Seguridad de la información en la red basada en el sistema de criptografía AES

Kevin Alejandro Herrera
Escuela de Matemática y Ciencias de la Computación
Universidad Nacional Autónoma de Honduras
e-mail: kherrerar@unah.hn

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

Seguridad de la información en la red basada en el sistema de criptografía AES

***Resumen*—**

Palabras claves: **AES, criptografía, seguridad de la información, cifrado, descifrado.**

I. Introducción

II. Planteamiento del problema

El avance tecnológico con el pasar del tiempo ha crecido de manera exponencial, con ello el flujo de información en la red ha aumentado su volumen, por tanto muchos de estos datos se encuentran vulnerables al no tener un sistema de seguridad adecuado que pueda resguardar dichos datos, estos mismos pertenecen a organizaciones, empresas o personas de interés.

A medida la tecnología avanza para bien también lo hace para mal y muchos de estos sistemas quedan obsoletos ante ataques cada vez indefendibles, para ello se debe implementar un método de seguridad informática; la criptografía en sus diferentes sistemas ayudará de manera eficiente a proteger esta información.

En particular el cifrado simétrico preserva la confidencialidad tanto en las transmisiones de información como en su almacenamiento, protegiendo los archivos y evitando que personas ajenas tengan acceso a la información.

III. Preliminares y Notación

III-A. Entradas y salidas

La salida y entrada del algoritmo AES consiste en una secuencia de *128 bits* (dígitos con valores de 0 o 1). Estas secuencias a veces se denominarán *bloques* y el número de los bits que contienen se denominarán *longitud*.

III-B. Bytes

***Definición** 1 (Byte):* Es una cadena de 8 *bits*, representados como $\{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$, estos toman el valor de 0 o 1. Es la unidad básica de procesamiento del algoritmo AES.

La principal y básica estructura algebraica usada en el sistema AES es el campo binario de Galois $GF(2)$ (llamados *bits*). Por tanto un *byte* se puede representar en un campo finito $GF(2^8)$ utilizando una representación polinomial a través de la fórmula:

$$\sum_{i=0}^{7} b_i x^i$$

En el algoritmo AES trataremos a los *bytes* en notación hexadecimal, como se puede ver en el cuadro I.

Cadena de bits	Carácter hexadecimal
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Cuadro I
Notación hexadecimal

III-C. Matrices de bytes

III-D. El estado

El algoritmo AES divide los datos de entrada en bloques de 4 palabras de 32 *bits*, decir $4 \times 32 = 128$ *bits*, se puede ver como una sucesión de 16 *bytes*.

Para comprender las operaciones que se realizan en AES, representaremos los 16 *bytes* como una matriz de 4×4 (cuatro filas por cuatro columnas) llamada **matriz de estado**. Un *byte* se representara por la letra B por tanto nuestra matriz de estado será la siguiente:

$$\begin{bmatrix} B_{15} & B_{14} & B_{13} & B_{12} \\ B_{11} & B_{10} & B_9 & B_8 \\ B_7 & B_6 & B_5 & B_4 \\ B_3 & B_2 & B_1 & B_0 \end{bmatrix}$$

III-E. Operaciones matemáticas

Todos los bytes en el algoritmo de AES son interpretados como elementos de un campo finito, estos elementos pueden sumarse y multiplicarse, pero estas operaciones son diferentes a las que se utilizan con los números usuales. A continuación se detallan.

III-E1. Suma (o resta): La suma y resta se caracteriza por tener el mismo resultado y está dado por el operador lógico *or* exclusivo (XOR) bit a bit (ver cuadro II). Esta operación se representa por el símbolo $\oplus$.

***Ejemplo** 1:* Consideremos $\{01011011\}$ y $\{11010110\}$

- $01011011 \oplus 11010110 = 10001101$ (Notación binaria)
- $5B \oplus D6 = 8D$ (Notación hexadecimal)
- $(x^6 + x^4 + x^3 + x + 1) \oplus (x^7 + x^6 + x^4 + x^2 + x) = x^7 + x^3 + x^2 + 1$ (Notación polinómica)

A	B	A $\oplus$ B
0	0	0
0	1	1
1	0	1
1	1	0

Cuadro II
OPERACIÓN XOR.

