Trabajo Práctico Especial

Secreto Compartido en Imágenes con Esteganografía

Criptografía y Seguridad

Integrantes

Nombre	***Correo***	***Legajo***
Ritorto, Bianca	britorto@itba.edu.ar	57082
Santoflaminio, Alejandro	asantoflaminio@itba.edu.ar	57042
Varela, Catalina	cavarela@itba.edu.ar	56433

Índice

Introducción

Se realizó una implementación en lenguaje C del algoritmo de Secreto Compartido en Imágenes para la encriptación y desencriptación de imágenes con marcas de agua y esteganografía.

En este documento analizaremos la teoría en la que nos basamos para la realización del proyecto, los recaudos que se tomaron llevando a restricciones, la utilidad de la implementación de estos algoritmos, así como el análisis de diversas cuestiones relacionadas al trabajo.

Base teórica

Para la realización de este trabajo práctico se utilizó como base teórica el paper titulado *"Verifiable Image Secret Sharing Using Matrix Projection"* por Nurfathiya Faradiena Azzahra y Kiki Ariyanti Sugeng de Universitas Indonesia. En este paper se describe el algoritmo de encripción y desencripción de imágenes, así como su verificación, a través de la utilización de operaciones matriciales.

Si bien el documento se encuentra bien organizado, es difícil seguirlo por la cantidad de operaciones matemáticas y variables que hay, lo cual significó una complejización a la hora de realizar el trabajo. La notación utilizada no cambia a lo largo del documento. Como error notamos que en el algoritmo de recuperación el último paso de la página 3 es en verdad el último de la distribución y no debería hacerse en la recuperación.

Algoritmo de distribución

El objetivo del algoritmo es separar la imágen en n partes encriptadas verificables, teniendo en cuenta que sólo k de ellas podrán reconstruir la imagen original.

Descripción del algoritmo

1. Tomando el par (k, n) mencionado y la matriz S que representa los píxeles de la imágen a encriptar, se construye una matriz aleatoria llamada A de $m \times k$ y rango k, con m[1] > 2(k-1)-1.
2. Una vez hecho esto se calcula la proyección de la matriz A como $\mathbb{S} = proj(A) \pmod p$ y se calcula una matriz remanente $R = (S - \mathbb{S}) \pmod p$, siendo p un número primo.
3. Luego, se eligen n vectores aleatorios x_j de $k \times 1$ y n coeficientes c_j, con $1 \leq j \leq n$.
4. Se calcula el vector $v_j = (A \times xj) \pmod p$ para $1 \leq j \leq n$.
5. Usando el algoritmo de Thien y Lin y la matriz R, se crean las sombras como $G_j = [g_1^{(j)} \ g_2^{(j)} \ \dots \ g_{ceil(m/k)}^{(j)}]$, siendo $g_t^{(j)}(i)$ el resultado del polinomio $g_t^{(j)}(i) = (I(i, k(t-1)+1)c_j^0 + \dots + I(i, k(t-1)+k)c_j^{k-1}) \pmod{251}$, donde $1 \leq t \leq ceil(m/k)$ y $1 \leq i \leq m$.

[1] En la implementación tomamos siempre m = n.

6. Se selecciona una imágen como watermark W y se computa la matriz remanente proveniente de la watermark como R_w= (W - $\mathbb{S}$) (mod p).
7. Por último, se divide las sombras de las imágenes Sh_j = [v_jG_j] entre los participantes y hacer la matriz R_w pública.

Algoritmo de recuperación

El objetivo del algoritmo es poder recuperar a partir de lo encriptado en el punto anterior, una imagen de tamaño n utilizando sólo k sombras, verificando que estas sombras sean válidas.

