TP Criptografía y Seguridad

Integrantes:
Bilevich, Andres Leonardo 59108
Lin, Scott 59339
Margossian, Gabriel Viken 59130

1. El Paper

1.1. Organización Formal

Nos pareció que la organización del paper es adecuada y no era confusa. Primero introduce el problema y presenta distintos algoritmos que se fueron desarrollando para propósitos similares al problema que se quiere resolver y explica un poco las características de cada uno. En el desarrollo muestra claramente cómo este se resuelve de forma práctica usando el algoritmo propuesto por el paper. También, utiliza gráficos muy intuitivos para hacer su funcionamiento más fácil de entender. Finalmente se muestran distintos gráficos de análisis y de comparación con las otras formas de estenografía que sirven para ver de forma más fácil las ventajas que ofrece el propuesto.

1.2. Los Algoritmos

Los algoritmos son explicados de forma muy clara, detallada, completa y fácil de entender, como se mencionó antes el uso de gráficos ayuda mucho a entender el funcionamiento de forma más visual.

1.3. La Notación

La notación dentro de todo es clara y consistente, pero encontramos algunos errores a lo largo del paper que son fáciles de corregir.

- Por un lado, cuando se hace mención de la ecuación de la carga útil (9) para un modelo (k,n) la ecuación tiene dos números cuatro, el que está en el numerador debería en realidad ser la k del modelo y no un 4.
- Por otro lado en las ecuaciones (6), (7) y (8) faltan algunos signo “-” que fueron reemplazados por espacios en blanco
- Además, en la ecuación (7) especifica un s1 que en realidad debería ser el s de la iteración anterior, sino estaría siempre usando el s1

2. Optimización de la carga útil

Con “Optimización de la Carga Útil” se refiere a que el método propuesto en este paper, logra guardar más información en cada pixel que el resto de los métodos anteriores. En vez de ocultar un píxel en cada píxel, puede llegar a guardar más información por pixel.

Lo que hace es crear grupos de 4 pixeles en las n imágenes portadoras y seccionando la imagen secreta en tramos de tamaño k, luego para cada uno de estos tramos crear un polinomio de grado k-1 donde los coeficientes son los valores de los píxeles de la imagen secreta para este tramo de tamaño k. Luego evalúa el polinomio en un punto por cada imagen portadora, los pares x se sacan del píxel superior izquierdo en cada grupo de 4 pixeles y se pasa por el polinomio. El y = f(x) resultante se oculta particionado en los bits menos significativos de los otros 3 pixeles del grupo, en un esquema (k,n) se van a calcular n puntos, por cada sección de tamaño k. Luego solo hace falta conseguir k puntos para poder interpolar el polinomio ya que este es de grado k-1.

Sabiendo esto, es lógico pensar que la cantidad de información que se puede guardar depende linealmente de la cantidad de píxeles que tienen las imágenes portadoras. ya que se puede guardar k píxeles en cada grupo de tamaño 4, solo que cuanto mayor sea el k, mas imágenes harán falta como mínimo para poder luego reconstruir la información.

Otra caracteristica que se puede destacar de este metodo es que si la imagen tiene mas colores que solo el blanco y negro, se podra guardar mas informacion en cada pixel. por lo que también hay una relación lineal conforme a cuantos colores tienen las imágenes. sabiendo esto, podemos deducir que se podrán guardar como máximo los bytes que indique la ecuación:

$$Payload(byte) \quad = \quad \frac{NPixeles * Ncolor * k}{4}$$

Donde NPixeles es el número de píxeles y Ncolor es el número de colores que contienen las imágenes camuflaje. De esta forma si se quiere guardar Payload(byte) se necesita encontrar imágenes portadoras que tengan una combinación de píxeles y colores que hagan que la ecuación sea verdadera (o por lo menos que tengan menos información que la que da Payload(byte)) además de elegir un k adecuado.