Algoritmo 1 Algoritmo de la suma en $GF(2^8)$

Entrada: $f(x) = \sum_{i=0}^{7} a_i x^i$, $g(x) = \sum_{i=0}^{7} b_i x^i \in GF(2^8)$
Salida: $h(x) = \sum_{i=0}^{7} c_i x^i \in GF(2^8)$
1: **para** $i = 0$ hasta 7 **hacer**
2: $\quad c_i \rightarrow (a_i + b_i) \bmod 2$
3: **fin para**

III-E2. Multiplicación: : Se calcula como el residuo de la multiplicación binaria de dos componentes, teniendo en cuenta la suma y resta en campo finito, con el polinomio irreducible[1] dado por:

$$m(x) = x^8 + x^4 + x^3 + x + 1$$

Esta operación se denota por el símbolo $\bullet$.

Para multiplicar dos *bytes* se escribirán en forma polinómica, teniendo en cuenta que la suma es equivalente a una operación XOR. Al realizar la multiplicación es probable que se tenga un resultado de más de 8 *bits* por tanto se necesitará hacer modulo de dicho polinomio con el polinomio irreducible $m(x)$ para así volver a tener 1 *byte*.

***Ejemplo** 2:* Consideremos $\{01010111\} \bullet \{10000011\}$

- $(x^6+x^4+x^2+x^1+x+1) \bullet (x^7+x+1) = x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1$

Evidentemente este nuevo polinomio no contiene 8 *bits*, entonces:

$$x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1 \bmod x^8+x^4+x^3+x+1$$

$$= x^7 + x^6 + 1$$

que corresponde a 11000001.

Algoritmo 2 Algoritmo de la multiplicación en $GF(2^8)$

Entrada: $f(x) = \sum_{i=0}^{7} a_i x^i$, $g(x) = \sum_{i=0}^{7} b_i x^i \in GF(2^8)$
Salida: $h(x) = \sum_{i=0}^{7} c_i x^i \in GF(2^8)$
1: $c \leftarrow 0$
2: **para** $i = 0$ hasta 7 **hacer**
3: $\quad c_i \leftarrow b_i f(x) + c$
4: $\quad f(x) \leftarrow f(x)(x^i) \bmod m(x)$
5: **fin para**

IV. SEGURIDAD INFORMÁTICA

***Definición** 2 (Seguridad de la información):* Conjunto de medidas preventivas de las organizaciones que permiten resguardar y proteger la información buscando mantener la confidencialidad, disponibilidad e integridad de la misma.

[1]Un polinomio es irreducible si sus únicos divisores son el mismo y la unidad.

Esta seguridad busca la preservación de tres pilares:

- La *confidencialidad* es el principio que garantiza que la información solo puede ser accedida por las personas que tienen autorización.
- La *integridad* es que la información sólo puede ser creada y modificada por quien esté autorizado a hacerlo.
- La *disponibilidad* es que la información debe ser accesible para su consulta o modificación cuando se requiera.

Estos tres conceptos forman lo que a menudo se denomina la triada CIA. Los tres conceptos encarnan los objetivos fundamentales de seguridad tanto para los datos como para la información y los servicios informáticos.

Figura 1. Tríada de la CIA

***Definición** 3 (Seguridad informática):* Conjunto de políticas y mecanismos que nos permiten garantizar la confidencialidad, la integridad y la disponibilidad de los recursos en un sistema.

Observación: El concepto de *seguridad de la información* no debe ser confundido con el de *seguridad informática*, ya que este último sólo se encarga de la seguridad en el medio informático. Sin embargo, la información puede encontrarse en diferentes medios o formas, y no solo en medios informáticos. Es decir que el primer concepto es mas amplio que el segundo.

V. HISTORIA DE LA CRIPTOGRAFÍA

La palabra *criptografía* proviene en un sentido etimológico del griego Kriptos = ocultar, Graphos = escritura, lo que significaría escritura oculta.

