Criptografía y seguridad (ITBA)

Trabajo Práctico: Esteganografía

Autores

Juan Ignacio Quintairos - Legajo 59715

Valentino Riera Torraca - Legajo 60212

Camila Borinsky - Legajo 60083

Introducción

Este trabajo práctico consistió en la implementación de 3 métodos de esteganografía. Como el objetivo del esteganografiado es ocultar un mensaje en un portador, la criptografía puede complementar estos métodos permitiendo ocultar un mensaje cifrado en un portador, resultando en que quede aún más oculto.

Implementamos los algoritmos para ocultar un contenido o *payload* en un archivo portador del tipo `.bmp` y además el algoritmo para extraer el contenido (el proceso inverso). Los métodos implementados fueron:

- LSB1: modifica el bit menos significativo de cada byte del portador reemplazándolo por un bit del contenido a ocultar.
- LSB4: modifica los 4 bits menos significativos de cada byte del portador reemplazándolo por 4 bits del contenido a ocultar.
- LSBI: versión mejorada del LSB1 presentada en el paper provisto por la cátedra que intenta maximizar la cantidad de bits del portador que se mantienen igual almacenando un patrón.

Además trabajamos con la librería de `openssl` para c permitiendo ocultar contenido encriptado por diferentes algoritmos y métodos de encripción.

Algoritmos:

- AES 128
- AES 192
- AES 256
- DES

Modos

- ECB
- OFB
- CFB
- CBC

En una segunda parte del trabajo práctico analizamos archivos provistos por la cátedra para intentar encontrar la información oculta que llevaban.

Implementación

A modo de facilitar el entendimiento de nuestro código y la estructura del proyecto agregamos esta sección con una breve descripción de la implementación.

- El archivo stegobmp.c es el entry point de nuestro programa en el que parseamos los parámetros y nos construimos estructuras para facilitar los flujos del sistema.
- En el directorio headers están presentes los .h
- En el directorio resources están presentes algunos archivos de prueba.
- El archivo reveal.c resuelve los pedidos de extracción de payload. Se cuenta con una primera función con cómputos generales que luego llama a otra dependiendo si se trata de lsb1 o lsb4, o si se trata de lsbi. Si se debe desencriptar se utilizan funcionalidades de cryptography.c
- El archivo hide.c resuelve los pedidos de ocultamiento de payload. Se cuenta con una primera función con cómputos generales que luego llama a otra dependiendo del algoritmo solicitado. Si se debe encriptar se utilizan funcionalidades de cryptography.c
- El archivo cryptography.c contiene funciones que utilizan la librería openssl para resolver los pedidos de encripción y desencripción.
- El archivo cmp_bits.c lo usamos para realizar pruebas de comparación de archivos byte a byte.

En el caso de ocultar/revelar un contenido encriptado la convención que seguimos fue:

`LSB(`tamaño del cifrado, `encrypt(`tamaño del `payload` || `payload` || extensión del `payload))`

Cuestiones a analizar

1. Discutir los 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?

   La organización formal del documento es académica y apropiada. Primero introduce el tema, brindando la idea básica de la técnica y un poco de historia y los conceptos, luego sigue la explicación de cómo se realizó la implementación y por último la sección con los resultados empíricos y análisis de los mismos.

   En cuanto a la claridad en la descripción del algoritmo, creemos que no se logró plasmar la idea del mismo de forma simple. Necesitamos varias leídas del mismo para entender verdaderamente cómo era la implementación. Hubiera servido tener alguna figura que tal vez ayudaba a la comprensión y no solo la explicación en prosa.

La notación utilizada es escasa, se explica bastante con palabras, en este sentido creo que ayuda un poco a la lectura porque cuando las explicaciones están muy cargadas con notación se vuelven muy difíciles de seguir.

No encontramos contradicciones pero encontramos un error pequeño en la primera línea del 3er párrafo en la explicación del First Scheme en la página 3 del pdf publicado por la cátedra (página 2 del paper). El error es simplemente un typo en la enumeración de los patrones pues dice "(00, 10, 10, 11)" donde entendemos debería decir (00, 01, 10, 11). Encontramos otro error en la sección del second scheme que explicamos al final del punto 9.

