

---

Prof. Maurizio Dècina

APPUNTI SULLE  
**COMUNICAZIONI SICURE**

# LARGE NUMBERS

Potenze di 2	$2^{10}$	————→	$\sim 10^3$	
	$2^{16}$	————→	$\sim 6,55$	$10^4$
	$2^{32}$	————→	$\sim 4,3$	$10^9$
	$2^{48}$	————→	$\sim 2,8$	$10^{14}$
	$2^{56}$	————→	$\sim 7,2$	$10^{16}$
	$2^{64}$	————→	$\sim 1,8$	$10^{19}$
	$2^{100}$	————→	$\sim 1,3$	$10^{30}$
	$2^{128}$	————→	$\sim 3,4$	$10^{38}$
	$2^{200}$	————→	$\sim 1,6$	$10^{60}$
	$2^{256}$	————→	$\sim 1,2$	$10^{77}$
	$2^{512}$	————→	$\sim 1,3$	$10^{154}$
	$2^{750}$	————→	$\sim 5,9$	$10^{225}$
	$2^{1024}$	————→	$\sim 1,8$	$10^{308}$

$$\frac{\# \text{ Cifre Esp. Binario}}{\log_2 10} = \# \text{ Cifre Esp. Decimale } [\log_2 10 \cong 3,33]$$

Secondi	1 h	3,6	$10^3$
	1 g	8,64	$10^4$
	1 mese	$\sim 2,6$	$10^6$
	1 anno	$\sim 3,15$	$10^7$
	10 anni	$\sim 3,15$	$10^8$
	100 anni	$\sim 3,15$	$10^9$
	1000 anni	$\sim 3,15$	$10^{10}$

Servono metà combinazioni in media per risolvere

Esempio Chiave da 56 bit      combinazioni =  $2^{56} \cong 7,2 \times 10^{16}$

$7,2 \times 10^{16} \times 10^{-9} = \sim 7,2 \times 10^7 \sim 2$  anni per provarle tutte

$10^{-9}$  s per combinazione

1 ns per combinazione [ $>1.000$  MIPS in multiprocessing]

---

# LARGE NUMBERS

---

- Probabilità di venire ucciso in un incidente automobilistico  
(negli U.S., per anno) 1 su 5.600 ( $2^{-12}$ )

- Età della Terra  $10^9$  ( $2^{30}$ ) anni

- Età dell'Universo  $10^{10}$  ( $2^{34}$ ) anni

Se l'Universo è Chiuso

- Vita totale dell'Universo,  
durata  $10^{11}$  ( $2^{37}$ ) anni  
 $10^{18}$  ( $2^{61}$ ) secondi

Se l'Universo è aperto

- Tempo affinché tutta la materia si liquidi  
a temperatura zero  $10^{65}$  ( $2^{216}$ ) anni  
 $10^{72}$  ( $2^{240}$ ) secondi
- Numero di atomi della Terra  $10^{51}$  ( $2^{170}$ )
- Numero di atomi  
nella Galassia  $10^{67}$  ( $2^{223}$ )  
nell'Universo  $10^{77}$  ( $2^{265}$ )
- Volume dell'Universo  $10^{84}$  ( $2^{280}$ )  $\text{cm}^3$

