Pasos para entender AES

❖ CIFRADO:

Para la ejemplificación del cifrado aes, se hará uso del archivo cifrado_aes.pyc, el cual encontrará en el directorio **/var/www/AES**

AES trabaja el método de cifrado siguiendo la estructura mostrada en la Figura N° 4.3.1:

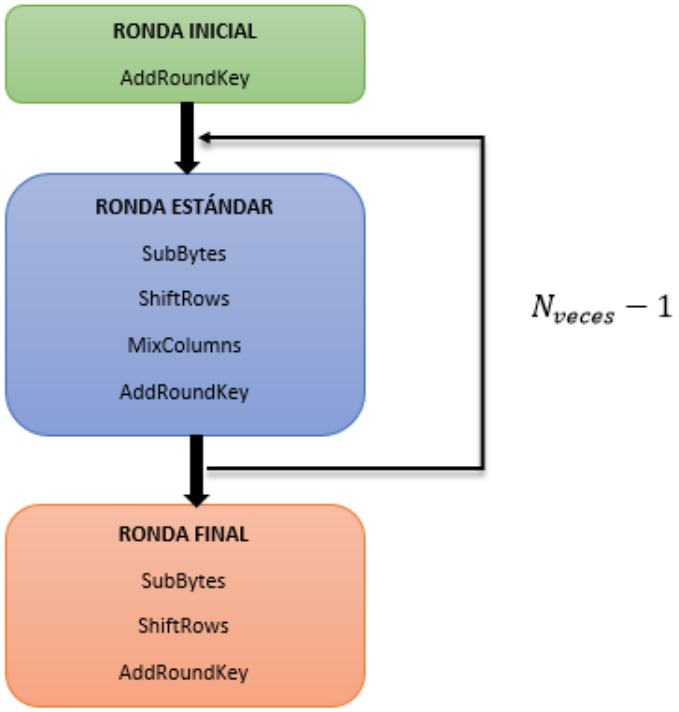

Figura N° 4.3.1: Cifrado AES.

RONDA INICIAL: Se realiza la operación XOR (⊕) entre el mensaje y la clave, elementos que constan de 16 bytes hexadecimales, los datos deben de ser acomodados en una matriz de 4x4 tomando en cuenta los índices de cada byte, como se muestra en la Figura N° 4.3.2:

00	04	08	12
01	05	09	13
02	06	10	14

03	07	11	15

Figura N° 4.3.2: Acomodo matricial de elementos

Copie el archivo cifrado_aes.pyc a su directorio AES y ejecute lo siguiente:

« python cifrado_aes.pyc --datos»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Datos»* que se encuentra en la práctica.

Teniendo los datos, los elementos, quedan de la siguiente forma:

Start of Round				**After SubBytes**				**After ShitfRows**				**After MixColumns**				**Round Key Value**			
32	88	31	e0													2b	28	ab	09
43	5a	31	37													7e	ae	f7	cf
f6	30	98	07													15	d2	15	4f
18	8d	a2	34													16	a6	88	3c

Figura N° 4.3.3: Ronda Inicial

En la Figura N° 4.3.3 se muestra el acomodo de los elementos para la ronda inicial, también conocida como ronda cero.

AddRoundKey: Haciendo uso de estos dos elementos, se realiza la operación $\oplus$ entre cada uno de los bytes del mensaje y la clave. Ejecute:

« python cifrado_aes.pyc --sround»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: SRound»* que se encuentra en la práctica.

El resultado final, son los nuevos valores de Start of Round para la primera ronda.

RONDA ESTÁNDAR:

En este caso, este proceso se realizará 9 veces, dado que la clave consta de 128 bits.

SubBytes: Para realizar este paso, se toman los valores obtenidos en *AddRoundKey*. En este paso, se hace uso de una matriz que costa de 16 filas y 16 columnas. Como ejemplo, se toma el primer byte del resultado obtenido anteriormente, es decir *«19»*, el número *«1»* se busca en la fila y el *«9»* en columna, la intersección de estas nos da un nuevo valor hexadecimal, que será el nuevo valor para dicha celda en la matriz.

