Cifrari a blocchi:
Advanced Encryption Standard

Barbara Masucci

Dipartimento di Informatica ed Applicazioni
Università di Salerno

masucci@dia.unisa.it
http://www.dia.unisa.it/professori/masucci

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 (112 per 3-DES)
  - blocchi di 64 bit
  - criteri costruttivi non chiari (ci sono trapdoor nelle S-box?)
  - lento nell'implementazione software (3-DES ancora di più)
- Obiettivo del NIST:
  - nuovo cifrario a blocchi per uso commerciale e governativo più sicuro ed efficiente di 3-DES

Processo di Selezione AES

- 12 Settembre 1997: il NIST indice un concorso pubblico per la nomina dell'AES
- 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 (MARS, RC6, RINJDAEL, SERPENT, TWOFISH)
- Pubblico scrutinio
- Terza conferenza AES, 13-14 aprile 2000 (presentazione analisi e testing)

Processo di Selezione AES

- 2 ottobre 2000: Scelta del finalista: RINJDAEL
- 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

Requisiti e Selezione per l'AES

Requisiti richiesti dal NIST:
- Cifrario a blocchi
- Lunghezza chiave: 128, 192, o 256 bit
- Lunghezza blocco: 128 bit
- Permette l'implementazione su smart-card
- Royalty-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.Schneier, 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

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 (anche 192 o 256 bit)
  - Lunghezza della chiave: 128, 192, o 256 bit
  - Numero di round: 10, 12 o 14
  - Schedulazione della chiave: 44, 52 o 60 sottochiavi a 32 bit
- Ogni round (tranne l'ultimo) è una composizione uniforme e parallela di 4 passi
  - SubBytes (sostituzione byte per byte mediante S-box)
  - ShiftRows (permutazione)
  - MixColumns (sostituzione che usa aritmetica sul campo finito $GF(2^8)$)
  - AddRound key (XOR bit a bit con chiave espansa)

Parametri AES

	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

1 word = 32 bit

Chiavi AES

- Con 128 bit: 2^{128} = 3.4x 10^{38} chiavi possibili
  - Una macchina che prova 2^{55} chiavi al secondo impiega 149.000 miliardi di anni per rompere AES
- Con 192 bit: 2^{192} = 6.2x 10^{57} chiavi possibili
  - …
- Con 256 bit: 2^{256} = 1.1x 10^{77} chiavi possibili
  - …

Si pensa che AES resterà sicuro per i prossimi 20 anni

Chiave e blocco

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

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

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

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

Barbara Masucci - DIA – Università di Salerno 10

State

- Operazioni effettuate su una matrice di byte, detta state
  - 4 righe
  - Nb colonne, costituite da word a 32 bit
  - $S_{r,c}$ 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

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

Barbara Masucci - DIA – Università di Salerno 11

State

input bytes **state** **output bytes**

in_0	in_4	in_8	in_{12}
in_1	in_5	in_9	in_{13}
in_2	in_6	in_{10}	in_{14}
in_3	in_7	in_{11}	in_{15}

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

out_0	out_4	out_8	out_{12}
out_1	out_5	out_9	out_{13}
out_2	out_6	out_{10}	out_{14}
out_3	out_7	out_{11}	out_{15}

$S_{r,c} \leftarrow in_{r+4c}$ $0 \le r < 3$ $0 \le c < Nb\text{-}1$

$out_{r+4c} \leftarrow S_{r,c}$ $0 \le r < 3$ $0 \le c < Nb\text{-}1$

Barbara Masucci - DIA – Università di Salerno 12

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

Barbara Masucci - DIA – Università di Salerno 13

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)

$\{b_7b_6b_5b_4b_3b_2b_1b_0\} \rightarrow b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0$

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

Addizione su byte

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

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

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

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

Moltiplicazione su byte

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

$\{01010111\} \cdot \{10000011\} = \{11000001\}$

{57} • {83} = {c1}

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

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

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

dividiamo per m(x) e teniamo il resto

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

$c_0 = a_0 \bullet b_0$

$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$

$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$

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

$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$

$c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$

$c_6 = a_3 \bullet b_3$

non va in una word!

