

Instituto Tecnológico de Buenos Aires



72.44 - Criptografía y Seguridad

TPE

Grupo 11

(62028) Nicolás Matías Margenat - nmargenat@itba.edu.ar

(62094) Juan Burda - jburda@itba.edu.ar

(62504) Elian Paredes - eparedes@itba.edu.ar

1. Introducción	2
2. Relativo al paper	2
2.1. Organización formal	2
2.2. Descripción del algoritmo	2
2.3. Notación y claridad	3
3. Comparación de algoritmos	4
4. LSBI	5
4.1. ¿Por qué es mejor?	5
4.2. Lugares para guardar la información de patrones invertidos	5
5. Estegoanálisis	6
6. Dificultades encontradas	8
7. Futuras extensiones	8

1. Introducción

En el presente informe se detallan algunos aspectos relativos al Trabajo Práctico (TP) que consistió en la implementación de diversos métodos de esteganografía.

En cuanto al enunciado del TP, se detallarán a continuación las secciones junto con las preguntas (“7. Cuestiones a analizar”) que responden.

- Sección 2 → I
- Sección 3 → II
- Sección 4 → VII, VIII
- Sección 5 → III, IV, V, VI
- Sección 6 → IX
- Sección 7 → X

2. Relativo al paper

2.1. Organización formal

El paper comienza con un resumen que introduce brevemente la esteganografía y describe la técnica propuesta de inversión de bits para mejorar la calidad de las imágenes esteganográficas en imágenes de color de 24 bits. La introducción proporciona contexto histórico sobre la esteganografía y explica su relevancia en la comunicación digital moderna.

A continuación, se explica la técnica estándar del bit menos significativo (LSB1), describiendo su funcionamiento básico para ocultar información en imágenes. Luego, se detalla cómo se representan las imágenes de color utilizando el modelo RGB en el contexto de imágenes de 24 bits.

El paper profundiza en el proceso de inserción y extracción de datos utilizando el método LSB1, antes de presentar la nueva técnica de inversión de bits propuesta. Esta sección detalla cómo el nuevo método mejora tanto la seguridad como la calidad de la imagen esteganográfica.

Los autores proporcionan los pasos detallados del algoritmo desarrollado, seguidos por los resultados de simulación. Esta sección presenta los resultados experimentales, incluyendo análisis de histogramas y valores PSNR (técnica estándar para testear la calidad de estego-imágenes) para diferentes imágenes de prueba, demostrando la eficacia del método propuesto.

El paper concluye con una discusión que resume los hallazgos, destaca las mejoras en seguridad y calidad de imagen, y confirma la efectividad del método propuesto.

2.2. Descripción del algoritmo

La descripción del algoritmo es clara, proporcionando tanto una descripción coloquial como pseudocódigo. En el caso de nuestro equipo, la manera más clara de entender el algoritmo resultó ser mirando la lista numerada de cada uno de los pasos que se muestra al

final de la sección 5 del paper. El código implementado sigue esa estructura, pero no se adhiere estrictamente al pseudocódigo presentado por los autores en el paper.

2.3. Notación y claridad

La claridad del paper deja bastante que desear. Hay errores de tipeo y mal uso del inglés que por momentos dificultan el entendimiento.

En la sección 5 del paper hay un error de tipeo que dificultó y retrasó la implementación del método propuesto (ver Figura 1) ya que en uno de los bytes falta uno de los bits, y los autores explican luego que para ese byte el valor no cambia pero si se observa el valor anterior el valor parece que cambia. No fue hasta que el equipo contó los bits que se evidenció este error de tipeo.

Secret message	: 1011	
Cover image	: 10001100	10101101
	A	B
	10101011	1010110□
	C	D
LSB stego-image	: 10001101	10101100
	A	B
	10101011	10101101
	C	D

Figura 1. Error de tipeo.

A lo largo del texto hay numerosos errores debido al mal uso del idioma inglés, en la Figura 2 se pueden ver algunos presentes en la sección 5 y 6. Estos errores entorpecen la lectura, dificultando el entendimiento del texto.

- | | |
|---|--|
| 3. From the stego-image, once again, calculate the pattern occurrences in the 2nd last and 3rd last bit of the stego-image. <u>From these patterns, we use them as a guide for the inversion steps.</u> | If msg bit = 1
Insert a 1 in the <u>lest</u> significant bit of the pixel value |
|---|--|

Figura 2. Mal uso del inglés.

