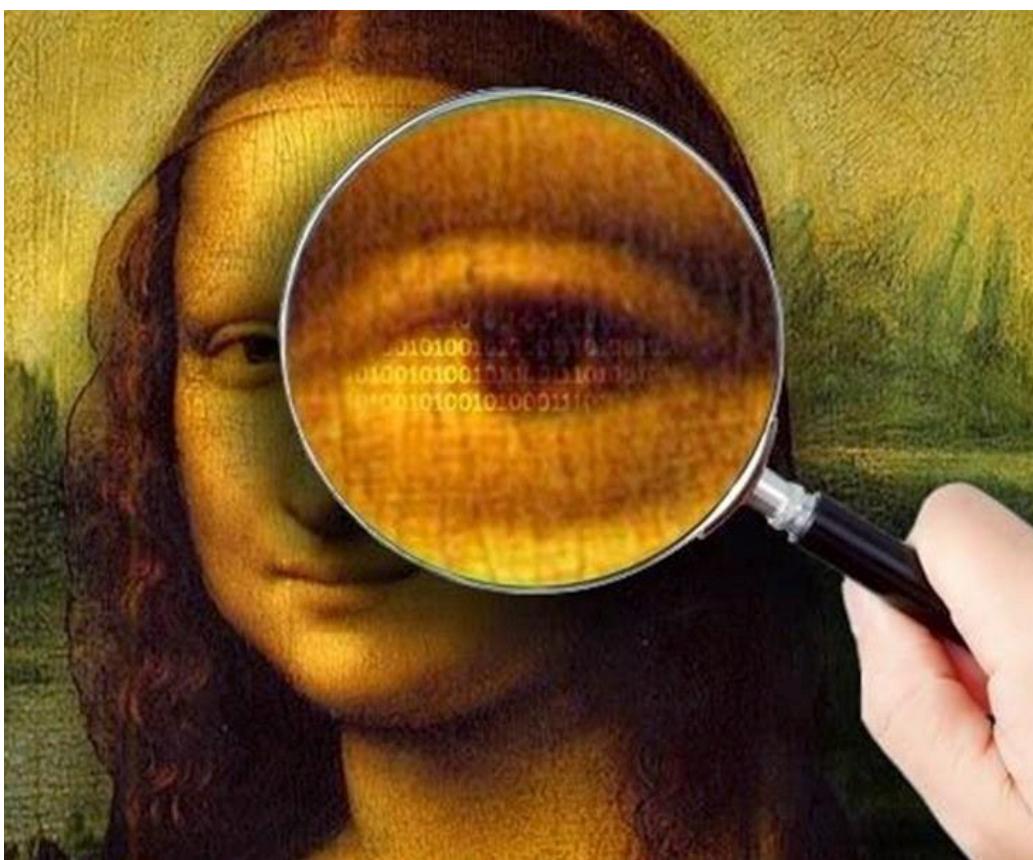


Criptografía y Seguridad

Primer Cuatrimestre 2022

Trabajo Práctico Especial

"Esteganografía - Cuestiones a analizar"



Integrantes del grupo

Tomas Cerdeira - 60051

Ignacio Villanueva - 59000

Santiago Garcia Montagner - 60352

Fecha de Entrega: 25 de junio de 2022

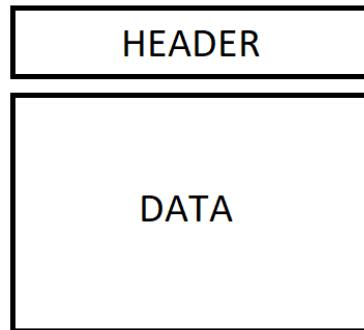
1. Discutir los siguientes aspectos relativos al documento.

a. Organización formal del documento.

b. La descripción del algoritmo.

c. La notación utilizada, ¿Es clara? ¿Hay algún error o contradicción?

a) Son 2 los documentos que entran en juego: el archivo a ocultar y la imagen (en este caso .bmp) portadora. Ambos archivos, en términos generales, siguen la siguiente “arquitectura”:



- **Header:** Es un bloque de bytes que se encuentra al principio del archivo y se utiliza para identificar el archivo. Las aplicaciones suelen leer este primer bloque para asegurarse de que el archivo es realmente de una formato soportado y que no está dañado.

