TPI Grupo-03:

Secreto Compartido con Esteganografía

Santiago Sandrini - 61447

Federico Gustavo Rojas Pelliccia - 60239

Leonardo Agustín D'Agostino - 60335

Roberto Franco Rodriguez Tulasne - 60089

Introducción

En este trabajo práctico se buscó introducirnos a las áreas de la criptografía visual y la esteganografía a través de la implementación de un algoritmo de secreto compartido en imágenes en lenguaje C. El detalle del mismo se encuentra en el paper "**(k,n) secret image sharing scheme capable of cheating detection**" **(Yan-Xiao Liu, Quin-Dong Sun y Ching-Nung Yang)**. El objetivo del algoritmo es ocultar una imágen secreta entre múltiples sombras, con la condición de que no se pueda obtener la imágen original a menos que se cuenten con una cantidad mayor o igual a k <= n de sombras y que a su vez no se pueda obtener ningún tipo de información adicional sobre el secreto al disponer entre 0 y k - 1 sombras. Además el algoritmo posee la particularidad de "detectar trampas", es decir que algún actor entregue una sombra falsa al querer recuperar el secreto. El mismo está basado en el esquema de Shamir de secreto compartido.

El programa implementado tiene como objetivo:

1- Distribuir una imagen secreta de extensión ".bmp" en otras imágenes de la misma extensión, siendo estas las sombras en el esquema (k, n) de secreto compartido.

2- Recuperar una imagen secreta de extensión ".bmp" a partir de k imágenes de la misma extensión.

Análisis del Paper

Organización

El paper tiene una **buena organización**, primero dando una breve introducción histórica y del estado del arte en cuanto a esquemas de secreto compartido visuales y polinómicos y comentando sobre esquemas con capacidades de detección de “cheating”, explicando también a que se refiere este concepto.

Luego de esta breve introducción, se procede a dar definiciones formales del esquema de shamir, la detección de cheating y un esquema de distribución de imágenes secretas con secreto compartido (Thien-Lin). Con esta sección preliminar el lector ya posee los conceptos y definiciones para comprender el nuevo esquema presentado en el paper.

La tercera sección ya presenta el esquema propuesto en sí y luego se procede a dar resultados, discusiones y conclusiones en secciones posteriores.

Lo único que resultó en confusiones fue la división del texto en dos columnas, una distribución tradicional facilitaría la lectura en formatos digitales al no tener que desplazarse hacia arriba al terminar de leer la primera columna de cada página. En formatos en papel esto no presentaría un problema.

Descripción y notación

En cuanto a la **descripción del algoritmo**, se encontró una descripción detallada y completa que le permite al lector poder comprender cómo implementar el esquema propuesto de una manera correcta y sin mayores dificultades. La inclusión de la sección preliminar es fundamental para que los lectores introduzcan o refresquen sus conocimientos del tema para poder comprender el resto del paper.

Se detalla en subsecciones la generación de sombras y la reconstrucción del secreto, a su vez realizando comparaciones con los esquemas ya existentes. De esta forma el

lector puede seguir de una forma estructurada los pasos de cada algoritmo y además comprender las mejoras y soluciones que ofrecen los escritores en el nuevo esquema propuesto.

No obstante, por la naturaleza del problema, se encontró que ciertas explicaciones pueden llegar a resultar confusas a primera lectura, necesitando ser revisadas múltiples veces; especialmente las **notaciones** cuentan con muchos subíndices que dificultan la comprensión inicial. Sin embargo se entiende que la inclusión de tantos subíndices es necesaria para la definición formal de los algoritmos, siendo la notación consistente. Cabe destacar que existe una inconsistencia entre fórmulas presentadas y utilizadas que será discutida más adelante.

Para agregar, manejar tantos subíndices vuelven más propensos tanto al lector como al escritor a cometer errores. Por ejemplo, en la página 3, en la sección de *"Image reconstruction phase"* del inciso 2 se encontró que falta un último subíndice al desarrollar la fórmula de la $f(x)_j$.

