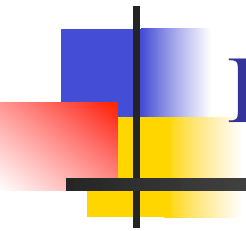


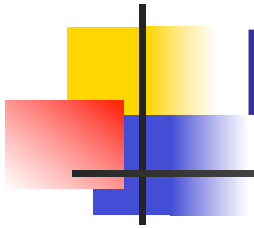
Corso di Crittografia

Prof. Dario Catalano

Advanced Encryption Standard

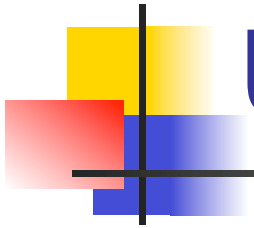
Perche' un nuovo standard?

- Quando si è deciso di sostituire DES, questo era ancora sicuro.
- Sorge naturale la questione del perche' sostituirlo.
- La sicurezza di un cifrario a blocchi dipende anche dalla dimensione del blocco.
  - Un cifrario a blocchi puo' essere "rotto" in tempo $2^{n/2}$ (n dimensione del blocco)
- Il DES non e' abbastanza rapido (in software)



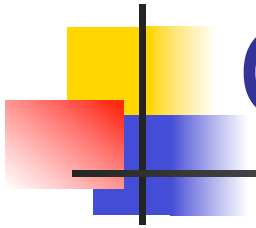
Mission Impossible

- Sostanzialmente vogliamo un nuovo cifrario che:
  - Abbia dimensioni del blocco e della chiave piu' grandi rispetto al DES
  - Sia piu' veloce del DES



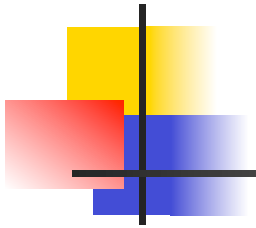
Un po’ di storia

- 1998: Il National Institute of Standards end Technology (NIST) annuncia una competizione per un nuovo cifrario a blocchi.
- A differenza del DES, il nuovo cifrario deve essere costruito dal “pubblico”.
- 15 sottomissioni
- 2001: Dopo due anni di studi, la scelta cade sull’algoritmo Rijndael (Vincent Rijmen & Joan Daemen).

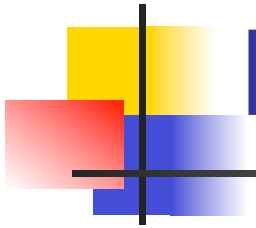


Qualche informazione su AES

- La lunghezza del blocco e' n=128
- La dimensione della chiave e' variabile: 128, 192 o 256 bit.
- Per questo motivo lo standard specifica tre cifrari a blocchi differenti: AES128, AES192 e AES256.
- Nel seguito indicheremo con AES l'algoritmo AES128.

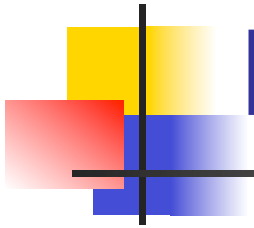


- AES è il principale cifrario a blocchi per molte app. commerciali
- Nel 2003 l'NSA ha annunciato che avrebbe utilizzato AES per cifrare documenti classificati
  - Fino al livello SECRET per ogni dimensione della chiave
  - TOP SECRET per chiavi di 196 o 256 bit.
  - Nessun algoritmo pubblico era mai stato precedentemente utilizzato per questi scopi.



Breve descrizione di AES

- AES si compone di tre livelli fondamentali (layers)
  - Ogni layer manipola tutti i bit dello stato.
- **Key Addition layer** In cui lo stato attuale viene aggiornato usando una chiave di round di 128 bit.
- **Byte Substitution layer** (S-Box) Lo stato modificato utilizzando una trasformazione non lineare.



