



**Trabajo Práctico de Implementación:
Estenografía**

Segundo Cuatrimestre 2024

Pedro Jeremías López Guzmán	60711
Lucía Digon	59030
Eduardo Federico Madero Torres	59494
Martín E. Zahnd	60401

Grupo $2^4 - 1$


Índice

1. Introducción	1
2. Cuestiones a analizar	1
2.1. Aspectos relativos al documento base	1
2.1.1. Organización formal del documento	1
2.1.2. Descripción del algoritmo	2
2.1.3. Notación utilizada	2
2.2. Ventajas y desventajas de cada algoritmo	2
2.3. Metodología y Proceso de Descubrimiento	3
2.3.1. avatar.bmp	3
2.3.2. hugo1.bmp	4
2.3.3. bogota.bmp	7
2.3.4. madridoso.bmp	8
2.4. Video oculto	9
2.5. Otro método de estanografiado	9
2.6. Propuesta del documento de Majeed y Sulaiman	9
2.7. Otra forma de guardado	9
3. Dificultades encontradas	10
4. Mejoras o posibles extensiones	11
5. Conclusiones	11

1. Introducción

La esteganografía es una técnica que se utiliza desde la antigüedad para ocultar la existencia de mensajes. A diferencia de la criptografía, que hace ilegible la información, la estenografía busca ocultar la información dentro de otro archivo “portador”, de manera que la información pase desapercibida, ocultando su existencia.

El objetivo de este trabajo práctico, es desarrollar un programa stegobmp, que permite ocultar y extraer archivos en imágenes en formato “.bmp”, mediante distintos métodos de esteganografía. En el informe se detallará el proyecto realizado, al igual que un análisis de ventajas y desventajas y otras cuestiones a analizar.



2. Cuestiones a analizar

A continuación, se discutirá la organización formal y claridad del documento base, el análisis comparativo de los métodos de esteganografía empleados y los resultados del proceso de estegoanálisis aplicado a los archivos ocultos.

2.1. Aspectos relativos al documento base


Para el desarrollo del proyecto, se contó con el documento “An Improved LSB Image Steganography Technique using bit-inverse in 24 bit colour image” de Majeed y Sulaiman [1], en el que se detalla el método LSBI.

2.1.1. Organización formal del documento

En términos generales, el documento se encuentra estructurado de manera lógica, comenzando con una introducción al tema y explicando por qué es necesaria la mejora que se plantea en el documento.

Sin embargo, la introducción incluye una explicación de los conceptos básicos de esteganografía que, para un lector experto, resulta innecesaria.

A continuación, se comenta el algoritmo y los resultados de una simulación, cerrando con una conclusión sobre el análisis realizado.



2.1.2. Descripción del algoritmo

La descripción del algoritmo en la sección “5. THE PROPOSED BIT INVERSION TECHNIQUE” es clara, mas la sección siguiente, “6. THE PROPOSED BIT INVERSION ALGORITHM”, presenta el algoritmo de manera confusa y requiere varias lecturas del mismo para comprenderlo. ✓

2.1.3. Notación utilizada

La notación no es consistente a lo largo del documento. En particular, se puede observar que ambos algoritmos, en las secciones “A LSB-BASED EMBEDDING AND EXTRACTING SECRET DATA ALGORITHM” y “6. THE PROPOSED BIT INVERSION ALGORITHM”, son descritos con notación sumamente diferente. ✓

2.2. Ventajas y desventajas de cada algoritmo

Con el fin de poder analizar las ventajas y desventajas de cada algoritmo, se decidió estenografiar un mismo archivo en un .bmp con cada uno de los tres algoritmos y comparar los resultados obtenidos. Para eso, se decidió utilizar el comando

```
< /dev/urandom tr -dc '[:alnum:]' | head -c 100000 > input_file.txt
```

Para generar un archivo de 100000 bytes y un bmp de 1944054 bytes.

Como se puede observar en la Tabla 1, el algoritmo LSBI es el algoritmo menos indicado cuando el tamaño importa. ✓

Método	Tamaño máximo del mensaje	Cantidad de bytes modificados
LSB1	Tamaño de imagen / 8	800072
LSB4	Tamaño de imagen / 2	200018
LSBI	2 * Tamaño de imagen / 3 / 8 - 4	1200112

Tabla 1: Utilizando un archivo de entrada de 100000 bytes

Método	Ventajas	Desventajas
LSB1	<ul style="list-style-type: none"> - Método simple y de fácil implementación. - Baja distorsión en la imagen portadora. 	<ul style="list-style-type: none"> - Baja capacidad de almacenamiento. - Fácil de detectar mediante análisis estadísticos.
LSB4	<ul style="list-style-type: none"> - Mayor capacidad de almacenamiento en comparación con LSB1. - Menor distorsión que otros métodos de alta capacidad. 	<ul style="list-style-type: none"> - Puede producir distorsión visible en la imagen. - Más susceptible a detección debido a la mayor modificación de los bits.
LSBI	<ul style="list-style-type: none"> - Mejora la seguridad mediante la inversión de bits en patrones específicos. - Difícil de detectar en comparación con LSB1 y LSB4. 	<ul style="list-style-type: none"> - Complejidad de implementación mayor que LSB1 y LSB4. - Requiere más procesamiento, lo que puede afectar el rendimiento.

