TRABAJO PRÁCTICO DE IMPLEMENTACIÓN: SECRETO COMPARTIDO CON ESTEGANOGRAFÍA

72.04 Criptografía y Seguridad

Primer cuatrimestre de 2021

Grupo 5

Integrantes:

Nombre	Apellido	Legajo
Florencia	Petrikovich	58637
Gonzalo	Hirsch	59089
Rodrigo Manuel	Navarro Lajous	57307

Fecha de entrega: 24 de Junio de 2021

Introducción

El concepto de Esquema de Secreto Compartido se remonta al año 1979 cuando Adi Shamir y George Blakley concibieron, de forma separada, este concepto. Yendo más allá en la historia, en 1994 Adi Shamir y Moni Naor introdujeron el concepto de criptografía visual, una extensión del esquema de secreto compartido pero aplicado a imágenes. Estas imágenes contenían información secreta y se encontraban distribuidas de manera segura. Para recuperar el secreto había que superponer estas distintas imágenes secretas.

En el Esquema de Secreto Compartido de Shamir, el secreto se divide en un número fijo de N partes, siendo que si se tiene un subconjunto de K partes diferentes ($N \geq K$), el secreto se puede reconstruir, de lo contrario el secreto queda indeterminado.

En este trabajo se busca realizar la implementación de la extensión al esquema de secreto compartido de Shamir para compartir una imagen secreta. Para ocultar de manera imperceptible a la imagen secreta, se utilizarán técnicas de esteganografía, la ciencia que se ocupa de la manera de ocultar un mensaje. El objeto que tendrá parte del secreto ocultado dentro de sí mismo se llama portador.

La esteganografía y la criptografía se complementan. Mensajes cifrados y ocultados con esteganografía tienen un nivel de seguridad mucho mayor en comparación con mensajes simplemente encriptados o simplemente ocultados con esteganografía.

Aspectos relativos al documento

Para el desarrollo de este trabajo práctico, se utilizó para la comprensión del algoritmo de Shamir el paper *Sistema de Imagen Secreta Compartida con Optimización de la Carga Útil* escrito por Espejel-Trujillo, Castillo-Camacho y otros. A continuación, se analizan cuestiones del documento.

Organización formal del documento

En lo relacionado a la organización formal del documento, se puede decir que en general estaba bien. Tiene secciones bien definidas que llevan un orden lógico entre ellas. Introducción, codificación, decodificación y resultados.

Si bien se considera que en general estaba bien estructurado, se resaltan dos aspectos que no ayudaban a la comprensión. Por un lado, la introducción es sumamente larga, aproximadamente tan larga como la descripción de la codificación y decodificación, y al hablar del estado del arte respecto la codificación, pero en detalles muy generales y relativamente vagos, se consideró que confunde al lector si uno no tiene una idea concreta de los algoritmos que menciona. Por otro lado, en varias secciones hacer referencia a figuras que se encuentran mucho después, por ejemplo, en la sección de codificación hace referencia a figuras que se encuentran dentro de la sección de decodificación, y eso es confuso.

Descripción del algoritmo de distribución y del algoritmo de recuperación

En términos generales las descripciones de los algoritmos eran claras. Sin embargo, resulta confuso que se haga referencia a figuras que se encuentran en otras secciones cuando realmente deberían encontrarse en las secciones que corresponden al algoritmo en cuestión.

Se resalta que en la codificación no era muy claro que iban a haber tantos polinomios como bloques del secreto. Tampoco mencionaba que hacer si habían X repetidos en diferentes bloques a utilizarse.

En la decodificación el algoritmo está descrito simple y claro, pero se hace énfasis en que tener figuras del algoritmo de codificación dentro de la descripción del algoritmo de decodificación dificulataba la comprensión del mismo. Vale la pena mencionar que las fórmulas de interpolación no contemplaban el caso en que un punto tenga $X = 0$, y tampoco eran totalmente correctas. Hizo falta la intervención de la cátedra para proveer una fórmula que efectivamente funcionara.

Notación utilizada

En términos de la notación utilizada, en general estaba bien y era clara. De todas maneras, hay algunos casos en que no era muy clara. Generalmente se seguían utilizando las notaciones con *i* y *j* pero sin recordar a que hacían referencia varias páginas después.

