Trabajo Práctico Especial

Alvarez Escalante, Tomás (60127)
tomalvarez@itba.edu.ar

Caeiro, Alejo (60692)
acaeiro@itba.edu.ar

Ferreiro, Lucas Agustín (61595)
lferreiro@itba.edu.ar

Gomez Kiss, Roman (61003)
romgomez@itba.edu.ar

21/06/2023

Implementación de '(k,n) secret image sharing scheme capable of cheating detection by Yan-Xiao Liu, Quin-Dong Sun and Ching-Nung Yang'

Grupo 15

72.44 Criptografía y Seguridad

Instituto Tecnológico de Buenos Aires

Tabla de contenidos

Introducción

El objetivo de este trabajo fue realizar un programa en lenguaje Java que implemente el algoritmo descrito en el documento "(k,n) secret image sharing scheme capable of cheating detection by Yan-Xiao Liu, Quin-Dong Sun and Ching-Nung Yang". Dicho algoritmo propone un esquema para compartir una imagen secreta basado en el método de Shamir y utilizando esteganografia para lograr que la imagen que se oculta en las sombras sea prácticamente imperceptible.

Guía de instalación y uso

Para compilar el programa se debe estar posicionado en la carpeta raíz del proyecto (*./TPE-SSS*) y correr:

```
./compile.sh
```

Tras esto se generará el archivo *ss.jar* en la carpeta raíz del proyecto. Por lo tanto, para correr el programa se deberá ejecutar:

```
java -jar ./ss.jar <modo> <imagenSecreta> <k> <directorio>
```

donde:

- **Modo:** tiene dos posibles valores:
  - **d:** indica que se va a distribuir una imagen secreta en otras imágenes.
  - **r:** indica que se va a recuperar una imagen secreta a partir de otras imágenes.
- **imagenSecreta:** corresponde al nombre de un archivo de extensión .bmp, y tiene dos usos según el modo elegido:
  - Con la opción (d) este archivo debe existir, ya que es la imagen a ocultar y debe ser una imagen en blanco y negro (8 bits por pixel).
  - Con la opción (r) será el nombre del archivo de salida, con la imagen secreta revelada.
- **k:** es la cantidad mínima de sombras necesarias para recuperar el secreto en un esquema (k, n), donde n es la cantidad de imágenes portadoras.
- **directorio:** path a un directorio que debe contener imágenes de extensión .bmp, de 8 bits por píxel, de igual tamaño que la imagen secreta. Además, deberá verificarse

que existan por lo menos k imágenes en el directorio. Este parámetro tiene dos posibles usos según el modo:

- Si se elige el modo (d), es el path al directorio donde se encuentran las imágenes en las que se distribuirá el secreto
- Si se elige el modo (r), es el path al directorio donde están las imágenes que contienen oculto el secreto.

Cuestiones a analizar

1.a Discutir la organización formal del documento/paper, ¿es adecuada o confusa?

La organización formal del documento es correcta, se cuenta con una introducción del tema a discutir, una sección de preliminares donde se comenta el método de Shamir, luego una explicación del método propuesto y por último la sección de resultados y conclusiones. El texto en dos columnas dificulta un poco el seguimiento, pero no se considera un problema grave.

1.b La descripción del algoritmo de distribución y la del algoritmo de recuperación en el paper, ¿es clara, confusa, detallada y/o completa?

La descripción del algoritmo propuesto en el paper es clara y completa gracias a que el documento cuenta con una sección de preliminares donde se explica el método de Shamir y una introducción al término de cheating detection en esquemas de secreto compartido, que facilitan luego la interpretación del propio algoritmo. Además, se agregan explícitamente los pasos a seguir para la distribución y recuperación de las imágenes, lo que permitió simplificar la compresión de este para su implementación.

Además, queremos mencionar que se encontró un error numérico al comienzo de la página 5 (**Sección 4: resultados y discusiones**):

"polynomials: $f_1(x) = 57 + 68x + 90x^2 + 231x^3$ and $g_1(x) = 161 + 104x + 42x^2 + 89x^3$, where $57 + 9 \times 161 = 0(mod 251)$, $68 + 9 \times 104 = 0(mod 251)$."

El error se encuentra en las cuentas $57 + 9 \times 161 = 0(mod 251)$ y $68 + 9 \times 104 = 0(mod 251)$, ya que estas deberían ser $57 \times 9 + 161 = 0(mod 251)$ y $68 \times 9 + 104 = 0(mod 251)$

1.c La notación utilizada en el paper, ¿es clara, cambia a lo largo del documento y/o hay algún error?

La notación utilizada a lo largo de todo el paper es clara y consistente. Sin embargo,

al contar con tantas variables, y en especial subíndices, a veces se dificulta la correcta compresión de que representa cada una de ellas.