Tabla 2: Comparación de ventajas y desventajas entre los métodos de esteganografía

2.3. Metodología y Proceso de Descubrimiento

Para comenzar nuestro análisis, establecimos una metodología sistemática explorando todas las posibles combinaciones de extracción sin encriptación para cada uno de los métodos implementados: LSB1, LSB4 y LSBI. La cátedra proporcionó 4 archivos: *avatar.bmp*, *bogota.bmp*, *hugo1.bmp* y *madridoso.bmp*.

2.3.1. avatar.bmp

Inicialmente se realizaron las siguientes pruebas:

```
./stegobmp -extract -p avatar.bmp -out avatar_lsbi -steg LSBI
./stegobmp -extract -p avatar.bmp -out avatar_ls4 -steg LSB4
./stegobmp -extract -p avatar.bmp -out avatar_ls1 -steg LSB1
```

Ninguno de los tres métodos produjo resultados interpretables, generándose únicamente archivos de salida vacíos.

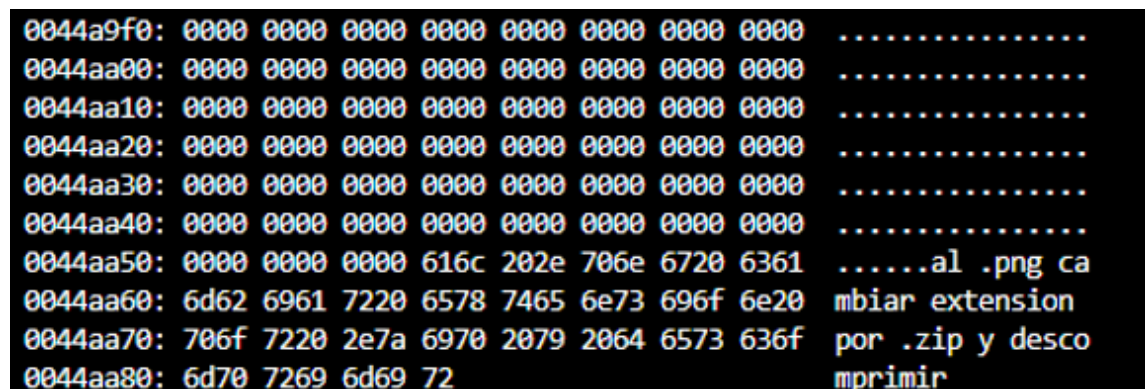
Para ver si había algún mensaje en texto plano dentro del archivo, se utilizó el comando `strings` de la siguiente manera:

```
strings avatar.bmp
```

Lo cual arrojó

```
...  
ECBuuu  
tttttt  
ttt111  
al .png cambiar extension por .zip y descomprimir
```

Esto mostró el mensaje oculto, y se utilizó la herramienta `xxd` para encontrar en dónde estaba el mensaje.



```
0044a9f0: 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0044aa00: 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0044aa10: 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0044aa20: 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0044aa30: 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0044aa40: 0000 0000 0000 0000 0000 0000 0000 0000 .....  
0044aa50: 0000 0000 0000 616c 202e 706e 6720 6361 .....al .png ca  
0044aa60: 6d62 6961 7220 6578 7465 6e73 696f 6e20 mbiar extension  
0044aa70: 706f 7220 2e7a 6970 2079 2064 6573 636f por .zip y desco  
0044aa80: 6d70 7269 6d69 72 mprimir
```

Figura 1: Análisis Binario

2.3.2. hugo1.bmp

La extracción con LSB1 produjo resultados significativos:

```
./stegobmp -extract -p hugo1.bmp -out salida -steg LSB1
```

