Trabajo Práctico de Implementación

Secreto Compartido con Esteganografía

Fecha de entrega: 21/06/2023
Grupo: 6

Alumnos:
Agustín Mattiussi - 61361
Julián Sasso - 61535
Malena Vasquez Currie - 60072
Sol V. Anselmo - 61278

Docente:
Ana María Arias Roig

Índice

1. Introducción

El objetivo de este trabajo es implementar el algoritmo de Imagen Secreta Compartida descripto en el documento "(k,n) secret image sharing scheme capable of cheating detection". Se usará para poder distribuir y recuperar imágenes secretas de extensión bmp.

2. Aspectos Relativos al Documento/Paper

2.1. Organización Formal

La organización del documento es adecuada y sigue los lineamientos de la mayoría de los papers académicos, donde cada subtítulo referencia al tema a detallar. Una crítica posible es la elección de presentar el texto en dos columnas ya que esto dificulta levemente la lectura.

2.2. Claridad de Descripción

Los algoritmos de distribución y reconstrucción tienen una descripción clara y completa. Al principio los autores brindan una explicación genérica, indicando los pasos necesarios para cada proceso. Hacia el final del documento presentan un ejemplo que ayuda a solidificar la información brindada y ayuda a visualizar mejor los algoritmos. Para lograr una mayor comprensión del paper, es recomendable contar con conocimientos previos de álgebra (particularmente polinomios y campos finitos).

2.3. Notación Utilizada

La notación utilizada es en su mayoría clara y correcta. Los símbolos elegidos le permiten al lector una fácil asociación con las variables que representan y se mantienen consistentes a lo largo del documento. Sin embargo, tuvimos dudas interpretando que debíamos concatenar los valores de los pares $v_{i,j} = (m_{i,j}, d_{i,j})$. A su vez, cuando el paper habla de la Image Reconstruction Phase define el índice como $j = 1, 2, \ldots, k$ dando a entender que j representa los primeros k valores de n cuando en realidad debe tomar cualquiera de los k valores pertenecientes a n.

3. Detección de Sombras Falsas

Dadas k sombras de una imagen secreta, dividida en t bloques de pixeles, el proceso para identificar las falsas es el siguiente:

- Extraer las partes $v_{i,j} = (m_{i,j}, d_{i,j})$ de cada secreto $S_1, ..., S_k$
- Para cada grupo de partes, reconstruir $f_i(x)$ y $g_i(x)$ a partir de los m y d obtenidos en el punto anterior usando Lagrange
- Sean $a_{i,0}, a_{i,1}, b_{i,0}, b_{i,1}$ los coeficientes de x^0 y x en $f_i(x)$ y $g_i(x)$ respectivamente. Verificar si existe un entero en común r_i que satisface que $r_i a_{i,0} + b_{i,0} = 0$ y $r_i a_{i,1} + b_{i,1} = 0$. En caso negativo, **existen sombras falsas**.

No es un algoritmo que se puede destacar por su eficiencia computacional, ya que de solo hacer la interpolación de Lagrange se cuenta con una complejidad temporal de $o(n^2)$. Igualmente, es eficiente en cuanto a resultados ya que cumple el objetivo. Otro posible método para hallar las funciones es la eliminación de Gauss con complejidad temporal de $o(n^3)$. Hay soluciones de $o(nlog^2(n))$.

4. Desventajas de Mod 251

La congruencia que se usa decide el tamaño de las partes en las cual se divide cada sombra (funciona como un límite superior), por lo que el espacio de acciones está limitado. Si bien el 251 sirve ya que es el último primo menor a 255 (máximo valor para una imagen en blanco y negro), causa que las partes sean más grandes. Esto puede llegar a complicar los cálculos ya que trabajar polinomios de coeficientes altos hace que sean más díficiles de resolver. Además, como los píxeles de las imágenes con las que trabajamos tienen 8 bits cada uno, estos toman valores entre 0 y 255. Por lo tanto, en los píxeles más blancos de la imagen, es decir los que tienen valores superiores a 251, puede haber cierta pérdida de información. A continuación, se mostrará un ejemplo de este caso donde tenemos una imagen que fue distruibuida en diferentes imágenes portadoras y como quedó tras hacer la reconstrucción.

