Criptografía Y Seguridad
Trabajo Práctico Especial

Informe

Integrantes:

Ignacio Ribas
Francisco Choi
Augusto Henestrosa

Índice

Sobre las cuestiones a analizar

Se discuten uno por uno los puntos a analizar de acuerdo al enunciado:

1. De los aspectos relativos al documento

a. La organización del documento es bastante clara. Las secciones que importan para el desarrollo del sistema propuesto, que en nuestro caso fueron básicamente las de esquema de codificación y de decodificación, quizás podrían tener títulos más grandes. En cuanto al documento en sí como paper, tiene intro, abstract, descripción del esquema propuesto, resultados experimentales y conclusión, como cualquier otro, así que en ese sentido está bien.
b. Los algoritmos de recuperación están claros. Encripción y desencripción están bastante claros en alto nivel, si bien es cierto que algunas aclaraciones en el mismo enunciado de este TPE fueron obviadas en el paper. Un ejemplo de esto es la aclaración de que para un mismo bloque, el valor de X en las imágenes portadoras debe ser diferente, y que para eso es necesario cambiar los números (sumar 1 (mod 256) al valor de X repetido hasta que deje de estarlo).
Igualmente para la interpolación de un polinomio de grado k-1 son necesarios k pares (X, Y), necesariamente sin que se repita X, así que posiblemente no habría sido difícil darse cuenta, dado que si se trazaban los errores en la desencripción, estos se generaban a partir de divisiones modulares por 0 en la interpolación.
También cabe destacar que fueron necesarias algunas consultas respecto de operaciones en el campo de Galois propuesto, dado que no quedaban claras leyendo únicamente el paper.
c. La notación en el documento es tan clara que los algoritmos a bajo nivel fueron fáciles de implementar basándose en este. Sin embargo, en la parte de interpolación de Lagrange, la notación no es clara: pareciera que los signos menos (-) no se muestran y en cambio se ve un espacio en blanco. Eso llevó a varios problemas y después de varios intentos fallidos de aplicar una interpolación de Lagrange en el código, se optó por usar el algoritmo de reducción de líneas de Gauss. Ese funcionó con el ejemplo en clase.

2. Sobre la optimización de la carga útil

La carga útil es la cantidad de bytes necesarios de cada imagen portadora, es decir, los que se utilizan para distribuir el secreto. Desviándose un poco de la definición del paper, la carga útil se define como:

$$Payload(byte) \ = \ NPixeles \ * \ \#(bytes\,por\,pixel) \ * \ 4 \ / \ k$$

donde NPixeles son los pixeles de la imagen secreta, y los bytes por píxel también son los de esa imagen.
De esta manera, a medida que aumenta k, se necesitan menos bytes de cada portadora.

3. Ventajas y desventajas de trabajar en GF(2^8) respecto de trabajar con congruencias módulo

Dado que se trabaja con bytes, pareciera correcto comparar GF(2^8) con la partición inducida por mod(256) o mod(251). Una ventaja de GF(2^8) es la facilidad para manipular los elementos de ese campo aprovechando la representación de byte (cada bit i ϵ [1, 8] representa el coeficiente de grado 8-i, yendo desde el bit más significativo al menos significativo). Las operaciones de suma y resta se pueden hacer con XOR directamente, multiplicación es mucho más tediosa en GF(2^8) que en congruencias módulo.

4. Sobre el trabajo con otro polinomio generador

Se podría trabajar con otro polinomio generador del GF(2^8), sólo cambiarían las tablas de multiplicación. Respecto a su uso como clave, no sería muy efectivo, ya que el fuerte del esquema es el secreto distribuido por interpolación polinómica, no el polinomio generador que se utilice. Si lo que se quisiera es esconder el generador como clave, si tuviera algo como un oráculo de distribución, usando un generador gp como clave, entonces eligiendo bien los bytes de la imagen secreta se podría obtener fácilmente la tabla de multiplicación de GF(2^8) con generador gp, y entonces, asumiendo que a cada generador le corresponda una tabla de multiplicación[1], se puede determinar el valor de gp. De la misma manera, tampoco resistiría un equivalente de Test de Eavesdropping.

5. ¿Por qué se pueden guardar secretos de todo tipo?

En cuanto se puedan representar como un conjunto ordenado de bytes, tanto lo que se quiere esconder como aquellos archivos que sean portadores pueden separarse en bloques para seguir los procedimientos de distribución y recuperación, siempre y cuando sea consistente con las disposiciones de los bloques para ambos algoritmos. A lo sumo cada uno tendrá su tratamiento de headers y el contenido empezará en un offset diferente (para garantizar la parte de “esteganografía” y que no haya ruido blanco o archivos inválidos), pero la idea es siempre la misma, en realidad siempre se trabaja sobre un array de bytes.
Si la estructura básica no pasa el tamaño de un byte, entonces las operaciones en GF(2^8) se pueden realizar sin problemas ya que cada valor que pueda tomar un byte puede representar un polinomio de ese campo directamente, y entonces pueden realizarse todas las operaciones necesarias de ambos esquemas.

