Cifrari a blocchi

a.a. 2019/20

Alfredo De Santis

Dipartimento di Informatica
Università di Salerno

ads@unisa.it

http://www.di-srv.unisa.it/~ads

Marzo 2020

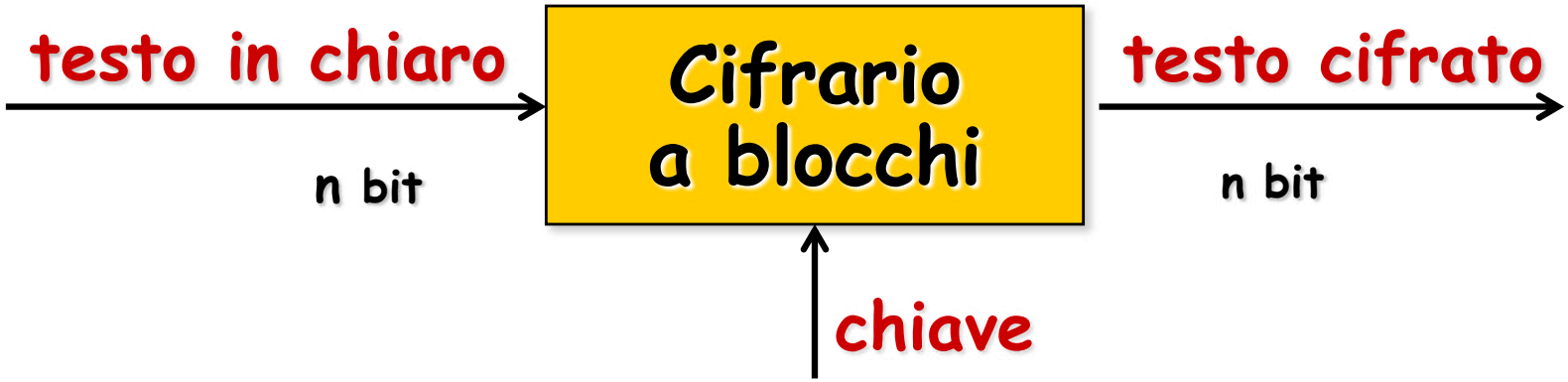
Indice

- Introduzione
- Cifrari di Feistel
- Data Encryption Standard (DES)
  - Storia
  - Descrizione
  - Sicurezza
- Modalità operative
- Lunghezza della chiave e sicurezza
- DES Doppio
- DES Triplo

Cifrari simmetrici

- Crittosistemi a chiave privata/segreta
- Alice e Bob conoscono la stessa chiave K
- Cifrari a blocchi
  - Messaggi divisi in blocchi e poi cifrati
- Stream cipher
  - Messaggi cifrati carattere per carattere

Cifrari a blocchi



- Alcuni esempi:
  - Data Encryption Standard (DES)
  - DES triplo
  - Blowfish
  - RC5, RC6
  - Advanced Encryption Standard (AES)

Cifrari a blocchi

- Blocchi di n bit
  - 2^n possibili input (testo in chiaro)
  - 2^n possibili output (testo cifrato)
  - $2^n!$ possibili trasformazioni (reversibili)

- La cifratura deve essere reversibile

  Funzione biettiva

Chiaro	Cifrato
00	11
01	10
10	00
11	01

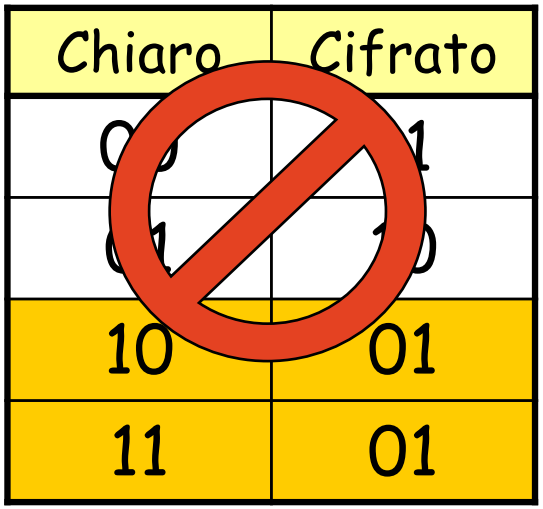
Chiaro	Cifrato
00	11
01	10
10	01
11	01

Cifrari a blocchi

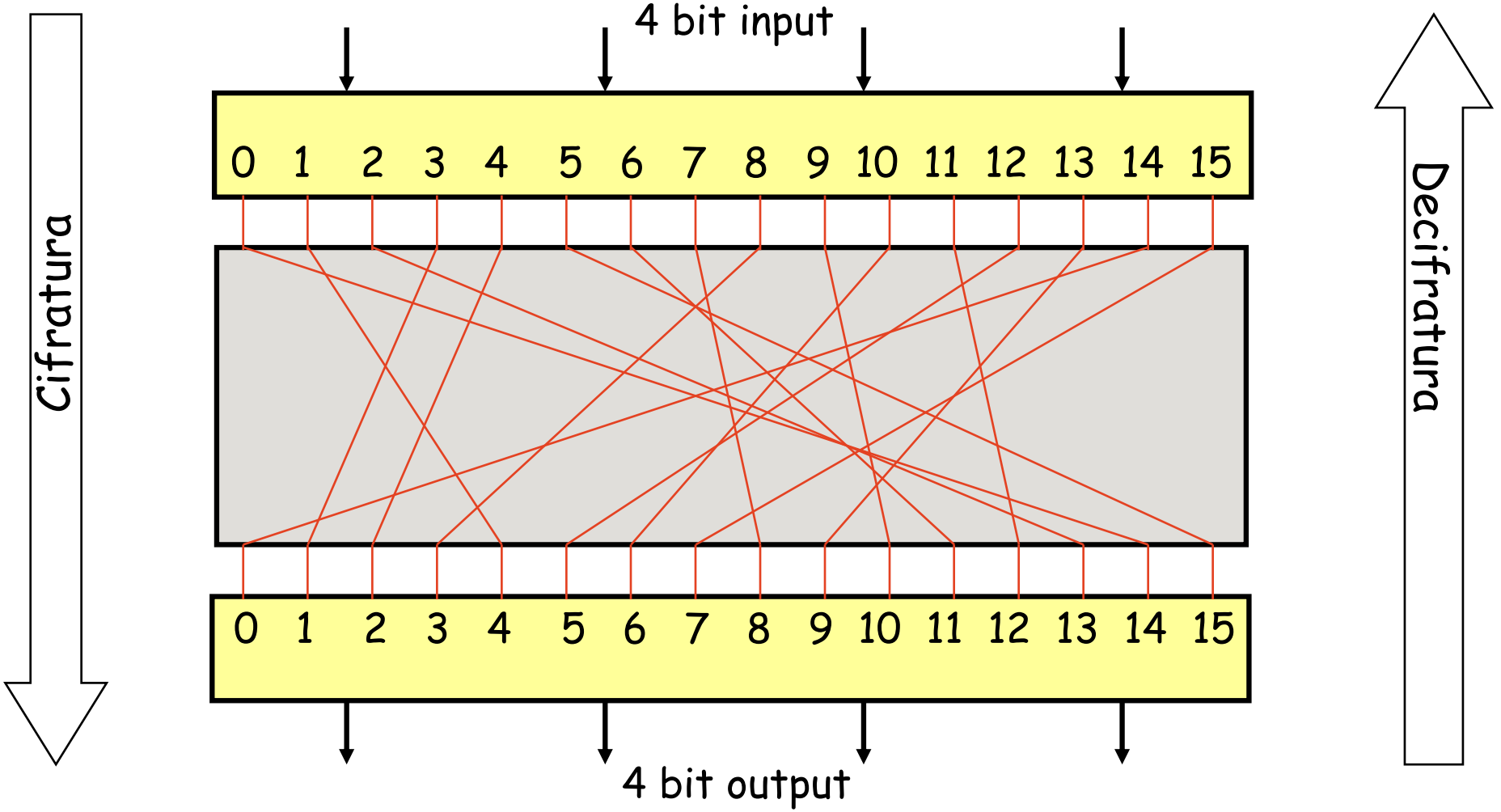
- Blocchi di n bit
  - 2^n possibili input (testo in chiaro)
  - 2^n possibili output (testo cifrato)
  - $2^n!$ possibili trasformazioni (reversibili)

- La cifratura deve essere reversibile
  - Funzione biettiva

Chiaro	Cifrato
00	11
01	10
10	00
11	01

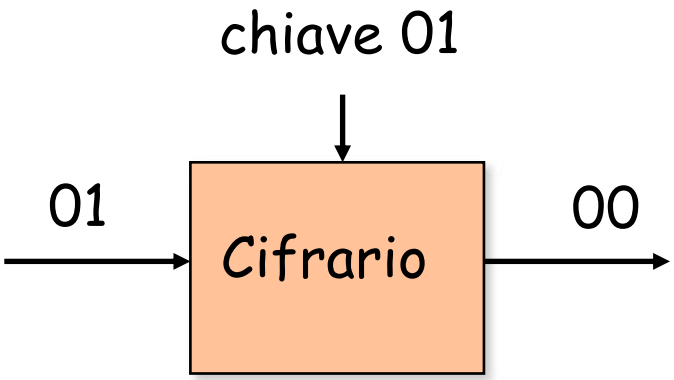


Esempio per n=4



Cifrari a blocchi

- Cifrario chiave 2 bit



Chiaro	Cifrato
00	11
01	10
10	00
11	01

chiave 00

Chiaro	Cifrato
00	01
01	00
10	10
11	11

chiave 01

Chiaro	Cifrato
00	10
01	11
10	01
11	00

chiave 10

Chiaro	Cifrato
00	00
01	01
10	11
11	10

chiave 11

Esempio per n=4

Mapping memorizzabile in una tabella

- La chiave è la tabella

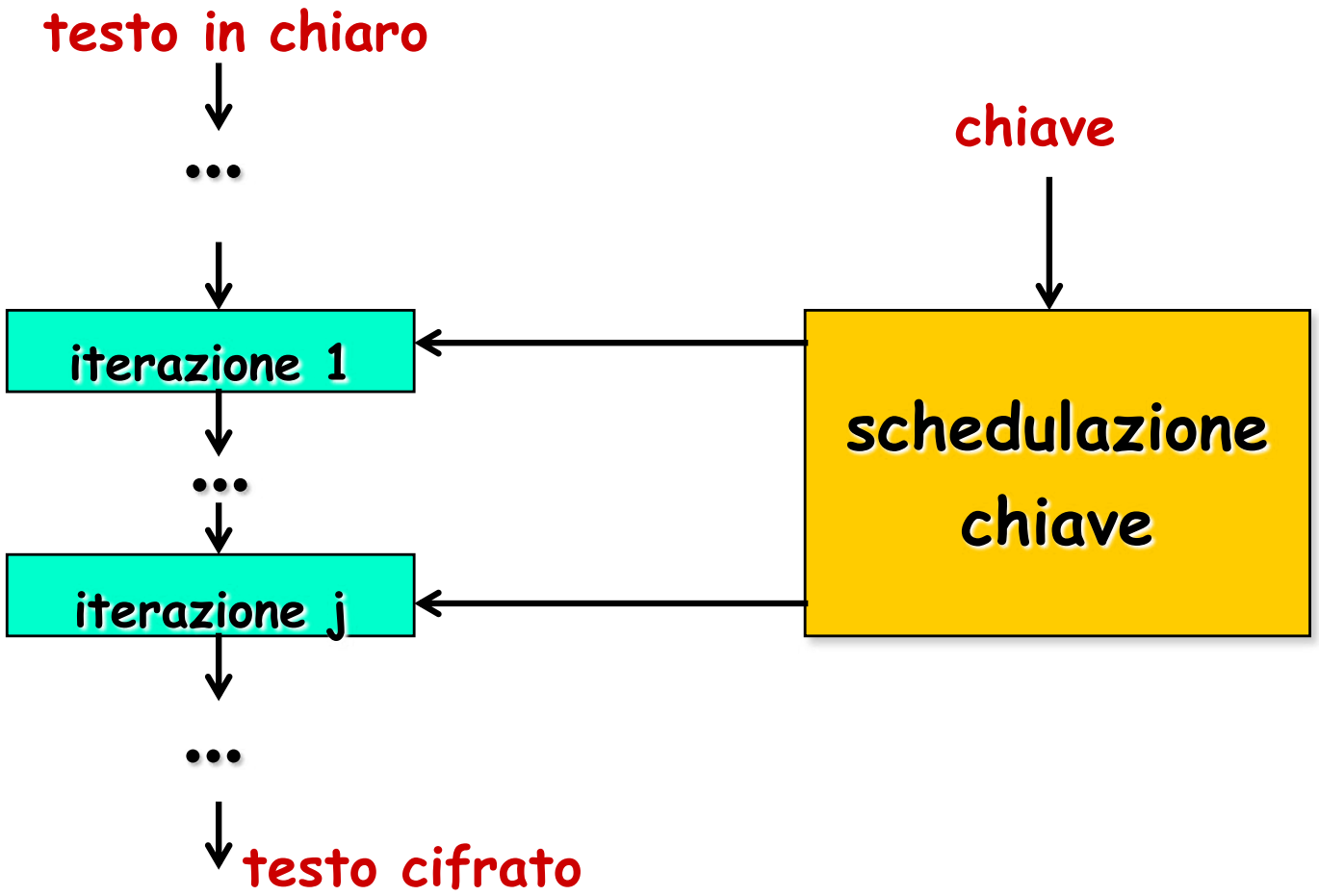
Chiaro	Cifrato
0000	1110
0001	0100
0010	1101
0011	0001
0100	1111
0101	1011
0110	1011
0111	1000

Chiaro	Cifrato
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Cifrari a blocchi

- L'uso della tabella è poco pratico
- Per blocchi di n bit la tabella ha dimensione $n \times 2^n$
  - Per n=64, dimensione $n \times 2^n = 2^{70} = 10^{21}$ bit
- Occorre un'approssimazione
  - Un metodo compatto per rappresentare la funzione di cifratura/decifratura

Struttura tipica di un cifrario a blocchi



Cifrari di Feistel

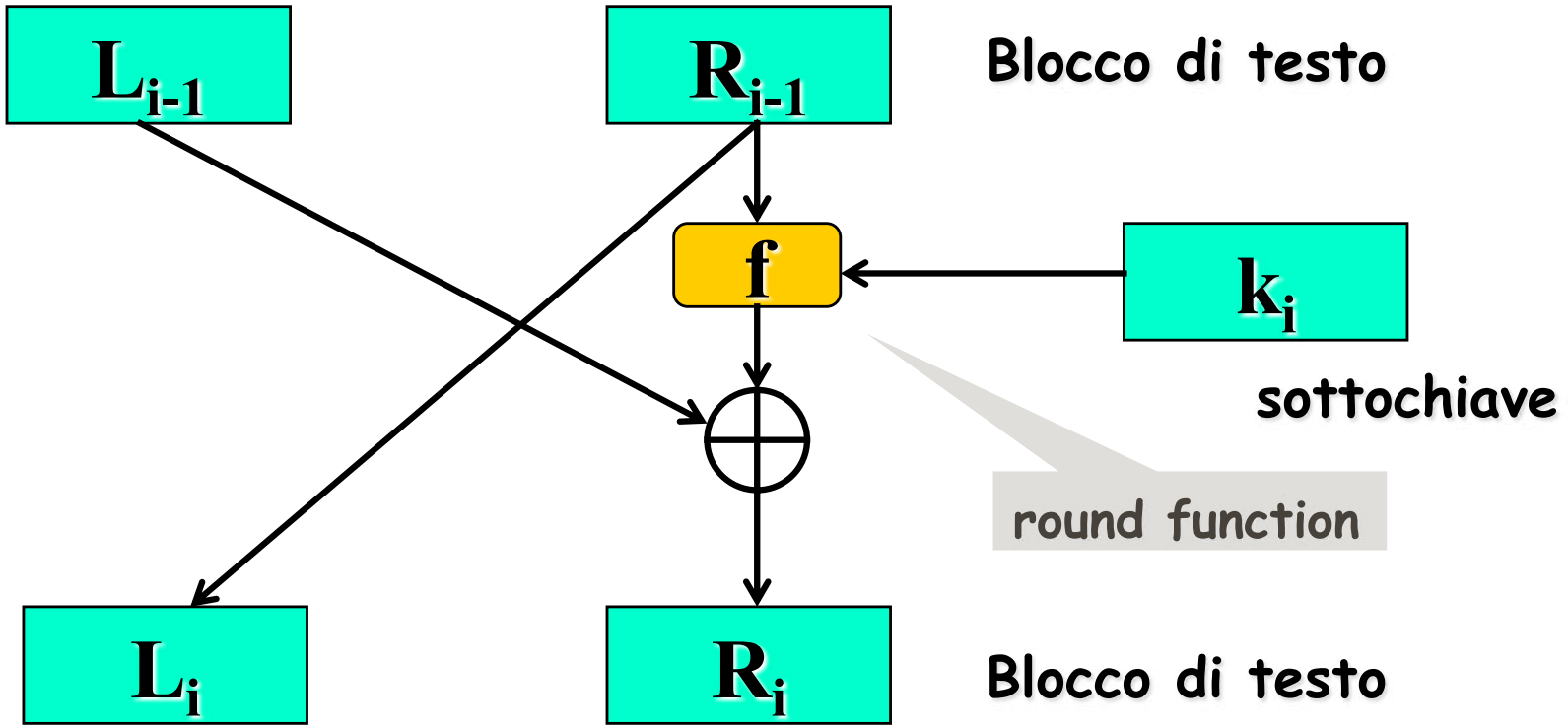
- Proposta di Horst Feistel (1973), basata su una sequenza di permutazioni e sostituzioni
- Si basa sui principi di Shannon (1949)
  - Diffusione
    - Indipendenza della struttura statistica del testo in chiaro da quello cifrato

      (distribuzione delle correlazioni statistiche del testo in chiaro)
  - Confusione
    - Relazione complicata fra testo cifrato e chiave segreta
- Tali principi rendono difficile gli attacchi statistici

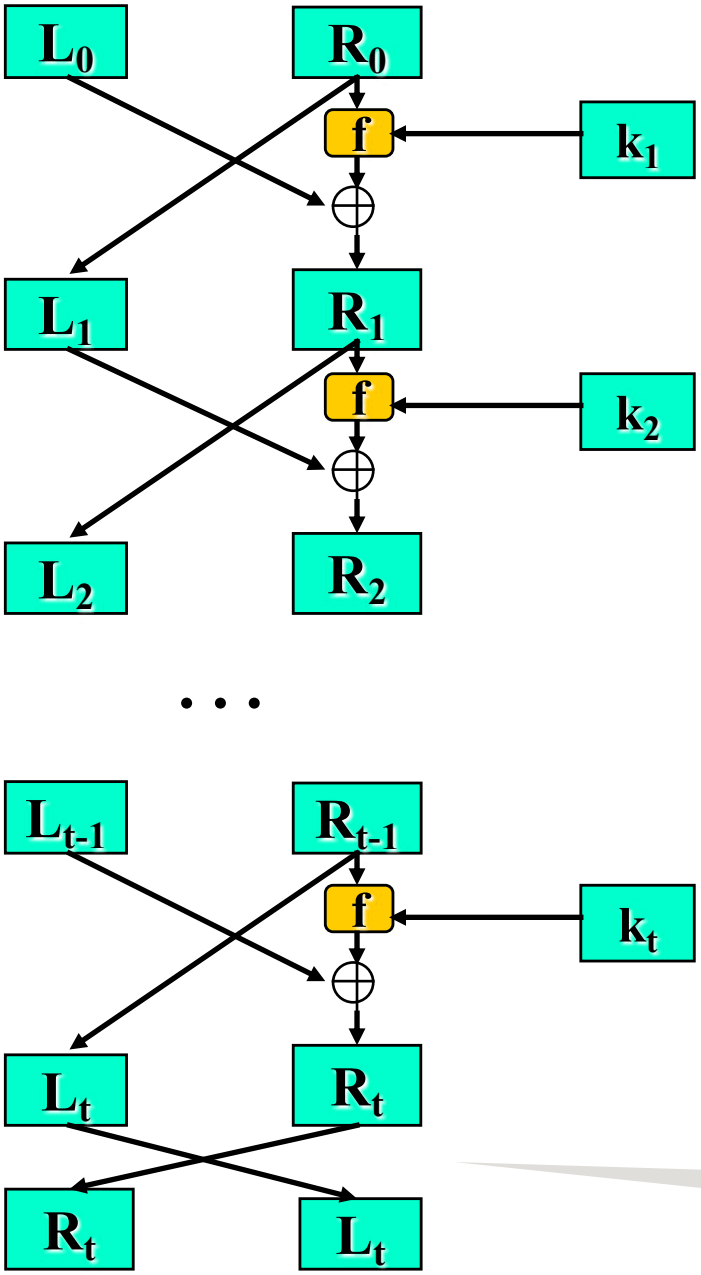
Cifrari di Feistel: Caratteristiche

- Dimensioni del blocco
  - Blocchi grandi migliorano la sicurezza ma riducono la velocità
- Dimensioni della chiave
  - Chiavi grandi migliorano la sicurezza ma riducono la velocità
- Ripetizione di round / iterazione
  - Tutti i round hanno la stessa struttura
- Algoritmo di schedulazione della chiave
  - A partire dalla chiave iniziale vengono prodotte tante sottochiavi quanti sono i round

Cifrari di Feistel: struttura di un round



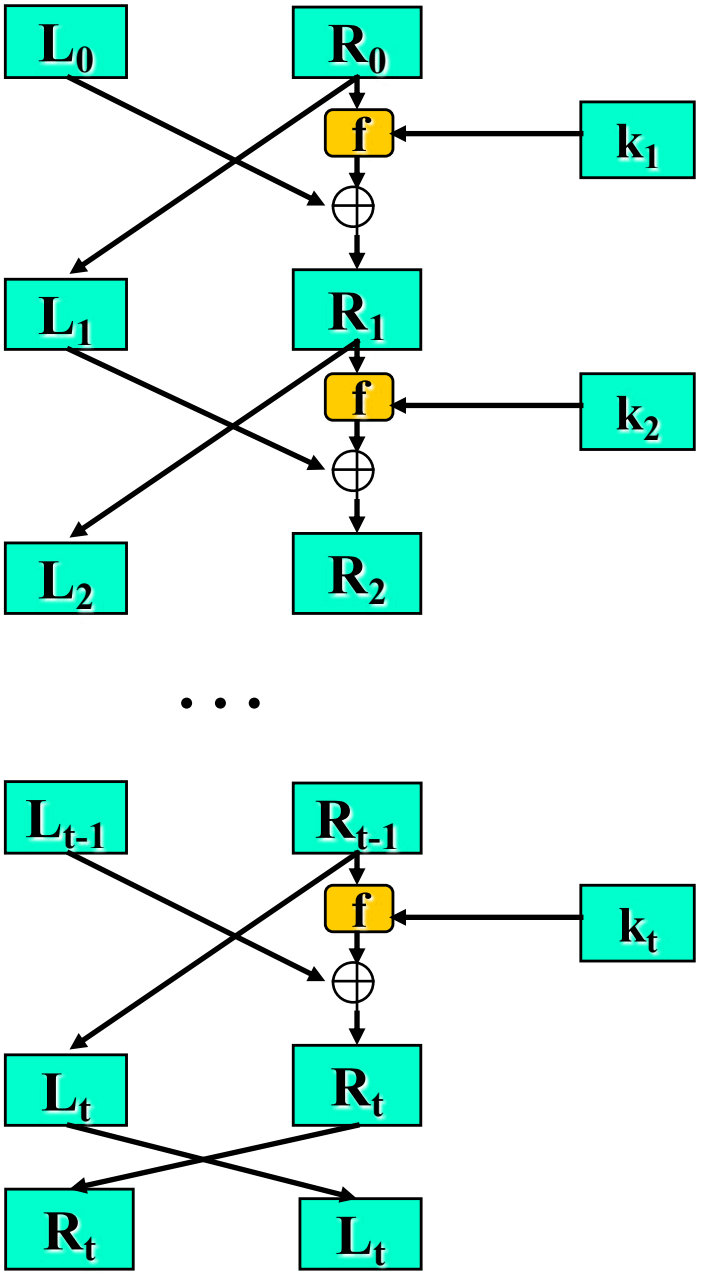
Cifrari di Feistel: cifratura



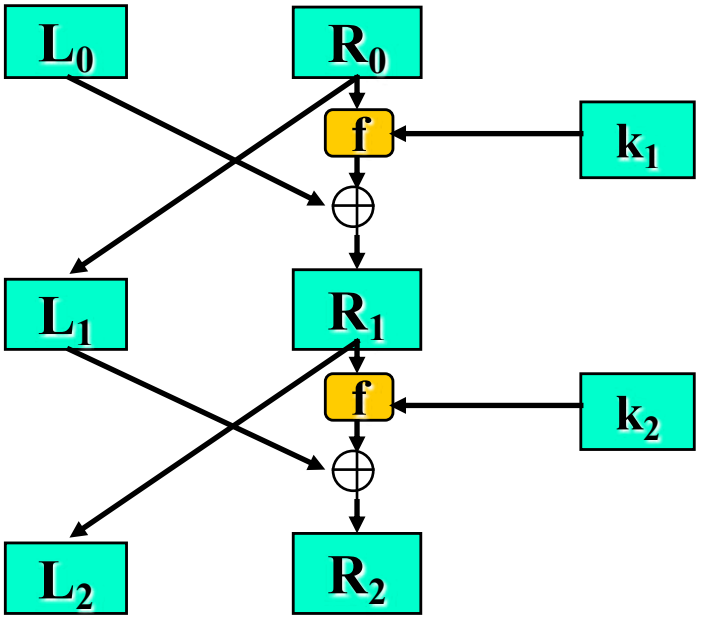
t round

scambio

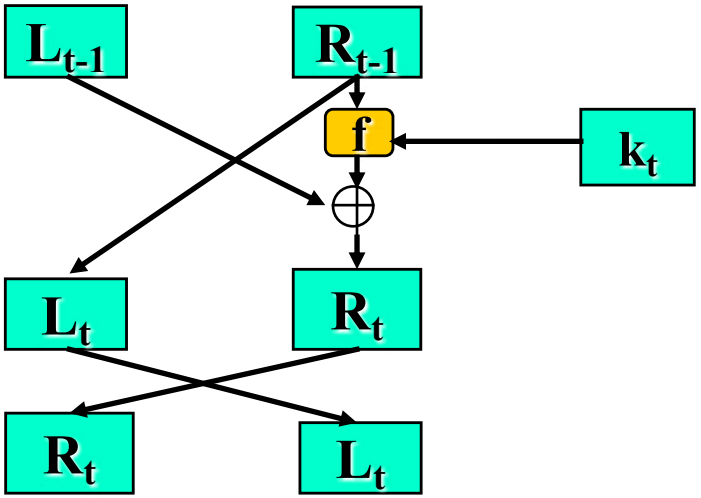
Cifrari di Feistel: decifratura

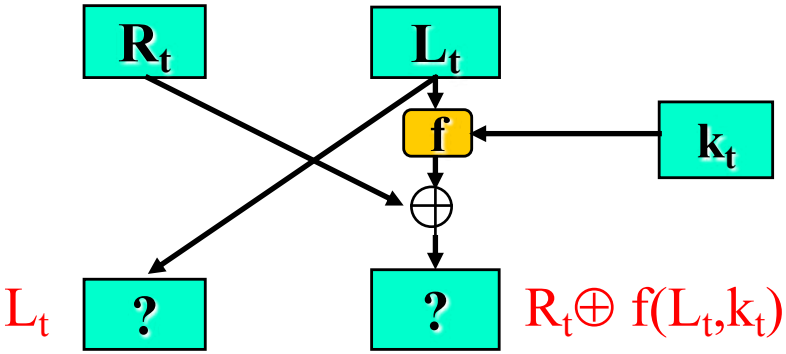


Cifrari di Feistel: decifratura

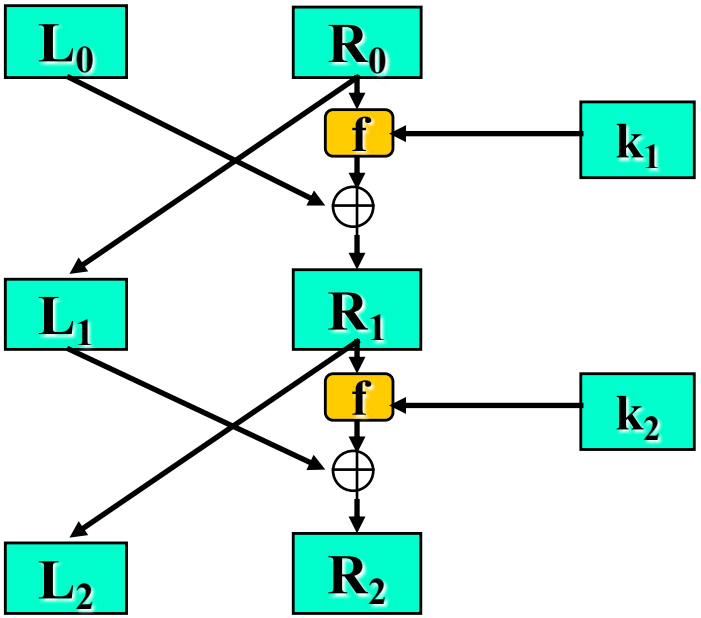


. . .

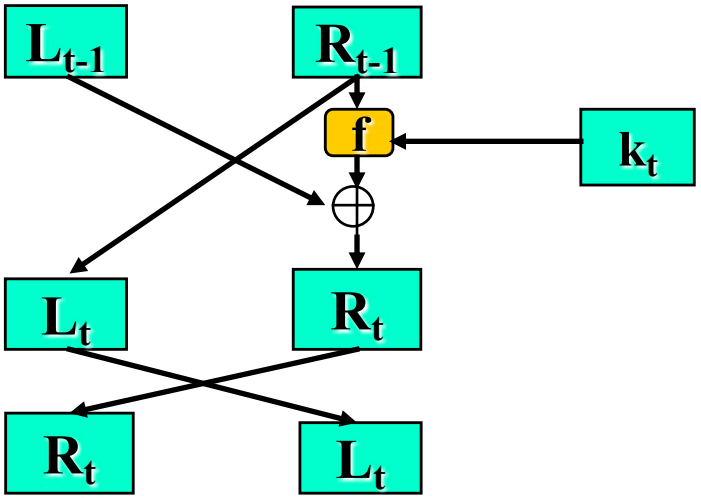


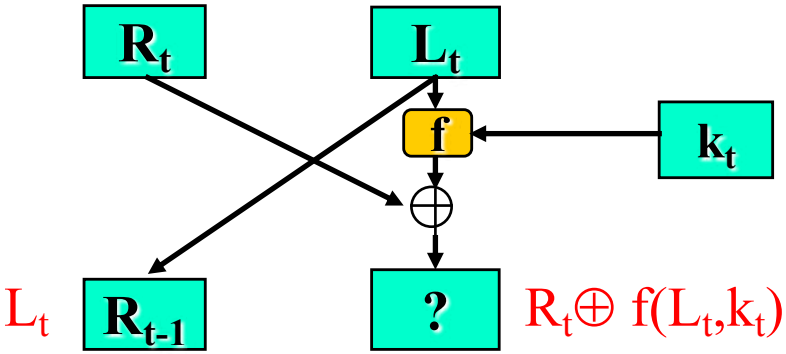


Cifrari di Feistel: decifratura



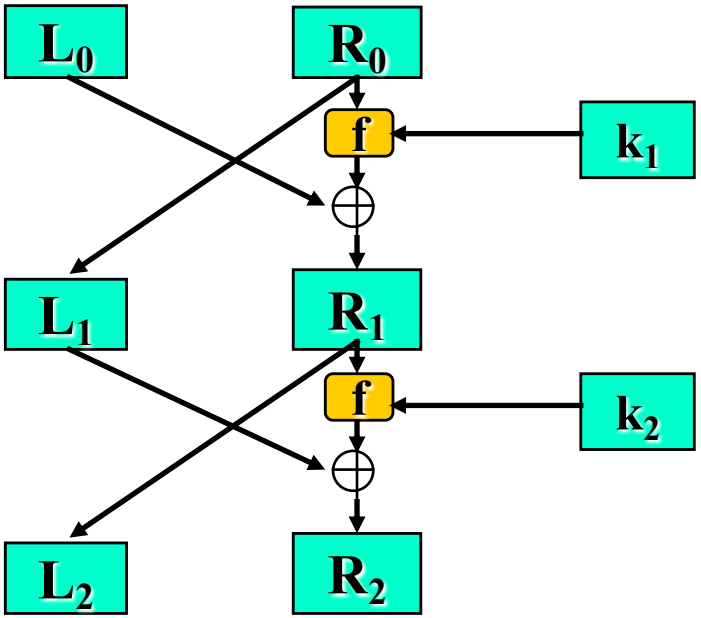
. . .

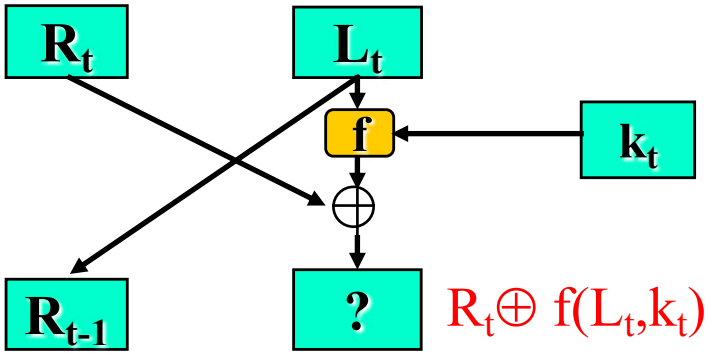




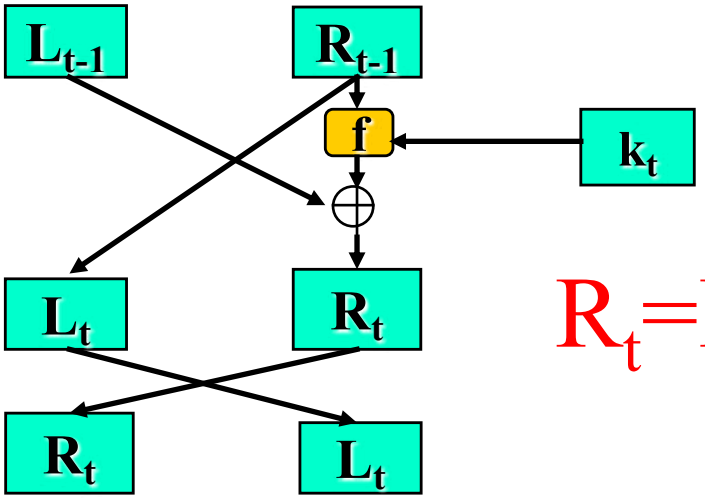
$$L_t = R_{t-1}$$

Cifrari di Feistel: decifratura



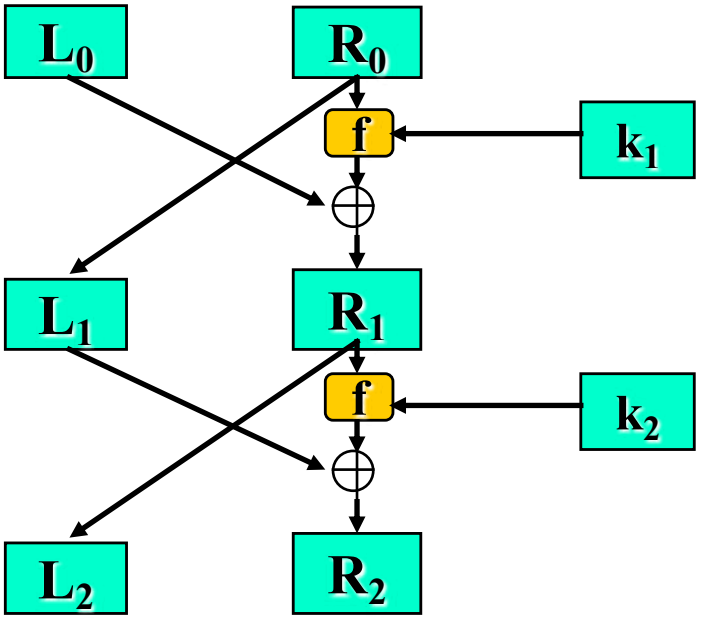


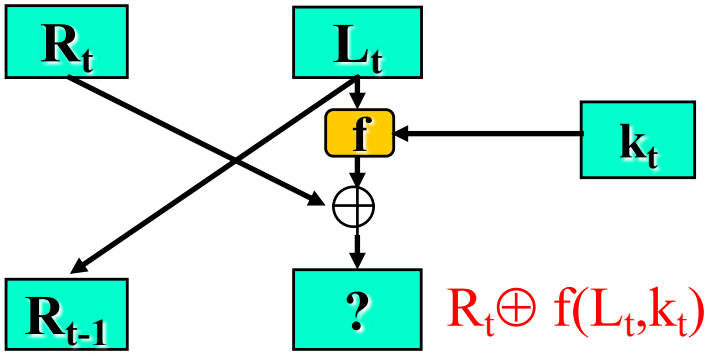
. . .



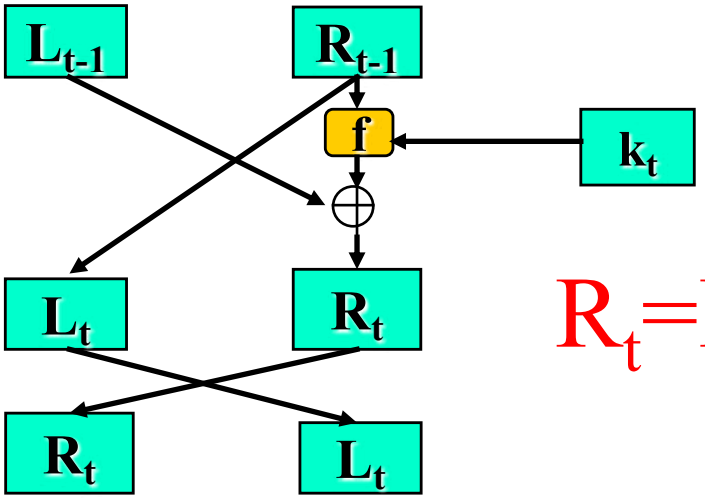
$$R_t = L_{t-1} \oplus f(R_{t-1}, k_t)$$

Cifrari di Feistel: decifratura





. . .

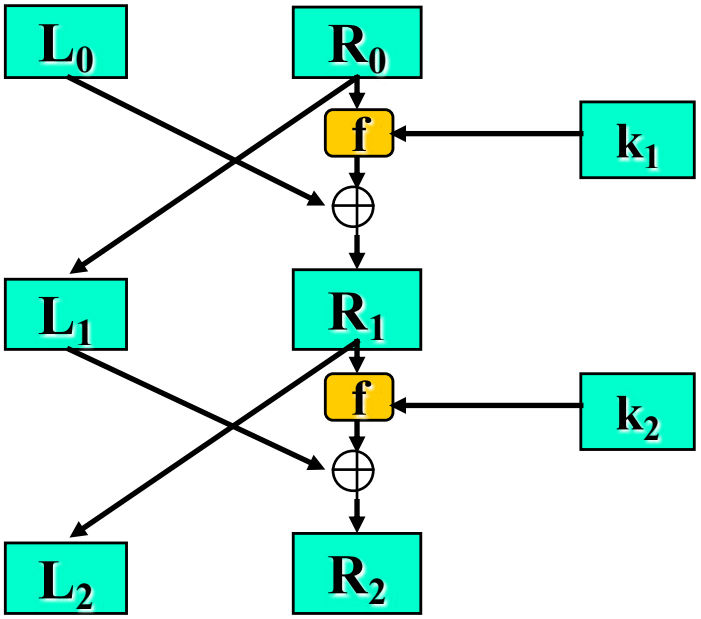


$$L_{t-1} = R_t \oplus f(R_{t-1},k_t)$$

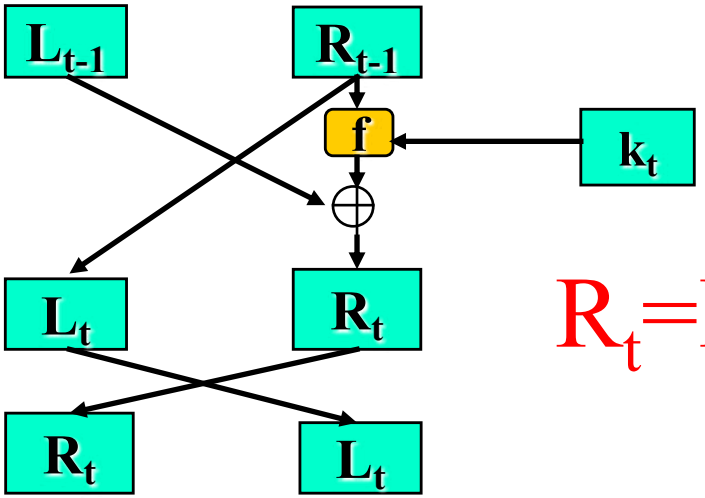
facendo $\oplus f(R_{t-1},k_t)$ ad ambo i membri

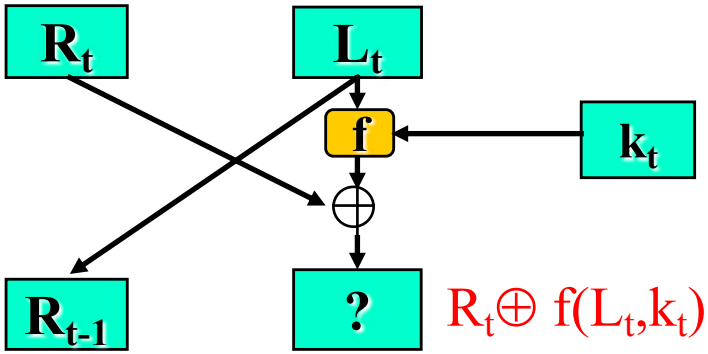
$$R_t = L_{t-1} \oplus f(R_{t-1},k_t)$$

Cifrari di Feistel: decifratura



. . .





$$L_{t-1}=R_t \oplus f(L_t,k_t)$$

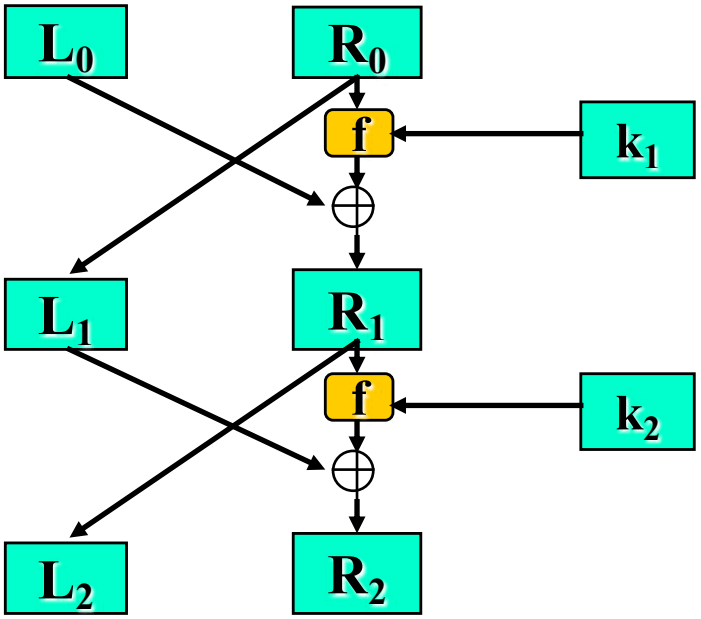
dato che $L_t=R_{t-1}$

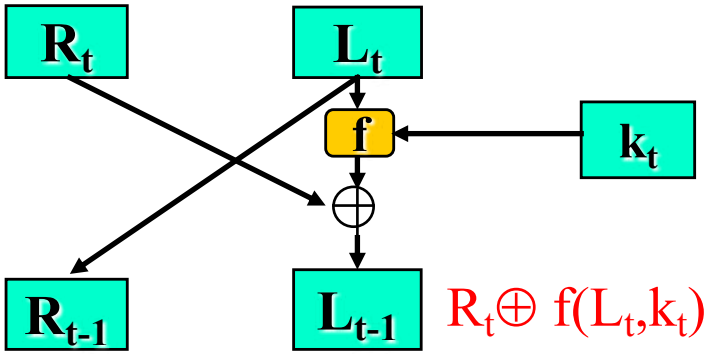
$$L_{t-1}=R_t \oplus f(R_{t-1},k_t)$$

facendo $\oplus f(R_{t-1},k_t)$ ad ambo i membri

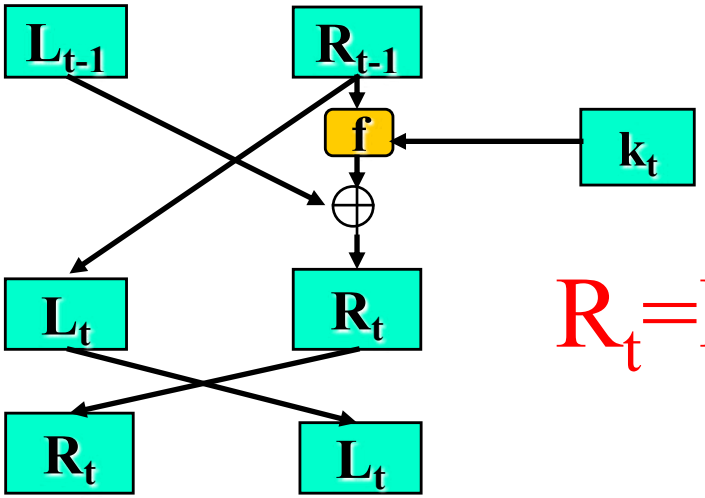
$$R_t=L_{t-1} \oplus f(R_{t-1},k_t)$$

Cifrari di Feistel: decifratura





. . .



