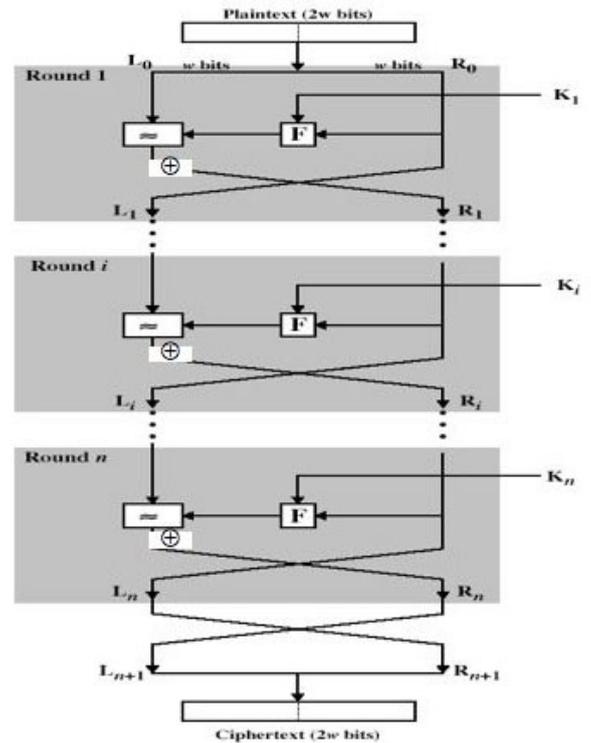


## cifrario di Feistel

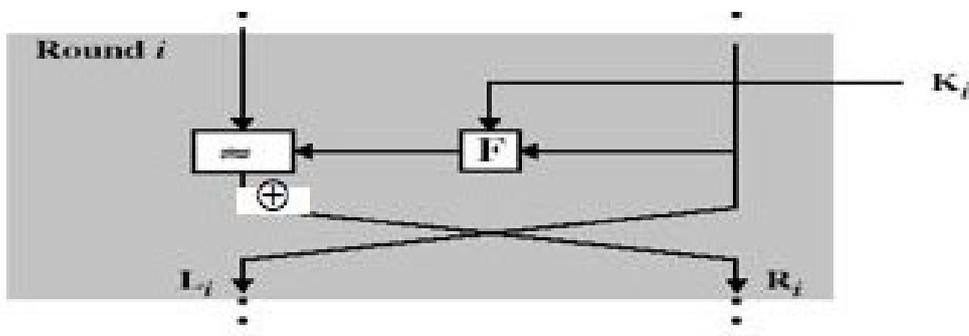
In un cifrario di Feistel, i blocchi hanno lunghezza  $m = 2w$ : a ogni iterazione, ogni stato è diviso in due blocchi di lung.  $w$ ,  $(L_i, R_i)$ . Servono  $R$  chiavi di round  $(k_1, \dots, k_R)$ .



## funzione di round

la funzione di round ha la seguente forma:

input:  $(L_{i-1}, R_{i-1}, k_i)$ , output  $(L_i, R_i)$



dove  $L_i = R_{i-1}$ ,

$R_i = L_{i-1} \oplus F(R_{i-1}, k_i)$ .

# Data Encryption Standard

Storia:

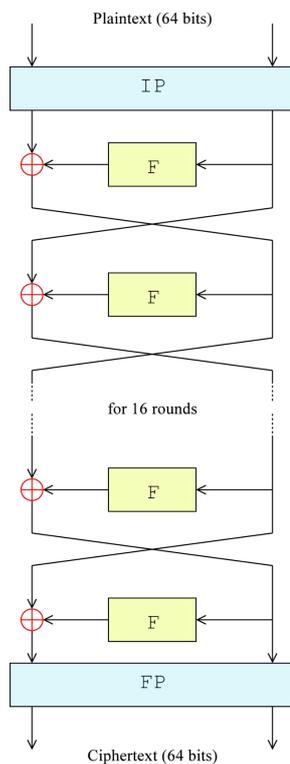
- pubblicato nel 1977 dal National Bureau of Standards (ora NIST National Institute of Standards and Technology) per uso commerciale e applicazioni del governo degli USA
- 1973-75 sviluppo del DES, dal LUCIFER (70) dell'IBM creato da Feistel e altri
- il DES sviluppato dall'IBM (Feistel, Coopersmith, Tuchman e altri)
- testato (e modificato) dall'NSA
- 75-77 il DES va al vaglio dei crittografi
- nel 1999 il NIST ha pubblicato una nuova versione chiamata triple DES (3DES) o TDEA – comunque il DES è stato molto più longevo del previsto (si credeva 10-15 anni)

