

storia AES

- nel settembre 97 il NIST comincia la selezione del successore del DES
- si chiamerà AES – Advanced Encryption Standard
- 21 proposte - solo 15 soddisfano i criteri necessari
- i 15 candidati vengono annunciati nella *First AES Candidate Conference*, nel 98
- seguono la *Second AES Candidate Conference*, a Roma nel 99, dopo la quale vengono annunciati i 5 finalisti
- sono MARS, RC6, Rijndael, Serpent, Twofish
- la *Third AES Candidate Conference* si tiene nell'aprile 2000
- nell'ottobre 2000 viene scelto Rijndael

- la competizione è stata molto internazionale
- Rijndael è stato proposta da due crittografi belgi: Daemen e Rijmen
- MARS (IBM), RC6 (Rivest e RSA Security), Twofish (Schneier e altri)
- Serpent è di Anderson (UK), Biham (Israele), Knudsen (Danimarca)

	Rijndael	Serpent	Twofish	MARS	RC6
General Security	2	3	3	3	2
Implementation Difficulty	3	3	2	1	1
Software Performance	3	1	1	2	2
Smart Card Performance	3	3	2	1	1
Hardware Performance	3	3	2	1	2
Design Features	2	1	3	2	1
Total	16	14	13	10	9

I won!!



(illustrazione di Jeff Moser da <http://www.moserware.com>)

AES

- preceduto da SHARK (attaccato con successo da Jakobsen e Knudsen)
- e da Square
- blocchi di lunghezza 128
- 3 lunghezze per la chiave: 128, 192, 256
- nr di round 10, 12, 14 a seconda della lunghezza della chiave
- descriviamo la versione a 10 round con chiave di 128 bit

AES - blocchi

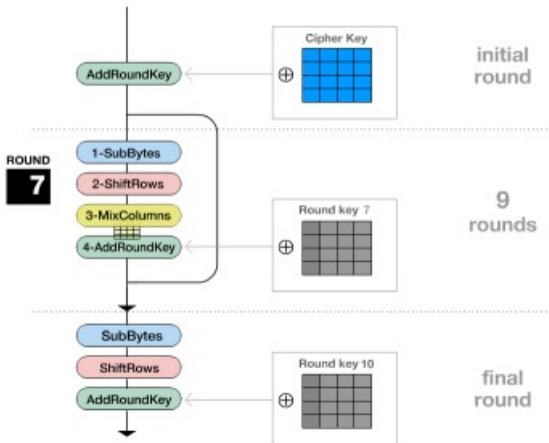
- in ogni momento, il blocco di 128 bit è pensato come una matrice 4×4 di byte (8 bit) – 16 byte = 128 bit

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

- spesso $s_{i,j}$ viene pensato come una coppia di cifre esadecimali (ognuna rappresenta 4 bit)
- ex: $s_{i,j} = 5D = 01011101$
- a ogni stadio dell'algoritmo, la tabella si chiama **Stato** (state)

AES - descrizione (10 round)

Encryption process



AES - descrizione (10 round)

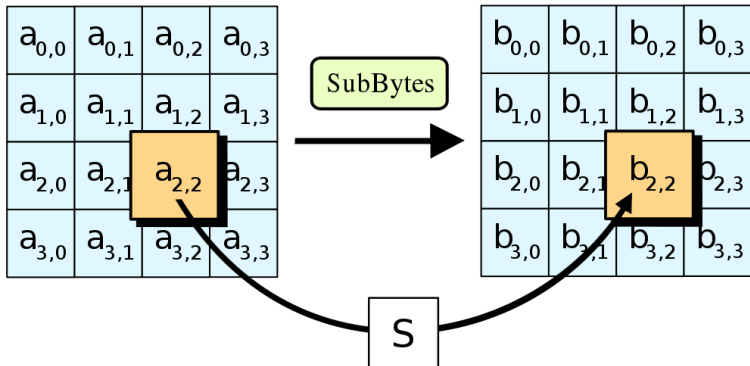
- dato un PT x , si inizializza la tabella **Stato** con x . Si esegue la trasformazione **ADDROUNDKEY**, che è uno XOR fra **Stato** e la chiave di round
- per i primi 9 round:
 - sostituzione **SUBBYTES** su **Stato** usano una S-box
 - permutazione **SHIFTRows**
 - trasformazione **MIXColumns**
 - trasformazione **ADDROUNDKEY**
- nel 10 round si salta **MIXColumns**: si esegue dunque **SUBBYTES**, **SHIFTRows** e **ADDROUNDKEY**
- l'output **Stato** è il CT y

