IIC3253

Criptografía simétrica y AES

Definiciones formales

- Funciones de hash ☑️
- Message Authentication Codes ☑️
- Esquemas criptográficos ☑️

Definimos una PRP con poder de computación arbitrario pero cantidad fija de pasos.

¿Cómo se vería un "adversario" si lo pensamos como una función en python?

```python
def adv(f: (string) -> string) -> bool:
  """
  parameters:
    f: An arbitrary permutation
  returns:
    b: Guess of f == Enc_k(.)
  """
```

Buscamos generalizar esta noción a una cantidad *polinomial* de pasos

```
def adv(f: (string) -> string) -> bool:
  """
  parameters:
    f: An arbitrary permutation
  returns:
    b: Guess of f == Enc_k(.)
  """
```

¿Podemos pedirle a **`adv`** que sea polinomial?

Si $|k|$ está fijo la definición anterior no funciona, pues podemos probar todas las llaves posibles en $\mathcal{O}(1)$

Nuevamente necesitamos introducir un parámetro de seguridad.

Un esquema es un triple (Gen, Enc, Dec) de algoritmos aleatorizados donde

$Gen(1^n)$ genera una llave k tal que $|k| \geq n$

¿Cómo se ve ahora el adversario?

```
def adv(n: str, f: (string) -> string) -> bool:
  """
  parameters:
    n: Security paramenter (unary)
    f: An arbitrary permutation
  returns:
    b: Guess of f == Enc_k(.) with k = Gen(1^n)
  """
```

Es decir, el adversario recibe también
el parámetro de seguridad

¿Cuándo diremos que un esquema
se ve en general como una PRP?

Intuitivamente, cuando ningún adversario *eficiente* puede distinguir la encriptación de una permutación al azar

¿Cuál sería la formalización?

Para todo adversario $\texttt{Adv}$ de tiempo polinomial:

$$\left| \Pr_{k \sim Gen(1^n)} [\texttt{Adv}(1^n, Enc_k(\cdot))] - \Pr_{\pi \sim \mathbb{U}} [\texttt{Adv}(1^n, \pi(\cdot))] \right|$$

Es una función despreciable

Definiciones formales

- Funciones de hash ☑️
- Message Authentication Codes ☑️
- Esquemas criptográficos ☑️

Vamos a la práctica

$Enc_k : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$ para cada $k \in \{0,1\}^n$

Buscamos una construcción
práctica que se vea como una PRP

Otra forma de verlo...

¿Cuántas permutaciones hay para $\{0,1\}^\ell$? $2^\ell!$

¿Cuántas permutaciones define $Enc_k : \{0,1\}^\ell \rightarrow \{0,1\}^\ell$ con $k \in \{0,1\}^n$? 2^n

Un atacante, en tiempo polinomial, no puede saber si saqué al azar una permutación de las $2^\ell!$ o de las 2^n

Estas 2^n permutaciones deben tener una representación **muy compacta**

Confusión / Difusión

¡No podemos "especificar" las tablas de permutación enteras, es demasiado!

¿Qué tamaño de permutaciones sería razonable especificar en el computador?

Una permutación de 8 bits tiene tamaño $8 \cdot 2^8 = 2048$, o 256 bytes ✓

Supongamos que $f_0, \ldots, f_7$ son 8 de dichas permutaciones y tenemos un mensaje de 64 bits

$$m = B_0 \quad B_1 \cdots B_6 \quad B_7$$

$$m^c = f_0(B_0) \quad f_1(B_1) \cdots f_6(B_6) \quad f_7(B_7)$$

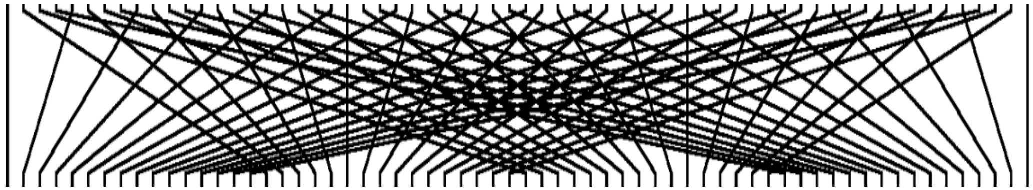
Imaginemos por un momento que la
llave define el orden de las funciones f

¿Qué tan PRP se ve $f(m) = m^c$?

Si cambio el primer byte, cambia el primer byte 🤔

Permutemos todos los bits de m^c

$$m^c = f_0(B_0) \quad f_1(B_1) \cdots f_6(B_6) \quad f_7(B_7)$$



$$D(m^c) = B_0^1 \qquad B_1^1 \quad \cdots \quad B_6^1 \qquad B_7^1$$

¿Qué tan PRP se ve $D(m^c)$?

Si cambio el primer byte siempre cambian
los mismos 8 bits de la salida 🤔

¿Qué podemos hacer?

¡Repetir!

$$m_0 = B_0^0 \quad B_1^0 \cdots B_6^0 \quad B_7^0$$

$$m_0^c = f_0(B_0^0) \quad f_1(B_1^0) \cdots f_6(B_6^0) \quad f_7(B_7^0)$$

$$m_1 = D(m_0^c) = B_0^1 \qquad B_1^1 \cdots B_6^1 \qquad B_7^1$$

$$m_1^c = f_0(B_0^1) \quad f_1(B_1^1) \cdots f_6(B_6^1) \quad f_7(B_7^1)$$

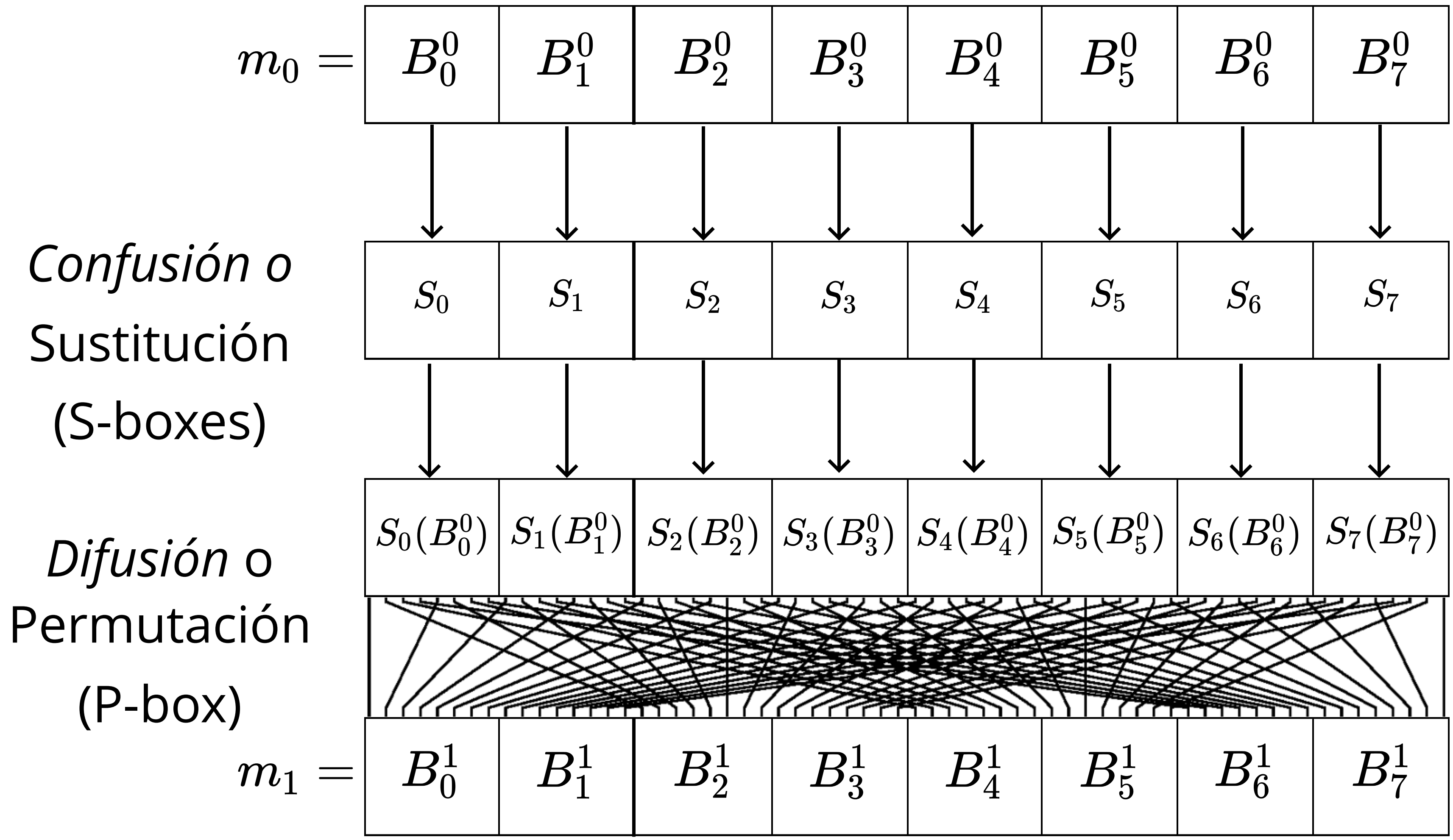
$$m_2 = D(m_1^c) = B_0^2 \qquad B_1^2 \cdots B_6^2 \qquad B_7^2$$

$$m_2^c = f_0(B_0^2) \quad f_1(B_1^2) \cdots f_6(B_6^2) \quad f_7(B_7^2)$$

$$\vdots$$

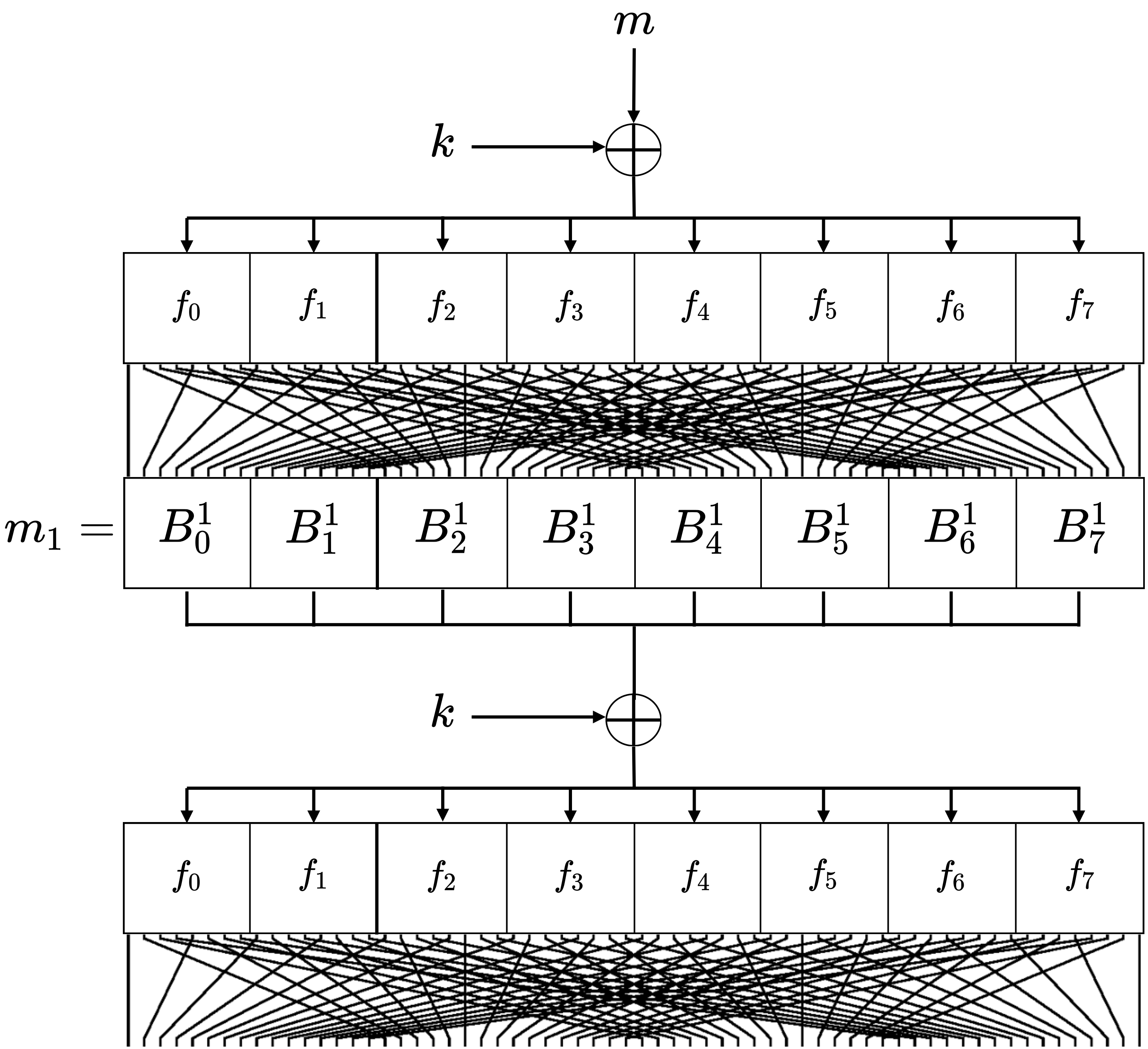
Lo que hemos buscado hasta ahora se
conoce como el "efecto avalancha":

"Si cambio un bit, la salida cambia de forma *aleatoria*"


$m_0 =$
B_0^0 B_1^0 B_2^0 B_3^0 B_4^0 B_5^0 B_6^0 B_7^0
Confusión o Sustitución (S-boxes)
S_0 S_1 S_2 S_3 S_4 S_5 S_6 S_7
$S_0(B_0^0)$ $S_1(B_1^0)$ $S_2(B_2^0)$ $S_3(B_3^0)$ $S_4(B_4^0)$ $S_5(B_5^0)$ $S_6(B_6^0)$ $S_7(B_7^0)$
Difusión o Permutación (P-box)
$m_1 =$
B_0^1 B_1^1 B_2^1 B_3^1 B_4^1 B_5^1 B_6^1 B_7^1

¿Y la llave?

¿Alguna idea?

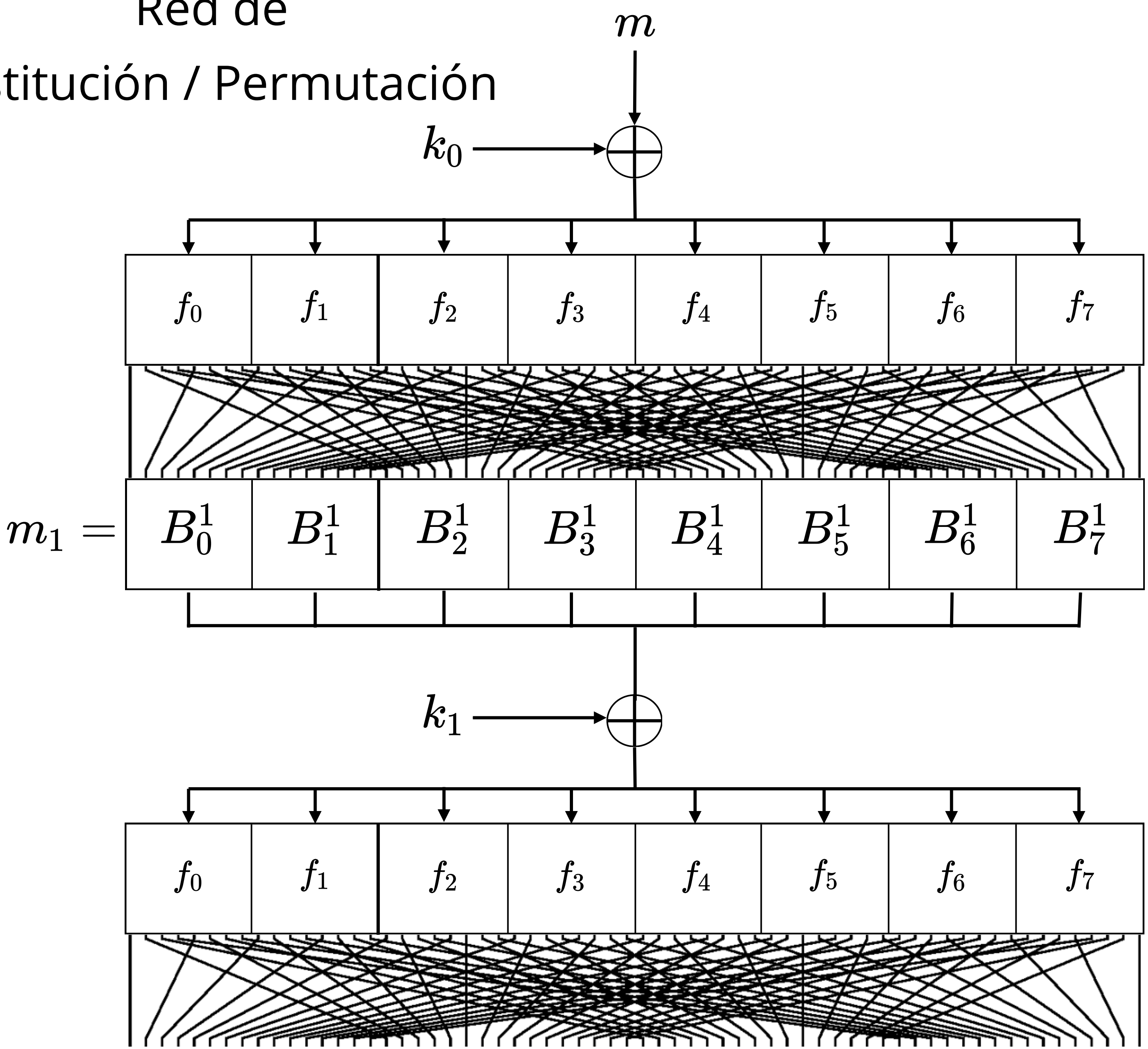


m
k
f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7
$m_1 = B_0^1\ B_1^1\ B_2^1\ B_3^1\ B_4^1\ B_5^1\ B_6^1\ B_7^1$
k
f_0 f_1 f_2 f_3 f_4 f_5 f_6 f_7

En la práctica no se usa directamente
la llave, se usa un *key schedule*

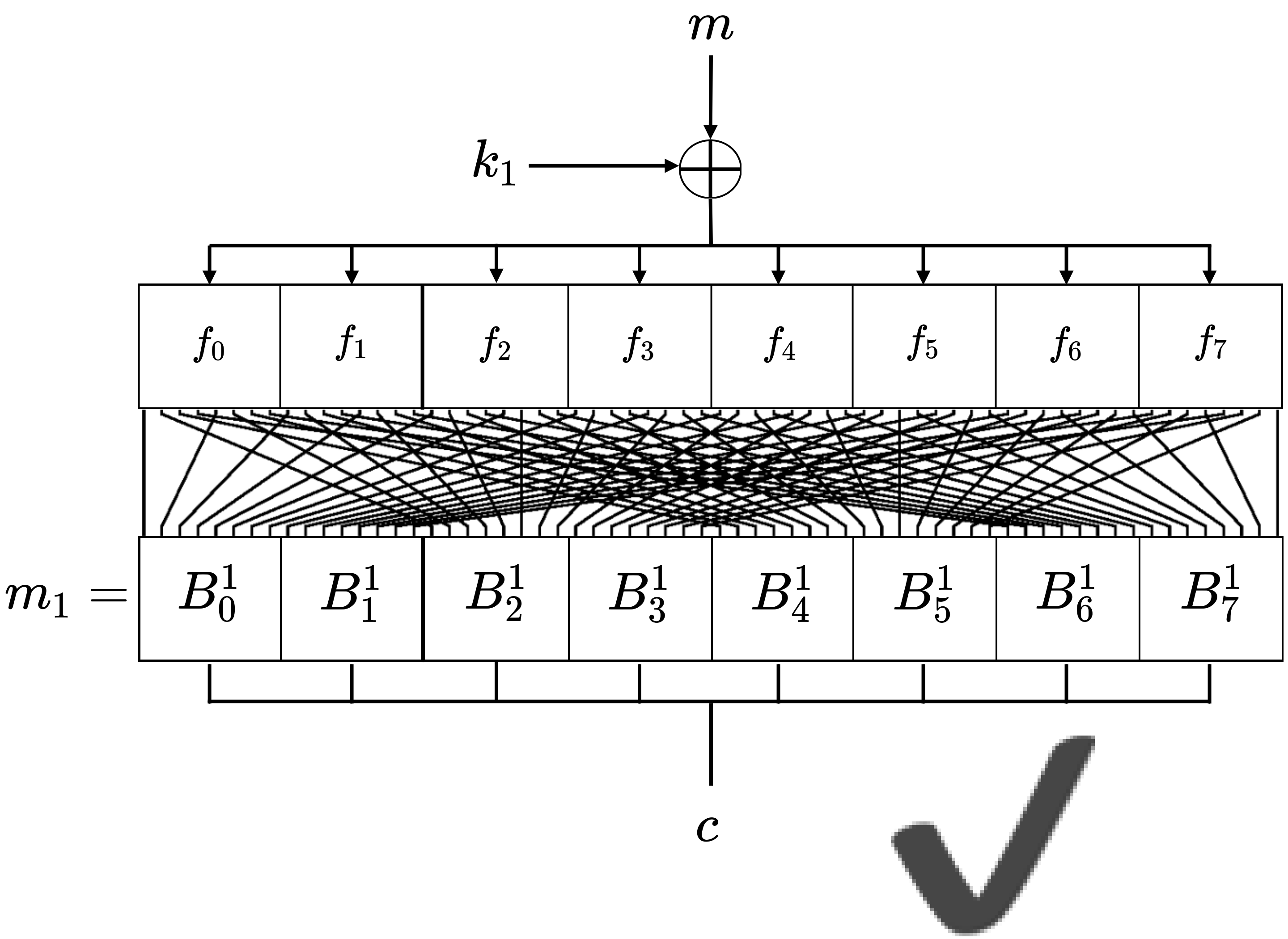
Es un algoritmo determinista que genera una
llave para cada ronda a partir de la llave original

Red de Sustitución / Permutación



Como un esquema criptográfico...

¿Podemos decriptar?


m
k_1
f_0
f_1
f_2
f_3
f_4
f_5
f_6
f_7
$m_1 =$
B_0^1
B_1^1
B_2^1
B_3^1
B_4^1
B_5^1
B_6^1
B_7^1
c

AES

En el llamado a propuestas para definir el
Advanced Encryption Standard (AES, 2001)

The security provided by an algorithm is the most important factor. . . . Algorithms will be judged on the following factors: . . .

- *The extent to which the algorithm output is indistinguishable from a random permutation . . .*

Antes se usaba Data Encryption Standard (DES)
o su extensión a 3 rondas (triple-DES o 3DES)

AES es un *block cipher*, lo que significa que la encriptación se define para *bloques* de largo fijo y luego se extiende a largo arbitrario

Esta extensión se conoce como "modo de operación", veremos cómo funciona más adelante

AES puede funcionar con llaves de 128, 192 y 256 bits; estudiaremos la versión de 128 bits (son todas similares)

¡AES es una red de sustitución/permutación!

- AES-128 tiene 10 rondas
- AES-192 tiene 12 rondas
- AES-256 tiene 14 rondas

Key Schedule

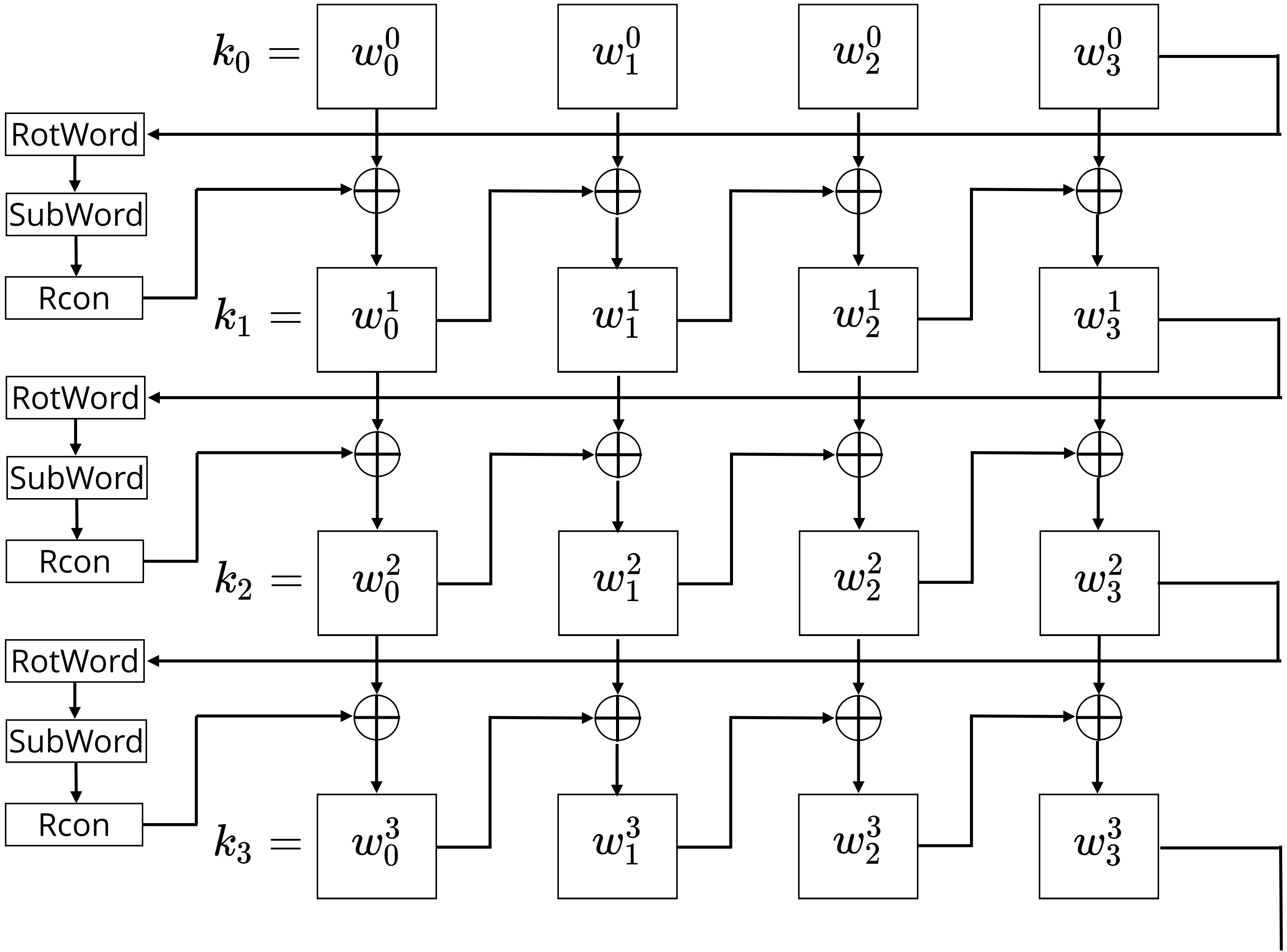
Con 10 rondas tenemos que generar 11 sub-llaves

128 bits = 16 bytes

Matriz de 4x4 bytes = 4 columnas de 4 bytes

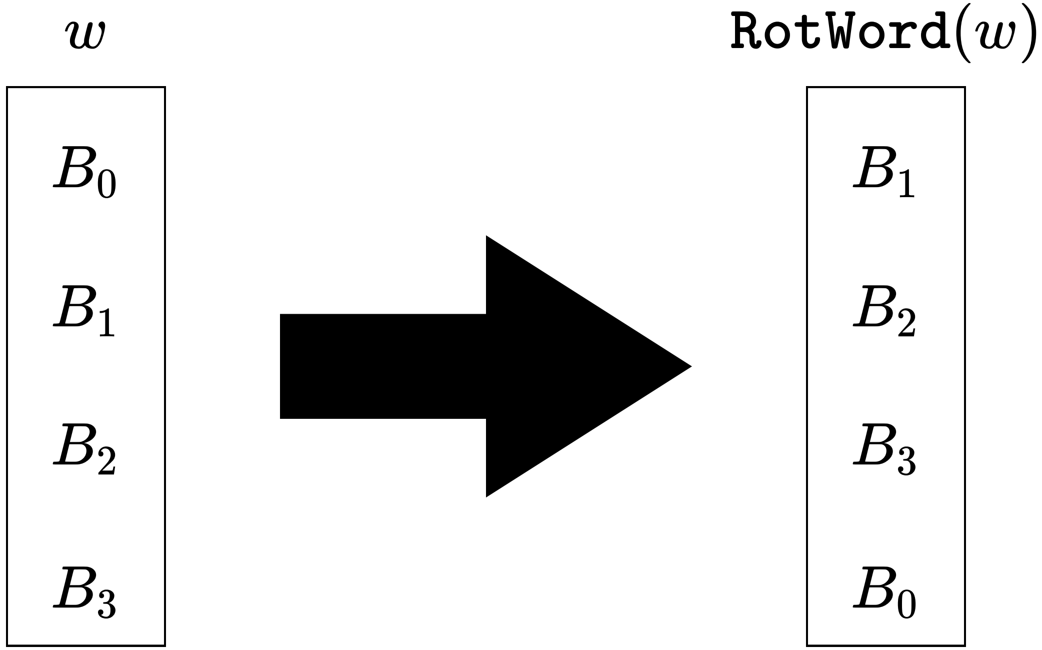
$k_0 = k$

w_0^0	w_1^0	w_2^0	w_3^0
k_0^0	k_4^0	k_8^0	k_{12}^0
k_1^0	k_5^0	k_9^0	k_{13}^0
k_2^0	k_6^0	k_{10}^0	k_{14}^0
k_3^0	k_7^0	k_{11}^0	k_{15}^0

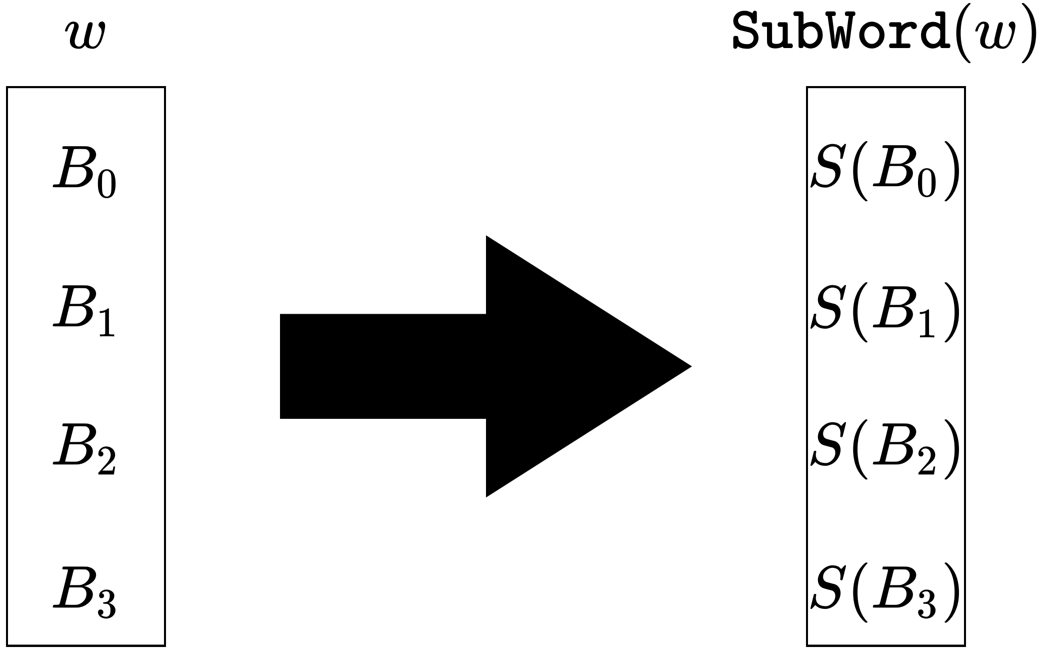


$k_0 =$
w_0^0
w_1^0
w_2^0
w_3^0
RotWord
SubWord
Rcon
$k_1 =$
w_0^1
w_1^1
w_2^1
w_3^1
RotWord
SubWord
Rcon
$k_2 =$
w_0^2
w_1^2
w_2^2
w_3^2
RotWord
SubWord
Rcon
$k_3 =$
w_0^3
w_1^3
w_2^3
w_3^3

RotWord



SubWord



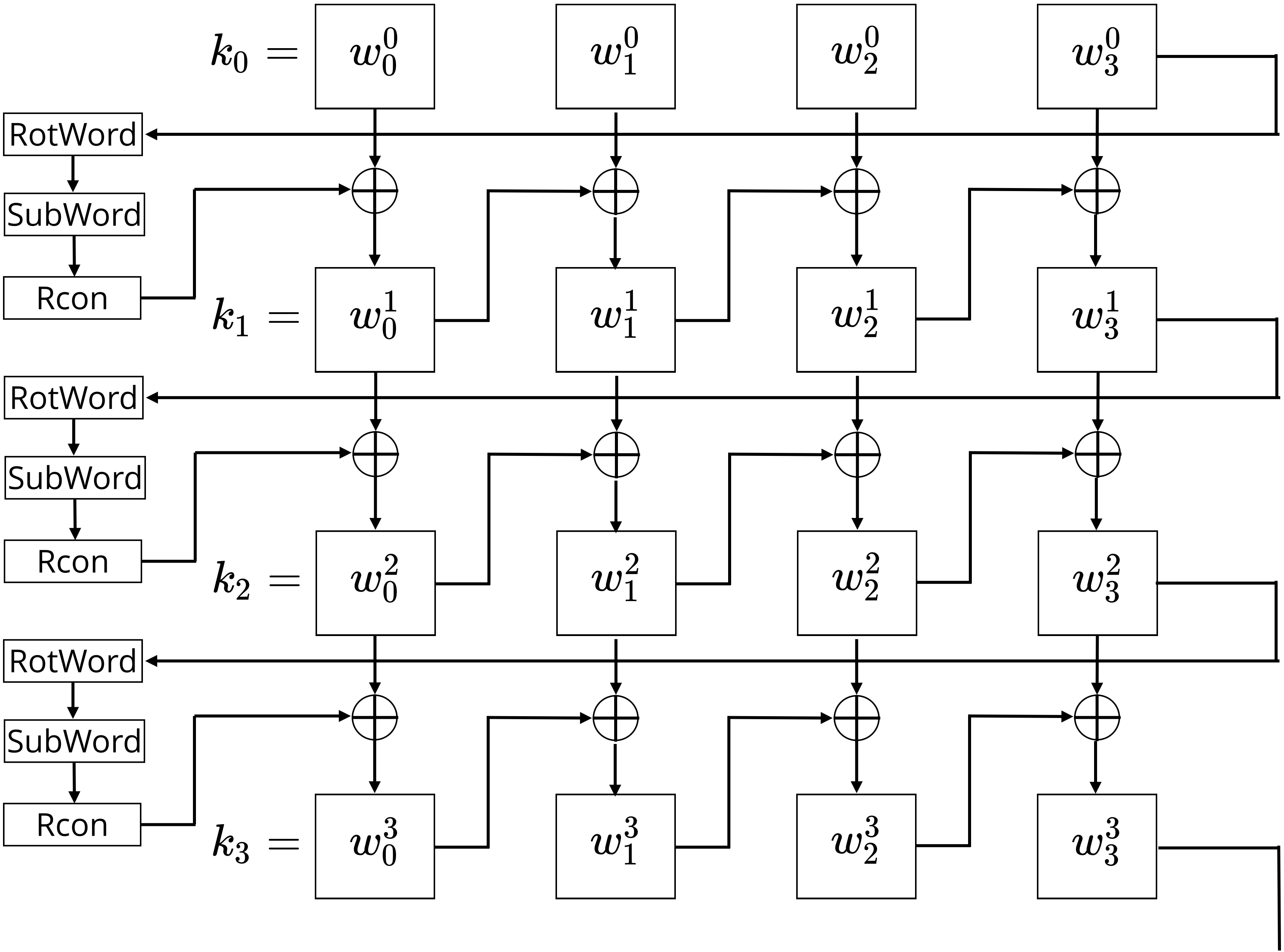
AES S-Box

$$\underset{S(B)}{\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \underset{B}{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

AES S-Box

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

Implementación


$k_0 =$
w_0^0
w_1^0
w_2^0
w_3^0
RotWord
SubWord
Rcon
$k_1 =$
w_0^1
w_1^1
w_2^1
w_3^1
RotWord
SubWord
Rcon
$k_2 =$
w_0^2
w_1^2
w_2^2
w_3^2
RotWord
SubWord
Rcon
$k_3 =$
w_0^3
w_1^3
w_2^3
w_3^3

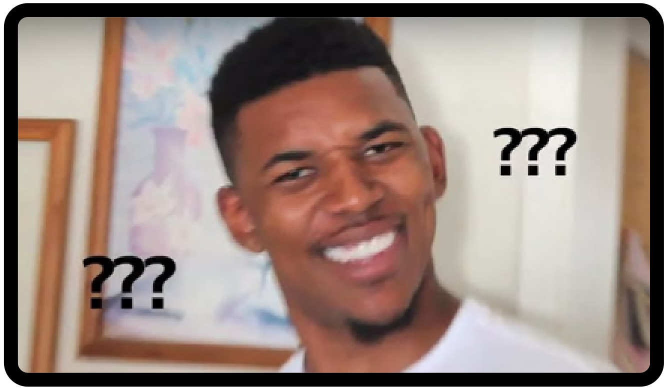
Rcon

$$
\begin{array}{c}
w \\
\boxed{\begin{array}{c} B_0 \\ B_1 \\ B_2 \\ B_3 \end{array}}
\end{array}
\quad \Longrightarrow \quad
\begin{array}{c}
\texttt{Rcon}(w) \\
\boxed{\begin{array}{c} B_0 \bigoplus rc_i \\ B_1 \\ B_2 \\ B_3 \end{array}}
\end{array}
$$

Para la ronda i, el valor rc_i se define como el vector de coeficientes del polinomio

$$x^{i-1} \mod x^8 + x^4 + x^3 + x + 1$$

en módulo 2



$$x^{i-1} \mod x^8+x^4+x^3+x+1$$

$$10 \mod 3 = 1$$

Ronda 10

¿Cuántas veces cabe $x^8+x^4+x^3+x+1$ en $x^{10-1}=x^9$?

x veces, ¿y el resto es?

el polinomio $-x^5-x^4-x^2-x \ = x^5+x^4+x^2+x$

vector de coeficientes = 00110110

que en decimal es 54

y en hexadecimal 36

Ronda 9

Cuántas veces cabe $x^8 + x^4 + x^3 + x + 1$ en $x^{9-1} = x^8$?

1 vez, ¿y el resto es?

el polinomio $-x^4 - x^3 - x - 1 \quad = x^4 + x^3 + x + 1$

vector de coeficientes? 00011011

que en decimal es 27

y en hexadecimal 1B

Ronda 8

¿Cuántas veces cabe $x^8+x^4+x^3+x+1$ en $x^{8-1}=x^7$?

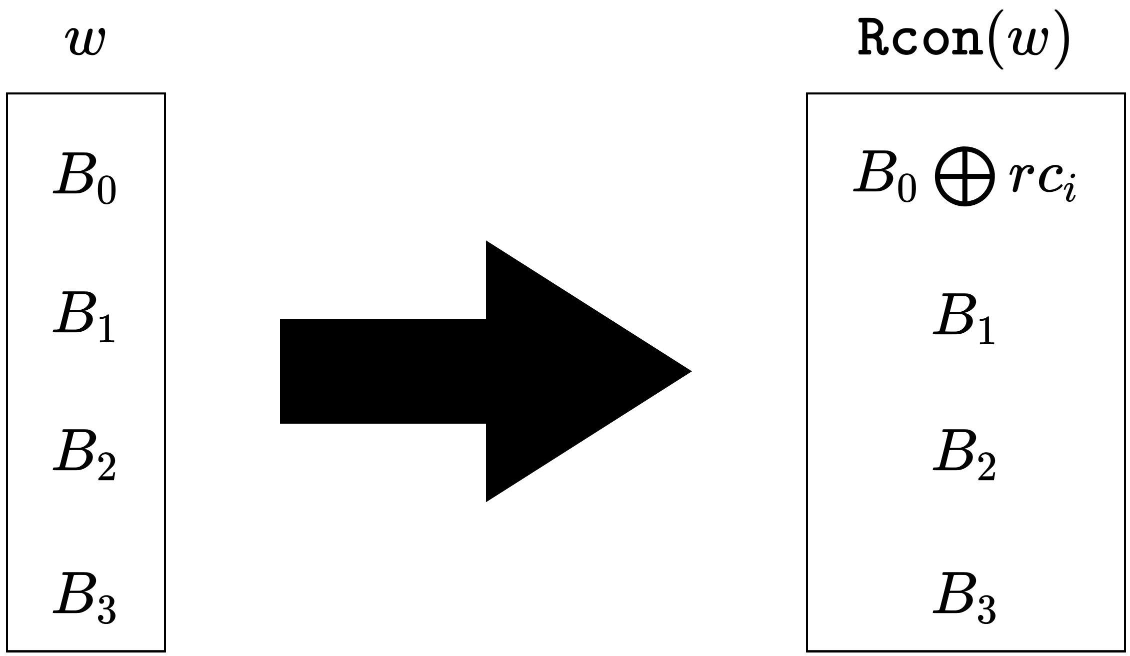
Ninguna, ¿y el resto es?

x^7, cuyos coeficientes en binario son

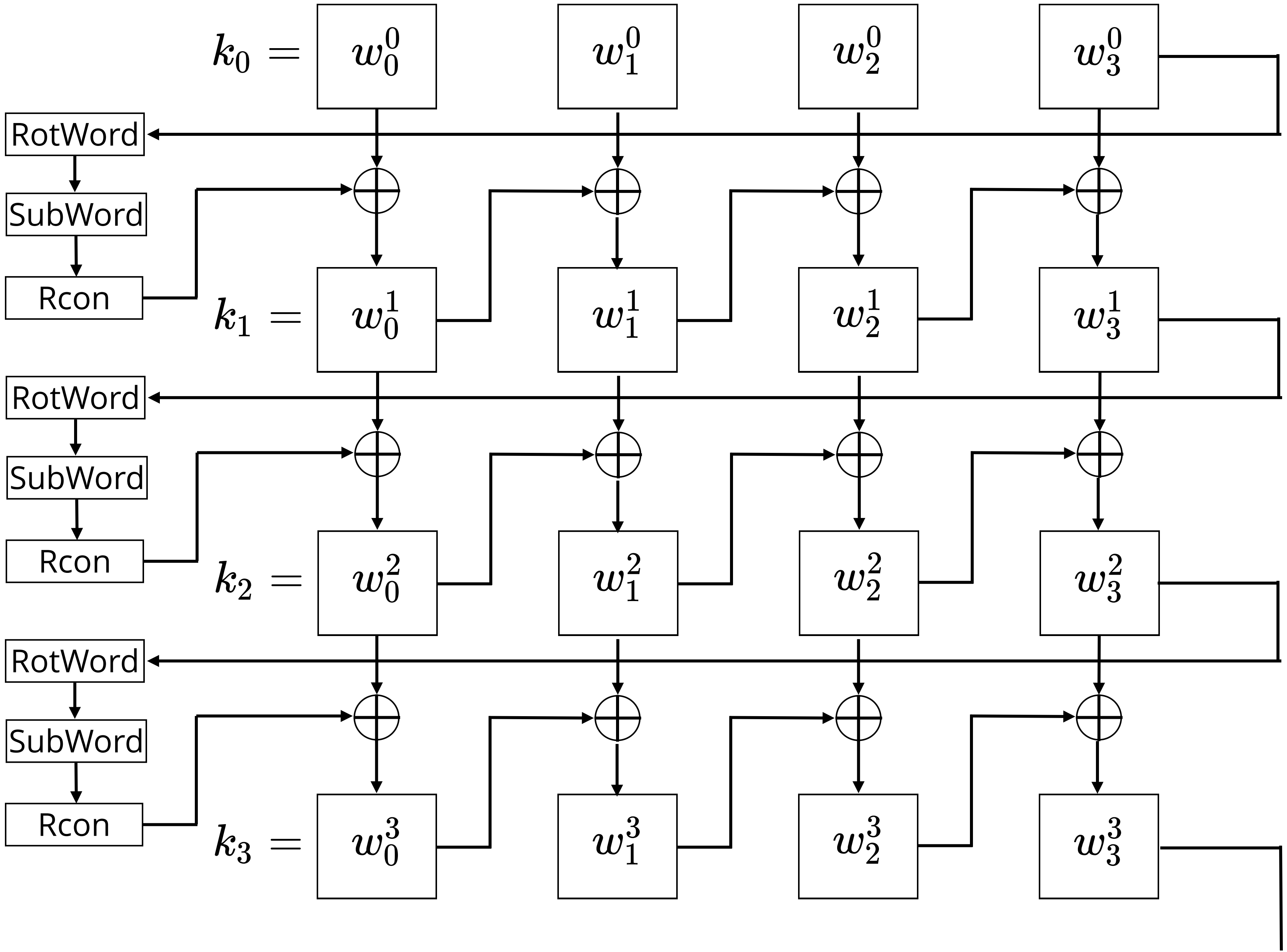
10000000

y en hexadecimal

80

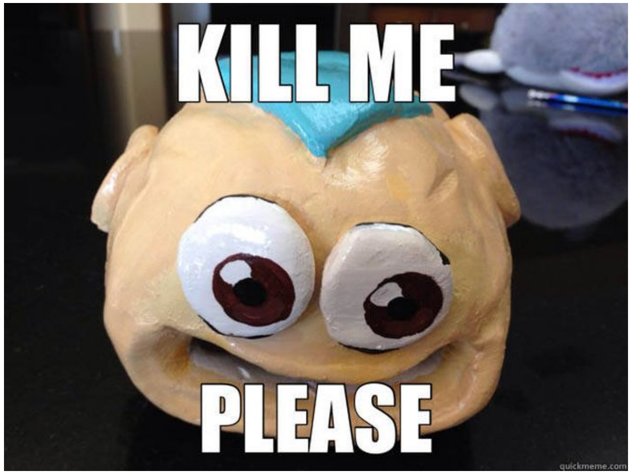


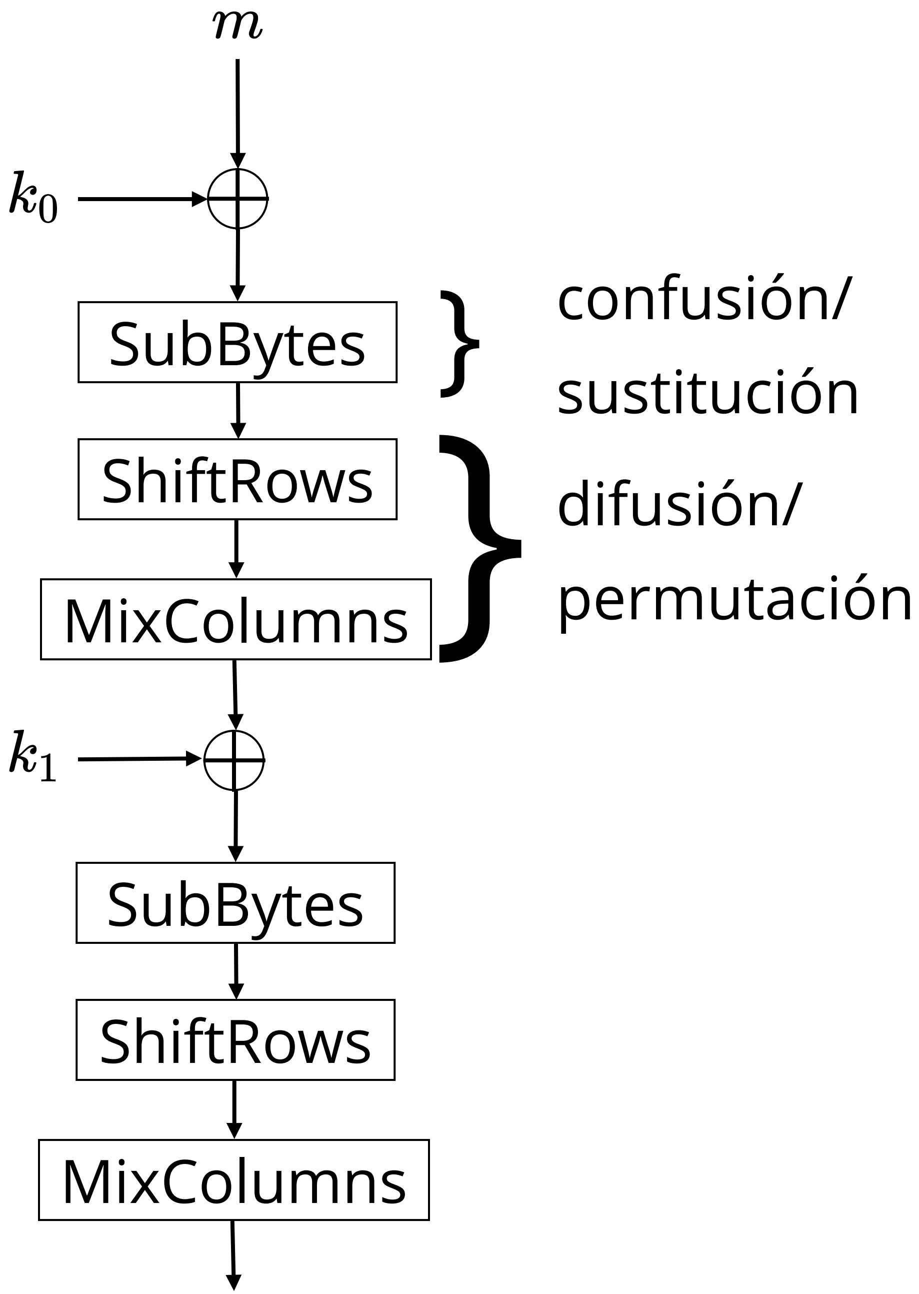
rc_i = [01 02 04 08 10 20 40 80 1B 36]


$k_0 =$
w_0^0
w_1^0
w_2^0
w_3^0
RotWord
SubWord
Rcon
$k_1 =$
w_0^1
w_1^1
w_2^1
w_3^1
RotWord
SubWord
Rcon
$k_2 =$
w_0^2
w_1^2
w_2^2
w_3^2
RotWord
SubWord
Rcon
$k_3 =$
w_0^3
w_1^3
w_2^3
w_3^3

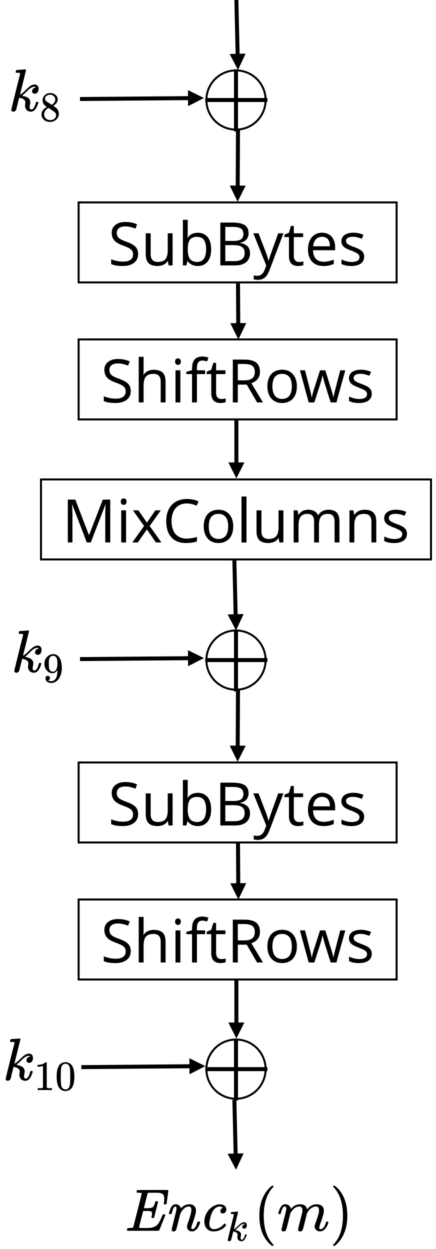
Tenemos $\{k_0, \dots, k_{10}\}$

Y todavía falta todo AES





m
k_0
SubBytes
ShiftRows
MixColumns
k_1
SubBytes
ShiftRows
MixColumns
confusión/
sustitución
difusión/
permutación



k_8
SubBytes
ShiftRows
MixColumns
k_9
SubBytes
ShiftRows
k_{10}
$Enc_k(m)$

Diremos que AES tiene un *estado* de 128 bits, que es lo que se va modificando en cada paso

Este estado lo representaremos como una matriz de 4x4 bytes

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

SubBytes

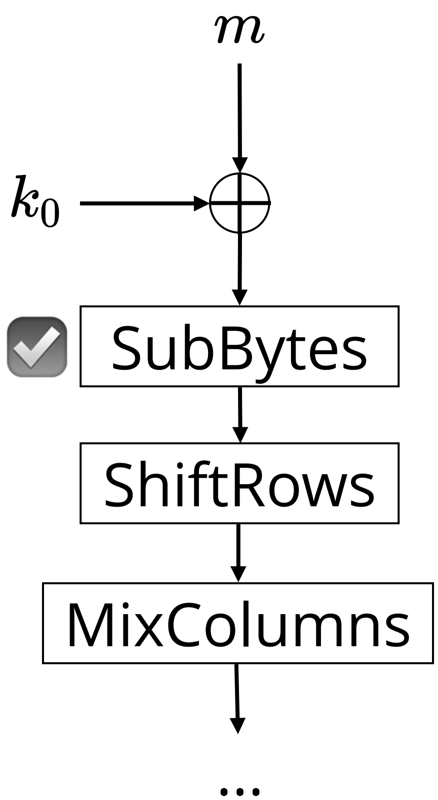
B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

$S(B_0)$	$S(B_4)$	$S(B_8)$	$S(B_{12})$
$S(B_1)$	$S(B_5)$	$S(B_9)$	$S(B_{13})$
$S(B_2)$	$S(B_6)$	$S(B_{10})$	$S(B_{14})$
$S(B_3)$	$S(B_7)$	$S(B_{11})$	$S(B_{15})$

AES S-Box

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

Hello S-Box my old friend!

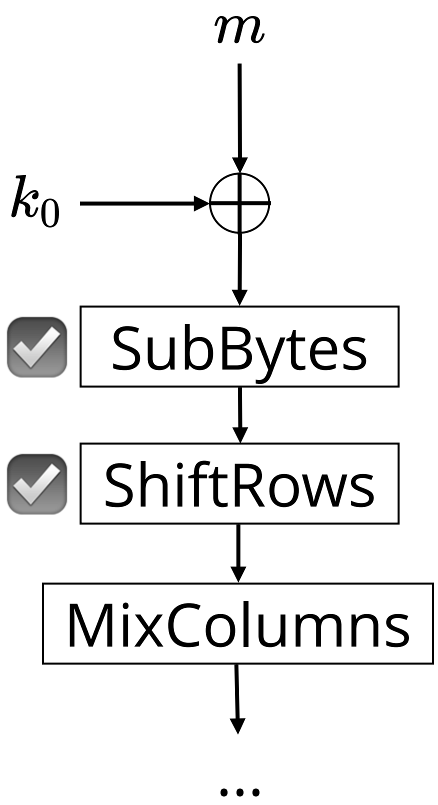


m
k_0
SubBytes
ShiftRows
MixColumns
...

ShiftRows

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

B_0	B_4	B_8	B_{12}
B_5	B_9	B_{13}	B_1
B_{10}	B_{14}	B_2	B_6
B_{15}	B_3	B_7	B_{11}


m
k_0
SubBytes
ShiftRows
MixColumns
...

MixColumns

$$\begin{array}{|c|c|c|c|}\hline 02 & 03 & 01 & 01 \\ \hline 01 & 02 & 03 & 01 \\ \hline 01 & 01 & 02 & 03 \\ \hline 03 & 01 & 01 & 02 \\ \hline \end{array} \bullet \begin{array}{|c|c|c|c|}\hline B_0 & B_4 & B_8 & B_{12} \\ \hline B_1 & B_5 & B_9 & B_{13} \\ \hline B_2 & B_6 & B_{10} & B_{14} \\ \hline B_3 & B_7 & B_{11} & B_{15} \\ \hline \end{array}$$

Donde el punto representa una multiplicación de matrices muy particular...

Ejemplo

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} 63 & EB & 9F & A0 \\ 2F & 93 & 92 & C0 \\ AF & C7 & AB & 30 \\ A2 & 20 & CB & 2B \end{bmatrix} = \begin{bmatrix} & & & \\ & & & \\ & & & \\ & & & \end{bmatrix}$$

x = 00000010 02 x 63 01100011 = $x^6 + x^5 + x + 1$

$x + 1$ = 00000011 03 x 2F 00101111 = $x^5 + x^3 + x^2 + x + 1$

1 = 00000001 01 x AF 10101111 = $x^7 + x^5 + x^3 + x^2 + x + 1$

1 = 00000001 01 X A2 10100010 = $x^7 + x^5 + x$

$$x(x^6 + x^5 + x + 1) \text{ +}$$

$$(x+1)(x^5 + x^3 + x^2 + x + 1) \text{ +}$$

$$(x^7 + x^5 + x^3 + x^2 + x + 1) \text{ +}$$

$$(x^7 + x^5 + x)$$

$$= 3x^7 + 2x^6 + 3x^5 + x^4 + 3x^3 + 4x^2 + 5x + 2$$

$$= x^7 + x^5 + x^4 + x^3 + x$$

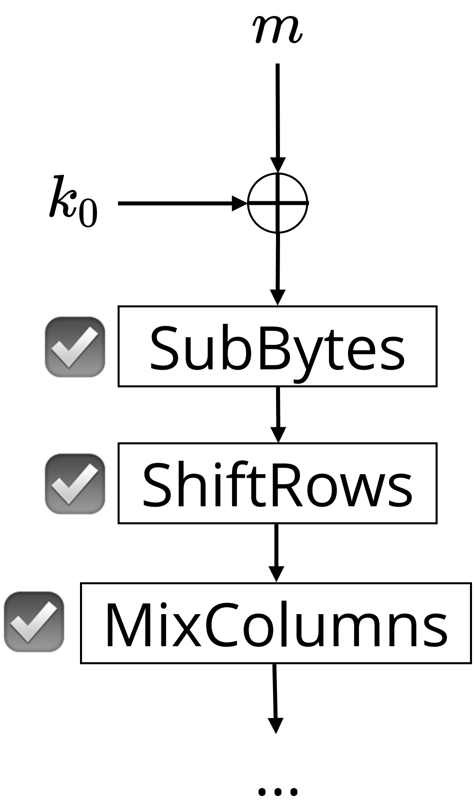
Si el polinomio resultante tuviera x^8,
le restamos $x^8 + x^4 + x^3 + x + 1$

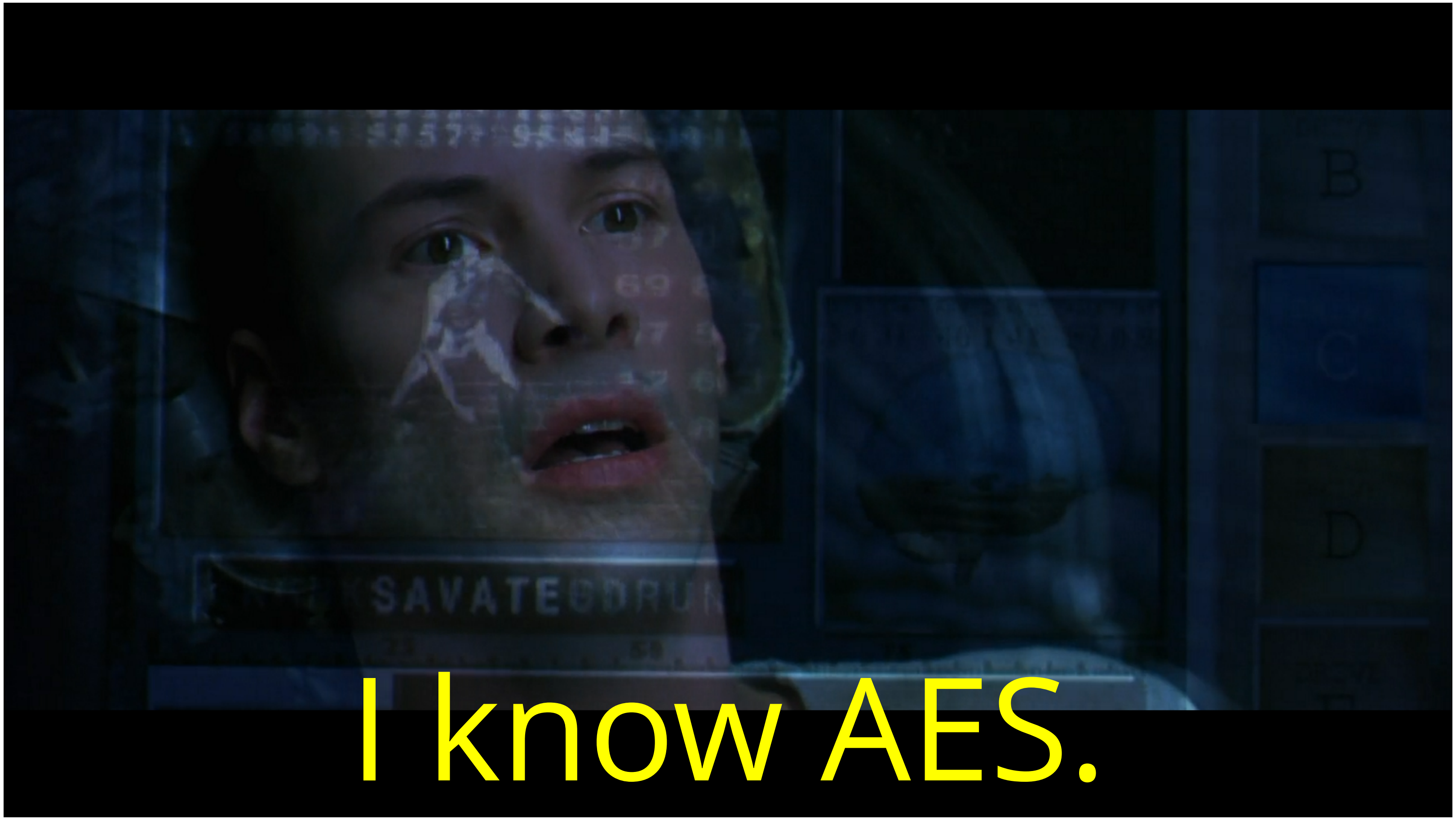
Volvemos a binario: 10111010

En hexadecimal: BA

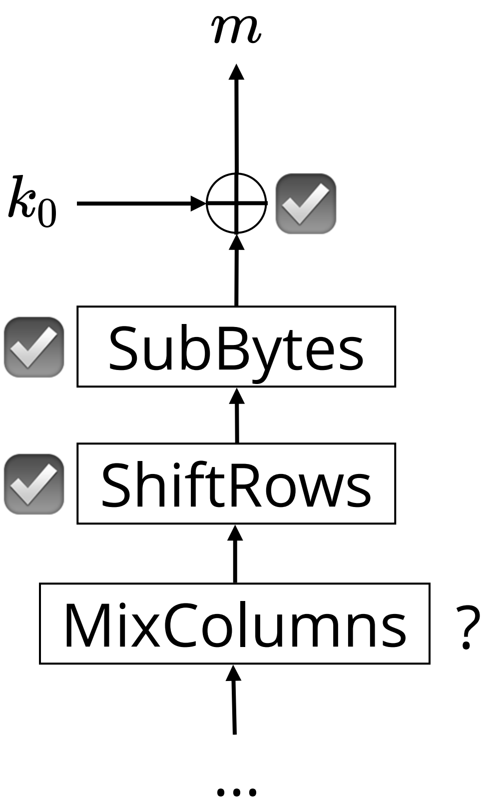
Ejemplo

$$
\begin{array}{|c|c|c|c|}
\hline
02 & 03 & 01 & 01 \\ \hline
01 & 02 & 03 & 01 \\ \hline
01 & 01 & 02 & 03 \\ \hline
03 & 01 & 01 & 02 \\ \hline
\end{array}
\bullet
\begin{array}{|c|c|c|c|}
\hline
\mathrm{63} & \mathrm{EB} & \mathrm{9F} & \mathrm{A0} \\ \hline
\mathrm{2F} & \mathrm{93} & \mathrm{92} & \mathrm{C0} \\ \hline
\mathrm{AF} & \mathrm{C7} & \mathrm{AB} & \mathrm{30} \\ \hline
\mathrm{A2} & \mathrm{20} & \mathrm{CB} & \mathrm{2B} \\ \hline
\end{array}
=
\begin{array}{|c|c|c|c|}
\hline
\mathrm{BA} & \mathrm{84} & \mathrm{E8} & \mathrm{1B} \\ \hline
\mathrm{75} & \mathrm{A4} & \mathrm{8D} & \mathrm{40} \\ \hline
\mathrm{F4} & \mathrm{8D} & \mathrm{06} & \mathrm{7D} \\ \hline
\mathrm{7A} & \mathrm{32} & \mathrm{0E} & \mathrm{5D} \\ \hline
\end{array}
$$


m
k_0
SubBytes
ShiftRows
MixColumns
...


SAVATE
I know AES.

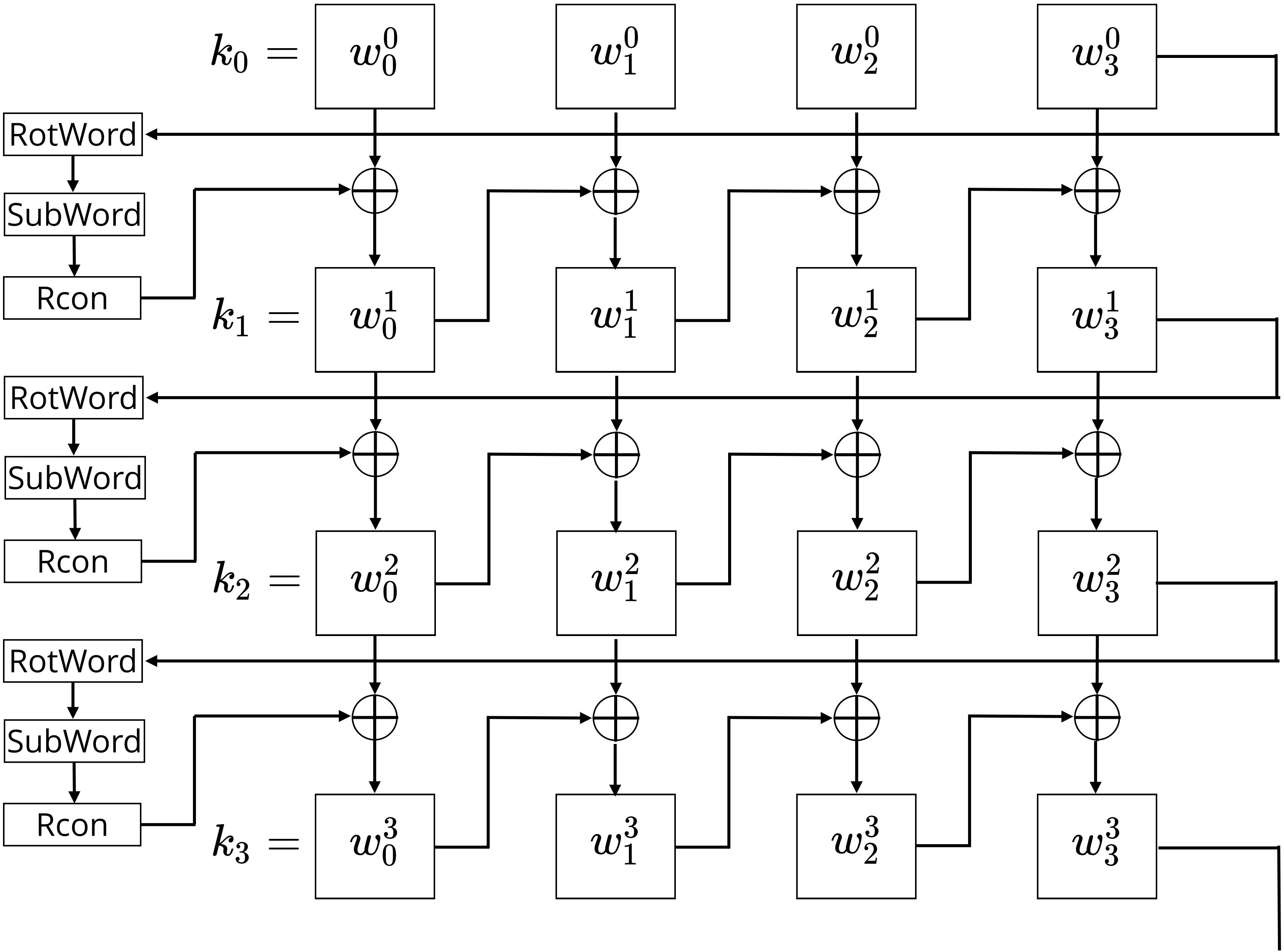
Wait. ¿cómo decriptamos?

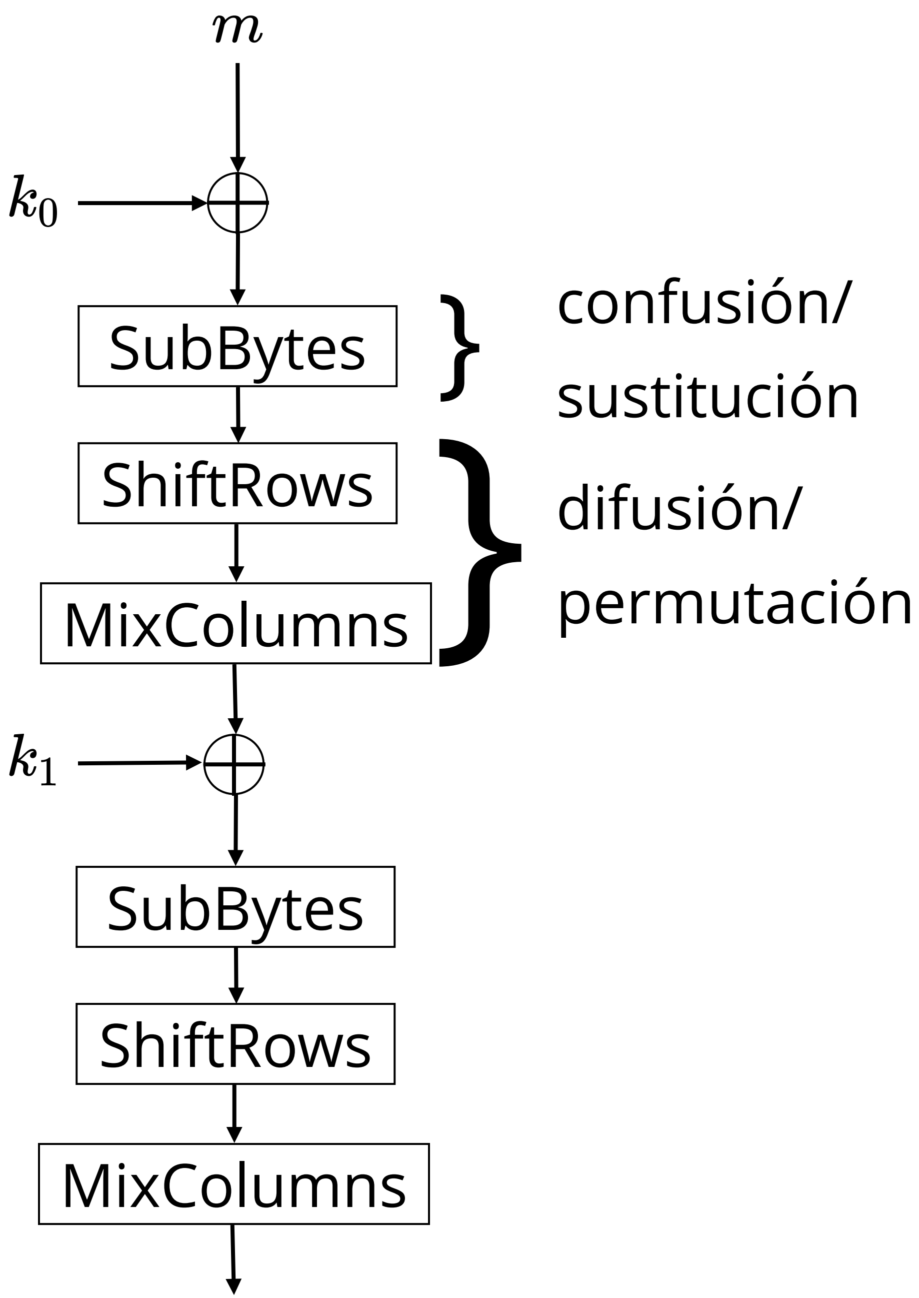


Para invertir MixColumns usamos la misma multiplicación retorcida, pero esta vez por la matriz

0E	0B	0D	09
09	0E	0B	0D
0D	09	0E	0B
0B	0D	09	0E

Lo consideraremos magia negra...


$k_0 =$
w_0^0
w_1^0
w_2^0
w_3^0
RotWord
SubWord
Rcon
$k_1 =$
w_0^1
w_1^1
w_2^1
w_3^1
RotWord
SubWord
Rcon
$k_2 =$
w_0^2
w_1^2
w_2^2
w_3^2
RotWord
SubWord
Rcon
$k_3 =$
w_0^3
w_1^3
w_2^3
w_3^3


m
k_0
SubBytes
ShiftRows
MixColumns
confusión/
sustitución
difusión/
permutación
k_1
SubBytes
ShiftRows
MixColumns

¿Cómo lo extendemos para mensajes de largo arbitrario?

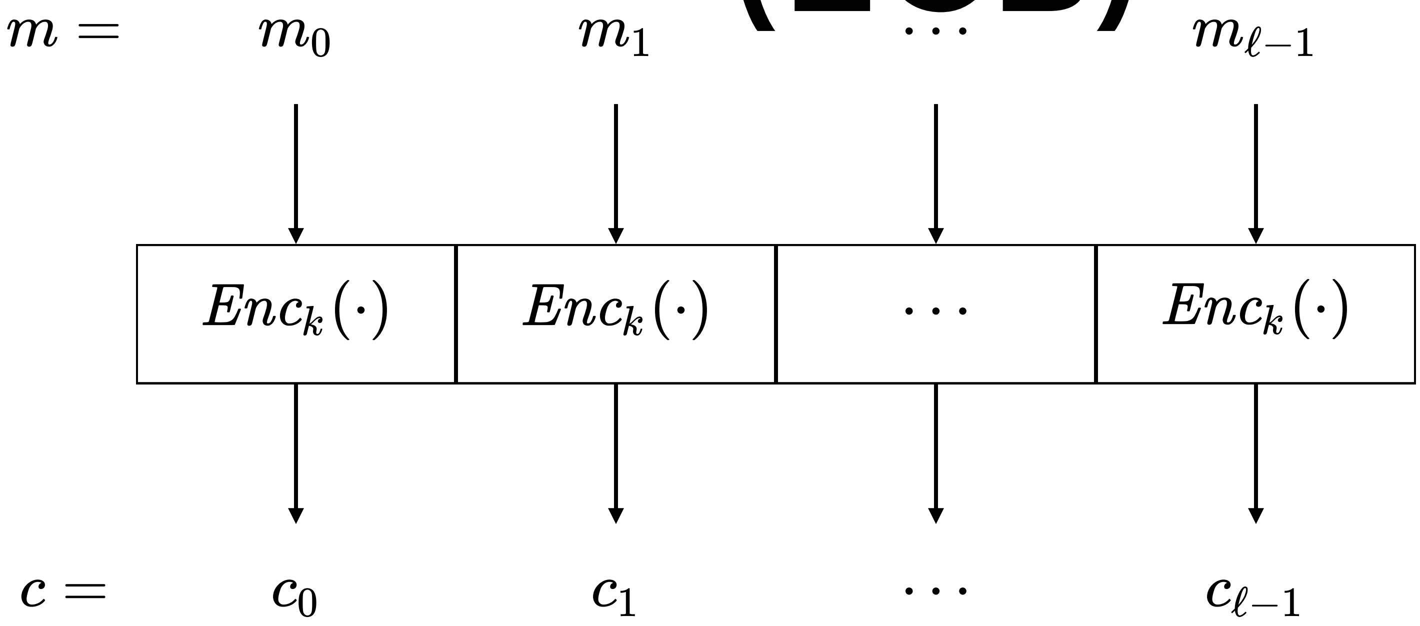
Modos de Operación

Supondremos que tenemos un mensaje m que dividiremos en ℓ *bloques*

$$m = m_0\, m_1\, \cdots\, m_{\ell-1}$$

Donde cada m_i tiene 128 bits. Si $|m|$ no divide a 128 le agregamos un *padding*

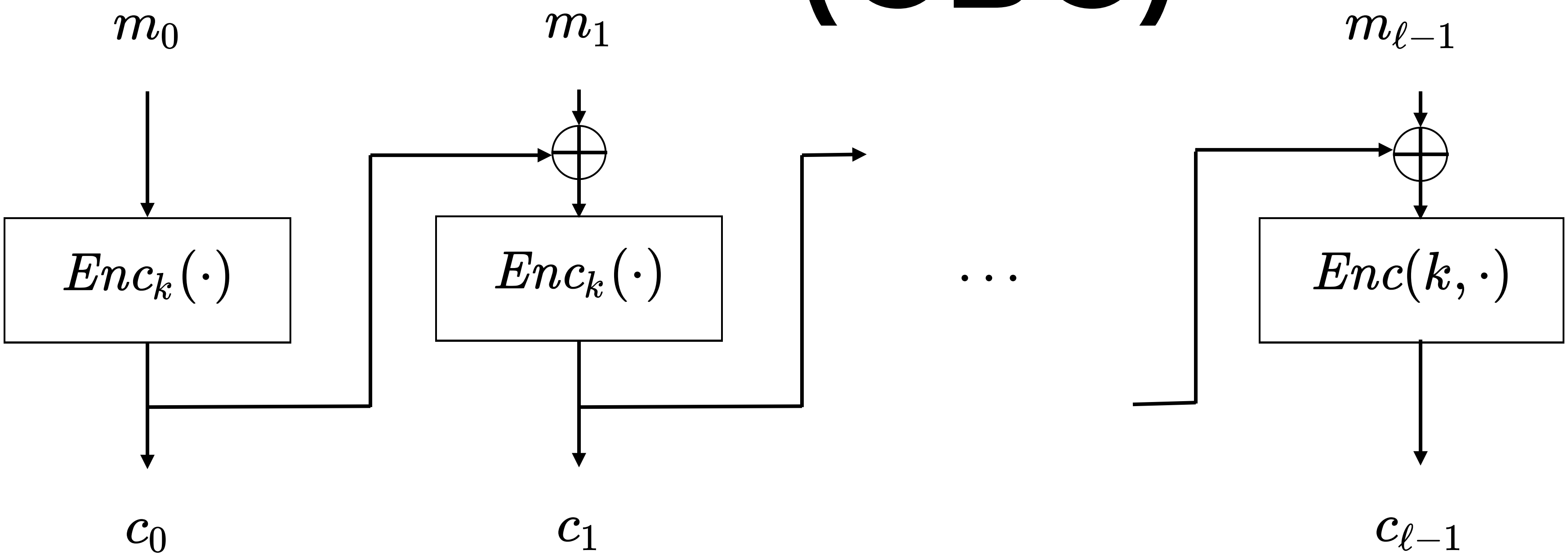
Electronic Code Book (ECB)



¿Problemas?

Si cambio sólo el primer bloque,
cambia sólo el primer bloque

Cipher Block Chaining* (CBC)



¿Problemas prácticos?

Si mando dos veces el mismo mensaje, el atacante lo sabrá

¿Qué le pedimos a un esquema de encriptación
con mensajes de largo arbitrario?

¿Se puede ver como una PRP?

En la práctica esperamos propiedades algo distintas

A jugar...

1. El verificador toma k en base a $Gen(1^n)$
2. El adversario puede preguntar lo que quiera a $Enc_k(\cdot)$
3. El adversario genera mensajes m_0 y m_1 y los envía al verificador
4. El verificador elige $b \in \{0,1\}$ y envía al adversario $c = Enc_k(m_b)$
5. El adversario puede preguntar lo que quiera a $Enc_k(\cdot)$
6. El adversario elige $b' \in \{0,1\}$ y gana si $b = b'$

¿Cómo puede tratar de hacer trampa el adversario?

➡ La función de encriptación **debe** ser aleatorizada

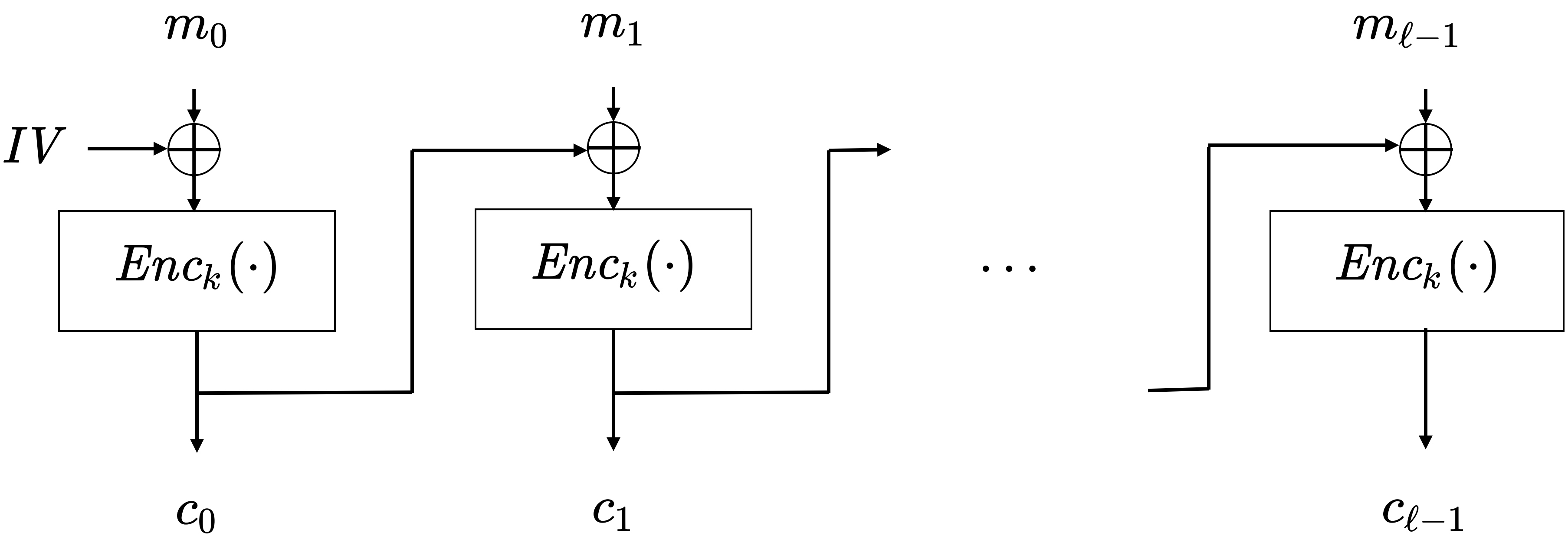
Un esquema criptográfico es seguro frente a ataques de texto elegido (CPA-secure) si

Para todo adversario A

$$\left|\frac{1}{2} - \mathbf{Pr}[A \text{ gane el juego }]\right|$$

Es una función despreciable en n

CBC



$IV =$ vector de inicialización aleatorio

Son 128 bits que se deben compartir cada vez que se encripta un mensaje.

Teorema 💫

Si Enc es una PRP, entonces el esquema CBC es CPA-secure


WELL, I THINK
WE'RE
DONE HERE
quickmeme.com