Criptografía y Seguridad (72.44)

TRABAJO PRÁCTICO: ESTEGANOGRAFÍA

Grupo 11

Pedro Remigio, 58451

Miguel Di Luca, 58460

Fermin Gomez, 58111

Cuestiones a analizar

1. **Discutir los siguientes aspectos relativos al documento.**
    - Organización formal del documento.
    - La descripción del algoritmo.
    - La notación utilizada, ¿es clara? ¿hay algún error o contradicción?

El documento trata de un paper de una mejora del algoritmo de esteganografía LSB. Está dividido en tres partes. La primera introduce al tema y explica la idea del algoritmo a mejorar e introduce el algoritmo de criptografía RC4, que es utilizado a lo largo del paper. La segunda parte explica el algoritmo implementado por los autores, con una máquina de estados. La tercera, presenta la implementación y da ejemplos de datos escondidos y encontrados en los archivos.

El algoritmo es una variación del algoritmo LSB (Least Significant Byte). En primer lugar se toma un valor de salto entre 0 y 256 usando potencias de dos, luego se encripta el mensaje usando RC4. Una vez que se tiene al mensaje encriptado, se procede a esconderlo usando el bit menos significativo de cada byte, haciendo uso del salto (hop) previamente obtenido.

En el momento que se llega al final del archivo, se vuelve a la ubicación inicial mas un k (el modulo del numero de vueltas por la foto). Vale aclarar que en el paper agregan una capa inicial de protección en el cual el usuario puede usar el software si sabe la contraseña.

La notación del paper no es clara. La máquina de estados dada en la sección 2 es muy grande y se subdivide en diversas partes, lo que hace al paper difícil de seguir. El hecho de que se aclare cómo se hace la decisión de la clave RC4, hace con que el lector haga suposiciones, como que la clave sea siempre la misma, que va en contra de la seguridad del método de criptografía usado.

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.**

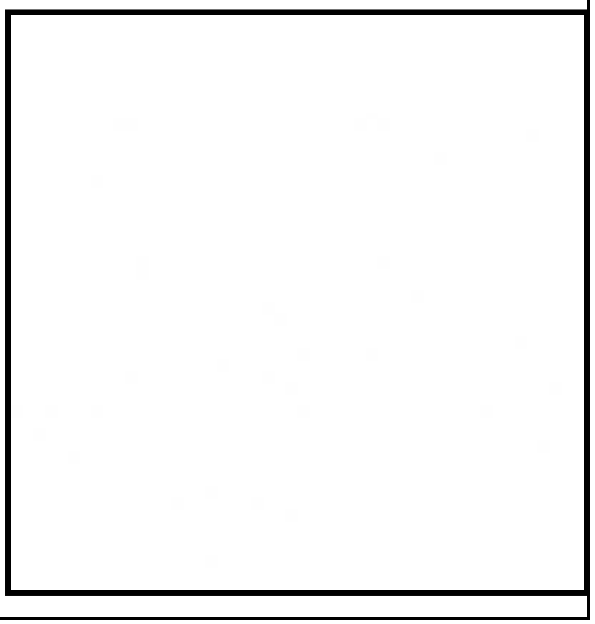
Como archivo portador se eligió cuadrado blanco de 400x400 lo que da a un archivo .bmp de 480,054 bytes. Lo llamamos white.bmp
Para esteganografiar utilizamos 2 archivos de distintos tamaños. Estos los creamos con contenido random utilizando el comando:

```
dd if=/dev/urandom bs=1 count=50000 of=./small-message.txt
```

`dd if=/dev/urandom bs=1 count=220000 of=./big-message.txt`
big-message.txt de 220000 bytes y small-message.txt de 50000 bytes.

A. LSB1

Resultados

Ocultando el archivo big-message.txt	Ocultando el archivo small-message.txt
No se pudo esteganografiar el mensaje	

Ventajas

No agrega ruido visible a la imagen portadora, esto se debe a que el método utiliza el bit menos significativo para ocultar el mensaje, por lo tanto los pixeles se ven modificados mínimamente, o incluso, algunos pueden no verse modificados en absoluto.

