Data Encryption Standard

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

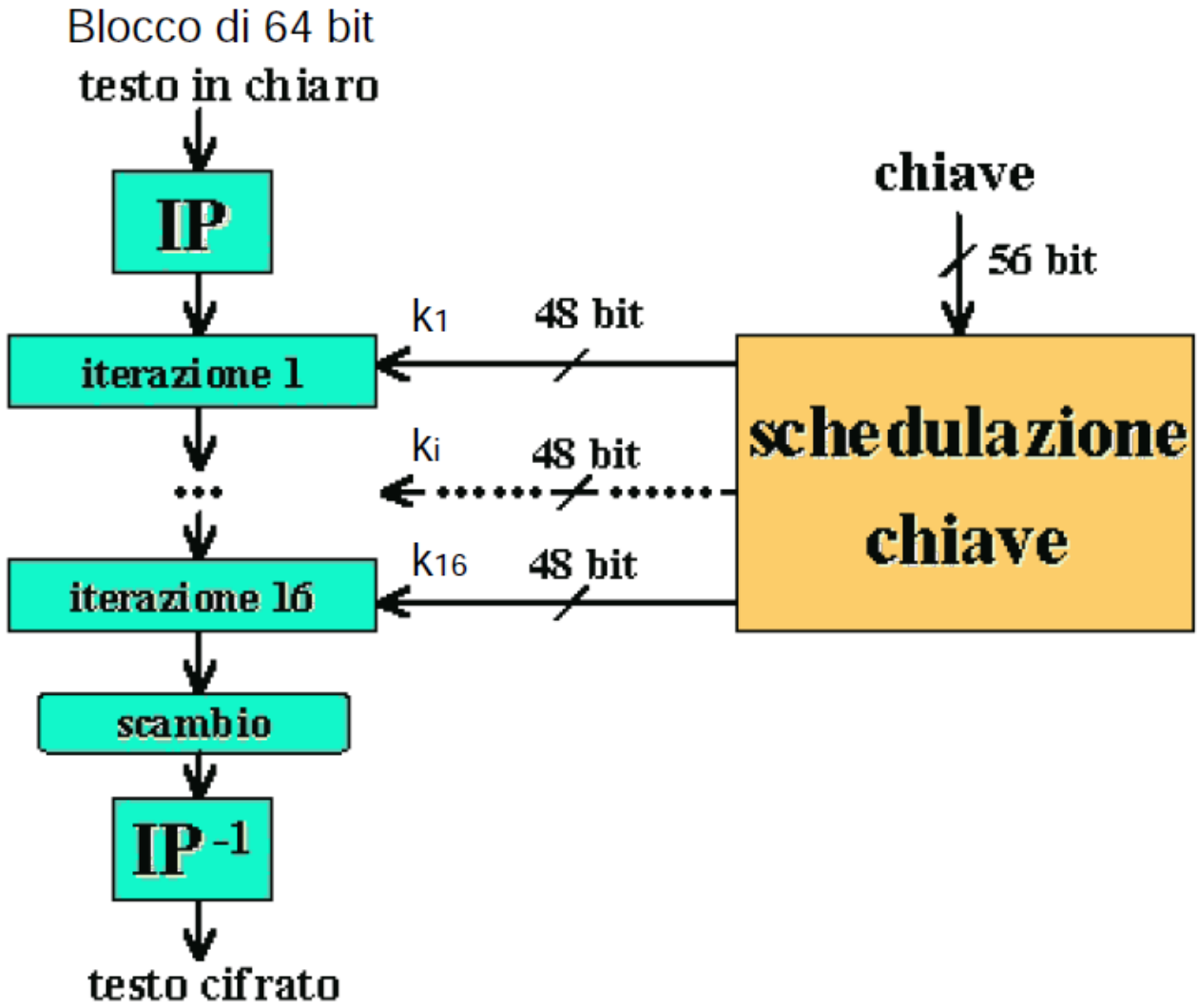
Proprietà desiderabili di un algoritmo di cifratura secondo Shannon.

- 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



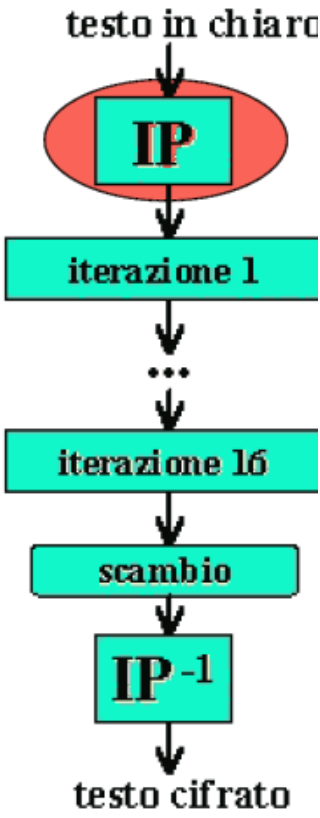
La Struttura

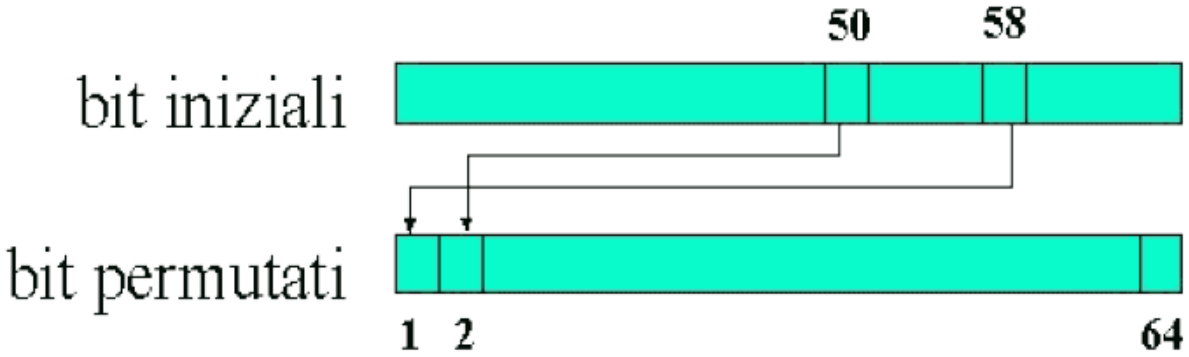
- Un blocco di testo in chiaro (64 bit) viene permutato dal blocco IP, il risultato di questa permutazione andrà in ingresso alla prima iterazione.
- In ogni iterazione il blocco di 64 bit viene opportunamente mescolato con una Kiesima chiave.
- In tutto 16 iterazioni.
- Dopo le 16 iterazioni al testo 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





IP

Dati permutati

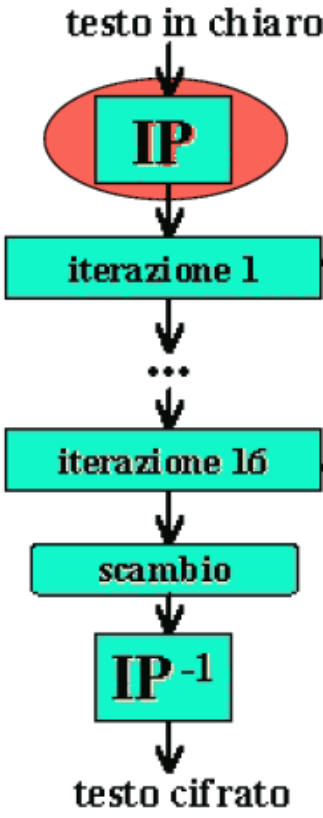
58	50	42	34	26	18	10	02
60	52	44	36	28	20	12	04
62	54	46	38	30	22	14	06
64	56	48	40	32	24	16	08
57	49	41	33	25	17	09	01
59	51	43	35	27	19	11	03
61	53	45	37	29	21	13	05
63	55	47	39	31	23	15	07

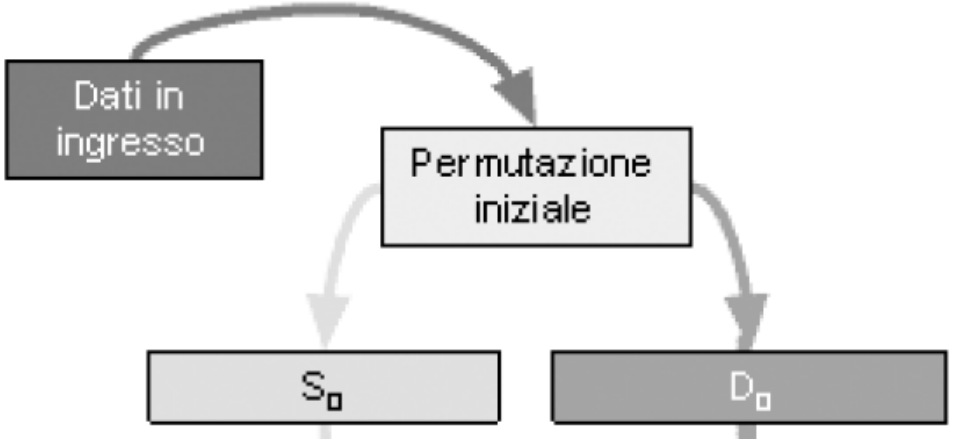
32bit

Parte sinistra
S_0

32bit

Parte destra
D_0



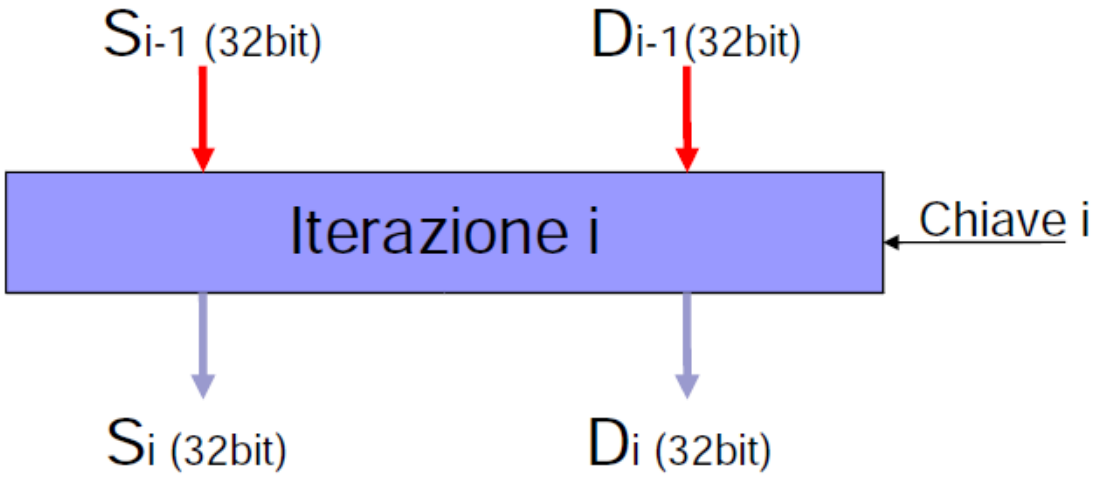


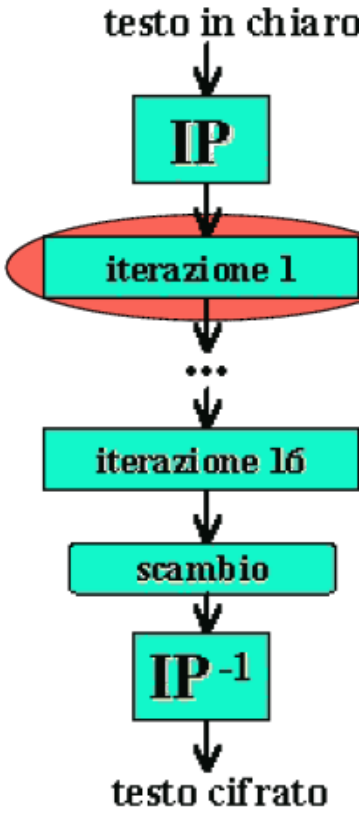
Input Iterazione 1

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 cosi via.
- Al termine della permutazione il blocco viene suddiviso in due sotto-blocchi da 32 bit ciascuno S_0 e D_0, questi saranno l'input della prima iterazione.