En el paper la fórmula figura como:

$$f(x)_j = a_{j,0} + a_{j,1}x + a_{j,2}x^2 + \ldots, \; + \; a_{j,k-}x^{k-1}$$

Cuando en realidad falta restarle 1 a la k del subíndice final:

$$f(x)_j = a_{j,0} + a_{j,1}x + a_{j,2}x^2 + \ldots, \; + \; a_{j,k-1}x^{k-1}$$

A su vez, se encontró también una inconsistencia de notación en la página 5, sección *"Results and discussion"*:

"Assume the first block B1 consists of the following 6 pixels: (57, 68, 90, 231, 42, 89), the dealer selects an integer r1 = 9, and generates two k − 1 = 3-th degree polynomials: f1(x) = 57 + 68x + 90x2 + 231x3 and g1(x) = 161 + 104x + 42x2 + 89x3, where 57 + 9 ∗ 161 = 0(mod251), 68 + 9 ∗ 104 = 0(mod251)."

En este ejemplo se utilizó una fórmula distinta para la selección de a_0, a_1, b_0 y b_1 a la presentada por el paper anteriormente:

“The dealer chooses a random integer ri, and computes two pixels bi,0, bi,1 which satisfy that: riai,0 + bi,0 = 0, riai,1 + bi,1 = 0 over GF(251).”

Por lo que el cálculo de b_0 y b_1 para mantener consistencia debería haber sido:

$$9 * 57 + b_0 = 0 (mód\ 251) \ -> \ b_0 = 240$$

$$9 * 68 + b_1 = 0 (mód\ 251) \ -> \ b_1 = 141$$

Sombras falsas

El esquema del documento parte del esquema desarrollado por Thien y Lin, y busca agregar un sistema de detección de sombras falsas. Dicho sistema se basa en el modelo propuesto inicialmente por Tompa y Woll. Si en alguna de las sombras se detecta “cheating”, entonces se debe frenar la reconstrucción. De lo contrario, se continúa.

Puntualmente el algoritmo planteado establece que en la fase de reconstrucción de la imagen, si entre los $a_{i0}, a_{i1}, b_{i0}\ y\ b_{i1}$ coeficientes de los polinomios $f_i(x)\ y\ g_i(x)$ de algún bloque, no se encuentre un valor r_i que satisfaga que $r_i a_{i0} + b_{i0} = 0$ y $r_i a_{i1} + b_{i1} = 0$ entonces hay sombras falsas y por consecuencia se detectó cheating. Se busca que los bytes $a_0, a_1, b_0\ y\ b_1$sean consistentes y se hayan generado de la misma variable r .

Eficacia del cheat detecting

En cuanto a la eficacia de detección de trampas, en las propias conclusiones del paper se encuentra una tabla comparativa con otros esquemas (Harn, Pieprzyk y Sergio), en dicha tabla se puede ver como el esquema propuesto puede detectar cheating con hasta k − 1 cheaters con probabilidad de cheating exitoso de $\frac{1}{251}$ por bloque, mientras que los otros esquemas pueden detectar cheating con menor cantidad de atacantes, o igual cantidad pero con mayor probabilidad de realizar un ataque exitoso.

En la sección 3 se incluye un teorema con demostración para probar que el esquema detecta cheating con hasta k - 1 cheaters. En el teorema se específica que para hacer cheating sin ser detectado se deben elegir dos polinomios f* y g* junto a un número r* que satisfaga:

$$r * a_0 * + b_0 * = 0$$

y

$$r * a_1 * + b_1 * = 0$$

Obteniendo así dos polinomios en la fase de reconstrucción de imágen:

$$f ** (x) = f(x) + f * (x)$$

$$g ** (x) = g(x) + g * (x)$$

Luego si existiese un r' en común tal que:

$$r'(a_0 *+ a_0) + b_0 *+ b_0 = 0$$

y

$$r'(a_1 *+ a_1) + b_1 *+ b_1 = 0$$

se evitaría la detección de cheating. Sin embargo esto sólo puede ocurrir si r* = r, y como el atacante no posee ninguna información de r (probado en un teorema anterior en el

paper), debe elegir r* al azar entre 251 valores por lo que la probabilidad de evitar la detección de cheating es de $\frac{1}{251}$.

