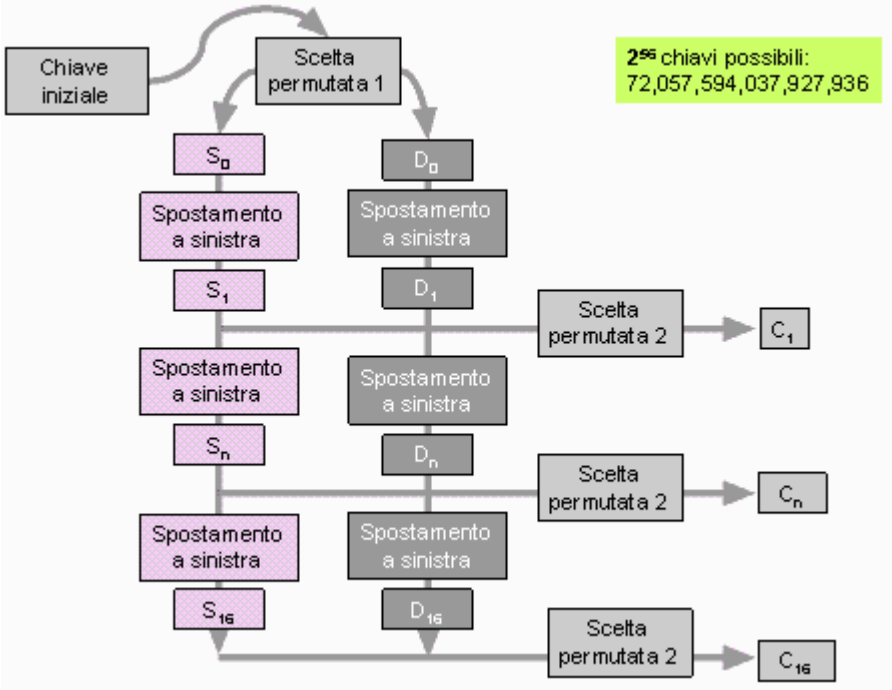
Operazioni sulla chiave

Partendo da una chiave iniziale di 64 bits vengono effettuate diverse operazioni per generare un insieme di chiavi da utilizzare nei singoli cicli. Notate che la scelta permutata 1 ci permetterà di ottenere 56 bits dai 64 iniziali da utilizzare come materiale di partenza per la generazione delle singole chiavi. E' per questo motivo che le chiavi possibili sono 72,057,594,037,927,936.

Lo schema generico delle operazioni effettuate sulla chiave è il seguente:



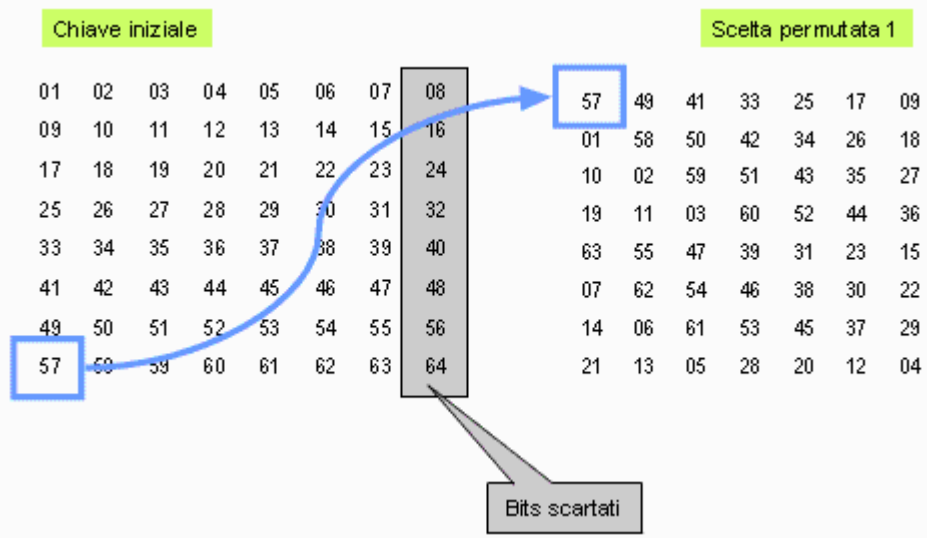
Vediamo adesso nello specifico le singole operazioni.

Chiave iniziale

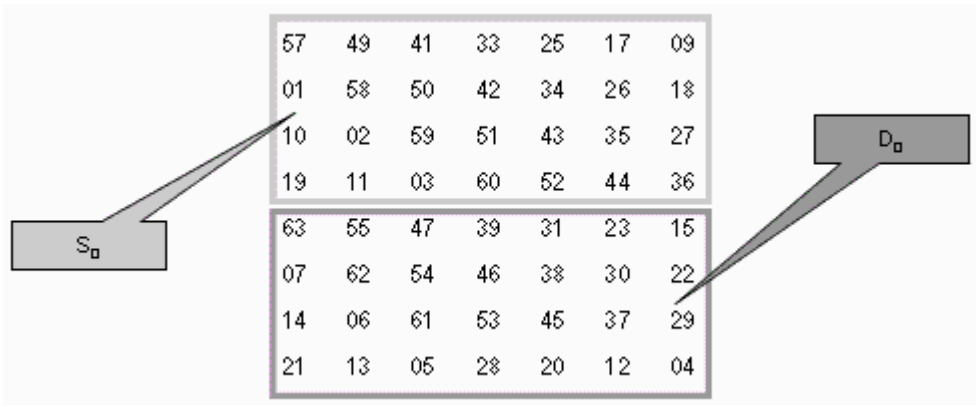
La chiave iniziale è costituita da 64 bits generati solitamente in modo casuale ma aventi una particolarità: l'ottavo bit di ogni riga non è altro che un controllo di disparità effettuato sui bits costituenti la riga e non verrà utilizzato nelle successive operazioni di elaborazione sulla chiave.

