



# Data Encryption Standard



***Alessandro Ballini***  
***Silvio Donnini***  
***Roberto Pariset***  
***Alberto Pettini***

# DES, breve storia

- 1973: Il National Bureau of Standards (NBS) pubblica un bando in cui richiede un algoritmo di cifratura:
  - La cui sicurezza risiedesse nella segretezza della chiave e non nel processo di cifratura
  - Che potesse essere realizzato efficientemente a livello hardware
- IBM propone una prima versione del DES
- NSA (National Security Agency) lo certifica ma propone delle variazioni:
  - Riduzione della lunghezza della chiave, da 128 bit a 56 bit
  - Modifica delle funzioni contenute nelle S-box

# DES, breve storia

- 1977: DES viene accettato e reso pubblicamente disponibile.
  - Primo cifrario certificato ufficialmente come sicuro e noto a tutti.
  - Certificato ufficialmente ogni 5 anni
- 1987: Prime manifestazioni di sfiducia sul DES
- 1998: Il DES viene rotto
- 2000: NBS sceglie il successore del DES, denominato Advanced Encryption Standard (AES).

# IL DES

cifrario simmetrico a blocchi

## ■ I cifrari a blocchi

- Il messaggio viene suddiviso in blocchi di n bit
- I blocchi vengono cifrati indipendentemente l'uno dall'altro.

## ■ I cifrari simmetrici:

- Cifratura e decifrazione con la stessa chiave.

# Le basi del DES

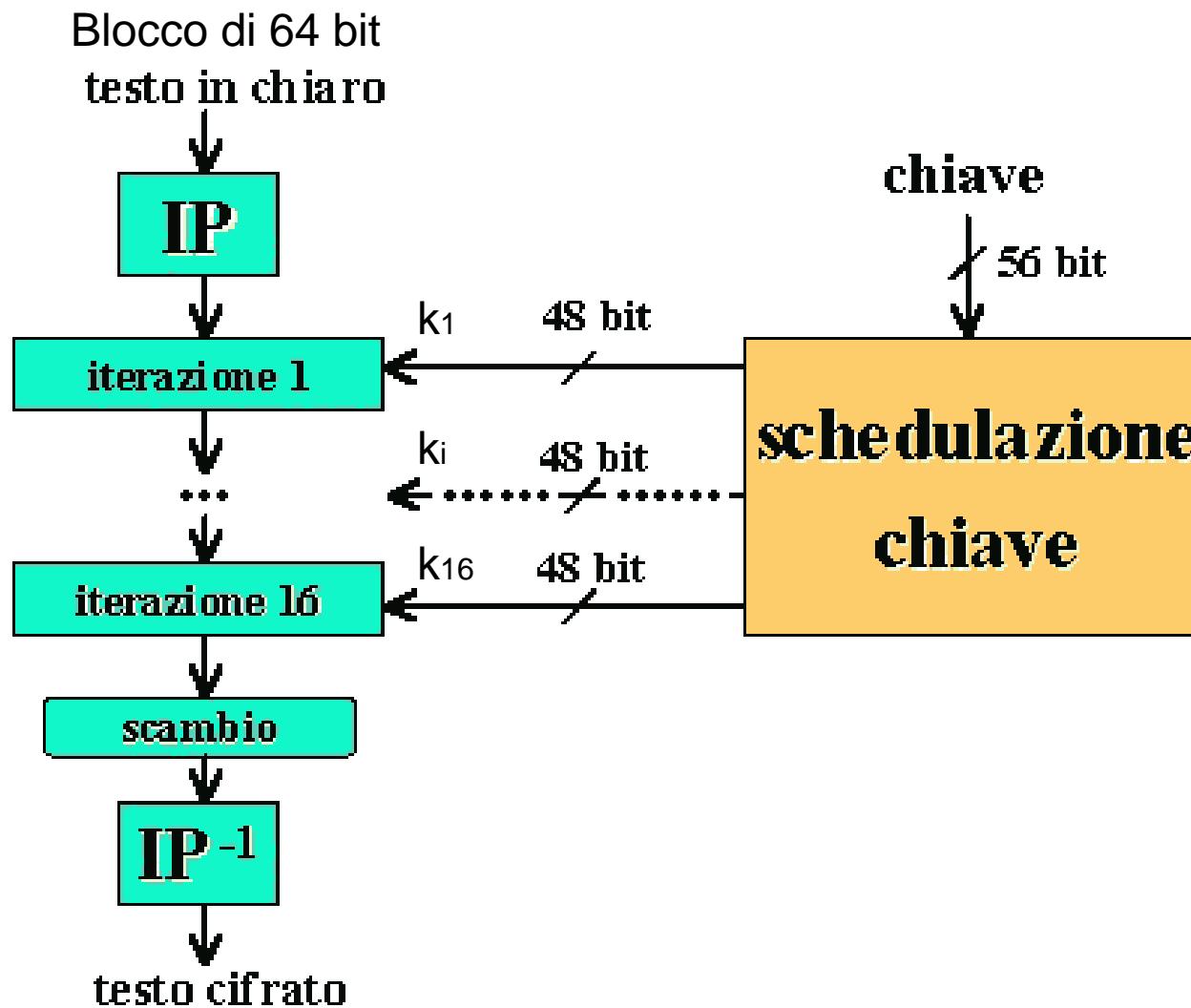
Proprietà desiderabili di un algoritmo di cifratura secondo Shannon.

- Diffusione
  - Alterare la struttura del testo in chiaro “spargendo” i caratteri su tutto il testo cifrato.
- Confusione
  - Combinare in modo complesso il messaggio e la chiave, per non permettere al crittoanalista di separare le due sequenze mediante l’analisi del crittogramma.

Nel DES diffusione e confusione sono soddisfatte attraverso una serie di permutazioni e combinazioni del messaggio con la chiave.

# La Struttura

## ■ Ad alto livello



# La Struttura

- Un blocco di testo in chiaro (64 bit) viene permutato dal blocco IP, il risultato di questa permutazione andrà in ingresso alla prima iterazione.
- In ogni iterazione il blocco di 64 bit viene opportunamente mescolato con una Kiesima chiave.
- In tutto 16 iterazioni.
- Dopo le 16 iterazioni al testo di 64 bit oramai cifrato viene applicata la permutazione iniziale inversa  $IP^{-1}$

# IP : permutazione iniziale

Blocco in ingresso

0	1	0	1	0	0	1	1
0	1	0	0	0	1	0	1
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	1
0	1	0	1	0	0	1	0
0	1	0	0	1	0	0	1
0	0	1	0	1	0	0	0
0	1	0	1	1	0	0	1

Permutazione iniziale

58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

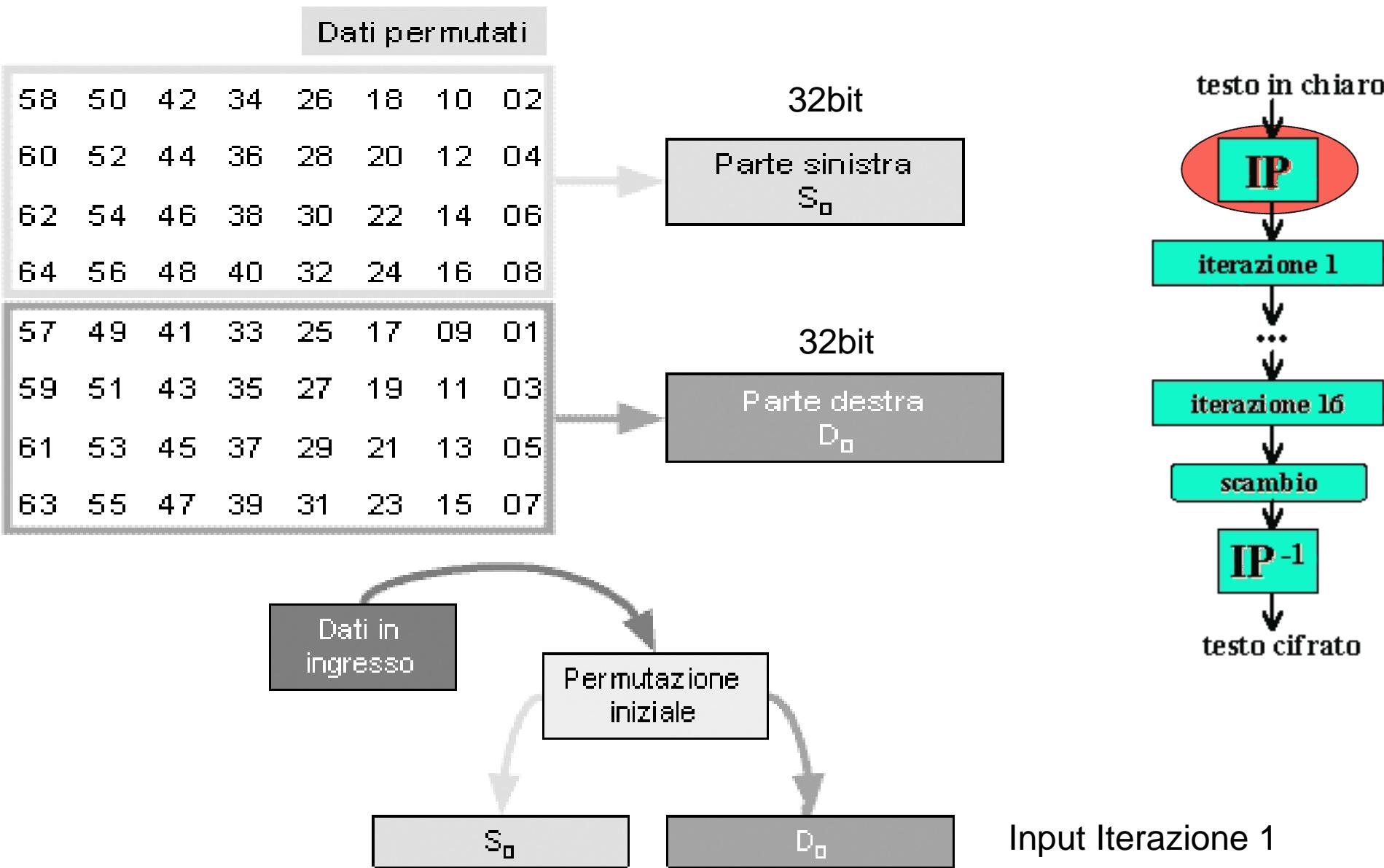
bit iniziali



bit permutati



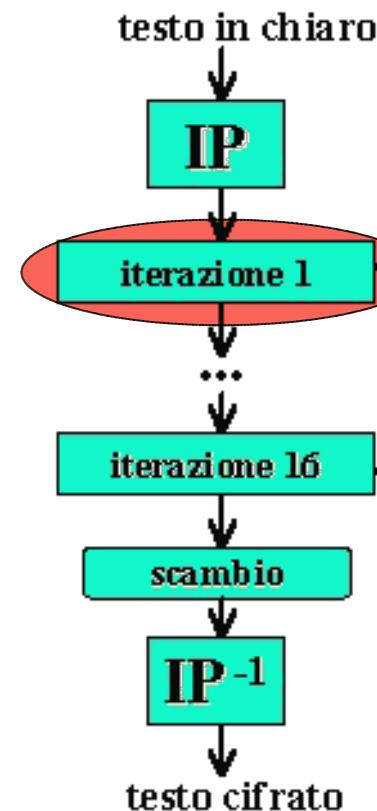
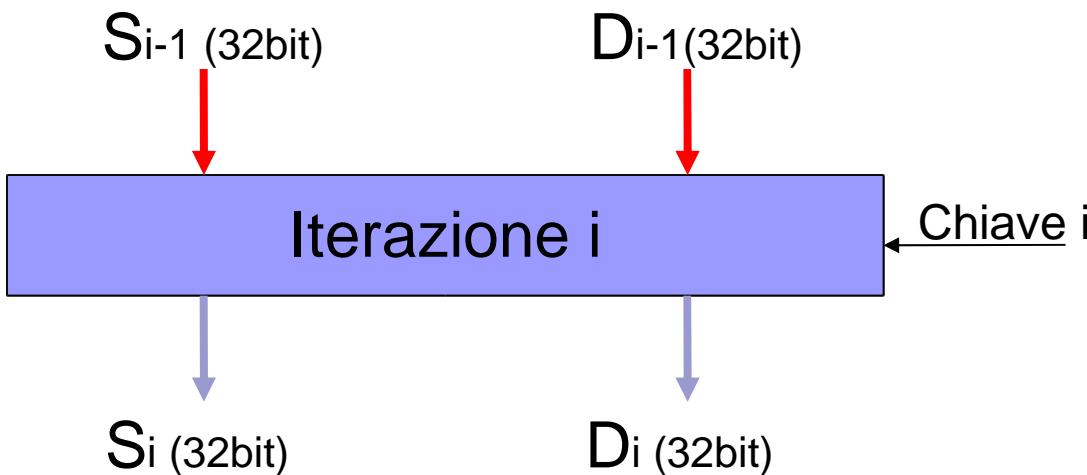
# IP



# IP : permutazione iniziale

- Nella permutazione iniziale i bit vengono scambiati in base alla matrice di permutazione IP, ovvero in posizione [1,1] andrà il bit in posizione 58 del testo in chiaro, in seconda posizione il bit 50 e così via.
- Al termine della permutazione il blocco viene suddiviso in due sottoblocchi da 32 bit ciascuno  $S_0$  e  $D_0$ , questi saranno l'input della prima iterazione.

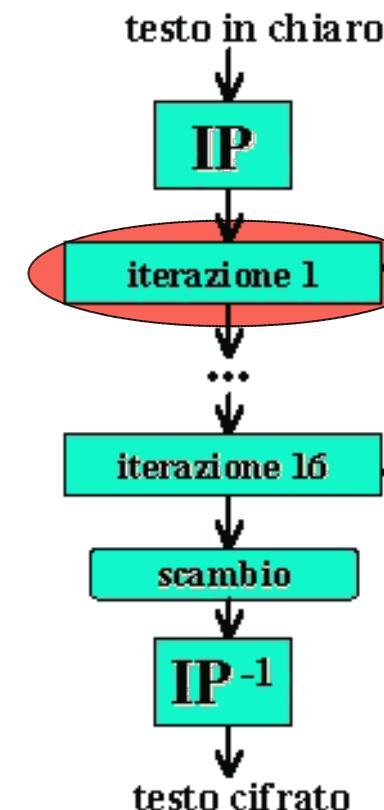
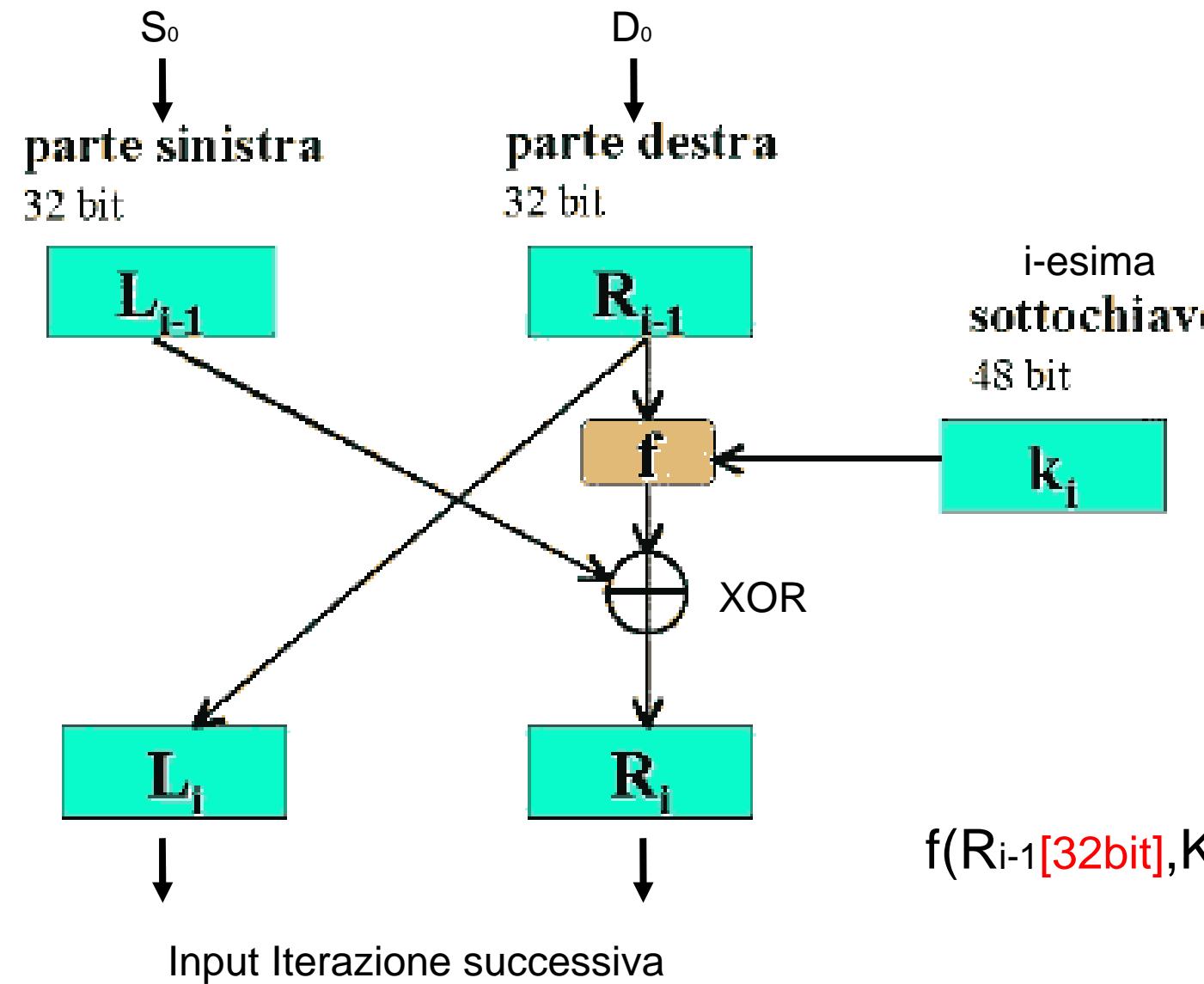
# Iterazione



# Iterazione

- Ogni Iterazione ha la seguente struttura :
  - Input : due blocchi  $S_{i-1}$  e  $D_{i-1}$  da 32bit, una chiave  $K_i$  da 48 bit.
  - Elaborazione : combinazione di  $S_{i-1}$  e  $D_{i-1}$  con  $K_i$ .
  - Output : due nuovi blocchi  $S_i$  e  $D_i$  da 32 bit.

