UNIVERSIDAD INTERNACIONAL SEK
ECUADOR

SEGURIDAD DE DATOS

Ing. José Luis Medina

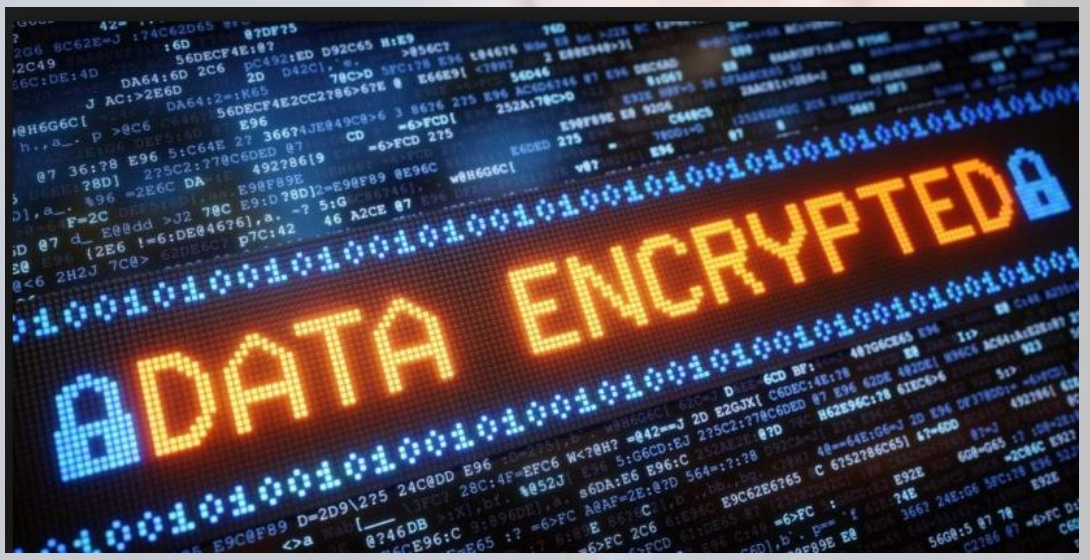
CRIPTOGRAFÍA - MODERNA

Principios de Kerckhoffs

- La efectividad del sistema no depende de la privacidad de su diseño.
- La clave debe ser fácil de recordar.
- Los criptogramas deberán dar resultados alfanuméricos.
- El sistema debe ser operable por una única persona.
- El sistema debe ser fácil de utilizar.
- Si el sistema no es teóricamente irrompible, al menos debe serlo en la práctica.

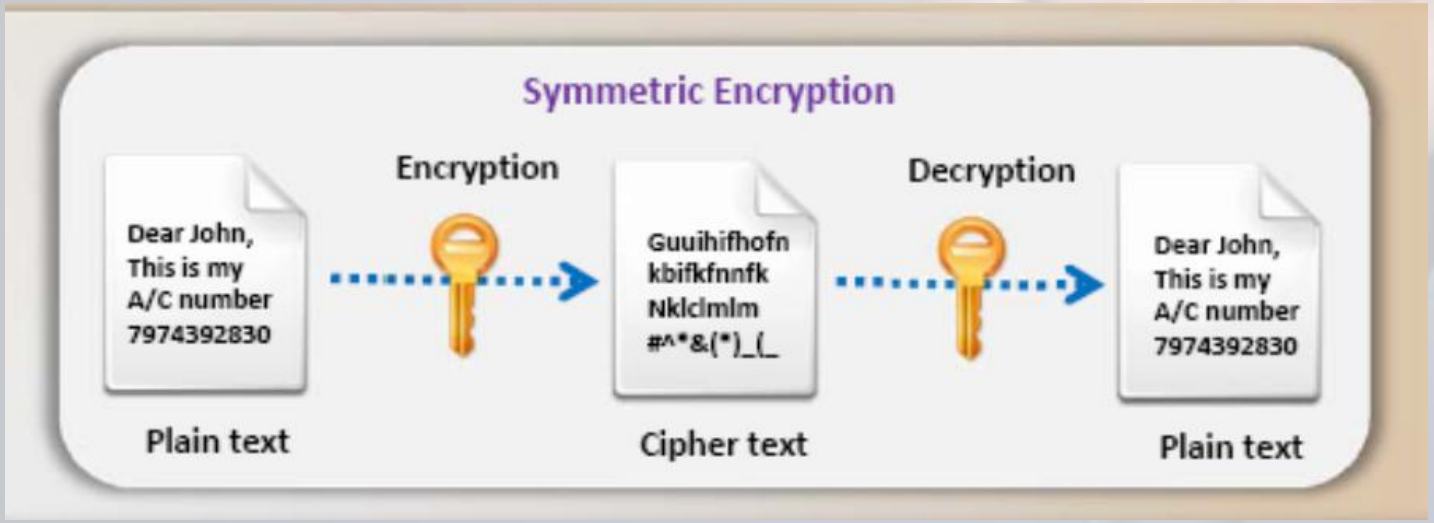
Criptografía - Moderna

- Dentro de la criptografía moderna, los algoritmos usados en sistemas de computación son mucho mas complejos, usan fórmulas matemáticas que indican las reglas como un texto plano debería convertirse en texto cifrado
- En algunas tecnologías de cifrado el origen y el destino utilizan la misma clave para ejecutar tareas de cifrado y descifrado, en otras tecnologías utilizan una clave para cifrar y otra clave para descifrar



Criptografía Simétrica

- Este método utiliza algoritmos que utilizan la misma clave para cifrar y descifrar la información
- A través de estos algoritmos se puede cifrar mensajes de correo electrónico, archivos de disco duro, registros de una base de datos, pero es necesario que las partes que deseen comunicarse de una manera segura conozcan la clave o llave para le ejecución de las operaciones.



- Si **10** personas necesitan comunicarse de una manera segura una con otra usando claves simétricas, entonces se requieren **45 claves** para el efecto
- Si **100** personas necesitan comunicarse de una manera segura una con otra usando claves simétricas, entonces ser requieren **4950 claves**
- La ecuación usada para calcular cuantas claves se requieren es:

