Criptografía y Seguridad

Trabajo práctico N°2

Secreto Compartido en Imágenes con Esteganografía

Autores:
Pablo Ballesty - 49359
Nicolás Magni - 48008

11 de junio de 2012

Índice

Resumen

El objetivo del presente informe es detallar las deciones tomadas durante el diseño y la implementación del Trabajo Práctico. Así también, se presentan los resultados obtenidos a partir de las imágenes que envió la cátedra y se realiza un análisis de los mismos. Se detallan las dificultades encontradas, y las soluciones propuestas. También se presentan conclusiones y posibles extensiones del algoritmo.

1. Introducción

El Trabajo Práctico consiste en la implementación de un algoritmo de secreto compartido, aplicado a imágenes. Los algoritmos de secreto compartido, consisten en ocultar información sensible, la cual llamaremos *el secreto*, utilizando n claves, de las cuales, para la recuperación del secreto, serán necesarias k (con $k \leq n$).

Como en este caso, las claves son imágenes, al momento de generar las mismas, se utiliza una técnica de *esteganografía* con el fin de que el ojo humano no note que se trata de claves.

El algoritmo que se implementa en el trabajo se extrajo de la publicación ***"Improvements in Geometry-Based Secret Image Sharing Approach with Steganography"*** realizada por **Mustafá Ulutas**, **Vasif V. Nabiyev** y **Guzin Ulutas**, de la Universidad de Karadeniz, Turquía.

2. Desarrollo

El algoritmo no resulta difícil de implementar, sin embargo, require de tener especial cuidado en ciertos aspectos, para obtener resultados exitosos. A continuación, se detallan las dificultades que presenta el algoritmo, y como éstas fueron resueltas en la práctica.

2.1. Algoritmo de distribución

El algoritmo de distribución consiste, en dadas n imágenes (las cuales llamamos *imágenes cover*), una imágen secreta S, y un número k. Generar n imágenes clave (las cuales llamamos *imágenes stego*), de forma tal que con k de las n generadas, pueda reconstruirse la imagen secreta especificada.

La estrategia que utiliza el algoritmo de distribución, explicada en forma general, consiste en:

- **Paso 1**: Particionar la imagen secreta S y las *imágenes cover* en bloques de k píxels. Para cada bloque de la imagen S repetir los pasos siguientes.
- **Paso 2**: Se considera a los k valores del bloque de la imagen S, como un punto en un espacio k-dimensional.
- **Paso 3**: Utilizandos los k valores del bloque de cada *imagen cover* se genera la ecuación de un hiperplano, siguiendo los pasos subyacentes.
- **Paso 4**: Se obtienen para el bloque seleccionado de cada *imagen cover* los valores $(a_1, a_2, \ldots, a_k)$. Estos valores corresponden a los bits más significativos de cada píxel, que quedan luego de quitar el espacio donde se va a almacenar B y el bit de autenticación. [1]
- **Paso 5**: Se calcula el valor de B, utilizando la ecuación 1 evaluando el punto del Paso 2. Y se embebe este en los espacios que se dejaron para el mismo en el Paso 4.
- **Paso 6**: Se calcula el bit de autenticación p, y se lo coloca en el lugar que se dejó para el mismo en el Paso 4.

$$(a_1x_1 + a_2x_2 + \cdots + a_kx_k) \bmod 251 \equiv b \tag{1}$$

Luego de llevar a cabo este algoritmo, si se toman k de las n *imágenes cover*, puede armarse el siguiente sistema:

$$\begin{cases} (a_{1,1}x_1 + a_{2,1}x_2 + \cdots + a_{k,1}x_k) \bmod 251 & \equiv B_1 \\ (a_{1,2}x_1 + a_{2,2}x_2 + \cdots + a_{k,2}x_k) \bmod 251 & \equiv B_2 \\ & \vdots \\ (a_{1,k}x_1 + a_{2,k}x_2 + \cdots + a_{k,k}x_k) \bmod 251 & \equiv B_k \end{cases} \tag{2}$$

Como al momento de generar cada ecuación, se calculó cada B_i de forma tal, que el punto de la imagen secreta S sea solución. Al resolver el sistema presentado en 2, se va a obtener el punto de la imagen secreta S. Si esto se realiza, para cada bloque de las *imágenes cover*, se recuperaría la imagen secreta.