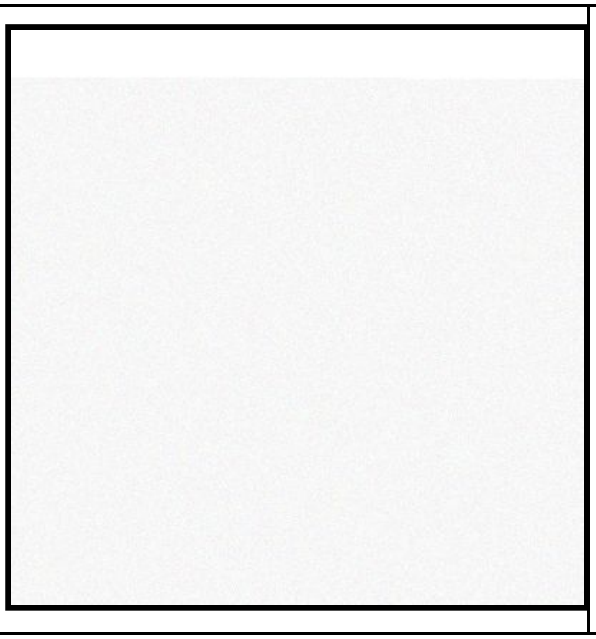
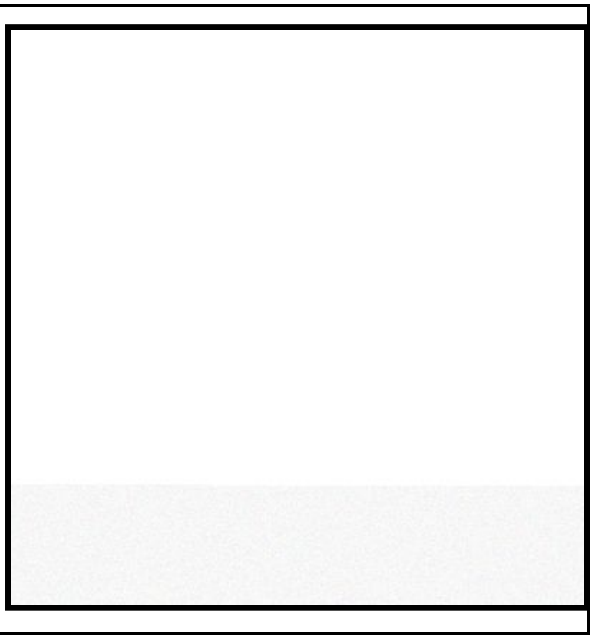
Desventajas

Cómo utiliza el bit menos significativo para ocultar el mensaje el total de bits disponibles para esconder el mensaje es del 12,5% del total de bits correspondientes del archivo portador. Por lo que el tamaño de los mensajes que puede esconder es muy chico.

B. LSB4

Resultados

Ocultando el archivo big-message.txt	Ocultando el archivo small-message.txt

Ventajas

Cómo utiliza los 4 bits menos significativo para ocultar el mensaje el total de bits disponibles para esconder el mensaje es del 50% del total de bits correspondientes del archivo portador. Permitiéndonos esconder mensajes de mayor tamaño que los otros métodos.

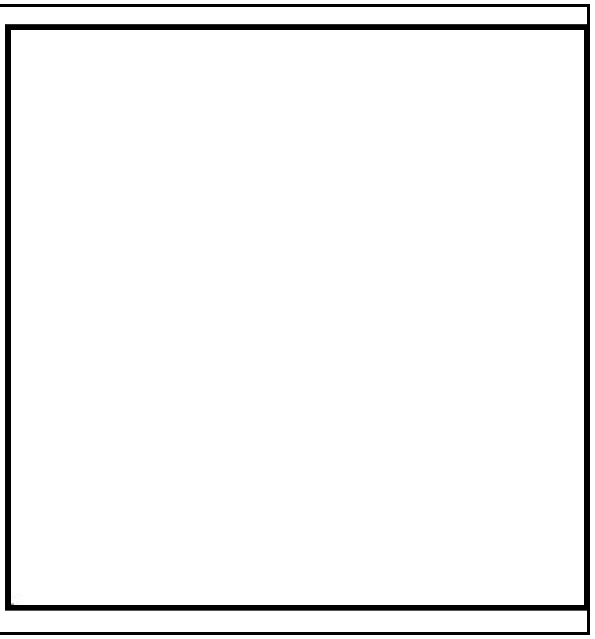
Desventajas

Al utilizar los 4 bits menos significativo para ocultar el mensaje en la imagen se puede puede notar una mayor presencia de ruido, ya que cada pixel se ve modificado de forma más drástica en comparación a LSB1.

C. LSBI

Resultados

Ocultando el archivo big-message.txt	Ocultando el archivo small-message.txt

No se pudo esteganografiar el mensaje	

Ventajas

Los saltos no son tan simples como LSB1, lo que hace con que sea más complicado que un atacante extraiga la imagen. Así como LSB1, el ruido en la foto es mínimo.