Scelta permutata 1

I 64 bits della chiave iniziale vengono sottoposti alla scelta permutata 1 e come potete notare, fra le altre cose, è proprio attraverso questa operazione che vengono eliminati gli otto bits di controllo.

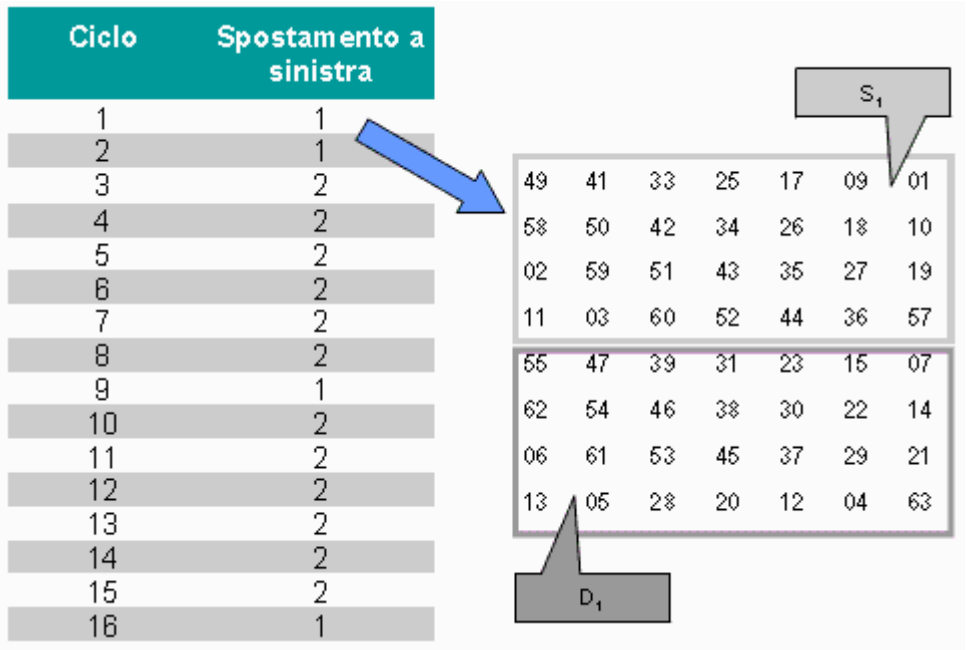


I 56 bits ottenuti vengono suddivisi in due parti da 28 bits per poi essere spostati singolarmente a sinistra.



Spostamento

Il valore dello spostamento è variabile ed è legato al ciclo secondo questa tabella:

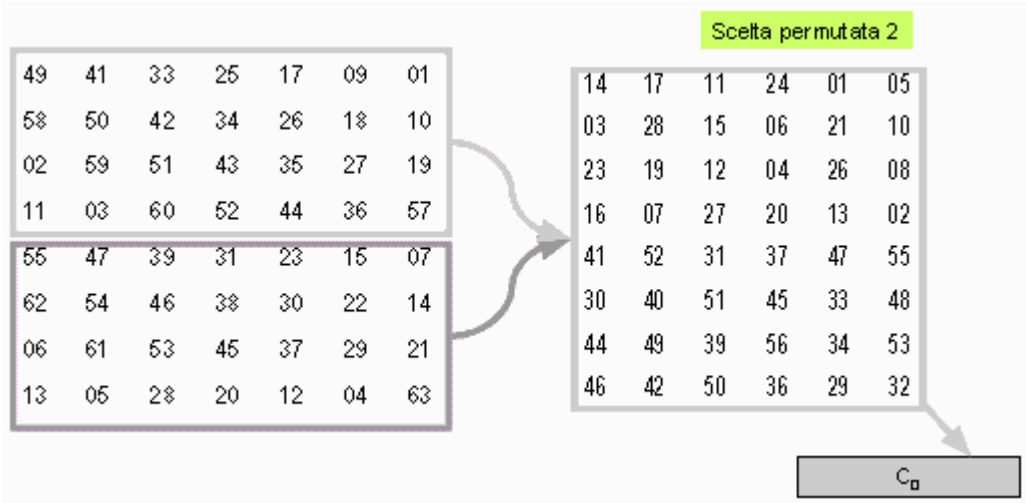


Ciclo	Spostamento a sinistra
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Per esempio, come potete vedere, per il primo ciclo, lo spostamento ha valore 1.

Scelta permutata 2

I due semiblocchi vengono quindi rielaborati utilizzando la scelta permutata 2.



Ed il valore ottenuto costituisce la prima chiave utilizzata nel primo ciclo. La procedura continuerà poi negli altri cicli, seguendo lo schema generale esposto in precedenza.

