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?)
- Attuale obiettivo del NIST:
  - nuovo cifrario a blocchi per uso commerciale e governativo
  - più sicuro ed efficiente del DES triplo

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)

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 (commenti entro il 29 maggio 2001)
- Eventuale revisione
- Proposta al *Secretary of Commerce* per approvazione come FIPS

Previsione approvazione **estate 2001**

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
  - Royality-free
- Piattaforma del NIST per l'analisi dei candidati:
  PC IBM-compatibile, 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

Documentazione dei Candidati

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

Finalisti e candidati per l'AES

RIJNDAEL	Joan Daemen, Vincent Rijmen
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.

Pronuncia: Reign Dahl, Rain Doll, Rhine Dahl

Prima di descrivere l'AES

Qualche preliminare matematico …

Operazioni su byte

- Addizione e Moltiplicazione definite sui 256 valori di un byte
- Struttura del campo finito GF(2^8)

Addizione

- XOR corrisponde ad addizione di polinomi

{01010111} ⊕ {10000011} = {11010100}

{57} ⊕ {83} = {d4}

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

Moltiplicazione su byte

- moltiplicazione in GF(2^8) (denotata da •) 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

Esempio moltiplicazione

{01010111} • {10000011} = {11000001}

{57} • {83} = {c1}

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

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

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

$$= x^7 + x^6 + 1$$

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

$c_0=a_0\bullet b_0$ $\quad c_4=a_3\bullet b_1\oplus a_2\bullet b_2\oplus a_1\bullet b_3$

$c_1=a_1\bullet b_0\oplus a_0\bullet b_1$ $\quad c_5=a_3\bullet b_2\oplus a_2\bullet b_3$

$c_2=a_2\bullet b_0\oplus a_1\bullet b_1\oplus a_0\bullet b_2$ $\quad c_6=a_3\bullet b_3$

$c_3=a_3\bullet b_0\oplus a_2\bullet b_1\oplus a_1\bullet b_2\oplus a_0\bullet b_3$

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

$c_0=a_0\bullet b_0$ $\quad c_4=a_3\bullet b_1\oplus a_2\bullet b_2\oplus a_1\bullet b_3$

$c_1=a_1\bullet b_0\oplus a_0\bullet b_1$ $\quad c_5=a_3\bullet b_2\oplus a_2\bullet b_3$

$c_2=a_2\bullet b_0\oplus a_1\bullet b_1\oplus a_0\bullet b_2$ $\quad c_6=a_3\bullet b_3$

$c_3=a_3\bullet b_0\oplus a_2\bullet b_1\oplus a_1\bullet b_2\oplus a_0\bullet b_3$

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

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(2^8)

- x^4+1 non è irriducibile su GF(2^8)
- Non tutti i polinomi hanno inverso mod x^4+1
- AES usa $a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

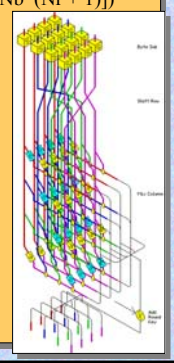
$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$

AES: Draft del FIPS

	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

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



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}

$S_{r,c} \leftarrow in_{r+4c}$ $0 \le r < 4$ $0 \le c < Nb$

$out_{r+4c} \leftarrow S_{r,c}$ $0 \le r < 4$ $0 \le c < Nb$

SubBytes(state)

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

S-box

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$		$S'_{0,0}$	$S'_{0,1}$	$S'_{0,2}$	$S'_{0,3}$
$S_{1,0}$			$S_{1,3}$		$S'_{1,0}$		$S'_{1,2}$	$S'_{1,3}$
$S_{2,0}$		$S_{2,2}$	$S_{2,3}$		$S'_{2,0}$		$S'_{2,2}$	$S'_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$		$S'_{3,0}$	$S'_{3,1}$	$S'_{3,2}$	$S'_{3,3}$

S-box

x \ y	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

Costruzione S-box

- Prendere l'inverso moltiplicativo in GF(2^8)
  - {00} resta {00}
- Trasformazione affine in GF(2^8)

