Criptografía y Seguridad (72.44)

Trabajo Práctico: Esteganografía

Cuestiones a analizar

25 de junio del 2020

Marina Fuster 57613

Clara Guzzetti 57100

Ignacio Vidaurreta 57250

Índice

1. Discutir los siguientes aspectos relativos al documento.

En referencia al archivo "*An improved LSB based Steganography with enhanced Security and Embedding/Extraction*", procedemos a contestar la siguientes preguntas:

a. Organización formal del documento.

El artículo posee un abstract y cinco secciones: la introducción (donde introduce brevemente el concepto de esteganografía, LSB y RC4), el diseño del sistema propuesto (donde presenta con detalle y gráficos los procesos de embeber y extraer), en tercer lugar se encuentra la implementación del sistema propuesto, cuarto, describe en dónde radica la seguridad de esta implementación y, por último, realiza conclusiones y sugerencias para trabajo futuro.

b. La descripción del algoritmo.

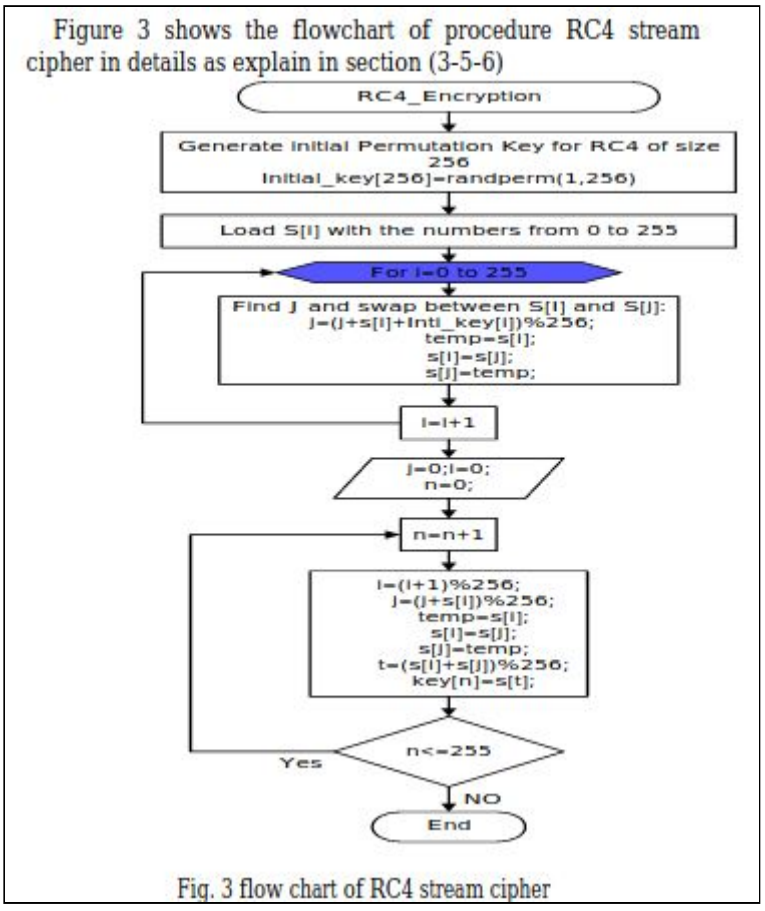
En el paper el sistema propuesto recibe como input el objeto a embeber y un *cover object* que es una imágen de grey-scale de tamaño 256x256 y envía como output un objeto esteganografiado.
Define 2 caminos: embeber el texto secreto en la imágen y extraer el texto de la imagen portadora.

i. **Proceso de ocultamiento**: Como se describe en el paper, el algoritmo consta de los siguientes pasos:
    1. Pasarle como input el texto secreto para ocultar en la llamada *cover image*
    2. Seleccionar la *cover image* (archivo BMP) de tamaño 256x256
    3. Abrir la imagen y poner la información en una matriz
    4. Seleccionar el valor de *hop* como una potencia de 2 dentro de la lista [2, 4, 8, 12, 16, 25, 32].
    5. Calcular el tamaño del texto secreto
    6. Aplicar RC4 en el texto secreto
    7. Sustituir el secreto encriptado del texto en la ubicación especificada (valor del hop) de la *cover image*.

ii. **Proceso de extracción**: permite recuperar el texto secreto con los siguientes pasos:
    1. Recibe la clave de extracción y el objeto esteganografiado.
    2. Se saca el texto encriptado del objeto esteganografiado haciendo el proceso inverso al paso 7 de ocultamiento
    3. Utilizando la clave se desencripta el texto utilizando RC4
    4. Fin

c. La notación utilizada, ¿es clara? ¿hay algún error o contradicción?

