Trabajo Práctico de implementación

Criptografía y Seguridad

Secret Image Sharing Scheme

Profesores

- Abad, Pablo
- Arias Roig, Ana
- Ramele, Rodrigo

Alumnos

- Bezdjian, Alejandro
- Colloca, Tomás
- Ducret, Argentino
- Suárez Bodean, Joaquín

Introducción

El objetivo del trabajo práctico es implementar el algoritmo descrito en el paper “An Efficient Secret Image Sharing Scheme” de Kuang-Shyr Wu y Tsung-Ming Lo. Dicho algoritmo de esteganografía sirve para ocultar una imagen dentro de otras y para recuperarla sin depender de una tabla predefinida o conocer de antemano cual es el número de las sombras.

Además se analizarán temas propuestos por la cátedra, se explicará de forma resumida cómo fue la implementación del algoritmo y por último se presentarán los resultados obtenidos de las imágenes brindadas por la cátedra.

Cuestiones a analizar

Documento de Luang-Shyr Wu y Tsung-Ming

Este paper abarca el tema de (r, n) threshold secret image sharing, explica cómo encriptar una imagen, el secreto, en n sombras y como no se puede obtener ningún tipo de información a partir de r-1 sombras. En otra palabras, un atacante que obtiene r-1 sombras sólo va a obtener r-1 ecuaciones con r incógnitas obteniendo así un sistema indeterminado.

Además, hicieron una extensión al paper "Thien-Lin algorithm", el cual utilizaba un procedimiento similar pero utilizando módulo 251. En esta extensión se utiliza módulo 257 y se propone una solución al problema de obtener 256 al aplicar módulo, ya que se pretende trabajar con valores en el rango [0, 255].

Organización formal del documento

El paper se encuentra organizado en las siguientes secciones: abstract, introduction, review, proposed method, results, conclusions y references. Entendemos que es la organización estándar de cualquier documento formal y que su redacción es correcta.

Se podría haber esperado una sección en la que se define la notación a usar dentro del paper y que los algoritmos fueran descriptos en forma de pseudocódigo.

Vale la pena destacar, que el documento no es extenso; brinda la información necesaria y de forma clara.

Existe un pequeño error en la notación en el algoritmo propuesto: cuando se describe el algoritmo de encripción, se llama " O " a la imagen secreta, pero al final del algoritmo de desencriptación, se utiliza " O' " (que es la notación utilizada en el algoritmo de Thien-Lin). Consideramos que este error no es demasiado grave ya que la letra utilizada es la misma (solo se le agrega un apóstrofe) y además la frase dice: " [...] to get the secret image O' " por lo que se entiende perfectamente a lo que se refería.

Algoritmo de distribución

Para encriptar una imagen O en n sombras (imágenes: $S_1, S_2, ..., S_n$) se deben seguir los siguientes pasos:

1. Generar la tabla de permutaciones R (obtenida por un generador de números pseudo aleatorios).
2. Aplicar la operación XOR entre la tabla R y la imagen original O. La imagen resultante la denotaremos Q.
3. Comenzar con un contador $j = 1$ que recorrerá las secciones.
4. De forma secuencial obtener r pixeles no procesados $(a_0, a_1, \ldots, a_{r-1})$ de Q para formar una sección. Está sección será la número j-esima, además crear un polinomio de grado $r-1$ como el siguiente:
$$f_j(x) = (a_0 + a_1 x + \ldots + a_{r-1} x^{r-1}) \; mod \; 257$$
5. Generar las n sombras $(f_j(1), f_j(2), \ldots, f_j(n))$ y asignarlos de forma secuencial a las n sombras $S_1, S_2, \ldots, S_n$.
6. En el caso de que $f_j(x) = 256$, al primer valor no nulo de $\{a_0, a_1, \ldots, a_{n-1}\}$ decrementarlo en 1. Es decir si a_0 es no nulo, reasignar $a_0 = a_o - 1$.
   Volver al paso 5.
7. Incrementar j en 1.
8. Repetir los pasos del 3 al 7 hasta que todos los píxeles de Q sean procesados.

Algoritmo de recuperación

Los siguientes pasas son los utilizados para recuperar una imagen a partir de r sombras:

1. Comenzar con $j = 1$, valor que indica la sección a procesar.
2. Obtener un pixel no procesado en la sección j de cada una de las r sombras.
3. Usar estos r pixeles en $f_j(1), f_j(2), \ldots, f_j(r)$ y aplicar interpolación de Lagrange y obtener así los coeficientes $a_0 + a_1 + \ldots + a_{r-1}$. Estos valores son los r píxeles de la secciones j de la imagen Q.
4. Incrementar j en 1.
5. Repetir los pasos 2 al 4 hasta que todos los píxeles de las sombras sean procesados.
6. Aplicar la operación XOR entre la tabla de permutación pre definida R y Q para obtener la imagen secreta O.

