Criptografía y Seguridad

TRABAJO PRÁCTICO 2 - INFORME

SECRETO COMPARTIDO EN IMÁGENES CON ESTEGANOGRAFÍA

GRUPO 6 - 1C 2017

DOCENTES

- Abad, Pablo Eduardo
- Arias Roig, Ana María
- Ramele, Rodrigo Ezequiel

INTEGRANTES

- Comercio Vázquez, Matías Nicolás - 55309
- Ibars Ingman, Gonzalo - 54126
- Mercado, Matías - 55019

Cuestiones analizadas

A continuación se responden las cuestiones solicitadas en la consigna del trabajo, en el orden que las enumera:

1.

a.

La sección *Introduction* presenta, de manera breve, cómo fueron evolucionando los sistemas de distribución de imágenes secretas (SIS), basándose de los papers presentados por Blakley y Shamir. Si bien deciden partir del paper escrito por Thein-Lin por su simplicidad, no dan una explicación del por qué lo es más que el resto de los papers mencionados.

En la sección *Review* se realiza un resumen de los métodos de secreto compartido propuestos por Shamir, y por Thien-Lin, que salvo por el error mencionado en la sección **7.** respecto del paper de Shamir, es claro.

En la sección *The Proposed Method* se explica la implementación del algoritmo que incluye modificaciones del paper Thein-Lin, utilizando módulo 257 en vez de módulo 251 y cambiando el método de permutación por una técnica de búsqueda en una tabla.

En la sección *Experimental Results* se muestran ejemplos de uso de este algoritmo sin entrar en detalles respecto de los parámetros utilizados, del tipo de imagen o de cómo almacenan las sombras y las tablas. También se explica de manera genérica lo mismo que en la sección *Abstract* respecto de la probabilidad para un atacante de intentar encontrar la imagen original.

En la sección *Conclusions* se describen las propiedades del método propuesto sin alegar el por qué se cumplen; en los resultados debieran exponerse tablas que muestran, por ejemplo, cuanta menos pérdida de píxeles originales existe utilizando este método en vez del de Thien-Lin. También debieran exponer el porcentaje de reducción de tamaño de las imágenes sombra respecto del método de Thien-Lin.

En conclusión el paper no presenta resultados concretos para poder comprobar las conclusiones que alega.

b.

En la sección *The Proposed Method* puede existir un problema en caso de que, o bien por el polinomio original, o por una ejecución de los pasos 4 y 5 lleven a que $f_j(x) = x^{r-1}$. En este caso si existe alguna sombra igual a 256, se le restará un 1 al coeficiente a_{r-1} y el polinomio ya no será de grado r-1. El resto de los coeficientes

deben ser obligatoriamente 0 para que esto ocurra ya que el paso 5 del algoritmo pide encontrar el primer coeficiente diferente de 0 buscando de manera ascendente desde a_0. Si bien luego del algoritmo se contempla el caso en que todos los coeficientes sean igual a 0 y demuestra que este polinomio nunca resultará en un valor igual a 256, el problema reside en que si el polinomio es $f_j(x) = 0$, según el paper de Shamir, teóricamente se deberían necesitar 0 sombras para recuperar el secreto; la cuestión se encuentra en que el secreto no se guarda únicamente en el a_0 sino que todos los coeficientes son parte del secreto con lo cual a fines del algoritmo no interesa si el polinomio resultante es igual a 0.

c.

La notación utilizada es consistente y clara, en general, a lo largo de todo el documento. Existe un pequeño detalle en el último paso de la recuperación de la imagen secreta en la sección *The Proposed Method* en donde se menciona a *O'*, que hace referencia a la obtención de una imagen que puede ser diferente a la original (definido en la sección *Review* cuando se explica el algoritmo de Thien-Lin). Citando el paper: *"[...] Similar to Thien-Lin's scheme, the following steps can be used to reveal the secret image O [...]".* En este caso *O* debiera ser igual a *O'*.

2.

La imagen recuperada no necesariamente será la misma que la original. El problema yace en que las sombras son obtenidas con un polinomio utilizando aritmética modular con módulo 257[1] con lo cual, si bien el intervalo en el que se encuentra la escala de grises es [0 ; 255], al intentar obtener la sombra se puede obtener el valor 256, y el tratamiento especial a este valor implica cambiar un coeficiente en el polinomio original, que por construcción del mismo es un pixel de la imagen original.

