Criptografía y Seguridad ***(72.04)***

TRABAJO PRÁCTICO DE IMPLEMENTACIÓN: SECRETO COMPARTIDO CON ESTEGANOGRAFÍA

1 Objetivos

- Introducirlos en el campo de la criptografía visual y sus aplicaciones, a través de la implementación de un algoritmo de Secreto Compartido.
- Introducirlos en el campo de la esteganografía y sus aplicaciones.
- Implementar y analizar un algoritmo descripto en un documento científico.

2 Consigna

Realizar un programa en **lenguaje C** que implemente el algoritmo de Imagen Secreta Compartida descripto en el documento "**(k,n) secret image sharing scheme capable of cheating detection**" cuyos autores son Yan-Xiao Liu, Quin-Dong Sun y Ching-Nung Yang de la Universidad de Tecnología de Xi'an (China).

El programa permitirá:

1) Distribuir una imagen secreta de extensión ".bmp" en otras imágenes también de extensión ".bmp" que serán las sombras en un esquema (k, n) de secreto compartido.

2) Recuperar una imagen secreta de extensión ".bmp" a partir de k imágenes, también de extensión ".bmp"

3 Introducción

La ***criptografía visual*** es un concepto introducido en 1994 por Adi Shamir y Moni Naor. En su presentación en EUROCRYPT'94 ellos consideran un nuevo tipo de esquema criptográfico que puede decodificar imágenes secretas sin usar cálculos criptográficos clásicos. En esencia, el sistema que ellos idearon era una extensión del concepto de ***esquemas de secreto compartido***, pero aplicado a imágenes. Las imágenes que tenían la información secreta, distribuida de manera segura, se podían luego superponer para recuperar la imagen secreta.

El concepto de Esquema de Secreto Compartido también fue, en parte, idea de Shamir. Adi Shamir y George Blakley conciben en 1979, aunque en forma separada, el concepto de Secreto Compartido como una manera de proteger claves.

Tanto Shamir como Blakley exponen que guardar la clave en un solo lugar es altamente riesgoso y guardar múltiples copias en diferentes lugares sólo aumenta la brecha de seguridad. Shamir, por ejemplo, concluye que el secreto (***D***) deberá dividirse en un número fijo de partes ($D_1, D_2, \ldots, D_n$) de forma tal que:

1. Conociendo un subconjunto de ***k*** cualesquiera de esas partes se pueda reconstruir ***D.***
2. Conociendo un subconjunto de ***k-1*** cualesquiera de esas partes el valor ***D*** quede ***indeterminado.***

El documento de Blakley describe una forma de lograr el objetivo de distribuir las sombras de la manera exigida, utilizando conceptos de ***geometría proyectiva***.

El documento que se pide implementar en este trabajo práctico propone un esquema para compartir una imagen secreta basado en el método de Shamir. Para lograr que la imagen que se oculta en las sombras sea prácticamente imperceptible, en el documento se menciona la posibilidad de hacer uso de métodos de ocultamiento, es decir, de ***esteganografía***.

La ***esteganografía*** (del griego στεγανοζ *steganos, encubierto u oculto* y γραπηοζ *graphos, escritura*) es la ciencia que se ocupa de la manera de **ocultar** un mensaje.

La existencia de un mensaje u objeto es ocultada dentro de otro, llamado **portador o camuflaje**. El objetivo es proteger información sensible, pero a diferencia de la criptografía que hace ininteligible dicha información, la esteganografía logra que la información pase completamente desapercibida al ocultar su existencia misma.

La criptografía y la esteganografía se complementan. Un mensaje cifrado mediante algoritmos criptográficos puede ser advertido por un intruso. Un mensaje cifrado que, además, ha sido ocultado mediante algún método de esteganografía, tiene un nivel de seguridad mucho mayor ya que los intrusos no pueden detectar su existencia. Y si por algún motivo un intruso detectara la existencia del mensaje, encontraría la información cifrada.

4 Detalles del sistema.

4.1 Generalidades

El programa debe recibir como parámetros:[1]

➢ **`d`** o bien **`r`**

➢ ***`imagenSecreta`***

➢ ***`número (k)`***

➢ ***`directorio`***

Significado de cada uno de los parámetros obligatorios:

➢ **`d`**: indica que se va a distribuir una imagen secreta en otras imágenes.

➢ **`r`**: indica que se va a recuperar una imagen secreta a partir de otras imágenes.