Operazioni sui dati

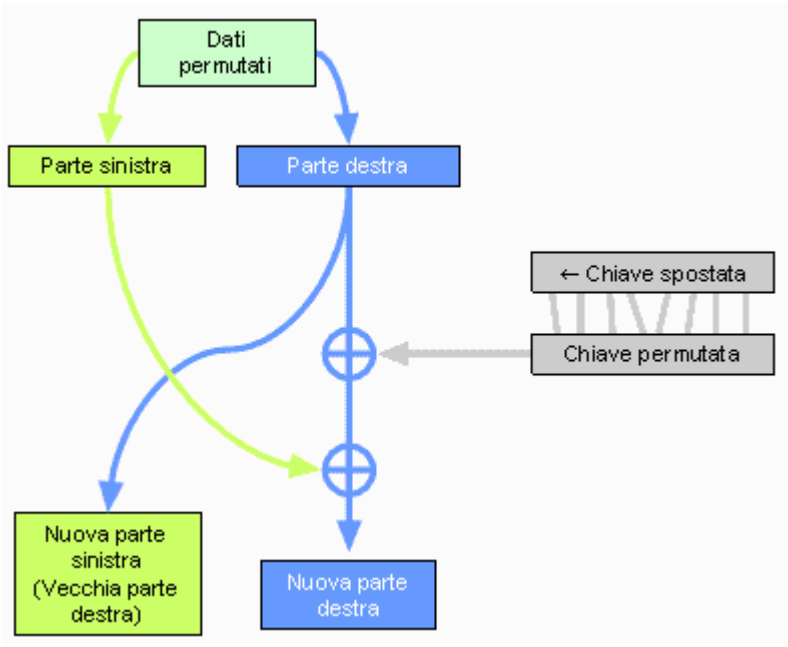
L'algoritmo elabora blocchi di dati di 64 bits utilizzando un insieme ben preciso di operazioni:

- primo ciclo;
- ciclo generico;
- ultimo ciclo.

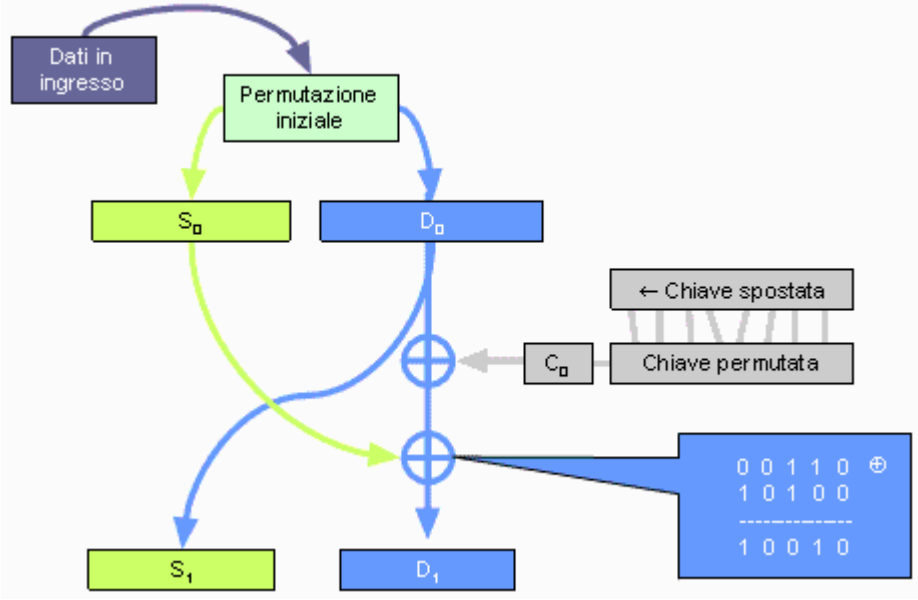
Iniziamo vedendo il primo ciclo.

Primo ciclo

Nel primo ciclo, i dati in ingresso vengono sottoposti ad una permutazione iniziale, suddivisi in due semiblocchi, scambiati di posto ed elaborati singolarmente. In particolare, sul semiblocco destro vengono applicati più procedimenti di somma binaria (OR esclusivo) utilizzando fra l'altro anche la chiave risultante dal corrispettivo ciclo di elaborazione sulla chiave.



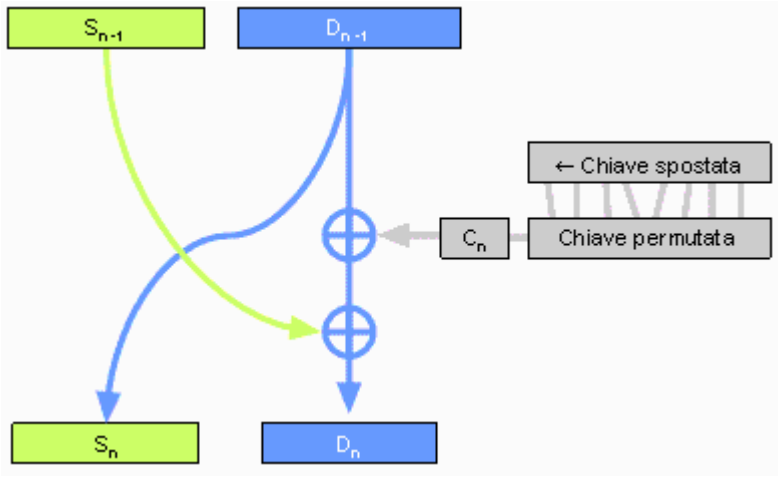
In particolare:



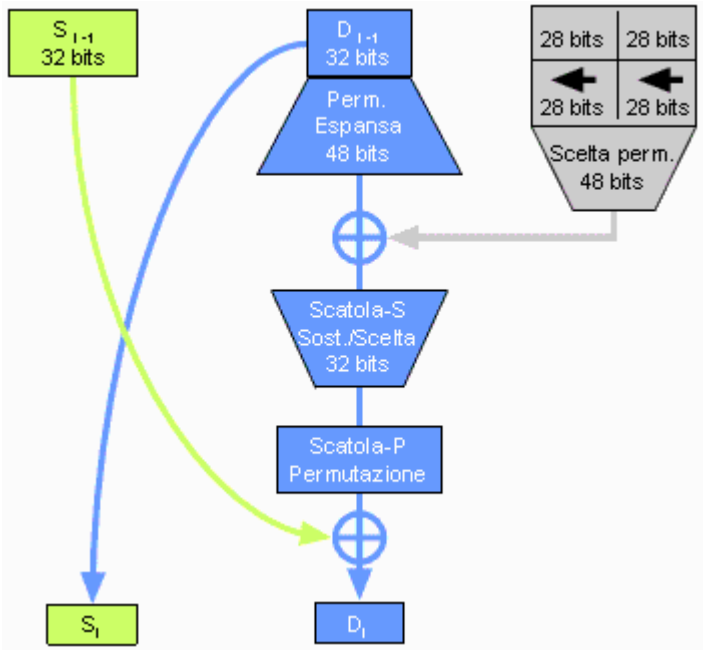
Al termine del primo ciclo i due semiblocchi ottenuti vengono sottoposti a quattordici cicli generici.

Ciclo *n*

Ogni ciclo generico è simile al primo ciclo tranne che per la mancanza della permutazione iniziale.



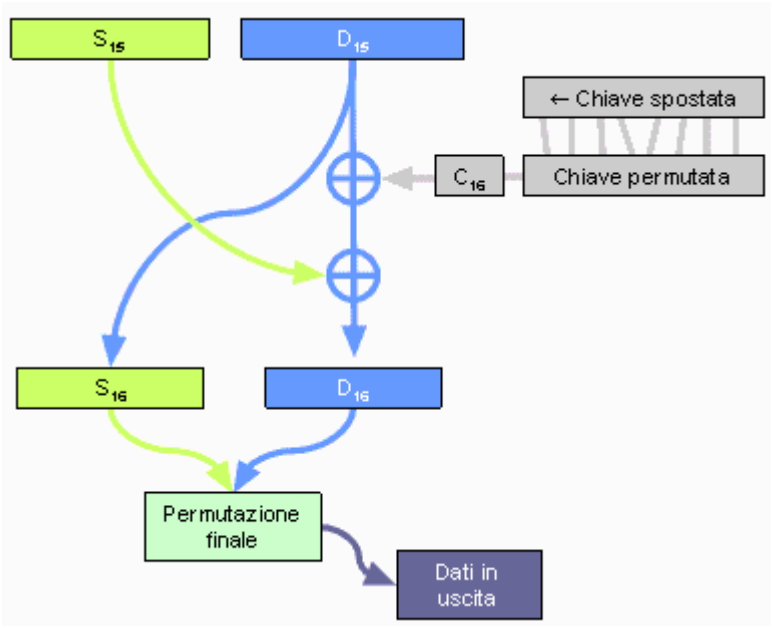
Vediamo nel dettaglio il ciclo generico:



Al termine del quattordicesimo ciclo generico, i due semiblocchi risultanti entrano nell'ultimo ciclo.

Ultimo ciclo

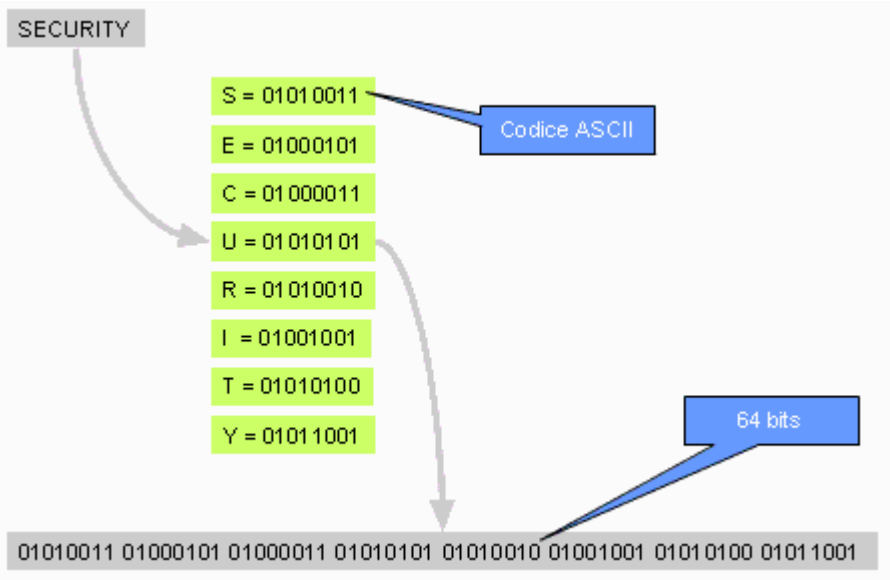
Anche in questo caso, le operazioni effettuate sono simili a quelle viste in precedenza, solo che al termine i dati ottenuti vengono sottoposti ad una permutazione finale.