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 este punto aplicamos los tres algoritmos ocultando la imagen .png del logo del itba dentro del archivo .bmp llamado lado.bmp entregado por la cátedra.

   Presentamos el siguiente cuadro comparativo:

Algoritmo	**Calidad del ocultamiento**	**Tamaño necesario del portador**	**Bits modificados**	**Tiempo de ocultamiento**
lsb1	Muy buena, indistinguible a nuestro ojo al solo modificar el LS bit	Grande. 8 veces el tamaño del payload	190182	Orden n. 1 pasada por original escondiendo payload
lsb4	Sigue siendo indistinguible al ojo humanos pero bastante menor la calidad al modificar 4 bits del pixel original	Chico. Solo 2 veces el tamaño del payload	205965	Orden n. 1 pasada por original escondiendo payload
lsbi	Mejor aún que lsb1 al intentar maximizar la cantidad de bits que se mantienen igual entre original y cargada con payload	Ídem lsb1, 8 veces el payload (+ 4 bytes para patrones)	169083	Orden n. Pero 2 pasadas por original al tener que setear la inversión para los patrones antes de esconder

`La comuna de bits modificados contiene la cantidad de bits que se modifican al aplicar los algoritmos respecto del portador orignal.`

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.

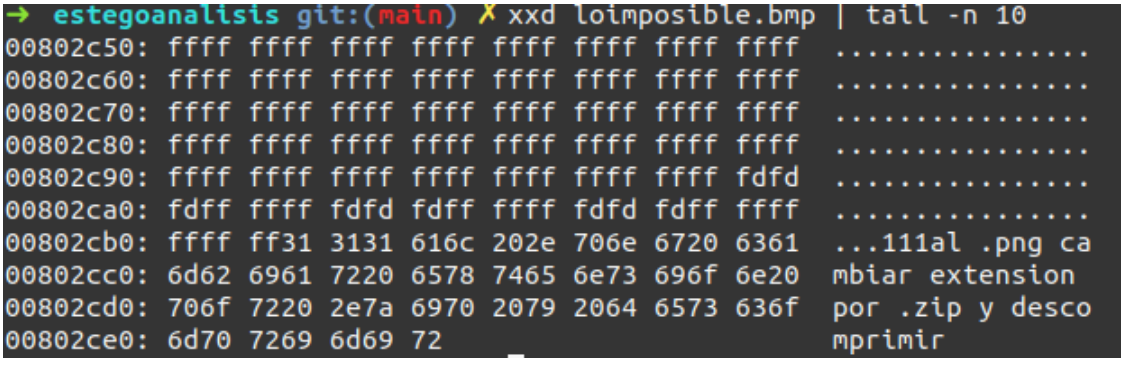
Como no teníamos información sobre cómo estaban ocultos los archivos que nos dieron empezamos probando extraer sin descencriptar para cada método (lsb1, lsb4 y lsbi) cada una de las imágenes. A partir de esta "fuerza bruta" llegamos a lo siguiente:

- `loimposible.bmp` no extrajo nada útil con ningún método
- `madrid.bmp` con lsb4 extrajo un png con una imagen de un buscaminas
- `sherlock.bmp` terminó de correr exitosamente la extracción con lsb1 pero no extrajo nada útil (sospechamos que estaba encriptado)
- `silence.bmp` al extraerlo con lsbi obtuvimos un archivo pdf cuyo contenido era:

Una vez probados todos los métodos de esteganografiado simple llegamos a un camino sin salida porque si bien sospechábamos que el `sherlock.bmp` ocultaba contenido encriptado no teníamos ni el método ni el algoritmo, aunque sí una posible clave. Decidimos probar con otros métodos no lsb y corrimos entonces los siguientes comandos:

```
xxd loimposible.bmp | head -n 10

xxd loimposible.bmp | tail -n 10
```



Como el único png que habíamos obtenido era el del buscaminas, le cambiamos la extensión a una copia de este archivo y extrajimos el zip y obtuvimos el algoritmo y modo de encripción: AES 256 con ECB (en la pregunta 4 explicamos cómo).