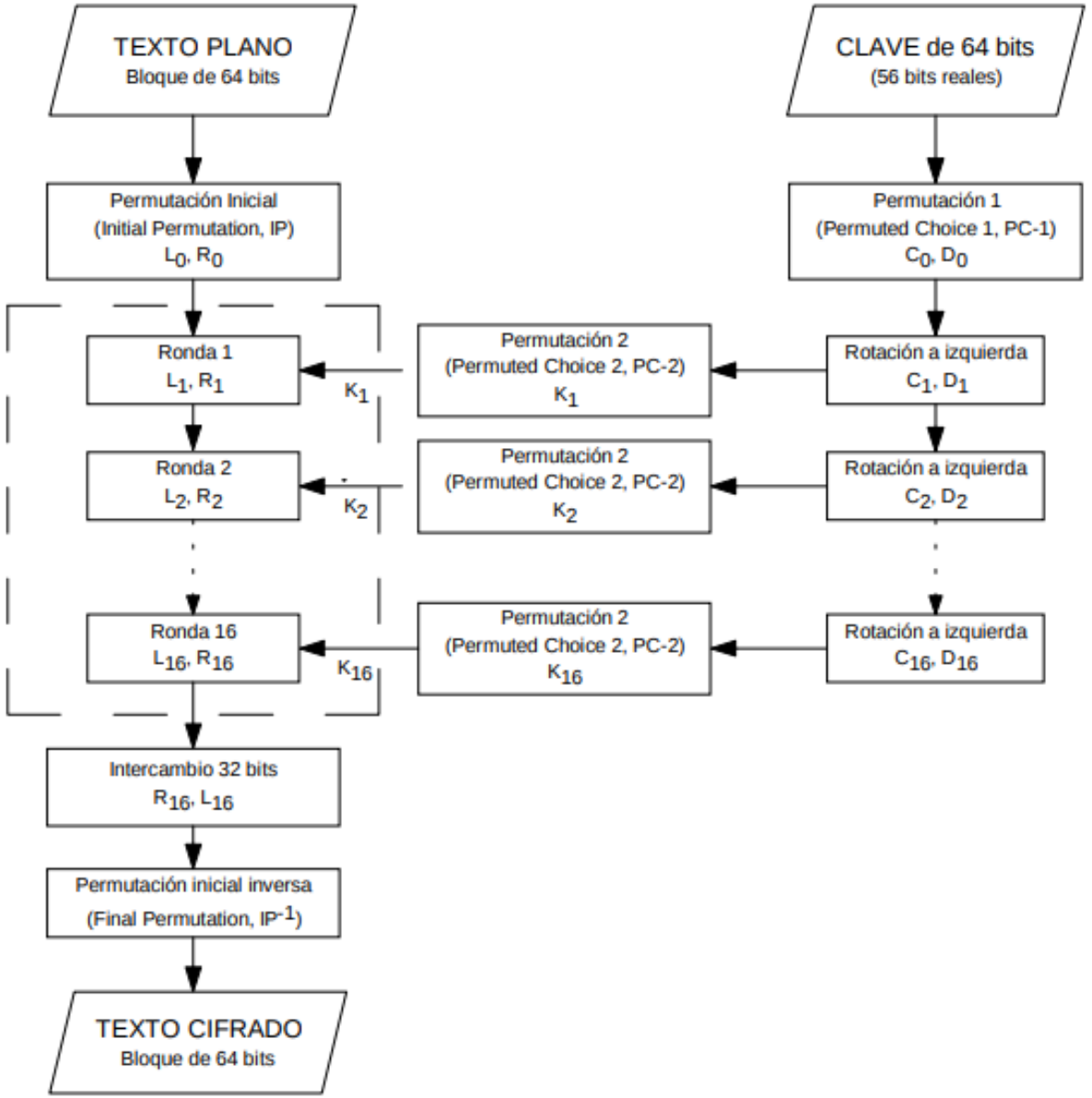
  N(N-1)/2 = número de claves

Simétrico - DES

- **D**ata **E**ncryption **S**tandard, ha sido implementado en la mayoría de productos comerciales usando criptografía
- Inicialmente propuesto en 1977 por el **NIST** (The National Insitute of Standards and Technology)
- Fue el esquema de cifrado más ampliamente usado
- El algoritmo en si se denomina **DEA** (**D**ata **E**ncryption **A**lgorithm)
- Fue retirado en mayo de 2005 por ya estar descontinuado

- La descripción del algoritmo para DES es la siguiente:
- Tamaño de bloque es de 64 bits (Texto)
- La clave utiliza 64 bits, sin embargo, realmente utiliza 56 bits puesto que 8 bits son destinados a la paridad
- Usa Feistel con pequeñas variaciones
- 16 rondas de procesamiento
- El proceso de descifrado es esencialmente el mismo que el de cifrado

(Data Encryption Standard)



Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Aplicada al **Texto** (**64 bits**)

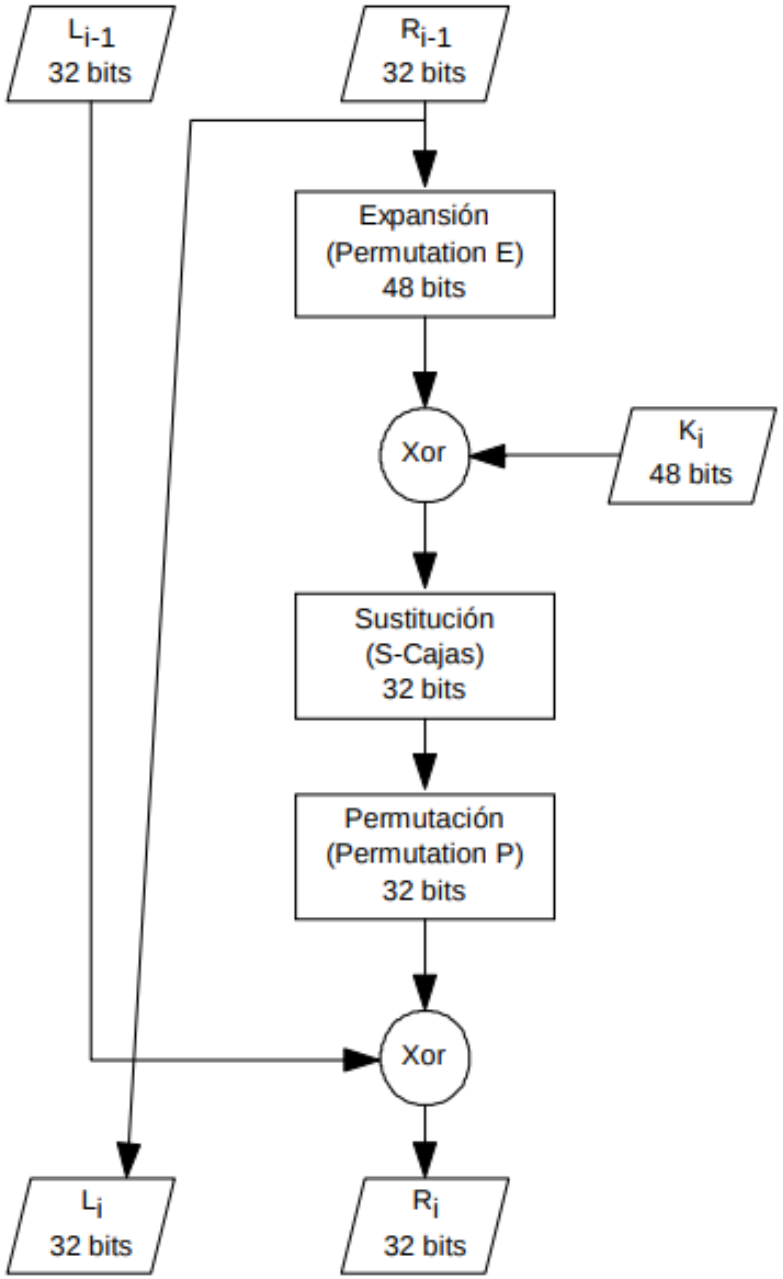
Permuted Choice 1 (PC-1)

57	49	41	33	25	17	9	1
58	50	42	34	26	18	10	2
59	51	43	35	27	19	11	3
60	52	44	36	63	55	47	39
31	23	15	7	62	54	46	38
30	22	14	6	61	53	45	37
29	21	13	5	28	20	12	4

Aplicada a la **clave** (**56 bits**)

Ronda	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits despl.	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

K1, K2....K16



Expansion (E)

32	1	2	3	4	5	4	5
6	7	8	9	8	9	10	11
12	13	12	13	14	15	16	17
16	17	18	19	20	21	20	21
22	23	24	25	24	25	26	27
28	29	28	29	30	31	32	1

Fila	Columna																S-Caja
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7	S_1
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0	
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13	
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10	S_2
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15	
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9	
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8	S_3
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7	
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12	
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15	S_4
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4	
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14	
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9	S_5
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14	
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3	
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11	S_6
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6	
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13	
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1	S_7
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2	
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12	
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7	S_8
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8	
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11	

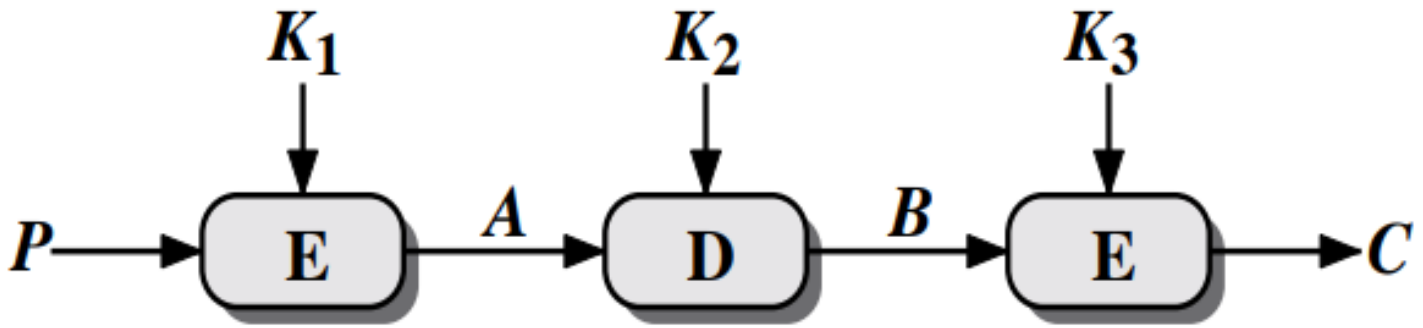
Tabla de permutación IP Inversa para el Descifrado