Esempio

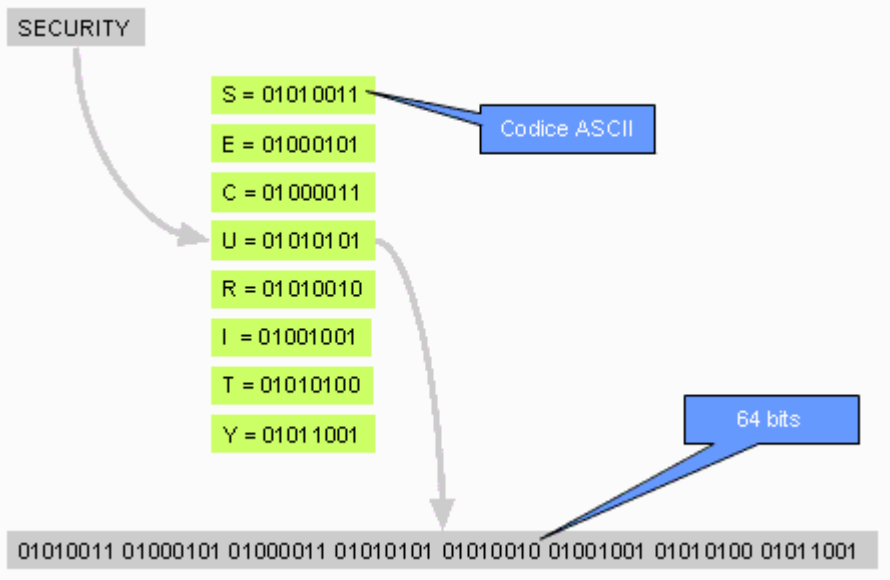
Dopo aver visto in teoria la struttura dell'algoritmo, vediamo insieme un esempio pratico utilizzando una sola parola. Per comodità e per rimanere nel tema, ho scelto la parola SECURITY.

Mi preme sottolineare che, per non appesantire eccessivamente l'esempio, non ho scelto una chiave e di conseguenza non vi mostrerò le elaborazioni effettuate sulla stessa.



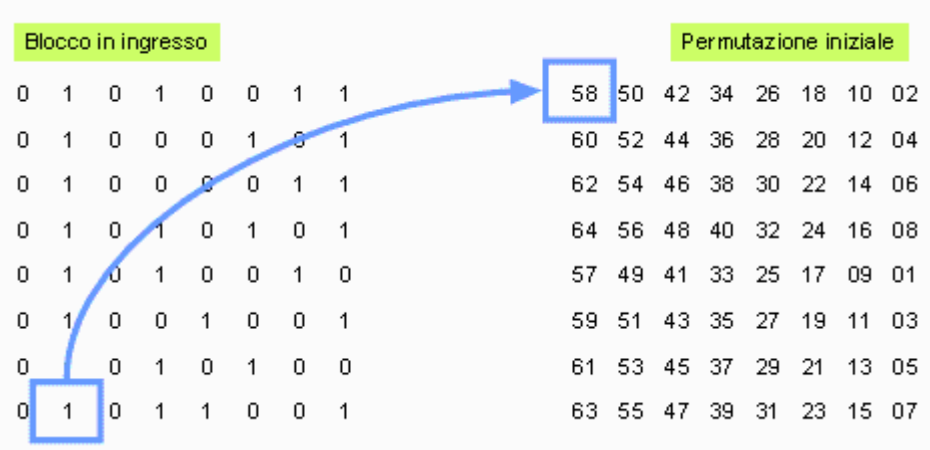
Blocco in ingresso

Suddividiamo la parola negli otto caratteri componenti e convertiamo i singoli caratteri in formato ASCII. Quindi concateniamo il tutto per ottenere il nostro primo blocco di 64 bits in ingresso.



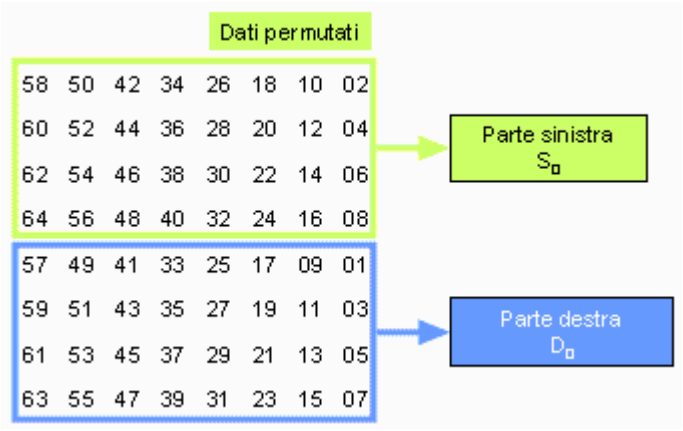
Permutazione iniziale

Sottoponiamo adesso i dati ottenuti alla permutazione iniziale. Notate che per facilità di lettura non ho riportato i dati binari del blocco di ingresso ma la posizione relativa all'interno del blocco. Così, per esempio, il cinquantottesimo bit del blocco in ingresso assumerà la prima posizione all'interno della matrice della permutazione iniziale.