Sabiendo ahora el algoritmo, modo de encripción y password pudimos, utilizando lsb1, extraer del archivo sherlock.bmp un archivo .wmv que contenía un video con unas escenas de una serie/película en la que se presenta un método de ocultar información en un tejido.

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

   La imagen `madrid.bmp` tenía oculta con lsb4 una imagen .png. Siguiendo las instrucciones que obtuvimos de otra de las imagenes, cambiamos la extensión del png a .zip y lo descomprimimos obteniendo un archivo llamado `sol9.txt`. Este archivo contenía las siguientes instrucciones:

```
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.
```

   El png en sí era una foto de un buscaminas entonces siguiendo los pasos y utilizando la foto obtuvimos el modo y algoritmo de encripción para extraer el contenido de otra de las imagenes.

   En binario lo que obtuvimos siguiendo los pasos fue:

```
01000001  A
01100101  e
01110011  s

01000101  E
```

```
01100011  c
01100010  b
```

Que al pasarlo a ascii obtuvimos que era: Aes Ecb

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?

   En el video se presenta una manera de ocultar información utilizando como portador un tejido. El método consiste en tejer de forma que si dejamos un hilo vertical encima del horizontal que lo atraviesa se interpreta como un 1, si lo dejamos por debajo, como un 0. De esta forma se puede ocultar información de forma binaria sobre un tejido, portador bastante inesperado. En la escena del video lo que se esconde es un nombre.

   Nos parece que este método de ocultamiento hace que sea increíblemente difícil detectar que el objeto contiene un mensaje oculto para quien no conoce el método. De todas formas, es un método que conlleva un proceso de encripción muy lento al tener que realizar un tejido, a su vez que la desencripción tampoco es demasiado veloz al tener que analizar cada cruzamiento de hilos e ir anotando a mano los 1s y 0s. Obviando además que requiere que al destinatario se le haga entrega de un objeto físico, lo cual no es ideal si no sabemos quién es para enviarlo, dónde está físicamente o si está muy lejos.

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

   En el archivo `loimposible.bmp` encontramos texto plano embebido dentro de la imagen directamente. Allí encontramos que "al .png cambiarle la extensión a .zip y descomprimirlo"

   Aunque no sea posible detectar a simple vista que el bmp tiene información embebida simplemente en texto plano no nos parece un método seguro para ocultar información pues con tan solo usar una herramienta para volcar el contenido hexadecimal del archivo ya se tiene el mensaje oculto. Si la imagen es pequeña podría notarse la diferencia a simple vista.

   Si se está buscando simplicidad a la hora de ocultar y no tanto mantener el mensaje oculto de personas con un mínimo de habilidades y conocimientos de archivos y programación podría ser una alternativa válida.

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

   La gran ventaja que tiene el algoritmo propuesto por estos autores es que se logra obtener una imagen con un payload oculto que se parece muchísimo a la imagen original. No se puede apreciar tanto esta mejora en archivos donde el número de bits invertidos en relación al .bmp original es parecida a la cantidad de bits no invertidos (ya presentes originalmente) en la imagen; pero si nos encontramos en un caso donde los invertidos superan ampliamente a los no invertidos, entonces la mejora es notoria al obtener un archivo con payload oculto muy parecido a la imagen original.

   El hecho de poder ocultar la lógica no tan simple de los patrones en tan solo 4 bits es también un punto muy fuerte del algoritmo.

La desventaja clara que presenta la propuesta es tener que hacer 2 pasadas sobre el .bmp original y el archivo del payload para identificar si se debe invertir o no para cada patrón. De todas formas, una vez hecha esa lógica el algoritmo se convierte en un LSB1 común y corriente estando únicamente atentos a si se debe invertir o no al ocultar.

