



Criptografia y Seguridad

Esteganografía

Trabajo Práctico Especial 2

Civile, Juan Pablo	50453
Crespo, Alvaro	50758
Susnisky, Dario	50592

10 de Junio de 2013

1. Introducción

En el presente Trabajo Práctico se muestran los resultados y análisis generados a partir del programa *stegobmp*, implementado según la especificación brindada por la cátedra. Dicho programa brinda la posibilidad de ocultar un archivo cualquiera en un archivo *bmp*, mediante un método de estenografiado específico, con la posibilidad de encriptarlo. De igual forma, el programa permite recuperar el archivo oculto a partir de un archivo *bmp*, que haya sido estenografiado con alguno de los métodos provistos.

Se presentan los resultados obtenidos a partir de las imágenes provistas por la cátedra, junto con el análisis de los mismos. También se incluye el análisis de algunas cuestiones interesantes, establecidas por la cátedra.

2. Desarrollo

2.1. Estegoanálisis de los archivos provistos

2.2. Cuestiones a analizar

1) Para la implementación del programa *stegobmp* se pide que la ocultación comience en el primer componente del primer pixel. ¿Sería mejor empezar en otra ubicación? ¿Por qué?

Dado que la ocultación que se utiliza se basa en el método LSB (*Least Significant Bit*), los bits “aprovechables” son pocos. En el caso de LSB1, se pueden utilizar 1 de cada 8 bits, (el bit menos significativo de cada byte). Si se usa LSB4, se pueden utilizar 4 de cada 8, lo que mejor bastante la cantidad de espacio “aprovechable”. Aún así, utilizando los últimos 4 bits de cada byte, ya se está introduciendo una disposición no menor, por lo que la cuestión del espacio resulta importante.

Por lo tanto, es conveniente empezar la ocultación desde el comienzo de la imagen, es decir el primer pixel, para poder disponer de toda la imagen para, en caso de ser necesario, ocultar en los bits menos significativos, la información del mensaje oculto.

Otra cuestión de interés, es que si se oculta información siempre en los bits menos significativos en orden, es más fácil localizar y extraer la información oculta, podría ser mejor solo ocultar en los bits menos significativos de una secuencia aleatoria basada en una clave o algo similar. De esta forma, el lugar donde se comienza la ocultación pierde su significado, ya que no importa por donde se comienza la ocultación, siempre y cuando se utilice todo el archivo para ocultar.

Otro aspecto importante a la hora de escoger el lugar de la imagen es el ruido, es decir, las zonas donde la imagen contiene ruido. Varios autores aseguran, y con razón, que lo mejor para minimizar la distorsión de la imagen al ocultar el mensaje, es ocultar el mensaje en las zonas más ruidosas de la imagen, ya que la distorsión atrae menos la atención. Para encontrar estas zonas, se utilizan filtros de selección de píxeles.

2) ¿Qué ventajas podría tener ocultar siempre en una misma componente? Por ejemplo, siempre en el bit menos significativo de la componente azul.

Es algo parecido a lo que se propone en el paper Enhanced Least Significant Bit algorithm For Image Steganography de Shilpa Gupta, Geeta Gujral y Neha Aggarwal. Se propone el “Enhanced Least Significant Bit (ELSB)”, que consiste justamente en ocultar siempre en la primer componente, es decir, en el bit menos significativo de la componente azul.

Las ventajas son las obvias. Se distorsiona menos la imagen, ya que solo se altera la banda de los azules. El salto de color es menor, ya que si, por ejemplo se cambiará la componente roja, el rango de colores que se saltea es mayor. Recordar que el color se define como la concatenación de 3 valores de 0 a 255, rojo, verde y azul, en ese orden, por lo que valor del color azul es menos significativo que los otros dos.

La desventaja es también clara: se requiere mayor tamaño de la imagen portadora. Para contrarrestar esto se podría hacer la modificación de solo cambiar el bit menos significativo, sino 2, 3 o 4, que igualmente generaría una menor distorsión y requieren menor tamaño. De hecho con usar 3 bits se requiere el mismo tamaño que al usar LSB-1, lo cual tiene sentido, es lo mismo, en términos de cantidades, tomar 1 bit de cada componente que tomar 3 de una sola componente.

