Reporte Reto 1 Codefest AD Astra 2024: Ciberseguridad en sistemas satelitales de observación de la Tierra

Grupo:
QuantumCommandos

Integrantes:
Santiago Castro Duque - Universidad de los Andes
Sebastian Pedraza Perez - Universidad de los Andes
Alejandro Jaramillo Castellanos - Universidad de los Andes
Jose David Florez Ruiz - Universidad de los Andes

Fecha:
2 de Agosto de 2024

Tabla de Contenidos

1. Introducción

El reto CODEFEST AD ASTRA 2024 busca desarrollar una solución de seguridad para proteger la transmisión de datos desde satélites de observación terrestre a estaciones en la Tierra. Este desafío es crucial, ya que la información satelital es valiosa para una variedad de aplicaciones, incluyendo el monitoreo ambiental, la gestión de recursos y la defensa nacional. En este contexto, la seguridad de los datos es primordial para evitar accesos no autorizados y garantizar la integridad de la información. Los participantes deben implementar un sistema que garantice la confidencialidad de los datos a través de técnicas de cifrado, asegurando que los datos sensibles sean inaccesibles para cualquier entidad no autorizada.

La estrategia de solución adoptada emplea el algoritmo de cifrado AES-CTR de 128 bits, conocido por su eficiencia y seguridad en la protección de datos. Este algoritmo se complementa con el uso de HKDF para generar llaves criptográficas dinámicas y únicas para cada sesión de comunicación. Para manejar de manera eficiente el uso de memoria en sistemas embebidos, se utiliza una técnica de "chunking", que procesa los datos en bloques de tamaño fijo, permitiendo la gestión de grandes volúmenes de datos sin requerir una gran cantidad de memoria RAM. Esta combinación de tecnologías asegura un cifrado robusto y una gestión eficiente de recursos, proporcionando una solución segura y práctica para la transmisión de datos críticos en misiones satelitales.

2. Estrategia de Cifrado

La estrategia de cifrado implementada en el proyecto se centra en asegurar la confidencialidad e integridad de las imágenes satelitales durante la transmisión. Para ello, se ha utilizado el algoritmo de cifrado simétrico AES-128-CTR, que es ampliamente reconocido por su balance

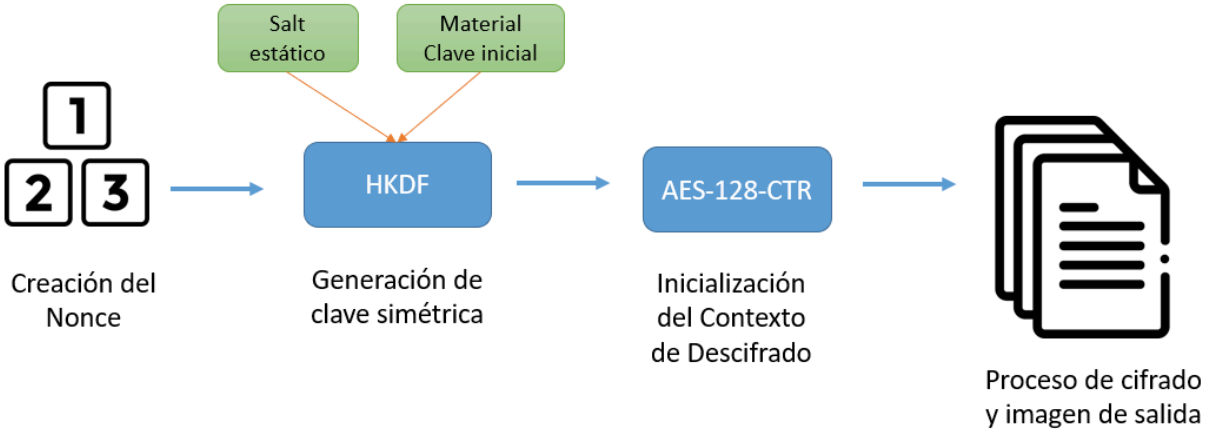
entre seguridad y eficiencia, especialmente adecuado para sistemas embebidos con recursos limitados.

Proceso de Cifrado

1. **Generación de la Clave y Nonce:** Se utiliza el mecanismo de derivación de claves HKDF (HMAC-based Key Derivation Function) para generar una clave de cifrado segura. El proceso comienza con una fase de extracción, donde se aplica HMAC-SHA256 utilizando una sal estática y un material clave inicial definido, obteniendo así una pseudoclave raíz (PRK). Posteriormente, se expande esta PRK con información adicional y un nonce aleatorio generado con RAND_bytes(), resultando en la clave de cifrado final. El nonce generado es crítico para garantizar la seguridad del cifrado en modo CTR (Counter), ya que asegura la unicidad de las secuencias de encriptado.