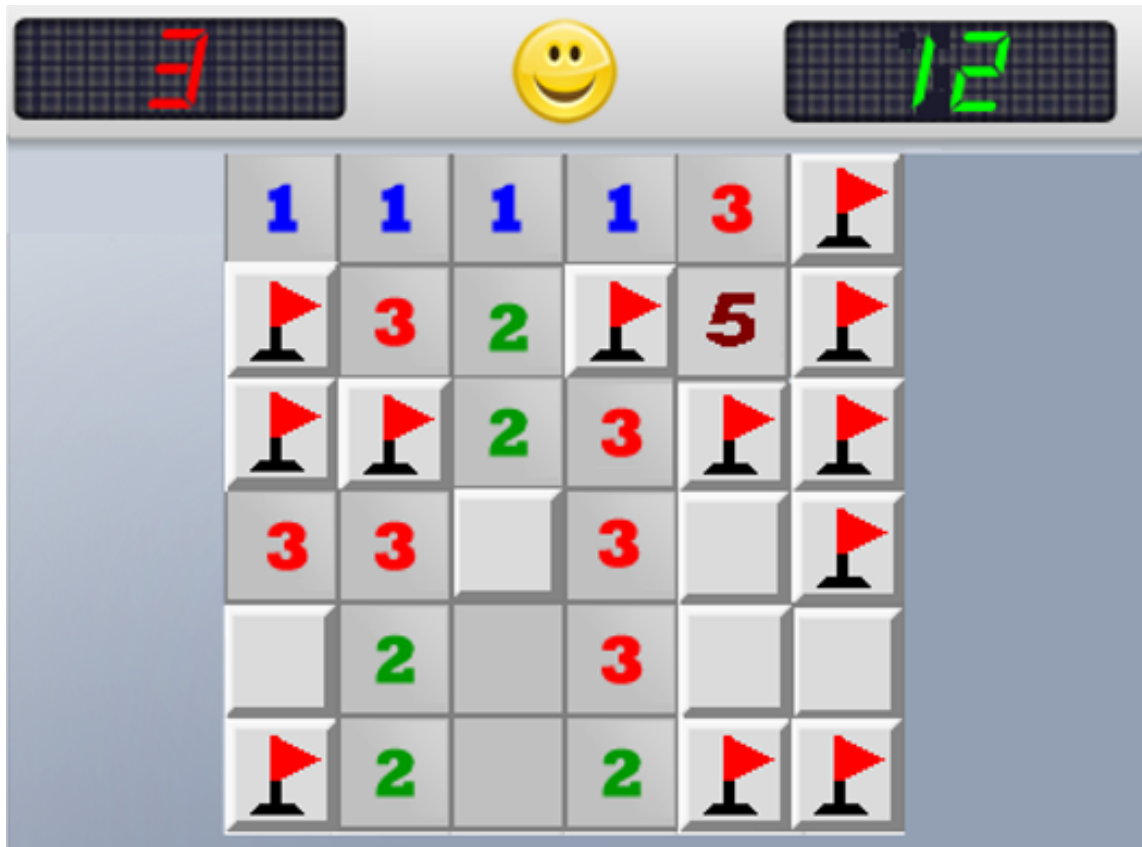


Figura 2: Imagen PNG obtenida

Este comando generó una imagen PNG.

Siguiendo las instrucciones descubiertas previamente, se cambió la extensión del archivo a .ZIP, lo cual permitió obtener un archivo de texto con las siguientes instrucciones:

- Cada mina es un 1
- Cada fila forma una letra
- Así encontrarás el algoritmo que tiene clave de 256 bits y el modo
- La password está en otro archivo
- Con algoritmo, modo y password hay un .wmv encriptado y oculto



Siguiendo estas instrucciones, se procedió a analizar el buscaminas donde:

1. Mina (bandera) = 1
2. Número visible/caja vacía = 0

Sabiendo que todas las filas comienzan por 01, se tomaron los 6 bits como los menos significativos del byte.



Figura 3: Solución de Buscaminas

El análisis fila por fila reveló:

- Fila 1: 01000001 = 'A'
- Fila 2: 01100101 = 'e'

La password es bienvenido



Figura 4: Contenido del PDF obtenido

- Fila 3: 01110011 = 's'
- Fila 4: 01000011 = 'C'
- Fila 5: 01000010 = 'B'
- Fila 6: 01100011 = 'C'

Esto proporcionó “AesCBC”, lo cual indica:

- Algoritmo: AES
- Clave: 256 bits
- Modo: CBC

2.3.3. bogota.bmp

A continuación, utilizando LSBI, se logró extraer un archivo PDF:

```
./stegobmp -extract -p bogota.bmp -out salida -steg LSBI
```

El PDF contenía la contraseña necesaria para la extracción final, siendo esta la palabra “bienvenido”.

2.3.4. madridoso.bmp

Finalmente, con toda la información recopilada, se procedió a extraer el video oculto:

```
./stegobmp -extract -p madridoso.bmp -out salida -steg LSB4  
-a aes256 -m cbc -pass "bienvenido"
```

La estructura del comando final fue:

- PASSWORD: “bienvenido”
- Algoritmo: AES-256
- Modo: cbc

Finalmente, se obtuvo un archivo de video que era el que se buscaba obtener.