Descripción del algoritmo

1. Una vez elegidas las k imágenes a utilizar de las n para recuperar la imagen original, crear la matriz B = [v_1 v_2 … v_k], siendo vj la primer columna de Sh_j.
2. Computar la matriz proyección $\mathbb{S}$ = (proj(B)) (mod p).
3. Construir la matriz remanente R usando el algoritmo de Thien y Lin, y usando G_j (obtenido como la matriz Sh_j sin la primer columna). R será entonces de la siguiente manera:

$$R = \begin{bmatrix} r_1^{(1)} & r_2^{(1)} & \cdots & r_y^{(1)} \\ r_1^{(2)} & r_2^{(2)} & \cdots & r_y^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ r_1^{(x)} & r_2^{(x)} & \cdots & r_y^{(x)} \end{bmatrix}$$

   Con r_y (x) = [I(x, k(y - 1) + 1) I(x, k(y - 1) + 2)], donde ≤ x ≤ m, y 1 ≤ y ≤ ceil(m/k).
4. Calcular la imagen secreta como S = ($\mathbb{S}$ + R) (mod p).
5. Verificar que las imágenes usadas para la recuperación sean correctas recuperando la imagen usada como watermark W = ($\mathbb{S}$ + R_w) (mod p).

Aclaración: ¿Por qué siempre hay que indicar n, aún al recuperar?

Es necesario indicar n, incluso al recuperar, porque el valor de n se utiliza para la obtención de las matrices S de la imagen secreta, ya que al saber que la watermark tiene el mismo tamaño que la secreta, podemos teniendo n saber la cantidad de matrices S que tenemos. Además nos permite obtener el tamaño de las sombras y por lo tanto se utiliza para determinar la dimensión de diversas matrices que se utilizan en las operaciones durante la recuperación.

Mejora por Azzahra y Sugeng

Este paper busca mejorar el método propuesto por Li Bai en *“An Image Secret Sharing Method”* al agregar la posibilidad de verificar si las imágenes utilizadas para la recuperación son efectivamente parte de las n generadas o si fueron alteradas. Esto se realiza agregando la watermark, con la matriz remanente de la watermark conocida. Si alguna de las k imágenes hubiese sido alterada, se obtendría como imagen watermark ruido.

Generación de las matrices X_i

Las matrices X_i se generan en la distribución de manera aleatoria (aunque respetando lo establecido en la consigna para obtener independencia lineal). Estas matrices son utilizadas para luego generar las matrices V que se corresponden a la primer columna de cada sombra.

En la recuperación se utilizan para justamente obtener la proyección de las matrices S.
La forma de generar las matrices X es válida, justamente porque se generan de tal forma q son linealmente independientes. Esto es porque cada vez que vamos a generar las matrices X establecemos un valor 'a' aleatoriamente (perteneciente a los enteros menores a 251) y cada valor de la matriz X (que tiene una sola columna) lo vamos completando de la forma: $a^0, a^1, \ldots, a^{k-1}$. Es importante asegurarse que cada una de nuestras n matrices X se genere con un valor de 'a' distinto.

Transformación de la marca de agua

Al distribuir generamos una transformación de la watermark, que se visualiza como una imagen puramente ruidosa[2]. Esta imagen no tiene aplicada esteganografía. Tal y como indica el paper esta imagen contiene la información correspondiente a las matrices Rw y que son de dominio público, por lo que no se ocultan.

Si fuera deseable ocultarla con esteganografía, al ser del mismo tamaño que la marca de agua (123200) se requeriría de portadoras de tamaño más grande que el de las provistas (si fuera a hacerse con LSB1 o LSB2) ya que no contienen suficiente espacio para guardar los datos esteganografiados.

De todas formas consideramos que para la implementación del trabajo y justamente por el propósito que tiene la generación de la transformación de la watermark, la idea es no aplicarle ningún tipo de ocultamiento porque su propósito es únicamente verificar la legitimidad de la imagen secreta.

Restricciones

Elección de pares (k,n)

En el trabajo implementado y , como se indicó en la consigna, tan solo se trabaja con los pares de k = 2 con n = 4 y de k = 4 con n = 8. Esto permite para las imágenes provistas de 280x440 de 8 bits por pixel, utilizar imágenes portadoras del mismo tamaño con 24 bits por

[2] En nuestro trabajo en particular se genera con el nombre generated_watermark.bmp dado que en la consigna no se indicaba que el usuario debe indicar un path para la misma. En la recuperación se genera un archivo llamado recovered_watermark.bmp.

pixel. Esto es porque se usa LSB1 para el esquema (4,8) y LSB2 para el esquema (2,4) y el tamaño de las portadoras se ajusta perfectamente para guardar las sombras generadas mediante ambos esquemas con su respectivo sistema de esteganografía.

