Documentación Image Lockdown

Introducción

En las últimas semanas, nuestro equipo se ha enfocado en desarrollar una solución que permita la encriptación y desencriptación de imágenes satelitales, garantizando una transmisión segura y eficiente. Este informe detalla el proceso seguido, las herramientas utilizadas, la solución implementada y los resultados obtenidos.

Herramientas Utilizadas

Para abordar el problema, recurrimos a diversas herramientas que facilitaron el desarrollo y optimización de nuestra solución. A continuación, se describen las principales herramientas empleadas:

OpenSSL

Utilizamos OpenSSL y su documentación extensiva para implementar los algoritmos de cifrado y descifrado. Esta herramienta nos proporcionó las librerías necesarias para trabajar con criptografía de alto nivel.

Valgrind

Valgrind fue crucial para detectar y corregir fugas de memoria en nuestro código. Gracias a esta herramienta, podemos asegurar una gestión eficiente de los recursos de memoria, evitando posibles errores y optimizando el rendimiento.

Comando `time` de Unix

Empleamos el comando `time` de Unix para medir el tiempo de ejecución de nuestras funciones. Esto nos permitió realizar pruebas de rendimiento y ajustar nuestro código para mejorar la eficiencia.

gdb

Utilizamos gdb como depurador para identificar y eliminar bugs en nuestras funciones. Con esta herramienta, pudimos optimizar el código y asegurar su estabilidad y fiabilidad.

Explicación de la Solución

Tipo de Cifrado y Llaves Dinámicas

Nuestra solución se basa en el algoritmo AES-CTR-256. El proceso de cifrado y descifrado se realiza de la siguiente manera:

Encriptar:

1. Se realiza una generación aleatoria del IV (de 16 bytes) y de la llave AES (de 32 bytes).
2. Se guarda el IV en los primeros 16 bytes del archivo de salida.
3. La llave AES se divide en 4 partes.
4. Cada una de las 4 partes de la llave se reordena.
5. Se guarda la llave reordenada en los siguientes 32 bytes del archivo de salida.
6. Para evitar llenar la memoria, el archivo se procesa en buffers.
7. El archivo cifrado se va guardando en el archivo de salida de manera progresiva hasta completar la encriptación de todo el archivo.

Desencriptar:

1. Se extrae el IV de los primeros 16 bytes del archivo de entrada.
2. Se extraen los siguientes 32 bytes que contienen la llave.
3. Los 4 fragmentos de la llave se reordenan para obtener la llave original.
4. Con la llave ya restaurada, se procede a desencriptar el archivo por buffers para evitar llenar la memoria.
5. El archivo desencriptado se va procesando y guardando en el archivo de salida de manera progresiva hasta completar todo el archivo.

