Esteganografía LSB en Imágenes BMP

La esteganografía digital es la disciplina que estudia técnicas para ocultar información dentro de objetos digitales como imágenes, audio o video, de manera que la existencia del mensaje permanezca oculta. A diferencia de la criptografía, que protege el contenido, la esteganografía busca ocultar la presencia misma del mensaje.

El método utilizado en este proyecto es LSB (Least Significant Bit), que consiste en modificar el bit menos significativo de cada componente de color (RGB) de un píxel. Esta modificación produce cambios mínimos en el color, generalmente imperceptibles al ojo humano.

Se utiliza el formato BMP sin compresión debido a que permite acceder directamente a los valores de los píxeles sin transformaciones que alteren los bits. En imágenes RGB de 24 bits, cada píxel contiene tres canales (Rojo, Verde y Azul) con 8 bits cada uno, permitiendo almacenar hasta 3 bits por píxel.

Nota: Los primero 54 bytes de la imagen no deben de usarse.

Las operaciones binarias utilizadas incluyen:

- AND (&): Para enmascarar bits (ejemplo: valor & 254 limpia el LSB).
- OR (|): Para establecer un bit específico.
- Desplazamiento a la derecha (>>): Para extraer componentes RGB.
- Desplazamiento a la izquierda (<<): Para reconstruir valores o desplazar bits.

Cada carácter se representa en 8 bits según el código **ASCCI**. Por ejemplo:
'A' = 65 = 01000001 en binario.

Para insertar un bit en el LSB:
1. Se limpia el LSB con: valor & 254 (11111110 en binario).

2. Se inserta el bit con: (valor & 254) | bit.

Capacidad máxima:
Si la imagen tiene W x H píxeles:
Capacidad en bits = W * H * 3
Capacidad en bytes = (W * H * 3) / 8
Restando 2 bytes para delimitadores.

Método cifrar (File bmp, File txt)

Proceso:

1. Se lee la imagen usando BufferedImage.
2. Se lee el contenido del archivo TXT.
3. Se agregan delimitadores, con caracteres poco comunes ASCII 219 (11011011 en binario) al inicio y final.
4. Se recorre la imagen por coordenadas (x,y).
5. Se extraen R, G, B usando desplazamientos.
6. Se inserta cada bit del mensaje en el LSB de los canales RGB.
7. Se reconstruye el nuevo valor RGB.
8. Se guarda la imagen como "imagen_oculta.bmp".

Ejemplo binario:
219 = 11011011
'H' = 01001000
'i' = 01101001
Cada píxel almacena 3 bits en orden R, G, B.

Método descifrar (File bmp)

Proceso:

1. Se lee la imagen.
2. Se extrae el LSB de cada canal RGB.
3. Se reconstruyen caracteres cada 8 bits.
4. Se detecta el delimitador 219 como inicio y fin.
5. Se usa StringBuilder para reconstruir el mensaje.
6. Se escribe el archivo "contraseña_descifrada.txt".

Reconstrucción:

caracter = (caracter << 1) | bit

Ventajas

- Imperceptibilidad visual: al modificar solo los LSBs, las alteraciones en color son mínimas y normalmente indetectables para el ojo humano.
- Simplicidad de implementación: algoritmo directo, fácil de comprender e implementar con operaciones bit a bit.
- Bajo coste computacional: operaciones aritméticas y bitwise son rápidas; complejidad temporal lineal.

Limitaciones

- Ausencia de cifrado criptográfico. El mensaje se inserta en texto plano (aunque dentro de la imagen); cualquiera con el método puede recuperar el texto. Se recomienda cifrar el mensaje antes de insertar (AES, ChaCha20) si la confidencialidad es importante.
- Vulnerabilidad ante compresión o edición de imagen. Formatos con compresión con pérdida (JPEG), reescalado o filtrado alteran los bits y destruyen el mensaje. Incluso ciertas herramientas de edición sin pérdida pueden reescribir mapas de bits y causar pérdidas.
- Limitación de capacidad. La cantidad máxima de información está limitada por floor((W*H*3)/8). Mensajes largos requieren imágenes grandes.
- Posible detección por análisis estadístico. Técnicas de esteganálisis (chi-square, RS analysis, análisis de histograma, detección de patrones LSB) pueden

identificar anomalías estadísticas en los LSBs. La inserción secuencial puede ser más detectable que una inserción pseudoaleatoria.

- Otros riesgos y defectos del código actual. No chequeo explícito de capacidad: el código no informa si la imagen es insuficiente para contener todo el mensaje; mejor calcular capacidad antes y abortar con mensaje claro. Uso de delimitador 219: si el texto contiene ese carácter la delimitación falla o requiere escape. Es más robusto almacenar la longitud del mensaje al inicio (por ejemplo, en los primeros N bits). Codificación: Files.readString y Files.writeString usan la codificación por defecto del sistema: conviene especificar UTF-8 para evitar problemas con caracteres no ASCII.

El método LSB implementado permite ocultar información dentro de imágenes BMP de manera sencilla y eficiente. Es adecuado para fines académicos y demostrativos. Para aplicaciones reales de seguridad, se recomienda añadir cifrado previo, uso de clave y dispersión pseudoaleatoria.

Autor

Abraham Mora
Estudiante de Ingeniería en Sistemas Computacionales – IPN