Es importante notar que esta probabilidad corresponde a evitar el cheat detection en un único bloque, por lo que la probabilidad de engañar al esquema por completo con una sombra falsa es de $(\frac{1}{251})^{|bloques|}$. Esto hace el método de detección muy eficaz, especialmente en comparación a otros mencionados en el paper, ya que la probabilidad de engañar al sistema es de $(\frac{1}{251})^{|bloques|}$, muy baja para un atacante por fuerza bruta, y además con detección de hasta k - 1 atacantes utilizando solo interpolaciones de Lagrange.

Desventajas del módulo 251

En cuanto a las desventajas de trabajar con aritmética modular con módulo 251, lo primero que se debe tener en cuenta es que las operaciones se vuelven computacionalmente más complejas, especialmente en el cálculo de inversos multiplicativos, que para optimizar su cálculo se debió utilizar una tabla estática con los inversos de los números en módulo 251. En relación a esto, también se debe tener en cuenta que los píxeles con valor 0 y 251 se deben transformar a 1 para que tengan inverso multiplicativo.

La conversión mencionada de píxeles 0 y 251, junto a los píxeles representados por 252, 253, 254, 255 (ya que los píxeles son representados por 8 bits [0,255]) causarán pérdida de información de la imágen original al ser transformados. Especialmente los valores 251, 252, 253, 254 y 255 que en escala de grises representan píxeles muy claros, se verán como píxeles muy oscuros al aplicarse aritmética modular módulo 251 sobre ellos. Esto se vé claramente en la Figura 1 de la siguiente página.

Este atributo se traslada como desventaja si se quiere utilizar el esquema para el ocultamiento de otro tipo de archivos, ver las siguientes secciones.

Ocultamiento de otro tipo de archivos

Como se mencionó en la sección anterior, utilizar aritmética modular con módulo 251 provoca pérdida de información al transformar valores fuera del rango modular o no inversibles. Esto genera limitaciones en las posibilidades de tipo de datos/archivos que se pueden ocultar con el esquema propuesto.

Al utilizar imágenes bmp en escala de grises, la pérdida de información al utilizar aritmética modular con módulo 251 no es algo muy significativo, uno puede recuperar la imagen con algunos píxeles erróneos y seguir interpretando la información correctamente.

Por ejemplo en la imágen obtenida a partir de las sombras enviadas por la cátedra:

Figura 1: imágen oculta en imágenes de cátedra

Aunque se pueden ver píxeles erróneos en la reconstrucción del secreto, la imágen puede seguir siendo interpretada por cualquier actor que conozca el edificio del Parlamento Británico en Londres.

Por otra parte, si se quisiera ocultar otro tipo de archivos, si los mismos requieren de todos los bits presentes correctamente para funcionar adecuadamente como por ejemplo un ejecutable, lo más probable que ocurra es que no se pueda recuperar el secreto ya que el resultado obtenido de dicha recuperación será un archivo corrupto o inutilizable. En consecuencia el secreto no se podría recuperar, se pierde.

También se podría querer ocultar archivos que, aunque no se corrompan por el esquema de ocultamiento, la pérdida de información sería tan grande que el archivo recuperado

no sería de utilidad. Podría ser por ejemplo un archivo pdf (aunque este también podría resultar corrupto).

