Trabajo práctico de implementación: secreto compartido con esteganografía

Grupo 12

Integrantes:

- Bartellini H., Mateo F. (61438).
- Brave, Jerónimo (61053).
- Colonnello, Joaquín (59574)
- García Matwieiszyn, Juan I. (61441).

Docentes:

- Abad, Pablo Eduardo.
- Arias Roig, Ana Maria.
- Ramele, Rodrigo Ezequiel.

Contenidos

Análisis del paper

Organización formal del documento

La organización del documento resultó clara y los pasos indicados en el mismo para seguir los algoritmos propuestos fueron efectivos para implementar el trabajo.

Descripción del algoritmo de distribución y del algoritmo de recuperación

Como se mencionó previamente, la descripción del algoritmo fue adecuada, detallada y completa.

Notación

La notación es clara y consistente a lo largo del documento, particularmente en aquellas partes donde se detalla el algoritmo. La misma puede resultar algo confusa al principio por un uso exagerado de subíndices, pero estos mismos subíndices hacen que una lectura detenida del documento sea muy eficaz para comprender e implementar los mecanismos propuestos.

Detección de sombras falsas

El paper propone que por cada bloque de información de longitud *2k-2* se generen dos polinomios de grado *k-1* que contengan como coeficientes los bytes del bloque en cuestión. Al recuperar estos polinomios usando el método de interpolación de Lagrange, vemos que igualmente tenemos coeficientes de más (dos polinomios de grado *k-1* contienen *k* coeficientes cada uno, dando un total de *2k*) ¿Cómo se explica este exceso de información?

El hecho de que haya dos coeficientes de más lo usaremos para **detección de fraudes.** Si llamamos *f* y *g* a nuestros polinomios, podemos calcular a partir de los dos primeros coeficientes de *f* los dos primeros coeficientes de *g* con la ayuda de un *nonce*, que llamaremos *r* y será un número aleatorio entre 1 y 251, elegido de forma uniforme. De esta forma, al recuperar *f* y *g*, podemos recuperar también *r* y verificar que se el mismo tanto para el primer como el segundo coeficiente.

Ahora, ¿cuán eficiente es? Asumiendo que el atacante sabe que estamos trabajando módulo 251, por cada bloque tiene una probabilidad de $\frac{1}{251}$ de acertarle al valor de *r* que fue elegido al azar. Ahora, se deberá acertar correctamente el valor de *r* por cada bloque que se tenga. Si llamamos *N* a la cantidad de bloques, ahora la probabilidad de que un atacante pueda de forma inadvertida utilizar una sombra falsa es de $(\frac{1}{251})^N$, que es significativamente menor computando que *N* puede estar en el órden de 10^4. Luego, la probabilidad de que un atacante acierte el valor de *r N* veces es despreciable.

Problemáticas de trabajar módulo 251

Se quiere modelar una estructura con valores de 0-255 (un pixel). Sin embargo, trabajando en $\mathbb{Z}_{251}$ sólamente se pueden tomar valores entre 0 y 250, lo que en teoría limita el espacio de representación que tenemos. Para remediar esta situación, se analizaron varias soluciones, a saber:

1. Una opción es truncar los píxeles más “claros” (valores mayores o iguales a 251) a 250. La desventaja es que reduce el contraste en zonas de alta luminosidad
2. También se puede bajar la luminosidad de la imagen para que ningún valor supere a 250. El problema con este método es que ahora gran parte de los pixeles se representaría con un valor distinto al original
3. Por último, se puede aplicar el módulo directamente. El problema acá es que estaríamos representando píxeles claros como oscuros

Para este trabajo elegimos la opción 1. Es menos disruptiva visualmente que la opción 3 y el paper asume que la imagen de entrada puede representarse en GF(251), cuando la opción 2 no permitiría distribuir y recuperar estas imágenes sin pérdida de información.

Otro inconveniente de trabajar módulo 251 es que obtener el inverso multiplicativo no es trivial, y se requiere almacenar dichos valores. Sin embargo, el hecho de que se trabaje módulo 251 (o cualquier otro número primo para el caso) es lo que permite que el esquema de interpolación pueda ser utilizado como método de secreto compartido.

Secretos de otros tipos (pdf, png, etc.)

En este caso, el algoritmo es utilizado **únicamente para esconder imágenes BMP**. ¿Podrían esconderse otro tipo de archivos?

La respuesta a esta pregunta no es tan sencilla. En principio, podemos decir que lo que hacemos con los BMP no es directamente replicable a otros formatos de archivo, en particular aquellos que se basan en compresión como JPG o MP3. Esto se debe a que en esta implementación nos basamos en el hecho de que la data de BMP se guarda en el archivo **de la misma forma en la que debe ser representada en la pantalla**. Es decir, permite maniobrar directamente sobre el arreglo de datos. Si quisiéramos replicar exactamente el mismo comportamiento con otro formato de archivo sería imposible (por ejemplo, no podríamos esconder mediante secreto compartido un PDF en otros tantos PDFs portadores).

