jPanel1;
    private javax.swing.JPanel jPanel2;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JScrollPane jScrollPane2;
    private javax.swing.JTextArea jTextArea1;
    private javax.swing.JTextArea jTextArea2;
    // End of variables declaration
}
```

g.2 código fuente del Splash screen

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package VIsta;

import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Splash extends javax.swing.JFrame implements Runnable {

    /**
     * Creates new form Splash
     */
    Thread t;
    public Splash() {
        initComponents();
    }
    @Override
    public void run(){
        try {
```

```
            //inicialisa el SplashScreen
            this.setTitle("            ");
            this.setLocationRelativeTo(null);
            this.setVisible(true);
            t.sleep(5000);
            //cierra el Splash
            this.dispose();
            //abre la ventana principal
            new Ventana().setVisible(true);
        } catch (InterruptedException ex) {
            Logger.getLogger(Splash.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    /**
     * This method is called from within the constructor to initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is always
     * regenerated by the Form Editor.
     */
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        setBackground(new java.awt.Color(0, 0, 0));
        setUndecorated(true);

        jLabel1.setIcon(new      javax.swing.ImageIcon(getClass().getResource("/VIsta/Splash_icon.png")));      //
NOI18N

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1,                                    javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addComponent(jLabel1)
        );

        pack();
    }// </editor-fold>
```

```
    /**
     * @param args the command line arguments
     */

    // Variables declaration - do not modify
    private javax.swing.JLabel jLabel1;
    // End of variables declaration
}
```

g.3 código fuente de la clase principal

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package VIsta;

/**
 *
 * @author Franck Corte Hernandez
 */
public class Principal {
    public static void main(String args[]) {
        new Thread(new Splash()).start();

    }
}
```