Creemos que existen ciertas mejoras que se le podrían hacer al paper. En primer lugar, estaría bueno que los algoritmos se intentaran explicar mediante pseudocódigo, en lugar de mediante diagramas de flujo. Esto quizá es un poco bias de nuestra parte ya que durante la carrera, solemos entender algoritmos a través de pseudocódigo. Para nosotros, la descripción de los algoritmos resultaba muy confusa en algunos casos. Por ejemplo para el caso de RC4:



Los diagramas de flujo no deberían tener tanto texto, sino mostrar el algoritmo de una manera más limpia, que luego se puede especificar utilizando pseudocódigo.
Otra cosa que no logramos entender es por qué le dicen *hope* al *hop*. Nos pareció un uso inadecuado de notación.

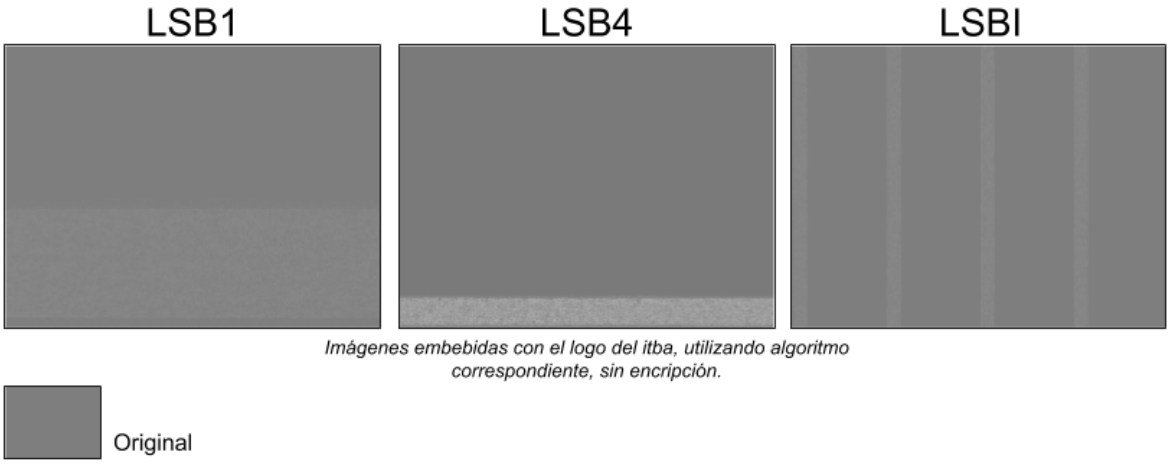
Finalmente, el uso de screenshots para la muestra de la implementación resultó contraproducente ya que no se podía entender bien qué era lo que estaba escrito. Quizá resultaría más claro transcribir el output a texto.

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.

Para abordar esta comparación, queríamos, en primer lugar, ver qué sucedía con las imágenes embebidas.

Imágenes proporcionadas por la cátedra con logo de ITBA embebido

En este caso, vemos que prácticamente no hay diferencia para el ojo humano entre estas tres imágenes excepto por una diferencia en la imagen correspondiente a LSB4 en el inferior. Creemos que la falta de percepción a simple vista se relaciona con el hecho de que dichas imágenes tienen bastantes matices, lo cual permite esconder mejor el logo del itba. Para comprobar esto, decidimos embeber el logo en imágenes lisas completamente rojas. En un paso posterior editamos aumentando el contraste y reduciendo la saturación de las imágenes finales para observar las diferencias con el ojo humano:



Imágenes embebidas con el logo del itba, utilizando algoritmo correspondiente, sin encripción.

En este caso vemos que es totalmente evidente al ojo humano que la imagen original fue modificada.