Polinomi con coefficienti in GF(2^8)

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

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

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

$c_0=a_0\bullet b_0$

$c_1=a_1\bullet b_0\oplus a_0\bullet b_1$

$c_2=a_2\bullet b_0\oplus a_1\bullet b_1\oplus a_0\bullet b_2$

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

$c_4=a_3\bullet b_1\oplus a_2\bullet b_2\oplus a_1\bullet b_3$

$c_5=a_3\bullet b_2\oplus a_2\bullet b_3$

$c_6=a_3\bullet b_3$

modulo x^4+1

Polinomi con coefficienti in GF(2^8)

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

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

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

cioè

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

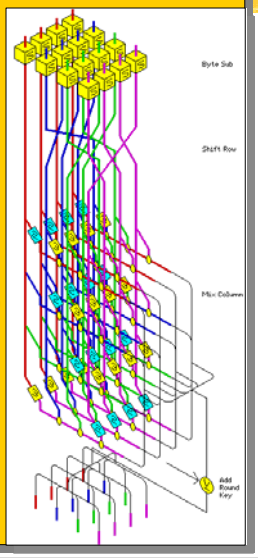
Polinomi con coefficienti in GF(2^8)

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

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

Pseudocodice per l'AES

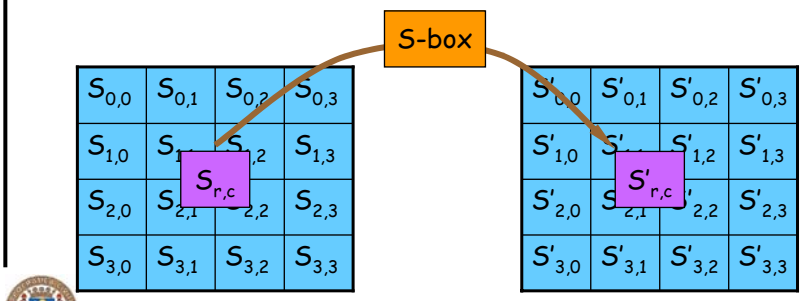
```
Cipher (byte in[4·Nb], byte out[4·Nb], word w[Nb· (Nr + 1)])
byte state[4,Nb]
state ← in
AddRoundKey (state, w)
for round = 1 to Nr - 1
        SubBytes (state)
        ShiftRows (state)
        MixColumns (state)
        AddRoundKey (state, w + round · Nb)
SubBytes (state)
ShiftRows (state)
AddRoundKey (state, w + Nr · Nb)
out ← state
```



SubBytes Transformation

Bytes trasformati mediante una S-box non lineare ma invertibile

$S'_{r,c} \leftarrow \text{S-box}(S_{r,c})$ $0 \le r < 3$ $0 \le c < Nb\text{-}1$



S-box

- Tabella 16x16 contenente una permutazione dei 256 valori possibili di un byte
- I primi 4 bit determinano l'indice di riga, gli altri 4 l'indice di colonna

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

Esempio: {53}→ {ed}

Costruzione S-box

- Inizializzare la S-box con i valori dei byte in ordine ascendente riga per riga
  - Prima riga: {00}, {01}, …, {0F},
  - Seconda riga: {10}, {11}, …, {1F},
  - …
- Sostituire ciascun byte con il suo inverso moltiplicativo in GF(2^8)
  - {00} resta {00}
- Applicare una trasformazione affine in GF(2^8)

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

i-esimo bit del byte {63}

Costruzione S-box

Forma matriciale della trasformazione

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

Proprietà S-box

- L'output non è una funzione lineare dell'input
- Non ha punti fissi diretti né opposti
  - S-box(a) ≠ a e S-box($\bar{a}$) ≠ $\bar{a}$
- E' invertibile
  - Inverse_S-box(S-box(a)) = a
- Non è self-invertibile
  - S-box(a) ≠ Inverse_S-box(a)
- Progettata per resistere ad attacchi crittoanalitici noti