En nuestro caso, al utilizar únicamente archivos de tipo BMP, el header de los mismos sigue el siguiente estándar:

Offset hex	Offset dec	Size	Purpose
00	0	2 bytes	The header field used to identify the BMP and DIB file is <code>0x42 0x4D</code> in hexadecimal , same as <code>BM</code> in ASCII. The following entries are possible: BM Windows 3.1x, 95, NT, ... etc. BA OS/2 struct bitmap array CI OS/2 struct color icon CP OS/2 const color pointer IC OS/2 struct icon PT OS/2 pointer
02	2	4 bytes	The size of the BMP file in bytes
06	6	2 bytes	Reserved; actual value depends on the application that creates the image, if created manually can be 0
08	8	2 bytes	Reserved; actual value depends on the application that creates the image, if created manually can be 0
0A	10	4 bytes	The offset, i.e. starting address, of the byte where the bitmap image data (pixel array) can be found.

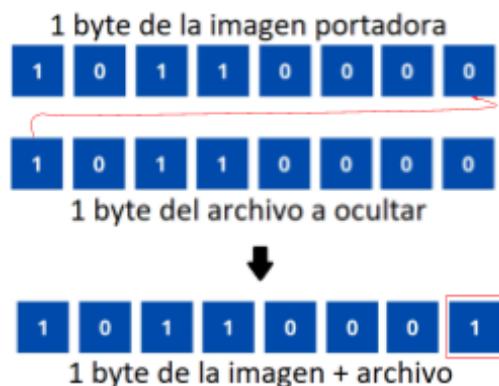
- **Data:** Aquí se encuentran los bytes en sí del archivo.

Leyendo del header los bytes entre 02 y 06 (en base a la imagen anterior), obtenemos el tamaño del archivo sabiendo así cuando comienza la data/bytes del (al terminar el header) y cuando se debe dejar de leer (una vez leídos el size).

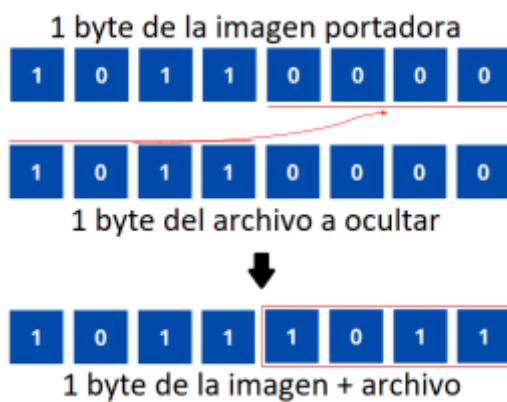
b) Para la implementación de este trabajo, tuvimos que investigar e implementar 3 algoritmos distintos de la misma familia: LSB1, LSB4 y LSBImproved.

Las siglas LSB significan “Least Significant Bit”, o en español “Bit menos significativo”. Esta familia de algoritmos se basa en modificar en el caso de LSB1 y LSBImproved un único bit de cada byte (en LSBImproved no siempre) y en el caso del LSB4 4 bits de cada byte, de la imagen portadora y cambiarlos por los del archivo a ocultar.

LSB1 - LSBImproved:



LSB4:



c) Error o contradicción no que hayamos visto, leído o encontrado... Con respecto a la notación, LSB1 y LSB4 si, es bastante clara y simple de entender a diferencia del LSBImproved.

2. 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.

- Cover Image:

Trabajo Práctico Especial

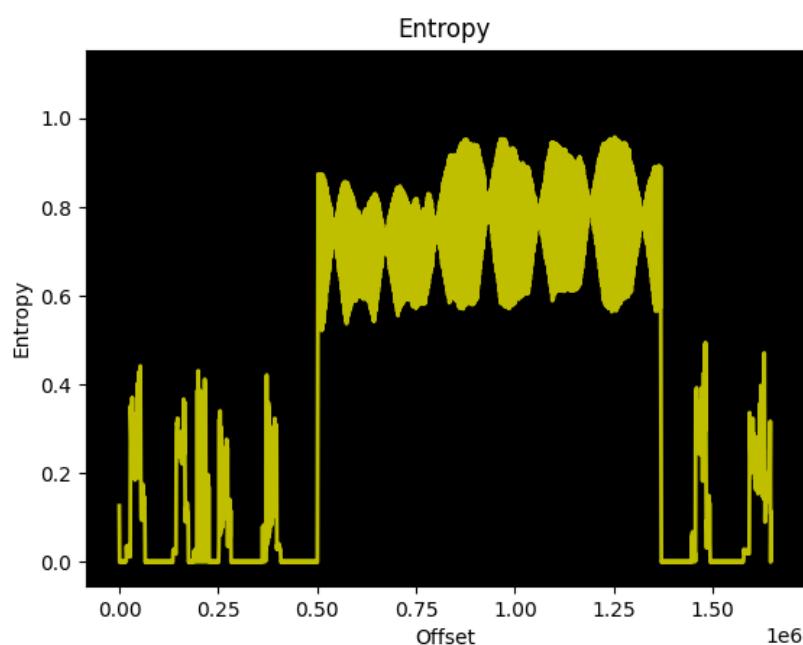
"Esteganografia - Cuestiones a analizar"



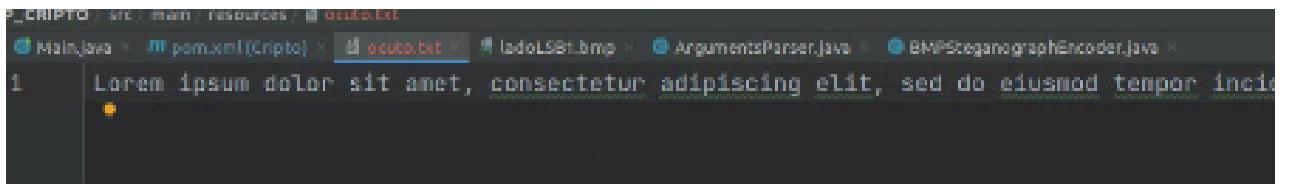
Integrantes del grupo

Tomas Cerdeira - 60051
Ignacio Villanueva - 59000
Santiago Garcia Montagner - 60352

Fecha de Entrega: 25 de junio de 2022



- Archivo a ocultar:



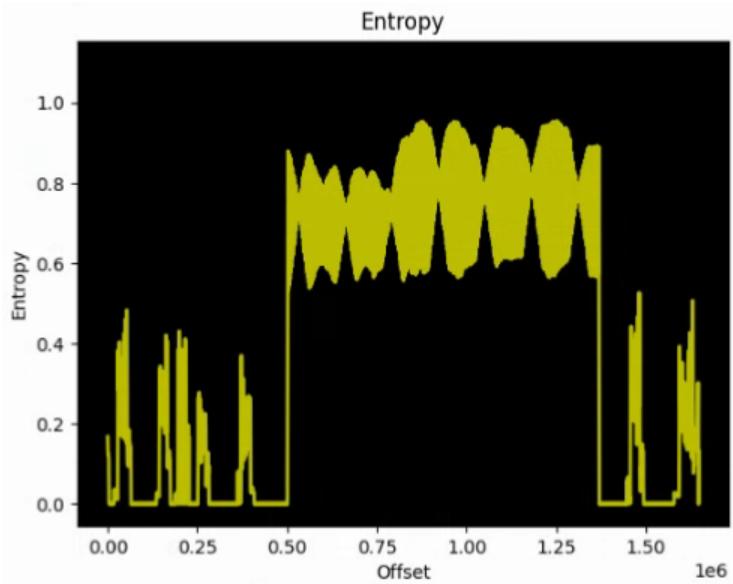
```

P_CRIPTO /src/main/resources/ > oculta.txt
Main.java < pax.xml(Cripto) < oculta.txt < lodoLSB1.bmp < ArgumentsParser.java < BMSteganographEncoder.java <
1 | Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.
| .