Sin embargo, existen algunos casos a tener en cuenta, donde pueden generarse algunos problemas, que ocasionan que el punto de la imagen secreta no pueda ser recuperado. En las secciones 2.1.1 y 2.1.2, se presentan estos problemas y las estrategias utilizadas para evitarlos.

[1] Se utilizó el esquema descripto en el enunciado. Es decir que el bit de autenticación p, queda en el primer píxel del bloque

2.1.1. Sistemas compatibles determinados

Uno de los problemas que presenta la estrategia descripta al comenzar esta sección, es que, pueden ocurrir casos en los cuales el sistema generado, no sea un Sistema Compatible Determinado (**SCD**). Si no se contemplaran estos casos, existirían posibilidades de generar sistemas sin solución, lo cual, concluiría en la imposibilidad de recuperar el punto de la imagen secreta, en la posición de bloque dada.

La estrategia que se tomó, para resolver este problema consistió en, generar el sistema dados los valores originales de los bloques de las *imágenes cover*, y siempre probar si el sistema es **SCD**, es decir, que se obtiene una solución. En caso de que el sistema no tenga solución, se elige de forma aleatoria una de las *imágenes cover* y también de forma aleatoria uno de los k píxels del bloque y se lo incrementa un valor *inc*, el cual se calcula utilizando la ecuación 3.

$$inc = \begin{cases} 2^{pos} & \text{si la suma sobre el píxel no va a producir overflow} \\ 200 & \text{si la suma de } 2^{pos} \text{ sobre el pixel produce overflow.} \end{cases} \quad (3)$$

El valor de *pos* corresponde al número de posiciones que deben dejarse en el píxel, para embeber el valor de B y el bit de autenticación p si fuere el caso. De esta forma, se busca incrementar los bits que van a corresponder a algún valor de a_i en la ecuación correspondiente al bloque. El propósito de incrementar en un valor grande el píxel, en caso de que se vaya a generar un overflow con el valor de incremento anterior, es que no haya saltos de colores muy pronunciados en la *imagen stego* que se genere como resultado. De esta forma, se logra que si se va a producir un overflow, el color correspondiente al píxel siga siendo similar al original, y no se pase del blanco al negro, lo cual sería muy notorio.

2.1.2. Validación de la distribución

Otro de los problemas que no se consideran en la estrategia descripta al comenzar esta sección, y relacionado con el problema presentado anteriormente. Es que no solo se debe verificar la compatibilidad del sistema compuesto por las n ecuaciones que se derivan de las *imágenes cover*. Sino que deben verificarse todos los sistemas conformados por k ecuaciones de las n generadas.

De esta forma deben generarse los $\binom{n}{k}$ sistemas, y probar que cada uno de estos corresponde a un **SCD**.

Este problema se resolvió generando todos los posibles sistemas, y si alguno no resulta un **SCD**, entonces se realiza el arreglo descripto en la sección anterior, pero solo se modifican las *imagenes cover* que incurrieron en el sistema generado. Sin embargo, si se realiza alguna corrección es necesario volver a probar la compatibilidad de todos los sistemas posibles nuevamente, puesto que el cambio de una *imagen cover*, puede haber afectado a sistemas que anteriormente no presentaban problemas de compatibilidad.

Vale la pena notar que, por ejemplo, en un esquema de $(k, n) = (4, 8)$, la cantidad de posibles combinaciones de ecuaciones resulta $\binom{8}{4} = \frac{8!}{(8-4)! * 4!} = 70$. Y que si en alguna de las mismas, es necesario realizar algún arreglo, es necesario probar todas las combinaciones nuevamente. Por lo tanto, la verificación de la distribución es uno de los procesos más costosos del algoritmo.

2.2. Algoritmo de reconstrucción

El algoritmo de reconstrucción es fácilmente implementable, y si se desarrolló el de distribución siguiendo una arquitectura medianamente modular, debe consistir de la reutilización de los métodos utilizados para comprobar si los sitemas que se generaban en la distribución eran compatibles.