Iterazione

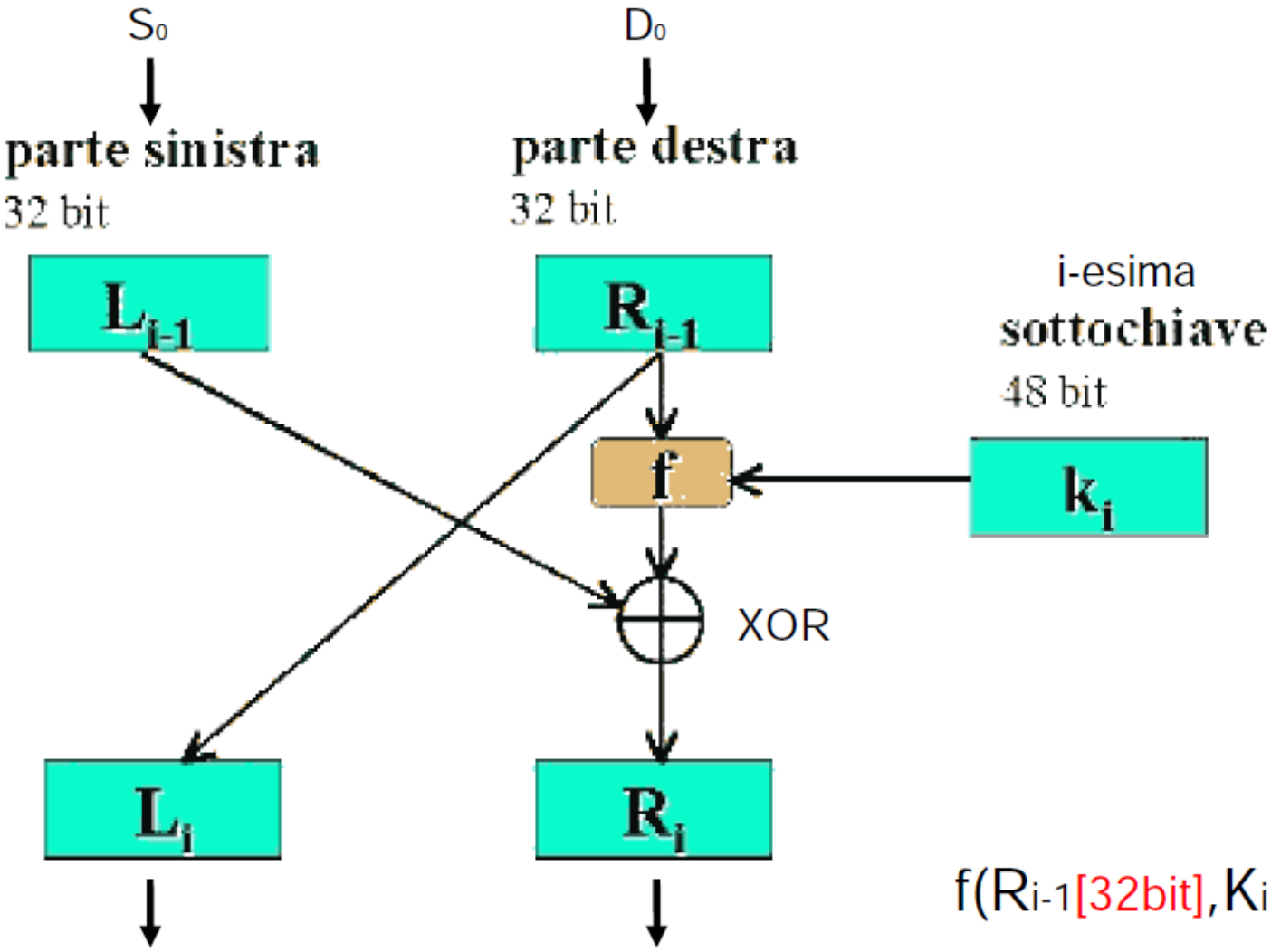

Si-1 (32bit)
Di-1(32bit)
Iterazione i
Chiave i
Si (32bit)
Di (32bit)

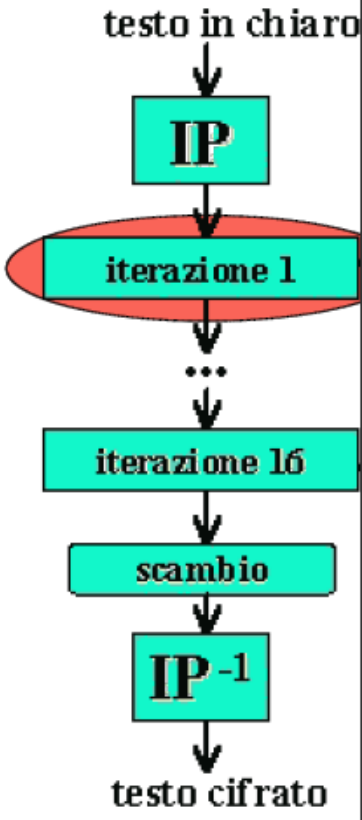

testo in chiaro
IP
iterazione 1
...
iterazione 16
scambio
IP -1
testo cifrato

Iterazione

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

Iterazione





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

Passi del ROUND

QUINDI:

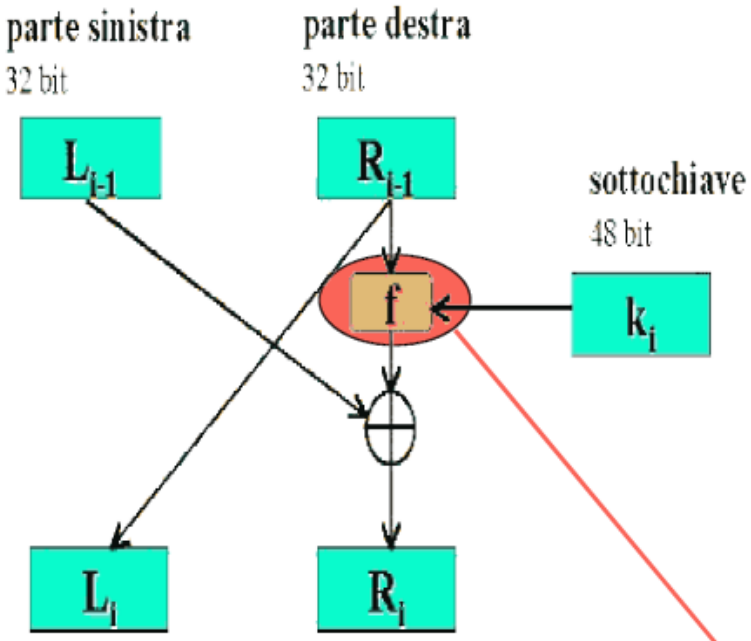
- $L_i = R_{i-1}$
- $R_i = L_{i-1}$ XOR Feistel(R_{i-1}, K_i)

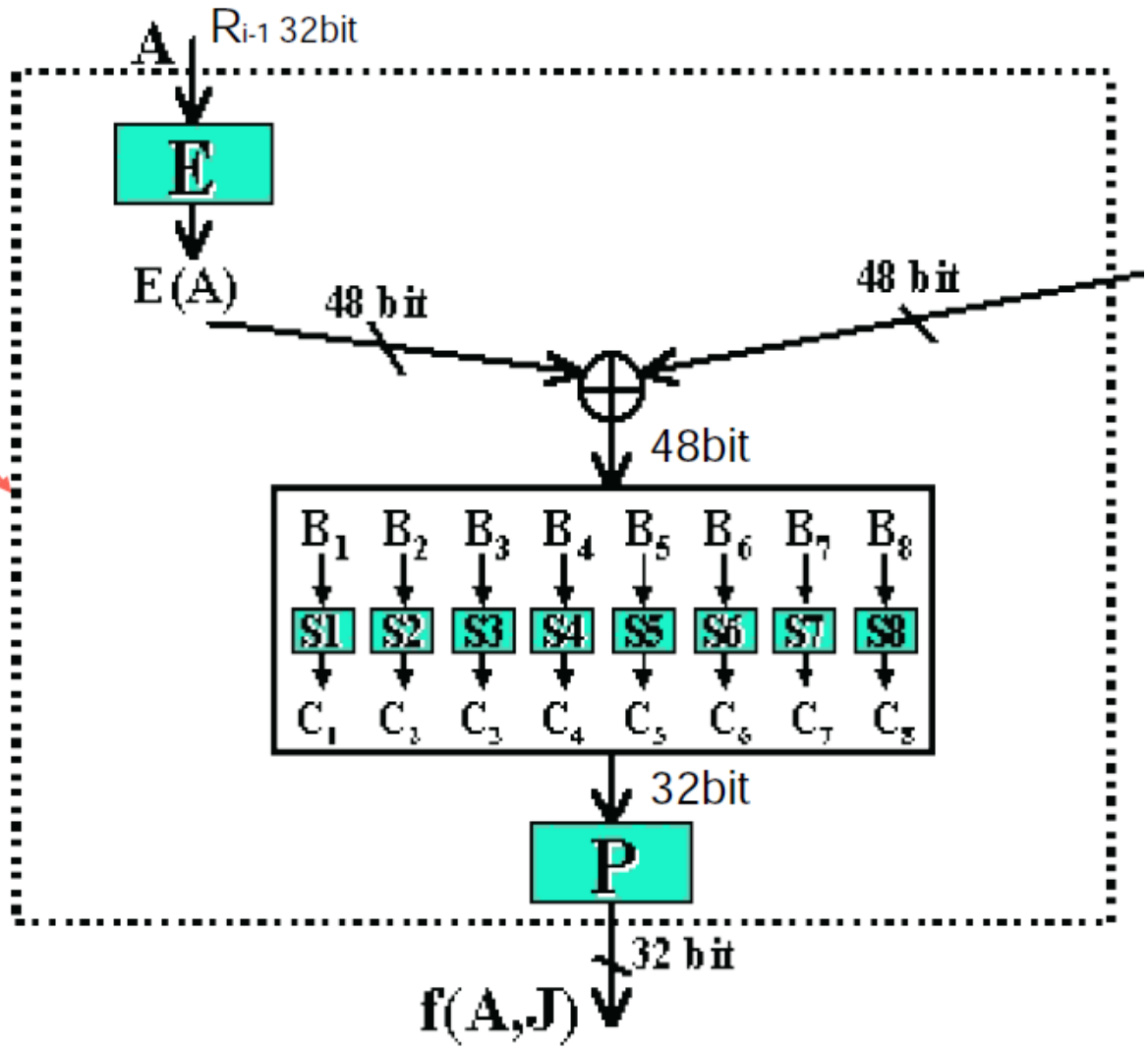
Iterazione

- Elaborazione dei blocchi nell'iterazione.
  - I due blocchi di Output Li e Ri sono ottenuti in questo modo :
    - Il nuovo blocco Li è il vecchio Ri-1
    - Il nuovo blocco Ri è ottenuto mettendo in XOR il vecchio blocco Li-1 con l'output della funzione F
  - La Funzione F combina il vecchio blocco di destra Ri-1 con la chiave Ki