$$L_{t-1}=R_t \oplus f(L_t,k_t)$$

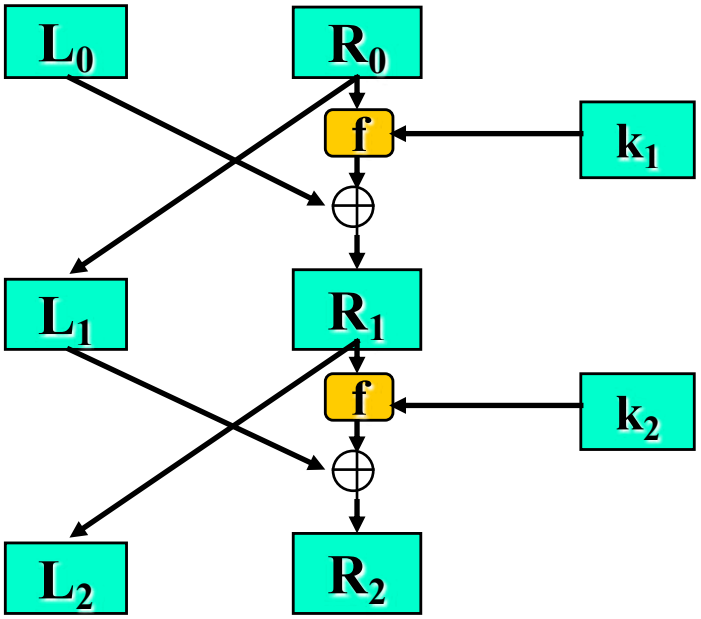
dato che $L_t=R_{t-1}$

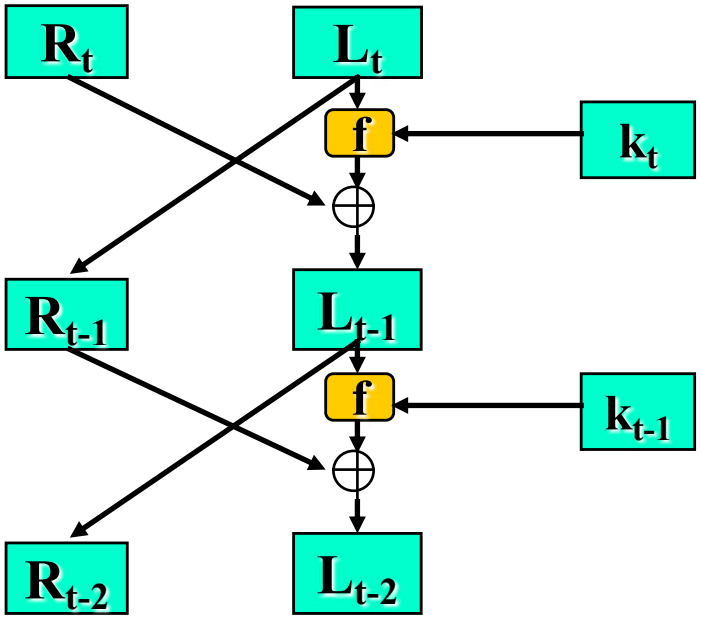
$$L_{t-1}=R_t \oplus f(R_{t-1},k_t)$$

facendo $\oplus f(R_{t-1},k_t)$ ad ambo i membri

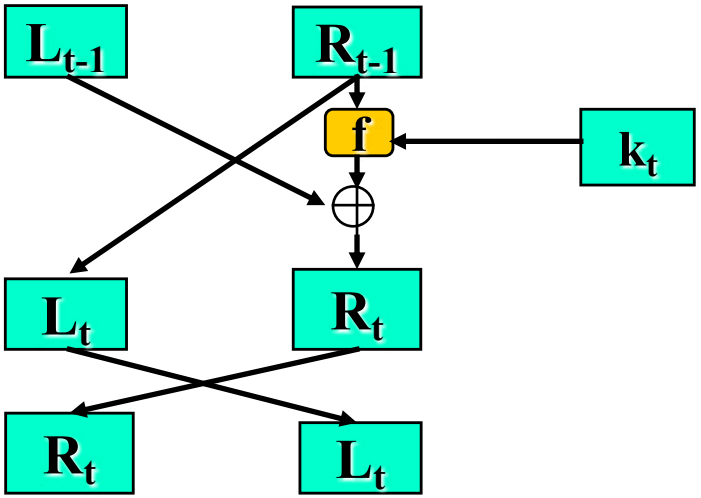
$$R_t=L_{t-1} \oplus f(R_{t-1},k_t)$$

Cifrari di Feistel: decifratura

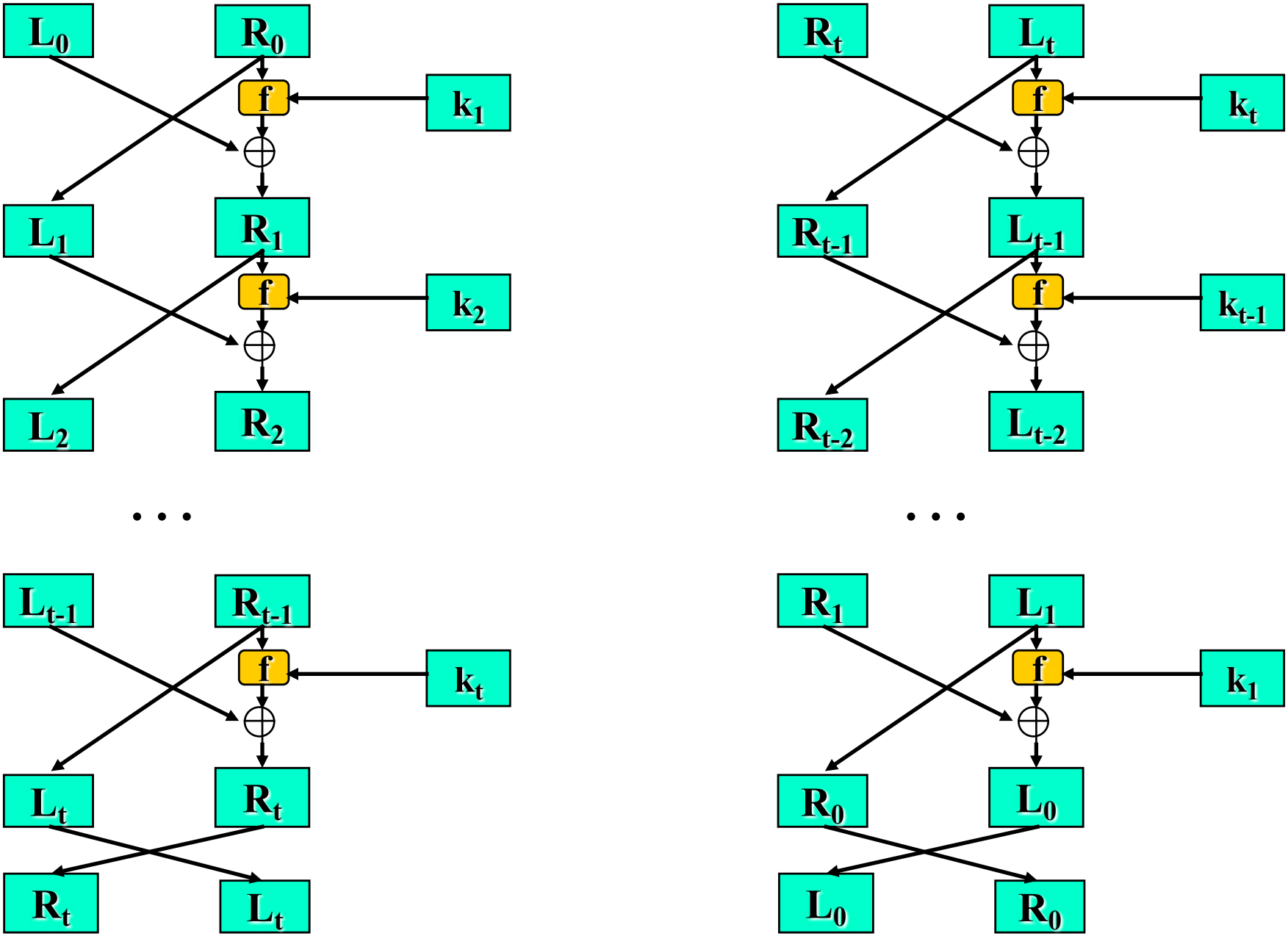




. . .



Cifrari di Feistel: decifratura



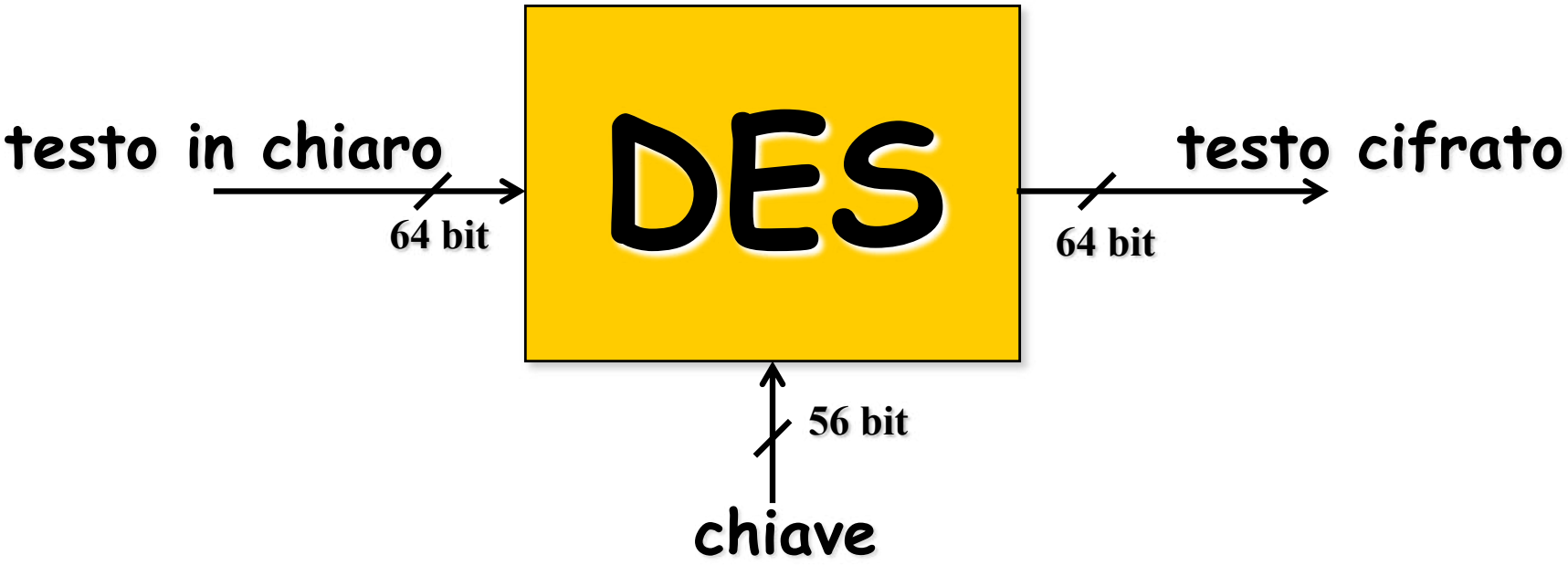
Cifrari di Feistel

- Cifratura:
  - Basta implementare un solo round
  - Lo stesso codice può essere usato per ogni round
- Decifratura
  - Usa lo stesso algoritmo per la cifratura
  - Usa le sottochiavi in ordine inverso
- Esempi di cifrari di Feistel
  - DES
  - Blowfish

Data Encryption Standard (DES)

- 15 maggio 1973: richiesta pubblica per uno standard della NBS (National Bureau of Standards), oggi NIST (National Institute of Standard and Technology)
- 27 agosto 1974: seconda richiesta
- 1975: Lucifer, sviluppato da IBM nel 1971, viene modificato dalla NSA (chiave da 128 a 56 bit)
- 1976: due workshop
- Standard pubblicato 15 gennaio 1977 (FIPS PUB 46)
- Riaffermato per successivi 5 anni nel
  1983 (FIPS PUB 46-1), 1987, 1992 (FIPS PUB 46-2)
- DES challanges (giugno 1997, luglio 1998, gennaio 1999)
- Riaffermato 1999 (FIPS PUB 46-3)
- Advanced Encryption Standard (AES), pubblicato 26 novembre 2001
- Ritirato FIPS 46-3 (19 maggio 2005)
- COPACOBANA (Cost-Optimized Parallel COde breaker), 2007

Data Encryption Standard



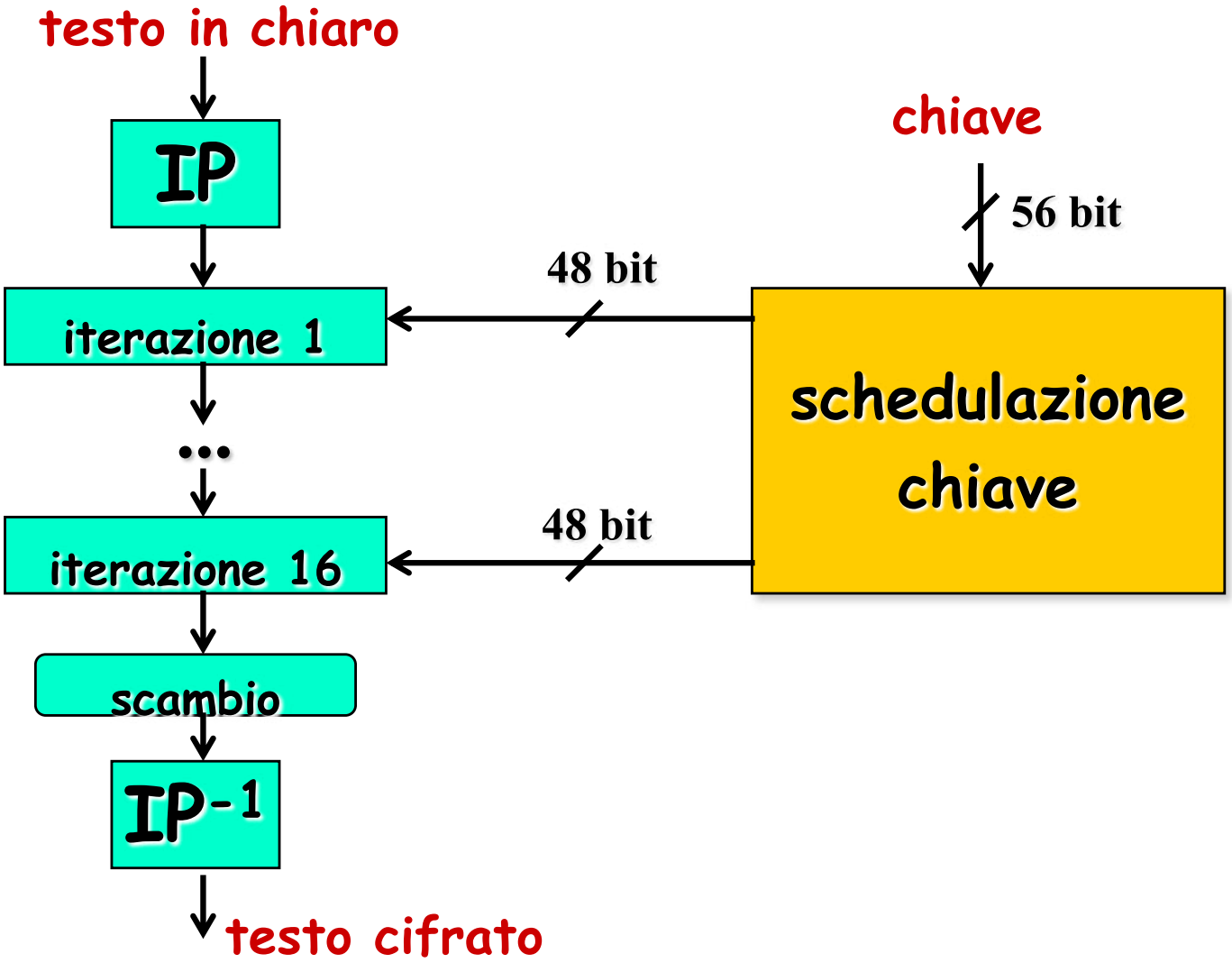
Lunghezza della Chiave

Nello standard DES la chiave è lunga 64 bit:

8 byte di cui l'ottavo bit è di parità

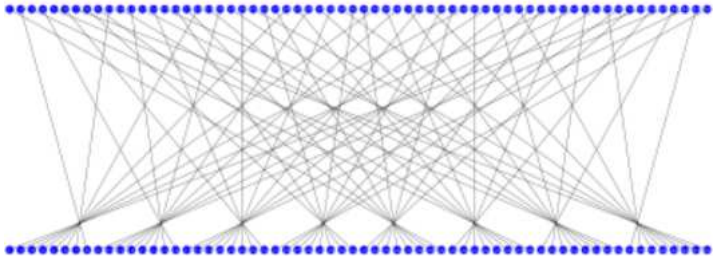
è lo xor dei precedenti 7 bit

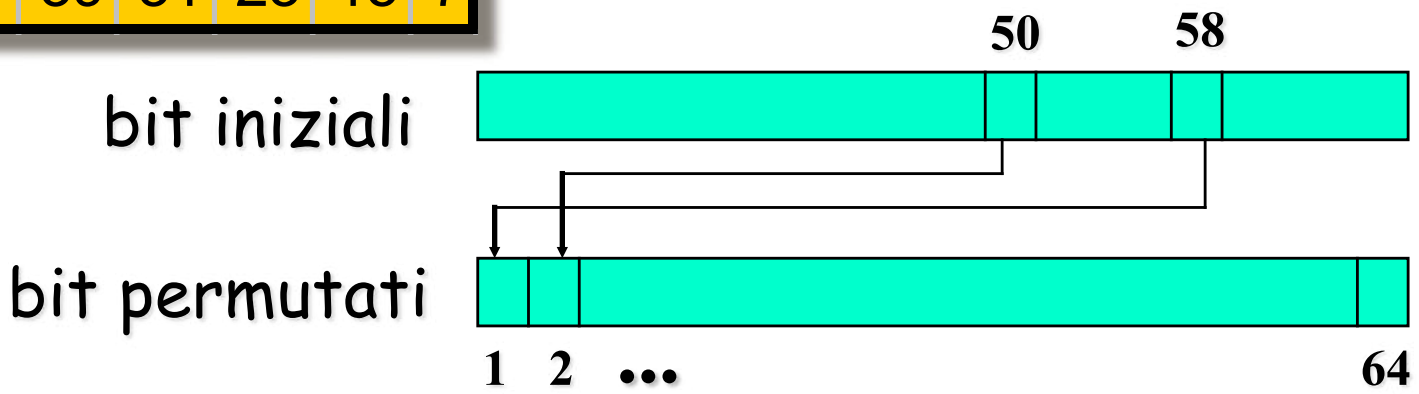
Struttura del DES



Permutazione Iniziale IP

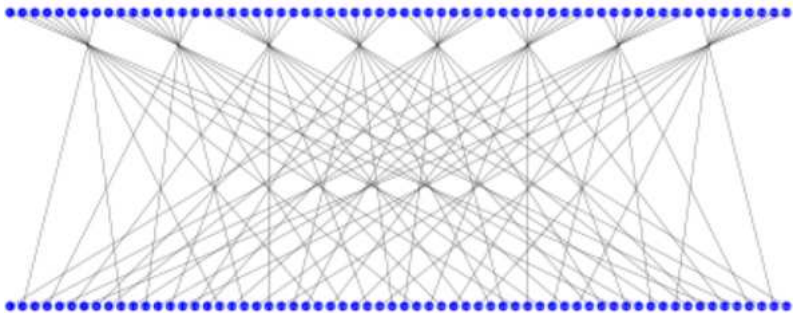
58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

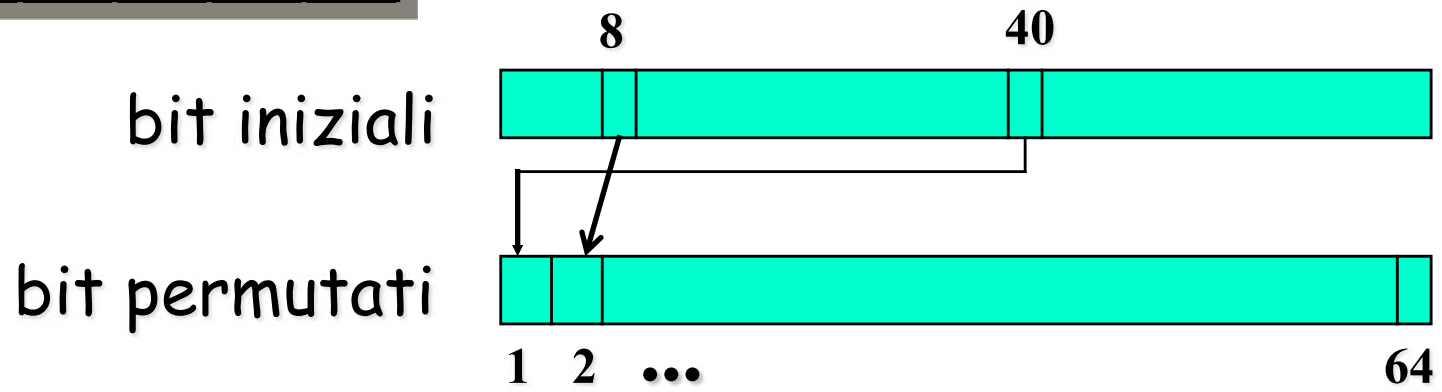




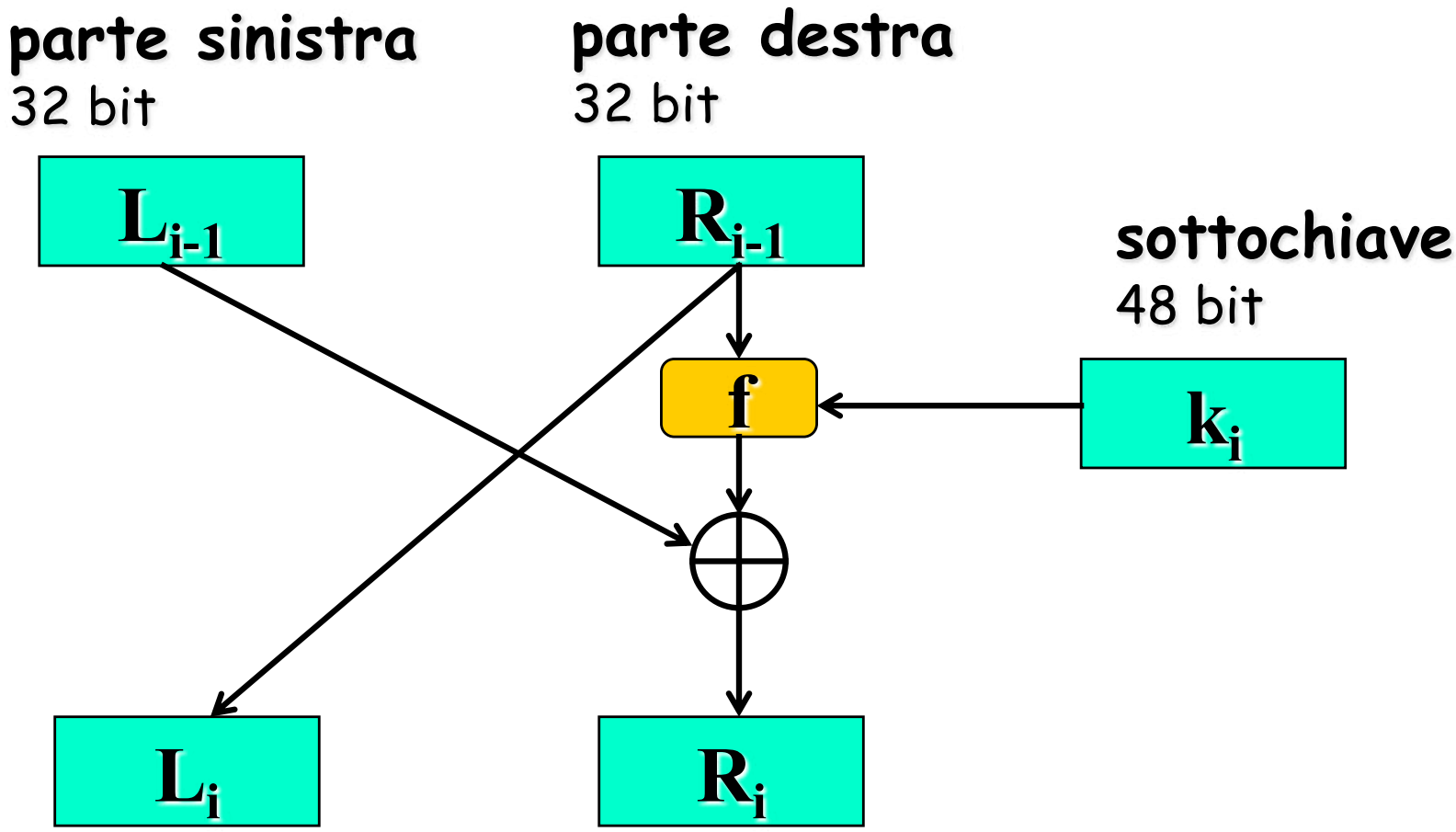
Permutazione Inversa IP^{-1}

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

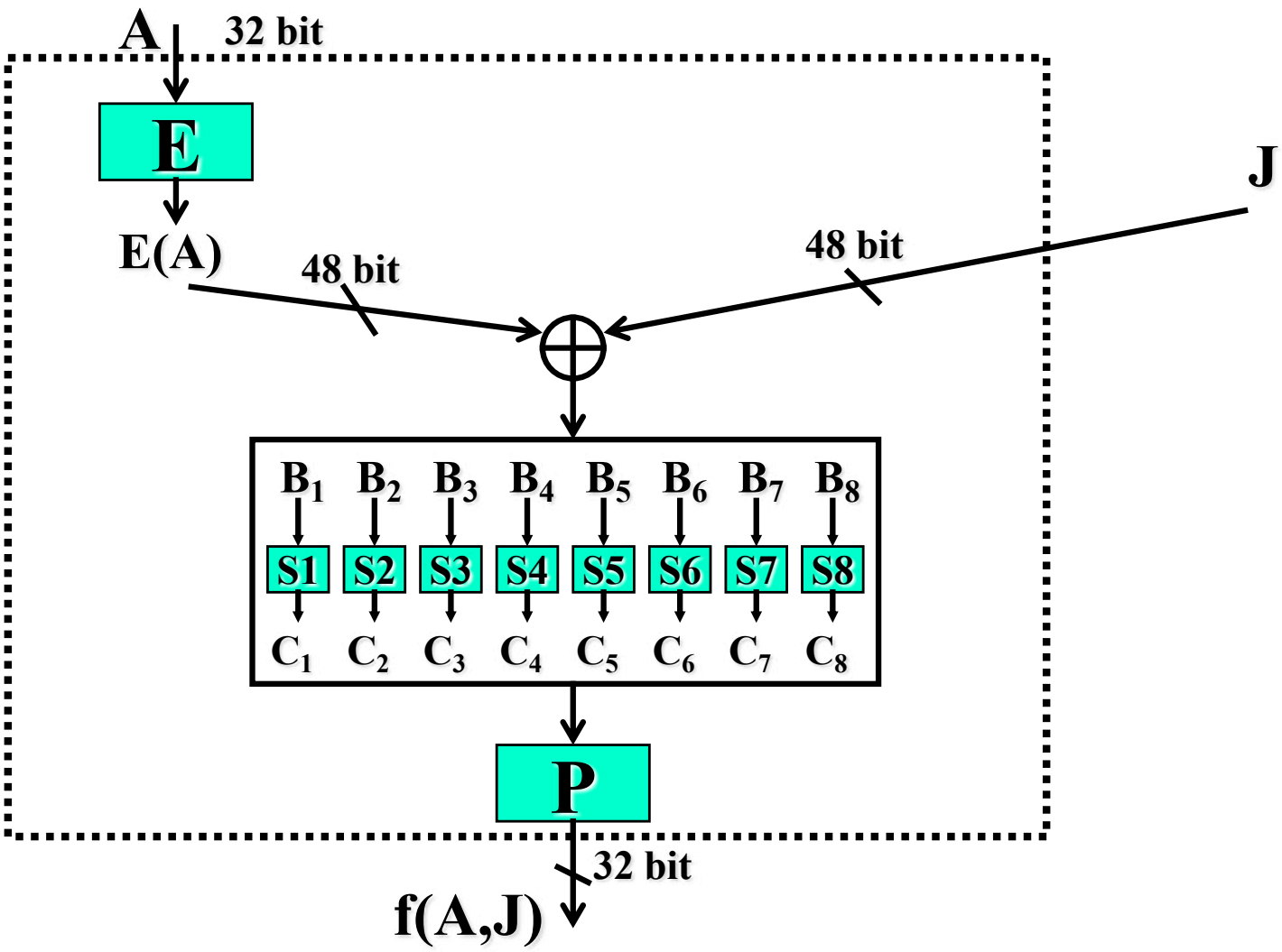




Singola Iterazione



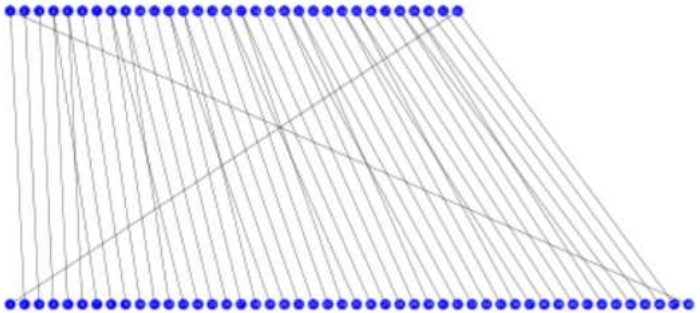
La funzione f

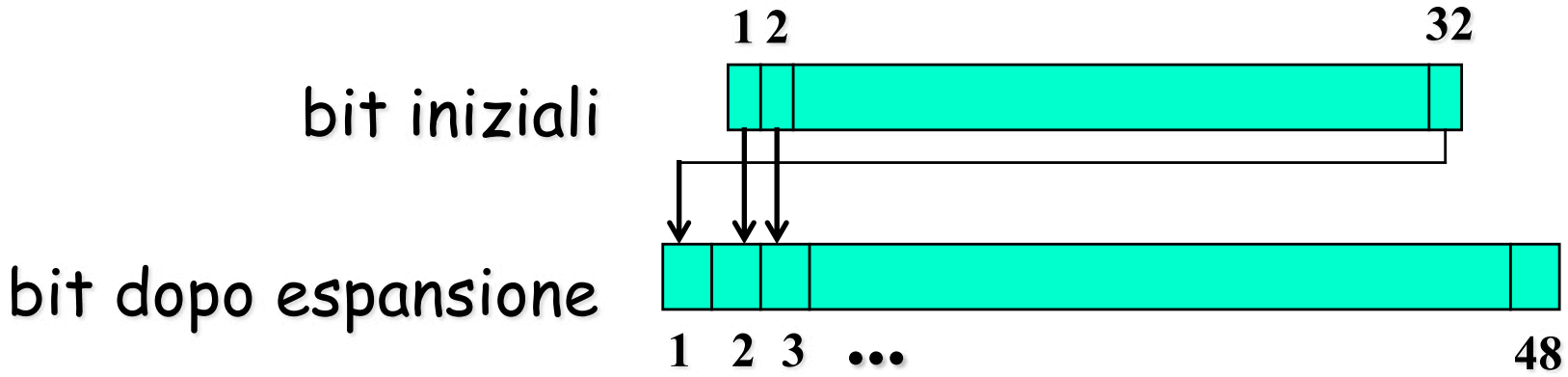


Espansione E

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

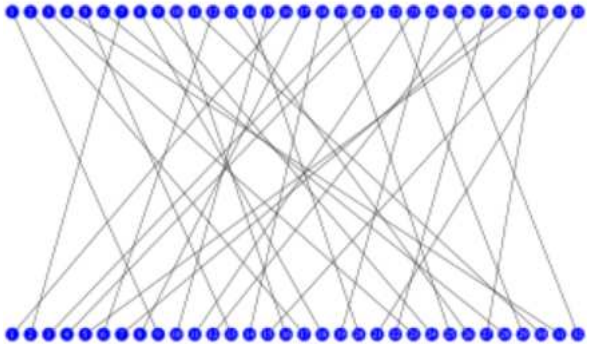
16 bit sono duplicati

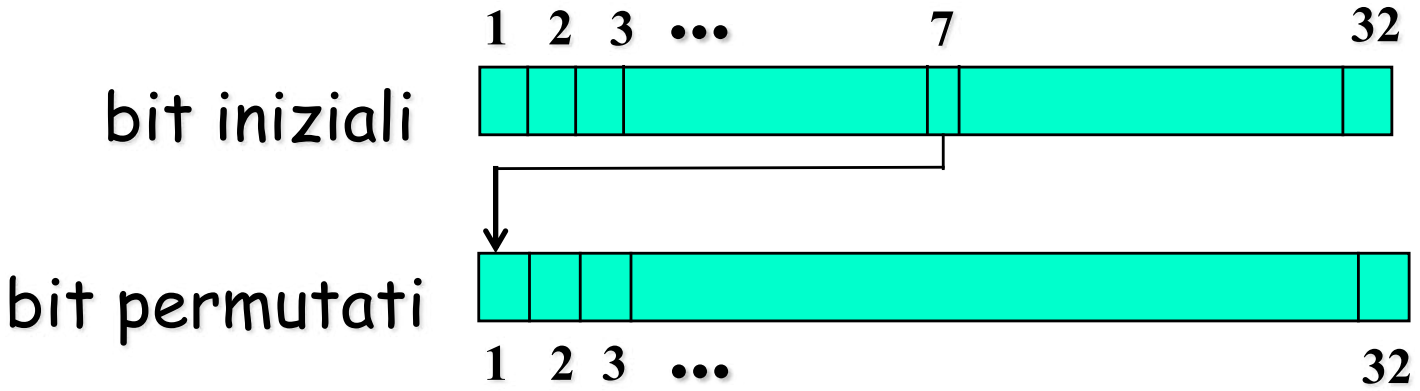




Permutazione P

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





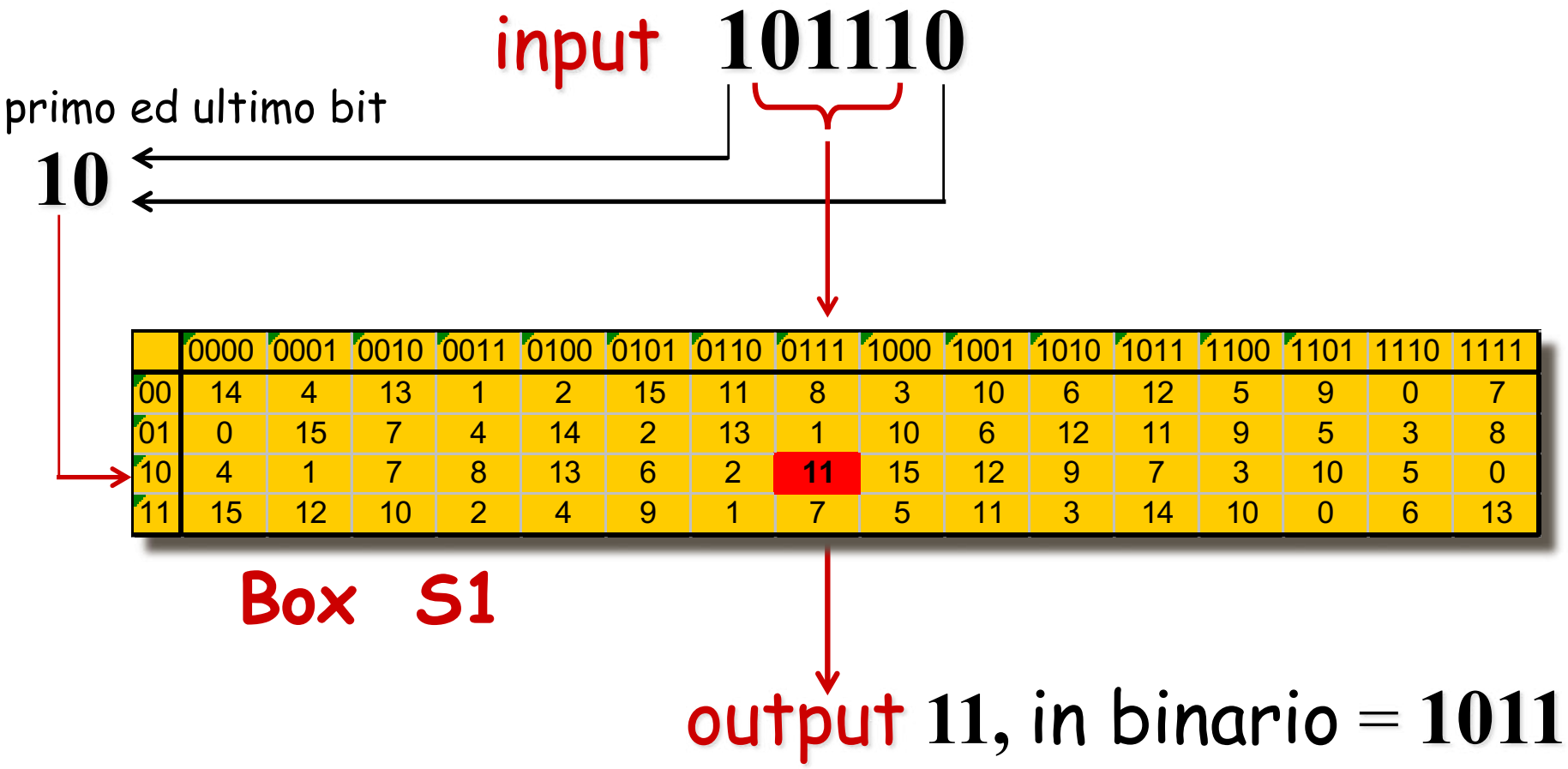
S-box

	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

6 bit in input specificano un elemento della tabella la cui conversione binaria dà i 4 bit output

Funzionamento delle S-box



	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	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

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

S-box non lineari

Se le S-box fossero state lineari, per es.,

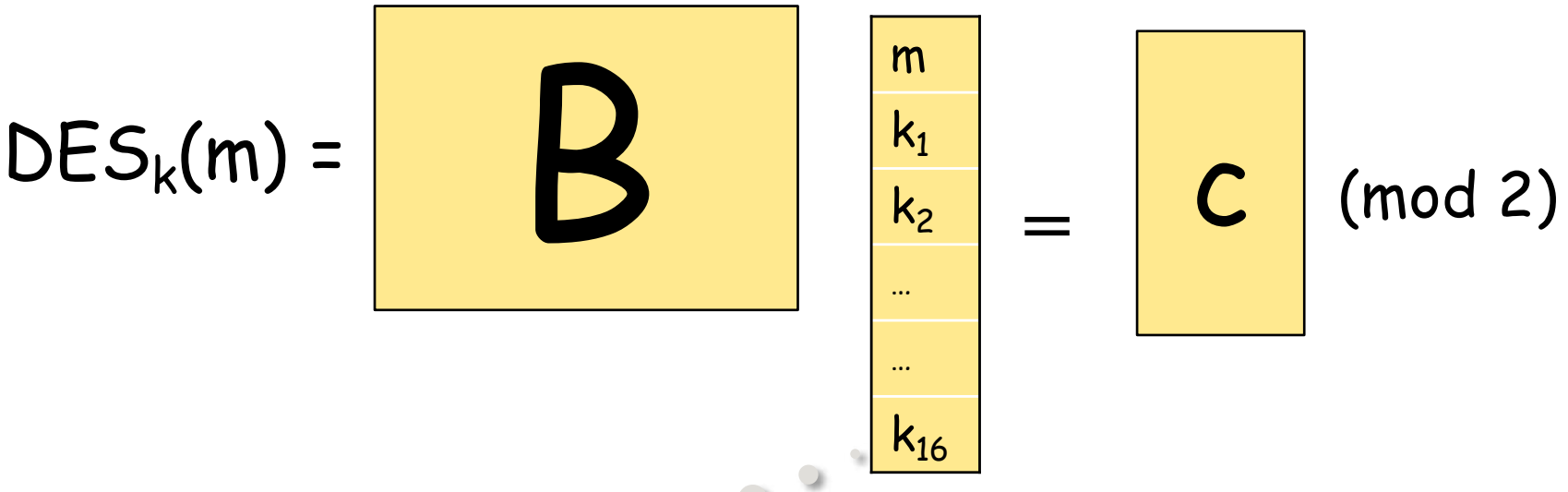
$S_i(x_1,x_2,\ldots,x_6) = (x_1x_3,\ x_1x_4x_5,\ x_1x_6,\ x_2x_3x_6)$

cioè:

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} x_1x_3 \\ x_1x_4x_5 \\ x_1x_6 \\ x_2x_3x_6 \end{bmatrix} \pmod 2$$

S-box non lineari

Allora il cifrario DES sarebbe stato lineare, cioè esistebbe una matrice 64x832 B tale che

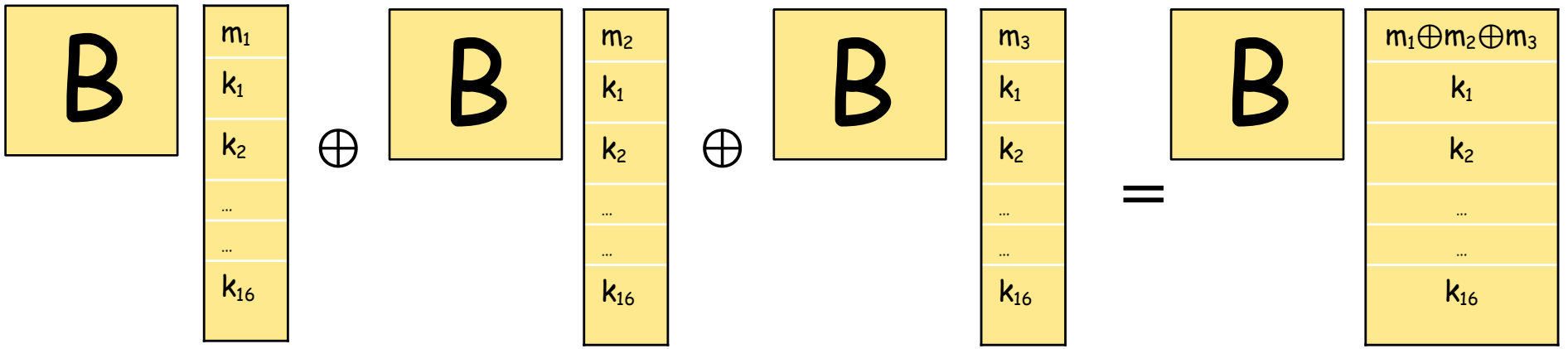


Lunghezza
64 + 16 · 48 = 832

S-box non lineari

Se il cifrario DES fosse stato lineare, allora

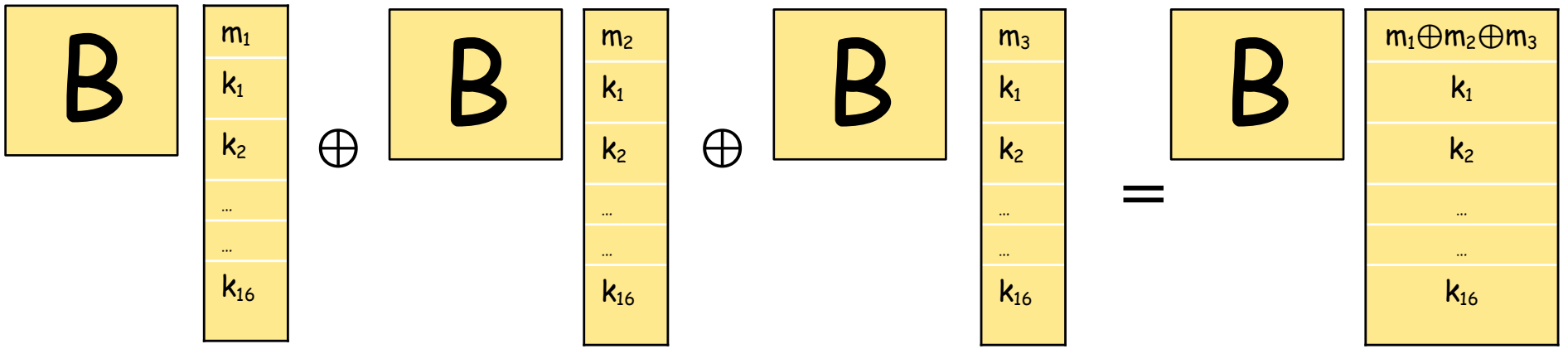
$$DES_k(m_1) \oplus DES_k(m_2) \oplus DES_k(m_3) = DES_k(m_1 \oplus m_2 \oplus m_3)$$



S-box non lineari

Se il cifrario DES fosse stato lineare, allora

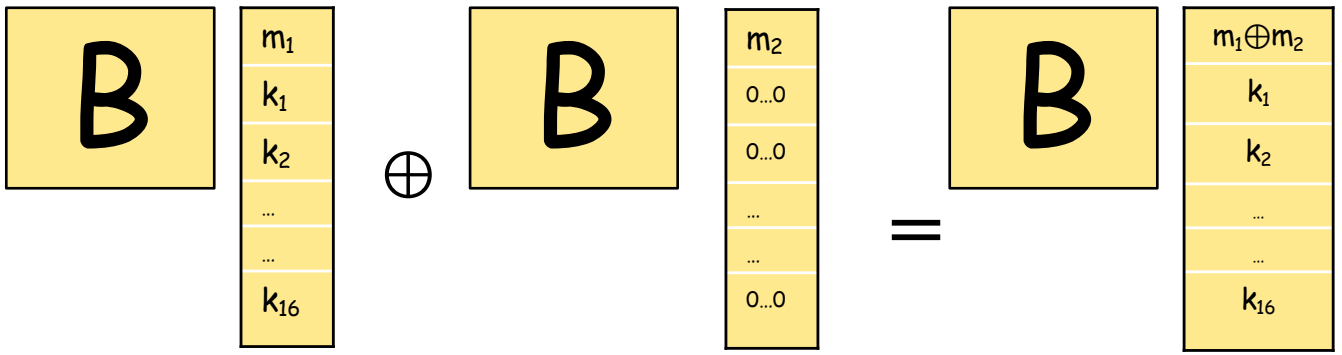
$$DES_k(m_1) \oplus DES_k(m_2) \oplus DES_k(m_3) = DES_k(m_1 \oplus m_2 \oplus m_3)$$