Permutación IP inversa

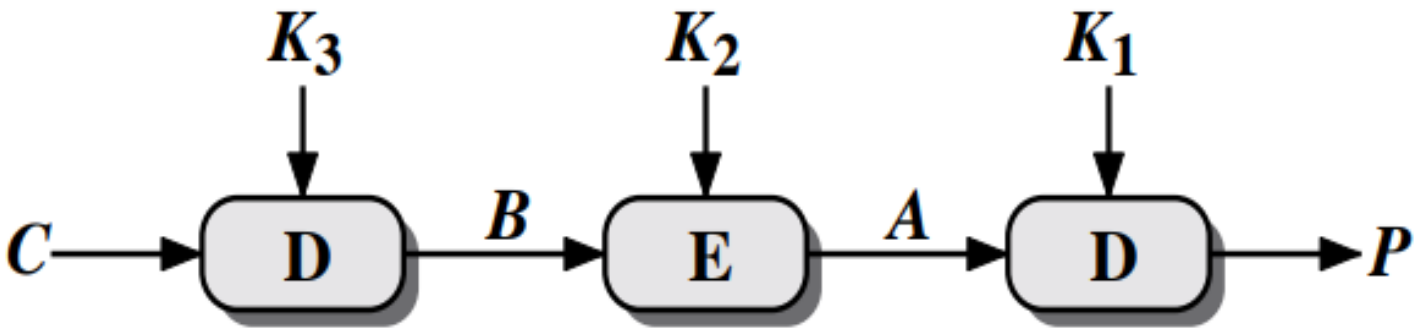
40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28	35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

Simétrico – Triple DES

- **Triple** **D**ata **E**ncryption **S**tandard o 3DES, fue propuesto por primera vez en 1985 [X9.17], se añadió al estandar DES en 1999 [FIPS 46-3]
- El tamaño efectivo de la clave es de 168 bits
- El algoritmo sigue siendo **DEA**, pero para este caso sería **TDEA**



