Criptografía y Seguridad

Secreto compartido en imagenes con esteganografía

Trabajo Práctico de Implementación

Alumnos: Serber Valeria (51021), di Tada Teresa (52354), Altamiranda Enzo (51265), Ontivero Cristian (51102)

Fecha de Entrega: Lunes 22 de Junio de 2015

Número de Grupo: 1

1 - Introducción

Se implementó un algoritmo de secreto compartido en imagenes que permite distribuir y recuperar imagenes en formato BMP.

2 - Objetivos

Implementar y analizar el algoritmo descrito en el documento *"Secret Image Sharing"* cuyos autores son *Chih-Ching Thien* y *JaChen Lin* para introducirnos a la criptografía visual y la esteganografía.

El programa realizado en lenguaje C permitirá distribuir una imagen secreta en formato BMP generando sombras que se ocultarán en otras imagenes del mismo formato, siguiendo una variación del esquema de umbral *(k,n)* de Shamir, para secreto compartido. También posibilitará el recupero de la imagen secreta a partir de *k* imagenes.

3 - Análisis del algoritmo

3.1 - Aspectos relativos al documento de Thien y Lin.

3.1.1 Organización formal del documento.

El paper presenta en orden: una introducción, un repaso del modelo de umbral *(k,n)* de Shamir, una propuesta de un método de secreto compartido en imagenes con explicación separada para encripción y desencripción del secreto, un ejemplo de cómo funciona con sus respectivas imagenes, un análisis de seguridad del algoritmo que analiza que se cumpla que con *k-1* sombras no se puede obtener suficiente información para revelar el secreto, un beneficio de utilizar el algoritmo y las respectivas conclusiones.

3.1.2 La descripción del algoritmo de distribución y la del algoritmo de recuperación.

Para la distribución de la imagen secreta, se dividió la misma en *n* sombras, con el objetivo de que la imagen pueda ser reconstruida con *k* o más sombras. Para eso se genera un polinomio de grado *k-1*, donde los coeficientes son *k* píxeles con valor entre 0 y 250. Se evalúa ese polinomio de *1* a *n* para obtener *n* píxeles, uno para cada sombra.

Inicialmente se truncan los valores de los píxeles que son mayores a 250 para que estén en el rango requerido. Se usa una semilla para generar una permutación de los píxeles de la imagen que agrega seguridad. Luego, secuencialmente se toman *k* píxeles que aún no han sido elegidos y se utilizan como coeficientes en el polinomio, y como se explicó anteriormente, se evalúa el mismo. Esto sigue repitiendose hasta que todos los píxeles de la imagen fueron procesados.

Para la recuperación de la imagen secreta se utilizan *k* sombras, de las *n* disponibles. De cada sombra, se elige un píxel que aún no ha sido utilizado. Se resuelve el sistema de ecuaciones que se genera con esos píxeles y cada ecuación correspondiente a cada sombra. En esta implementación se hizo con el algoritmo de eliminación *Gauss-Jordan*, para obtener los coeficientes de los polinomios. Esto se repite hasta que todos los píxeles de las *k* sombras son procesados. Los coeficientes resultantes son los píxeles de la imagen permutada anteriormente, por lo tanto, luego se aplica permutación inversa para obtener la imagen secreta.

3.1.3 La notación utilizada, ¿es clara? ¿cambia a lo largo del documento?

La notación utilizada en el documento es clara y concisa, salvo algunas excepciones como por ejemplo cuando habla de secciones de pixeles sin explicar a qué se refiere hasta más adelante. Asume que el lector tiene cierto conocimiento sobre polinomios y resolución de sistemas de ecuaciones modulares. Menciona que el sistema debe resolverse con interpolación de Lagrange, pero no explica cómo hacerlo y no sugiere otros métodos posibles de resolución que también pueden servir. La notación es consistente a lo largo de todo el documento.

3.2 Aspectos relativos al algoritmo de Thien y Lin.

3.2.1 Método para evitar el truncamiento de píxeles mayores que 250 bits. ¿Puede haber otras alternativas?

Para evitar el truncamiento de píxeles mayores o iguales a 250 bits se sugiere partir los mismos en dos píxeles distintos, uno con el valor 250 y otro con el valor del píxel menos 250. Estos se van colocando en un array junto a los demás píxeles en el orden correspondiente a la imagen y el array se utiliza para generar los coeficientes con secciones que contengan *r* pixeles. Al descifrar un secreto si aparece un píxel de valor 250 leo el siguiente y los uno.

Si se usara un número primo más chico se podría utilizar el mísmo método de dividir el número en dos.

