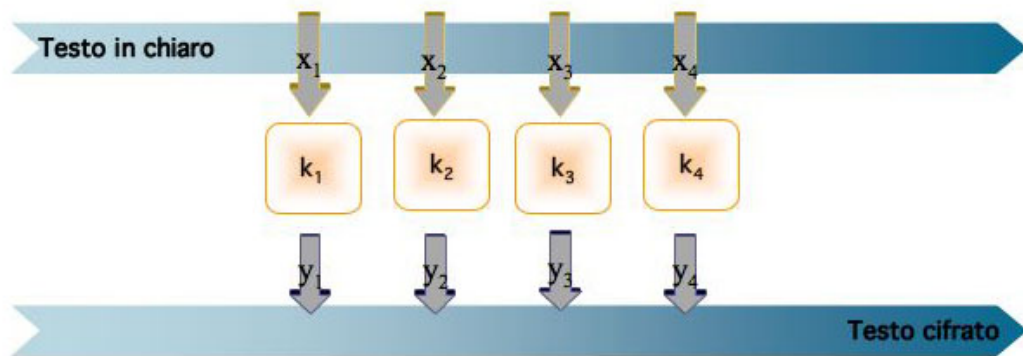


crittosistemi a flusso

Prendiamo $\mathbb{Z}_2$ come alfabeto;

in un *cifrario a flusso*, il testo in chiaro $x = x_1x_2 \dots$ viene cifrato bit per bit con la chiave $k = k_1, k_2, \dots$ per ottenere il testo cifrato.



Nell'one-time pad, $x_i \oplus k_i = y_i$

crittosistemi a blocchi

- I cifrari a blocchi (o a pacchetti) lavorano su blocchi di informazione ognuno dei quali è cifrato e decifrato indipendentemente
- Il testo in chiaro è considerato come una sequenza di blocchi di m bit.
- Il testo cifrato ha la stessa lunghezza del testo in chiaro.
- Di fatto opera come una sostituzione su stringhe molto lunghe (64 bit o più)
- Quindi come una permutazione di 2^{64} numeri.
- dovrebbe essere una **permutazione pseudocasuale (PRP)** indistinguibile da una permutazione casuale
- Il DES opera su blocchi di 64 bit; l'AES su blocchi di 128 bit

crittosistemi prodotto

- Shannon in *Communication Theory of Secrecy Systems* introduce l'idea di combinare diversi crittosistemi prendendo il loro “prodotto”.
- DES, AES sono CS prodotto
- consideriamo crittosistemi in cui $\mathcal{P} = \mathcal{C}$ (CS endomorfi):
 $S_1 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_1, \mathcal{E}_1, \mathcal{D}_1)$, $S_2 = (\mathcal{P}, \mathcal{P}, \mathcal{K}_2, \mathcal{E}_2, \mathcal{D}_2)$.
- Il prodotto $S_1 \times S_2$ è il crittosistema

$$(\mathcal{P}, \mathcal{P}, \mathcal{K}_1 \times \mathcal{K}_2, \mathcal{E}, \mathcal{D});$$

- data una chiave $k = (k_1, k_2)$, la corrispondente funzione di cifratura è

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

e la funzione di decifratura è

$$d_{(k_1, k_2)}(y) = d_{k_1}(d_{k_2}(y)).$$

crittosistemi prodotto

Infatti,

$$\begin{aligned} d_{(k_1, k_2)}(e_{(k_1, k_2)}(x)) &= d_{(k_1, k_2)}(e_{k_2}(e_{k_1}(x))) \\ &= d_{k_1}(d_{k_2}(e_{k_2}(e_{k_1}(x)))) \\ &= d_{k_1}(e_{k_1}(x)) \\ &= x. \end{aligned}$$

crittosistemi prodotto: esempio

- sia M il cifrario moltiplicativo: $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$;
 $\mathcal{K} = \{a \in \mathbb{Z}_{26} \mid (a, 26) = 1\}$;

$$e_a(x) = ax, \quad d_a(y) = a^{-1}y;$$

- sia S il cifrario additivo; $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$;

$$e_k(x) = x + k, \quad d_k(y) = y - k;$$

- allora $M \times S$ è il cifrario affine!
- $e_{(a,k)}(x) = e_k(ax) = ax + k$

- osserviamo che anche $S \times M$ è il cifrario affine
- $e_{(k,a)}(x) = e_a(x + k) = ax + ka$
- i due crittosistemi **commutano**
- il prodotto di crittosistemi non è sempre commutativo
- ma è sempre **associativo**
- $(S_1 \times S_2) \times S_3 = S_1 \times (S_2 \times S_3)$

cifrari iterati

- spesso si considera il prodotto di un crittosistema S per se stesso; $S \times S = S^2$
- un crittosistema si dice **idempotente** se $S \times S = S$: (additivo, sostituzione, Vigenère sono idempotenti)
- se un crittosistema S non è idempotente, usare il CS S^n può aumentare la sicurezza
- questa idea è usata per esempio nel DES, che consiste di 16 iterazioni, o **round**

- per costruire crittosistemi non idempotenti da iterare
- si considera il prodotto di crittosistemi semplici, eventualmente idempotenti, ma che **non** commutino fra loro.
- perchè se S_1 e S_2 sono idempotenti e commutano, anche il loro prodotto $S_1 \times S_2$ è idempotente
 - $(S_1 \times S_2) \times (S_1 \times S_2) = S_1 \times (S_2 \times S_1) \times S_2$
 $= S_1 \times S_1 \times S_2 \times S_2 = S_1 \times S_2$

SPN

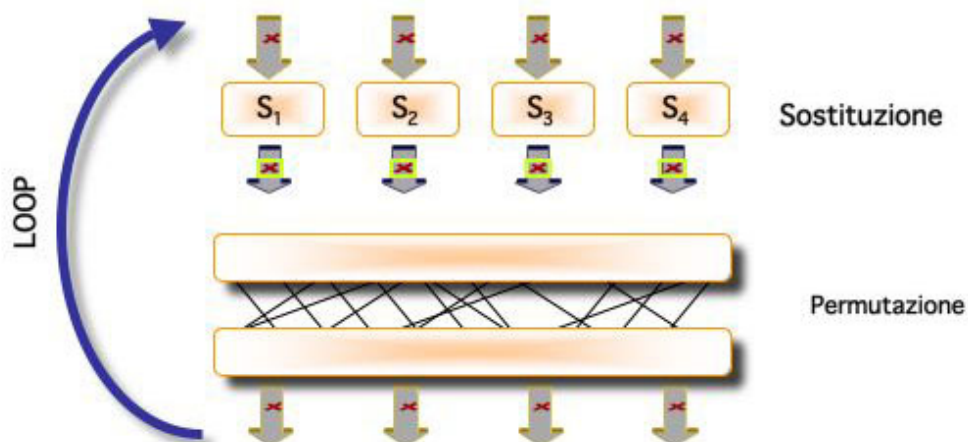
- questo è quello che succede nelle reti a sostituzione-permutazione (Substitution-permutation network, o SPN)
- introdotte da Shannon in *Communication Theory of Secrecy Systems*
- queste reti sono la base della crittografia moderna
- forniscono confusione e diffusione di un messaggio

Shannon ha suggerito di usare crittosistemi che producano:

confusione: rende la relazione tra ciphertext e chiave la più complessa possibile.

diffusione “sparge” la struttura statistica del testo in chiaro su tutto il testo cifrato.

SPN

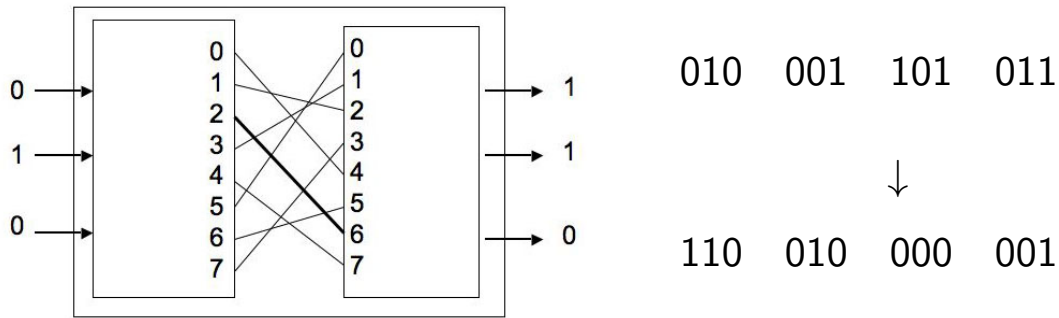


SPN - Esempio

Blocchi lunghi $m = 3 \times 4$. Un blocco è:

010 001 101 011

Le S-box sono permutazioni π_S di $\{0, 1, \dots, 7\}$: hanno come input e output una stringa binaria di lunghezza tre. Per semplicità, consideriamo una sola S-box.



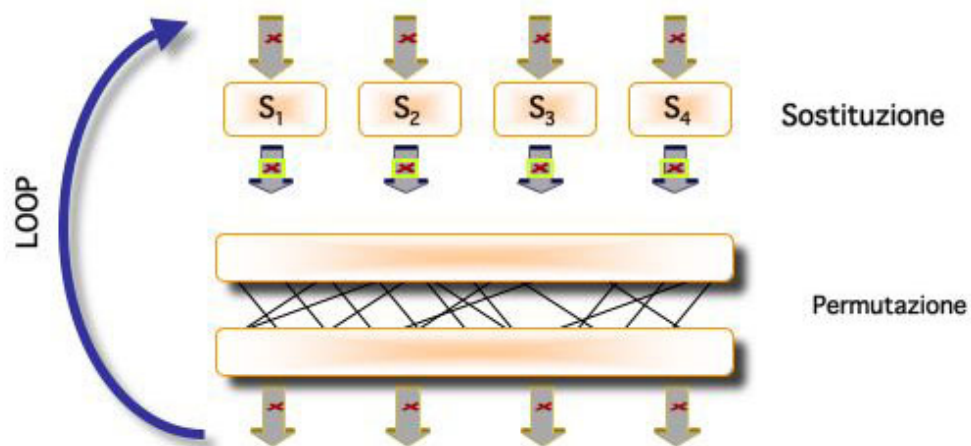
La P-box è una permutazione π_P di $\{1, 2, \dots, 12\}$.

1	2	3	4	5	6	7	8	9	10	11	12
5	7	2	12	9	6	4	3	11	8	1	10

Quindi

010	001	101	011
	↓	π_S	
110	010	000	001
	↓	π_P	
000	010	101	100

SPN



key schedule

Abbiamo descritto un'iterazione.

In genere, in un SPN a R round, una chiave dà luogo a R chiavi di round (key schedule).

Nell' i -simo round, prima della S-box e della P-box, il testo in input viene combinato con la chiave tramite uno XOR.

Supponiamo che le rete del nostro esempio abbia 2 round: la chiave potrebbe essere una stringa binaria k di lunghezza 12, che produce due chiavi di round (k_1, k_2) , per esempio $k_1 = k, k_2 = k \oplus 1$.

Vediamo come funziona l'SPN dell'esempio, se il testo in chiaro è $x = 001\ 110\ 101\ 001$, e la chiave è $k = 011\ 111\ 000\ 010$. Le chiavi di round sono $k_1 = k = 011\ 111\ 000\ 010$; $k_2 = k \oplus 1 = 100\ 000\ 111\ 101$

$$\begin{array}{rcll}
 x = & 001 & 110 & 101 & 001 & \oplus \\
 k_1 = & 011 & 111 & 000 & 010 & \\
 & 010 & 001 & 101 & 011 & \\
 & \downarrow & \pi_S & & & \\
 & 110 & 010 & 000 & 001 & \\
 & \downarrow & \pi_P & & & \\
 & 000 & 010 & 101 & 100 & \\
 & & & & & \oplus \\
 k_2 = & 100 & 000 & 111 & 101 & \\
 & 100 & 010 & 010 & 001 & \\
 & \downarrow & \pi_S & & & \\
 & 111 & 110 & 110 & 010 & \\
 & \downarrow & \pi_P & & & \\
 & 111 & 110 & 101 & 001 & = y
 \end{array}$$

SPN - descrizione generale

Siano l , k e R interi positivi, $m = l \cdot k$, sia $\pi_{S_i} : \{0, 1\}^l \rightarrow \{0, 1\}^l$ una permutazione, (per $i = 1, \dots, k$) e

$\pi_P : \{1, \dots, m\} \rightarrow \{1, \dots, m\}$ un'altra permutazione. Siano $\mathcal{P} = \mathcal{C} = \mathbb{Z}_2^m$, sia $\mathcal{K} = \mathbb{Z}_2^m$ assieme a un algoritmo di key scheduling che, data una chiave k , produce R chiavi di round $k_1, \dots, k_R$.

La cifratura funziona secondo il seguente schema informale:

per $i = 1, \dots, R$

- ① input = x $i = 1$, o l'output del round $i - 1$ per $i > 1$.
- ② fare lo XOR fra l'input e k_i .
- ③ a questo, applicare π_{S_i} .
- ④ a questo, applicare π_P .

l'output del R -simo round è il testo cifrato y .