En nuestro caso la tabla de permutación no es predefinida sino que se debe generar con una semilla la cual será ocultada en las imágenes portadoras. Pudiendo recuperarla y volver a generar la tabla de permutaciones para revertir la operación XOR realizada.

Es importante mencionar que en ningún momento el paper explica cómo resolver utilizando interpolación de Lagrange, ni sugiere utilizar Gauss para resolver el sistema de ecuaciones lineales modular. Tampoco se menciona como ocultar ni recuperar las sombras respecto de las imágenes

que las contienen. En adición, el paper no contempla casos bordes como por ejemplo cuando el tamaño de la imagen no es un múltiplo de *r*.

Notaciones utilizadas

Las notaciones utilizadas en el paper son claras, y además respetan las anotaciones del paper inicial de Thien y Lin, por lo tanto no varían a lo largo del documento.

Imagen recuperada

La imagen recuperada podría no ser igual a la original, como se mencionó en secciones anteriores hay un caso en el cual se puede perder información que es cuando el valor obtenido al aplicar módulo 257 es 256 (valor no representable en el rango [0, 255]).

Asumiendo que lo descrito en el párrafo anterior ocurrió al generar las sombras, a la hora de recuperar la imagen secreta es imposible predecir o saber si esto sucedió en algún píxel por lo cual no se puede recuperar esa informacion perdida, sin embargo la imagen secreta recuperada será idéntica para el ojo humano ya que la pérdida de información puede ser considerada despreciable.

Criterio utilizado para k distinto de 8

Es importante notar que las imágenes a utilizar como sombras deben tener al menos M’ = M / k * 8 píxeles, siendo M el primer múltiplo de k mayor o igual a la cantidad de píxeles de la imagen secreta. En caso en que la imagen tenga una cantidad de píxeles no múltiplo de k, se la rellena con ceros. Notar que cuando k = 8, M’ = M.

En adición, se decidió ocultar el ancho y alto de la imagen a ocultar en los bytes [38_{10} ; 42_{10}) y [42_{10} ; 46_{10}) respectivamente, que no suelen ser utilizados en las imágenes .bmp. De esta forma, se puede recuperar el *header* de la imagen que se ocultó. Se podría también haber utilizado los segundos bits menos significativos para esconder esta información de no haber notado que dichos bytes no suelen ser utilizados. Esta opción puede habilitarse con el argumento -wh para el caso k = 8 para no necesitar que las imágenes portadoras tengan que tener la misma dimensión entre sí e igual a la imagen a ocultar.

Por último, en caso de que las imágenes portadoras tengan distinto tamaño, se toma la cantidad de secciones de la imagen como el mínimo de la cantidad de píxeles de cada imagen portadora. Finalmente, utilizando el ancho y alto de la imagen, se logra recortar aquellos píxeles obtenidos que no pertenecían a la imagen original si todas las imágenes portadoras tienen una cantidad de píxeles estrictamente mayor a M’.

Implementación

Extensiones

Para imágenes de color de 24 bits por pixel podríamos variar el algoritmo de la siguiente manera:

Para cada banda de 8 píxeles R, G y B aplicar el algoritmo. Esto genera las sombras S_i^R, S_i^G y S_i^B con $1 \leq i \leq n$. A su vez, se reemplaza cada bit menos significativo de cada banda de las imágenes portadoras con el bit correspondiente de la sombra y banda en cuestión.

Dificultades durante el desarrollo

En líneas generales el algoritmo es simple y está descrito de forma clara. Una dificultad que se encontró fue el manejo del tipo de datos byte en Java, debido a que es un tipo de datos con signo. Esto hace que al leer los bytes de la imagen, los números superiores a 127 sean tomados como negativos. Para solucionar esto es necesario aplicarles la máscara 0xFF.

La segunda dificultad encontrada fue que en el paper dice que es necesario utilizar el algoritmo interpolador de Lagrange para recuperar los coeficientes del polinomio. Si bien esto es realizable en papel, es mucho más difícil hacerlo algorítmicamente en código. La solución a este problema es utilizar otro método diferente al de Lagrange que se explicará el la siguiente sección.

Método resolución del sistema de ecuaciones lineares modular utilizando Gauss

Para resolver el sistema de ecuaciones del paso 3 del algoritmo de recuperación se utilizó Gauss. De esta forma, si $f_j(i) = c_o^i \,.\, a_0 + \ldots + c_{r-1}^i \,.\, a_{r-1} = b_j^i$, con b_j^i el píxel en cuestión para la sombra i, se crea la matriz C|B con k filas donde la i-ésima fila de C está compuesta por $c_o^i, \ldots, c_{r-1}^i$ y el i-ésimo valor de B es igual a b_j^i. Luego se escalona C|B utilizando operaciones elementales de fila, y finalmente se resuelve desde la fila r a la 1 cada ecuación modular de la forma $x' \,.\, a_{r-1} = b' \bmod (257)$ donde x' y b' son valores conocidos. Las filas r-1 … 1 se pueden resolver reemplazando con los coeficientes ya hallados.