(a) Encryption

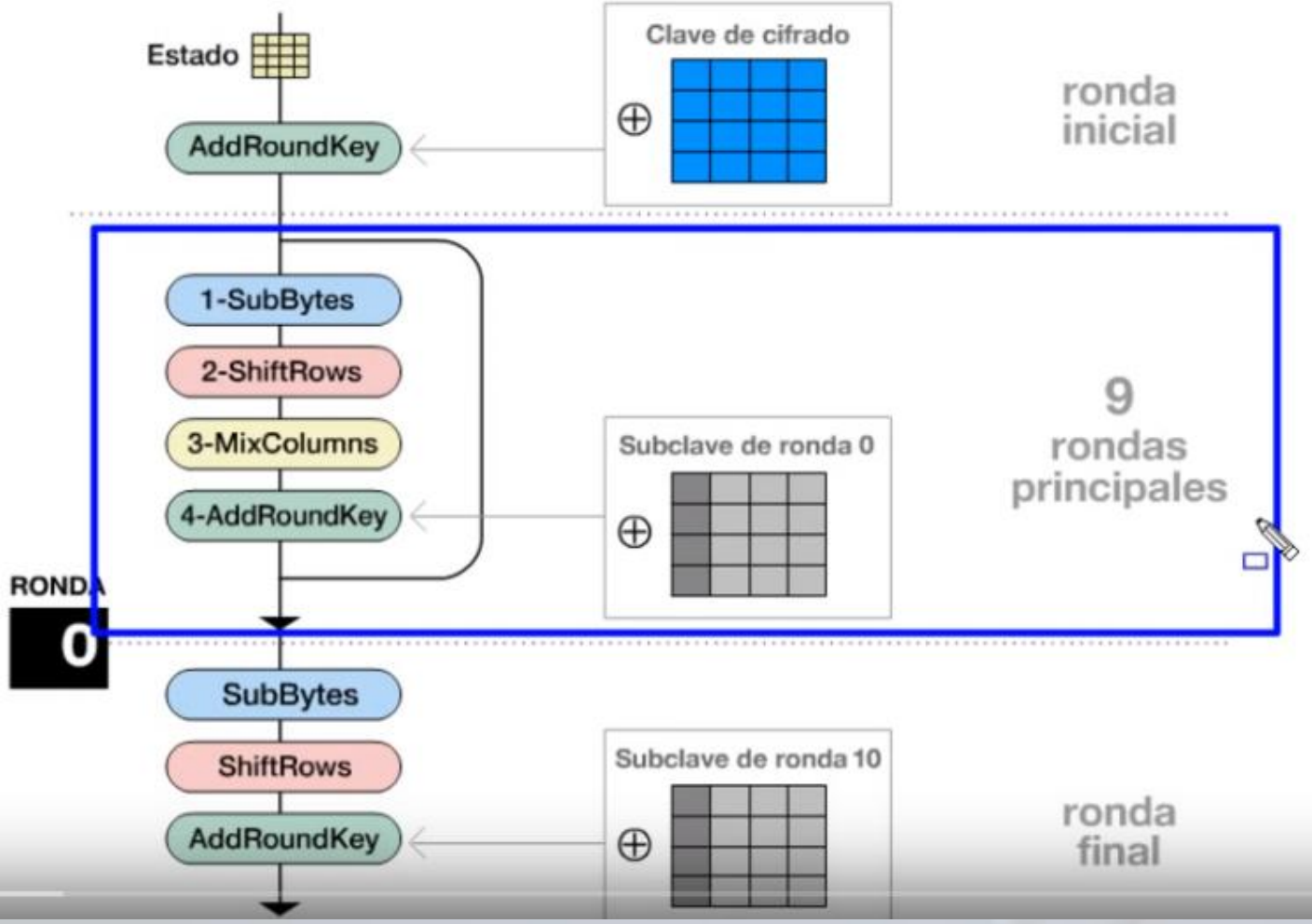


(b) Decryption

Simétrico – AES

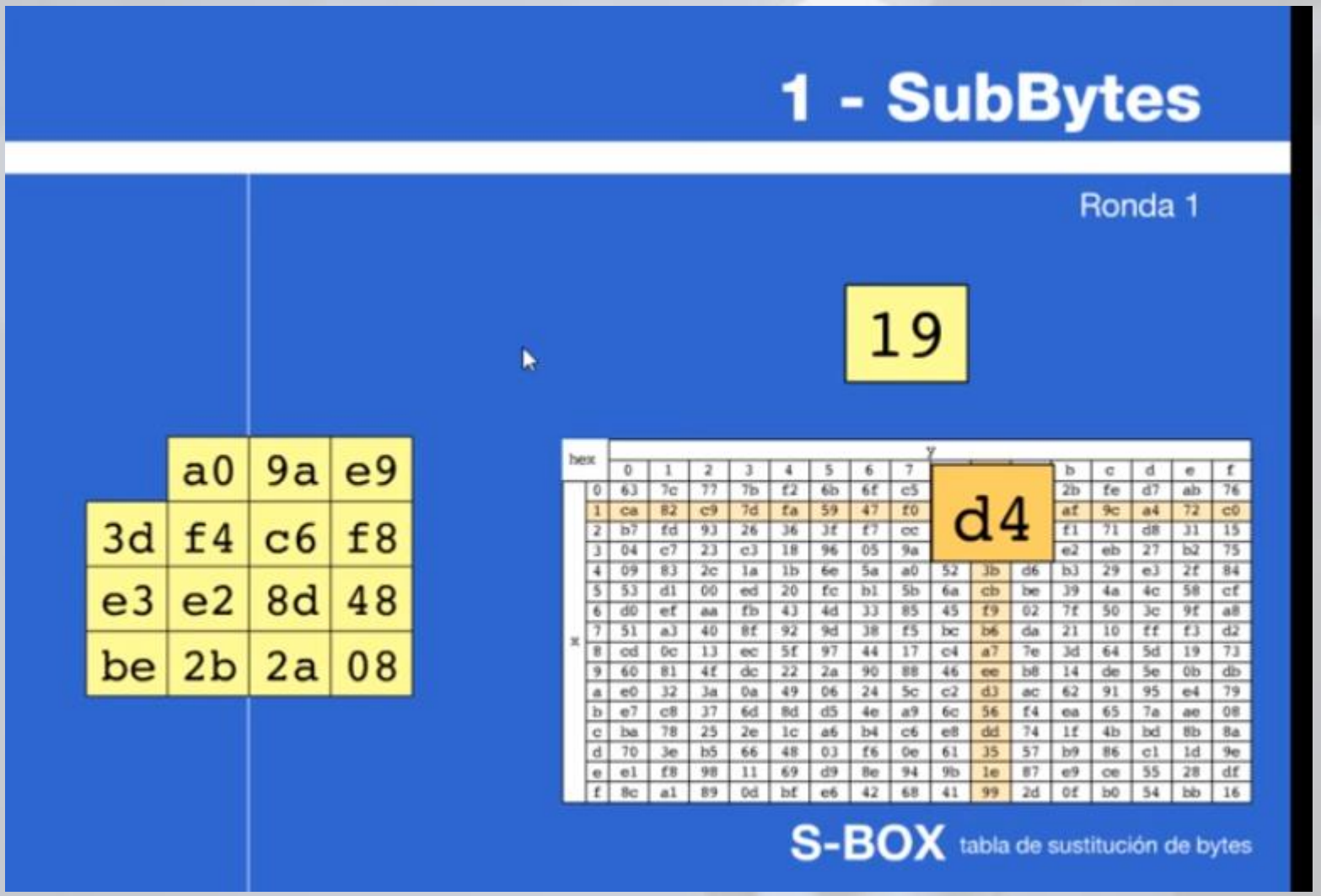
- **A**dvanced **E**ncryption **S**tandard**,** después de que DES fue el estandard utilizado por alrededor de 20 años y fue crackeado, la NIST decidió un nuevo estandard y fue seleccionado AES
- Utiliza cifrado simétrico de bloque, con tamaños de bloque de 128 bits y soportar claves de 128, 192 y 256 bits
- **AES** es el estandard utilizado en la actualidad

Proceso de Cifrado



Los 4 tipos de transformación:

- 1-SubBytes
- 2-ShiftRows
- 3-MixColumns
- 4-AddRoundKey



2 - ShiftRows

Ronda 1

d4	e0	b8	1e
bf	b4	41	27
5d	52		98
f1	e5	30	

ae

rotación de 3 bytes

3 - MixColumns

Ronda 1

e0	b8	1e	
b4	41	27	
52	11	98	
ae	f1	e5	

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \cdot \begin{bmatrix} d4 \\ bf \\ 5d \\ 30 \end{bmatrix}$$

Los cuatro bytes de cada columna son multiplicados dentro del Campo de Galois por una determinada matriz.

4 - AddRoundKey

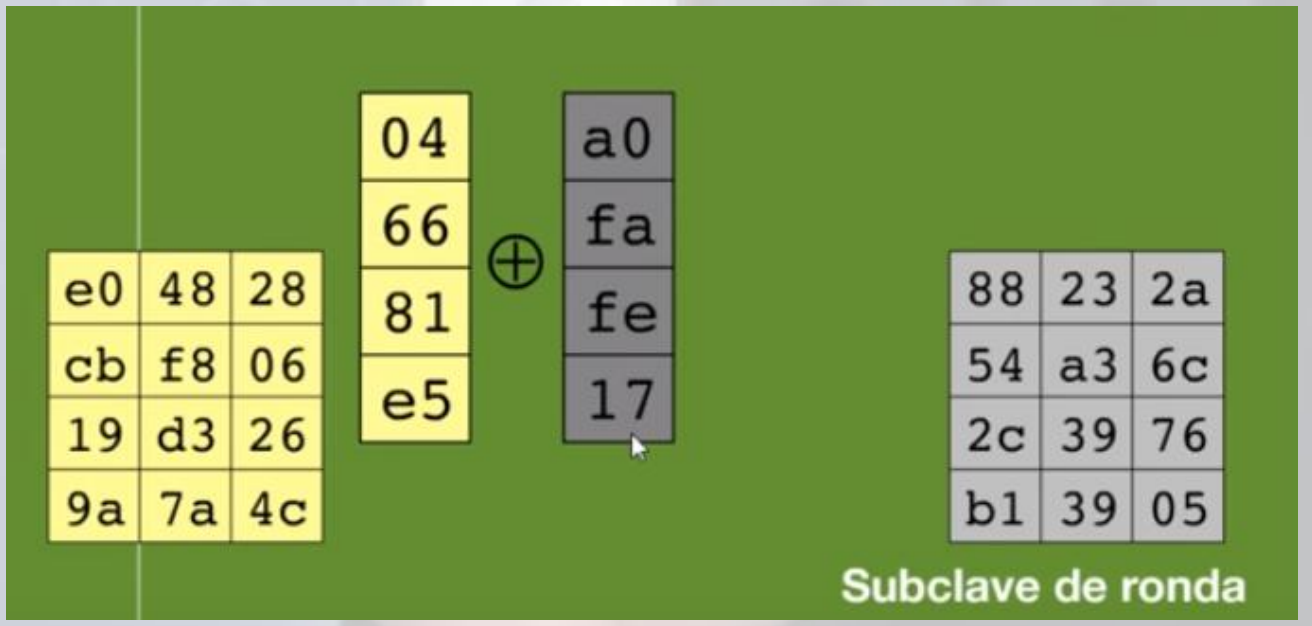
Ronda 1

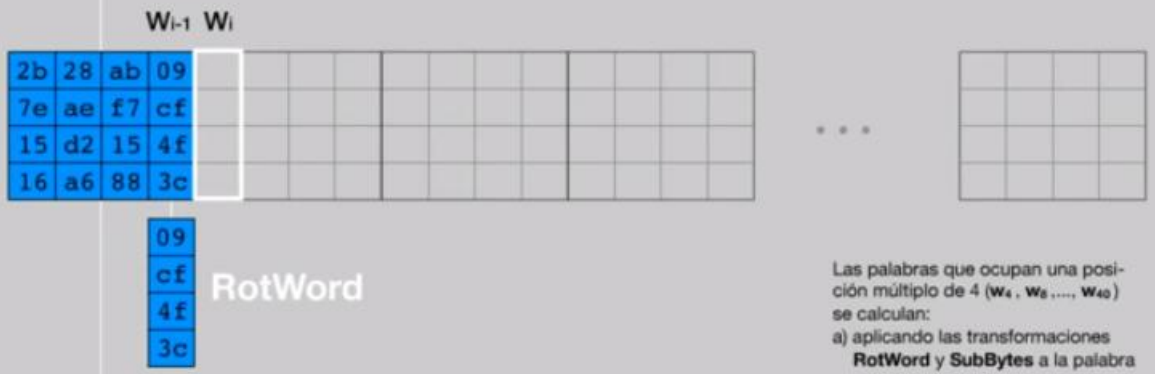
04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

Subclave de ronda

(producida como subclave de la ronda 1 durante el cálculo de las subclaves – ver diapositiva 19)