Molte controversie sulla sicurezza dell'algoritmo:

si è creduto che l'NSA abbia interferito con la progettazione.

La chiave è stata portata dai 128 bit del LUCIFER a 56 bit.

Si crede(va) che ci fosse una trapdoor nella progettazione delle S-box.



Il DES consiste di:

- Una permutazione iniziale  $IP$  dei 64 bit
- 16 round identici “alla Feistel”
- Una permutazione finale, che è l’inversa di quella iniziale:  $FP = IP^{-1}$

## permutazioni

Le due permutazioni, iniziale e finale, sono l’unica cosa che rende il DES diverso da un cifrario di Feistel.

Non servono a aumentare la sicurezza, ma sono legate al processo di caricamento del plaintext nei chip.

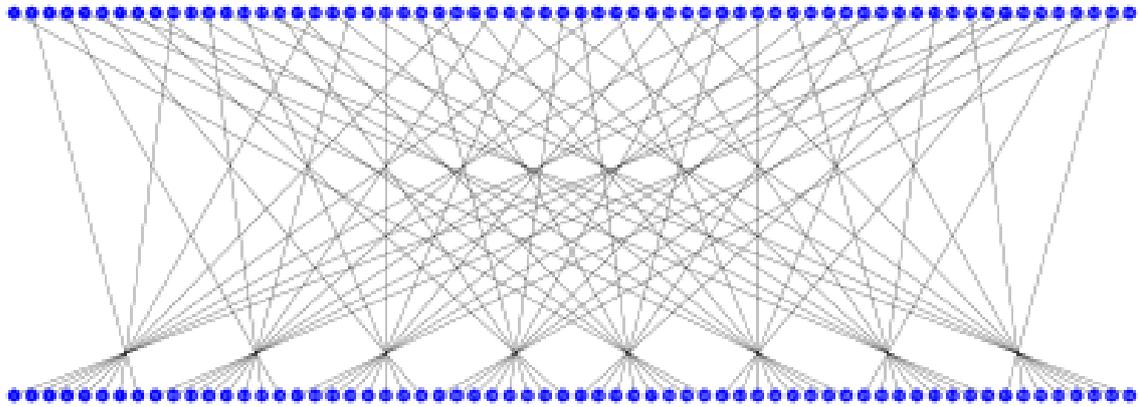
IP

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

cioè:

il primo bit in output è il 58simo dell’input;  
 il secondo bit in output è il 50simo dell’input. . .

$58 \rightarrow 1, 50 \rightarrow 2 \dots$

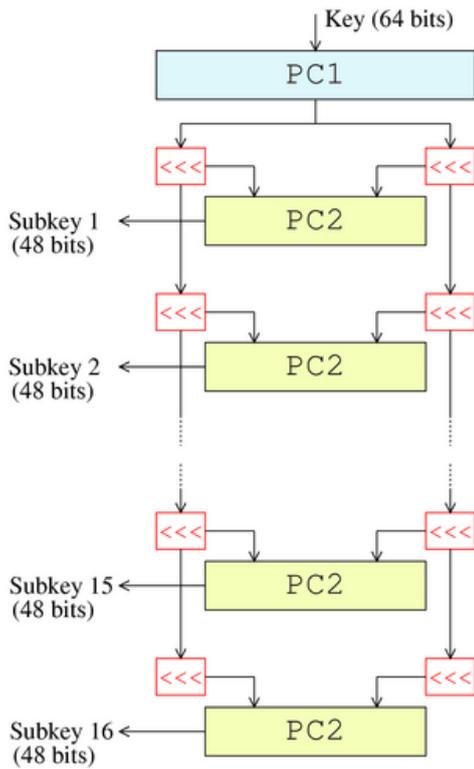


## chiave

La chiave: 64 bit, di cui 8 sono un controllo di parità. La chiave ha solo 56 bit.