AES



(a) Input, state array, and output

AES - SUBBYTES



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

AES - SUBBYTES

- i byte vengono trattati come elementi del campo $GF(2^8) = \mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x + 1)$; (Rijndael's finite field)
- la sostituzione - SUB-BYTES prende in input un byte $a_7 \dots a_1 a_0$ e dà in output un byte $b_7 \dots b_1 b_0$
 - ① $a = a_7 x^7 + \dots + a_1 x + a_0 \in GF(2^8)$
 - ② si calcola $a^{-1} = \hat{a}_7 x^7 + \dots + \hat{a}_1 x + \hat{a}_0 \in GF(2^8)$
 - ③ si applica una trasformazione affine a $\hat{a}_7 \dots \hat{a}_1 \hat{a}_0$ (della forma $Ax + B$, A matrice invertibile 8×8 , B matrice 8×1)
 - ④ il risultato dà l'output $b_7 \dots b_1 b_0$
- si può mostrare che la trasformazione $a \rightarrow a^{-1}$ fornisce non-linearità e “resiste bene” alla crittoanalisi lineare e differenziale
- senza trasformazione affine la sostituzione non è sicura – questo è il problema di SHARK

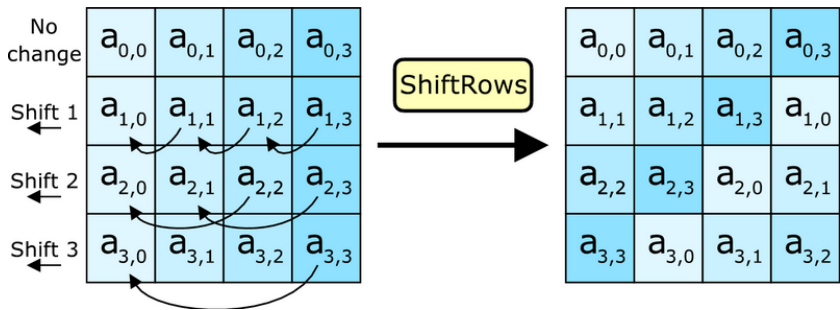
trasformazione affine

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

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

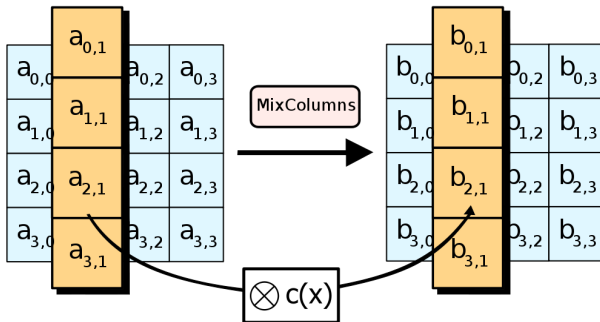
- se si parte da 53, la tabella dà *ED*
- $53 = 01010011$ rappresenta il polinomio $x^6 + x^4 + x + 1$
- si può mostrare (usando per esempio l'algoritmo di Euclide esteso) che l'inverso di questo elemento in $GF(2^8)$ è il polinomio $x^7 + x^6 + x^3 + x$, che in notazione binaria è 11001010
- la trasformazione affine dà $11001010 \rightarrow 11101101$
- in esadecimale $11101101 \rightarrow ED$

AES - SHIFTRows



- funziona come sopra
- i 4 byte di una stessa colonna vengono sparsi sulle 4 colonne
- fornisce diffusione

AES - MixColumns



- opera su ogni colonna
- può essere descritta come prodotto di matrici in $GF(2^8)$ – è una trasformazione lineare

$$\begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} =$$

$$= \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$$

- la moltiplicazione è sempre quella di $GF(2^8) = \mathbb{Z}_2[x]/(x^8 + x^4 + x^3 + x + 1)$

-

$$(02 \quad 03 \quad 01 \quad 01) \begin{pmatrix} 87 \\ 6E \\ 46 \\ A6 \end{pmatrix}$$

$$= (02 \bullet 87) \oplus (03 \bullet 6E) \oplus (01 \bullet 46) \oplus (01 \bullet A6) = 47$$

- la matrice è circolante
- viene da un codice (correttore di errori) MDS (maximum distance separable)

chiave

- `ADDROUNDKEY` è uno XOR
- viene usata 11 volte nella versione a 10 round
- la chiave è a 128 bit (sempre nella versione a 10 round)
- abbiamo bisogno di ottenere 11 chiavi di round a 16 byte (=128 bit)
- si usa la procedura `KEYEXPANSION`
- è la parte dell'algoritmo che è stata più criticata (soprattutto per la versione con chiave a 256 bit)