Cambios propuestos para el algoritmo

Modificaciones

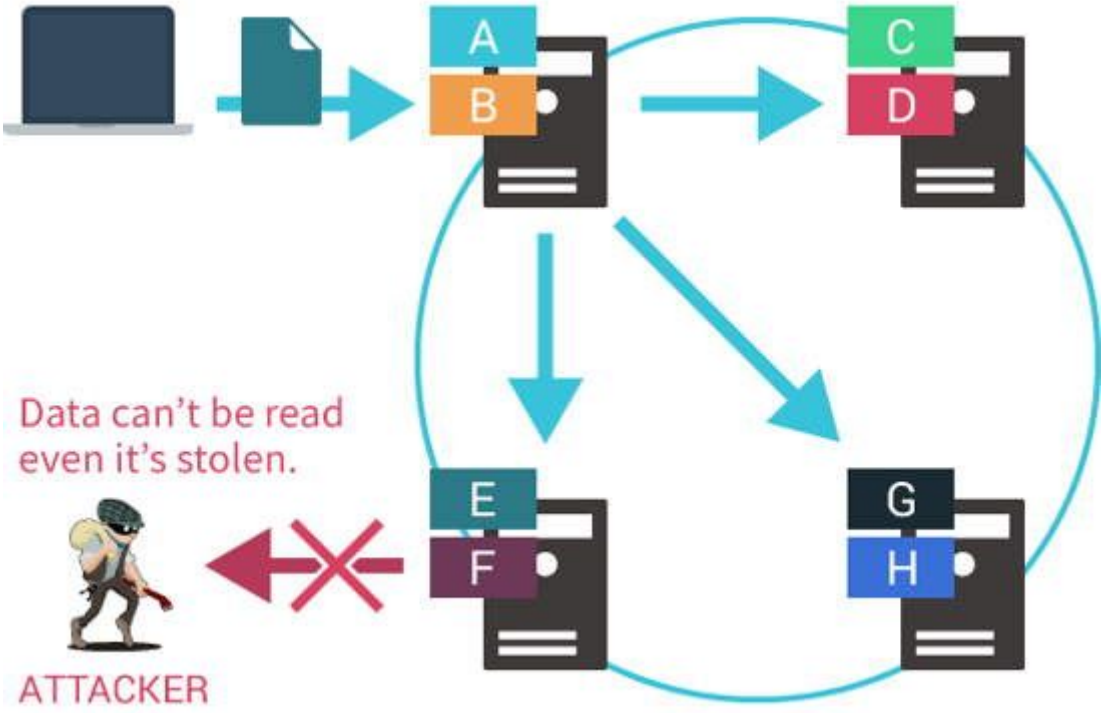
Como se mencionó anteriormente, se modificó la forma de resolver el sistema utilizando Gauss en lugar de utilizar el polinomio interpolador de Lagrange. El algoritmo permite guardar la semilla que genera la tabla de permutación en las imágenes portadoras, de esta forma reduciendo el tamaño de la clave. Por otro lado, se explicó anteriormente como se podría extender el algoritmo para imágenes de color. En adición, si se decidiera no perder en absoluto información, se podría guardar cada valor obtenido con el polinomio en 9 bits en lugar de 8, y de esta forma no perder información cuando el resultado de evaluar el mismo da 256.

Situaciones de uso

Se utiliza en situaciones que se necesita evitar tener un único punto de falla, como ya se mencionó anteriormente este algoritmo propone dividir el secreto en n sombras las cuales se deben guardar en lugares diferentes y seguros con lo cual se evita tener un único punto de falla. Si un atacante logra obtener entre una sombra y r-1 no lograría obtener nada de información sobre el secreto.

Normalmente se utiliza en ámbito militar o comercial para proteger información sensible a la cual se le aplicará el algoritmo y n personas tendrán cada una de las sombras guardada en un lugar seguro. Al momento de recuperar la información se deberán juntar la r sombras requeridas.

Es importante que luego de aplicar el algoritmo se destruye todo rastro del secreto original y datos que el algoritmo haya podido guardar para así evitar que un atacante logre inferir informacion gracias a estas pistas.



Vale la pena notar que este algoritmo también se podría aplicar a otro tipo de secreto que no sea necesariamente una imagen, ya que lo que importa es la representación en bytes de la misma. Sin embargo, es posible que la pérdida de información por el caso mencionado puede que sí sea evidente al ojo del humano, y sea necesario utilizar 9 bits para codificar el resultado del polinomio para no tener dicha pérdida.

Resultados

Se aplicó el algoritmo de recuperación sobre las sombras brindadas por la cátedra. El secreto hallado se encuentra a continuación:

Bibliografía

- Kuang-Shyr Wu, Tsung-Ming Lo (2013). An Efficient Secret Image Sharing Scheme. Trans Tech Publications, Switzerland. Recuperado en: https://www.scientific.net/AMM.284-287.3025