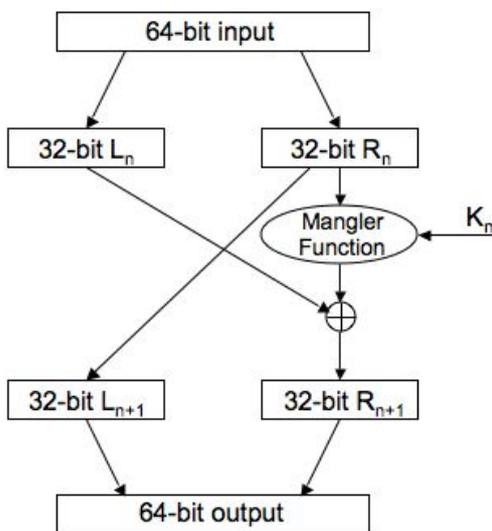
Dalla chiave  $k$  ottengo 16 chiavi di round  $(k_1, \dots, k_{16})$  di 48 bit ciascuna.

## key schedule

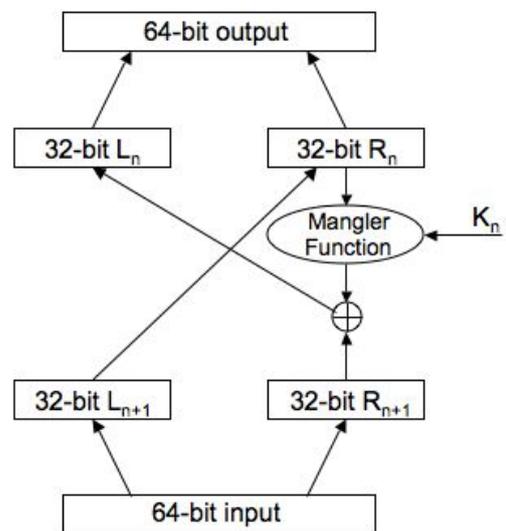


- produce le 16 chiavi di round (a 48 bit)
1. permutazione iniziale  $PC_1$  della chiave
  2. divisione dei 56 bit in 2 metà a 28 bit
  3. a ogni round
    - 3.1. Left shift ogni metà (28bit) separatamente di 1 o 2 posti (questo sarà l'input del prossimo round)
    - 3.2. riunione delle due metà, e
- permutazione/contrazione:  $PC_2$  ha 56 bit in input, output a 48 bit

