INFORME DE LABORATORIO 1

Autores: *María Del Mar Arbeláez Sandoval, Julián Mauricio Sánchez Ceballos*

Laboratorio de Electrónica Digital 3
Departamento de Ingeniería Electrónica y de Telecomunicaciones
Universidad de Antioquia

Resumen

El trabajo de esta practica consiste en la implementación del algoritmo de cifrado AES128 (Advanced Encryption Standar) en lenguaje C, esto con el objetivo de generar un primer acercamiento a la programación en lenguaje C, programación muy útil y principal recurso de software para el desarrollo de las demás practicas del laboratorio. Durante la practica se realizaron funciones acordes a la implementación de el algoritmo de cifrado ya mencionado, para el desarrollo de esta practica se hace importante el entendimiento de operaciones a nivel de bit y el uso de punteros, dos conceptos de vital importancia en el mundo de la programación de sistemas embebidos, obteniendo como resultado de esta practica el mejoramiento de las habilidades de manipulación de información binaria en el lenguaje de bajo nivel C.

Palabras clave: lenguaje C, estandar AES-128, criptografia, bits, operaciones bitwise.

INTRODUCCIÓN.

El estándar NISRT FIPS 127, conocido como el Estándar de Cifrado Avanzado (AES por sus cifras en inglés) es de vital importancia en el ámbito de la seguridad de la información debido a su papel fundamental en el manejo de datos sensibles, desarrollado para reemplazar estándares antiguos que se habían vuelto vulnerables a los ataques con el aumento de la computación, con este nuevo estándar se desarrolló un algoritmo de cifrado de tipo simétrico que ofrece un nivel superior de seguridad, este nuevo estándar también proporciona una base confiable para el desarrollo de políticas y procedimientos de seguridad informática. La implementación del estándar en lenguaje C es una excelente oportunidad para que un programador mejore sus habilidades de manipulación de datos binarios, punteros y la estructura de codificación en C.

Para la implementación del estándar primero se realiza una lectura y compresión del estándar y de la matemática detrás de ella. en primer lugar se define un campo finito, espacio matemático donde se realizan todas las operaciones matemáticas del estándar de cifrado. Luego se procede a la programación de las subrutinas y por último a la integración de todo el código.

El AES-128 hace uso de operaciones en el campo finito GF(2^8) , que es un campo finito con 256 elementos. Este campo finito es crucial para las transformaciones internas del algoritmo.

CAMPO FINITO GF(2^8).

En Matemática, un campo finito es un campo que contiene un numero finito de elementos, este campo tiene 256 elementos y se define como el conjunto de todos los polinomios de grado menor a 8 con coeficientes 0, 1:

$$a(x) = a_7x^7 + a_6x^6 + ... + a_1x + a_0$$

Donde cada a_i puede ser 0 ó 1. Para operar en GF(2^8), se usa un polinomio irreducible de grado 8 comúnmente $x^8+x^4+x^3+x+1$. Este polinomio actúa como el modulo en las operaciones aritméticas dentro de este campo

0.0.1. Operaciones en GF(2^8)

1. **Suma y resta:** Las operaciones de suma y resta en GF(2^8) son iguales y se realizan mediante la operación XOR bit a bit.

2. **Multiplicación:** la multiplicación en GF(2^8) se lleva a cabo mediante la multiplicación de polinomios, seguida de una reducción de modulo $x^8 + x^4 + x^3 + x + 1$ para mantener el resultado dentro de GF(2^8). Este proceso garantiza que los resultados de la multiplicación sigan siendo polinomios de grado menor a 8.

3. **Inverso multiplicativo:** Para calcular el inverso de un elemento en GF(2^8), se utilizan algoritmos como el de Euclides extendido, ya que el campo finito garantiza que cada elemento no nulo tiene un único inverso multiplicativo.

Este campo finito se utiliza en dos procesos:

1. **SubBytes:** Esta transformación no lineal sustituye cada byte del estado inverso multiplicativo en GF(2^8), seguido de una transformación afín. La S-box de AES deriva precisamente de esta operación, proporcionando resistencia a criptoanálisis lineal y diferencial. En la implementación mostrada en este laboratorio, este calculo no se realiza y la s-box se presenta como una variable constante la cual no se calcula para efectos de optimización del código.