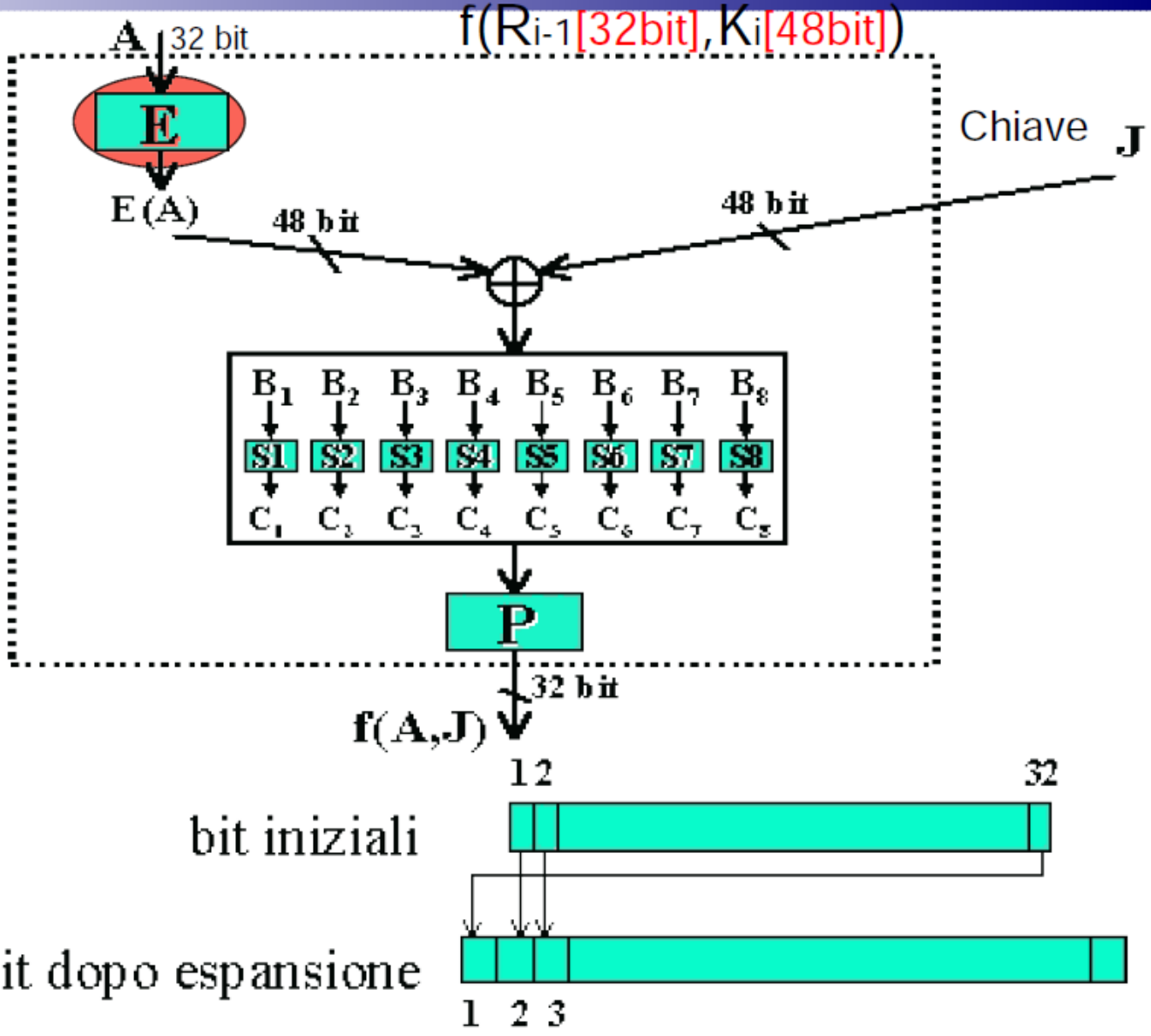
La Funzione f

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





Il 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

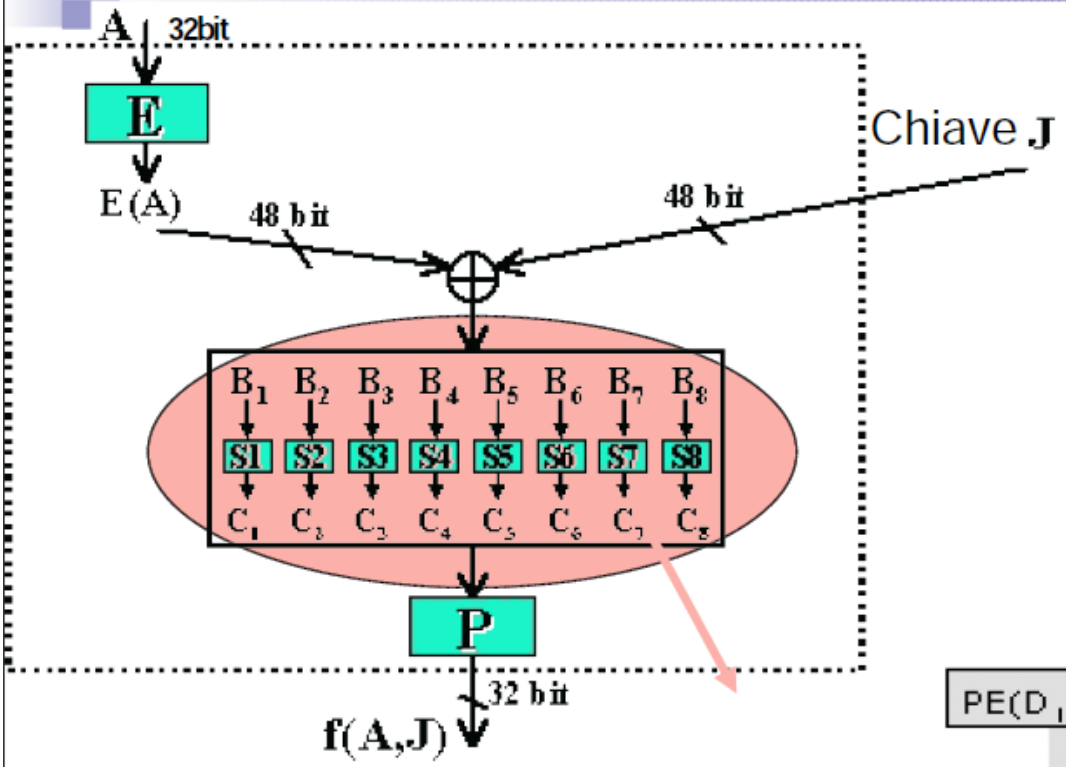
Alcuni bit vengono ripetuti

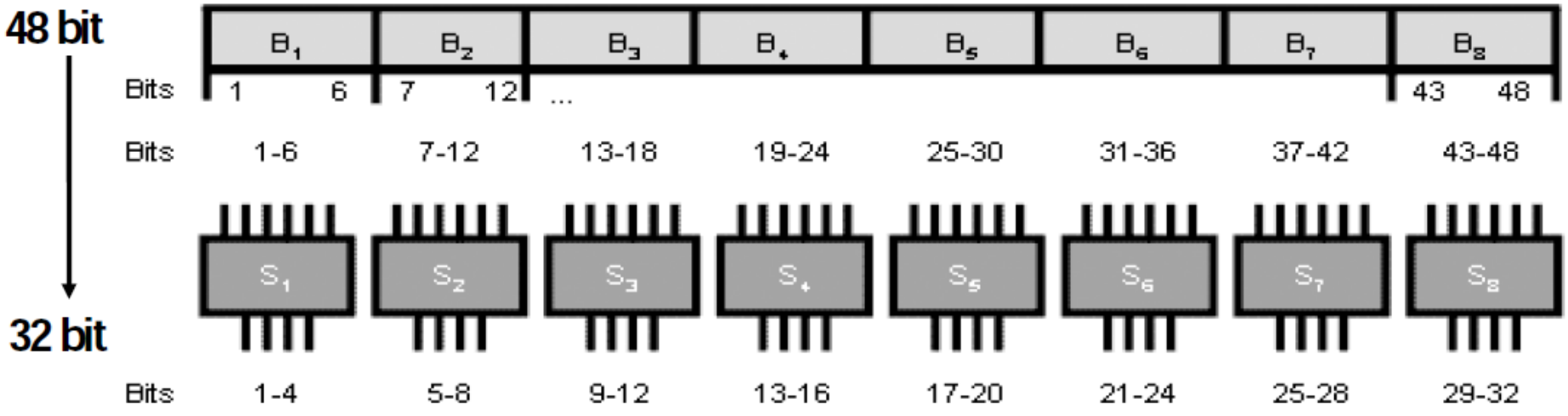
Funzione F

- Inizialmente i 32 bit del vecchio blocco di destra Ri-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 Ki chiave anch'essa di 48 bit.
- La nuova sequenza B di 48 bit ottenuta viene suddivisa in 8 sotto-blocchi da 6 bit ciascuno.
- La stringhe Bi di 6 bit vengono date in ingresso alle otto funzioni SBOX.

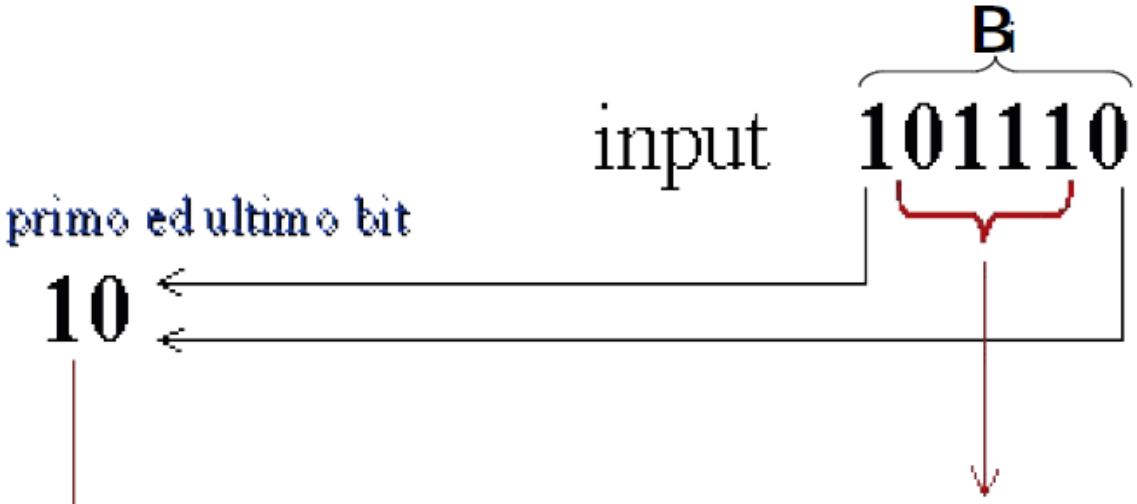
Le S-Box

Parte cruciale su cui si basa la sicurezza del cifrario





Le S-Box



	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	0110	0111	1111
00	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
01	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
10	4	1	7	8	13	6	2	11	15	12	9	7	3	10	5	0
11	15	12	10	2	4	9	1	7	5	11	3	14	10	0	6	13

Box S1

output 11 in binario = **1011**

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 cosi 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.

Le 8 S-Box

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

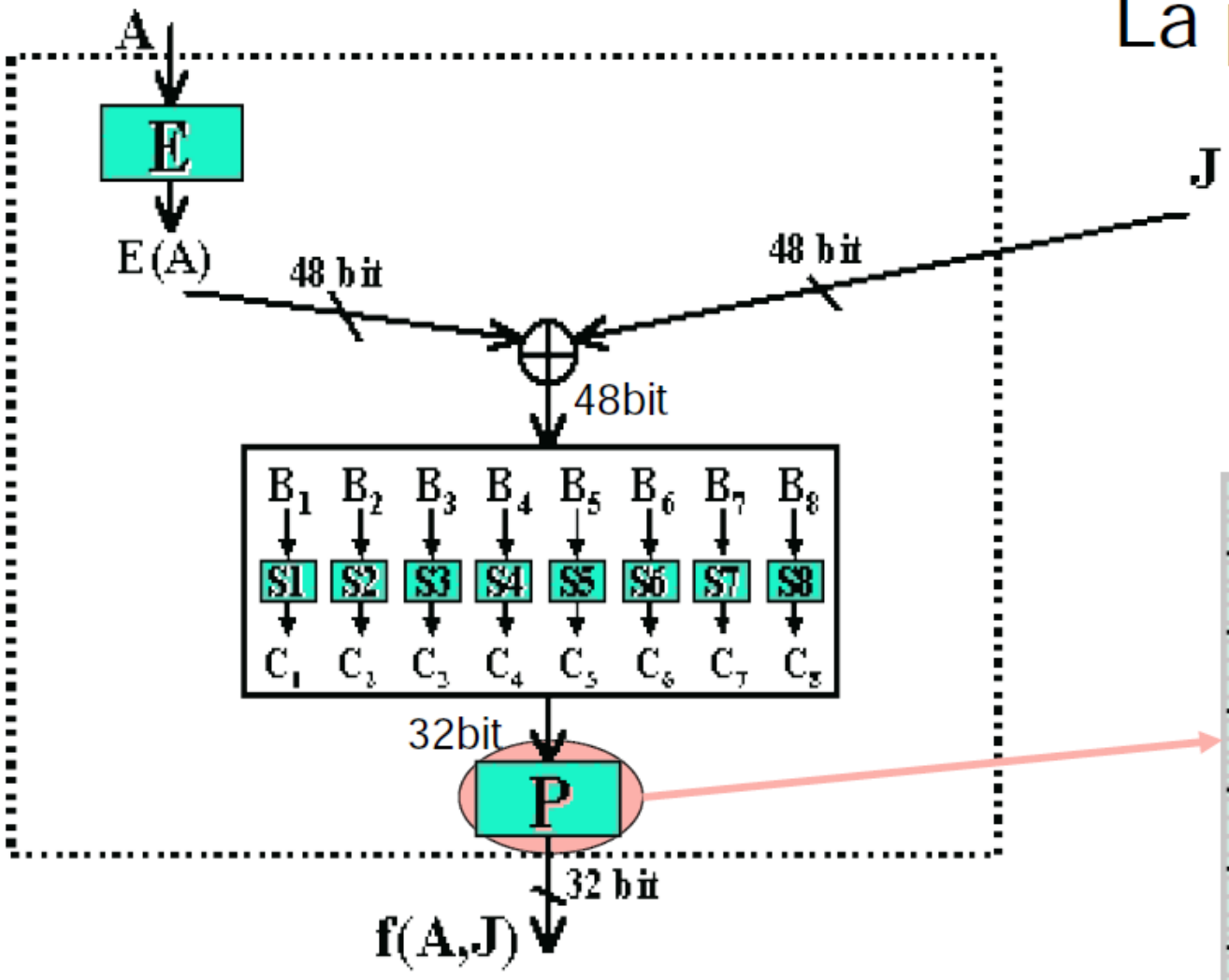
S_5															
2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

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

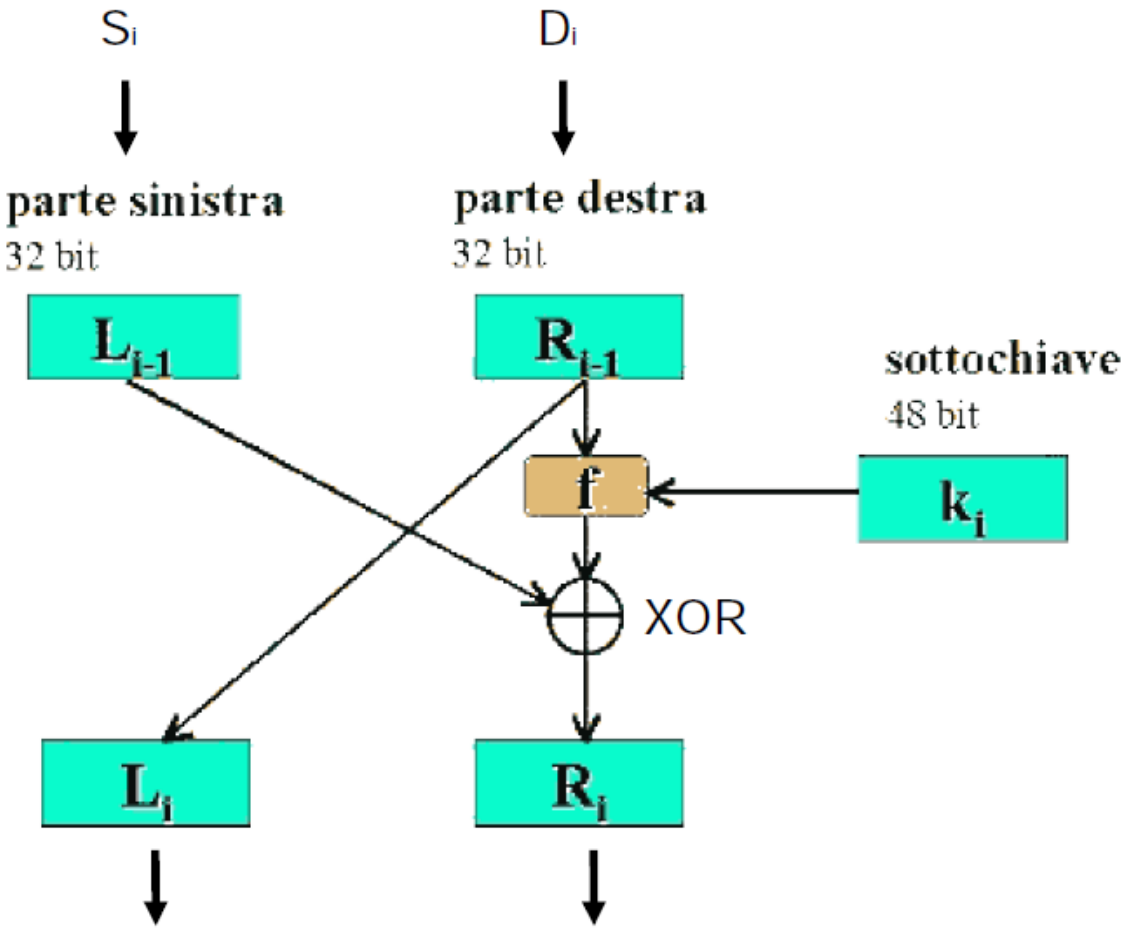
S_8															
13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

