Informe TP Especial

Integrantes:

- Greth, Juan Cruz 57370
- Martín, Fernando Ezequiel 57025

Materia: “Criptografía y Seguridad” (72.44)

Tema: “Esteganografía”

Fecha de entrega: 25/6/2020

Índice

Introducción

En el siguiente documento vamos a describir la implementación, ventajas y desventajas de los algoritmos implementados. En el trabajo práctico se realizó la implementación de los algoritmos LSB1, LSB4 y LSBI. Para esto utilizamos el paper y la documentación provista por la cátedra. A la vez, para complementar los algoritmos de esteganografiado se desarrolló algoritmos de encripción para agregarle una capa mas de seguridad a los archivos a ocultar.
Por último ,vamos a presentar los resultados de aplicar estos algoritmos a los archivos provistos por la cátedra para evaluar las implementaciones realizadas en el trabajo.

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_ En cuanto a la organización del documento, nos pareció relativamente correcta, pero hubiéramos puesto la sección de “SECURITY OF PROPOSED SYSTEM” antes de la descripción del algoritmo, debido a que nos parece más beneficioso saber que nos va a ofrecer este algoritmo y luego leer como va a lograr ese cometido en el transcurso de la descripción del mismo.

B_ El algoritmo LSBI utiliza los primeros 6 bytes de la data de la imagen portadora (entendemos por data a la información sin el header) como clave RC4. A la vez en el primer byte se encuentra el salto. Este depende de la posición del primer bit en 1 en el byte. El salto será igual a 2 elevado a la posición de este primer bit en 1. Por ejemplo, si el byte es 10010010 entonces el salto será igual a 2^7, dado que el primer bit en 1 es en la posición 7. En caso de que el primer byte sea igual a 0 el salto será 256. Una vez obtenido el salto, se encripta con RC4 la información con la clave obtenida de los primeros 6 bytes y se procede a esteganografía el archivo. El primer byte a utilizar es el 7, en donde se inserta el en el bit menos significativo el primer bit del primer byte de la data encriptada. El siguiente byte a utilizar es 7 más el salto. En este se insertará el segundo bit de la data en el bit menos significativo. El proceso continua hasta terminar de guardar toda la información. Es muy probable que, dado que el salto de bytes es grande, se llegue al final del archivo. En este caso el salto da la vuelta y vuelve a empezar por los primeros bytes de la imagen portadora. Por ejemplo, si la imagen tiene 200 bytes, estás en la posición 190 y el salto es 16, el siguiente el byte será el 6.

C_ Con respecto a la notación, nos pareció adecuada pero posee un par de ambigüedades al momentos de nombrar propiedades del algoritmo.

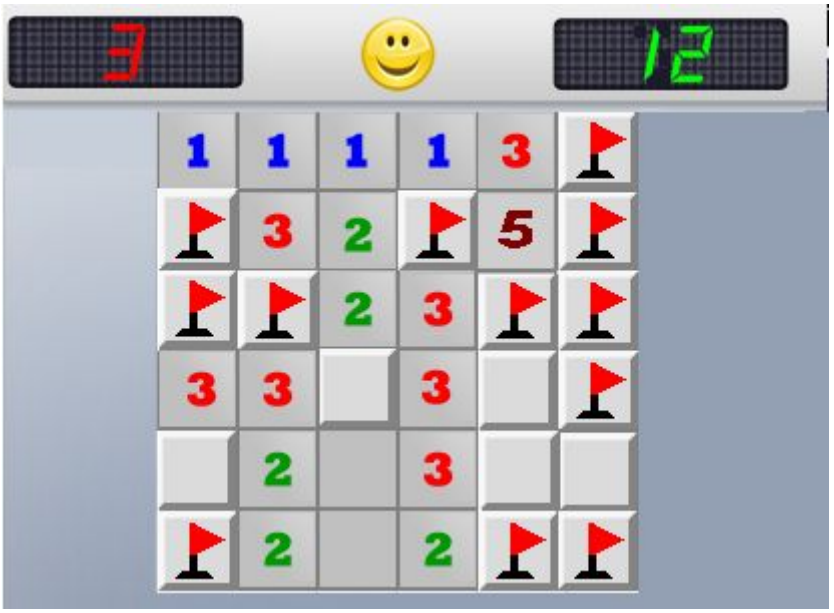
2. Esteganografíar un mismo archivo en un .bmp con cada uno de los tres algoritmos, y comparar los resultados obtenidos. Hacer un cuadro comparativo de los tres algoritmos estableciendo ventajas y desventajas.