## DES round



**Encryption**



**Decryption**

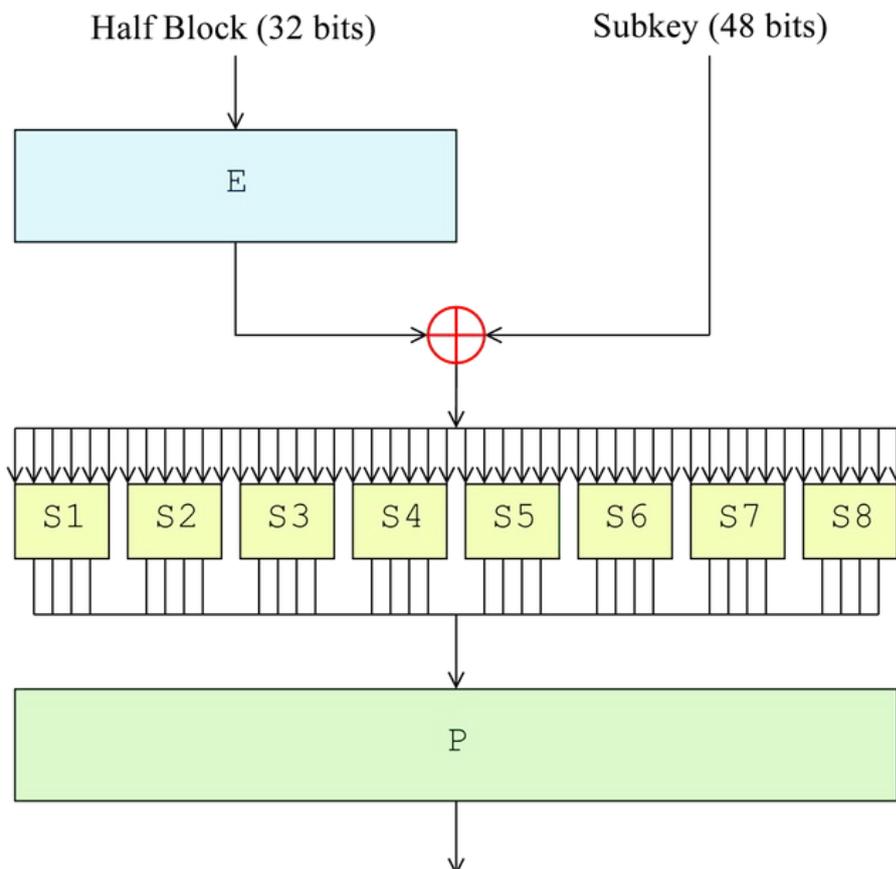
## funzione $F$ , o mangler

Dettagli della funzione  $F$ :

al round  $i$

$$F(R_{i-1}, K_i) = P(S(E(R_{i-1}) \oplus K_i))$$

- 1 Prende i 48 bit della chiave  $k_i$ .
- 2 Espande i 32 bit di destra,  $R_{i-1}$ , a 48 bit
- 3 Esegue lo XOR della chiave e dei 48 bit e invia il risultato in input alle S-Box
- 4 Le S-box sono tabelle di sostituzione e contrazione predefinite. In output danno 32 bit
- 5 Questi 32 bit vengono permutati

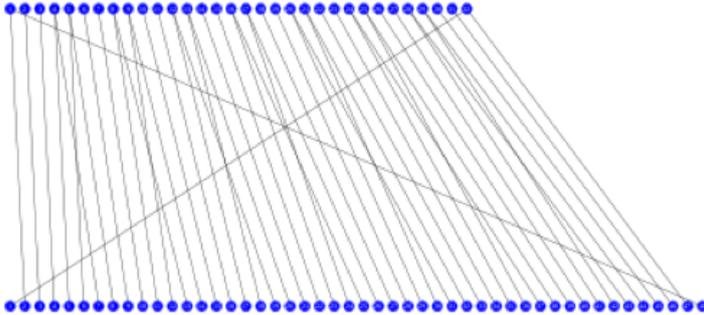


# espansione E

E

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

$1 \rightarrow 2, \quad 1 \rightarrow 32$   
 $2 \rightarrow 3 \dots$   
 16 bit ripetuti  
 mai due volte in  
 un blocco da 6  
 diffusione



$S_1$

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

$S_2$

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

$S_3$

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

$S_4$

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

$S_5$

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

$S_6$

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

$S_7$

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

$S_8$

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

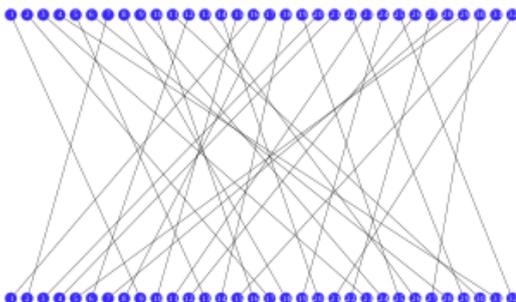
S <sub>5</sub>															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Dato un input di 6 bit, i 4 bit di output si trovano scegliendo la riga usando i due bit esterni, la colonna usando i 4 bit interni.  
 Esempio: in 011011, i bit esterni sono 01 e quelli interni sono 1101.  
 Righe e colonne sono numerate partendo da 0; l'output corrisponde al valore nella seconda riga, 14sima colonna, cioè 9.  
 Quindi 011011 → 1001

