Criptografía y Seguridad

Trabajo práctico de implementación

Secreto compartido en imágenes con esteganografía

Integrantes

- Alderete, Facundo
- de la Puerta Echeverría, María
- Menzella, Facundo
- Romarión, Germán

Docentes

- Abad, Pablo
- Roig Arias, Ana María
- Ramele, Rodrigo

Fecha de entrega: 22 de Junio de 2015

Objetivo y consigna

El objetivo de este trabajo práctico es aprender sobre criptografía visual y sus aplicaciones, implementando un algoritmo de Secreto Compartido en Imágenes.

Como consigna, se nos pidió realizar un programa en lenguaje C que implemente el algoritmo descrito en el documento "*Secret Image Sharing*" de Chih-Ching Thien y Ja-Chen Lin. El programa deberá distribuir una imagen secreta en otras imágenes que serán las sombras en un esquema *(k, n)* de secreto compartido y luego recuperar una imagen secreta a partir de *k* imágenes. Todas las imágenes serán de extensión ".bmp".

Análisis general

Aspectos relativos al documento de Thien y Lin

Organización formal del documento

La organización del documento nos parece adecuada ya que es clara y ordenada, sobretodo en la descripción de los pasos a seguir del algoritmo.

El *paper* describe en su introducción el origen del concepto *secret sharing* y cómo se aplicaría a imágenes, lo cual resulta útil para entender por qué surgió este método en primer lugar.

En la segunda sección se describe con detalle cómo se dividen una imagen en *n* sombras y por qué hacen falta *r* o más sombras para recuperarla. A continuación, se detalla el método propuesto por los autores para compartir imágenes secretas, exponiendo el "*step by step*" tanto para distribuir la imagen como para revelar el secreto.

Luego se muestran resultados experimentales en los que se ve una imagen, su permutación, las *shadows* generadas y luego la recuperación exitosa de la misma y se hace un análisis de seguridad sobre el método propuesto, detallando sus beneficios.

Por último, las conclusiones explican las ventajas de utilizar este método.

Descripción del algoritmo de distribución y de recuperación

La descripción del algoritmo comienza con la etapa de distribución cuyo primer paso consiste en recorrer el arreglo que representa los píxeles de la imagen y truncar su valor a 250 si se usa el método con pérdida. De lo contrario, si el valor de píxel es mayor o igual a 250, se guarda dicho valor y a continuación la diferencia entre el valor del píxel y el valor 250.

Luego, se divide la imagen en secciones de *r* píxeles cada una. Dichos píxeles representan a los coeficientes de un polinomio de grado *r - 1* que luego se evaluará para todos los valores de *n*, es decir la cantidad de sombras a generar. A continuación se explica que cada uno de estas evaluaciones del polinomio se asignan a las *n* sombras. El proceso de selección y de transformación de los *r* píxeles continúa hasta haber recorrido toda la imagen original.

A continuación, se detalla la fase de revelado. En primer lugar, se debe tomar el primer píxel de las *r* sombras y usar la interpolación de Lagrange junto al valor de *n* para obtener los coeficientes correspondientes a los *r* píxeles de la imagen original. Luego de haber obtenido todos los píxeles, se deben permutar los píxeles de la imagen obtenida para recuperar la imagen original.

Notación utilizada

La notación utilizada nos resultó clara. No notamos cambios a lo largo del documento.

Truncamiento de píxeles mayores a 250

Tal como se explicó anteriormente, el documento detalla un método de revelado sin pérdidas que maneja los píxeles cuyo valor es superior a 250. El proceso descrito incrementará el tamaño de la sombra generada, aunque usualmente es en poca cantidad ya que son pocos los valores que superan este número. El método indica que si se halla un valor superior a 250, entonces se debe “partir” al píxel (p_i) en dos valores, uno de 250 y otro de (p_i - 250), y guardar en ese mismo orden. Luego se repiten los pasos descritos anteriormente.

Criterio utilizado para elegir imágenes portadoras y para ocultar las sombras en el caso de *k* distinto de 8

El criterio utilizado para *k* distinto de 8, fue que las imágenes portadoras deben ser de tamaño igual al tamaño de las *shadows* por 8.

```
sizeof(porter_image) = 8 * sizeof(shadow)
```

De esta forma, siempre se utiliza la totalidad de la imagen portadora, guardando en cada byte exactamente un bit. Entendemos que no hay una mejor solución, o por lo menos no encontramos una que en el trade-off de tiempo, claridad y precisión fuese mejor que la elegida.