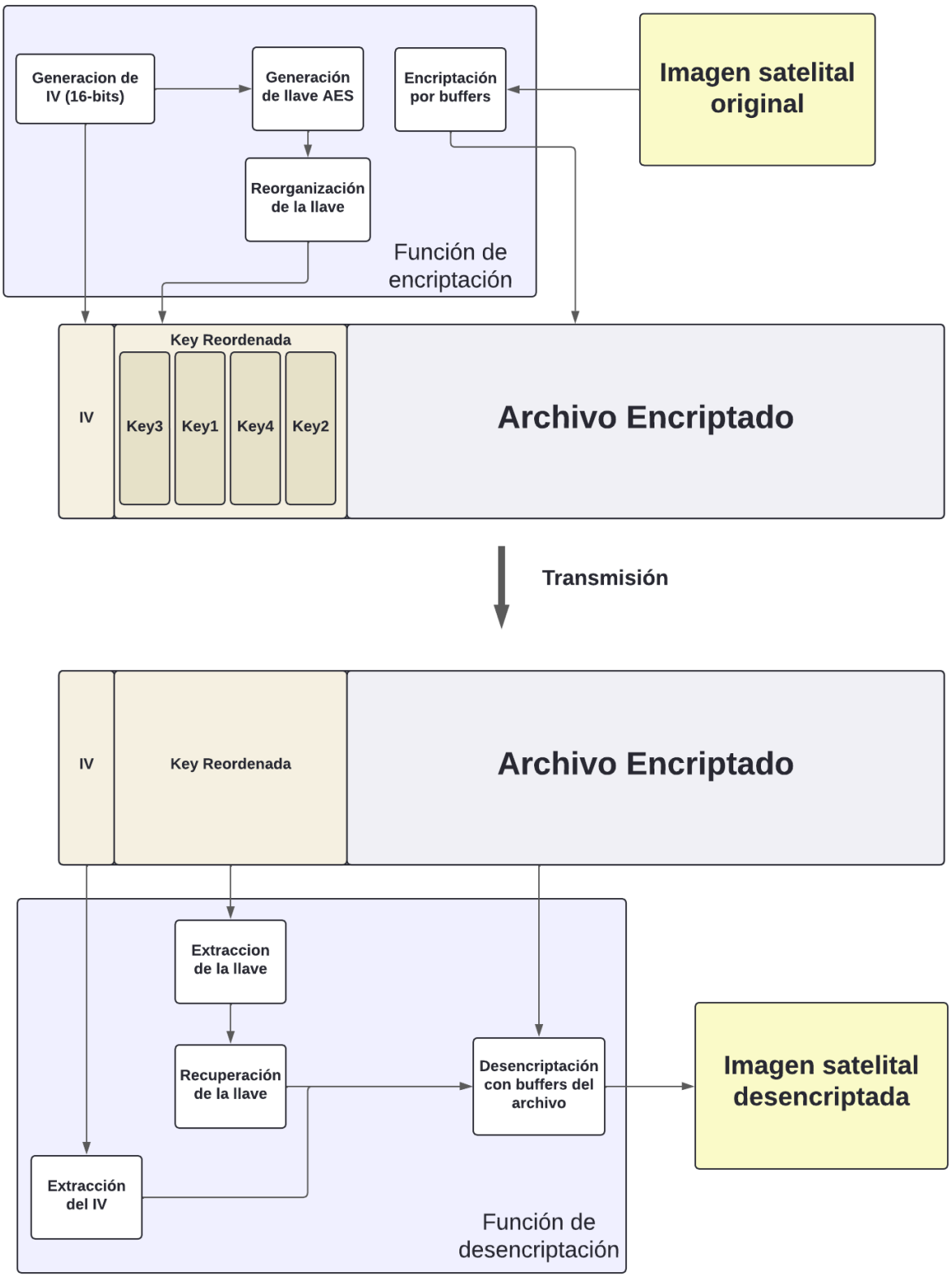

Fig 1. Esquema de funciones implementadas

Con estos pasos, en caso de que la señal sea interceptada, solo nosotros poseemos el conocimiento necesario para entender la estructura y organización interna de los datos almacenados en el archivo. La forma en que la llave ha sido fragmentada, reordenada y distribuida permanece oculta para cualquier tercero, lo que añade una capa adicional de seguridad. Incluso si alguien tuviera acceso al archivo cifrado, sin conocer la disposición exacta de la llave y cómo fue manipulada durante el proceso de encriptación, sería prácticamente imposible reconstruir la información original o acceder al contenido de manera no autorizada. Esto garantiza que nuestros datos permanezcan seguros y protegidos frente a cualquier intento de interceptación o vulneración.

Uso del Buffer y Pre-asignación de Memoria

Para manejar archivos grandes y evitar problemas de memoria, implementamos un buffer de 1MB. El código obtiene el tamaño del archivo y pre-asigna la memoria necesaria. Esta técnica reduce significativamente el tiempo de ejecución y asegura la eficiencia del programa, evitando sobrecargas y mejorando el rendimiento en sistemas con recursos limitados.

Tests Llevados a Cabo y Manejo de Errores

Verificación del algoritmo

Para verificar la efectividad de nuestras funciones, utilizamos el algoritmo **MD5** para comparar la imagen de entrada con la imagen desencriptada. Este enfoque nos permitió verificar la integridad de los datos sin necesidad de revisiones visuales. Al comparar los códigos MD5 generados, pudimos confirmar que la imagen desencriptada era idéntica a la original, asegurando así el correcto funcionamiento de nuestro sistema.

Conclusión

En conclusión, hemos desarrollado una solución eficiente y segura para la encriptación y desencriptación de imágenes satelitales. Utilizando herramientas como OpenSSL, Valgrind, el comando `time` de Unix y gdb, pudimos crear un sistema robusto que garantiza la integridad y seguridad de los datos transmitidos. Estamos ansiosos por recibir su feedback y responder cualquier pregunta que puedan tener.

Referencias

https://www.openssl.org
https://www.geeksforgeeks.org/rsa-algorithm-cryptography/?ref=gcse_ind
https://www.tutorialspoint.com/advanced-encryption-standard-aes