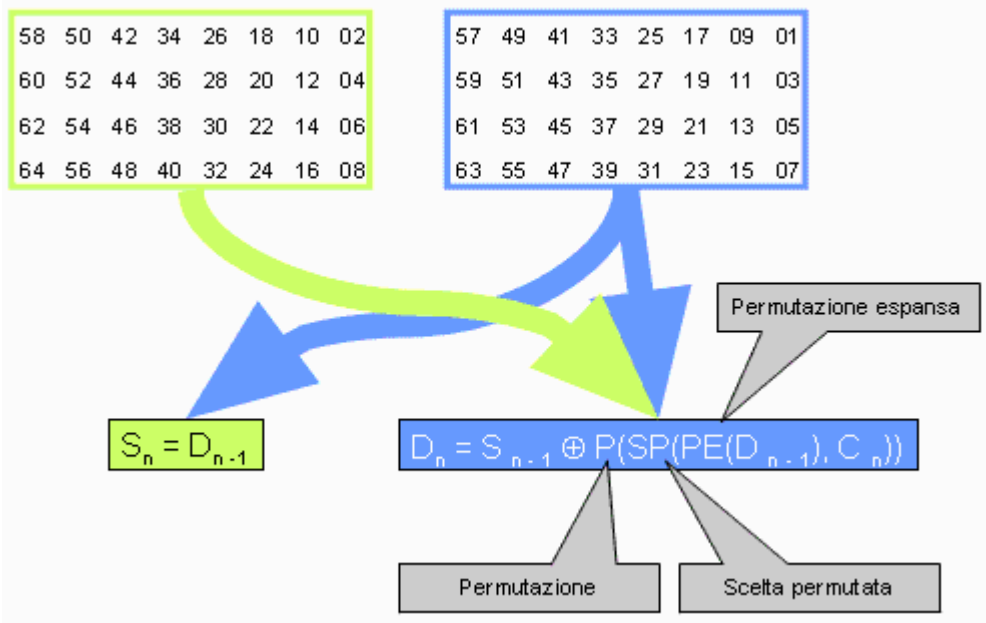


Suddivisione dei dati permutati

Successivamente alla permutazione iniziale, vi è una suddivisione dei dati in due semiblocchi, sinistro e destro.

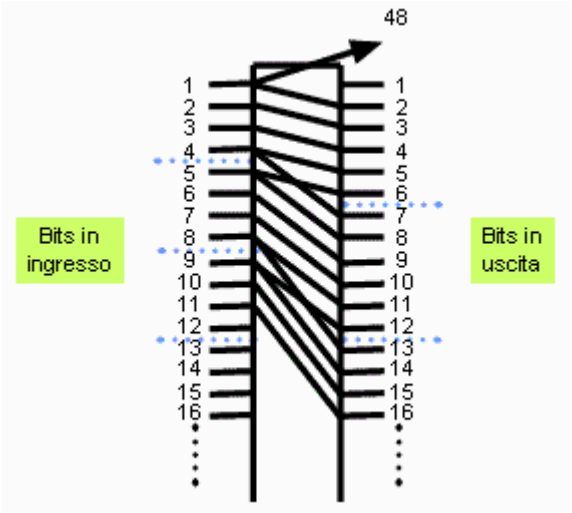


I due semiblocchi ottenuti verranno elaborati seguendo due percorsi separati, come riassunto in questo schema:

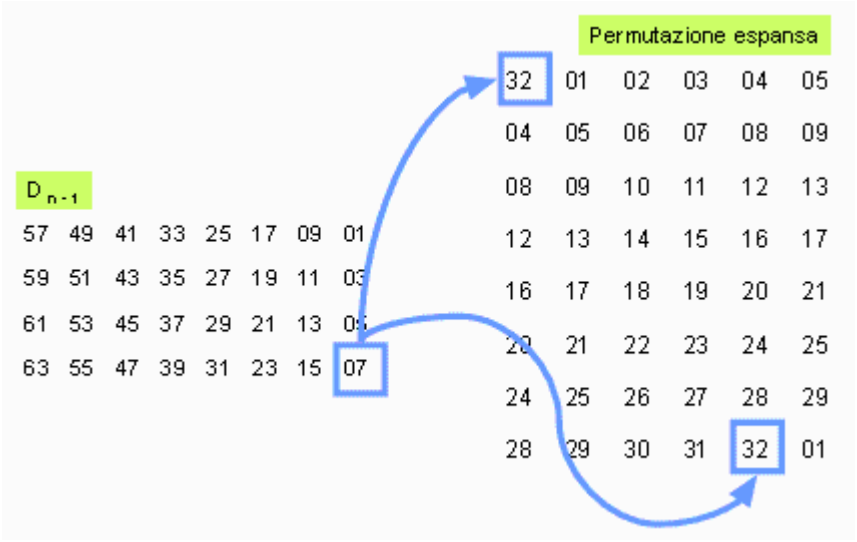


Permutazione espansa

La parte destra dei dati viene sottoposta ad una permutazione espansa in cui la maggior parte dei bits viene traslata di posto ed allo stesso tempo alcuni di questi vengono duplicati in più posizioni.



in pratica riprendendo il risultato del passo precedente:

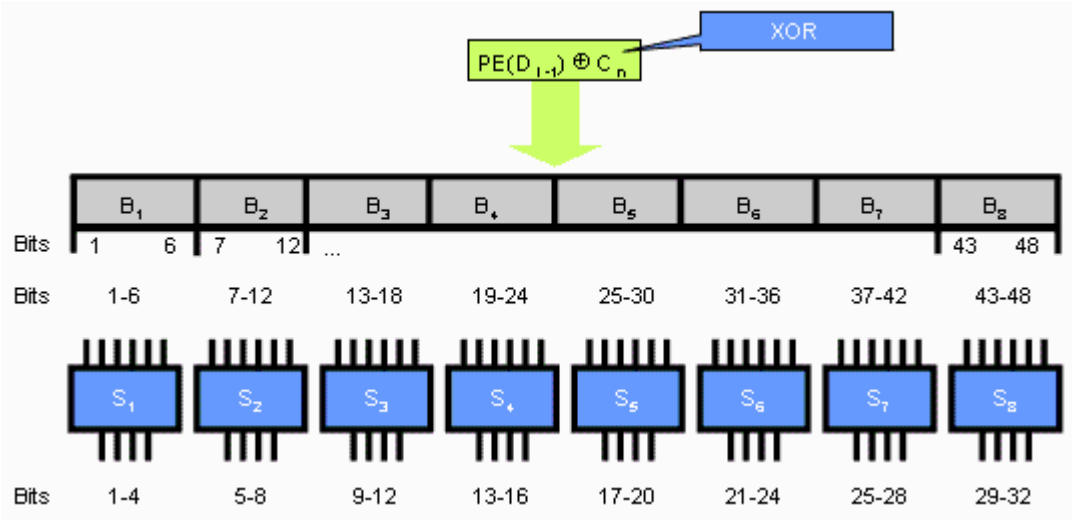


Dopo la permutazione espansa, i dati ottenuti vengono combinati mediante una funzione di somma binaria (OR esclusivo) con quelli relativi della chiave appositamente generata per il ciclo corrente.