En la sección 6 los autores proponen un pseudocódigo para implementar su algoritmo (ver Figura 3), y se utiliza un valor n en el primer *for loop* que nunca se explica a qué hace referencia, pues no se sabe si se trata de los n bytes del tamaño de la imagen portadora, o si se trata de los bytes que ocupa la imagen a ocultar.


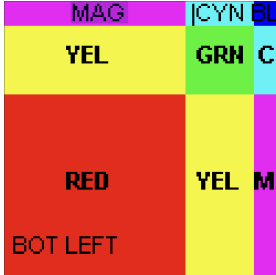
```


pic = cover image
msg = secret message
For i = 1 to n
    Get char from msg
    For each 2 bits



```

Figura 3. Notación no explicada.

3. Comparación de algoritmos

Recursos utilizados			
Carrier		Payload	
	Tamaño: 1,44 MB Dimensiones: 800 x 600 píxeles Profundidad de color: 24 bits Compresión: No		Tamaño: 46 KB Dimensiones: 124 x 124 píxeles Profundidad de color: 24 bits Compresión: No

Comparación entre algoritmos			
Algoritmo	Características	Observaciones	Resultado
LSB1	<ul style="list-style-type: none"> Sustituye el último bit de cada byte de la imagen. Alta capacidad de ocultación. Fácil de detectar. 	A simple vista, no se puede detectar la presencia del archivo embebido dentro de la imagen portadora.	

LSB4	<ul style="list-style-type: none"> • Sustituye los últimos 4 bits de cada byte. • Mayor capacidad de almacenamiento. • Fácil de detectar. 	Se logra observar en la parte inferior de la imagen cierta distorsión, que da un indicio de que existe algún archivo embebido en la misma	
LSBI	<ul style="list-style-type: none"> • Usa una técnica adaptativa para modificar sólo algunos bits. • Minimiza el cambio visual en la imagen. • Optimiza el balance entre capacidad y calidad. 	Al igual que ocurre con LSB1, no se detecta a simple vista la presencia del archivo embebido.	

4. LSBI

4.1. ¿Por qué es mejor?

La propuesta de Majeed y Sulaiman resulta mejor pues modifica menos la imagen que cualquiera de los otros algoritmos. Esto sucede porque invierte aquellos bytes con el mismo patrón en el 2do y 3er bit menos significativo si se cumple que cambian su último bit más de lo que no cambian. Además, en ningún momento se toca el byte que corresponde al canal rojo de la imagen, siendo que este es el canal al que nuestro ojo es más sensible (según citas de los autores en el paper). Consecuentemente, al observar la imagen a simple vista no podemos detectar que hay algo oculto (como sí suele suceder con LSB4, por ejemplo), y cuando se realiza un análisis mediante PSNR tampoco se notan diferencias significativas.

4.2. Lugares para guardar la información de patrones invertidos

En la implementación realizada se guarda la información de los patrones invertidos al comienzo del archivo. Una alternativa posible sería guardarlo al final del archivo, o incluso en cualquier posición prefijada del archivo siempre y cuando se tenga el cuidado de no incluir información del payload en dicha posición.

5. Estegoanálisis

Luego de haber implementado los algoritmos de esteganografiado tanto para embeber como para extraer archivos se procedió a la etapa de estegoanálisis. En esta etapa, se debían analizar los archivos *back.bmp*, *kings.bmp*, *roma.bmp* y *sherlock.bmp* provistos por la cátedra con el motivo de descubrir los mensajes ocultos. Además, se sabe por enunciado que algunos mensajes ocultos contienen, a su vez, otros mensajes ocultos, que uno de los archivos contiene el fragmento de una película y que otro de los archivos contiene un mensaje que no está esteganografiado con LSB1, LSB4 ni LSBI.

En un primer momento se intentó una aproximación de “fuerza bruta” utilizando los 3 modos de esteganografía para extraer los archivos ocultos sin encriptación, ya que no se contaba con ninguna contraseña para descryptar los mismos.

Para el archivo *roma.bmp* con LSB4 se extrajo el siguiente archivo .png:



Figura 4: archivo oculto extraído de *roma.bmp*

A su vez, mediante LSBI se extrajo un archivo .pdf del archivo *back.bmp*, que, al abrirlo, contenía en texto “al .png cambiarle la extensión por .zip y descomprimir”

Para los dos archivos restantes, no se pudo extraer ningún archivo válido utilizando los 3 métodos de esteganografía. Sin embargo, utilizando el comando “strings” en *kings.bmp* se descubrió el texto “la password es desafío” al final del mismo.