Sin embargo, si se quiere, también existe una forma de utilizar el mismo esquema para esconder un archivo binario genérico, y es generar una imagen BMP cuya data sea exactamente el contenido en binario de ese archivo, con algún padding para compensar. Cabe destacar que es evidente que la imagen generada por esta data binaria no tendrá ningún valor visual y será muy difícil distinguirla del ruido. Sin embargo, al encontrar una forma de guardarla en formato BMP, podremos ocultarla en otras imágenes mediante esteganografía.

Lugar alternativo para el número de sombra

Existen otros campos reservados en el header que podrían utilizarse, pero hay muchos campos afectarían como se ve la imagen.

También se podría guardar entre el header y la data, pero no hay garantía de que los programas que procesen el BMP mantengan los datos en esta sección, pudiendo afectar la distribución de las portadoras.

Como último se podría codificar sobre los píxeles de la portadora, como el secreto. En este caso habría que replantear el encoding para k=3 y k=5, que usan la totalidad de la portadora y no dejan espacio extra.

¿Qué ocurre si r, a0, a1, b0 o b1 son 0?

Si cualquiera de estos valores fuera cero se rompería el método de detección de engaños propuesto en el paper. Para analizarlo, veamos de qué forma se utiliza el *nonce* para generar los coeficientes de los polinomios. Llamemos a_i al i-ésimo coeficiente de *f* y b_i al i-ésimo coeficiente de *g*. Los valores de a_i quedan completamente determinados por el secreto a guardar para todo *i* entre 0 y *k-1*, mientras que los coeficientes b_i están determinados por la data para todo i entre 2 y *k - 1*. Ahora, veamos cómo elegir b_0 y b_1.

1. Se elige un valor aleatorio para *r*, bajo la restricción $0 < r < 251$
2. Se eligen b_0 y b_1 tales que $ra_0 + b_0 \equiv 0 \pmod{251} \land ra_1 + b_1 \equiv 0 \pmod{251} \Leftrightarrow ra_0 + b_0 \equiv -ra_0 \pmod{251} \land b_1 \equiv -ra_1 \pmod{251}$
3. Cuando estamos recuperando la información oculta, podemos usar el hecho que $r \equiv -b_0(a_0)^{-1} \pmod{251} \land r \equiv -b_1(a_1)^{-1}$, lo que permite comprobar fácilmente que los coeficientes recuperados son correctos

Ahora, es fácil notar que en cualquier de las ecuaciones previamente definidas, tener cualquier parámetro en cero vuelve trivial la resolución de estos mecanismos. Por lo tanto, se pierde toda la parte de detección de engaños.

Esteganografía, *k* y el tamaño de las imágenes

Empecemos notando que, al utilizar LSB2, la información se esconderá en los dos bits menos significativos del byte, mientras que usando LSB4 la información se esconderá en los cuatro bits menos significativos del byte. Es decir, que usando LSB4 el tamaño de los bytes en donde se debe guardar la información debe ser al menos cuatro veces más largo que el secreto, mientras que dos veces si usamos LSB2. Cabe preguntar, ¿por qué utilizamos LSB2 y LSB4 según el valor de *k*?

La respuesta es simple: si utilizamos LSB2 con *k* igual tres o cuatro, la sombra a guardar no entraría en el tamaño de la imágen. Es preciso notar que a medida que incrementamos el *k*, el tamaño de cada sombra disminuye (lo cual es lógico: es más ligero distribuir el mismo secreto entre 10 que entre 2). Es por este motivo que, si *k* es pequeño, nos vemos obligados a usar LSB4.

¿Qué ocurriría si se usaran imágenes en color como portadoras?

Al implementar el algoritmo, la sección de data se utiliza exclusivamente para guardar la información de la imagen secreta, como si la misma fuera un gran arreglo de bytes. Es decir, no le importa al algoritmo si esta data está organizada en un byte para cada color o no, simplemente necesita tener unos cuantos bytes en los que pueda esconder la información que requiere. Por tanto, se podría usar perfectamente imágenes a color como portadoras, y la extensión del algoritmo a las mismas sería sencilla.

Acerca del algoritmo

Facilidad de implementación

El algoritmo no trae mayores problemas para implementar, además de estar claramente planteado y explicado en el paper mencionado previamente. La complejidad de implementación viene únicamente de entender el papel y después plasmar ese algoritmo a código.

Extensión o modificación del algoritmo

El algoritmo se puede extender fácilmente para, por ejemplo, poder utilizar imágenes que no sean de exactamente un tamaño dado, aplicando por ejemplo algún tipo de padding. También se pueden considerar esquemas similares en los que las imágenes portadoras no necesariamente deban ser iguales, lo que permitiría más flexibilidad en el uso.

Casos de uso

Los algoritmos de secreto compartido son útiles en casos en los que se busca requerir un cierto nivel de consenso para desbloquear información. Un ejemplo podría ser la distribución de una clave privada usada para firmar documentos, donde se requiera la participación de un cierto número de integrantes para emitir un documento en nombre de la organización.

Otro uso podría ser un álbum de figuritas criptográfico usando un sistema de secreto compartido (k, k). Se distribuyen copias de cada figurita, en forma de imagen con una sombra embebida, con distintas probabilidades según la “rareza” que se le asigne a cada una. Los participantes deben reunir todas las figuritas para desbloquear un mensaje, imagen especial o clave criptográfica que acredita que completaron el álbum.