Ejecute:

« python cifrado_aes.pyc --subbytes»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: SubBytes»* que se encuentra en la práctica. . La matriz que se utiliza se muestra en la Figura N° 4.3.4.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Figura N° 4.3.4: Matriz SubBytes

Los valores obtenidos al realizar este paso, son los valores colocados en el campo de After SubBytes para la primera ronda.

Shift Rows: Para realizar este paso, se toma en cuenta el resultado obtenido en *SubBytes*. En este paso se hace un corrimiento circular de *n* número de bytes como se indican en la Figura N° 4.3.5:

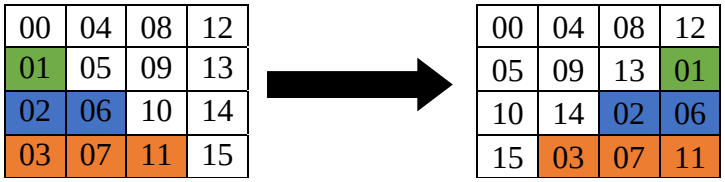

Figura N° 4.3.5: Corrimiento de bytes

Ejecute:

« python cifrado_aes.pyc --srows»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Shift Rows»* que se encuentra en la práctica.

Mix Columns: Para realizar este paso, se toma en cuenta el resultado obtenido en *Shift Rows*. Mix Columns hace uso del teorema de Galios, y su uso en AES, consta de distintos pasos:

1. Se hace una multiplicación de matrices, por lo cual se hace uso de la matriz obtenida en Shift Rows y una matriz llamada fija, esta es:

02	03	01	01
01	02	03	01

01	01	02	03
03	01	01	02

Figura N° 4.3.6: Matriz fija

Tomando el primer byte de cada una de las matrices se realiza: (02 • d4)[1], por lo cual, tomando como ejemplo para la Figura N° 4.3.7:

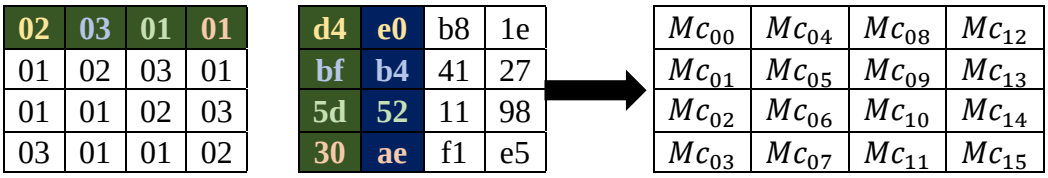

Figura N° 4.3.7: Multiplicación de matrices

El resultado para obtener el índice Mc_{00} es:

$$Mc_{00} = (02 \bullet \text{d4}) \oplus (03 \bullet \text{bf}) \oplus (01 \bullet 5\text{d}) \oplus (01 \bullet 30)$$

Del mismo modo se realizan para los demás índices, para Mc_{04} quedaría:

$$Mc_{04} = (02 \bullet \text{e0}) \oplus (03 \bullet \text{b4}) \oplus (01 \bullet 52) \oplus (01 \bullet \text{ae})$$

Para obtener todos los valores, se hace uso de las tablas L y E.

2. La tabla L se utiliza para obtener su valor equivalente, como lo realizado en SubBytes de cada uno de los elementos que se van a multiplicar.

   Como ejemplo, se toman los valores (02 • d4) de estos dos, se debe tomar su equivalente en la tabla L, se busca *«d»* en la fila y *«4»* en columna, lo mismo con *«0»* se busca en fila y el *«2»* en las columnas, los resultados son *«41»* y *«19»* respectivamente.

   De los dos valores obtenidos se realiza una suma de los hexadecimales, el resultado es *«5a»*.