KEY EXPANSION

- si lavora sulle colonne - ogni colonna è pensata come una parola di 4 byte
- ci servono 44 parole (11 byte) $w[0], w[1], \dots, w[43]$
- la chiave di partenza occupa le prime 4 parole
- le rimanenti 40 parole vengono generate 4 alla volta



(b) Key and expanded key

KEYEXPANSION

- la chiave di partenza occupa le prime 4 parole
- le rimanenti 40 parole vengono generate 4 alla volta
- ogni parola $w[i]$ ($i \geq 4$) dipende solo dalla precedente $w[i - 1]$ e da quella che si trova 4 posizioni prima, $w[i - 4]$
- per le posizioni $i \not\equiv 0 \pmod{4}$, $w[i] = w[i - 1] \oplus w[i - 4]$
- per le posizioni $i \equiv 0 \pmod{4}$, $w[i] = g(w[i - 1]) \oplus w[i - 4]$

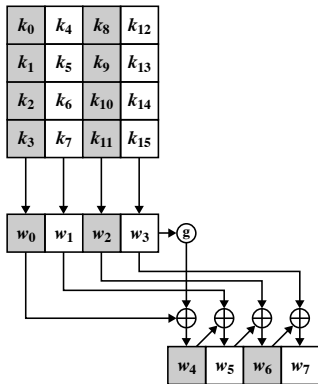


Figure 5.6 AES Key Expansion

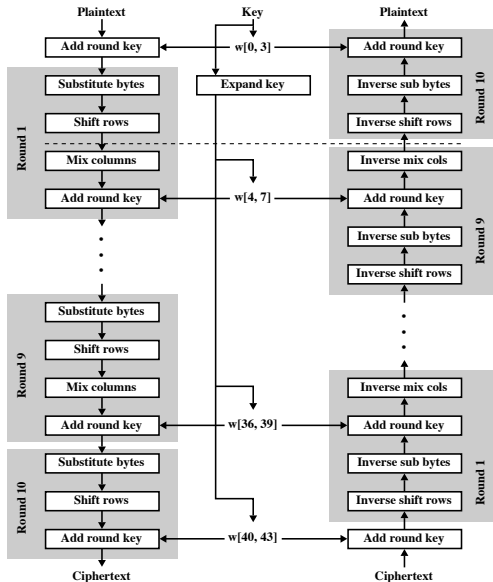
- la funzione g nella KEYEXPANSION usa la S-box SUBBYTES della cifratura

- per prima cosa si ruota la parola:
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \rightarrow \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_0 \end{bmatrix}$$

- a questa parola si applica la sostituzione SUBBYTES
- il primo byte della parola ottenuta viene messo in XOR con una costante che dipende dal round

decifratura

- non è un cifrario di Feistel – cifratura e decifratura sono un po' differenti
- chiaramente bisogna applicare tutte le trasformazioni nell'ordine inverso, e usare le chiavi nell'ordine inverso.
- inoltre devo usare le operazioni inverse di MIXCOLUMNS, SHIFTRows e SUBBYTES – solo ADDROUNDKEY coincide con la sua inversa



(a) Encryption

(b) Decryption

- per esempio, INVERSEMIXCOLUMNS è la trasformazione

$$\begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix} \cdot \begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,2} & a_{0,3} \\ a_{1,0} & a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,0} & a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,0} & a_{3,1} & a_{3,2} & a_{3,3} \end{pmatrix} =$$

$$= \begin{pmatrix} b_{0,0} & b_{0,1} & b_{0,2} & b_{0,3} \\ b_{1,0} & b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,0} & b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,0} & b_{3,1} & b_{3,2} & b_{3,3} \end{pmatrix}$$

- è un po' più pesante da implementare

- primi attacchi teorici nel 2009
- alle versioni con chiavi lunghe
- Dall'abstract dell'articolo di Alex Biryukov, Orr Dunkelman, Nathan Keller, Dmitry Khovratovich e Adi Shamir:

AES is the best known and most widely used block cipher. Its three versions (AES-128, AES-192, and AES-256) differ in their key sizes (128 bits, 192 bits and 256 bits) and in their number of rounds (10, 12, and 14, respectively). In the case of AES-128, there is no known attack which is faster than the 2^{128} complexity of exhaustive search. However, AES-192 and AES-256 were recently shown to be breakable by attacks which require 2^{176} and 2^{119} time, respectively. While these complexities are much faster than exhaustive search, they are completely non-practical, and do not seem to pose any real threat to the security of AES-based systems.