El algoritmo consiste en dadas n *imágenes stego* siendo $n >= k$. Seguir los siguientes pasos

- **Paso 1**: Se toman las *imágenes stego* recibidas. [2]
- **Paso 2**: Se particiona cada *imagen stego* en bloques de k píxels.
- **Paso 3**: Se extraen los grupos de k pixels de cada *imagen cover*. Y se realizan para cada uno, los Pasos 4,5 y 6.
- **Paso 4**: Se extrae el bit de autenticación p, de la posición correspondiente del bloque.
- **Paso 5**: Se calcula el bit de autenticación, utilizando el bloque con la posición de p en 0. Si el bit que se obtiene es distinto al extraído en el Paso 4, se avisará al usuario que la imagen que se recupera, pudo haber sido manipulada. [3]
- **Paso 6**: Se extraen los valores $(a_1, a_2, \ldots, a_k)$ y B del bloque.

[2] El *paper* propone tomar k imágenes de las n dadas. Por facilidad de implementación, puesto que el resultado es el mismo, se toman todas las imágenes recibidas.

[3] En la implementación, el mensaje consiste en la cantidad de bloques que fallaron en la autenticación.

- **Paso 7**: Se genera el sistema de ecuaciones, utilizando los datos que se extrageron en el Paso 6. Y se resuelve el sistema, para obtener el bloque de k píxels secreto. Utilizando el método que se explica en 2.2.1.

En las sección 2.2.1, se presenta el método utilizado para la resolución de los sistemas de ecuaciones, el cual también es utilizado en el algoritmo de distribución para validar **SCD**.

2.2.1. Método de solución de sistemas de ecuaciones en aritmética modular

El método que se utiliza para la resolución de los sitemas de ecuaciones en ambos algoritmos, el de distribución y el de recuperación, es el método de **Gauss-Jordan**, con unas modificaciones para trabajar en aritmética modular.

El método recibe la matriz del Paso 7, del algoritmo de recuperación explicado al comenzar esta sección. Donde cada fila, esta conformada por los valores que se extrajeron del Paso 4, para cada *imagen stego*, de la siguiente manera:

a_1	a_2	...	a_k	B

El mismo utiliza las siguientes 3 rutinas:

- **mulRow(*row*, *number*)**: Multiplica todos los valores de la fila *row* por *number* en módulo 251.
- **subRowNTimes(*row1*, *row2*, *ntimes*)**: Resta a la fila *row1* la fila *row2 ntimes* veces en módulo 251.
- **switchRows(*row1*, *row2*)**: Intercambia las posiciones de las filas *row1* y *row2*.

El método consiste en realizar sobre la matriz, los siguientes pasos:

- **Paso 1 (ReacomodateMatrix)**: Realiza *switchRows* de forma tal de dejar en la diagonal de la matriz, todos números mayores a 0. Si esto no se logra, el sistema ya no tiene solución única y se termina.
- **Paso 2**: Para cada número en la diagonal de la matriz se realizan los Pasos 3, 4 y 5.
- **Paso 3**: Se toma el número de la diagonal al cual llamamos x, se busca el número c, tal que $x * c \bmod 251 \equiv 1$.[4]
- **Paso 4:** Se realiza *mulRow* de la fila en la que se encuentra x por c.
- **Paso 5:** Para todas aquellas filas que contengan valores distintos de 0, en la columna de x. Se realiza *subRowNTimes* de la fila correspondiente por la fila de x, la cantidad de veces del valor encontrado.
- **Paso 6:** En este paso la matriz debería haber quedado diagonalizada, y con todos sus valores en 1. Se recorre la diagonal, verificando que los valores sean 1, en caso de ser verificado, se devuelve la solución que corresponde a los valores en la columna de B, en otro caso el sistema no posee solución única.

Cabe destacar, que como el método de resolución trabaja siempre en módulo 251, el resultado que se obtiene no va a ser exactamente el mismo que el secreto provisto al momento de la distribución.

2.3. Comparación con el algoritmo original de Shamir

El algoritmo propuesto en el *paper* se basa en la utilización de esteganografía junto con el método de Blackley, resulta interesante nombras las ventajas que posee el mismo contra el método original propuesto por Shamir.