	LSB1	LSB4	LSBI
Observación	Podemos notar cómo los cambios se acumulan en el final de la imagen pues a pesar de que la información se embebe al comienzo de la matriz que la representa, la	Los cambios se acumulan también al final de la imagen, pero en este caso de una manera más notoria dado que se modifican los últimos cuatro bytes en	Podemos identificar un “paso” entre las zonas con diferencias, que reflejan el HOP del algoritmo LSBI. Esto produce que la diferencia se disperse en diferentes zonas

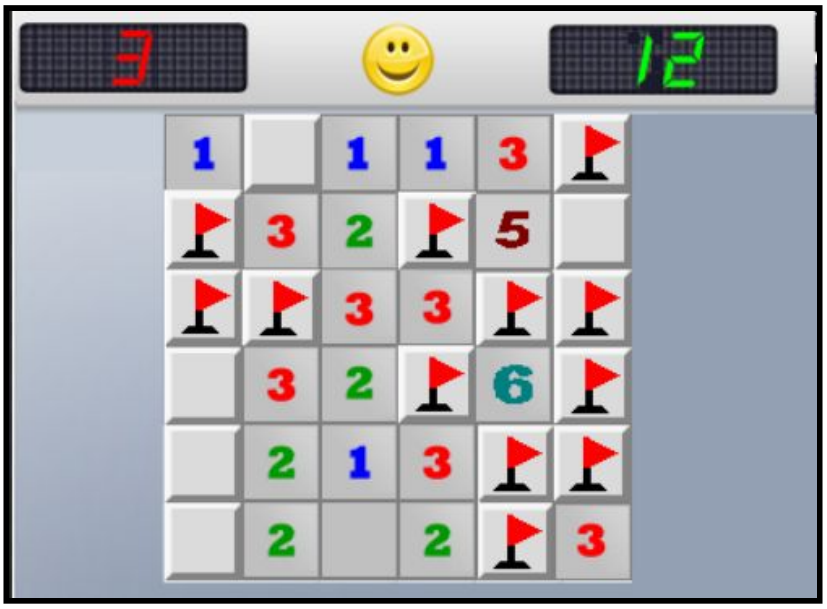
	imagen en sí se renderiza de abajo hacia arriba, de izquierda a derecha. Se observa una distribución uniforme en la zona diferente ya que el algoritmo es secuencial en cuanto a los bytes y embebe un solo bit en cada uno.	lugar de uno sólo. Por esa razón pasa menos desapercibido. Así también la diferencia se encuentra en una porción más acotada de la imagen ya que la información entra en menos bytes, mostrando una compresión respecto de la superficie diferente en LSB1	de la imagen en lugar de acumularse al fondo de la misma. La intensidad de la diferencia se asemeja a la de LSB1 ya que como aquel algoritmo, sólo modifica un bit por cada byte en el que inyecta información.
Ventajas	★ Baja intensidad en la zona modificada	★ Zona modificada de menor tamaño	★ Baja intensidad en la zona modificada ★ Dispersión de las zonas modificadas gracias al HOP
Desventajas	➢ Acumulación de la zona modificada al final de la imagen	➢ Acumulación de la zona modificada al final de la imagen	➢ Riesgo de encontrar un patrón de zonas diferentes

Más allá de las ventajas y desventajas cabe notar el rol de la imagen portadora a la hora de elegir un algoritmo de estaganografiado. El ojo humano podría detectar diferentes cambios dependiendo de cómo le afecta a la misma la información a embeber y el algoritmo elegido.

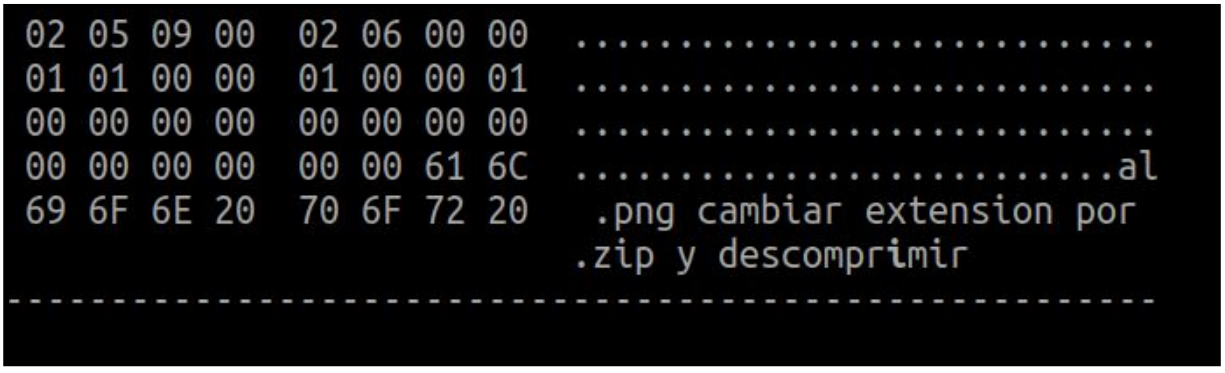
3. Explicar detalladamente el procedimiento realizado para descubrir qué se había ocultado en cada archivo y de qué modo. Indicar qué se encontró en cada archivo.