➢ ***`imagenSecreta`***: Corresponde al nombre de un archivo de extensión .bmp. En el caso de que se haya elegido la opción (d) este archivo debe existir ya que es la imagen a ocultar y debe ser una imagen en blanco y negro (8 bits por pixel) Si se eligió la opción (r) este archivo será el archivo de salida, con la imagen secreta revelada al finalizar el programa.

➢ ***`Número k:`*** El número corresponde a la cantidad mínima de sombras necesarias para recuperar el secreto en un esquema (k, n).

➢ ***`directorio`*** El directorio donde se encuentran las imágenes en las que se distribuirá el secreto (en el caso de que se haya elegido la opción (d)), o donde están las imágenes que contienen oculto el secreto (en el caso de que se haya elegido la opción (r)). Debe contener imágenes de extensión .bmp, de 8 bits por píxel, de igual tamaño que la imagen secreta. Además, deberá verificarse que existan por lo menos k imágenes en el directorio.

Ejemplos: (Asumiendo que en los directorios indicados hay 8 imágenes)

➢ Distribuir la imagen “Albert.bmp” según esquema (4,8) ocultando las sombras en las imágenes del directorio “covers/280x440”:

```
./ss d Albert.bmp 4 covers/280x440/
```

➢ Recuperar la imagen “secreto.bmp”, en un esquema (4,8) buscando imágenes en el directorio “shares/280x440/”

```
./ss r secreto.bmp 4 shares/280x440/
```

4.2 Algoritmo de Distribución

En la distribución hay que tener en cuenta los siguientes aspectos:

4.2.1 Imagen secreta

La imagen secreta debe ser de formato BMP, de 8 bits por píxel. (1 byte = 1 píxel)

El formato BMP es un formato de archivos **binario** de imagen bastante simple. Consta de dos partes:

i. encabezado → de 54 bytes

ii. Cuerpo → de tamaño variable.

El encabezado contiene información acerca del archivo: tamaño de archivo, ancho de imagen, alto de imagen, bits por píxel, si está comprimido, etc

IMPORTANTE: Leer bien el valor que indica en qué offset empieza la matriz de píxeles, ya que puede comenzar inmediatamente después de los 54 bytes del encabezado, o bien empezar más adelante.

En el cuerpo del archivo bmp, están los bits que definen la imagen propiamente dicha. Si la

[1] Respetar el orden y sintaxis de los parámetros.

imagen es de 8 bits por píxel, es una imagen en tonos de grises: el píxel de valor 0x00 es de color negro y el píxel 0xFF es de color blanco.

Tener cuidado al elegir la imagen: revisarla con algún editor hexadecimal para asegurarse que no tenga información extra al final (metadata) y que se ajuste al formato que se pide.

Como la imagen se va a subdividir en bloques de 2k-2 bytes, el total de pixeles debe ser divisible por 2k-2.

4.2.2 Valores de k y de n

El valor de k puede ser mayor o igual que 2 y menor o igual que n.

Pero para poder ocultarlo fácilmente por esteganografía, usaremos sólo valores de k entre 3 y 8. Es decir, k puede ser 3,4,5,6,7,8. Sino, se rechaza.

4.2.3 Trabajo en el cuerpo Z_{251}

Todas las operaciones se realizarán en el cuerpo entero de congruencias módulo 251.

Tener en cuenta que la división en Z251 significa multiplicar por el inverso.

Así, por ejemplo, si tuviéramos que dividir por 2, sería lo mismo que multiplicar por 126.

El inverso de 2 es 126, ya que $2.126 \equiv 252 \equiv 1\ (251)$.

Conviene hacerse una tabla de inversos al inicio del programa y tenerla disponible para hacer todas las divisiones.

4.2.4 Elección del entero aleatorio ri

El entero ri debe elegirse aleatoriamente en el rango [1,251]

El entero ri no puede ser cero.

4.2.5 Coeficientes a_0, a_1, b_0 y b_1

Estos coeficientes deben ser distintos de 0.

Si a_0 o a_1 fueran 0, se considerarán como 1 para hacer el cálculo de b_0 y b_1.

4.2.6 Imágenes Portadoras y ocultamiento por esteganografía

Para esta implementación, se usarán imágenes portadoras de formato BMP y de 8 bits por píxel, de igual tamaño que la imagen secreta. Para facilitar toda la operatoria, asumiremos que son de igual alto y ancho que la imagen secreta.

IMPORTANTE:

De la imagen secreta sólo se guardan los pixeles (no el encabezado).