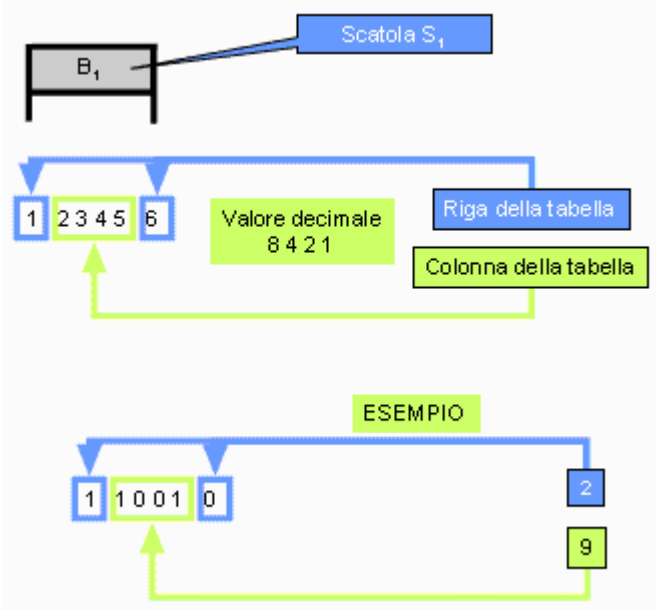
Chiaramente, questa elaborazione non comporta uno spostamento all'interno delle posizioni occupate dai dati, e come avrete potuto notare, a differenza dei passi precedenti, ho disposto il risultato finale su sei colonne per otto righe. Il perché di tutto questo lo capirete passando al prossimo passaggio: le Scatole-S.

Scelta permutata -scatole S

Le sostituzioni vengono effettuate in otto 'scatole' chiamate appunto Scatole-S (S-boxes) in cui da sei bits in ingresso vengono ottenuti quattro bits in uscita.



Utilizzando un semplice procedimento, ogni scatola elabora la singola riga in modo da ottenere, dai valori binari, due valori decimali. La logica seguita è questa: i due bits esterni trasformati in decimale danno la riga all'interno della scatola, mentre i quattro rimanenti, interni, danno la colonna.



Vediamo adesso come sono composte le otto scatole:

S_1	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
01	00	15	07	04	14	02	13	01	10	06	12	11	09	05	03	08
02	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
03	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

S_2	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	15	01	08	14	06	11	03	04	09	07	02	13	12	00	05	10
01	03	13	04	07	15	02	08	14	12	00	01	10	06	09	11	05
02	00	14	07	11	10	04	13	01	05	08	12	06	09	03	02	15
03	13	08	10	01	03	15	04	02	11	06	07	12	00	05	14	09

S_3	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	10	00	09	14	06	03	15	05	01	13	12	07	11	04	02	8
01	13	07	00	09	03	04	06	10	02	08	05	14	12	11	15	1
02	13	06	04	09	08	15	03	00	11	01	02	12	05	10	14	07
03	01	10	13	00	06	09	08	07	04	15	14	03	11	05	2	12

S_4	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	07	13	14	03	00	06	09	10	01	02	08	05	11	12	04	15
01	13	08	11	05	06	15	00	03	04	07	02	12	01	10	14	09
02	10	06	09	00	12	11	07	13	15	01	03	14	05	02	08	04
03	03	15	00	06	10	01	13	08	09	04	05	11	12	07	02	14

S_5	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	02	12	04	01	07	10	11	06	08	05	03	15	13	00	14	09
01	14	11	02	12	04	07	13	01	05	00	15	10	03	09	08	06
02	04	02	01	11	10	13	07	08	15	09	12	05	06	03	00	14
03	11	08	12	07	01	14	02	13	06	15	00	09	10	04	05	03

S_6	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	12	01	10	15	09	02	06	08	00	13	03	04	14	07	05	11
01	10	15	04	02	07	12	09	05	06	01	13	14	00	11	03	08
02	09	14	15	05	02	08	12	03	07	00	04	10	01	13	11	06
03	04	03	02	12	09	05	15	10	11	14	01	07	06	00	08	13

S_7	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	04	11	02	14	15	00	08	13	03	12	09	07	05	10	06	01
01	13	00	11	07	04	09	01	10	14	03	05	12	02	15	08	06
02	01	04	11	13	12	03	07	14	10	15	06	08	00	05	09	02
03	06	11	13	08	01	04	10	07	09	05	00	15	14	02	03	12

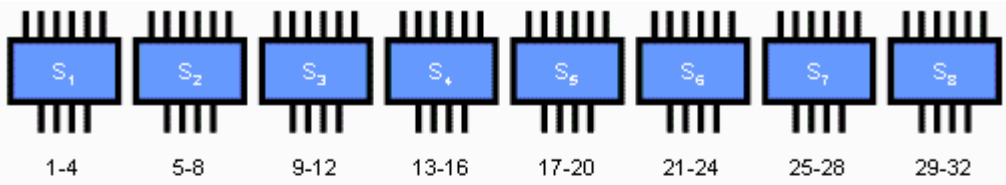
S_8	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
00	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
01	01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02
02	07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
03	02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11

Il valore decimale in uscita dalla singola scatola sarà quello indicato dalle coordinate costituite da riga e colonna.