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

cioè

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{bmatrix} \leftarrow \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

ShiftRows(state)

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

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

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$

MixColumns(state)

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

MixColumns()

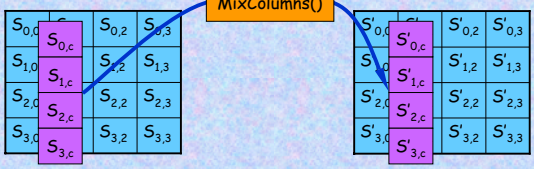
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

$S'_{0,0}$	$S'_{0,1}$	$S'_{0,2}$	$S'_{0,3}$
$S'_{1,0}$	$S'_{1,1}$	$S'_{1,2}$	$S'_{1,3}$
$S'_{2,0}$	$S'_{2,1}$	$S'_{2,2}$	$S'_{2,3}$
$S'_{3,0}$	$S'_{3,1}$	$S'_{3,2}$	$S'_{3,3}$

MixColumns(state)

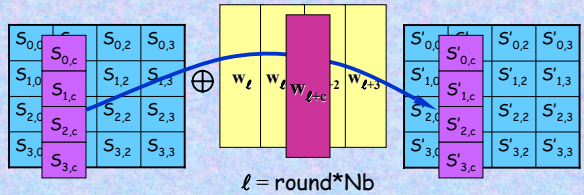
$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} \leftarrow \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

MixColumns()



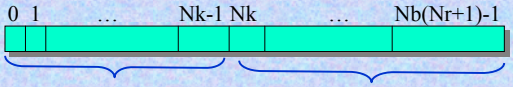
AddRoundKey()

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



Espansione chiave

Chiave schedulata word w[Nb(Nr+1)]



Chiave byte key[4*Nk]

w[i]←w[i-1] xor w[i-Nk]

Eccetto per i multiplo di Nk
e per Nk = 8 and i mod Nk = 4

Espansione chiave

KeyExpansion (byte key[4 * Nk], word w[Nb * (Nr+1)], 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)]

Rcon[i]=[x^{i-1},{00},{00},{00}]

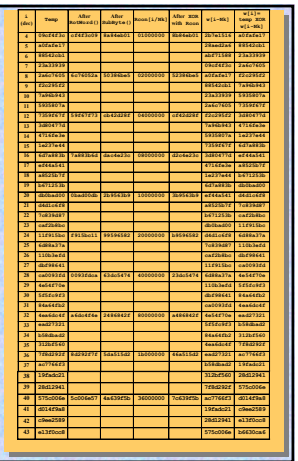
RotWord (word w)
[a,b,c,d] ← w
Output [b,c,d,a]

Espansione chiave Esempio

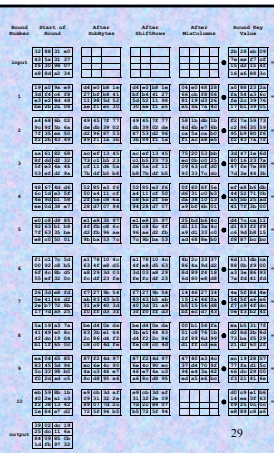
Key
2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