3. Ahora con este resultado se obtiene su equivalente en la tabla E, del mismo modo, buscando la intersección de *«5»* en fila y *«a»* en columna. El resultado es *«b3»*

   Se realizan el mismo proceso para (03• bf), (01 • 5d), (01 • 30)

4. Se realiza la operación $\oplus$ con los 4 elementos hexadecimales obtenidos, es decir:

[1] El operador • es la multiplicación de un anillo de polinomios con coeficientes en GF(2^8). Para realizar dicha operación en el cifrado y descifrado AES, se hace uso de las tablas L y E.

$$Mc_{00} = \text{b3} \oplus \text{da} \oplus \text{5d} \oplus 30 = 04$$

La tabla L se muestra en la Figura N° 4.3.8

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

Figura N° 4.3.8: Tabla L.

La tabla E se muestra en la Figura N° 4.3.9

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

Figura N° 4.3.9: Tabla E.

Ejecute:

« python cifrado_aes.pyc --mcolumns»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Mix Columns»* que se encuentra en la práctica.

Finalmente se realiza nuevamente AddRoundKey, que se recuerda que es la operación ⊕, en este caso, entre la matriz correspondiente a Mix Columns y la subclave para esta ronda, el resultado será Start of Round de la siguiente ronda.

Obtención de las claves:

Teniendo la clave, se extrae la última columna, sombreada en color amarillo, de esta, el primer elemento pasa a ser el último, y de esta última columna se obtiene sus valores equivalentes, haciendo uso de la misma matriz usada en SubBytes. El proceso se muestra en la Figura N ° 4.3.10

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

→

2b	28	ab	cf
7e	ae	f7	4f
15	d2	15	3c
16	a6	88	09

→

2b	28	ab	8a
7e	ae	f7	84
15	d2	15	eb
16	a6	88	01

Figura N° 4.3.10: Primer paso para obtener claves

Posteriormente, se hace uso de la matriz RCON, la cual se muestra en la Figura N° 4.3.11:

10	10	10	10	10	10	10	10	10	10	10	10
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	00	00	00	00	00	00

Figura N° 4.3.11: Matriz RCON

En este caso, esta consta de 10 columnas dado que son 10 rondas a realizar.

Se realiza la operación ⊕ entre la primera columna de la clave, el resultado mostrado en la Figura N° 4.3.9 y la primera columna de la matriz RCON, esto por ser la primera clave, para la segunda ronda se hará uso de la segunda columna y asi sucesivamente, el resultado de dicha operación será la primera columna para la nueva clave, como se muestra en la Figura N° 4.3.12.

2b		8a		10		a0
7e	⊕	84	⊕	00	=	fa
15		eb		00		fe
16		01		00		17

Figura N° 4.3.12: Obtención de la primera columna para la primera subclave

Finalmente para obtener las siguientes columnas, se realiza un ⊕ de la columna correspondiente de la clave original y la columna del resultado anterior inmediato, es decir, para la columna dos,

se realiza la operación ⊕ con el resultado obtenido en la Figura N° 4.3.12. Para la columna tres, se realiza la operación ⊕ con el resultado obtenido para la columna dos y así sucesivamente.

Ejecute:

« python cifrado_aes.pyc --keys»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Sub Claves»* que se encuentra en la práctica.

Para obtener la siguiente Subclave, se realiza el mismo proceso, solo que ahora la subclave a utilizar será la obtenida en la ronda anterior, no la original.

RECORDANDO: En el apartado de **RONDA ESTANDAR** se menciona al final la realización de la AddRoundKey entre MixColumns y la Subclave, cuyo resultado será los bytes de Start of Round para la siguiente ronda. Para visualizar el resultado ejecute:

« python cifrado_aes.pyc --sroundmc»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: SRound MColumns - Subclave»* que se encuentra en la práctica.

MixColumns es el último paso de la ronda estándar, por lo cual, el proceso se realiza 9 veces. Ejecute:

« python cifrado_aes.pyc --restandar»

Lo anterior le permitirá observar los datos obtenidos en las 9 rondas.