Esercizio: mostrare la validità dell'uguaglianza

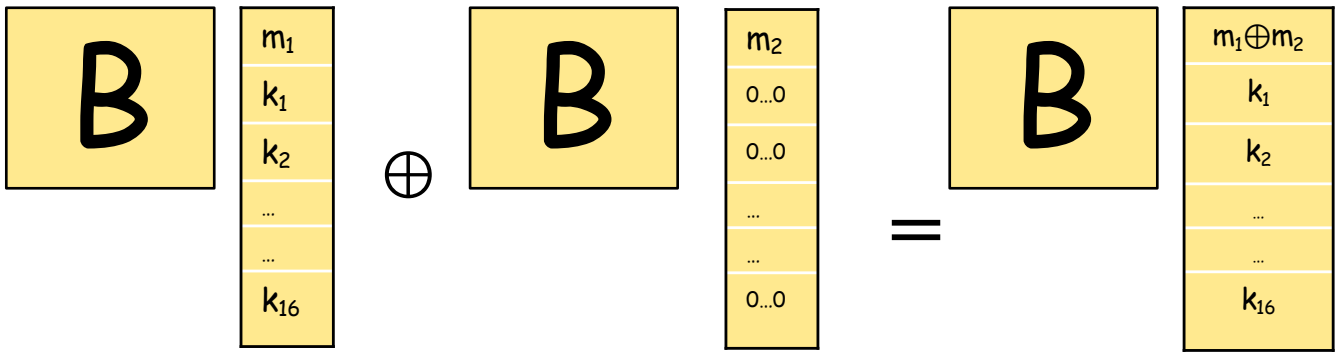
S-box non lineari

Ancora, se il cifrario DES fosse stato lineare, allora



S-box non lineari

Ancora, se il cifrario DES fosse stato lineare, allora

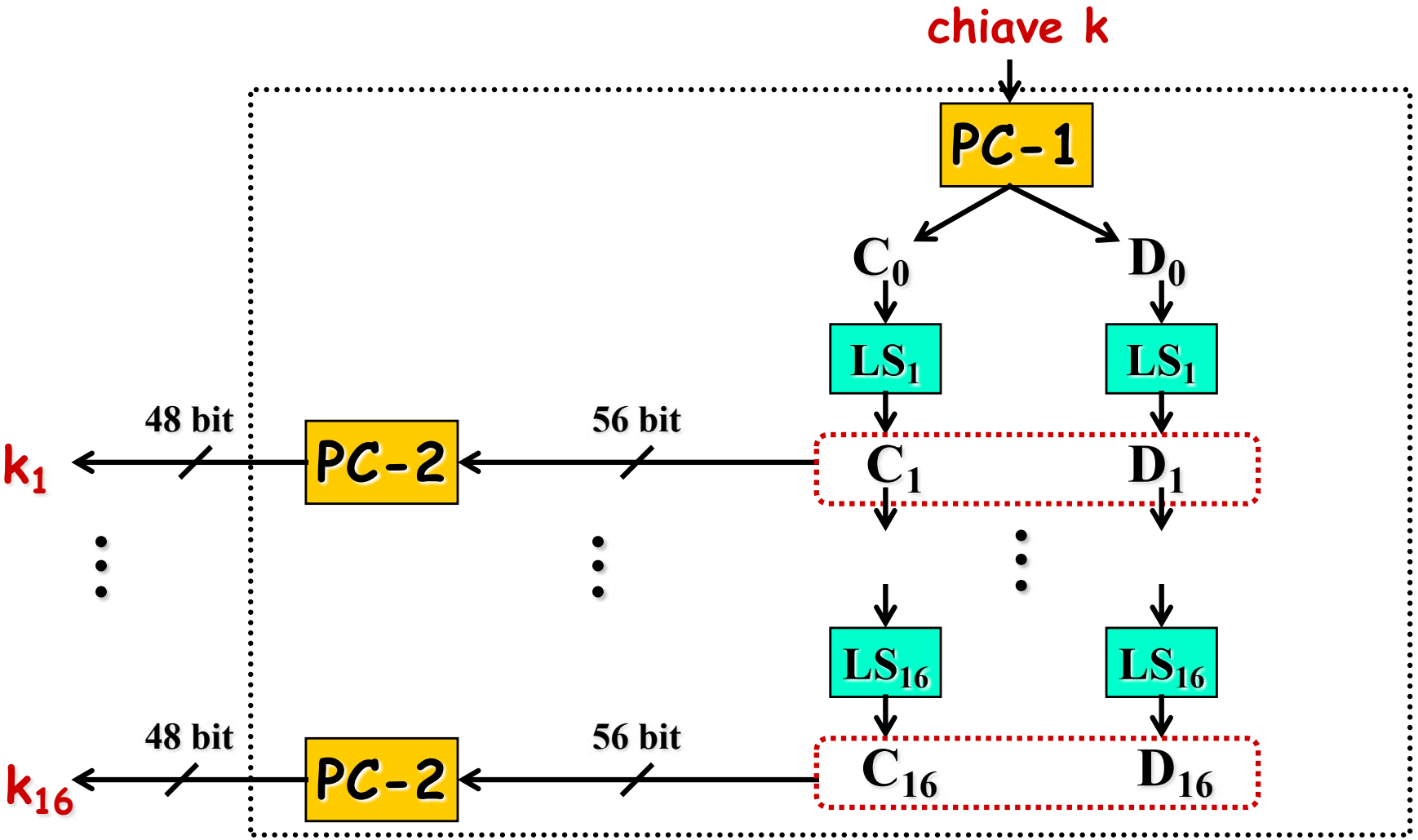


Esercizio: mostrare la validità dell'uguaglianza

Proprietà delle S-box [NBS, 1976]

- Ogni riga è una permutazione degli interi 0,..,15
- Nessuna S-box è una funzione lineare dei suoi input
- Cambiando un solo bit di input ad una S-box variano almeno due bit nell'output
- Per ogni S-box S e per ogni input x a 6 bit:
  $S(x)$ e $S(x \oplus 001100)$ differiscono in almeno due bit
- Per ogni S-box, per ogni input x e per ogni bit d,g,
  $S(x) \neq S(x \oplus 11dg00)$
- Per ogni S-box, il numero degli input per i quali il bit di output è 0 è circa uguale al numero degli input per i quali tale bit è 1

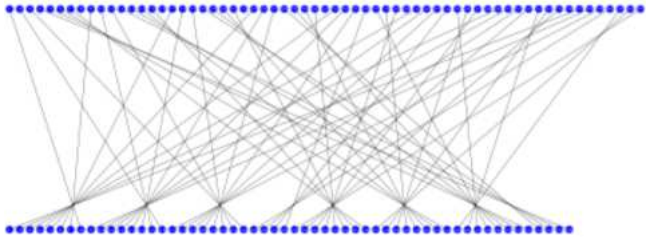
Schedulazione delle chiavi

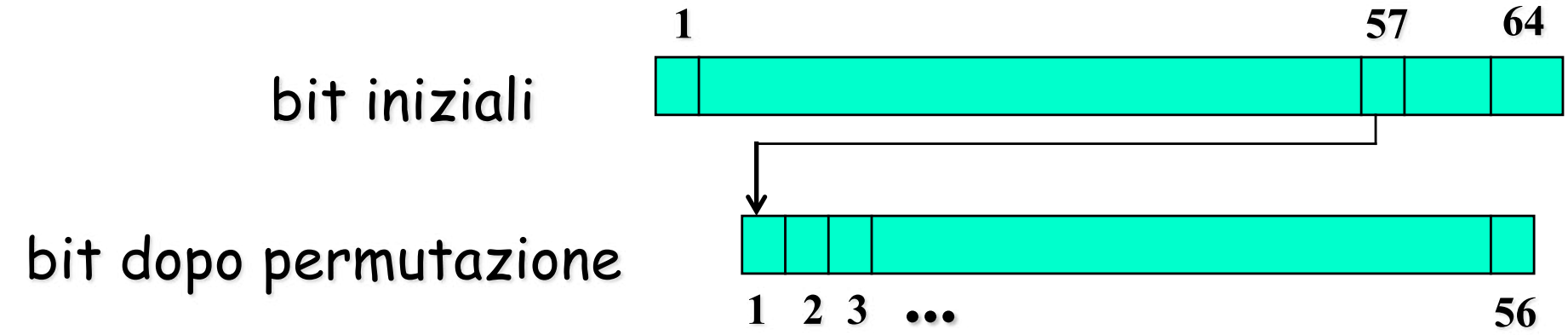


Permutazione PC-1

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

i bit in posizione 8, 16, 24, 32, 40, 48, 56, 64
sono di parità e non compaiono



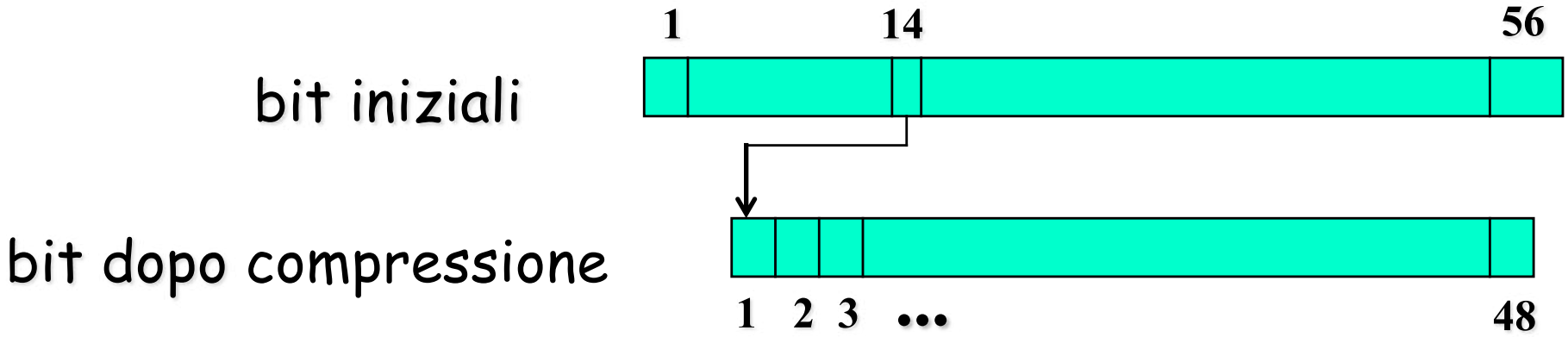


Compressione-permutazione PC-2

14	17	11	24	1	5	3	28	15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	26	41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56	34	53	46	42	50	36	29	32

8 bit soppressi in posizione

9, 18, 22, 25, 35, 38, 43 e 54

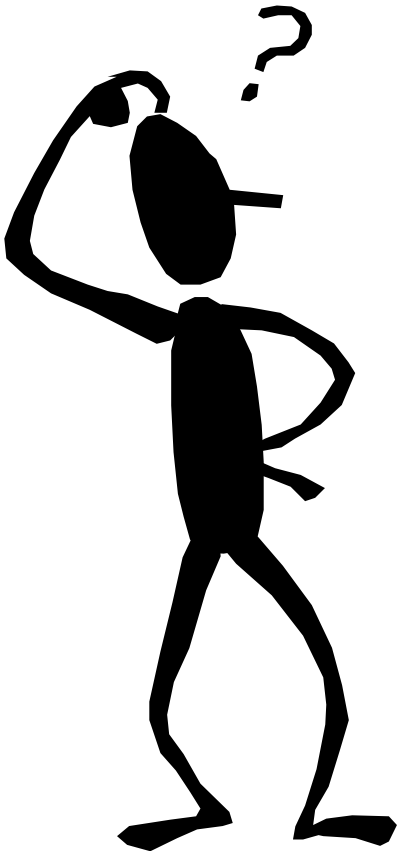


Schedulazione shift a sinistra

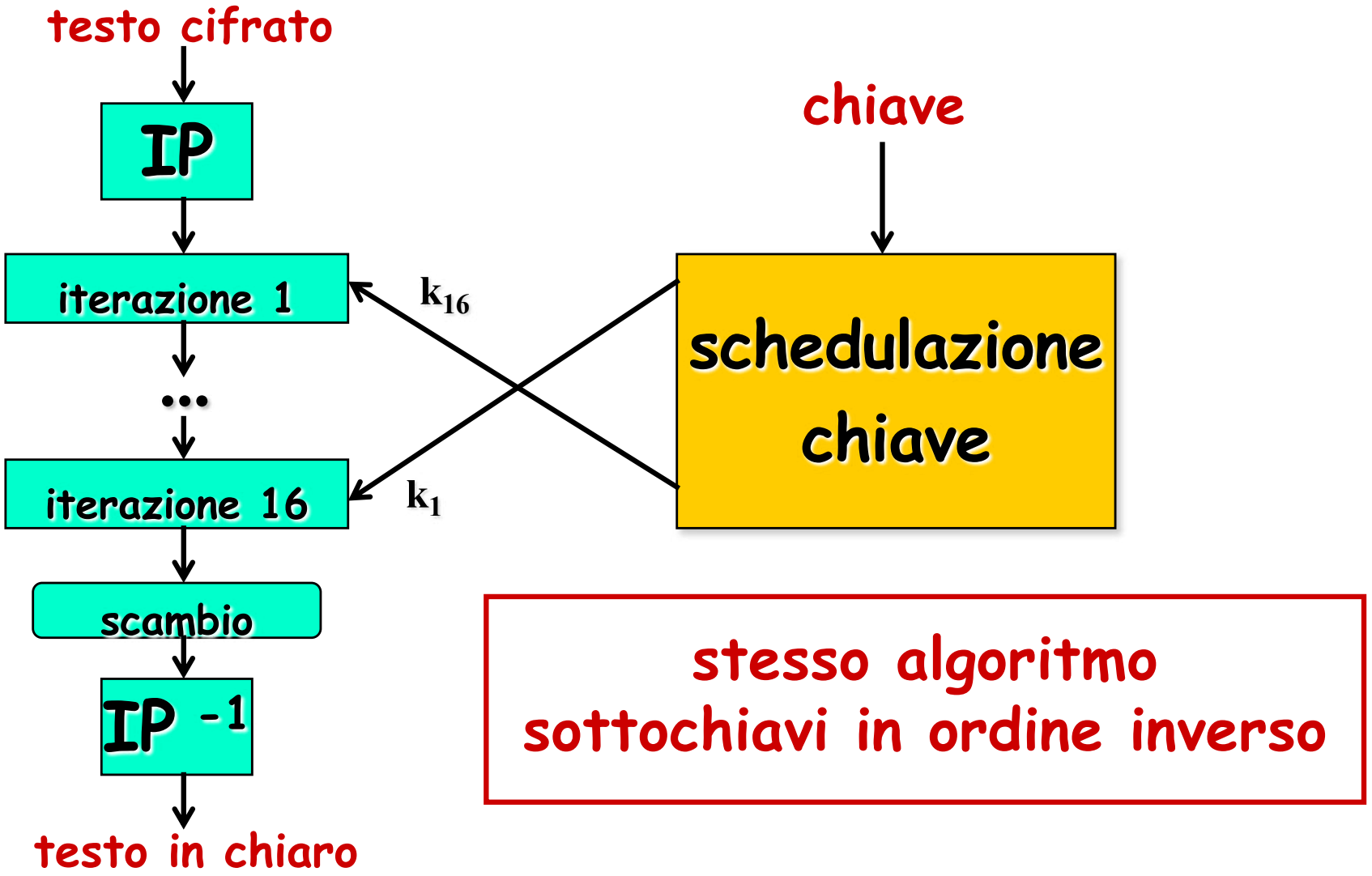
iterazione	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
shift	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Totale shift nelle 16 iterazioni = 28 posizioni

Decifratura DES



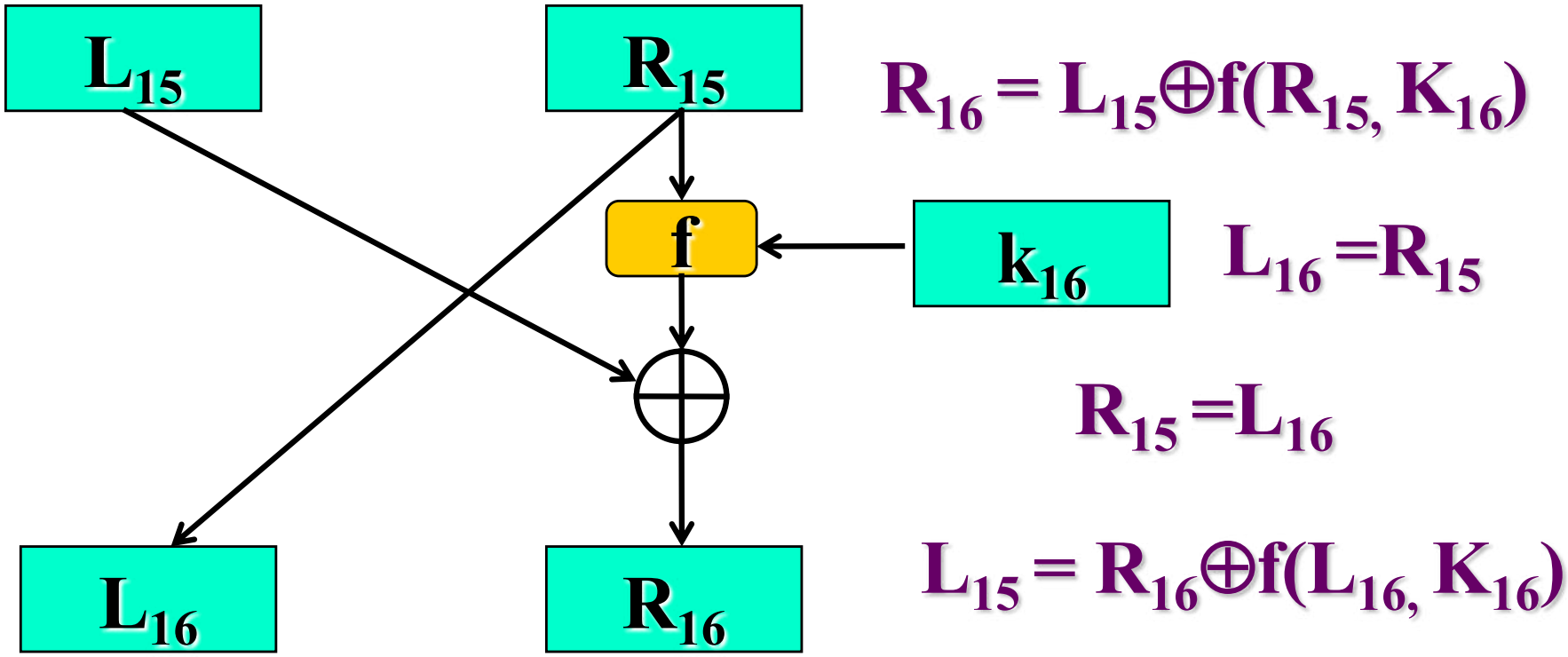
Decifratura DES



stesso algoritmo
sottochiavi in ordine inverso

Decifratura DES

L'ultimo round dell'operazione di cifratura:



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

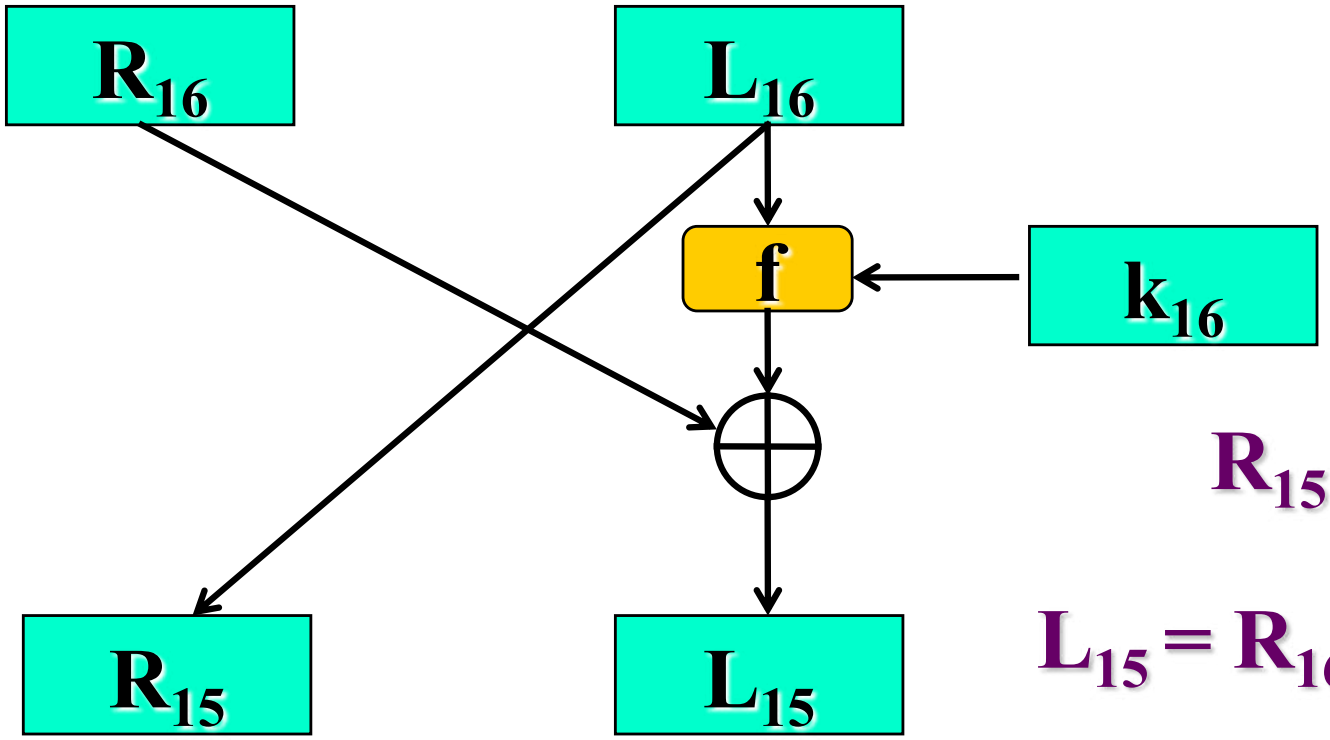
$$L_{16} = R_{15}$$

$$R_{15} = L_{16}$$

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

Decifratura DES

Il primo round dell'operazione di decifratura:



$$R_{15} = L_{16}$$

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

Avalanche Effect

- Un piccolo cambiamento del testo in chiaro oppure della chiave produce un grande cambiamento del testo cifrato

- DES mostra un forte avalanche effect

Avalanche Effect

Esempio

Plaintext 1	00000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
Plaintext 2	10000000	00000000	00000000	00000000	00000000	00000000	00000000	00000000
Key	00000001	1001011	0100100	1100010	0011100	0011000	0011100	0110010

(a) Change in Plaintext

Round	Number of bits that differ
0	1
1	6
2	21
3	35
4	39
5	34
6	32
7	31
8	29
9	42
10	44
11	32
12	30
13	30
14	26
15	29
16	34

Avalanche Effect

Esempio

plaintext	01101000	10000101	00101111	01111010	00010011	01110110	11101011	10100100
Key 1	1110010	1111011	1101111	0011000	0011101	0000100	0110001	11011100
Key 2	0110010	1111011	1101111	0011000	0011101	0000100	0110001	11011100

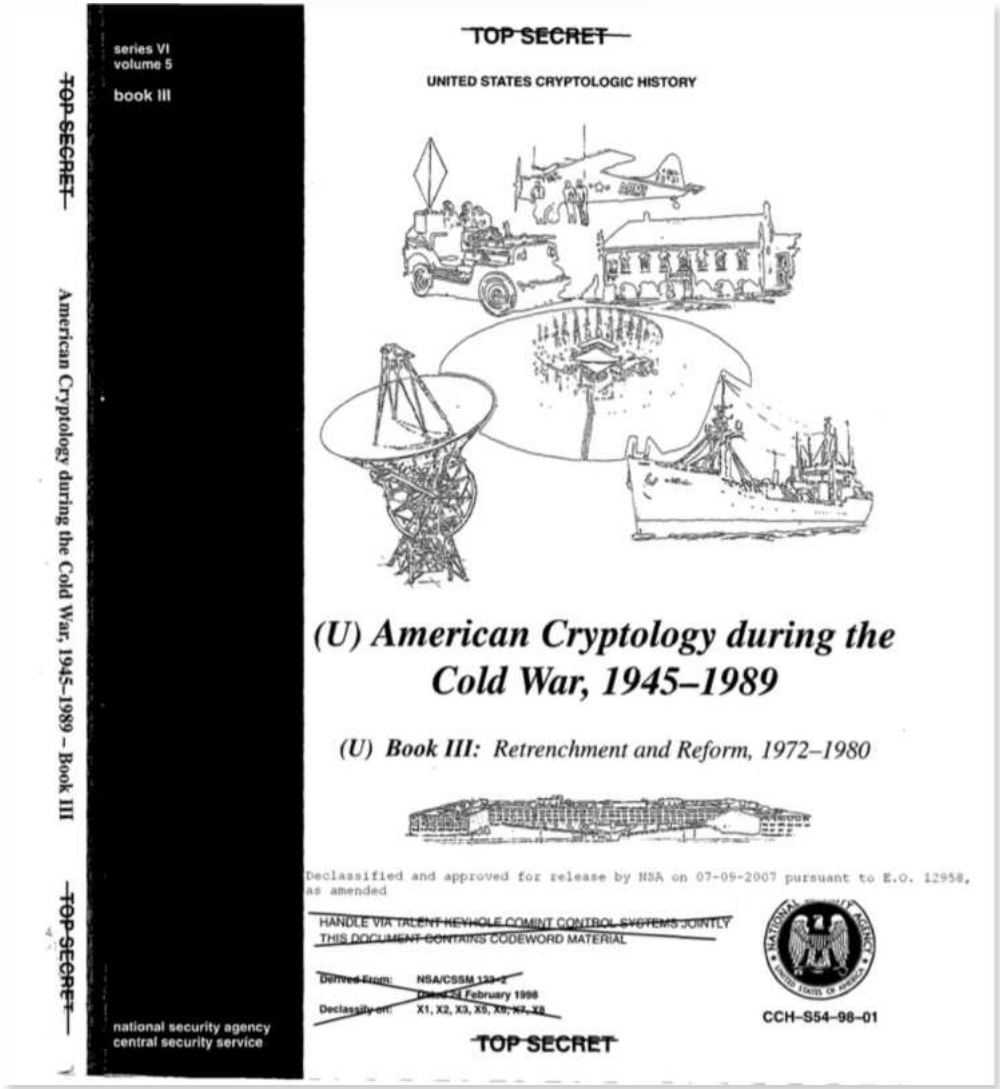
(b) Change in Key	
Round	Number of bits that differ
0	0
1	2
2	14
3	28
4	32
5	30
6	32
7	35
8	34
9	40
10	38
11	31
12	33
13	28
14	26
15	34
16	35

Osservazioni sul DES

- Molti aspetti circondati da misteri
  - Si ritiene che autorità governative possano avere forzato alcuni aspetti per poter mantenere il controllo
- Esempi
  - Restrizione ad un massimo di 56 bit per la chiave
    - Poter decifrare messaggi
  - Costruzioni delle S-box
    - Nascondono qualcosa?
  - Gli shift
    - Perché a volte 1 e a volte 2?

Coinvolgimento NSA nella progettazione del DES

Declassificato 2009



Coinvolgimento NSA nella progettazione del DES

In 1973 NBS solicited private industry for a data encryption standard (DES). The first offerings were disappointing, so NSA began working on its own algorithm. Then Howard Rosenblum, deputy director for research and engineering, discovered that Walter Tuchman of IBM was working on a modification to Lucifer for general use. NSA gave Tuchman a clearance and brought him in to work jointly with the Agency on his Lucifer modification."

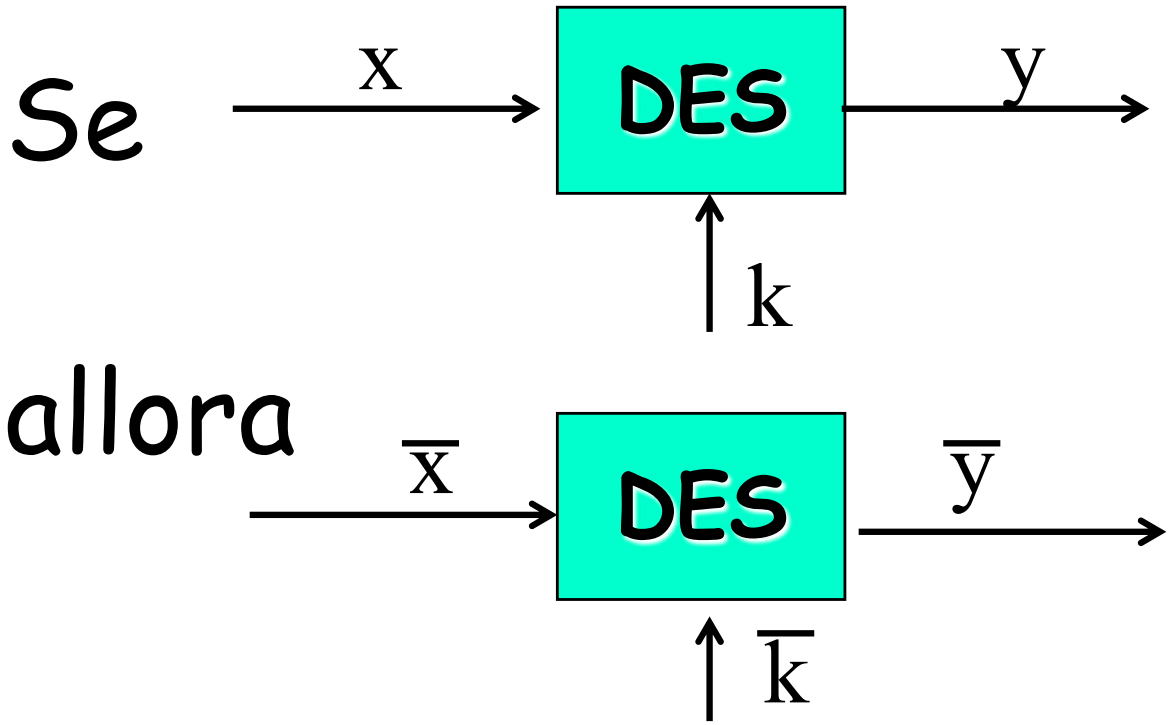
...

NSA worked closely with IBM to strengthen the algorithm against all except brute force attacks and to strengthen substitution tables, called S-boxes. Conversely, NSA tried to convince IBM to reduce the length of the key from 64 to 48 bits. Ultimately they compromised on a 56-bit key.

Attacchi al DES

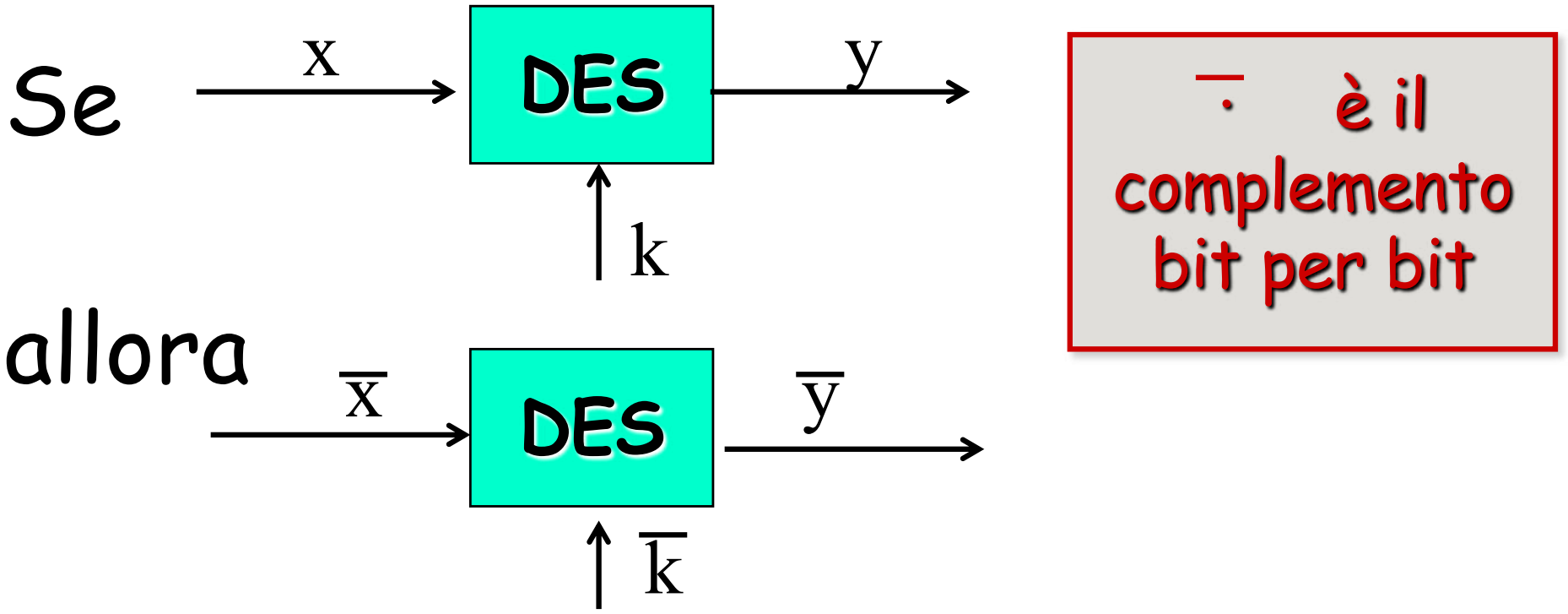
- Forza bruta (ricerca esaustiva)
  - Esaminare 2^{56} chiavi
- Anni '70: improponibile
  - Ma forse non per tutti
- Oggi: fattibile

Proprietà del complemento



$\overline{\cdot}$ è il complemento bit per bit

Proprietà del complemento



Esercizio: provare la proprietà del complemento

Proprietà del complemento

Può essere usata per migliorare il

running time della ricerca esaustiva?

- Attacco known plaintext: NO
- Attacco chosen plaintext: SI
  - Idea:
    - Scegliamo un opportuno testo in chiaro ed otteniamo il testo cifrato
    - Calcoliamo una singola cifratura con una possibile chiave
    - Se non è la chiave, non lo sarà nemmeno il valore complementato
  - Eliminiamo due chiavi candidate mediante una singola cifratura
  - Il running time della ricerca esaustiva migliora di un fattore $\frac{1}{2}$

Proprietà del complemento

Può essere usata per migliorare il

running time della ricerca esaustiva?

- Attacco known plaintext: NO
- Attacco chosen plaintext: SI
  - Scegliamo un plaintext x
  - Otteniamo due testi cifrati (x, y_1) e $(\overline{x}, y_2)$
  - Per un valore h, calcoliamo DES(h, x)
    - se $DES(h, x) \neq y_1$, allora chiave$\neq$h
    - se $DES(h, x) \neq \overline{y}_2$ allora $DES(\overline{h}, \overline{x}) \neq y_2$ e quindi chiave$\neq\overline{h}$

Ricerca esaustiva

- Numero chiavi DES = $2^{56} \approx 7{,}2056 \cdot 10^{16}$ = 72 milioni di miliardi di combinazioni distinte
- Considerando un computer che svolge:
  - 1 cifratura DES al microsecondo, ci vogliono circa 1142 anni per provare la metà dello spazio delle chiavi $2^{55} \approx 3{,}6 \cdot 10^{16}$ chiavi
  - 1 milione di cifrature DES al microsecondo, ci vogliono solo circa 10 ore

Ricerca esaustiva

Con macchine parallele è possibile diminuire il tempo richiesto dalla ricerca esaustiva

- 1977: ipotesi di macchina in grado di rompere il DES in un giorno
  - costo: 20 milioni di dollari
  - 10^6 chip, in grado di testare 10^6 chiavi al secondo
- 1993: ipotesi di macchina in grado di rompere il DES in tre ore e mezza
  - Costo: 1 milione di dollari
  - 10 macchine in parallelo, ciascuna con 5.760 chip

DES challenges

- Proposte da RSA Data Security:
  - $10000 al primo che rompe la *challenge*
- Messaggio cifrato con DES
  - Messaggio in lingua inglese
  - Ogni lettera codificata in ASCII
  - La stringa dei bit divisa in blocchi di 64 bit
  - Blocchi cifrati con la stessa chiave segreta

DES challenges

Giugno 1997: dopo 96 giorni, testato 24% delle 2^{56} chiavi,

- Progetto DESCHALL (abbreviazione per DES Challenge)
- Rocke Verser, programmatore, Colorado) scrisse e distribuì un client di ricerca
- Server: 486-based PS/2 con memoria 56MB
- Client scritti per diversi pc (anche a 64 bit)
- Scaricati da circa 78.000 differenti indirizzi IP
- Chiave trovata da Michael K. Sanders (Pentium 90 MHz, RAM 16M, FreeBSD 2.2.1, velocità 250.000 chiavi/secondo)
- Chiave 8558891AB0C851B6
- Messaggio: "Strong cryptography makes the world a safer place"
- Premio di $10000: $6000 a Verser e $4000 a Sanders

http://gilchrist.ca/jeff/distrib-des.html
https://web.archive.org/web/20071201071615/http://home.earthlink.net:80/~rcv007/despr4.htm

DES challenges

- Proposte da RSA Data Security: $10.000 al primo che rompe la *challenge* se rotta entro il 25% del miglior tempo precedente

- **Inizio 1998:** 41 giorni
  - distributed.net
  - Messaggio: "The secret message is: Many hands make light work"
- **Luglio 1998**: 56 ore, Deep Crack, EFF, 250.000 dollari
  - "There are many people who will not believe a truth until they can see it with their own eyes. Showing them a physical machine that can crack DES in a few days is the only way to convince some people that they really cannot trust their security to DES."
  - Messaggio: "The secret message is: It's time for those 128-, 192-, and 256-bit keys."
- **Gennaio 1999**: 22 ore 15 minuti testando 245 miliardi di chiavi al secondo, distributed.net con 100.000 computer e EFF
  - Messaggio: "See you in Rome (second AES Conference, March 22-23, 1999)"

distributed.net

- Organizzazione no profit, fondata nel febbraio 1997
- Progetto di calcolo distribuito per risolvere problemi
  - Usando tempo CPU o GPU, altrimenti non utilizzato

Progetti:

Project	Description	Status
RC5-72	RSA Labs' 72bit RC5 Encryption Challenge	active
OGR-24	Optimal Golomb Rulers	completed
OGR-25	Optimal Golomb Rulers	completed
OGR-26	Optimal Golomb Rulers	completed
OGR-27	Optimal Golomb Rulers	completed
OGR-28	Optimal Golomb Rulers	active
RC5-56	RSA Labs' 56bit RC5 Encryption Challenge	completed
RC5-64	RSA Labs' 64bit RC5 Encryption Challenge	completed
RC5-64 (all)	RC5-64, plus the work done after the key was found	completed

23 marzo 2020

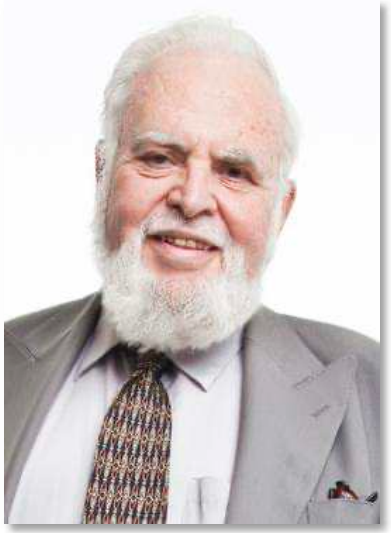
Aggregate Statistics

Total Blocks to Search:	1,099,511,627,776
Total Blocks Tested:	71,160,873,384
Overall Rate:	130 Blocks/sec
Total Keys to Search:	4,722,366,482,869,645,213,696
Total Keys Tested:	305,633,623,939,076,849,664
Overall Rate:	559,630,786,742 Keys/sec
Percent Complete:	6.472%
Time Working:	6,321 days

Progress Meters

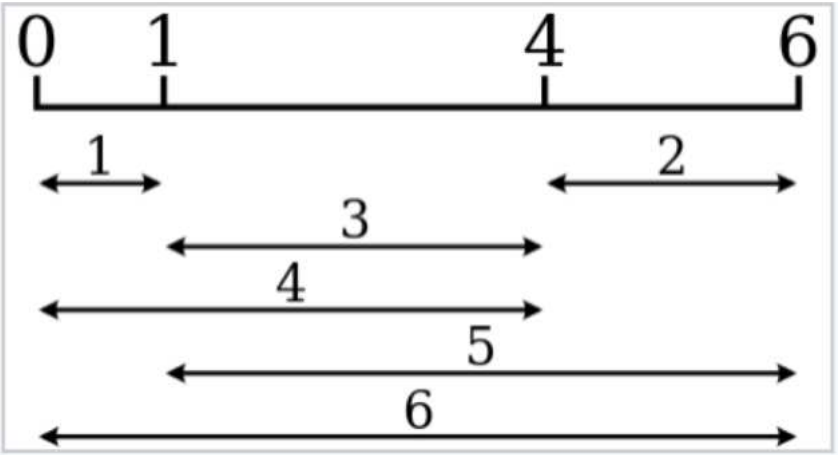
distributed.net

Solomon Wolf Golomb



Regolo di Golomb, è un insieme
di tacche poste a posizioni intere su un regolo, tale che non ci sia alcuna coppia di tacche poste alla stessa distanza

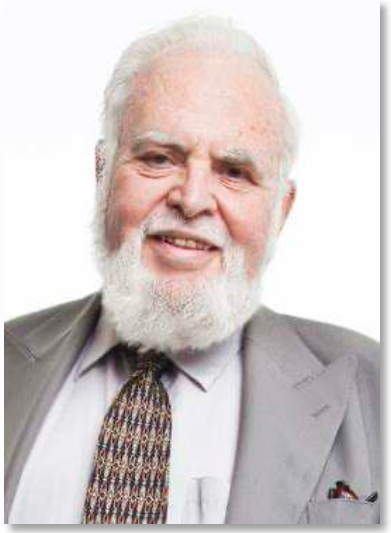
ottimale = più corto



OGR-4

distributed.net

Solomon Wolf Golomb



Regolo di Golomb, è un insieme
di tacche poste a posizioni intere su un regolo, tale che non ci sia alcuna coppia di tacche poste alla stessa distanza

ottimale = più corto

OGR-27
0 3 15 41 66 95 97 106 142 152 220 221
225 242 295 330 338 354 382 388 402
415 486 504 523 546 553
Trovato il 25 febbraio 2014
dopo 1.822 giorni con 19.919 partecipanti

distributed.net

Progetti:

Project	Description	Status
RC5-72	RSA Labs' 72bit RC5 Encryption Challenge	active
OGR-24	Optimal Golomb Rulers	completed
OGR-25	Optimal Golomb Rulers	completed
OGR-26	Optimal Golomb Rulers	completed
OGR-27	Optimal Golomb Rulers	completed
OGR-28	Optimal Golomb Rulers	active
RC5-56	RSA Labs' 56bit RC5 Encryption Challenge	completed
RC5-64	RSA Labs' 64bit RC5 Encryption Challenge	completed
RC5-64 (all)	RC5-64, plus the work done after the key was found	completed

Aggregate Statistics

17 marzo 2018

Total Blocks to Search:	1,099,511,627,776
Total Blocks Tested:	55,842,215,909
Overall Rate:	116 Blocks/sec
Total Keys to Search:	4,722,366,482,869,645,213,696
Total Keys Tested:	239,840,491,065,325,912,064
Overall Rate:	497,211,465,082 Keys/sec
Percent Complete:	5.079%
Time Working:	5,583 days

Progress Meters

Regolo di Golomb, è un insieme di tacche poste a posizioni intere su un regolo, tale che non ci sia alcuna coppia di tacche poste alla stessa distanza

Deep Crack

"Deep" si riferisce a Deep Blue, computer di IBM

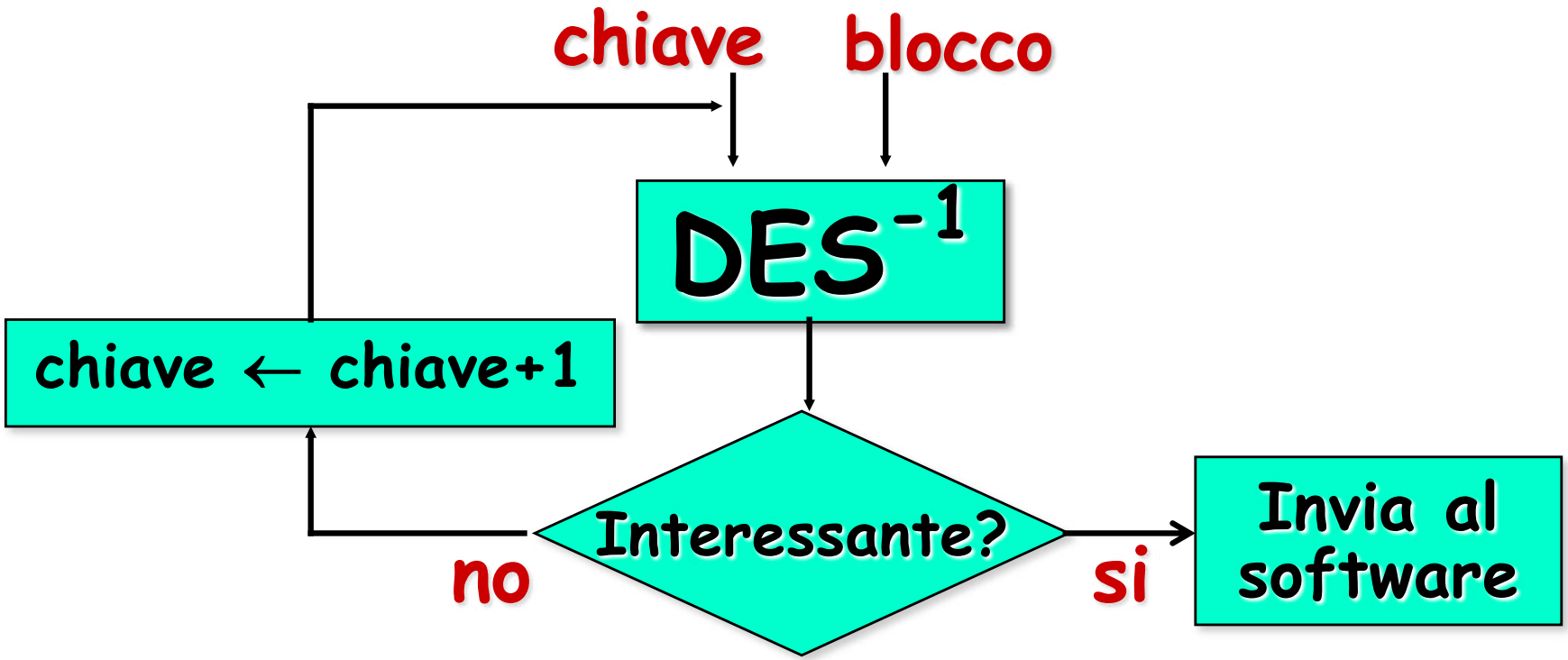
- fu il primo a vincere a scacchi contro il campione mondiale, Garry Kasparov
  - una singola partita, 1996
  - un match, 1997