Dichos pixeles se leen y guardan en orden [0, 1, 2, …] (es decir como se presentan los bytes después del offset).

Una vez obtenidas las sombras con el algoritmo propuesto, se ocultan en las imágenes portadoras.

Usaremos los métodos LSB2 y LSB4.

LSB2 - Inserción en los dos bits menos significativos

Se toman de a 2 los bits del mensaje y se ocultan en los 2 bits menos significativos de la componente del píxel que corresponda.

LSB4 - Inserción en los cuatro bits menos significativos

Es una variante del anterior, donde 4 bits del mensaje se ocultan en los 4 bits menos significativos de la componente del píxel que corresponda.

Para k = 3 y 4, usaremos LSB4. Para k = 5, 6, 7 y 8 usaremos LSB2.

En algunos casos, todos los bytes de la portadora serán usados, y en otros no. Al haber elegido imágenes portadoras del mismo tamaño de la imagen secreta, y con el uso de LSB4 para k menor que 5 y LSB2 para los otros valores de k se asegura que habrá lugar para ocultar toda la imagen secreta.

Por ejemplo, suponiendo que el primer byte de la sombra es **11011010**, siendo k = 6, y el offset de la portadora empieza en 1078.

Por ser k = 6, elegimos LSB2.

Luego, de la portadora se usarán los últimos 2 bits de los 4 bytes 1078,1079,1080 y 1081.

	Original	Después de ocultar **11011010** en LSB2
Byte 1078	01110111	0111011**11**
Byte 1079	01110111	0111010**01**
Byte 1080	01110110	0111011**10**
Byte 1081	01110100	0111011**10**

4.2.6 Ocultamiento del número de sombra.

Para la correcta resolución de la recuperación se necesitará saber cuál fue el número de sombra que se calculó en cada caso.

El número de sombra (valor entre 1 y n) se guardará en plano, directamente en el valor del byte reservado (campo **`bfReserved1`** del encabezado, inmediatamente después de size)

4.3 Algoritmo de Recuperación

En la recuperación hay que tener en cuenta los siguientes aspectos:

4.3.1 Imágenes portadoras

Las imágenes portadoras deben ser de formato BMP, de 8 bits por píxel y todas del mismo tamaño (ancho y alto) entre sí. Si no se tienen k imágenes que cumplan esta condición, se muestra mensaje de error y no se realiza nada.

Luego al resolver, la imagen secreta conservará los datos de encabezado de cualquiera de las imágenes portadoras, siendo entonces de igual formato BMP, de 8 bits por píxel y de igual ancho y alto.

4.3.2 Recuperación del secreto

Se obtiene recuperando los bytes de cada una de las k sombras y el número de sombra guardado en el campo **`bfReserved1`** del encabezado

Luego deberá obtenerse los coeficientes de los $f_i(x)$ y $g_i(x)$ de cada bloque. Existen distintas maneras de hacerlo (interpolación de Lagrange, método de Gauss, etc). En cualquier caso, recordar que todas las operaciones son en congruencias módulo 251.

No olvidarse de validar el valor de r para cada bloque.

5 Cuestiones a analizar.

Deberán analizarse las siguientes cuestiones:

1. Discutir los siguientes aspectos relativos al documento/paper.
    a. Organización formal del documento (¿es adecuada? ¿es confusa?)
    b. La descripción del algoritmo de distribución y la del algoritmo de recuperación. (¿es clara? ¿es confusa? ¿es detallada? ¿es completa?)
    c. La notación utilizada, ¿es clara? ¿cambia a lo largo del documento? ¿hay algún error?
2. El título del documento hace referencia a que es capaz de detectar sombras falsas (cheating detection). ¿cómo lo hace? ¿es un método eficaz?
3. ¿Qué desventajas ofrece trabajar con congruencias módulo 251?
4. Con este método, ¿se podrían guardar secretos de cualquier tipo (imágenes, pdf, ejecutables). ¿por qué? (relacionarlo con la pregunta 3)
5. ¿En qué otro lugar o de qué otra manera podría guardarse el número de sombra?

6. ¿Qué ocurriría si se permite que r, a_0, a_1, b_0 o b_1 sean 0?
7. Explicar la relación entre el método de esteganografía (LSB2 o LSB4), el valor k y el tamaño del secreto y las portadoras.
8. Analizar cómo resultaría el algoritmo si se usaran imágenes en color (24bits por píxel) como portadoras.
9. Discutir los siguientes aspectos relativos al algoritmo implementado:
    a. Facilidad de implementación
    b. Posibilidad de extender el algoritmo o modificarlo.
10. ¿En qué situaciones aplicarían este tipo de algoritmos?