Llene la tabla *«Actividad 4.3: Ronda 1 – Ronda 9»* que se encuentra en su práctica de acuerdo a los datos obtenidos al ejecutar lo anterior.

RONDA FINAL: La ronda final realiza las operaciones *SubBytes, ShiftRows y AddRoundKey*, además de obtener la *Subclave* correspondiente para dicha ronda.

El orden de dicha ronda es:

- *SRound*: Que es el resultado de ⊕ entre el resultado MixColumns y la Subclave correspondientes a la ronda nueve, es decir *AddRoundKey*
- *SubBytes:* Resultado de realizar dicha operación utilizando SRound.
- *SRows:* Resultado de realizar dicha operación utilizando el resultado SubBytes.
- *Key:* La clave correspondiente a la ronda diez.

Finalmente, para obtener la cadena correspondiente al cifrado, se realiza la operación ⊕ entre el resultado arrojado por *SRows* y *Key*. Ejecute:

« python cifrado_aes.pyc --rfinal»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Ronda Final»* que se encuentra en la práctica.

❖ **DESCIFRADO:**

Para la ejemplificación del descifrado aes, se hará uso del archivo descifrado_aes.pyc, el cual encontrará en el directorio **/var/www/AES**

AES trabaja el método de cifrado siguiendo la estructura mostrada en la Figura N° 4.3.13:

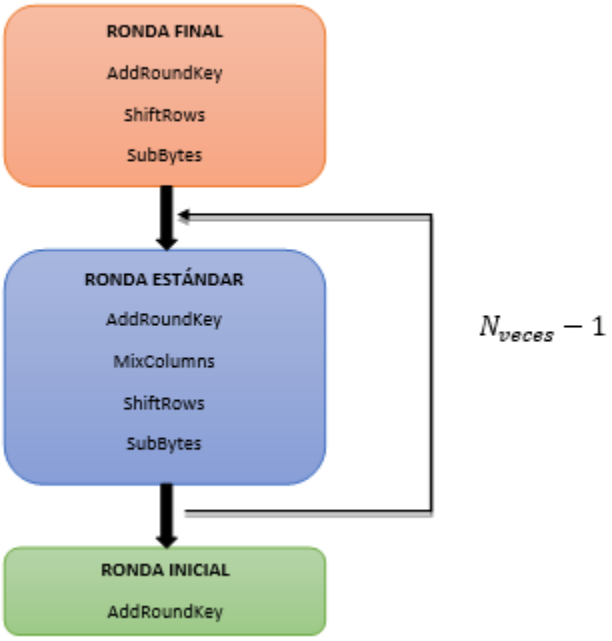

Figura N° 4.3.13: Descifrado AES

El descifrado AES es el proceso inverso del cifrado, por lo cual para que sea entendible, se comenzará por la explicación de la ronda final, no sin antes especificar que la obtención de las subclaves se realiza exactamente igual que en el cifrado, por lo cual, estas no sufren cambio alguno. Antes, ejecute:

python descifrado_aes.pyc --datos»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Datos descifrado»* que se encuentra en la práctica.

RONDA FINAL: Al ser el proceso inverso, recuerde, ¿Cuál fue el último paso realizado en el cifrado? Fue la realización de la operación XOR(⊕) entre el resultado de *Shift Rows* y la *Subclave[10]*, es decir *AddRoundKey*. Ahora, lo que tenemos es el mensaje cifrado, como se mencionó, las subclaves no sufren cambios por lo cual son las mismas a utilizar, entonces se realiza la operación ⊕ entre el *mensaje cifrado* y la *Subclave[10]*.

ACLARACIÓN: Para que no exista confusión, como entradas, el descifrado recibe una clave y un mensaje cifrado, para poder realizar satisfactoriamente el descifrado, la clave debe ser la misma

que se utilizó en el cifrado, por lo cual las subclaves puedes obtenerse a partir de la clave original y alojarse en alguna lista, para después hacer uso de ellas. Ejecute:

« python descifrado_aes.pyc --subclaves»

Lo que se muestra son las diez subclaves alojadas en una lista llamada subclaves. Cada una de estas será utilizada cuando se requiera, para este caso, se necesita la número diez, después se hará uso de la nueve y así sucesivamente.

