Cuvalles

Proyecto final de encriptación

Orlando Aguilar Vivanco

09/10/11

Introducción

El presente trabajo tiene como fin describir un sistema que oculta información dentro de una imagen. Este tipo de técnicas se conocen como estenografía, definida como: “un sistema utilizado para **encriptar** la información y hacerla invisible a los ojos de los demás, haciéndola sólo accesible para el emisor y el receptor”[1].

Esto es importante debido a que en la actualidad el cifrado de la información es vital en casi toda actividad humana. Debemos recordar que la información es poder, y a quien le damos a conocer nuestra información le estamos dando poder sobre nosotros.

Este trabajo no sólo intenta describir el sistema, también produjo el sistema. El sistema se programó en java y es el proyecto final de la clase de criptografía.

Objetivos

El objetivo del trabajo es la generación de un sistema que pueda encriptar una cadena de texto de cualquier magnitud y además pueda ocultar el resultado de la encriptación dentro de una imagen png.

El objetivo de este trabajo se ha logrado, y además se logró comprimir los datos de la cadena reduciendo los requisitos de tamaño de la imagen en la que se almacenan los datos.

Breve descripción

El proyecto consiste en tomar imágenes en formato png, con 3 canales de información y ocultar en ellas información previamente comprimida, cifrada y separada.

La información dentro de la imagen se encuentra dispersa, y la dispersión se encuentra ordenada por funciones que permiten seleccionar el canal y el avance desde la posición actual de almacenaje a la siguiente, de tal forma que la información no tiene un orden tan fácil de predecir en el encriptado (Se encuentra en los tres canales y con posiciones de avance distinto).

Como ya se había comentado, este proyecto generó un software, programado en java. Capturas del software se encuentran en los anexos 1 y 2.

A continuación describo más ampliamente el proceso de compresión, cifrado y ocultación de la información en la imagen.

Compresión

Lo primero que se hace es comprimir la cadena transformándolo en un arreglo de bytes y enviándola como parámetro al stream Gzip de JAVA, luego se toma el arreglo resultado y se transforma

en una cadena nuevamente usando la función byteArray2String.

```
public static String comprime(String s)throws IOException{
   ByteArrayOutputStream os= new ByteArrayOutputStream(s.length());
   GZIPOutputStream out = new GZIPOutputStream(os);
   byte[] bt= s.getBytes();
   out.write(bt);
   out.finish();
   out.close();
   return byteArray2String(os.toByteArray());
 }


 public static String byteArray2String(byte[] b){
    tam=b.length;
   StringBuilder build= new StringBuilder(b.length/2+ (b.length%2));
    for (int z=0,p=0;z<b.length;z+=2,p++){
     int aux;
     if ((z+1)<b.length){
        aux=((int)(b[z]&0xFF)) | ((int)((b[z+1]&0xFF)<<8));
     }else
        aux=b[z];
     build.append((char) aux);
   }
   return build.toString();
 }
```

Cifrado

El algoritmo de cifrado está basado en el Feistel[2]. Divide la cadena en bloques de 64 bits y luego cada bloque los divide en 2 bloques realizando operaciones en cada medio bloque y luego haciendo intercambios. Debido a la forma en que se aplican las operaciones para encriptar (se aplica un xor al resultado de la función de encripción), la función que se aplica no necesita ser reversible. En el programa se usa la función (((t%k)|0xACF00)&t ^ val) con t el trozo del bloque al que se le aplicará la función, k la llave de un arreglo interno al programa y val una llave simétrica.

Algoritmo para transformar long en cadenas y cadenas en long:

```
public static long str4ToLong(String c){
        long l=0;
        int t=c.length();
        t=t>4? 4 : t;
        for (int z=0;z<t;z++){
           l|=Long.rotateLeft((long)c.charAt(z),16*z);
```

```
  }
  return l;
 }
 public static String longToStr4(long l){
  StringBuilder res= new StringBuilder();
  for (int z=0;z<4;z++){
     char c=(char)(Long.rotateRight(l,16*z)&0xFFFF);
     res.append(c);
  }
  return res.toString();
 }
```

Ocultación de la información en la imagen

Para ocultar la información en la imagen se usan 2 funciones. Una función se usa para seleccionar el avance en los canales de color de la imagen. Otra se usa para seleccionar el canal que se usará para ocultar la información. La información sólo se oculta en los primeros 3 bits menos significativos, ya que es en estos bits donde el ojo humano es incapaz de reconocer los cambios, o al menos es muy poco probable que los identifique a simple vista.