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

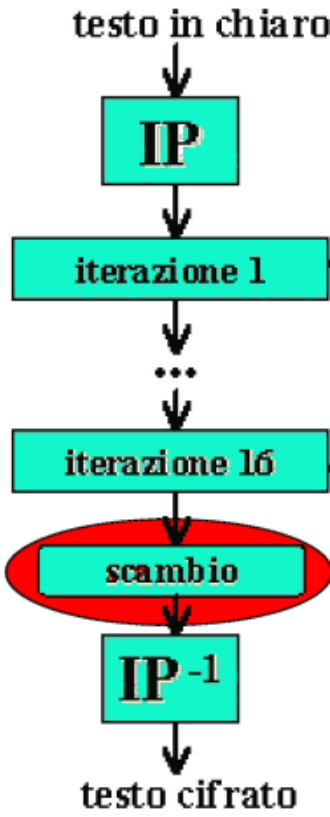

Si
Di
parte sinistra
32 bit
parte destra
32 bit
Li-1
Ri-1
sottochiave
48 bit
f
ki
XOR
Li
Ri
Input Iterazione successiva


testo in chiaro
IP
iterazione 1
...
iterazione 16
scambio
IP -1
testo cifrato

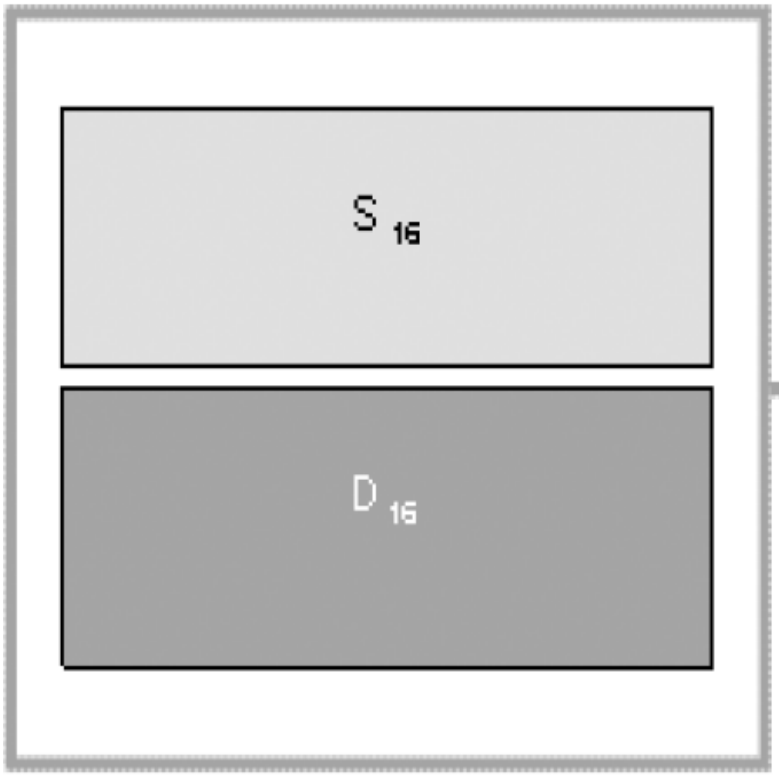
Scambio

- 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.

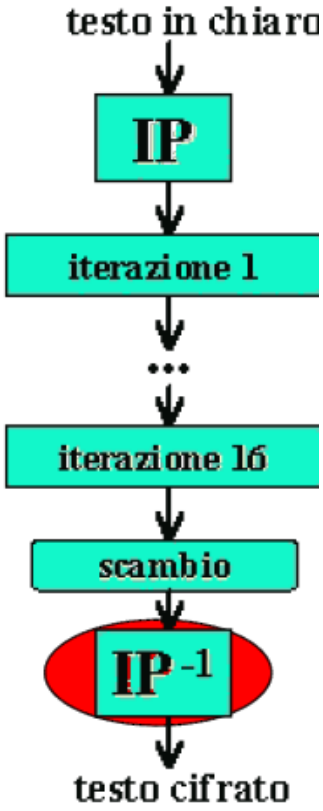


IP^{-1}

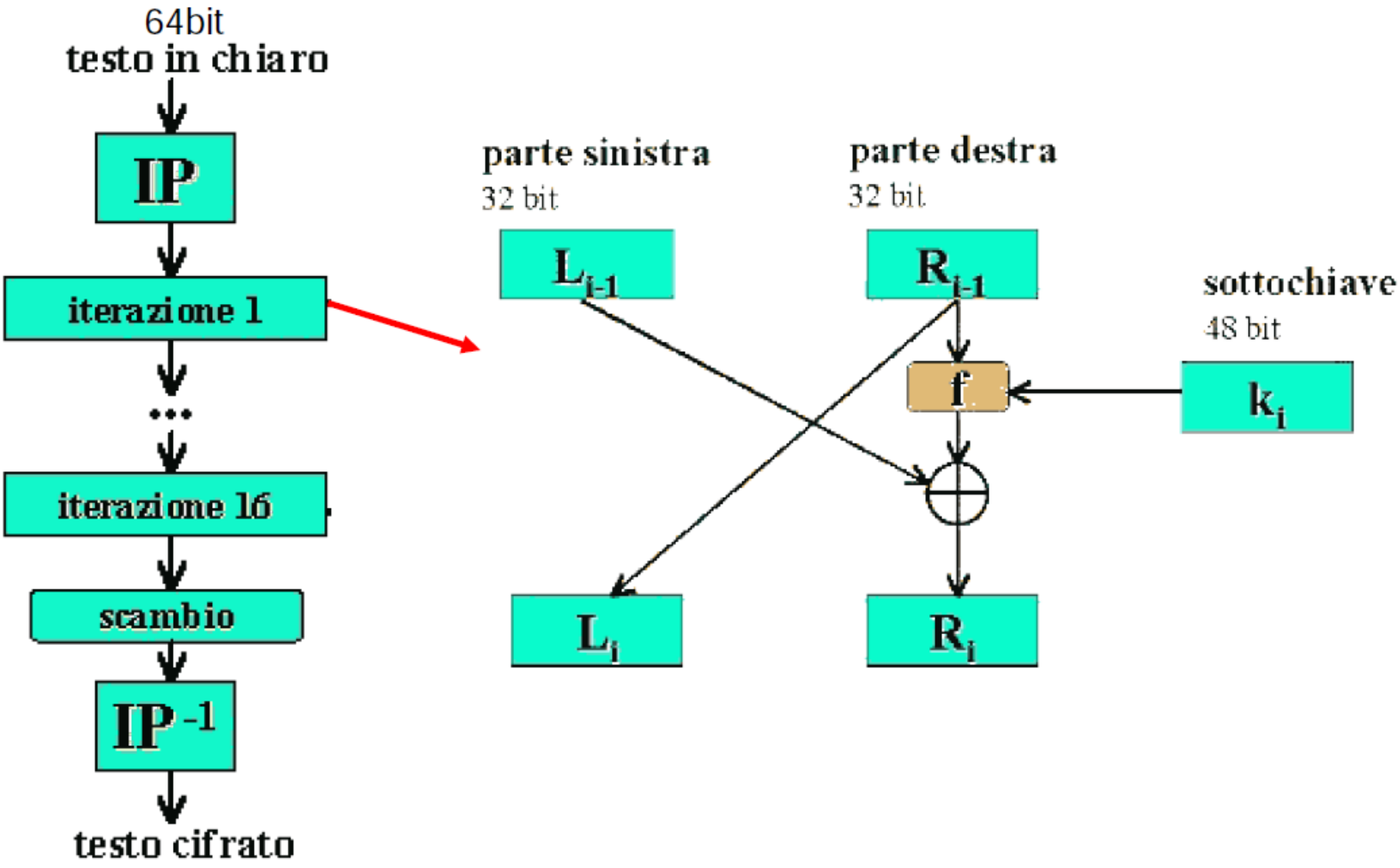


Permutazione finale

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

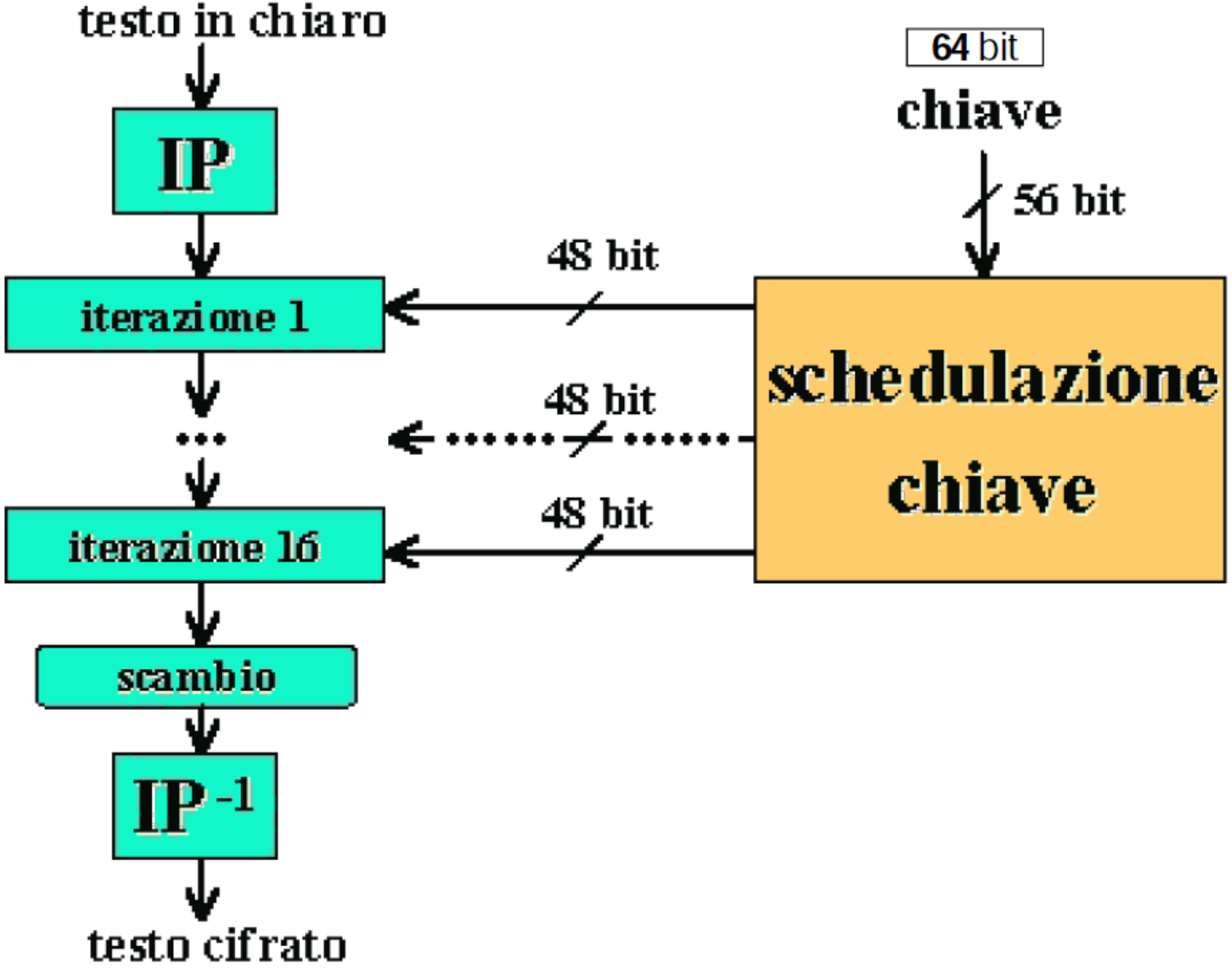


Ricapitolando…



Generazione delle chiavi e decifrazione

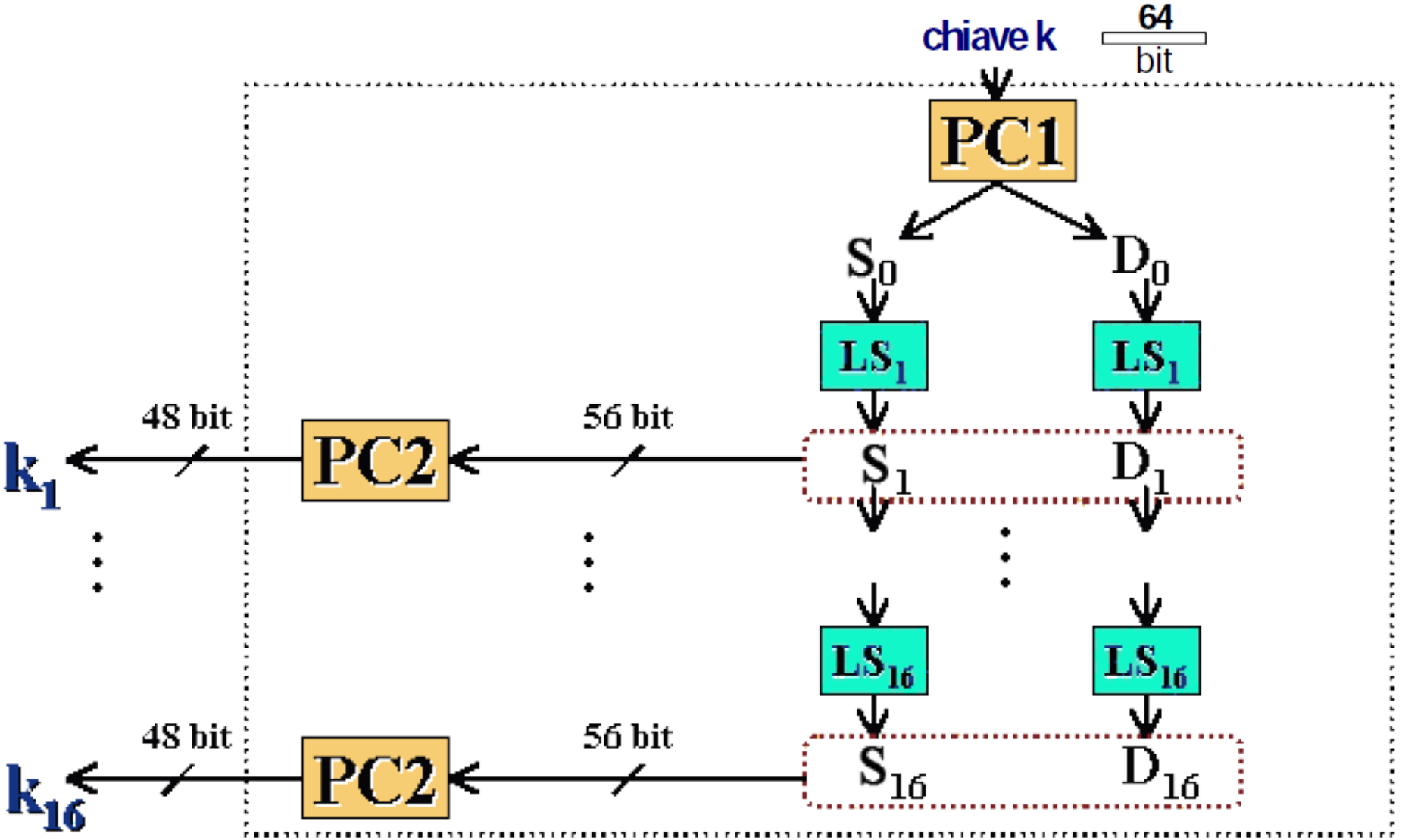
Generazione delle Chiavi (1)