En su clasificación dentro de las ciencias, la *criptografía* proviene de una rama de las matemáticas como se muestra en la figura 2 que fue iniciada por el matemático Claude Shannon [2]. en 1948, denominada: Teoría de la información.

***Definición** 4 (Criptografía):* Ciencia encargada de diseñar funciones o dispositivos, capaces de transformar mensajes legibles a mensajes cifrados de tal manera que esta transformación (cifrar) y su transformación inversa (descifrar) sólo pueden ser factibles con el conocimiento de una o mas llaves.

***Definición** 5 (Criptoanálisis):* Ciencia que estudia los métodos que se utilizan para descifrar mensajes sin necesidad

[2]Claude Elwood Shannon fue un matemático, ingeniero eléctrico y criptógrafo estadounidense recordado como *el padre de la teoría de la información*

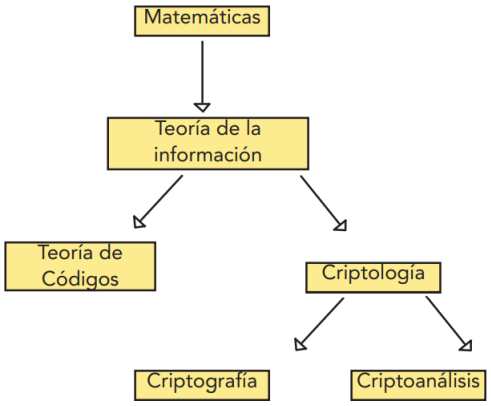

Figura 2. Origen de la criptografía.

de tener la llave con la que dicho mensaje fue cifrado.

La *criptografía* se puede clasificar en dos:

- La criptografía clásica
- La criptografía moderna

VI. Advanced Encryption Standard (AES)

VI-A. *Estructura del algoritmo*

La estructura de este algoritmo consiste en una serie de rondas donde se realizan un conjunto de cuatro transformaciones orientadas a *bytes*, el número de rondas depende de la longitud de la llave.

Vamos a asumir que la longitud de la llave escogida es de 128 *bits* ya que la longitud de la llave no afecta a las diversas trasformaciones que realiza el cifrado.

	Longitud de llave *(Nk words)*	**Tamaño de bloque** *(Nb words)*	**Número de rondas** *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Cuadro III
Número de rondas en función de la longitud de la llave

Básicamente se aplican cuatro trasformaciones a la matriz de estado durante el número determinado de rondas, las cuales se detallaran a continuación.

VI-B. *Cifrado AES*

Para definir este proceso, vamos a asumir que la longitud de la llave es de 128 *bits*, ya que la longitud de la llave no afecta a las trasformaciones que realiza el cifrado.
En el proceso de cifrado se aplican cuatro transformaciones a la matriz de estado inicial, dichas transformaciones son las siguientes:

VI-B1. SubBytes(): Esta operación consiste en una transformación no lineal, donde se realiza una sustitución de cada byte por otro byte establecido en la caja de sustitución (S-box) ver cuadro IV .

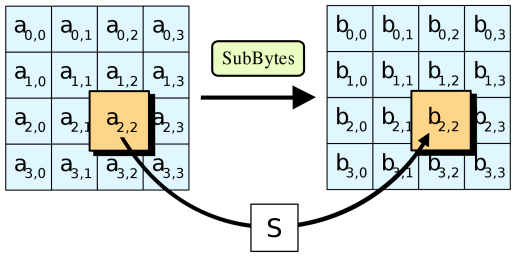

Figura 3. Transformación SubBytes

AES define una matriz de valores de *bytes* de 16×16 llamada S-Box, que contiene una permutación de todos los 256 valores de 8 *bits* posibles.