Deep Crack

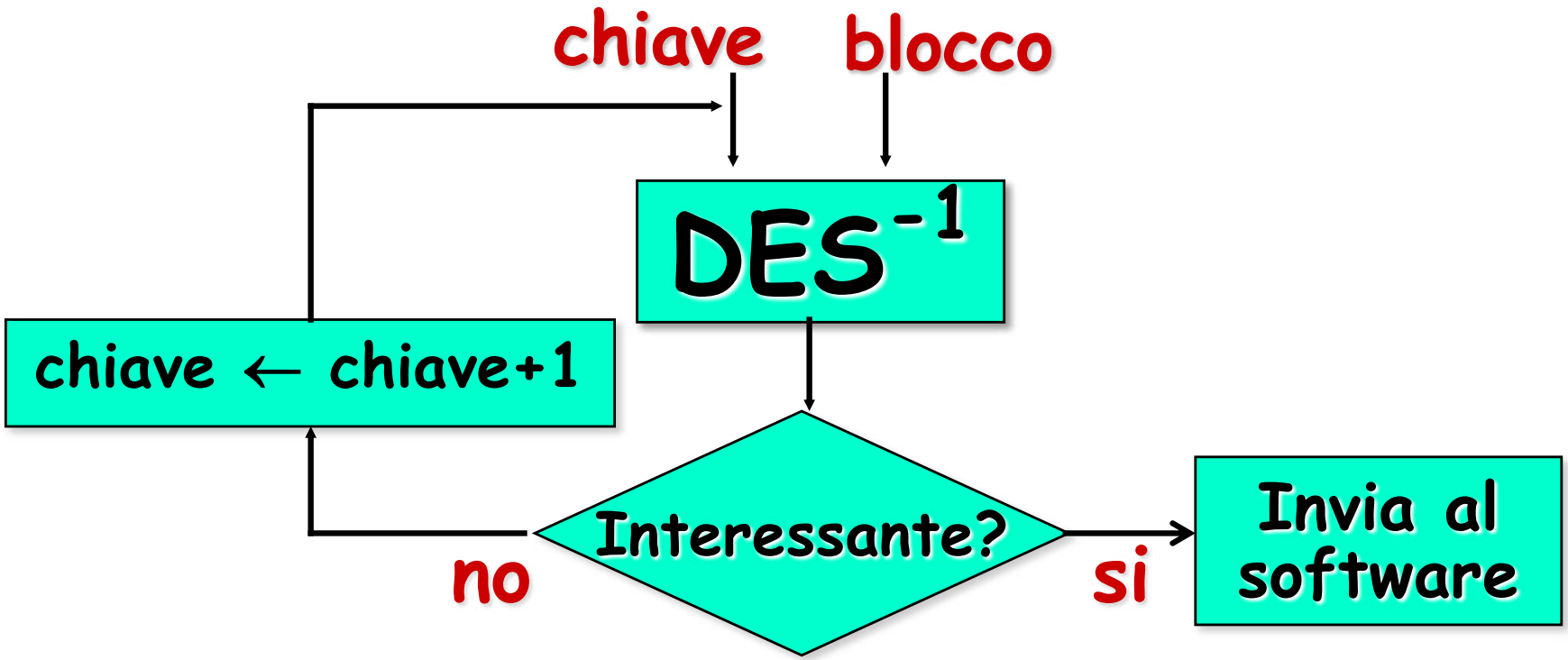
Coordinamento di un singolo PC

- Assegna intervalli di chiavi alle singole unità
- Riceve insiemi di possibili chiavi dalle singole unità
- Seleziona la chiave tra tutti i candidati

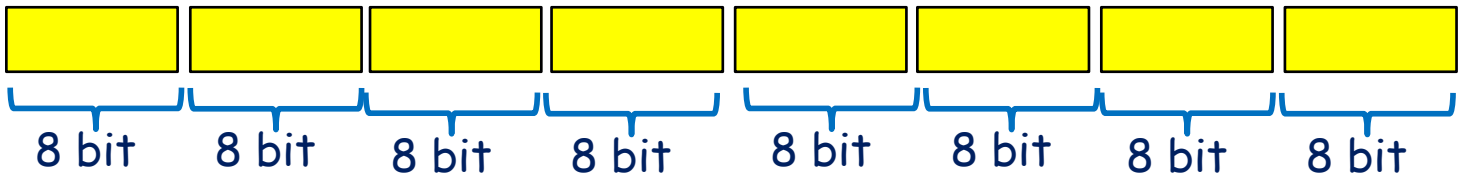
Deep Crack: Unità di ricerca



Deep Crack: Unità di ricerca



Interessante se un blocco decifrato è formato da tutti caratteri ASCII



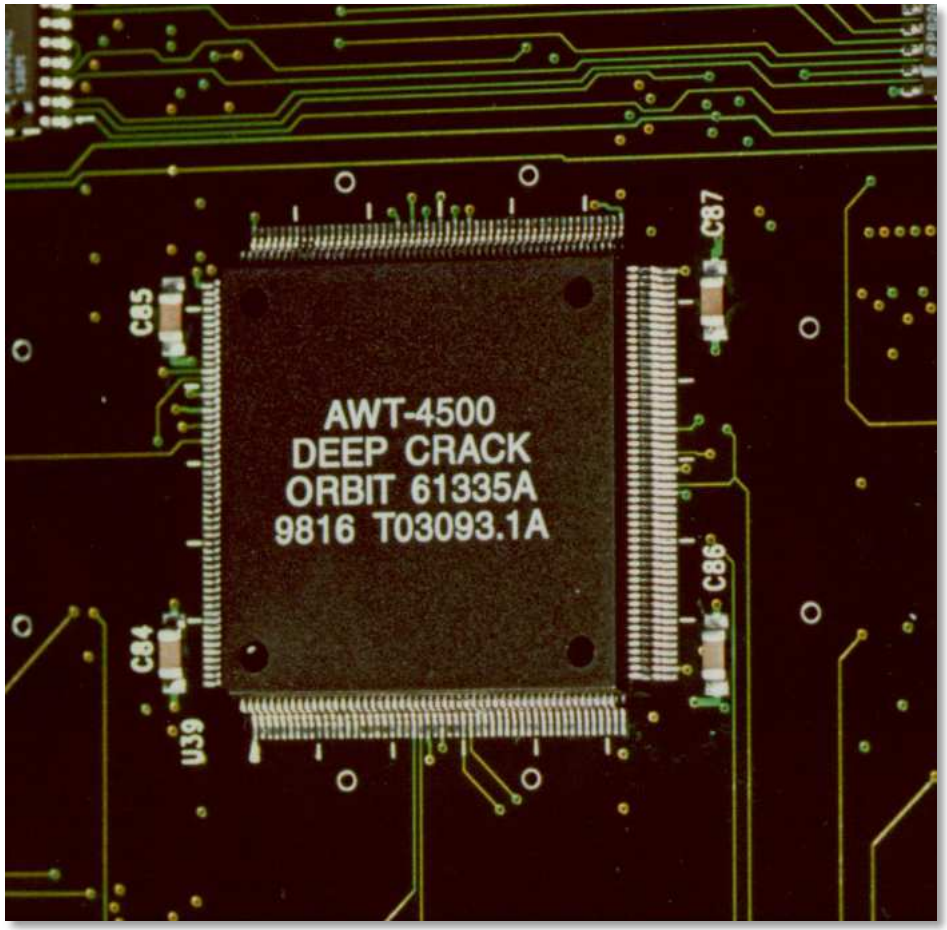
Deep Crack: Unità di ricerca

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

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

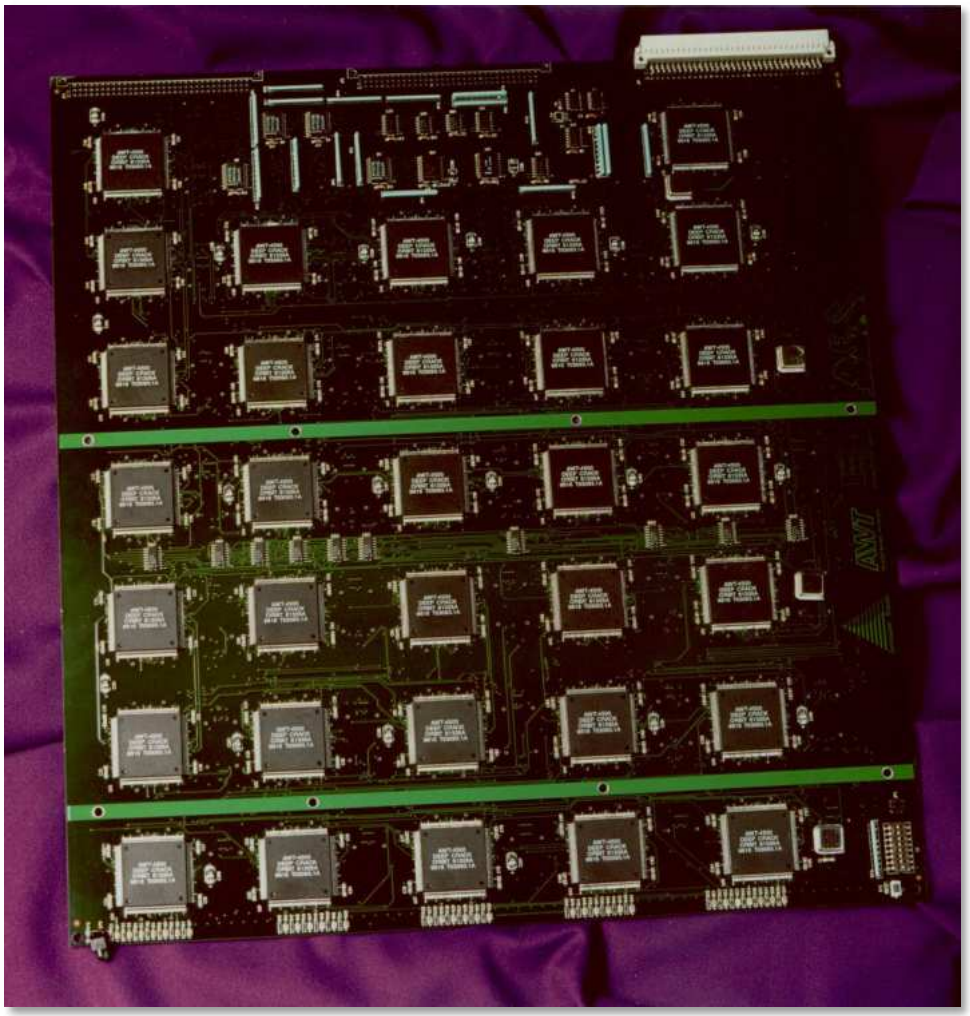
Chip

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



Board

- 64 processori
- 32 per faccia
- 40 cm X 40 cm
- Prova 64·60.000.000 = 3.840.000.000 chiavi al secondo
- Prova tutte le chiavi in ≈218 giorni



Chassis

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

EFF DES Cracker

Prestazioni

Device	Quanti nella prossima device	Chiavi/secondo	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

EFF

- International non-profit digital rights group based in the United States
- Fondata 6 luglio 1990
- https://www.eff.org

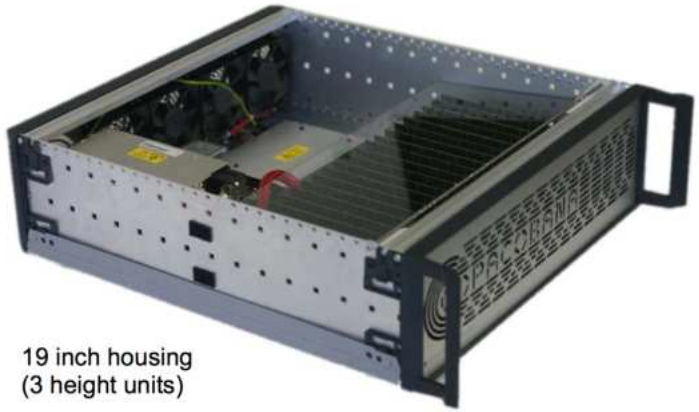
COPACOBANA

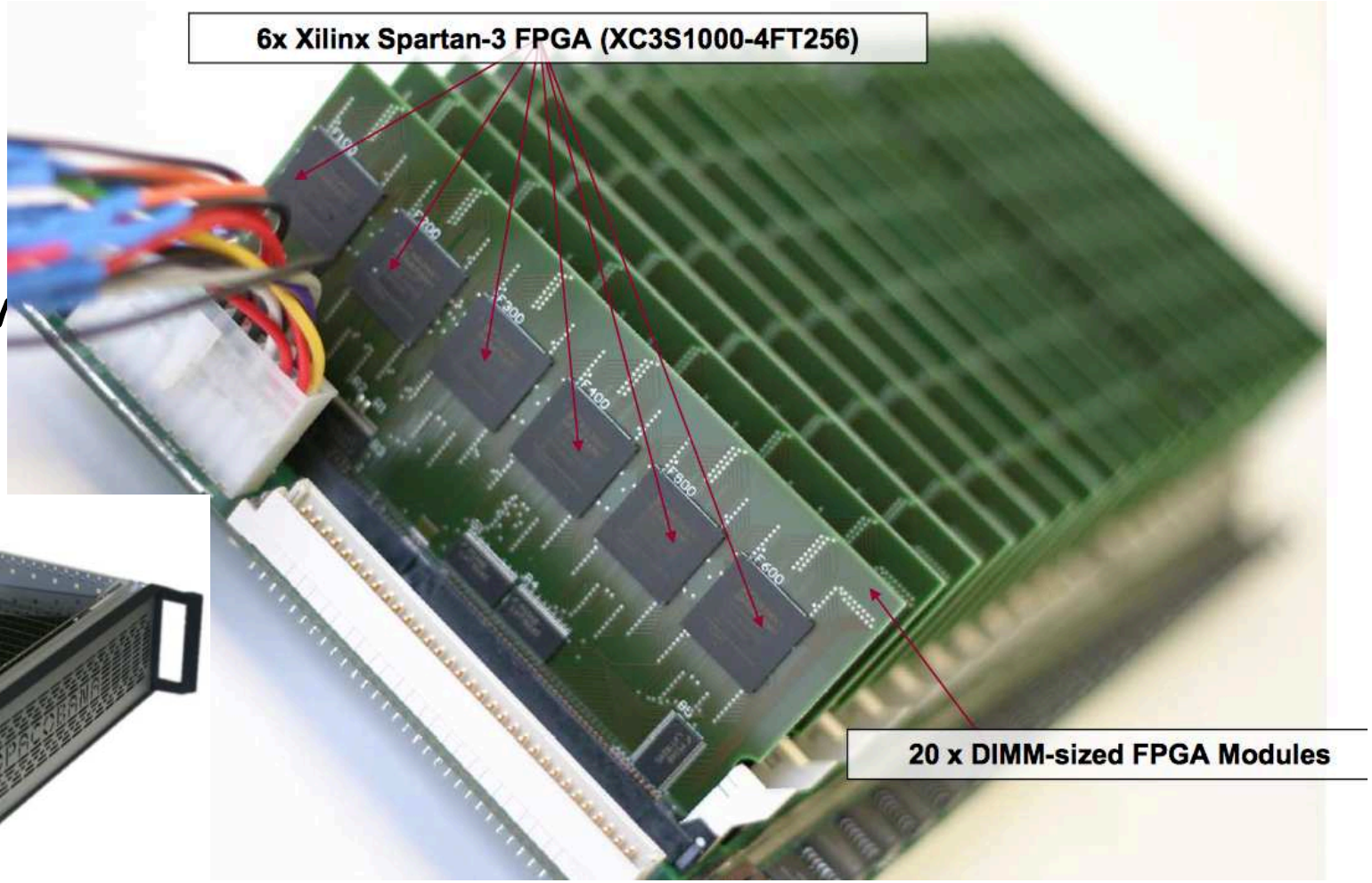
- *Cost-Optimized Parallel COde breaker*
- Costruito da team Università di Bochum e Kiel, Germania
- Utilizzo in parallelo di dispositivi Field Programmable Gate Array (FPGA) riprogrammabili e reperibili normalmente in commercio
  - Possibile riprogrammarne la logica, quindi possibile utilizzare l'apparecchiatura per altri cifrari
  - Costo contenuto
- Diversamente da hardware dedicato solo alla analisi di uno specifico cifrario
  - Come le Bombe per Enigma
  - Come Deep Crack per il DES

COPACOBANA

- *Cost-Optimized Parallel COde breaker*
- Costruito da team Università di Bochum e Kiel, Germania
- Utilizzo in parallelo di dispositivi Field Programmable Gate Array (FPGA) riprogrammabili e reperibili normalmente in commercio
  - Possibile riprogrammarne la logica, quindi possibile utilizzare l'apparecchiatura per altri cifrari
  - Costo contenuto
- Marzo 2007: 120 FPGA di tipo XILINX Spartan3-1000
  - € 8.000, tempo medio per rompere DES di 6,4 giorni
- Maggio 2008: COPACOBANA RIVYERA con 128 Virtex-4 SX 35 FPGA
  - tempo medio per rompere DES meno di un giorno

- *Cost-Opti*
- Costruito
  Germania





- Marzo 2007: 120 FPGA di tipo XILINX Spartan3-1000
  - € 8.000, tempo medio per rompere DES di 6,4 giorni
- Maggio 2008: COPACOBANA RIVYERA con 128 Virtex-4 SX 35 FPGA
  - tempo medio per rompere DES meno di un giorno

COPACOBANA

- *Cost-Optimized Parallel COde breaker*
- Costruito da team Università di Bochum e Kiel, Germania
-

Original:

6xSpartan-3 XC3S1000

-
- Maggio 2008: COPACOBANA RIVYERA con 128 Virtex-4 SX 35 FPGA
  - tempo medio per rompere DES meno di un giorno

Attacchi sofisticati al DES

- Crittoanalisi differenziale
  (Eli Biham e Adi Shamir, 1990)

- Crittoanalisi lineare
  (Mitsuru Matsui, 1993)

Crittoanalisi differenziale

- Informalmente, analizza come le differenze in input diventano differenze nell'output
- La differenza in genere è lo XOR
- Analisi dei differenziali per ogni S-box (ΔX, ΔY), dove $\Delta Y = S(X \oplus \Delta X) \oplus S(X)$
- Efficace con un numero basso di iterazioni
  - chosen ciphertext attack
    - Con 8 iterazioni, necessarie 2^{14} coppie (plaintext, ciphertext) di testi scelti
    - Con 16 iterazioni, necessarie 2^{47} coppie (plaintext, ciphertext) di testi scelti

DES è resistente alla crittoanalisi differenziale,
ma con piccole modifiche diventa vulnerabile!

Crittoanalisi differenziale

Attacco già noto in fase di progetto del DES

Differential cryptanalysis was well known, however, to the IBM team that designed DES, as early as 1974. Knowledge of this technique, and the necessity to strengthen DES against attacks using it, played a large part in the design of the S-boxes and the permutation *P*.

After discussions with NSA, it was decided that disclosure of the design considerations would reveal the technique of differential cryptanalysis, a powerful technique that can be used against many ciphers. This in turn would weaken the competitive advantage the United States enjoyed over other countries in the field of cryptography.

The Data Encryption Standard (DES) and its strength against attacks

by D. Coppersmith

The Data Encryption Standard (DES) was developed by an IBM team around 1974 and adopted as a national standard in 1977. Since that time, many cryptanalysts have attempted to find shortcuts for breaking the system. In this paper, we examine one such attempt, the method of differential cryptanalysis, published by Biham and Shamir. We show some of the safeguards against differential cryptanalysis that were built into the system from the beginning, with the result that more than 10^{15} bytes of chosen plaintext are required for this attack to succeed.

Introduction

Cryptography has long been in use by governments, particularly in the realms of military and diplomatic communication. It is hard to imagine military communication without cryptography; cryptanalysis, or secretly deciphering the opponent's messages, is perhaps of even greater value. Much has been written about cryptography in the military; see reference [1] for example.

During the early 1970s, it became apparent that the commercial sector also has a legitimate need for cryptography. Corporate secrets must be transmitted between distant sites, without the possibility of eavesdropping by industrial spies. Personal data on databases need to be protected against espionage and alteration.

A familiar example is the communication between an automatic teller machine (ATM) and a central computer. The user inserts a magnetic card and types a few numbers. The ATM sends messages to the computer. The computer checks the account balance and returns a message authorizing the ATM to dispense funds. Obviously, if these messages are unprotected, a thief can tap the wires, find the message authorizing the dispensing of funds, and send multiple copies of that message to the ATM, thereby "cleaning out" the supply of cash from the ATM.

In the early 1970s, a banking customer asked IBM to develop a system for encrypting ATM data. With this problem as a starting point, a team was formed from

Disclaimer
The present author participated in the design and test of DES, particularly in the design of the S-boxes and in strengthening them against differential cryptanalysis. Naturally, this author has strong opinions about DES and its history. Any opinions in this paper are those of the author and are not necessarily shared by IBM.

Crittoanalisi lineare

Ricerca approssimazioni affini per il cifrario

- Recupera la chiave a partire da 2^{47} coppie (plaintext, ciphertext) di testi noti
- Implementato nel 1993: 10 giorni su 12 macchine

Modalità operative del DES

Come cifrare più di 64 bit?

Modalità operative del DES

Come cifrare più di 64 bit?

- Electronic codebook chaining (ECB)
- Cipher block chaining (CBC)
- Cipher feedback (CFB)
- Output feedback (OFB)
- Counter (CTR)

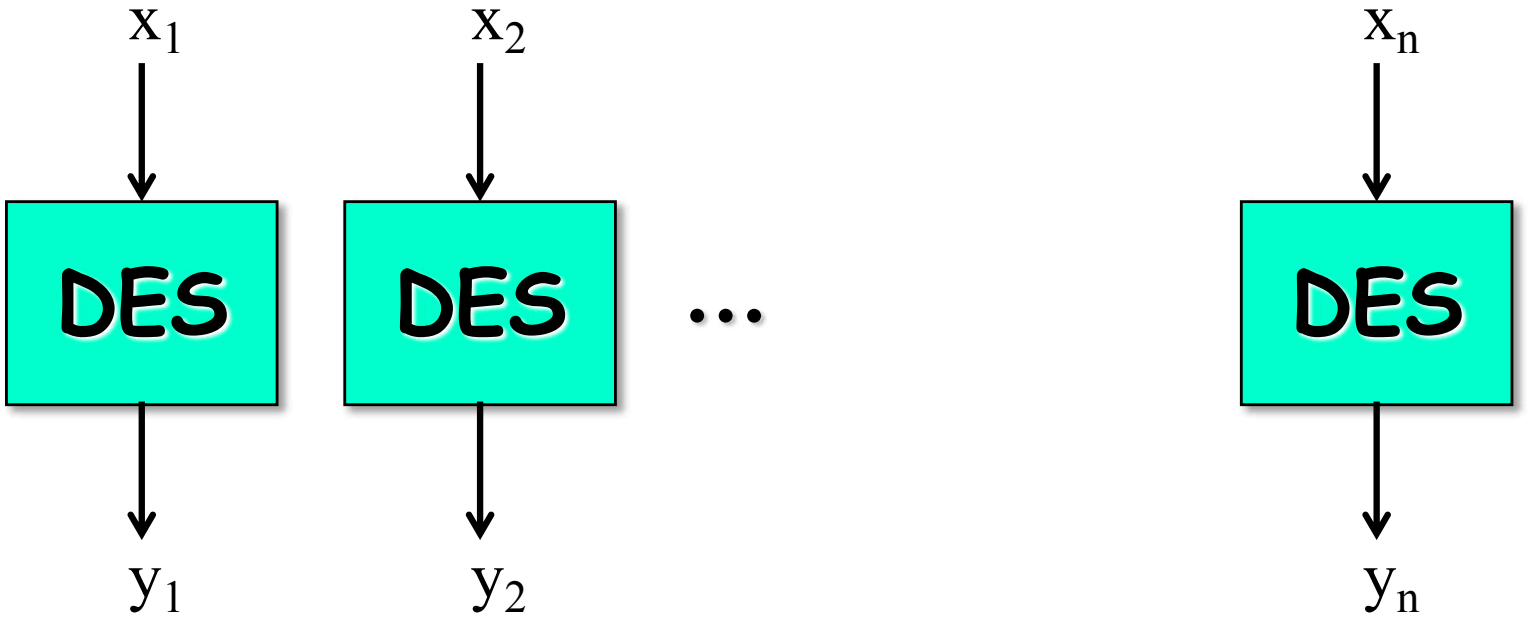
Modalità operative dei cifrari a blocchi

- NBS FIPS PUB 81, DES modes of operation, National Bureau of Standards, 1981
  - ECB – CBC – CFB - OFB
- NIST SP 800-38A, Recommendation for block cipher modes of operation, National Institute of Standards and Technology, 2001
  - Aggiornamento di ECB – CBC – CFB – OFB ed in più CTR
- NIST SP 800-38E, Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices, gennaio 2010
  - XTS-AES già nel IEEE Std 1619-2007
  - Lo vedremo in seguito (dopo l'AES).

NIST National Institute of Standards and Technology
FIPS Federal Information Processing Standards

Electronic codebook chaining (ECB)

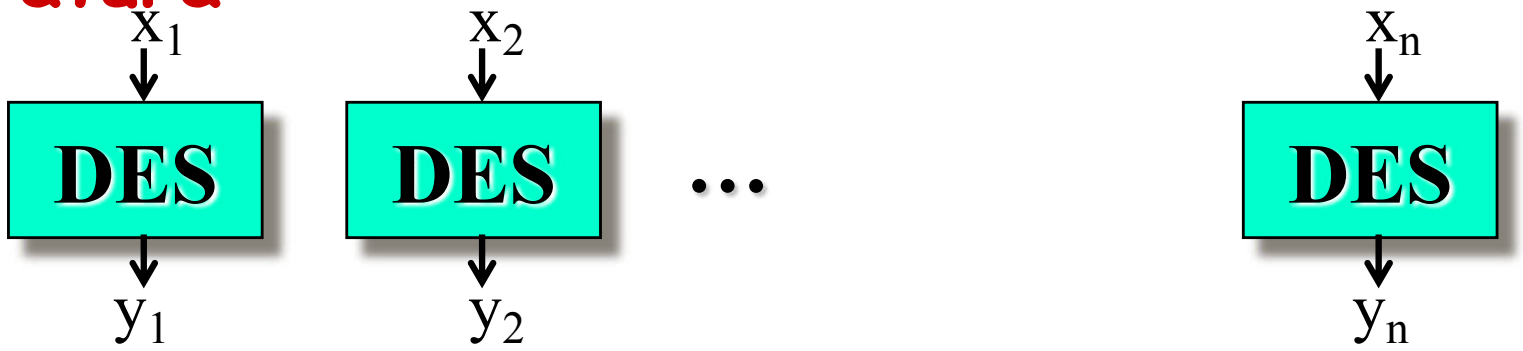
messaggio in chiaro $x = x_1 x_2 \ldots x_n$ (diviso in n blocchi di 64 bit)



messaggio cifrato $y = y_1 y_2 \ldots y_n$

Electronic codebook chaining (ECB)

cifratura



decifratura

Electronic codebook chaining (ECB)

- Cifratura parallelizzabile?
- Decifratura parallelizzabile?

Electronic codebook chaining (ECB)

- Cifratura parallelizzabile? SI
- Decifratura parallelizzabile? SI

Electronic codebook chaining (ECB)

- Se la lunghezza del messaggio non è multiplo di 64?
  - Possibile soluzione: Padding con 100…00
- L'ECB è il metodo più veloce
- Eventuali errori non si propagano

- Non c'è dipendenza tra i blocchi
  - Possibili attacchi di sostituzione

- Non sicuro per messaggi lunghi
  - Allo stesso blocco in chiaro corrisponde sempre lo stesso cifrato

Electronic codebook chaining (ECB)

- Se la lunghezza del messaggio non è multiplo di 64?
  - Possibile soluzione: Padding con 100…00
- L'ECB è il metodo più veloce
- Eventuali errori non si propagano

- Non c'è dipendenza tra i blocchi
  - Possibili attacchi di sostituzione

- Non sicuro per messaggi lunghi
  - Allo stesso blocco in chiaro corrisponde sempre lo stesso cifrato
  - Ad esempio, che succede se cifriamo i colori di una immagine bitmap con grandi aree di colore uniformi?

Electronic codebook chaining (ECB)

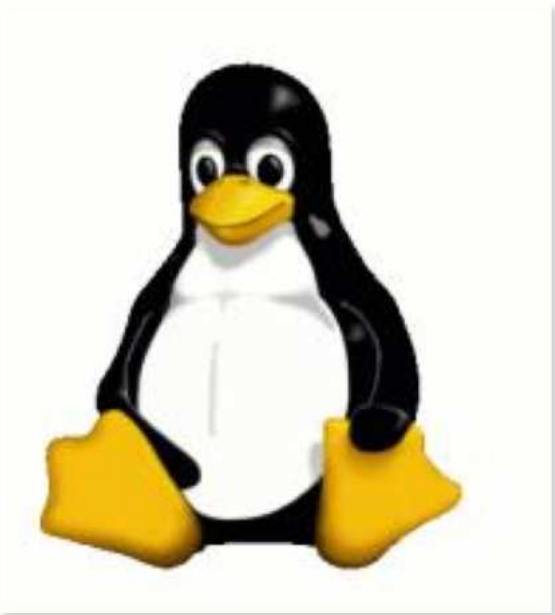
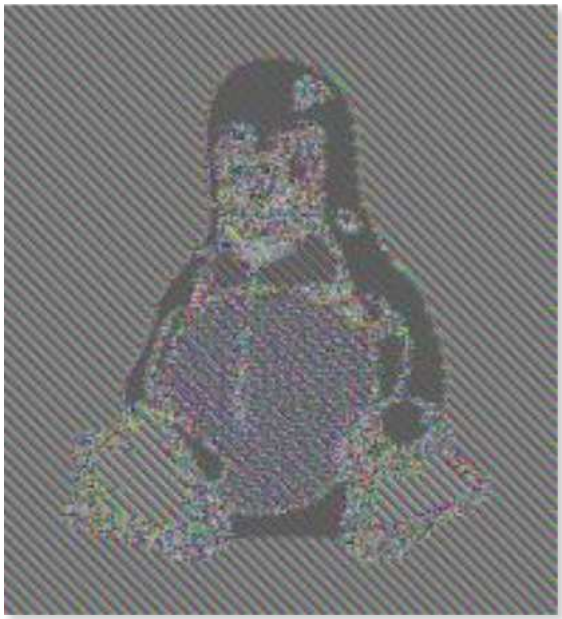
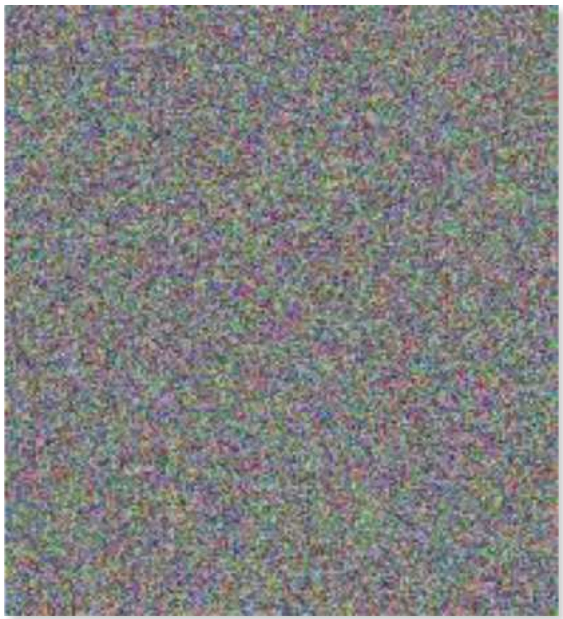


Immagine originale



Cifratura con ECB

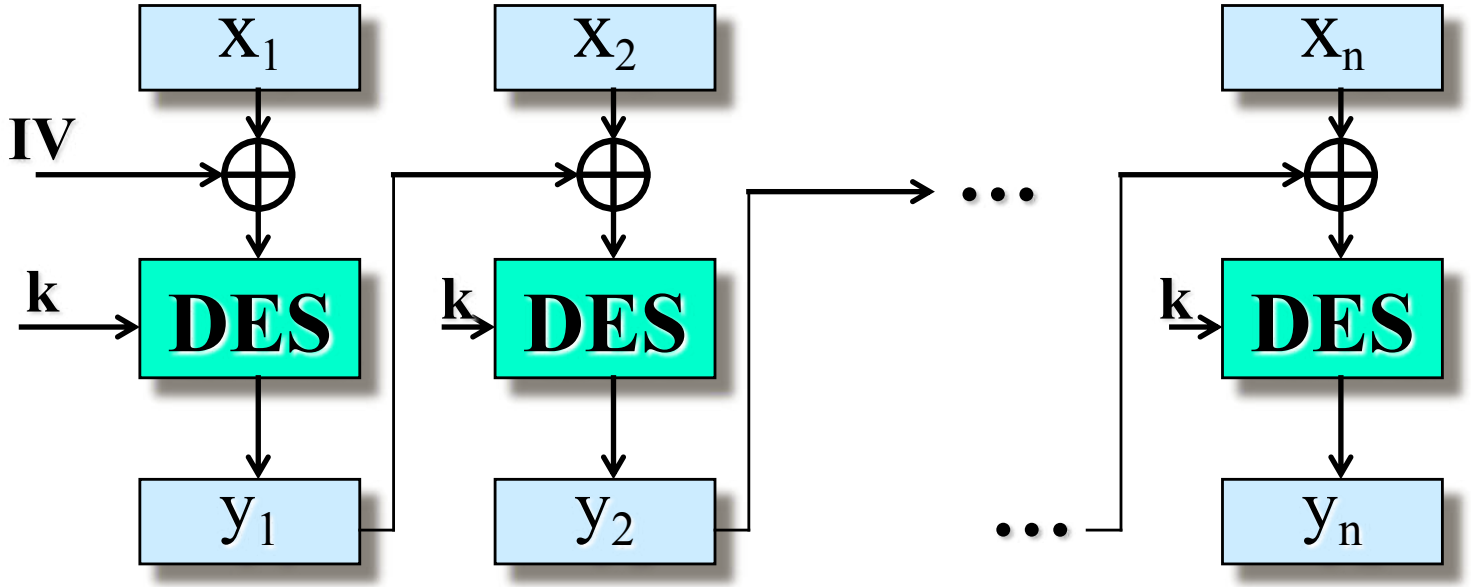


Cifratura con altre modalità

- Non sicuro per messaggi lunghi
  - Allo stesso blocco in chiaro corrisponde sempre lo stesso cifrato
  - Ad esempio, che succede se cifriamo i colori di una immagine bitmap con grandi aree di colore uniformi?

Cipher Block Chaining (CBC)

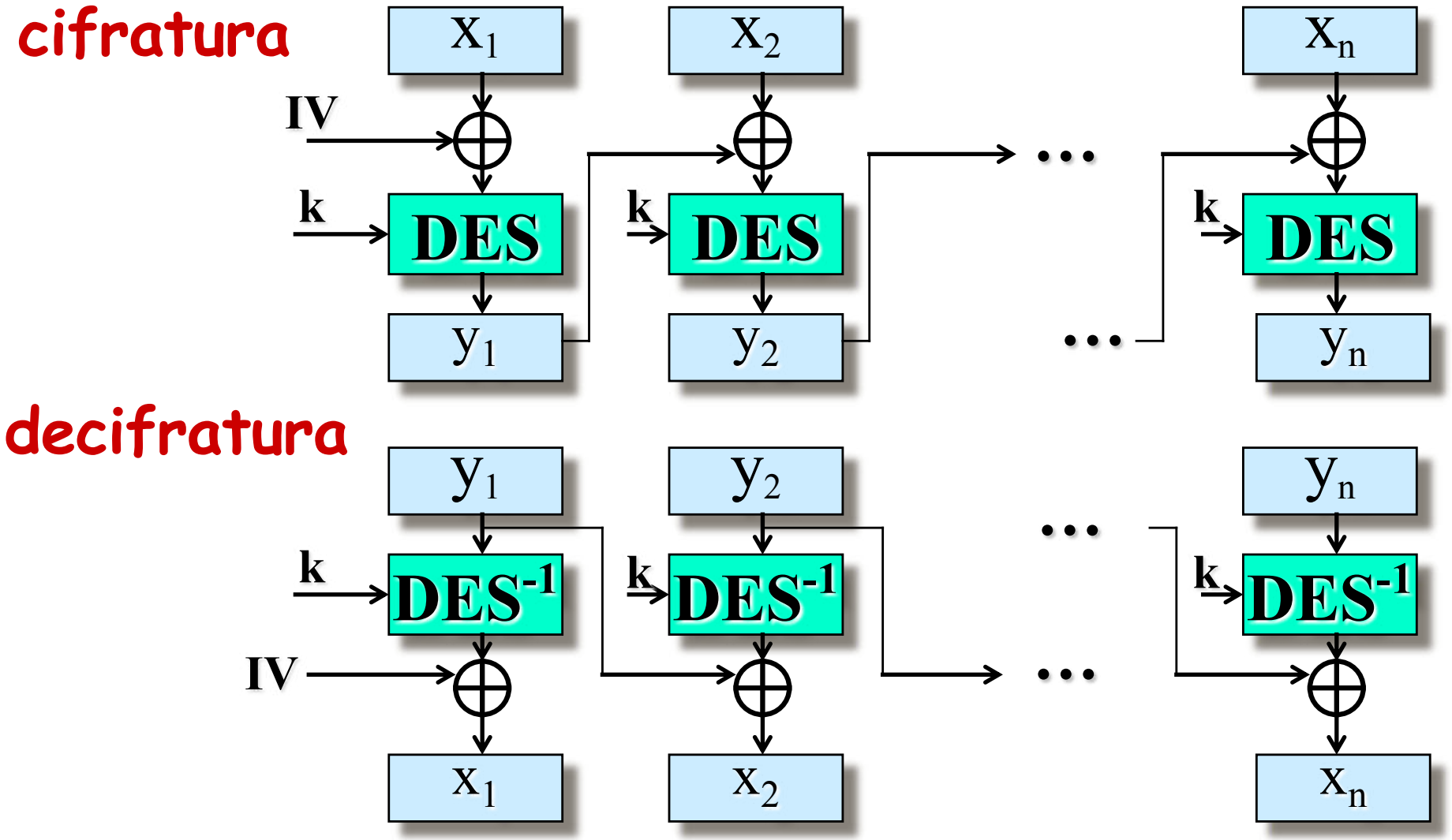
messaggio in chiaro $x=x_1x_2...x_n$ (diviso in n blocchi di 64 bit)



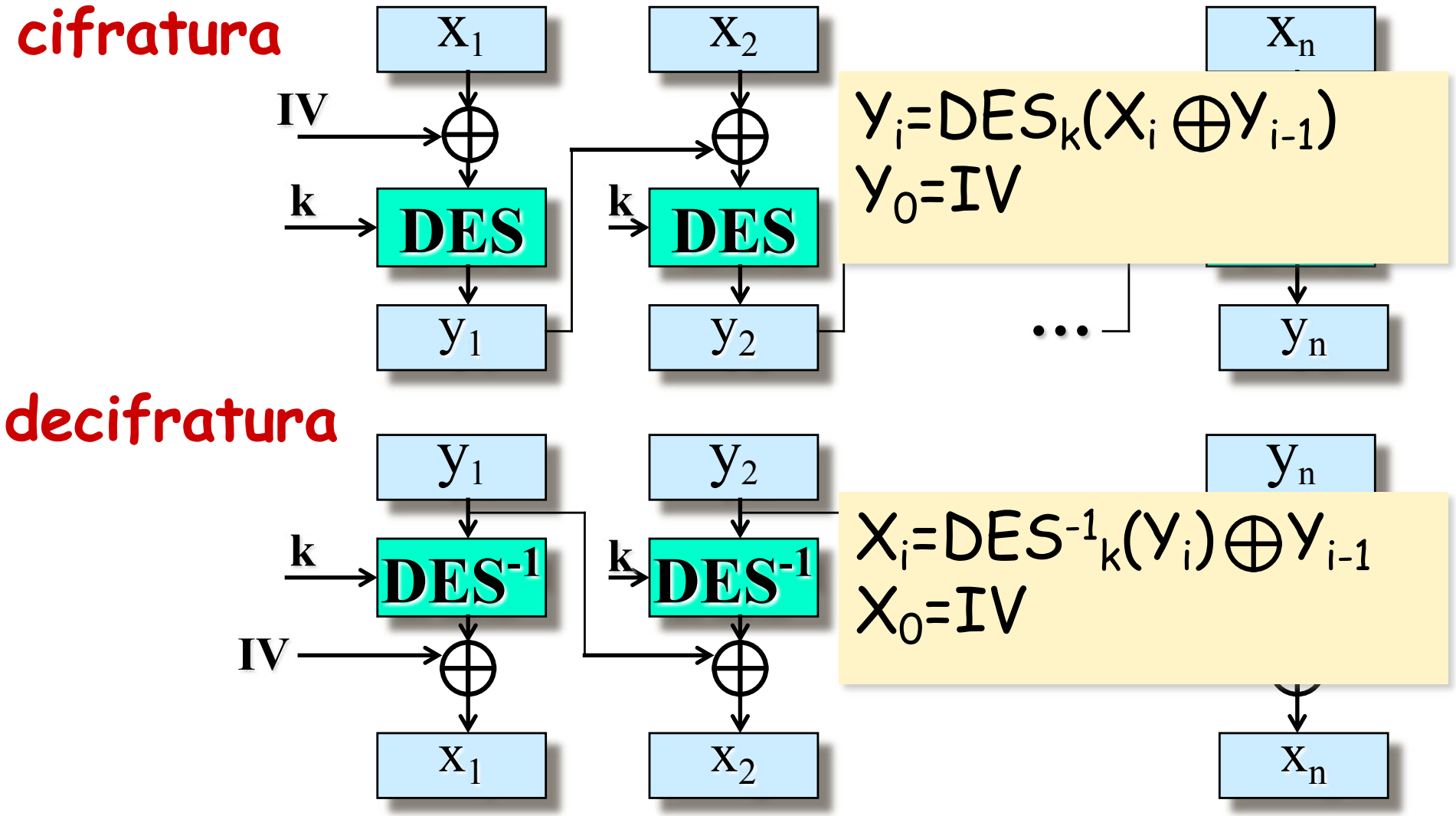
messaggio cifrato $y = y_1y_2...y_n$

vettore di inizializzazione IV di solito pubblico,
(potrebbe anche essere scelto a caso e tenuto nascosto)

Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC)

- Cifratura parallelizzabile?
- Decifratura parallelizzabile?
- Che succede se decifriamo con IV errato?

Cipher Block Chaining (CBC)

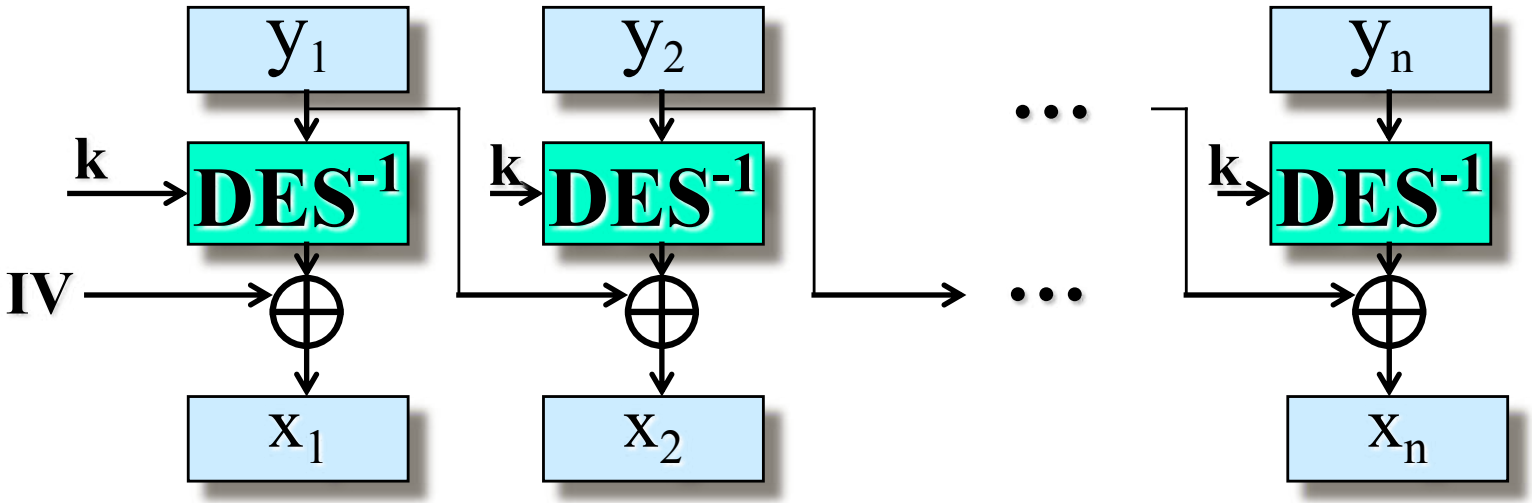
- Cifratura parallelizzabile? NO
- Decifratura parallelizzabile?
- Che succede se decifriamo con IV errato?

Cipher Block Chaining (CBC)

- Cifratura parallelizzabile? NO
- Decifratura parallelizzabile? SI
- Che succede se decifriamo con IV errato?

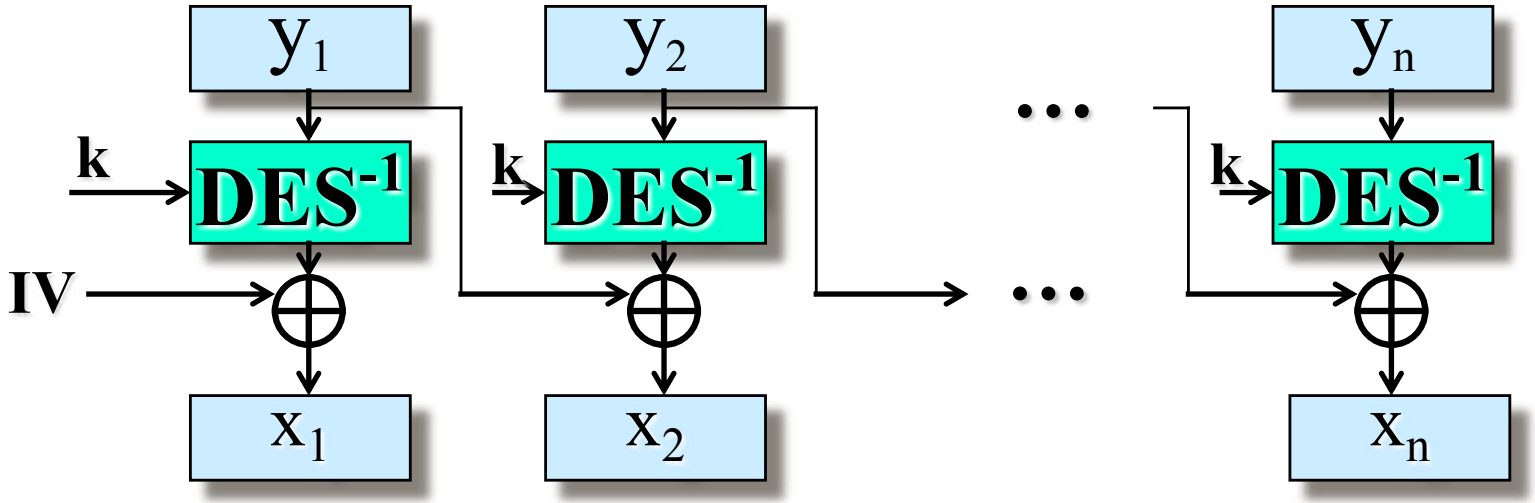
Cipher Block Chaining (CBC)

- Cifratura parallelizzabile? NO
- Decifratura parallelizzabile? SI
- Che succede se decifriamo con IV errato?



Cipher Block Chaining (CBC)

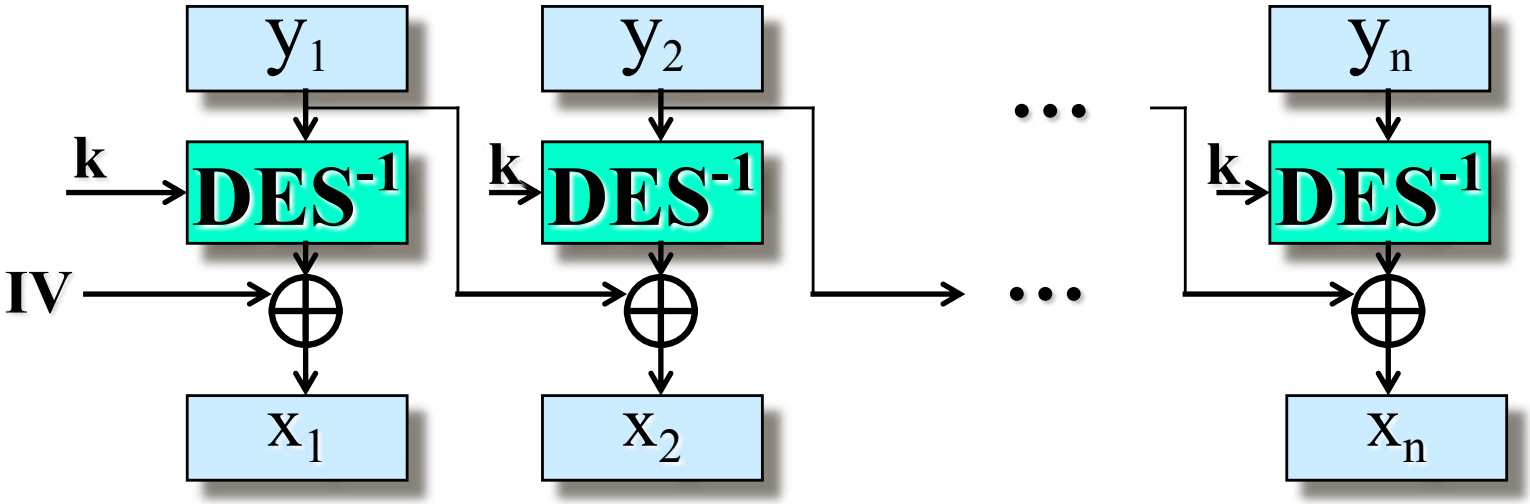
- Cifratura parallelizzabile? NO
- Decifratura parallelizzabile? SI
- Che succede se decifriamo con IV errato?



Solo il primo blocco è errato!

Cipher Block Chaining (CBC)

- *Explicit Initialization Vectors*
  - Aggiunta di un blocco casuale iniziale al testo in chiaro
  - Primo blocco del testo decifrato viene scartato
  - IV non deve essere comunicato
- The Transport Layer Security (TLS) Protocol Version 1.1 (Aprile 2006)



Cipher Block Chaining (CBC)

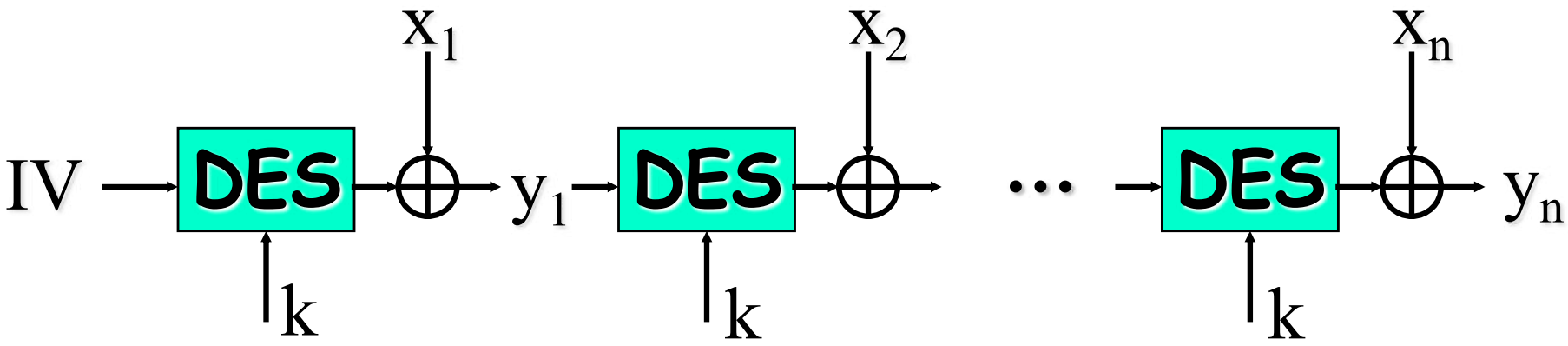
- Meno veloce dell'ECB
- Propagazione errori

- C'è dipendenza tra i blocchi
- Non possibili attacchi di sostituzione

Cipher feedback (CFB)

messaggio in chiaro $x=x_1x_2...x_n$

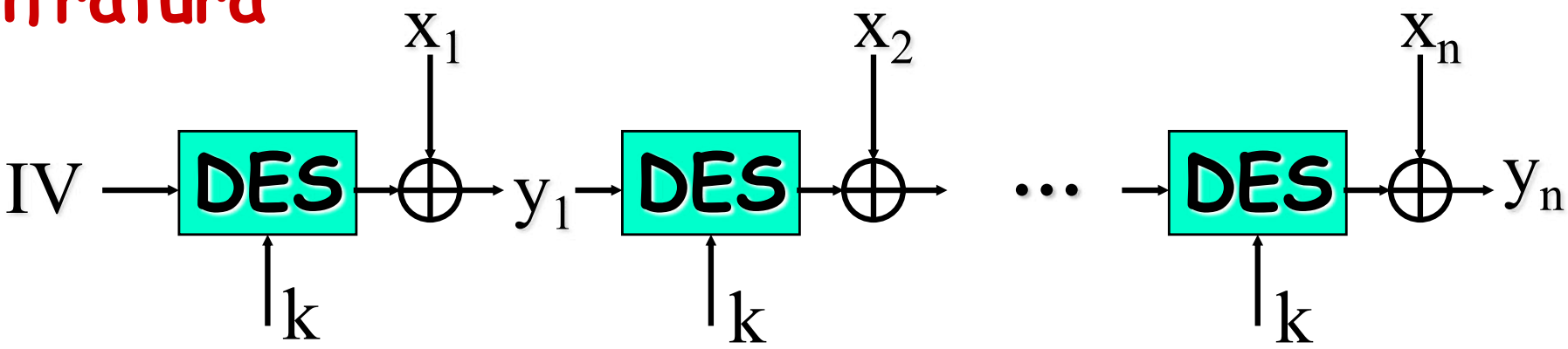
(diviso in n blocchi di 64 bit)



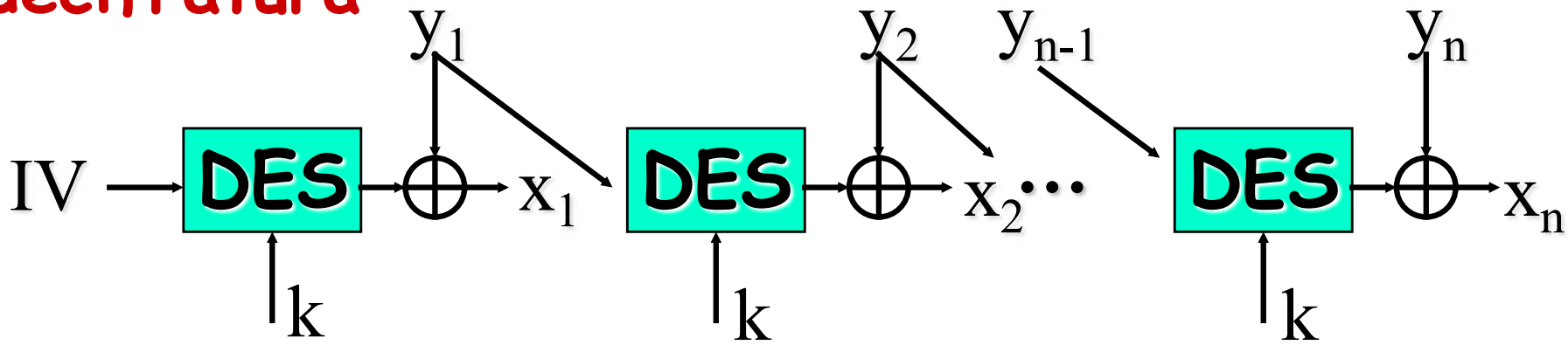
messaggio cifrato $y = y_1y_2...y_n$

Cipher feedback (CFB)

cifratura

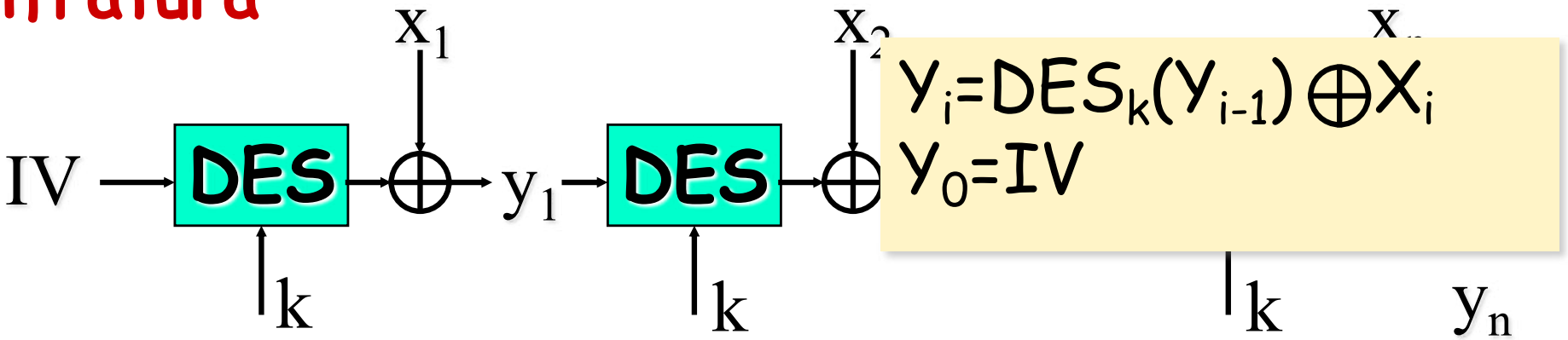


decifratura

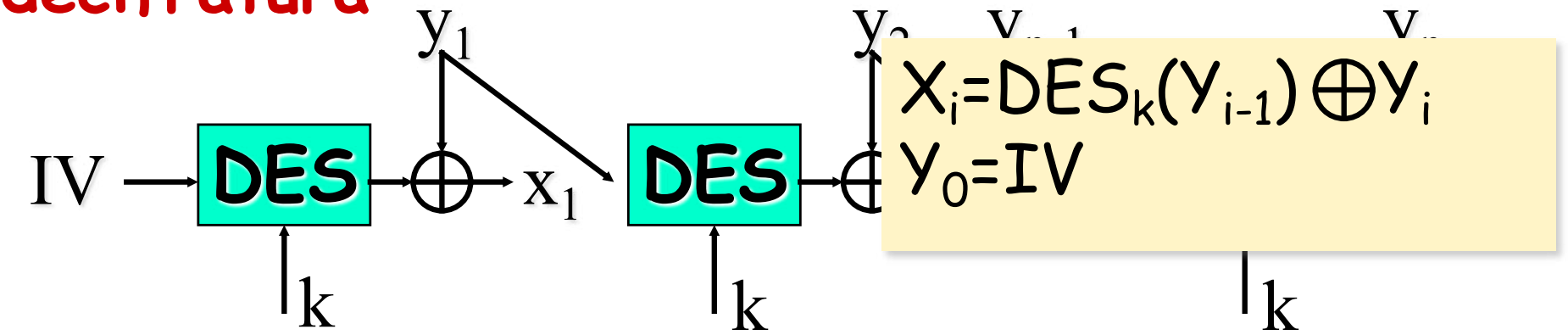


Cipher feedback (CFB)

cifratura



decifratura



Cipher feedback (CFB)

- Cifratura parallelizzabile?
- Decifratura parallelizzabile?
- Che succede se decifriamo con IV errato?

Cipher feedback (CFB)

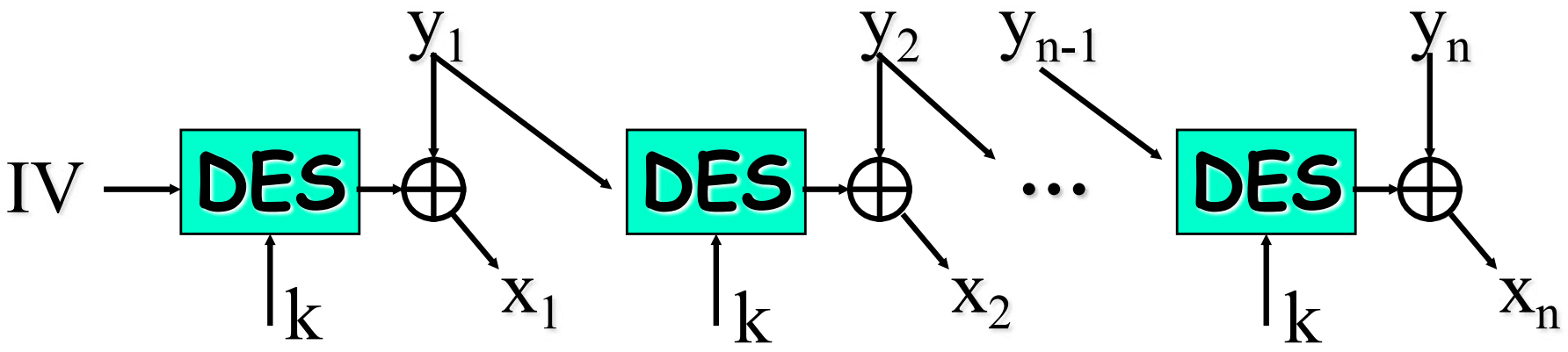
- Cifratura parallelizzabile? NO
- Decifratura parallelizzabile?
- Che succede se decifriamo con IV errato?

Cipher feedback (CFB)

- Cifratura parallelizzabile? NO
- Decifratura parallelizzabile? SI
- Che succede se decifriamo con IV errato?

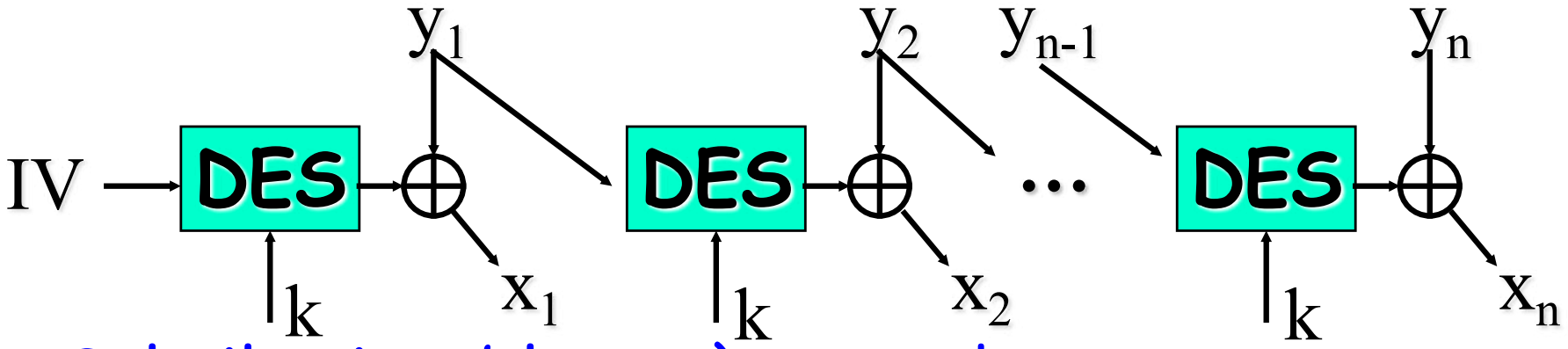
Cipher feedback (CFB)

- Cifratura parallelizzabile? NO
- Decifratura parallelizzabile? SI
- Che succede se decifriamo con IV errato?



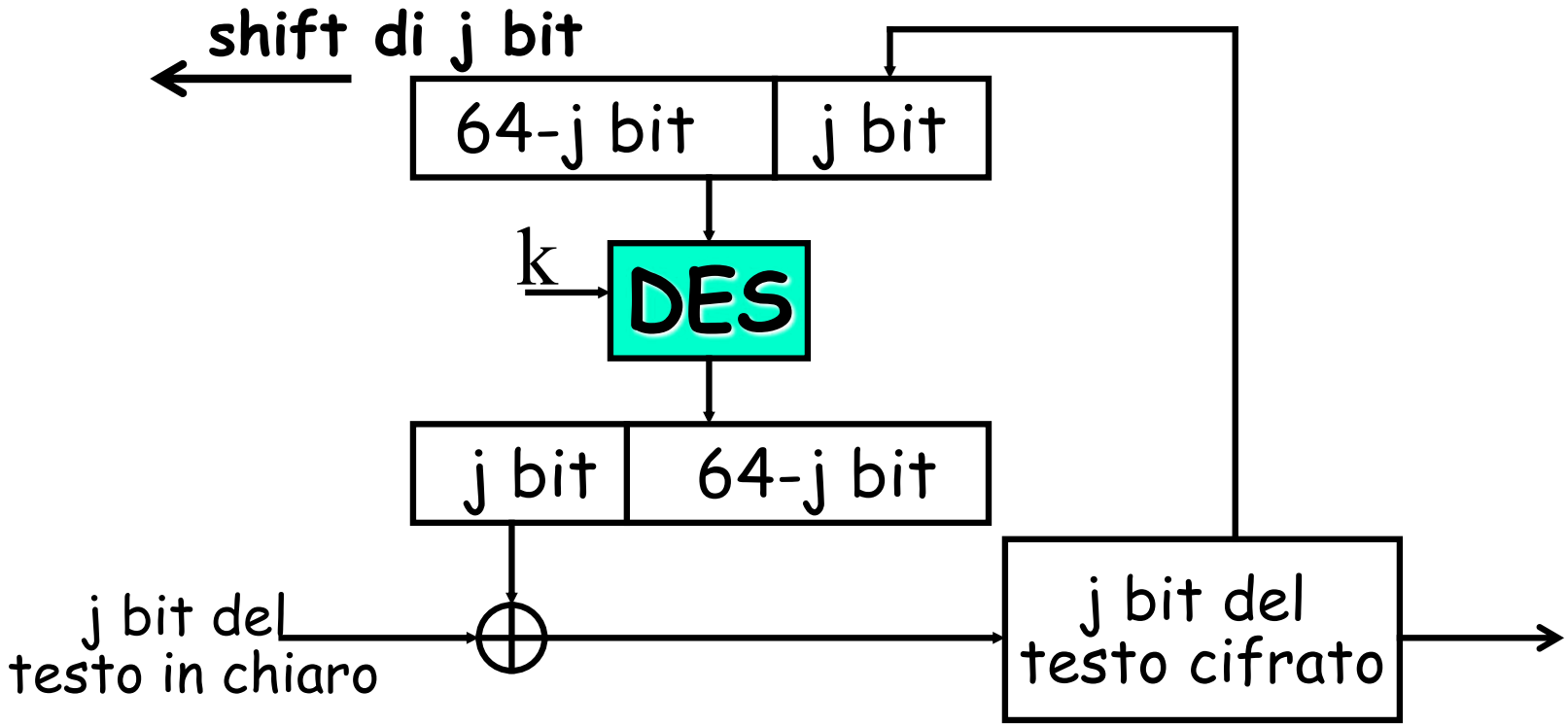
Cipher feedback (CFB)

- Cifratura parallelizzabile? NO
- Decifratura parallelizzabile? SI
- Che succede se decifriamo con IV errato?



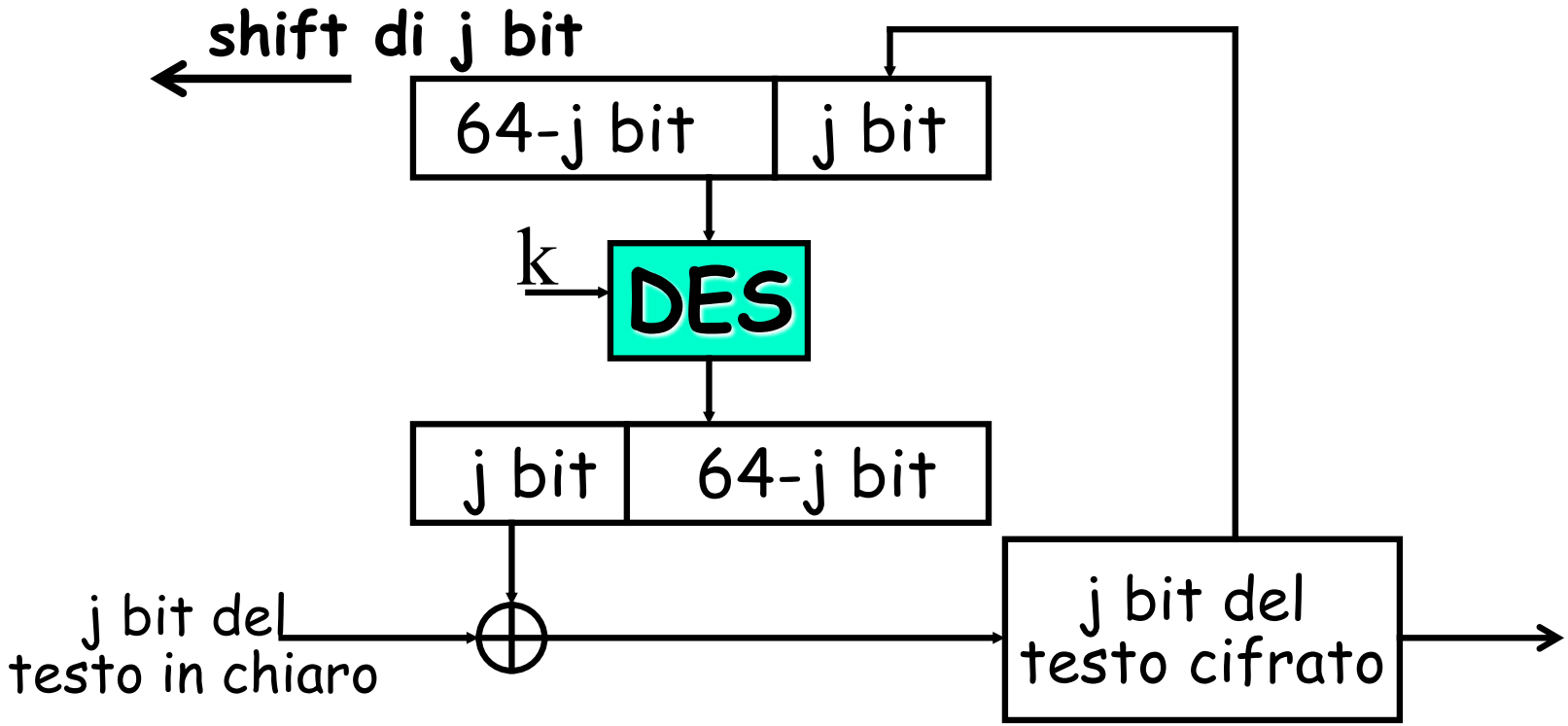
Solo il primo blocco è errato!

j-bit CFB



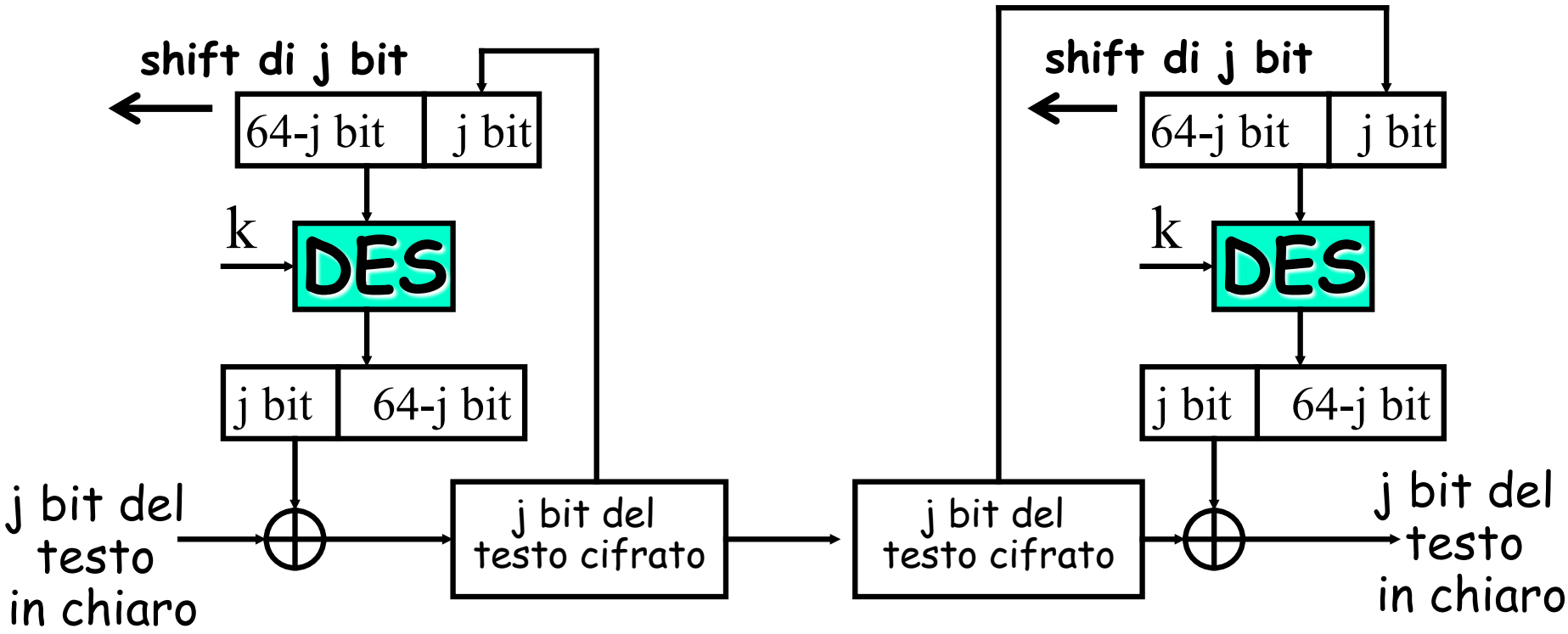
Si inizia cifrando IV

j-bit CFB



CFB è uguale a **64-bit CFB**

j-bit CFB



Cifratura

Decifratura

j-bit CFB

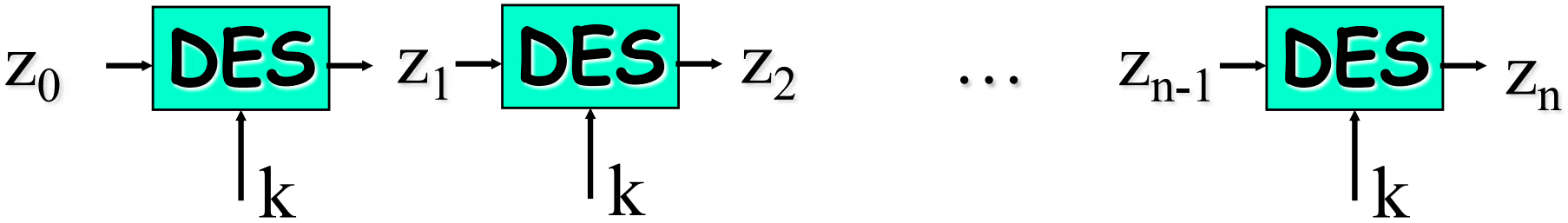
- j può essere scelto a piacimento, ad es. j=8
- Si possono utilizzare j bit cifrati senza aspettarne 64

- Più lento al decrescere di j

Output feedback (OFB)

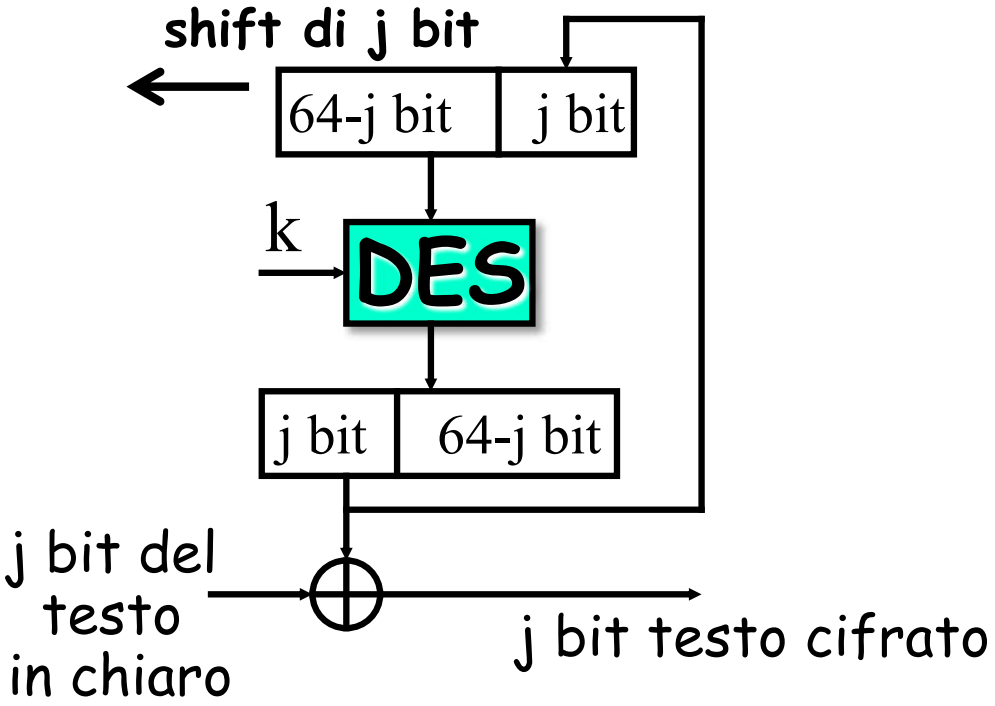
messaggio in chiaro $x=x_1x_2...x_n$
(diviso in n blocchi di 64 bit)

sequenza $z=z_0z_1...z_n$ indipendente dal messaggio, z_0=IV



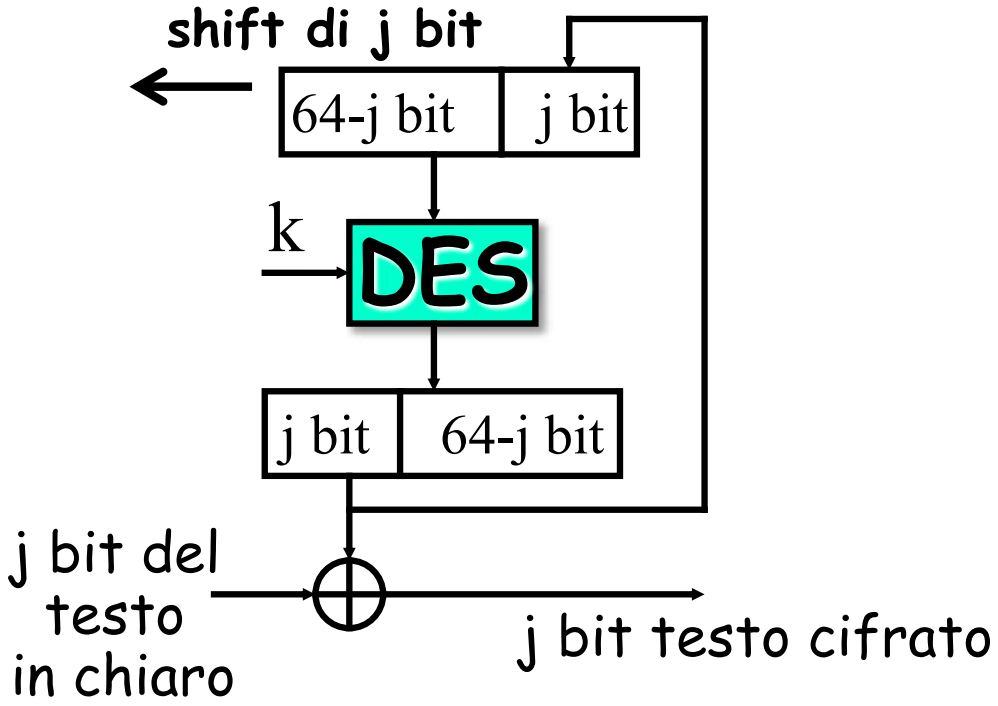
messaggio cifrato $y = y_1y_2...y_n$ $\quad y_i=x_i \oplus z_i$

j-bit OFB



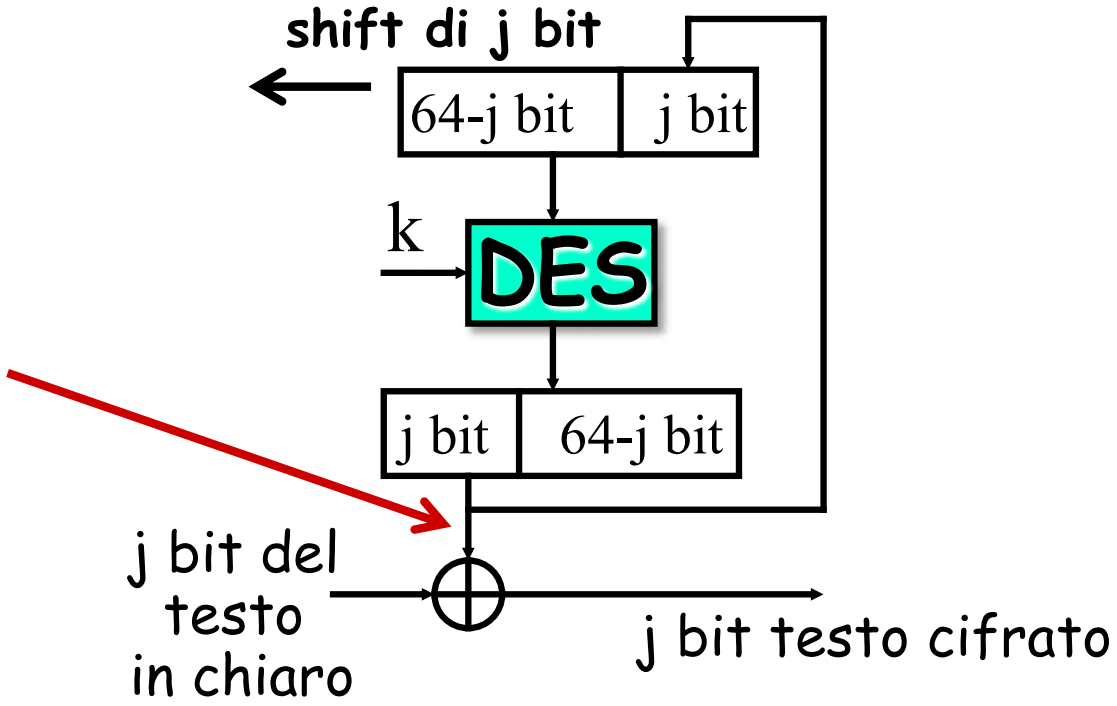
Si inizia cifrando IV

j-bit OFB



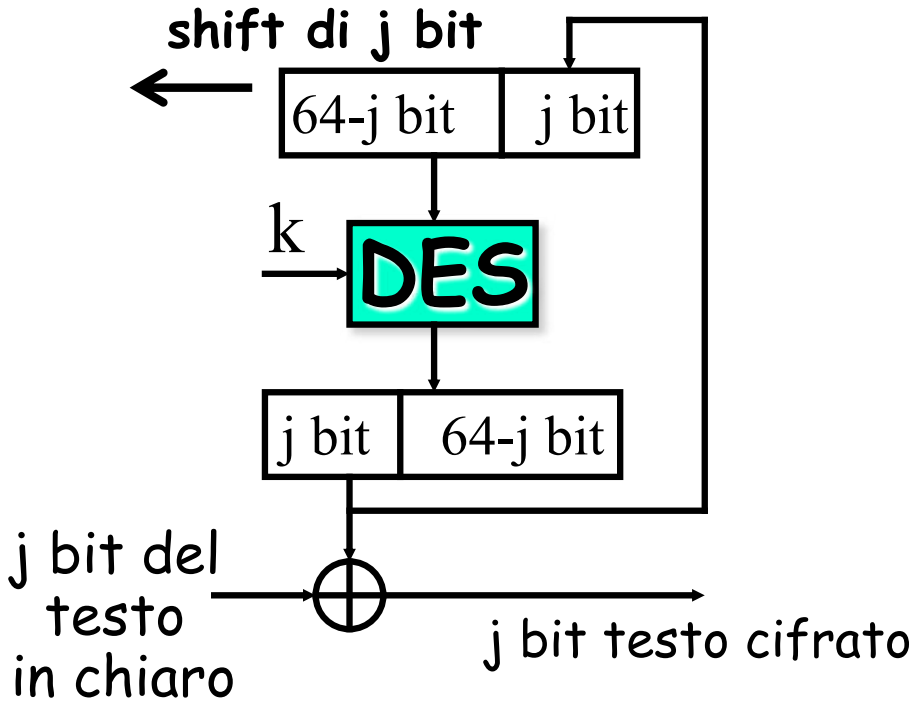
OFB è uguale a 64-bit OFB

j-bit OFB

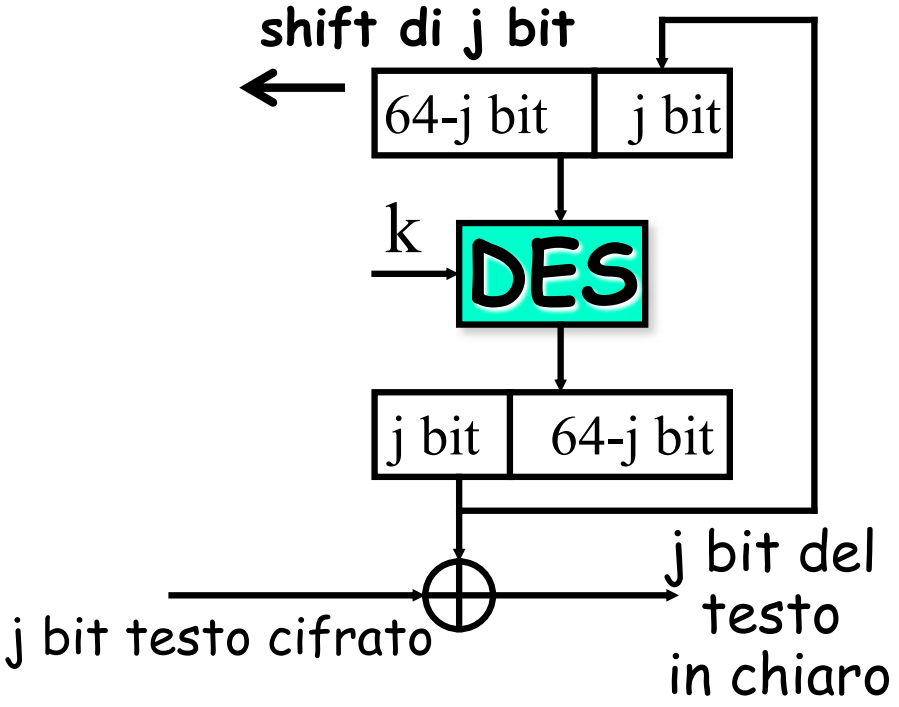


I valori input allo xor possono essere precomputati

j-bit OFB



Cifratura



Decifratura

j-bit OFB

- XOR veloci da realizzare

- Non c'è dipendenza tra i blocchi

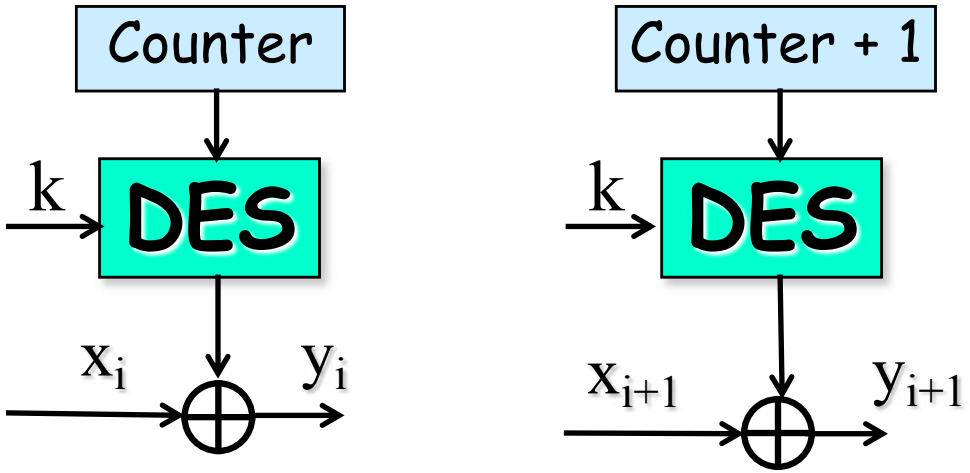
  - Cambiando un bit nel testo in chiaro, cambia un solo bit nel cifrato

Counter

Impiega un contatore di 64 bit

messaggio in chiaro $x=x_1x_2...x_n$
(diviso in n blocchi di 64 bit)

messaggio cifrato $y = y_1y_2...y_n$

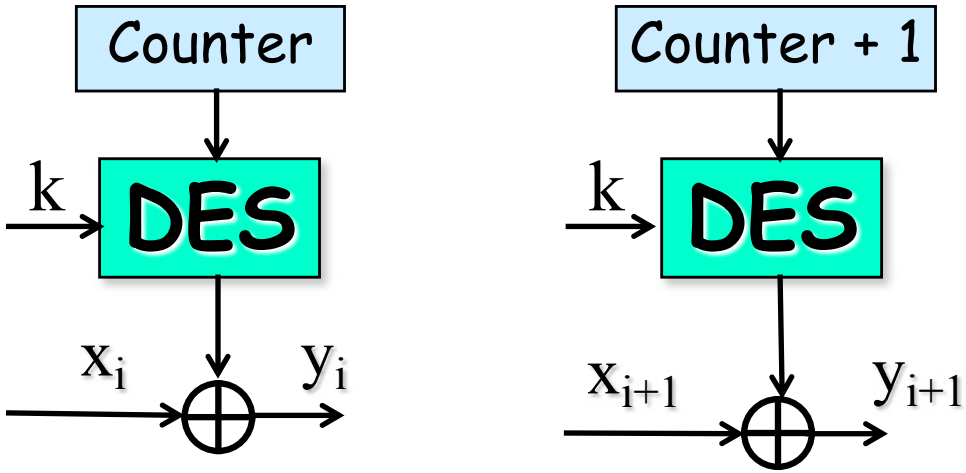


Counter

Impiega un contatore di 64 bit

messaggio in chiaro $x=x_1x_2...x_n$
(diviso in n blocchi di 64 bit)

messaggio cifrato $y = y_1y_2...y_n$



Evitare riuso della stessa chiave e dei valori Counter, Counter+1, Counter+2, …, già utilizzati

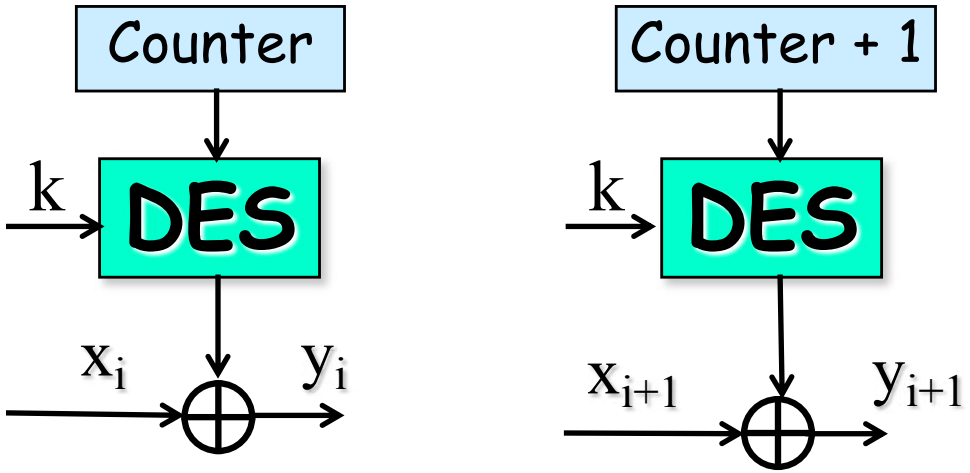
- Si otterrebbero correlazioni tra i relativi blocchi di testo in chiaro

Counter

Impiega un contatore di 64 bit

Vantaggi:

- Efficienza hardware e software
- Preelaborazione
- Accesso casuale
- Sicurezza dimostrabile
- Semplice

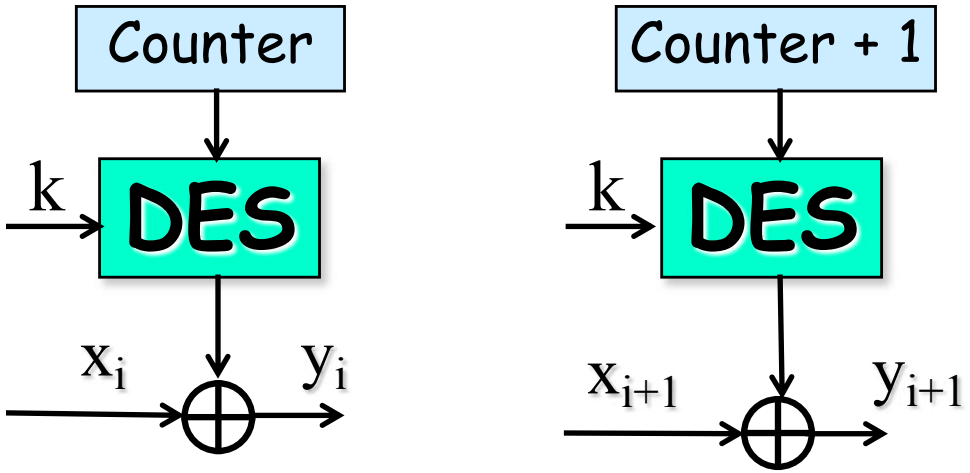


Utilizzo

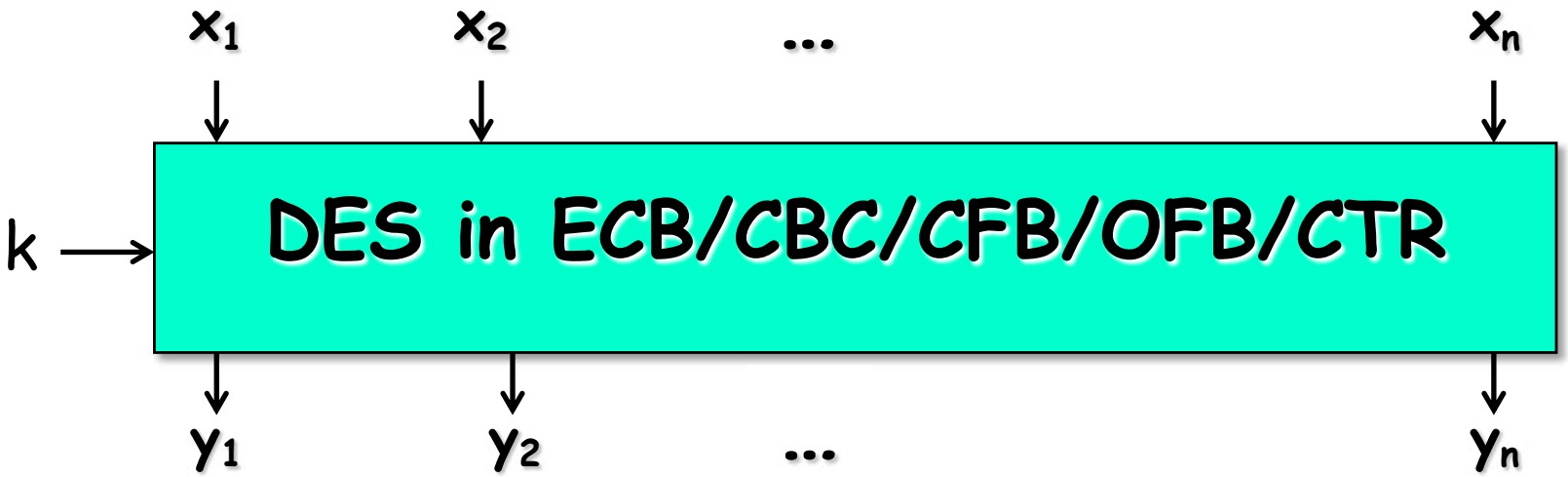
Impiega un contatore di 64 bit

Vantaggi:

- Efficienza hardware e software
- Preelaborazione
- Accesso casuale
- Sicurezza dimostrabile
- Semplice



Esercizio



Blocco y_1 trasmesso non correttamente
(alcuni 0 diventano 1 e viceversa)

Quanti e quali blocchi sono decifrati non correttamente?