# Iterazione

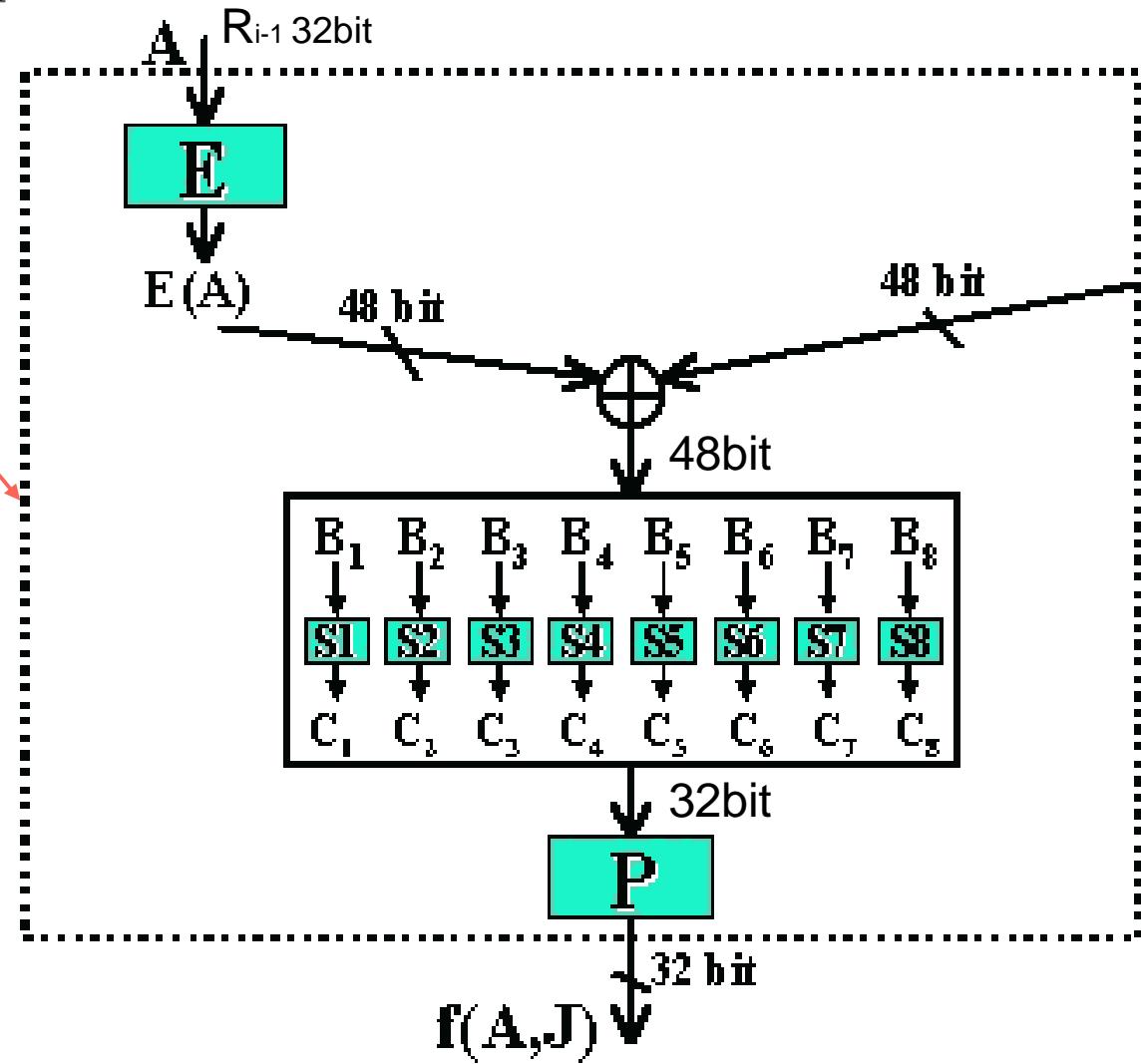
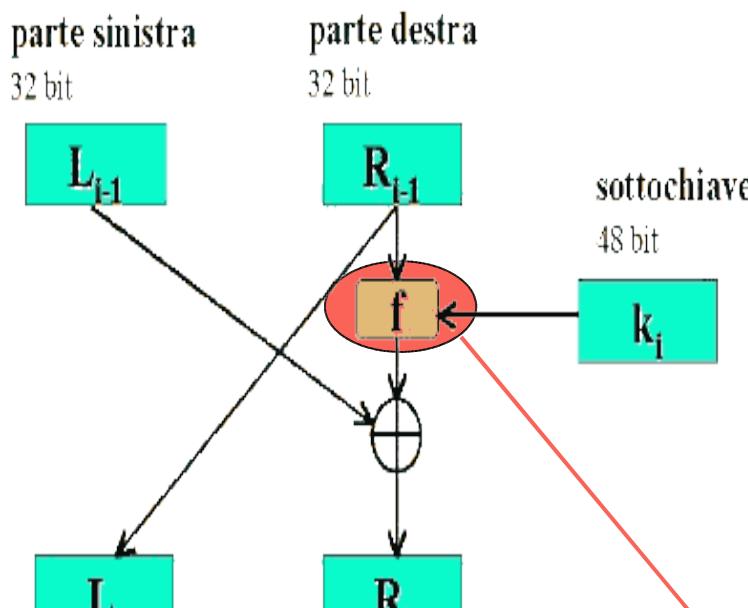


# Iterazione

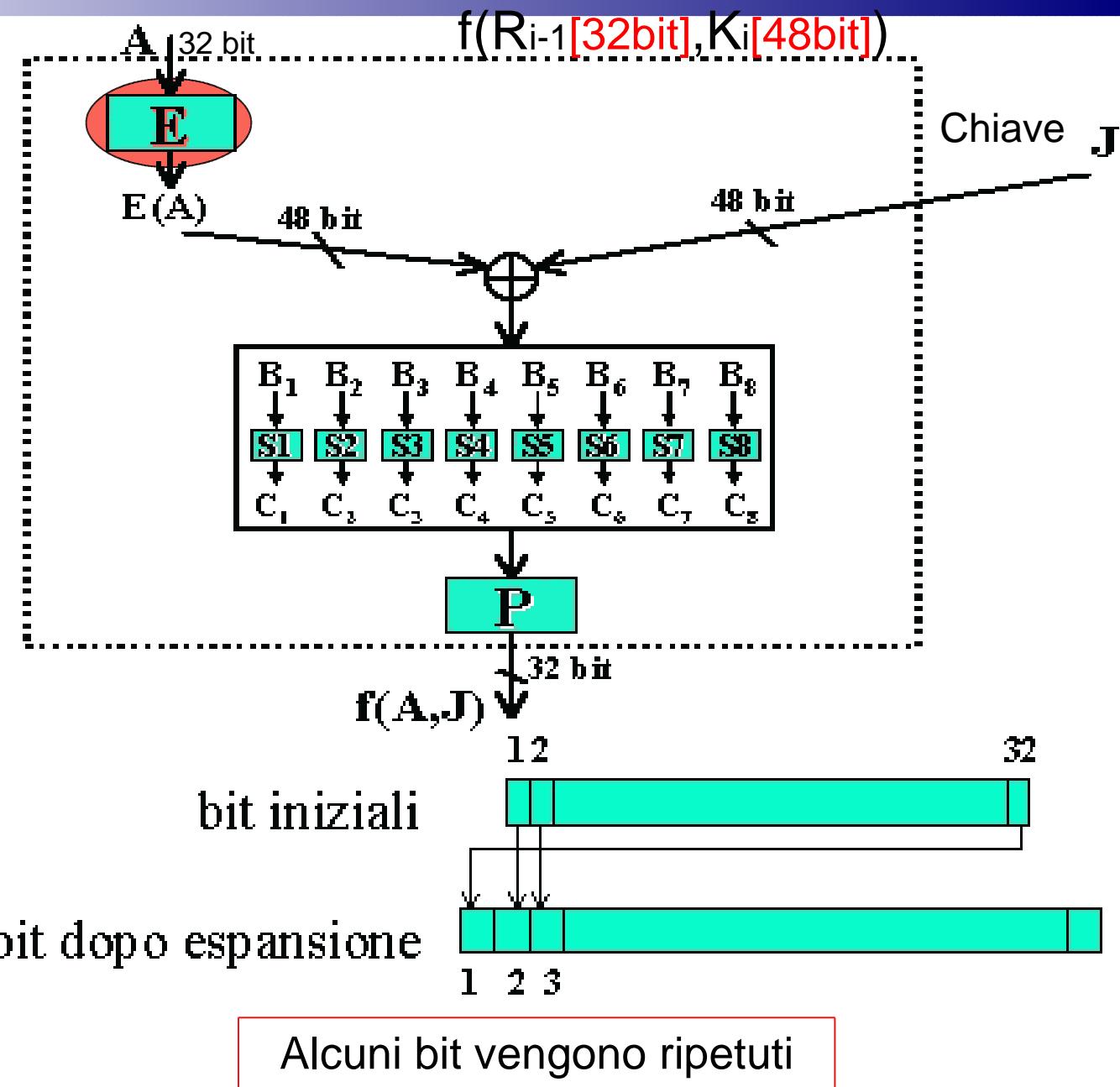
- Elaborazione dei blocchi nell'iterazione.
  - I due blocchi di Output  $L_i$  e  $R_i$  sono ottenuti in questo modo :
    - Il nuovo blocco  $L_i$  è il vecchio  $R_{i-1}$
    - Il nuovo blocco  $R_i$  è ottenuto mettendo in XOR il vecchio blocco  $L_{i-1}$  con l'output della funzione  $F$
  - La Funzione  $F$  combina il vecchio blocco di destra  $R_{i-1}$  con la chiave  $K_i$

# La Funzione f

$$f(R_{i-1}[32\text{bit}], K_i[48\text{bit}])$$



# II Blocco E



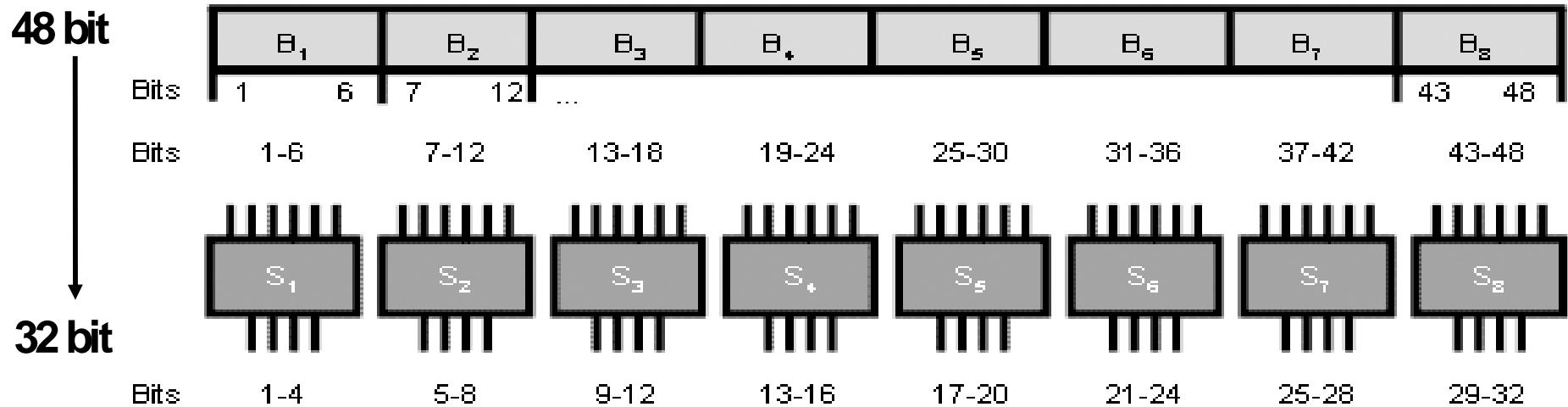
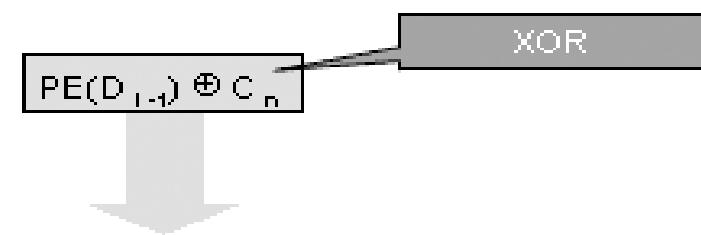
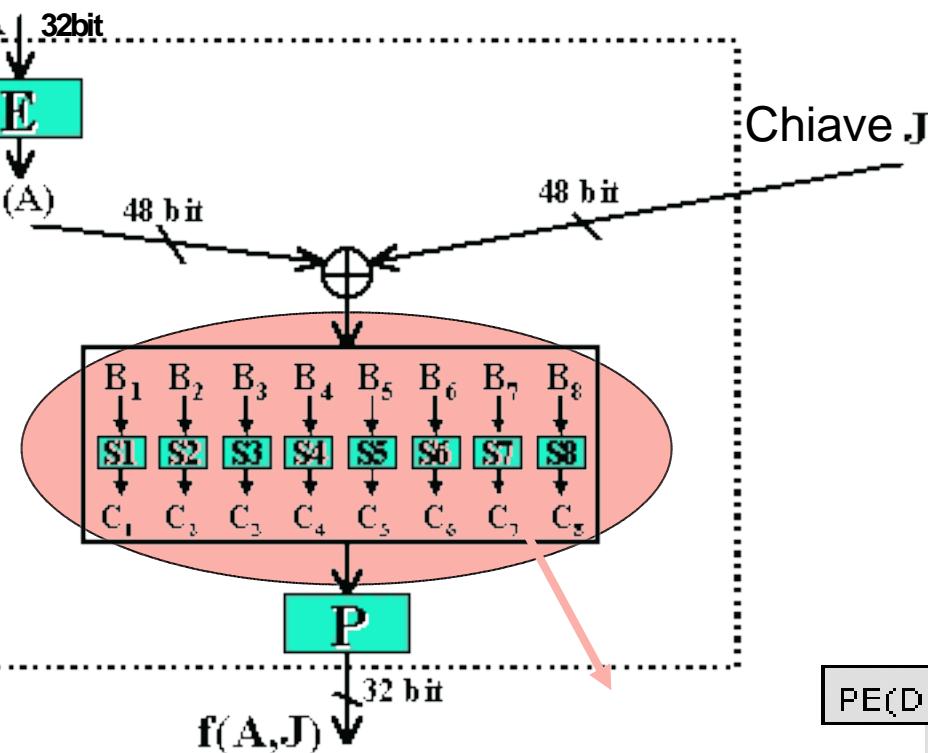
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

# Funzione F

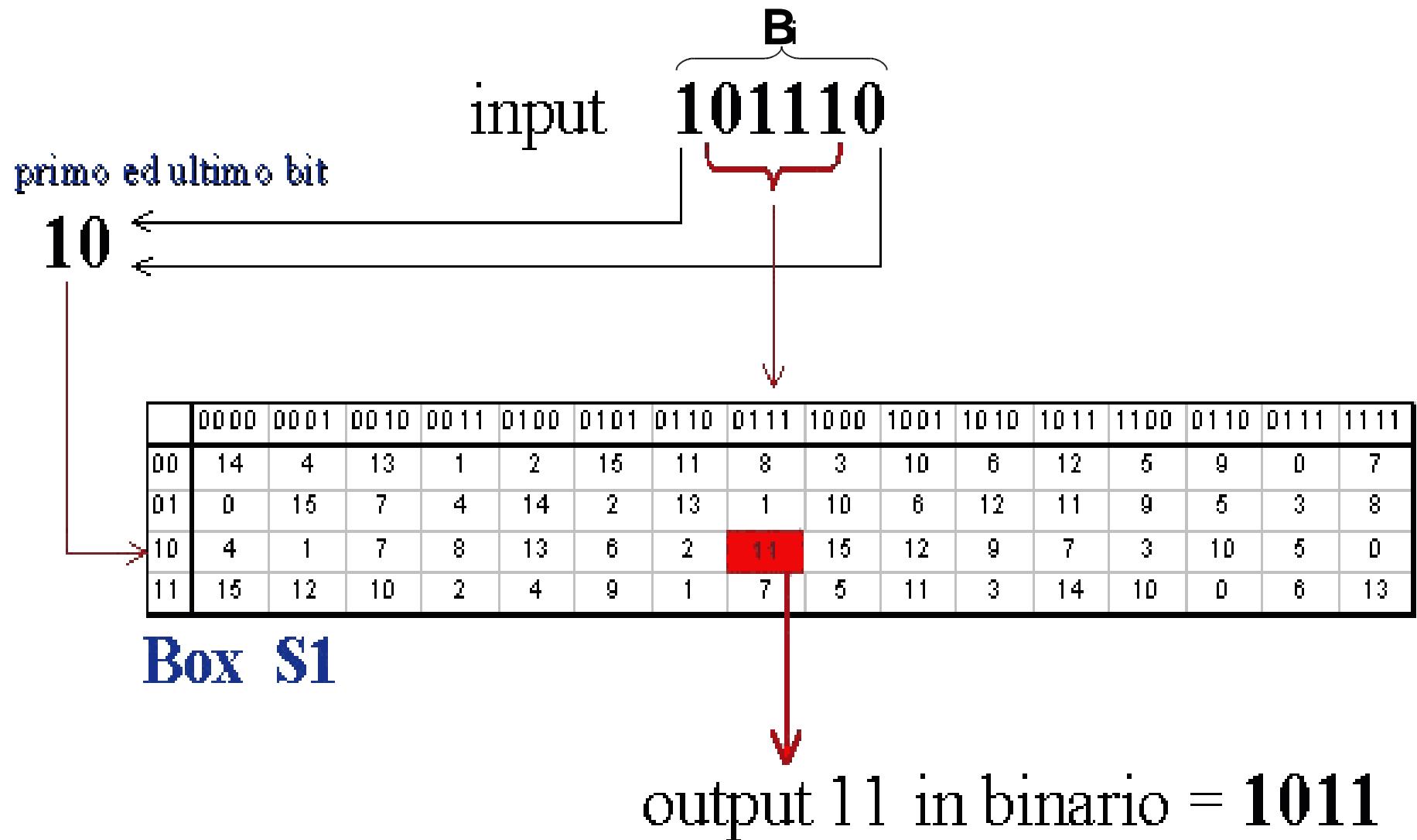
- Inizialmente i 32 bit del vecchio blocco di destra  $R_{i-1}$  vengono espansi a 48 bit dal blocco E mediante la matrice E, semplicemente alcuni bit vengono ripetuti.
- I 48 bit ottenuti dall'espansione vengono messi in XOR con la  $K_i$  chiave anch'essa di 48 bit.
- La nuova sequenza B di 48 bit ottenuta viene suddivisa in 8 sottoblocchi da 6 bit ciascuno.
- Le stringhe  $B_i$  di 6 bit vengono date in ingresso alle otto funzioni SBOX.