2. El título del paper hace referencia a que es capaz de detectar sombras falsas (cheating detection), ¿cómo lo hace?, ¿es un método eficaz?

Como se comenta en la sección 3 del documento, la detección de sombras falsas se realiza mediante la construcción de un (k,n) threshold que extiende del esquema propuesto por Thien y Lin en la sección 2.3. La demostración de que dicho esquema detecta cheating se encuentra especificada en el paper, en la cual se especifica que como el atacante puede obtener los polinomios $f^{**}(x) = f^{*}(x) + f(x)$ y $g^{**}(x) = g^{*}(x) + g(x)$ en la etapa de recuperación, donde $f^{*}(x)$ y $g^{*}(x)$ son elegidos por el atacante, entonces puede elegir un número r^{*} que satisface $r^{*}a_0^{*} + b_0^{*} = 0$ y $r^{*}a_1^{*} + b_1^{*} = 0$. Entonces, según el algoritmo, si existe un r' que satisface $r'(a_0 + a_0^{*}) + b_0 + b_0^{*} = 0$ y $r'(a_1 + a_1^{*}) + b_1 + b_1^{*} = 0$ NO se detecta cheating, el cual solo ocurre cuando $r^{*} = r$ (con una probabilidad de $\frac{1}{251}$).

Es decir, se detecta cheating en las imágenes portadoras cuando en la fase de recuperación de la imagen secreta, al momento de calcular los coeficientes para rearmar los polinomios $f(x)$ y $g(x)$ con Lagrange para alguno de los bloques, no se encuentra un valor r_i que satisfaga simultáneamente $r_i * a_{i,0} + b_{i,0} = 0$ y $r_i * a_{i,1} + b_{i,1} = 0$.

Dado que todos los píxeles están en GF(251), y dada la posibilidad de éxito del atacante mencionada anteriormente, se puede concluir que el esquema es efectivo y eficaz para detectar cheating. Además, en la **Sección 5: conclusiones** se especifica una tabla donde se comprueba que el método para detectar cheating resulta más eficiente que otros esquemas (como el esquema de Harn, el de Pieprzyk o el de Sergio), pues es capaz de detectar cheating de hasta $k - 1$ atacantes unicamente con interpolaciones de Lagrange.

3. ¿Qué desventajas ofrece trabajar con congruencias módulo 251?

La desventaja de trabajar de este modo es que las operaciones terminan siendo más complejas computacionalmente y para optimizarlas se debe contar con una tabla estática con los inversos de los números en el campo de Galois 251. Además, GF(251) es un campo relativamente chico, lo que puede conllevar a que sea difícil trabajar con secretos largos. Otra desventaja de trabajar en GF(251) es que la seguridad de los secretos compartidos es vulnerable a ataques de fuerza bruta con una probabilidad de éxito del atacante de $\frac{1}{251}$.

Para poder implementar el algoritmo de cheating detection, se necesita que los coeficientes derivados de los píxeles sean reversibles para cuando se realiza la recuperación de la imagen, por ende, los píxeles con valor 0 y 251 deberán ser transformados a 1 (ya que son congruentes a 0 $mod(251)$). Encima, como los píxeles de las imágenes están representados con 8 bits, es decir, toman valores de 0 a 255, se puede tener perdida de información

cuando se necesitan transformar los píxeles representados por 252, 253, 254 y 255.

4. Con este método, ¿se podrían guardar secretos de cualquier tipo (imágenes, pdf, ejecutables)? ¿Por qué? Relacionarlo con la pregunta anterior

Por lo mencionado en la pregunta anterior, cualquier tipo de archivo a guardar es susceptible a perdida de información como consecuencia de trabajar en módulo 251. En el caso de este trabajo, al estar utilizando imágenes de formato .bmp, dicha perdida de información se traduce en cambios en algunos de píxeles de la imagen. A continuación se puede observar como ejemplo la imagen secreta recuperada con las imágenes portadoras recibidas por la cátedra:

Figura 1: Imagen secreta recuperada con las imágenes portadoras para el grupo 15.

Podemos notar los bordes completamente negros o alguna secciones donde se nota claramente la perdida de información al recuperar la imagen. Pero como se mencionó anteriormente, en este caso, dicha perdida no es de gran importancia, y solo afecta en la claridad y los detalles de la imagen.

Por ende podemos concluir que en archivos ejecutables o archivos donde cada bit es

relevante, no sería recomendable utilizar este método, ya que al momento de recuperarlo, se puede llegar a corromper el mismo. En cuanto a los archivos pdfs o similares, dependerá exclusivamente del fin del mismo, ya que si solamente se lo utiliza para entender el contenido, el cambio en algunos bits no presentaría diferencias significativas.

5. ¿En qué otro lugar o de qué otra manera podría guardarse el número de sombra?

