

Cifrari a blocco

- RC2
- RC5
- RC6
- Altri cifrari

AES 0

RC2

Ron Rivest

- Facile da implementare su processori 16 bit
- Usato in S/MIME con chiave 40, 64 e 128 bit
- Descritto in RFC 2268, marzo 1998

AES 1

RC2

AES 2

Input/output round

AES 3

RC2: mixing round

AES 4

Mix up R[i]

$$R[i] \leftarrow R[i] + K[j] + (R[i-1] \& R[i-2]) + (\sim R[i-1] \& R[i-3])$$

$$R[i] \leftarrow R[i] \ll s[i]$$

$$j \leftarrow j+1$$

Indici R[i] mod 4

- 1 if i=0
- 2 if i=1
- 3 if i=2
- 5 if i=3

AES 5



RC2: mixing round

```

R[0] ← R[0] + K[j] + (R[3] & R[2]) + (~R[3] & R[1])
R[0] ← R[0] << 1
j ← j+1
R[1] ← R[1] + K[j] + (R[0] & R[3]) + (~R[0] & R[2])
R[1] ← R[1] << 2
j ← j+1
R[2] ← R[2] + K[j] + (R[1] & R[0]) + (~R[1] & R[3])
R[2] ← R[2] << 3
j ← j+1
R[3] ← R[3] + K[j] + (R[2] & R[1]) + (~R[2] & R[0])
R[3] ← R[3] << 5
j ← j+1

```

AES

6



RC2: mashing round

R[0], R[1], R[2], R[3]

```

Mash R[0]
Mash R[1]
Mash R[2]
Mash R[3]

```

K[R[0]&63]

K[R[1]&63]

K[R[2]&63]

K[R[3]&63]

R[0], R[1], R[2], R[3]

AES

7



Mash R[i]

$$R[i] \leftarrow R[i] + K[R[i-1] \& 63]$$

AES

8



RC2: mashing round

```

R[0] ← R[0] + K[ R[3] & 63 ]
R[1] ← R[1] + K[ R[0] & 63 ]
R[2] ← R[2] + K[ R[1] & 63 ]
R[3] ← R[3] + K[ R[2] & 63 ]

```

AES

9



RC2: espansione chiave

chiave nei byte $L[0], \dots, L[T-1]$ $1 \leq T \leq 128$

```

P[0, ..., 255] ← espansione binaria π
for i=T to 127 do
  L[i] ← P[ L[i-1] + L[ i-T ] ]
L[128-T] ← P[ L[128-T] ]
for i=127-T downto 0 do
  L[i] ← P[ L[i+1] Å L[ i+T ] ]

```

sottochiavi byte $\{L[0], L[1], \dots, L[127]\}$

ovvero word 16 bit $K[0], \dots, K[63]$

$$K[i] = L[2i] + 256 \cdot L[2i+1]$$

AES

10



Tabella P

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

AES

11



RC2

Esercizio:

Decifrazione?



AES

12



RC5

Ron Rivest [1994]

- ❑ Algoritmo semplice
- ❑ Orientato alla parola macchina
- ❑ Usa operazioni comuni dei processori
- ❑ Parametrizzato
 - Lunghezza parola macchina
 - Numero iterazioni
 - Lunghezza chiave
- ❑ Usa poca memoria (smart card e altre device)
- ❑ Rotazioni data-dependent

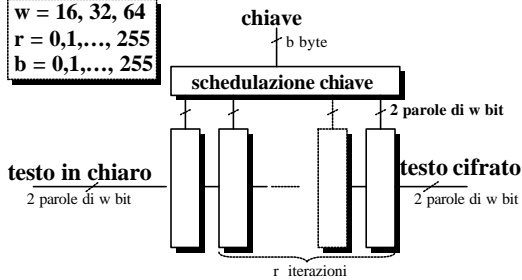
AES

13



RC5-w/r/b

$w = 16, 32, 64$
 $r = 0, 1, \dots, 255$
 $b = 0, 1, \dots, 255$



AES

14



RC5

Operazioni su parole di w bit:

- $a+b$ somma modulo 2^w
- $a-b$ sottrazione modulo 2^w
- $a \oplus b$ XOR bit a bit
- $a \ll b$ shift a sinistra di a di un numero di bit dato dai $\log w$ bit meno significativi di b
- $a \gg b$ shift a destra di a di un numero di bit dato dai $\log w$ bit meno significativi di b