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

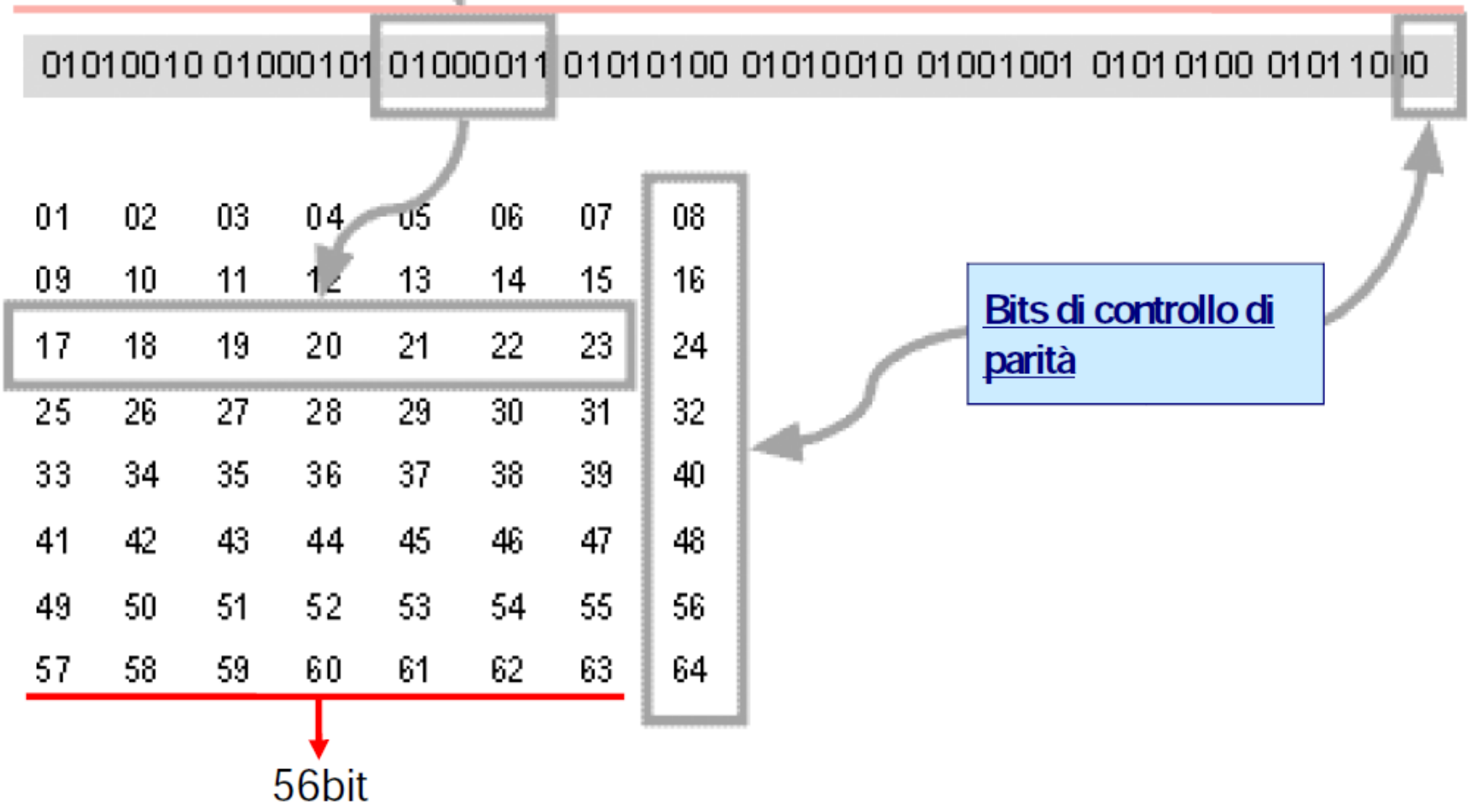
- Elementi fondamentali sono:
  - PC1 (Permutate Choise 1)
  - LS_i (Left Shift i)
  - PC2 (Permutate Choise 2)

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***



Bits di controllo di parità

PC1 (1)

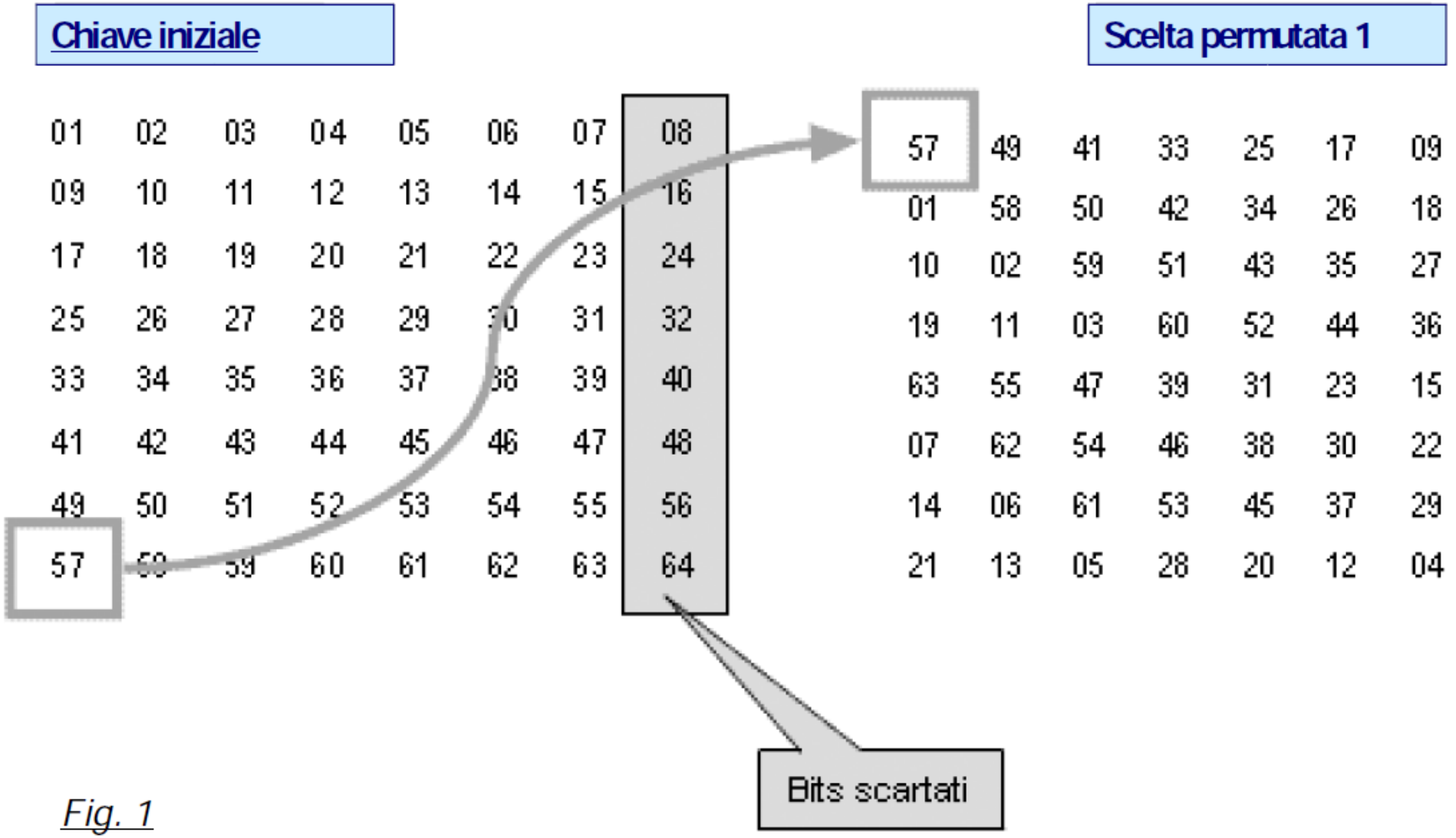
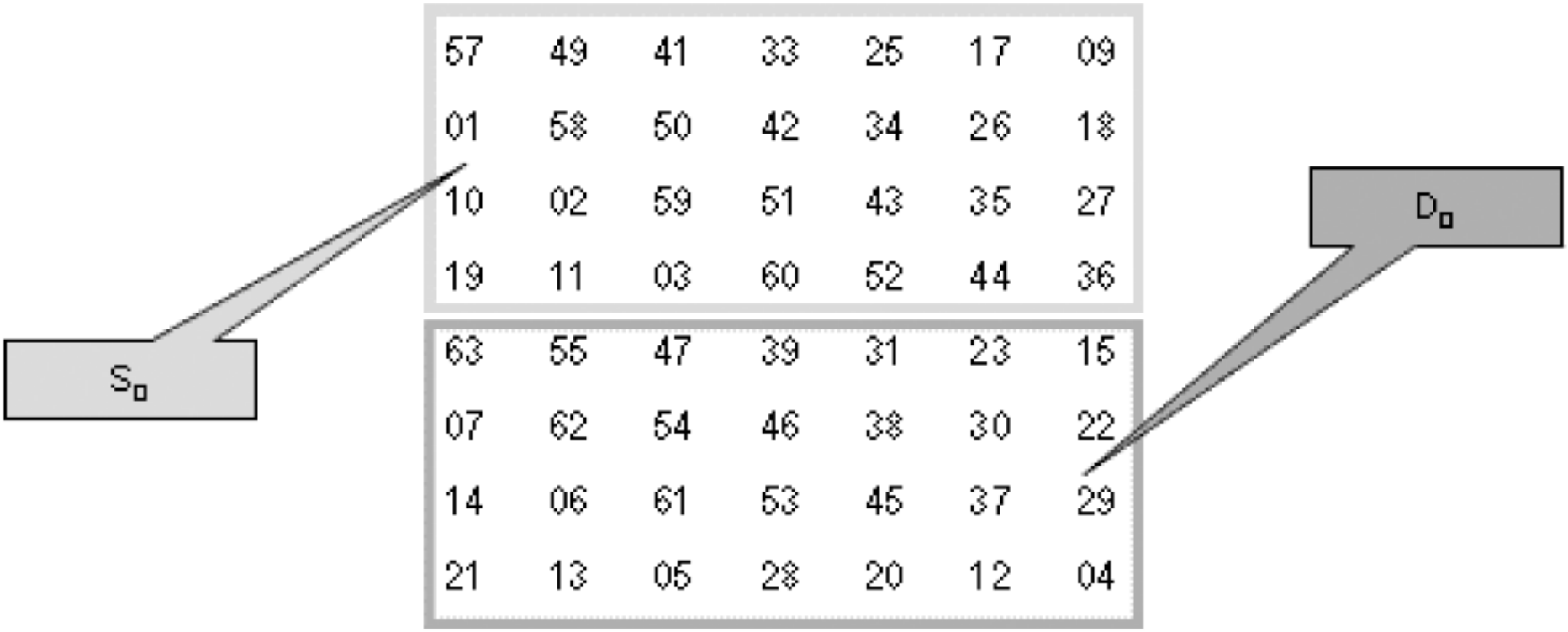

Chiave iniziale
01 02 03 04 05 06 07 08
09 10 11 12 13 14 15 16
17 18 19 20 21 22 23 24
25 26 27 28 29 30 31 32
33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48
49 50 51 52 53 54 55 56
57 58 59 60 61 62 63 64
Scelta permutata 1
57 49 41 33 25 17 09
01 58 50 42 34 26 18
10 02 59 51 43 35 27
19 11 03 60 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
Bits scartati

Fig. 1

PC1 (2)



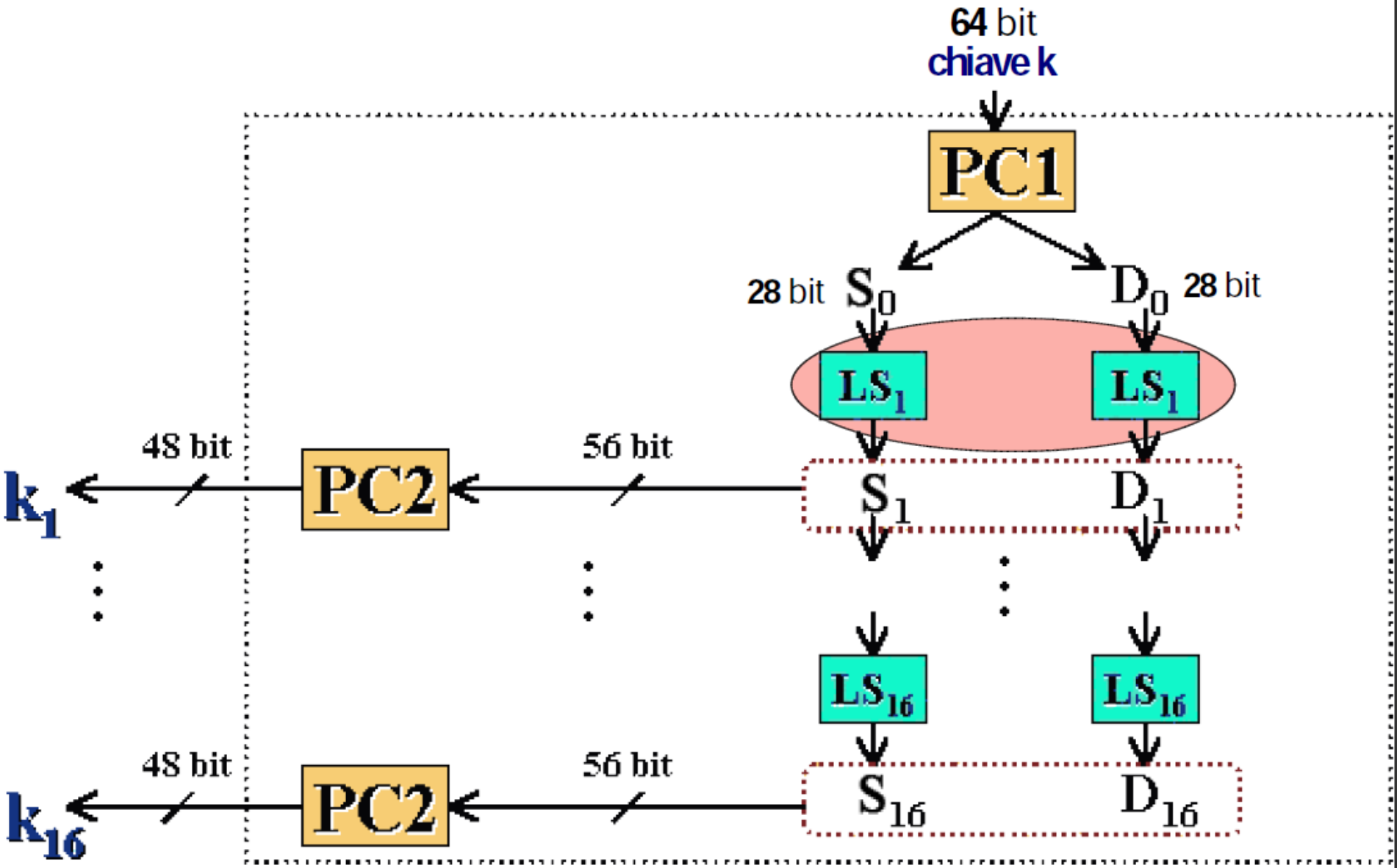
- Vengono prodotte due semi-chiavi, S_0 e D_0