	LSB1	**LSB4**	**LSBI**
Ventajas	La gran ventaja de este algoritmo es que es casi imperceptible el cambio de color de los pixeles al ocultar un archivo en una imagen .bmp.	La ventaja de utilizar este algoritmo es que al utilizar 4 bits por byte para esconder información, permite esconder archivos de gran tamaño.	La ventaja de usar este algoritmo es que en el caso de que alguien quiera extraer el archivo oculto en la imagen, se le va a hacer realmente difícil, ya que la distribución de los bytes que tienen información del archivo, están definidos bajo un patrón que no es trivial, además que utiliza un algoritmo de encriptación RC4 con una clave arbitraria evitando interpretar la información de manera sencilla. Además, cuenta con las ventajas de LSB1 debido que utiliza la misma cantidad de bits por byte para esconder un archivo.
Desventajas	La desventaja de este algoritmo es	La desventaja es que se hace muy	Una de las desventajas de

	que al nada más utilizar el último bit de cada byte, para poder esconder 1 byte de archivo, necesito 8 bytes de imagen portadora, lo que me limita bastante el tamaño de archivo que puedo esconder.	notorio el cambio de tonalidad de los píxeles, debido a que al cambiar 4 bits, estamos influyendo de gran manera en el tono del color representado.	este algoritmo es la misma que en el caso de LSB1, al esconder 1 bit por byte, el archivo a esconder esta muy limitado. Y la segunda desventaja que le encontramos es el tiempo de procesamiento, debido que tiene que hacer muchos pasos para poder esconder como extraer un archivo.

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

De entrada contábamos con cuatro archivos provistos por la cátedra. Sabíamos que los archivos ocultaban algún tipo de información, pero a priori no teníamos información de que se escondía y como. Por lo que en un primer momento recurrimos a intentar desesteganografiar cada archivos sin password con los tres modos desarrollados. De eso logramos obtener dos archivos. En madridoso, desesteganografiando con LSB4, obtuvimos una imagen png con un buscaminas.



Mientras que en quito con LSBI obtuvimos un pdf que nos revelaba que la password era compartido.

Hasta ese momento teníamos una password pero no sabíamos con qué modo y schema intentar desesteganografiar. Por lo que seguimos buscando información en los archivos. Al leer el hexdump del archivos titanic1.bmp encontramos al final lo siguiente:

```
00240030: EB CF AD E4 C6 A3 61 6C 20 2E 70 6E 67 20 63 61    kO-dF#al..png.ca
00240040: 6D 62 69 61 72 20 65 78 74 65 6E 73 69 6F 6E 20    mbiar.extension.
00240050: 70 6F 72 20 2E 7A 69 70 20 79 20 64 65 73 63 6F    por..zip.y.desco
00240060: 6D 70 72 69 6D 69 72                               mprimir
```