3) Esteganografiar un mismo archivo en un .bmp con cada uno de los tres algoritmos, y comparar los resultados obtenidos. Hacer un cuadro comparativo de los tres algoritmos estableciendo ventajas y desventajas.

TODO Bueno, claramente cuantos mas bits modificas mas se va l carajo la imagen.
Ver bien...

Algoritmo	Ventajas	Desventajas
LSB-1	Menos distorsión	Requiere mayor tamaño del portador
LSB-4	Reduce tamaño requerido del portadora	Más distorsión
LSB Enhanced	Mínima distorsión	Muchísimo tamaño del portador. No hay una relación directa entre el tamaño del portador y del mensaje.

4) Para la implementación del programa *stegobmp* se pide que la extensión del archivo se oculte después del contenido completo del archivo. ¿por qué no conviene ponerla al comienzo, después del tamaño de archivo?

Podría argumentarse que esteganografiar la extensión por separado no es necesario. Pero lo cierto es que se necesita para rearmar el archivo rápidamente, y sin dejar librado al usuario del programa que deba adivinar o saber el formato del archivo oculto. Por que si bien el contenido entero podría ocultarse y recuperarse exitosamente, sin la extensión no se podría nombrar correctamente el archivo, y los sistemas operativos no sabrían con que programas abrirlos. Por otro lado, es cierto que no es una buena idea que un atacante pueda, descubriendo una secuencia de unos pocos bits que forman 4 chars identificables, tener mucha información respecto del archivo que se encuentra oculto. Es por esto que quizás, no sea conveniente poner la extensión al principio, donde puede ser más fácilmente encontrada. De todas formas, esto es discutible, ya que

si un atacante logra descifrar los 4 bytes que determinan el tamaño del archivo, podría encontrar con facilidad donde se encuentran los caracteres de la extensión, siempre y cuando no se utilice alguna forma de randomización o alteración del orden de los bits ocultados, como con el uso de una clave.

5) Explicar detalladamente el procedimiento realizado para descubrir qué se había ocultado en cada archivo y de qué modo.

En un principio, se intentó extraer cada archivo con todos los algoritmos de estenografiado, para ver a qué algoritmo respondía cada uno. De esta forma, se encontró que el archivo *miserables2.bmp* había sido estenografiado con el algoritmo LSB1, y el mensaje oculto era un archivo *.png*, el cual contenía una imagen de un tablero del juego *buscaminas*, a medio terminar. En segundo lugar, se encontró que el archivo *medianocheparis1.bmp* había sido estenografiado con el algoritmo LSB4, y el mensaje oculto era un archivo *.wmv*, pero no se podía reproducir correctamente con ningún reproductor de vídeo, tanto en Linux como en Windows como en Mac. Dejando ese llamativo descubrimiento de lado por un momento, se extrajo satisfactoriamente el archivo *.pdf* del archivo *loimposible.bmp*, el cual contenía el texto “La password es bienvenido”. Luego, se procedió a analizar el cuarto archivo, *secretodesusojos1.bmp*, y tras un largo análisis, se descubrió, vía un editor hexadecimal, que hacia el final del archivo, en la representación en caracteres ASCII, se podía leer el siguiente mensaje:

“al .png cambiar extension por .zip y descomprimir.”

Ante esta revelación, se siguieron dichas instrucciones, y lo que se obtuvo fue un pequeño archivo de texto, *sol8.txt*, que contenía lo siguiente:

*“cada mina es un 1.
cada fila forma una letra.
Los ascii de las letras empiezan todos en 01.
Así encontrarás el algoritmo que tiene clave de 128 bits y el modo
La password está en otro archivo
Con algoritmo, modo y password hay un .wmv encriptado y oculto.”*

Con este nuevo descubrimiento, se resolvió el mensaje encodeado en la imagen *.png*. **Resolviendo** el tablero del buscaminas, se obtuvo el siguiente resultado:

Binario	Decimal	Hexadecimal	ASCII
0100001	101	41	A
01100101	101	65	e
01110011	115	73 hex	s
01000011	67	43 hex	C
01100000	98	62 hex	b
01100011	99	63 hex	c

Al romper este código, la respuesta era clara, se debía usar el programa *stegobmp* para extraer correctamente el archivo *.wmv*, con el algoritmo de encriptación *aes128*, en modo *cbc* y utilizando como contraseña **bienvenido**.