$Nk = 4$

w_0 = 2b7e1516 w_1 = 28aed2a6
w_2 = abf71588 w_3 = 09cf4f3c



Cifratura Esempio



Input
32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34

Key
2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

Pseudocodice decifrazione AES

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
byte  state[4,Nb]
state ← in
AddRoundKey(state, w + Nr * Nb)
for round = Nr - 1 step -1 to 1
        InvShiftRows(state)
        InvSubBytes(state)
        AddRoundKey(state, w + round * Nb)
        InvMixColumns(state)
InvShiftRows(state)
InvSubBytes(state)
AddRoundKey(state, w)
out ← state
```

InvShiftRows(state)

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

$shift(1,4)=1 \quad shift(2,4)=2 \quad shift(3,4)=3$

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$

$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,0}$
$S_{2,2}$	$S_{2,3}$	$S_{2,0}$	$S_{2,1}$
$S_{3,3}$	$S_{3,0}$	$S_{3,1}$	$S_{3,2}$

InvSubBytes(state)

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

Inverse_S-box

Inverse_S-box

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

InvMixColumns(state)

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

InvMixColumns()

InvMixColumns(state)

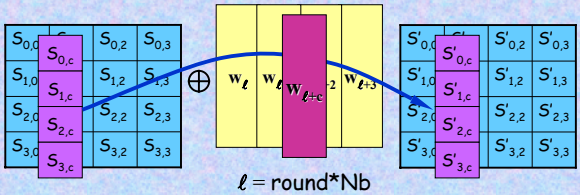
$$\begin{bmatrix} S'_{0,c} \\ S'_{1,c} \\ S'_{2,c} \\ S'_{3,c} \end{bmatrix} \leftarrow \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} S_{0,c} \\ S_{1,c} \\ S_{2,c} \\ S_{3,c} \end{bmatrix}$$

InvMixColumns()

AddRoundKey()

È l'inversa di se stessa!

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



$\ell = \text{round}*Nb$

Implementazioni

B. R. Gladman

Pentium Pro 200MHz

C, C++

Cicli
Mbits/second
M = 10^6 non 2^{20}

Tables used (last round)			0	0	1	0	0	1	1	4	4
Loops unrolled			n	n	n	n	y	n	y	n	y
128 bit block (fixed)	code size (bytes)		9490	5602	5666	4402	10194	4402	10242	4146	9954
	table size (bytes)		576	2680	4728	8824	8824	10872	10872	17016	17016
	128 bit key	key (encrypt)	381	394	371	394	391	374	374	369	369
		key (decrypt)	381	1623	1618	1623	1623	1620	1614	1601	1581
		encrypt	2340	730	686	424	419	396	375	386	362
		decrypt	2126	705	673	443	401	396	369	392	366
		encrypt (mhz)	10.9	35.0	37.3	60.3	61.0	64.6	68.2	66.3	70.7
		decrypt (mhz)	12.0	36.3	38.0	57.7	63.8	64.6	69.3	65.3	69.9
	192 bit key	key (encrypt)	390	1846	1833	1854	1857	1836	1836	1802	1785
		key (decrypt)	390	1853	1838	1845	1859	1834	1838	1800	1791
		encrypt	2839	869	817	493	498	461	439	466	433
		decrypt	2568	843	802	515	474	466	441	475	429
		encrypt (mhz)	9.0	29.4	31.3	51.9	51.4	55.5	58.3	54.9	59.1
		decrypt (mhz)	9.9	30.3	31.9	49.7	54.0	54.9	58.0	53.8	59.6
	256 bit key	key (encrypt)	580	2291	2239	2291	2291	2233	2245	2098	2108
		key (decrypt)	580	2291	2259	2285	2283	2249	2261	2162	2193
		encrypt	3351	1027	961	570	549	549	509	556	493
		decrypt	3027	991	945	590	549	547	507	554	496
		encrypt (mhz)	7.6	24.9	26.6	44.9	46.6	46.6	50.2	46.0	51.9
		decrypt (mhz)	8.4	25.8	27.0	43.3	46.6	46.8	50.4	46.2	51.6

Implementazioni

B. R. Gladman

Pentium Pro 200MHz

C, C++

Cicli
Mbits/second
M = 10^6 non 2^{20}

Tables used (last round)			0	0	1	0	0	1	1	4	4
Loops unrolled			n	n	n	n	y	n	y	n	y