---

$n$  oggetti diversi  $\rightarrow$  presi a gruppi di  $k$

PERMUTAZIONI  $(k = n)$   $n!$

DISPOSIZIONI  $\frac{n!}{(n-k)!}$

DISPOSIZIONI CON RIPETIZIONE  $n^k$

COMBINAZIONI  $\binom{n}{k}$

COMBINAZIONI CON RIPETIZIONE  $\binom{n+k-1}{k}$

$n = 1, 2 \dots$

$k = 1, 2 \dots$

---

FORMULA DI STIRLING

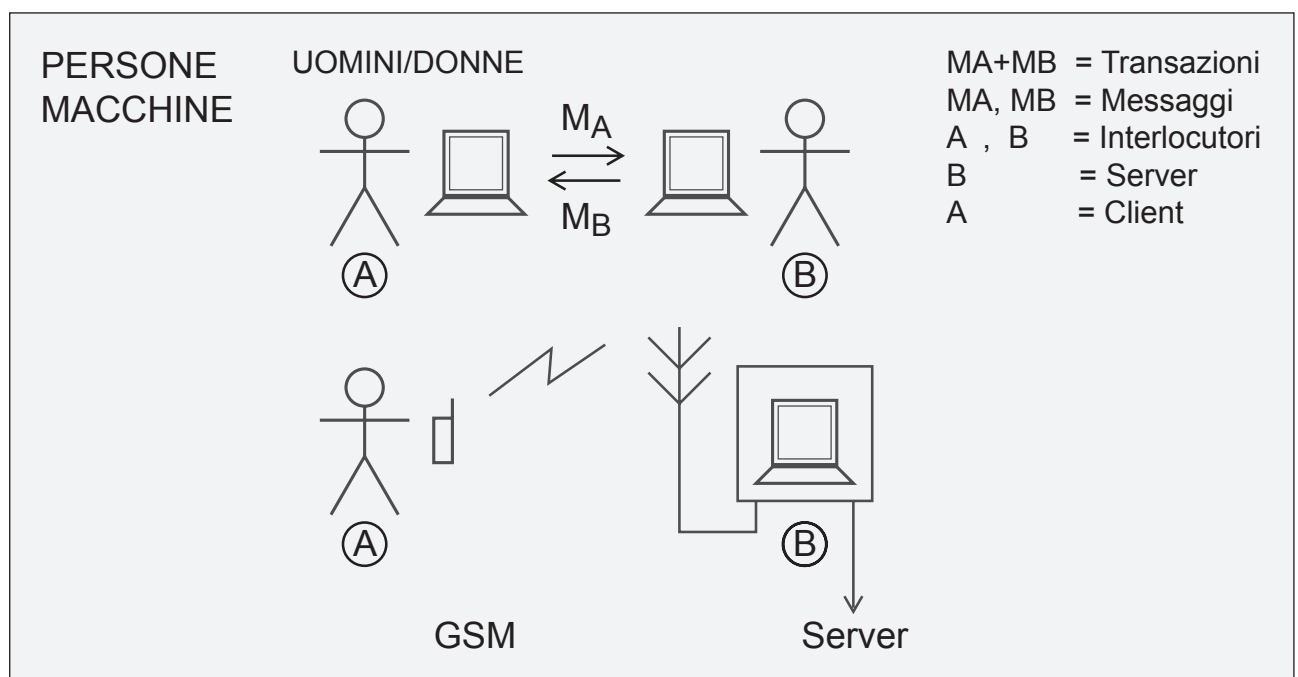
$$n! \simeq \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

$n$  intero  $> 0$ , per  $n \gg 1$

---

# ASPETTI DELLA SICUREZZA

- SEGRETEZZA {DEI MESSAGGI {RISERVATEZZA  
CONFIDENZIALITA'}
- IDENTIFICAZIONE {AUTENTICAZIONE  
DEGLI INTERLOCUTORI} MACCHINE  
PERSONE
- INTEGRITA' {AUTENTICAZIONE DEI MESSAGGI
- FIRMA {DEI MESSAGGI {FIRMA DI PERSONE
- CERTIFICATO {DI IDENTITA' DI PERSONE
- NON RIPUDIO {DELLE TRANSAZIONI
- ANONIMITA' {DELLE TRANSAZIONI
- DISPONIBILITA' {DEL SERVIZIO} {(DENIAL OF SERVICE ATTACK)
- AFFIDABILITA' {DEL SERVIZIO
- AUTORIZZAZIONE {A INTERLOCUTORI IDENTIFICATI}



---

# NUMERI INTERI

---

- Dato  $m$  intero positivo,  $m > 0$ , l'espressione

‘ $m$  divide  $n$ ’

ove  $n$  è intero,  $n \geq 0$ , significa che

‘ $n$  è divisibile per  $m$ ’

in modo perfetto, cioè senza resto:

$$\frac{n}{m} = k \text{ intero} \quad \begin{array}{l} n \geq 0 \\ k \geq 0 \\ m > 0 \end{array}$$

e si scrive

$$m \backslash n$$

Quindi  $n$  è multiplo di  $m$ :

$$n = km, \text{ per qualche intero } k.$$

- Esiste solo un multiplo di  $m = 0$ , cioè zero stesso.

Cioè 0 non ‘divide’ alcun numero, ovvero nessun intero è divisibile per 0.

