CODENEAS

Nicolás Bedoya Figueroa
Manuela Pacheco Malagón
Nicolas Rozo Fajardo
Luis Felipe Torres Galvis

CODEFEST AD ASTRA 2024 FASE FINAL:

Solución de Cifrado para Comunicación Satelital

CIFRADO
DESCIFRADO
USO DE LLAVES DINÁMICAS
GESTIÓN DE MEMORIA EN SISTEMA EMBEBIDO
LIBRERÍAS UTILIZADAS
VERIFICACIÓN Y VALIDACIÓN
REFERENCIAS

CIFRADO

Cumpliendo con los estándares de cifrado para misiones espaciales[3], el cifrado de las imágenes se lleva a cabo haciendo uso del esquema de cifrado AES en modo CTR (counter mode). Con el fin de mejorar el rendimiento del programa y facilitar la implementación, se hace uso de las funcionalidades de la librería Crypto++ y se configura tanto la llave criptográfica como el vector de inicialización haciendo uso de los resultados generados por la estrategia de generación de llaves dinámicas propuesta. Adicionalmente al cifrado de la imagen, el programa calcula el hash de la imagen original usando el algoritmo SHA-256 y lo almacena para su posterior uso en la verificación de integridad de la imagen descifrada.

Es importante destacar que el archivo generado (ya sea .tif o .tiff) no puede ser visualizado debido a que, con el fin de aumentar la seguridad, nuestro programa cifra la totalidad del archivo de entrada, lo cual implica que los headers y etiquetas necesarias para visualizar la imagen se encuentran cifradas. Esta decisión se tomó con el fin de aumentar la confidencialidad ya que, si no se cifraban los headers y etiquetas, sería posible consultar datos sensibles como el título de la imagen o su autor haciendo uso de la imagen resultante del cifrado. Aun así, se puede verificar que la imagen se cifró correctamente ya sea utilizando el modo de descifrado del programa o calculando su hash.

DESCIFRADO

El descifrado de las imágenes se lleva a cabo haciendo uso del esquema de cifrado AES en modo CTR (counter mode) y utilizando las funcionalidades de la librería Crypto ++. Aún así, cabe aclarar que antes de que inicie el proceso de descifrado, el programa se encarga de recuperar tanto la semilla como el hash de la imagen con el fin de reconstruir tanto la llave dinámica como el vector de inicialización, siguiendo el esquema presentado en la estrategia de generación de llaves dinámicas.

Una vez se completa el proceso de descifrado de la imagen, el programa calcula el hash de la imagen obtenida haciendo uso de la función SHA-256 y lo compara con el hash calculado de la imagen original para revisar la integridad de la imagen descifrada. En caso de que los valores no coincidan, el programa informa al usuario que la integridad de la imagen se vio comprometida.

USO DE LLAVES DINÁMICAS

El objetivo principal de la implementación de llaves dinámicas es definir un método que genere llaves de manera continua y eficiente de tal manera que sean difíciles de invertir, logrando un equilibrio entre seguridad y desempeño. A su vez, una correcta generación de llaves dinámicas

garantiza que si una o más llaves se ven comprometidas, el resto de la secuencia permanecerá segura.

La estrategia propuesta integra conceptos de probabilidad, generación de números aleatorios y uso de funciones de hash para garantizar que las llaves generadas son seguras y pueden ser construidas o recuperadas tanto por la estación terrena como por el satélite. A continuación, se explica de forma detallada el proceso de generación de llaves.

- Al usar el modo de encriptado del programa, este genera la semilla que servirá como base para el proceso de construcción de la llave. Para esto, se obtiene el tiempo actual del sistema y se construye un generador de número pseudoaleatorios usando el valor obtenido. Posteriormente, se construye una distribución de probabilidad uniforme (con rango 0 a 9999) haciendo uso del generador de números pseudoaleatorios. Finalmente, se genera un número haciendo uso de la distribución de probabilidad creada y a este se le aplica la función SHA-256 para obtener una cadena alfanumérica de 32 bytes, la cual actuará como semilla en el proceso.

- Posteriormente, se aplica una función de hash numérica a la semilla para obtener un número y, con este, construir nuevamente un generador de número pseudoaleatorios. Al igual que en el paso anterior, este generador se utiliza para construir una distribución de probabilidad uniforme la cual es utilizada para generar cadenas alfanuméricas pseudoaleatorias de 32 bytes.

- Haciendo uso de la funcionalidad de construcción de cadenas alfanuméricas pseudoaleatorias, el programa construye 5 llaves temporales, es decir, 5 cadenas alfanuméricas de 32 bytes y, posteriormente, construye la llave semilla la cual es el resultado de unir las 5 llaves temporales haciendo uso del operador lógico XOR.