Desventajas

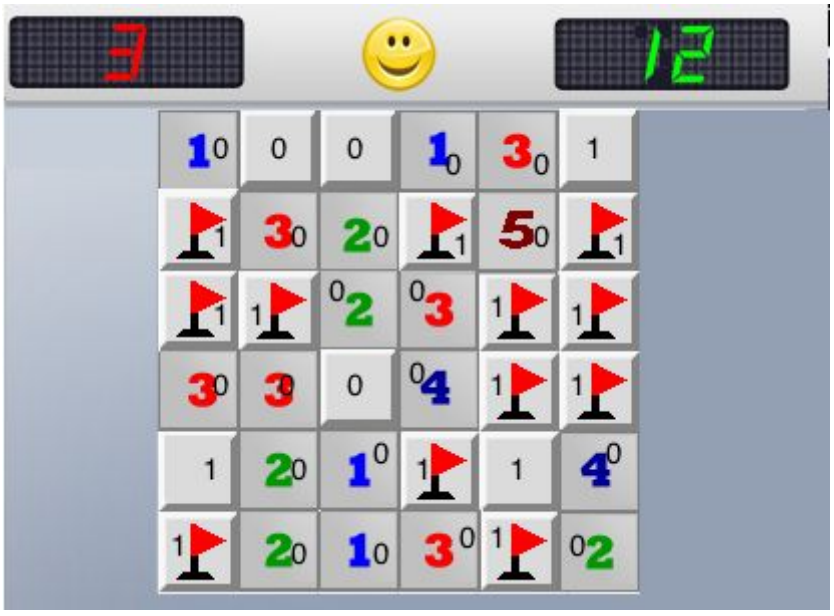
Cómo utiliza el bit menos significativo para ocultar el mensaje el total de bits disponibles para esconder el mensaje es del 12,5% del total de bits correspondientes del archivo portador. Los 6 primeros bytes con la contraseña del RC4 puede modificar dos pixeles totalmente, lo que se puede notar si se hace zoom en es zona de la foto.

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.**

En una primera instancia se comenzó analizando los headers de los archivos con el program Hex Fiend que nos permite visualizar el archivo en formato hexa. En *buenosaires.bmp,* se noto que el tamaño especificado en el header no correspondía con el tamaño actual del archivo.
Se procedió en analizar por qué este archivo era más grande de lo especificado. Con ayuda del editor hexa pudimos ver que es lo que había luego de los bytes originales del archivo. Encontramos un mensaje oculto que decía:
“al .png cambiar extensión por .zip y descomprimir”

Para continuar con el análisis se fue probando extraer mensajes ocultos con los métodos LSB1, LSB4 y LSBI con los demás archivos hasta que con alguna combinación obtuvieramos un mensaje con sentido.

Usando LSB4 en el archivo lado.*bmp,* obtuvimos un archivo con extensión *.png* que contenía la imagen de un tablero de buscaminas.
Habiendo encontrado un ".png" se procedió en seguir la sugerencia obtenida del archivo *buenosaires.bmp*. Se le cambio la extension a ".zip" y luego se lo descomprime. Se descubrió un archivo s*ol13.txt* que cuyo contenido tenía otra sugerencia(ver punto 4).
Siguiendo los pasos sugeridos, terminamos de resolver el juego buscaminas y interpretando el tablero obtuvimos:



```
01000001 -> A
01100101 -> E
01110011 -> S
01000011 -> C
01100110 -> F
01100010 -> B
```

Concluimos que en uno de los archivos el mensaje oculto tiene extension .wmv y está encriptado usando AES en modo CFB.

Usando LSBI en el archivo lima.*bmp* se obtuvo un archivo con extensión ".pdf" que en su contenido decía: "La password es solucion".

Como sabemos que en uno de los archivos el mensaje oculto está cifrado y queda un archivo por analizar, procedemos con la misma metodología pero ahora utilizando como algoritmo de encripción AES256, modo CFB y contrasena solucion.
Finalmente usando LSB1 en *silence.bmp* se obtuvo un archivo con extensión ".wmv"(ver punto 5).

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

Dentro del archivo lado.*bmp* había un mensaje oculto con extensión ".png" que a su vez contenía oculto dentro el archivo s*ol13.txt*. Su contenido decía:

```
cada mina es un 1.
cada fila forma una letra.
Los ascii de las letras empiezan todos en 01.
Asi encontraras el algoritmo que tiene clave de 256 bits y el
modo
La password esta en otro archivo
Con algoritmo, modo y password hay un .wmv encriptado y
oculto.
```

Para ocultar ambos mensajes se aprovechó que los archivos .zip para almacenar su header utilizan los últimos bytes, mientras que los .png los primeros. Efectivamente al abrir con un editor de texto el mensaje oculto se puede visualizar en los primeros bytes de este el magic number `89 50 4E 47 0D 0A 1A 0A`, correspondiente a un archivo .png y en los últimos bytes el magic number `50 4B 03 04`, correspondiente a un archivo .zip.