Respecto de la notación de la codificación, no era muy clara la notación que se utilizaba para los polinomios ya que había una P y en realidad hay varios polinomios distintos dependiendo del bloque. Se considera que faltaría un subíndice para poder indicar a qué polinomio hace referencia.

Respecto de la notación de la decodificación, las fórmulas de la interpolación estaban mal, ya que no servían y no contemplaban el caso en que un punto fuera cero, por lo que potencialmente podía haber un error de división por cero.

Carga útil

El título del documento es "*Sistema de Imagen Secreta Compartida con Optimización de la Carga Útil*", al hablar de optimización de carga útil se refiere a que en comparación con algoritmos previos a este, la carga de información por byte de la imagen portadora se incrementa. Esto es debido a que en los algoritmos anteriores que menciona se esconde 1 solo bit por byte, siendo que para guardar 1 byte de secreto son necesarios 8 bytes de imagen portadora. El algoritmo propone una forma de reducir esa relación, de tal manera que el secreto puede ser tan grande como la imagen portadora, y en algunos casos, hasta mayor que la portadora (depende de los parámetros).

Durante el desarrollo del algoritmo se demuestra que mediante el esquema propuesto se pueden esconder K bytes del secreto en 1 bloque 2x2 entre las N imágenes portadoras. Si el K resulta ser 4, se va a utilizar la imagen portadora completa, pero en caso de que se utilice un K mayor a 4, se puede ver que la cantidad de información por bloque se guarda es mayor, necesitando menos bloques de la imagen portadora. En sí, se van a necesitar tantos bloques de portadoras como longitud(secreto)/K.

En comparación con los métodos anteriores que necesitaban portadoras de un tamaño mucho mayor al secreto, en este caso se pueden usar portadoras más chicas que el secreto. De todas formas, esta optimización no viene sin un precio, utilizar un K muy grande implica que se necesitan N >= K imagenes.

Campo de Galois

Las operaciones de Galois son un elemento muy importante en el documento propuesto por la cátedra. Las mayores ventajas que brindan son poder operar dentro del rango [0, 255] (en este caso), sabiendo que los resultados de las operaciones siempre van a encontrarse

dentro del rango mencionado. Esto brinda la posibilidad de operar sin pérdidas de información, en comparación con las operaciones módulo que podrían ser módulo 251 o 257 (ambos primos).

Por otro lado, la principal desventaja que poseen es el costo computacional realizar todas las operaciones con Galois. Esto puede mitigarse haciendo uso de “Look Up Tables” precomputadas dado el polinomio generador.

Polinomio generador

El polinomio generador utilizado era una medida de la cátedra para poder asegurarse de que las operaciones de todos los grupos sean iguales. De todas maneras, se podría operar con otro polinomio dentro de $GF(2^8)$, siempre y cuando las imágenes portadoras hayan sido generadas con ese mismo polinomio.

El polinomio en sí puede ser utilizado como clave, proporciona una capa de seguridad extra porque alguien teniendo todas las imágenes portadoras no podría recuperar el secreto a menos que sepa el polinomio utilizado para codificar el secreto dentro de las imágenes. Es análogo a dos partes teniendo que ponerse de acuerdo con el uso de un generador. Si se utilizara como clave sería necesario buscar una buena manera de resguardarlo.

Secretos

En el documento propuesto se menciona que se pueden codificar documentos de todo tipo. Esto es verdad y viene de la mano de 2 factores muy importantes. El primer factor es que al utilizar operaciones de Galois se puede codificar y decodificar el secreto sin pérdidas, por lo que el ejecutable o el PDF codificado se puede recuperar entero. Si hubieran pérdidas, el ejecutable o el PDF no podrían ser recuperados o en caso de que se recupere con alguna pérdida, no podrían utilizarse. El segundo factor es que todos esos secretos son reducibles a un arreglo de bytes, y para el algoritmo es indiferente si el arreglo de bytes proviene de una imagen, PDF o ejecutable. Es cuestión de saber cómo guardarlo al momento de recuperar el contenido.

Implementación

Adaptaciones para el guardado de una imagen completa

Si se desea guardar el encabezado de la imagen al igual que el arreglo de pixeles, no se tiene que adaptar la implementación actual. El algoritmo simplemente necesita que su secreto se pueda reducir a un arreglo de bytes. El header, como el arreglo de pixeles, son bytes, lo cual permite que se escondan dentro de las sombras junto con los pixeles. Sin embargo, para que funcione el algoritmo (con o sin encabezado), se debe poner limitaciones sobre los posibles valores de K.