El avance en cada canal se refiere a la posición x en la que se ocultará el bit, cada canal contiene un contador de la última posición x en la que se ocultó un bit, así que la próxima vez que se vaya a ocultar un bit en ese canal se actualiza el contador con la función apropiada y se usa esa nueva posición para ocultarlo.

Funciones de avance:

```
int nextColor(){
    angle++;
    return (int) abs(sin(angle)*8) %3;
  }
  int nextAvance(){
    return (int)abs((sin(++angle)*cos(++angle/10.0f)*20.0f) % 6)+1;
  }
  void next() throws LimitExcededException{
    c=nextColor();
    i[c]+=nextAvance();
    if (i[c]>=ancho){
      j[c]++;
      i[c]=i[c] % ancho;
      if (j[c]>=alto)
        throw new LimitExcededException();
```

```
    }
    iAct=i[c];
    jAct=j[c];
    pix=nextColor();
  }
```

Guardado de la imagen.

La imagen se guarda en formato png, ese formato es un archivo de compresión sin pérdida, así aseguramos que la información se podrá recuperar, y que nuestra imagen no ocupará mucho espacio en almacenamiento secundario.

Recuperación de la información cifrada de la imagen.

La recuperación de información original sigue el mismo algoritmo de ocultación de información. Con cada uno de los bits recuperados se genera una cadena que contiene el texto encriptado.

Descifrado de la información

El descifrado de la información se realiza con el mismo algoritmo de cifrado, sólo que el orden de las llaves invierte.

Descompresión de la información

El algoritmo de descompresión transforma la cadena en un arreglo de bytes, realizando el proceso inverso al de la compresión y luego se pasa por un stream de descompresión Gzip para luego recuperar la cadena original.

```
 public static String descomprime(String s)throws IOException{
   byte[] bt= String2byteArray(s);
   ByteArrayInputStream os= new ByteArrayInputStream(bt);
   GZIPInputStream in = new GZIPInputStream(os);
   StringBuilder build= new StringBuilder(bt.length);
   int arr;
   while((arr=in.read())!=-1)
     build.append((char) arr);
   in.close();
   return build.toString();
  }


public static byte[] String2byteArray(String a){
   byte [] r=new byte[tam];
```

```
  for (int z=0,p=0;p<a.length();z+=2,p++){
    int aux=a.charAt(p);
    r[z]=(byte)(aux&0xFF);
    if ((z+1)<tam)
      r[z+1]=(byte)(aux>>8);
  }
  return r;
}
```

Conclusiones

Para concluir me gustaría decir que es fascinante la criptografía, durante la elaboración de este sistema indagué varias fuentes de todo tipo, desde matemáticas hasta de programación ampliando mi base de conocimientos en muchos campos.

Quizá este sistema no es el más seguro, quizá no sea el algoritmo de estenografía más importante, pero es fácilmente ampliable y ampliamente personalizable, lo que lo hace susceptible a mejoras en un futuro.