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Subclaves descifrado»* que se encuentra en la práctica.

Ahora, ejecute:

« python descifrado_aes.pyc --daddroundkey»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: AddRoundKey descifrado»* que se encuentra en la práctica.

Compare ¿El resultado mostrado es el mismo al último paso realizado en el cifrado?

Siguiendo la lógica de ser el proceso inverso, los siguientes pasos a realizar son:

ShiftRows: En el caso de *Shift Rows*, la rotación de bytes ahora es hacia el lado izquierdo, como se muestra en la Figura N° 4.3.14, a este método se le conoce como *InvShiftRows*.

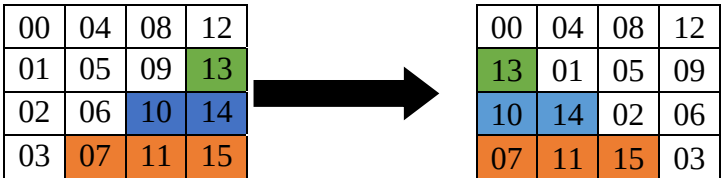

Figura N° 4.3.14: InvShiftRows

Recuerde que se hace uso de los valores obtenidos en AddRoundKey. Ejecute:

« python descifrado_aes.pyc --invsrows»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Inv Shift Rows»* que se encuentra en la práctica.

SubBytes: Conocido como *InvSubBytes*, realiza el mismo procedimiento que *SubBytes* en el cifrado, solo que ahora, hace uso de una matriz diferente. En la Figura N° 4.3.15 se muestra la matriz a utilizar.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Figura N° 4.3.14: Matriz inversa SubBytes

Recuerde que siempre hará uso de los datos obtenidos en el paso anterior. Ejecute:

« python descifrado_aes.pyc --invsubbytes»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Inv SubBytes»* que se encuentra en la práctica.

Para visualizar los datos ejecute: *« python descifrado_aes.pyc --rfinal»*

RONDA ESTANDAR: La ronda estándar comenzará con *AddRoundKey*, entre la subclave correspondiente, es decir la subclave nueve, y el último resultado obtenido, es decir, el *InvSubBytes* de la ronda anterior.

Mix Columns: Se realiza el mismo procedimiento que en el cifrado, solo que la matriz fija a utilizar es la siguiente:

0e	0b	0d	09
09	0e	0b	0d
0d	09	0e	0b
0b	0d	09	0e

Figura N° 4.3.15: Matriz fija Inv MixColumns

Recuerde que se utilizan los valores obtenidos en el paso anterior inmediato. Ejecute:

« python descifrado_aes.pyc --invmcolumns»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3: Inv Mix Columns»* que se encuentra en la práctica.

Los siguientes dos pasos ya los conoce, se realiza Shift Rows y SubBytes. El proceso se realiza nueve veces. Ejecute:

« python descifrado_aes.pyc --restandar»

Llene la tabla *«Ronda 9 – Ronda 1 Inversa»* que se encuentra en su práctica de acuerdo a los datos obtenidos al ejecutar lo anterior.

Visualice esta tabla con la tabla *«Actividad 4.3: Ronda 9 – Ronda 1»*. ¿Hay similitudes?

RONDA INICIAL: Finalmente se realiza la operación *AddRoundKey* entre la clave original y el último conjunto de bytes arrogado de la ronda uno, es decir, los valores correspondientes a *SubBytes*, finalmente se obtiene el mensaje original. Ejecute:

« python descifrado_aes.pyc --rinicial»

Coloque una captura de pantalla de lo mostrado en el recuadro *«Actividad 4.3Ronda inicial descifrado»* que se encuentra en la práctica.

Como observó, el descifrado es exactamente el proceso inverso al cifrado, solo tenga cuidado en hacer uso correcto de matrices para cada uno de los casos.