Al elegir pares (k,n) distintos de los establecidos, en este trabajo en particular sucedería que por los tamaños y características de las imágenes a ocultar (tenemos 123200 bytes en las imágenes secreto) perderiamos datos, dado que generamos varias matrices S a partir de esta imagen.

Ahora bien 123200 descompuesto en números primos es $2^6 \times 5^2 \times 7 \times 11$, por lo tanto tenemos que elegir un n cuyo cuadrado este entre esos números, es decir que divida a 123200. Por ello 2 y 4 son valores válidos para n. Podríamos tomar otros valores que cumplan este requisito para n si tuviéramos portadoras de diferente tamaño.

Teniendo en cuenta estos valores de n, no es posible tomar otro valor para k. Es decir para n siendo igual a 4 debemos tomar a k como 2, esto porque debemos armar nuestro sistema de ecuaciones en la recuperación para armar la matriz R y necesitamos de al menos 2 sombras para hacerlo y resolver el sistema. Lo mismo sucede para n siendo 8, vamos a necesitar 4 sombras para resolver todas las variables que van a ser los valores pertenecientes a la matriz R.

Control de rangos

Las matrices A generadas durante el algoritmo deben cumplir el requisito de ser de rango k. Además controlamos que el resultado de su transpuesta multiplicado por A también sea de este mismo rango. Esto para asegurar de que esta multiplicación tenga inversa y no tener errores durante el cálculo de la proyección de S.

También con este mismo fin, en las matrices X, se las genero de tal manera que siempre sean linealmente independientes entre sí.

Guardado de sombras

En el programa, las sombras se guardan en las imágenes portadoras. Lo que se hace es para cada una de las sombras generadas en cada matriz S de la distribución generamos una cantidad n de sombras. Al tener n imágenes portadoras, guardamos en la imagen número ‘p’ todas las sombras ‘p’ de forma concatenada (aplicando esteganografía una vez concatenadas todas). Es decir, se generan en la distribución por cada matriz S, una cantidad n de sombras con números del 0 al 7. Así elegimos una portadora donde siempre vamos a guardar los datos correspondientes a un mismo número de sombra.

Número de sombra

Las imágenes utilizadas para compartir las sombras son imágenes de extensión bmp. Este tipo de imágenes contiene headers que pueden utilizarse para transmitir información. Gracias a esto, podemos guardar en el header en la variable reserved1 el número de

sombra generada. Es decir en una portadora, indicamos en el valor reserved1 el valor 0 si allí estamos guardando todas las sombras 0 que se obtienen en el algoritmo de distribución. Consideramos que otro lugar donde podría guardarse dicha sombra es dentro del header en la variable reserved2. Pensamos como opción guardarlo en el nombre de archivo pero sería bastante inseguro y además al cambiar el nombre del mismo se perdería.

Implementación

Si bien la implementación del trabajo práctico se hizo en base al paper proporcionado, sufrimos varias complicaciones a la hora de hacerlo.

En primera instancia, fue difícil hacer una correspondencia entre el paper y la consigna porque ciertas cosas no eran del todo claras. No sabíamos que había que hacer varias matrices S y no era del todo claro el cómo guardar las sombras en las portadoras. Además, el hecho de no tener una referencia sobre esteganografía del estilo del paper *"Verifiable Image Secret Sharing Using Matrix Projection"*, hizo que implementarlo tuviese un grado extra de complejidad.

Para ello se tuvo que tener en cuenta el tamaño de las imágenes BMP así como sus bits por píxel para poder calcular dinámicamente la cantidad de matrices S que vamos a tener que generar en el algoritmo y luego generar las sombras y almacenarlas de forma concatenada como se mencionó anteriormente.