2. **MixColumns:** En esta transformación, se multiplica cada columna del estado por una matriz fija, utilizando multiplicaciones en GF(2^8). Esto difunde la influencia de cada byte sobre toda la columna, aumentando la confusión y difusión del algoritmo.

AES-128.

La implementación del estándar se realiza en primer lugar con la elaboracion de subrutinas, estas subrutinas son llamadas en cada ronda del algoritmo, subrutinas que son las encargadas de realizar las transformaciones a los datos.

1. **EXPANSIÓN DE CLAVE:** Genera un conjunto de claves para cada ronda a partir de la clave original de 128 bits. Se producen 11 subclaves de 128 bits, una para cada ronda de cifrado, dentro de esta subrutina hay algunas operaciones que se realizan:
   a) **RotWord:** Rotar los bytes de una palabra (4 bytes).
   b) **SubWord:** sustituir cada byte por su equivalente en la S-box.
   c) **XOR:** Realiza operaciones con la constante de ronda R-con y la palabra anterior.

   En la siguiente imagen se muestra el pseudocódigo para la subrutina de Expansión de Clave:

Algorithm 2 Pseudocode for KEYEXPANSION()

```
1:  procedure KeyExpansion(key)
2:      i ← 0
3:      while i ≤ Nk − 1 do
4:          w[i] ← key[4 ∗ i..4 ∗ i + 3]
5:          i ← i + 1
6:      end while                          ▷ When the loop concludes, i = Nk.
7:      while i ≤ 4 ∗ Nr + 3 do
8:          temp ← w[i − 1]
9:          if i mod Nk = 0 then
10:             temp ← SubWord(RotWord(temp)) ⊕ Rcon[i/Nk]
11:         else if Nk > 6 and i mod Nk = 4 then
12:             temp ← SubWord(temp)
13:         end if
14:         w[i] ← w[i − Nk] ⊕ temp
15:         i ← i + 1
16:     end while
17:     return w
18: end procedure
```

Figura **0-1**: KeyExpansion para AES-128

2. **ADDROUNDKEY:** Aplica una subclave al estado actúa mediante una operación XOR. este proceso se realiza al inicio y al final de cada ronda.

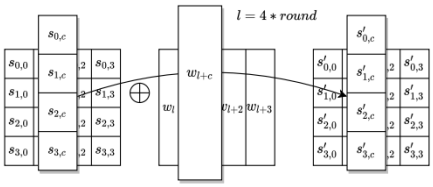

Figura **0-2**: AddRoundKey para AES-128

3. **SUBBYTES:** Sustituye cada byte del estado con un valor correspondiente en la S-box, obtenida mediante el inverso multiplicativo en GF(2^8) y una transformacion afin.

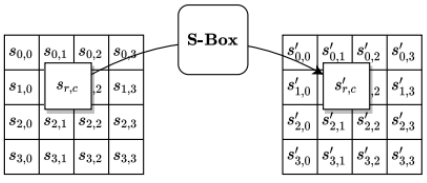

Figura **0-3**: SubBytes para AES-128

4. **SHIFTROWS:** Realiza un desplazamiento de cíclico en las filas del estado, desplazando cada fila a la izquierda un numero diferente de posiciones.

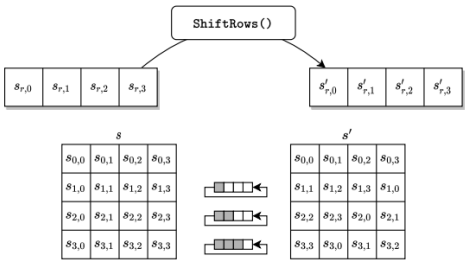

Figura **0-4**: ShiftRows para AES-128

5. **MIXCOLUMNS:** Mezcla los bytes de cada columna del estado utilizando una multiplicación matricial en el campo finito GF(2^8).