```

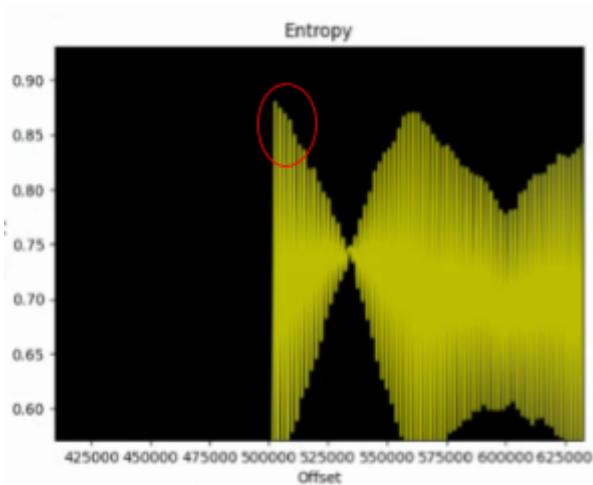
- Stego Image:

LSB1

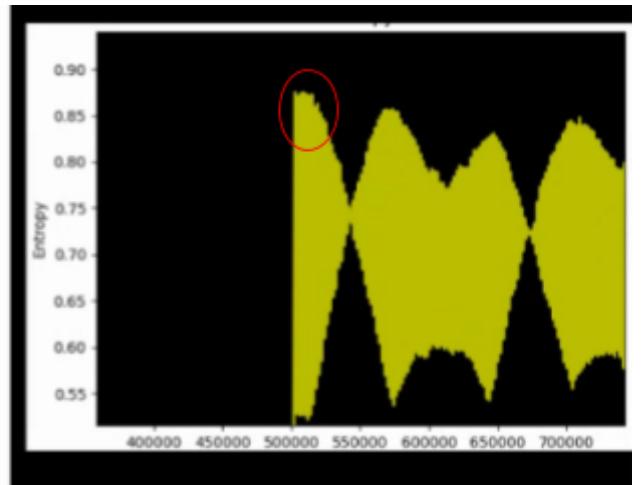


Comparando alrededor del offset 0.5 y la con la original:

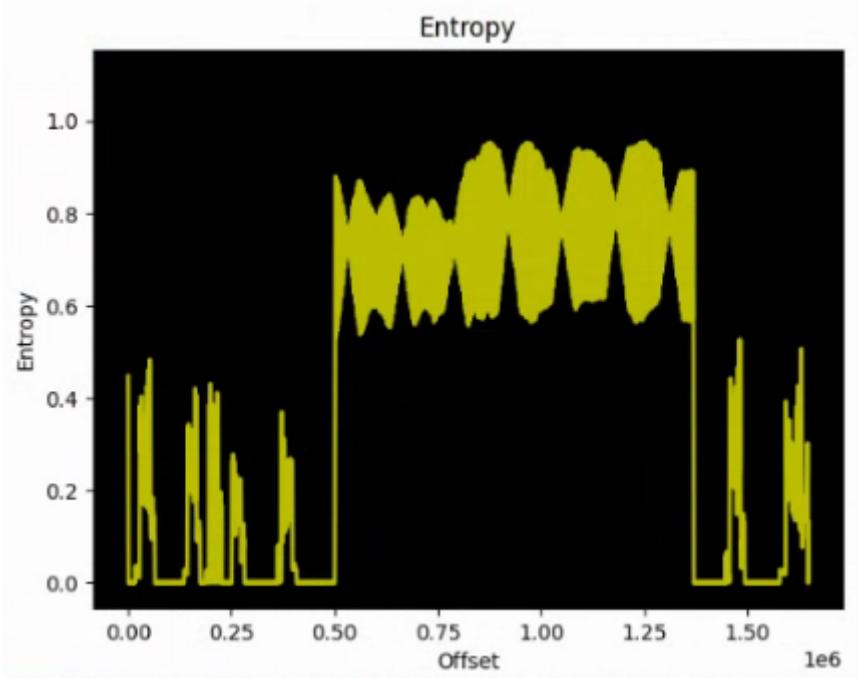
IZQUIERDA → STEGO



DERECHA → ORIGINAL

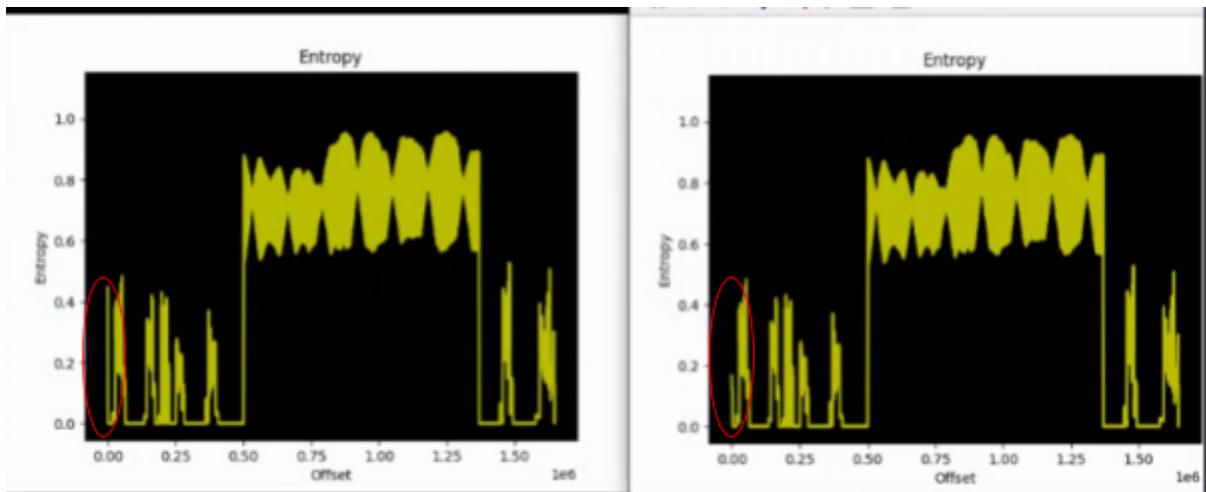


LSB4

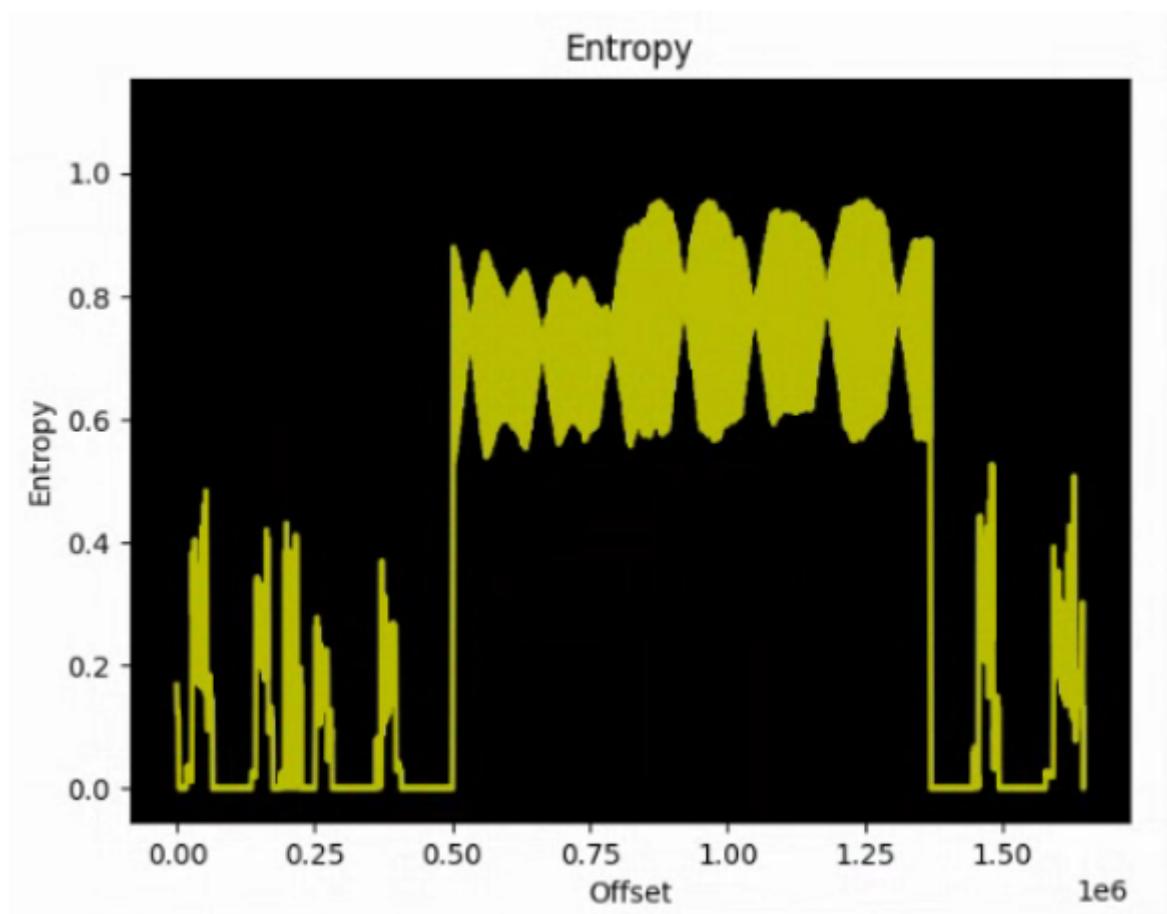


IZQUIERDA → STEGO

DERECHA → ORIGINAL

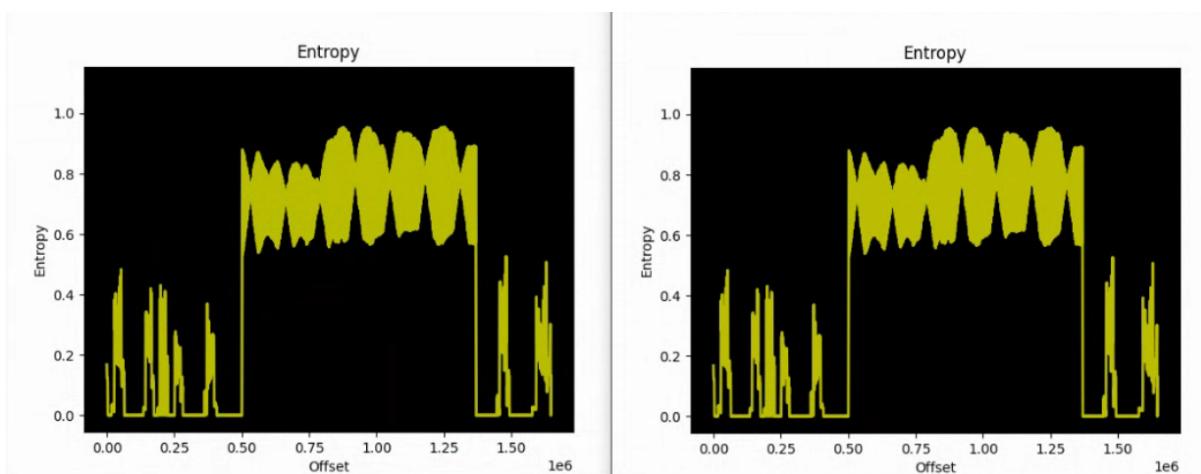


LSBImproved



IZQUIERDA → STEGO

DERECHA → ORIGINAL



NO OBSERVAMOS UNA DIFERENCIA (a grandes rasgos)

Algoritmo	Funcionamiento	Ventajas	Desventajas
LSB1	sustituir el bit menos significativo de cada byte del archivo portador por un bit del mensaje.	<ul style="list-style-type: none"> • El más simple de implementar • Cada pixel de la imagen solo se ve alterado en 1 bit \Rightarrow sólo se pierde 1 bit de información por cada byte 	<ul style="list-style-type: none"> • Los archivos a ocultar deben ser chicos* • De igual manera, el archivo portador debe ser muy grande
LSB4	4 bits del mensaje se ocultan en los 4 bits menos significativos del archivo portador.	<ul style="list-style-type: none"> • Simple de implementar • Se pueden ocultar archivos "grandes"** 	<ul style="list-style-type: none"> • Cada pixel de la imagen se ve alterado en 4 bits \Rightarrow se pierde 50% de información por cada byte
LSB Improved	LINK PAPER	<ul style="list-style-type: none"> • La calidad de la imagen original es casi mantenida • Hace que sea más difícil darse cuenta que se llevó a cabo una esteganografía 	<ul style="list-style-type: none"> • El más complejo de implementar • Se debe guardar el patrón utilizado

* tamaño en bytes del archivo a ocultar = tamaño en bytes de la imagen / 8

** tamaño en bytes del archivo a ocultar = tamaño en bytes de la imagen / 2

3. Explicar detalladamente el procedimiento realizado para descubrir qué se había ocultado en cada archivo y de qué modo. Indicar qué se encontró en cada archivo.