- Dato  $a$  intero positivo,  $a > 0$ , questi è un numero ‘primo’

soltanto se 1 e  $a$  dividono  $a$  stesso, cioè se  $a$  è divisibile soltanto per 1 e per  $a$  stesso

$$\left. \begin{array}{l} \frac{a}{1} = a \\ \frac{a}{a} = 1 \end{array} \right\} \begin{array}{l} \text{se queste sono le sole divisioni senza resto,} \\ \text{allora } a \text{ è primo} \end{array} \quad \begin{array}{l} a = p \\ p > 0. \end{array}$$

$$\text{per } a = 0 \quad \left\{ \begin{array}{l} \frac{0}{1} = 0 \\ \frac{0}{0} = \text{indefinito} \end{array} \right.$$

---

$p_i \Rightarrow 1, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, \dots$

$i = 1, 2, 3, \dots$

I numeri primi sono infiniti

- Tutti gli altri numeri interi sono ‘composti’

$$m = \prod_{k=1}^n p_k, \quad p_1 \leq p_2 \leq p_3 \dots \leq p_n, \quad m > 0$$

$p_k$  numeri primi - Teorema fondamentale dell’Aritmetica -  
ovvero

$$m = \prod_{i=1}^n p_i^{e_i}, \quad 1 \leq i \leq n, \quad m > 0, \quad e_i > 0$$

Ad es:  $91 = 7 \times 13$        $20 = 2^2 \cdot 5$

- Il massimo comun divisore ( $\equiv \text{mcd}$ ) di due numeri interi  $m$  e  $n$  è l’intero più grande che li divide entrambi

$$\text{mcd}(m, n) = \max \{k \mid k \mid m \text{ e } k \mid n\}$$

l’espressione

$k$  divide  $m$  (o  $m$  è divisibile per  $k$ )

$$k \mid m \equiv k > 0 \text{ e } m = qk$$

per un certo intero  $q$

Ad es:  $\text{mcd}(12, 18) = 6$

- Due numeri positivi  $m$  e  $n$  interi sono ‘primi tra loro’ se

$$\text{mcd}(n, m) = 1$$

e si scrive

$$n \perp m \text{ o } m \perp n$$

inoltre:

$$\text{mcd}(0, n) = n$$

$$\text{mcd}(0, 0) = \text{indefinito}$$

---

# ARITMETICA MODULARE

---

Dato  $a \leq 0$ , intero

e  $m > 0$ , intero

$$a \bmod m = r$$

significa che

$$a - \left\lfloor \frac{a}{m} \right\rfloor m = r$$

e cioè che

$$a = km + r$$

$$k = \left\lfloor \frac{a}{m} \right\rfloor$$

$$a = \left\lfloor \frac{a}{m} \right\rfloor m + r$$

con

$$0 \leq r \leq m - 1$$

sempre  $r \geq 0$   
i residui di  $a$  modulo  $m$   
sono sempre  $\geq 0$ .

Ad esempio:

$$3 \bmod 7 = 3$$

$$(k = 0) \quad a = 3, m = 7$$

$$7 \bmod 3 = 1$$

$$(k = 2) \quad a = 7, m = 3$$

$$-1 \bmod 7 = 6$$

$$(k = -1) \quad -1 = k \cdot 7 + 6, k = -1$$

$$-2 \bmod 3 = 1$$

$$(k = -1) \quad -2 = k \cdot 3 + 1, k = -1$$

$$2 \bmod 3 = 2$$

$$(k = 0)$$

$$9 \bmod 5 = 4$$

$$(k = 1)$$

$(a \bmod 0 = a, \text{ per definizione})$

$(0 \bmod m = 0, \text{ per definizione})$



---

# EXCLUSIVE OR

---

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

$$(0 + 0) \bmod 2 = 0 \rightarrow 0 \bmod 2 = 0$$

$$(0 + 1) \bmod 2 = 1 \rightarrow 1 \bmod 2 = 1$$

$$(1 + 0) \bmod 2 = 1 \rightarrow 1 \bmod 2 = 1$$

$$(1 + 1) \bmod 2 = 0 \rightarrow 2 \bmod 2 = 0$$

---

Si dice inoltre che  $b$  è congruente con  $a \bmod m$

$$b \equiv a \pmod{m}$$

se  $b$  e  $a$  hanno lo stesso resto quando divisi per  $m$ ,

$$m > 0$$

$$b, a, \geq 0$$

e cioè detti

$$\begin{cases} a = k_1m + r_1 \\ b = k_2m + r_2 \end{cases}$$

$a$  e  $b$  sono congruenti  $\bmod m$  se e solo se

$$r_1 = r_2 = r$$

$$b \bmod m = a \bmod m = r$$

$$(b - a) \bmod m = 0$$

e cioè  $(b - a)$  è ‘DIVISIBILE’ (perfettamente divisibile) per  $m$  (resto = 0).