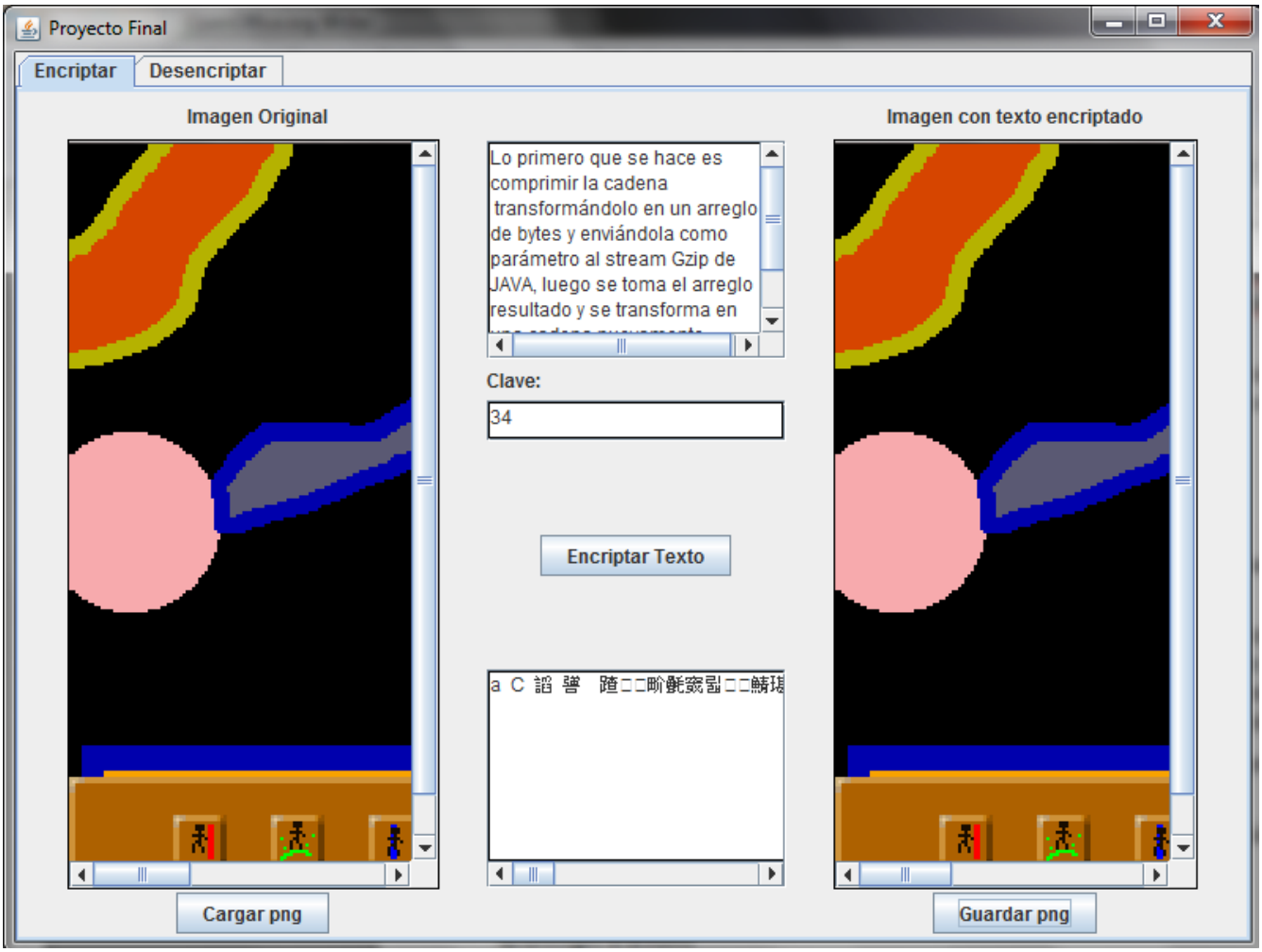
El desarrollo de un sistema de criptografía es sólo el comienzo de la búsqueda de su destrucción y de su fortalecimiento.