2. **Inicialización del Cifrador:** Se configura el contexto de cifrado utilizando la clave derivada y el nonce, inicializando el cifrador AES-128-CTR. Este modo de operación se elige porque, a diferencia de otros modos como CBC, no requiere relleno (padding) y permite la encriptación de bloques de tamaño variable, lo cual es ideal para archivos de imagen.

3. **Cifrado de Datos:** El archivo de entrada, que contiene la imagen a cifrar, se procesa en fragmentos de tamaño fijo para manejar eficientemente el uso de memoria. Cada fragmento se encripta utilizando el cifrador configurado y se escribe en el archivo de salida. Antes de iniciar el proceso de cifrado, se guarda el nonce en el archivo de salida para su uso posterior durante el descifrado.



3. Estrategia de Descifrado

La estrategia de descifrado se enfoca en revertir el proceso de cifrado de manera segura y eficiente, garantizando la recuperación íntegra de las imágenes satelitales cifradas. Se utiliza el algoritmo simétrico AES-128-CTR, que asegura que los datos cifrados puedan ser descifrados correctamente solo con la clave y el nonce adecuados.

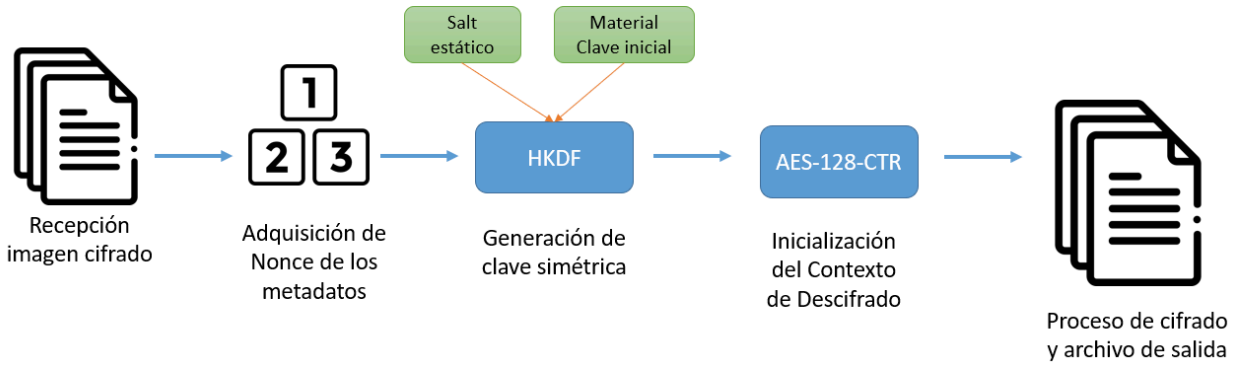
Proceso de Descifrado

1. **Extracción del Nonce:** El proceso de descifrado comienza leyendo el nonce almacenado al inicio del archivo cifrado. Este nonce es esencial para la descifrado ya que se utiliza para inicializar el vector de inicialización (IV) en el contexto del descifrado.

2. **Generación de la Clave de Descifrado:** Utilizando el mismo procedimiento que en el cifrado, se deriva la clave de descifrado. Se emplea la función HKDF (HMAC-based Key Derivation Function) con el mismo salt estático y material clave inicial, combinado con el nonce extraído. Esta clave debe coincidir exactamente con la clave utilizada en el proceso de cifrado para garantizar la correcta restauración de los datos originales.

3. **Inicialización del Contexto de Descifrado:** Se configura un contexto de descifrado utilizando el algoritmo AES-128-CTR con la clave y el IV (inicializado con el nonce). El modo CTR es elegido debido a su capacidad para manejar datos de longitud variable y su resistencia contra ciertos tipos de ataques que afectan a otros modos de operación.

4. **Proceso de Descifrado:** El archivo cifrado se lee en fragmentos. Cada fragmento es procesado mediante el contexto de descifrado, que transforma los datos cifrados de vuelta a su estado original. Estos datos descifrados se escriben en el archivo de salida especificado, garantizando la integridad y coherencia de los datos restaurados.