6 Organización de los grupos

El trabajo será realizado en grupos de, máximo, 3 integrantes.

7 Sugerencias

Se sugiere encararlo en forma modularizada, probando por separado las cuestiones:

- manejo de archivos bmp

- operaciones en campos finitos

- Interpolación polinómica.

- esquema propuesto con bloques pequeños (Por ej.: k = 4, n = 10, secreto de 60 bytes)

Una vez que se aseguran de que por separado funciona, integrarlo.

8 Entrega

La fecha de entrega es el día **21 de junio**.

Cada grupo enviará por mail a la cátedra el archivo con el proyecto realizado (en C o en Java), junto con la documentación correspondiente al uso del programa.

Además presentarán el mismo día un informe con la solución correspondiente a la recuperación del secreto a partir de los archivos que se le entregarán oportunamente al grupo y el detalle de lo analizado en el punto 5 (Cuestiones a analizar).

No se aceptará:

- Que no haya README con indicaciones de compilación / ejecución.

- Que no funcione en Pampero.

- Que no respeten el orden de los argumentos del programa.

- Que no funcione para los archivos que se les dé al grupo (es la manera de asegurar que respetaron el algoritmo propuesto)

9 Sobre los archivos a entregar por mail.

- El entregable debe ser un archivo comprimido cuyo nombre debe cumplir el formato: `grupoXX.(zip|tar.gz|rar)` donde `XX` es el numero de grupo.
- Debe respetar la estructura de carpetas:
    - **docs/** (Documentación e informe)
    - **src/** (Fuentes)
    - **README.txt** (en el root, **incluir comentarios pertinentes** para la ejecucion correcta de scripts y binarios asi como también dependencias de la aplicación)
    - Incluir makefile en el root. Debe generar sólo el binario a ejecutar. **No debe incluirse el binario en la entrega**.
- Excluir de la entrega:
    - Enunciado
    - Cualquier tipo de binario generado por el make.

- Carpetas .svn y __MACOSX
- Archivos de prueba entregados por la cátedra.

- Deben incluirse **únicamente los printf explicitados** en el enunciado. En caso de incluirse más printf que los especificados, deben ejecutarse únicamente especificando una opción de verbose.

- **Es condición necesaria de aprobación su correcto funcionamiento en entorno pampero de ITBA.**

- Debe respetarse la sintaxis de ejecución del enunciado. Respetar incluso las mayúsculas y minúsculas.
- Utilizar códigos de error correctos. Por ejemplo, utilizar EXIT_FAILURE y EXIT_SUCCESS de stdlib.h.
- El programa debe explicitar errores. Por ejemplo, si hubo un error en un parámetro de entrada, se debe informar al usuario su error e informar la sintaxis correcta.

10 Criterios de Aprobación

Para aprobar el trabajo, se tendrán en cuenta:

- Entrega en la fecha indicada.
- **Que el programa pueda efectuar la distribución del secreto y la recuperación del mismo para los archivos entregados por la cátedra.**
- Que el contenido del informe sea correcto y completo, esto es, que estén contestadas todas las cuestiones del punto 5.
- Que el archivo ejecutable y el código en C se ajusten a los requerimientos y a lo establecido en el apartado 9.

La nota se conformará en un 60% por el programa y en un 40% por el informe. Son obligatorios el informe y el programa.

Si el trabajo, presentado en la fecha 21 de junio, resultara luego desaprobado, se podrá recuperar una sola vez. El trabajo recuperado sólo podrá tener una nota máxima de 4 (cuatro)

Para la entrega, así como para cualquier inconveniente, el mail de contacto es:

- Ana Arias: mroig@itba.edu.ar

11 Material de lectura:

- Liu, X., Sun, Q., Yang C. "(k,n) secret image sharing scheme capable of cheating detection", EURASIP Journal on Wireless Comunication and Networking,2018. **Disponible en https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-018-1084-7 (Visitado Febrero 2023)**
- Capítulo 15 de Computer Security – Art and Science, Matt Bishop, Addison-Wesley, 2004
- Capítulo 10 y 12 de Handbook of Applied Cryptography, Alfred J. Menezes, Paul C. Van Oorschot, Scott A. Vanstone, CRC Press, 1997