Se podrían elegir dos píxeles en particular y guardar el número de sombra en los primeros 4 bits de ambos pixeles. De esta forma, el número de sombra no será sobreescrito ni con LSB4 ni con LSB2, ya que estos bits no se ven afectados al momento de distribuir el secreto. Hay que tener en cuenta que esta alternativa tiene implícito un trade off, el cual es que estos dos píxeles se verán ligeramente diferentes comparados con el de la imagen original.

6. ¿Qué ocurriría si se permite que r, a_0, a_1, b_0 o b_1 sean 0?

En primer lugar, si r es 0, esto significa que el polinomio utilizado para compartir secretos es una función constante. Esto haría que el esquema fuera vulnerable a ataques, ya que un atacante podría adivinar fácilmente el valor de la constante y reconstruir la imagen secreta sin necesidad de compartirla.

En segundo lugar, necesitamos que a_0 y a_1 sean distintos de 0 para que en el sistema de ecuaciones de cheating detection ambos sean invertibles, y así poder calcular el valor de r.

En tercer lugar, si b_0 o b_1 son 0, esto significa r, a_0 o a_1 son 0 siguiendo el sistema de ecuaciones para cheating detection, lo cual resulta en un absurdo por lo mencionado anteriormente. Esto quiere decir que las sombras entregadas a los participantes carecen efectivamente de sentido. Esto haría que el esquema fuera ineficaz, ya que los participantes no podrían reconstruir la imagen secreta, incluso si enviaran sus sombras honestamente.

En conclusión, permitir que r, a_0, a_1, b_0 y/o b_1 tomen el valor 0 debilitaría significativamente la seguridad y la eficacia del método propuesto en el documento.

7. Explicar la relación entre el método de esteganografía (LSB2 o LSB4), el valor k y el tamaño del secreto y las portadoras.

La relación entre k y el método de estenografía la sabemos por consigna y es:

- Si $k = 3, 4$ se debe utilizar LSB4.
- Si $k = 5, 6, 7, 8$ se debe utilizar LSB2.

Como se demostrará a continuación, a menor valor de k, se utilizarán más bloques

para ocultar la imagen. Como se tendrán más bloques, y, por tanto, más polinomios, para lograr insertar correctamente la información en las imágenes portadoras se deberán utilizar los 4 bits menos significativos de cada pixel. A medida que se aumente el k, y a su vez la cantidad de bloques disminuya, se podrá reducir el número de bits a utilizar, pues se deberá ocultar menos información.

Haciendo el cálculo numérico para el tamaño de las imágenes que recibimos para verificar el algoritmo (280x440), podemos ver claramente como se va reduciendo la cantidad de bloques a medida que se aumenta el k. Es decir, la cantidad de bloques $\#B_k$ está dado por:

$$\#B_k = \frac{280 * 440}{2k - 2}$$

$$\#B_3 = 30800 \qquad \#B_5 = 15400,$$
$$\#B_6 = 12320 \qquad \#B_8 = 70400$$

Notemos que no se realizó la cuenta con $k = 4$ y $k = 7$ ya que el tamaño de la imagen no es divisible por $2k - 2$ con dichos valores.

Para cada bloque, al usar el algoritmo de secreto compartido con cheating detection, se deben guardar $f(x)$ y $g(x)$. Si se usara LSB2, se necesitarían 8 píxeles para ocultar cada bloque.

$$30800 * 8 = 246400 > 123200 = 280 * 440$$

$$15400 * 8 = 123200 = 280 * 440$$

$$12320 * 8 = 98560 < 123200 = 280 * 440$$

Se puede ver que a partir de $k = 5$ se podrá realmente usar LSB2 para ocultar el secreto en las imágenes, ya que el tamaño es igual al de la imagen, y a medida que se siga aumentando el k, sobrarán más píxeles para ocultar el secreto.

Para $k = 3$, si se usara LSB4 se puede ver que se necesitarían 4 píxeles para ocultar cada bloque.

$$30800 * 4 = 123200 = 280 * 440$$

En general, es mejor utilizar un k de mayor tamaño, para así poder utilizar un método de esteganografia que considere menos cantidad de bits. A diferencia del método de Shamir visto en clase, donde el secreto se conseguía evaluando $f(0)$, ahora este se consigue extrayendo todos los coeficientes de los polinomios. Por lo tanto, por lo demostrado anteriormente, podemos concluir que al usar LSB2 será menos notorio el ocultamiento en las imágenes portadoras. A continuación se puede observar un ejemplo:

Figura 2: Imagen original (sin secreto)

