Explicacion AES.cpp

Función chartobyte(): Esta función convierte una cadena de caracteres (char) en un arreglo de bytes (byte). Este proceso es esencial cuando se maneja texto que necesita ser transformado en una representación binaria para ser procesado por los algoritmos de cifrado. Cada carácter en la cadena de entrada se convierte directamente en su valor ASCII correspondiente en el arreglo de salida.

Argumentos:

- const char* input: Entrada, cadena de caracteres a ser convertida.
- byte* output: Salida, arreglo de bytes que almacenará los valores convertidos.
- int length: Longitud de la cadena de entrada.

Función padToMultipleOf16(): Esta función asegura que el texto plano tenga una longitud que sea un múltiplo de 16 bytes, lo cual es necesario para que el texto pueda ser procesado por el cifrado AES. La función expande el texto plano añadiendo valores aleatorios hasta que su longitud sea un múltiplo de 16. Este padding es crucial para el cifrado en bloques, ya que AES opera en bloques de 128 bits (16 bytes).

Argumentos:

- byte *plaintext: Entrada, arreglo de bytes que representa el texto plano original.
- size_t length: Longitud del texto plano original.
- byte *&plaintextexpanded: Salida, referencia a un arreglo de bytes que contendrá el texto plano expandido.
- size_t &expandedLength: Salida, referencia al tamaño del texto plano expandido.

Función KeyExpansion(): La función KeyExpansion toma la clave original y la expande para generar las claves de ronda necesarias para el proceso de cifrado AES. Dependiendo del tamaño de la clave (128, 192 o 256 bits), se generan diferentes cantidades de claves de ronda. Este proceso es fundamental porque en AES cada ronda del cifrado utiliza una clave derivada diferente, aumentando así la complejidad y seguridad del cifrado.

Argumentos:

- byte *key: Entrada, clave original.
- size_t keySize: Tamaño de la clave original en bits (128, 192 o 256).
- byte *expandedKey: Salida, arreglo de bytes que contendrá la clave expandida.
- size_t &expandedKeySize: Salida, tamaño de la clave expandida.

Función SubWord(): SubWord es una función auxiliar que aplica la sustitución de bytes a una palabra de 32 bits usando la tabla sbox. Este proceso es parte fundamental de la expansión de la clave, transformando cada palabra en una nueva palabra de acuerdo con la tabla sbox. Esto añade un nivel adicional de seguridad al proceso de cifrado, asegurando que la clave de cada ronda sea suficientemente diferente y más difícil de predecir.

Argumentos:

- uint32_t word: Entrada, palabra de 32 bits a la que se aplicará la sustitución de bytes.

Retorna una palabra de 32 bits con los bytes sustituidos.

Función RotWord(): RotWord es otra función auxiliar que realiza una rotación cíclica de una palabra de 32 bits hacia la izquierda. Este proceso es parte de la generación de claves de ronda en la expansión de la clave. La rotación ayuda a asegurar que los bytes de la clave se dispersen y se mezclen, aumentando la complejidad del cifrado y ayudando a proteger contra ataques que buscan patrones en las claves.

Argumentos:

- uint32_t word: Entrada, palabra de 32 bits a la que se aplicará la rotación cíclica.

Retorna una palabra de 32 bits rotada cíclicamente hacia la izquierda

Función addRoundKey(): La función addRoundKey aplica la operación XOR entre el estado actual (state) y la clave de la ronda (roundKey). Este paso es crucial en el proceso de cifrado AES, ya que combina el estado con la clave de la ronda actual, introduciendo la clave secreta en el proceso de cifrado. Esta operación se realiza en cada ronda de AES y es esencial para asegurar que el cifrado sea dependiente de la clave.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual.
- byte *roundKey: Entrada, clave de la ronda que se aplicará al estado.

Función SubBytes(): La función SubBytes realiza la sustitución de bytes en el estado utilizando la tabla sbox. Este proceso es una transformación no lineal que reemplaza cada byte del estado con un byte correspondiente de la tabla sbox. Esta sustitución introduce confusión en el cifrado, dificultando la tarea de los atacantes que intenten analizar la relación entre la clave de cifrado y el texto cifrado.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual en la ronda de cifrado.

