Instituto Tecnológico de Buenos Aires

Criptografía y Seguridad

Ingeniería en Informática

Esteganografía en BMPs

Trabajo Práctico Especial de Implementación

Titulares: Ramele, Rodrigo; Roig, Ana

Semestre: 2018A

Grupo: 12

Repositorio: https://github.com/pcostesi/cripto-2018a

Revisión: 29 de junio de 2018

Entrega: 28 de Junio de 2018 a las 23:59hs

Autor: Costesich, Pablo Alejandro (#50109)

Resumen

Se presenta un trabajo práctico especial donde se implementa un programa ‘`stegobmp.jar`‘ con modos de inclusión y extracción de archivos varios a BMPs V3 mediante sistemas de ocultamiento LSB1, LSB4 y LSBE. Se incluye la posibilidad de cifrar el contenido previo a la inclusión utilizando AES en sus variantes 128, 192 y 256, así como también DES. Se soportan los modos ECB, CFB, OFB y CBC.

Se analiza el contenido de cuatro archivos provistos por la cátedra, así como propios, y se detalla el resultado de las particularidades de algunos modos de operación y contenidos. Se propone una serie de mejoras a la implementación.

Dedicado a Jeff Atwood y Joel Spolsky

Índice

1. Introducción

En el repositorio de GitHub se encuentra tanto este informe como el código fuente del programa stegobmp en la carpeta homónima. El mismo está realizado en Java 10 y debe ser construido con Maven 3 o posterior. El informe está realizado en LaTeXy se encuentra en la carpeta report.

Adicionalmente, se provee una serie de archivos de soporte para compilar tanto el informe como la aplicación.

1.1. Compilación

Dado que es necesario utilizar Java 10 para compilar la aplicación por el uso extensivo de funcionalidades provistas por tal versión, el modo recomendado es mediante el archivo de Docker provisto. El mismo contiene todas las dependencias necesarias.

Se provee un helper script para construir el binario mediante Docker, `build.sh`.

El resultado de la compilación es una imagen de Docker que contiene tanto el Jar File como un ejecutable de Windows .EXE.

1.2. Modo de uso

Una vez generado el jar o el exe, debe utilizarse como se describe en la consigna. El ejecutable sigue estrictamente dicha interfaz y advertirá ante cualquier parámetro incorrecto. Bajo ciertas condiciones, algunas excepciones como errores de cifrado y tamaño son expuestos al usuario.

1.3. Generación del informe

El informe también es generado mediante una imagen de Docker que contiene una distribución completa de LaTeX. La misma puede encontrarse en https://github.com/blang/latex-docker.

Basta correr el archivo `latex.sh` para generar el reporte en pdf.

2. Implementación

2.1. Aplicación CLI

2.1.1. Identificación de la línea de comandos

La línea de comandos es parseada con Args4J, que se encarga de todos los detalles como la generación del texto de ayuda y las opciones mutuamente excluyentes.

2.1.2. Métodos de instanciación

Se puede llamar al programa de distintas maneras:

- Embed: genera una instancia de un Encoder<>, con o sin cifrado, dependiendo de si se encuentran presentes los flags correctos.
- Extract: genera una instancia de un Decodedr<>, con o sin cifrado, dependiendo de si se encuentran presentes los flags correctos.
- Guess: aplica fuerza bruta para decifrar archivos.

2.2. Cifrado

Se utiliza CipherInputStream ya que abstrae del manejo de bloques y buffers.

2.2.1. `CipherHelper`

3. Cuestiones a resolver

A continuación se detalla una serie de cuestiones planteadas por la cátedra.

3.1. Posición de esteganografiado

> Para la implementación del programa stegobmp se pide que la ocultación comience en el primer componente del primer pixel. ¿Sería mejor empezar en otra ubicación? ¿Por qué?

Al realizar siempre la oculatción en el primer píxel, existe la posibilidad de notar las alteraciones en el BMP para LSB1, LSB4 y LSBE en las porciones de color blanco en la parte inferior de la imagen. El hecho es evidenciable a simple vista en la imagen "lado.bmp"provista por la cátedra, que aparenta tener una ligera sombra de ruido.º "lluvia"hasta determinada altura. Esto hace que una persona pueda sospechar del contenido de los archivos.

Alterar el header es inviable. La ventaja de ocultar desde el comienzo del bitmap (es decir, luego del header) es que se logra un máximo de aprovechamiento del archivo.