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

Esempio: {ed}→ {53}

ShiftRows Transformation

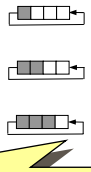
Righe shiftate di posizioni differenti (shift ciclico a sx)

$S'_{r,c} \leftarrow S_{r,(c+shift(r,Nb))modNb}$

$0 \le r < 3 \quad 0 \le c < Nb\text{-}1$

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

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



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

I 4 byte di una colonna sono spostati su colonne differenti

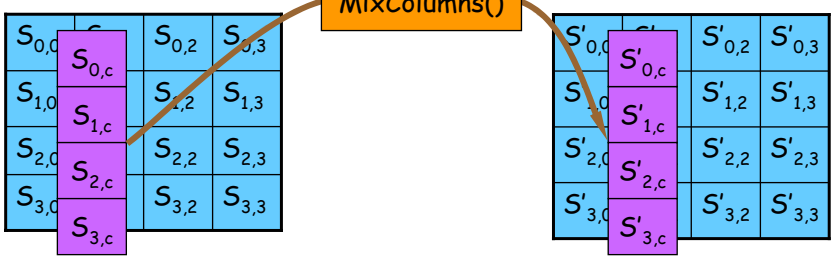
MixColumns Transformation

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

Moltiplicazione mod x^4+1 con polinomio fissato

$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$

MixColumns()

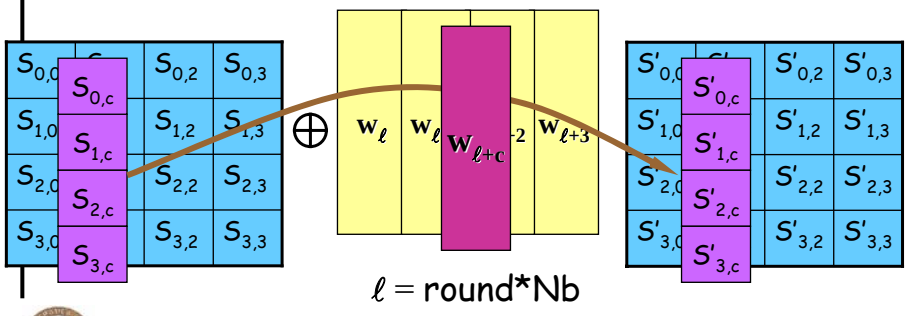


Bytes nelle colonne combinati linearmente

AddRoundKey Transformation

round key (word)

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



$\ell = round*Nb$

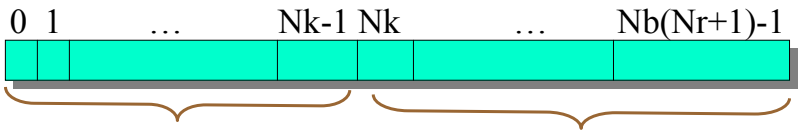
Espansione chiave

- A partire dalla chiave iniziale di 4*Nk byte, genera un array di Nb*(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]
    - Rcon[i]=[RC[i],{00},{00},{00}] — Word i cui tre byte più a destra sono sempre {00}
    - RC[i]=2 • RC[i-1]
    - RC[1]= {01}

i	1	2	3	4	5	6	7	8	9	10
RC[i]	{01}	{02}	{04}	{08}	{10}	{20}	{40}	{80}	{1B}	{36}

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 multipli 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]=[RC[i],{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`