192 bit block (fixed)	code size (bytes)		9490	8050	8098	6114	15410	6146	15426	5874	15122
	table size (bytes)		576	2680	4728	8824	8824	10872	10872	17016	17016
	128 bit key	key (encrypt)	381	634	665	630	630	670	670	676	670
		key (decrypt)	381	2892	2880	2879	2895	2894	2887	2907	2891
		encrypt	2340	1327	1250	749	755	680	675	674	670
		decrypt	2126	1257	1190	726	721	664	651	655	651
		encrypt (mhz)	10.9	19.2	20.4	34.1	33.9	37.6	37.9	37.9	38.2
		decrypt (mhz)	12.0	20.3	21.5	35.2	35.5	38.5	39.3	39.0	39.3
	192 bit key	key (encrypt)	390	2704	2643	2693	2699	2667	2674	2650	2659
		key (decrypt)	390	2706	2645	2717	2698	2652	2640	2676	2673
		encrypt	2839	1335	1250	747	756	678	676	673	666
		decrypt	2568	1245	1181	730	721	661	654	655	648
		encrypt (mhz)	9.0	19.1	20.4	34.2	33.8	37.7	37.8	38.0	38.4
		decrypt (mhz)	9.9	20.5	21.6	35.0	35.5	38.7	39.1	39.0	39.5
	256 bit key	key (encrypt)	580	3415	3356	3421	3425	3350	3350	3260	3251
		key (decrypt)	580	3414	3339	3413	3414	3353	3357	3237	3258
		encrypt	3351	1555	1487	884	871	814	787	798	781
		decrypt	3027	1472	1403	851	821	786	766	768	743
		encrypt (mhz)	7.6	16.4	17.2	28.9	29.3	31.4	32.5	32.0	32.7
		decrypt (mhz)	8.4	17.3	18.2	30.0	31.1	32.5	33.4	33.3	34.4

Implementazioni

B. R. Gladman

Pentium Pro 200MHz

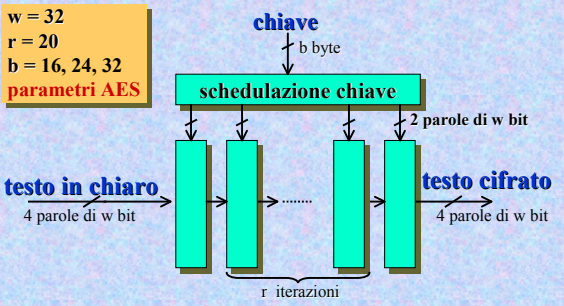
C, C++

Cicli
Mbits/second
M = 10^6 non 2^{20}

Tables used (last round)			0	0	1	0	0	1	1	4	4
Loops unrolled			n	n	n	n	y	n	y	n	y
256 bit block (fixed)	code size (bytes)		9490	10578	10610	7906	20482	7970	20434	7650	20098
	table size (bytes)		576	2680	4728	8824	8824	10872	10872	17016	17016
	128 bit key	key (encrypt)	381	930	1001	945	945	1002	998	1020	1010
		key (decrypt)	381	4519	4521	4516	4519	4508	4488	4550	4507
		encrypt	2340	2096	2010	1167	1161	1101	1058	1095	1053
		decrypt	2126	1989	1922	1147	1119	1076	1031	1070	1016
		encrypt (mhz)	10.9	12.2	12.7	21.9	22.0	23.2	24.1	23.3	24.3
		decrypt (mhz)	12.0	12.8	13.3	22.3	22.8	23.7	24.8	23.9	25.1
	192 bit key	key (encrypt)	390	4232	4138	4213	4238	4152	4155	4129	4133
		key (decrypt)	390	4268	4132	4266	4269	4138	4150	4199	4177
		encrypt	2839	2105	2007	1168	1152	1100	1071	1087	1055
		decrypt	2568	1995	1915	1157	1121	1081	1025	1067	1008
		encrypt (mhz)	9.0	12.1	12.7	21.9	22.2	23.2	23.9	23.5	24.2
		decrypt (mhz)	9.9	12.8	13.3	22.1	22.8	23.6	24.9	23.9	25.3
	256 bit key	key (encrypt)	580	4524	4407	4520	4522	4407	4407	4236	4231
		key (decrypt)	580	4536	4408	4540	4540	4408	4413	4276	4295
		encrypt	3351	2104	2009	1168	1157	1102	1064	1094	1058
		decrypt	3027	1984	1905	1158	1118	1080	1030	1069	1007
		encrypt (mhz)	7.6	12.1	12.7	21.9	22.1	23.2	24.0	23.4	24.1
		decrypt (mhz)	8.4	12.9	13.4	22.1	22.8	23.7	24.8	23.9	25.4

RC6-w/r/b

w = 32
r = 20
b = 16, 24, 32
parametri AES



RC6

Operazioni su parole di w bit:

- a+b somma modulo 2^w
- a-b sottrazione modulo 2^w
- a⊕b XOR bit a bit
- a · b moltiplicazione modulo 2^w
- a « b shift a sinistra di a di un numero di bit dato dai log w bit meno significativi di b
- a » b shift a destra di a di un numero di bit dato dai log w bit meno significativi di b

RC6: cifratura

Input: testo in chiaro (A,B,C,D)
Chiave schedulata: S[0,…,2r+3]