3. Ventajas y Desventajas de GF(2^8) respecto de Módulo

Trabajar con un campo de Galois permite tener una representación binaria de los elementos del campo, esto simplifica las operaciones dentro del campo ya que la suma se reduce a un XOR entre los bits de 2 elementos del campo.
Por otro lado, la multiplicación es más compleja ya que hay que multiplicar y reducir. Sin embargo, si mantenemos el polinomio generador constante, podemos hacer esta operación rápidamente con una tabla de valores

Adicionalmente se indica en el paper que:

“Para incrementar la carga útil de los datos secretos, la interpolación de Lagrange se resuelve de manera iterativa bajo un campo GF(28). Los resultados muestran que el sistema propuesto presenta una mayor carga útil comparado con los resultados de otros métodos propuestos previamente. Además, la calidad de las imágenes camuflaje es mayor que en los algoritmos convencionales cuando la misma cantidad de datos secretos es compartida.”

4. Cambiando el polinomio generador

Se puede trabajar con otro polinomio generador pero el polinomio que se utilice en el campo de Galois $GF(2^n)$ debe ser un polinomio irreducible de grado n, con un número impar de términos y debe tener una constante.
El polinomio podría ser otro parámetro que se le pase al programa y cambiaría la forma en la que se encripta y se recupera la información. En ese caso el polinomio generador se comportaría como una clave.

5. Guardado de distintos tipos de secreto

El algoritmo almacena en las imágenes portadoras información binaria. No discrimina si mi secreto es una imagen, un documento, un ejecutable, un audio o un video. Se podría guardar cualquiera de estos fácilmente escondiendo el binario del archivo en las imágenes.

Sin embargo, se debe considerar que la cantidad de información que yo quiero esconder debe cumplir la ecuación de la carga útil, explicada en la sección 2.

Cabe aclarar que el hecho de estar trabajando con GF(2^8) nos da mucha facilidad para trabajar con bits y ya que como la información de cualquier archivo se puede descomponer como un array de bits esto genera que el algoritmo pueda trabajar con cualquier archivo sin discriminar el tipo del mismo. Y pueda efectuar operaciones de suma y multiplicación en campos de Galois fácilmente.

6. Cómo podríamos guardar un archivo de imagen completo

Ya que el algoritmo está preparado para guardar cualquier tipo de archivo, se podría simplemente modificar el algoritmo para que guarde toda la información de la imagen en otras imágenes, incluido el header. La única consideración que habría que tener es que las imágenes portadoras deberán cumplir con la ecuación de la carga útil, que se explica en la sección 2.

7. Usando imágenes a color

Si se quisieran utilizar imágenes de a color, habría que modificar la forma en la que se distribuye la información de las secciones de tamaño k. Cada píxel de esta sección de la imagen secreta seguiría correspondiendo a un coeficiente del polinomio que se genera con esta sección, Además también se tendrían que generar n puntos tomando n valores de x y calculando n valores de y (un punto por cada imagen portadora por cada sección).

La principal diferencia sería en cómo se guardará la información de de el valor y luego de ser pasado por el polinomio (y = f(x)) ya que ahora cada pixel de la imagen portadora cuenta con 24 bits de información en vez de 8 (correspondiente a 3 bytes, uno por cada color (RGB))

Sabiendo que ahora cada pixel dispone de más información se podría distribuir los el valor de la siguiente forma:

1. Se toma un bloque de 4 pixeles de la imagen portadora i (con i = 1...n)
2. este bloque de 4 pixeles contiene X, W, V, U, solo que ahora cada uno de estos pixeles tiene 3 canales, el canal R el G y el B
3. se toma el canal R de X (denominado Xr) y se lo pasa por la función polinómica
4. el resultado (Yr) se descompone en 3 pedazos al igual que antes, solo que estos pedazos ahora se guardan en los 3 bits menos significativos del canal R de W, V y U (denominados Wr, Vr, y Ur)
5. De esta forma, si se repite el mismo proceso en los otros canales (G y B) se pueden almacenar 3 segmentos de tamaño k (equivalentes a 3 polinomios) en cada bloque de 4 pixeles, uno por cada canal (RGB)

De esta forma se guarda el triple de información por cada segmento de tamaño k de datos y el resto del algoritmo funciona de una forma muy similar al algoritmo original, solo que ahora a la hora de interpolar los polinomios hay que tener la forma en la que se distribuyen los segmentos.