Para la recuperación se procede a leer de las portadoras también de forma dinámica sabiendo su tamaño y realizando las operaciones correspondientes.

Con respecto a la esteganografía, tal y como se indica en la consigna se implementó LSB1 para el caso del esquema 4,8 y LSB2 para el caso del esquema 2,4. Se verificó siempre de hacer los chequeos correspondientes con respecto al tamaño necesario de las portadoras así como para calcular el tamaño de los datos ocultos al recuperar. La esteganografía se aplicó en el caso de la distribución al final, cuando ya se tenían todas las sombras de un mismo número concatenadas y en el caso de la recuperación al principio para obtener nuevamente las sombras concatenadas que corresponden a un mismo número.

Posibilidad de extender el algoritmo o modificarlo

Existe la posibilidad de extender el algoritmo o modificarlo usando otros métodos de esteganografía. Sin embargo, si se quisiese hacer esto, sería necesario contar con imágenes portadoras de otros tamaños.

Aplicaciones

Existen diferentes usos y aplicaciones para este tipo de algoritmos con el objetivo de compartir imágenes de manera segura. A continuación listamos algunos:

- compartir imágenes sensibles o que no debieran ser vistas por cualquiera;
- enviar un mensaje encriptado a través de la imagen;

- poder tener control de autoría sobre imágenes, de manera de no sean plagiadas de ser interceptadas;
- tener la capacidad de poder verificar que la imagen no fue alterada por un tercero.

Conclusión

En conclusión, el trabajo permitió combinar varios aspectos que van desde el manejo de archivos BMP, ocultamiento de imágenes mediante operaciones matriciales y el uso de esteganografía.

Fue difícil combinar los distintos aspectos y realizar todas las operaciones en aritmética modular (en este caso con módulo 251) para el correcto manejo de los píxeles. Además requirió de investigación adicional, no sólo mediante la lectura de los papers provistos sino también para combinar lo anterior con la esteganografía.

A continuación se muestra el resultado de la ejecución del programa con los archivos provistos por la cátedra.

Resultados obtenidos con (2, 4)

Los resultados obtenidos con k = 2 y n = 4 fueron óptimos, con la imagen R_w generada siendo ruido (sin poder notar que la imágen watermark es la Torre Eiffel) y con la imagen original siendo recuperada satisfactoriamente sin ningún tipo de ruido. Se muestran la imagen Rw generada durante la distribución y las imágenes recuperadas.

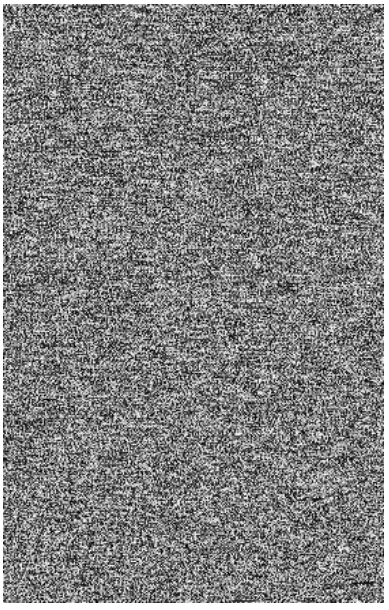

Fig. 1: Imagen R_w con (2, 4).

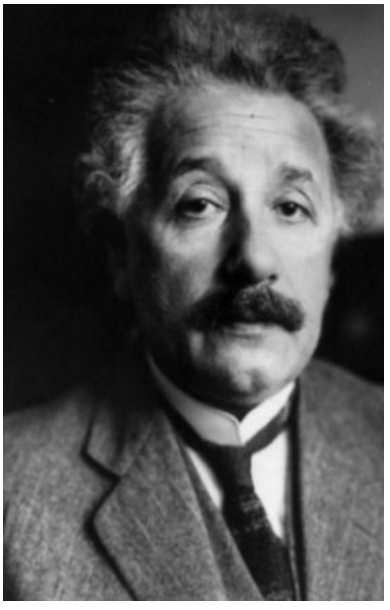

Fig. 2: Imagen recuperada de las imágenes con (2,4).