# Le S-Box

Parte cruciale su cui si basa la sicurezza del cifrario



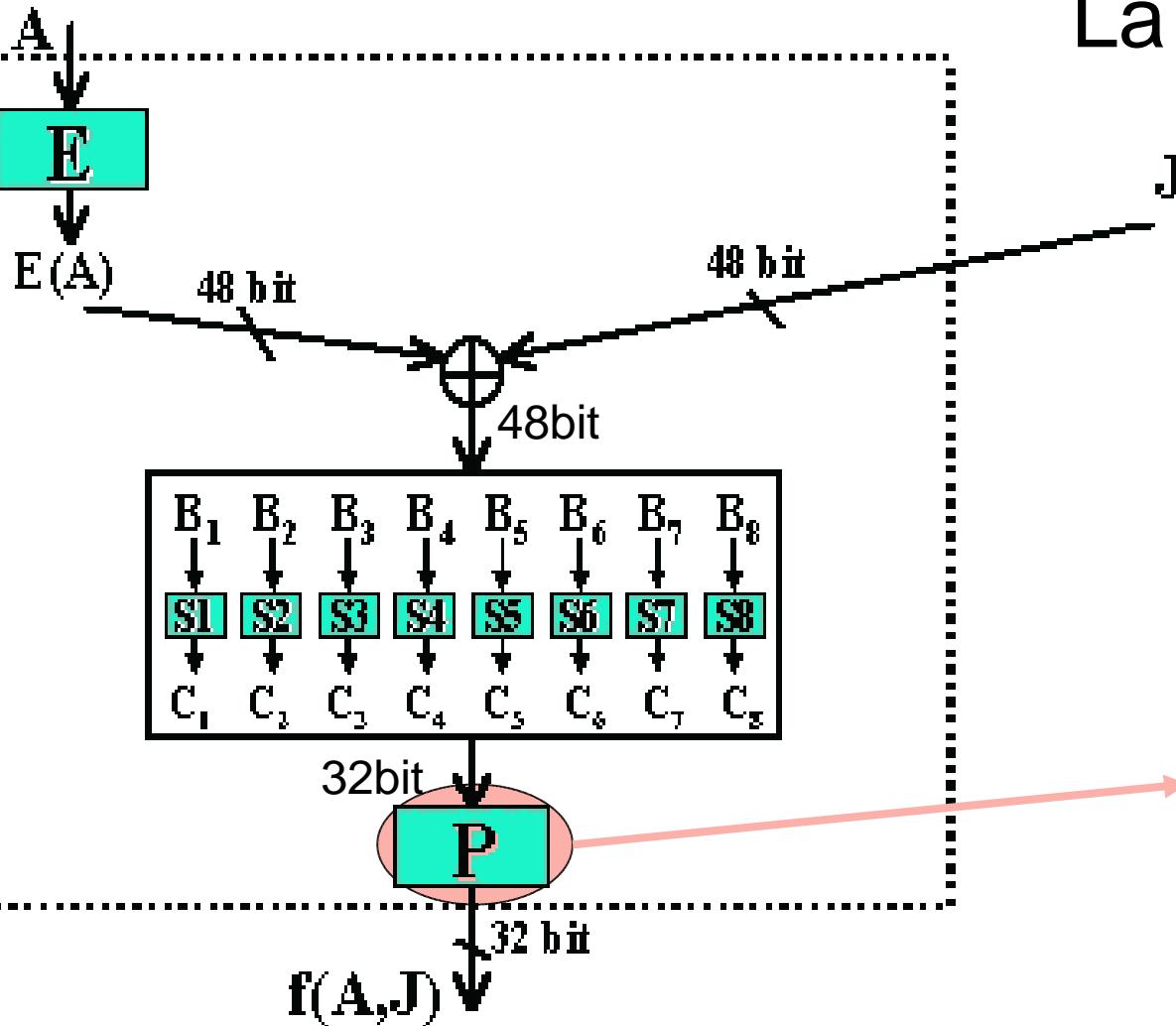
# Le S-Box



# SBOX

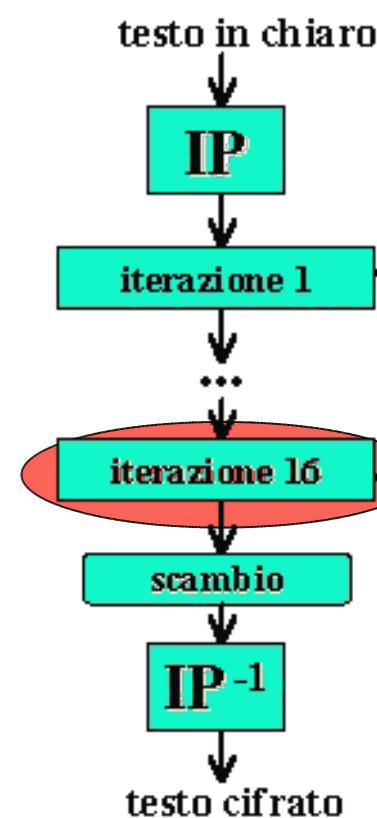
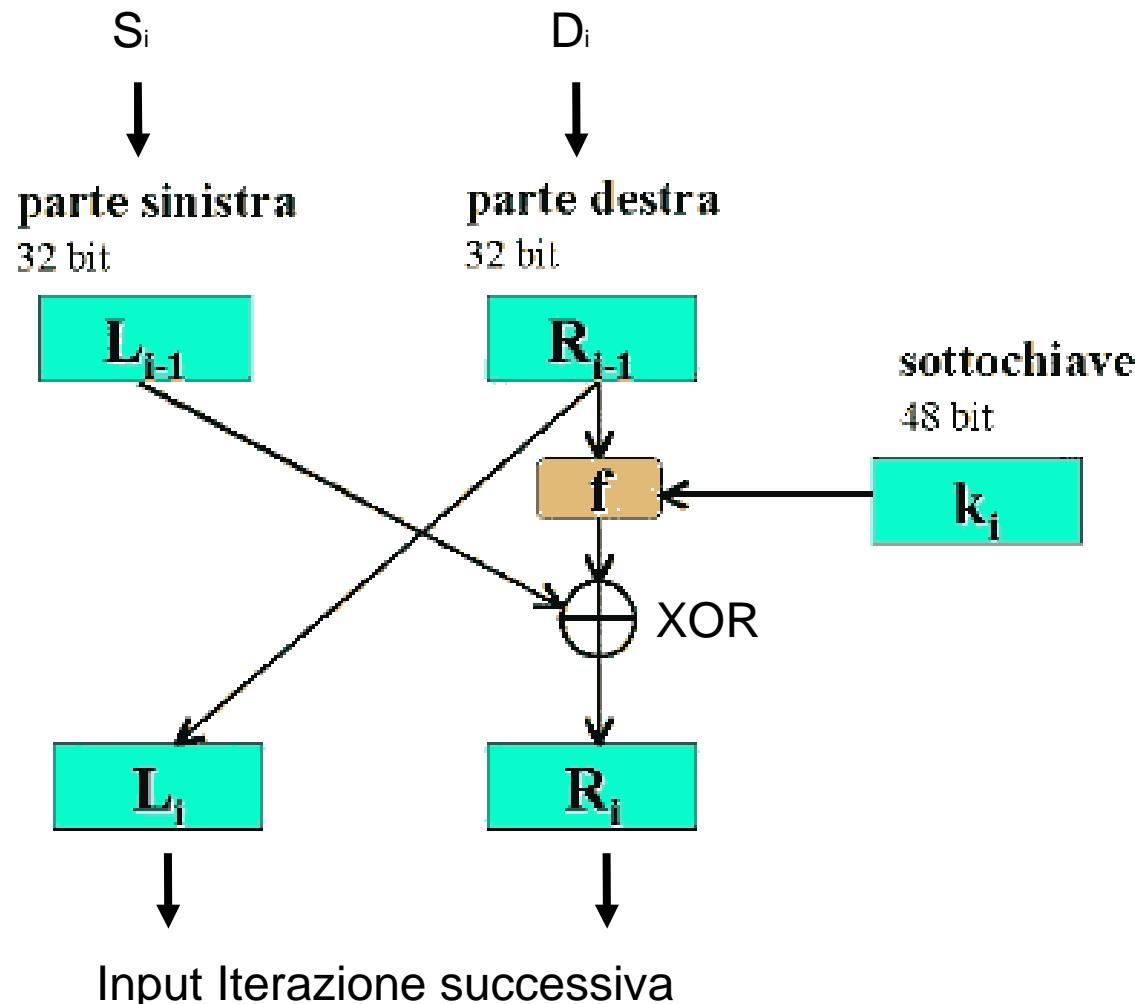
- Le SBOX hanno il compito di trasformare un blocco in ingresso Bi di 6 bit in un blocco in uscita di 4 bit.
- Dalla figura nella slide precedente possiamo vedere come ciò avviene: dal primo e l'ultimo bit del blocco Bi in input ricaviamo l'indice di riga della SBOX, dai 4 bit centrali restanti invece ricaviamo l'indice di colonna.
- Otteniamo così l'elemento  $S[i,j]=11$  ; trasformandolo in binario otteniamo i 4 bit di output.
- Mettendo in sequenza gli output delle otto SBOX otteniamo la stringa di 32bit, questa verrà sottoposta ad un'ultima permutazione P.

# La permutazione P



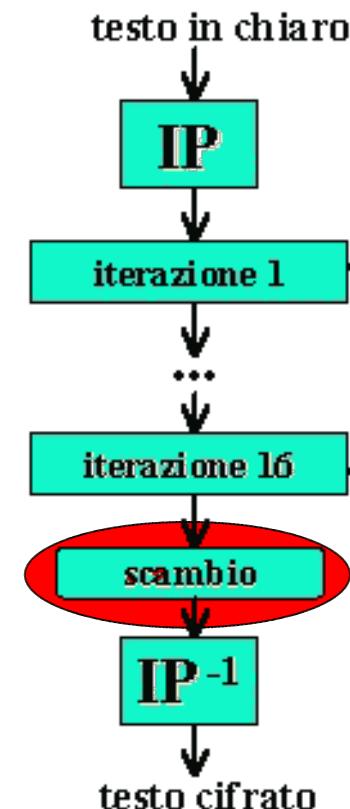
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

# Iterazione

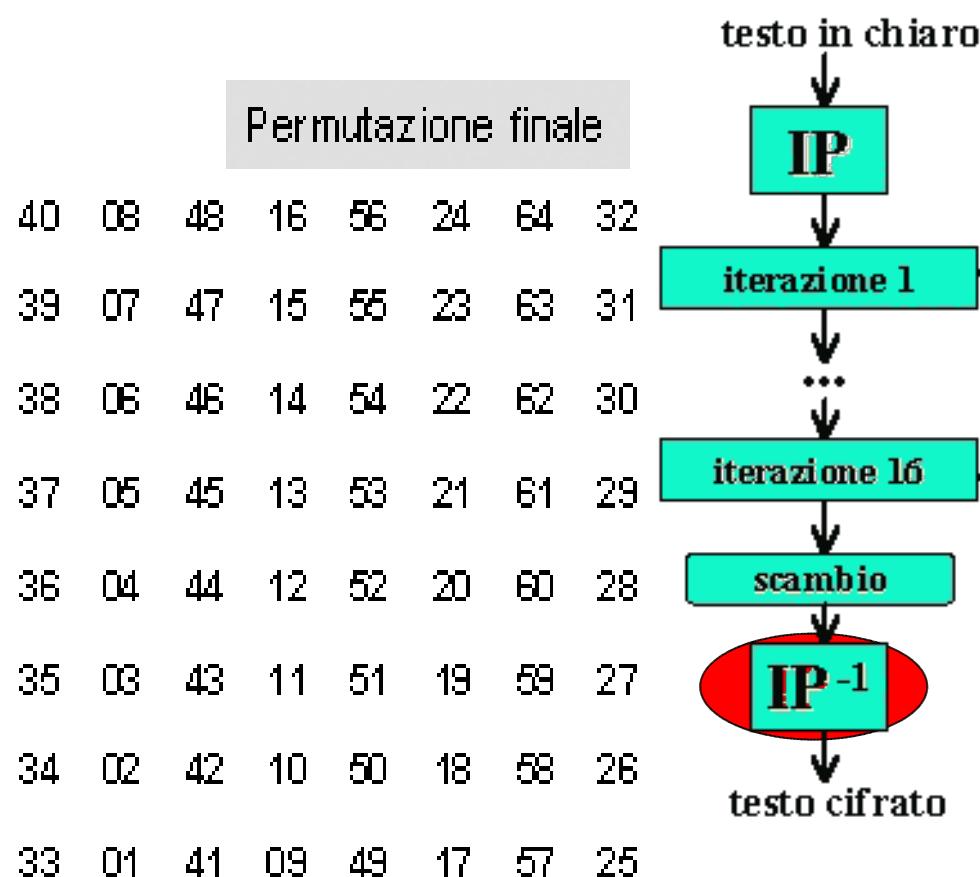
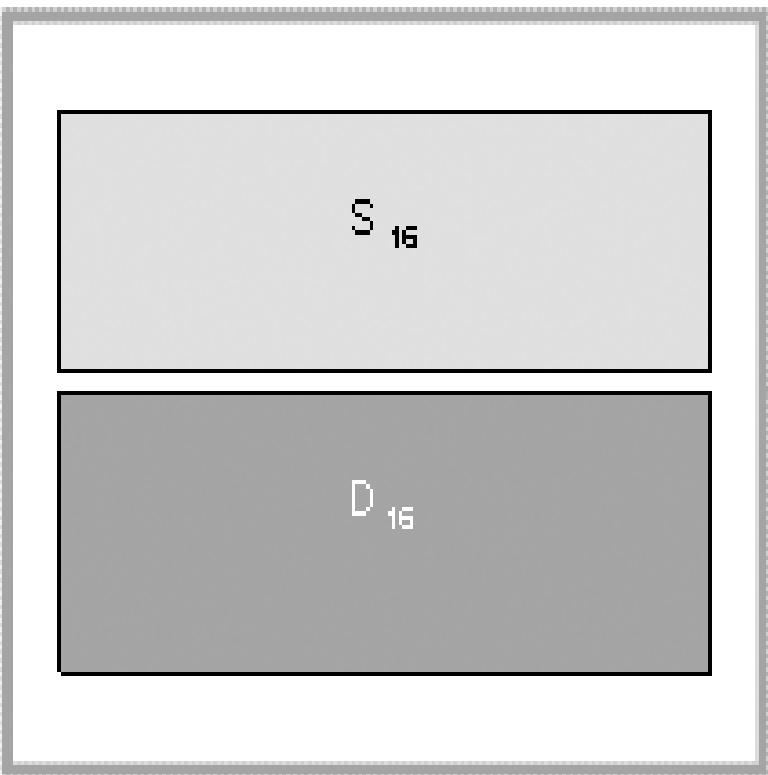


# Scambio

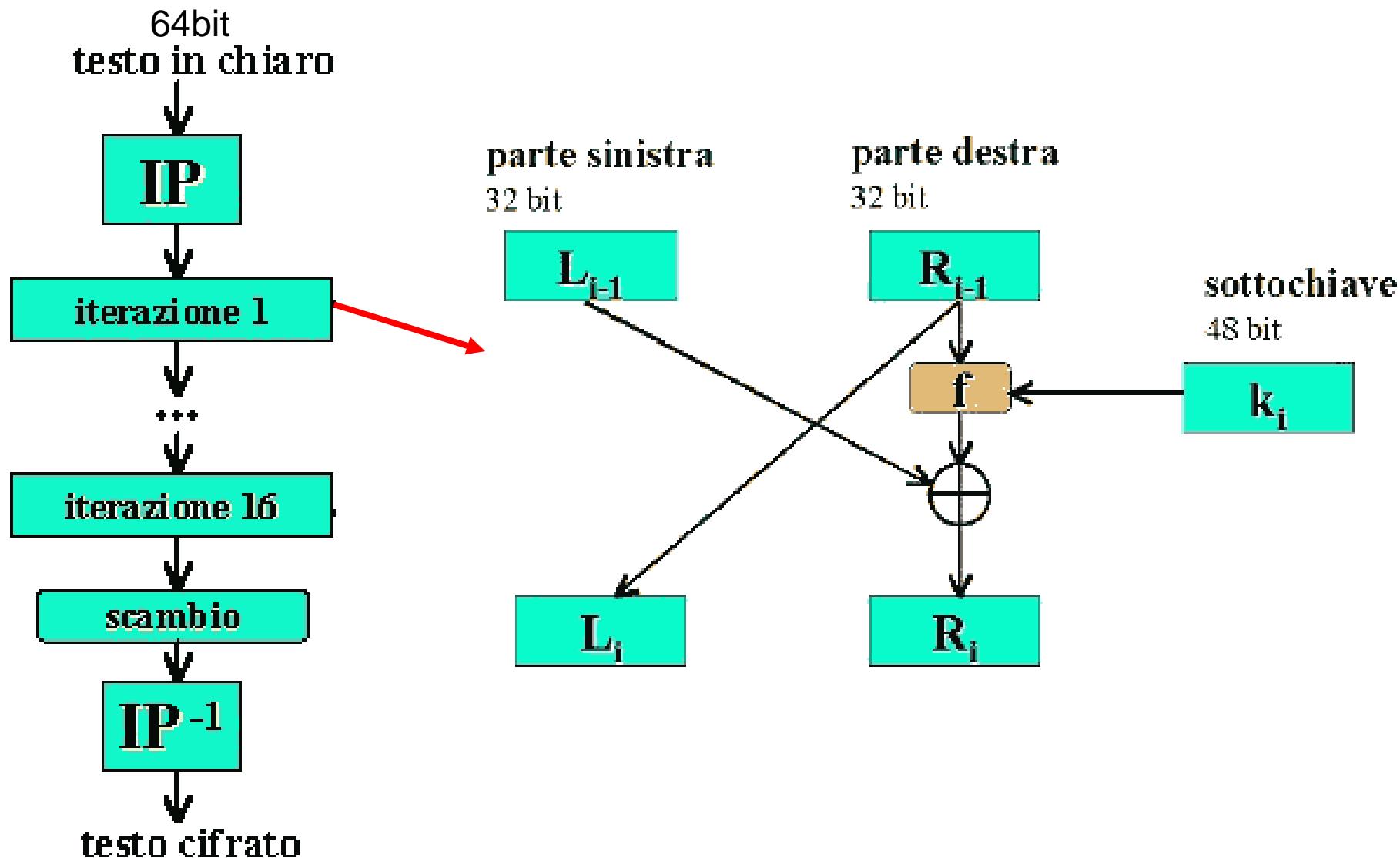
- Le parti  $L_{16}$  ed  $R_{16}$  vengono invertite
- In questo modo il blocco viene predisposto per il processo di decifrazione, come vedremo in seguito.



