Universidad de Oviedo

Departamento de Informática

Campus de Gijón

Cifrado de datos con algoritmos Simétricos

Presentación

Daniel F. García

Definiciones iniciales

Criptografía

Ciencia que estudia las técnicas empleadas para encriptar (cifrar) la información

Término criptografía (proviene del griego) → { "Kryptos" (oculto); "Grafos" (escritura) }

Criptoanálisis

Ciencia que estudia herramientas y técnicas para romper los sistemas de cifrado definidos por la criptografía

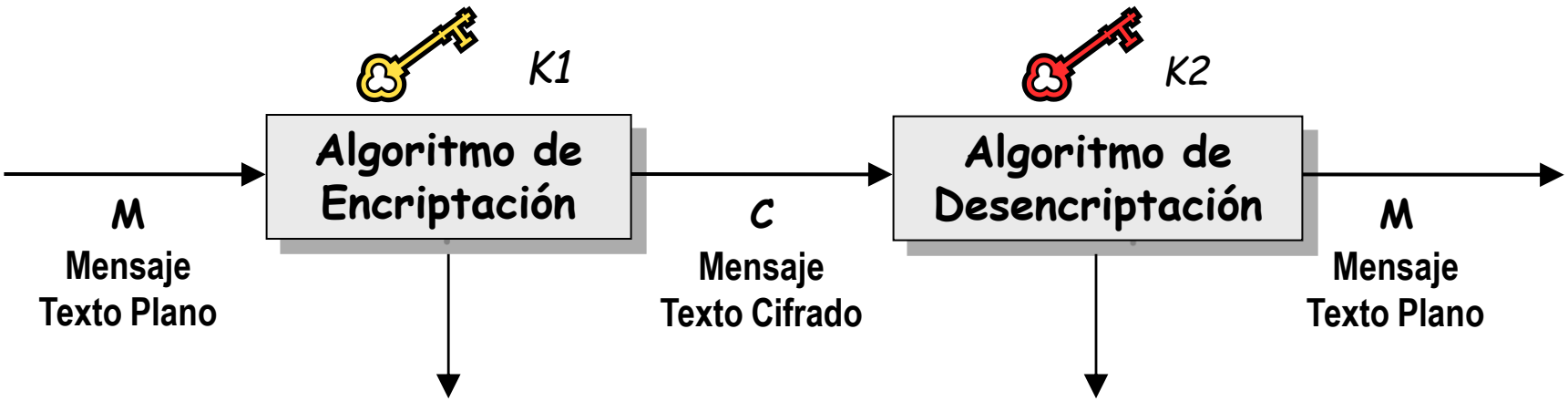
Criptología

Ciencia que desarrolla sistemas para cifrar la información (criptografía) y para desbaratarlos (criptoanálisis)

Funcionamiento de un sistema criptográfico 1

Base: Algoritmos de Encriptación (cifrado) y Desencriptación (descifrado)

Son los elementos básicos para implementar una infraestructura de seguridad



Tipos de Algoritmos → { Simétricos: $K1 = K2$; Asimétricos: $K1 \neq K2$

Funcionamiento de un sistema criptográfico 2

Funcionamiento de los algoritmos → Depende de las claves

(La clave determina el resultado que produce el algoritmo)

Recomendación actual

El algoritmo de cifrado / descifrado debe ser público y estar bien documentado

¿Por qué? **Para que expertos en criptografía determinen su robustez y sus debilidades**

Aunque el algoritmo sea público sin la clave no se puede descifrar la información

(Siempre que el algoritmo sea lo suficientemente robusto)

Tipos de sistemas criptográficos según las claves usadas

SIMÉTRICOS ó de clave privada o secreta

Se emplea la misma clave para el cifrado y el descifrado

ASIMÉTRICOS ó de clave pública

Se emplea una clave para el cifrado y otra para el descifrado

Una de las claves es pública y la otra es privada

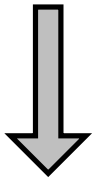
Funcionamiento de un sistema criptográfico 3

Los algoritmos criptográficos ⋯

Realizan una secuencia de operaciones sobre los símbolos del texto original

- Transposiciones (cambiar el orden de los símbolos)
- Sustituciones (reemplazar unos símbolos por otros)

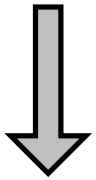
Texto original: **criptografia**



Sustitución de símbolos

(reemplazar cada letra por la segunda siguiente en el alfabeto)

Resultado: **etkrvqitchkc**



Transposición de símbolos en bloques de tres

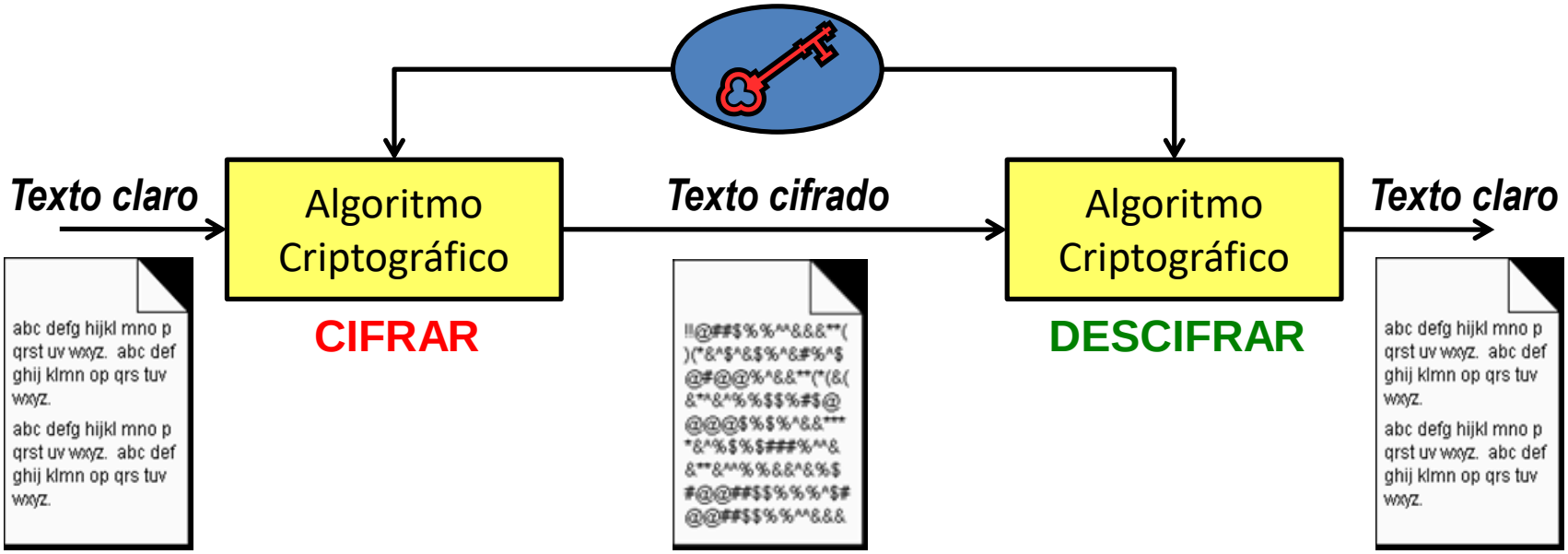
Símbolo 3ª Posición → 1ª
Símbolo 1ª Posición → 2ª
Símbolo 2ª Posición → 3ª

Resultado: **ketqrvcitchk**

Criptografía simétrica 1

Los algoritmos criptográficos simétricos:

UTILIZAN LA **MISMA** CLAVE PARA CIFRAR Y DESCIFRAR



La seguridad se basa en mantener secreta la clave, no el algoritmo

Ventaja: Elevada velocidad de cifrado

Problema: Gestión de claves
- Aumento del número de claves: n (n-1) / 2 para n usuarios
- Distribución de cada clave al transmisor y al receptor

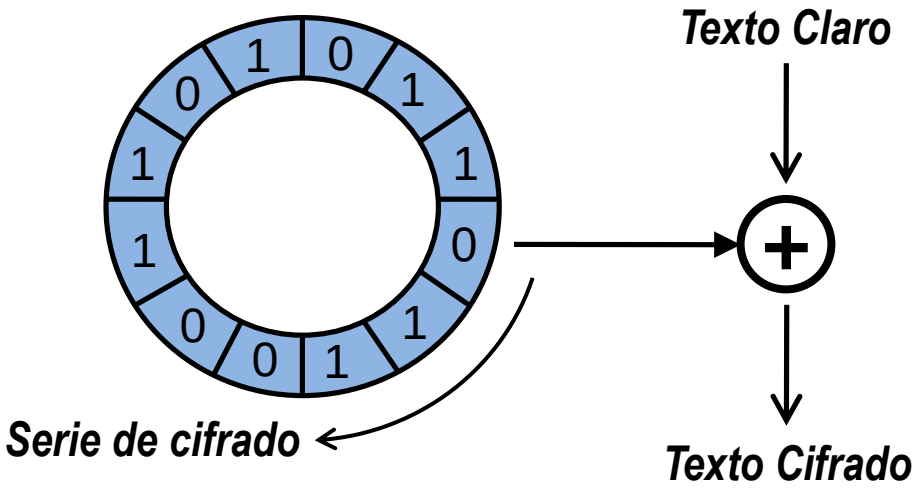
Tipos de algoritmos simétricos →
- De flujo (ó serie)
- De bloque

Algoritmos criptográficos simétricos de flujo 1

Un cifrador de flujo (*stream cipher*) combina los dígitos de texto plano con una serie pseudoaleatoria de dígitos de cifrado (*keystream*), típicamente mediante una operación o-exclusiva (xor)

Los dígitos de texto plano son cifrados uno de cada vez

En la práctica los dígitos son bits o bytes



Tipos de cifradores de flujo { Síncronos
Auto-sincronizables

Algoritmos criptográficos simétricos de flujo 2

Cifradores de Flujo Síncronos

La secuencia de dígitos de cifrado se genera independientemente del texto plano y del texto cifrado, y se combina con:

- El texto plano (para cifrar)
- El texto cifrado (para descifrar)

El transmisor y el receptor deben estar perfectamente sincronizados para que el proceso de descifrado sea correcto

Si se añaden o eliminan dígitos del texto durante la transmisión, se pierde la sincronización