PC1 (spiegazioni)

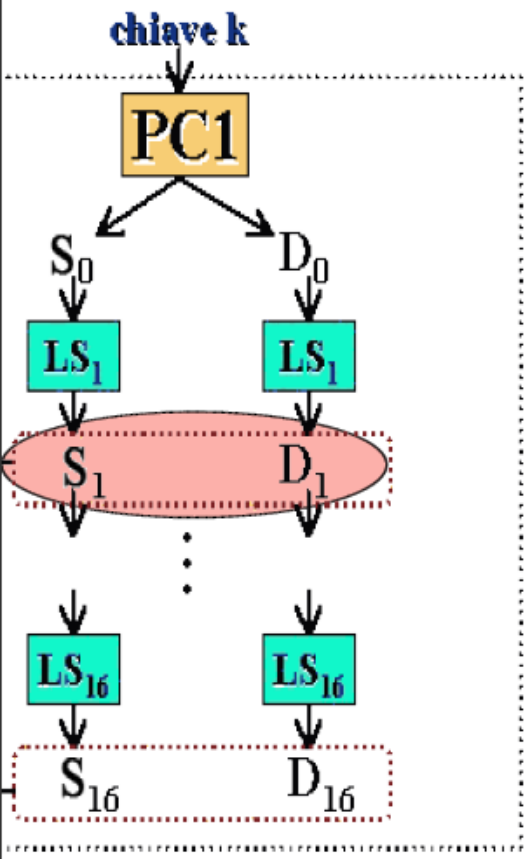
- Accetta in ingresso la chiave di 64 bit assegnata al processo di cifratura. Tale chiave contiene 8 bit per il controllo di parità (uno per ogni byte). Non contenendo informazione i bit di parità possono essere eliminati, portando la chiave da 64 a 56 bit. In una rappresentazione matriciale della chiave (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 ***S0*** e ***D0*** (sotto-chiave Sinistra e sotto-chiave Destra)


64 bit
chiave k
PC1
28 bit
S_0
D_0
28 bit
LS_1
LS_1
48 bit
56 bit
PC2
k_1
S_1
D_1
LS_{16}
LS_{16}
48 bit
56 bit
PC2
k_{16}
S_{16}
D_{16}

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

Tabella 1

= 28

Shift ciclico

S_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

D_1

LSi dettagli

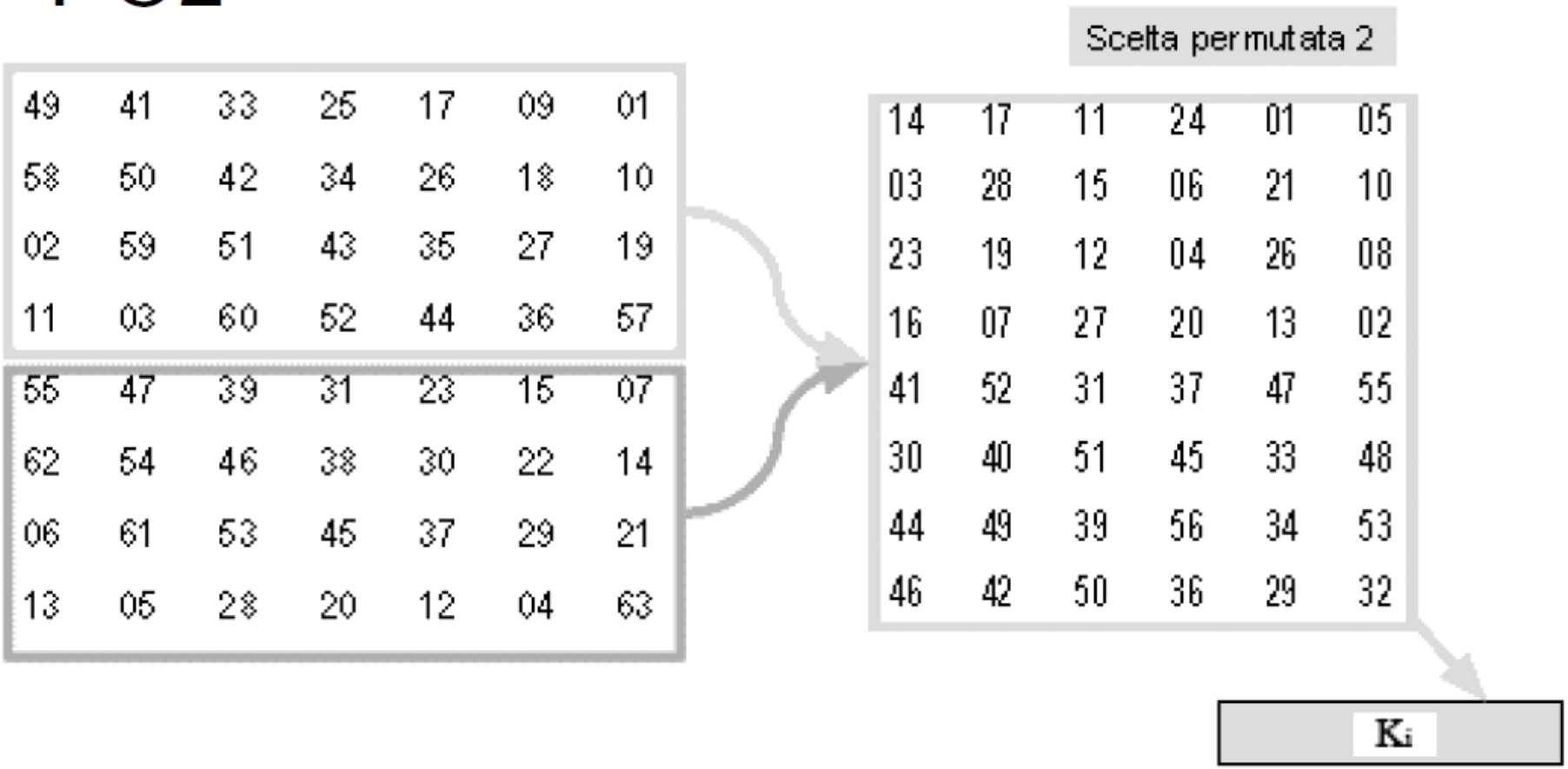
- Ad ogni iterazione le due semichiavi S_i e D_i di 28 bit vengono elaborate dal blocco LS_i in parallelo.

- LS_i 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.

- LS_i produce in output due nuove semichiavi S_{i+1} e D_{i+1}.

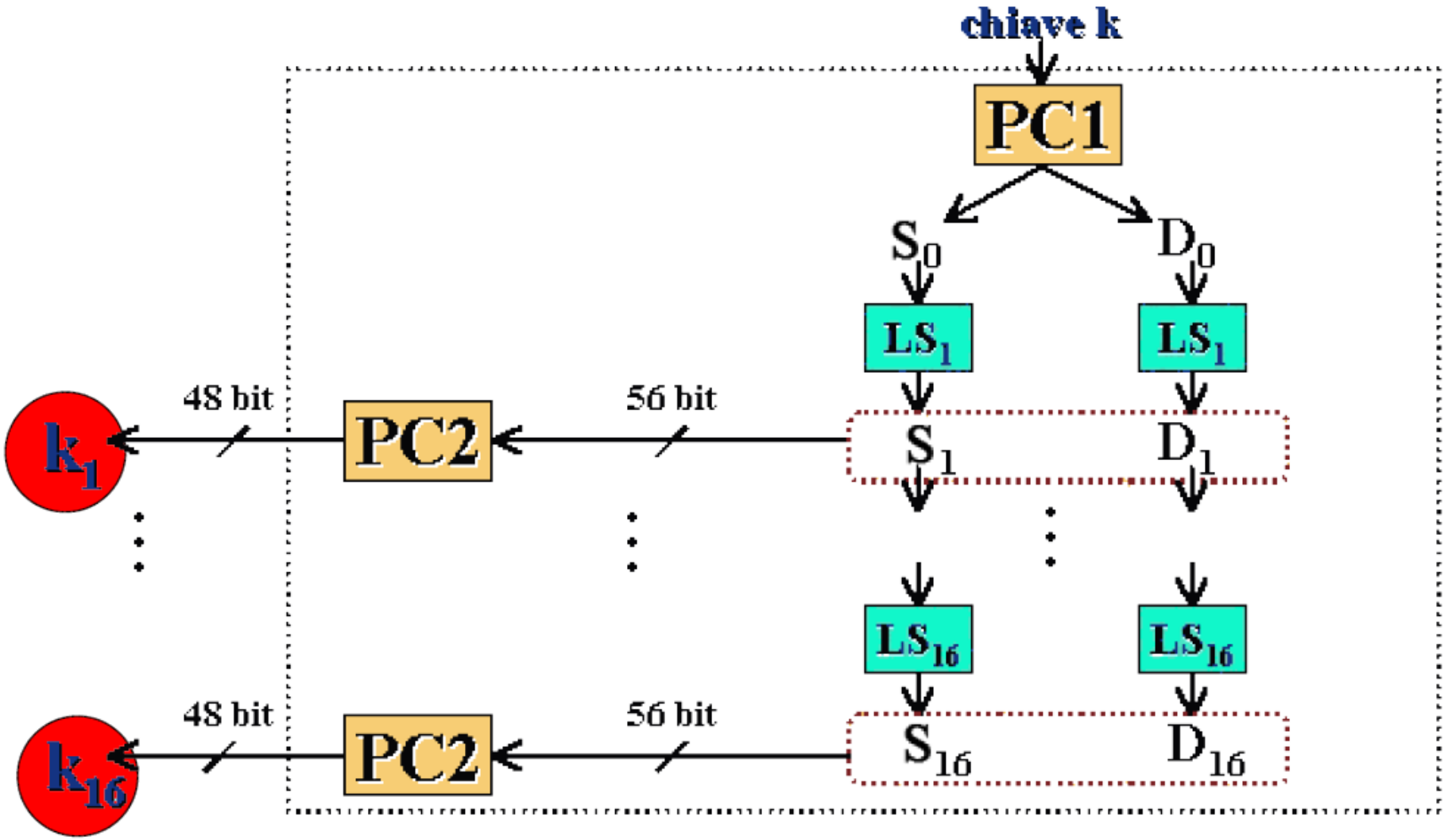
PC2



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)