Finalmente, al realizar la extracción con desencriptación del archivo *medianocheparis1.bmp*, el que había respondido al algoritmo LSB4 y había arrojado un *.wmv* corrupto, se obtuvo el archivo *.wmv* deseado. El video mostraba una escena de la película **Wanted (Se busca en castellano)** la cuál habla de una mística máquina de telar, que produce unas telas que tienen un código secreto, según la disposición de los hilos, que establece objetivos para una selecta comunidad secreta de asesinos.

6) ¿Qué se encontró en cada archivo?

Como se dijo en la respuesta anterior, estos son los resultados para cada archivo:

Archivo portador (BMP)	Alg. esteganografiado	Mensaje oculto
miserables2	LSB1	Imagen PNG(archivo ZIP c/ archivo de texto)
medianocheparis1	LSB4 + aes-128-cbc	Video WMV
loimposible	LSBE	Documento PDF
secretodesusojos1	Misterioso	Texto (pista para desencriptar el WMV)

7) Algunos mensajes ocultos tenían, a su vez, otros mensajes ocultos. Indica cuál era ese mensaje y cómo se había ocultado.

El mensaje oculto que consistía en una imagen *.png*, era a su vez, un archivo portador de otro mensaje oculto, y además en la imagen se encontraba un código que, descifrado, proporcionaba el algoritmo y el modo de encriptación para poder extraer correctamente otro de los mensajes ocultos.

Para empezar, al archivo se le podía cambiar la extensión a *.zip*, y descomprimirla, obteniendo un pequeño archivo de texto que contenía instrucciones a seguir. Algunas de esas instrucciones decían también, que la imagen contenía otro mensaje oculto, el algoritmo y el modo de encriptación para lograr extraer el otro de los mensajes ocultos. La imagen consistía en un tablero del juego *Buscaminas*, y si se completaba y se seguían las instrucciones para su decodificación, se podía extraer este otro mensaje oculto. En este caso era *aes* y *cbe*.

8) Uno de los archivos ocultos era una porción de un video, donde se ve ejemplificado una manera de ocultar información ¿cuál fue el portador?

Como se explicó en la pregunta 5, el archivo *.wmv* que estaba esteganografiado y encriptado, contenía escenas de una película muy conocida, en la que se ocultaban y extraían mensajes a través de telas. El método consistía en fijarse la disposición de los hilos y según como estaban entrelazados, se podía obtener un código binario que representaba un nombre de una persona. Esa persona era un objetivo para que una selecta comunidad secreta de asesinos matara.

9) ¿De qué se trató el método de estenografiado que no era LSB? ¿Es un método eficaz? ¿Por qué?

El método que no era LSB, fue un método un tanto “casero” y rudimentario. Lo que se hizo fue simplemente agregar al final de la imagen en formato *.bmp* un corto mensaje oculto, con una pista clave para conseguir la desencriptación de otro de los mensajes ocultos. La razón por la que el ocultamiento funciona es porque el tamaño de la imagen

(la cantidad de píxeles) está establecida en el header del archivo, y en eso se basan los programas que imprimen la imagen a pantalla. Si se agregan bytes luego del supuesto “final” de la imagen, es decir luego del último píxel contabilizado en el header, estos no se verán en los típicos programas que renderizan imágenes. Pero si, por el contrario, se utiliza un editor de texto hexadecimal, se puede ver que entre el supuesto final del la imagen y el EOF (el carácter *end of file*), puede existir información, como en el caso en cuestión, que no será percibida con los programas que comúnmente se utilizan para abrir imágenes.