Si se corrompe un dígito durante la transmisión, solo afecta a un digito del texto descifrado
Y el error no se propaga a otros dígitos del texto

Cifradores de Flujo Auto-sincronizables

Usan varios de los N dígitos del texto cifrado previamente para generar la serie de dígitos de cifrado (*keystream*)

El receptor se sincroniza automáticamente con el transmisor después de recibir N dígitos del texto cifrado

Algoritmos criptográficos simétricos de bloque 1

Un cifrador de bloques (*block cipher*) trabaja con grupos de bits de longitud fija denominados bloques

Transforma N bits de texto plano en N bits de texto cifrado

PERO ⋯

Tras dividir un texto plano en bloques,
el cifrado de cada bloque se puede realizar así:

- De modo independiente para cada bloque
- De modo que el cifrado de unos bloques afecte al cifrado de otros bloques

Modos de funcionamiento básicos de los cifradores de bloque:

ECB: *Electronic CodeBook* (Libro electrónico de códigos)

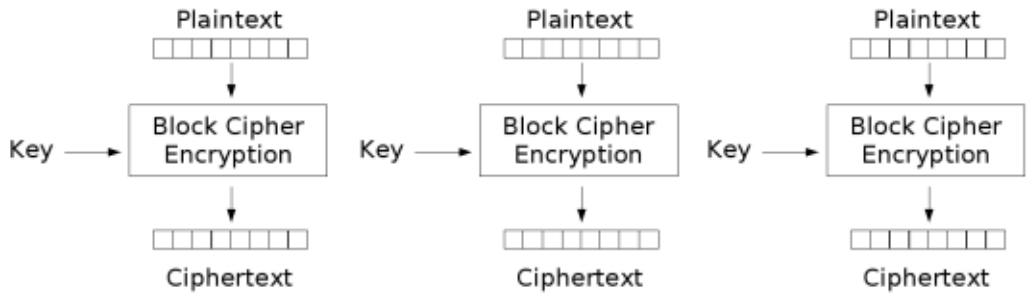
CBC: *Cipher Block Chaining* (Encadenamiento de bloques)

CFB: *Cipher FeedBack* (Realimentación de bloques)

OFB: *Output FeedBack* (Realimentación del bloque de salida)

Operación en modo ECB: Electronic CodeBook

Cada bloque se cifra con la clave (key) de forma independiente de otros bloques

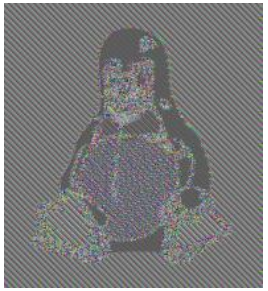


Electronic Codebook (ECB) mode encryption

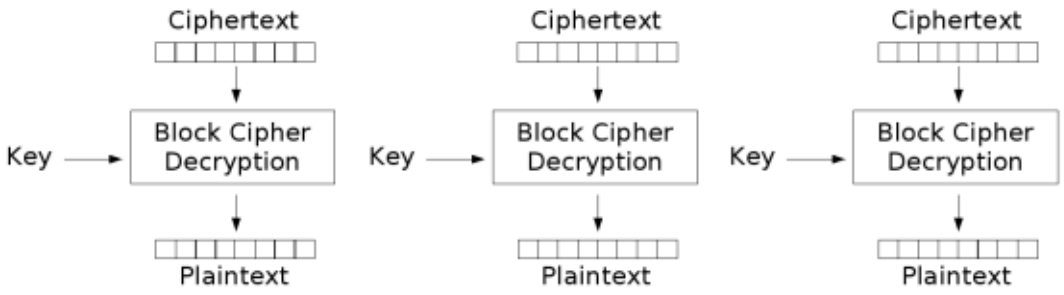
Original

Resultado ←→ Codificar con un gran libro electrónico de códigos

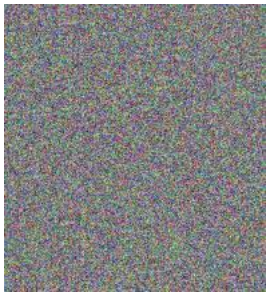
Bloques de texto plano iguales, SIEMPRE generan el mismo bloque cifrado



Modo ECB



Electronic Codebook (ECB) mode decryption

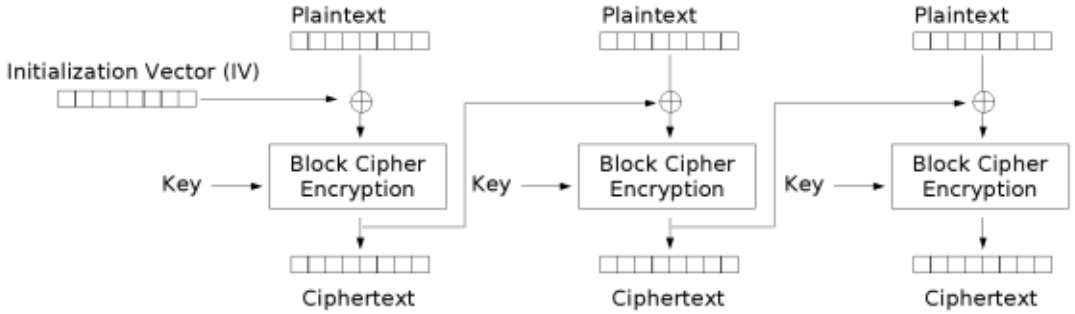


Otros Modos

Operación en modo CBC: Cipher Block Chaining

Antes de encriptar cada bloque de texto plano se le hace una operación XOR con el bloque de texto cifrado previo

Al primer bloque de texto plano se le hace una operación XOR con un **vector de inicialización**



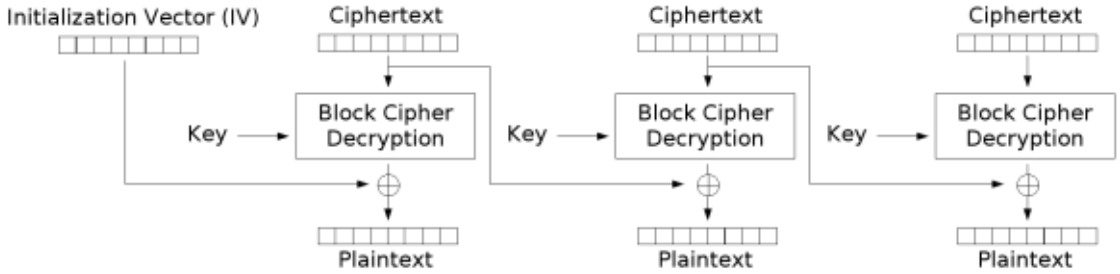
Cipher Block Chaining (CBC) mode encryption

← Se aleatoriza el texto de entrada eliminando sus patrones

La encriptación NO es paralelizable

La desencriptación SI es paralelizable

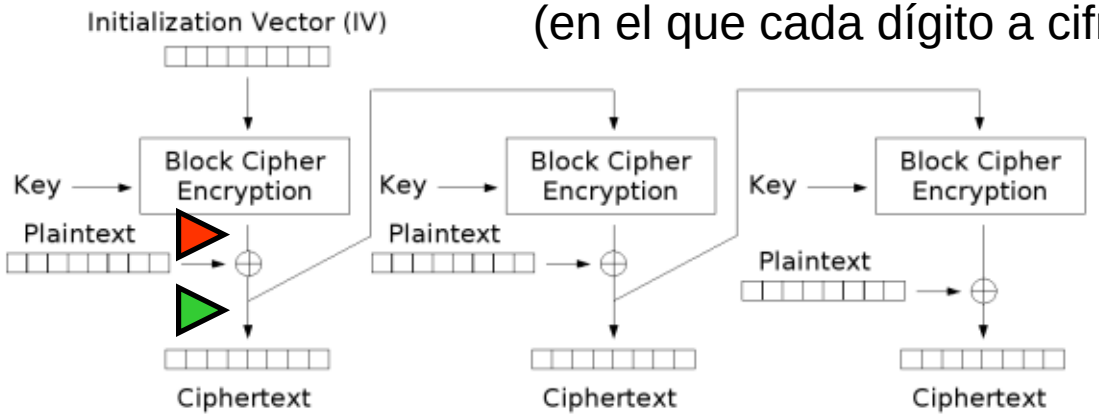
Un cambio de UN bit en el texto plano
Afecta a todos los bloques cifrados siguientes



Cipher Block Chaining (CBC) mode decryption

Operación en modo CFB: Cipher FeedBack

Su funcionamiento es similar al de un cifrador de flujo auto-sincronizable (en el que cada dígito a cifrar es un bloque)



Cipher Feedback (CFB) mode encryption

Trozo de

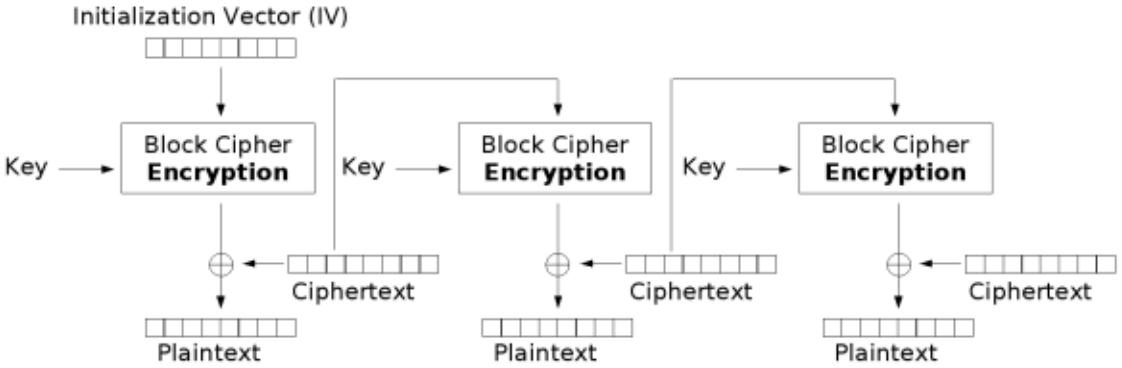
Salida Cifrador ←→ Serie de cifrado

Ptext XOR Serie → Texto Cifrado

Texto Cifrado → Entrada al Cifrador

Genera nuevo trozo de serie de cifrado

La desencriptación es idéntica a la encriptación cambiando simplemente el rol que juegan el PlainText y el CiperText en la operación XOR de salida



Cipher Feedback (CFB) mode decryption

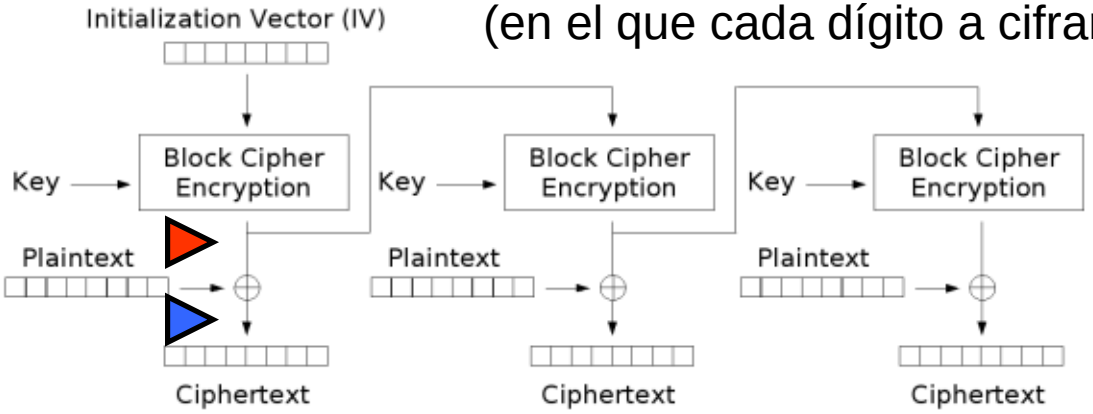
La encriptación NO es paralelizable

La desencriptación SI es paralelizable

¿Bloque descifrador?

Operación en modo OFB: Output FeedBack

Su funcionamiento es similar al de un cifrador de flujo síncrono (en el que cada dígito a cifrar es un bloque)

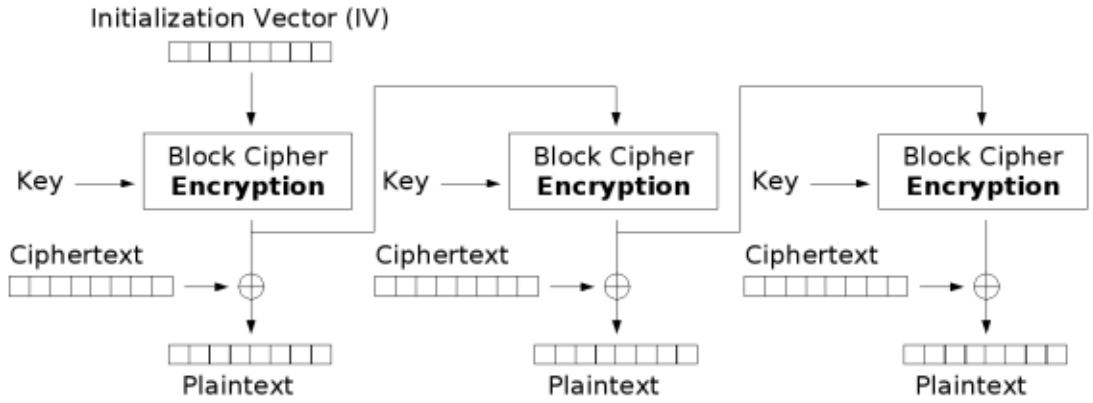


Output Feedback (OFB) mode encryption

Trozo de

- Salida Cifrador ←→ Serie de cifrado
  Trozo de serie → Entrada al Cifrador
  Genera nuevo trozo de serie de cifrado
- Ptext XOR Serie → Texto Cifrado

La desencriptación es idéntica a la encriptación cambiando simplemente el rol que juegan el PlainText y el CipherText en la operación XOR de salida



Output Feedback (OFB) mode decryption

La encriptación y desencriptación NO son paralelizables (La entrada de cada encriptador es la salida del anterior)

Pero … Se pueden hacer todas las operaciones con el cifrador a priori

El paso final de cifrado o descifrado (xor) Se puede hacer en paralelo una vez que El texto plano o cifrado estén disponibles

Estandarización de los modos de operación

- Inicialmente (1980) los modos de operación se definieron en el documento FIPS 81 para el algoritmo simétrico DES (Data Encryption Standard)

  https://csrc.nist.gov/pubs/fips/81/final
  DES Modes of Operation

  FIPS = Federal Information Processing Standards

- En 2001 el NIST (National Institute of Standards and Technology, US Dep. Commerce)
  - Revisó los modos de operación aprobados
  - Incluyó el algoritmo simétrico AES (Advanced Encryption Standard)
  - Añadió el modo CTR (Counter Mode)

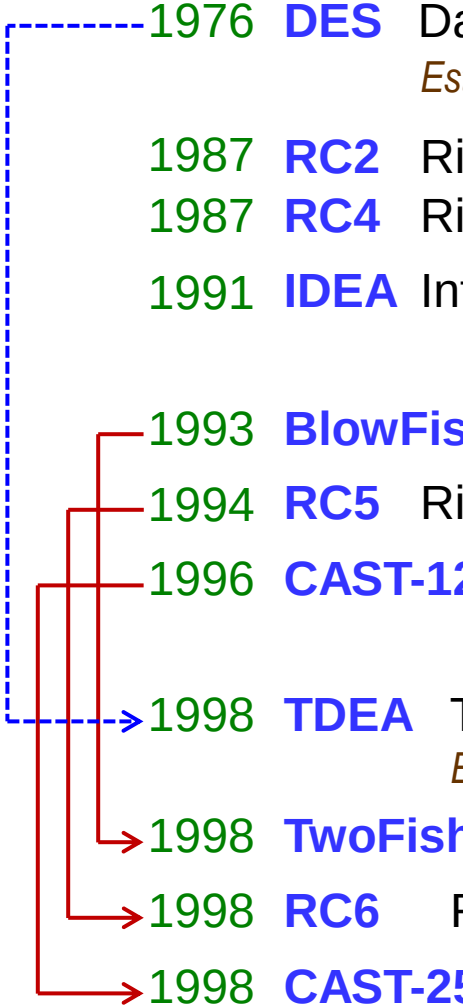
  https://csrc.nist.gov/pubs/sp/800/38/a/sup/final
  Recommendation for Block Cipher Modes of Operation

- En 2010 el NIST añadió el modo XTS para el algoritmo AES

  XTS = **X**EX **T**weakable Block Cipher with Ciphertext **S**tealing (XEX = XOR Encrypt XOR)

  https://csrc.nist.gov/pubs/sp/800/38/e/final
  Recommendation for Block Cipher Modes of Operation:
  The XTS-AES Mode for Confidentiality on Storage Devices

Visión de algoritmos criptográficos simétricos



1976 **DES** Data Encryption Standard Estándar FIPS46 → Revisiones → FIPS46-3
Estándar ANSI (ANS X3.92) https://csrc.nist.gov/pubs/fips/46-3/final

1987 **RC2** Rivest Cipher 2 (Block) https://www.rfc-editor.org/rfc/rfc2268

1987 **RC4** Rivest Cipher 4 (Stream) https://datatracker.ietf.org/doc/id/draft-kaukonen-cipher-arcfour-03.txt

1991 **IDEA** International Data Encryption Algorithm (Block)

1993 **BlowFish**

1994 **RC5** Rivest Cipher 5 (Block) https://people.csail.mit.edu/rivest/pubs/Riv94.pdf

1996 **CAST-128** https://www.rfc-editor.org/rfc/rfc2144

1998 **TDEA** Triple Data Encryption Algorithm (También denominado Triple DES o 3DES)
Estándar ANSI (ANS X9.52) https://csrc.nist.gov/pubs/sp/800/67/r2/final

1998 **TwoFish**

1998 **RC6** Rivest Cipher 6 (Block)

1998 **CAST-256** https://www.rfc-editor.org/rfc/rfc2612

2000 **AES** Advanced Encryption Standard (Block)

AES (Advanced Encryption Standard) Introducción

- Adoptado como un estándar por el gobierno federal USA el 26-Mayo-2002

  https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf

- Seleccionado entre 15 competidores en un proceso que duró 5 años
  - En enero de 1997 el NIST solicitó nuevos algoritmos de encriptación
  - En agosto de 1998 se seleccionaron 15 algoritmos
  - En abril de 1999 se seleccionaron los 5 finalistas → MARS, RC6, Rijndael, Serpent y Twofish
  - En octubre de 2000 se seleccionó Rindjael como AES – FIPS-197

    Rindjael fue diseñado por Joan Daemen y Vincent Rijmen (Univ Leuven, Bélgica)

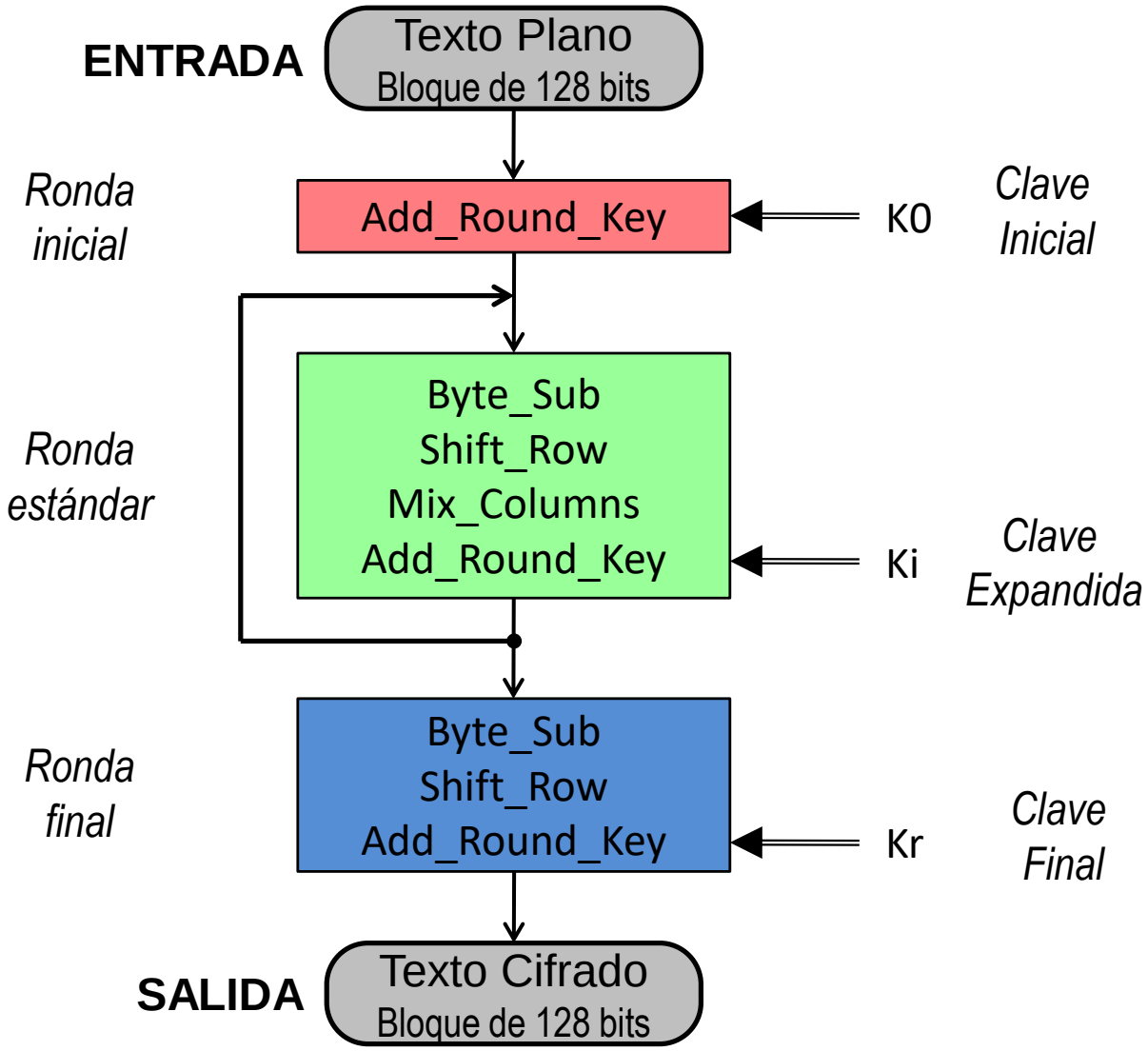
- Cifrador / Descifrador simétrico que trabaja con bloques de 128 bits

  El bloque se organiza en forma de matriz de 4x4 bytes denominada el *estado*

- Tamaños de clave posibles: 128, 192, 256 bits

- Principio de diseño: red de sustitución - permutación

AES: Vista general del proceso de cifrado



AES: Ronda inicial

La entrada y la clave tienen 16 bytes (128 bits)

Relleno

ENTRADA:	**A**	**T**	**T**	**A**	**C**	**K**		**A**	**T**		**D**	**O**	**W**	**N**	**!**	01
CLAVE:	**S**	**O**	**M**	**E**		**1**	**2**	**8**		**B**	**I**	**T**		**K**	**E**	**Y**

El texto de entrada y la clave se organizan en matrices por columnas sucesivas

Entrada

A	C	T	W
T	K		N
T		D	!
A	A	O	01

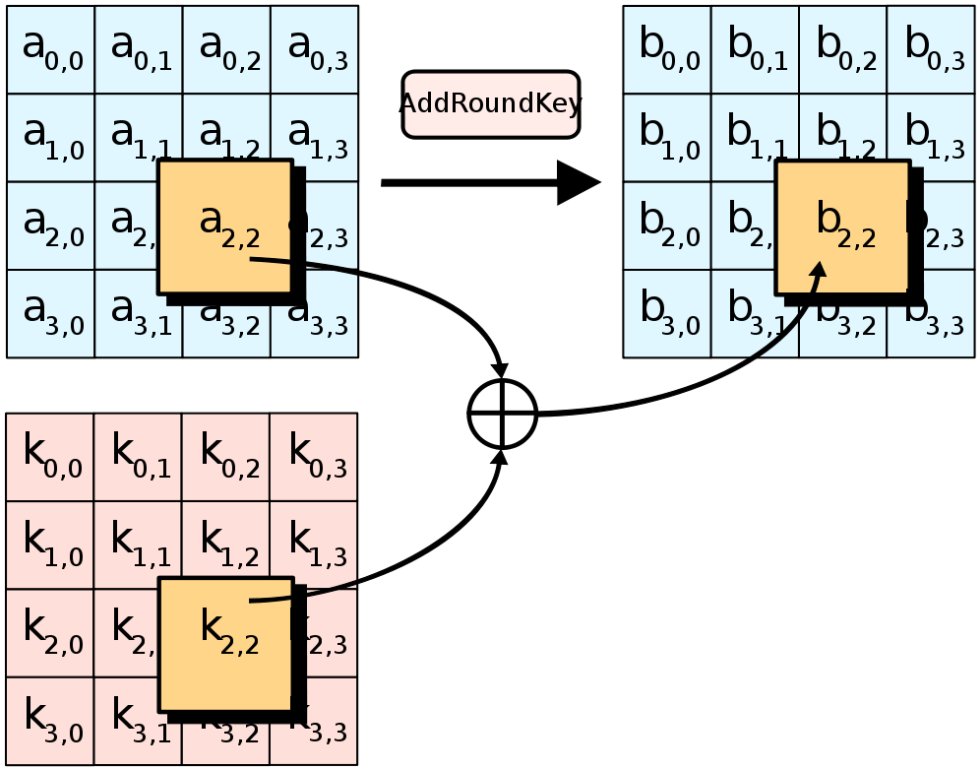
Clave

S			
O	1	B	K
M	2	I	E
E	8	T	Y

AES: Ronda inicial - Concepto

Operación *Add_Round_Key*

Cada byte del estado es combinado con un byte de la clave de la ronda correspondiente usando la operación XOR



AES: Ronda inicial - Ejemplo

Ejemplo de Operación *Add_Round_Key*

Entrada			
A	C	T	W
T	K		N
T		D	!
A	A	O	01

$\oplus$

K0 = Clave Inicial			
S			
O	1	B	K
M	2	I	E
E	8	T	Y

=

12	63	74	77
1B	7A	62	05
19	12	0D	64
04	79	15	58

A = 65d = 41h = $0100\ 0001_b$
S = 83d = 53h = $0101\ 0011_b$

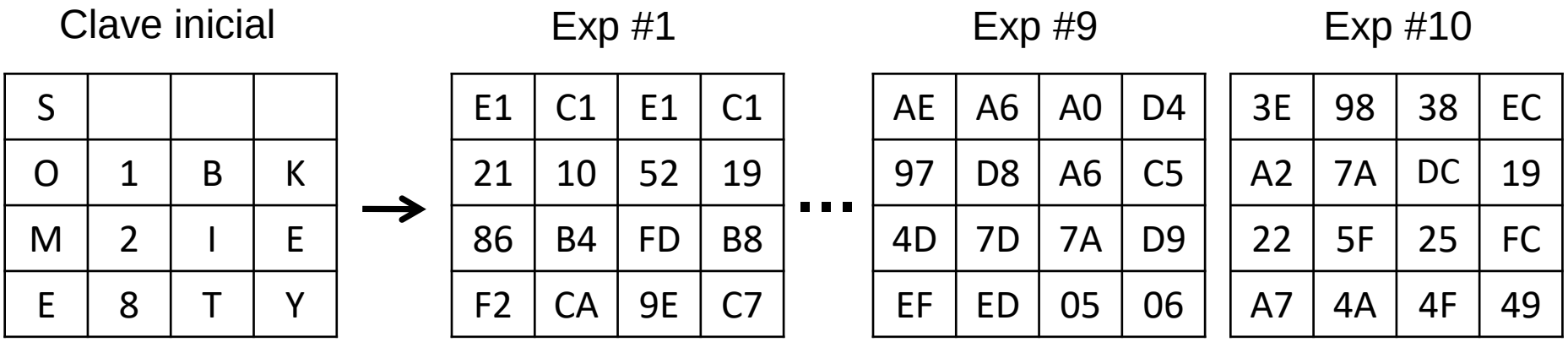
XOR = $0001\ 0010_b$

A y S interpretados como código ASCII

AES: Expansión de la clave - concepto

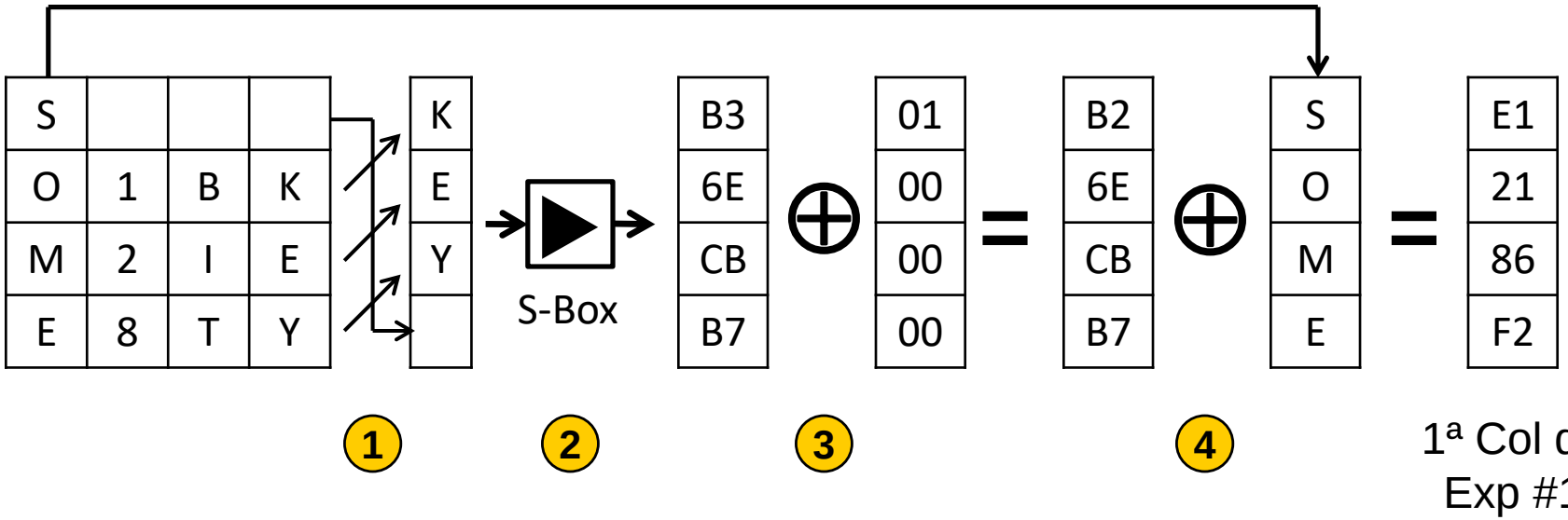
La operación *Add_Round_Key* realizada en cada ronda estándar necesita una **clave especifica para cada ronda estándar**

La clave de cada ronda estándar se deriva de la clave inicial mediante un proceso denominado **Expansión de la Clave**



AES: Expansión de la clave – pasos (1)

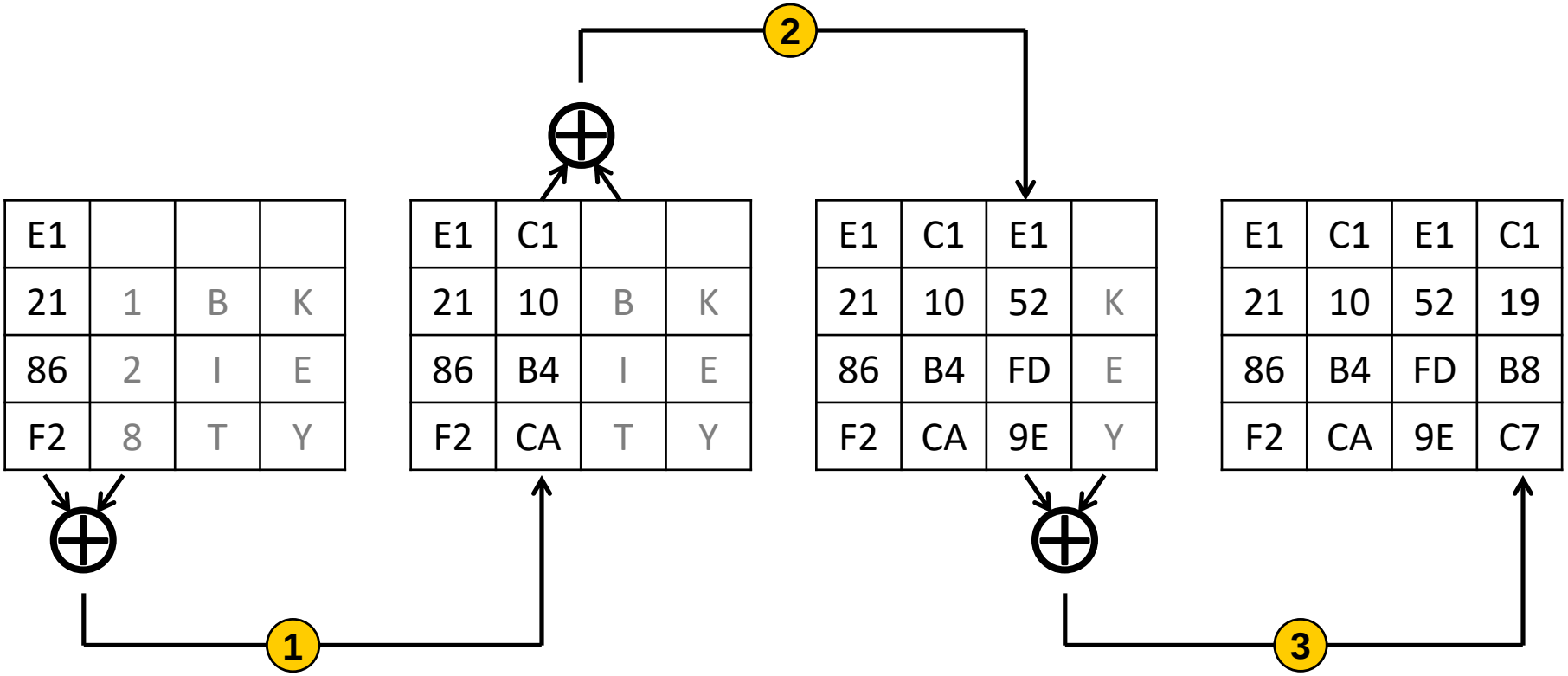
Obtención de la primera columna de la nueva clave



S-Box = *Substitution-Box*

AES: Expansión de la clave – pasos (2)

Obtención de las siguientes columnas de la nueva clave:



AES: Ronda intermedia - Concepto

En todas las **rondas intermedias** se realizan 4 operaciones sucesivas:

- 1. Byte_Sub
  Se substituye cada byte del estado por otro extraído de una tabla
- 2. Shift_Row
  Se hacen desplazamientos con las filas del estado
- 3. Mix_Columns
  Se mezclan los bits de los bytes de las columnas del estado
- 4. Add_Round_Key
  Se hace el XOR de cada byte del estado con la clave de la ronda

Tamaño de la clave	Repeticiones de la Ronda intermedia
128	9
192	11
256	13

AES: Ronda intermedia – Byte_Sub (1)

Objetivo → Introducir "**confusión**"

Ocultar la relación entre el mensaje original y el mensaje cifrado

Ejemplo

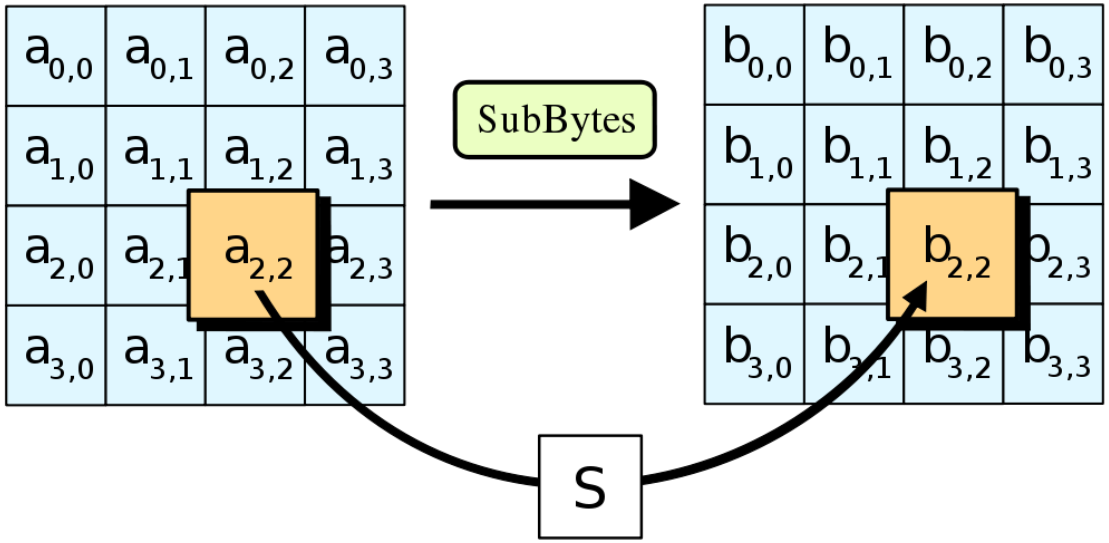
Texto plano: ATTACK AT DAWN

↓↓↓↓↓↓ ↓↓ ↓↓↓↓

Texto cifrado: DWWDFN DW GDZQ

A + 3 letras = D

En AES para transformar un byte se usa una "Substitution Box" (S-Box)



AES: Ronda intermedia – Byte_Sub (2)

Los bytes procesados por una S-Box pueden interpretarse como elementos de un Campo de Gaulois GF(2^8) y admiten una representación polinomial:

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_01 \longleftrightarrow \sum_{i=0}^{7} b_i x^i$$

Los coeficientes b_i de estos polinomios son binarios (0 ó 1)

Ejemplo: byte 0110 0011 → Polinomio representativo: x^6+x^5+x+1

SUMA

Para sumar dos elementos se hace la suma en módulo 2 de los coeficientes de las potencias correspondientes de los dos polinomios

También se puede hacer la operación XOR entre los bits correspondientes de los coeficientes de los polinomios expresados mediante bytes

A = 57h = 0101 0111 = $x^6 + x^4 + x^2 + x + 1$
B = 83h = 1000 0011 = $x^7 + x + 1$

XOR D4h = 1101 0100 = $(x^7 + x^6 + x^4 + x^2 + 2x + 2) \bmod 2$
= $x^7 + x^6 + x^4 + x^2$ = 1101 0100

AES: Ronda intermedia – Byte_Sub (3)

MULTIPLICACIÓN

Para multiplicar dos elementos se multiplican los polinomios MODULO un polinomio irreducible de grado 8

Polinomio irreducible → Sus únicos divisores son 1 y él mismo

Polinomio irreducible usado en AES: $m(x) = x^8 + x^4 + x^3 + x + 1$

A = 57h = 0101 0111 = $x^6 + x^4 + x^2 + x + 1$

B = 83h = 1000 0011 = $x^7 + x + 1$

Multiplicar A·B →

$$x^{13} + x^{11} + x^9 + x^8 + x^7$$
$$x^7 + x^5 + x^3 + x^2 + x$$
$$x^6 + x^4 + x^2 + x + 1$$

$$x^{13}+x^{11}+x^9+x^8+\cancel{2x^7}+x^6+x^5+x^4+x^3+\cancel{2x^2}+\cancel{2x}+1$$

$$(x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1) \text{ MOD } (x^8+x^4+x^3+x+1)$$

NO cabe en un GF(2^8)

AES: Ronda intermedia – Byte_Sub (4)

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

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

$x^5 + x^3$

$x^{13} + x^9 + x^8 + x^6 + x^5$ ← Multiplico m(x) por x^5

$x^{11} + x^4 + x^3 + 1$ ← Resta = Suma MOD 2

$x^{11} + x^7 + x^6 + x^4 + x^3$ ← Multiplico m(x) por x^3

$x^7 + x^6 + 1$ ← Resta = Suma MOD 2

SI cabe en un GF(2^8)

A·B = C1h = 1100 0001 = $x^7 + x^6 + 1$

AES: Ronda intermedia – Byte_Sub (5)

¿Por qué aprender a multiplicar elementos de un GF(2^8)?

¡Porque la 1ª operación que hace una S-Box es cambiar un byte por su inverso!

Para cualquier polinomio binario p(x) no nulo y de grado menor que 8
Se define su inverso $p^{-1}(x)$ como el polinomio que verifica:

$$p(x) \cdot p^{-1}(x) \text{ MOD } m(x) = 1$$

Dados los 255 polinomios posibles de GF(2^8) → Encontrar sus 255 inversos

[Son 255 y no 256 porque el polinomio 00h no tiene inverso]

La ecuación: $p(x) \cdot ¿? \text{ MOD } m(x) = 1$

Se puede resolver “por fuerza bruta” probando hasta encontrar la solución

Con todas las soluciones se construye una tabla

AES: Ronda intermedia – Byte_Sub (6)

Tabla de inversos en un campo GF(2^8)

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

Ejemplos:

INV(C4) → DA

INV(DA) → C4

Se verifica que:

SI

INV(X) → Y

ENTONCES

INV(Y) → X

AES: Ronda intermedia – Byte_Sub (7)

La 2ª operación que hace una S-Box es esta transformación afín:

$$a_i = b_i \oplus b_{(i+4)mod8} \oplus b_{(i+5)mod8} \oplus b_{(i+6)mod8} \oplus b_{(i+7)mod8} \oplus c_i$$

b_i es el i-esimo bit del byte a transformar ($0 \le i \le 7$)

c_i es el i-esimo bit del byte 63h = 0110 0011

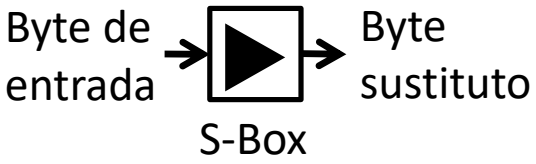
$$a_0 = b_0 \oplus b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus c_0$$
$$a_1 = b_1 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_0 \oplus c_1$$
$$a_2 = b_2 \oplus b_6 \oplus b_7 \oplus b_0 \oplus b_1 \oplus c_2$$
$$a_3 = b_3 \oplus b_7 \oplus b_0 \oplus b_1 \oplus b_2 \oplus c_3$$
$$a_4 = b_4 \oplus b_0 \oplus b_1 \oplus b_2 \oplus b_3 \oplus c_4$$
$$a_5 = b_5 \oplus b_1 \oplus b_2 \oplus b_3 \oplus b_4 \oplus c_5$$
$$a_6 = b_6 \oplus b_2 \oplus b_3 \oplus b_4 \oplus b_5 \oplus c_6$$
$$a_7 = b_7 \oplus b_3 \oplus b_4 \oplus b_5 \oplus b_6 \oplus c_7$$

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} = \begin{pmatrix} 1&0&0&0&1&1&1&1 \\ 1&1&0&0&0&1&1&1 \\ 1&1&1&0&0&0&1&1 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&1&0&0&0 \\ 0&1&1&1&1&1&0&0 \\ 0&0&1&1&1&1&1&0 \\ 0&0&0&1&1&1&1&1 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

AES: Ronda intermedia – Byte_Sub (8)

Combinando las dos operaciones → TABLA de sustitución de bytes

	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



5A **BE**

AES: Ronda intermedia – Byte_Sub - Ejemplo

Resultado de Add_Round_Key
(En la Ronda Inicial)

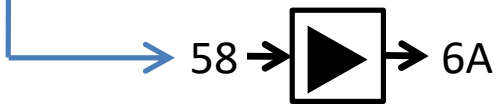
||

Entrada

12	63	74	77
1B	7A	62	05
19	12	0D	64
04	79	15	58

Salida

C9	FB	92	F5
AF	DA	AA	6B
D4	C9	D7	43
F2	B6	59	6A



S-Box

AES: Ronda intermedia – Shift_Row (1)

Objetivo → Introducir "**difusión**"

Es conveniente dispersar el mensaje
Por ejemplo, realizando una transposición de columnas

Ejemplo

Texto inicial:

A	T	T	A
C	K	A	T
D	A	W	N

Texto transpuesto: ACD TKA TAW ATN

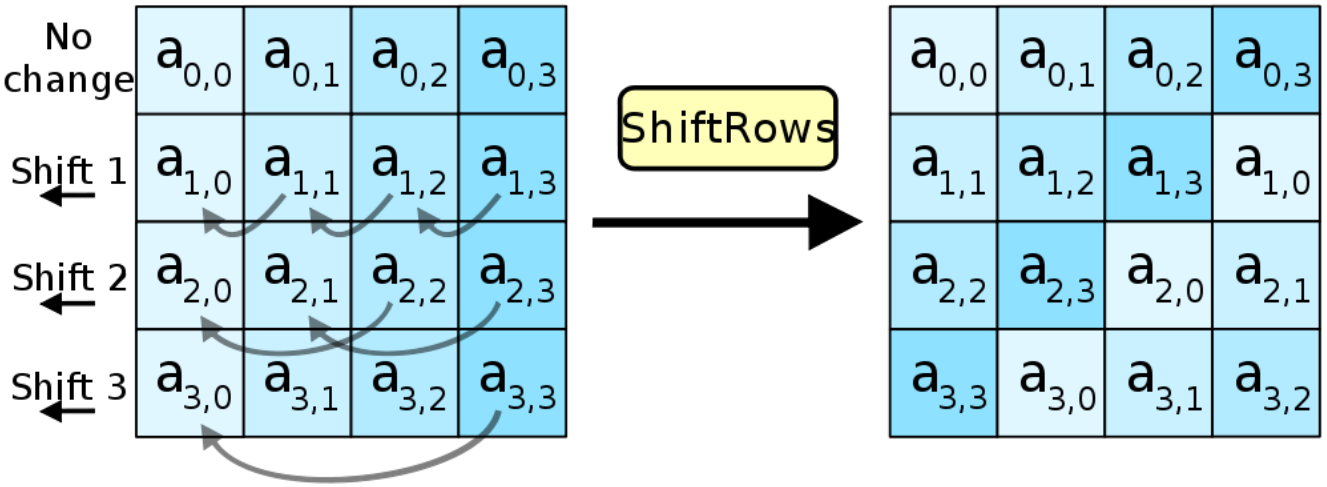
En AES para introducir la difusión se usa una técnica de desplazamiento de filas

AES: Ronda intermedia – Shift_Row (2)

Operación *Shift_Row*

Cada fila F del estado es desplazada cíclicamente (circularmente) B bytes a la izquierda

Fila	Bytes
0	0
1	1
2	2
3	3

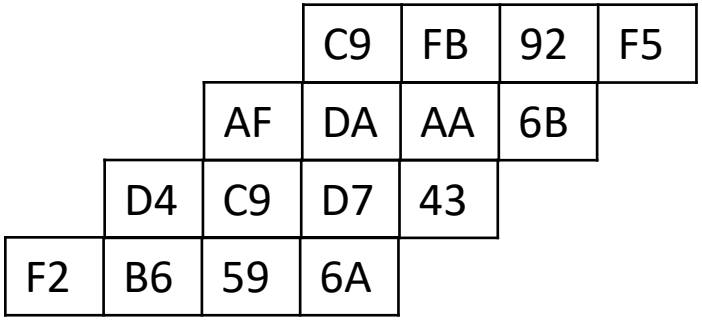


AES: Ronda intermedia – Shift_Row - Ejemplo

Ejemplo de Operación *Shift_Row*

Cada fila F del estado es desplazada cíclicamente (circularmente) B bytes a la izquierda

C9	FB	92	F5
AF	DA	AA	6B
D4	C9	D7	43
F2	B6	59	6A



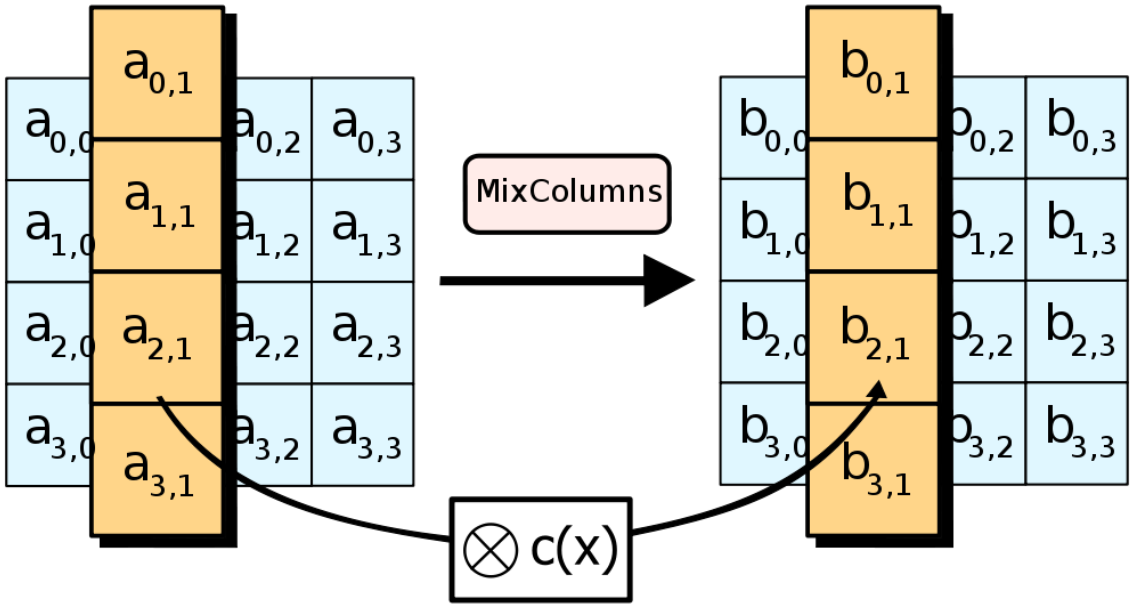
C9	FB	92	F5
DA	AA	6B	AF
D7	43	D4	C9
6A	F2	B6	59

AES: Ronda intermedia – Mix_Columns (1)

Objetivo → Introducir **MAS** "**difusión**"

Los cuatro bytes de cada columna del estado se combinan usando una función

La función MixColumns transforma 4 bytes (entrada) en otros 4 bytes (salida) (cada byte de entrada afecta a los cuatro bytes de salida)



Durante la transformación **NO** hay interacción entre las columnas del estado

AES: Ronda intermedia – Mix_Columns (2)