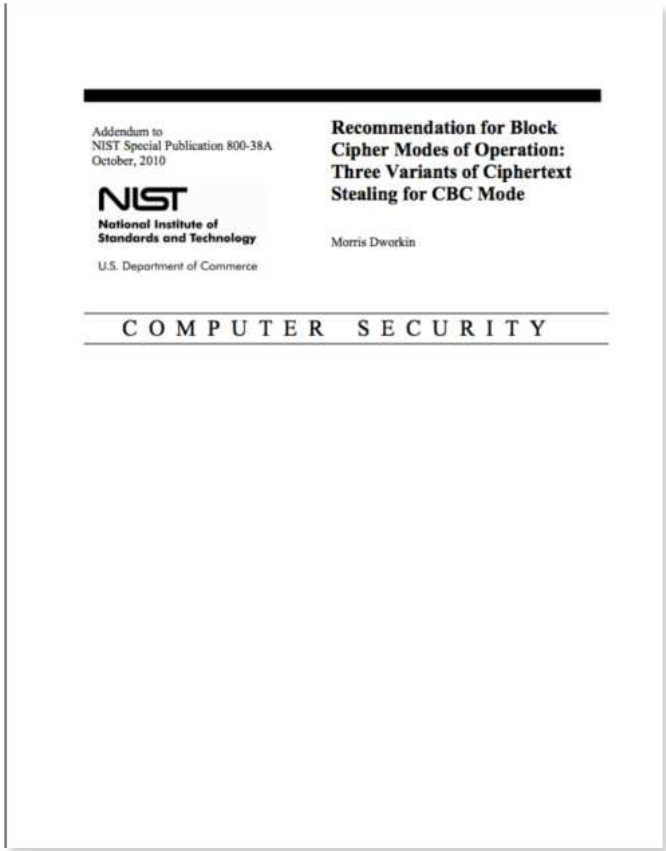
Ciphertext Stealing

Metodo per evitare una espansione del testo cifrato quando la sua lunghezza non è multipla del blocco del cifrario

- Ci sono vari metodi
- Tre varianti in NIST SP800-38A:

  CBC-CS1, CBC-CS2, CBS-CS3

http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-38a-add.pdf



Addendum to
NIST Special Publication 800-38A
October, 2010

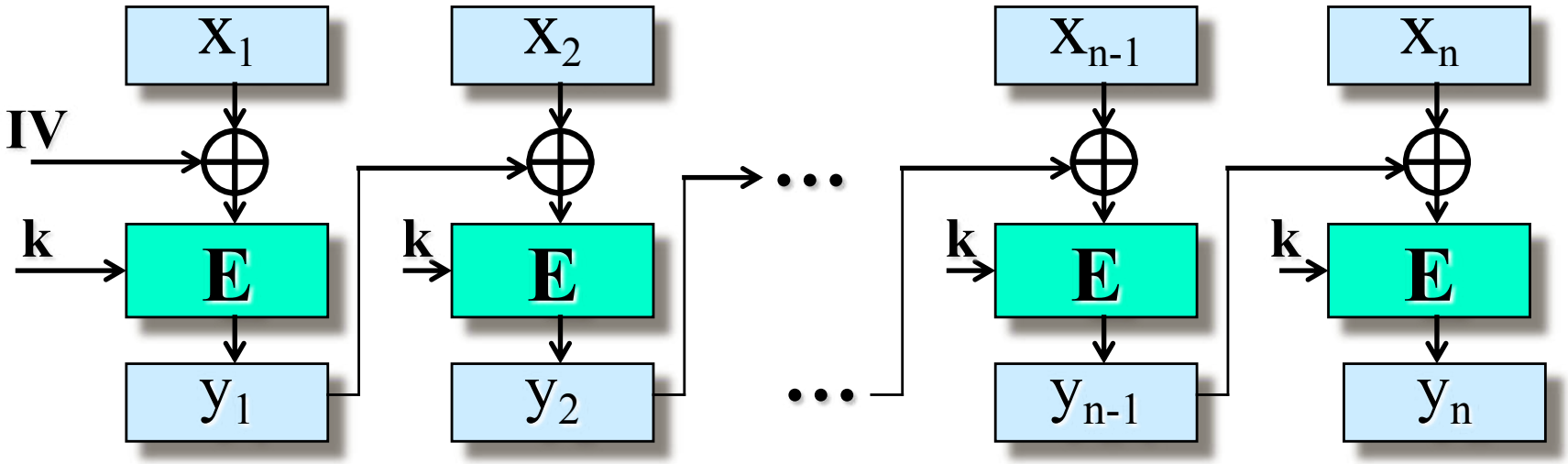
NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Recommendation for Block Cipher Modes of Operation: Three Variants of Ciphertext Stealing for CBC Mode

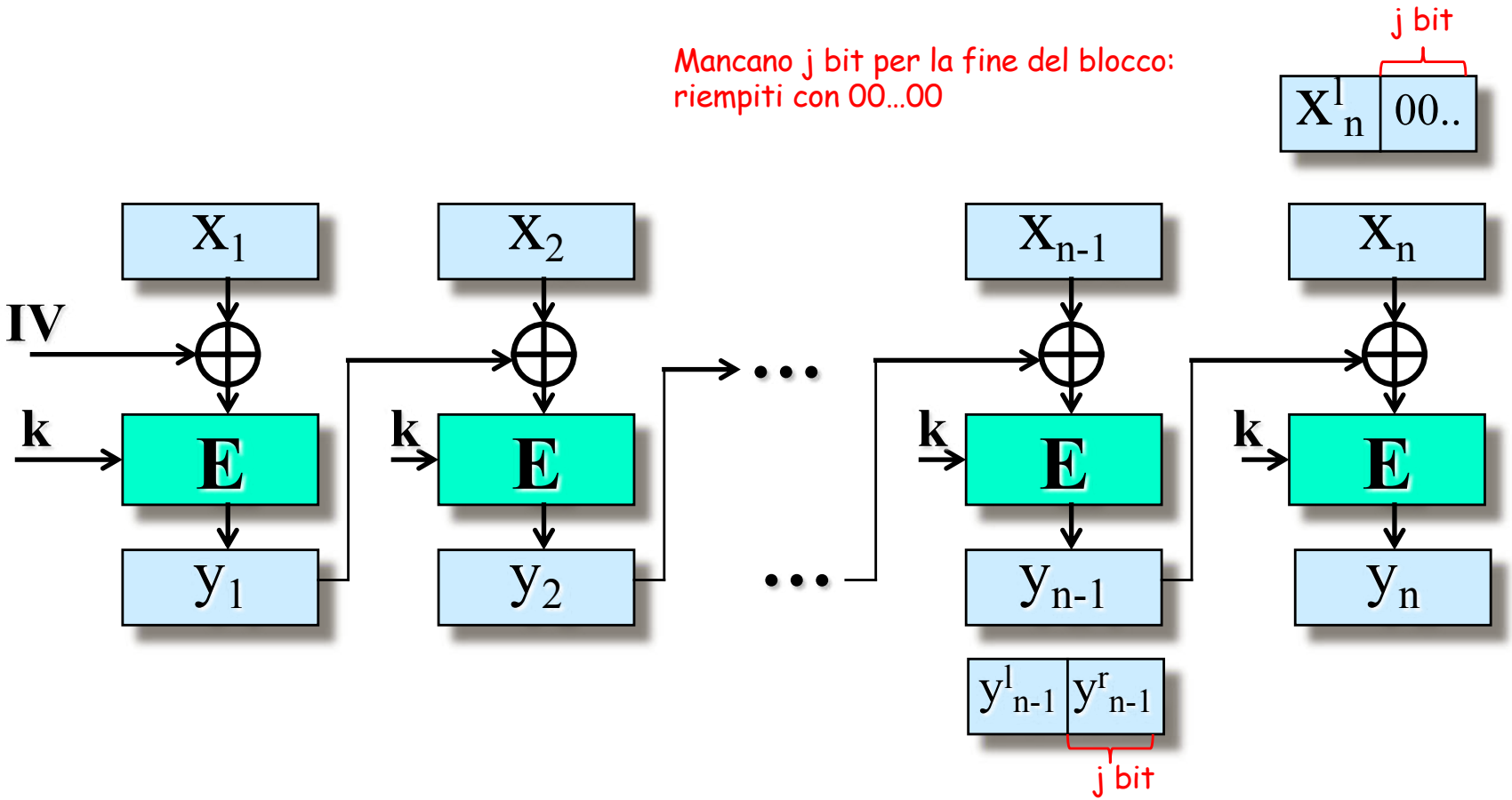
Morris Dworkin

COMPUTER SECURITY

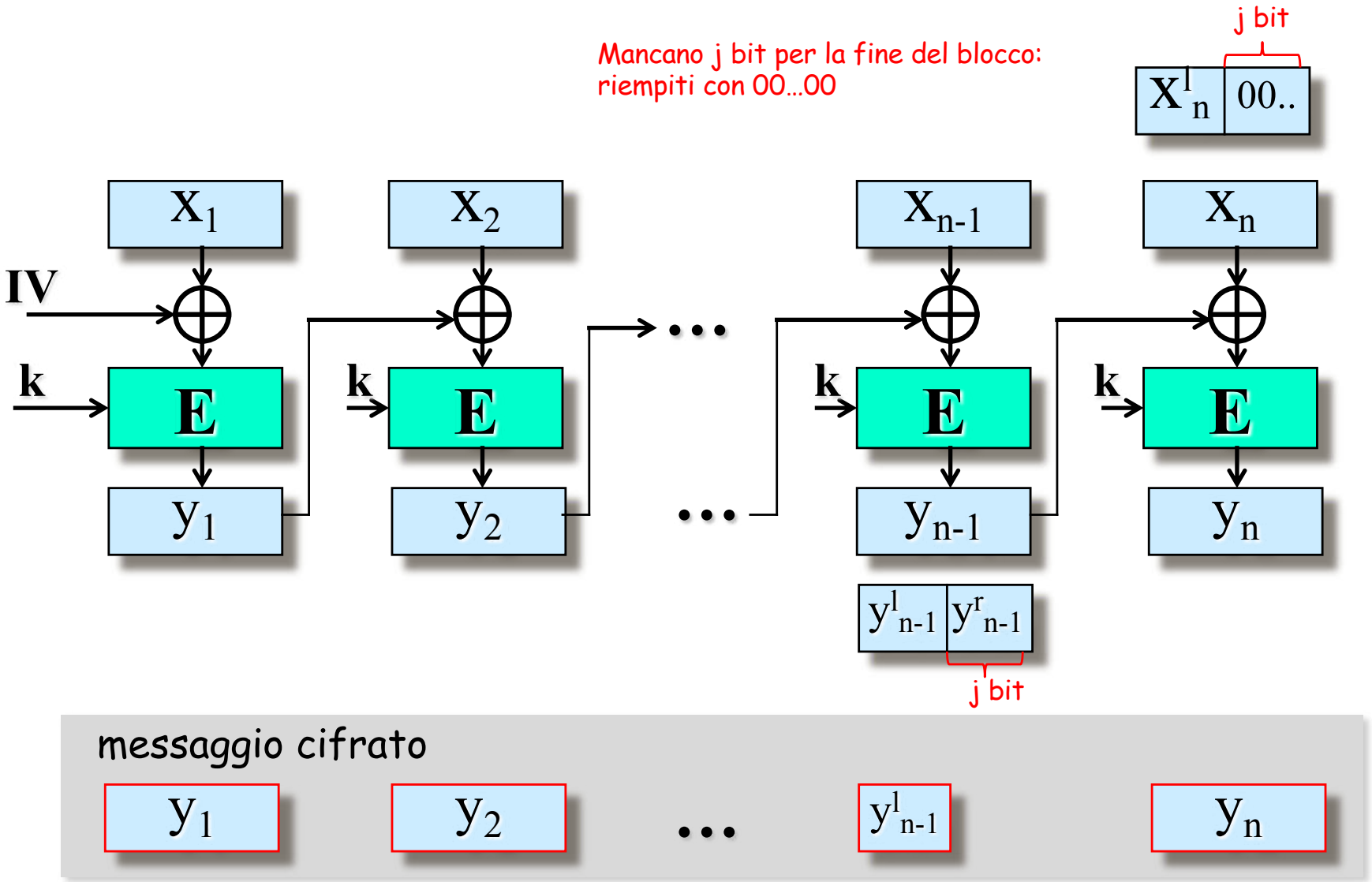
Cipher Block Chaining (CBC)



Cipher Block Chaining (CBC) con Ciphertext Stealing CS1



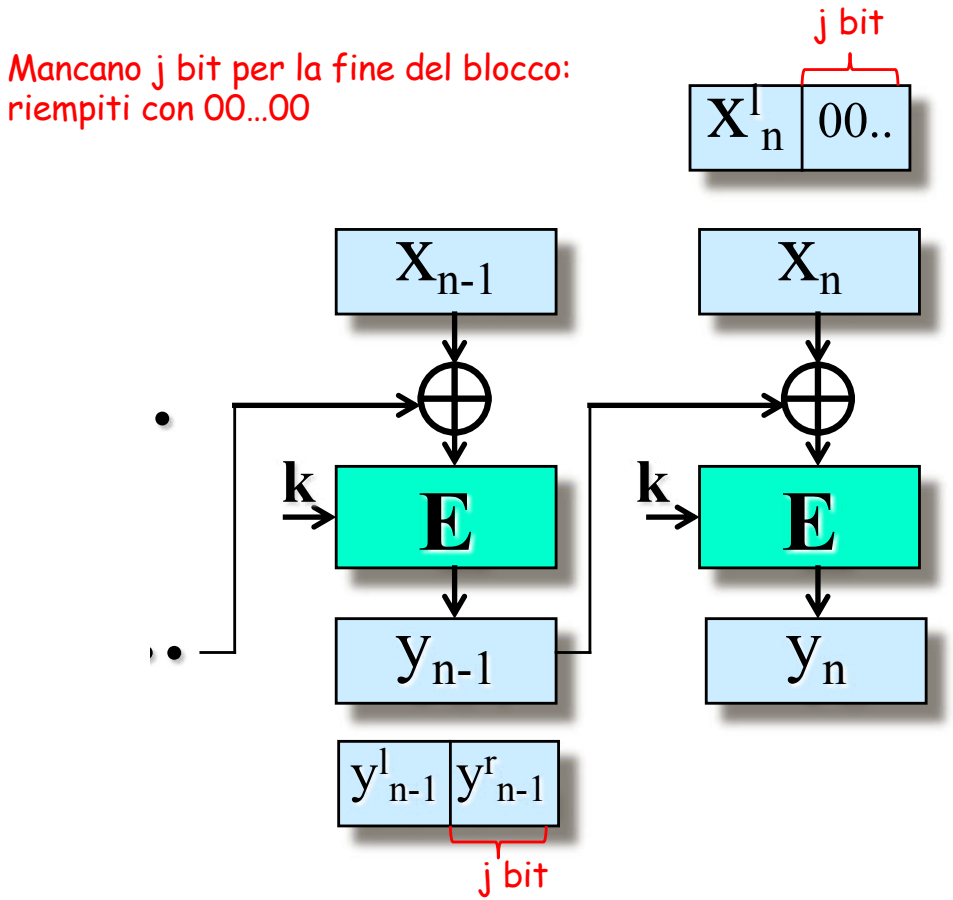
Cipher Block Chaining (CBC) con Ciphertext Stealing CS1



Cipher Block Chaining (CBC) con Ciphertext Stealing CS1

Decifratura

- y^r_{n-1} non c'è nel cifrato,
- Come lo recupero?

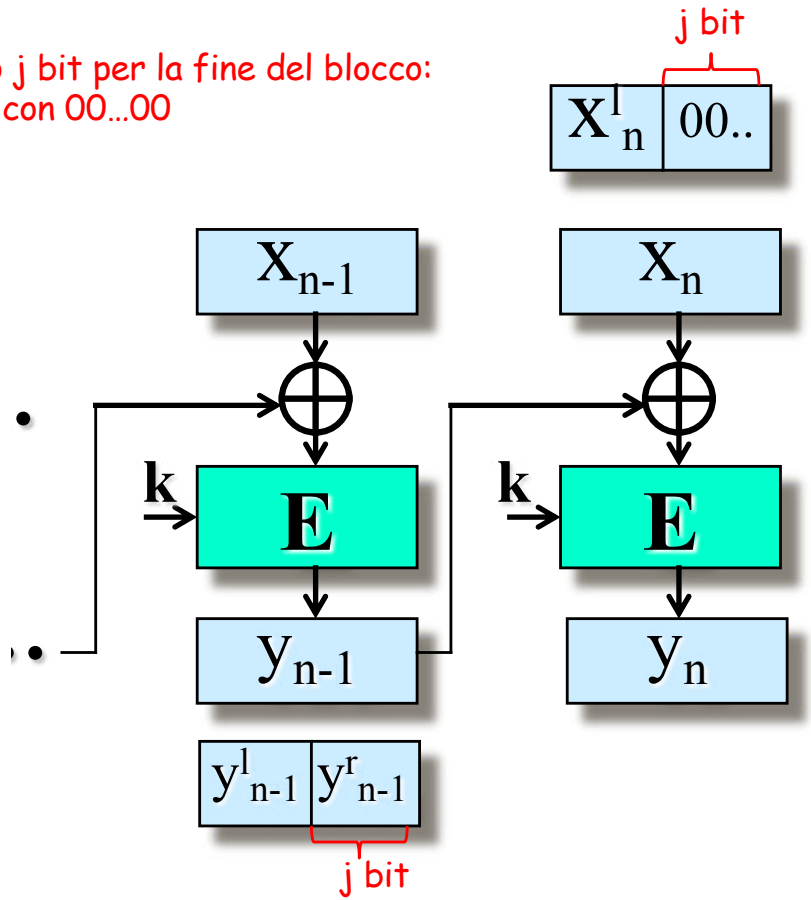


messaggio cifrato

y_1 y_2 ... y^l_{n-1} y_n

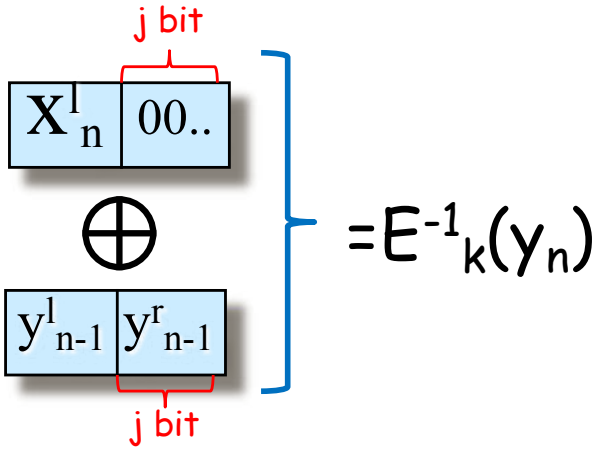
Cipher Block Chaining (CBC) con Ciphertext Stealing CS1

Mancano j bit per la fine del blocco: riempiti con 00...00



Decifratura

- y^l_{n-1} non c'è nel cifrato,
- Come lo recupero?
- Osserva che

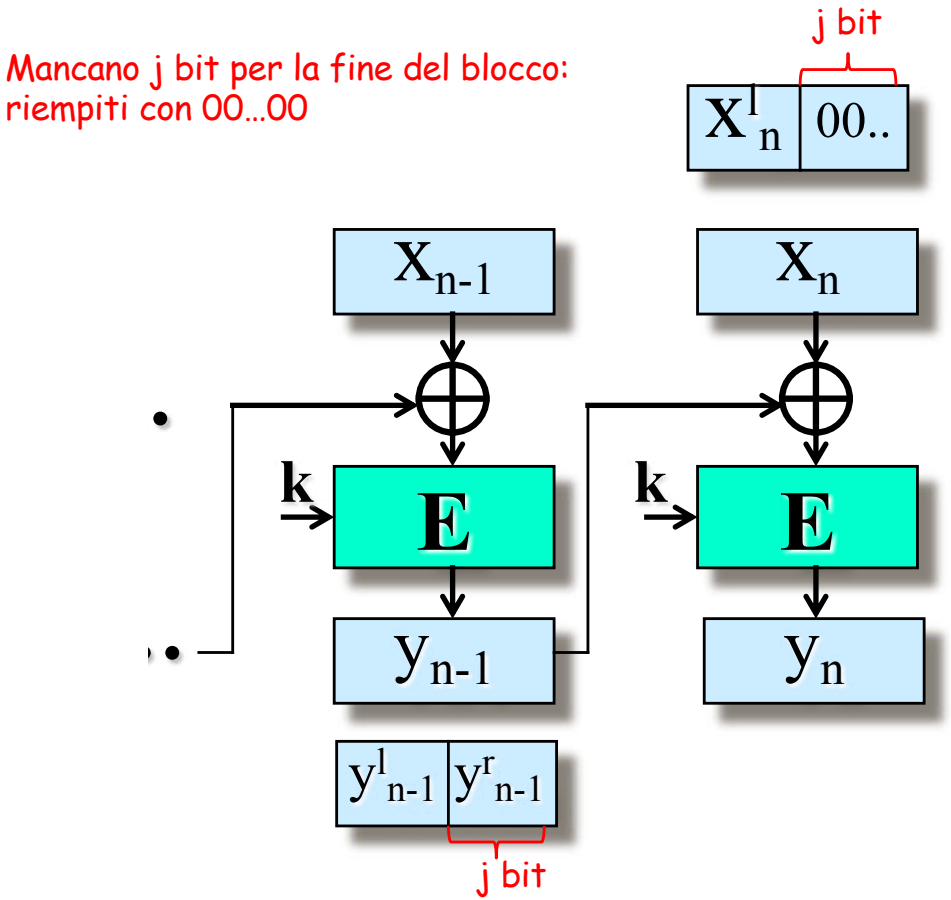


messaggio cifrato

Cipher Block Chaining (CBC) con Ciphertext Stealing CS1

Decifratura

- y^l_{n-1} non c'è nel cifrato,
- Come lo recupero?
- Quindi, $y^r_{n-1}=E^{-1}_k(y_n)$



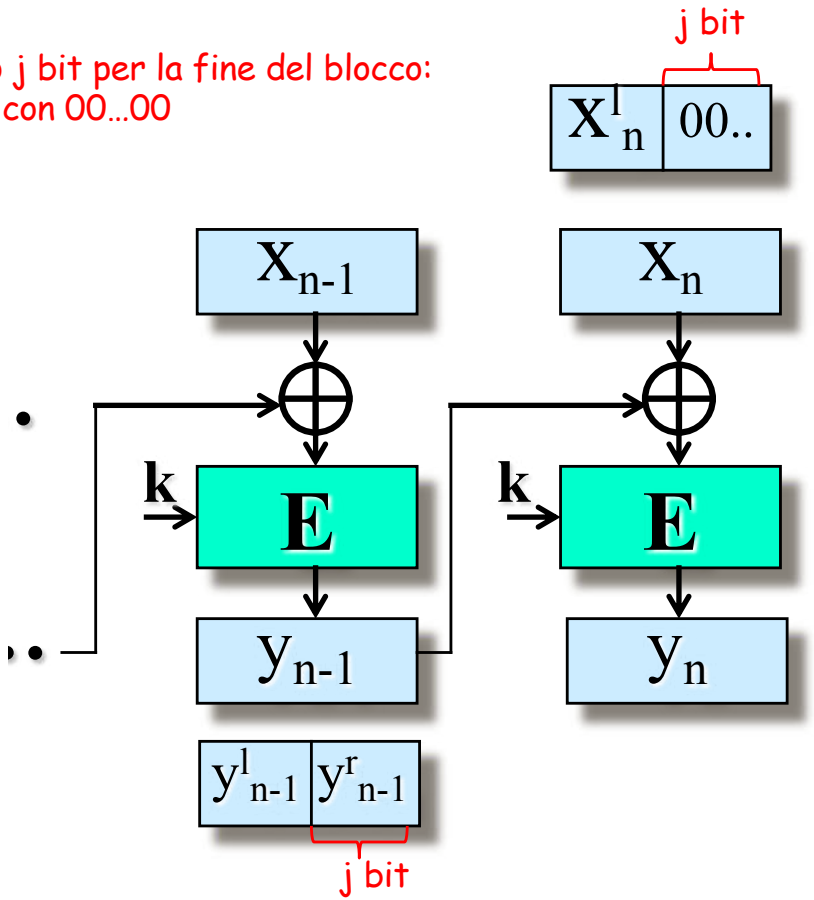
messaggio cifrato

y_1 y_2 ... y^l_{n-1} y_n

Cipher Block Chaining (CBC) con Ciphertext Stealing CS1

Esercizio:

Descrivere la decifratura



messaggio cifrato

y_1 y_2 ... y^l_{n-1} y_n

Altre modalità uso dei cifrari a blocchi

Authenticated encryption mode

- Counter with CBC-MAC (CCM) NIST, Special Publication 800-38C
- Galois/Counter Mode (GCM) NIST, Special Publication 800-38D

Authentication modes

- Cipher-based MAC (CMAC) NIST, Special Publication 800-38B

Format Preserving Encryption (dati non binari)

- NIST, Special Publication 800-38F

Key Wrapping (protezione chiavi crittografiche)

- NIST, Special Publication 800-38G

http://csrc.nist.gov/groups/ST/toolkit/BCM/current_modes.html

Altre modalità uso dei cifrari a blocchi

Proposte al NIST

- Authenticated encryption modes
- Authentication modes
- Encryption modes
- Other modes

http://csrc.nist.gov/groups/ST/toolkit/BCM/modes_development.html

Back to Top

Encryption Modes

Mode	Full Mode Name	Available Documentation
2DEM	*2D-Encryption Mode* *A. A. Belal, M. A. Abdel-Gawad*	SP I AD I IP CD I TV I SU
ABC	*Accumulated Block Chaining* *L. Knudsen*	SP I AD I IP TV I SU
BPS	*Format Preserving Encryption Proposal* *E. Brier, T. Peyrin, J. Stern*	SP I AD I IP TV I SU
CSPEM	*Character Set Preserving Encryption Mode* *Gary S. Sarasin*	SP I AD I IP TV I SU
CTR	*Counter Mode Encryption* *H. Lipmaa, P. Rogaway, D. Wagner*	SP I AD I IP TV I SU
DFF	*Delegatable Feistel-based Format-preserving Encryption Mode* *J. Vance, M. Bellare* (Posted on Nov. 9, 2015 as replacement for VAES3 proposal)	SP I AD I IP1 IP2 I TV I SU
FCEM	*Format Controlling Encryption Mode* *U. Mattsson* (Posted Jun 30, 2009)	SP I AD I IP TV I SU
FFX	*Format-preserving Feistel-based Encryption Mode* *M. Bellare, P. Rogaway, T. Spies* (April 12, 2010: Version 1.1 replacing Version 1.0)	SP I SP2 I AD IP I TV I SU
IGE	*Infinite Garble Extension* *V. Gligor, P. Donescu*	SP I AD I IP TV I SU
RAC	*Random Access Counter* *J. Anderson* (Posted May 15, 2015)	SP I AD I IP CD I TV I SU
VFPE	*VISA Format Preserving Encryption* *VISA USA Inc., Attention John Sheets or Kim R. Wagner*	SP I AD I IP TV I SU
XBC	*Cross Block Chaining (XBC)* *Andre Watson* (Posted Oct 16, 2014) (Added link to code Jan 18, 2017)	SP I AD I IP CD I TV I SU

Sicurezza contro attacchi "Brute Force"

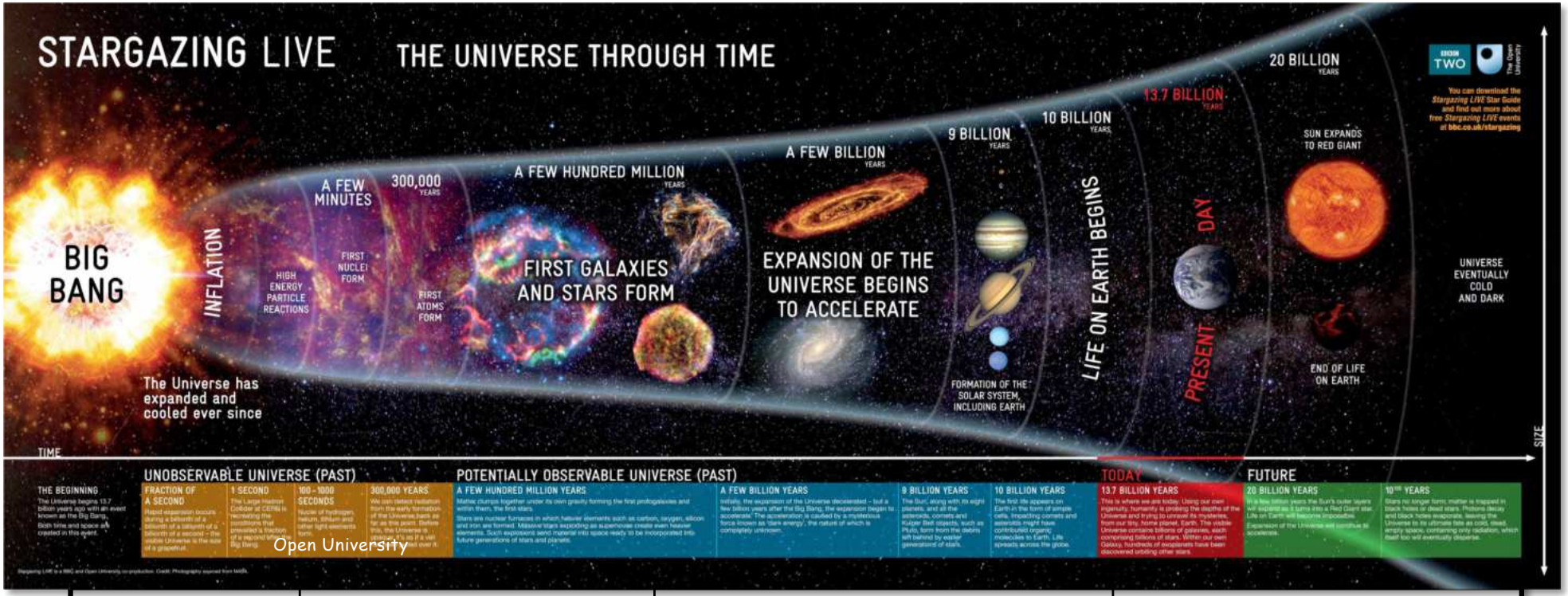
Quanto deve essere lunga la chiave per evitare attacchi di ricerca esaustiva?

Sicurezza contro attacchi "Brute Force"

Grandezza chiavi (bits)	Numero di possibili chiavi	Tempo medio necessario con 1 decifratura/μs	Tempo medio necessario con 10^6 decifrature/μs
32	$2^{32} \approx 4{,}3 \times 10^9$	2^{31} μs = 35,8 minuti	2,15 millisecondi
56	$2^{56} \approx 7{,}2 \times 10^{16}$	2^{55} μs = 1142 anni	10,01 ore
112	$2^{112} \approx 5{,}19 \times 10^{33}$	2^{111} μs = $8{,}2 \times 10^{19}$ anni	$8{,}2 \times 10^{13}$ anni
128	$2^{128} \approx 3{,}4 \times 10^{38}$	2^{127} μs = $5{,}4 \times 10^{24}$ anni	$5{,}4 \times 10^{18}$ anni
168	$2^{168} \approx 3{,}7 \times 10^{50}$	2^{167} μs = $5{,}9 \times 10^{36}$ anni	$5{,}9 \times 10^{30}$ anni
256	$2^{256} \approx 1{,}15 \times 10^{77}$	2^{255} μs = $1{,}83 \times 10^{63}$ anni	$1{,}83 \times 10^{57}$ anni

Micro (μ) è 0,000 001 (un milionesimo)

Sicurezza contro attacchi "Brute Force"



256	$2^{256} \approx 1{,}15 \times 10^{77}$	$2^{255}\ \mu s \quad = 1{,}83 \times 10^{63}$ anni	$1{,}83 \times 10^{57}$ anni

Stima età universo 13,798 ± 0,037 miliardi anni $\approx 4 \times 10^{17}$ secondi

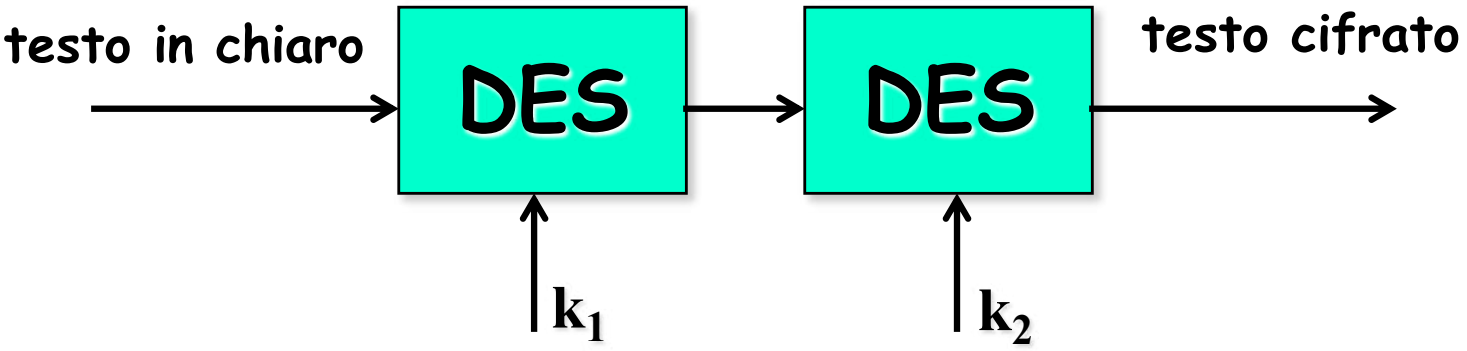
(atre stime 13-15 miliardi anni) $\approx 4 \times 10^{23}$ microsecondi

Cifratura multipla

- Debolezza del DES: chiave piccola (56 bit)
- Come costruire un cifrario più sicuro a partire dal DES (senza modificarne la struttura)?
- Cifratura multipla

  Cifrare il messaggio varie volte con chiavi differenti, sperando che questo aumenti la sicurezza...

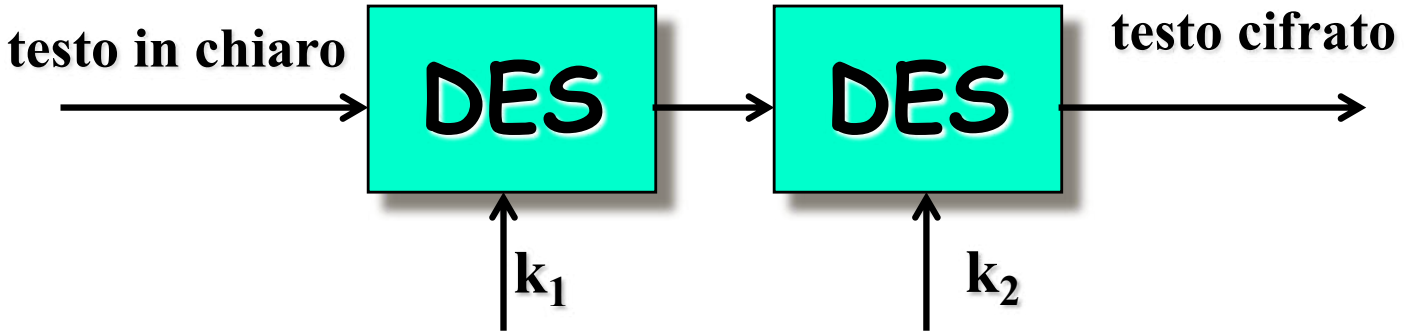
DES Doppio



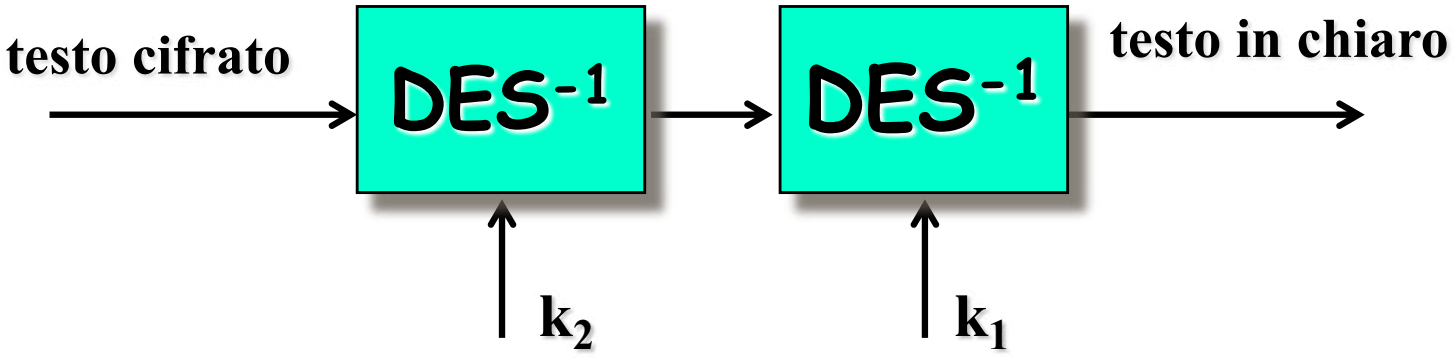
- lunghezza blocco = 64 bit
- chiave (k_1, k_2) lunga 56+56 = 112 bit

DES Doppio

Cifratura



Decifratura



Sicurezza DES doppio

Quanto è "sicuro"
il DES doppio?

DES ≡ DES doppio ?

Data una coppia (k_1, k_2), se esistesse k_3 tale che

$$DES_{k_3}(\cdot) = DES_{k_2}(DES_{k_1}(\cdot))$$

la doppia cifratura sarebbe equivalente

ad una cifratura singola

DES non forma un gruppo

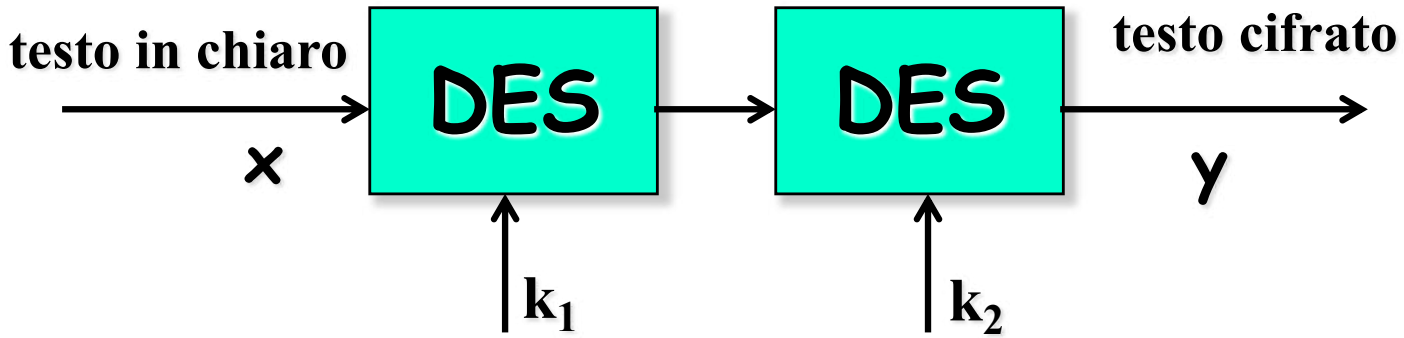
- Esistono $2^{64}! > 10^{347.380.000.000.000.000.000} > 10^{10^{20}}$ permutazioni
  - Corrispondenti ai 2^{64} input possibili
- Ci sono $2^{56} < 10^{17} << 10^{10^{20}}$ permutazioni definite da DES
  - A ciascuna chiave, DES fa corrispondere una permutazione
- Se DES viene applicato due volte con chiavi diverse può produrre una delle permutazioni non definite da DES

L'insieme delle 2^{56} permutazioni definite dalle 2^{56} chiavi DES non è chiuso per composizione

(dimostrato solo nel 1992)

DES Doppio

attacco *meet in the middle*

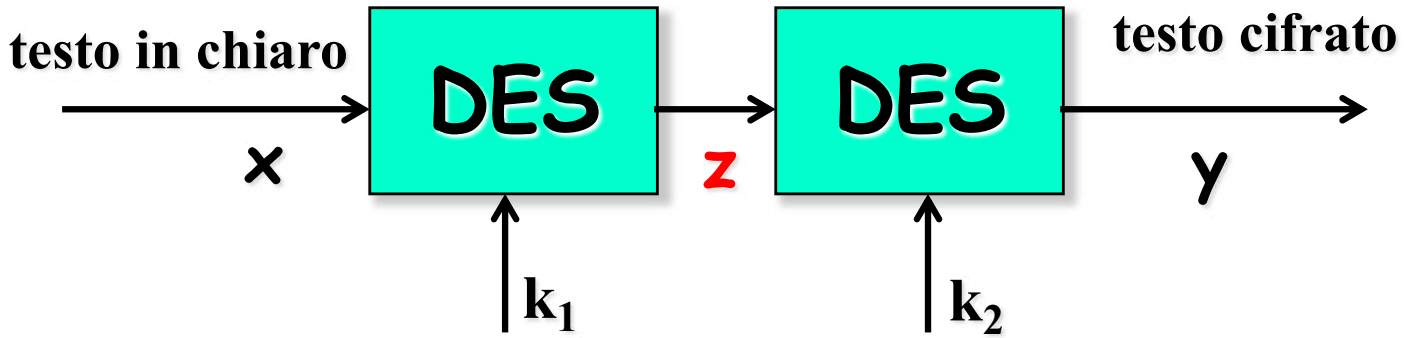


- Known Plaintext Attack

  Si conoscono coppie (x,y), ma non la chiave (k_1,k_2)

DES Doppio

attacco *meet in the middle*



Data una coppia nota (x,y)

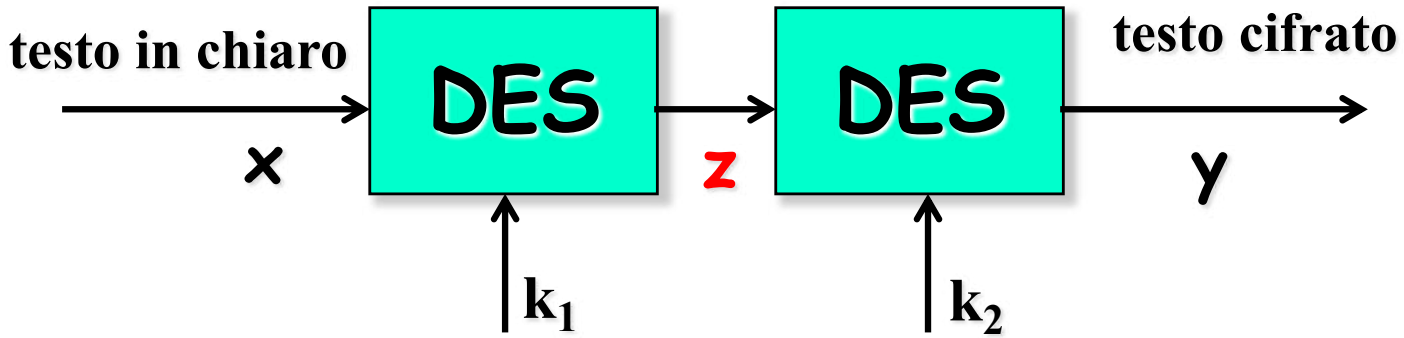
Cifro x con i 2^{56} valori possibili di k_1

Decifro y con i 2^{56} valori possibili di k_2

Se trovo match per z allora ho trovato chiave!