$Nk = 4$

$w_0 = 2b7e1516\ w_1 = 28aed2a6$

$w_2 = abf71588\ w_3 = 09cf4f3c$

i (dec)	Temp	After RotWord()	After SubByte()	Rcon[i/Nk]	After XOR with Rcon	w[i-Nk]	w[i]= temp XOR w[i-Nk]
4	09cf4f3c	cf4f3c09	8a84eb01	01000000	8b84eb01	2b7e1516	a0fafe17
5	a0fafe17					28aed2a6	88542cb1
6	88542cb1					abf71588	23a33939
7	23a33939					09cf4f3c	2a6c7605
8	2a6c7605	6c76052a	50386be5	02000000	52386be5	a0fafe17	f2c295f2
9	f2c295f2					88542cb1	7a96b943
10	7a96b943					23a33939	5935807a
11	5935807a					2a6c7605	7359f67f
12	7359f67f	59f67f73	cb42d28f	04000000	cf42d28f	f2c295f2	3d80477d
13	3d80477d					7a96b943	4716fe3e
14	4716fe3e					5935807a	1e237e44
15	1e237e44					7359f67f	6d7a883b
16	6d7a883b	7a883b6d	dac4e23c	08000000	d2c4e23c	3d80477d	ef44a541
17	ef44a541					4716fe3e	a8525b7f
18	a8525b7f					1e237e44	b671253b
19	b671253b					6d7a883b	db0bad00
20	db0bad00	0bad00db	2b9563b9	10000000	3b9563b9	ef44a541	d4d1c6f8
21	d4d1c6f8					a8525b7f	7c839d87
22	7c839d87					b671253b	caf2b8bc
23	caf2b8bc					db0bad00	11f915bc
24	11f915bc	f915bc11	99596582	20000000	b9596582	d4d1c6f8	6d88a37a
25	6d88a37a					7c839d87	110b3efd
26	110b3efd					caf2b8bc	dbf98641
27	dbf98641					11f915bc	ca0093fd
28	ca0093fd	0093fdca	63dc5474	40000000	23dc5474	6d88a37a	4e54f70e
29	4e54f70e					110b3efd	5f5fc9f3
30	5f5fc9f3					dbf98641	84a64fb2
31	84a64fb2					ca0093fd	4ea6dc4f
32	4ea6dc4f	a6dc4f4e	248642f	80000000	a486842f	4e54f70e	ead27321
33	ead27321					5f5fc9f3	b58dbad2
34	b58dbad2					84a64fb2	312bf560
35	312bf560					4ea6dc4f	7f8d292f
36	7f8d292f	8d292f7f	5da515d2	1b000000	46a515d2	ead27321	ac7766f3
37	ac7766f3					b58dbad2	19fadc21
38	19fadc21					312bf560	28d12941
39	28d12941					7f8d292f	575c006e
40	575c006e	5c006e57	4a639f5b	36000000	7c639f5b	ac7766f3	d014f9a8
41	d014f9a8					19fadc21	c9ee2589
42	c9ee2589					28d12941	e13f0cc8
43	e13f0cc8					575c006e	b6630ca6

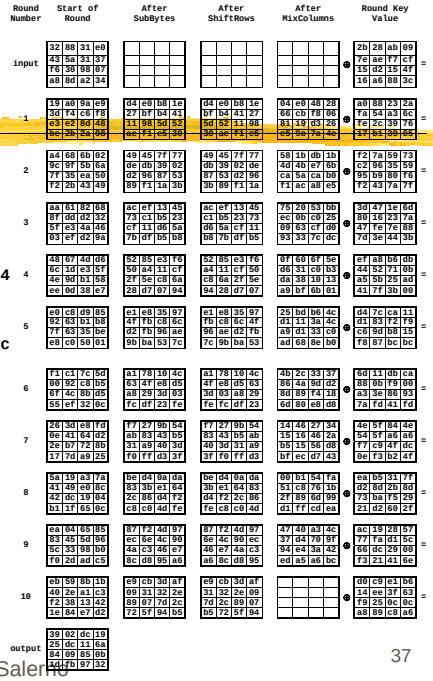
Cifratura Esempio

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`



Decifratura AES

- L'algoritmo di decifratura non è lo stesso della cifratura
  - Usa una diversa sequenza di trasformazioni
  - Usa le trasformazioni inverse
  - Svantaggio: necessaria una doppia implementazione
- Esiste anche un algoritmo di decifratura che ha la stessa struttura di quello di cifratura
  - Stessa sequenza di trasformazioni
  - Usa le trasformazioni inverse