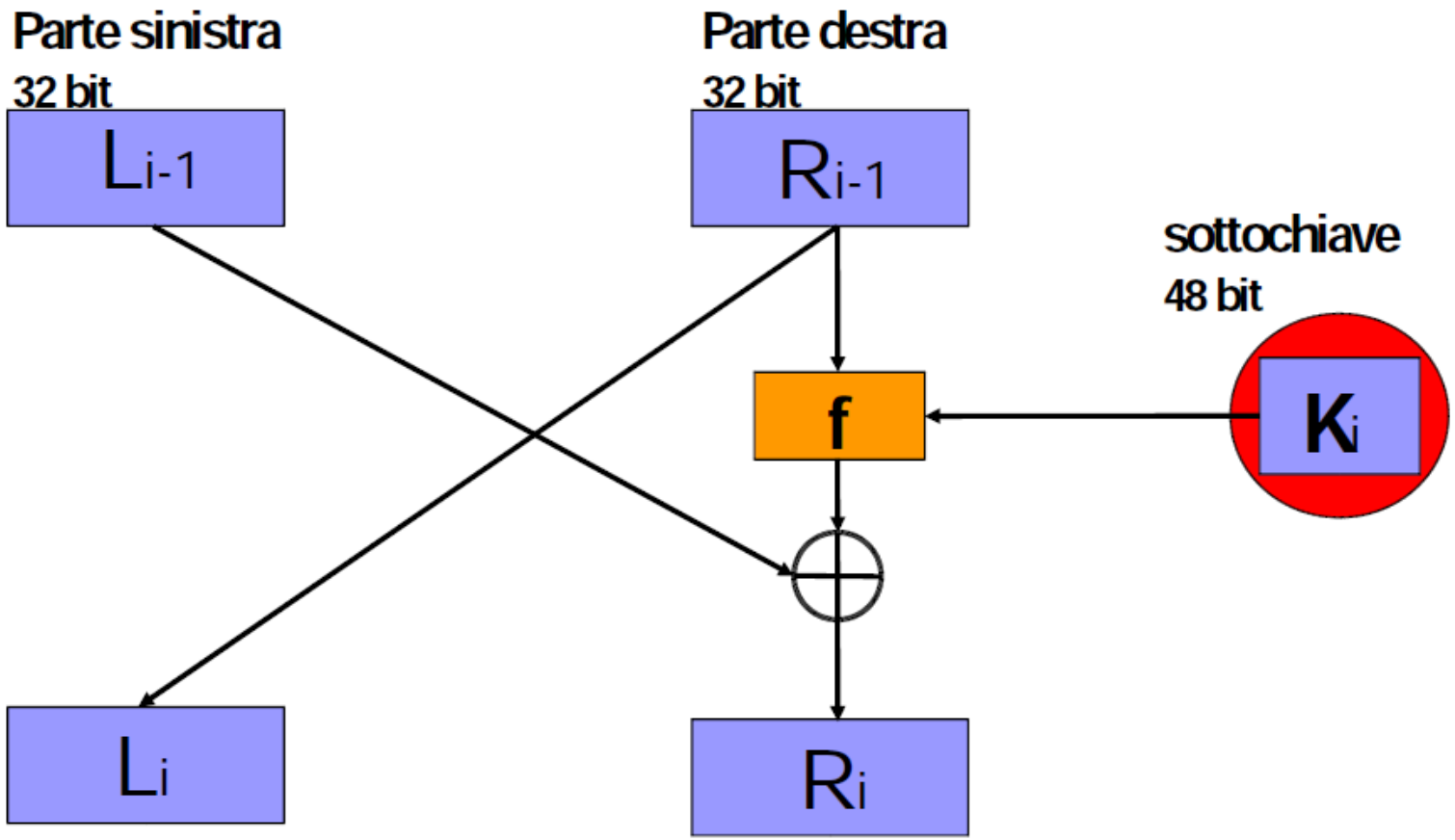
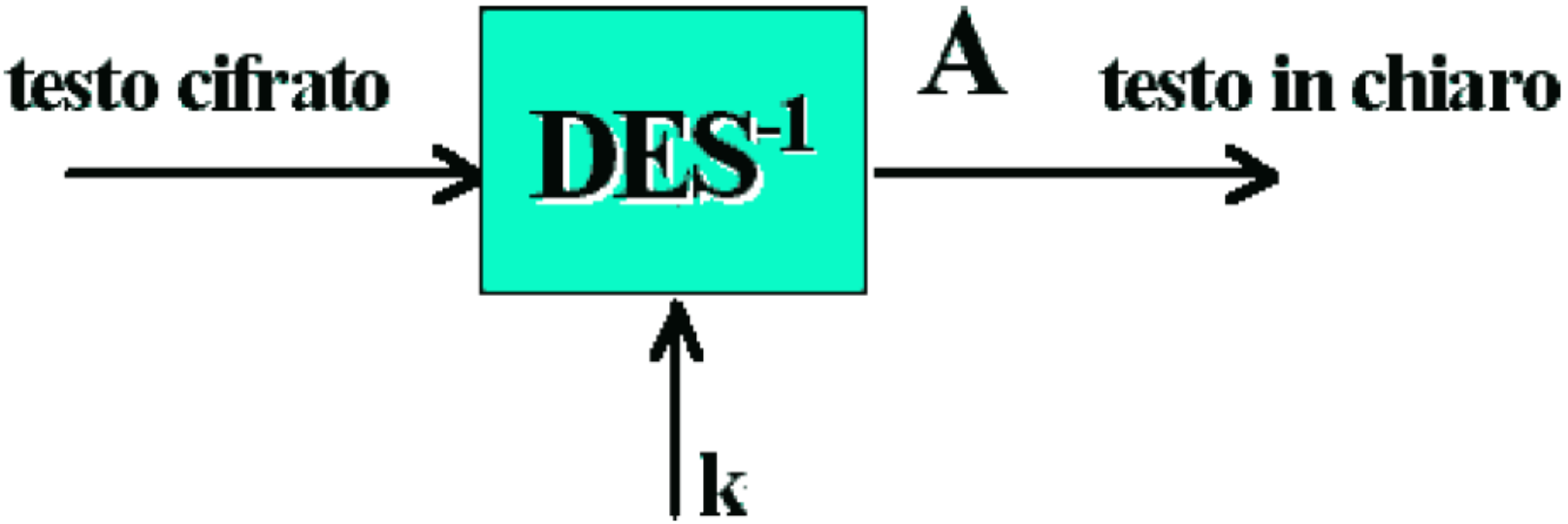


Fig. 3

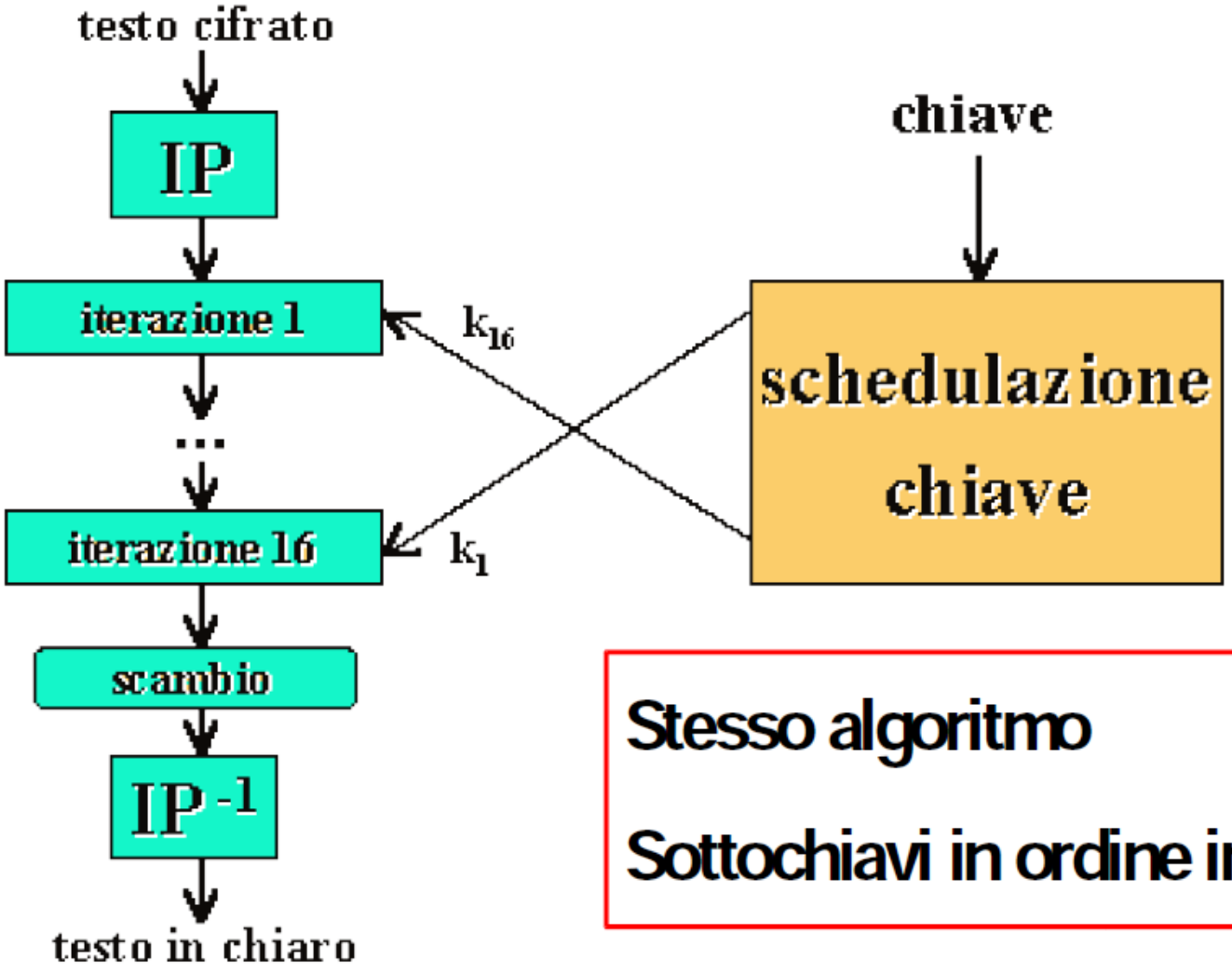
PC2 dettagli

- PC2 opera ad ogni itarazione 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)



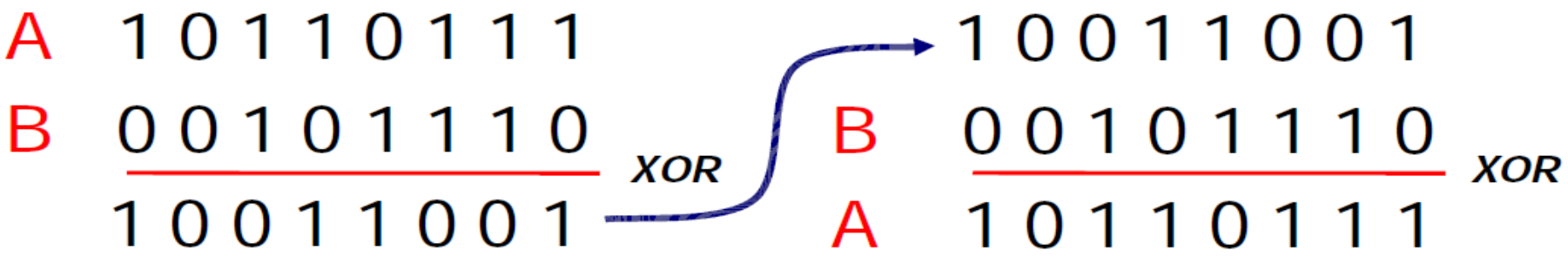
Stesso algoritmo

Sottochiavi in ordine inverso

Proprietà XOR

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

(A XOR B) XOR B = A
(A XOR B) XOR A = B



C=A XOR **B** ⟹ **A = C** XOR **B**

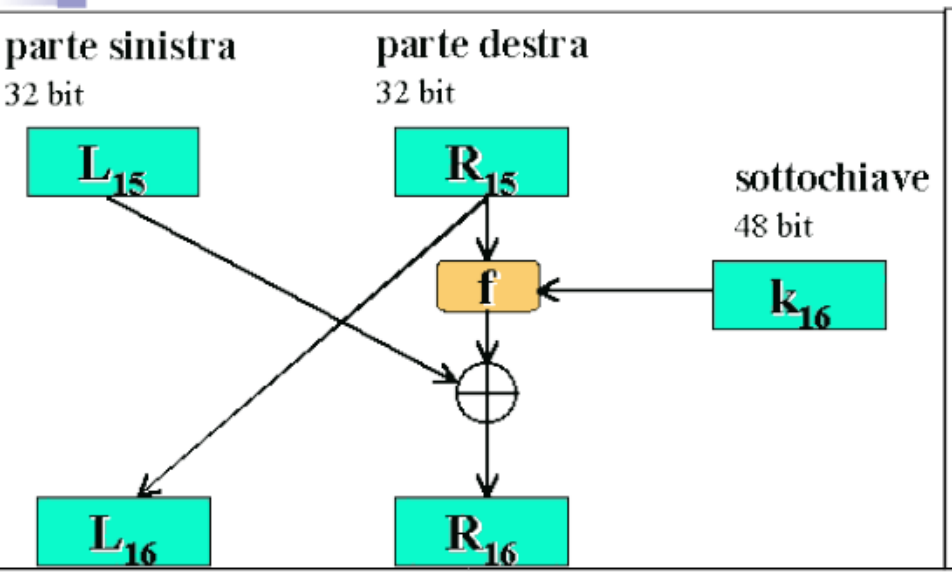


Fig.A (ultima iterazione cifratura)

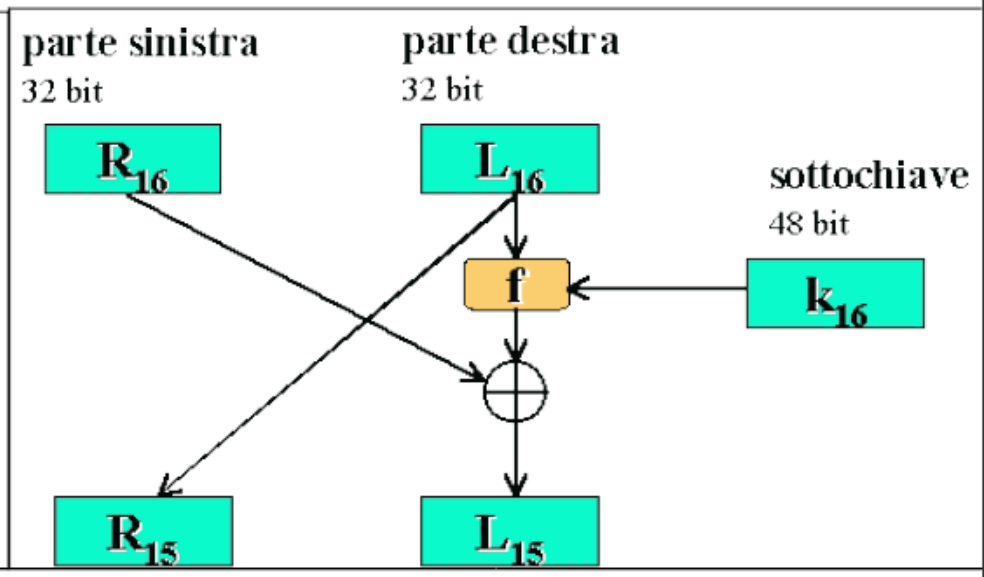


Fig.B (prima iterazione decifratura)