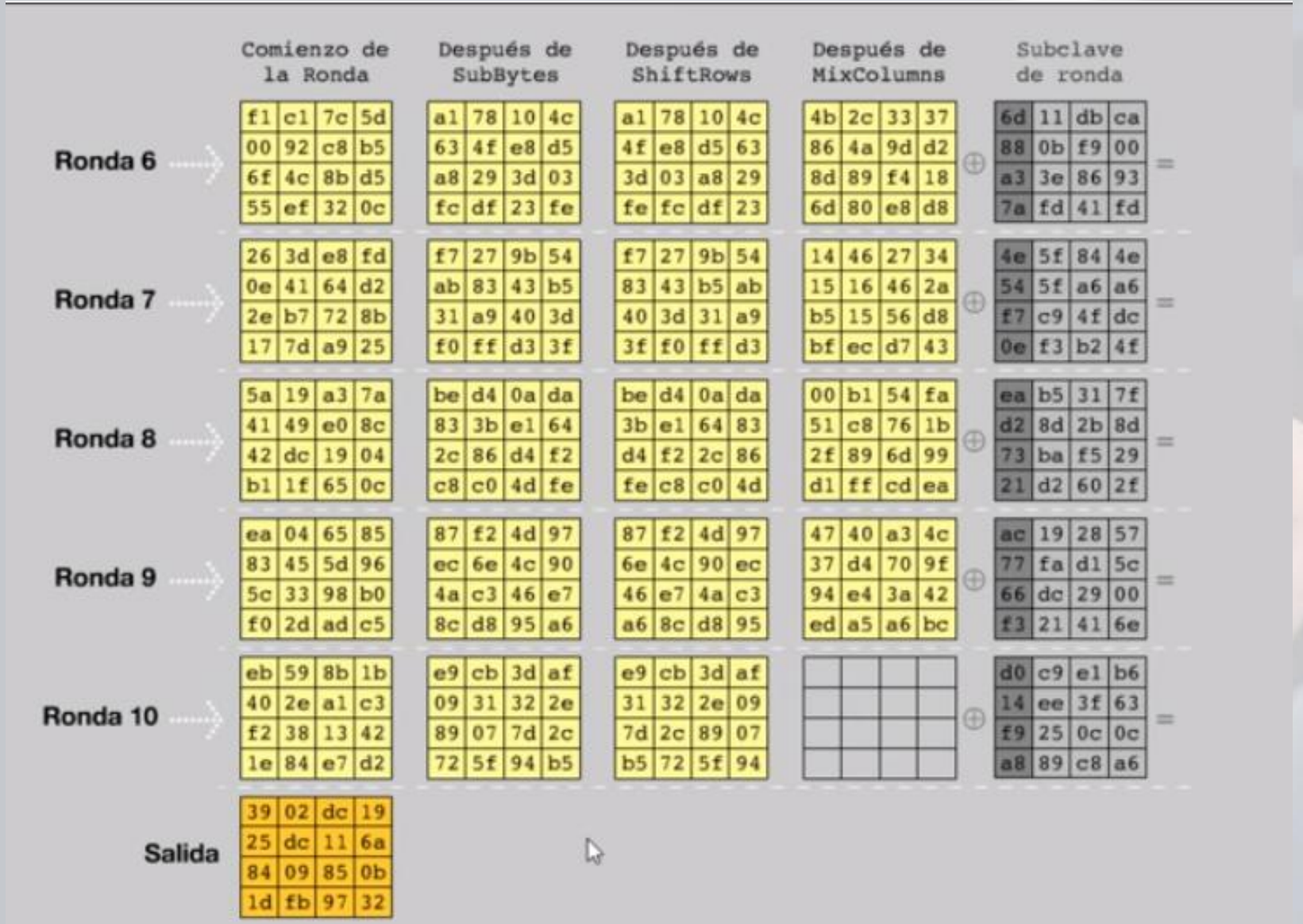




Las palabras que ocupan una posición múltiplo de 4 (w_4, w_8, ..., w_{40}) se calculan:

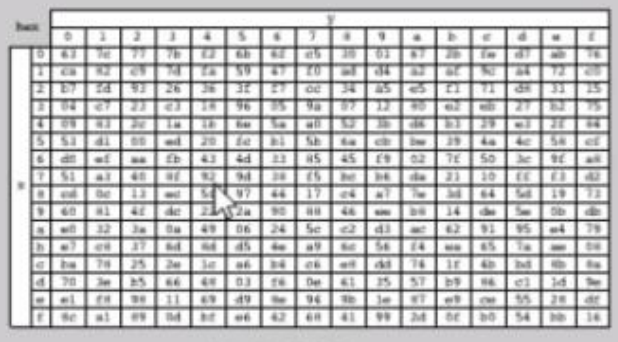
a) aplicando las transformaciones **RotWord** y **SubBytes** a la palabra anterior w_{i-1}.

	Comienzo de la Ronda	Después de SubBytes	Después de ShiftRows	Después de MixColumns	Subclave de ronda
Entrada	32 88 31 e0 43 5a 31 37 f6 30 98 07 a8 8d a2 34				2b 28 ab 09 7e ae f7 cf 15 d2 15 4f 16 a6 88 3c
Ronda 1	19 a0 9a e9 3d f4 c6 f8 e3 e2 8d 48 be 2b 2a 08	d4 e0 b8 1e 27 bf b4 41 11 98 5d 52 ae f1 e5 30	d4 e0 b8 1e bf b4 41 27 5d 52 11 98 30 ae f1 e5	04 e0 48 28 66 cb f8 06 81 19 d3 26 e5 9a 7a 4c	a0 88 23 2a fa 54 a3 6c fe 2c 39 76 17 b1 39 05
Ronda 2	a4 68 6b 02 9c 9f 5b 6a 7f 35 ea 50 f2 2b 43 49	49 45 7f 77 de db 39 02 d2 96 87 53 89 f1 1a 3b	49 45 7f 77 db 39 02 de 87 53 d2 96 3b 89 f1 1a	58 1b db 1b 4d 4b e7 6b ca 5a ca b0 f1 ac a8 e5	f2 7a 59 73 c2 96 35 59 95 b9 80 f6 f2 43 7a 7f
Ronda 3	aa 61 82 68 8f dd d2 32 5f e3 4a 46 03 ef d2 9a	ac ef 13 45 73 c1 b5 23 cf 11 d6 5a 7b df b5 b8	ac ef 13 45 c1 b5 23 73 d6 5a cf 11 b8 7b df b5	75 20 53 bb ec 0b c0 25 09 63 cf d0 93 33 7c dc	3d 47 1e 6d 80 16 23 7a 47 fe 7e 88 7d 3e 44 3b
Ronda 4	48 67 4d d6 6c 1d e3 5f 4e 9d b1 58 ee 0d 38 e7	52 85 e3 f6 50 a4 11 cf 2f 5e c8 6a 28 d7 07 94	52 85 e3 f6 a4 11 cf 50 c8 6a 2f 5e 94 28 d7 07	0f 60 6f 5e d6 31 c0 b3 da 38 10 13 a9 bf 6b 01	ef a8 b6 db 44 52 71 0b a5 5b 25 ad 41 7f 3b 00
Ronda 5	e0 c8 d9 85 92 63 b1 b8 7f 63 35 be e8 c0 50 01	e1 e8 35 97 4f fb c8 6c d2 fb 96 ae 9b ba 53 7c	e1 e8 35 97 fb c8 6c 4f 96 ae d2 fb 7c 9b ba 53	25 bd b6 4c d1 11 3a 4c a9 d1 33 c0 ad 68 8e b0	d4 7c ca 11 d1 83 f2 f9 c6 9d b8 15 f8 87 bc bc