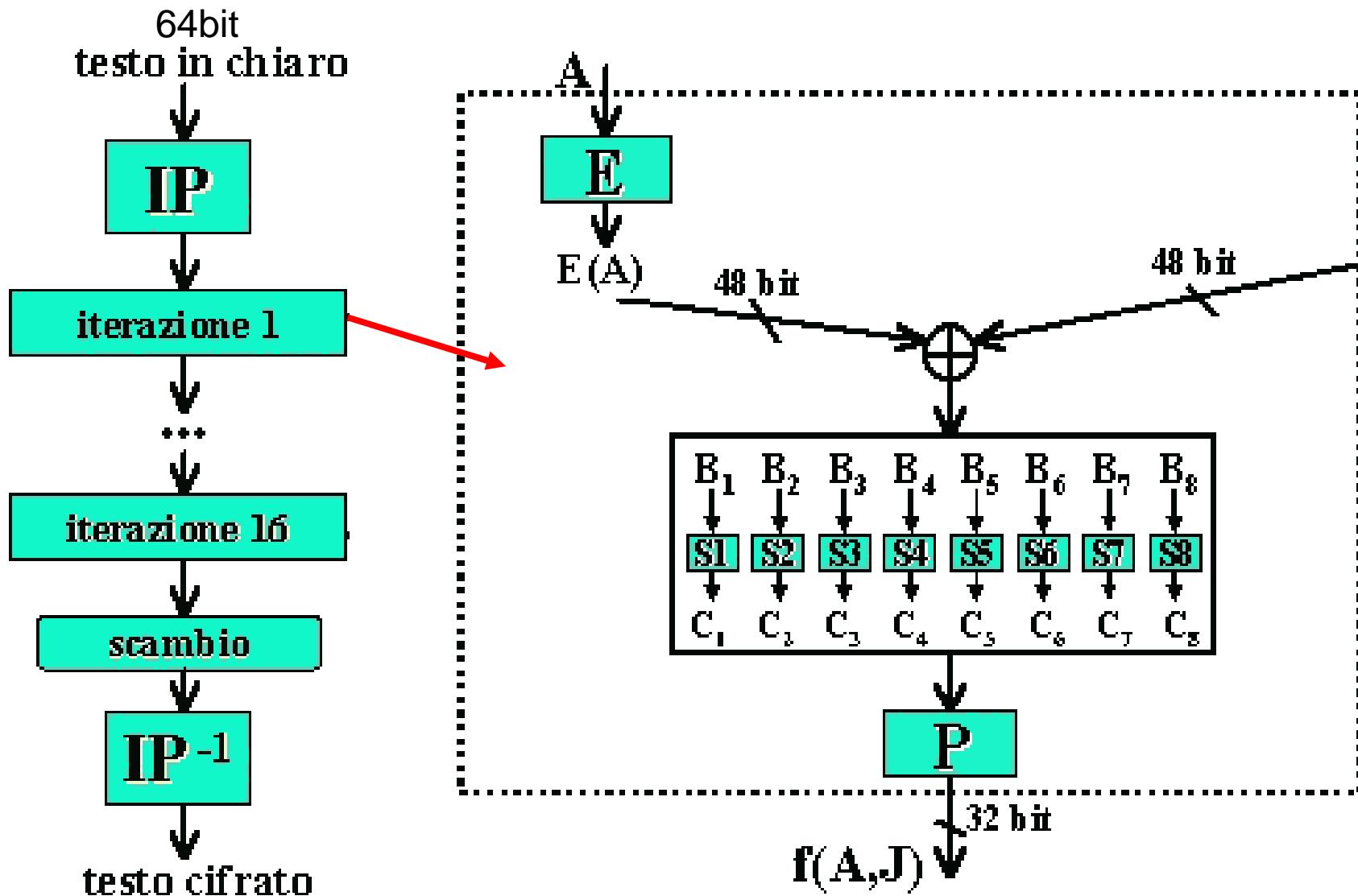
# IP 1



# Ricapitolando...



# Ricapitolando...

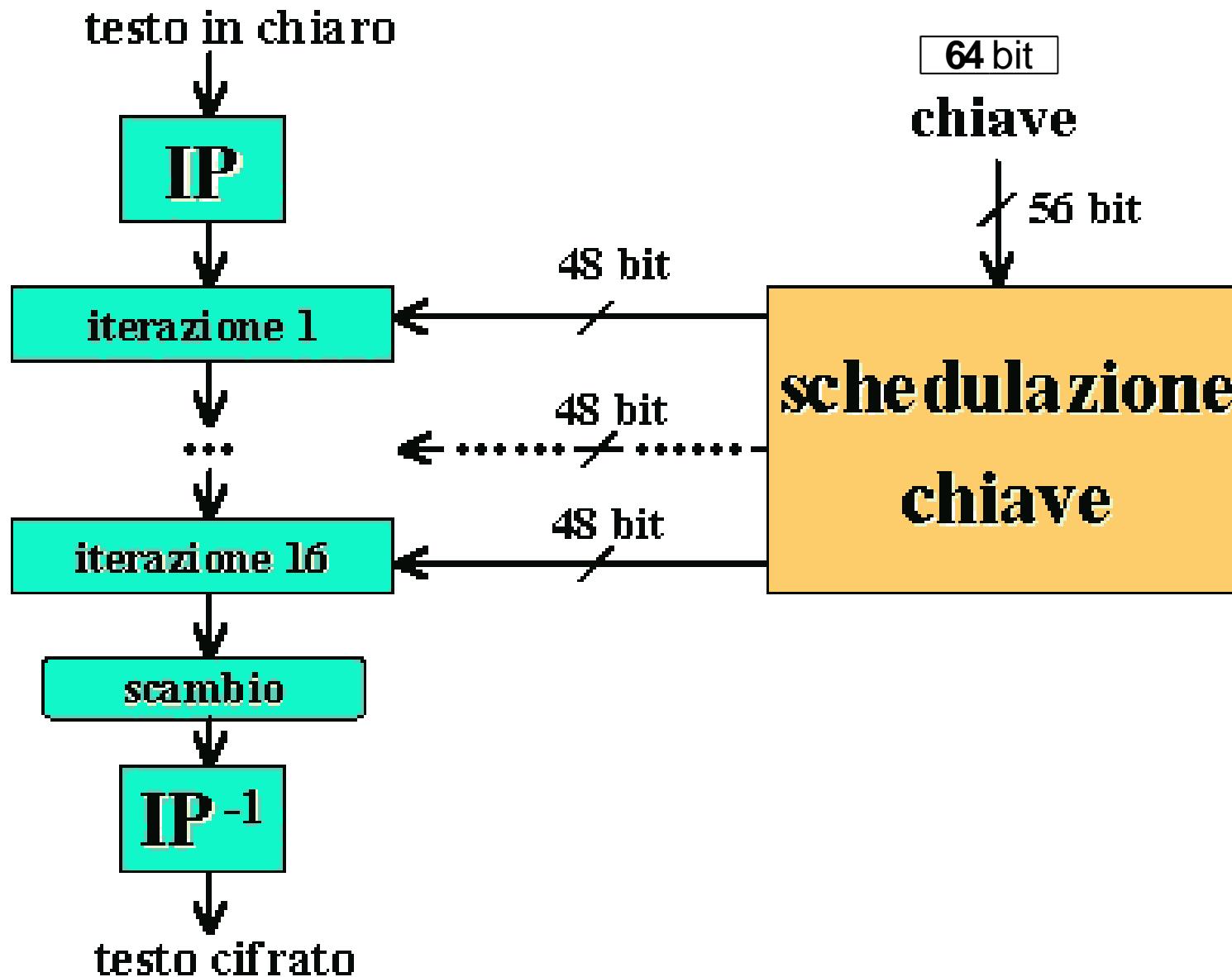




# Generazione delle chiavi e decifrazione

Pettini Alberto

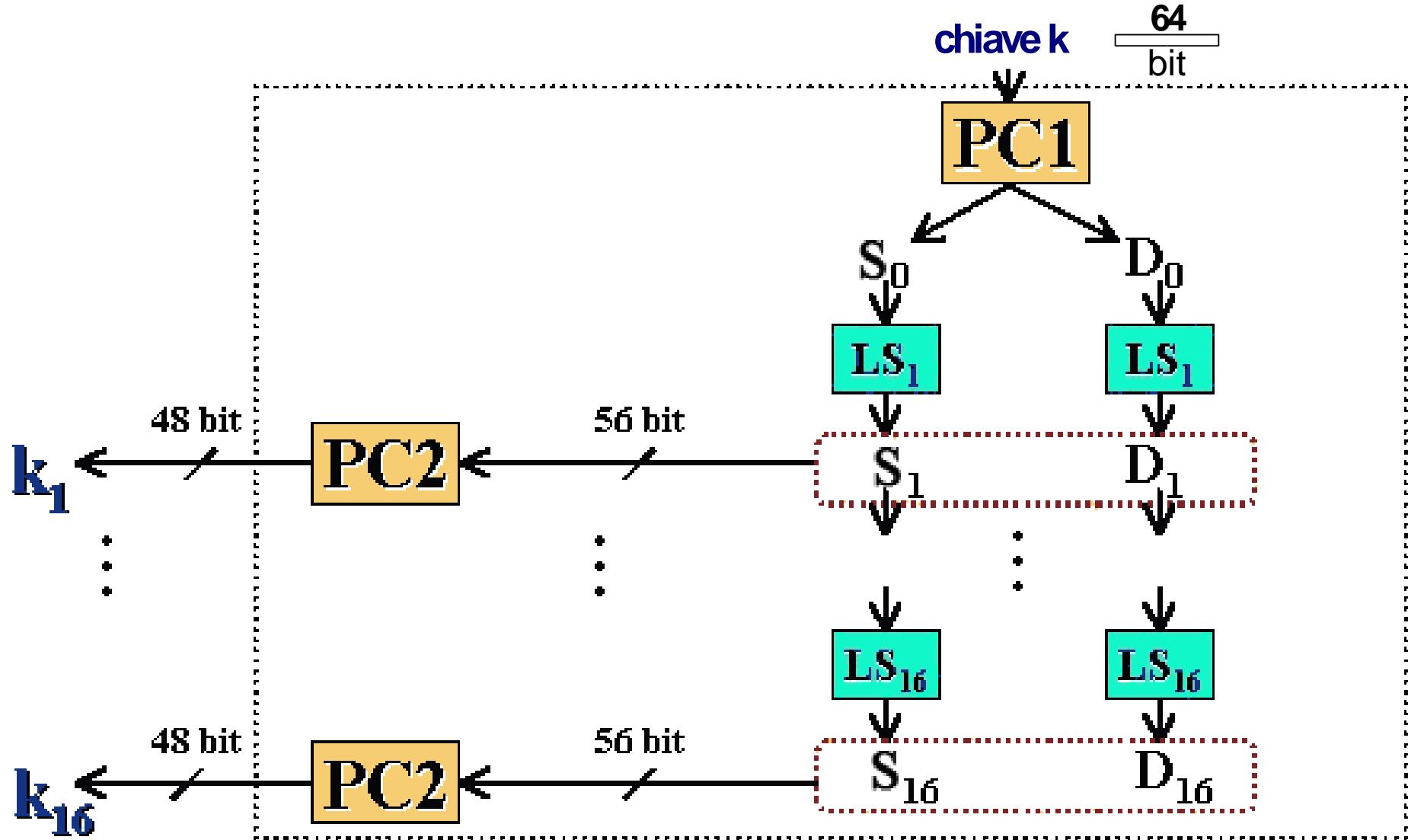
# Generazione delle Chiavi (1)



# Generazione delle Chiavi (2)

- Le chiavi vengono generate da uno Scheduler. Esso prende in input la chiave di 64 bit (generata in modo random automaticamente dal sistema una volta per l'intero procedimento di cifratura)
- Lo Scheduler interviene ad ogni iterazione del DES producendo una sotto-chiave diversa di 48 bit per ciascun round
- Dalla chiave di partenza vengono dunque prodotte 16 sotto-chiavi diverse

# Generazione delle Chiavi (3)



# Generazione delle Chiavi (3)

- Lo Scheduler si compone di diverse box, attraverso le quali la chiave viene ridotta di dimensioni (in termini di bit) e rimaneggiata per produrre le diverse sotto-chiavi
- Elementi fondamentali sono:
  - PC1 (Permute Choise 1)
  - LS<sub>i</sub> (Left Shift i)
  - PC2 (Permute Choise 2)

# Riduzione bit (PC1)

Dati della chiave  
generati in modo  
casuale

64  
bit

- $K$  chiave di 64 bit di cui 8 utilizzati per controllo di parità
- Le chiavi usate nei round sono tutte derivate da  $K$

01010010 01000101 01000011 01010100 01010010 01001001 01010100 01011000

01	02	03	04	05	06	07	08
09	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

56bit

Bits di controllo di  
parità

# PC1 (1)

Chiave iniziale

01	02	03	04	05	06	07	08
09	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Scelta permutata 1

57	49	41	33	25	17	09
01	58	50	42	34	26	18
10	02	59	51	43	35	27
19	11	03	60	52	44	36
63	55	47	39	31	23	15
07	62	54	46	38	30	22
14	06	61	53	45	37	29
21	13	05	28	20	12	04

Fig. 1

Bits scartati

# PC1 (2)

57	49	41	33	25	17	09
01	58	50	42	34	26	18
10	02	59	51	43	35	27
19	11	03	60	52	44	36
63	55	47	39	31	23	15
07	62	54	46	38	30	22
14	06	61	53	45	37	29
21	13	05	28	20	12	04

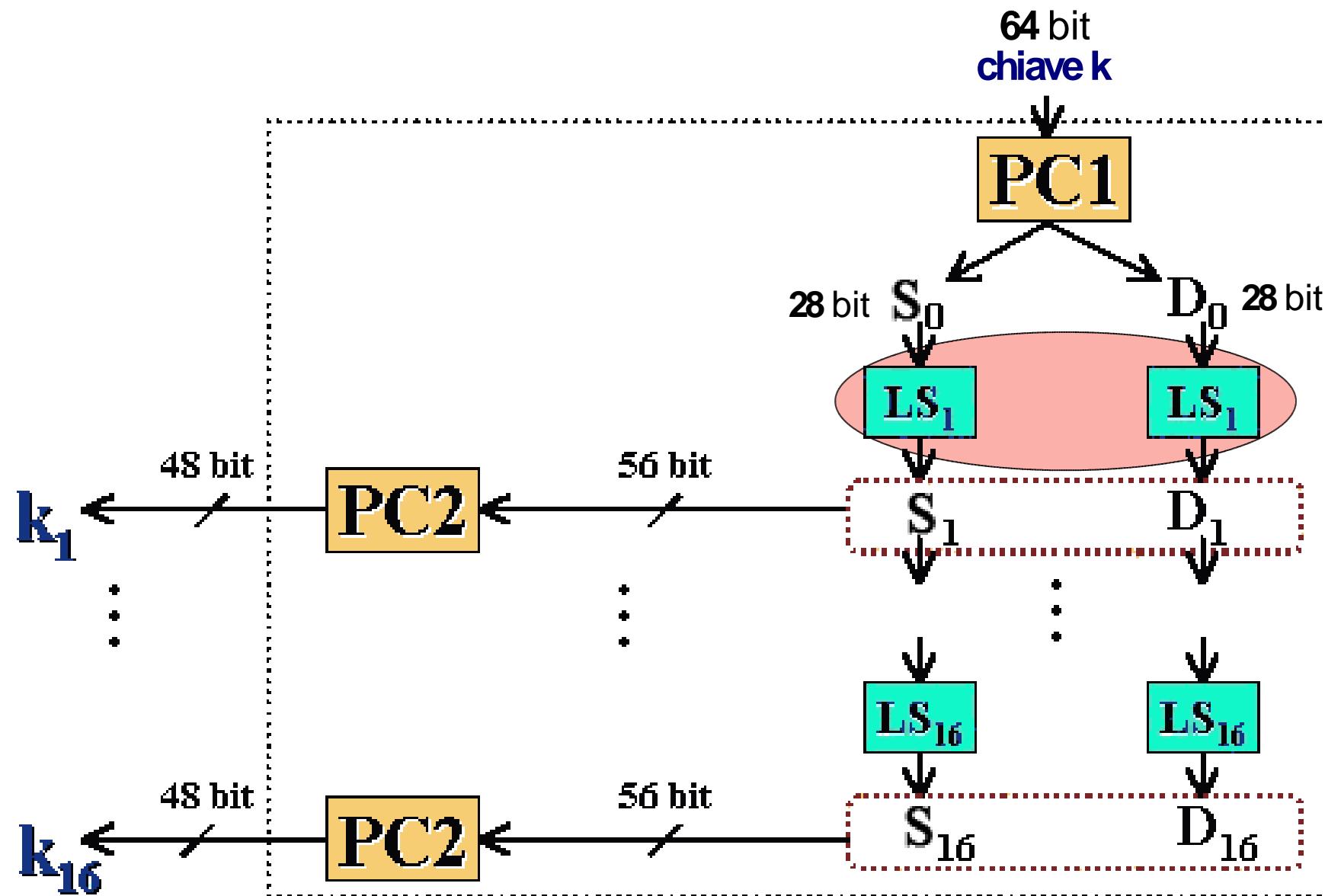
$S_0$

$D_0$

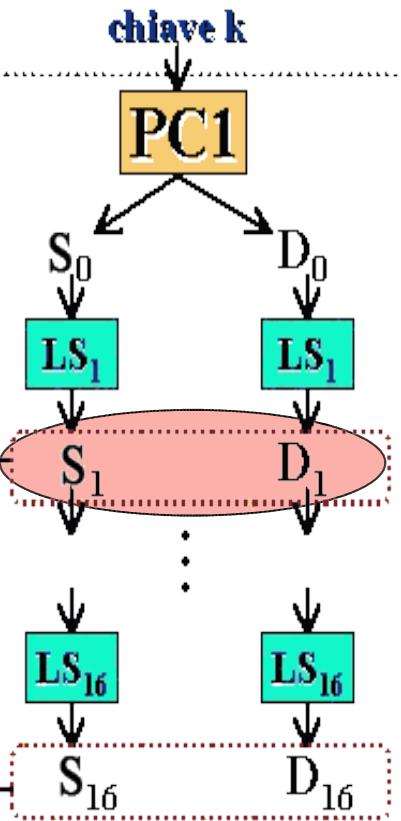
- Vengono prodotte due semi-chiavi,  $S_0$  e  $D_0$

# PC1 (spiegazioni)

- Accetta in ingresso la chiave di 64 bit assegnata al processo di cifratura. Tale chiave contiene 8 bit per il controllo di parità (uno per ogni byte). Non contenendo informazione i bit di parità possono essere eliminati, portando la chiave da 64 a 56 bit. In una rappresentazione matriciale della chiave (matrice 8x8) i bit da scartare sono quindi 8,16,24,32,40,48,56,64. La matrice uscente è quindi 8x7.
- Esegue una permutazione di bit secondo lo schema in Fig.1. Il bit 57 ad esempio viene mappato nel bit 1 della nuova matrice, il bit 49 nel bit 2 etc..
- La matrice così ottenuta viene suddivisa in due (per righe), andando a produrre due sotto-chiavi di 28 bit ciascuna, etichettate come **S<sub>0</sub>** e **D<sub>0</sub>** (sotto-chiave Sinistra e sotto-chiave Destra)



# LSi



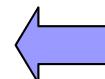
Ciclo	Spostamento a sinistra
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

49	41	33	25	17	09	01	
58	50	42	34	26	18	10	
02	59	51	43	35	27	19	
11	03	60	52	44	36	57	
55	47	39	31	23	15	07	
62	54	46	38	30	22	14	
06	61	53	45	37	29	21	
13	05	28	20	12	04	63	

Tabella 1

= 28

Shift ciclico



# LSi dettagli

- Ad ogni iterazione le due semichiavi  $S_i$  e  $D_i$  di 28 bit vengono elaborate dal blocco  $LSi$  in parallelo.
- $LSi$  effettua uno shift verso sinistra di tutti i bit delle semichiavi ad ogni iterazione in accordo con la Tabella 1. Come si può notare lo spostamento è di una posizione nelle itarazioni 1-2-9-16, di due posizioni in tutte le altre.
- Se sommiamo i numero totale di spostamenti otteniamo un numero di bit pari a 28. Ciò significa che lo shift è ciclico, ovvero alla sedicesima itarazione il primo bit della matrice di partenza è tornato in posizione 1. Questo dettaglio ci sarà utile nella fase di decifrazione.
- $LSi$  produce in output due nuove semichiavi  $S_{i+1}$  e  $D_{i+1}$ .