En un primer momento se barajó la opción de que en el caso que *k* fuese menor que 8, entonces debíamos sobreescribir los 2 bits menos significativos de la imagen. A pesar de no ser una mala alternativa, requiere persistir exactamente cuántos bytes se utilizan en cada imagen portadora. Como los únicos bytes reservados de la imagen bmp van del 6 al 9 (los cuales son usados para transmitir la semilla y el valor de n con el cual fueron generadas), se optó por la metodología mencionada anteriormente. Además, el hecho de introducir más ruido a la imagen expone en cierta forma el “secreto”, o por lo menos expone que “algo no anda bien”.

Otra condición utilizada es que para $k \neq 8$ las imagenes deben ser cuadradas. Esto se hizo para que se mantenga una concordancia entre el tamaño de las imagenes portadoras y el secreto. Utilizando el k, se puede calcular que tanto “más grande” o “más chica” es una imagen portadora con respecto a la imagen secreto. De no ser cuadradas, habría que guardar el ancho y el alto en las imagenes portadoras y, como se explico anteriormente, descartamos esa posibilidad.

Aspectos relativos al algoritmo implementado

Alternativa al método de interpolación de Lagrange

El método de interpolación de Lagrange para obtener un polinomio no resulta conveniente a la hora de tener que implementarlo en computadora.

Dado que representamos a las imágenes y sombras como matrices, decidimos implementar el método de eliminación de Gauss-Jordan para hallar la inversa de una matriz.

Dada la siguiente ecuación:

$$A * B = C,$$

siendo:

A: una matriz cuadrada de dimensión *k*, cuyas filas representan a un polinomio de grado *k - 1*, evaluado para los valores de n correspondientes.

B: una matriz columna de *k* filas que contiene a los coeficientes (píxeles de la imagen original) de los polinomios de A.

C: una matriz columna de *k* filas que contiene los valores de los píxeles de las sombras.

Para poder hallar los valores de B, es imprescindible hallar la inversa de A:

$$A^{-1} * A * B = A^{-1} * C$$
$$B = A^{-1} * C$$

El algoritmo que halla A^{-1} sigue los siguientes pasos:

1. Crear la matriz identidad de dimensión *k*.
2. Para cada columna de A:

a. Se posiciona al puntero en el elemento **e** ubicado en la diagonal.

b. Si el valor en dicha posición no es cero, se multiplica a toda la fila por el inverso multiplicativo modular (módulo 251) de ese valor. Se refleja esta operación en la matriz identidad.

c. Para cada fila **f** de A, donde **f** no contiene a **e**:

i. Se define **g** la fila que contiene a **e**.

ii. Se le sustrae a la fila **f** la fila **g***e**.

iii. Se refleja la operación en la matriz identidad.

Luego de haber seguido dichos pasos, en la matriz A quedará la identidad, y en la matriz identidad quedará la inversa de A.

Facilidad de la implementación

Las dificultades en la implementación son en parte relacionadas al lenguaje de programación. En un principio optamos por desarrollar el trabajo práctico en Java, ya que es el lenguaje con el que más cómodos nos sentimos. Sin embargo, tuvimos dificultades con los tipos

de datos que maneja. Este problema surgió en una fase temprana del desarrollo, con lo cual no nos resultó tedioso tener que migrar a C, un lenguaje más amigable con los tipos de datos que necesitamos para este trabajo, y que ofrece un mejor manejo a nivel bit.

Posibilidad de extender el algoritmo a imágenes a color

El algoritmo se puede extender, realizando las validaciones de independencia lineal en cada canal de color rojo, verde y azul. Ver: C.-C. Chang, C.-C. Lin, C.-H. Lin, and Y.-H. Chen 'A novel secret image sharing scheme in color images using small shadow images'.

Dificultades en la lectura del documento y/o en la implementación

El documento no presenta grandes dificultades en la comprensión.

Extensiones o modificaciones al algoritmo

Actualmente nuestra implementación no contempla imagenes comprimidas. Podria extenderse para soportarlo.

Decidimos implementar el método de distribución y recuperación de imágenes con pérdida de información, ya que truncamos todos los valores de los píxeles mayores a 251. Habiendo dicho esto, existe la posibilidad de extender al algoritmo para ofrecerle al usuario la opción de distribuir y recuperar imágenes usando el método sin pérdida de datos.