Comienzo de la Ronda
Después de SubBytes
Después de ShiftRows
Después de MixColumns
Subclave de ronda
Ronda 6
f1 c1 7c 5d
00 92 c8 b5
6f 4c 8b d5
55 ef 32 0c
a1 78 10 4c
63 4f e8 d5
a8 29 3d 03
fc df 23 fe
a1 78 10 4c
4f e8 d5 63
3d 03 a8 29
fe fc df 23
4b 2c 33 37
86 4a 9d d2
8d 89 f4 18
6d 80 e8 d8
6d 11 db ca
88 0b f9 00
a3 3e 86 93
7a fd 41 fd
Ronda 7
26 3d e8 fd
0e 41 64 d2
2e b7 72 8b
17 7d a9 25
f7 27 9b 54
ab 83 43 b5
31 a9 40 3d
f0 ff d3 3f
f7 27 9b 54
83 43 b5 ab
40 3d 31 a9
3f f0 ff d3
14 46 27 34
15 16 46 2a
b5 15 56 d8
bf ec d7 43
4e 5f 84 4e
54 5f a6 a6
f7 c9 4f dc
0e f3 b2 4f
Ronda 8
5a 19 a3 7a
41 49 e0 8c
42 dc 19 04
b1 1f 65 0c
be d4 0a da
83 3b e1 64
2c 86 d4 f2
c8 c0 4d fe
be d4 0a da
3b e1 64 83
d4 f2 2c 86
fe c8 c0 4d
00 b1 54 fa
51 c8 76 1b
2f 89 6d 99
d1 ff cd ea
ea b5 31 7f
d2 8d 2b 8d
73 ba f5 29
21 d2 60 2f
Ronda 9
ea 04 65 85
83 45 5d 96
5c 33 98 b0
f0 2d ad c5
87 f2 4d 97
ec 6e 4c 90
4a c3 46 e7
8c d8 95 a6
87 f2 4d 97
6e 4c 90 ec
46 e7 4a c3
a6 8c d8 95
47 40 a3 4c
37 d4 70 9f
94 e4 3a 42
ed a5 a6 bc
ac 19 28 57
77 fa d1 5c
66 dc 29 00
f3 21 41 6e
Ronda 10
eb 59 8b 1b
40 2e a1 c3
f2 38 13 42
1e 84 e7 d2
e9 cb 3d af
09 31 32 2e
89 07 7d 2c
72 5f 94 b5
e9 cb 3d af
31 32 2e 09
7d 2c 89 07
b5 72 5f 94
d0 c9 e1 b6
14 ee 3f 63
f9 25 0c 0c
a8 89 c8 a6
Salida
39 02 dc 19
25 dc 11 6a
84 09 85 0b
1d fb 97 32

Cálculo de las Subclaves

SubBytes



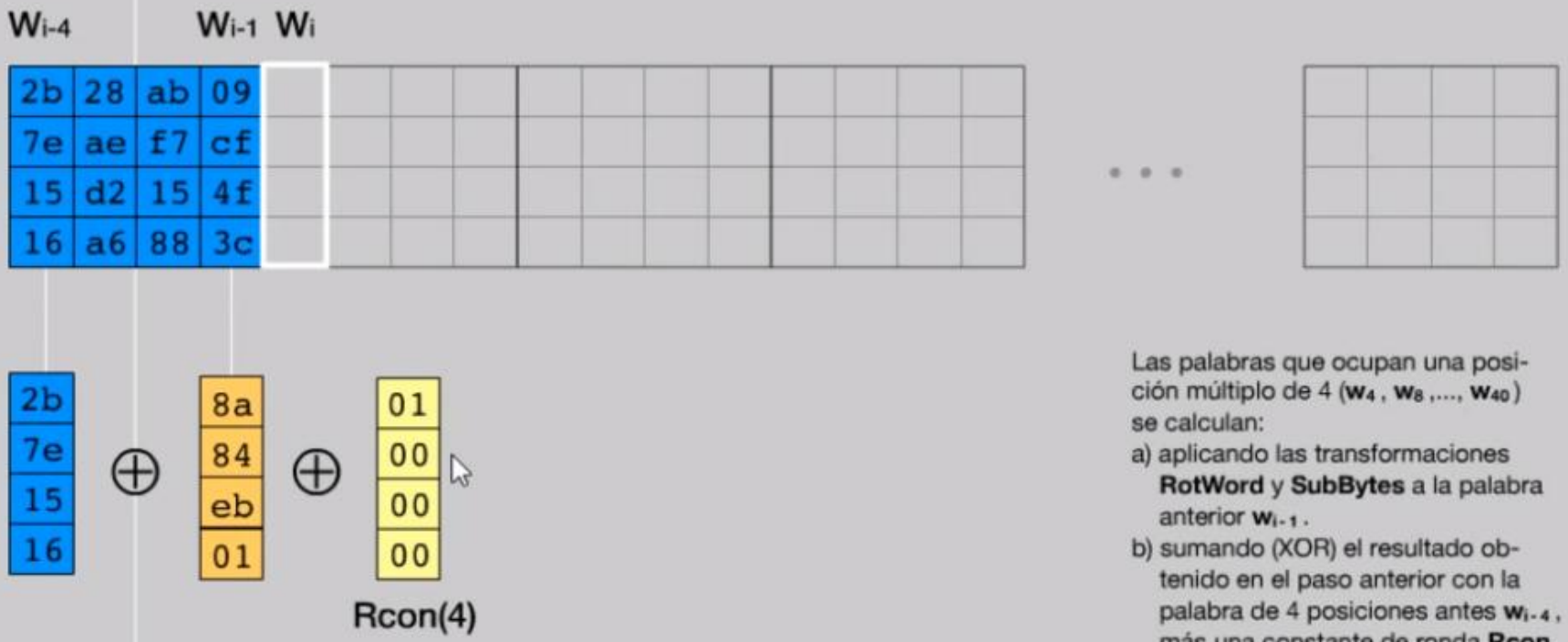
S-BOX tabla de sustitución de bytes

Las palabras que ocupan una posición múltiplo de 4 (w_4 , w_8 ,..., w_{40}) se calculan:

a) aplicando las transformaciones **RotWord** y **SubBytes** a la palabra anterior w_{i-1} .

Rcon

Cálculo de las Subclaves



Las palabras que ocupan una posición múltiplo de 4 (w_4, w_8,..., w_{40}) se calculan:

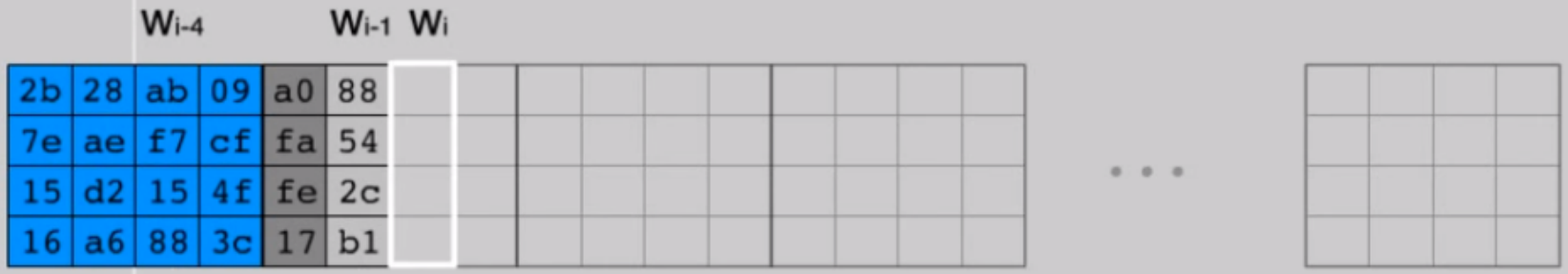
a) aplicando las transformaciones **RotWord** y **SubBytes** a la palabra anterior w_{i-1}.

b) sumando (XOR) el resultado obtenido en el paso anterior con la palabra de 4 posiciones antes w_{i-4}, más una constante de ronda **Rcon**.