En el caso en que ninguna sombra obtenida sea igual a 256, los polinomios originales se mantendrán, y se podrá recuperar la imagen original sin modificación alguna en sus píxeles.

En las las imágenes 1 y 2 se puede realizar una comparación entre la imagen original y la recuperada utilizando un esquema de secreto compartido (8,8). Si bien es posible observar una notable diferencia en algunos píxeles entre ambas imágenes, los píxeles de la imagen se mantuvieron intactos en su gran mayoría, a simple vista.

[1] Por ser un número primo, el conjunto de enteros que se forma módulo un número primo p es un campo en el que la interpolación es posible. Además, es el número primero más cercano a 256.

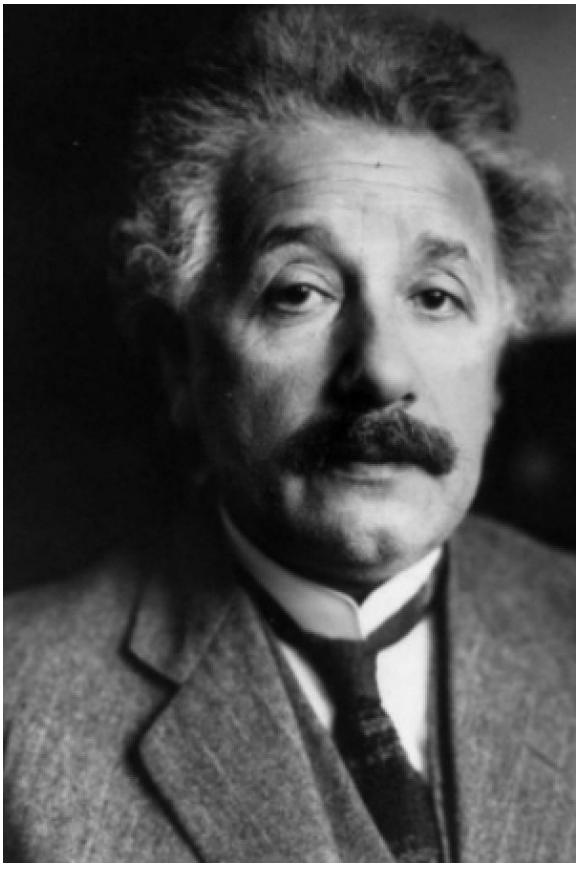

Imagen 1: Imagen original

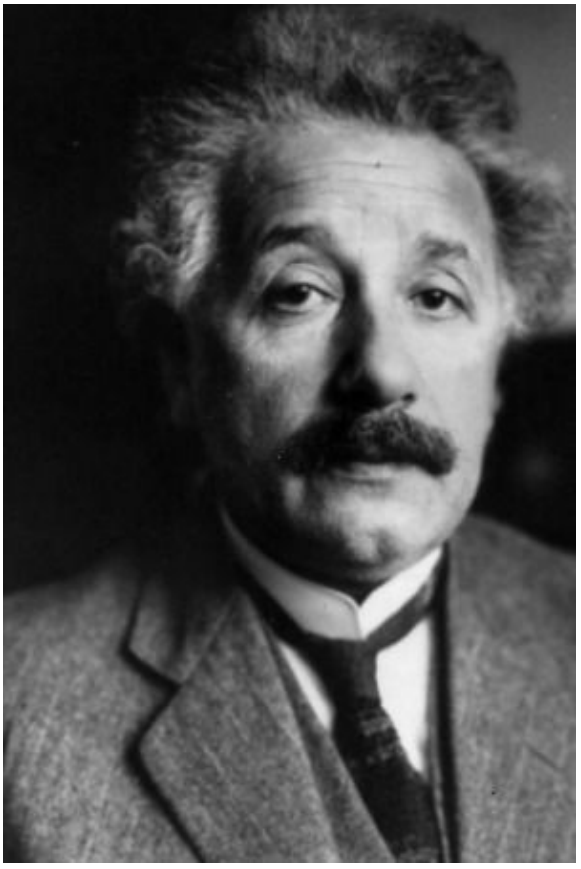

Imagen 2: Imagen recuperada

3.