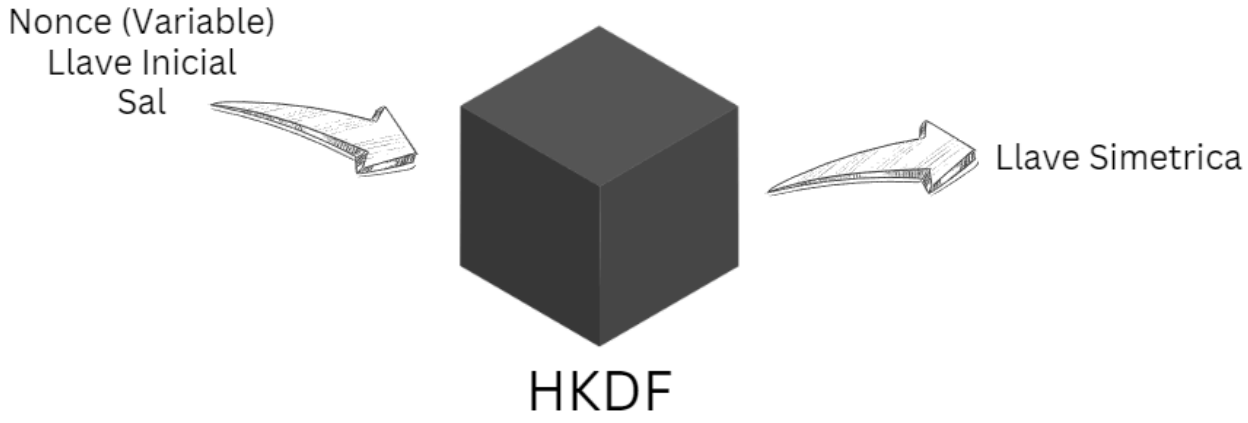


4. Estrategia para Uso de Llaves Dinámicas

En la estrategia de uso de claves dinámicas, se ha implementado un enfoque que garantiza que cada sesión de comunicación entre el satélite y la estación terrestre utilice una clave única y segura. Para ello, se emplea el protocolo de derivación de claves basado en HMAC (HKDF), un método robusto que permite generar claves criptográficas a partir de un material clave inicial, una sal y un valor compartido. Este enfoque aporta alta entropía y resistencia a ataques.

El proceso comienza con la generación de un material clave inicial (inputKeyMaterial) estático, que es una constante predeterminada específica para la misión. Este material clave se utiliza junto con una sal estática (static_hash) y un nonce generado aleatoriamente para derivar la clave de sesión única mediante HKDF. El nonce, un número único generado para cada operación de cifrado, asegura que la clave resultante sea única para cada sesión, incluso si el material clave y la sal permanecen constantes. Para facilitar el proceso de descifrado, el nonce se transmite junto con los metadatos de la imagen encriptada, permitiendo que el receptor recupere la clave de sesión adecuada y garantizando la coherencia y seguridad del proceso de descifrado. A pesar de que el hash, la sal y la clave inicial sean estáticos y estén codificados, mantenerlos confidenciales asegura la generación de una clave completamente segura basada en el nonce.

El uso de HKDF se realiza en dos fases: extracción y expansión. En la fase de extracción, se obtiene una pseudoclave raíz (PRK) utilizando HMAC-SHA256 con el inputKeyMaterial y la sal. En la fase de expansión, se genera la clave de sesión final (OKM) aplicando HMAC sobre la PRK, el nonce y cualquier otra información adicional relevante. Este proceso asegura que la clave final no sólo dependa de la clave inicial, sino también de factores dinámicos como el nonce, lo que previene la reutilización de claves y aumenta la seguridad del sistema.