Bibliografía

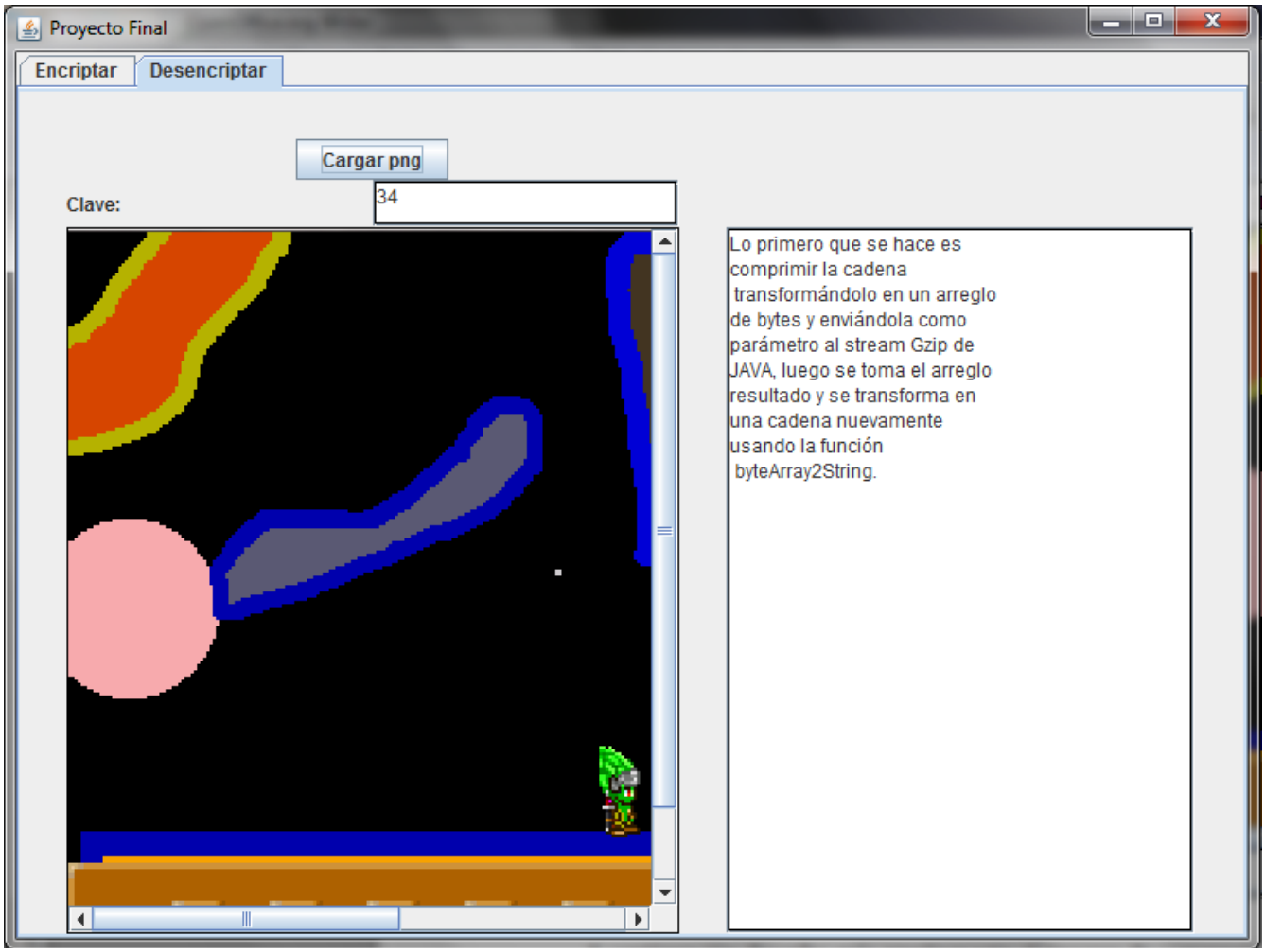
[1] Tecnología 21, http://tecnologia21.com/estenografia-esconder-archivo, consultado el 13/10/11

[2] Cryptography_An_Introducion_Book, Nigel Smart, pp 125,126

Anexo 1


Proyecto Final
Encriptar
Desencriptar
Imagen Original
Lo primero que se hace es comprimir la cadena transformándolo en un arreglo de bytes y enviándola como parámetro al stream Gzip de JAVA, luego se toma el arreglo resultado y se transforma en
Clave:
34
Encriptar Texto
Imagen con texto encriptado
Cargar png
Guardar png

Anexo 2


Proyecto Final
Encriptar
Desencriptar
Cargar png
Clave:
34
Lo primero que se hace es comprimir la cadena transformándolo en un arreglo de bytes y enviándola como parámetro al stream Gzip de JAVA, luego se toma el arreglo resultado y se transforma en una cadena nuevamente usando la función byteArray2String.