Breve descrizione di AES (II)

- **Diffusion layer. "**Diffonde" le trasformazioni effettuate su tutti i bit dello stato.
  - Consiste di due procedure (entrambe effettuano operazioni lineari)

L'algoritmo AES

$AES_k(M)$ // $|M|=128$ e $|K|=128$

$(K_0,\ldots,K_{10}) \leftarrow$ *Espandi_Chiave* (K) // $|K_i|=128$

$s \leftarrow M \oplus K_0$ // s e' chiamato lo stato

for r=1 to 10 do

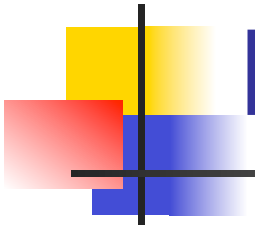
s ← *S* (s)

s ← *Shift_Rows* (s)

if r ≤ 9 then *mix_cols* (s)

$s \leftarrow s \oplus K_r$

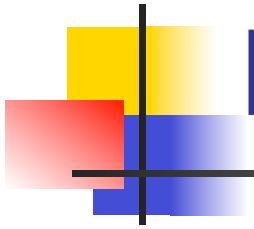
Return s



Descrizione generale di AES

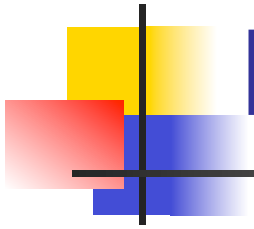
- Descriveremo AES in termini delle 4 funzioni: expand, S, shift-rows e mix-cols
- Il valore s e' chiamato lo *stato*.
- Tale valore e' inizializzato a quello del messaggio M. Lo stato finale e' il crittotesto C
- Cio' che avviene in ogni iterazione del ciclo for e' detto round.
- AES consiste di 10 rounds

La funzione expand



- E’ la funzione di espansione della chiave.
- Prende in input una stringa di 128 bit e produce 11 sotto chiavi di 128 bit ciascuna.
- Adesso ripassiamo un po’ di matematica…

La S-box di AES - I



- Ogni elemento (tranne 0) a ammette un *inverso* $inv(a)=a^{-1}$.
- Un inverso e' quell'elemento tale che $a \cdot inv(a)=1$
- Cerchiamo di capire come calcolare tali inversi.

La S-box di AES - II

- Consideriamo la seguente trasformazione e chiamiamola AFF (l'aritmetica e' in GF(2))

$$x'_0 x'_1 x'_2 x'_3 x'_4 x'_5 x'_6 x'_7 = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

NB: L'indicizzazione dei bytes e' al contrario

La S-box di AES - III

Definiamo $S:\{0,1\}^8 \rightarrow \{0,1\}^8$ I

S(a)

a ← inv'(a)

a' ← AFF (a)

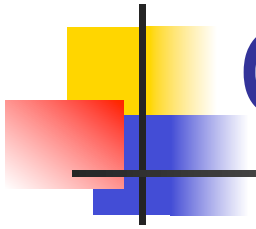
Return a'

Non abbiamo ancora finito (!)

- Ora che sappiamo come calcolare la funzione di base, estendiamo la funzione S (senza cambiar nome) in modo che possa prendere input piu' grandi.
- Data una stringa di m bytes A=A[1]…[A[m], poniamo

$$S(A)=S(A[1])\ldots S(A[m])$$

- Dunque S prende in input uno stato s (16 bytes) e lo modifica secondo le procedure descritte.



Ottimizzazioni

- Con molti (ma non tutti) gli elementi del campo di Galois di Rijndael e' possibile "percorrere" tutti gli elementi del campo (tranne lo zero) via esponenziazione.
- Tali elementi vengono detti generatori.