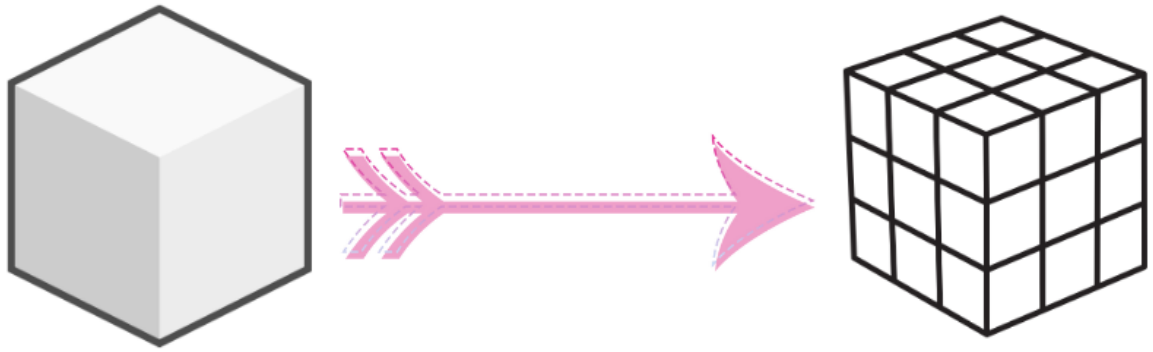


5. Estrategia para Gestión de Memoria en Sistema Embebido

La estrategia de manejo de memoria implementada en este proyecto está diseñada para ser eficiente incluso cuando el tamaño de la imagen procesada supera la cantidad de RAM disponible en el sistema embebido. Esto se logra mediante el uso de buffers de tamaño fijo, que permiten procesar la imagen en fragmentos más pequeños, en lugar de cargarla completamente en la memoria. Al dividir la imagen en porciones de 64 KB para el buffer de entrada, el sistema puede leer y procesar cada segmento de datos de manera independiente.

Durante el proceso de cifrado o descifrado, cada fragmento es leído del almacenamiento (como un disco o memoria flash), procesado y luego escrito inmediatamente al archivo de salida. Esto se realiza en un ciclo continuo hasta que toda la imagen ha sido completamente procesada. Al no intentar cargar la imagen completa en la RAM, el uso de memoria se mantiene bajo, limitado principalmente por el tamaño del buffer y las estructuras de datos auxiliares necesarias para el procesamiento.

Además, este enfoque asegura que el sistema nunca se quede sin memoria, ya que solo se utiliza una pequeña cantidad fija de RAM en cualquier momento dado, independientemente del tamaño total de la imagen. Este manejo cuidadoso de la memoria es crucial en sistemas embebidos, que a menudo tienen limitaciones estrictas de recursos. Así, esta estrategia permite procesar imágenes de tamaño arbitrario sin riesgo de agotar la memoria disponible, asegurando la continuidad y estabilidad de las operaciones criptográficas.



6. Librerías Utilizadas

- **iostream:** Proporciona funcionalidades para la entrada y salida estándar. Se utiliza para mostrar mensajes en la consola y manejar la entrada de datos de los usuarios. Fuente: Biblioteca estándar de C++. Licencia: No aplica licencia específica, ya que es parte del estándar del lenguaje C++, regulado por la especificación ISO C++.