Por lo tanto, al utilizar este esquema para ocultar información, se debe analizar cuál es el impacto de pérdida de información al recuperar el archivo, y uno decidir si el esquema es adecuado o no para ocultar dicha información. El ocultamiento por esteganografía es adecuado en archivos como imágenes, videos o audios, pero no resulta útil para archivos que puedan resultar corrompidos o inutilizables al alterar bytes arbitrarios del mismo.

Número de sombra

Como se detalla en la consigna, es necesario conocer el número de la sombra que se calculó en cada caso para poder recuperar el secreto. Esto se hace guardándolo en plano en el byte reservado del encabezado.

Otra forma de guardar el número de sombra podría ser utilizando los primeros 4 bits más significativos de dos píxeles elegidos específicamente, evitando así que se sobreescriban estos bits al utilizar LSB2 o LSB4 para ocultar información, ya que estos sobreescriben los bits menos significativos. El problema de esto es que estos píxeles se verían modificados con respecto a los de la imágen original (además de su modificación si ocultan información).

Una segunda manera sería con un archivo auxiliar, donde se guarde el número de sombra junto a un hash identificatorio de la misma. Para tener aún mayor seguridad de que no se repetirán los hash de cada sombra, se puede utilizar un *“salt”* junto a la función de hash y guardar este valor también. De esta forma si por alguna razón dos sombras fueran exactamente iguales, sus valores de hash no lo serían.

Valores de $r,\ a_0,\ a_1,\ b_0,\ b_1$

La principal razón por la que estos valores no pueden ser cero es para poder realizar correctamente el cheat detecting:

$$a_0 * r + b_0 = 0\ (mód\ 251) \qquad y \qquad a_1 * r + b_1 = 0 (mód\ 251),\ con\ r\ E\ [1, 250]$$

Por un lado, si los cuatro valores $a_0,\ a_1,\ b_0,\ b_1$ fueran igual a cero entonces la detección de cheating sería trivial ya que r podría tomar cualquier valor y entonces no se podría detectar si hubo trampas. En cambio si cualquier par $a_0,\ b_0$ o $a_1,\ b_1$ fuese cero, la seguridad se reduciría a que r deba solo resolver una de las ecuaciones, en lugar de necesitar que r resuelva ambas ecuaciones en simultáneo, ya que la otra ecuación será siempre cero.

A su vez, para que $b_0,\ b_1$ puedan valer cero (por cómo se calculan estos valores en la construcción de sombras), r debe valer cero o su respectivo $a_0,\ a_1$ debe valer cero. En el caso de que r fuese cero, nuevamente no se podría realizar la detección de cheating, ya que las ecuaciones estarían resueltas para cualquier valor de $a_0,\ a_1$. Análogamente si $a_0,\ a_1$ fueran ambos cero, se podría utilizar cualquier r ya que ambas ecuaciones estarían resueltas. El caso de que solo $a_0\ o\ a_1$ fuese cero se desarrolló en el párrafo anterior, reduciendo el cheat detecting a resolver una única ecuación en lugar de dos en simultáneo.

Por lo tanto los valores $r,\ a_0,\ a_1,\ b_0,\ b_1$ no pueden ser cero (ni ningún valor que en congruencia módulo 251 sea cero), ya que pueden inhabilitar por completo la detección de cheating o reducir drásticamente su detectabilidad.

Método de esteganografía

Lo primero que se debe mencionar es que al utilizar un k más alto el número de bloques será más chico según la fórmula:

$$\#B_k = \frac{height * width}{2k - 2}$$

Es por esto que a menor valor de k habrá mayor cantidad de bloques, por lo que se requerirán mayor cantidad de bits para ocultar la información (mayor cantidad de polinomios) por lo que a su vez se modificarán mayor cantidad de píxeles en las sombras y mayor cantidad de bits por píxel. Al contrario, al utilizar un k más alto y tener una cantidad de bloques menor, se tendrá también una cantidad menor de polinomios, osea una cantidad menor de información a ocultar y a su vez se utilizarán menor cantidad de píxeles en las sombras y menor cantidad de bits por píxel.