(a) Imagen original

(b) Imagen reconstruida

5. Guardar Secretos de Cualquier Tipo

El método de Shamir con detección de trampas se puede utilizar para guardar secretos de cualquier tipo de archivo, incluyendo imágenes, PDF y ejecutables. Esto se debe a que el esquema de Shamir no está directamente relacionado con el tipo de datos que se desea proteger, sino con la distribución y reconstrucción de un secreto en general. En otras palabras, como este método utiliza un polinomio de grado *k-1* para dividir un secreto en acciones, cada acción puede representar cualquier tipo de dato.

Sin embargo, como se mencionó en la sección anterior, existe la posibilidad de que haya cierta pérdida de información al distribuir el secreto. Por lo tanto, para otro tipo de archivos, si bien es posible utilizar este método, no sería lo más recomendable ya que al distribuir el secreto en ejecutables por ejemplo, al modificar algún bit puedo corromper el archivo.

6. Guardar el Número de Sombra

Siguendo las instruciones del documento, el número de sombra es guardado en el primer bit reservado del header de las imagenes bmp. Pero este valor podría tambien almacenarse de otras maneras sin afectar la integridad de la imagen:

- **En el segundo bit reservado del header**.
- **En la metadata de la imagen**: al utilizar campos específicos de los metadatos de la imagen, no se afectará directamente la estructura interna de la imagen, y el numero de sombra será accesible para su recuperación posteriormente.
- **En almacenamiento externo**: si bien sería mas difícil la recuperación del número de sombra, el almacenamiento en un entorno externo proporcionaría un nivel adicional de seguridad.

7. Valores Nulos para a_0, a_1, b_0, b_1, r_o

Como se mencionó en la Sección 3, la forma de determinar si existen sombras falsas es usando $r_i a_{i,0} + b_{i,0} = 0$ y $r_i a_{i,1} + b_{i,1} = 0$. Si se permite que los coeficientes $a_{i,0}, a_{i,1}, b_{i,0}, b_{i,1}$ sean nulos, resultará que cualquier valor r_i cumple la condición. Es decir, el esquema devolverá un resultado incorrecto ya que no pudo hacer el análisis correspondiente. Además, $a_{i,0}, a_{i,1}$ no pueden ser nulos debido a que deben ser invertibles para poder detectar situaciones donde se esté usando una sombra falsa (cheating).

Por otro lado, en caso de que r_i sea nulo, $b_{i,0}, b_{i,1}$ serían 0 sin importar los valores de $a_{i,0}, a_{i,1}$ por lo que no se podría reconstruir el esquema ni recuperar el secreto aunque se utilicen todas sombras lícitas. El funcionamiento del esquema quedará ligado al valor de $b_{i,0}$ o $b_{i,1}$ ya que y las ecuaciones planteadas siempre resultarán verdaderas.

8. Método de Esteganografía

Nuestra implementación trabaja con valores de k entre 3 y 8. Para los valores de k mayor a 4, se utiliza el método LSB2 que consiste en ocultar el mensaje secreto de a 2 bits en los 2 bits menos significativos de la componente del píxel que correspoda. El método LSB4 es análogo. A su vez, el tamaño de la imagen (el secreto) debe ser divisible por 2k-2. Por lo que podemos inferir que a mayor valor de k, se utilizan menos bloques para ocultar la imagen.

Por último, es necesario que las imágenes portadoras sean del mismo tamaño que el secreto debido a que en el peor de los casos, todos los bytes de la portadora serán modificados. En otras palabras, debemos asegurarnos que el tamaño de las imagenes portadoras tengan el mismo tamaño que la imagen secreta para asegurarnos la correspondencia adecuada de píxeles cuando se hace el ocultamiento.

En conclusión, es una mejor opción utilizar un k de mayor tamaño para asi requerir una menor cantidad de bits en las portadoras y hacer un ocultamiento más imperceptible.

9. Imágenes Portadoras en Color

Dada una imagen bmp, la información puede ocultarse en los bits de menor importancia (más a la derecha). Para imagenes portadoras en color se debe tener en cuenta que estas requieren 24 bits para representar cada pixel: 8 para rojo, 8 para verde y 8 para azul. Por lo que, si se usaran imagenes bmp de color, el algoritmo deberá asignar los componentes RGB. Sin embargo, permitiendo que este algoritmo utilice imágenes a color, los bits modificados en las imágenes portadoras será más evidente.