-	**x0**	**x1**	**x2**	**x3**	**x4**	**x5**	**x6**	**x7**	**x8**	**x9**	**xA**	**xB**	**xC**	**xD**	**xE**	**xF**
0x	63	7E	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1x	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2x	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3x	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4x	09	83	2C	1A	1B	6E	5A	A0	52	3B	D6	B3	29	E3	2F	84
5x	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6x	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7x	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8x	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9x	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
Ax	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
Bx	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
Cx	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
Dx	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
Ex	E1	F8	98	11	69	D9	8E	94	9B	16	87	E9	CE	55	28	DF
Fx	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

Cuadro IV
S-box

***Ejemplo** 3:* Si necesitamos realizar la transformación *SubBytes* al número $\{19\}$, no tenemos que hacer nada mas que mirar el cuadro IV , fijándonos en la fila **1x** y la columna **x9**, así obteniendo $\{D4\}$ que es la trasformación de $\{19\}$.

VI-B2. ShiftRows(): Esta transformación consiste en una rotación cíclica hacia la izquierda de las filas de la matriz de estado, de manera que la primera fila permanece igual, la segunda fila se rota hacia la izquierda una posición, la tercera fila se rota hacia la izquierda dos posiciones y la ultima fila se rota tres posiciones a la izquierda.

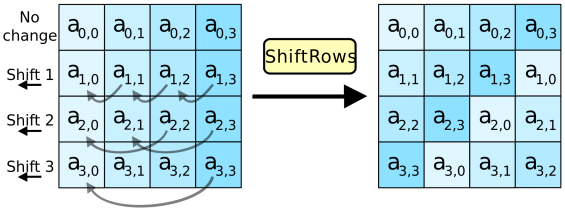

Figura 4. Transformación ShiftRows

VI-B3. MixColumns(): Esta trasformación consiste en multiplicar las columnas de bytes módulo $x^4 + 1$ por el polinomio

$$c(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

Es mas sencillo verlo como una multiplicación matricial. Sea:

$b(x) = c(x) \otimes a(x)$:

$$\begin{bmatrix} b_{0,c} \\ b_{1,c} \\ b_{2,c} \\ b_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_{0,c} \\ a_{1,c} \\ a_{2,c} \\ a_{3,c} \end{bmatrix} \text{ para } 0 \leq c \leq \mathbf{Nb}.$$

***Observación** 1:* $\otimes$ indica la multiplicación de dos polinomios (cada uno de grado < 4) modulo $x^4 + 1$.

Los cuatro *bytes* de la columna son reemplazados como se ve a continuación:

$$b_{0,c} = (\{02\} \bullet a_{0,c}) \oplus (\{03\} \bullet a_{1,c}) \oplus a_{2,c} \oplus a_{3,c}$$
$$b_{1,0} = a_{0,c} \oplus (\{02\} \oplus a_{1,c}) \oplus (\{03\} \bullet a_{2,c}) \oplus a_{3,c}$$
$$b_{2,c} = a_{0,c} \oplus a_{1,c} \oplus (\{02\} \bullet a_{2,c}) \oplus (\{03\} \bullet a_{3,c})$$
$$b_{3,c} = (\{03\} \bullet a_{0,c}) \oplus a_{1,c} \oplus a_{2,c} \oplus (\{02\} \bullet a_{3,c})$$

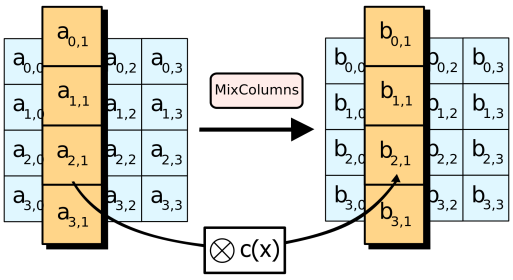

Figura 5. Transformación MixColumns

VI-B4. AddRoundKey(): Esta transformación consiste en aplicar la operación OR-Exclusivo (XOR) entre cada byte de la matriz de estado que proviene de la trasformación anterior (MixColumns) y una subclave que se genera a partir de la clave del sistema para esa ronda.