L'algoritmo prosegue, convertendo il valore decimale appena ottenuto in binario.

Permutazione

I dati, estratti delle Scatole-S vengono introdotti all'interno di una Scatola-P (P-Box). Dove vengono permutati.



Permutazione

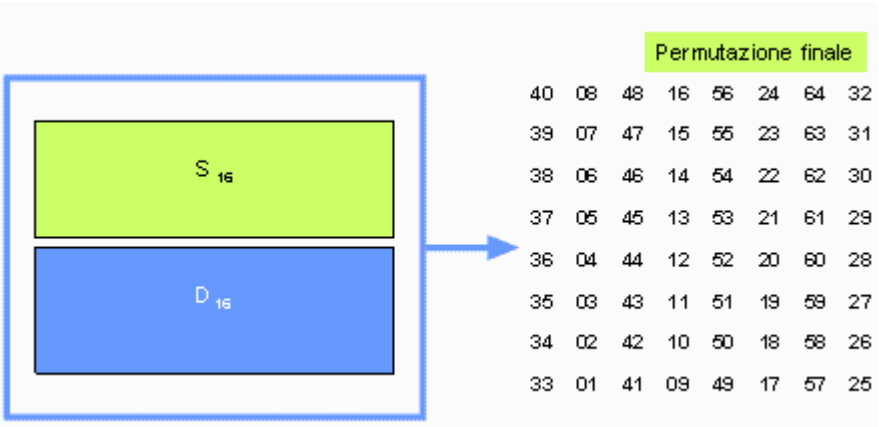
16	07	20	21
29	12	28	17
01	15	23	26
05	18	31	10
02	8	24	14
32	27	03	09
19	13	30	06
22	11	04	25

Il passo successivo prevede la somma binaria (OR esclusivo) dei dati della parte destra con quelli della parte sinistra calcolata in precedenza.

Tutta questa procedura appena vista, viene ripetuta per quattordici cicli, fino ad arrivare in quello finale che si caratterizza per la presenza di un'ultima permutazione, chiamata appunto permutazione finale.

Permutazione finale

Nella fase finale, i dati vengono sottoposti ad un'ultima permutazione. Vediamola nel dettaglio:



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

Modalità di funzionamento

ECB (Electronic Code Book)

Costituisce la modalità di funzionamento predefinita. I dati vengono suddivisi in blocchi di 64 bits ed ogni blocco viene cifrato singolarmente indipendentemente dal precedente. In questo modo, nell'eventualità si verificasse un errore, questo sarebbe limitato al blocco contenente l'errore stesso. Il principale rischio è dato dal fatto che, vista la totale assenza di ulteriori misure di sicurezza se non quelle proprie dell'algoritmo, sarebbe possibile scambiare l'ordine dei blocchi cifrati risultanti senza che questo possa essere rilevato. Comunque, anche se effettivamente è la meno sicura, per le sue caratteristiche di velocità e di semplicità di implementazione costituisce la modalità più utilizzata.

CBC (Cipher Block Chaining)

In questa modalità, ogni blocco di testo, dopo essere stato cifrato in modalità ECB, viene sottoposto ad una ulteriore operazione di OR esclusivo (XOR) con il successivo blocco ancora da cifrare. In questo modo, tutti i blocchi verranno resi dipendenti dai blocchi che li precedono così che, per trovarne il testo in chiaro, si renda necessario non solo conoscerne il relativo testo cifrato ma anche la chiave, ed il testo cifrato del blocco precedente. Da notare che, il primo blocco, non avendo un blocco precedente, verrà sottoposto all'operazione di XOR con un numero di 64 bits chiamato Vettore di Inizializzazione (IV - Initialization Vector).

Anche se più sicura ma al tempo stesso più lenta della modalità ECB, la logica seguita fa si che, se durante la trasmissione si introduce un errore, questo verrà trascinato lungo tutta la sequenza dei blocchi rimanenti, visto che ogni blocco è dipendente dal precedente.

CFB (Cipher Feedback)

Questa modalità permette di cifrare blocchi in chiaro inferiori a 64 bits così da eliminare la necessità dell'ulteriore elaborazione che normalmente si rende necessaria con i files aventi grandezza non multipla di 8 bytes.

Il testo in chiaro non viene sottoposto all'algoritmo DES ma, ad un'operazione di OR esclusivo (XOR) con i dati uscenti dall'algoritmo, generati utilizzando inizialmente un blocco casuale di 64 bits, chiamato Shift Register, ed in uscita, un componente aggiuntivo, M-Box, che selezionerà gli M bits più a sinistra del blocco risultante così da farlo corrispondere alla grandezza del blocco che vogliamo cifrare.

Il testo cifrato ottenuto al termine dell'operazione, diverrà il nuovo dato in ingresso all'interno dello Shift Register, da utilizzare con il successivo blocco da cifrare.

Presenta tutte le caratteristiche sia positive che negative della modalità CBC.

OFB (Output Feedback)

E' simile alla modalità CFB da cui si differenzia per il fatto che è il blocco in uscita dal ciclo DES che viene riutilizzato all'interno dello Shift Register, piuttosto che il risultato finale di tutta la procedura.

A differenza delle modalità CFB e CBC, un eventuale errore di trasmissione non compromette tutta la procedura, visto che, una volta in possesso del valore iniziale casuale dello Shift Register, tutti i successivi concatenamenti vengono derivati da quest'ultimo.

Allo stesso tempo però, questa modalità è meno sicura della CFB perchè, anche non conoscendo la chiave, basta il testo cifrato ed i dati in uscita dal ciclo DES per risalire al testo in chiaro dell'ultimo blocco.