3.3 Criterio utilizado para elegir imagenes portadoras en el caso de k distinto de 8 y propuestas descartadas.

En el caso de que k sea distinto de 8 se buscan imagenes en el directorio proporcionado que tengan el tamaño adecuado, ya sea la cantidad de bytes necesarios o más. Si no se consiguen de éste tipo o la cantidad deseada el programa genera un error. Si la imagen es más grande, se oculta dejando los últimos bytes sin nada oculto.

No hubo propuestas descartadas ya que desde el comienzo se utilizaron imagenes con más bytes de las necesarias y no hubo problemas al respecto ni se necesitó optimizar espacio.

3.4 Aspectos relativos al algoritmo implementado:

3.5.1 Facilidad de implementación

El algoritmo no presentó demasiada dificultad de implementación salvo por el hecho de que había que tener cuidado en los índices de la eliminación de *Gauss-Jordan*, tener cuidado con los módulos y detalles que eran pequeños pero si estaban mal el algoritmo no hacía lo que queríamos que hiciera.

3.5.2 Posibilidad de extender el algoritmo para que se usen imagenes en color.

Si la imagen tuviera 16 bits por píxel permitiría utilizar 65.536 colores en lugar de 256 como las de 8 bits por píxel. En este caso para el algoritmo de Shamir se debería utilizar un primo menor a 65.536, como por ejemplo 65.521 y se podría truncar los bits mayores a este número de la misma manera que explica el documento para imagenes en tonos grises.

3.5 ¿Qué dificultades tuvieron en la lectura del documento y /o en la implementación?

En la implementación tuvimos diversas dificultades. Una de ellas fue que al levantar bytes no habíamos tenido en cuenta si la computadora usaba *big endian* o *little endian*, por lo cual a veces fallaba porque estábamos asumiendo uno en particular. Por otro lado, al hacer la tabla de inversos multiplicativos para realizar la eliminación de *Gauss-Jordan* nos olvidamos del último inverso multiplicativo del número 250 (que es 250) por lo cual había errores en la resolución del sistema de ecuaciones con las imagenes de la cátedra pero no con las nuestras porque justo ese número no lo usaban.

En la lectura del documento no tuvimos demasiadas dificultades, salvo que no se proporcionaba una forma para resolver el sistema de ecuaciones y tuvimos que elegir la que mejor nos parecía.

3.6 ¿Qué extensiones o modificaciones harían a la implementación o al algoritmo?

Se podría extender para un mayor soporte de las diferentes variaciones del formato BMP (en el código se asume una cabecera DIB del tipo “BITMAPINFOHEADER”, pero existen otras variantes). También sería posible seguir el método propuesto en el documento que evita el truncado de valores, evitando la pérdida de información.

Otra modificación viable al programa sería simplificar la lista de parámetros necesarios, combinando la funcionalidad de “-secret” con “-r” y “-d” (es decir, seguir “-r” y “-d” por el nombre de la imagen revelada o a distribuir).

Con respecto al tamaño de la imagen a revelar, si bien se optó por que sea conocida por el usuario y se ingrese como parámetro del programa, se podría agregar al final de las imagenes, luego del arreglo de píxeles de la imagen, ya que la cabecera del BMP permite saber dónde comienza la imagen y hasta dónde llega.

3.7 ¿En qué situaciones aplicarían este tipo de algoritmos?

En cualquier situación en la que se desee repartir la responsabilidad y capacidad de recuperar el secreto entre varios individuos, y/o se necesite redundancia para poder recuperar el secreto. Por ejemplo, tendría sentido en ámbitos militares en los que se desea que nadie tenga acceso a armas nucleares por sí solo, sino que necesite que varios estén de acuerdo, pero a su vez se quiera distribuir esta capacidad entre varias personas de forma tal que si algunas no pueden aportar su parte de la clave por cualquier razón (por ejemplo se encuentran prisioneros), se pueda igual recuperar el secreto.

Otro uso menos drástico podría ser, por ejemplo, para resguardar la clave privada de un certificado raíz de una autoridad certificante importante, dificultando las firmas que realiza y que no se pueda robar la misma de un solo lugar o a una sola persona.

4 - Solución del recupero de imagenes

Se corrió el algoritmo con la opción de recuperar, con *k* igual a 8, *width* 300 y *height* 300 con las 8 últimas imagenes dadas por la cátedra para el grupo 1, obteniendo la imagen secreta de la Figura 1.

Figura 1: *imagen secreta de la cátedra*

5 - Conclusiones

El algoritmo de secreto compartido en imagenes con esteganografía permite esconder imagenes efectivamente de manera que se pueda recuperar una imagen secreta a partir de un conjunto de sombras sin necesidad de obtener todas las sombras generadas. Esto es conveniente porque puedo protegerme si pierdo sombras o de que alguien que posee una sola pueda leer el secreto.

Es una alternativa óptima para utilizar el método de Shamir ya que al tener las sombras más chicas que las imagenes originales, se pueden ocultar en diferentes imagenes captando poca atención.

.