El siguiente ejemplo es útil para visualizar esto:

Figura 2: imágen original

Figura 3: imágen con k=8 LSB2

Figura 4: imágen con k=3 LSB4

Interpretando el ejemplo, se ve claramente cómo utilizar un valor más alto de k es preferible para disimular el ocultamiento de información en las sombras.

La consigna del TPI especifica lo siguiente: *"Para k = 3 y 4, usaremos LSB4. Para k = 5, 6, 7 y 8 usaremos LSB2."*. Sabiendo que el tamaño de las imágenes son de 280 x 440, las relaciones se obtienen con los siguientes cálculos:

$$\#B_3 = \frac{280 * 440}{3*2-2} = 30800 \qquad \#B_4 = \frac{280 * 440}{4*2-2} \ (no\ divisible)$$

$$\#B_5 = \frac{280 * 440}{5*2-2} = 15400 \qquad \#B_6 = \frac{280 * 440}{6*2-2} = 12320$$

$$\#B_7 = \frac{280 * 440}{7*2-2} \ (no\ divisible) \qquad \#B_8 = \frac{280 * 440}{8*2-2} = 8800$$

Luego para cada bloque, se deben guardar los valores de f(j) y g(j) = 16 bits. Por lo que para cada bloque, si se usa LSB2 (2 bits por píxel) se deberán utilizar 8 píxeles.

$$K = 8,\ 8800 * 8 = 70400 < 280 * 440 = 123200$$

$$K = 6,\ 12320 * 8 = 98560 < 280 * 440 = 123200$$

$$K = 5,\ 15400 * 8 = 123200 = 280 * 440 = 123200 -> límite$$

Luego para valores de K menores a 5 ya no se puede utilizar LSB2 para fotos de tamaño 280 x 440, no alcanzan los píxeles. Con LSB4 (4 bits por píxel) se requieren utilizar 4 píxeles por bloque.

$$K = 3,\ 30800 * 4 = 12320 = 280 * 440 = 123200 -> límite$$

Imágenes a color

Si se tuviera que utilizar una imagen a color se tendrían que tomar las siguientes consideraciones:

El cambio sería que cada píxel de 24 bits de una imagen a color se representaría por 3 canales (RGB), los primeros 8 bits son la intensidad del rojo, los siguientes 8 bits la intensidad del verde y los últimos 8 bits la intensidad del azul.[1] Esto es indistinguible para el esquema a una imágen en escala de grises tres veces más grande que la a color, sin embargo para el ojo humano si será más difícil la interpretación de los píxeles con pérdida de información mencionados anteriormente, ya que la aritmética modular módulo 251 puede afectar a cualquier de los tres canales del píxel en simultáneo, provocando que este tome un color distinto al original. Sin embargo esto sólo ocurriría en los píxeles con pérdida de información, mientras que el resto se mantendrían como se mostraban en la imágen original. El esquema podría extenderse para manipular imágenes a color, con esta particularidad.

De igual forma, siempre se debe tener en cuenta el tamaño del secreto y de las sombras, para calcular los límites de k con los cuales se utiliza LSB2 o LSB4.

[1] RGB Color Codes Chart 🎨 (rapidtables.com)

Análisis de implementación

Facilidad de implementación

La implementación del esquema en lenguaje C tuvo sus dificultades, aunque no asociadas a la implementación propia del esquema del paper. De hecho, las explicaciones detalladas paso a paso en el paper para construcción y recuperación resultaron muy comprensibles para trasladarse a código, aunque cabe destacar que las sugerencias y observaciones dadas por la profesora también fueron un complemento fundamental para la correcta implementación.

La mayor complicación de implementación que enfrentamos fue la interpolación del polinomio, en un principio pensamos en utilizar el método de Gauss o la interpolación “normal” de Lagrange, pero tras no llegar a resultados correctos con estos optamos por utilizar la versión “encajada” de la interpolación de Lagrange.