Se dice que es una técnica un tanto rudimentaria ya que el ocultamiento se puede lograr sin mucho esfuerzo, ni conocimiento de algoritmos o formatos de archivos. En linux es tan simple como correr por línea de comandos:

```
echo "este es mi mensaje" >> imagen.bmp
```

o

```
cat mensaje.txt >> imagen.bmp
```

Donde *mensaje.txt* contiene el mensaje que se desea ocultar, y *imagen.bmp*.

Algo similar puede lograrse, desde Windows, corriendo:

```
imagen.bmp + mensaje.txt nuevaImagen.bmp
```

En donde no se modifica la imagen en cuestión, sino que se crea una nuevo archivo que contiene la imagen con el mensaje oculto en ella.

10) ¿Qué mejoras o futuras extensiones harías al programa *stegobmp*?

Una posible mejora al programa *stegobmp* es agregarle opcionalmente el uso de una contraseña de esteganografiado que defina el orden aleatorio en el que se ocultan los bits del mensaje oculto.

Otra opción, podría ser que tanto para extraer el mensaje oculto, se requiera el uso de la imagen original (cover image). Esto fuerza que, aparte de conocer la clave, se debe conocer el original de la imagen.

Algunas variantes para explorar podrían ser la aplicación de *Direct Cosine Transformation* o *Wavelet Transformation*, técnicas que son levemente más complejas que el esteganografiado con LSB.

Algo interesante, sería agregarle el soporte al programa para poder ocultar en otra clase de archivos portadores, como por ejemplo archivos de sonido, o de texto. Algunas técnicas orientadas a archivos de texto (que utilicen archivos de texto como portadores) como Line Shift Coding Protocol o Word Shift Coding Protocol serían muy interesantes, o White Space Manipulation (SNOW ya lo hace y es open-source).

A. Anexo



Figura 1: El secreto de sus ojos. Esteganografiada con el método “misterioso” (appendear texto al final del archivo BMP).



Figura 2: Los miserables. Esteganografiada con el método LSB1.

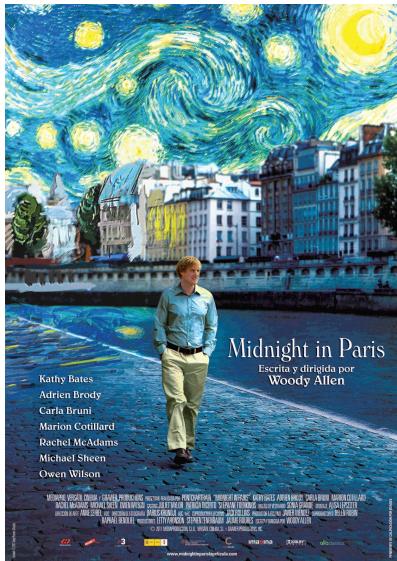


Figura 3: Los miserables. Esteganografiada con el método LSB4 y usando encriptación para el mensaje oculto usando aes de 128 bits en modo cbc y con contraseña *bienvenido*.



Figura 4: Lo imposible. Esteganografiada con el método LSBE.

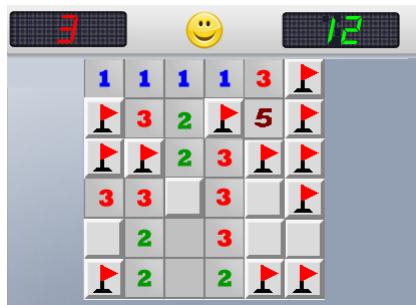


Figura 5: Imagen PNG. Obtenida utilizando el método LSB1 en la imagen de la figura 2. A la vez, se puede cambiar su extensión por ZIP y descomprimir para obtener un archivo de texto. Además tiene codificado dentro, en forma binaria el algoritmo y el modo de encriptación para desencriptar el archivo WMV oculto en la imagen de la figura 2.

La password es bienvenido

Figura 6: Documento PDF. Obtenido a partir de la imagen de la figura 4. Brinda la contraseña para desencriptar el archivo WMV oculto en la imagen de la figura 2.