- Consideriamo il messaggio cifrato $R16\ L16$ a meno della permutazione finale IP^(-1). Dalla Figura A si ricavano le seguenti relazioni:

$$L16 = R15 \text{ e } R16 = L15 \oplus f(R15, K16)$$

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

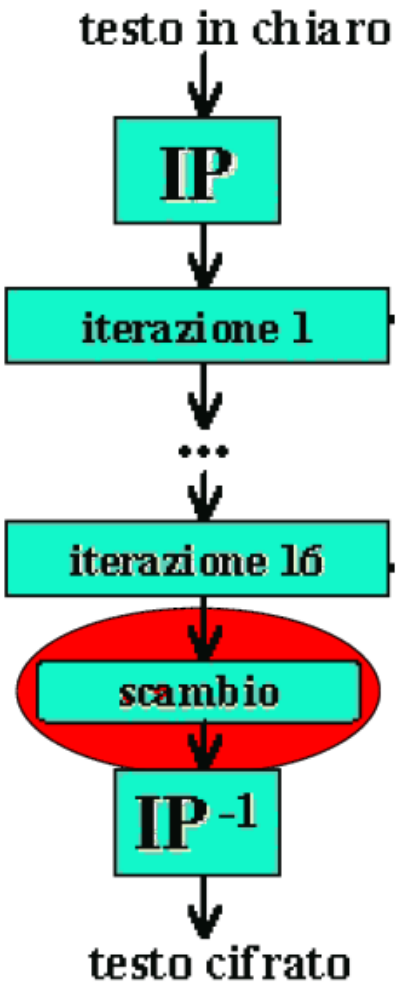
$$R15 = L16 \text{ e } L15 = R16 \oplus f(R15, K16) = R16 \oplus f(L16, K16)$$

- In conclusione: da $R16$ e $L16$ abbiamo ricavato $R15$ ed $L15$ (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:

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

- 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)

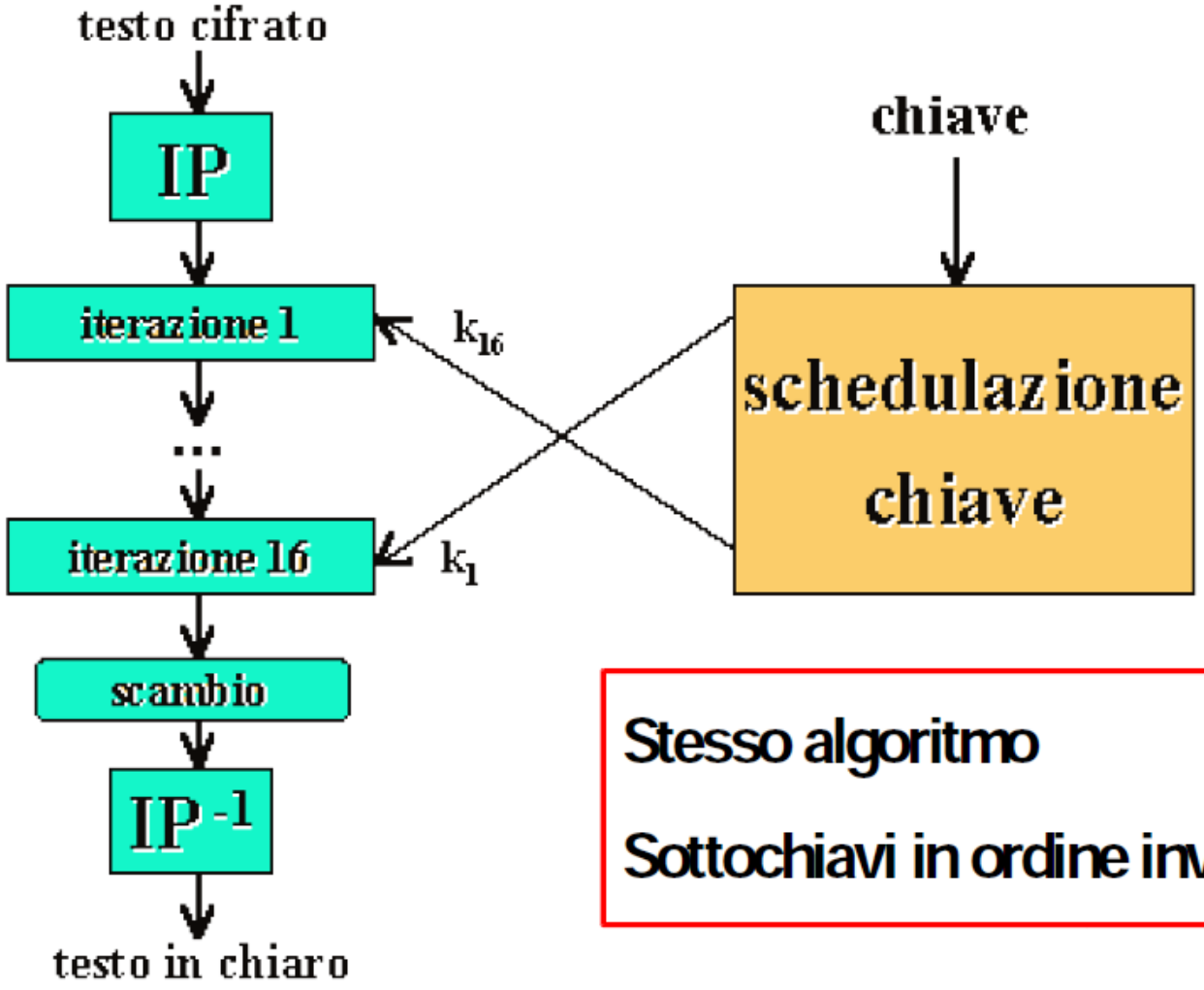
- Dal momento che per la fase di decifratura le chiavi devono essere schedulate in ordine inverso rispetto alla cifratura, dobbiamo effettuare delle variazioni al dispositivo di schedulazione definito in precedenza.

- 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 *Li* e *Ri* 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)



Stesso algoritmo

Sottochiavi in ordine inverso

La Sicurezza del DES

Sin dalla sua presentazione la **sicurezza del DES** è stata messa in discussione; con una chiave di 56 bit le chiavi possibili sono 2^{56} numero enorme, assolutamente fuori della portata per un essere umano, ma non di quella dei moderni computer. $2^{56} = 72.057.594.037.927.936$

In altre parole il DES non è affatto al sicuro dal più semplice degli attacchi crittanalitici, quello **esaustivo** (in inglese: ***brute-force***) che semplicemente **prova** una per una **tutte le chiavi**.

Nel 1993 Wiener presentò un progetto di computer dedicato in grado di decrittare il DES, unico difetto il costo stimato in un milione di dollari!

Nel **1998** un gruppo di tre aziende Cryptographic Research, Advanced wireless technologies, Electronic Frontier Foundation comunicò di aver realizzato ***DES Cracker*** una macchina per la ricerca delle chiavi dal costo di 250.000$ in grado di **forzare** la **chiave** del **DES** in **56 ore**. Comunicazione che mostrò definitivamente che la **chiave** di **56 bit era troppo corta**.

Il DES a 64 bit ha dovuto così cedere il passo ad un DES a **128 bit** e al **triplo DES**. Nel 2001 è stato presentato il nuovo protocollo **AES** destinato a sostituire progressivamente il DES.

Ricerca esaustiva

- Numero chiavi DES = $2^{56} \approx 7{,}2056 \cdot 10^{16}$
- Un computer a 5.000 Mhz che testa una chiave per 10 cicli di clock impiega

  144.115.188 secondi ≈ 834 giorni ≈ 2 anni e 3 mesi

  per provare $2^{55} \approx 3{,}6 \cdot 10^{16}$ chiavi

5.000.000.000/10 = 500.000.000 di chiavi al secondo
72.056.000.000.000.000/500.000.000= 144.115.188 secondi

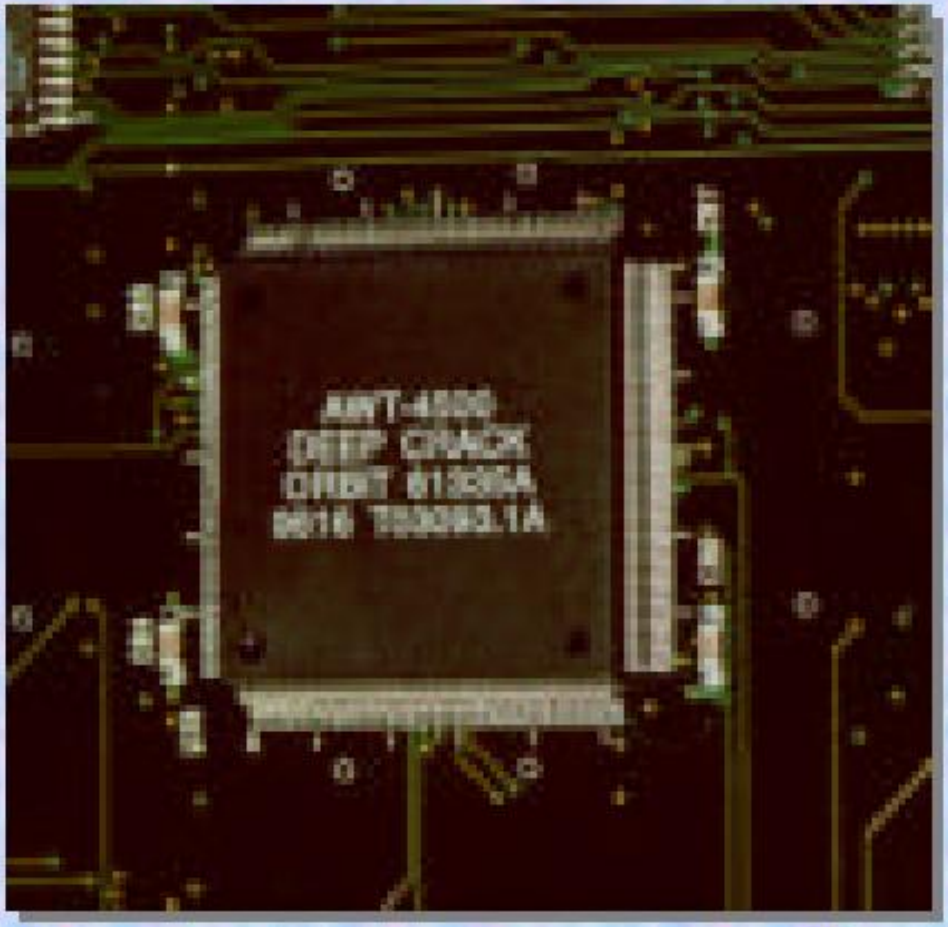
Deep Crack: Unità di ricerca

- Clock di 40Mhz
- Una decifratura in 16 cicli di clock
- Numero chiavi provate al secondo

$$\frac{40.000.000}{16} = 2.500.000$$

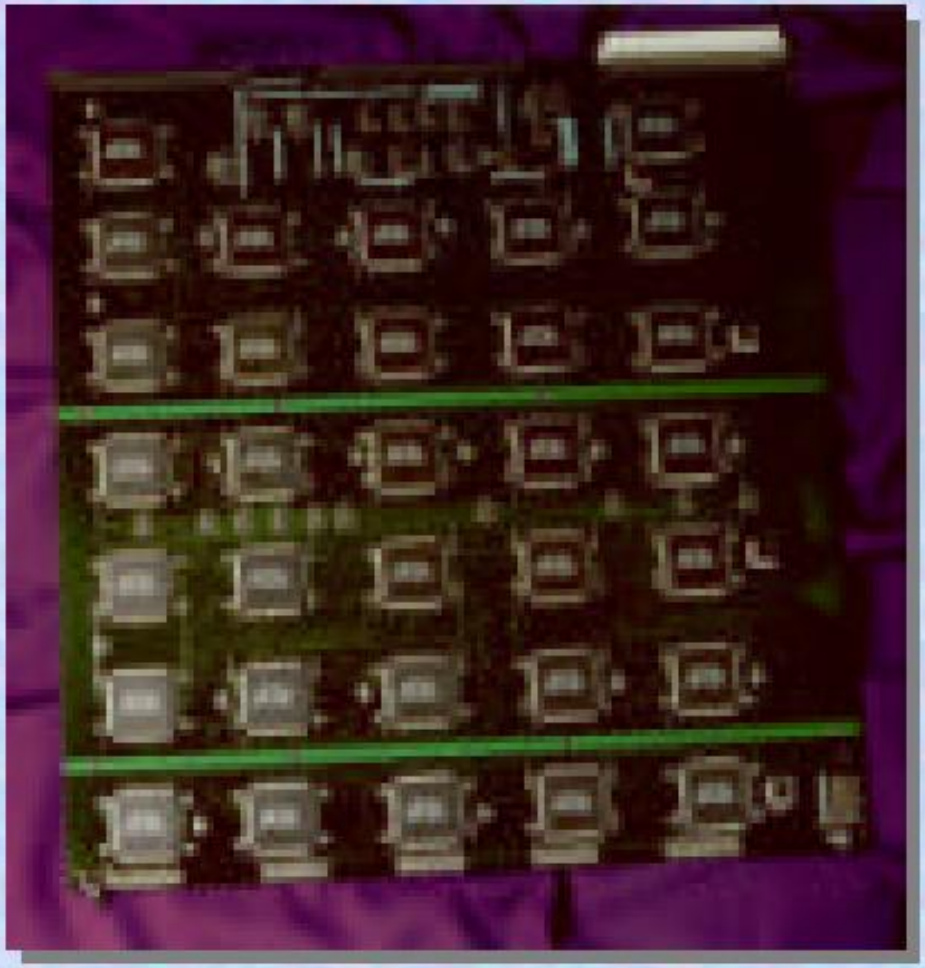
Chip

- 24 unità di ricerca
- Prova 24 · 2.500.000 = 60.000.000 chiavi al sec.
- Prova tutte le chiavi in 13.900 giorni (≈38 anni)



Board

- 64 chip
- 32 per faccia
- 40 cm X 40 cm
- Prova $64 \cdot 60.000.000 = 3.840.000.000$ chiavi al sec.
- Prova tutte le chiavi in ≈218 giorni



Chassis

- 12 schede
- Prova 12 · 3.840.000.000 = 46.080.000.000 chiavi al sec.
- Prova tutte le chiavi in ≈18 giorni

EFF DES Cracker

Prestazioni

Device	Quanti nella prossima device	Chiavi/sec	Num. medio Giorni per ricerca
Unità di ricerca	24	2.500.000	166.800
Chip	64	60.000.000	6.950
Board	12	3.840.000.000	109
Chassis	2	46.080.000.000	9,05
EFF DES Cracker		92.160.000.000	4,524