Los pasos que seguimos, fueron los siguientes:

- Ir probando por cada imagen, todos los algoritmos de esteganografía que fuimos implementando y analizando la salida obtenida para hallar la información oculta.
 - En los casos que no funcionaba, es decir obteniendo basura, suponíamos que el algoritmo que se utilizó para esteganografiar ese archivo no era con el que estábamos probando.
- 1) *sherlock.bmp*: Usando LSB Improved, se revela un pdf con el siguiente contenido:

al .png cambiarle la extensión por .zip y descomprimir

- 2) *paris.bmp*: Usando LSB1 se descubrió una imagen .png de buscaminas.



Como se indicó previamente, se convierte este archivo en .zip, y al descomprimirlo se encuentra un .txt con indicaciones sobre cómo interpretar la imagen del buscaminas.

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

Interpretando la imagen del buscaminas obtenemos el algoritmo y modo de encriptación:

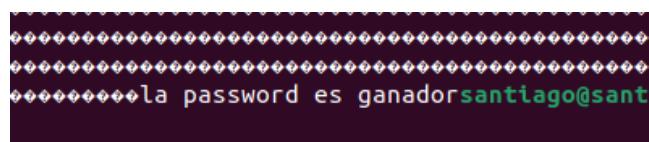
0100 0001 → a
0110 0101 → e
0111 0011 → s
0100 0011 → c
0110 0110 → f
0110 0010 → b

Es decir, AES-CFB

Esto resulta en obtener el algoritmo y modo.

- 3) *kings.bmp*: Luego de intentar con todos los algoritmos de esteganografía y que ninguno obtenga información pensamos que tal vez la información escondida no estaba esteganografiada con LSB.

Intentamos diversos algoritmos/formas y al final solamente había que levantar el archivo bmp en plano y allí se escondía la password a utilizar para el último archivo.



Hasta este momento, obtuvimos:

- Algoritmo de cifrado: AES
- Modo de bloques: CFB
- Longitud de la clave: 192
- Password: ganador

- 4) *medianochе1.bmp*: Usando LSB4 y la información de encriptación obtenida en otros archivos, obtenemos un video del recorte de una película/serie que se habla de información oculta en un archivo/mail.



[LINK AL VIDEO](#)

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

Del archivo *paris.bmp*, luego de desestenografiar con LSB1, se obtiene como resultado una archivo PNG de un buscaminas. Luego, por información obtenida de otro archivo, le cambiamos la extensión a ZIP y descubrimos que también era un TXT comprimido. El último tenía información de cómo interpretar la imagen del buscaminas.

En el archivo *kings.bmp*, no se le encontró información esteganografía con los métodos LSB. Pero, si tenía información oculta en el. Esta, se encontraba escrita en texto plano al final del archivo.

5. Uno de los archivos ocultos era una porción de un video, donde se ve ejemplificado una manera de ocultar información ¿qué se ocultaba según el video y sobre qué portador?

El video da a entender que:

- Se envió un mail con un archivo
- El archivo tiene un tamaño mayor a lo que debería
- Tal vez el “agente” oculto datos en el archivo ([con acento español](#))



Por lo tanto, la única información que podemos sacar del mismo es que:

qué se ocultaba según el video: datos

sobre qué portador: un archivo

6. ¿De qué se trató el método de estenografiado que no era LSB1 ni LSB4 ni LSBI? ¿Es un método eficaz? ¿Por qué?

- De alguna manera, no sabemos bien como, al simplemente cambiarle la extensión al archivo .png obtenido al desesteganografiar paris.bmp usando LSB1, por .zip, se obtiene la información ocultada.
 - Nos pareció bastante eficaz, si...
 - Otra forma que vimos que usaron en los archivos que nos enviaron, en específico en kings.bmp, enviar información oculta al final de la data de la imagen.
 - Nos pareció eficaz ya que cumple con el objetivo de esconder la información dentro del archivo, pero no es una forma segura ya que es muy simple encontrar la información escondida.

Imagen "kings.bmp":



abriendo la imagen con un editor de texto y yendo hasta el final:

yyyyyyyyyyyyyy la password es ganador

7. ¿Por qué la propuesta del documento de Akhtar, Khan y Johri es realmente una mejora respecto de LSB común?

La calidad de una esteganografía se puede determinar por 3 aspectos: imperceptibilidad, capacidad de almacenamiento y robustez. El método propuesto tiene grandes mejoras en la imperceptibilidad, ya que menos bits son modificados. Es robusto, ya que al analizar el archivo no se puede identificar tan fácilmente como LSB1 o LSB4 si es que hubo una esteganografía. Por último, en términos de mejoras de capacidad de almacenamiento no hay ninguna, ya que tiene la misma capacidad que LSB1 y menos que LSB4.