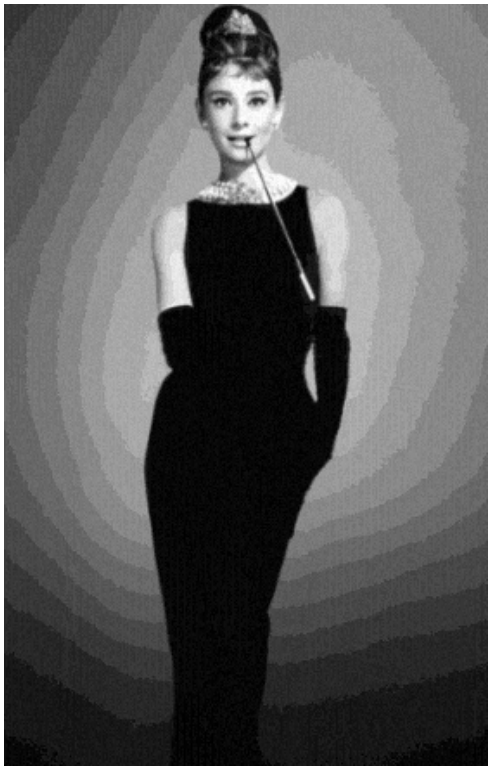

Figura 3: Imagen con secreto con LSB4

Figura 4: Imagen con secreto con LSB2

8. Analizar cómo resultaría el algoritmo si se usaran imágenes en color (24 bits por píxel) como portadoras.

Si se utilizaran imágenes en color como portadoras con 24 bits por píxel, el método propuesto seguiría siendo aplicable, pero habría algunas consideraciones adicionales a tener en cuenta. En primer lugar, el tamaño de las sombras sería mayor debido al aumento del número de bits por píxel, teniendo que volver a calcular el tamaño del secreto como se mostró anteriormente para encontrar los umbrales a partir de los cuales se podría utilizar LSB4 y LSB2. Esto aumentaría la complejidad computacional del esquema, ya que se requerirían más cálculos durante la etapa de reconstrucción del secreto.

Por último, el uso de imágenes en color requeriría un procesamiento adicional para separar la imagen en sus canales de color individuales (por ejemplo, RGB), lo cual también solicitaría cálculos y espacio de almacenamiento adicionales. También se requerirán consideraciones adicionales para asegurar la seguridad del secreto compartido. Un ejemplo de esto podría ser asegurarse de que los canales de color se compartan de forma independiente para evitar que un atacante reconstruya la imagen secreta combinando las partes de diferentes canales de color.

9.a Discutir la facilidad de implementación del algoritmo implementado.

Al haber escogido el lenguaje Java para realizar el trabajo no contamos con ninguna gran dificultad al momento de enfrentar la implementación de este. Las únicas complejidades encontradas las podemos relacionar con un correcto entendimiento del algoritmo mencionado en el paper y los pasos que este conllevaba, como también darnos cuenta de que el método de Lagrange simplificado era más sencillo de implementar.

Cabe destacar que al utilizar IntelliJ IDEA, contamos desde el primer momento con herramientas como el debugger y el refactoring, que permitieron avanzar rápidamente con el desarrollo del proyecto.

9.b Discutir la posibilidad de extender el algoritmo o modificarlo.

Podría extender el método agregando complejidad a los polinomios, con grado mayor a 8, abriendo la posibilidad de un mejor ocultamiento esteganográfico al desbloquear la posibilidad de usar LSB1. También podría introducirse alguna variabilidad en como se insertan los datos a ocultar a medida que se incremente el valor de k, ya que el secreto no entrará 'justo' en la imagen portadora.

Además, se podría extender la funcionalidad y generalizar el algoritmo para usar imágenes a color de 24 bits, modificando algunos cálculos internos para el tamaño de bloques, como bien se mencionó en la pregunta 8.

10. ¿En qué situaciones aplicarían este tipo de algoritmos?

Una posible aplicación del algoritmo es en el campo de la transmisión segura de imágenes. Por ejemplo, el algoritmo podría utilizarse para compartir imágenes sensibles (como imágenes médicas o documentos clasificados) entre un grupo de personas autorizadas, asegurando que las imágenes se mantengan confidenciales y no puedan ser accedidas por partes no autorizadas.

Otra posible aplicación del algoritmo es en el área de la copia de seguridad y recuperación segura de datos. Un ejemplo podría ser utilizar el algoritmo para compartir una copia de seguridad de una imagen crítica del sistema entre un grupo de administradores, asegurando que la copia de seguridad se mantenga segura y pueda ser recuperada en caso de fallo del sistema o pérdida de datos.

Conclusión

El método propuesto en el paper logra realizar una correcta extensión del método de Shamir de secreto compartido para poder ocultar un secreto en imágenes portadoras y detectar cheating en la etapa de recuperación. Sin embargo, es de suma importancia tener en cuenta que, a causa de estar planteado en $GF(251)$, el algoritmo puede conllevar a perdida de información del secreto original al momento de querer recuperarlo.

Bibliografía

- https://jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-018-1084-7