Función ShiftRows(): La función ShiftRows reorganiza los bytes en las filas del estado mediante desplazamientos cíclicos hacia la izquierda. Este paso es parte del proceso de difusión en AES, asegurando que los bytes dentro de cada bloque se mezclen a medida que avanzan las rondas de cifrado. Al realizar estos desplazamientos, la dependencia entre los bytes de diferentes columnas aumenta, complicando el análisis de patrones en el texto cifrado.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual en la ronda de cifrado.

Función MixColums(): MixColumns mezcla los bytes en cada columna del estado utilizando multiplicaciones en un campo de Galois. Esta operación proporciona difusión al distribuir los bytes en la columna a través de combinaciones lineales, lo que asegura que un pequeño cambio en el estado tenga un impacto significativo en el bloque de salida. Esto aumenta la resistencia del cifrado frente a ataques criptográficos que intentan explotar patrones de correlación.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual en la ronda de cifrado.

Función roundEncrypt(): La función roundEncrypt realiza una ronda completa de cifrado AES, que incluye las operaciones de SubBytes, ShiftRows, MixColumns, y addRoundKey. Cada ronda de cifrado transforma el estado, incrementando la seguridad del bloque cifrado. Esta función encapsula los pasos de una ronda en un solo método, facilitando la aplicación de múltiples rondas en la estructura general del algoritmo AES.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual en la ronda de cifrado.
- byte *roundKey: Entrada, clave de la ronda que se aplicará al estado.

Función xorBlock(): La función xorBlock realiza la operación XOR entre dos bloques de bytes (block1 y block2), almacenando el resultado en output. Este proceso es común en modos de operación como el CBC (Cipher Block Chaining) donde se necesita combinar el bloque de texto plano con el bloque de cifrado previo antes de aplicar el cifrado. La

operación XOR es fundamental en criptografía por su simplicidad y su capacidad para mezclar datos de manera reversible.

Argumentos:

- byte *block1: Entrada, primer bloque de bytes.
- byte *block2: Entrada, segundo bloque de bytes.
- byte *output: Salida, bloque de bytes que almacenará el resultado de la operación XOR.

Función Encrypt(): La función Encrypt es la encargada de realizar el cifrado completo de un texto plano utilizando el algoritmo AES. Dependiendo del modo de operación (ECB, CBC, CFB, OFB), aplica diferentes métodos de cifrado sobre los bloques del texto plano. La función realiza la expansión de la clave, el padding del texto plano para que sea un múltiplo de 16 bytes, y finalmente, cifra cada bloque de acuerdo con el modo de operación seleccionado. Retorna el texto cifrado.

Argumentos:

- const char *plain_Text: Entrada, texto plano que será cifrado.
- size_t plain_Text_Size: Entrada, tamaño del texto plano en bytes.
- byte *key: Entrada, clave de cifrado.
- int keySize: Entrada, tamaño de la clave en bits (128, 192, o 256).
- byte *IV: Entrada, vector de inicialización, usado en modos como CBC, CFB, y OFB.
- const char *Modo: Entrada, modo de operación (ECB, CBC, CFB, OFB).
- size_t &cipherTextSize: Salida, tamaño del texto cifrado en bytes.

Retorna un puntero a byte que contine el texto cifrado.

Función invSubBytes(): La función invSubBytes realiza la sustitución inversa de bytes en el estado utilizando la tabla invsbox. Este proceso es la contrapartida del paso SubBytes en el cifrado AES. Durante la descodificación, se utiliza invSubBytes para revertir la sustitución de bytes aplicada durante el cifrado, devolviendo cada byte a su valor original en el estado previo al cifrado.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual en la ronda de descifrado.

Función invShiftRows(): La función invShiftRows deshace la operación de ShiftRows realizada durante el cifrado, revirtiendo el desplazamiento cíclico de las filas en el estado.