El mensaje oculto dentro de lado.*bmp* es la concatenación de los binarios de un archivo .png con un archivo .zip (bytes .png || bytes .zip). Este se puede interpretar de dos maneras distintas cambiando la extensión.

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

El archivo portador fue *silence.bmp.* El video muestra una forma de estenografía que a diferencia de ocultar mensajes en archivos .bmp utiliza tejidos como medio portador del mensaje. La metodología consiste en hacer que un hilo vertical saltee la trama por encima de los demás, lo cual sería interpretado como un 1. Caso contrario está por debajo, se lo interpreta como un 0.

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

Este archivo era *buenosaires.bmp.* El método consiste en agregar los bytes de mensaje a ocultar a continuación de los bytes del archivo portador.

El método no es eficaz ya que muy rápidamente se puede detectar que el tamaño especificado y el actual no coinciden haciendo evidente la alteración del archivo portador.

7. **Para la implementación del algoritmo del documento de Juneja y Sandhu, se tomó como clave RC4 los primeros pixeles de la imagen portadora. ¿de qué otra manera podría considerarse o generarse o guardarse la clave RC4?**

Se podría tomar el tamaño del archivo bmp encontrados en el header. Otra manera sería generar una clave aleatoria y pasarla de alguna forma, desde escribirla en bytes al final del archivo (como en el archivo de buenosaires.bmp) como en el propio nombre del archivo (aunque dependiendo del nombre puede resultar bastante obvio).

8. **Según el libro de Katz, hay una forma más segura de usar RC4. ¿se podría implementar en este algoritmo LSBI?**

Katz sugiere ignorar los primeros 1024 bits al menos del flujo del algoritmo. Eso pasa pues el algoritmo tiene un cierto bias en los primeros valores que puede ser inseguro para ciertos ataques. El libro menciona como ejemplo el ataque que es posible al protocolo de encripción WEP usado en redes wireless.

Para que se use en este algoritmo LSBI, se tendria que de alguna manera definir cuántos bits se ignorarán. Además, el “ignorar” en RC4 significa encriptar / pasar información en el flujo para hacer las alteraciones al array propuesto en el algoritmo. Eso significa que las entradas que deberia pasar deberian ser conocidas por las dos partes (tanto el lado que está embebiendo el mensaje como el que está extrayendo). Una posible manera seria setear quizás otros 8 bytes al principio del archivo para definir el número de bits a ignorar y el MSB en los saltos para saber con qué valor pasar por el RC4.

9. **¿Por qué la propuesta del documento de Juneja y Sandhu es realmente una mejora respecto de LSB común?**

Porque esconder la foto con un simple LSB común resulta mucho más fácil de probar. No existen claves especiales o reglas de salto. Entonces con probar el método sobre un archivo, rápidamente encontrar el mensaje escondido. La propuesta no solo mejora la manera de esconder a la información, como pasa la misma por otra capa de encriptación, sin alterar de forma significativa el tiempo.

10. **En el documento, Juneja y Sandhu indican que la inserción de los bits en la imagen es aleatoria. ¿es realmente así? ¿de qué otra manera podría hacerse los “saltos” de inserción de bits?**

No es realmente aleatoria. El paper específica saltos de valores entre 1 y 256 siendo todos potencias de 2. En el caso de lo implementado en el trabajo práctico, fijamos el estado del hop como el primer bit más significativo del primer byte. Si el salto es aleatorio, es necesario que el otro lado que quiera ver al mensaje tenga acceso al orden “aleatorio” de los saltos de bytes. Se podría usar alguna secuencia conocida de saltos usando los valores de los primeros bytes de la portada. Quizás sea posible también hacer saltos que no sean potencias de 2, sino que valores más diferentes.

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

Primeramente, entender el flujo del algoritmo de esteganografía. Son muchos mini pasos que uno tiene que estar al tanto para esconder y extraer bien a los mensajes. Otro tema es en el momento de extraer a la imagen. Para encontrar el 0 final de la extensión, para cada byte leido, hay que desencriptar con el uso de RC4 para ver si se terminó.

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

- Soportar no solo archivos .bmp para ocultar mensajes sino que cualquier archivo.
- También podríamos agregar un algoritmo que funcione de manera similar a LSB, pero de forma dinámica. Es decir, utiliza el bit menos significativo de cada byte hasta que llega al final del mensaje, una vez que sucede esto, vuelve al comienzo y utiliza el segundo bit menos significativo, y así sucesivamente. Con esto nos garantizamos el producir el menor ruido posible y a la vez, soportar mensajes de mayor tamaño.
- Algo que se implementó fue el algoritmo LSBX, siendo X cualquier número entre 1 y 8. Aunque solo se permite al usuario ejecutar LSB1 y LSB4 por los propósitos del trabajo.