Pseudocodice decifrazione AES

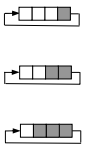
```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])
byte  state[4,Nb]
state ← in
AddRoundKey(state, w + Nr * Nb)
for round = Nr - 1 step -1 to 1
        InvShiftRows(state)
        InvSubBytes(state)
        AddRoundKey(state, w + round * Nb)
        InvMixColumns(state)
InvShiftRows(state)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

InvShiftRows Transformation

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

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

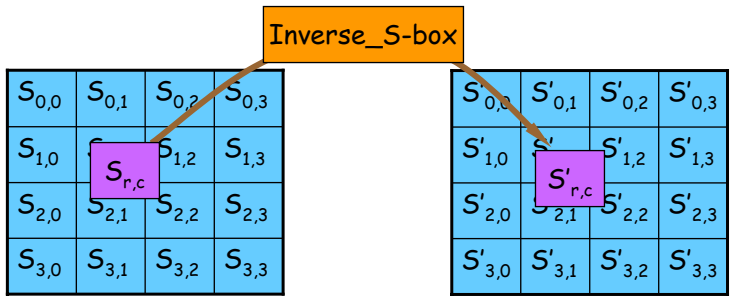
$S_{0,0}$	$S_{0,1}$	$S_{0,2}$	$S_{0,3}$
$S_{1,0}$	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$
$S_{2,0}$	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$
$S_{3,0}$	$S_{3,1}$	$S_{3,2}$	$S_{3,3}$



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

InvSubBytes Transformation

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



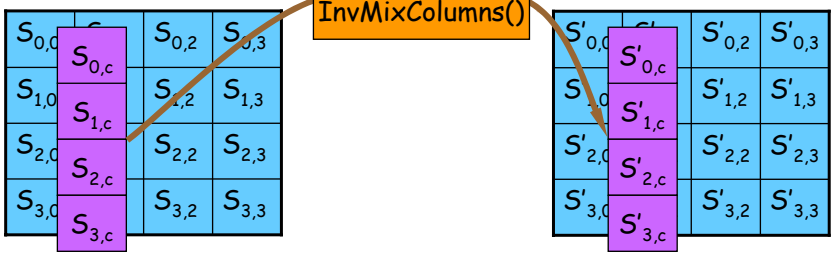
InvMixColumns Transformation

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

Moltiplicazione mod x^4+1 con polinomio fissato

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

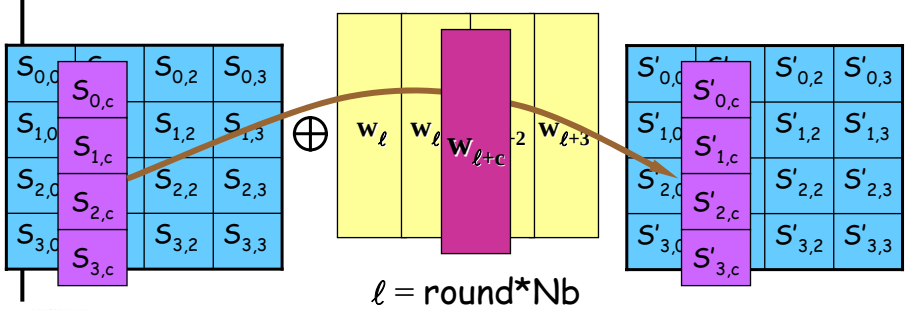
InvMixColumns()



AddRoundKey Transformation

È l'inversa di se stessa!

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



Bibliografia

- Cryptography and Network Security by W. Stallings (2003)
  - cap. 5 (AES) + appendice
- Tesina di Sicurezza su reti
  - Advanced Encryption Standard