## permutazione P

P			
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

16 → 1, 7 → 2  
 i quattro bit in uscita da una S-box vengono sparsi  
 diffusione



## progettazione delle S-box

- ① 6 bit in entrata, 4 in uscita (il massimo possibile per i chip del 74)
- ② gli output NON sono una combinazione lineare degli input (né approssimabili da una funzione lineare)
- ③ le righe sono permutazioni di  $\{0, 1, \dots, 15\}$
- ④ cambiando un bit nell'input, cambiano almeno due bit in output
- ⑤  $S(x)$  e  $S(x + 001100)$  differiscono in almeno 2 bit

il ruolo di P, E, e PC2 è di spargere gli output delle S-box in modo tale che ogni bit in output dipenda da tutti i bit in input in pochi round

Il DES ha un forte *effetto valanga*: una proprietà desiderabile per un algoritmo di cifratura è che un cambiamento di un solo bit nell'input o nella chiave comporti una modifica di almeno metà dei bit di uscita. Già presente dopo poche iterazioni

Table 3.6 Avalanche Effect in **DES**: Change in Plaintext

Round		$\delta$
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402e68	33
7	9616fe2367117cf2 cf402e682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		$\delta$
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP <sup>-1</sup>	da02ce3a89ecac3b 057cde97d7683f2a	32

Table 3.7 Avalanche Effect in **DES**: Change in Key

Round		$\delta$
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		$\delta$
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeaaa	33
14	c6a62c4e56b0bd75 b9bdeaaa2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP <sup>-1</sup>	da02ce3a89ecac3b ee92b50606b62b0b	30

## crittoanalisi del DES

Di fatto, il modo più efficiente di violare il DES è un attacco a forza bruta: provare ogni possibile chiave, una dopo l'altra.

Ci sono due attacchi "a livello teorico" applicabili a ogni cifrario a blocchi

**crittoanalisi differenziale:** "scoperta" da Biham e Shamir (ca 89).

Il DES resiste molto bene alla CD: la tecnica era nota all'IBM e/o all'NSA al momento della progettazione, ed è stata tenuta segreta.

Per violare il DES con la CD, servono  $2^{47}$  chosen plaintext.

**crittoanalisi lineare:** scoperta da Matsui nel 93, migliorata via via.

Servono  $\sim 2^{41}$  known plaintext. Non ci sono prove che il DES sia stato sviluppato per resistere a questo tipo di attacco.

## attacchi a forza bruta

Lo spazio delle chiavi del DES ha grandezza  $2^{56}$ . Già al momento dell'introduzione era considerato inadeguato.

Proposte per una macchina DES-cracker

1977: Diffie-Hellmann propongono una macchina che può trovare la chiave in un giorno. Costo stimato \$ 20.000.000

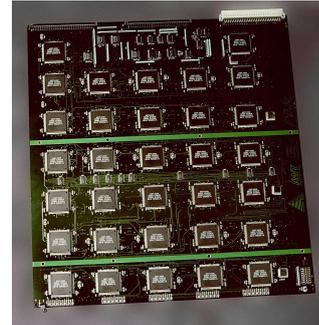
1993: Wiener, 7 ore, \$ 1.000.000

(Probabilmente, mai realizzate)

Nel 97 , la RSA Security ha sponsorizzato una gara, offrendo \$ 10.000 ai primi a decrittare un messaggio cifrato con il DES. La gara è stata vinta dal progetto DESCHALL, usando il calcolo distribuito.

There are many people who will not believe a truth until they can see it with their own eyes. Showing them a physical machine that can crack DES in a few days is the only way to convince some people that they really cannot trust their security to DES.