8. ¿De qué otra manera o en qué otro lugar podría guardarse el registro de los patrones invertidos?

En el header de BMP existen 4 bytes reservados para la aplicación que genera la imagen en la posición 0x06 a 0x09. Se podrían utilizar estos bytes para guardar los bits del patrón, y de esta forma tener 4 bytes extra para los datos.

Offset hex	Offset dec	Size	Purpose
00	0	2 bytes	The header field used to identify the BMP and DIB file is 0x42 0x4D in hexadecimal, same as BH in ASCII. The following entries are possible: BM Windows 3.1x, 95, NT, ... etc. BA OS/2 struct bitmap array CI OS/2 struct color icon CP OS/2 const color pointer IC OS/2 struct icon PT OS/2 pointer
02	2	4 bytes	The size of the BMP file in bytes
06	6	2 bytes	Reserved; actual value depends on the application that creates the image, if created manually can be 0
08	8	2 bytes	Reserved; actual value depends on the application that creates the image, if created manually can be 0
0A	10	4 bytes	The offset, i.e. starting address, of the byte where the bitmap image data (pixel array) can be found.

9. Leer el Segundo esquema y analizar (sin implementar) cuáles serían las ventajas que pueden verse.

El segundo esquema planteado en el paper se asimila a LSBI, donde se identifican diferentes patrones de bytes para saber si LSB debe ser invertido o no. En este caso se asume que el receptor del mensaje tiene acceso a la imagen original, por lo que puede saber si cada LSB fue invertido o no. Esto nos permite tener hasta 8 patrones distintos, lo cual implica que tendremos como mínimo la misma probabilidad que el primer esquema de LSBI, pero probabilísticamente la cantidad de bits invertido será menor que todos los anteriores métodos de estenografía.

10. Leer el Segundo esquema e indicar qué desventajas o inconvenientes podría tener su implementación.

Un inconveniente que puede surgir es la necesidad de tener la imagen original de antemano. Esto implica que se tiene que tener una mayor cantidad de información de antemano, lo que puede generar un riesgo de seguridad, ya que esta información tiene que ser transmitida por algún medio.

11. ¿Qué dificultades encontraron en la implementación del algoritmo del paper?

Sentimos que podrían haber hecho un mejor trabajo al momento de explicar su funcionamiento... desde incluir más ejemplos, hasta utilizar algún tipo de dibujo/representación gráfica de cómo y qué bits son los que se modifican, cuales quedan iguales, etc, etc.

Luego de leerlo y releerlo múltiples veces, nos armamos un documento de texto con colores, negritas y subrayados para entender mejor el ejemplo expuesto:

IMAGEN:

10001100 10101101 10101011 10101101

ESCONDER: 1011

10001100 10101101 10101011 10101101

10001101 10101100 10101011 10101101 → 2 LSB cambiados y 2 LSB NO cambiados

10001100 10101101 10101011 10101100

como acompañamiento a:

First Scheme

Four message bits 1011 are to be hidden into four cover image pixels 10001100, 10101101, 10101011 and 10101101.

After plain LSB steganography, stego-image pixels are **10001101, 10101100, 10101011 and 10101101**. Two pixels i.e. first and second of cover image have changed. Now, we see that the second and third LSB of three cover image pixels are 0 and 1 respectively. For two of these three pixels, LSB has changed. If we invert the LSB of these three pixels, cover image pixels will be **10001100, 10101101, 10101011 and 10101100**. Now, there is only one pixel of stego-image which differs from cover image i.e. the last one. Thus, the PSNR would increase improving the quality of stego-image. For correct de-steganography, we need to store the fact that we have inverted the LSBs of those pixels in which second and third LSB are 0 and 1 respectively.

12. ¿Qué mejoras o futuras extensiones harías al programa stegobmp?

- Permitir esteganografía archivos en otro/s tipo de archivo portador
 - no solo en .bmp's
- Hacer un front-end (tal vez con javafx) para poder interactuar con el programa de manera más simple y rápida
- Desacoplar y emprolijar un poco el código
 - Además de hacer su debida documentación