




# IIC3253

Criptografía simétrica: definición y construcciones prácticas

# 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?

```
1 def adv(f: (string) -> string) -> bool:
2     """
3     parameters:
4         f: An arbitrary permutation
5     returns:
6         b: Guess of f == Enc(k, .)
7     """
```

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

```
1 def adv(f: (string) -> string) -> bool:
2     """
3     parameters:
4         f: An arbitrary permutation
5     returns:
6         b: Guess of f == Enc(k, .)
7     """
```

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?

```
1 def adv(n: str, f: (string) -> string) -> bool:
2     """
3     parameters:
4         n: Security parameter (unary)
5         f: An arbitrary permutation
6     returns:
7         b: Guess of f == Enc(Gen(n^1), .)
8     """
```

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  $\mathbf{Adv}$  de tiempo polinomial:

$$\left| \Pr_{k \sim \text{Gen}(1^n)} [\mathbf{Adv}(1^n, \text{Enc}(k, \cdot))] - \Pr_{f \sim \mathbb{U}} [\mathbf{Adv}(1^n, f(\cdot))] \right|$$

Es una función despreciable

# Definiciones formales

- Funciones de hash 
- Message Authentication Codes 
- Esquemas criptográficos 



**Vamos a la práctica**

$$Enc : \{0, 1\}^n \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$$

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  $2^n$   
 $Enc : \{0, 1\}^n \times \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$

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  $8 \cdot 2^8 = 4096$ , o 512 bytes ✓

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

$$m = b_0 \quad b_1 \cdots b_6 \quad b_7$$

$$m^c = f_0(b_0) \quad f_1(b_1) \cdots f_6(b_6) \quad f_7(b_7)$$

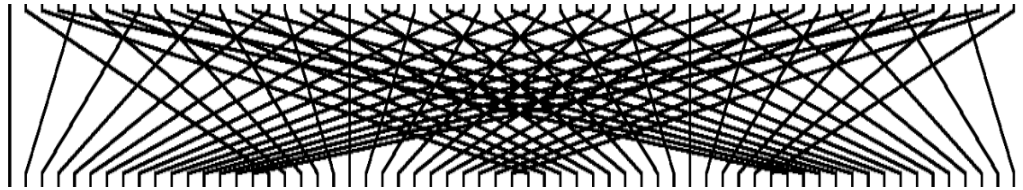
Imaginemos por un momento que la llave define el orden de las funciones  $f$

Qué tan PRP se ve  $f(m) = m_c$ ?

Si cambio el primer byte, cambia el primer byte 🤔

Permutemos todos los bits de  $m_c$

$$m^c = f_0(b_0) \quad f_1(b_1) \cdots f_6(b_6) \quad f_7(b_7)$$



$$D(m^c) = b_0^1 \quad b_1^1 \quad \cdots \quad b_6^1 \quad b_7^1$$

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!

$$m_0 = b_0^0 \quad b_1^0 \cdots b_{14}^0 \quad b_{15}^0$$

$$m_0^c = f_0(b_0^0) \quad f_1(b_1^0) \cdots f_{14}(b_{14}^0) \quad f_{15}(b_{15}^0)$$

$$m_1 = D(m_0^c) = b_0^1 \quad b_1^1 \cdots b_{14}^1 \quad b_{15}^1$$

$$m_1^c = f_0(b_0^1) \quad f_1(b_1^1) \cdots f_{14}(b_{14}^1) \quad f_{15}(b_{15}^1)$$

$$m_2 = D(m_1^c) = b_0^2 \quad b_1^2 \cdots b_{14}^2 \quad b_{15}^2$$

$$m_2^c = f_0(b_0^2) \quad f_1(b_1^2) \cdots f_{14}(b_{14}^2) \quad f_{15}(b_{15}^2)$$

⋮

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)*

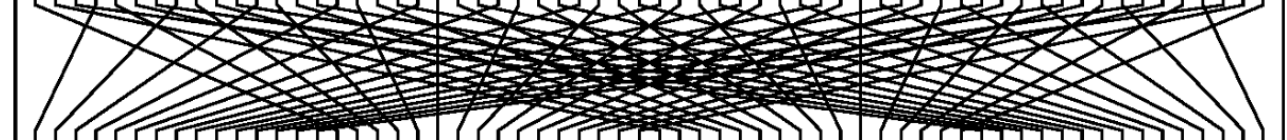


$s_0$	$s_1$	$s_2$	$s_3$	$s_4$	$s_5$	$s_6$	$s_7$
-------	-------	-------	-------	-------	-------	-------	-------



$s_0(b_0^0)$	$s_1(b_1^0)$	$s_2(b_2^0)$	$s_3(b_3^0)$	$s_4(b_4^0)$	$s_5(b_5^0)$	$s_6(b_6^0)$	$s_7(b_7^0)$
--------------	--------------	--------------	--------------	--------------	--------------	--------------	--------------

*Difusión o  
Permutación  
(P-box)*

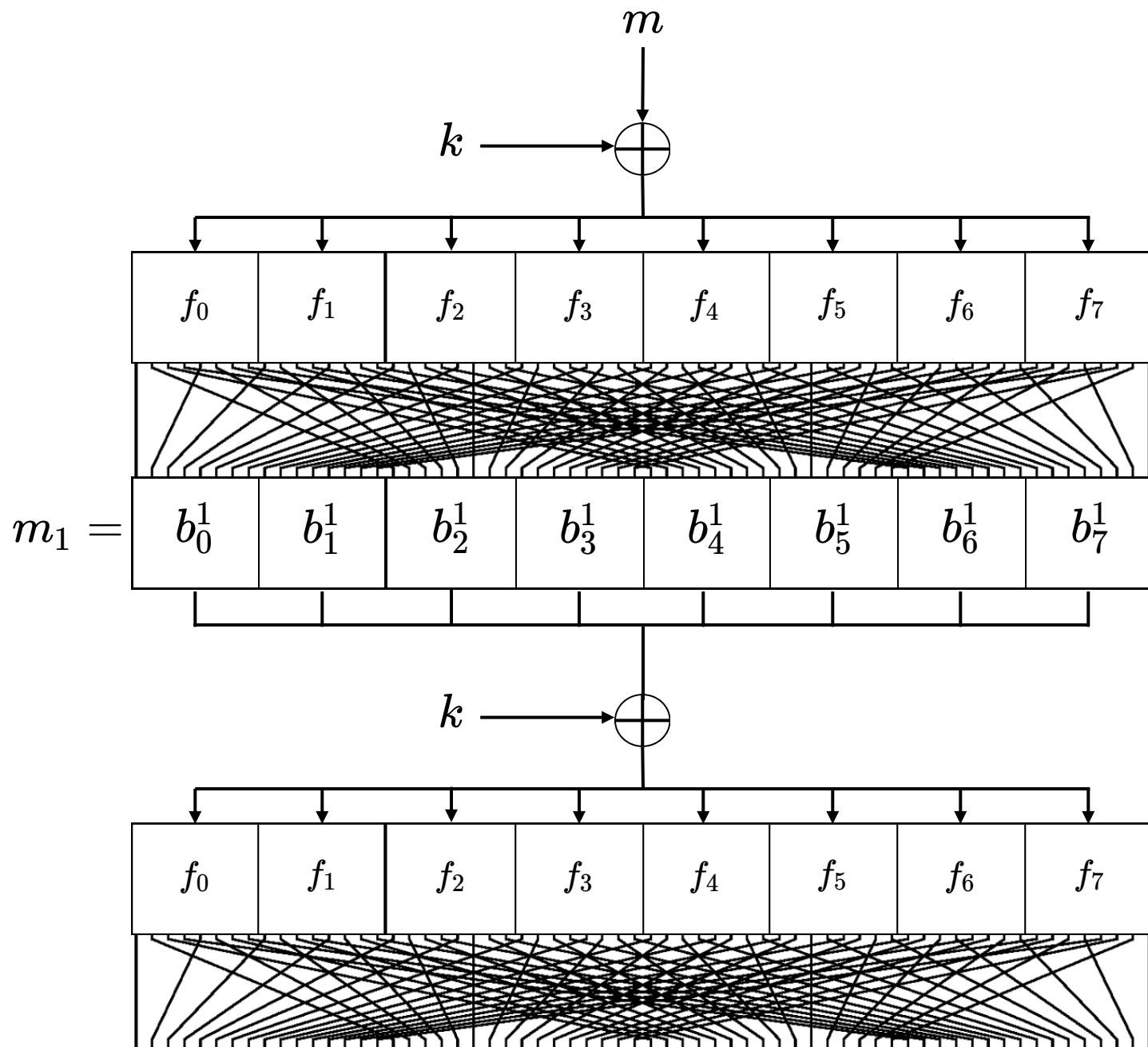


$m_1 =$

$b_0^1$	$b_1^1$	$b_2^1$	$b_3^1$	$b_4^1$	$b_5^1$	$b_6^1$	$b_7^1$
---------	---------	---------	---------	---------	---------	---------	---------

Y la llave?

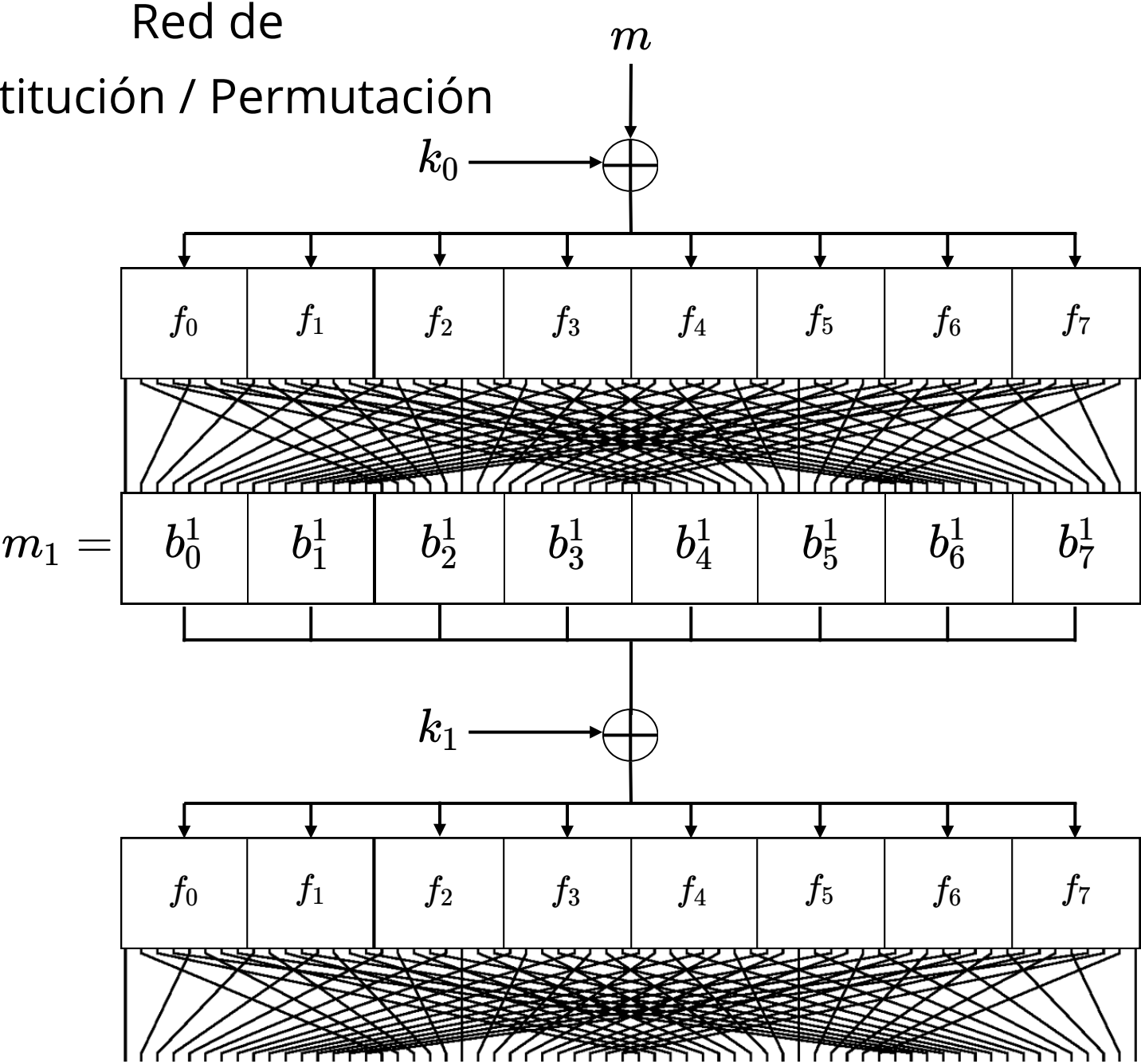
Alguna idea?



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

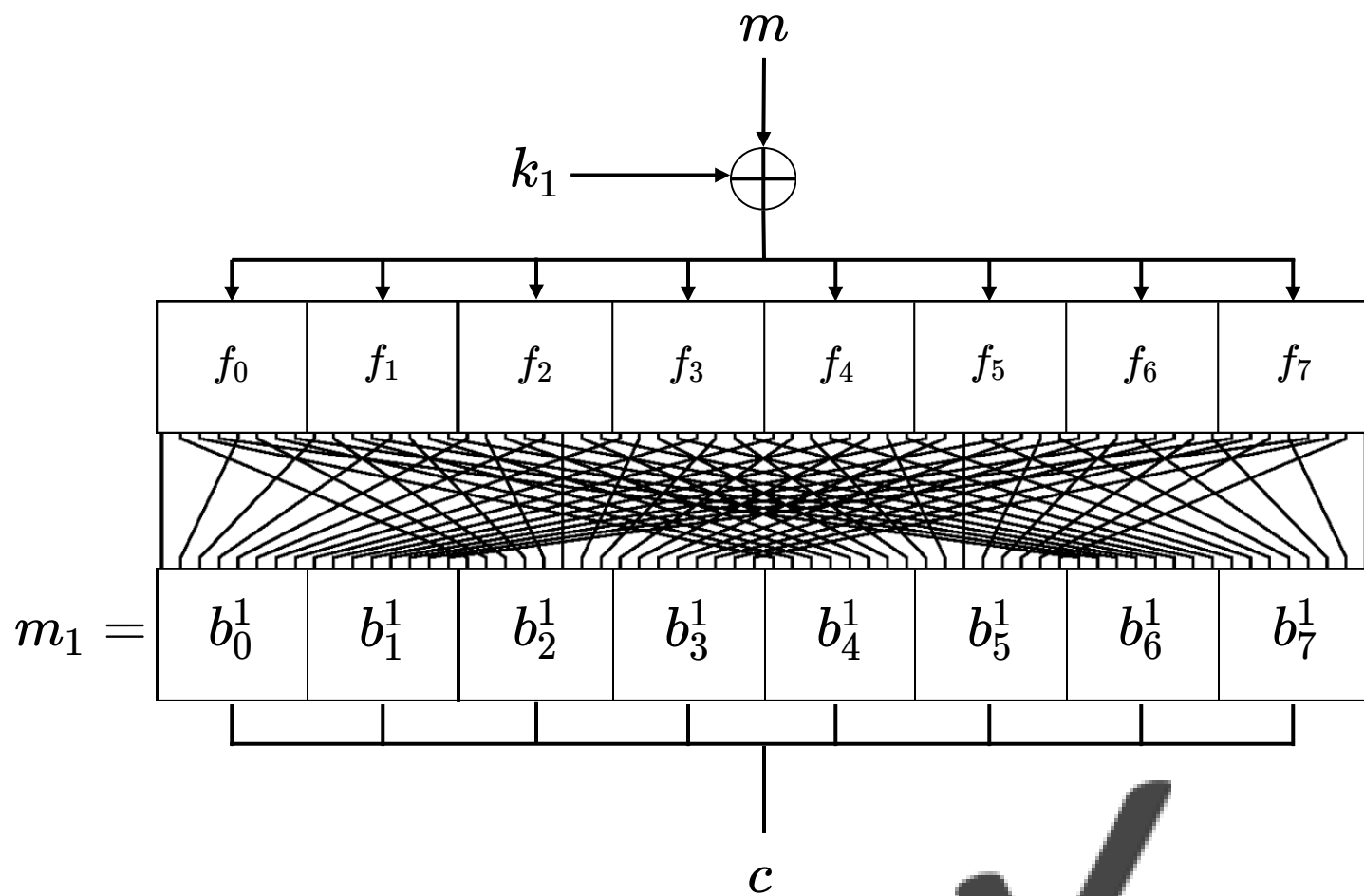
Es un algoritmo determinista que genera una  
llave para cada ronda a partir de la llave original

Red de  
Sustitución / Permutación



Como un esquema criptográfico...

Podemos decriptar?



**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

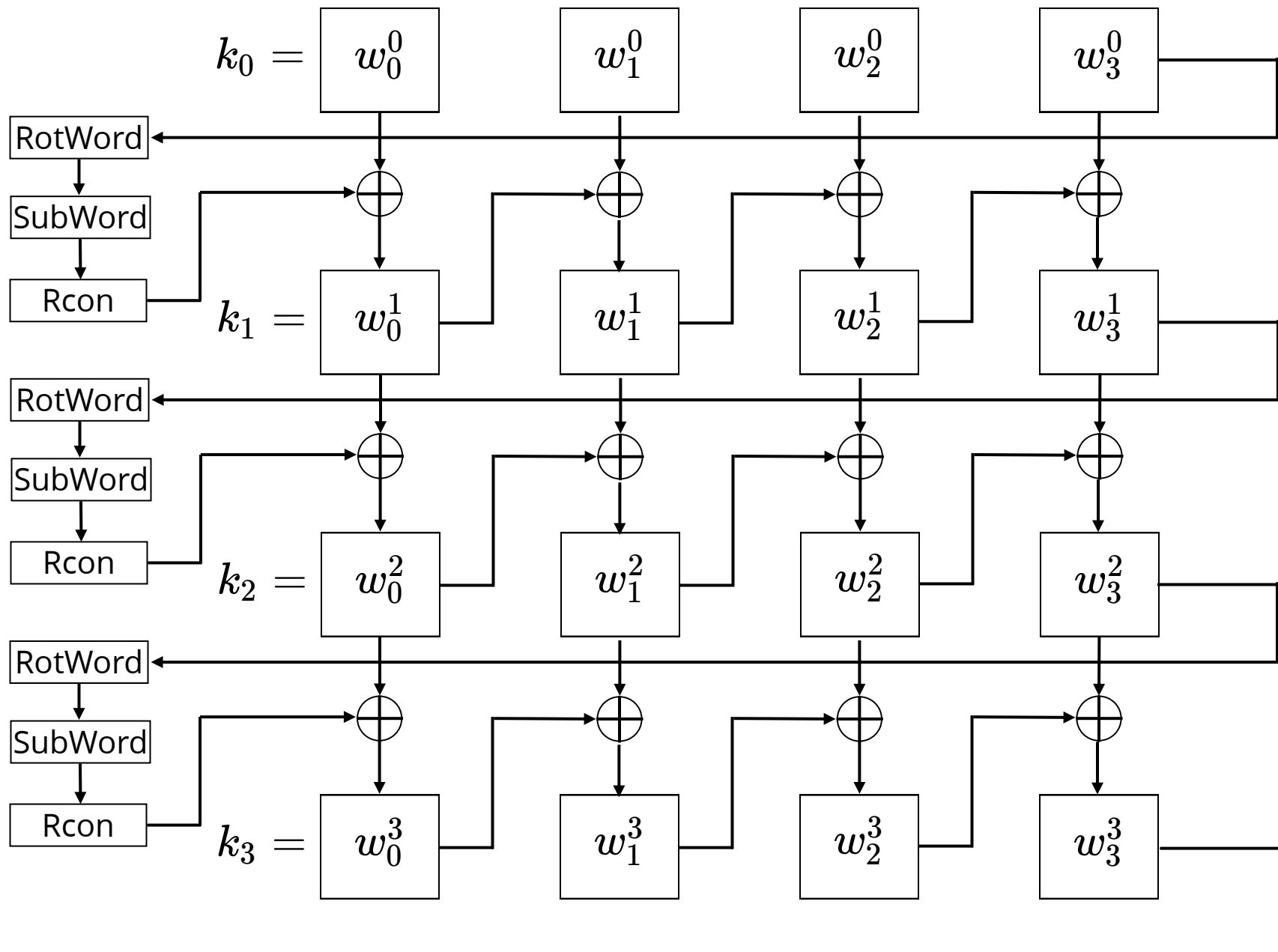
# Key Schedule

Con 10 rondas tenemos que generar 11 sub-llaves

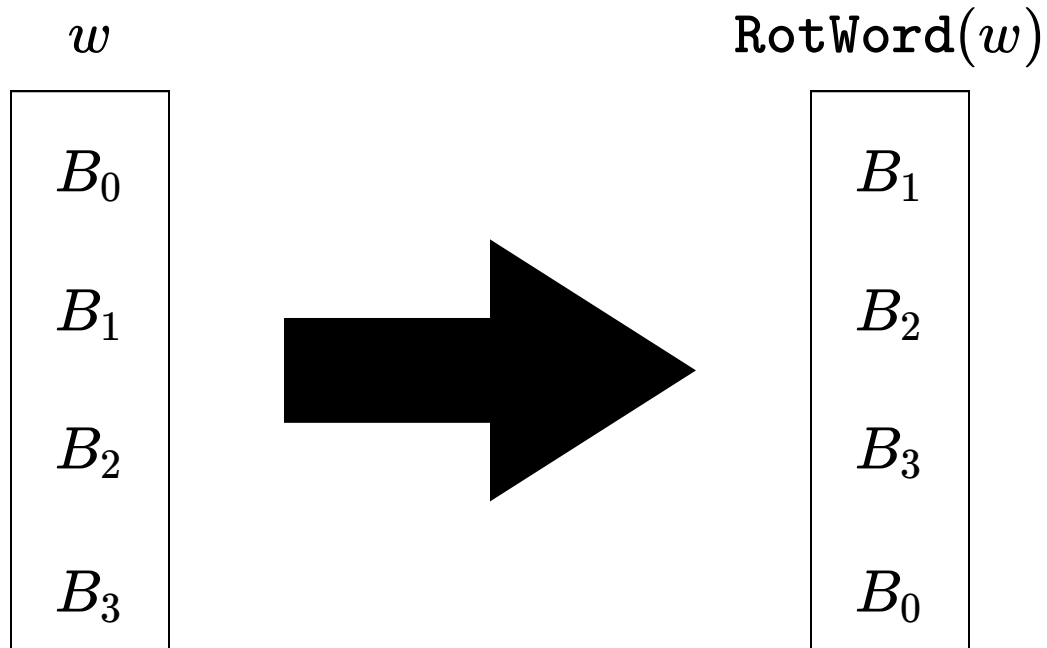
128 bits = 16 bytes

Matriz de 4x4 bytes = 4 columnas de 4 bytes

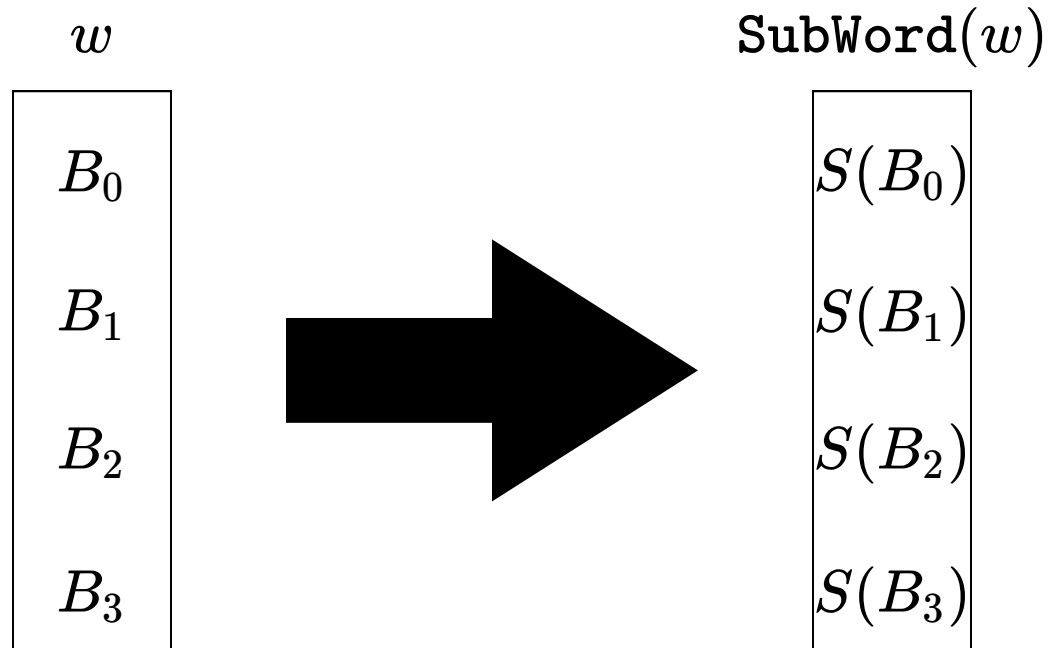
$k_0 = k$	$w_0^0$	$w_1^0$	$w_2^0$	$w_3^0$
	$k_0^0$	$k_4^0$	$k_8^0$	$k_{12}^0$
	$k_1^0$	$k_5^0$	$k_9^0$	$k_{13}^0$
	$k_2^0$	$k_6^0$	$k_{10}^0$	$k_{14}^0$
	$k_3^0$	$k_7^0$	$k_{11}^0$	$k_0^{15}$



# RotWord



# SubWord



# AES S-Box

$S(B)$

$B$

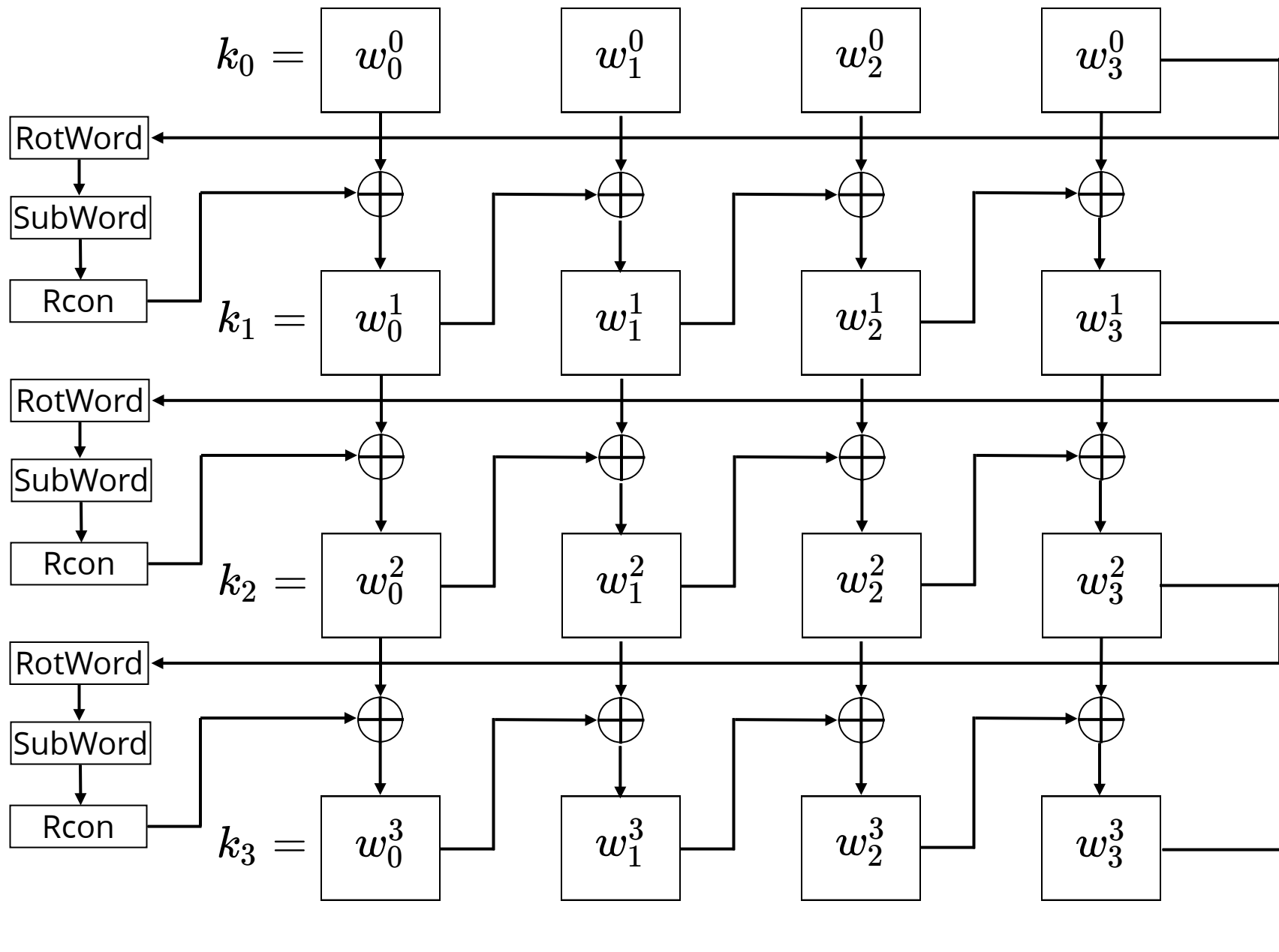
$$\begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \\ s_4 \\ s_5 \\ s_6 \\ s_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\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



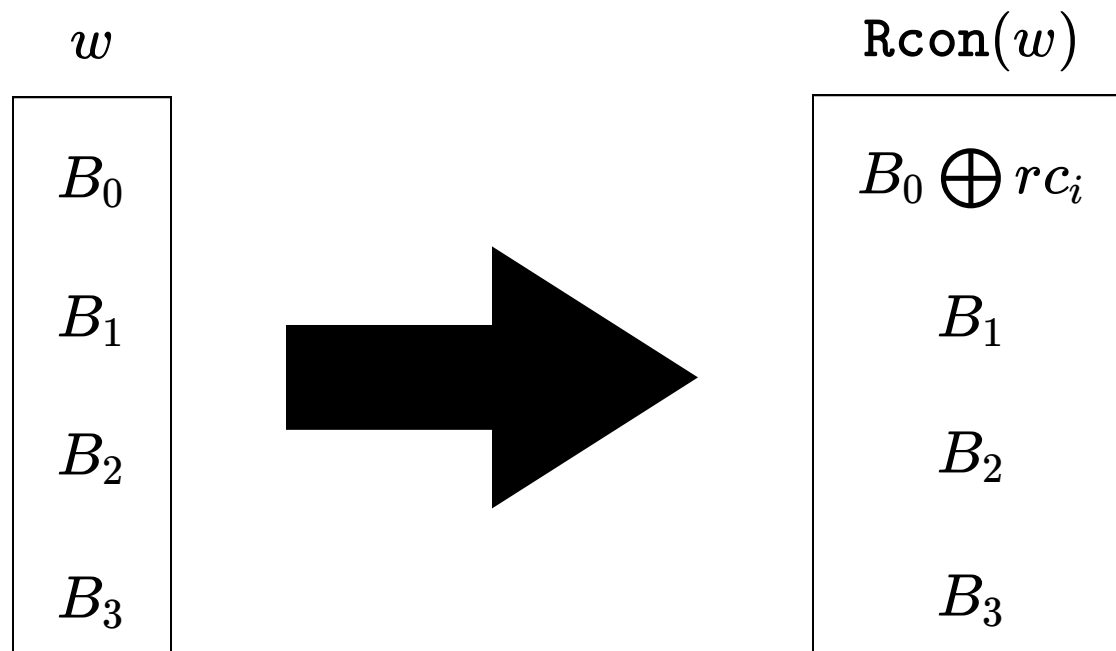
# AES S-Box

	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



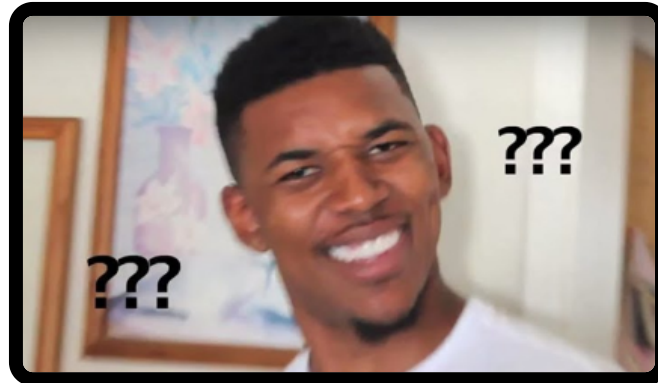
# Rcon



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

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

en módulo 2



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

$$10 \bmod 3 = 1$$

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  $-x^4 - x^3 - x - 1 = x^4 + x^3 + x + 1$

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?

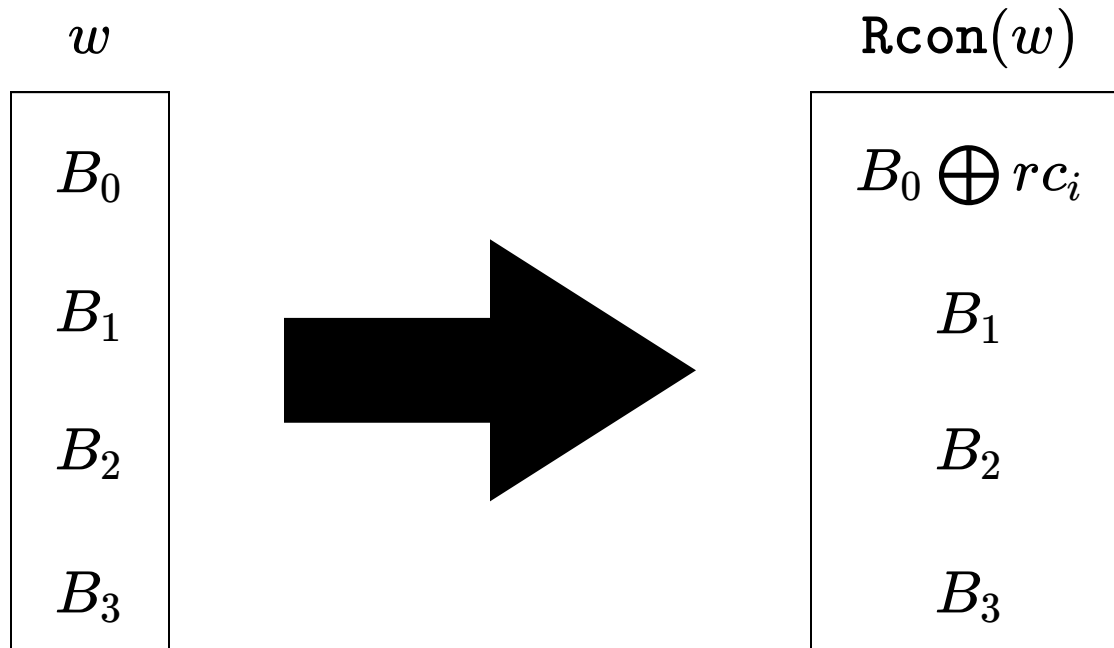
$x^7$ , cuyos coeficientes en binario son

10000000

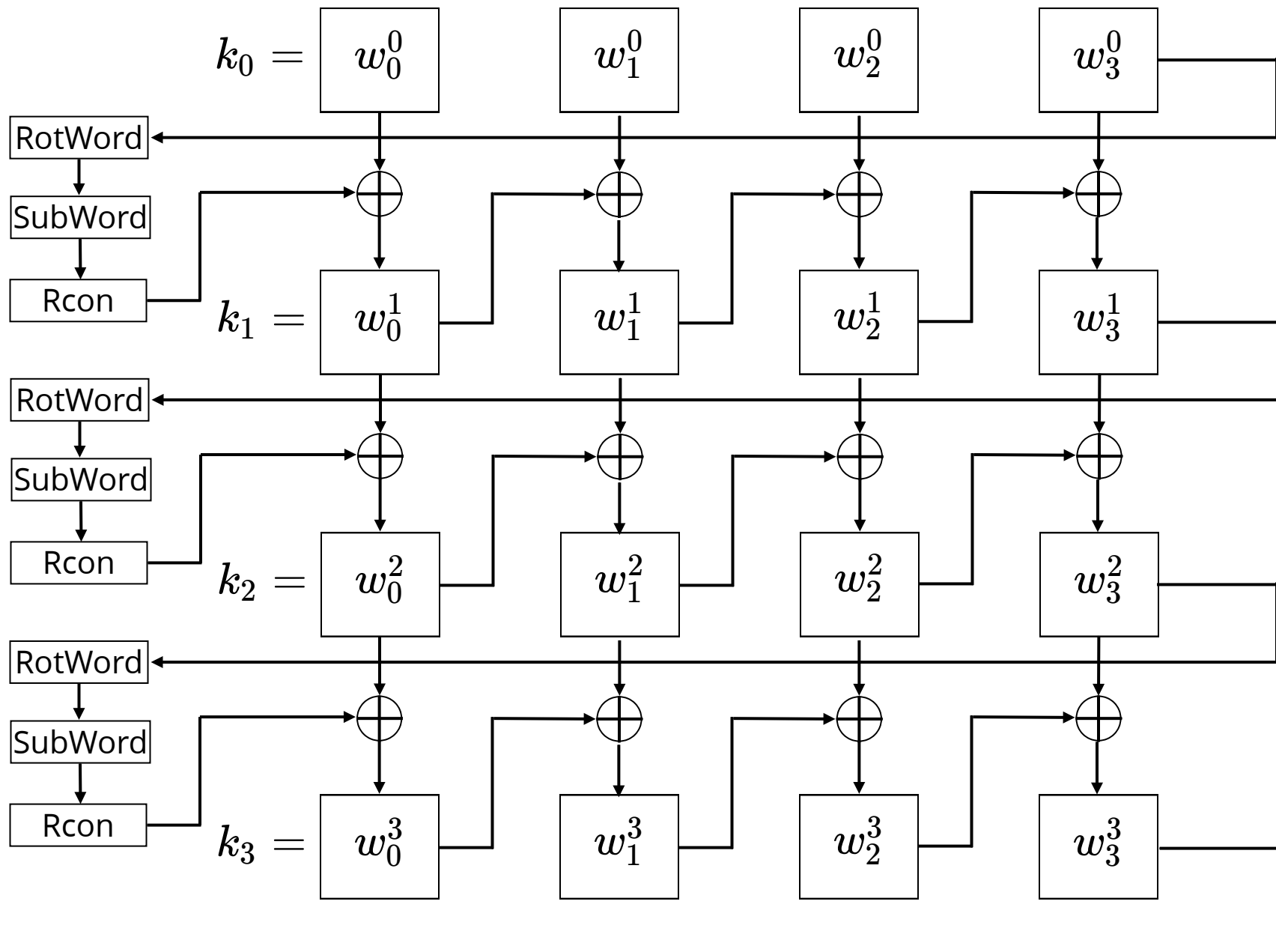
y en hexadecimal

80





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



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

**To be continued...**