3.2. Ocultamiento en una componente puntual

> ¿Qué ventajas podría tener ocultar siempre en una misma componente? Por ejemplo, siempre en el bit menos significativo de la componente azul.

La principal ventaja es que se reduce el nivel de alteración del archivo, lo que dificulta un poco más el análisis estadístico.

3.3. Comparativa entre sistemas LS*

> 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.

Al realizar un esteganografiado 'pom.xml' (disponible en el repositorio) en un archivo 'lado.bmp' provisto por la cátedra y sin contenido oculto previo, se puede observar para los distintos algoritmos lo siguiente:

Cuadro 1: Comparativa de Sistemas LSB

Ventaja	**LSB1**	**LSB4**	**LSBE**
Tiempo de ejecución	Medio	Alto	Bajo
Alteración visible al ojo	Media a baja	Alta (ruidosa)	Imperceptible
Capacidad de ocultamiento	Media (1:8)	Alta (1:2)	Variable (baja)
Depende del bmp portador	Tamaño fijo	Tamaño fijo	Sí, muy variable

Por detalles de implementación, la copia a buffers muy pequeños en LSB4 hace que sea el más lento. LSB1 es un poco más rápido porque el tamaño de buffer es cuatro veces mayor. LSBE no usa el mismo mecanismo, por lo cual no se ve impactado del mismo modo.

De los tres sistemas, el más evidente al ojo es LSB4, ya que altera más bytes del portador. Asimismo, es el que más capacidad de almacenaje tiene, que sólo depende del tamaño de la imagen y no del contenido. De manera similar, LSB1 altera los bytes de manera menos evidente a costa de la capacidad de ocultamiento. LSBE es imperceptible, pero a costa de depender tanto del tamaño de la imagen como del contenido.

3.4. Esteganografiado de la extensión

> Para la implementación del programa stegobmp se pide que la extensión del archivo se oculte después del contenido completo del archivo. ¿por qué no conviene ponerla al comienzo, después del tamaño de archivo?

Si se serializa al comienzo, luego del tamaño del payload, es trivial detectar si existe esteganografía con sólo leer unos pocos bytes, y detectar si está cifrado o no. Se puede generar una máscara estática sobre los bytes del archivo para detectar la presencia del caracter esteganografiado ".", con lo que se puede extraer la extensión para LSB1 y LSB4, y de manera similar se puede realizar para LSBE. Esto nos da una señal inequívoca de qué sistema se emplea: si encontramos dicha marca entonces el archivo oculto no estará cifrado.

Si el archivo estuviera cifrado, se tiene una serie de bytes al comienzo por cifrar que no cambian. Dado que el IV depende de la clave y no se le puso sal, que se sugirió el uso de MD5 (roto), y que el conjunto de extensiones es pequeño, se introduce una reducción considerable en el espacio de búsqueda de claves; realizar un ataque de fuerza bruta sobre los primeros bytes es factible.

3.5. Modo de análisis de archivos provistos

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

El análisis se realiza en primera instancia mediante la herramienta `stegobmp` en Guess Mode (implementado para realizar aproximaciones iniciales rápidas). El mismo realiza un análisis de fuerza bruta sobre el sistema de esteganografía empleado, usando tanto el tamaño como la extensión a modo de pistas.

Para aquellos archivos con resultados negativos ante el proceso anterior, se realiza un dump de strings usando la herramienta `strings` de Unix. En el ejemplo provisto por la cátedra es fácil encontrar la clave al final de l archivo. Si no fuera así, al contenido se le hace un grep buscando palabras disponibles en un diccionario (se puede encontrar un diccionario de palabras en `/usr/share/dict/words`).

Si el resultado anterior descubre palabras, se emplean como posibles passwords de los archivos que no arrojen resultados ante ninguno de los dos métodos previamente mencionados, y para el archivo donde fue encontrada esa clave. El método utilizado es nuevamente el Guess Mode.

3.6. Contenido de los archivos con secretos

> ¿Qué se encontró en cada archivo?

La cátedra provee para el Grupo 12 cuatro archivos con las siguientes características:

3.6.1. `montevideo.bmp`

El análisis del archivo en modo guess sin password no arroja resultados concluyentes. Ni la extensión ni el tamaño del archivo son válidos para cada sistema.