```
B ← B + S[0]
D ← D + S[1]
for i ← 1 to r do
      t ← (B • (2B+1)) « log w
      u ← (D • (2D+1)) « log w
      A ← ((A⊕ t) « u) + S[2i]
      C ← ((C ⊕ u) « t) + S[2i+1]
      (A,B,C,D) ← (B,C,D,A)
A ← A + S[2r+2]
C ← C + S[2r+3]
```

Output: testo cifrato (A,B,C,D)

RC6: decifratura

```
B ← B + S[0]
D ← D + S[1]
for i = 1 to r do
      t ← (B • (2B+1)) « log w
      u ← (D • (2D+1)) « log w
      A ← ((A⊕ t) « u) + S[2i]
      C ← ((C ⊕ u) « t) + S[2i+1]
      (A,B,C,D) ← (B,C,D,A)
A ← A + S[2r+2]
C ← C + S[2r+3]
```

cifratura

```
C ← C - S[2r+3]
A ← A - S[2r+2]
for i = r downto 1 do
      (A,B,C,D) ← (B,C,D,A)
      u ← (D • (2D+1)) « log w
      t ← (B • (2B+1)) « log w
      C ← ((C-S[2i+1]) » t) ⊕ u
      A ← ((A-S[2i]) » u) ⊕ t
D ← D - S[1]
B ← B - S[0]
```

decifratura

RC6: schedulazione chiave

L [0,…,c-1] è un array di c = ⌈8b/w⌉ parole di w bit