- Finalmente, se construye la llave criptográfica final aplicando la función SHA-256 a la llave semilla generada en el paso anterior. Esta llave es la que se usará tanto en el proceso de cifrado como de descifrado.

A su vez, con el fin de aumentar la seguridad del sistema y la calidad del cifrado, el vector de inicialización también es generado de forma dinámica bajo la siguiente estrategia.

- Una vez generada la semilla que se menciona en el paso 1 de la estrategia de generación de llaves dinámicas, se hace uso de la funcionalidad de construcción de cadenas alfanuméricas pseudoaleatorias y se genera una cadena alfanumérica de 16 bytes. Esta cadena actúa como vector de inicialización en el proceso de encriptado y desencriptado.

Esta estrategia resulta segura y efectiva ya que, en un entorno de comunicación real, solo debería comunicarse la semilla (cadena alfanumérica de 32 bytes) entre el satélite y la estación

terrena y, a partir de esta, ambos actores podrían reconstruir tanto la llave como el vector de inicialización. A su vez, esta estrategia garantiza un mayor nivel de seguridad ya que, aunque se comprometa la semilla, el atacante debería tener conocimiento de la estructura del código para poder replicarla.

Ahora bien, dado que el algoritmo y estrategia planteados están diseñados para simular un escenario de ejecución real, se tomó la decisión de compartir tanto la semilla como el hash de la imagen al mismo tiempo que se transfiere la imagen. A continuación, se presentan los pasos que garantizan la seguridad y éxito asociada al intercambio de la semilla:

1. Al momento de usar la funcionalidad de cifrado de imagen, se generan un par de llaves asimétricas (llave pública y privada) de 32 bytes las cuales son almacenadas en dos archivos dentro del sistema. Cabe resaltar que el almacenamiento de estas llaves se realiza para simular la situación de la vida real en la cual las dos partes conocen sus llaves y las tienen almacenadas de forma segura.

2. Una vez creadas las llaves, luego de encriptar la imagen original se calcula su hash y se crea un paquete concatenando la llave con el hash calculado. Este paquete es cifrado con la llave pública generada y se concatena al final del vector que representa la imagen cifrada para su posterior almacenamiento, el cual, en un escenario real representa el envio de la información desde el satélite hasta la estación terrena.

3. Al momento de usar la funcionalidad de descifrado, se recupera el paquete concatenado al final del vector y, haciendo uso de la llave privada (la cual solo conoce la estación terrena) se recupera tanto la semilla como el hash de la imagen original. Una vez obtenidos estos valores, se utiliza el algoritmo propuesto para reconstruir tanto la llave simétrica como el vector de inicialización y se descifra el contenido de la imagen.

4. Finalmente, se calcula el valor de hash de la imagen descifrada y se compara con el valor recuperado del paquete. Esto permite revisar la integridad de la imagen y, en caso de que existan, detectar posibles errores.

Se puede afirmar que la estrategia usada para compartir la semilla y el hash calculado es efectiva y segura ya que la información enviada se encuentra cifrada y sólo puede ser descifrada por la estación terrena (garantizando autenticación), se hace uso de la imagen para transmitir la información mitigando los problemas de comunicación y se entregan de forma segura los datos que permiten reconstruir los parámetros necesarios para desencriptar una imagen exitosamente.

GESTIÓN DE MEMORIA EN SISTEMA EMBEBIDO

Uno de los principales desafíos consistía en la lectura y procesamiento de imágenes de gran tamaño en un ambiente con recursos limitados, esto es, en un ambiente en el que el tamaño de la imagen es mayor a la capacidad de la memoria RAM. Para abordar esto, se implementó un proceso de carga de imágenes por chunks o fragmentos que asegura una manipulación eficiente y segura de estos volúmenes de datos, garantizando que se cumpla con las restricciones de memoria del sistema embebido y que se preserve la integridad de los datos de la imagen.

El proceso inicia con la apertura de un flujo hacia los archivos de entrada y salida en modo binario, evitando así el uso de librerías externas y la alteración de los datos durante la lectura y escritura. Con el fin de manejar la memoria adecuadamente, se utiliza un buffer para la lectura de datos, procesando la imagen en bloques del tamaño del buffer hasta alcanzar el final del archivo. Cada bloque de datos leído se cifra o decifra, y se almacena en un vector temporal, para posteriormente ser escrito en el archivo de salida.

Luego de realizar diversas pruebas de rendimiento y uso de memoria, se optó por un tamaño de chunk o buffer de 1 GB para garantizar que el uso total de memoria del programa, el cual también incluye el vector temporal y las estructuras de datos auxiliares generadas por la librería criptográfica, se mantenga por debajo de los 2 GB, cumpliendo con el límite de 4 GB del sistema satelital, y permitiendo la lectura eficiente de una cantidad significativa de datos en cada operación.