# PC2

49	41	33	25	17	09	01
58	50	42	34	26	18	10
02	59	51	43	35	27	19
11	03	60	52	44	36	57
55	47	39	31	23	15	07
62	54	46	38	30	22	14
06	61	53	45	37	29	21
13	05	28	20	12	04	63

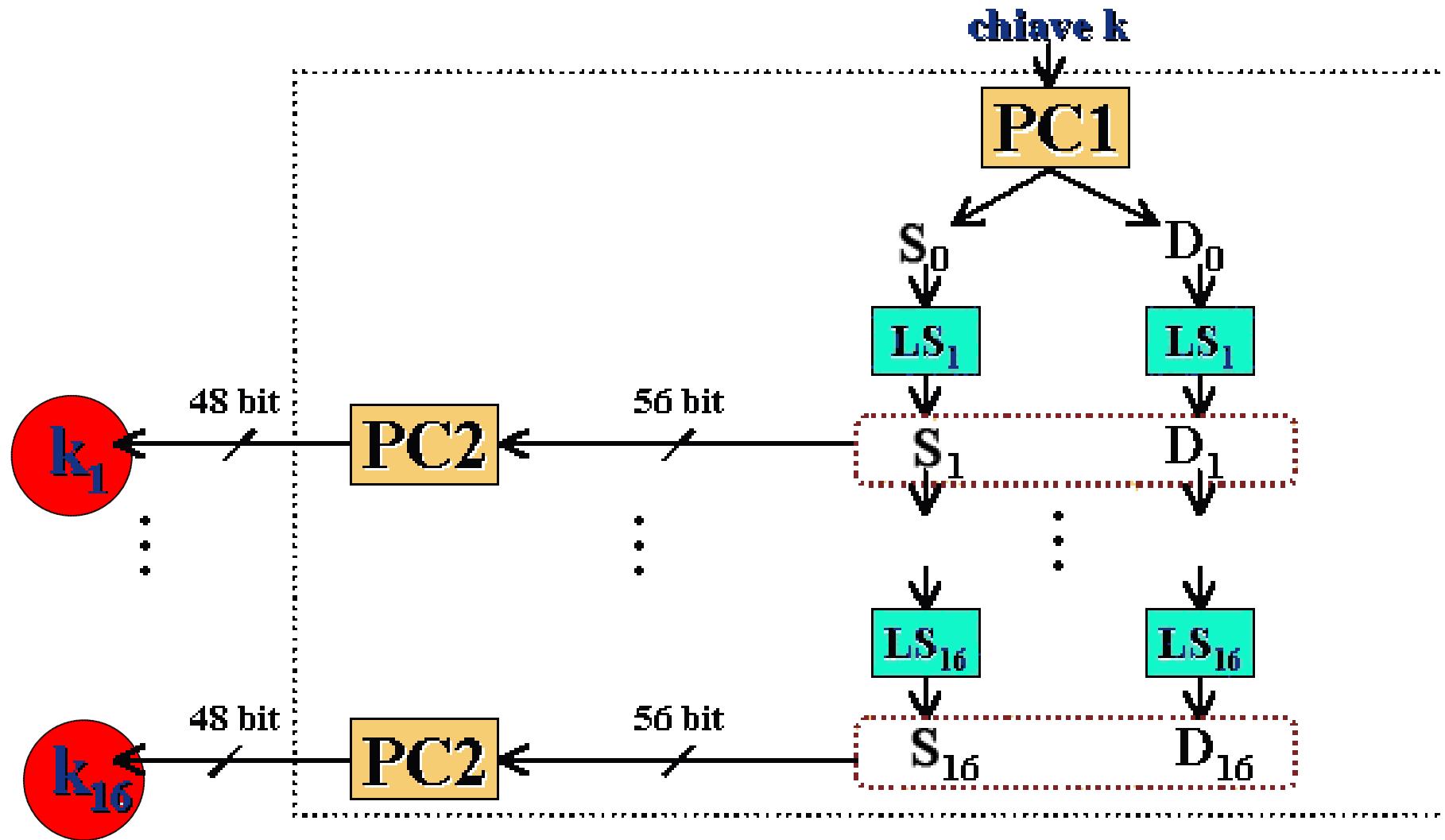
Scelta permutata 2						
14	17	11	24	01	05	
03	28	15	06	21	10	
23	19	12	04	26	08	
16	07	27	20	13	02	
41	52	31	37	47	55	
30	40	51	45	33	48	
44	49	39	56	34	53	
46	42	50	36	29	32	

K<sub>i</sub>

I bit che vengono soppressi dalla compressione sono quelli in posizione 9, 18, 22, 25, 35, 38, 43 e 54 della stringa input

Fig. 2

# Output di PC2 (1)



# Output di PC2 (2)

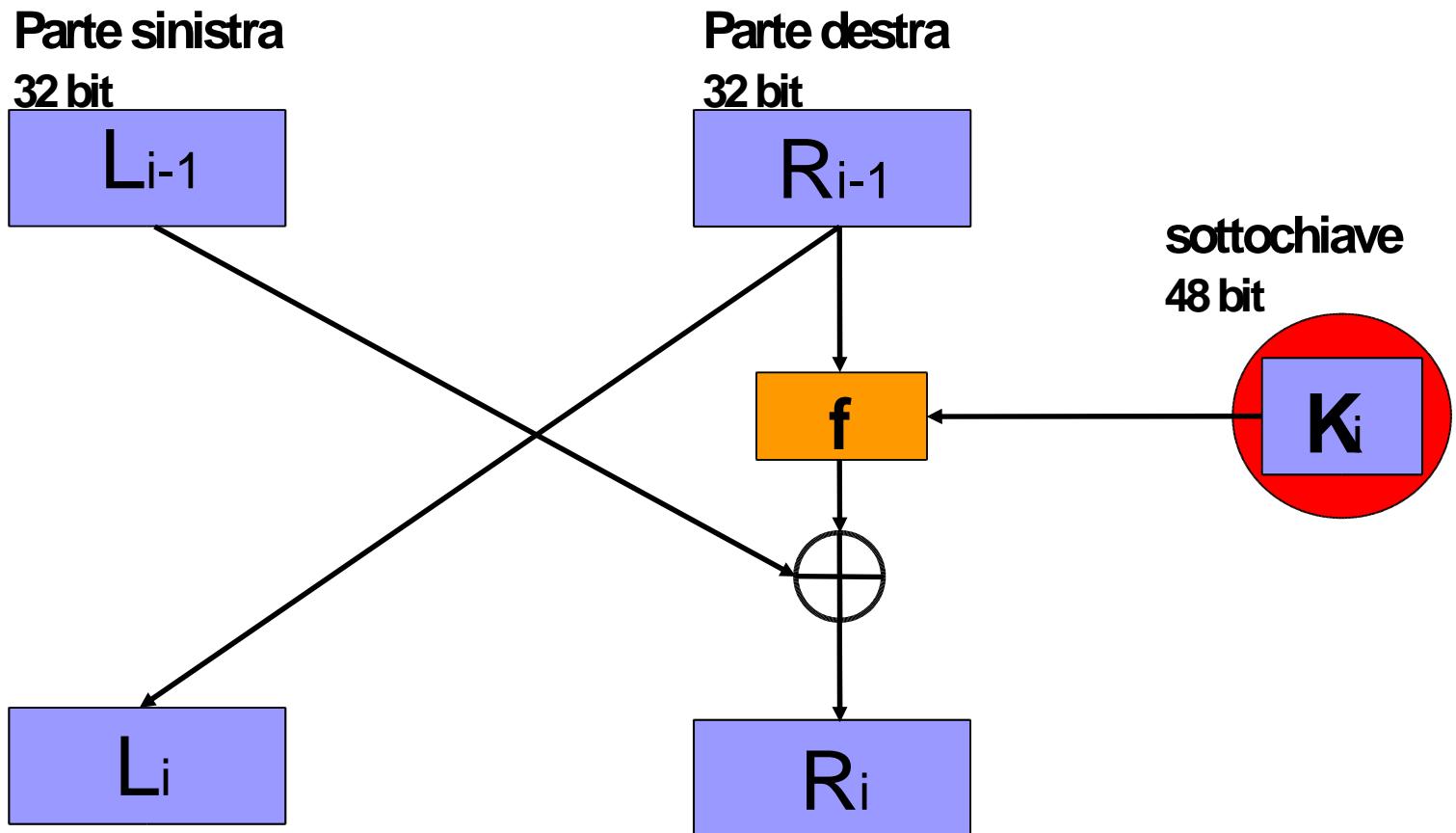
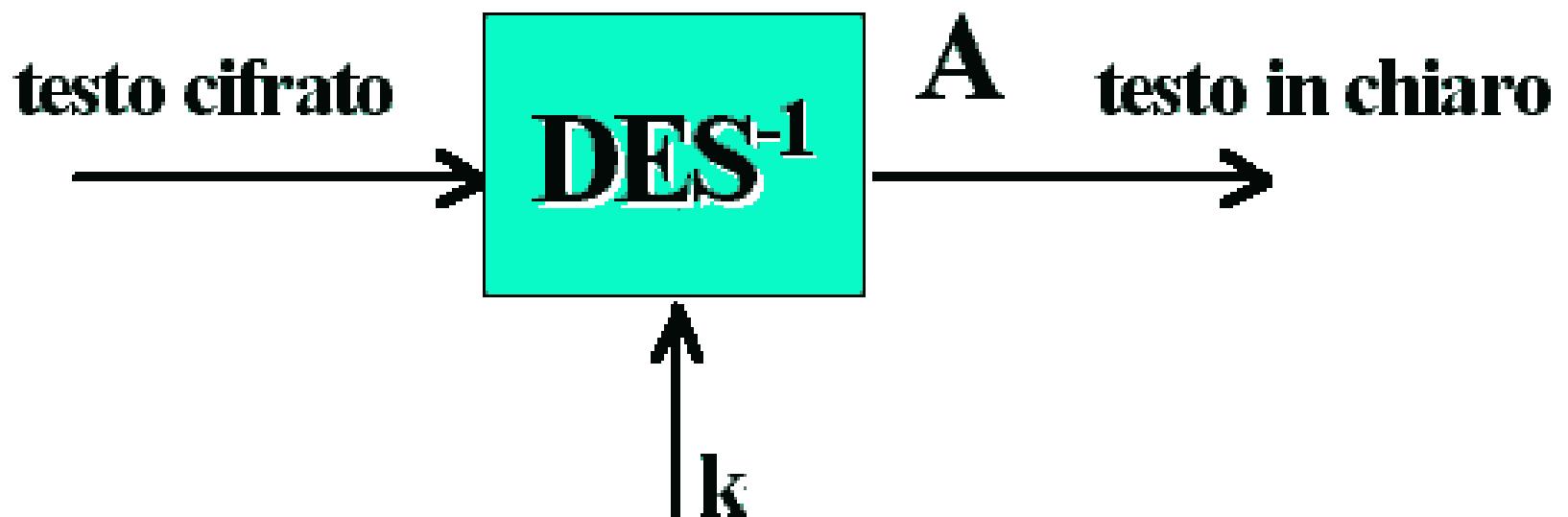


Fig. 3

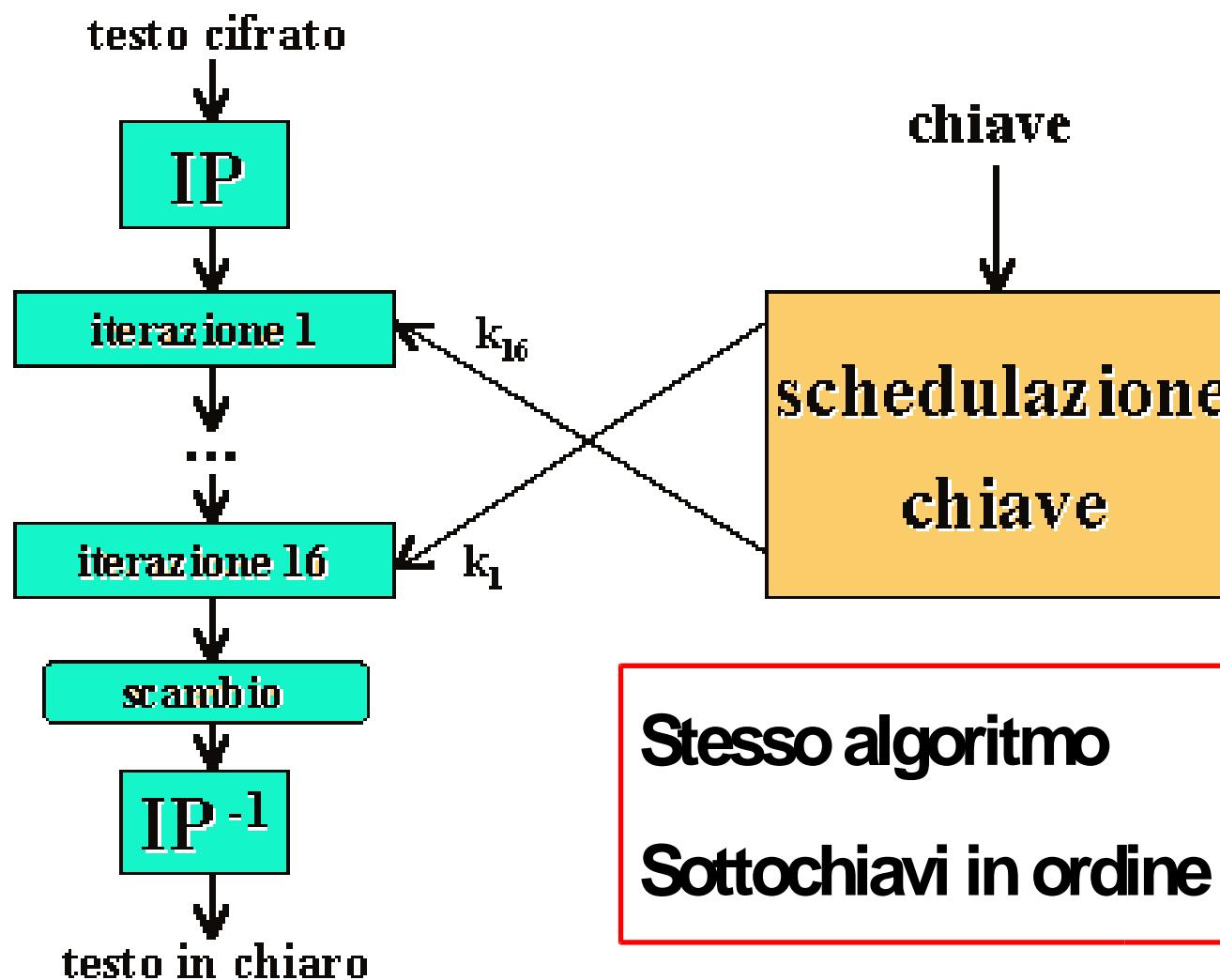
# PC2 dettagli

- PC2 opera ad ogni iterazione per produrre in output una chiave da 48 bit.
- Prende in ingresso le due sotto-chiavi  $S_i$  e  $D_i$  di 28 bit ciascuna e le ricompone in una da 56 bit.
- Effettua una seconda scelta permutata (Fig. 2), che rimescola e comprime la chiave in input (vengono eliminati i bit in posizione 9, 18, 22, 25, 35, 38, 43 e 54) per ottenere in output una chiave da 48 bit.
- La chiave  $K_i$  così ottenuta va quindi in ingresso alla funzione  $f$  dell'  $i$ -esimo round del DES (Fig. 3)

# Decifrazione (1)



# Decifrazione (2)

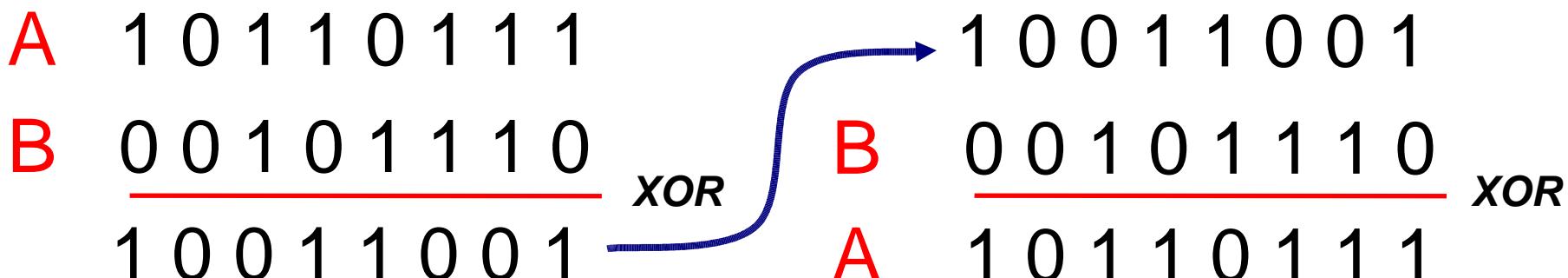


**Stesso algoritmo**  
**Sottochiavi in ordine inverso**

# Proprietà XOR

- E' una operazione reversibile:
  - reciproco annullamento di due **XOR** identici
  - non comporta perdita di informazione

$$\begin{aligned}(A \text{ XOR } B) \text{ XOR } B &= A \\ (A \text{ XOR } B) \text{ XOR } A &= B\end{aligned}$$



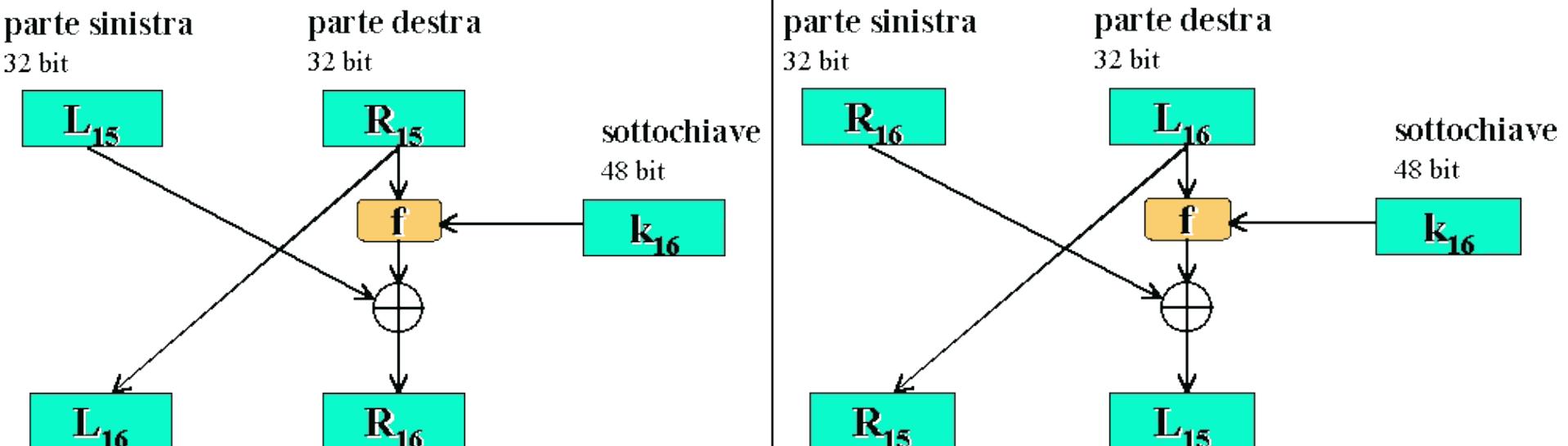
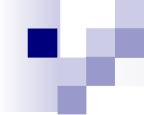


Fig.A (ultima iterazione cifratura)

Fig.B (prima iterazione decifratura)

- Consideriamo il messaggio cifrato  $R_{16} L_{16}$  a meno della permutazione finale  $IP^{-1}$ . Dalla Figura A si ricavano le seguenti relazioni:

$$L_{16} = R_{15} \text{ e } R_{16} = L_{15} \oplus f(R_{15}, K_{16})$$

- Da qui riscrivendo le due equazioni possiamo ricavare i *valori precedenti*:

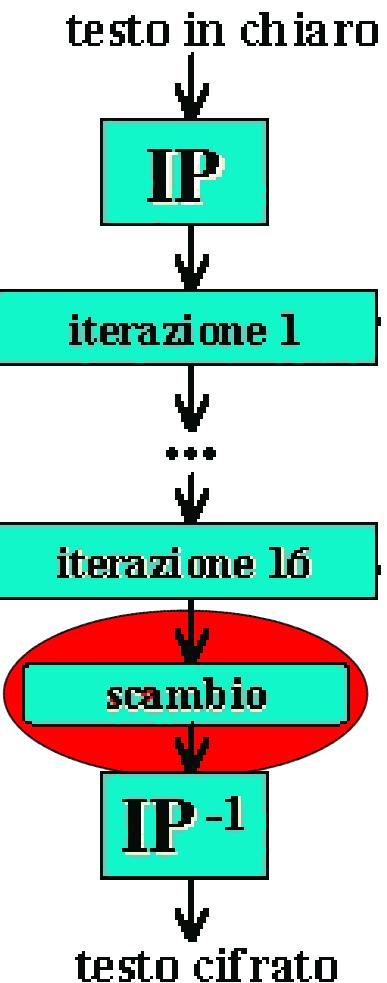
$$R_{15} = L_{16} \text{ e } L_{15} = R_{16} \oplus f(R_{15}, K_{16}) = R_{16} \oplus f(L_{16}, K_{16})$$

- In conclusione: da  $R_{16}$  e  $L_{16}$  abbiamo ricavato  $R_{15}$  ed  $L_{15}$  (Figura B)

- Quindi, invertendo il ruolo di  $R$  ed  $L$  e utilizzando le chiavi in maniera inversa, da  $K_{16}$  a  $K_1$ , si può ritornare al messaggio in chiaro passando attraverso le seguenti coppie:

$(R_{16}, L_{16}) \rightarrow (R_{15}, L_{15}) \rightarrow (R_{14}, L_{14}) \rightarrow \dots \rightarrow (R_0, L_0)$