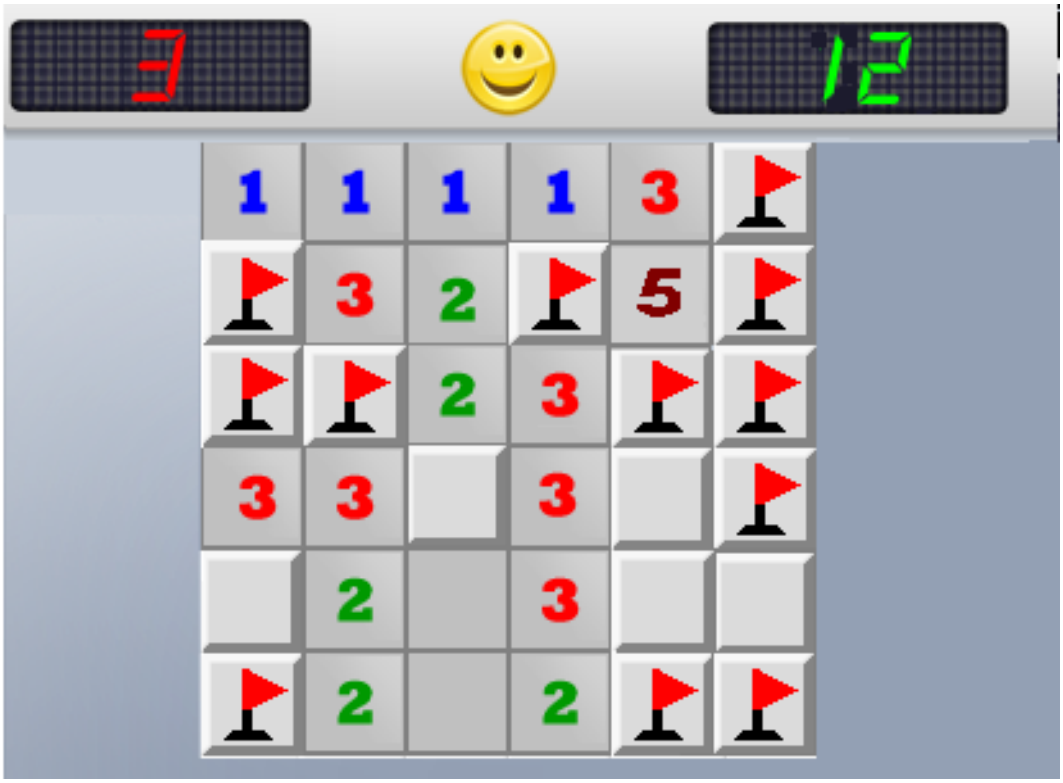
Se procede a utilizar el binario strings. Arroja, sorprendentemente, el resultado de: `h4la password es exitoso` al final del archivo. Ninguna combinación arroja resultados concluyentes. Se asume que el archivo no tiene más datos pasibles de criptoanálisis por ahora.

3.6.2. `silence.bmp`

El resultado inicial del Guess Mode sin password arroja lo siguiente:

```
Guessing if silence.bmp can be decoded with LSB4Splitter
- silence.bmp seems to be a .png encoded with LSB4
Guessing if silence.bmp can be decoded with LSB1Splitter
- Hmm, nope.
Guessing if silence.bmp can be decoded with LSBESplitter
- Hmm, nope.
```

Se renombra la salida para que tenga extensión png y se aprecia una imagen un juego clon de Buscaminas:



Se concluye con alto grado de certidumbre que el archivo silence.bmp contiene un .png que fue esteganografiado con LSB4 y sin password.

3.6.3. `quito.bmp`

Un análisis inicial del archivo con Guess Mode revela la siguiente información:

```
Guessing if quito.bmp can be decoded with LSB4Splitter
- Hmm, nope.
Guessing if quito.bmp can be decoded with LSB1Splitter
- quito.bmp seems to be a .wmv encoded with LSB1
Guessing if quito.bmp can be decoded with LSBESplitter
- Hmm, nope.
```

El análisis inicial nos hace sospechar de un .wmv, por lo que se renombra el resultado con tal extensión. Sin embargo, el archivo no es reconocido por su cabecera. El mismo comienza con 13F44B79, pero se espera 3026B27 según esta lista: .