8. Cómo tomamos los bloques

Los bloques se podrían tomar de muchas formas distintas, la forma que propone el paper permite solamente afectar los tres bits menos significativos de cada bloque, pero por ejemplo se podrían tomar bloques de tamaño tres y almacenar en cada uno un segmento, donde uno de los bloques se utilizan como x y dos de los bloques se utilizan como y (la mitad del y en los cuatro bits menos significativos de cada pixel)

También se podría hacer lo inverso y tomar bloques de tamaño 5 donde 1 se utiliza para almacenar el x y los otros cuatro se utilizan para almacenar en sus dos bits menos significativos, los cuatro pedazos de y

Cabe aclarar que estas dos opciones son mucho más difíciles de utilizar para cubrir toda la imagen ya que no son cuadradas y además no toman en cuenta el bit de paridad, pero solo se proponen como ejemplo. la realidad es que podrás tomar un montón de distintas formas y tamaños distintos para los bloques y cada una tendría ventajas y desventajas.

9. Aspectos de la implementación

9.1. Facilidad de implementación

La dificultad de la implementación se redujo considerablemente cuando comenzamos a utilizar tablas para la multiplicación de Galois y para los inversos multiplicativos de la interpolación de Lagrange.

La funcionalidad que más nos costó implementar fue por diferencia la interpolación de lagrange. Sin embargo, el resto de las funciones y la estructura general del trabajo no presentaron una gran dificultad.

9.2. Posibilidad de extender el algoritmo o modificarlo

Podríamos hacer que el programa reciba el polinomio generador como argumento y deberíamos generar las tablas para la multiplicación de Galois y el inverso multiplicativo en la interpolación de Lagrange en cada caso. O trabajar sin el uso de tablas (lo que consumiría menos memoria pero requeriría de un esfuerzo computacional mucho mayor)

Por otro lado también se podría expandir el proyecto para que no solo pueda ocultar imágenes sino cualquier archivo ya sea de texto, imagen o hasta ejecutables ya que el algoritmo del paper lo permite, además se podría expandir el sistema para que pueda ocultar en imágenes a color de distintos tamaños.

10. Aplicaciones del algoritmo

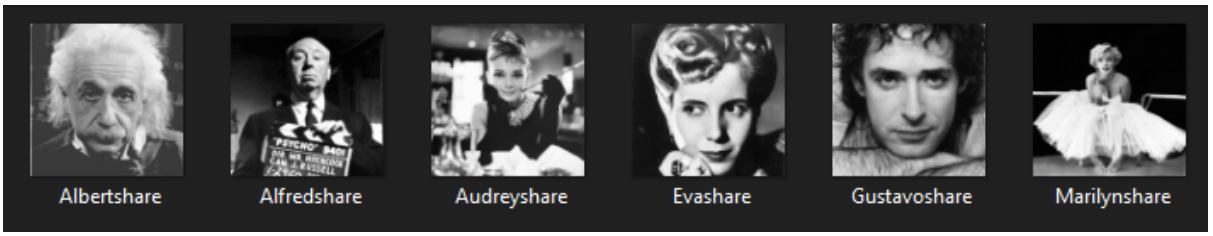
Este algoritmo es muy útil cuando se quiere distribuir información en n partes, donde solamente hace falta k para poder conseguir esa información, por ejemplo esto se podría utilizar para distribuir una llave importante entre los gerentes de una empresa para que solo teniendo k llaves dentro de las n se pueda utilizar esta llave.

Secreto de la cátedra

Después de correr el siguiente comando:

```
./main r output/secret.bmp 6 sharesg13/
```

Donde la carpeta “sharesg13” contenía los siguientes archivos provistos por la cátedra:



Se obtuvo la siguiente imagen secreta: