



## Advanced Encryption Standard (AES)

Alfredo De Santis  
Sicurezza su Reti

AES

0



## Advanced Encryption Standard (AES)

- ❑ Il *National Institute of Standard and Technology* (NIST) propose il DES come standard nel 1977 ...
- ❑ DES riaffermato nel 1993 fino a Dicembre 1998
- ❑ Critiche al DES:
  - chiave di soli 56 bit
  - criteri costruttivi non chiari (ci sono trapdoor nelle S-box?)
- ❑ Obiettivo del NIST:
  - nuovo cifrario a blocchi per uso commerciale e governativo
  - più sicuro ed efficiente del DES triplo

AES

1



## Processo di Selezione AES

- ❑ 12 Settembre 1997: il NIST indice un concorso pubblico per la nomina dell'AES (deadline 15 giugno 1998)
- ❑ Pubblico scrutinio (<http://www.nist.gov/AES>)
- ❑ Prima conferenza AES, 20-23 agosto 1998 (presentazione di 15 candidature)
- ❑ Pubblico scrutinio
- ❑ Seconda conferenza AES, 22-23 marzo 1999 (presentazione analisi e testing)
- ❑ 9 Agosto 1999: annuncio dei 5 finalisti
- ❑ Pubblico scrutinio
- ❑ Terza conferenza AES, 13-14 aprile 2000 (presentazione analisi e testing)

AES

2



## Processo di Selezione AES

- ❑ 2 ottobre 2000: Scelta del finalista
- ❑ 28 febbraio 2001: Pubblicazione di un Draft di *Federal Information Processing Standard (FIPS)*
- ❑ Pubblico scrutinio di 90 giorni
- ❑ Proposta al *Secretary of Commerce* per approvazione
- ❑ Pubblicato sul *Federal Register*, 6 dic 2001,
  - annuncio approvazione come FIPS 197
  - effettivo a partire dal 26 maggio 2002

AES

3



## Requisiti e Selezione per l'AES

- ❑ Requisiti richiesti dal NIST:
  - Cifrario a blocchi
  - Lunghezza chiave tra 128 e 256 bit
  - Lunghezza testo in chiaro 128 bit (anche 64 e 256 possibilmente)
  - Permette l'implementazione su smart-card
  - Royalty-free
- ❑ Piattaforma del NIST per l'analisi dei candidati:
  - PC IBM-compatible, Pentium Pro 200MHz, 64MB RAM, WINDOWS 95
  - Compilatori Borland C++ 5.0 ed il Java Development Kit (JDK) 1.1
- ❑ Selezione del NIST basata su:
  - Sicurezza
  - Efficienza implementazioni hardware e software
  - Grandezza codice e memoria utilizzata

AES

4



## Documentazione dei Candidati

- ❑ Descrizione algoritmo
- ❑ Analisi algoritmo (vantaggi e limiti)
- ❑ Stima dell'efficienza computazionale
- ❑ Analisi dell'algoritmo rispetto agli attacchi di crittoanalisi più conosciuti (ad esempio known o chosen plaintext)
- ❑ Implementazione di riferimento in ANSI C
- ❑ Implementazione ottimizzata dell'algoritmo implementata in ANSI C e Java

AES

5



## Finalisti e candidati per l'AES

RIJNDAEL	Joan Daemen, Vincent Rijmen	Pronuncia: Reign Dahl, Rain Doll, Rhine Dahl
MARS	IBM	
RC6	RSA Laboratories	
SERPENT	R. Anderson, E. Biham, L. Knudsen	
TWOFISH	B. Schneider, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson	
CAST-256	Entrust Technologies, INC.	
CRYPTON	Future System, INC.	
DEAL	R. Outerbridge, L. Knudsen	
DFC	CNRS	
E2	Nippon Telegraph and Telephone Corp.	
FROG	TecApro Internacional S.A.	
HPC	L. Brown, J. Pieprzyk, J. Seberry	
LOKI97	L. Brown, J. Pieprzyk, J. Seberry	
MAGENTA	Deutsche Telekom AG	
SAFER+	Cylink Corp.	

AES

6



## AES: Rijndael

- ❑ Non è un cifrario di Feistel
  - Lavora in parallelo sull'intero blocco in input
- ❑ E' un cifrario a blocchi iterato
  - Taglia del blocco: 128 bit
  - Lunghezza della chiave: 128, 192, o 256 bit
  - Numero di round: 10, 12 o 14
- ❑ Ogni round (tranne l'ultimo) è una composizione uniforme e parallela di 4 passi
  - SubBytes (sostituzione mediante S-box)
  - ShiftRows (permutazione)
  - MixColumns (sostituzione che usa aritmetica su GF(2<sup>8</sup>))
  - AddRound key (XOR con chiave espansa)

AES

7



## FIPS 197

	Key Length (Nk words)	Block Size (Nb words)	Number of Rounds (Nr)
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

AES

8



## Lunghezza chiavi AES

- ❑ Con 128 bit:  $2^{128} = 3.4 \times 10^{38}$  chiavi possibili
    - Una macchina che prova 2<sup>55</sup> chiavi al secondo impiega 149.000 miliardi di anni per rompere AES
  - ❑ Con 192 bit:  $2^{192} = 6.2 \times 10^{57}$  chiavi possibili
    - ...
  - ❑ Con 256 bit:  $2^{256} = 1.1 \times 10^{77}$  chiavi possibili
    - ...
- Si pensa che AES resterà sicuro per i prossimi 20 anni

AES

9



## Chiave e blocco

- ❑ Chiave di lunghezza variabile (128, 192, 256 bit)
  - rappresentata come una matrice di byte con 4 righe e Nk colonne, Nk = length. chiave / 32
    - chiave 128 bit = 16 byte → Nk = 4
    - chiave 192 bit = 24 byte → Nk = 6
    - chiave 256 bit = 32 byte → Nk = 8

K <sub>0,0</sub>	K <sub>0,1</sub>	K <sub>0,2</sub>	K <sub>0,3</sub>
K <sub>1,0</sub>	K <sub>1,1</sub>	K <sub>1,2</sub>	K <sub>1,3</sub>
K <sub>2,0</sub>	K <sub>2,1</sub>	K <sub>2,2</sub>	K <sub>2,3</sub>
K <sub>3,0</sub>	K <sub>3,1</sub>	K <sub>3,2</sub>	K <sub>3,3</sub>

- ❑ Blocco di lunghezza 128 bit = 16 byte
  - rappresentato come una matrice di byte con 4 righe e Nb colonne, Nb = length. chiave / 32
    - blocco 128 bit = 16 byte → Nb = 4

in <sub>0</sub>	in <sub>4</sub>	in <sub>8</sub>	in <sub>12</sub>
in <sub>1</sub>	in <sub>5</sub>	in <sub>9</sub>	in <sub>13</sub>
in <sub>2</sub>	in <sub>6</sub>	in <sub>10</sub>	in <sub>14</sub>
in <sub>3</sub>	in <sub>7</sub>	in <sub>11</sub>	in <sub>15</sub>

AES

10



## State

- ❑ Operazioni effettuate su una matrice di byte, detta state
  - 4 righe
  - Nb colonne, costituite da word a 32 bit
  - S<sub>r,c</sub> byte in riga r e colonna c
- ❑ Matrice di byte in input copiata nella matrice state

$$S_{r,c} \leftarrow in_{r+4c}$$

- ❑ Al termine, matrice state copiata nella matrice output

$$out_{r+4c} \leftarrow S_{r,c}$$

AES

11



# state

input bytes

state

output bytes

$in_0$	$in_4$	$in_8$	$in_{12}$	$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$	$out_0$	$out_4$	$out_8$	$out_{12}$
$in_1$	$in_5$	$in_9$	$in_{13}$	$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$out_1$	$out_5$	$out_9$	$out_{13}$
$in_2$	$in_6$	$in_{10}$	$in_{14}$	$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$out_2$	$out_6$	$out_{10}$	$out_{14}$
$in_3$	$in_7$	$in_{11}$	$in_{15}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$	$out_3$	$out_7$	$out_{11}$	$out_{15}$

$$S_{r,c} \leftarrow in_{r+4c}$$

$0 \leq r < 4 \quad 0 \leq c < Nb$

$$out_{r+4c} \leftarrow S_{r,c}$$

$0 \leq r < 4 \quad 0 \leq c < Nb$

AES

12



# Prima di descrivere l'AES



Qualche preliminare matematico ...

AES

13



# Preliminari

Il byte è l'unità di base nella computazione dell'AES



- Rappresentazione dei byte
- Operazioni sui byte
  - Addizione e moltiplicazione
- Struttura di  $GF(2^8)$ 
  - Campo finito con 256 elementi

AES

14



# Byte

- I valori dei byte sono rappresentati in notazione esadecimale

- Due cifre esadecimali per ciascun byte

{11010100} → d4

- Ciascun byte è interpretato come un elemento del campo finito  $GF(2^8)$

{ $b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0$ } →  $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$

{11010100} →  $(x^7 + x^6 + x^4 + x^2)$

AES

15



# Addizione su byte

- Addizione in  $GF(2^8)$  corrisponde al polinomio i cui coefficienti sono la somma modulo 2 dei coefficienti dei due polinomi

$$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$$

$$\{01010111\} \oplus \{10000011\} = \{11010100\}$$

$$\{57\} \oplus \{83\} = \{d4\}$$

AES

16



# Moltiplicazione su byte

- moltiplicazione in  $GF(2^8)$  (denotata da  $\bullet$ ) corrisponde alla moltiplicazione di polinomi modulo un polinomio irriducibile di grado 8

- Polinomio irriducibile per AES:

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

unici divisori:  
1 e se stesso

AES

17



# Moltiplicazione su byte

❑ Moltiplicazione in  $GF(2^8)$  (denotata da  $\bullet$ ) corrisponde alla moltiplicazione di polinomi modulo un polinomio irriducibile di grado 8

- Il risultato è un polinomio di grado = 7

❑ Polinomio irriducibile per AES:

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

- E' solo uno dei 30 polinomi irriducibili di grado 8

unici divisori:  
1 e se stesso



# Esempio moltiplicazione

$$\{01010111\} \cdot \{1000011\} = \{11000001\}$$

$$\{57\} \cdot \{83\} = \{c1\}$$

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

$$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \text{ modulo } x^8 + x^4 + x^3 + x + 1 = x^7 + x^6 + 1$$



# Proprietà

❑ Moltiplicazione

- Associativa
- Identità {01}
- Esiste inverso  $a^{-1}(x)$  per ogni  $a(x)$
- $a(x) \bullet (b(x) + c(x)) = a(x) \bullet b(x) + a(x) \bullet c(x)$

❑ Struttura del campo finito  $GF(2^8)$



# Polinomi con coefficienti in $GF(2^8)$

❑ Word  $[a_0, a_1, a_2, a_3] \rightarrow$  polinomio  $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \quad b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

Addizione  $a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$

Moltiplicazione  $c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$

$$\begin{aligned} c_0 &= a_0 \bullet b_0 & c_4 &= a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3 \\ c_1 &= a_1 \bullet b_0 \oplus a_0 \bullet b_1 & c_5 &= a_3 \bullet b_2 \oplus a_2 \bullet b_3 \\ c_2 &= a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 & c_6 &= a_3 \bullet b_3 \\ c_3 &= a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3 \end{aligned}$$

**non va in una word!**



# Polinomi con coefficienti in $GF(2^8)$

❑ Word  $[a_0, a_1, a_2, a_3] \rightarrow$  polinomio  $a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \quad b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

Addizione  $a(x) + b(x) = (a_3 \oplus b_3)x^3 + (a_2 \oplus b_2)x^2 + (a_1 \oplus b_1)x + (a_0 \oplus b_0)$

Moltiplicazione  $c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$

$$\begin{aligned} c_0 &= a_0 \bullet b_0 & c_4 &= a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3 \\ c_1 &= a_1 \bullet b_0 \oplus a_0 \bullet b_1 & c_5 &= a_3 \bullet b_2 \oplus a_2 \bullet b_3 \\ c_2 &= a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 & c_6 &= a_3 \bullet b_3 \\ c_3 &= a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3 \end{aligned}$$

**modulo  $x^4 + 1$**



# Polinomi con coefficienti in $GF(2^8)$

$$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0 \quad b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$$

Moltiplicazione mod  $x^4 + 1$   $d(x) = d_3x^3 + d_2x^2 + d_1x + d_0$

$$\begin{cases} d_0 = (a_0 \bullet b_0) \oplus (a_3 \bullet b_1) \oplus (a_2 \bullet b_2) \oplus (a_1 \bullet b_3) \\ d_1 = (a_1 \bullet b_0) \oplus (a_0 \bullet b_1) \oplus (a_3 \bullet b_2) \oplus (a_2 \bullet b_3) \\ d_2 = (a_2 \bullet b_0) \oplus (a_1 \bullet b_1) \oplus (a_0 \bullet b_2) \oplus (a_3 \bullet b_3) \\ d_3 = (a_3 \bullet b_0) \oplus (a_2 \bullet b_1) \oplus (a_1 \bullet b_2) \oplus (a_0 \bullet b_3) \end{cases}$$

**$x^i \text{ mod } (x^4 + 1) = x^{i \text{ mod } 4}$**

$$\text{cioè} \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$



## Polinomi con coefficienti in GF(2<sup>8</sup>)

- $x^{4+1}$  non è irriducibile su GF(2<sup>8</sup>)
- Non tutti i polinomi hanno inverso mod  $x^{4+1}$
- AES usa  $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$   
 $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$



## Pseudocodice per l'AES

Cipher (byte in[4·Nb], byte out[4·Nb], word w[Nb·(Nr + 1)])

byte state[4,Nb]

state ← in

AddRoundKey (state, w)

for round = 1 to Nr - 1

SubBytes (state)

ShiftRows (state)

MixColumns (state)

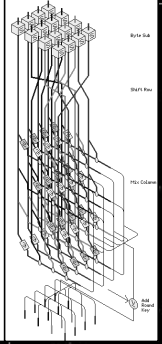
AddRoundKey (state, w + round · Nb)

SubBytes (state)

ShiftRows (state)

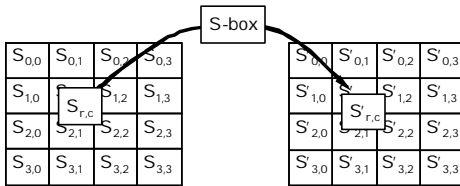
AddRoundKey (state, w + Nr · Nb)

out ← state



## SubBytes(state)

$$S'_{r,c} \leftarrow \text{S-box}(S_{r,c}) \quad 0 \leq r < 4 \quad 0 \leq c < Nb$$



## S-box

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7e	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	ad	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	3e	5e	0b	2b
	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

Esempio: {53} → {ed}



## Costruzione S-box

- Prendere l'inverso moltiplicativo in GF(2<sup>8</sup>)  
{00} resta {00}
- Trasformazione affine in GF(2<sup>8</sup>)

$$b'_i \leftarrow b_i \oplus b_{(i+4) \bmod 8} \oplus b_{(i+5) \bmod 8} \oplus b_{(i+6) \bmod 8} \oplus b_{(i+7) \bmod 8} \oplus \{01100011\}$$

cioè

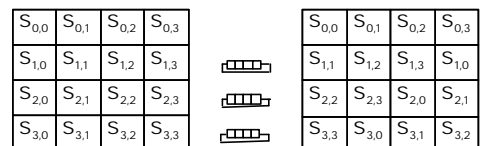
$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$



## ShiftRows(state)

$$S'_{r,c} \leftarrow S_{r,(c+\text{shift}(r,Nb)) \bmod Nb} \quad 0 \leq r < 4 \quad 0 \leq c < Nb$$

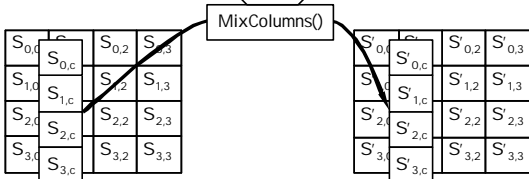
shift(1,4)=1    shift(2,4)=2    shift(3,4)=3



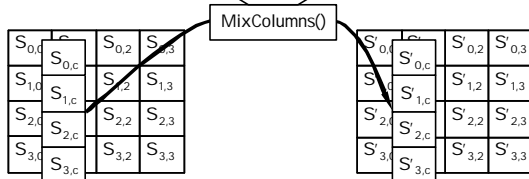
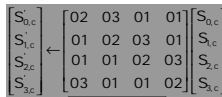


# MixColumns(state)

moltiplicazione per  $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$  modulo  $x^4+1$



# MixColumns(state)



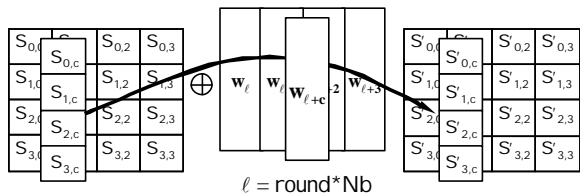
Bytes nelle colonne combinati linearmente



# AddRoundKey()

Round key di Nb words

$$[S'_{0,c}, S'_{1,c}, S'_{2,c}, S'_{3,c}] \leftarrow [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [w_{\text{round} \cdot \text{Nb} + c}] \quad 0 \leq c < \text{Nb}$$



# Espansione chiave

□ A partire dalla chiave iniziale di  $4 \cdot \text{Nk}$  byte, genera un array di  $\text{Nb} \cdot (\text{Nr} + 1)$  word (chiavi schedulate)

- Nb word per AddRoundKey iniziale
- Nb word per AddRoundKey in ciascun round

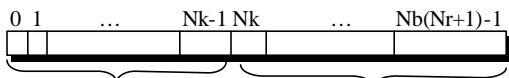
□ Utilizza due routine e un array di word costanti

- SubWord()
  - Applica la S-box a ciascun byte della word in input
- RotWord()
  - Applica uno shift ciclico a sinistra sulla word in input
- Rcon[i]
  - $\text{Rcon}[i] = [x^{-i}, \{00\}, \{00\}, \{00\}]$
  - $\text{Rcon}[i] = 2 \bullet \text{Rcon}[i-1]$
  - $\text{Rcon}[1] = \{01\}$



# Espansione chiave

Chiave schedulata word  $w[\text{Nb}(\text{Nr}+1)]$



Chiave byte  $\text{key}[4 \cdot \text{Nk}]$

$w[i] \leftarrow w[i-1] \text{ xor } w[i-\text{Nk}]$   
Eccetto per  $i$  multiplo di  $\text{Nk}$   
e per  $\text{Nk} = 8$  and  $i \text{ mod } \text{Nk} = 4$



# Espansione chiave

**KeyExpansion** (byte  $\text{key}[4 \cdot \text{Nk}]$ , word  $w[\text{Nb} \cdot (\text{Nr} + 1)]$ ,  $\text{Nk}$ )

```

i ← 0
while (i < Nk)
  w[i] ← word[key[4*i], key[4*i+1], key[4*i+2], key[4*i+3]]
  i ← i + 1
i ← Nk
while (i < Nb * (Nr + 1))
  word temp ← w[i - 1]
  if (i mod Nk = 0)
    temp ← SubWord(RotWord(temp)) xor Rcon[i/Nk]
  else if (Nk = 8 and i mod Nk = 4)
    temp ← SubWord(temp)
  w[i] ← w[i - Nk] xor temp
  i ← i + 1

```

**SubWord**(word w)  
[a,b,c,d] ← w  
Output [S-box(a), S-box(b), S-box(c), S-box(d)]

$\text{Rcon}[i] = [x^{-i}, \{00\}, \{00\}, \{00\}]$

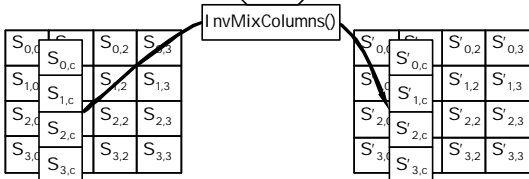
**RotWord**(word w)  
[a,b,c,d] ← w  
Output [b,c,d,a]





# InvMixColumns(state)

moltiplicazione per  
 $a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$   
 modulo  $x^4+1$



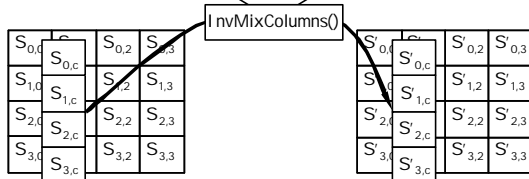
AES

42



# InvMixColumns(state)

$S_{0,c} \leftarrow \{0e\} S_{0,c} + \{0b\} S_{1,c} + \{0d\} S_{2,c} + \{09\} S_{3,c}$   
 $S_{1,c} \leftarrow \{09\} S_{0,c} + \{0e\} S_{1,c} + \{0b\} S_{2,c} + \{0d\} S_{3,c}$   
 $S_{2,c} \leftarrow \{0d\} S_{0,c} + \{0b\} S_{1,c} + \{0e\} S_{2,c} + \{09\} S_{3,c}$   
 $S_{3,c} \leftarrow \{0b\} S_{0,c} + \{0d\} S_{1,c} + \{09\} S_{2,c} + \{0e\} S_{3,c}$



AES

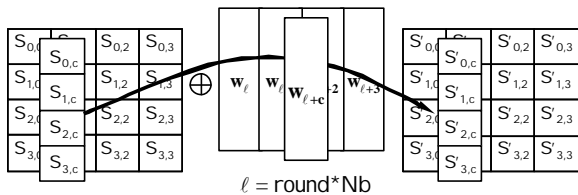
43



# AddRoundKey()

È l'inversa di se stessa!

$$[S'_{0,c}, S'_{1,c}, S'_{2,c}, S'_{3,c}] \leftarrow [S_{0,c}, S_{1,c}, S_{2,c}, S_{3,c}] \oplus [W_{\text{round}+N\text{b}+c}] \quad 0 \leq c < N\text{b}$$



AES

44



# Implementazioni

B. R. Gladman

Pentium Pro 200MHz

C, C++

Cicli  
Mbits/second  
M = 10<sup>6</sup> non 2<sup>20</sup>

Tables used (last round)	0	0	1	0	0	1	1	4	4
Loop/unrolled	a	a	a	a	a	a	a	a	a
code size (bytes)	9490	5002	5666	4402	10194	4402	10142	4146	9934
table size (bytes)	576	2680	4728	8824	8824	10872	10872	17016	17016
128 bit key	key (encrypt)	381	394	371	394	391	374	369	369
	key (decrypt)	381	1623	1618	1623	1623	1620	1614	1601
	encrypt (mb/s)	2340	730	686	424	419	396	375	386
	decrypt (mb/s)	2126	706	631	441	401	396	369	392
192 bit key	key (encrypt)	819	350	373	403	410	446	462	463
	key (decrypt)	120	363	380	377	468	446	463	463
	encrypt (mb/s)	990	1846	1833	1854	1857	1836	1836	1802
	decrypt (mb/s)	890	1853	1838	1845	1859	1834	1838	1800
256 bit key	key (encrypt)	381	394	371	394	391	374	369	369
	key (decrypt)	381	1623	1618	1623	1623	1620	1614	1601
	encrypt (mb/s)	2340	730	686	424	419	396	375	386
	decrypt (mb/s)	2126	706	631	441	401	396	369	392



# Implementazioni

B. R. Gladman

Pentium Pro 200MHz

C, C++

Cicli  
Mbits/second  
M = 10<sup>6</sup> non 2<sup>20</sup>

Tables used (last round)	0	0	1	0	0	1	1	4	4
Loop/unrolled	a	a	a	a	a	a	a	a	a
code size (bytes)	9490	5002	5666	4114	15410	6144	15426	5874	15122
table size (bytes)	576	2680	4728	8824	8824	10872	10872	17016	17016
128 bit key	key (encrypt)	381	634	665	620	670	670	676	670
	key (decrypt)	381	2892	2880	2879	2895	2894	2887	2907
	encrypt (mb/s)	2340	1227	1250	749	755	690	674	670
	decrypt (mb/s)	2126	1257	1190	726	721	664	651	655
192 bit key	key (encrypt)	819	192	204	364	339	374	374	379
	key (decrypt)	120	303	315	362	365	385	393	393
	encrypt (mb/s)	990	2704	2641	2693	2699	2667	2674	2650
	decrypt (mb/s)	890	2706	2643	2717	2698	2652	2640	2676
256 bit key	key (encrypt)	381	634	665	620	670	670	676	670
	key (decrypt)	381	2892	2880	2879	2895	2894	2887	2907
	encrypt (mb/s)	2340	1227	1250	749	755	690	674	670
	decrypt (mb/s)	2126	1257	1190	726	721	664	651	655



# Implementazioni

B. R. Gladman

Pentium Pro 200MHz

C, C++

Cicli  
Mbits/second  
M = 10<sup>6</sup> non 2<sup>20</sup>

Tables used (last round)	0	0	1	0	0	1	1	4	4
Loop/unrolled	a	a	a	a	a	a	a	a	a
code size (bytes)	1491	1075	1060	792	2042	792	2042	740	2098
table size (bytes)	576	2680	4728	8824	8824	10872	10872	17016	17016
128 bit key	key (encrypt)	261	436	395	361	361	392	392	392
	key (decrypt)	261	4316	4312	4316	4316	4308	4308	4307
	encrypt (mb/s)	2340	2096	2060	1191	1191	1191	1191	1191
	decrypt (mb/s)	2126	1989	1922	1191	1191	1191	1191	1191
192 bit key	key (encrypt)	361	1623	1623	1623	1623	1623	1623	1623
	key (decrypt)	361	4216	4216	4216	4216	4216	4216	4216
	encrypt (mb/s)	2340	2096	2060	1191	1191	1191	1191	1191
	decrypt (mb/s)	2126	1989	1922	1191	1191	1191	1191	1191
256 bit key	key (encrypt)	261	436	395	361	361	392	392	392
	key (decrypt)	261	4316	4312	4316	4316	4308	4308	4307
	encrypt (mb/s)	2340	2096	2060	1191	1191	1191	1191	1191
	decrypt (mb/s)	2126	1989	1922	1191	1191	1191	1191	1191