Figura 5: Video obtenido

2.4. Video oculto

En el video se puede observar a una mujer que detecta una anomalía en el tamaño del archivo, que es mayor de lo que debería ser. De esta forma, se puede ver que el portador es el archivo en el que se envió, donde evidentemente se está escondiendo un mensaje, el cual es desconocido. El problema de este método es que algo tan sencillo como el tamaño del archivo levanta sospechas, y por lo tanto no es un muy buen método.



2.5. Otro método de estanografiado

El método de esteganografiado que no era LSB1, LSB4 ni LSBI consistió en almacenar el mensaje directamente en el archivo BMP en texto plano.

Herramientas como `strings` hacen de este método una pésima opción porque es posible recuperar el mensaje independientemente de la posición del mismo dentro del archivo.



2.6. Propuesta del documento de Majeed y Sulaiman

La propuesta de Majeed y Sulaiman representa una mejora sobre el método LSB común ya que el método propuesto hace que más difícil la detección de la información oculta a simple vista. En el método LSB común se reemplazan los bits menos significativos de los pixeles de la imagen para almacenar los datos. Si bien esto resulta sencillo, también aumenta la posibilidad de que sean visibles imperfecciones en la imagen y generen sospechas al receptor.



La más importante de las mejoras de LSBI es utilizar el color rojo, que es visible con el 65 % de nuestros conos [1], como ruido para quien esté observando la imagen.

2.7. Otra forma de guardado

Podrían ser guardados al final del archivo, o dentro del header BMP utilizando los bytes reservados, que actualmente están en desuso.



Además, otro método sería utilizar este patrón como parte de la key. ✓

De todas formas, al ser 4 bits sólo hay 16 posibles combinaciones, por lo cual es muy sencillo descubrirlos por fuerza bruta. B

3. Dificultades encontradas

Una dificultad encontrada a la hora de implementación fue con la encriptación/desencriptación de los archivos. Una vez que obtenidas las instrucciones y la clave para desencriptar el archivo *madridoso.bmp*, se observó una falla en la desencriptación. De esta manera, se pudo detectar que había un problema. Luego de varias pruebas y de revisar el código, se pudo ver que el problema era que se estaba aplicando mal la salt en la función de OpenSSL. Hubo un error de interpretación, se interpretó que la salt era un entero en lugar de un arreglo de 8 bytes casteado. Una vez detectado el error, la solución fue bastante simple y se pudo extraer correctamente el archivo.

El diff del código es el siguiente:

```
- static const uint8_t *__SALT = 0x0000000000000000;  
+ static const unsigned char *__SALT[8] = {0, 0, 0, 0, 0, 0, 0, 0};
```

Por otra parte, extraer los bits en orden para almacenarlos en los bytes del archivo significó una cuidadosa y precisa manipulación de los valores. Esto llevó su tiempo y una evaluación exhaustiva con debuggers.

Por último, también se tuvo especial consideración al momento de analizar las estructuras utilizadas. Para los headers del archivo bmp se utilizaron dos estructuras, BmpHeader y BmpHeaderInfo. El problema es que el Header del archivo ocupa 14 bytes, pero al realizar una estructura en C de ese tamaño, el compilador optimiza la lectura a la estructura agregando padding para que el tamaño sea de 16 bytes, arruinando la posibilidad de hacer una lectura directa del archivo a la estructura. Se solucionó deshabilitando el padding en las declaraciones de ambas estructuras. ✓

4. Mejoras o posibles extensiones

Entre las mejoras y posibles extensiones que se podrían implementar en el programa stegobmp, además de las ya mencionadas como el soporte para más formatos de archivo y la generalización del algoritmo LSB, se podrían incluir: la implementación de técnicas de compresión previas al ocultamiento de datos para optimizar el espacio utilizado, la incorporación de mecanismos de detección de capacidad máxima que alerten al usuario antes de comenzar el proceso, y la adición de métodos de verificación de integridad para asegurar que el mensaje extraído coincida exactamente con el mensaje original embebido. También sería valioso agregar soporte para el procesamiento por lotes (batch processing) que permita trabajar con múltiples archivos simultáneamente. Adicionalmente, se podría considerar la implementación de técnicas de watermarking digital para complementar la funcionalidad actual del programa.



5. Conclusiones

En el trabajo práctico se implementaron y analizaron distintos métodos de estenografía en imágenes BMP. La secuencia de descubrimientos reveló un diseño cuidadoso en la ocultación de información, donde cada archivo contenía pistas o información necesaria para la extracción exitosa del siguiente, culminando en la recuperación del archivo de video encriptado.

Referencias

- [1] M. A. Majeed and R. Sulaiman, "An improved lsb image steganography technique using bit-inverse in 24 bit colour image," *Journal of Theoretical and Applied Information Technology*, 2015.