Luego de extraer correctamente el logo del ITBA de los archivos provistos por la cátedra a fines de testear los algoritmos de estaganografiado, comenzamos a hacer prueba y error con los archivos provistos definitivos.

En primer lugar, sospechamos que, al no tener información sobre las imágenes, alguna de las mismas debía contener un archivo sin encripción. Probamos con django.bmp y con kings.bmp. En ***kings.bmp*** (con LSB1) encontramos la siguiente imagen:



Luego de obtener dicha imagen, procedimos a probar LSB1, LSB4 y LSBI con el resto de los archivos, suponiendo que necesitábamos más datos para continuar. El único archivo que nos devolvió algo fue *lima.bmp* y esto fue descrito más abajo. Como la información no era suficiente para continuar y, como en el enunciado daba la pista de que había otras formas de estegoanálisis, probamos revisar los archivos con hexedit. En el archivo ***django.bmp***, hallamos lo siguiente:



Una vez obtenida dicha información, cambiamos la extensión de nuestro archivo *challenge.png* a *challenge.zip* y extraemos. Dentro, encontramos el archivo ***sol4.txt***.

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 192 bits y el modo
La password esta en otro archivo
Con algoritmo, modo y password hay un .wmv encriptado y oculto.

Contenido del archivo sol4.txt

De dichas instrucciones obtenemos que el algoritmo es “aes192” y el modo “ecb”.

Por otra parte, del archivo ***lima.bmp***, extrajimos con LSBI y obtuvimos la password necesaria.

Con todas estas herramientas, probamos LSB1, LSB4 y LSBI en el archivo ***loimposible.bmp*** con algoritmo aes192, modo ECB y password “metadata”.

Probando llegamos a la respuesta correcta con LSB4 corriendo el siguiente comando:

./stegobmp -extract -out “final_answer” -p “challenge/loimposible.bmp” -steg LSB4 -a aes192 -m ECB -pass metadata

Esto nos permitió encontrar un archivo de video wmv, con una escena de una película (nos quedamos con ganas de saber el nombre y el objetivo igual).

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

El archivo oculto que tenía otro mensaje oculto fue la imagen del buscaminas. Se había ocultado un archivo pdf cambiando la extensión. Es interesante esta práctica porque, por un lado, se engaña a la persona creyendo que la imagen png es lo que se quería encontrar. Por otro lado, si uno conoce esta práctica, las extensiones comunes a probar no son demasiadas, y se podría encontrar información oculta haciendo fuerza bruta sobre las mismas.

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 portador del archivo de video fue la imagen de *loimposible.bmp*.

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

El método de esteganografiado consiste en agregar directamente los bytes correspondientes al mensaje al final de la imagen. Abriendo el archivo en hexedit podemos ver a la derecha el mensaje oculto correspondiente en texto plano.

No consideramos que el mismo sea un método eficaz ya que inyecta bytes de manera directa en lugar de modificar ciertos bits de los bytes originales. Al mismo tiempo la información puede ser accedida por la simple observación del hex dump, que provee muchas veces la columna de la derecha con la traducción ASCII correspondiente, como es el caso de lo que descubrimos en *django.bmp*. Esto deja al mensaje muy expuesto.

Considerando que la criptografía y la estenografía se complementan, creemos que éste método es el menos eficaz en comparación con el resto de los métodos abarcados por este trabajo práctico.

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

Una idea que quizá haría un poco más complejo al algoritmo pero que creemos que lo haría más seguro sería concatenar la clave al stream de información a embeber(sin aplicarle RC4, es decir lo estaríamos pasando en plaintext).

La estructura del stream quedaría de la siguiente manera:

SIZE_OF_KEY | KEY | STREAM

Otra ventaja que nos da esto es que podemos hacer que la clave sea de tamaño variable, ya no estamos obligados a que sean 6 bytes (esto implicaría también que se modifique el código actual de RC4 para que tome claves de tamaño variable, pero esto debería ser trivial).