LIBRERÍAS UTILIZADAS

Crypto++

Crypto++ es una librería de código abierto en C++ de algoritmos y esquemas criptográficos. Su elección se basó en:

- **Flexibilidad:** Su variedad de algoritmos facilita la adaptación y prueba de diferentes enfoques criptográficos. Permitió cambiar el modo de cifrado a Counter, que es el recomendado en los estándares para cifrado en misiones espaciales. [3]
- **Documentación y Comunidad:** Cuenta con extensa documentación y una comunidad activa, facilitando la implementación y resolución de problemas.
- **Rendimiento optimizado:** Uso eficiente de la memoria, ideal para nuestros requisitos de sistemas embebidos

Crypto++ está distribuida bajo la **Boost Software License 1.0**.[2]

Librería Estándar de C++

Fundamental para el desarrollo de cualquier aplicación en C++, proporciona una colección robusta de funcionalidades. Se utilizó para entrada y salida estándar, manipulación de cadenas, estructuras de datos, generación de números aleatorios y manipulación de datos y de archivos.

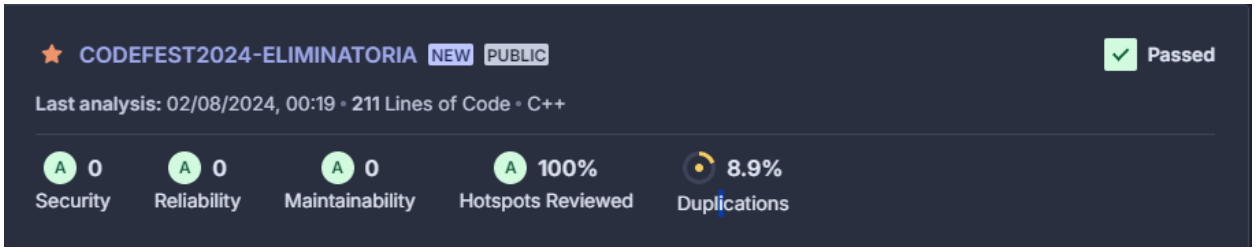
VERIFICACIÓN Y VALIDACIÓN

Para la verificación y validación de código se decidió hacer uso de la herramienta SonarCloud, puesto que esta permitía emparejar el proyecto de forma sencilla, a la par que brinda un análisis significativo. Bajo esta premisa se creó una Quality Gate para medir diversos atributos, como se puede evidenciar a continuación.

Métrica	Operador	Valor
Calificación de mantenibilidad	es peor que	A
Calificación de confiabilidad	es peor que	A
Puntos de acceso de seguridad revisados	es menor que	100%
Clasificación de seguridad	es peor que	A

Estas métricas definieron la puerta de calidad para la evaluación del código generado. En ese orden de ideas si, por ejemplo, se tiene un punto de acceso de seguridad que presente una vulnerabilidad, la calificación de esa métrica sería menor al 100%, haciendo que falle la puerta de calidad. Es decir, se deben de cumplir absolutamente todas las métricas definidas para aprobar.

Cabe aclarar que aunque usualmente se añade la métrica de duplicación de código a la puerta de calidad, en este caso no se añadió debido a que hay fragmentos de código que fueron duplicados para mejorar el rendimiento del programa y garantizar una mayor seguridad y confidencialidad en los proceso de encriptado, desencriptado, generación de llave y generación de hash. Aún así, aunque la métrica no se incluyó en la puerta de calidad, su valor es menor al 10%, que definimos como el umbral aceptable de duplicación. Los resultados de la evaluación de calidad del código se presentan a continuación.



En la esquina superior derecha se puede observar que la puerta de calidad fue aprobada correctamente, puesto que no se presenta ningún problema de seguridad, confiabilidad y mantenibilidad, además de que todos los puntos de acceso son seguros. Como se mencionó previamente, la métrica de duplicaciones no fue tomada en cuenta por las razones expuestas.

REFERENCIAS

[1] Ngo, Huy & Wu, Xianping & Le, Phu & Campbell, Wilson & Bala, Srinivasan. (2010). Dynamic Key Cryptography and Applications. International Journal of Network Security. 10.

[2] Boost Software License - Version 1.0 - August 17th, 2003. https://www.boost.org/LICENSE_1_0.txt

[3] CCSDS 350.9-G-2 INFORMATIONAL REPORT. (2023). *CCSDS REPORT CONCERNING CRYPTOGRAPHIC ALGORITHMS*. https://public.ccsds.org/Pubs/350x9g2.pdf