Procedemos a usar la password encontrada (exitoso) en `montevideo.bmp`:

```
Guessing if quito.bmp can be decoded with LSB1Splitter + decoder aes192 cbc
- quito.bmp seems to be a .wmv encoded with LSB1
```

3.6.4. `lado.bmp`

El resultado inicial del Guess Mode sin password arroja lo siguiente:

```
Guessing if lado.bmp can be decoded with LSB4Splitter
- Hmm, nope.
Guessing if lado.bmp can be decoded with LSB1Splitter
- Extension does not seem to be valid
Guessing if lado.bmp can be decoded with LSBESplitter
- lado.bmp seems to be a .pdf encoded with LSBE
```

El resultado es un .pdf que contiene la frase:

```
al .png cambiarle la extension por .zip y descomprimir
```

3.7. Ocultamiento de otros mensajes

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

El mensaje oculto dentro de otro está en `silence.bmp` y el resultado es un .zip.

El mismo se descubrió en base a otro archivo oculto extraído de `lado.bmp` que contiene la frase:

```
al .png cambiarle la extension por .zip y descomprimir
```

Al zip se le ejecuta el comando 'unzip' de mac. El resultado es un archivo sol9.txt con el siguiente texto:

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.

Se resuelve el juego (que está incompleto) y la secuencia da:

01000001
01100101
01110011

01000011
01100010
01100011

Que traduce a Aes Cbc de 192.

Se ejecuta con password 'exitoso' el stegobmp. Guessing if quito.bmp can be decoded with LSB1Splitter + decoder aes192 cbc - quito.bmp seems to be a .wmv encoded with LSB1

Se obtiene un video data.wmv.

3.8. Ocultamiento del video

> 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 fue `quito.bmp`.

El audio dice: 'Hay que ser cerdo para cortar con alguien por e-mail'

'Ya, como que hay un protocolo aceptable para partirle el corazón a alguien... Eh, un segundo, qué raro... Este archivo es más grande de lo que debería, lo que significa que hay algún error de compresión.'

'O nuestro agente ocultó datos en el archivo.'

3.9. Método NoLSB

> ¿De qué se trató el método de estenografiado que no era LSB? ¿Es un método eficaz? ¿Por qué?

El método fue simplemente agregar al final un texto plano a `montevideo.bmp`. El mismo fue fácil de encontrar ya que simplemente con un editor hexadecimal se puede ver. Es vulnerable a búsqueda de texto plano.

3.10. Mejoras y propuestas

> ¿Qué mejoras o futuras extensiones harías al programa stegobmp?

3.10.1. Guess Mode

Se propone, como se encuentra implementado, el modo "guess", que verifica mediante fuerza bruta el sistema utilizado para archivos esteganografiados no cifrados usando la extensión y verificación de tamaño. De obtenerse tamaños negativos o mayores de lo que puede sostener el archivo portador, éstos son descartados rápidamente. Si el tamaño tiene sentido, se verifica que la extensión tenga caracteres válidos (comience por punto, tenga caracteres legibles válidos de directorio y termine en cero) y que tenga un tamaño razonable.

El mismo sistema se emplea para archivos cifrados cuando se pasa la clave. Se ignora tanto el modo como el algoritmo y se saca el contenido por fuerza bruta.

3.10.2. Bulk Report Mode

Se aconseja que la funcionalidad de Guess Mode sea extendida para proveer un "Bulk Report Mode", que verifique todos los archivos BMP de un directorio y produzca un reporte para cada uno, junto con el archivo oculto (de encontrarse y tener un tamaño razonable límite).

3.10.3. Cambio de posición de ocultamiento

Se propone el siguiente modo: luego de ocultar los primeros cuatro bytes de tamaño (en Big Endian), insertar un número variable de bytes al azar en el rango 1 a 255 como padding hasta alcanzar como máximo N - 1 bytes ocultables (donde N es el payload esteganografiable, que sin contraseña es |archivo| + |extensión con punto y byte '0'| al final). Usar el byte '0' como marca, y colocar el payload. Continuar agregando bytes al azar hasta completar el archivo. Es decir, se emplea el bitmap entero, generando un nivel de ruido similar en todas las secciones del BMP.

3.10.4. API de Streams

Es posible convertir el sistema en dos clases wrappers de Input y Output streams. Esto permite agregar la funcionalidad de esteganografía a cualquier programa que quiera emplearla de modo transparente. Es destacable que la implementación puede soportar canales asincrónicos con mínimas modificaciones.

3.10.5. Cifrado asimétrico

Se puede cifrar una clave simétrica (de archivo, similar a lo que sería una de sesión) con una asimétrica. Las ventajas son:

- Sólo el receptor del mensaje puede leer su contenido real.
- Cualquier persona con la llave pública puede ocultar un archivo, pero no leerlo.
- Se puede agregar una firma para validar que el mensaje dentro del archivo es de una fuente confiable.