AES

15



RC5: cifratura

Input: testo in chiaro (A,B)
Chiave schedulata: $S[0, \dots, 2r+1]$

```

A ← A + S[0]
B ← B + S[1]
for i = 1 to r do
  A ← ((A ⊕ B) ≪ B) + S[2i]
  B ← ((B ⊕ A) ≪ A) + S[2i+1]

```

Output: testo cifrato (A,B)

AES

16



RC5: decifrazione

```

A ← A + S[0]
B ← B + S[1]
for i = 1 to r do
  A ← ((A ⊕ B) ≪ B) + S[2i]
  B ← ((B ⊕ A) ≪ A) + S[2i+1]

```

cifratura

```

for i = r downto 1 do
  B ← ((B - S[2i+1]) ≫ A) ⊕ A
  A ← ((A - S[2i]) ≫ B) ⊕ B
B ← B - S[1]
A ← A - S[0]

```

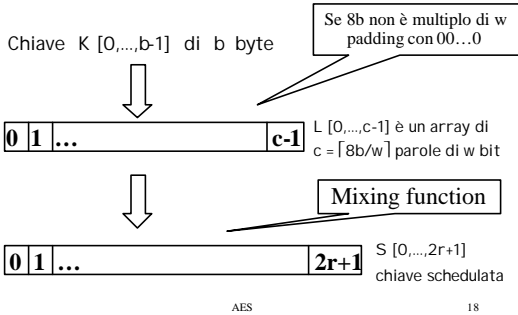
decifrazione

AES

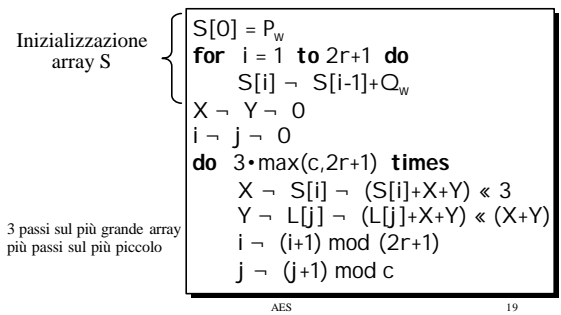
17



RC5: schedulazione chiave



RC6: schedulazione chiave



Costanti magiche

P_w = espansione binaria numero di Nepero
 $e = 2.71828182459045\dots$ (decimale) $P_w = \text{Odd}[(e-2)2^w]$

Q_w = espansione binaria rapporto aureo $Q_w = \text{Odd}[(\phi-1)2^w]$
 $j = (1 + \sqrt{5})/2 = 1.61803398874989\dots$ (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

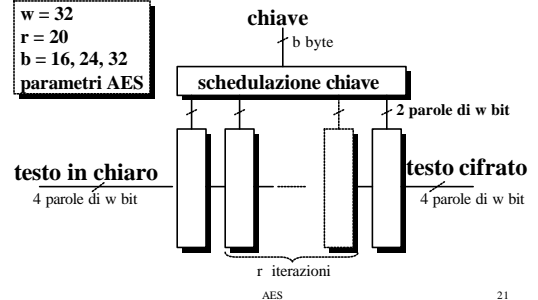
AES

20



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
- $a \oplus b$ XOR bit a bit
- $a \cdot b$ moltiplicazione modulo 2^w
- $a \ll b$ shift a sinistra di a di un numero di bit dato dai $\log w$ bit meno significativi di b
- $a \gg b$ shift a destra di a di un numero di bit dato dai $\log w$ bit meno significativi di b

AES

22



RC6: cifratura

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

```

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)

AES

23



RC6: decifrazione

```

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]

```

decifrazione

AES

24



RC6: schedulazione chiave

$L[0, \dots, c-1]$ è un array di $c = \lceil 8b/w \rceil$ parole di w bit

```

L[0, ..., c-1] = chiave con padding di 0 se necessario
S[0] = P_w
for i = 1 to 2r+3 do
  S[i] ← S[i-1] + Q_w
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

```

25



Costanti magiche

P_w = espansione binaria del numero di Nepero

$e = 2.71828182459045\dots$ (decimale) $P_w = \text{Odd}[(e-2)2^w]$

Q_w = espansione binaria del rapporto aureo

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

$j = (1 + \sqrt{5})/2 = 1.61803398874989\dots$ (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

AES

26



RC6: Prestazioni Pentium 200 MHz

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

Misurazioni della RSA

AES

27



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

AES

28



RC6: shedulazione chiave

RC6-32/20/16

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

AES

29



Implementazione ad 8 bit

Insieme istruzioni e tempi: Phillips 80C51

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