Fig. 3: Imagen watermark recuperada de las imágenes share con (2, 4).

Resultados obtenidos con (4, 8)

Los resultados obtenidos con k = 4 y n = 8 también fueron óptimos. Se muestran la imagen Rw generada durante la distribución y las imágenes recuperadas.

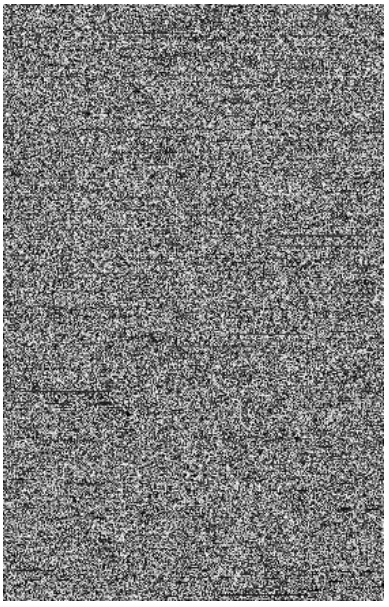

Fig. 4: Imagen R_w con (4, 8).

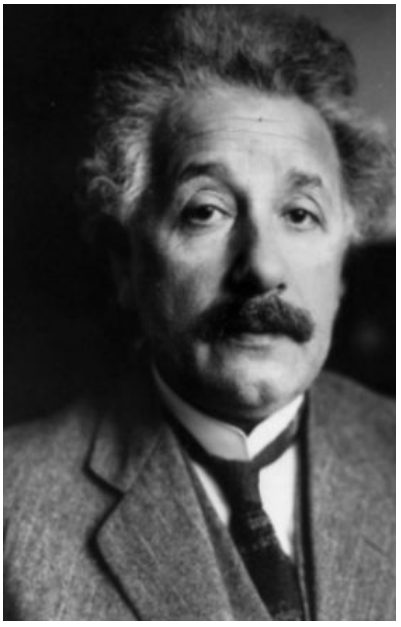

Fig. 5: Imagen recuperada de las imágenes share con (4, 8).

Fig. 6: Imagen watermark recuperada de las imágenes share con (4, 8).

Resultados obtenidos con (2, 4) para los archivos del grupo

Para los archivos provistos únicamente al grupo 3, los resultados fueron los siguientes.

Fig. 7: Imagen secreta recuperada (2,4)

Fig. 8: Imagen recuperada de la marca de agua (2,4)

Resultados obtenidos con (4, 8) para los archivos del grupo

Para los archivos provistos únicamente al grupo 3, los resultados fueron los siguientes.

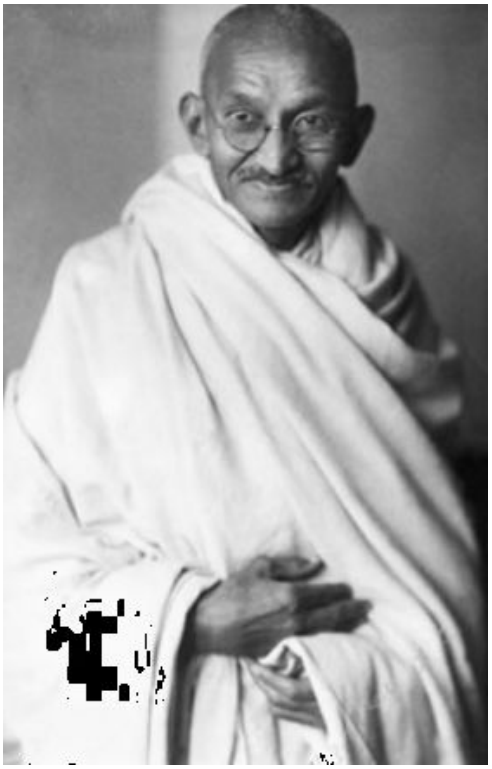

Fig. 9: Imagen secreta recuperada (4,8)

Fig. 10: Imagen recuperada de la marca de agua (4,8)