10. Aspectos Relativos al Algoritmo Implementado

10.1. Facilidad de Implementación

La implementación se realizó de forma modularizada, tal como recomendaba el enunciado. La división fue la siguiente:

- Manejo de archivos bmp
- Operaciones en campos finitos
- Interpolación polinómica
- Esquema propuesto con bloques pequeños

Esta decisión facilitó el desarrollo ya que agilizó los tiempos y contuvo los errores. Las únicas complicaciones enfrentadas durante el proceso fue trabajar con congruencia 251 e implementar la interpolación polinómica. Esto último se debió a que Lagrange require una incognita x como parte de su reconstrucción, y hacerlo en lenguaje C no nos resultó sencillo. Por lo que se optó por trabajar con el método de Gauss.

10.2. Posibilidad de Extender el Algoritmo

Una posible modificación podría servir para permitir que se acepten imágenes en diferentes formatos o a color, siempre teniendo en cuenta que los archivos deben poder leerse como bits para operar. Además, se podría utilizar el método de ocultamiento LSB1 que solo modifique el último bit menos significativo de la imágen. Este método, generaría cambios casi imperceptibles en las imágenes portadoras, mejorando el nivel de seguridad del esquema de secreto compartido.

Por otro lado, cabe destacar que en nuestra implementación la distribución del secreto en las imágenes portadoras se hace in-place. Una posible mejora sería que se realice una copia de las mismas en un directorio y realizar la distribución en este con el objetivo de preservar las imágenes originales.

11. Situaciones donde Aplicaríamos este Tipo de Algoritmos

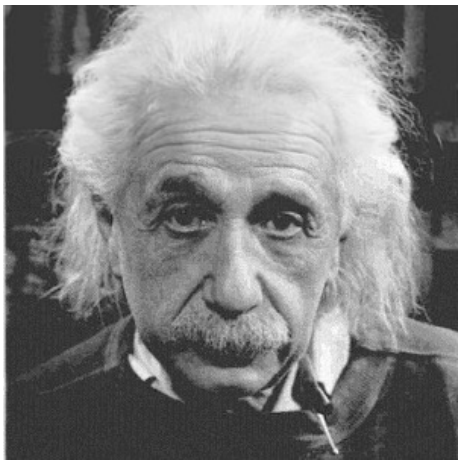
Este tipo de algoritmos se puede aplicar en cualquier situación donde se necesita garantizar la seguridad o integridad de información entrante. En particular, el Esquema de Shamir se puede aplicar en cualquier escenario en el que sea necesario proteger un secreto distribuyendo su acceso o recuperarlo entre múltiples entidades.

12. Recuperación del Secreto

La cátedra nos dió una colección de imágenes que tienen oculto con k = 4 la siguiente imagen secreta:

Figura 2: Imagen recuperada

Estas imágenes son:

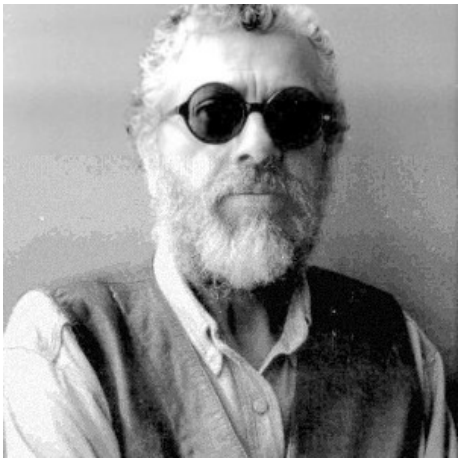


(a) Albertshare.bmp

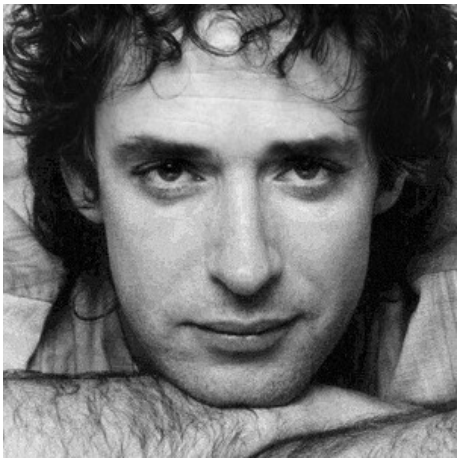
(b) Alfredshare.bmp

(c) Audreyshare.bmp

(d) Evashare.bmp



(e) Facundoshare.bmp



(f) Gustavoshare.bmp

(g) Jamesshare.bmp

(h) Marilynshare.bmp