Dado que no dió resultado esta estrategia con el archivo restante *sherlock.bmp*, se decidió seguir las instrucciones del archivo .pdf, por lo que se cambió la extensión del archivo de la Figura 4 de .png a .zip y se descomprimió, dando como resultado el archivo *sol11.txt* que indica cómo interpretar la imagen del buscaminas para obtener tanto el algoritmo como el modo del archivo encriptado:

```
1  cada mina es un 1.  
2  cada fila forma una letra.  
3  Los ascii de las letras empiezan todos en 01.  
4  Asi encontraras el algoritmo que tiene clave de 192 bits y el modo  
5  La password esta en otro archivo  
6  Con algoritmo, modo y password hay un .wmv encriptado y oculto. |
```

Figura 5: archivo *sol11.txt* extraído de *roma.zip*

Resolviendo el código del buscaminas se obtuvo “AesOfb”, indicando tanto el algoritmo como el modo.

Entonces, el archivo *roma.bmp* contenía 3 mensajes ocultos: el primero oculto por esteganografía dando como resultado un .png, el segundo dentro del mismo archivo interpretando las celdas del buscaminas, y el tercero cambiando la extensión del archivo a .zip.

Finalmente, al desencriptar con algoritmo AES con clave de 192, modo Ofb, LSB1 y contraseña “desafío” (obtenida del archivo *kings.bmp*), se obtuvo la porción de película en la que dos personajes descubren y explican un algoritmo de encriptación oculto en un telar.



Figura 6: telar de la película en la que se oculta un mensaje secreto

En el vídeo se explica como si el hilo vertical de la trama está encima representa un 1, mientras que si está por debajo representa un 0, de modo tal que se puede obtener un código binario que indica “... un nombre y un objetivo” si se convierte a texto.

Por otro lado, el método de esteganografiado distinto a LSB1, LSB4 y LSBI es el encontrado tanto en la imagen .png de *roma.bmp* como en el archivo *kings.bmp*. El mismo consiste en colocar el stream de bytes que se quiere ocultar al final del archivo de forma continua.

Este método no es eficaz en términos de almacenamiento debido a que incrementa el tamaño del archivo tanto como la información que se quiere ocultar, ya que no altera bytes de la imagen sino que anexa los mismos al final del archivo. Por otro lado, no es muy seguro debido a que toda la información oculta está de forma continua y no entremezclada. Para el caso del .zip oculto, es fácilmente detectable mediante un editor de binarios (cómo Hex Editor) identificando el header del archivo. Para el caso del texto oculto, el comando “strings” que se utilizó en el estegoanálisis imprime todo el texto legible en el archivo, haciendo este método trivial.

6. Dificultades encontradas

El principal problema que se tuvo al implementar LSBI fue entender el paper. Como se mencionó en la sección 2, había numerosos errores tanto de semántica y ortografía, como también errores de tipeo en partes clave del paper.

Otro problema que se tuvo fue, dado que se deben saltar los bytes rojos, se debe contemplar que no siempre se iniciará a leer la información a partir de un byte azul. Teniendo esto en cuenta se armaron dos funciones. La primer función *get_color_from_byte_index* devuelve el color de un byte dado su offset en el array de colores; la otra función toma como argumento un color y devuelve el siguiente en el patrón.

7. Futuras extensiones

En cuanto a las mejoras que se pueden introducir en el programa implementado sería optimizar el LSBI, ya que se realizan iteraciones demás con el objetivo de facilitar el entendimiento del código y evitar errores por parte del equipo.

Se podrían agregar punteros a funciones para facilitar el manejo de los distintos algoritmos en *subcommands.c*, evitando así tener tantos *if-else* que dificultan la lectura del código.

El manejo de errores también podría ser más riguroso, y se deberían mostrar mensajes de error entendibles por el usuario para cada problema que se produzca.

Se podrían introducir más algoritmos de esteganografiado, así como también más opciones de encriptación y modos.

Finalmente, se podría haber realizado una UI muy simple para evitar que el usuario tenga que introducir los parámetros por línea de comandos. El equipo pensó una implementación posible con dos ventanas *drag-and-drop*, una para el archivo portador y otro para el *payload*. Se podrían poner también *dropdowns* con las distintas opciones de algoritmos, tanto de esteganografiado como de encriptación y sus modos, y una barra simple para introducir una contraseña. De esta manera, el programa sería mucho más usable desde un punto de vista de UX/UI, haciéndolo más fácil de utilizar.