- En el método de Shamir las *imágenes cover* deben ser cuatro veces más grandes que la imagen secreta. Por lo tanto, de esto se desprenden dos grandes desventajas, se requerirá más ancho de banda en caso de que estas sean enviadas por la red, y requerirá más espacio en disco para alojar las imágenes.
- Otra de las desventajas que posee el método original de Shamir es que produce imágenes ruidosas.

3. Lectura del *Paper*

Durante la lectura del *paper* en el que se basa el Trabajo Práctico se notaron diferentes características del mismo, las cuales se destacan a continuación.

- **Organización formal del documento**:
  - El Abstract o Resumen del documento no posee título y tiene el mismo formato que los datos de Contacto y Copyright. A simple vista, parece ser parte de la información de Contacto y resulta fácil saltearlo para el lector.
  - No hay una división jerarquizada de las secciones. Se encuentran al mismo nivel Introducción, el Método de Blackley's, el Método propuesto y los Resultados.
- **Notación utilizada**
  - A lo largo de la lectura, cada vez que se hace referencia a una imagen, no queda muy claro a cual lo hace, producto de llamarlas de diferentes modos. En la sección de recuperación, llama a la misma imagen como *shared* y como *stego*.

[4] Al comenzar se crea un vector con todos los números c, con el objetivo de encontrar el mismo de forma eficiente.

- En las explicaciones hace uso de índices y posiciones, sin aclarar cual es el primer elemento o el último. Por lo tanto, no se entiende de qué forma toma algunos valores.
- La explicación de los algoritmos mediante pasos, donde un paso hace un *for-each* sobre los pasos subyacentes, no es la mejor para el caso. Hubiese estado mejor, la presentación de *pseudocódigo*.
- Para el cálculo de algunos valores presenta ecuaciones muy complejas, que luego de analizarlas son cálculos simples que podrían haberse expresado de una forma más clara (Ecuaciones 3.2, 3.3 y 3.4).

- **Varios**
  - Se dan ejemplos, pero no muestra los cálculos intermedios, ni realiza todos los cálculos del ejemplo. Por lo tanto, el mismo es de escasa utilidad.

4. Posibles extensiones

A continuación se presentan, dos posibles extensiones del algoritmo implementado. La primera se refiere a la utilización del mismo en imágenes a color, y en la segunda se propone una forma diference de conformar los bloques que utiliza el algoritmo.

4.1. Imágenes en color

Luego de implementar el algoritmo se notó, que el mismo puede ser fácilmente utilizado para imágenes a color. Aprovechando que las imágenes se representan generalmente mediante **RGB**, que consiste en guardar para cada píxel la intensidad del color Rojo, Verbe y Azul. Por lo tanto, puede pensarse a cada canal de la imagen, como una imagen en escala de grises, como las que se utilizaron en el trabajo, y aplicar el algoritmo a cada canal por separado, para luego unir los resultados.

4.2. Selección de los bloques

Durante la implementación de la solución del problema de la compatibilidad de los sistemas, se notó que muchas veces los k coeficientes de la ecuación correspondiente a un bloque eran iguales. Esto no resulta extraño, ya que en una imagen es muy probable que dos píxels adyacentes tengan la misma intensidad de color, o intensidad similiar, que termina siendo igual al quitar los primeros bits del píxel. Se nos ocurre que resultaría interesante cambiar la conformación de los bloques, y en vez de tomar k píxels adyances, tomar píxels de diferentes partes de la imagen, obviamente de forma consistente. Se nos ocurre que de esta forma, se reducirían gran cantidad de sistemas incompatibles, y no sería tan necesaria la modificacion de las *imágenes cover*.

5. Posibles aplicaciones

5.1. Envío de imágenes sensibles a través de redes no confiables

Una de las aplicaciones que se nos ocurre, es en un escenario donde uno desea enviar una imagen sensible (ejemplo: mapa del tesoro) a través de diferentes redes no confiables, se podría distribuir la imagen en n imágenes, siendo n la cantidad de redes no confiables a utilizar, y enviar cada una por una red diferente al *end-point* deseado. El *end-point* una vez que recibe las tres podrá reconstruir la imagen sensible. Mientras que aquellos que hayan robado alguna de las imágenes de la red, no podrán reconstruir el secreto, y mejor aún, creerán que lo han conseguido.