El criterio utilizado para la elección de imágenes portadoras en el caso de que k fuera distinto de 8 es que k divida al tamaño total del archivo, en unidades de bytes, de la imagen a guardar; en caso de no cumplirse dicha condición se devuelve un error.

4.

Para ocultar las sombras se plantearon dos propuestas.

En la primera de ellas, a diferencia del caso en que k = 8, se oculta todo el secreto (header + datos) en las sombras especificadas. La única condición que tienen que cumplir las sombras en este caso es la mencionada en el punto anterior. Para recuperar todos los bytes de un archivo secreto distribuido entre k sombras, lo que se hace es recuperar (es decir, recolectar de las sombras los bytes ofuscados del secreto y desofuscarlos) la mínima cantidad de bytes necesarios (correspondientes al header del secreto guardado) para determinar con exactitud

cuál es la cantidad total de bytes que se deben recuperar del secreto. Una vez establecida esta cantidad, se siguen recolectando el resto de los bytes (todavía ofuscados) y finalmente se concatenan todos los bytes recolectados (los iniciales del header + los que se recolectaron a continuación), para desofuscarlos todos juntos, usando la semilla almacenada en cualquiera de las sombras (es en todas la misma). Esta opción provee la ventaja de poder almacenar cualquier secreto en cualquier otra imagen, independientemente de que se cumpla alguna relación de tamaños entre la imagen secreto y las sombras (con excepción de la mencionada en el punto anterior).

En la segunda propuesta se planteó que, en lugar de almacenar el header del secreto, el mismo pudiese deducirse de alguna relación entre los tamaños de la imagen secreta y las sombras, siguiendo un *approach* similar al caso con k = 8. La única ventaja que proveía esta alternativa con respecto a la primera era el hecho de no guardar los bytes de header. Por el contrario, agregaba diversas complejidades y restricciones, no sólo al algoritmo de distribución y recuperación, sino también a la limitación en cuanto a las imágenes a usar como sombras.

Es por todo lo recientemente mencionado que se optó por implementar la primera opción.

5.

a.

Respecto de la implementación del paper, no existieron problemas mayores. La dificultad del trabajo, en general, estuvo en la recuperación de los coeficientes a partir de las sombras leídas de las sombras.

Si bien la lectura y escritura de los valores en el header de la imagen implica tener cuidado en el orden en que se van guardando los bytes para formar el valor completo, no se presentaron inconvenientes. Respecto de la lectura y el almacenamiento de las sombras en el LSB (Least Significant Bit) hubo que tener más cuidado porque se manejan operaciones a nivel bit.

Para recuperar cada polinomio se decidió implementar (y no depender de librerías) el método de eliminación de Gauss con aritmética modular. La mayor dificultad en el trabajo residió en adaptar el método para que soporte operaciones modulares.

b.

Una posibilidad de extensión del algoritmo para utilizar imágenes de 24 bits por píxel es almacenar en el LSB de cada color del pixel, de la imagen sombra, un bit de la sombra que corresponde a ese color en la imagen original. Habría que analizar en este caso qué tanto puede afectar a los colores de las imágenes sombra cambiar los 3 bits menos significativos de los colores, es decir, qué tan notable es la diferencia entre los colores originales y los cambiados.

Además de los mencionados cambios, se deben adaptar los tipos primitivos con los que se trabajó en la implementación actual para soportar el nuevo tamaño de píxel (es decir, para soportar el cambio de 8 a 24 bits).

6.

En la sección *Abstract* cuando menciona que existen *(256 x 256)/r* cantidad de polinomios, no se explica el por qué. La razón de la misma se explica en la sección *Review*, en donde define lo que es una sección de procesamiento. Dada la imagen de 256 x 256 píxeles se divide por r, siendo r la cantidad de píxeles que se utilizarán para formar un polinomio de grado r-1; por lo tanto se obtendrán *(256 x 256)/r* polinomios.

En la sección *Review* cuando menciona los pasos del algoritmo de Thien-Lin, en el paso 3 pide colocar el número de sección de procesamiento representado por la variable j a 1 sin definir lo que es una sección. La definición de una sección de procesamiento se encuentra en el siguiente paso.