Aplicaciones del algoritmo

Los algoritmos que combinan esteganografía con secreto compartido son aplicables, pero no están limitados, a las siguientes áreas:

a. Comunicaciones confidenciales y almacenamiento de datos secretos

Si el método es propiamente usado, y la imagen portadora no resulta alterada para el ojo humano, entonces es muy conveniente para almacenar cualquier tipo de información: imágenes, contraseñas, marcas de agua, firmas digitales, etc.

b. Protección ante alteración de datos

Como se requieren *k* componentes para recuperar la información original, resulta más complicado para un atacante reconstruir dicha información y alterarla.

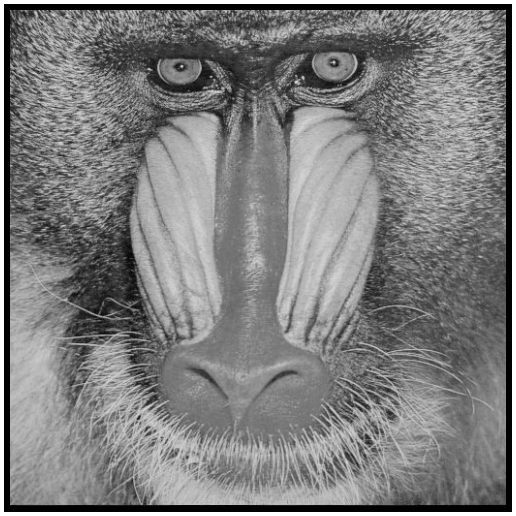
c. Terrorismo/hacking

Algunas organizaciones con fines maliciosos usan estos métodos en sitios masivos (blogs, sitios de compra/venta, redes sociales, etc.) donde realizan alguna publicación, y ocultan en sus avatar información acerca de objetivos o del próximo ataque. Dicha información puede ser recuperada a partir de los avatares de k publicaciones.

Resultados obtenidos

Todas las pruebas (8, n) se realizaron escondiendo la siguiente imagen:

En las siguientes imagenes portadoras:



Las pruebas para (k < 8, n) se realizaron escondiendo la imagen:

En la siguiente imagen :

Se entiende que el caso k > 8 es análogo a el caso de k < 8, por eso no se adjuntan imagenes de prueba. Lo único a tener en cuenta es que se deben proporcionar las imagenes que satisfagan las condiciones mencionadas en los puntos anteriores.

Imagenes reveladas de la catedra

Hemos tenido problemas para revelar la imagen asignada a nuestro grupo, ya que la semilla guardada en las imágenes portadoras no parece ser la indicada. Probablemente se

deba a que la cátedra corrió el programa de distribución con una implementación de la librería Random C distinta de la nuestra.

A modo de compensación, procedemos a mostrar las imágenes escondidas en las imágenes portadoras que la cátedra le asignó a todos los grupos.

Conclusiones

Comprobamos que el método funciona tanto en modo distribución como en modo recuperación. De todas formas, al haber testeado el algoritmo con k considerablemente pequeños, no sabemos cómo se comportaría con k grandes.

También verificamos que *secret-sharing* resulta muy efectivo al momento de querer ocultar un secreto en una imagen, ya que no resulta posible que el ojo humano pueda distinguir si una imagen fue alterada para ocultar información.

Entendemos que al ser un método para compartir un secreto, tiene sentido que la persona que inicia el proceso (la persona que desea compartir un secreto) se tome el trabajo de buscar las imagenes portadoras adecuadas, las cuales deben cumplir con ciertas precondiciones: formato *bmp*, color blanco y negro, tamaño exacto para almacenar las sombras, y sin compresión.

Anexo 1 - Ejecución del programa

En la carpeta lena se encuentra la imagen lena512 la cual es una imagen bmp de tamaño 512x512. En la carpeta *lena_porters2k* se encuentran dos imagenes: *barbara1.bmp* y *mandril.bmp*, ambas bmp de tamaño 512 x 512.

Para correr el programa en modo distribución, colocar en la carpeta *porters* las *n* imágenes elegidas para distribuir (se pueden colocar *n* imágenes iguales) y ejecutar:

```
make clean
make sii
./sii -d -s ../lena/lena512.bmp -k <min> -n <porters> -dir ../porters/
```

Para correr el programa en modo *recover*, colocar en la carpeta *porters* las imágenes necesarias para recuperar el secreto. Si acaba de realizar el paso de distribución mencionado anteriormente, simplemente ejecute:

```
./sii -r -s ../output/ -k <min> -dir ../porters/
```

En la carpeta ../output encontrara la imagen resultante. Dependiendo del sistema operativo, existe la posibilidad de que la imagen no pueda ser abierta correctamente. Basta con arrastrar la imagen a cualquier navegador, como puede ser Google Chrome, para ver la imagen de lena completamente recuperada.