Suponiendo que L es la longitud en bytes de la imagen completa (encabezado y píxeles), la cantidad de polinomios que se utilizaran para la codificación son $\frac{L}{K}$ (1). Por ende, las imágenes portadoras deben contar como mínimo con esta cantidad de bloques de píxeles, dado que si contara con menos no se podría esconder todo el secreto. Tomando los bloques de píxeles como bloques 2x2 y *n* como la cantidad de pixeles en la imagen portadora, K debe cumplir la condición (2).

$$\#Polinomios = \frac{L}{K} \tag{1}$$

$$n \geq 4 \times \frac{L}{K} \;\Rightarrow\; \frac{K}{L} \times n \geq 4 \;\Rightarrow\; L \geq K \geq 4 \times \frac{L}{n} \tag{2}$$

Por ende, verificando el cumplimiento de esta condición antes del codificado, se puede guardar secretos de cualquier tipo, incluyendo la imágen completa.

Aspectos del algoritmo

Facilidad de implementación

Una vez entendido el algoritmo, su implementación se consideró sencilla. La complejidad se atribuye al análisis y comprensión del paper, dado que algunas áreas no se consideran que fueron muy claras (como las fórmulas de interpolación, la cantidad de polinomios existentes y manejo de colisiones). Lo que se considera que fue más complejo en la implementación fue el manejo de archivos bmp, al igual que la transformación eficiente del arreglo de pixeles a bloques de píxeles.

Posibilidad de extensión o modificación del algoritmo

El algoritmo en cuestión es un algoritmo muy genérico, y ya de por sí es muy flexible. Si bien como mencionamos es muy genérico, se podrían implementar pequeñas extensiones o modificaciones.

El algoritmo de interpolación podría cambiarse siempre y cuando se mantengan las operaciones de Galois y se asegure que funciona correctamente. Otra potencial modificación o extensión es cambiar la manera en que se evitan las colisiones de X para distintos bloques, algo que no está contemplado en el documento en sí. También se podría cambiar cómo se codifican los bits en los bytes del bloque, pero eso potencialmente puede afectar a la calidad de la imagen portadora. Por último, se podría extender para poder codificar imágenes enteras (encabezado + contenido) para no perder el encabezado cuando se codifica.

De todas maneras, es importante resaltar que el algoritmo es muy bueno, funciona extremadamente bien y es lo suficientemente genérico como para que se puedan usar la cantidad de imágenes que se quieran.

Aplicación de algoritmos de secreto compartido

Algoritmos de secreto compartidos se pueden aplicar en una gran variedad de situaciones. Estos algoritmos requieren de la combinación de información guardada por más de una entidad para lograr la decodificación de información encriptada con algoritmos simétricos. Se utilizan en situaciones de alto riesgo donde una sola entidad no debería tener el poder para obtener la información encriptada. Las sombras se distribuyen a varias entidades y solamente juntando K de las sombras se puede recuperar la información.

Un ejemplo de esta situación es la encriptación de código nucleares. Una sola entidad no tiene el poder de obtenerlos, sino que varias se deben juntar y optar por su desencriptación.

Este tipo de algoritmo también permite establecer *jerarquías*. Si se desea que el poder de una entidad se mayor a las demás al momento de desencriptar, se le otorga a la misma más sombras.

El algoritmo de Shamir también permite la posible pérdida de información (sombras) sin resultar en pérdida del secreto. Esto se debe a que el algoritmo codifica la información del secreto en N sombras y solo necesita K sombras, con K <= N, para recuperar el secreto. Resulta útil entonces en situaciones donde existe la posibilidad de pérdida de algunas sombras.

Por último, el algoritmo tiene usos prácticos en situaciones en las cuales se quiere que dos entidades puedan recuperar el secreto por su cuenta utilizando información distinta. En este caso, se le podría otorgar a cada entidad K sombras diferentes para que ambas tengan el poder de desencriptación. Si se requiere que los conjuntos de K sombras de las entidades sean disjuntos, se debería cumplir que $N \geq 2K$.