Sin embargo, tiene sus desventajas, entre las que encontramos

- ➢ Requeriría agregar una nueva lógica de concatenación de la key al stream que agrega complejidad
- ➢ Aumenta el tamaño del stream a embeber
- ➢ La seguridad de este método depende de qué tan seguro hagamos la transmisión del hop (que en el sistema actual, no es para nada seguro porque está en el primer byte, por lo que la seguridad de este mecanismo si bien es mayor a la forma en la que lo hacíamos antes, sigue sin ser muy grande)

Otras alternativas:

- Podría ser que los bytes que se agarran para la clave no estén todos juntos porque puede ser más sencillo para un atacante ir probando con los primeros n bytes. De todas maneras presenta el problema de cómo detectar cuáles son a la hora de realizar el embed y no pisar esa información

- Podría generarse la clave a partir del header del archivo portador (tiene el beneficio añadido de que esta sección no se modifica, con lo cual no nos tenemos que preocupar por la posibilidad de modificarlo accidentalmente nosotros)
- Podría generarse a partir del tamaño del archivo portador
- Podría generarse a partir de una private o public key dependiendo la intención detrás del ocultamiento de información

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

En su libro, Katz indica que se suele preferir o impulsar el uso de cifrados en bloque frente a cifrados de flujo. Los primeros suelen ser eficientes para los ambientes usuales y parecieran ser más robustos que los de flujo.

Con respecto al cifrado RC4, el cual es el que aquí nos compete, Katz indica que los primeros bytes del flujo de salida generado por RC4 suelen contener un "bias". Nos recomienda que, en caso de utilizar RC4, los primeros 1024 bits (128 bytes) del flujo de salida deberían ser descartados.

Esta medida no se podría implementar directamente en nuestro algoritmo LSBI. Esto sucede porque en los primeros bytes del stream tenemos la longitud del mismo y parte de la información. Si nos tomáramos el trabajo de completar con basura antes de cifrar, y luego pasar por RC4, entonces sí, podríamos implementar esta medida.

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

La mejora principal que obtenemos del paper es la inserción de bits mediante a saltos. El hecho de poder realizar saltos de longitud variable hace que sea mucho más difícil la detección de que hay información oculta dentro del archivo.

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?

Claramente no es aleatoria porque los saltos tienen un patrón dado por el hop. Sin embargo, para alguien que no conoce el algoritmo entendemos que puede ser difícil encontrar o reconocer que existe un patrón (sobre todo si el hop es muy grande y además hace unas pasadas sobre el archivo).

Una forma en la cual podrían hacerse los saltos incrementando la dificultad de encontrar el patrón sería utilizar un hop variable. Se sigue manteniendo el problema del valor secreto inicial, como en la implementación de Juneja y Sandhu, pero ahora en cada inserción la multiplicamos por un cierto valor o buscamos alguna forma inteligente de alterarlo. Esto permitiría que en cada inserción el hop cambie hasta volver a empezar, momento en el cual se restartearía. Otra alternativa de saltos variables o indexados podría ser por ejemplo en base a paridad o a números primos.

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

- Encontrar la extensión (ya no terminaba con \0 por estar encriptado)

- Combinarlo con RC4
- Nos resultó difícil darnos cuenta qué cosas eran lo suficientemente similares para colocarlas bajo una misma implementación y cuándo debíamos hacerlo por separado. Consideramos grandes blockers aquellos problemas en donde pensábamos en bytes cuando debíamos pensar en bits, ya que nos costó muchísimo encontrar nuestro error cuando nos confundimos.
- El álgebra modular que utilizamos para movernos en la matriz a veces resultaba un poco más complicado.

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

- ➔ Mayor cobertura de testing.
- ➔ Mejor aprovechamiento del sistema de logging para informar de los eventos destacados al usuario.
- ➔ Mejor reutilización de código para los algoritmos de estenografiado.
- ➔ Podría añadirse un programa que automatice el estegoanálisis dadas ciertas imágenes portadoras que determine las probabilidades de haber combinado ciertos algoritmos y métodos de encripción dependiendo del éxito o fracaso de la ejecución.
- ➔ Sería interesante añadir un programa que devuelva la probabilidad de que una imagen contenga un archivo oculto, de manera que, en caso de que el análisis del programa hipotético mencionado arriba falle, uno pueda decidir si dicha probabilidad es suficiente para comenzar un estegoanálisis manual. Ambos programas se complementarían.
- ➔ Mejor manejo de la búsqueda del tamaño de la extensión en LSBI.
- ➔ Mejoras en cuanto al manejo de recursos de memoria. Creemos que hay demasiado traspaso de la información de un lado a otro, lo cual provoca alto uso de mallocs que se hace difícil controlar.
- ➔ Intentar paralelizar algunas tareas. Quizá se podría hacer más eficiente mediante a la utilización de threads (o quizás no, no investigamos mucho sobre posibilidades de paralelización)