La transformación de las columnas de puede representar matricialmente:

$$\begin{pmatrix} b_{0,C} \\ b_{1,C} \\ b_{2,C} \\ b_{3,C} \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{pmatrix} \begin{pmatrix} a_{0,C} \\ a_{1,C} \\ a_{2,C} \\ a_{3,C} \end{pmatrix}$$

$$b_{0,C} = 2\,a_{0,C} \oplus 3\,a_{1,C} \oplus a_{2,C} \oplus a_{3,C}$$
$$b_{1,C} = a_{0,C} \oplus 2\,a_{1,C} \oplus 3\,a_{2,C} \oplus a_{3,C}$$
$$b_{2,C} = a_{0,C} \oplus a_{1,C} \oplus 2\,a_{2,C} \oplus 3\,a_{3,C}$$
$$b_{3,C} = 3\,a_{0,C} \oplus a_{1,C} \oplus a_{2,C} \oplus 2\,a_{3,C}$$

La operación de multiplicación de los dígitos de la matriz por los elementos $a_{F,C}$ se interpreta así:

- x1 → NO modificar al elemento $a_{F,C}$
- x2 → Desplazar 1 bit a la Izda los bits del elemento $a_{F,C}$
- x3 → Desplazar 1 bit a la Izda los bits del elemento $a_{F,C}$ y después hacer la operación XOR con el valor inicial de $a_{F,C}$ (sin desplazarlo)

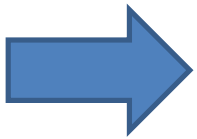
Casos x2 y x3 → Después del desplazamiento de $a_{F,C}$

SI $a_{F,C}$_Desplazado > 0xFF

ENTONCES hacer $a_{F,C}$_Desplazado XOR 0x1B

AES: Ronda intermedia – Mix_Columns - Ejemplo

C9	FB	92	F5
DA	AA	6B	AF
D7	43	D4	C9
6A	F2	B6	59



41	B9	E0	8B
6E	83	95	A9
18	DA	8B	38
99	00	65	D0

$a_{0,0}$ C9 1100 1001
$a_{0,0}$<<1 1 1001 0010
1B 0001 1011
XOR
$2a_{0,0}$ = XOR 1000 1001

$a_{1,0}$ DA 1101 1010
$a_{1,0}$<<1 1 1011 0100
1B 0001 1011
XOR
XOR 1010 1111
DA 1101 1010
XOR
$3a_{1,0}$ = XOR 0111 0101

$b_{0,0} = 2\,a_{0,0} \oplus 3\,a_{1,0} \oplus a_{2,0} \oplus a_{3,0}$

$2a_{0,0}$ = 1000 1001
$3a_{1,0}$ = 0111 0101
$a_{2,0}$ = 1101 0111 = D7
$a_{3,0}$ = 0110 1010 = 6A

$b_{0,0}$ = XOR = 0100 0001 = 41

AES: Ronda intermedia – Add_Round_Key

Se hace la misma operación que en la ronda inicial:

Cada byte del estado es combinado con un byte de la clave de la ronda correspondiente usando la operación XOR

Resultado de MixColumns

41	B9	E0	8B
6E	83	95	A9
18	DA	8B	38
99	00	65	D0

$\oplus$

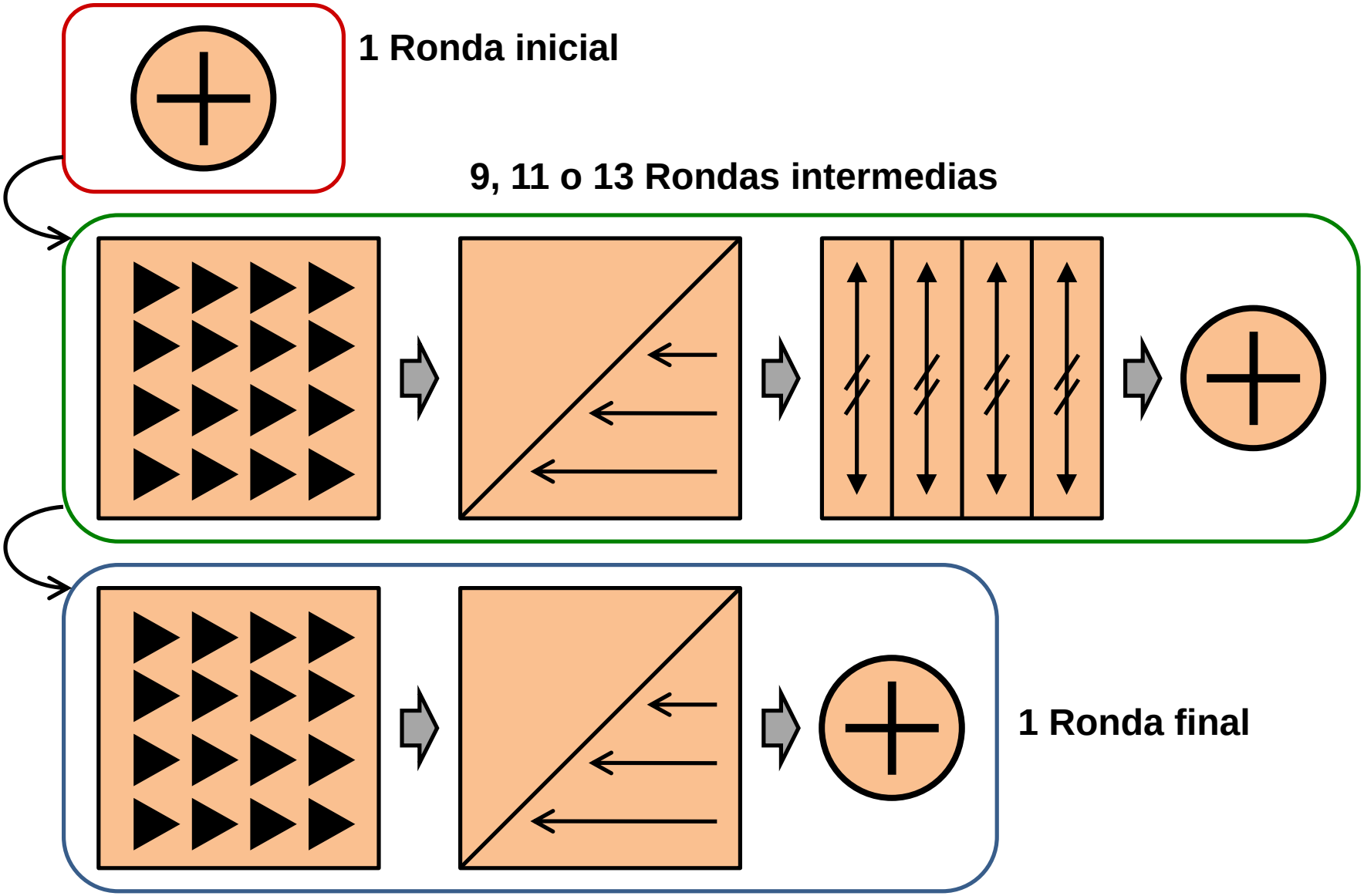
Clave Expandida #1

E1	C1	E1	C1
21	10	52	19
86	B4	FD	B8
F2	CA	9E	C7

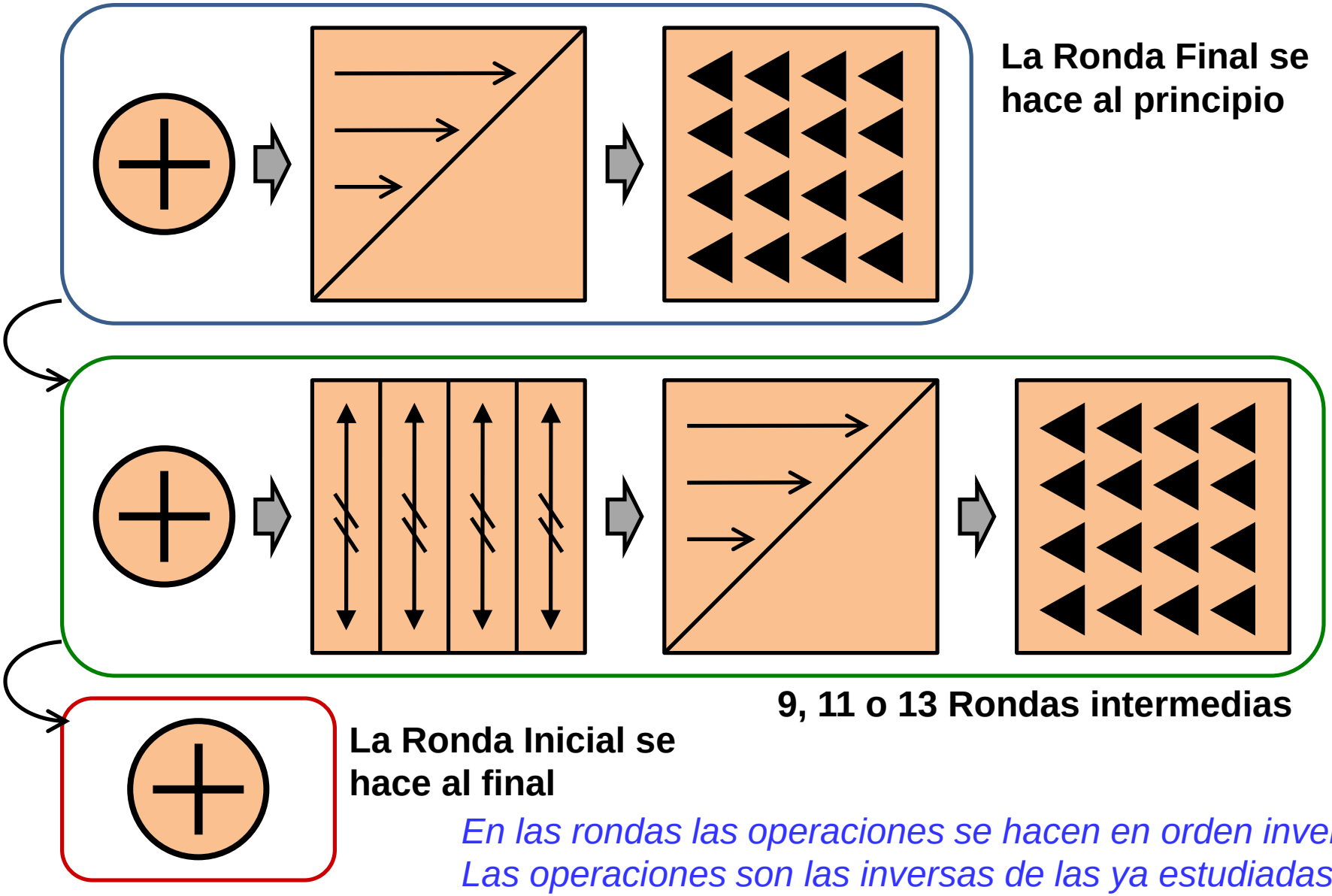
$=$

A0	78	01	4A
4F	93	C7	B0
9E	6E	76	80
6B	CA	FB	17

Resumen gráfico de AES para encriptar



Resumen gráfico de AES para **des**encriptar



La Ronda Final se hace al principio

9, 11 o 13 Rondas intermedias

La Ronda Inicial se hace al final

En las rondas las operaciones se hacen en orden inverso
Las operaciones son las inversas de las ya estudiadas