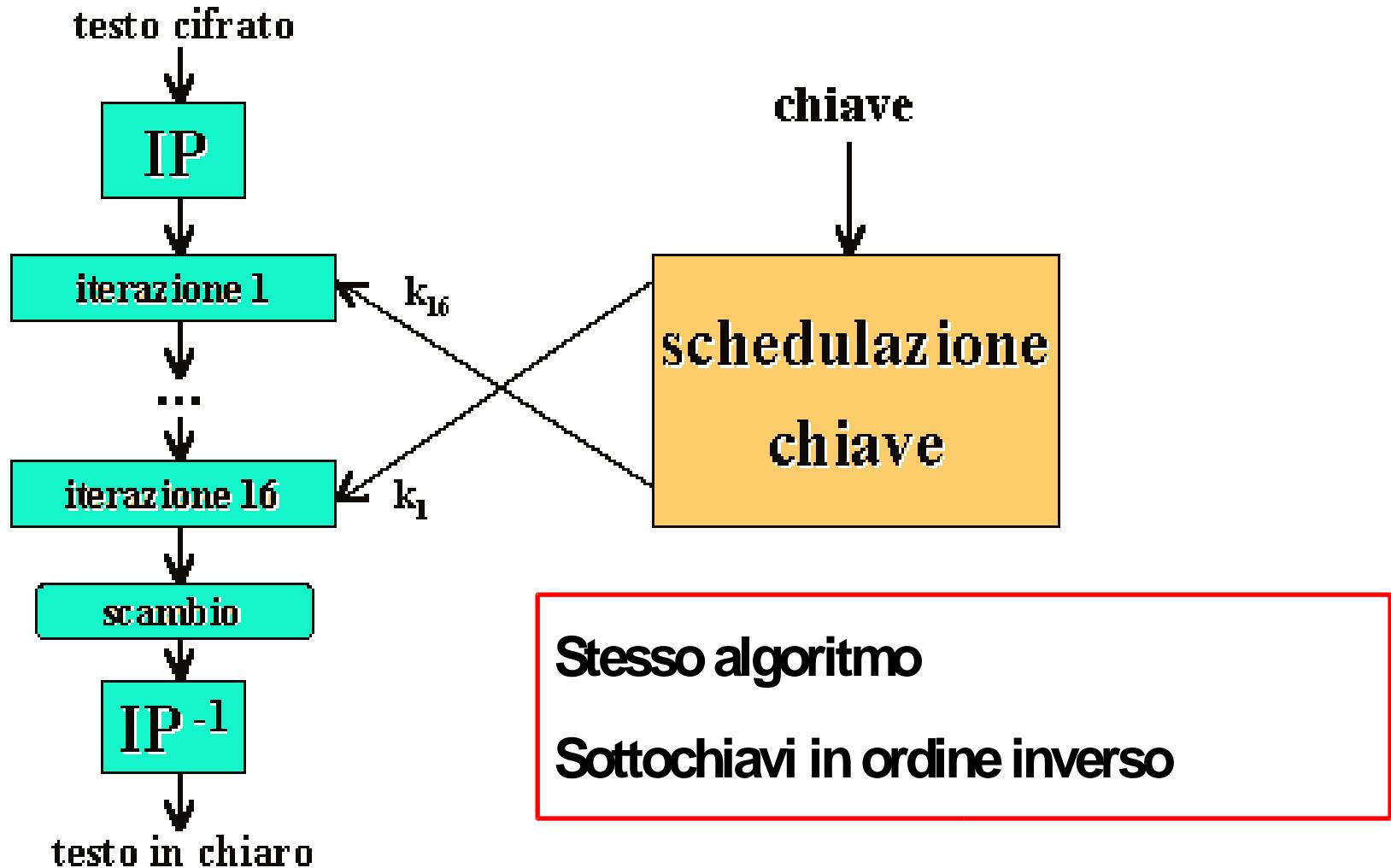
- Questo è il motivo per cui alla fine dell' algoritmo di cifratura si effettua lo scambio tra le parti  $L$  e  $R$ .



# Chiave – Schedulazione inversa (1)

- Dal momento che per la fase di decifratura le chiavi devono essere schedulate in ordine inverso rispetto alla cifratura, dobbiamo effettuare delle variazioni al dispositivo di schedulazione definito in precedenza.
- In particolare dobbiamo fare in modo che pur prendendo in input la chiave  $K$ , tale dispositivo produca le chiavi nell'ordine desiderato:
  - Per ottenere ciò invertiamo lo shift che veniva applicato alle semi-chiavi  $L_i$  e  $R_i$  effettuandolo verso destra anziché verso sinistra
  - Applichiamo la tabella di  $LS$  in ordine inverso 
  - In questo modo al primo passo di decifrazione ottengo da  $LS$  la sedicesima sotto-chiave di cifratura.
- Per ottenere la  $i$ -esima chiave **non** devo ricalcolare tutte le  $i-1$  che la precedono!

# Chiave – Schedulazione inversa (2)





# Roberto Pariset



# Crittoanalisi del DES

- Debolezze del DES
- Attacchi al DES
- Modalità operative
- 2DES e 3DES

# Caratteristiche del DES

- DES non è un cifrario perfetto
  - Il crittogramma, in assenza della chiave, dà informazioni sul messaggio
  - La chiave è più corta del messaggio
- DES non è un cifrario ideale
  - La distribuzione statistica del testo in chiaro dipende dalla distribuzione statistica del crittogramma

# Debolezze del DES

- Chiavi
  - Deboli
  - Semi-deboli
  - Particolari
  - Lunghezza della chiave
  - Complementazione
- Shift
- S-Box
- Numero di iterazioni

# Debolezze Della Chiave (1)

## ■ **Chiavi deboli**

- Generano un'unica sottochiave
- La sottochiave viene usata per 16 round
- Esistono 4 chiavi di questo tipo
- Proprietà:  $E_k(E_k(x)) = x$
- Ovvero: se ricifriamo un testo cifrato con una weak key otteniamo il testo in chiaro

# Debolezze Della Chiave (2)

## ■ **Chiavi semi-deboli**

- Generano 2 sottochiavi diverse
- Ognuna utilizzata in 8 round
- Esistono 6 coppie di chiavi di questo tipo
- Proprietà:  $E_{k1}( E_{k2}( x ) ) = x$
- Ovvero: alcune coppie di chiavi trasformano un messaggio in un unico crittogramma. Una chiave della coppia può decrittare un messaggio crittato con l'altra chiave



# Debolezze Della Chiave (3)

## ■ Chiavi particolari

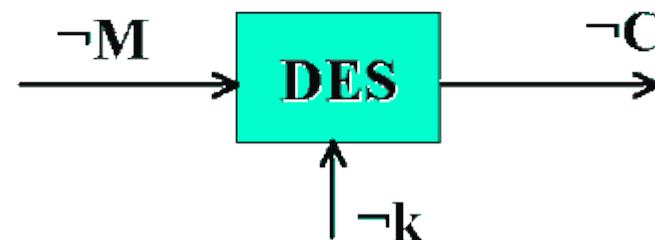
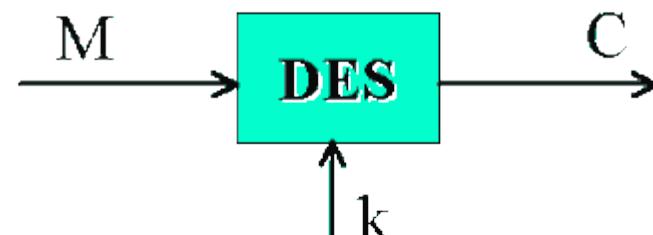
- Generano 4 sottochiavi diverse
- Ognuna usata in 4 round
- Esistono 48 chiavi di questo tipo
- Nessuna proprietà attualmente scoperta

- **Lunghezza:** una chiave di 56 bit è troppo piccola per essere sicura
- Le chiavi raramente sono veramente casuali

# Proprietà di complementazione (1)

- Voglio perpetrare un attacco
- Conosco sia un testo in chiaro sia il relativo testo cifrato
- Voglio determinare la chiave con un attacco brute force
- Occorrerebbero  $2^{56}$  tentativi, ma...

# Proprietà di complementazione (2)



- Sfruttando la proprietà di complementazione ne occorrono la metà ( $2^{55}$ )
- Se ho trovato il complemento di  $C$  mi basta complementare la chiave

# Attacchi al DES

- Tipi di attacchi
  - Known plaintext, chosen plaintext, known ciphertext
- Brute forcing
- Crittoanalisi
  - Differenziale
  - Lineare
  - Basata su chiavi



# Brute forcing

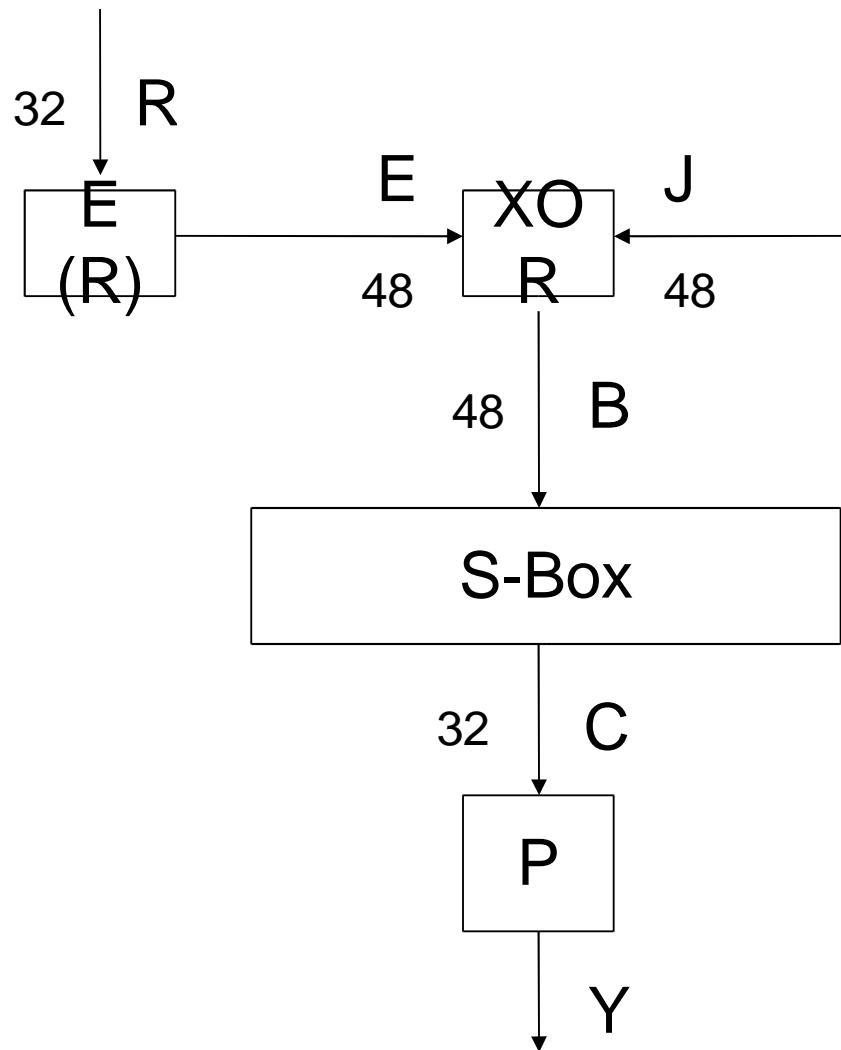
## ■ Forza bruta

- Un attacco brute force a DES è un problema NP-completo
- Il DES ha una chiave a 56 bit
- Questo attacco esegue una ricerca esaustiva sullo spazio delle chiavi
- Complessità temporale dell'ordine di  $2^{56}$  ( $2^{55}$  in media)
- Complessità spaziale costante

# Crittoanalisi differenziale (1)

- Ideata da Biham e Shamir nel 1990
- Si basa sul confronto tra le *differenze* di due testi in chiaro e quelle dei testi cifrati corrispondenti
- Differenze ottenute tramite XOR bit a bit
- Obiettivo: scoprire la sottochiave utilizzata nell'ultimo round, e quindi la chiave

# Crittoanalisi differenziale (2)



$$Y^* = Y \text{ XOR } Y'$$

$S_j$  con  $j=1..8$   
prende in ingresso 6  
bit e ne restituisce 4

- $E$  è lineare
- $P$  è lineare
- XOR è lineare

# Crittoanalisi differenziale (3)

- Sia  $B_j$  una stringa binaria di 6 bit (tot  $2^6$ )
- Sia  $C_j$  una stringa binaria di 4 bit (tot  $2^4$ )
- Le stringhe  $B_j$  vengono prese in input da una certa  $S_j$
- Le stringhe  $C_j$  vengono prodotte in output da una certa  $S_j$ ; perciò  $C_j = S_j( B_j )$
- Definiamo input XOR:  $B_j^* = B_j \text{ XOR } B_j'$
- Definiamo output XOR:  $C_j^* = C_j \text{ XOR } C_j'$

# Crittoanalisi differenziale (4)

- Sia  $D(B^*) = \{ (B, B'): B \text{ XOR } B' = B^* \}$  e si calcoli tale insieme per ogni  $B^*$
- Per ogni coppia in  $D(B^*)$  si calcoli quindi l'output XOR rispetto ad una certa  $S$ , al fine di determinare una tabella delle occorrenze
- Si noti che per alcune stringhe  $C^*$  **non** ci sono coppie corrispondenti in  $D(B^*)$
- Statistiche per determinare la distribuzione delle occorrenze per l'output XOR

# Crittoanalisi differenziale (5)

- Ad esempio, le 64 coppie in  $D(110100)$  producono la seguente distribuzione per l'output XOR, relativamente ad S1:

0000	0001	0010	0011	0100	0101	0110	0111
0	8	16	6	2	0	0	12

1000	1001	1010	1011	1100	1101	1110	1111
6	0	0	0	0	8	0	6

- 110100 causa 0111 da S1 con probabilità 12/64

# Crittoanalisi differenziale (6)

- Si definisca l'insieme  $\text{IN}(B^*, C^*)$ , relativo a una certa  $S$ , come:  $\{B : S(B) \text{ XOR } S(B \text{ XOR } B^*) = C^*\}$
- Per ognuna delle 8  $S$  è possibile rappresentare l'insieme  $\text{IN}$  costruendo una **tabella di distribuzione delle differenze** avente:
  - $2^6$  righe, pari al numero dei possibili input XOR
  - $2^4$  colonne, pari al numero dei possibili output XOR
  - elementi che rappresentano il numero di volte che un certo output XOR occorre per un certo input XOR

# Crittoanalisi differenziale (7)

- $B = E \text{ XOR } J$
- $B \text{ XOR } B' = (E \text{ XOR } J) \text{ XOR } (E' \text{ XOR } J)$
- $B \text{ XOR } B' = E \text{ XOR } E'$  ovvero  $B^* = E^*$
- **L'input XOR non dipende da J**
- E ed  $E'$  sono noti  $\rightarrow$  informazioni su  $J_r$

# Crittoanalisi differenziale (8)

- La coppia di blocchi in ingresso ( $E, E'$ ) dell'ultimo round **è nota** perché è contenuta nei blocchi prodotti in uscita ( $C, C'$ )
- È possibile estendere le tavelle di distribuzione delle differenze e lavorare sulla funzione  $F$  invece che sulle singole  $S$
- Le occorrenze (ovvero le probabilità) in  $F$  si ottengono moltiplicando quelle delle otto  $S$

# Crittoanalisi differenziale (9)

- Si scelga una certa differenza tra due plaintext e sia questa differenza  $E^*$
- Si calcoli  $D(E^*)$
- Per ogni sottochiave  $J_r$ , si critti una coppia di plaintext in  $D(E^*)$  e si conservi la coppia dei crittogrammi ottenuti ( $C, C'$ )

# Crittoanalisi differenziale (10)

- Per ogni coppia ottenuta ( $C, C'$ ) si calcoli l'output XOR atteso (in base a  $E^*$  e alla tabella di distribuzione delle differenze) per tutte le S della S-Box
- Per ogni possibile sottochiave si conti il numero di coppie che hanno un output XOR pari a quello atteso
- La sottochiave corretta è quella suggerita da tutte le coppie

# Crittoanalisi differenziale (11)

- È così possibile determinare i 48 bit di  $J_r$
- Data  $J_r$  è possibile ottenere 48 bit di  $K$
- $K$  è di 56 bit
- Gli 8 bit mancanti si ottengono con una ricerca esaustiva sulle  $2^8 = 256$  combinazioni possibili

# Crittoanalisi differenziale (12)

- Con un DES a meno di 16 round si può identificare la chiave più efficientemente che con un attacco brute force
- Con un DES a 16-18 round i due attacchi sono invece sostanzialmente equivalenti
- Non applicabile ad un DES con 19 round o più, perché occorrerebbero più di  $2^{64}$  chosen plaintext

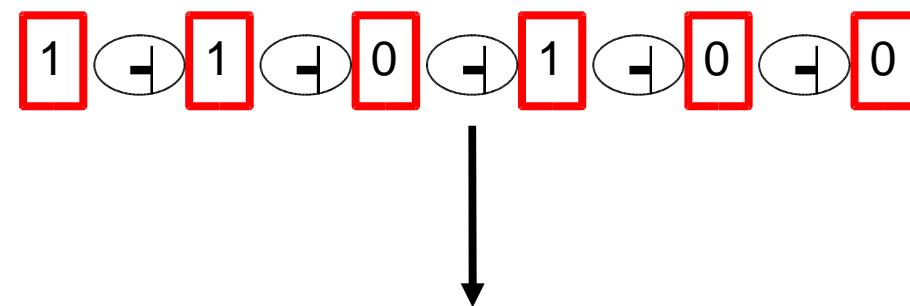
# Crittoanalisi differenziale (13)

- L'IBM conosceva la crittoanalisi differenziale e ha progettato il DES con 16 round per resistere a questo tipo di attacco
- Richiede  $2^{47}$  chosen plaintext oppure  $2^{55}$  known plaintext
- La probabilità di successo aumenta (e diminuisce) linearmente col numero di plaintext disponibili

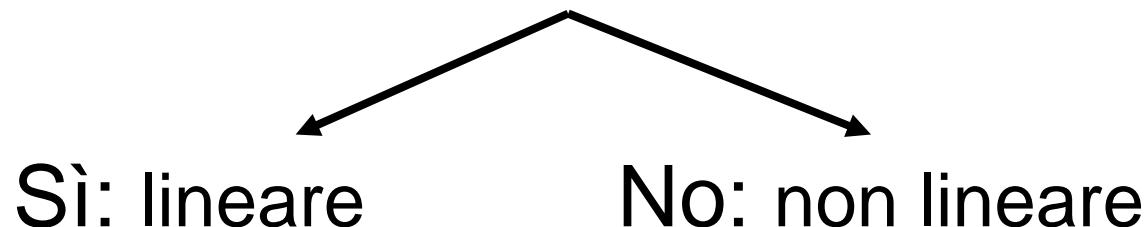
# Crittoanalisi lineare (1)

- Ideata da Mitsuru Matsui nel 1994
- Richiede  $2^{47}$  known plaintext
- Cerca di trovare un'approssimazione lineare che descriva le trasformazioni operate dal DES
- Nella S-Box nessun bit di output è una funzione lineare del bit di input

Mettendo in XOR n bit del testo in chiaro, n bit del testo cifrato e mettendo in XOR il risultato ottenuto, ottengo un bit



La probabilità di ottenere 0 è uguale  
alla probabilità di ottenere 1?