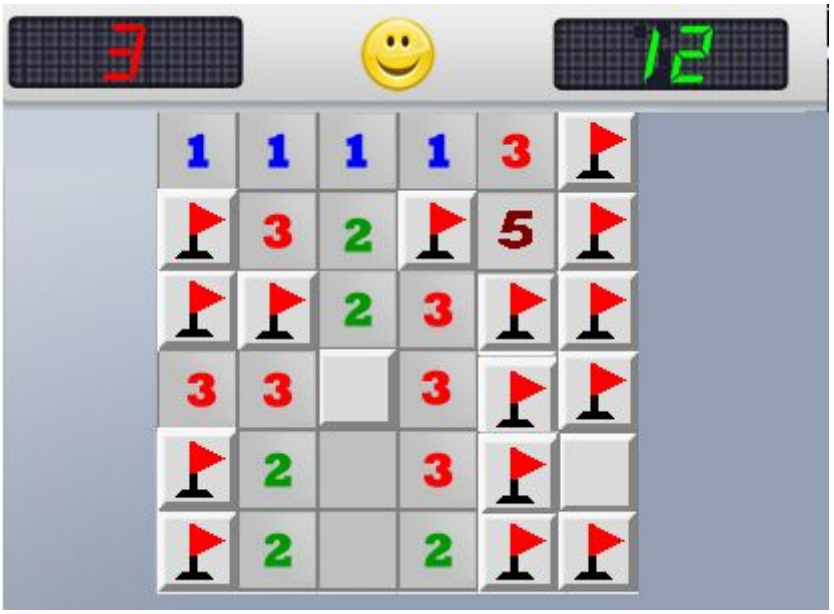
Por lo tanto procedimos a, como decía el mensaje oculto, cambiar el png obtenido de madridoso a un .zip y descomprimir. En este encontramos un txt con el siguiente mensaje:

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.

Este mensaje nos revelaba que existia un archivo .wmv encriptado y oculto en alguno de las imágenes provistas por la cátedra. Este archivo estaba encriptado con una clave de 256 bits. Sabiendo ya la clave, nos faltaba encontrar el modo, el schema y el algoritmo.

Leyendo el mensaje, este nos dice que cada fila del buscaminas formaba una letra en binario. Esta comenzaba con 01 y luego continuaba con cada posición de la imagen según si en esa posición se encontraba una mina o no. Si se encontraba una mina completaba con un 1, caso contrario un 0. Una vez recorrida toda la fila obteniamos 8 bits que podíamos convertir en una letra. Al completar las 6 filas nos arrojo un mensaje que no tenia ningun sentido. Luego de re-analizar el mensaje y la foto nos dimos cuenta que el buscaminas no estaba completo, por lo que marcamos las minas faltantes en el el juego para poder obtener el mensaje.



Con el buscaminas completo realizamos el mismo procedimiento anterior y esta vez pudimos obtener un mensaje.

fila	binario	letra
1	01000001	a
2	01100101	e
3	01110011	s
4	01000011	c
5	01100010	b
6	01100011	c

Leyendo las letras que obtuvimos de descifrar el buscaminas, supimos que el modo de encriptación era CBC, y el schema era AES.

Por lo tanto teníamos la password, el modo y el schema de encripción, pero seguíamos sin saber que algoritmo utilizar y en qué archivo. Al no encontrar más información, intentamos desesteganografiar los archivos con fuerza bruta. Desesteganografiando lima.bmp con LSB1 y demás datos encontrados obtuvimos el video que buscábamos. Este se puede observar en el siguiente link. Se trata de una escena de la película ***Se Busca*** en la que el personaje que interpreta Morgan Freeman explica una técnica de encriptación antigua.

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 que contenía el video era lima.bmp.

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

Encontramos dos métodos de esteganografiado no relacionados con los algoritmos implementados. El primero fue el de cambiar la extensión del png a zip y encontrar un un txt oculto en el mismo. Para realizar esto se crea una imagen que resulta de la copia de una imagen portadora más el zip a esconder. Dado que primero se encuentra la imagen, al intentar abrir el png, el SO lo primero que encuentra es el header de la imagen. En este obtiene el tamaño de la misma. Luego abre la imagen leyendo solo hasta el final del png. Mientras que si lo intentas descomprimir, en el formato zip el header se encuentra al final de la data, al igual que en nuestro archivo. Por lo tanto es capaz también de interpretarlo como un zip y descomprimir el archivo. Esta técnica es muy simple y se puede realizar sin ningún tipo de programa, pero no es un método eficaz. Con un simple análisis, observando la diferencia entre el tamaño de la imagen en el header y el tamaño del archivo podemos intuir que al final se esconde información y separarlos. Así encontraríamos el zip y, a menos que este esté protegido por una password, podríamos descomprimirlo y obtener la información oculta.