Análisis

Imágenes en color (24 bits por píxel)

En el caso de poseer una imagen de 24 bits por pixel, el algoritmo resultaría en lo siguiente. Usando los datos del encabezado de la imagen se puede ver cuántos bits por pixel se utilizan. Esa información hay que complementarla con la información que contiene el encabezado sobre las máscaras que se usan de cada color para determinar qué bits corresponden a cada color. Sabiendo que por píxel hay 24 bits, en este caso, se puede recuperar el arreglo de pixeles tomando 24 bits por pixel en vez de 8 como se hace actualmente.

Se consideró que no es tan simple como cambiar los bits menos significativos de cada píxel, porque potencialmente se podría estar cambiando mucho la componente para un solo color, generando un cambio bastante notorio en la imagen portadora. Las máscaras de los colores sirven para poder determinar los bits menos significativos de cada color para poder hacer los cambios de forma que pasen desapercibidos.

Por último, las operaciones utilizando Galois deberían ser hechas dentro de un $GF(2^{24})$ para que el rango de valores con los que se opera sean correctos.

Esquema de bloques

En términos de cómo se toman los bloques de las imágenes portadoras, entendemos que se puede cambiar, pero el esquema 2x2 posee ventajas por sobre otros esquemas.

Se podría utilizar un esquema 4x1 (o 1x4), pero el principal problema que brinda este esquema es que se reducen los bytes útiles de la imagen portadora ya que es más complicado que una imagen tenga una dimensión divisible por 4, comparado con que sea divisible por 2. Estos bytes que no completan un bloque utilizando el nuevo esquema podrían ser no modificados o habría que modificar el recupero para que si lo sean. En ambos casos complica la implementación.

Otra ventaja de hacer uso del bloque 2x2 en comparación con el 4x1 es que como los cambios en W, V y U son en función del valor de X, en un esquema 2x2 los cambios se hacen en los píxeles que rodean al X, minimizando el impacto del cambio. Si se usara el esquema 4x1 los pixeles estarían alejados del X, potencialmente haciendo que el impacto del cambio sea mayor.

Resultados

Imágenes de prueba

Dadas las imágenes de prueba provistas por la cátedra, parte de la prueba para determinar si la implementación funcionaba correctamente era poder recuperar exitosamente el secreto distribuido entre las imágenes en cuestión.

Fueron provistas 5 imágenes de prueba, las cuales eran todas necesarias para poder hacer el recupero exitoso ya que durante la codificación el K era 5. Se pudo encontrar el secreto distribuido entre las imágenes, la cual se puede ver en la figura 1.

Para realizar pruebas más exhaustivas sobre el funcionamiento del recupero del secreto, se intercambiaron imágenes portadoras con otro grupo (G2) para poder hacer las pruebas con ese conjunto de imágenes. El resultado del recupero se observa en la figura 2.

Figura 1: Secreto de la cátedra

Figura 2: Secreto de grupo G2

Casos no contemplados

Como fue mencionado en secciones anteriores, el algoritmo no cubría algunos posibles casos que podían darse. Estos casos podrían ser que dos bloques en la misma posición de imágenes portadoras diferentes tuvieran el mismo X, por lo que la interpolación no se podía resolver ya que se tienen dos puntos iguales. Otro caso que no contemplaba es que al interpolar haya un punto de la forma (0, F(0)), ya que eso generaba una división por cero al calcular Y'.

En los casos que se muestran en las próximas secciones, se tomó una de las fotos de prueba provistas por la cátedra que no tenía secreto, y se la utilizó como secreto para ser distribuida entre otras de las imágenes de prueba. Se utilizó K = 4.

Sin contemplar los casos

En este caso se muestran los resultados sin contemplar los casos mencionados anteriormente. Como se puede ver en la figura 3, hay dos clases de errores en la imagen. Hay varios puntos distribuidos que claramente no tienen el valor correcto, y hay parches de puntos contiguos que todos tienen valores incorrectos.