I generatori di Rijndael (inf. presa da internet)

3 5 6 9 11 14 17 18 19 20 23 24 25 26 28 30 31 33 34 35 39 40 42 44 48 49 60
62 63 65 69 70 71 72 73 75 76 78 79 82 84 86 87 88 89 90 91 95 100 101 104
105 109 110 112 113 118 119 121 122 123 126 129 132 134 135 136 138 142
143 144 147 149 150 152 153 155 157 160 164 165 166 167 169 170 172 173
178 180 183 184 185 186 190 191 192 193 196 200 201 206 207 208 214 215
218 220 221 222 226 227 229 230 231 233 234 235 238 240 241 244 245 246
248 251 253 254 255

- Oppure, in esadecimale

03 05 06 09 0b 0e 11 12 13 14 17 18 19 1a 1c 1e 1f 21 22 23 27 28 2a 2c 30 31
3c 3e 3f 41 45 46 47 48 49 4b 4c 4e 4f 52 54 56 57 58 59 5a 5b 5f 64 65 68 69
6d 6e 70 71 76 77 79 7a 7b 7e 81 84 86 87 88 8a 8e 8f 90 93 95 96 98 99 9b 9d
a0 a4 a5 a6 a7 a9 aa ac ad b2 b4 b7 b8 b9 ba be bf c0 c1 c4 c8 c9 ce cf d0 d6
d7 da dc dd de e2 e3 e5 e6 e7 e9 ea eb ee f0 f1 f4 f5 f6 f8 fb fd fe ff

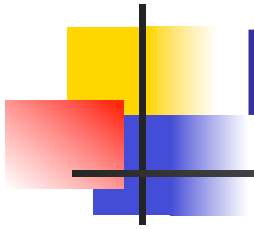
Algoritmo Inverso(a)

1. Utilizzando la tavola calcoliamo $b=\log_g a$
2. Poniamo $c=255-b$
3. L'inverso cercato e' dunque g^c.

- Adesso chiamiamo inv' la funzione inv "completata" con inv'(0)=0.

Calcolando S(0), S(1),…,S(256)… ecco cosa otteniamo

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



La funzione shift-rows

Sia $s=s_0s_1...s_{15}$ stato

shift-rows$(s_0s_1...s_{15})=s_0s_5s_{10}s_{15}s_4s_9s_{14}s_3s_8s_{13}s_2s_7s_{12}s_1s_6s_{11}$

Funzione mix-cols

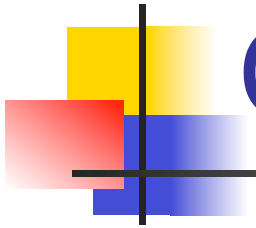
- Consideriamo la matrice ($s=s_0s_1...s_{15}$ stato)

s_0	s_4	s_8	s_{12}
s_1	s_5	s_9	s_{13}
s_2	s_6	s_{10}	s_{14}
s_3	s_7	s_{11}	s_{15}

- Definiamo mix-cols su parole di 4 bytes
- La funzione prende ciascuna delle colonne e applica la la stessa trasformazione a ciascuna di esse.

La trasformazione mix-cols

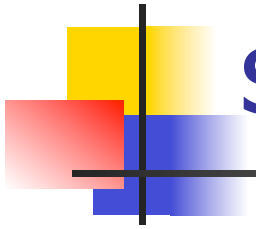
$$\begin{pmatrix} a'_0 \\ a'_1 \\ a'_2 \\ a'_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 02 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}$$



Osservazioni

- Sorprendentemente AES non contiene feistel rounds
- Abbiamo piu' informazioni disponibili (rispetto al DES).
- La sicurezza dei sistemi a blocchi considerati affidabili, tuttavia, si basa sul fatto che non siamo riusciti a romperli.

Limiti della Sicurezza basata su Key Recovery – I



- Fino ad ora abbiamo analizzato la sicurezza dei sistemi a blocchi carcando di curare la seguente domanda.
- Date q coppie msg-crittotesto, quanto difficile e' ritrovare la chiave?
- Abbiamo considerato sicuro un sistema a blocchi capace di resistere ad un tale attacco (security against key recovery)

Limiti della Sicurezza basata su Key Recovery – II

- Consideriamo il seguente cifrario a blocchi

$$E:\{0,1\}^{128} \times \{0,1\}^{256} \rightarrow \{0,1\}^{256}$$

$$E_K(M)=AES_K(M[1])||M[2]$$