Soporte Hardware para AES

Intel lanzo en 2010 procesadores con las 6 instrucciones AES-NI:
(Advanced Encryption Standard New Instructions)

AESENC	Ronda de cifrado
AESENCLAST	Última ronda de cifrado
AESDEC	Ronda de descifrado
AESDECLAST	Última ronda de descifrado
AESKEYGENASSIST	Genera claves para las rondas de cifrado
AESIMC	Genera claves de descifrado a partir de las de cifrado

La mayoría de procesadores Intel, AMD y ARM incluyen AES-NI

Es posible que además haya que activar el soporte AES en la BIOS

La mejora en prestaciones al usar AES-NI es de al menos x10

Discos Autocifrados con AES 1

Los discos autocifrados (Self-Encrypting Drives, SEDs) incluyen en su firmware un criptoprocesador AES para almacenar la información cifrada

La mayoría de los SSDs (Solid-State Disks) son SEDs

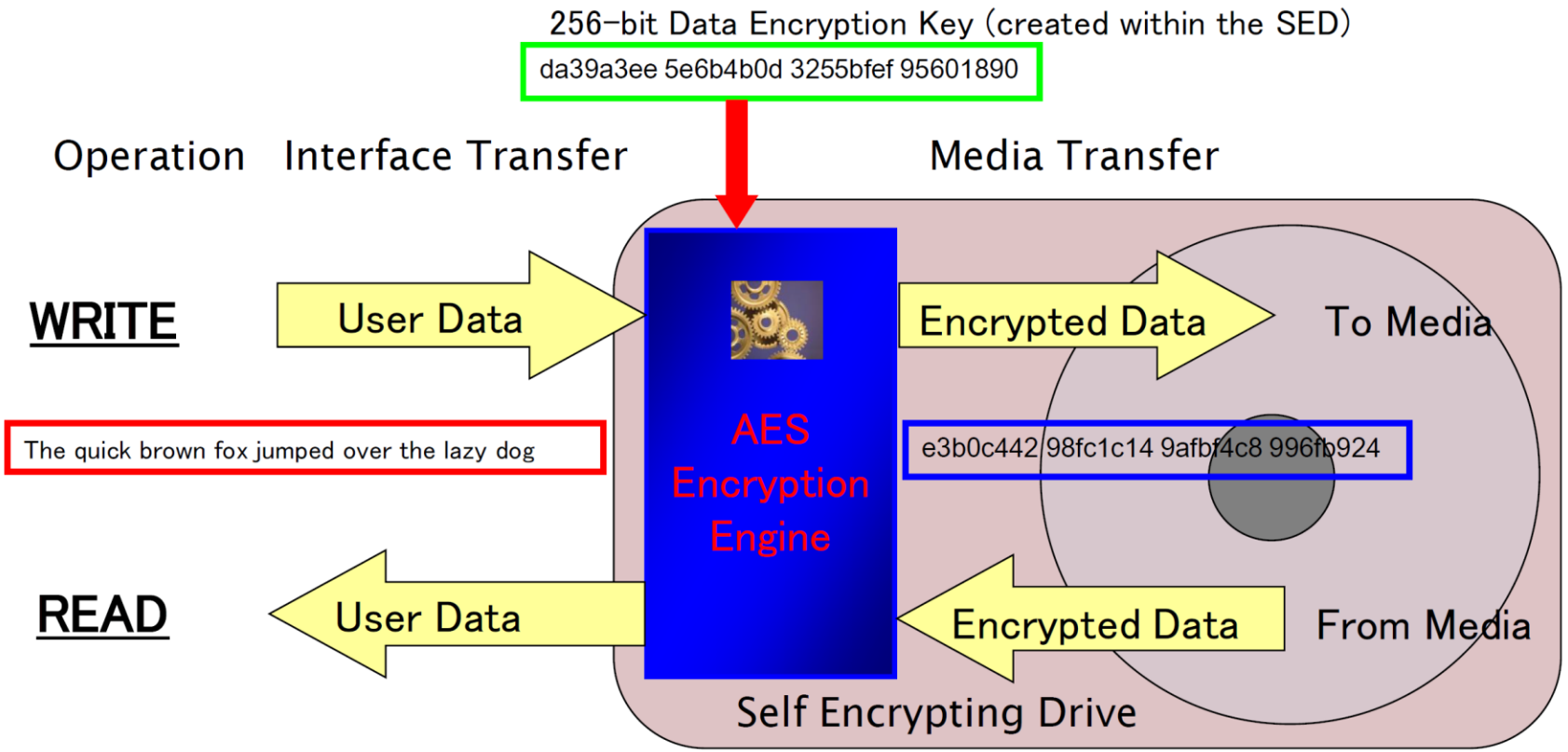


Figura: Trusted Computing Group

Discos Autocifrados con AES 2

La seguridad de un SED se proporciona mediante 2 mecanismos: Boqueo + Cifrado

Se utilizan 2 Claves en un SED:

- **DEK** (Data Encryption Key) Usada para **cifrar**/descifrar datos
  - Generada por el disco y nunca sale del disco
  - Almacenada en el disco en un formato cifrado con la AK
  - Si se cambia o borra, los datos del disco son indescifrables ←→ Borrado criptográfico

- **AK** (Authentication Key) Usada para **desbloquear** el disco
  - Un hash de la clave se almacena en el disco
  - El hash de clave se genera a partir de la clave (contraseña) proporcionada por el usuario
  - Un vez autenticada, esta clave se usa para descifrar la DEK

Al **apagar** el SED → Se bloquea automáticamente

Al **encender** el SED → Permanece bloqueado

Tras proporcionar la contraseña correcta la AK desbloquea el SED
El criptoprocesador permite la escritura y lectura de datos

Discos Autocifrados con AES 3

Proceso de desbloqueo de un SED

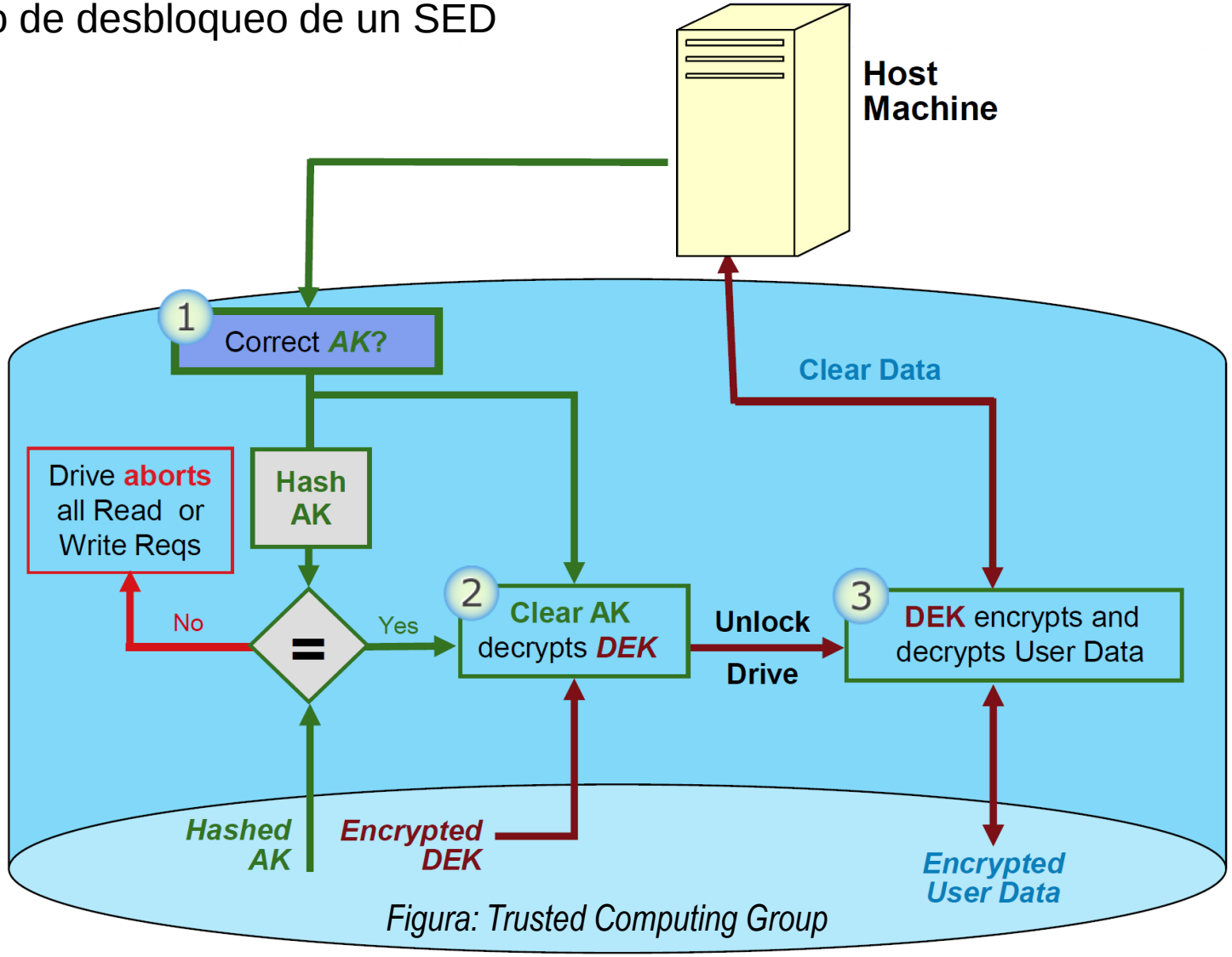


Figura: Trusted Computing Group