Otra complicación fue el ocultamiento y recuperación de sombras con esteganografía, hubo que detenerse a pensar y probar las operaciones con bits (shifts, or, etc).

A su vez tuvimos complicaciones propias de implementaciones en C con el manejo de memoria (liberación y leaks, tamaños, buffer overflow, chequeos, etc).

Extensiones o modificaciones

Una primera posible extensión es la de utilizar el esquema para ocultar imágenes a color en imágenes sombra también a color, como se detallo en la sección correspondiente. O también extenderlo para utilizar otro tipo de archivos como archivos de audio o videos.

Una segunda extensión es aumentar la cota superior de k, para poder así reducir aún más la cantidad de bloques a utilizar y permitir el uso de LSB1. A su vez al llegar a valores de k muy altos que utilizan muy pocos píxeles de la imágen se podría utilizar alguna función para seleccionar dónde se inserta el mensaje oculto, es decir en qué píxeles de

la sombra. De esta forma se pueden obtener sombras con un ocultamiento muy disimulado.

Una tercer extensión sería realizar algún tratamiento de los bits afectados por la aritmética modular, por ejemplo como mencionamos los píxeles con valor 251, 252, 253, 254, 255 en escala de grises pasarán de ser muy claros a muy oscuros con módulo 251. En nuestro caso se podría transformar dichos píxeles al valor 250, para mantener la claridad de los mismos, seguiremos teniendo pérdida de información pero los píxeles visualmente serán más representativos de su valor original, no tendrán la misma luminosidad que sus contrapartes originales pero al menos se mantendrán en un tono muy claro en lugar de ser muy oscuros.

Aplicaciones del algoritmo

Con este esquema lo que se busca principalmente es obtener ciertos atributos para la manipulación de imágenes sensibles o confidenciales como:

- Almacenamiento seguro: al dividir la imágen secreta en sombras y almacenarlas en distintos puntos, si alguna base de datos o dispositivo de almacenamiento fuese comprometido, la imágen seguiría siendo secreta.

- Transmisión segura de imágen: si algún canal de transmisión a algún actor estuviese comprometido, el atacante podría acceder únicamente a la sombra transmitida por ese canal y así no tendría acceso a la imágen.

- Control de acceso: al requerir la reunión de múltiples sombras distribuidas a distintos actores autorizados, se elimina el acceso no autorizado a la imágen secreta a menos que se dispongan de las sombras necesarias para obtener el secreto. Además con la detección de cheating se obtiene accountability ya que el esquema detectará intentos de manipulación (tampering) o falsificación (cheating) de sombras. Esto último es muy útil por ejemplo contra atacantes internos a la organización.

- Alta disponibilidad: mientras exista una cantidad determinada de actores que presenten su respectiva sombra, la imágen secreta seguirá siendo recuperable, incluso si ocurriese que un grupo de actores no están disponibles (ej: se les cayo el servidor) o incluso si se les daño/corrompió/perdió su sombra.

- Digital Rights Management: relacionado con el control de acceso, al dividir la imágen en sombras se puede proteger el copyright de la misma, ya que al distribuir sombras de la imágen, los actores no podrán realizar copiado y distribución no autorizada de la imágen o modificaciones de la misma.

- Multi-factor Authentication / Authentication: donde la sombra representaría algo que tengo, podría combinarse con algo que soy, como por ejemplo para acceder a un datacenter crítico de una empresa donde ningún empleado debería poder acceder sin supervisión de otros. Se podría requerir el escaneo de retina, junto a la presentación de una sombra, y a su vez se requerirá de otros actores con sus respectivas sombras para formar el secreto y acceder.

- Recuperación de datos: con este esquema también se podría realizar una recuperación de la imágen si alguna de las partes pierde la misma y desea recuperarla. Solo basta utilizar su sombra con la de los otros actores requeridos para recuperar la información pérdida.