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 dell'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 Ago 98 (presentazione di 15 candidature)
- Pubblico scrutinio
- Seconda conferenza AES, 22-23 Mar 99 (presentazione analisi e testing)
- 9 Agosto 1999: annuncio dei 5 finalisti
- Pubblico scrutinio (fino al 15 maggio 2000)
- Terza conferenza AES, 13-14 aprile 2000 (presentazione analisi e testing)
- Agosto 2000: Scelta del finalista/finalisti da includere in una proposta di *Federal Information Processing Standard* (FIPS)
- Pubblico scrutinio
- Eventuale revisione… 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 per l'AES

- 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

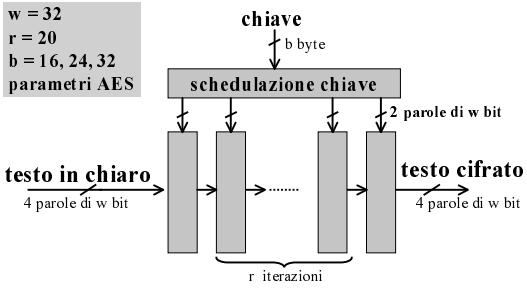
5 finalisti e 15 candidati per l'AES

MARS	IBM
RC6	RSA Laboratories
RIJNDAEL	Joan Daemen, Vincent Rijmen
SERPENT	Ross Anderson, Eli Biham, Lars Knudsen
TWOFISH	B.Schneider, J.Kelsey, D.Whiting, D.Wagner, C.Hall, N.Ferguson

CAST256	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.

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 $c = \lceil 8b/w \rceil$ 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 = i = j = 0
v = 3•max(c,2r+4)
for s = 1 to v do
      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

$e = 2.71828182459045\ldots$ (decimale)

Q_w = espansione binaria del rapporto aureo

$\varphi = (1+\sqrt{5})/2 = 1.61803398874989\ldots$ (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

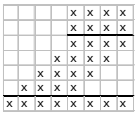
- Insieme istruzioni e tempi: Phillips 80C51
- 6 addizioni
- 2 " ⊕ "
- 2 " quadrati "
- 2 " « 5 "
- 2 " « *variabile* "

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

```
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]
```

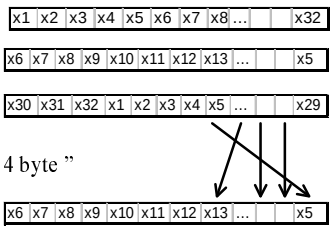

Implementazione ad 8 bit

- 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 ad 8 bit

- " « 5 "
  - " » 3"
  - " permutazione 4 byte "



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

Implementazione ad 8 bit

- " « z "

  se $z' = 1,2,3,4 \rightarrow z'$ volte " « 1 "

  se $z' = 5,6,7 \rightarrow 8-z'$ volte " » 1 "

  In media <2 shift a 32 bit

  poi "permutazione byte"

  – z' determinato dagli ultimi 5 bit di z

  – permutazioni controllate da salti (JB)

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