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

Rcon

Cálculo de las Subclaves

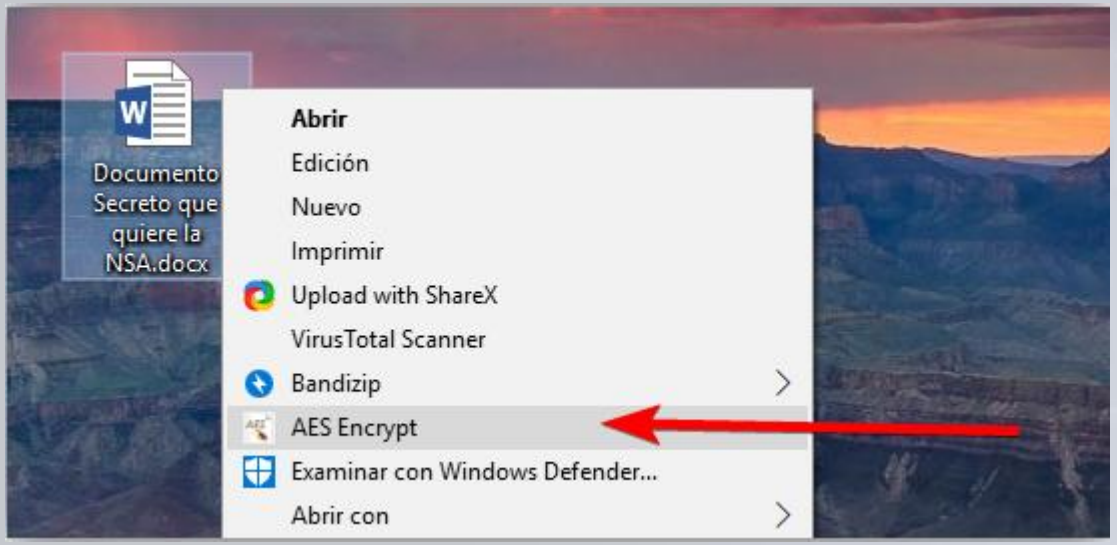


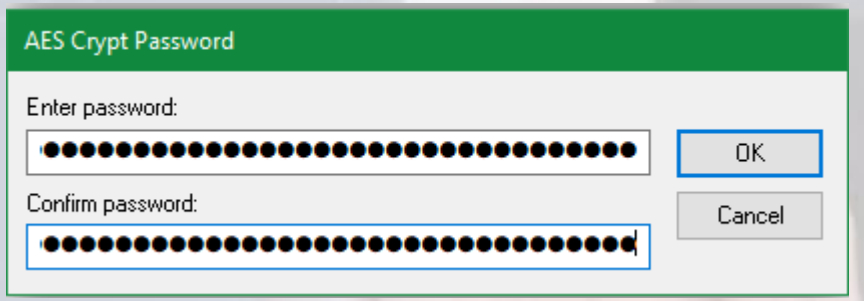
23
a3
39
39

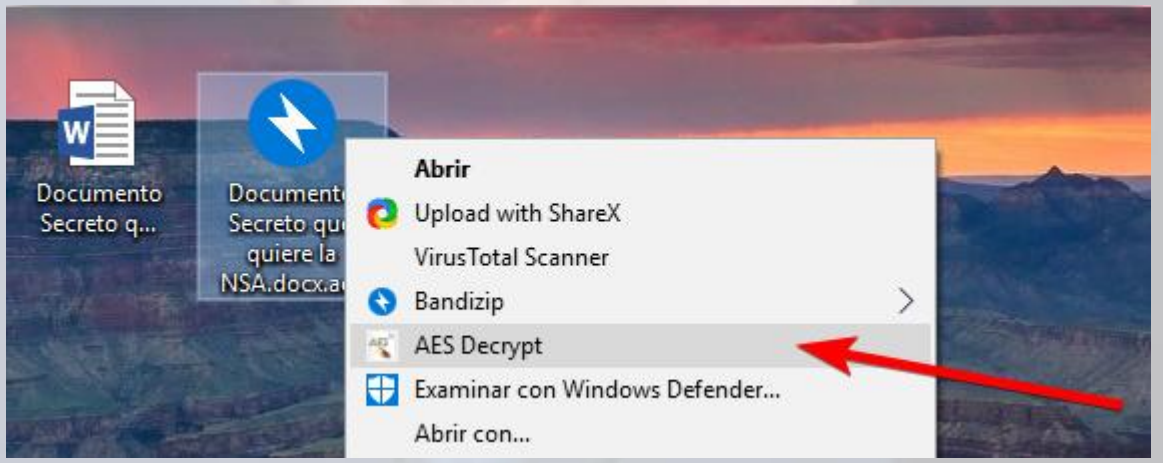
Las restantes palabras de 32 bits w_i se calculan sumando (XOR) la palabra anterior w_{i-1} con la palabra de cuatro posiciones antes w_{i-4}.

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

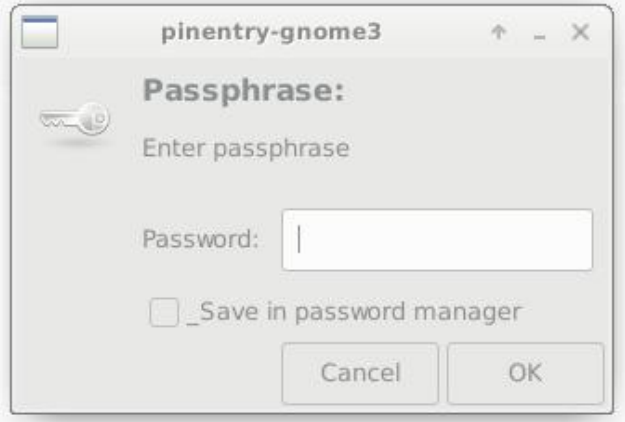
Rcon


Documento Secreto que quiere la NSA.docx
Abrir
Edición
Nuevo
Imprimir
Upload with ShareX
VirusTotal Scanner
Bandizip
AES Encrypt
Examinar con Windows Defender...
Abrir con


AES Crypt Password
Enter password:
Confirm password:
OK
Cancel


Documento Secreto q...
Abrir
Upload with ShareX
VirusTotal Scanner
Bandizip
AES Decrypt
Examinar con Windows Defender...
Abrir con...

```
joseluis@themordor:~$ gpg --symmetric --cipher-algo AES256 mensaje.txt
```




```
joseluis@themordor:~$ gpg --version
gpg (GnuPG) 2.1.18
libgcrypt 1.7.6-beta
Copyright (C) 2017 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Home: /home/joseluis/.gnupg
Supported algorithms:
Pubkey: RSA, ELG, DSA, ECDH, ECDSA, EDDSA
Cipher: IDEA, 3DES, CAST5, BLOWFISH, AES, AES192, AES256, TWOFISH,
        CAMELLIA128, CAMELLIA192, CAMELLIA256
Hash: SHA1, RIPEMD160, SHA256, SHA384, SHA512, SHA224
Compression: Uncompressed, ZIP, ZLIB, BZIP2
joseluis@themordor:~$
```

```
joseluis@themordor:~$ gpg -d mensaje.txt.gpg
gpg: AES256 encrypted data
gpg: encrypted with 1 passphrase
Este es una mensaje de texto que lo voy ha firmar electrónicamente

Saludos,

Jose Luis Medina
joseluis@themordor:~$
```

Algoritmos - Asimétricos

- **A**dvanced **E**ncryption **S**tandard**,** después de que DES fue el estandard utilizado por alrededor de 20 años y fue crackeado, la NIST decidió un nuevo estandard y fue seleccionado AES
- Utiliza cifrado simétrico de bloque, con tamaños de bloque de 128 bits y soportar claves de 128, 192 y 256 bits
- **AES** es el estandard utilizado en la actualidad

- El Criptosistema **RSA**: se basa en la dificultad computacional de la factorización de números enteros muy grandes
- El Criptosistema **ElGamal**: se basa en la dificultad computacional que supone resolver el problema de un logaritmo discreto
- EL Criptosistema basado en **Curvas Elípticas**: es una variante mejorada y mas eficiente que el criptosistema ELGamal (La seguridad que proporciona ElGamal con claves de longitud de 1024 bits **NO** es superior a la generada por las curvas elípticas con longitudes de clave de 160 bits)