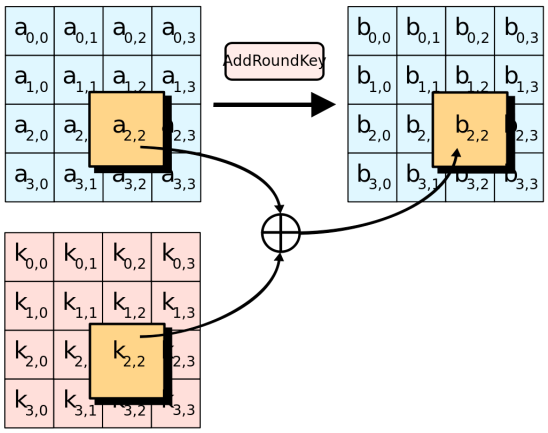

Figura 6. Transformación AddRoundKey

En la figura que representa el proceso de cifrado (Figura 7) esta transformación se realiza en la ronda inicial (ronda 0). Como se definió previamente nuestra longitud de clave será de 128 *bits*, por tanto la subclave de la ronda 0 es la propia clave de cifrado.

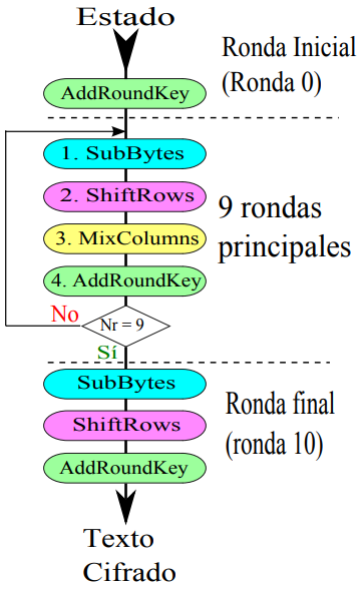

Figura 7. Proceso de cifrado AES

VI-C. Cifrado inverso AES

AES es un cifrador de clave simétrica por tanto la llave que se utiliza para *cifrar* se utilizará para *descifrar*.

Básicamente, el proceso de *descifrado* consiste en aplicar en orden inverso las trasformaciones aplicadas en el proceso de *cifrado*.

Las transformaciones a aplicar son las siguientes:

VI-C1. InvSubBytes(): La transformación *InvSubBytes* es, al igual que la transformación *SubBytes*, una sustitución no lineal de bytes.

-	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xA	xB	xC	xD	xE	xF
0x	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1x	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2x	54	7B	94	32	46	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3x	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4x	72	F8	F6	64	86	68	98	16	D4	4A	5C	CC	5D	65	B6	92
5x	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6x	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7x	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8x	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9x	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
Ax	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
Bx	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
Cx	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
Dx	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
Ex	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
Fx	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Cuadro V
INVS-BOX

El cuadro V es inverso al cuadro IV, que representaba la S-box, este cuadro inverso se llama InvS-Box.

***Ejemplo** 4:* Recordemos el ejemplo que se propuso en la transformación *SubBytes*, donde el byte $\{19\}$ tomaba el valor $\{D4\}$. Si ahora queremos aplicar la trasnformación *InvSubBytes* al byte $\{D4\}$, miramos en el cuadro V la fila **Dx** y la columna **x4**, dando como resultado el byte $\{19\}$.

VI-C2. InvShiftRows(): Esta transformación consiste en una rotación cíclica de *bytes* hacia la derecha. Por tanto, es una rotación en dirección opuesta a la que se propuso en la transformación *ShiftRows*.

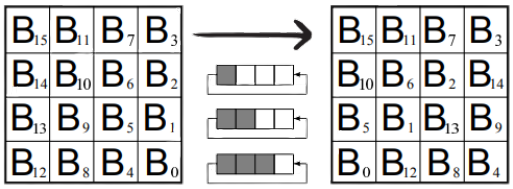

Figura 8. Transformación InvShiftRows

VI-C3. InvMixColumns(): Es la operación inversa de *MixColumns*, en esta transformación cada columna se trata como un polinomio y luego se multiplica módulo $x^4 + 1$ con un polinomio fijo:

$$c^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$$

De nuevo es más fácil verlo como una multiplicación matricial y recordando la observación 1, tenemos:

$$a(x) = c^{-1}(x) \otimes b(x)$$

$$\begin{bmatrix} a_{0,c} \\ a_{1,c} \\ a_{2,c} \\ a_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} b_{0,c} \\ b_{1,c} \\ b_{2,c} \\ b_{3,c} \end{bmatrix} \text{ para } 0 \leq c \leq \mathbf{Nb}.$$

donde los *bytes* son reemplazados de la siguiente manera:

$$a_{0,c} = (\{0e\} \bullet b_{0,c}) \oplus (\{0b\} \bullet b_{1,c}) \oplus (\{0d\} \bullet b_{2,c}) \oplus (\{09\} \bullet b_{3,c})$$

$$a_{1,c} = (\{09\} \bullet b_{0,c}) \oplus (\{0e\} \bullet b_{1,c}) \oplus (\{0b\} \bullet b_{2,c}) \oplus (\{0d\} \bullet b_{3,c})$$

$$a_{2,c} = (\{0d\} \bullet b_{0,c}) \oplus (\{09\} \bullet b_{1,c}) \oplus (\{0e\} \bullet b_{2,c}) \oplus (\{0b\} \bullet b_{3,c})$$

$$a_{3,c} = (\{0b\} \bullet b_{0,c}) \oplus (\{0d\} \bullet b_{1,c}) \oplus (\{09\} \bullet b_{2,c}) \oplus (\{0e\} \bullet b_{3,c})$$

VI-C4. AddRoundKey(): Esta transformación es la que se describió en la Sección *VI-B4*, es su propio inverso, ya que solo implica una aplicación de la operación XOR.

En la figura 9 se encuentra el diagrama del cifrado inverso de AES.

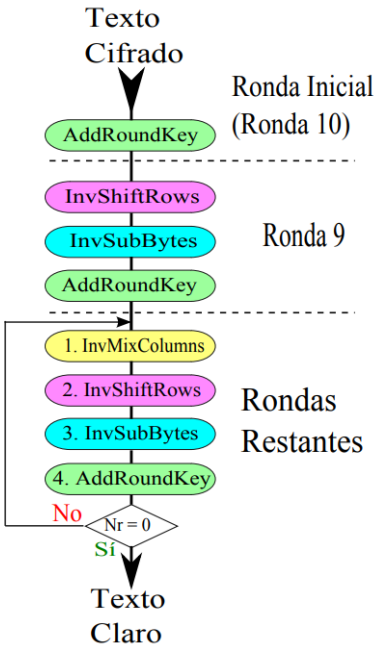

Figura 9. Proceso de cifrado inverso AES

VI-D. Cálculo de subclaves

El sistema de criptografía AES, concretamente en la transformación *AddRoundKey*, utiliza diferentes subclaves a lo largo del cifrado/descifrado, todas derivadas de la clave original.

***Definición** 6 (Clave expandida):* Es una sucesión de todas las subclaves, puede verse como una matriz de 4 filas por $[4 \times (Nr + 1)]$ columnas.

Por tanto la clave expandida depende del número de rondas (*Nr*) y este a su vez depende de la longitud de la llave (ver cuadro III).

En este proyecto de investigación se considera una longitud de llave de 128 bits, entonces nuestra matriz de clave expandida sera de 4 filas por (4×11), es decir, 44 columnas.

Un elemento importante de AES necesario para el cálculo de sublcaves es la matriz Rcon. Dicha matriz, según [4] es de la siguiente forma:

$$\text{Rcon} = \begin{bmatrix} 01 & 02 & 04 & 08 & 10 & 20 & 40 & 80 & 1B & 36 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \\ 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 & 00 \end{bmatrix}$$

A la primera columna de Rcon le asignaremos Rcon[1] y asi sucesivamente hasta la última columna, Rcon[10].

0	1	2	3	4	5	6	7	8	9	10	11	...	40	41	42	43
2B	28	AB	09													
7E	AE	F7	CF													
15	D2	15	4F													
16	A6	88	3C													

Figura 10. Matriz de clave expandida

VI-D1. Cálculo de subclave a partir de una clave de 128 bits: Como se ve en la figura 11, la matriz de clave expandida es de 4 filas por 44 columnas.