```

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]

```

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

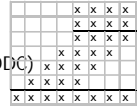
AES

30



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)

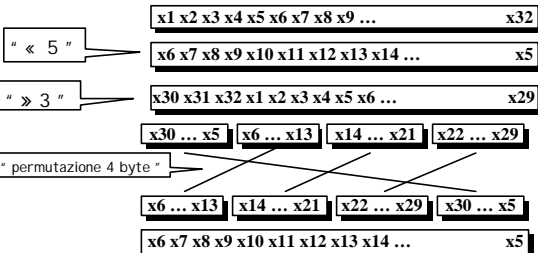


AES

31



Implementazione "« 5"



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

AES

32



Implementazione "» 3"

Bastano 3 shift "» 1" a 32 bit

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

AES

33



Implementazione "« z"

❑ Calcolare $z' = (\text{ultimi 5 bit di } z) \bmod 8$

$\left\{ \begin{array}{l} \text{se } z' = 1,2,3,4 \text{ @ } z' \text{ volte "« 1"} \\ \text{se } z' = 5,6,7 \text{ @ } 8-z' \text{ volte "» 1"} \end{array} \right.$

❑ poi "permutazione byte"

In media <2 shift a 32 bit

❑ permutazioni controllate da salti (JB)

AES

34



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

AES

35



Implementazione ad 8 bit

- Numero cicli stimato = $174 \times 20 \times 4 = 13.920$

Indirizzamento, azzeramento, overhead

- Implementazione su Intel 8051: **13.535 cicli**
- Intel: un ciclo \approx un microsecondo su MCS 51
- Velocità cifratura
 $1.000.000 / 13.920 \times 128 = 9.2 \text{ Kbit / sec}$

AES

36



International Data Encryption Algorithm (IDEA)

Xuejia Lai, James Massey [1990], [1991]



- Più efficiente su processori a 16 bit
- Usato nel PGP

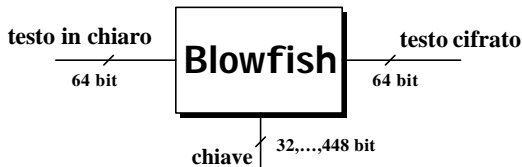
AES

37



Blowfish

Bruce Schneier [1993], [1994]



- velocità cifratura: 26 cicli di clock per byte su processori a 32 bit
- memoria usata <5K, non adatto per smart card
- semplici operazioni: addizione, XOR, table lookup
- non adatto per applicazioni con frequenti cambiamenti di chiave

AES

38



SAFER

James Massey [1994], [1995]



- Secure And Fast Encryption Routine
- operazioni sui byte, $GF(2^8)$
- SAFER K-64, per Cylink Corp., chiave 64 bit
- SAFER K-128, schedulazione chiave (128 bit) fatta dal Minister of Home Affairs, Singapore,

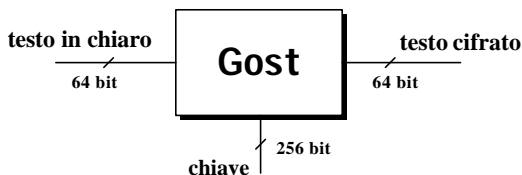
AES

39



Gost

Gosudarstvennyi Standard 28147-89 [1989]



- Feistel Cypher, 32 round
- S-box non specificate nello standard
 - Fornite da organizzazioni governative?
 - Generate a caso dai costruttori?

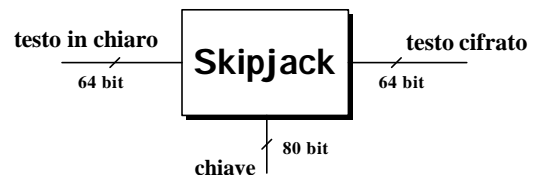
AES

40



Skipjack

NSA [1985-1990]



- usato nel Clipper Chip
- 32 iterazioni su 4 parole da 16 bit ciascuna

AES

41



Ancora cifrari a blocchi

- Varianti del DES: DESX, GDES, CRYPT(3), RDES, sⁿDES, ...
- Madryga, NewDES, FEAL, REDOC, LOKI, CA-1.1, Khufu, Knafre, MMB, CAST, 3-Way, ...
- ... AES