El segundo método que encontramos fue al leer el código binario de la imagen titanic1.bmp. En este se ocultaba una frase al final del archivo. El procedimiento para esconder la información es similar al 1. La diferencia es que para encontrarlo la única alternativa es leer el final de cada archivo. Al igual que en el caso anterior el método es simple, pero no eficaz. Comparando de nuevo el tamaño de la imagen en el header con el del archivo, podemos intuir que se esconde algo. La diferencia es que en este caso es un texto plano al final del archivo.

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 obtener de muchas formas, por ejemplo en lugar de ser los primeros 6 bytes, podrían ser los últimos 6 bytes, otra forma podría ser invertir los bytes de posición, es decir, en lugar de usar los byte ordenados del 1 al 6, podríamos usarlos del 6 al 1, o con cualquier otra distribución estipulada. También podríamos usar más bytes (mientras cumpla con la longitud de la clave de RC4).

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

Según Katz, para que RC4 se usa de manera segura si los 1024 bits iniciales de la salida se descartan. En nuestro caso, sería imposible implementar esto con lo que hicimos hoy, debido que lo que mandamos a encriptar es toda la información del archivo, si al momento de la salida, omitimos los primeros 128 bytes, perdemos una gran parte de la información que queres ocultar en la imagen.
Si quisiéramos adaptar nuestro sistema para que funcione con esta decisión, deberíamos anteponer 128 bytes de basura a nuestro byte array con la información del archivo a ocultar, de manera tal que al momento de la salida, poder suprimirlos sin perder información relevante y al momento de desencriptar lo recolectado a la imagen, habría que anteponer 128 bytes basura a todo lo extraído de la imagen para poder obtener la información correcta. (Habría que hacer la implementación correspondiente para poder desencriptar la los primeros 4 bytes obtenidos para saber la longitud del archivo).

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

La propuesta hecha en el documento es mucho más segura que el lsb común por dos razones. La primera de estas es que los bytes en los que se va a ocultar información, no están contiguos en la imagen, esto quiere decir que al momento de buscar en qué bytes hay data escondida, se les va a hacer muy difícil a los atacantes si no conocen el salto.
Y la segunda de las razones es que, como si fuera poco tener que encontrar el salto, la información que esta guardada en los bytes esta encriptada con un algoritmo RC4, esto quiere decir que si el atacante no conoce la ubicación de la clave, no podrá desencriptar la información que vaya consiguiendo para verificar si esta tiene sentido o no.

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?

Esta afirmación no es 100% cierta, ya que la inserción de los bits, depende de la imagen portadora, es decir que en una imagen portadora, se modificarán siempre los mismos bits no importa el archivo que se quiera ocultar.
Esto ocurre debido a que el salto lo obtiene a partir del primer byte de la imagen.
Otra manera de realizar el salto de inserción es en lugar de saltar de a bytes, hacerlo de a pixeles, cambiando el bit menos significativo del pixel correspondiente.

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

La principal dificultad que tuvimos fue entender el algoritmo LSBI. La parte “matemática” en que byte ocultar una vez que terminaba de recorrer todo la imagen portadora y había que realizar un salto en módulo no era trivial. Una vez que entendimos como realizar el salto, pudimos concretar la extracción de la imagen de prueba y completar el algoritmo.

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

Posiblemente como mejoras futuras, intentaremos que el programa soporte otro tipo de imágenes, y por qué no, otro tipo de datos (como videos por ejemplo).
Otra cosa que nos parecería interesante agregar son otros tipos de esteganografiados.