```
L [0,…,c-1] = chiave con padding di 0 se necessario
S[0] = Pw
for  i ← 1  to 2r+3  do
      S[i] ← S[i-1]+Qw
A ← B  ← 0
i ← j ← 0
do 3·max(c,2r+4) times
      A ← S[i] ← (S[i]+A+B) « 3
      B ← L[j] ← (L[j]+A+B) « (A+B)
      i ← (i+1) mod (2r+4)
      j ← (j+1) mod c
```


Costanti magiche

P_w = espansione binaria del numero di Nepero

e = 2.71828182459045… (decimale) $P_w = Odd[(e-2)2^w]$

Q_w = espansione binaria del rapporto aureo

$Q_w = Odd[(\phi-1)2^w]$

$$\varphi = (1+\sqrt{5})/2 = 1.61803398874989.....(\text{decimale})$$

w	16 bit	32 bit	64 bit
P_w	b7 e1	b7 e1 51 63	b7 e1 51 62 8a ed 2a 6b
Q_w	9E 37	9E 37 79 b9	9E 37 79 b9 7f 4a 7c 15

RC6: Prestazioni Pentium 200 MHz

		cicli/ blocchi	blocchi/ sec	Mbyte/ sec
ANSI C	cifratura	616	325000	5,19
ANSI C	decifratura	566	353000	5,65
JAVA (JDK)	cifratura	16200	12300	0,197
JAVA (JDK)	decifratura	16500	12100	0,194
JAVA (JIT)	cifratura	1010	197000	3,15
JAVA (JIT)	decifratura	955	209000	3,35
assembly	cifratura	254	787000	12,60
assembly	decifratura	254	788000	12,60

Misurazioni della RSA

RC6: Prestazioni

- C e Assembly:
  - Pentium II, 266 MHz, 32 Mbyte RAM, Windows 95, misure scalate a 200MHz
  - Borland C++ Development Suite 5.0
- JAVA
  - Pentium Pro 180 MHz, 64 Mbyte RAM, Windows NT 4.0, misure scalate a 200MHz
  - Compilazione: Javasoft JDK 1.1.6
  - Prestazioni bytecode misurate con
    Interprete Javasoft JDK 1.1.6 (compilazione JIT disabilitata)
    Symantec Java! JustInTime Compiler versione 210.054

RC6: shedulazione chiave

RC6-32/20/16

	cicli	µsecs	key setup/sec
ANSI C	4710	23,5	42500
JAVA (JDK)	107000	537	1860
JAVA (JIT)	14300	71,4	14000

RC6-32/20/32

	cicli	µsecs	key setup/sec
ANSI C	4710	23,6	42400
JAVA (JDK)	107000	548	1820
JAVA (JIT)	15000	75,1	13300

Implementazione ad 8 bit

Insieme istruzioni e tempi: Phillips 80C51

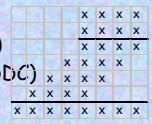
- 6 addizioni
- 2 " ⊕ "
- 2 " quadrati "
- 2 " « 5 "
- 2 " « *variabile* "

$B \cdot (2B+1) = 2B^2+B$

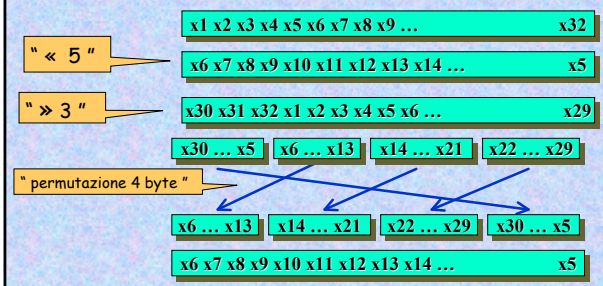
```
B ← B + S[0]
D ← D + S[1]
for i=1 to r do
    t ← (B · (2B+1)) « log w
    u ← (D · (2D+1)) « log w
    A ← ((A⊕ t) « u) + S[2i]
    C ← ((C ⊕ u) « t) + S[2i+1]
    (A,B,C,D) ← (B,C,D,A)
A ← A + S[2r+2]
C ← C + S[2r+3]
```


Implementazione ad 8 bit

- addizione a 32 bit
  - 4 addizioni ad 8 bit con riporto (ADDC)
- "⊕" a 32 bit
  - 4 "⊕" ad 8 bit (XRL)
- " quadrato " a 32 bit
  - 6 moltiplicazioni 8 bit X 8 bit (MUL)
  - 11 addizioni ad 8 bit con riporto (ADDC)



Implementazione " « 5 "



" » 1" a 32 bit → 4 rotazioni a destra 1 bit con riporto (RRC)

Implementazione " » 3 "

Bastano 3 shift " » 1" a 32 bit

" » 1" a 32 bit →
rotazione a destra 1 bit con riporto (RRC)

Implementazione " « *z* "

- Calcolare z' = (ultimi 5 bit di z) mod 8
  - se $z' = 1,2,3,4 \rightarrow z'$ volte " « 1 "
  - se $z' = 5,6,7 \rightarrow 8-z'$ volte " » 1 "
- poi "permutazione byte"
- permutazioni controllate da salti (JB)

In media <2 shift a 32 bit

Implementazione ad 8 bit

	istruzioni	cicli per operazione	cicli
+	4 ADDC	4	4 X 6 = 24
xor	4 XRL	4	4 X 2 = 8
quadrato	6 MUL, 11 ADDC	35	35 X 2 = 70
« 5	12 RRC	12	12 X 2 = 24
« z (media)	8 RRC/RLC, 8 JB	24	24 X 2 = 48
totale			174

Implementazione ad 8 bit

- Numero cicli stimato = 174 X 20 X 4 = 13.920

Indirizzamento, azzeramento, overhead

- Implementazione su Intel 8051: **13.535 cicli**
- Intel: un ciclo ≈ un microsecondo su MCS 51
- Velocità cifratura

  1.000.000 / 13.920 X 128 = 9.2 Kbit / sec