Nel 98, la EFF (Electronic Frontier Foundation) ha costruito una macchina al costo di circa \$ 250.000. La macchina ha trovato una chiave in 2 giorni circa.



COPACOBANA (2006, univ. di Bochum e Kiel) - \$ 10.000

## rafforzare il DES

- L'uso del DES è molto diffuso. Una volta che si è rivelato non più sicuro si può
  - passare a un algoritmo migliore, completamente diverso (AES)
    - può essere costoso
  - trovare il modo di rafforzare la sicurezza del DES
- tramite due iterazioni: si usano due chiavi  $(k_1, k_2)$ , e la cifratura è

$$e_{k_2}(e_{k_1}(x))$$

- Questo per il DES porta il numero di possibili chiavi a  $2^{112}$  (basta)

- ma attenzione - bisogna essere sicuri che il DES non sia idempotente!
- la cifratura non deve essere chiusa rispetto alla composizione
- se per ogni coppia di chiavi  $(k_1, k_2)$  esiste una chiave  $k_3$  con

$$e_{k_2}(e_{k_1}(x)) = e_{k_3}(x),$$

non c'è nessun miglioramento: due (o più) iterazioni non servono a niente

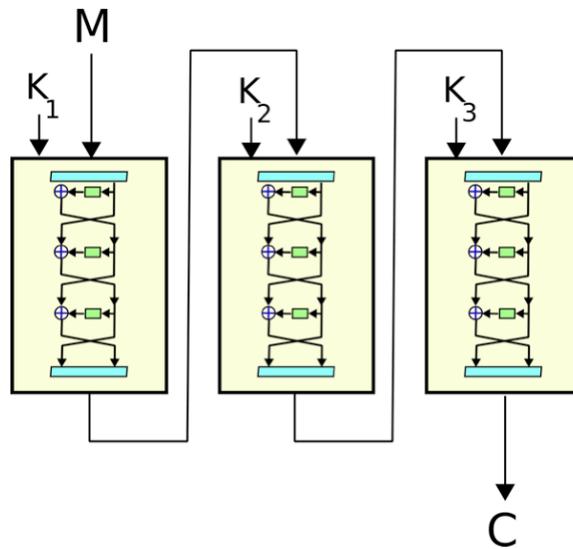
- Già osservato che i cifrari a sostituzione hanno questa proprietà - la composizione di due sostituzioni è una sostituzione (di più: formano gruppo)
- Le cifrature DES non sono chiuse rispetto alla composizione, né formano gruppo.
- Questo non è stato chiaro per molto tempo. Dimostrato solo nel '92 (Campbell e Wiener)

## meet-in-the-middle

- si può mostrare che due iterazioni non bastano - il DES a due iterazioni è vulnerabile a un attacco meet-in-the-middle
- funziona per ogni cifrario iterato due volte - ha bisogno di spazio
- MIM attack: known plaintext: so che  $e_{k_2}(e_{k_1}(x)) = y$
- calcolo  $e_k(x)$  per ogni chiave  $k$  (e memorizzo i risultati)
- brute force  $d_{k'}(y)$  per ogni chiave  $k'$
- ogni corrispondenza può rivelare le due chiavi - per accertarsene, si testano altre coppie

# triplo DES

Servono tre iterazioni: triplo DES



- il triplo DES può avere uno schema EDE
- $y = e_{k_3}(d_{k_2}(e_{k_1}(x)))$
- oppure uno schema EEE  $y = e_{k_3}(e_{k_2}(e_{k_1}(x)))$

- schema EDE

- 2-DES: con due chiavi e tre iterazioni:

$$y = e_{k_1}(d_{k_2}(e_{k_1}(x)))$$

- 3-DES: con tre chiavi e tre iterazioni:

$$y = e_{k_3}(d_{k_2}(e_{k_1}(x)))$$

- Il vantaggio dell'EDE è la compatibilità col DES ordinario:  
con  $k_1 = k_2 = k_3$ , 3-DES=DES
- è ancora uno standard NIST, ma è lento!
- Sempre più spesso rimpiazzato dall'AES.