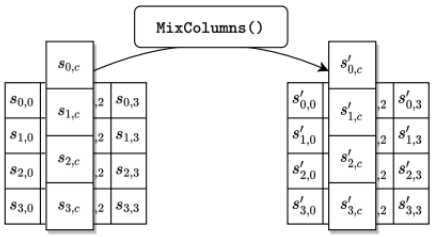

Figura **0-5**: MixColumns para AES-128

El AES-128 realiza 10 rondas de transformaciones en el estado en donde cada ronda incluye las operaciones de Subbytes, shiftRows, MixColumns y Addroundkey. para esto se muestra el pseudocódigo para el proceso de cifrado:

Algorithm 1 Pseudocode for CIPHER()

```
1:  procedure CIPHER(in, Nr, w)
2:      state ← in
3:      state ← ADDROUNDKEY(state, w[0..3])
4:      for round from 1 to Nr − 1 do
5:          state ← SUBBYTES(state)
6:          state ← SHIFTROWS(state)
7:          state ← MIXCOLUMNS(state)
8:          state ← ADDROUNDKEY(state, w[4 ∗ round..4 ∗ round + 3])
9:      end for
10:     state ← SUBBYTES(state)
11:     state ← SHIFTROWS(state)
12:     state ← ADDROUNDKEY(state, w[4 ∗ Nr..4 ∗ Nr + 3])
13:     return state
14: end procedure
```

Figura **0-6**: Pseudocódigo para el cifrado en AES128

IMPLEMENTACIÓN.

Para el uso del código es necesario entender la implementación y la estructura del proyecto. Para esto se generó una documentación en Doxyfile la cual se puede obtener en la carpeta principal del proyecto utilizando el comando:

```
doxygen Doxyfile
```

Este comando creará una carpeta llamada HTML en la cual se utiliza el archivo llamado `index.html` al abrir este archivo se obtiene la documentación de la implementación del estándar.

COMPILACIÓN

Al ejecutar el siguiente comando en la carpeta de `src`:

```
gcc main.c ../include/AES_Func.c -o main.exe
```

Esto generará el archivo `main.exe` el cual ejecuta el programa del estándar.

EJECUCIÓN

En la carpeta `src` ejecutar el siguiente comando:

```
./main.exe ../TextFiles/text.txt ../TextFiles/key.txt
```

Dentro de la carpeta `TextFiles` debe existir el nombre del archivo que se desea cifrar, indicado en este ejemplo como `text.txt` así como la clave, indicada en el ejemplo como `key.txt`.

TIEMPOS DE EJECUCIÓN

Después de 20 ejecuciones, los resultados obtenidos se muestran en la siguiente tabla:

Archivo	T. promedio (s)	desviación (s)
1 G	190.001	3.5838
100 M	19.2889	0.5747
10 M	1.9296	0.05778

Cuadro **0-1**: Resultados de ejecución

los valores mínimos y máximos se muestran a continuación:

Archivo	T. máximo (s)	T mínimo (s)
1 G	203.292	187.439
100 M	20.67	18.262
10 M	2.046	1.808

Cuadro **0-2**: Tiempos Máximos y mínimos.

CONCLUSIONES.

- La implementación del estándar AES-128 en C se pudo evidenciar la capacidad del lenguaje para realizar operaciones de bajo nivel, como la manipulación de bits y bytes. Esto es de vital importancia en el desarrollo de un curso de sistemas embebidos, donde estas habilidades junto con las habilidad de programación de punteros es muy necesaria.
- Se pudo evidenciar que el lenguaje C permite optimizar el rendimiento del código especialmente en operaciones matemáticas complejas como las necesarias para la parte operativa del campo finito GF(2^8), esto se ve altamente reflejado en la linealidad que presenta los cambios en tiempo dependiendo del tamaño del archivo que se desea desencriptar. Esta característica de C además de ser una ventaja para aplicaciones en criptografía, también es una ventaja para aplicaciones donde el tiempo de ejecución sea un factor critico como lo son en algunos casos el diseño de sistemas embebidos
- Finalmente durante la practica se evidenció la importancia del trabajo ordenado y modular, esto basado en el uso modular de funciones y rutinas para la implementación del estándar AES-128. Estas características mostraron ser eficientes a la hora del entendimiento y compresión del código, como aspectos claves en el desarrollo de software seguro y confiable.

BIBLIOGRAFIA:

- **NIST (2001).** *FIPS PUB 197: Advanced Encryption Standard (AES).* National Institute of Standards and Technology.
- *Harbison, S. P., & Steele, G. L. (2002). C: A Reference Manual (5th Edition). Prentice Hall.* l.
- *van Heesch, D. (2008). Doxygen Manual: The Definitive Guide to Code Documentation with Doxygen. CreateSpace Independent Publishing Platform.*