Esto es esencial para la descodificación, ya que vuelve a colocar los bytes en sus posiciones originales, devolviendo el estado a una configuración anterior al cifrado y ayudando a restaurar el texto plano original.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual en la ronda de descifrado.

Función invMixColums(): invMixColumns invierte la operación de MixColumns realizada durante el cifrado. Esta función realiza multiplicaciones en el campo de Galois usando las tablas mul9, mul11, mul13 y mul14 para revertir la mezcla de bytes en las columnas del estado. Este paso es necesario en el proceso de descifrado para recuperar el estado antes de la mezcla de columnas, permitiendo así la restauración del texto plano original.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual en la ronda de descifrado.

Función roundDecrypt(): La función roundDecrypt ejecuta una ronda completa de descifrado AES, que incluye los pasos de addRoundKey, invMixColumns, invShiftRows, e invSubBytes. Cada ronda de descifrado revierte las operaciones de una ronda de cifrado, deshaciendo los cambios realizados en el estado para recuperar el texto plano original. Esta función encapsula los pasos de una ronda de descifrado, haciendo que el proceso de descifrado completo sea más estructurado y manejable.

Argumentos:

- byte *state: Entrada/Salida, arreglo de bytes que representa el estado actual en la ronda de descifrado.
- byte *roundKey: Entrada, clave de la ronda que se aplicará al estado.

Función Decrypt(): La función Decrypt se encarga de realizar el descifrado de un texto cifrado usando el algoritmo AES. Dependiendo del modo de operación especificado (ECB, CBC, CFB, OFB), aplica diferentes métodos para descifrar los bloques de datos cifrados. La función realiza la expansión de la clave y luego procesa cada bloque de datos cifrados para devolver el texto plano original.

Argumentos:

- byte *textoCifrado: Entrada, puntero al texto cifrado que será descifrado.

- byte *textoDescifrado: Salida, puntero al buffer donde se almacenará el texto descifrado.
- size_t length: Entrada, tamaño del texto cifrado en bytes.
- byte *key: Entrada, clave utilizada para el descifrado.
- int keySize: Entrada, tamaño de la clave en bits (128, 192, o 256).
- byte *IV: Entrada, vector de inicialización usado en modos como CBC, CFB, y OFB.
- const char *Modo: Entrada, modo de operación (ECB, CBC, CFB, OFB).

Explicacion BasicUsageAES.ino

Definición de Claves y Vectores

El código comienza definiendo tres claves de diferentes tamaños para los modos AES-128, AES-192 y AES-256. También se define un vector de inicialización (IV) que se usará en los modos de cifrado que requieren un valor inicial.

Texto Plano y Preparación

Se establece un texto plano que será cifrado. Este texto se convierte en una secuencia de bytes para ser procesado por la función de cifrado.

Configuración Inicial

En la función setup(), que se ejecuta una vez al inicio del programa, se inicializa la comunicación serie para mostrar los resultados en la consola.

Cifrado del Texto Plano

Se llama a la función Encrypt para cifrar el texto plano utilizando la clave AES-192 y el vector de inicialización con el modo de cifrado "OFB". El tamaño del texto cifrado se almacena para su uso posterior.

Descifrado del Texto Cifrado

El texto cifrado resultante se descifra usando la función Decrypt con los mismos parámetros utilizados para el cifrado. Esto debería devolver el texto original si todo ha funcionado correctamente.

Impresión de Resultados

El programa imprime el texto plano, el texto cifrado en formato hexadecimal y el texto descifrado en la consola serie. De esta manera, se puede verificar que el proceso de cifrado y descifrado ha funcionado correctamente.

Conversión de Bytes a Caracteres

Para facilitar la visualización, el código también convierte el texto cifrado y descifrado de una secuencia de bytes a una cadena de caracteres para que pueda ser leída más fácilmente en la consola.

Funciones Auxiliares

Finalmente, el código incluye algunas funciones auxiliares para convertir bytes en caracteres y para imprimir secuencias de bytes en formato hexadecimal, lo cual es útil para visualizar los datos cifrados.