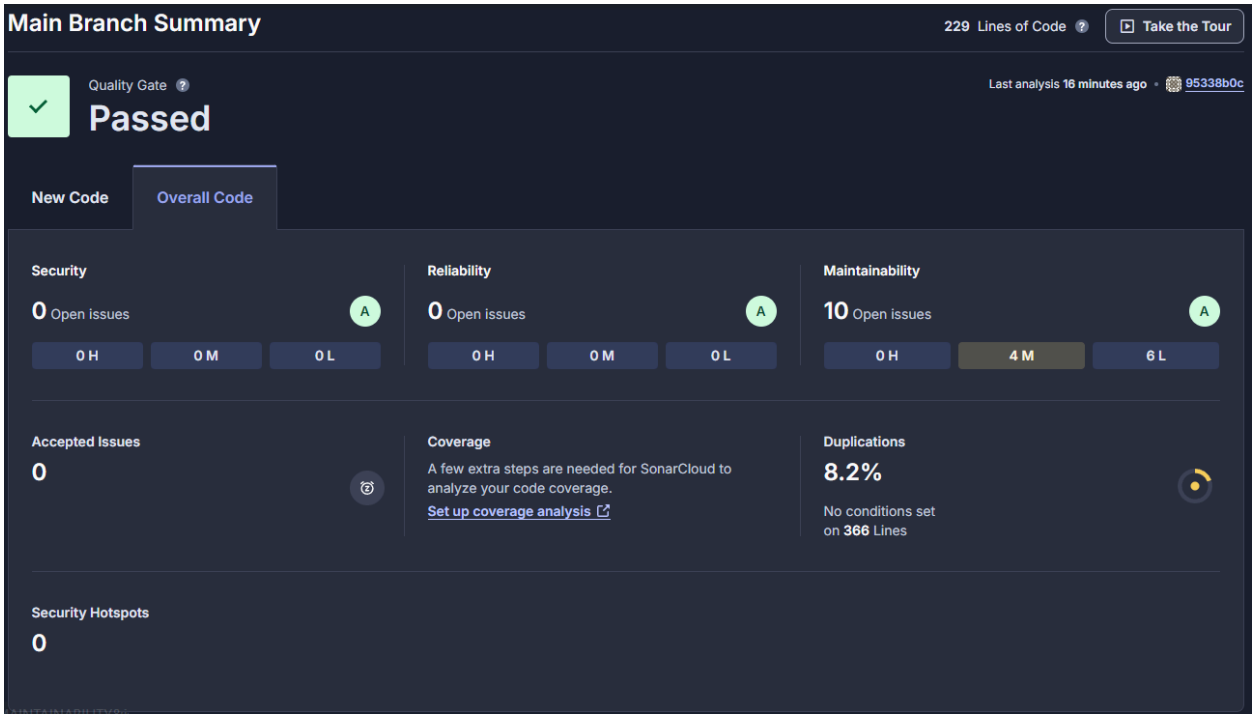
- **string:** Ofrece soporte para la manipulación de cadenas de caracteres. Es fundamental para gestionar las rutas de archivos y las operaciones con cadenas en el código. Fuente: Biblioteca estándar de C++. Licencia: Forma parte del estándar del lenguaje C++ y está regulada por la especificación ISO C++.
- **cstdlib:** Incluye funciones para realizar operaciones generales de propósito, como la gestión de memoria dinámica y la generación de números aleatorios. En este proyecto, se utiliza principalmente para manejar la terminación del programa en caso de errores. Fuente: Biblioteca estándar de C++. Licencia: No tiene una licencia específica, siendo parte del estándar ISO C++.
- **openssl/evp.h:** Parte de la biblioteca OpenSSL, proporciona una interfaz de alto nivel para las funciones criptográficas. Se utiliza para inicializar y manejar contextos de cifrado y descifrado, como en los algoritmos AES-128-CTR. Fuente: Biblioteca OpenSSL. Licencia: OpenSSL License y Apache License 2.0. Los usuarios pueden elegir entre estas dos licencias. OpenSSL License y Apache License 2.0.
- **openssl/hmac.h:** Ofrece funciones para la generación de HMAC. Es crucial para la implementación del HKDF, utilizado para derivar llaves criptográficas seguras. Fuente: Biblioteca OpenSSL. Licencia: Disponible bajo la OpenSSL License y Apache License 2.0. OpenSSL License y Apache License 2.0.
- **openssl/err.h:** Permite la gestión y reporte de errores específicos de OpenSSL. Se utiliza para capturar y mostrar errores en las operaciones criptográficas, facilitando la depuración y manejo seguro de fallas. Fuente: Biblioteca OpenSSL. Licencia: Igual que las otras cabeceras de OpenSSL, disponible bajo OpenSSL License y Apache License 2.0. OpenSSL License y Apache License 2.0.
- **openssl/rand.h:** Proporciona funciones para la generación de números aleatorios de alta calidad, esenciales para la creación de nonces encriptados. Garantiza que los números generados sean suficientemente aleatorios para mantener la seguridad criptográfica. Fuente: Biblioteca OpenSSL. Licencia: Igual que las otras cabeceras de OpenSSL, bajo OpenSSL License y Apache License 2.0. OpenSSL License y Apache License 2.0.
- **fstream:** Proporciona funciones para la lectura y escritura de archivos. Se utiliza para manejar la entrada y salida de datos binarios de archivos durante los procesos de cifrado y descifrado. Fuente: Biblioteca estándar de C++. Licencia: Parte del estándar del lenguaje C++ según la especificación ISO C++.
- **vector:** Parte de la biblioteca estándar de C++, ofrece una estructura de datos de matriz dinámica. Se utiliza para almacenar datos de longitud variable, como los buffers de datos en proceso de cifrado/descifrado y la gestión de claves derivadas. Fuente: Biblioteca estándar de C++. Licencia: No tiene una licencia específica, regulada por la especificación ISO C++.