1. El primer paso para calcular las subclaves es introducir la clave original en las primeras columnas. En el caso de 128 *bits*, ocupa 4 columnas, como se puede apreciar en la figura 11 .

   Ahora se deberá calcular las columnas de las diferentes subclaves una a una. Básicamente, la primera columna del grupo de *Nk* columnas (en este caso 4) aplica una

serie de operaciones que se describirán a continuación, mientras que el resto de columnas del grupo se calculan de manera análoga.

Se denomina C_i como la columna actual que queremos encontrar, en este momento tenemos que $i = 4$.

2. Se aplica sobre C_{i-1} una rotación hacia arriba.
   Es decir,
$$\begin{bmatrix} 09 \\ CF \\ 4F \\ 3C \end{bmatrix} \Rightarrow \begin{bmatrix} CF \\ 4F \\ 3C \\ 09 \end{bmatrix}$$
3. Sobre esta columna rotada, se aplica la transformación *SubBytes* (ver cuadro IV).
   Entonces,
$$\begin{bmatrix} CF \\ 4F \\ 3C \\ 09 \end{bmatrix} \Rightarrow \begin{bmatrix} 8A \\ 84 \\ EB \\ 01 \end{bmatrix}$$
4. A la trasformación *SubBytes* que resulto en el paso anterior se le suman C_{i-nk} y Rcon$[i/nk]$.
   Por tanto,
$$\begin{bmatrix} 8A \\ 84 \\ EB \\ 01 \end{bmatrix} \oplus \begin{bmatrix} 2B \\ 7E \\ 15 \\ 16 \end{bmatrix} \oplus \begin{bmatrix} 01 \\ 00 \\ 00 \\ 00 \end{bmatrix} = \begin{bmatrix} A0 \\ FA \\ FE \\ 17 \end{bmatrix} = C_4$$
   Ahora $i = 5$
5. Para encontrar C_i, se realiza la suma de C_{i-1} y C_{i-nk}.
   Es decir,
$$\begin{bmatrix} A0 \\ FA \\ FE \\ 17 \end{bmatrix} \oplus \begin{bmatrix} 28 \\ AE \\ D2 \\ A6 \end{bmatrix} = \begin{bmatrix} 88 \\ 54 \\ 2C \\ B1 \end{bmatrix} = C_5$$

Una vez hecho esto, las dos columnas que quedan por calcular de la primera subclave, C_6 y C_7, se hallan aplicando el paso 5.

0	1	2	3	4	5	6	7	8	9	10	11	...	40	41	42	43
2B	28	AB	09	A0	88	09	2A									
7E	AE	F7	CF	FA	54	CF	6C									
15	D2	15	4F	FE	2C	39	76									
16	A6	88	3C	17	B1	39	05									

Figura 11. Matriz de clave expandida con subclave para la primera ronda

Para calcular una nueva subclave, volvemos a realizar el paso 2 y se comienza de nuevo.

VII. Resultados

VIII. Conclusiones

IX. Trabajo a futuro

Referencias

[1] Stallings, W. (2016). *Cryptography and Network Security: Principles and Practice* (7.ª ed.). Pearson Education.

[2] Tanenbaum, & Wheterall. (2011). *Redes de computadoras* (5.ª ed.). Pearson Education.

[3] Tapia-Recillas, H. (2011). *Sobre algunas aplicaciones de los campos de Galois.* Miscelánea Matemática de la sociedad matemática Mexicana. https://miscelaneamatematica.org/download/tbl_articulos.pdf2.b83a7656c55ef738.353330362e706466.pdf

[4] NIST. (2001). *Announcing the Advanced Encryption Standard (AES).* Federal Information Processing Standards Publication 197.

[5] GitHub Docs. *Repositorio implementación AES en python.* https://github.com/KevinHe23/AES-python.

[6] GitHub Docs. *Repositorio simulación app-web-aes.* https://github.com/KevinHe23/app-web-aes.