# Crittoanalisi lineare (2)

- Intuitivamente: al variare dell'input, l'output varia in maniera difficile da prevedere
- I sistemi lineari sono facili da forzare (e.g. i cifrari additivi)
- Un sistema di crittoanalisi non lineare è impossibile da rappresentare con un insieme di equazioni lineari
- Un sistema non lineare si può però *approssimare* con equazioni lineari



# Crittoanalisi lineare (3)

- Troviamo approssimazioni lineari della S-Box tramite XOR dei valori di input e output
- Le approssimazioni lineari dei vari round possono essere unite (in maniera simile alla crittoanalisi differenziale)
- Le S-Box non sono state progettate per resistere a questo tipo di attacco, che infatti risulta il più efficace conosciuto

# Crittoanalisi basata su chiavi

- Esamina le differenze tra chiavi
- Simile alla crittoanalisi differenziale
- Non dipende dal numero di round del DES
- Importanza dello shifing delle chiavi: se fosse fisso a due posizioni si potrebbe ottenere la chiave con soli  $2^{17}$  chosen plaintext



# Attacchi Pratici: Deep Crack

- Progetto della Electronic Frontier Foundation
- Nel 1998 EFF dimostra che il DES può essere realmente forzato
- Deep Crack: una macchina parallela da 250.000 dollari
- Forzatura del DES in 2 giorni

# Deep Crack

- Deep Crack è formato da 24 unità
- Ogni unità lavora con una chiave e due blocchi
  - Con la chiave prova a decifrare il crittogramma corrente
  - Se il testo ottenuto è significativo, prova a decifrare anche l'altro crittogramma
  - Se il risultato è ancora significativo, la chiave viene segnalata (falso positivo?)
  - Altrimenti passa alla chiave successiva
- Cosa definisce un risultato significativo?



# MODALITÀ OPERATIVE

Silvio Donnini

# Modalità Operative

- Metodi esterni all'algoritmo
- Specificano **come** usare il DES
- 4 modalità standard (FIPS)
  - Electronic CodeBook
  - Cipher Block Chaining
  - Cipher Feedback
  - Output Feedback

# Electronic Codebook

Modalità vista fino ad adesso, nessuna variante

## cifratura



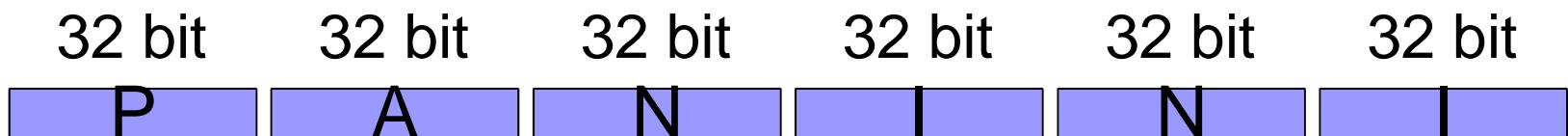
## decifratura



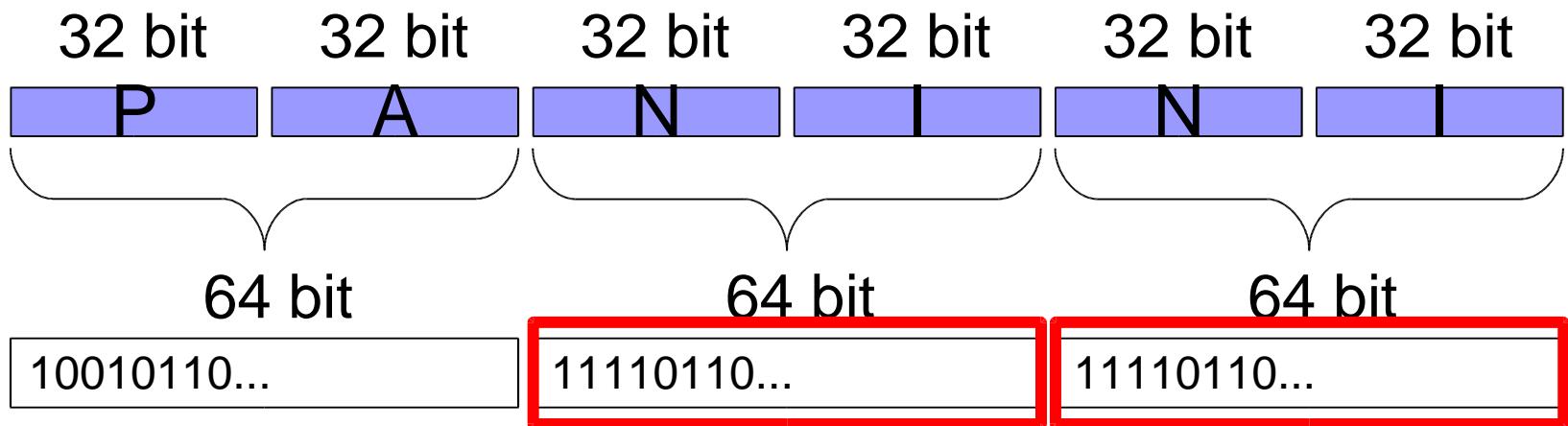
**Vantaggi:** rapidità di esecuzione.

# Ma...

- Ho un testo codificato secondo lo standard UTF-32



- UTF-32 utilizza 32 bit per un carattere



A bigrammi uguali corrispondono blocchi cifrati uguali

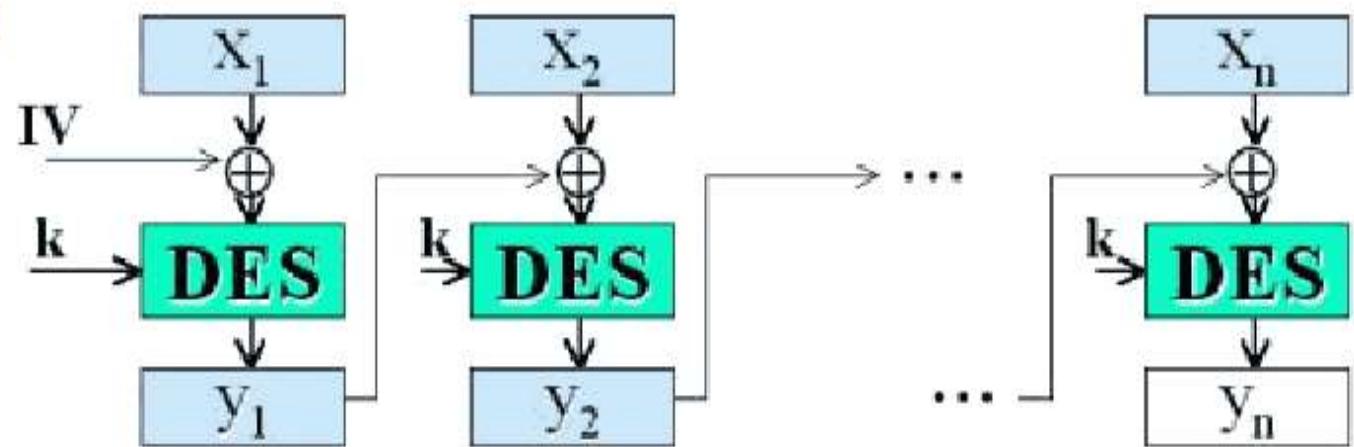
Vulnerabile ad attacchi di tipo statistico

# Inoltre

- Durante la trasmissione del messaggio
  - Potrebbe verificarsi la perdita di un blocco
  - Un nemico potrebbe cambiare l'ordine dei blocchi
- Bisognerebbe introdurre una forma di dipendenza tra i blocchi

# Cipher Block Chaining (1)

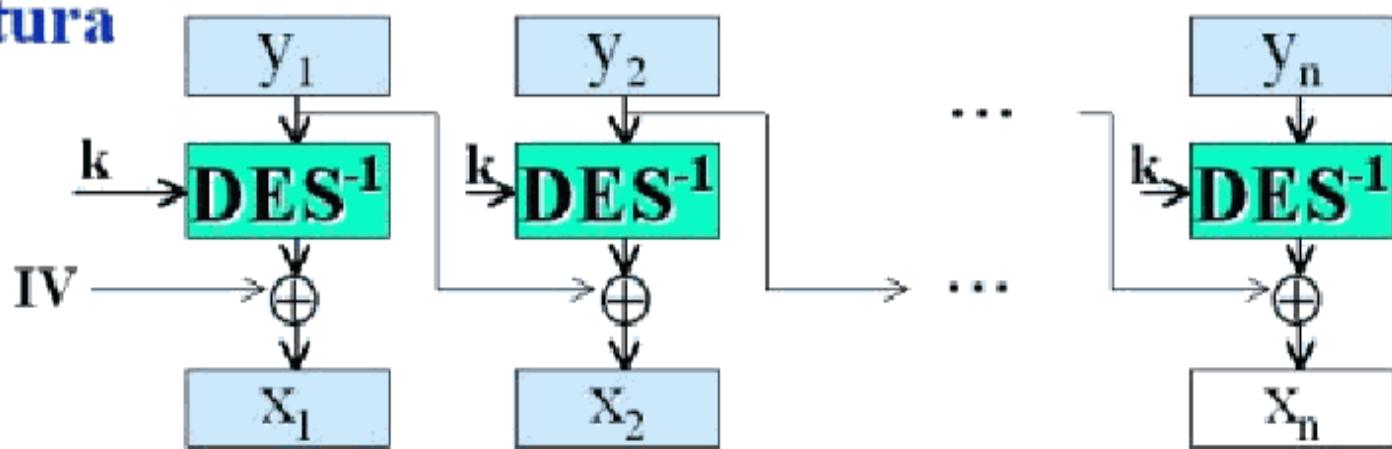
cifratura



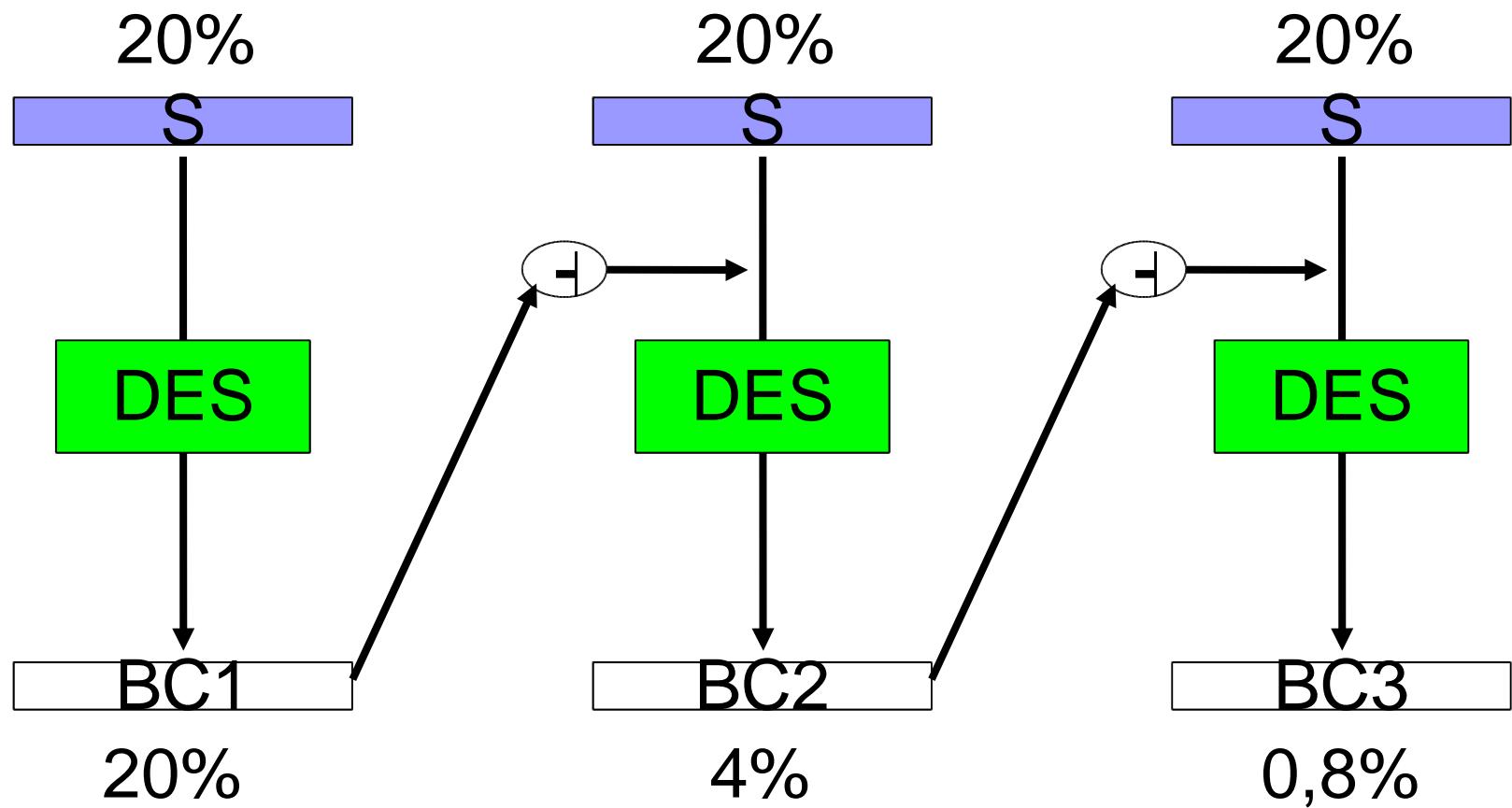
Ogni blocco in chiaro in XOR con il  
precedente blocco cifrato

# Cipher Block Chaining (2)

decifratura



# È diventato un cifrario ideale?



Le probabilità non sono indipendenti



# **Cipher Block Chaining (3)**

- La distribuzione statistica del testo cifrato è dipendente dalla distribuzione statistica del testo in chiaro
- Tuttavia la dipendenza è molto ben mascherata
- CBC aggiunge diffusione al testo cifrato



# Pro e contro

## ■ **Vantaggi:**

- Mi accorgo se è stato modificato qualunque blocco in qualunque posizione

## ■ **Svantaggi:**

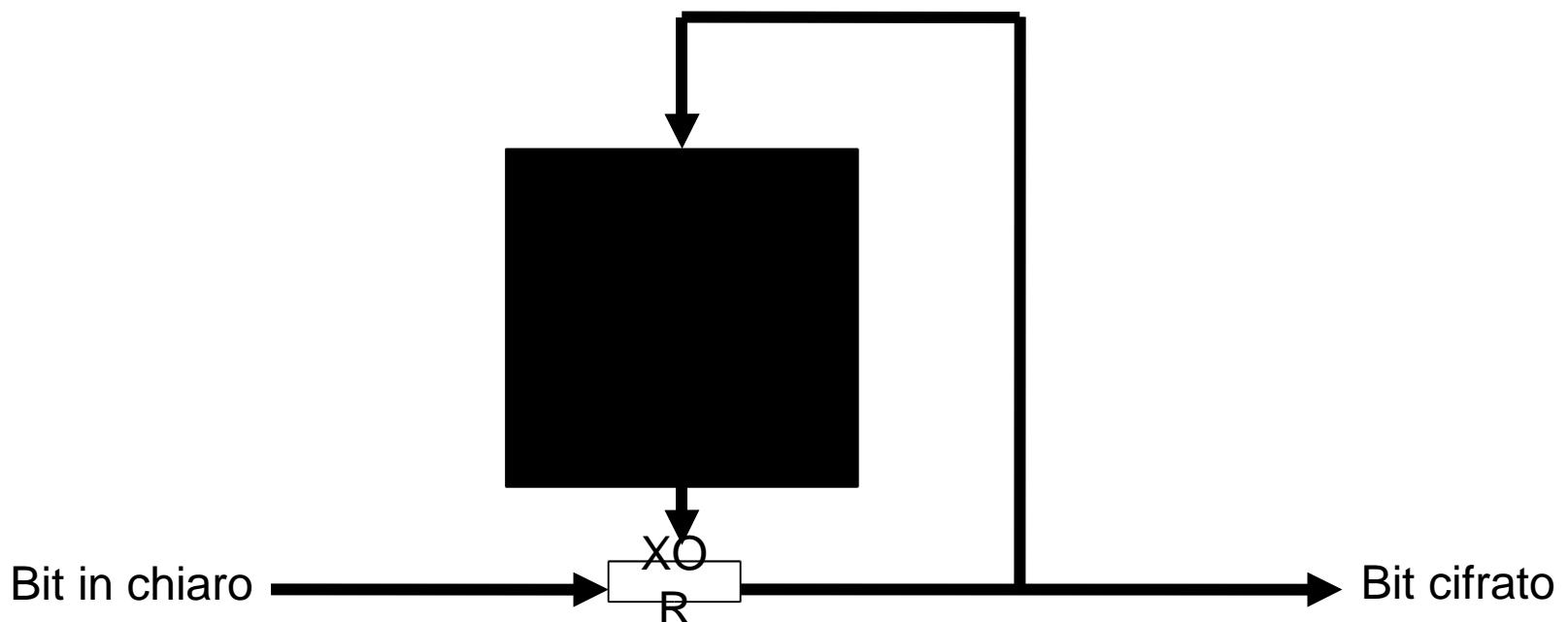
- Vengono eseguite più operazioni rispetto all'ECC, e quindi rispetto ad esso è più lento

# Cifrare bit per bit

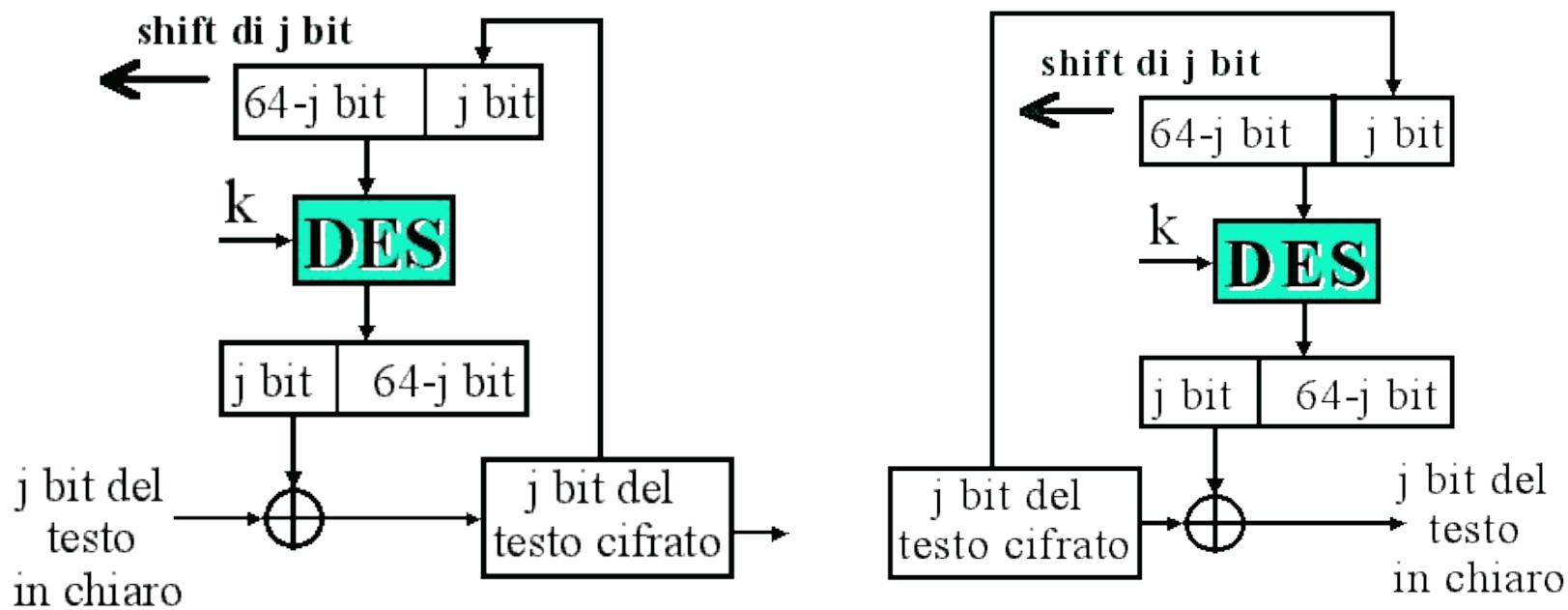
- Devo trasmettere piccoli flussi di dati (dimensioni di un byte) attraverso la rete
- ECB o CBC non sono adatti
- Esiste una modalità che mi permette di cifrare un bit alla volta