DES Doppio

attacco *meet in the middle*



Data una coppia nota (x,y)

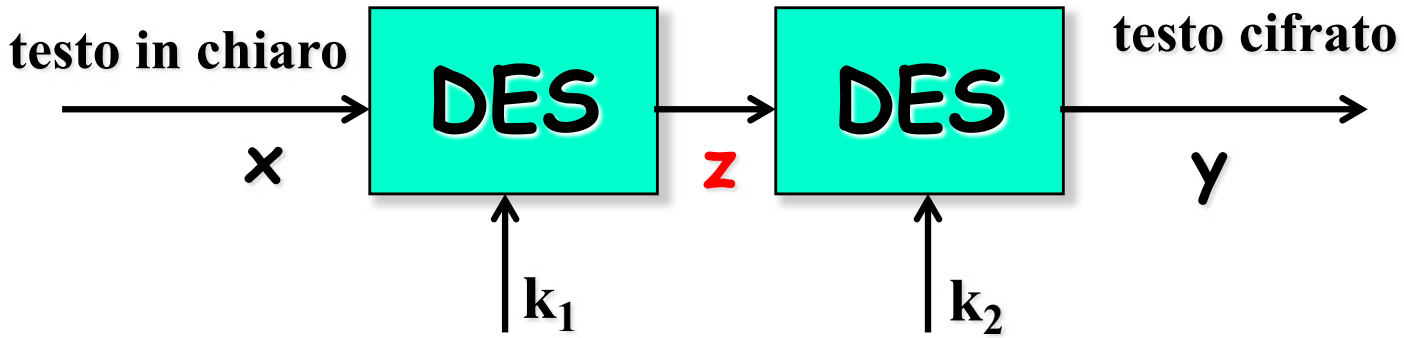
Cifro x con i 2^{56} valori possibili di k_1

Decifro y con i 2^{56} valori possibili di k_2

Se trovo $DES_{ka}(x) = DES^{-1}_{kb}(y)$
allora ho trovato chiave (k_a,k_b)!

DES Doppio

attacco *meet in the middle*



Data una coppia nota (x,y)

Eseguo tutte le cifrature di x per i 2^{56} valori possibili di k_1 in una tabella

Eseguo le decifrature di y per i 2^{56} valori possibili di k_2 e cerco le corrispondenze nella tabella

DES Doppio:

attacco meet in the middle

Known Plaintext Attack

Input: x, $y = DES_{k_2}(DES_{k_1}(x))$

Costruisci tabella

chiave	testo cifrato
00…00	$DES_{00\ldots00}(x)$
00…01	$DES_{00\ldots01}(x)$
…	…
11…11	$DES_{11\ldots11}(x)$

for $k_b \in \{0,1\}^{56}$

do $z = DES^{-1}_{k_b}(y)$

if per qualche k_a, (k_a, z) è nella tabella

then return la chiave è (k_a, k_b)

DES Doppio:
attacco *meet in the middle*

- **Complessità spazio**: 2^{56} righe nella tabella
- **Complessità tempo**:
  - 2^{56} cifrature per x (costruzione tabella)
  - 2^{56} decifrature per y
  - 2^{56} ricerche in tabella
    - O(1) se tabella hash
    - 56 se array ordinato

chiave	testo cifrato
00…00	$DES_{00\ldots00}(x)$
00…01	$DES_{00\ldots01}(x)$
…	…
11…11	$DES_{11\ldots11}(x)$

DES Doppio:
attacco *meet in the middle*

L'output (k_1,k_2) è sicuramente la chiave cercata?

DES Doppio:
attacco *meet in the middle*

Dato x, y qual'è il numero di chiavi (k_1,k_2) tali che

$$y = DES_{k_2}(DES_{k_1}(x))\ ?$$

E' troppo complicato calcolarlo esattamente per il DES

DES Doppio:
attacco *meet in the middle*

Dato x, qual'è il numero medio di chiavi (k_1,k_2) per un valore cifrato

$$DES_{k_2}(DES_{k_1}(x)) ?$$

E' un'approssimazione, ma riusciamo a calcolarlo

DES Doppio:
attacco *meet in the middle*

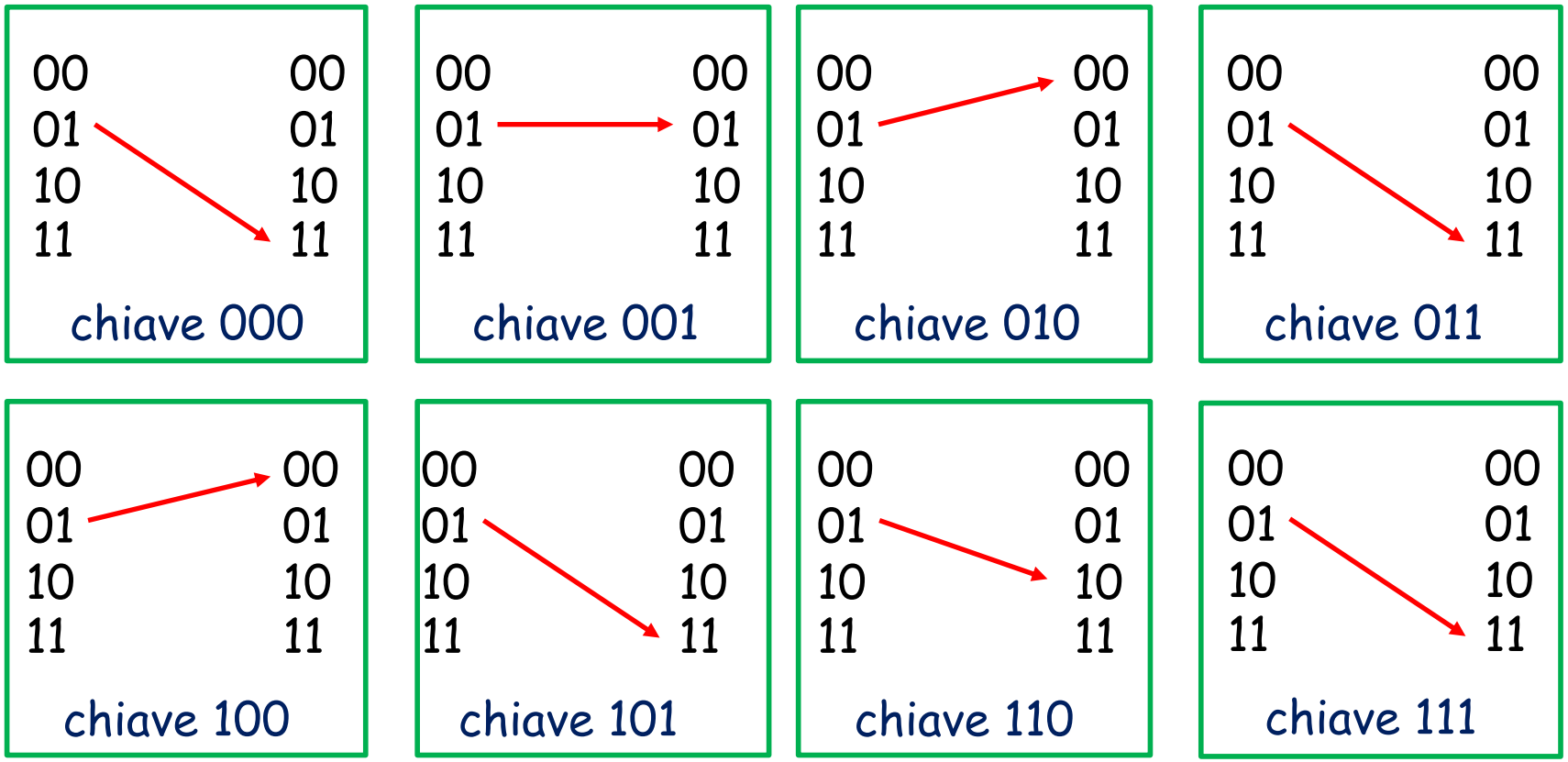
Dato x, qual'è il numero medio di chiavi (k_1,k_2) per un valore cifrato

$$DES_{k_2}(DES_{k_1}(x)) ?$$

E' un'approssimazione, ma riusciamo a calcolarlo

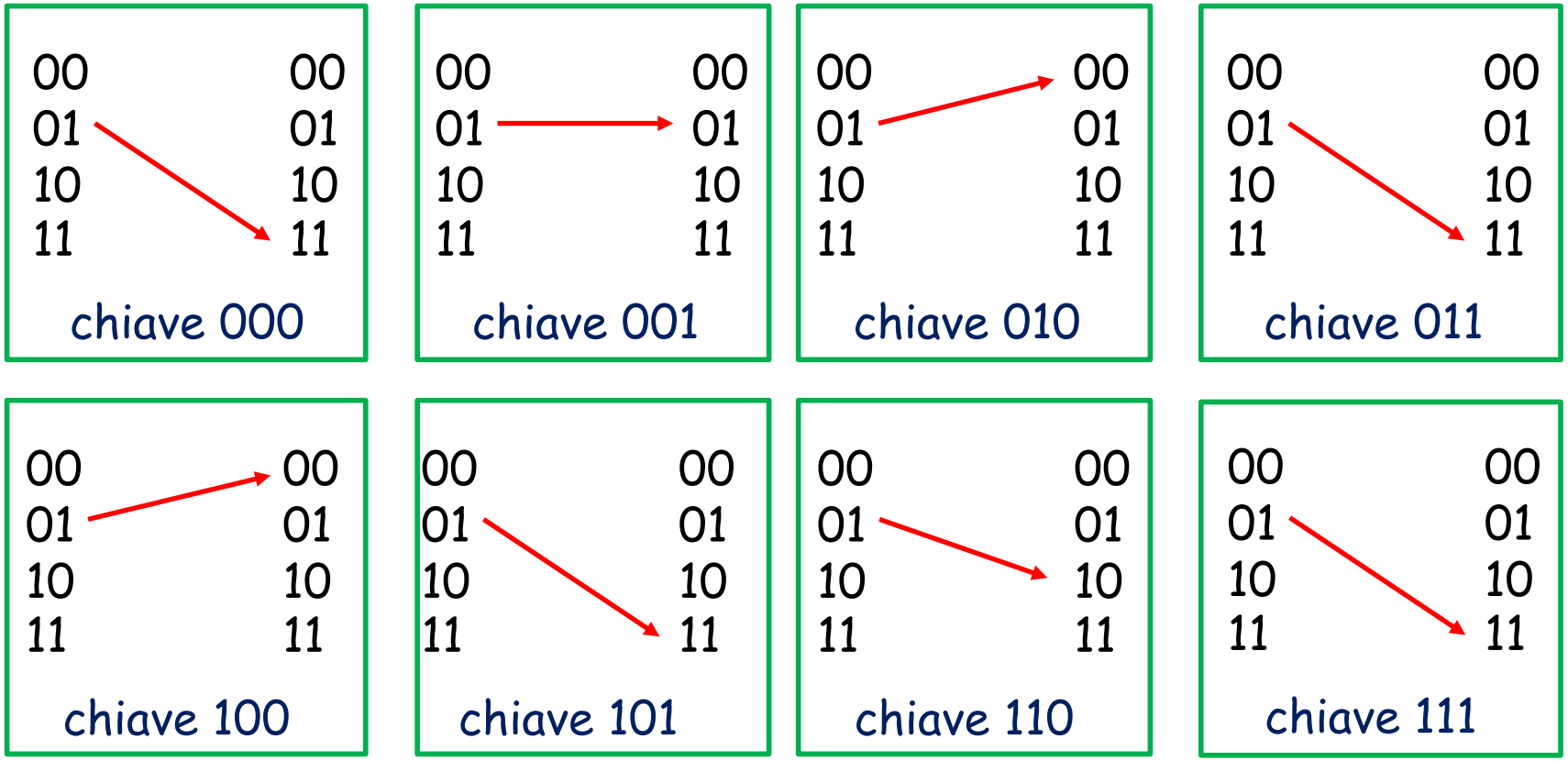
Vediamo prima un semplice esempio per un cifrario a blocchi con chiave di 3 bit

Esempio semplificato per il calcolo



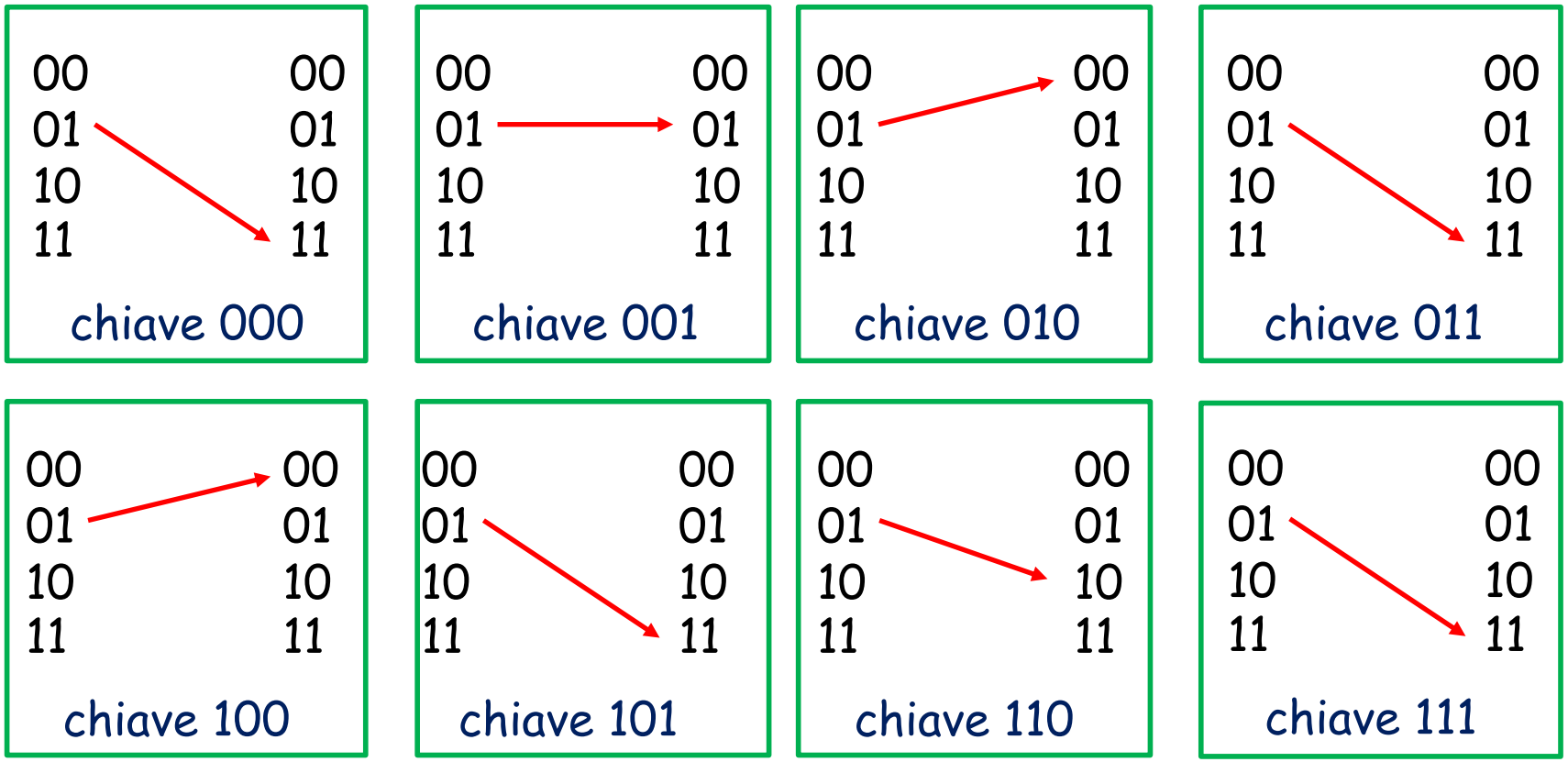
Testo cifrato 00	2 chiavi
Testo cifrato 01	1 chiavi
Testo cifrato 10	1 chiavi
Testo cifrato 11	4 chiavi

Esempio semplificato per il calcolo



Dato 01 qual'è il numero medio di chiavi k per un valore cifrato ?

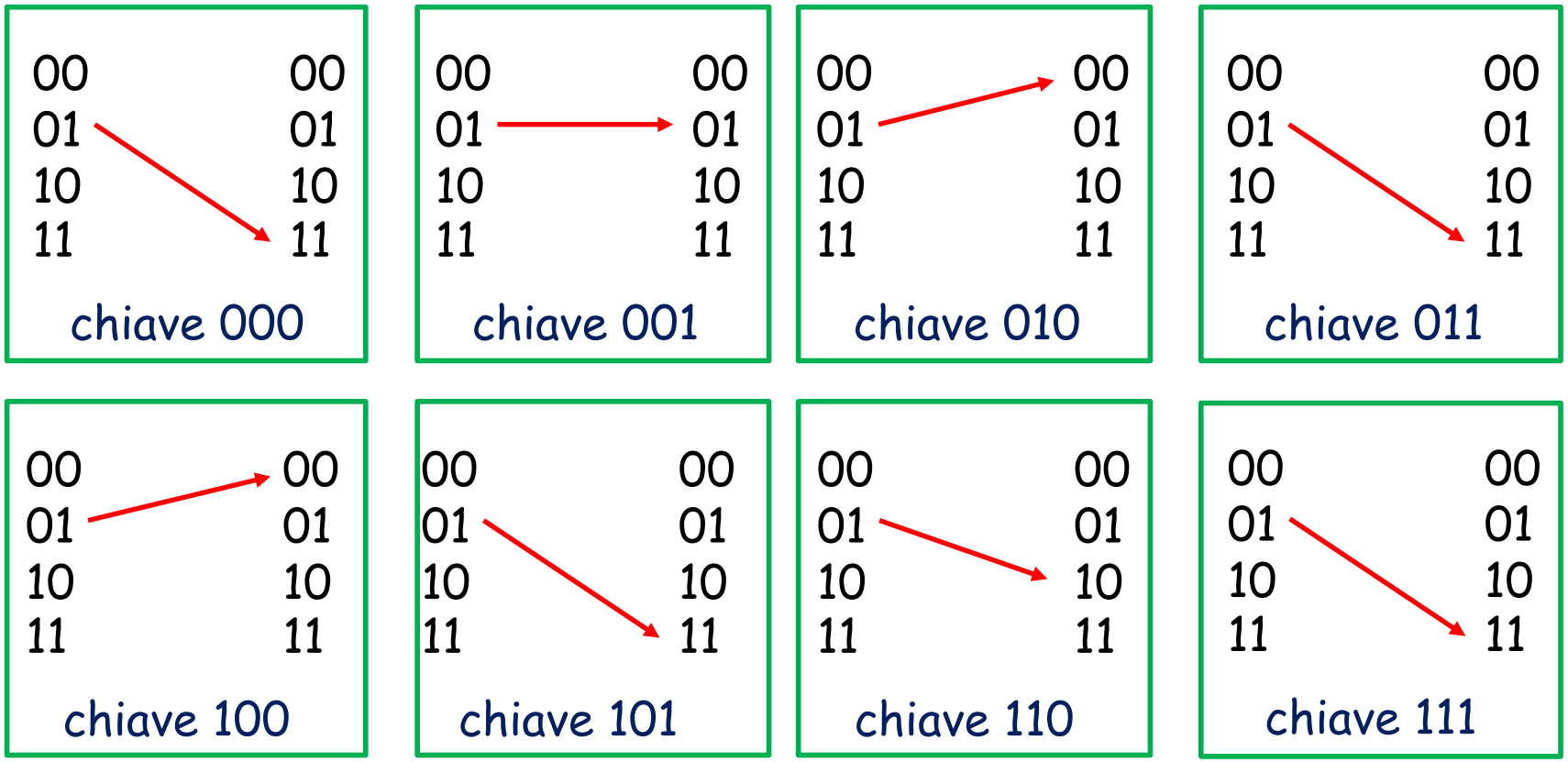
Esempio semplificato per il calcolo



Testo cifrato 00	2 chiavi
Testo cifrato 01	1 chiavi
Testo cifrato 10	1 chiavi
Testo cifrato 11	4 chiavi

in media (2+1+1+4)/4 = 2 chiavi

Esempio semplificato per il calcolo



Testo cifrato 00	2 chiavi
Testo cifrato 01	1 chiavi
Testo cifrato 10	1 chiavi
Testo cifrato 11	4 chiavi

in media (2+1+1+4)/4 = 2 chiavi

cioè $\frac{\#chiavi}{\#y} = \frac{8}{4} = 2$ chiavi

DES Doppio: attacco *meet in the middle*

Dato x, qual è il numero medio di chiavi (k_1,k_2) per un testo cifrato

$$DES_{k_2}(DES_{k_1}(x)) ?$$

Fissato x, ci sono 2^{112} chiavi e 2^{64} testi cifrati y

$$\frac{\#\text{chiavi}}{\#y} = \frac{2^{112}}{2^{64}} = 2^{48}$$

DES Doppio: attacco *meet in the middle*

Dato x, qual è il numero medio di chiavi (k_1, k_2) per un testo cifrato

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

Fissato x, ci sono 2^{112} chiavi e 2^{64}

$$\frac{\#\text{chiavi}}{\#y} = \frac{2^{112}}{2^{64}} = 2^{48}$$

2^{48} sono troppi

Come è possibile modificare l'attacco per avere un numero più piccolo?

DES Doppio: attacco *meet in the middle*

Dato x, qual è il numero medio di chiavi (k_1,k_2) per un testo cifrato

$$DES_{k_2}(DES_{k_1}(x)) ?$$

Fissato x, ci sono 2^{112} chiavi e 2^{64} testi cifrati y

$$\frac{\#\text{chiavi}}{\#y} = \frac{2^{112}}{2^{64}} = 2^{48}$$

Uso una seconda coppia (x',y') per verificare la chiave trovata!

DES Doppio: attacco *meet in the middle*

Known Plaintext Attack

Input: $x, y = DES_{k_2}(DES_{k_1}(x))$

$x', y' = DES_{k_2}(DES_{k_1}(x'))$

Costruisci tabella

chiave	testo cifrato
00…00	$DES_{00\ldots00}(x)$
00…01	$DES_{00\ldots01}(x)$
…	…
11…11	$DES_{11\ldots11}(x)$

for $k_b \in \{0,1\}^{56}$

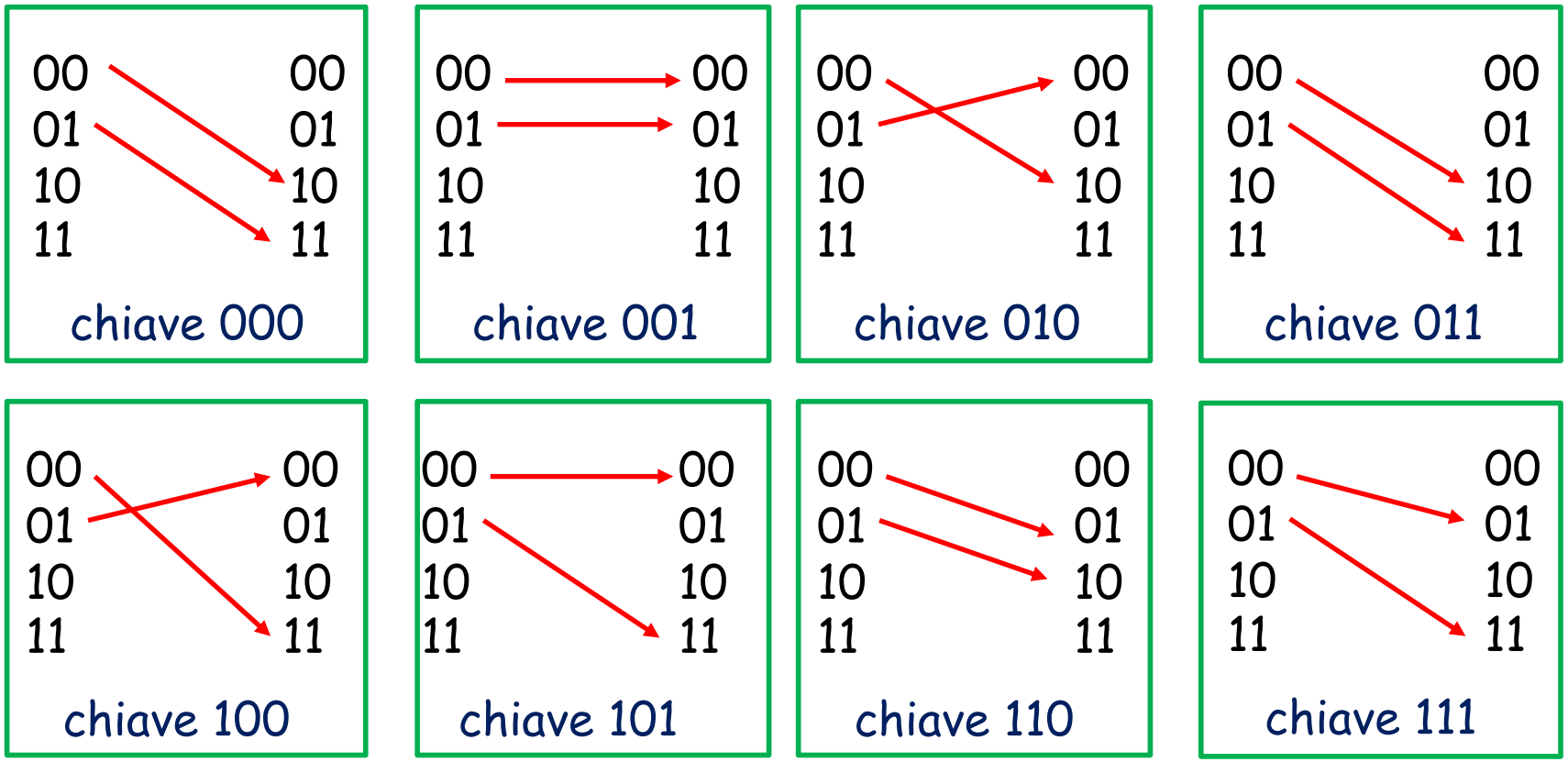
do $z = DES^{-1}{}_{k_b}(y)$

if per qualche k_a, (k_a, z) è nella tabella

e $y' = DES_{k_b}(DES_{k_a}(x'))$

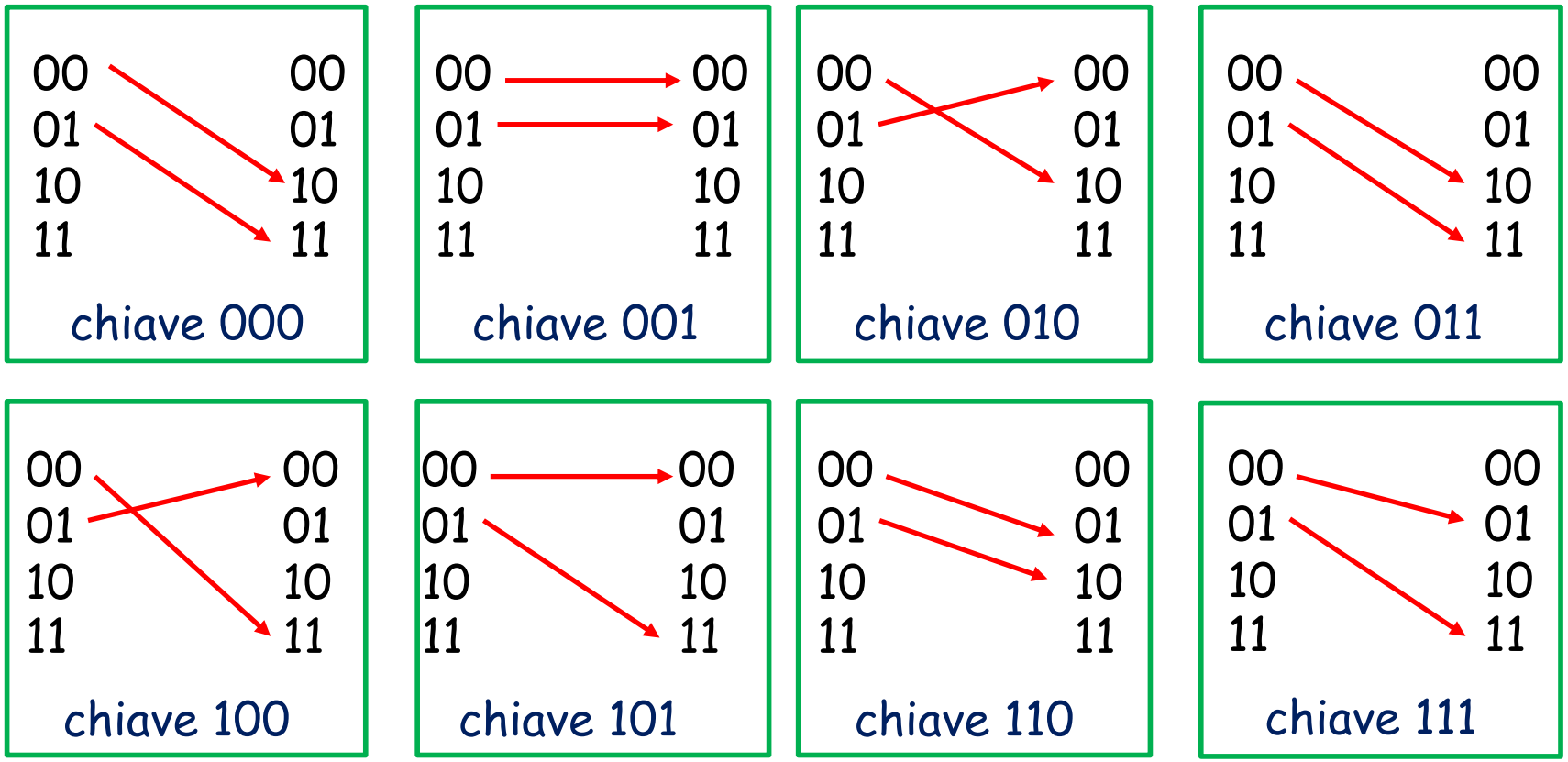
then return la chiave è (k_a, k_b)

Esempio semplificato per il calcolo



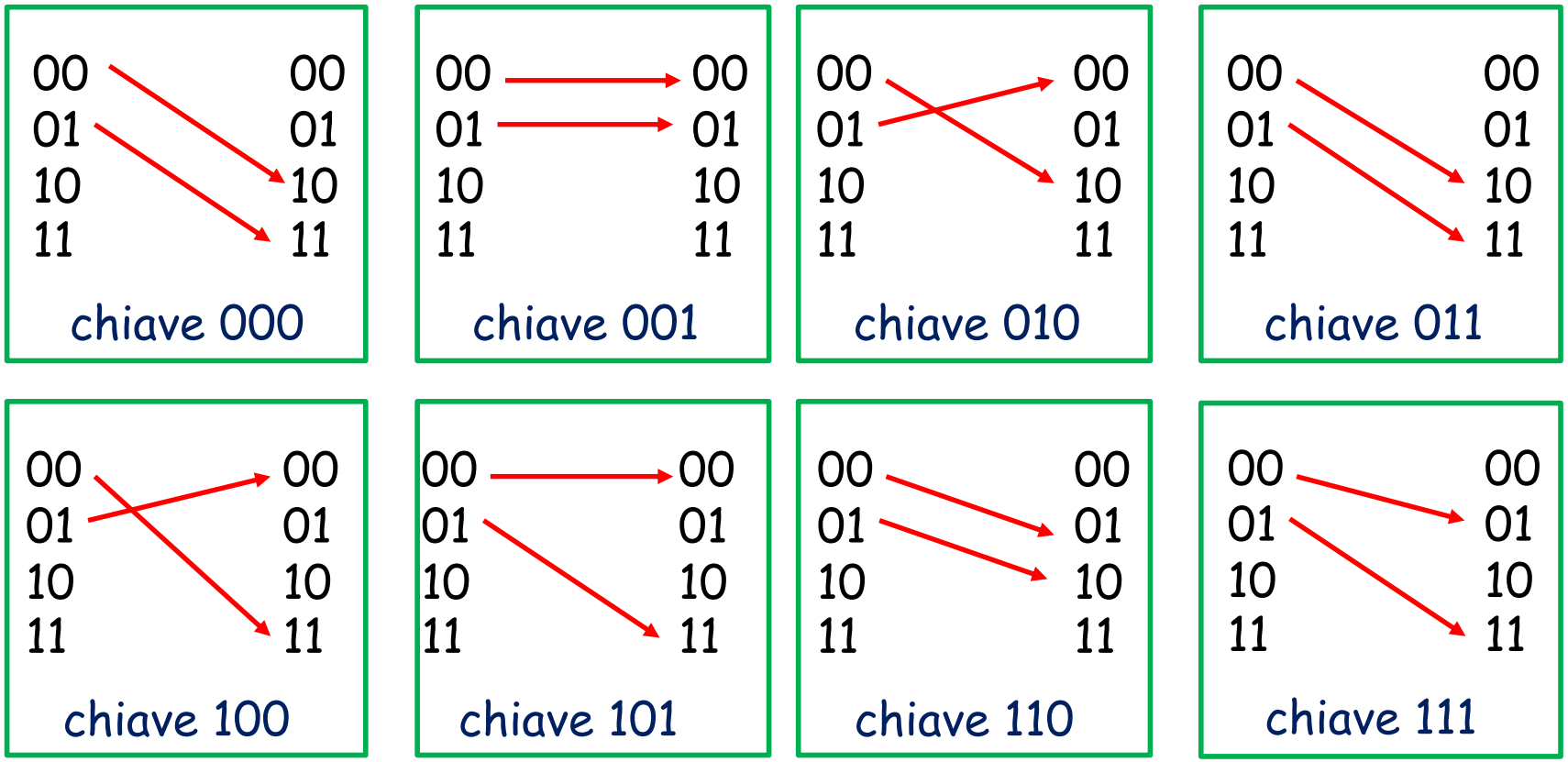
Testi cifrati 00 01	1 chiave
Testo cifrati 00 11	1 chiave
Testo cifrati 01 10	1 chiave
Testo cifrati 01 11	1 chiave
Testo cifrati 10 11	2 chiavi
Testo cifrati 10 00	1 chiave
Testo cifrati 11 00	1 chiave
Altri 9 coppie testi cifrati	0 chiavi

Esempio semplificato per il calcolo



Dati 01 e 00 qual'è il numero medio di chiavi k per coppia valori cifrati ?

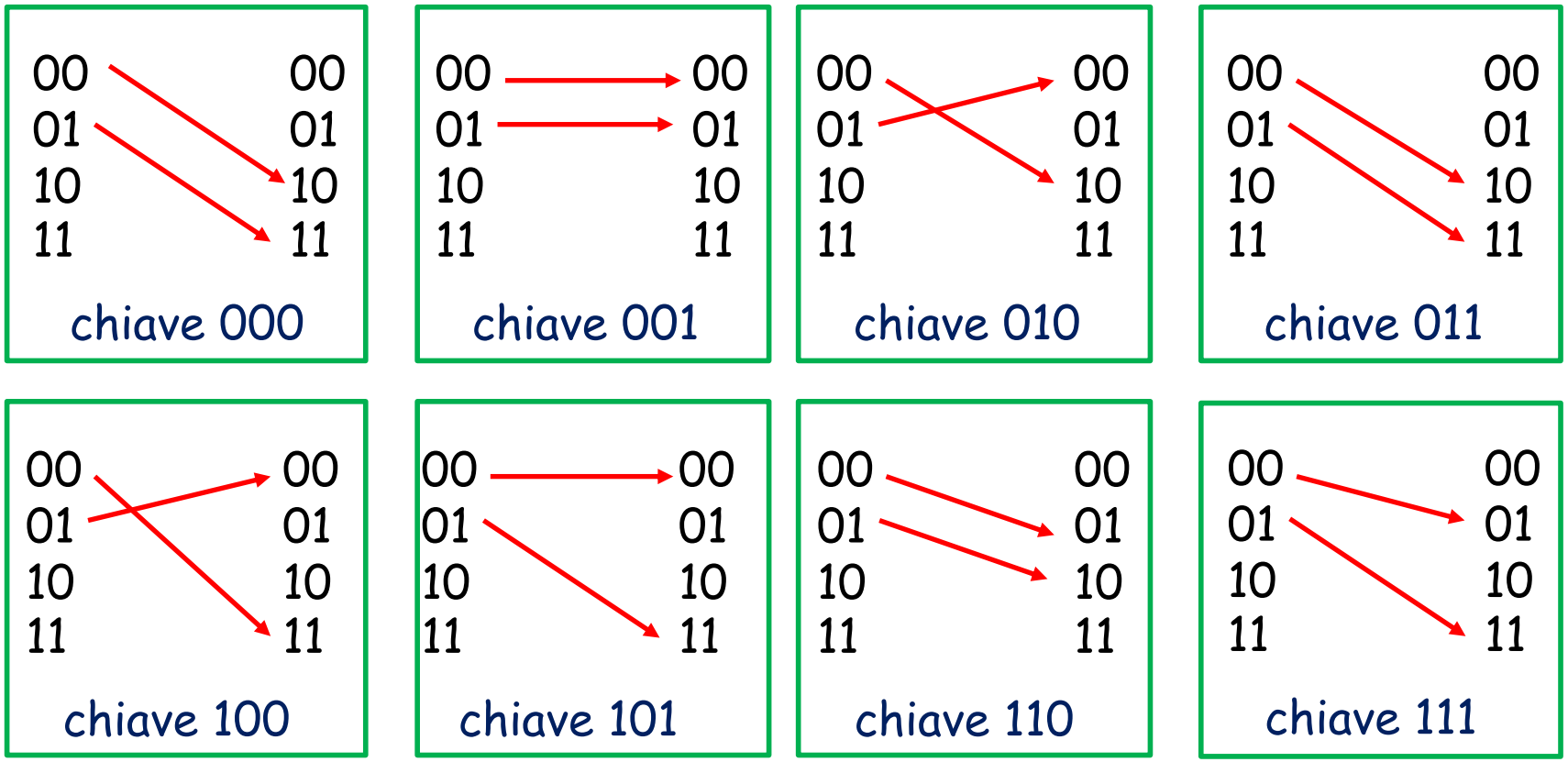
Esempio semplificato per il calcolo



Testi cifrati 00 01	1 chiave
Testo cifrati 00 11	1 chiave
Testo cifrati 01 10	1 chiave
Testo cifrati 01 11	1 chiave
Testo cifrati 10 11	2 chiavi
Testo cifrati 10 00	1 chiave
Testo cifrati 11 00	1 chiave
Altri 9 coppie testi cifrati	0 chiavi

in media (1+1+1+1+2+1+1)/16 = 1/2 chiavi

Esempio semplificato per il calcolo



Testi cifrati 00 01	1 chiave
Testo cifrati 00 11	1 chiave
Testo cifrati 01 10	1 chiave
Testo cifrati 01 11	1 chiave
Testo cifrati 10 11	2 chiavi
Testo cifrati 10 00	1 chiave
Testo cifrati 11 00	1 chiave
Altri 9 coppie testi cifrati	0 chiavi

in media (1+1+1+1+2+1+1)/16 = 1/2 chiavi

cioè $\frac{\#chiavi}{\#\ y,y'} = \frac{8}{16} = \frac{1}{2}$ chiavi

DES Doppio: attacco *meet in the middle*

Dati x, y, qual è il numero medio di chiavi (k_1,k_2) per i testi cifrati

$$y = DES_{k_2}(DES_{k_1}(x))$$

$$y' = DES_{k_2}(DES_{k_1}(x'))\ ?$$

Fissati x, x´, ci sono 2^{112} chiavi e 2^{128} testi cifrati y, y´

$$\frac{\#\text{chiavi}}{\#y,y'} = \frac{2^{112}}{2^{128}} = 2^{-16}$$

DES Doppio: attacco *meet in the middle*

Esercizio:

Supponiamo di avere tre coppie

$$y = DES_{k_2}(DES_{k_1}(x))$$
$$y' = DES_{k_2}(DES_{k_1}(x'))$$
$$y'' = DES_{k_2}(DES_{k_1}(x''))$$

- Come si modifica l'attacco *meet in the middle*?
- Che complessità ha l'algoritmo?
- Che vantaggio si ottiene?