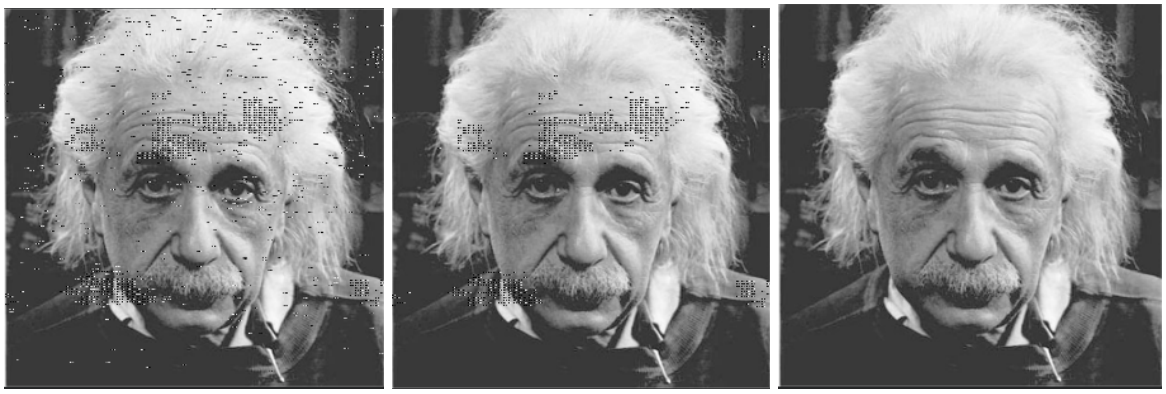
Contemplando posibles X iguales

En este caso se corrigió el algoritmo para contemplar la posibilidad que dos bloques en imágenes distintas posean el mismo X. En este caso hubo que realizar, como se vio en clase, la operación de probar si sumando o restando 1 se encontraba un X que no sea repetido. Los resultados de contemplar este caso fueron que los múltiples puntos erróneos distribuidos ya no aparecían más al recuperar el secreto, pero los parches de puntos erróneos seguían apareciendo, como puede ser visto en la figura 4.

Contemplando todos los casos

El último caso que hubo que contemplar era la división por cero en la interpolación. Para esto se realizó la sugerencia de la cátedra en torno a cómo ordenar los puntos recibidos para evitar este error y los resultados fueron positivos. Se obtuvo una extracción limpia y sin problemas de la imagen secreta distribuida (ver figura 5).

Dado que se fueron contemplando los casos progresivamente, podemos atribuir a cada caso no contemplado su efecto en el recupero de la imagen secreta, como puede ser visto en la progresión de las imágenes.



Figuras 3, 4 y 5 (en orden): Progresión de mejora de la calidad del secreto codificado y luego decodificado

Verificación de pérdida de información

Una última prueba que se realizó fue ver que el MD5 de la imagen distribuida sea el mismo que la imagen recuperada para poder ver si efectivamente habían o no pérdidas. Esta

prueba funciona bajo la suposición de que el encabezado de las imágenes de prueba era el mismo en todos los casos.

Efectivamente los encabezados de las imágenes de prueba eran los mismos, ya que el MD5 del secreto distribuido y el secreto recuperado era el mismo. Estos MD5 se pueden ver a continuación:

```
bash$ md5 images_bk/Albert.bmp
MD5 (images_bk/Albert.bmp) = 0fae4cd90c0f40bd2e7b3fea90000e76
bash$ md5 Albert_Secret.bmp
MD5 (Albert_Secret.bmp) = 0fae4cd90c0f40bd2e7b3fea90000e76
```

Errores encontrados

Para el desarrollo de la implementación se hizo uso de una librería de Python llamada **pyfinite** (para las operaciones de campos de Galois), si bien no es una librería muy utilizada, cumple todos los requerimientos y funciona como era esperado.

Durante el desarrollo se encontró que la librería poseía un error. Al realizar una división (dentro de las operaciones de Galois) del estilo 0 / N, la librería devolvía un valor incorrecto, haciendo que la implementación falle. Se resolvió esto a nivel aplicación teniendo en cuenta estos casos, pero al mismo tiempo se informó al creador de la librería sobre el error para que pueda ser resuelto. El informe terminó en una nueva versión de la librería lanzada con la corrección incluida en base a lo encontrado por el grupo. Se puede ver la issue de Github aquí.

Conclusión

Después del análisis e implementación del algoritmo descrito en el documento, se puede concluir que es un algoritmo muy potente, claramente superior a los algoritmos con los que el documento se compara. Ya de por sí la optimización que plantea es realmente importante. De todas formas, estaría bueno que el documento esté más completo en términos de algunas consideraciones que faltaban, y algunas fórmulas mal escritas porque lo que brindan es algo muy bueno. Es una buena experiencia poder interpretar e implementar el algoritmo descrito en un documento de seguridad, y se considera que se pudieron afianzar y comprender mejor algunos conceptos en esta aplicación práctica.