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

   Se nos ocurre que se podría guardar también en los 4 bits menos significativos pero del final del archivo para dificultarle el trabajo de revelar el secreto a alguien que tenga conocimiento de lsbi, aunque de todas formas no nos parece una mejora importante. Lo mismo si cambiamos el orden de guardado de los patrones en donde sea que guardemos.

   Otra opción sería enviar los patrones por otro canal o de otra forma y que actúe como una suerte de key, aunque no sería muy segura al solo haber 2^4 posibilidades. Es fácilemente atacable por fuerza bruta.

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

   La ventaja que presenta este nuevo esquema es que ahora la imagen con payload oculto es aún más parecida a la imagen original que para el primer esquema. Esto ocurre porque no solo se trabaja con los 4 patrones que forma el 2do y 3er LSB sino que también se abre cada patrón para considerar si se debería invertir los casos donde había un 0 originalemente en la imagen o un 1. Tenemos ahora 8 patrones, lo que nos da mayor calidad de ocultamiento.

   Para casos específicos en los que la mayoría de las veces se tuvo que haber invertido para un bit original del portador y no invertir para el bit contrario (que casi no agrega calidad de ocultamiento con el esquema 1 pues la mitad de los casos sería detrimental la inversión) la ventaja de usar el esquema 2 es muy clara pues aprovecha esa distinción del LSB.

   No estamos seguros por qué en el 3er párrafo de esta sección en su 4ta línea habla de tener que guardar un máximo de 8 patrones, cuando nos parece que siempre se deberían guardar 8 patrones al tener que guardar un 0 o un 1 siempre. De otra forma no sabríamos si hay que invertir o no al no tener valor alguno, ni sabríamos cuál es el patrón que se saltearon si no tenemos 8 valores.

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

    La primera que encontramos no es tanto de implementación sino de su uso por el hecho de la necesidad de que el destinatario de nuestro archivo con payload oculto tenga en su posesión también la imagen original. Sería un problema grande en casos en los que no sepamos la identidad con seguridad del destinatario y enviarle 2 veces una misma imagen por un canal público (pues nos parece no tendría mucha gracia ocultar información que se le enviaría directamente a un destinatario de manera segura) llamaría la atención y daría una oportunidad fácil de aprovechar de comparar los dos archivos lado a lado.

    Una desventaja mínima es tener ahora 8 bits de patrones a ocultar en vez de 4, pero no debería cambiar prácticamente nada.

También está el hecho de tener una lógica un poco más compleja, teniendo ahora que prestar atención también al LSB de los bytes de la imagen original para cada patrón.

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

    La mayor dificultad que encontramos fue entender el algoritmo. Luego de comprender como funcionaba la implementación no nos resultó demasiado complicada, lo más importante fue agregar la funcionalidad que difería de los otros algoritmos LSB en el uso de los primeros 4 bytes para guardar los patrones. A diferencia de los otros algoritmos tuvimos que hacer 2 pasadas por lo que vamos a esconder para precalcular el patrón. Un detalle que encontramos casi al final fue que al precalcular la cantidad de inversiones no habíamos considerado el offset de 4 bytes donde luego se guarda el patrón.

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

    Algunas mejoras que se nos ocurren son:

    - Soportar otros tipos de archivos como portadores. No trabajar solo con bmp, sino también otro formato de imágenes o ni siquiera imágenes. La dificultad está en atarse a los formatos de cada tipo de archivo (por ejemplo el parseo del header que hacemos sobre bmps)
    - Soportar otras versiones de imágenes BMP como portadores y no solo la 3. Deberíamos estar atentos a las diferencias del formato.
    - Soportar archivos BMP comprimidos como portadores. Deberíamos estar atentos a las diferencias del formato.
    - Implementar detección automática del LSB usado y extracción automática del payload.
    - Soportar otras técnicas de esteganografía además de los LSB implementados.
    - Soportar encriptación de a bloques, actualmente nuestra implementación requiere guardar todo el archivo en memoria (el no encriptado o a desencriptar).