# Cipher Feedback (1)

Idea di base:



# Cipher Feedback (2)



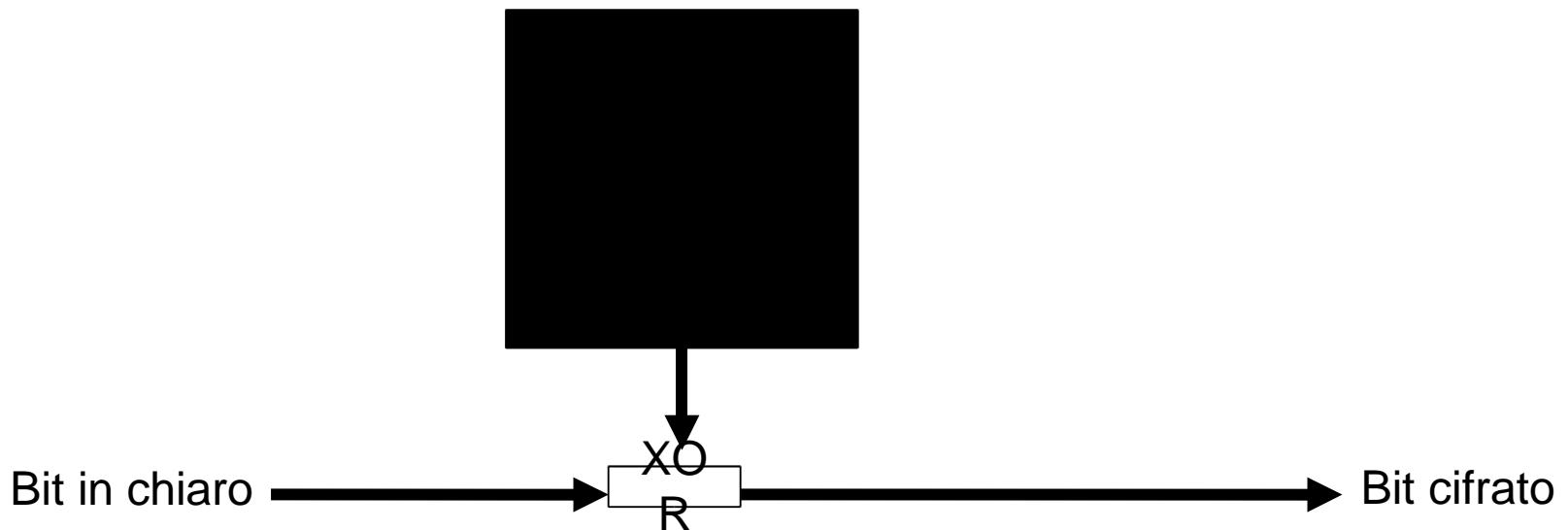


# Cipher Feedback (3)

- Consente di cifrare una quantità di dati compresa tra 1 e 64 bit
- Da utilizzare in presenza di input irregolari
- Un errore in un bit corrompe i 64 bit successivi
  - Può non essere accettabile

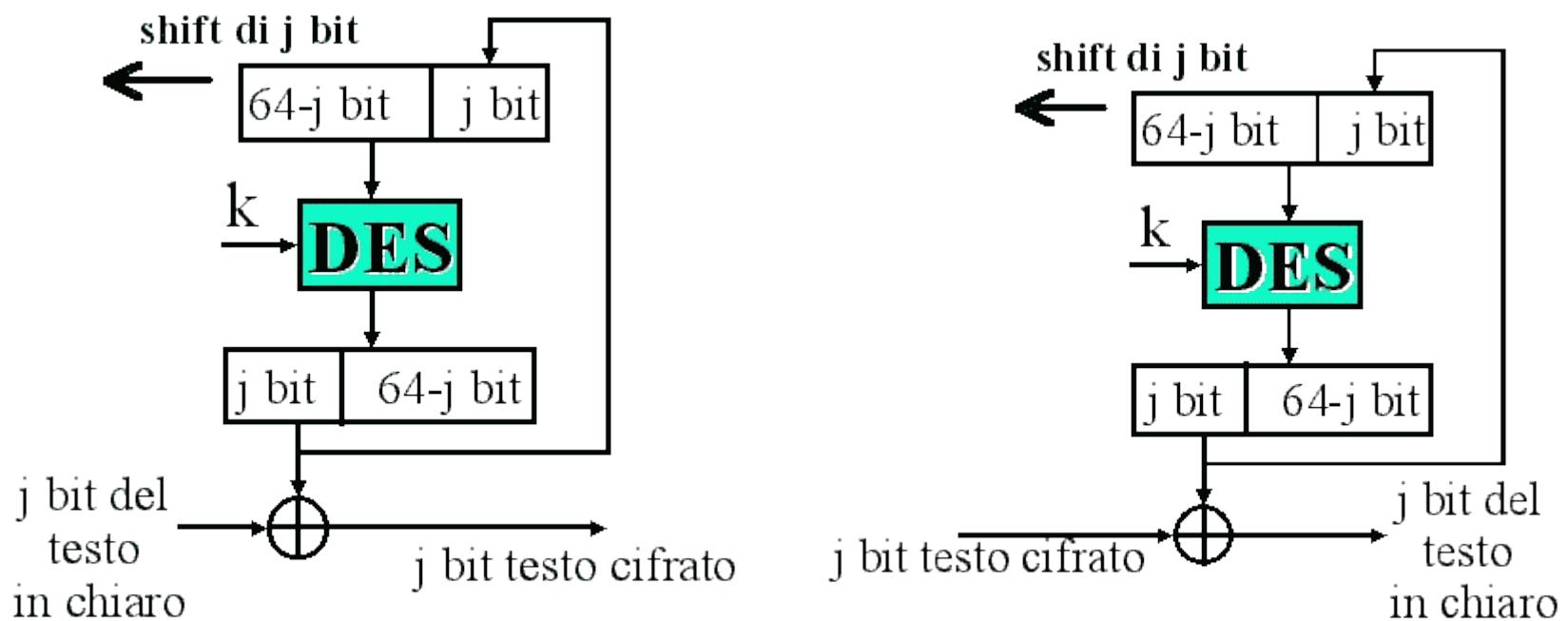
# Output Feedback (1)

- Idea di base:



- La scatola produce bit di cifratura sempre diversi ma dipendenti da un'unica chiave

# Output Feedback (2)



# Output Feedback (3)

## ■ **Vantaggi:**

- un errore su un bit resta su un bit senza propagarsi

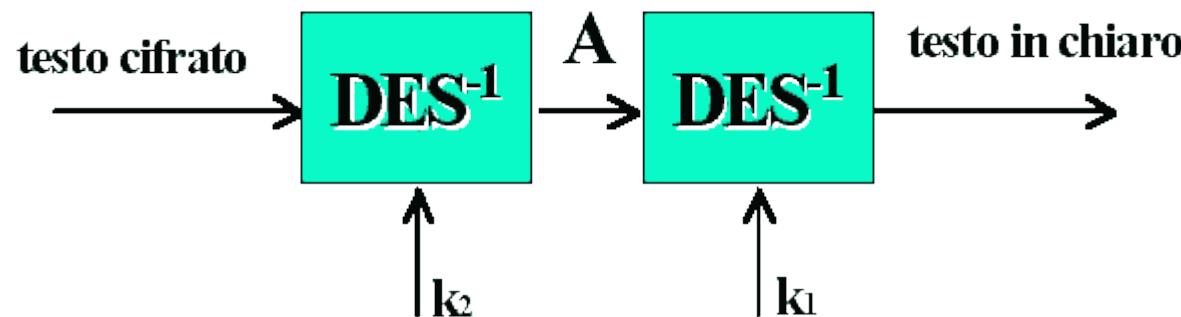
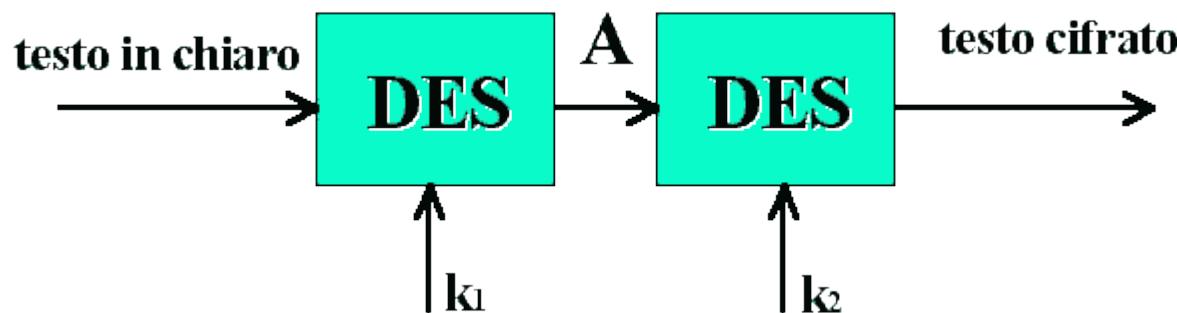
## ■ **Svantaggi:**

- non c'è alcuna dipendenza tra i bit cifrati, facili da sostituire (stesso problema di ECB)



# VARIANTI

# 2DES



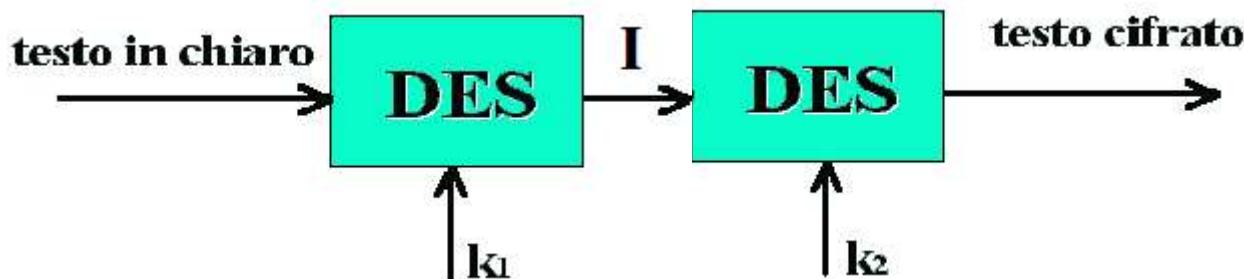
Due DES in sequenza con chiavi differenti

# 2DES: posso farlo?

- Date due chiavi  $K_1$  e  $K_2$ , **non** esiste sempre una terza chiave  $K$  tale che cifrare un testo con essa sia equivalente a cifrarlo prima con  $K_1$  e poi con  $K_2$
- In altre parole, l'insieme delle operazioni di cifratura **non** è chiuso rispetto alla composizione funzionale
- Quindi 2DES ha senso
- Ma è anche doppiamente sicuro?

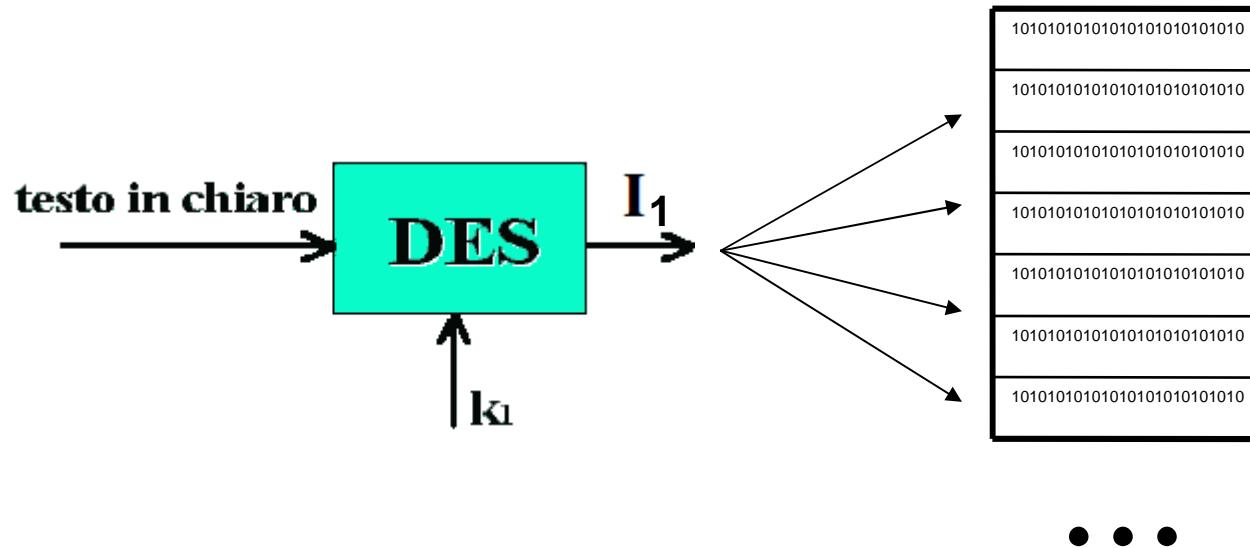
# Attacco meet-in-the-middle (1)

- Conosco un blocco di testo in chiaro e il corrispondente blocco di testo cifrato
- Spezzo l'algoritmo in due



# Attacco meet-in-the-middle (2)

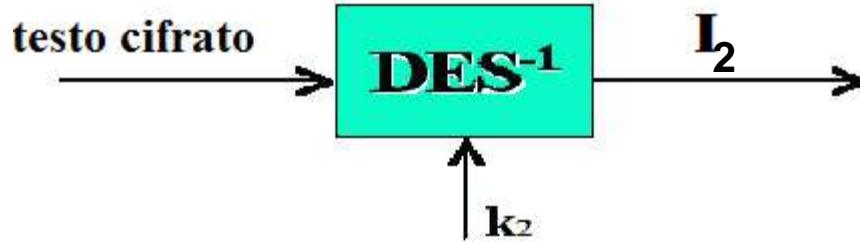
- Cifratura esaustiva con le  $2^{56}$  chiavi



- Il risultato in una enorme tabella hash

# Attacco meet-in-the-middle (3)

- Nella seconda parte uso DES<sup>-1</sup>



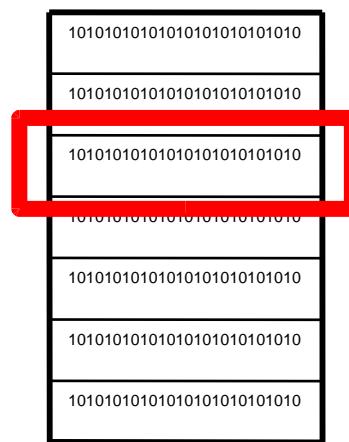
10101010101010101010101010
10101010101010101010101010
10101010101010101010101010
10101010101010101010101010
10101010101010101010101010
10101010101010101010101010
10101010101010101010101010

• • •

- Cerco il risultato nella tabella

# Attacco meet-in-the-middle (4)

- Se durante il secondo passo trovo un  $I_2$  uguale ad un record della tabella...



- La chiave che ha prodotto  $I_1$  e quella che ha prodotto  $I_2$  sono quasi sicuramente le chiavi usate da 2DES per cifrare il testo in chiaro

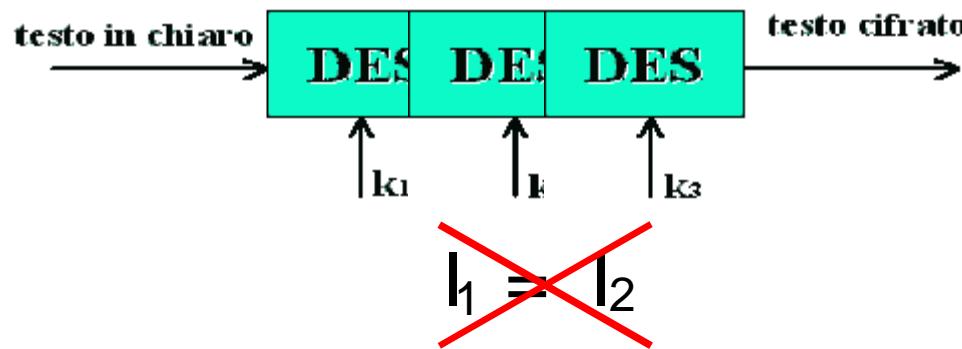
# Complessità dell'attacco

- 2 attacchi brute force al DES:

$$2^{56} + 2^{56} = 2^{57}$$

- $2^{56}$  ricerche nella tabella
- Complessità temporale totale  $\approx 2^{57}$
- Complessità spaziale non trascurabile  
 $2^{56}$  record
- Proviamo ad aumentare la sicurezza del 2DES

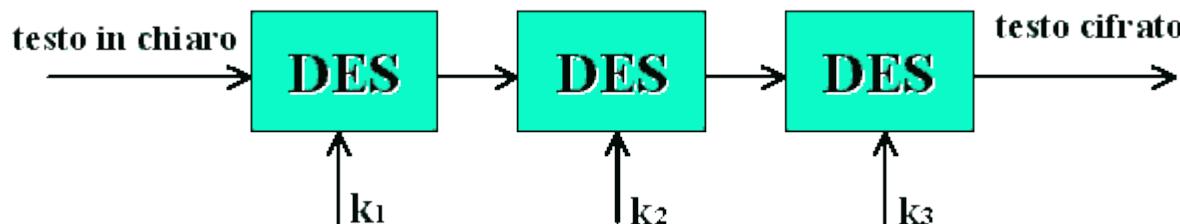
# Un miglioramento: 3DES



- **3DES:** L'attacco meet-in-the-middle non è più realizzabile
- Posso cercare di spezzare in due parti il 3DES, ma una delle due necessita la ricerca su uno spazio di  $2^{112}$  chiavi

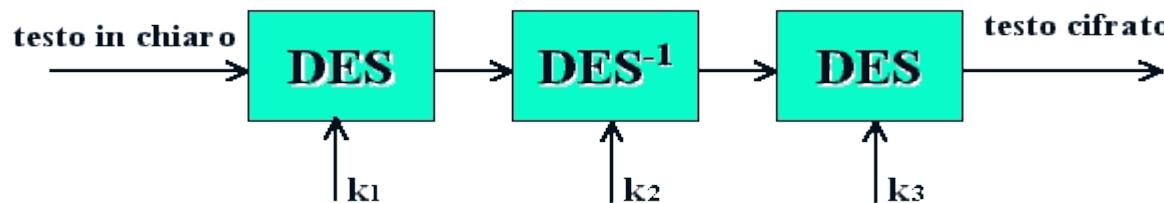
# 3DES: due modalità

## ■ Modalità EEE



## ■ Modalità EDE

### Cifratura



# Rompere 3DES (?)

- Un attacco brute force al DES è un problema NP-completo
  - Non è dimostrato che  $P \neq NP$
  - C'è ancora speranza di poter rompere il 3DES
- 
- 2005: Il 3DES è abbastanza sicuro? Sì

# Bibliografia

- **Bruce Scheider.** *Applied Cryptography*. Wiley & Sons, 1996.
- **Wenbo Mao.** *Modern Cryptography, Theory And Practice*. Prentice, 2003.
- **Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone.** *Handbook Of Applied Cryptography*. MIT Press, 1996
- **A Tanenbaum** *Reti Di Calcolatori*. Prentice-Hall, Pearson, 2003.
- **Keith W. Campbell, Michael J. Wiener.** *DES is not a Group*. Rapporto tecnico Bell-Northern Research, 1992