6. Adaptación para guardar el archivo de imagen completo (es decir, con encabezado y pixeles)

En la implementación, siguiendo la línea de lo que está en el paper, se divide la imagen secreta en bloques de longitud k, pero desde que comienzan los bytes de los pixeles en sí. La única modificación que habría que hacer es contar el encabezado en la imagen secreta (es trivial el cambio), garantizando que, para cada imagen portadora (siguiendo la idea del punto 2)

[1] Esto se asume pero no presentamos una prueba. Bastaría con probarlo para GF(2^8).

$$\#Q \geq \#SB/k$$

donde #Q es la cantidad de bloques de 4 bytes en la imagen portadora, #SB es la cantidad de bytes de la imagen secreta y k es la cantidad de imágenes necesarias para recuperar el secreto en un esquema (k,n).
Con esa modificación para la distribución, la recuperación requiere el mismo cambio: los bloques de longitud k que se van recuperando se van concatenando desde el byte 1 (offset 0) para formar la imagen secreta.

7. Variación del algoritmo para imágenes en color

Para empezar, se podría tratar a las imágenes en color de la misma manera: siguen siendo arrays de bytes. Se divide la imagen secreta en bloques de k bytes y se siguen usando los mismos bloques de 2x2 para las imágenes portadoras. En este caso, hay que tener en cuenta que son 3 bytes por pixel para la disposición de la imagen. Esto modifica un poco los cálculos de los offsets en los que se encuentran X, U, V y W, pero no debería significar grandes cambios.

8. Sobre la forma de los bloques de las imágenes portadoras

En el paper se presentan los algoritmos considerando XUVW como un bloque de 4, dos filas y dos columnas. Es posible cualquier disposición de XUVW siempre y cuando sea fácil de identificar a qué bloque pertenece. Por ejemplo, se podrían ubicar los 4 valores contiguamente en lugar de tener wLength bytes de separación entre X y V (o U y W), donde wLength es la longitud del bitmap.
Es posible que se haya elegido esta disposición de 2x2 para que el efecto de modificar los 3 bits menos significativos de U, V y W sea aún menos perceptible por el ojo humano. De esta manera, cada valor X de un bloque está a una distancia de 2 bytes de otro valor de X en cualquier dirección. En contraste, en el ejemplo propuesto con XUVW contiguos, por ejemplo, horizontalmente, los valores de X están separados entre sí por 3 bytes, mientras que verticalmente esa separación depende del resto de dividir la longitud del bitmap por 4.
Más gráficamente, en el ejemplo de bloques 2x2 se tienen una sombra de esta manera:

XU XU XU XU XU
VW VW VW VW VW
XU XU XU XU XU
VW VW VW VW VW

Disposición en bloques de 2x2

Se puede observar cómo las X están, horizontal, vertical y diagonalmente separadas por un byte. Mientras tanto, es posible que con la distribución propuesta sea más notable el cambio (si bien no debería ser perceptible en ninguno de los dos tipos de bloques ya que sólo se modifican los 3 bits menos significativos de U, V y W). Se presentan 2 casos de longitudes de imagen.

XUVWXUVWXU

VWXUVWXUVW
XUVWXUVWXU
VWXUVWXUVW

XUVW XUVW XUVW
XUVW XUVW XUVW
XUVW XUVW XUVW
XUVW XUVW XUVW

Disposición en bloques de 1x4

Es posible que esta disposición de los bytes otorgue una mejor calidad de las imágenes de camuflaje.

9. Sobre el algoritmo implementado

a. Facilidad de implementación: el algoritmo fue en sí muy fácil de implementar ya que, aprovechando la notación de elementos de GF(2^8) como cadenas de bits, se pudo hacer uso de operaciones de manipulación de bits nativas de C. En sí los módulos son relativamente chicos y se puede probar fácilmente cada uno (el código se divide en módulos de operaciones en GF(2^8), encripción, desencripción (estos dos últimos se combinaron en cryptosystem), interpolación de polinomios y manejo de BMPs).
b. Por como está implementado, habría que hacer unas modificaciones para que funcione con diferentes tipos de archivos. No sería difícil hacer un módulo de encripción/desencripción genérico para cualquier array de bytes, pero se deberían especificar aspectos como estrategias para la disposición de los bloques en las imágenes portadoras, si existe algún parámetro de alto y ancho del contenido del archivo, etc.

10. Situaciones en las que se pueden utilizar algoritmos de secreto compartido

Cualquier situación en la que se requiera del consentimiento o accionar consciente de más de una persona para efectuar una determinada acción se sirve del uso de este tipo de algoritmos. El tipo de situación más común se menciona en el paper y tiene que ver con la divulgación de información de carácter confidencial, que requiere del consentimiento de más de una persona. Otros objetivos respecto de un secreto, como su modificación o acceso, también pueden utilizar secreto compartido para garantizar que lo que se vaya a hacer sea una acción premeditada y acordada entre partes de igual autoridad.