- Dentro de las **Ventajas** de un criptposistema de **Clave Pública** tenemos:
  - Eficacia en la gestión y distribución de claves
  - La vida útil de las claves es de unos 2 años (aprox.), superior a la de las claves
  - utilizadas en los criptosistemas de clave secreta
  - Se pueden diseñar a partir de ellos esquemas de firma digital
- Dentro de las **Desventajas** de un criptosistema de **Clave Pública** tenemos:
  - La longitud de las claves (1024-4096 bits) es superior a la utilizada en los
  - criptosistemas de clave secreta.
  - Lentitud en el proceso de cifrado/descifrado.
  - Poco eficaces a la hora de cifrar grandes cantidades de datos.
  - Los algoritmos se basan en complejos resultados matemáticos

Algoritmos - RSA

Rivest, Shamir y Adleman

- **R**ivest, **S**hamir, y **A**dleman
- Hoy en día es el algoritmo más usado en clave pública
- Puede usarse para cifrar y descifrar archivos, así como también para aplicarlo en firma electrónica
- Usa diferente longitud de clave: 512, 768, 1024, 2048 bits
- La seguridad reside en la dificultad del factoreo de números enteros grandes

Algorithm Key generation for the RSA signature scheme

SUMMARY: each entity creates an RSA public key and a corresponding private key. Each entity A should do the following:

1. Generate two large distinct random primes p and q, each roughly the same size.
2. Compute $n = pq$ and $\phi = (p-1)(q-1)$.
3. Select a random integer e, $1 < e < \phi$, such that $\gcd(e, \phi) = 1$.
4. Use the extended Euclidean algorithm (Algorithm 2.107) to compute the unique integer d, $1 < d < \phi$, such that $ed \equiv 1 \pmod{\phi}$.
5. A's public key is (n, e); A's private key is d.

Algorithm RSA signature generation and verification

SUMMARY: entity A signs a message $m \in \mathcal{M}$. Any entity B can verify A's signature and recover the message m from the signature.

1. *Signature generation.* Entity A should do the following:
   (a) Compute $\widetilde{m} = R(m)$, an integer in the range $[0, n-1]$.
   (b) Compute $s = \widetilde{m}^d \bmod n$.
   (c) A's signature for m is s.
2. *Verification.* To verify A's signature s and recover the message m, B should:
   (a) Obtain A's authentic public key (n, e).
   (b) Compute $\widetilde{m} = s^e \bmod n$.
   (c) Verify that $\widetilde{m} \in \mathcal{M}_R$; if not, reject the signature.
   (d) Recover $m = R^{-1}(\widetilde{m})$.

- Los algoritmos enviados usando el **RSA** se representan mediante números y el funcionamiento se basa en el producto de dos números primos grandes (mayores de 10 a 100 cifras) elegidos al azar para conformar la clave de cifrado
- La seguridad de este algoritmo radica en que no hay maneras rápidas conocidas de factorizar un número grande en sus factores primos utilizando para ello computadores tradicionales

```
P  = 61    <= first prime number (destroy this after computing E and D)
Q  = 53    <= second prime number (destroy this after computing E and D)
PQ = 3233  <= modulus (give this to others)
E  = 17    <= public exponent (give this to others)
D  = 2753  <= private exponent (keep this secret!)

Your public key is (E,PQ).
Your private key is D.

The encryption function is: encrypt(T) = (T^E) mod PQ
                                       = (T^17) mod 3233

The decryption function is: decrypt(C) = (C^D) mod PQ
                                       = (C^2753) mod 3233

To encrypt the plaintext value 123, do this:

encrypt(123) = (123^17) mod 3233
             = 337587917446653715596592958817679803 mod 3233
             = 855

To decrypt the cipher text value 855, do this:

decrypt(855) = (855*2753) mod 3233
             = 123
```

- **RSA Cifrado**:

Se utiliza $k_{Pb} = (e, N) = (45, 21331)$
Se realiza la potenciación $c = m^e$ mód (N) de la siguiente manera:

1. Se clacula el número de letras por bloque: $log_{27} N = \frac{logN}{log27} = log_{27} 21331 = 3,02439626... \rightarrow$ 3 letras por bloque
2. Se calcula por bloques:
   SEC $= 20\ 5\ 3 = 20(27)^0 + 5(27)^1 + 3(27)^2 = 2342$
   RET $= ...$
3. Se cifra por bloques: 2342^{45} mód $(21331) = 12635$
   ...
4. Se manda $c = (12635\#...\#...)$

- **RSA Descifrado**:

Se utiliza $k_{Pv} = (d, N) = (933, 21331)$
Se realiza la potenciación $m = c^d$ mód (N) de la siguiente manera:

1. Se descifra: 12635^{933} mód $(21331) = 2342$

   ...

2. Se recupera el mensaje:
   $2342 = 27(86) + 20$; $86 = 27(3) + 5$; $3 = 27(0) + 3$

   ...

   m = 20 5 3 ... = SEC ...

- # Instalación de GnuPG:

```
joseluis — joseluis@ty
root@tyrio:~# apt-get install gnupg
Reading package lists... Done
Building dependency tree
Reading state information... Done
gnupg is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
root@tyrio:~#
```

```
root@tyrio:~# gpg --gen-key
gpg (GnuPG) 1.4.18; Copyright (C) 2014 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection? 1
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048)
Requested keysize is 2048 bits
Please specify how long the key should be valid.
         0 = key does not expire
      <n>  = key expires in n days
      <n>w = key expires in n weeks
      <n>m = key expires in n months
      <n>y = key expires in n years
Key is valid for? (0) 2y
Key expires at Fri 09 Nov 2018 07:08:24 PM ECT
Is this correct? (y/N) y

You need a user ID to identify your key; the software constructs the user ID
from the Real Name, Comment and Email Address in this form:
    "Heinrich Heine (Der Dichter) <heinrichh@duesseldorf.de>"

Real name: Jose Luis Medina [redacted]
Email address: [redacted]@gmail.com
Comment: Clave de Jose Luis Medina [redacted]
You selected this USER-ID:
    "Jose Luis Medina Balseca (Clave de Jose Luis Medina B) <[redacted]@gmail.com>"

Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? O
You need a Passphrase to protect your secret key.

passphrase not correctly repeated; try again.
We need to generate a lot of random bytes. It is a good idea to perform
some other action (type on the keyboard, move the mouse, utilize the
disks) during the prime generation; this gives the random number
generator a better chance to gain enough entropy.

Not enough random bytes available.  Please do some other work to give
the OS a chance to collect more entropy! (Need 200 more bytes)
```

- GnuPG:

```
root@tyrio:~# gpg -k
/root/.gnupg/pubring.gpg
------------------------
pub   4096R/D39DC0E3 2014-10-28
uid                  Michal Papis (RVM signing) <mpapis@gmail.com>
sub   2048R/C71866D7 2015-11-02
sub   4096R/BF04FF17 2014-10-28 [expires: 2017-03-08]

pub   2048R/2F503F51 2016-11-10 [expires: 2018-11-10]
uid                  Jose Luis Medina [redacted] (Clave de Jose Luis Medina [redacted]) <[redacted]@gmail.com>
sub   2048R/FDFFB03C 2016-11-10 [expires: 2018-11-10]

root@tyrio:~# gpg --output CLAVEPUB.gpg --export 2F503F51
```

```
root@tyrio:~# gpg --encrypt --recipient 2F503F51 documentos.txt
```

```
root@tyrio:~# gpg -d documentos.txt.gpg

You need a passphrase to unlock the secret key for
user: "Jose Luis Medina [redacted] (Clave de Jose Luis Medina[redacted] <[redacted]@gmail.com>"
2048-bit RSA key, ID FDFFB03C, created 2016-11-10 (main key ID 2F503F51)

Enter passphrase:
```

Bibliografía

- **STALLINGS Williams,** Cryptography and Network Security, Principles and Practice, Fifht Edition

- Video Explicación **DES,** https://www.youtube.com/watch?v=De4ikZa0G4c
- Video Explicación **AES,** https://www.youtube.com/watch?v=BctIBJ2NdHk