5.2. Persistencia de imágenes en filesystem público

Otra aplicación posible, pero esta vez solo explotando la bondad de la esteganografía, si uno quisiese guardar una imagen en un filesystem público, y no quiere que los vecinos puedan ver a simple vista la imagen. Podría guardar las *imágenes stego* necesarios para obtenerla en el filesystem. Sin embargo, en esta situación, si un atacante prueba recuperar secretos en base a imágenes públicas. Descubrirá nuestro secreto.

6. Resultados

En esta sección se presentan los resultados obtenidos a partir de las imágenes enviadas por la cátedra. [5]

6.1. 2 sombras

Se obtuvo la imagen sin ningún problema de compatibilidad en los sistemas, se obtuvieron 113875 errores de autenticación. Las imágenes y el resultado obtenido se muestran en la Figura 2.

[5] No se pudieron encontrar errores en el cálculo del bit de autenticación ni la verificación de SCD. Utilizando imágenes distribuidas y recuperadas con el programa, no se obtienen errores.

6.2. 3 sombras

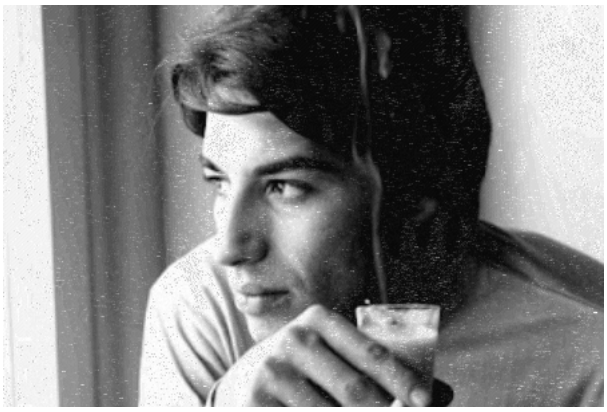
Se obtuvo la imagen con 397 problemas de compatibilidad en los sistemas, se obtuvieron 86112 errores de autenticación. Las imágenes y el resultado obtenido se muestran en la Figura 1.

6.3. 4 sombras

Se obtuvo la imagen sin ningún problema de compatibilidad en los sistemas, se obtuvieron 56527 errores de autenticación. Las imágenes y el resultado obtenido se muestran en la Figura 3.

A. Figuras

(a) Jim3.bmp



(b) Roberto3.bmp

(c) Whitney3.bmp

(d) Shakira3.bmp

(e) Recuperación obtenida

Figura 1: Recuperación con 3 sombras

(a) Carlitos2.bmp

(b) Gandhi2.bmp

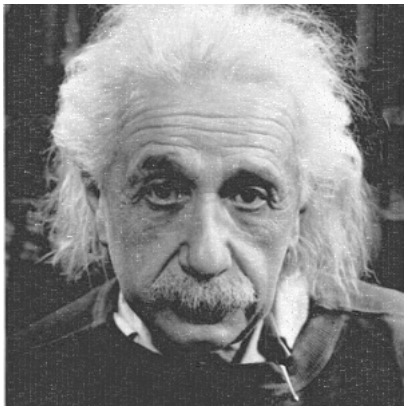
(c) Gandhi2.bmp

(d) Recuperación obtenida

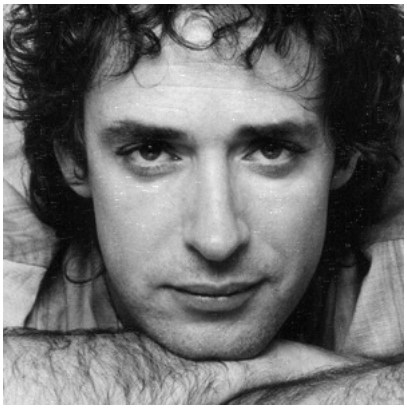
Figura 2: Recuperación con 2 sombras

(a) James4.bmp

(b) Eva4.bmp



(c) Albert4.bmp



(d) Gustavo4.bmp

(e) Marilyn4.bmp

(f) Recuperación obtenida

Figura 3: Recuperación con 4 sombras