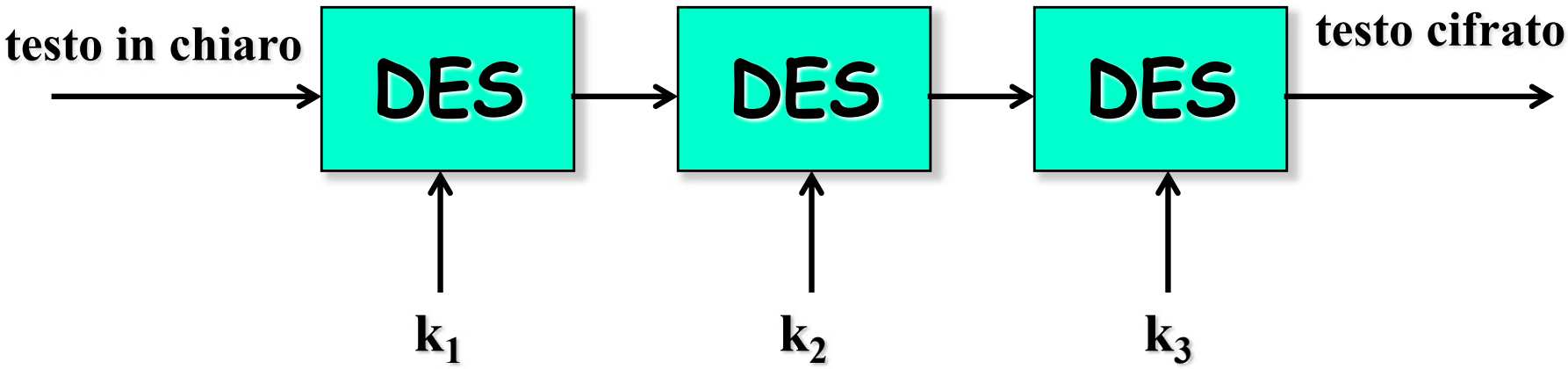
DES Doppio: attacco *meet in the middle*

Complessità *Known Plaintext Attack* $\approx 2^{56}$

Ricerca esaustiva su tutte le chiavi $\approx 2^{112}$

"Equivalente" ad un cifrario con una chiave di 56 bit, e non 112 bit

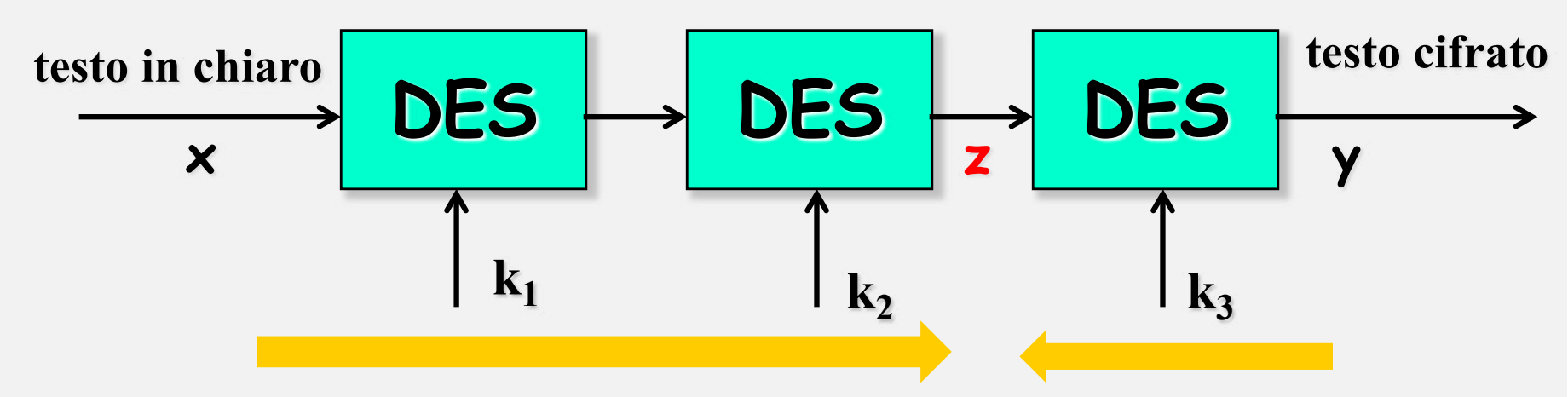
DES Triplicato



- lunghezza blocco = 64 bit
- chiave (k_1,k_2,k_3) lunga 56 + 56 + 56 = 168 bit

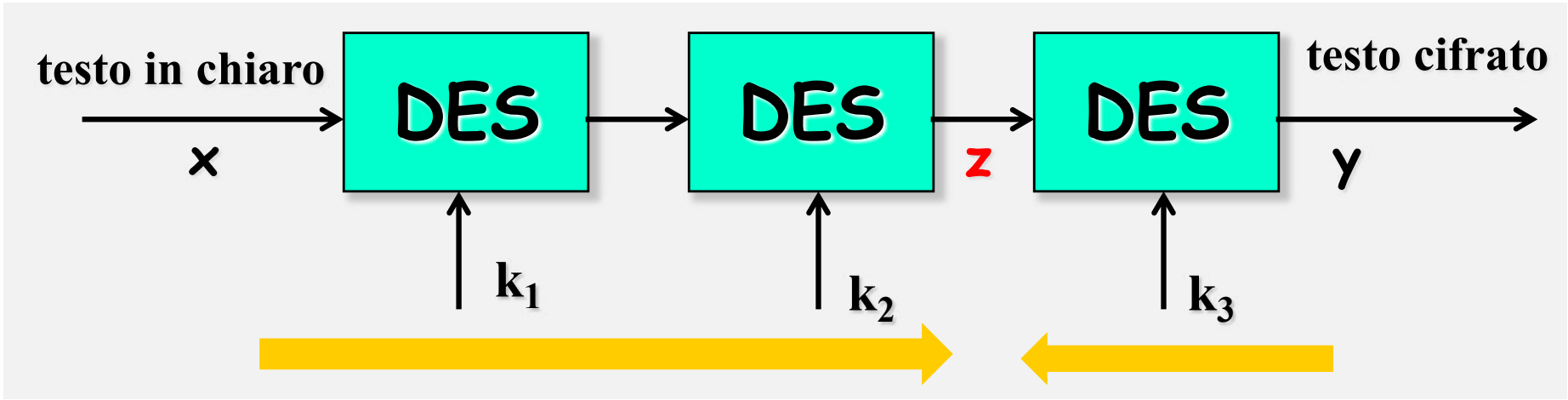
DES Triplicato: *attacco meet in the middle*

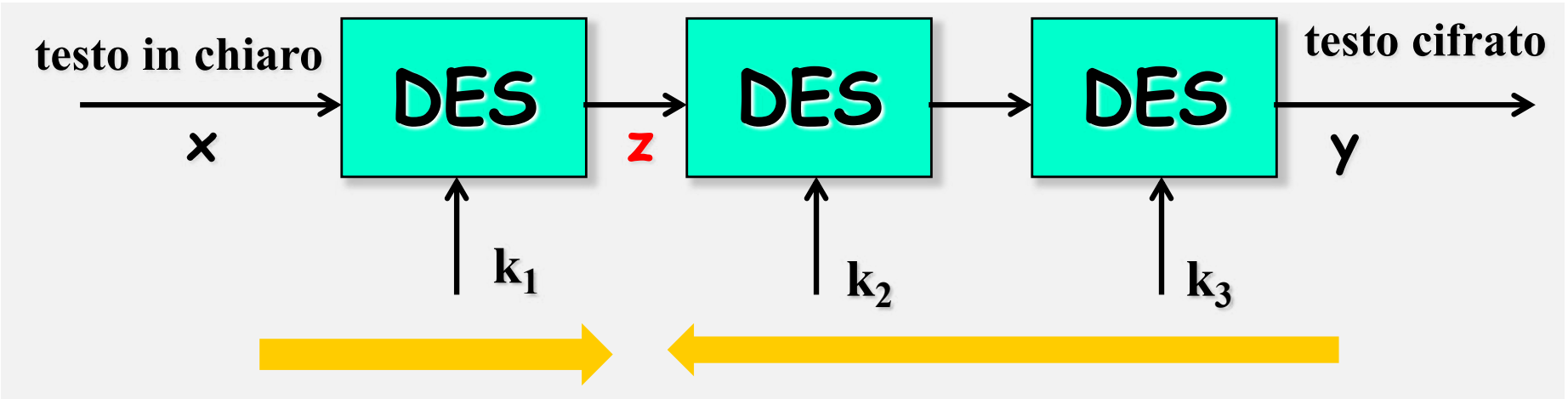
L'attacco può essere eseguito in 2 punti "centrali":



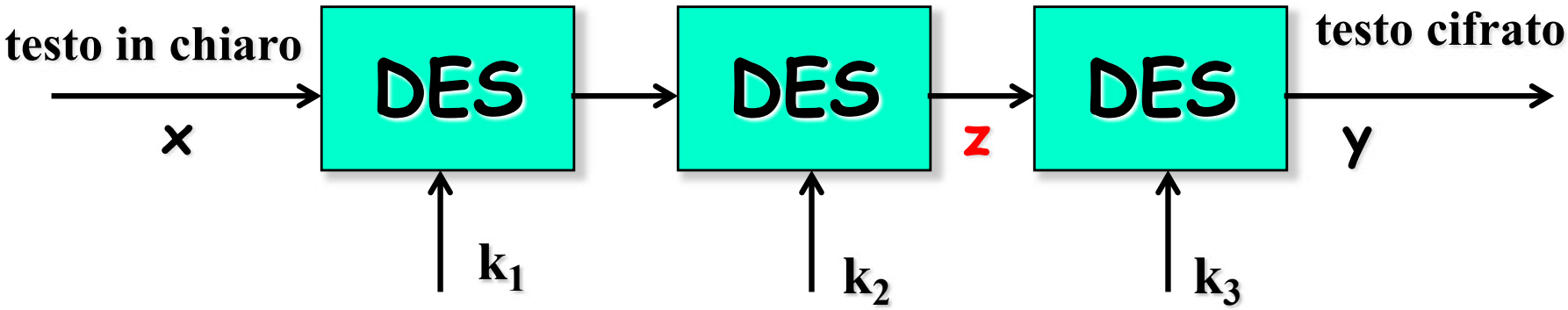
DES Triplicato: *attacco meet in the middle*

L'attacco può essere eseguito in 2 punti "centrali":





DES Triplicato: *attacco meet in the middle*



Known Plaintext Attack

Input: x, y = $DES_{k_3}(DES_{k_2}(DES_{k_1}(x)))$

Costruisci tabella

chiave	testo cifrato
00…00	$DES_{00…00}(DES_{00…00}(x))$
00…01	$DES_{00…00}(DES_{00…01}(x))$
…	…
11…11	$DES_{11…11}(DES_{11…11}(x))$

for $k_c \in \{0,1\}^{56}$

do $z = DES^{-1}_{k_c}(y)$

if per qualche k_a, k_b, $(k_a k_b, z)$ è nella tabella

then return la chiave è (k_a, k_b, k_c)

DES Triplicato: attacco *meet in the middle*

Known Plaintext Attack

Input: x, $y = DES_{k_3}(DES_{k_2}(DES_{k_1}(x)))$

Costruisci tabella

chiave	testo cifrato
00...00	$DES_{00...00}(DES_{00...00}(x))$
00...01	$DES_{00...00}(DES_{00...01}(x))$
...	...
11...11	$DES_{11...11}(DES_{11...11}(x))$

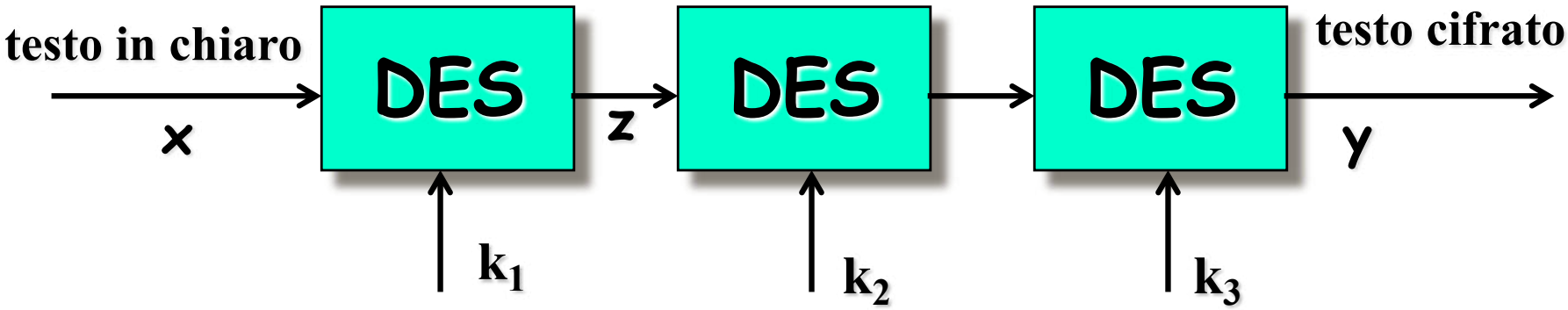
for $k_c \in \{0,1\}^{56}$
  do $z = DES^{-1}_{k_c}(y)$
    if per qualche k_a, k_b, $(k_a k_b, z)$ è nella tabella
      then return la chiave è (k_a, k_b, k_c)

Complessità spazio: 2^{112} righe nella tabella

Complessità tempo:

- 2^{112} cifrature per x (costruzione tabella)
- 2^{56} decifrature per y
- 2^{56} ricerche in tabella

DES Triplicato: attacco *meet in the middle*



Known Plaintext Attack

Input: x, y = $DES_{k_3}(DES_{k_2}(DES_{k_1}(x)))$

Costruisci tabella

chiave	testo cifrato
00...00	$DES_{00...00}(x)$
00...01	$DES_{00...01}(x)$
...	...
11...11	$DES_{11...11}(x)$

for $k_c, k_b \in \{0,1\}^{56} \times \{0,1\}^{56}$

do $z = DES^{-1}_{k_c}(DES^{-1}_{k_c}(y))$

if per qualche k_a, (k_a, z) è nella tabella

then return la chiave è (k_a, k_b, k_c)

DES Triplicato: attacco *meet in the middle*

Known Plaintext Attack

Input: x, $y = DES_{k_3}(DES_{k_2}(DES_{k_1}(x)))$

Costruisci tabella

chiave	testo cifrato
00…00	$DES_{00\ldots00}(x)$
00…01	$DES_{00\ldots01}(x)$
…	…
11…11	$DES_{11\ldots11}(x)$

for $k_c,k_b \in \{0,1\}^{56} \times \{0,1\}^{56}$

do $z = DES^{-1}_{k_b}(DES^{-1}_{k_c}(y))$

if per qualche k_1, (k_1, z) è nella tabella

then return la chiave è (k_a,k_b,k_c)

Complessità spazio: 2^{56} righe nella tabella

Complessità tempo:

- 2^{56} cifrature per x (costruzione tabella)
- 2^{112} decifrature per y
- 2^{112} ricerche in tabella

DES Triplicato: attacco *meet in the middle*

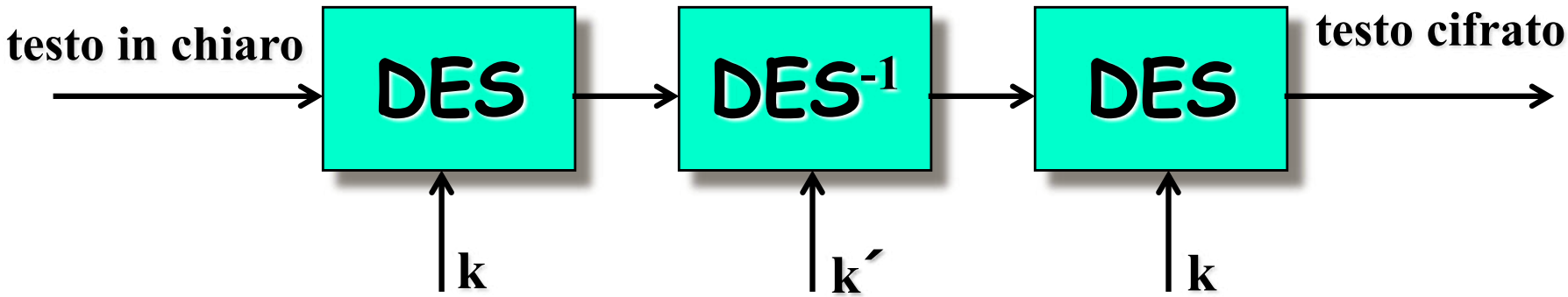
- Complessità *Known Plaintext Attack* $\approx 2^{112}$
- Ricerca esaustiva su tutte le chiavi $\approx 2^{168}$

DES Triplicato: attacco *meet in the middle*

Complessità *Known Plaintext Attack* $\approx 2^{112}$

"Equivalente" ad un cifrario con una chiave di 112 bit, e non 168 bit

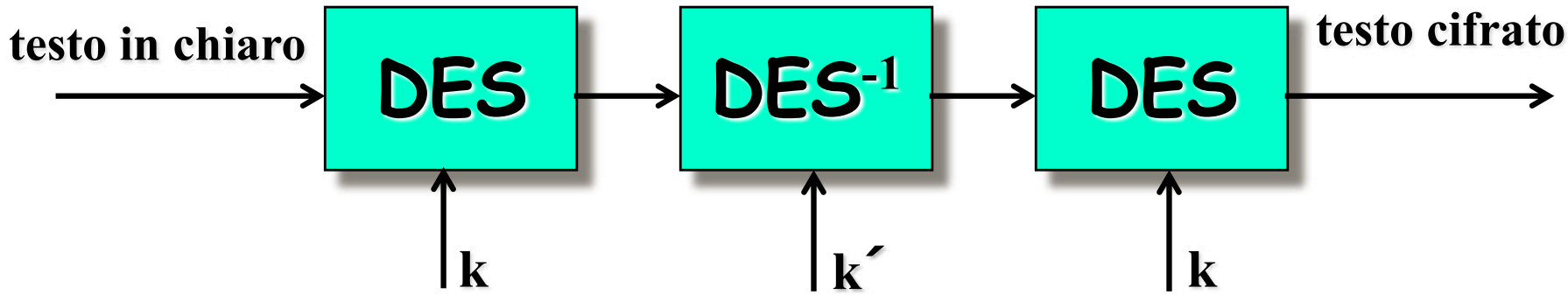
DES Triplo



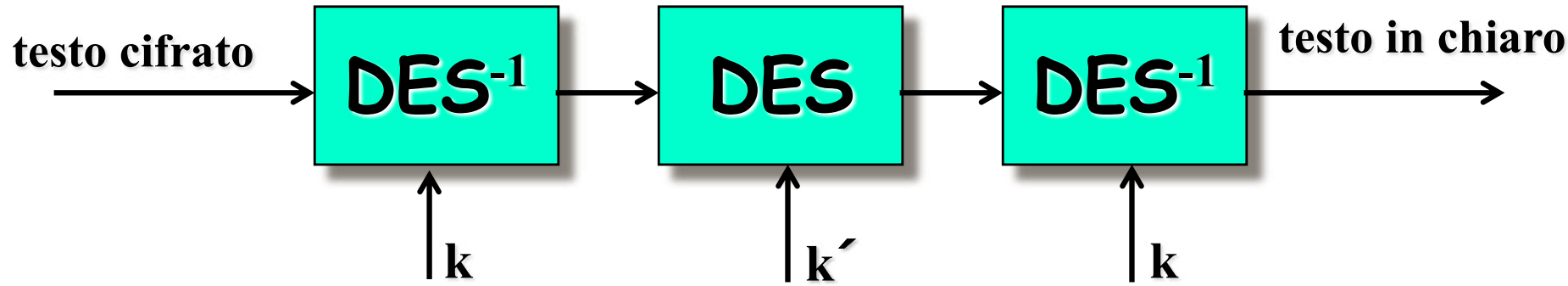
- lunghezza blocco = 64 bit
- chiave (k, k´) lunga 56+56 = 112 bit
- spesso chiamato $EDE_{k,k'}$ (acronimo per Encrypt Decrypt Encrypt)
- adottato negli standard X9.17 e ISO 8732

Decifratura DES Triplo

Cifratura



Decifratura



Termini

- **Triple DES**
- **TDEA** Triple Data Encryption Algorithm, Triple DEA

Entrambi i termini sono in FIPS PUB 46-3 (1999)

- **3TDEA** Three key TDEA
- **2TDEA** Two key TDEA
  - NIST SP 800-57 "Recommendation for Key Management"
  - NIST SP 800-78 "Cryptographic Algorithms and Key Sizes for Personal Identity Verification"

Security lifetime

Recommended algorithms and minimum key sizes

Algorithm security lifetimes	Symmetric key algorithms (Encryption & MAC)
Through 2010 (min. of 80 bits of strength)	2TDEA[23] 3TDEA AES-128 AES-192 AES-256
Through 2030 (min. of 112 bits of strength)	3TDEA AES-128 AES-192 AES-256
Beyond 2030 (min. of 128 bits of strength)	AES-128 AES-192 AES-256

NIST SP 800-57
"Recommendation for Key Management – Part 1: General", March 2007

Sicurezza contro attacchi "Brute Force"

Grandezza chiavi (bits)	Numero di possibili chiavi	Tempo medio necessario con 1 decifratura/μs	Tempo medio necessario con 10^6 decifrature/μs
32	$2^{32} \approx 4{,}3 \times 10^9$	2^{31} μs = 35,8 minuti	2,15 millisecondi
56	$2^{56} \approx 7{,}2 \times 10^{16}$	2^{55} μs = 1142 anni	10,01 ore
112	$2^{112} \approx 5{,}19 \times 10^{33}$	2^{111} μs = $8{,}2 \times 10^{19}$ anni	$8{,}2 \times 10^{13}$ anni
128	$2^{128} \approx 3{,}4 \times 10^{38}$	2^{127} μs = $5{,}4 \times 10^{24}$ anni	$5{,}4 \times 10^{18}$ anni
168	$2^{168} \approx 3{,}7 \times 10^{50}$	2^{167} μs = $5{,}9 \times 10^{36}$ anni	$5{,}9 \times 10^{30}$ anni
256	$2^{256} \approx 1{,}15 \times 10^{77}$	2^{255} μs = $1{,}83 \times 10^{63}$ anni	$1{,}83 \times 10^{57}$ anni

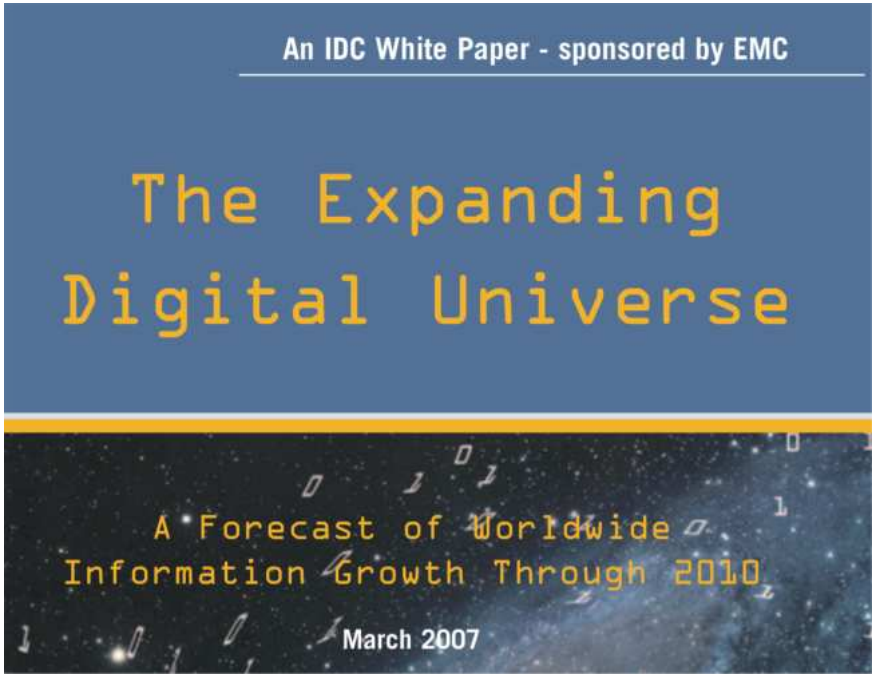
Stima età universo 13,798 ± 0,037 miliardi anni $\approx 4 \times 10^{17}$ secondi
$\approx 4 \times 10^{23}$ microsecondi

(atre stime 13-15 miliardi anni)

Universo digitale in espansione

Nel 2006 informazione digitale creata e catturata nel mondo: 161 exabyte

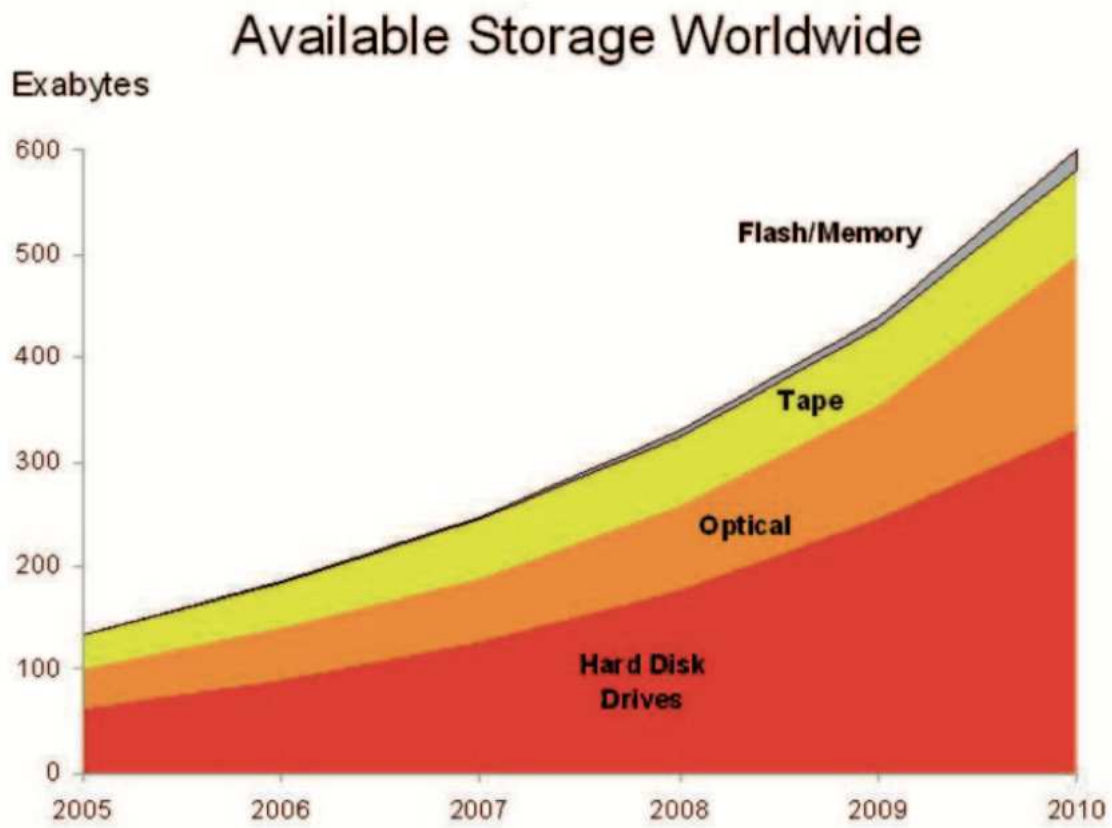
Bit (b)	1 or 0
Byte (B)	8 bits
Kilobyte (KB)	1,000 bytes
Megabyte (MB)	1,000 KB
Gigabyte (GB)	1,000 MB
Terabyte (TB)	1,000, GB
Petabyte (PB)	1,000 TB
Exabyte (EB)	1,000 PB
Zettabyte (ZB)	1,000 EB



https://www.tobb.org.tr/BilgiHizmetleri/Documents/Raporlar/Expanding_Digital_Universe_IDC_WhitePaper_022507.pdf

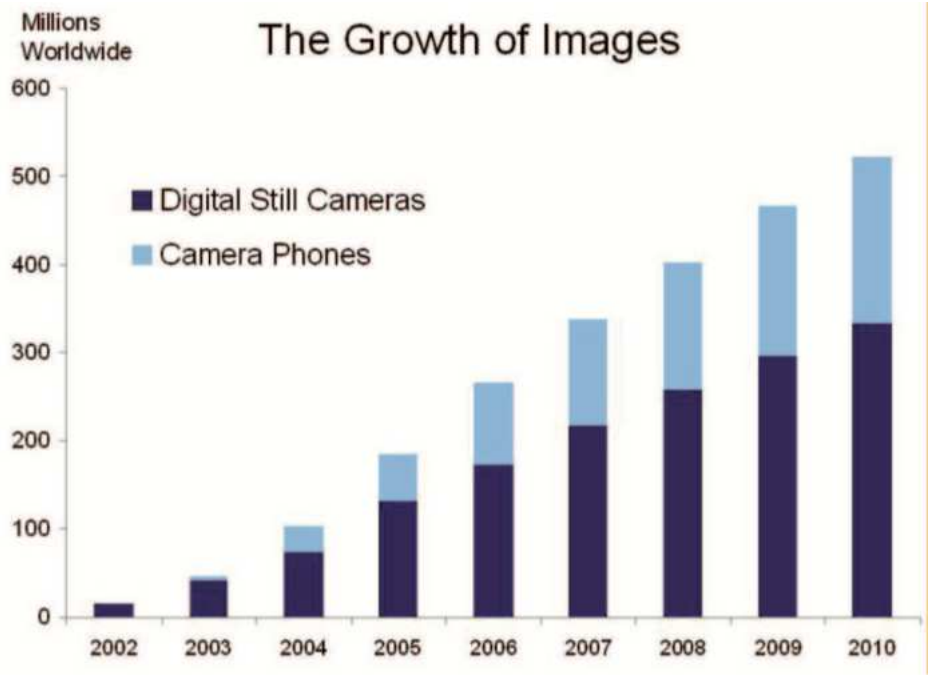
Universo digitale in espansione

- Nel 2006 informazione digitale creata e catturata nel mondo: 161 exabyte
- Previsione per il 2010: 988 exabyte

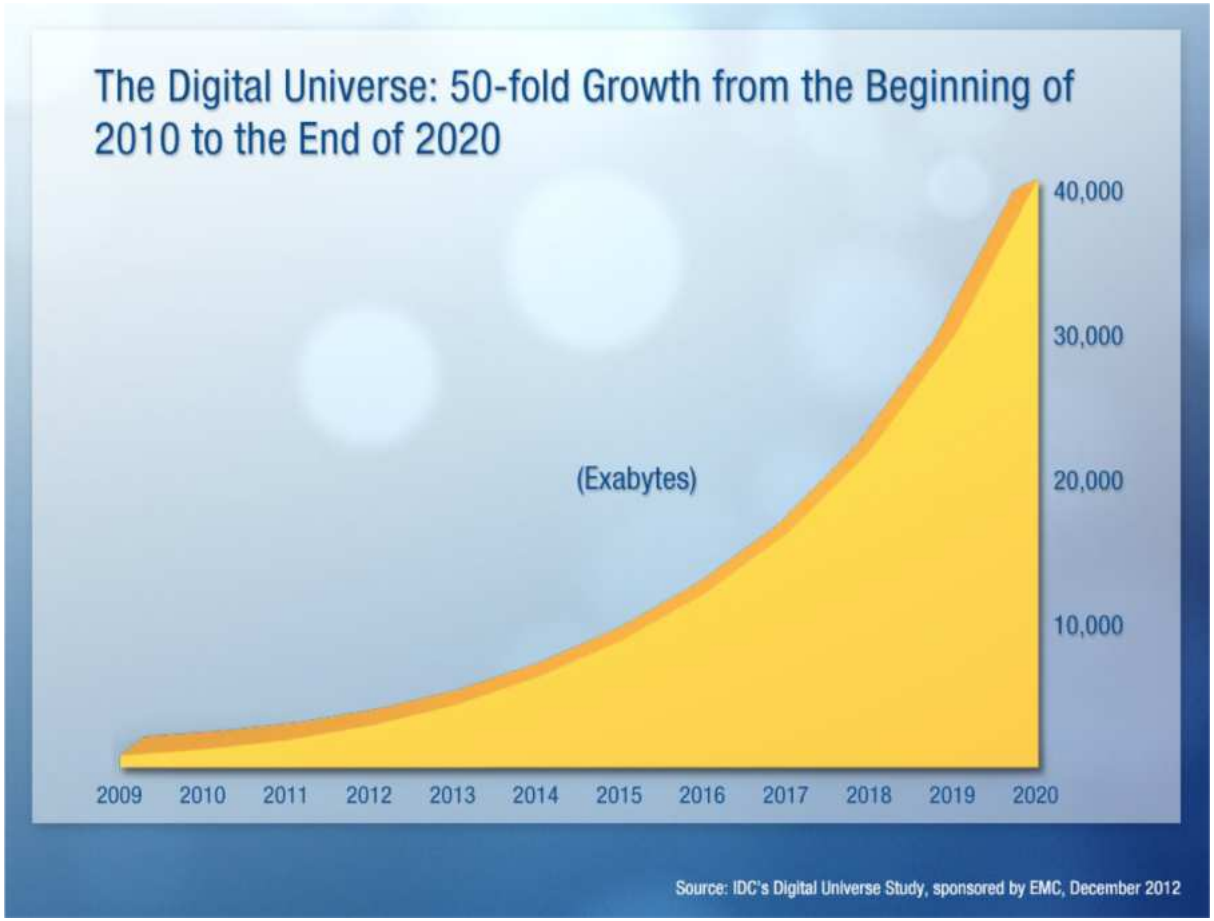


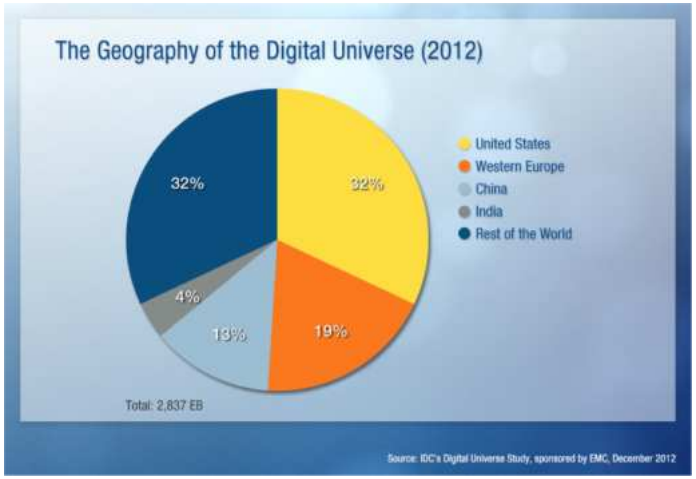
Universo digitale in espansione

Alcune stime:



Universo digitale in espansione

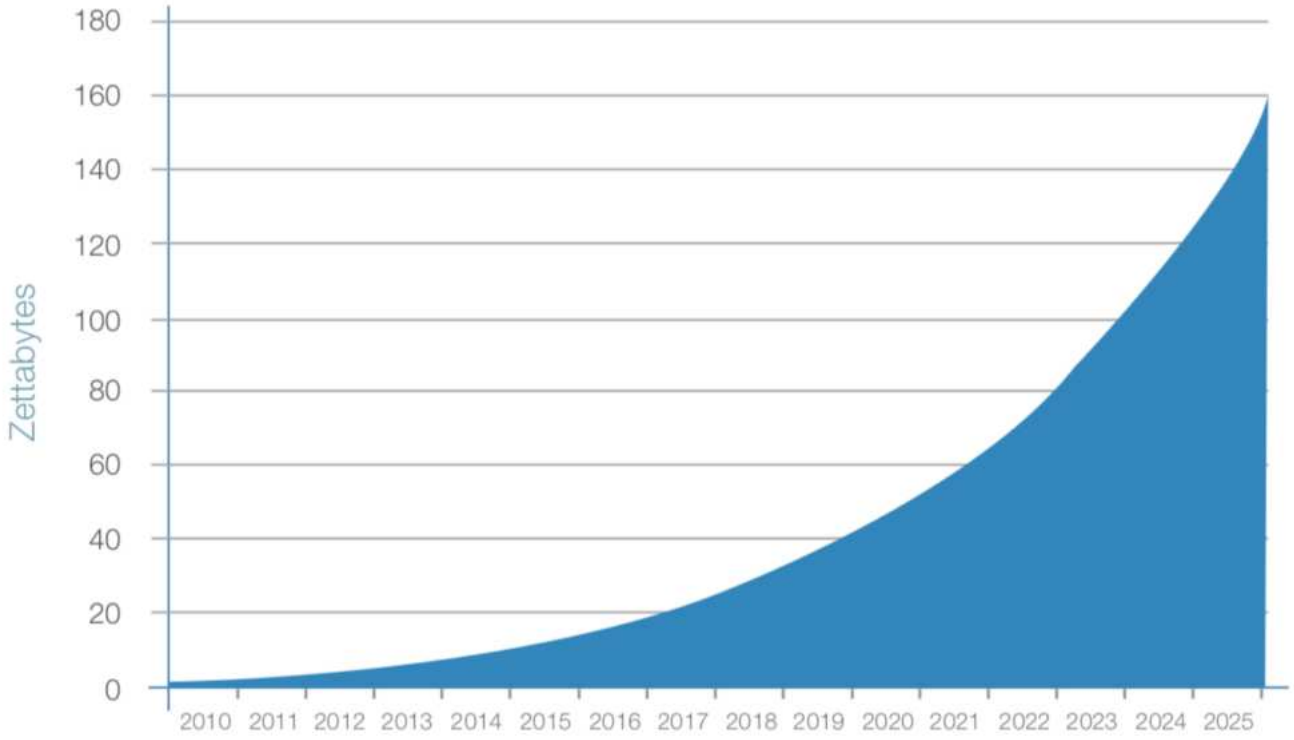




Exabyte = 10^{18} bytes

40.000 exabytes = $4 \times 10^{22} \approx 4{,}06 \times 2^{75}$

Universo digitale in espansione

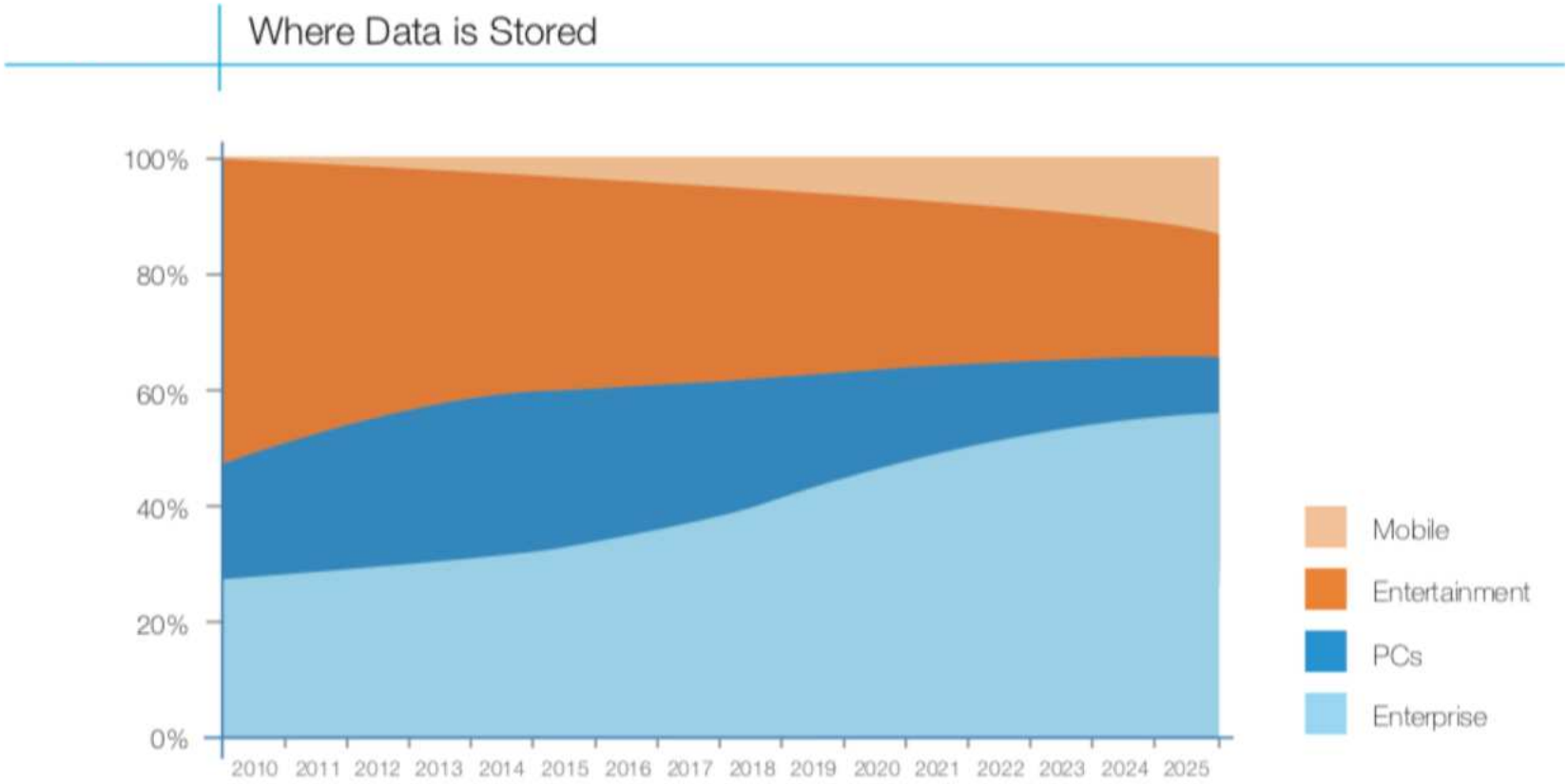


Source: IDC's Data Age 2025 study, sponsored by Seagate, April 2017

Zettabyte = 10^{21} bytes

163 zettabytes = $1{,}63 \times 10^{23} \approx 1{,}08 \times 2^{77}$

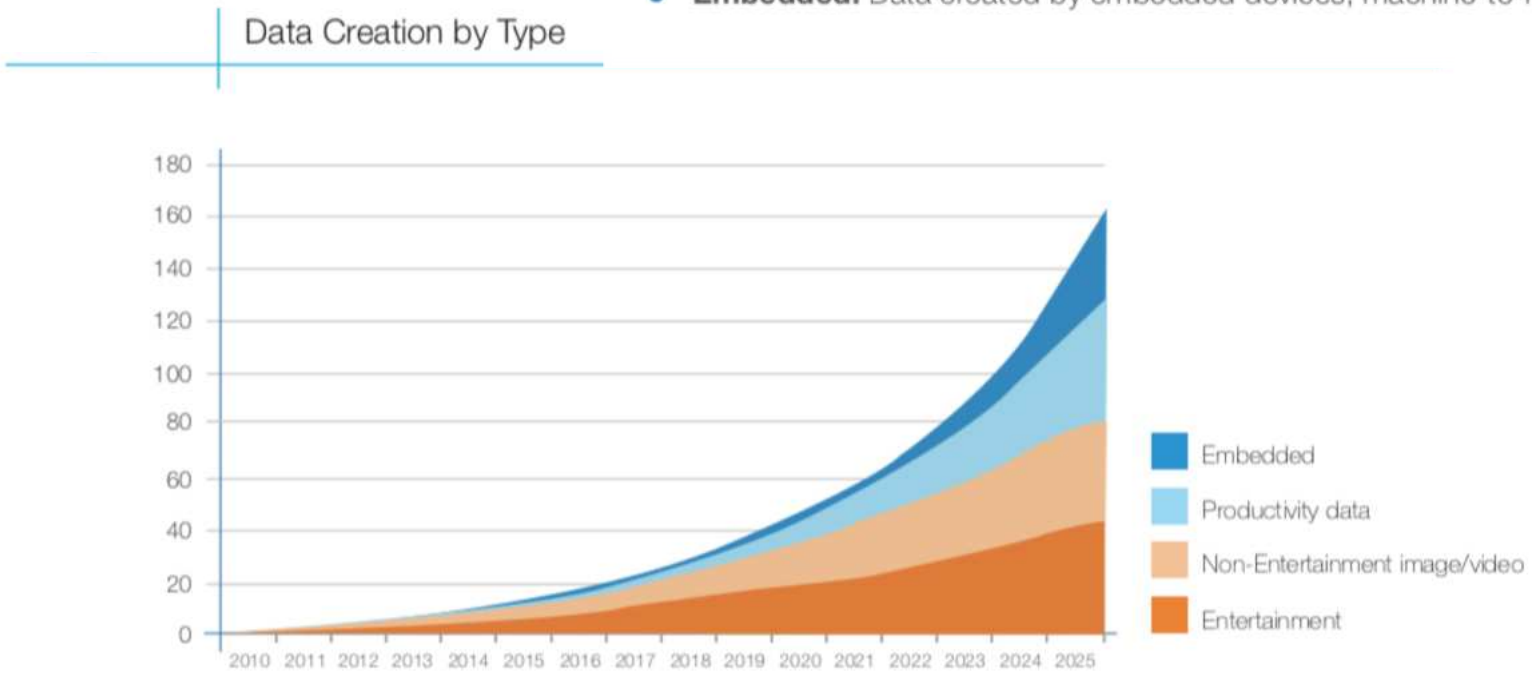
Universo digitale in espansione



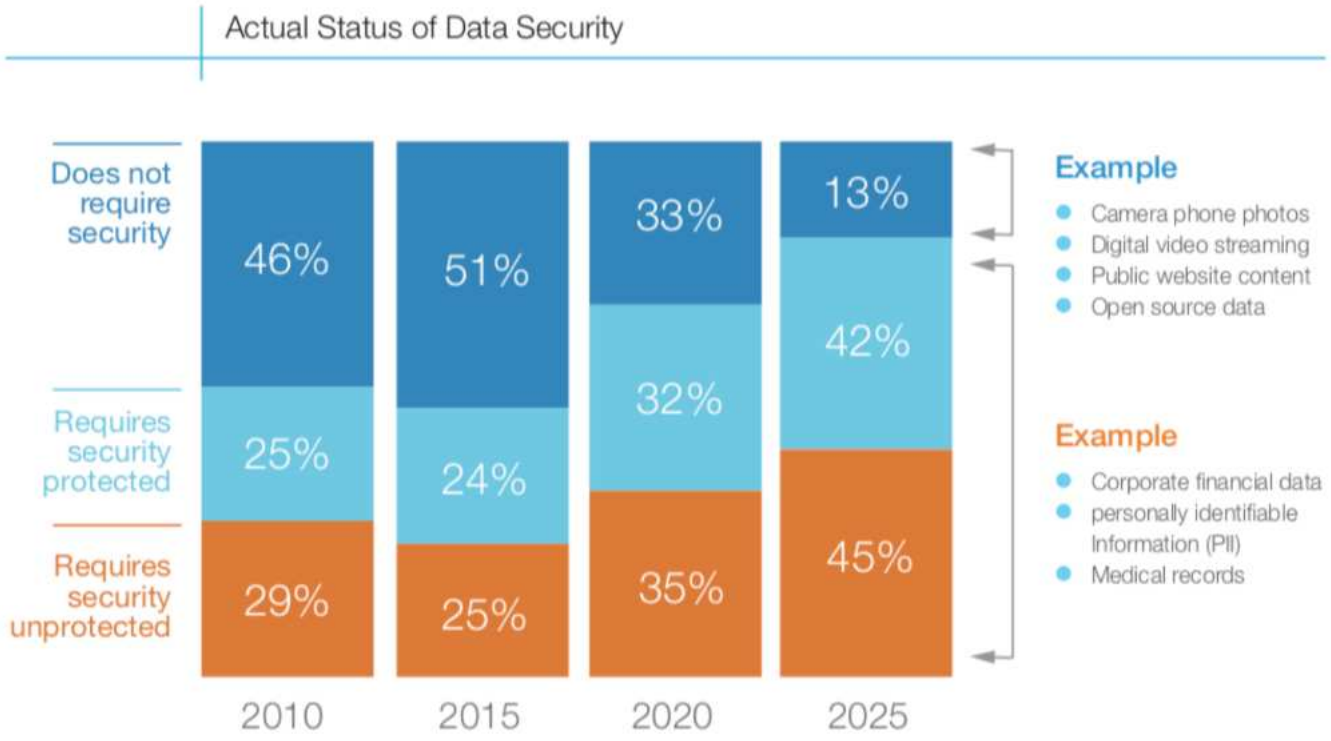
- Nel 2010 DVD e Blu-ray
- Mercato consumer per video poi servizi streaming

Universo digitale in espansione

- **Entertainment.** Image and video content created or consumed for entertainment purposes.
- **Non-entertainment image/video.** Image and video content for non-entertainment purposes, such as video surveillance footage or advertising.
- **Productivity data.** Traditional productivity-driven data such as files on PCs and servers, log files, and metadata.
- **Embedded.** Data created by embedded devices, machine-to-machine, and IoT.



Universo digitale in espansione



Limiti fisici

- Quanta informazione può essere mai memorizzata?
- Quante operazioni possono essere effettuate?

Computational capacity of the universe

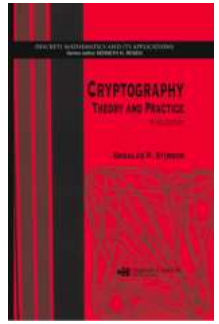
Seth Lloyd

Merely by existing, all physical systems register information. And by evolving dynamically in time, they transform and process that information. The laws of physics determine the amount of information that a physical system can register (number of bits) and the number of elementary logic operations that a system can perform (number of ops). The universe is a physical system. This paper quantifies the amount of information that the universe can register and the number of elementary operations that it can have performed over its history. The universe can have performed no more than 10^{120} ops on 10^{90} bits.

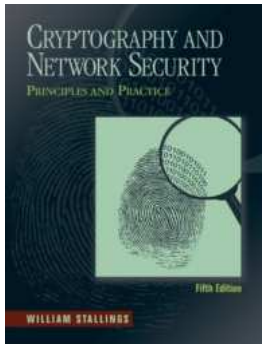
24 Oct 2001 https://arxiv.org/pdf/quant-ph/0110141.pdf
https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.88.237901

Bibliografia

- Cryptography: Theory and Practice
  D. Stinson (1995)
  - cap. 3



- Cryptography and Network Security
  by W. Stallings, 2010
  - cap. 3 (DES), cap. 6 (modalità operative, cifrature multiple)



- Tesina di Sicurezza su reti
  - Data Encryption Standard

Domande?