7. Estrategia de Verificación y Validación

La estrategia de verificación y validación del proyecto se llevó a cabo mediante un conjunto de pruebas exhaustivas utilizando imágenes de gran tamaño, proporcionadas tanto por los organizadores del reto como por el equipo de desarrollo. Para automatizar el proceso de pruebas, se desarrolló un script denominado run tests.bat, que permite ejecutar de manera sistemática la encriptación y desencriptación de todas las imágenes ubicadas en la carpeta ./tests/input. El script realiza varias tareas clave: limpia las carpetas de salida ./tests/encrypted y ./tests/output, compila el código fuente, y procesa cada archivo de imagen aplicando las funciones de encriptación y desencriptación. Además, se mide el tiempo de ejecución para cada imagen, así como el tiempo total de procesamiento.

```
PS C:\Local Working\Codefest_AD_Astra-2024> .\tests\run_tests.bat
input_path=.\tests\input\PIA05566.tif
output_path=.\tests\encrypted\PIA05566.tif
Processed (encrypted) file
Encrypted image
input_path=.\tests\encrypted\PIA05566.tif
output_path=.\tests\output\PIA05566.tif
Processed (decrypted) file
Decrypted image
La duracion de la instancia para PIA05566.tif fue: 0h:0m:0s:908ms
input_path=.\tests\input\PIA12833.tif
output_path=.\tests\encrypted\PIA12833.tif
Processed (encrypted) file
Encrypted image
input_path=.\tests\encrypted\PIA12833.tif
output_path=.\tests\output\PIA12833.tif
Processed (decrypted) file
Decrypted image
La duracion de la instancia para PIA12833.tif fue: 0h:0m:0s:25ms
Comparando imagenes entre .\tests\input y .\tests\output
Los archivos .\tests\input\PIA05566.tif y .\tests\output\PIA05566.tif tienen el mismo tamanio.
Los archivos .\tests\input\PIA12833.tif y .\tests\output\PIA12833.tif tienen el mismo tamanio.
El tiempo total de ejecucion fue: 0h:0m:2s:957ms
Pruebas completadas.
```

Una parte esencial de la validación consistió en comparar el tamaño de los archivos de entrada y salida para asegurar que el proceso de cifrado y descifrado se realizó correctamente. Si los tamaños de los archivos no coinciden, se genera una alerta, indicando una posible falla en el proceso. Además, se utilizó SonarQube para analizar la calidad interna del código, logrando pasar sus pruebas básicas de calidad. A pesar de detectar 10 issues menores que no representan riesgos significativos para la seguridad o la funcionalidad, estos no pudieron ser solucionados debido a limitaciones de tiempo. Esta combinación de pruebas automatizadas y análisis de calidad asegura que la solución cumple con los estándares de seguridad y rendimiento esperados, garantizando la integridad de la información procesada.



8. Conclusiones

En conclusión, el proyecto ha logrado cumplir con éxito los objetivos propuestos, garantizando la protección de los datos transmitidos desde satélites de observación terrestre mediante la implementación del algoritmo de cifrado AES-CTR de 128 bits y el uso de HKDF para la generación de llaves dinámicas, asegurando así la confidencialidad y seguridad de la información. La aplicación de la técnica de "chunking" ha permitido manejar eficientemente los recursos de memoria en sistemas embebidos, posibilitando el procesamiento de grandes volúmenes de datos sin comprometer la estabilidad del sistema. Estos logros han sido alcanzados gracias al esfuerzo y la colaboración del equipo, cuyo trabajo coordinado y compromiso han sido fundamentales para desarrollar una solución robusta y eficaz que cumple con los estándares de seguridad requeridos para proteger la valiosa información satelital.

9. Referencias

https://asecuritysite.com/hash/HKDF

OpenSSL. (n.d.). OpenSSL: Cryptography and SSL/TLS toolkit. Retrieved from https://www.openssl.org/

Wikipedia. (2023). Advanced Encryption Standard. Retrieved from https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

Krawczyk, H., & Eronen, P. (2010). HMAC-based Extract-and-Expand Key Derivation Function (HKDF) (RFC 5869). Retrieved from https://tools.ietf.org/html/rfc5869

Wikipedia. (2023). HMAC. Retrieved from https://en.wikipedia.org/wiki/HMAC

Wikipedia. (2023). SHA-2. Retrieved from https://en.wikipedia.org/wiki/SHA-2

OpenSSL. (n.d.). EVP_EncryptInit_ex - OpenSSL [Manual]. Retrieved from https://www.openssl.org/docs/man1.1.1/man3/EVP_EncryptInit.html
OpenSSL. (n.d.). RAND_bytes - OpenSSL [Manual]. Retrieved from https://www.openssl.org/docs/man1.1.1/man3/RAND_bytes.html

Wikipedia. (2023). Initialization Vector. Retrieved from https://en.wikipedia.org/wiki/Initialization_vector