Si bien no es una dificultad en la lectura del documento, en la sección *Review* del paper existe un error también encontrado en el paper original de Shamir en el que se menciona que el coeficiente a_{r-1} pueden ser igual a 0; esto es incorrecto ya que en tal caso el polinomio no sería de grado r-1, con lo cual a_{r-1} debe pertenecer al intervalo de enteros (0 ; p-1]. El resto de los coeficientes sí pertenece al intervalo [0 ; p-1].

7.

Una posible extensión a este algoritmo puede ser el de almacenar los cambios que se realizaron al polinomio original en el caso de obtener como $f_j(x) = 256$ en el paso 5, es decir, almacenar el índice de los coeficientes cambiados y cuantas veces se le aplicó un cambio a cada uno de esos coeficientes. Para evitar tener que guardar por cada polinomio cuantas veces se les aplicó un cambio, se utilizaría una variable como header cuyo valor sea igual a la cantidad de índices que fueron cambiados, y luego leer las tuplas con la siguiente estructura: <índice, cantidad de cambios>. Si bien agrega overhead evita que al aplicar el algoritmo de recuperación se obtengan píxeles distintos a los de la imagen original.

8.

Este tipo de algoritmos se pueden aplicar en los siguientes casos de manera útil:

- Esconder datos confidenciales a través de un canal seguro o no seguro.

- Según el tipo de algoritmo utilizado, para realizar más difícil la tarea a un atacante de encontrar información escondida dentro de una imagen.

Existen alegaciones sobre el servicio de inteligencia exterior ruso (SVR RF) aplicando técnicas de esteganografía para poder realizar comunicación con agentes “ilegales” o que se encuentran sin cubrimiento diplomático localizados en el extranjero.

Secreto recuperado

Dado el conjunto de imágenes enviadas al Grupo 6 que representaban imágenes sombra en un esquema de secreto compartido (8,8), se recuperó la siguiente imagen:

Imagen 3: Imagen recuperada a partir del conjunto de sombras del Grupo 6

Las imágenes sombra pertenecen a la siguiente lista:

- Albertossd.bmp
- Audreyssd.bmp
- Carlitosssd.bmp

- Gandhissd.bmp
- Gracessd.bmp
- Jamesssd.bmp
- Johnssd.bmp
- Marilynssd.bmp

Ejemplos de ejecución

A continuación se exponen combinaciones de imágenes que cumplen con las restricciones mencionadas en la sección Cuestiones analizadas:

Ejemplo 1

> java -jar target/steganography.jar -d -secret clave.bmp -k 2 -n 4 -dir varias

- clave.bmp corresponde al archivo Alfred.bmp
- En el directorio varias se encuentran los siguientes archivos:
    - boats.bmp
    - brickwall.bmp
    - carpet.bmp
    - man.bmp

Para recuperar la imagen con el nombre "*clave.bmp*", ejecutar:

> java -jar target/steganography.jar -r -secret clave.bmp -k 2 -n 4 -dir varias

Ejemplo 2

> java -jar target/steganography.jar -d -secret clave.bmp -k 2

- clave.bmp corresponde al archivo baboon.bmp
- En el directorio actual se encuentran los siguientes archivos:
    - boats.bmp
    - man.bmp'

Para recuperar la imagen con el nombre "*secreto.bmp*", ejecutar:

> java -jar target/steganography.jar -r -secret secreto.bmp -k 2

Aclaraciones

Es posible que en el caso de que se intente recuperar un secreto con un esquema distinto al usado para su distribución el programa devuelva una excepción (mismo caso si se especifican sombras incorrectas). Si bien es de conocimiento del grupo de que esto ocurre, ***se entiende que el programa desarrollado es una herramienta*** a utilizar distribuir o recuperar secretos dentro de parámetros válidos, ***y que es responsabilidad del que usa dicha herramienta hacer un correcto uso de la misma.*** Es imposible (o demasiado complejo para el objetivo de este trabajo) hacer que el programa verifique o valide que los parámetros utilizados para la recuperación de secretos se correspondan con los usados al momento de su distribución.