Ad esempio:

$$13 \bmod 3 = 1$$

$$22 \bmod 3 = 1$$

$$22 \equiv 13 \pmod{3}$$

Si dice anche che l’operazione

$$a \bmod m = r, \quad \text{ove } r \text{ è il residuo di } a,$$

è la riduzione modulare di  $a$ , modulo  $m$ .

L’insieme dei numeri interi

$$0, 1, 2, 3, \dots, m-1$$

è l’insieme completo dei residui, modulo  $m$ .

---

L'insieme

$$\mathbb{Z}_m \ (m > 0; |\mathbb{Z}_m| = m)$$

$$r_i \in \mathbb{Z}_m, \ r_i = 0, 1, 2, \dots, m-1$$

è lo spazio dell' 'aritmetica modulo'  $m$ .

Valgono tutte le proprietà dell'aritmetica

$$(a \pm b) \bmod m \equiv [(a \bmod m) \pm (b \bmod m)] \bmod m$$

$$(a \times b) \bmod m \equiv [(a \bmod m) \times (b \bmod m)] \bmod m$$

$$[a \times (b + c)] \bmod m \equiv \{[(a \times b) \bmod m] + [(a \times c) \bmod m]\} \bmod m$$

---

# POTENZE DI NUMERI MODULO $m$

---

$$a^x \bmod m$$

Esempio

$$a^4 \bmod m$$

$$\rightarrow (a \times a \times a \times a) \bmod m$$

oppure

$$\rightarrow [(a \times a) \bmod m]^2 \bmod m$$

$$3^4 \bmod 2 \rightarrow 81 \bmod 2 = 1$$

oppure

$$\rightarrow (9 \bmod 2)^2 \bmod 2 = (1)^2 \bmod 2 = 1.$$

$$a^x \bmod m = b$$

dato  $x$  e  $a$  trovare  $b$  in  $\mathbb{Z}_m$ .

---

# LOGARITMO DISCRETO

---

$$3^x \bmod 17 = 15$$

dato  $a$ ,  $m$  e  $b$  trovare  $x$ .

per tentativi, in questo caso

$$x = 6$$

$$3^6 = 729$$

$$729 \cdot \bmod 17 = 15$$

infatti

$$729 = \left\lfloor \frac{729}{17} \right\rfloor \times 17 + 15 =$$

$$= 42 \times 17 + 15 = \quad (k = 42)$$

$$714 + 15$$

Purtroppo non tutti i logaritmi discreti hanno soluzioni

Ad esempio:

$$3^x \bmod 13 = 7$$

non ha soluzione.

Trovare l'intero  $x$ ,  $x \geq 0$

tale che

$$a^x \bmod m = b$$

assegnato con  $a$  e  $b$  interi

positivi

cioè

$$a^x = km + r$$

$$k \geq 0 \text{ intero}$$

$$\log_a^D a^x = x = \log_a (km + r)$$

$$k \geq 0 \text{ intero}$$

$$x \cong \log_a^D b$$

$$a^x \equiv b \pmod{m}$$

---

Logaritmi discreti  $\log^D$

$x$  si dice logaritmo discreto

$$a^x \bmod m = b$$

assegnato  $b$

Trovare  $x$  tale che

$$a^x \equiv b \pmod{m}.$$

$$3^x \bmod 17 = 15$$

$$\begin{cases} a = 3 \\ b = 15 \\ m = 17 \end{cases}$$

→ allora  $x = 6$  infatti:

$$3^6 \bmod 17 = 15$$

$$729 \bmod 17 =$$

$$= 729 - \left\lfloor \frac{729}{17} \right\rfloor \cdot 17 =$$

$$= 729 - 714 = 15.$$

---

# ALGORITMO DI EUCLIDE

---

Calcolo ricorsivo

$$\text{mcd}(n, m) = \text{mcd}(m, n)$$

dato

$$0 \leq m < n$$

si calcola con la ricorrenza

$$\begin{cases} * & \text{mcd}(0, n) = n \\ * & \text{mcd}(m, n) = \end{cases} \quad \begin{cases} n > 0 \\ \text{mcd}(0, 0) = \text{indefinito} \end{cases}$$
$$\text{mcd}(n \bmod m, m) \quad (m > 0)$$

Ad esempio

$$\begin{aligned} \text{mcd}(12, 18) &= \\ \text{mcd}(18 \bmod 12, 12) &= \text{mcd}(6, 12) = \\ \text{mcd}(12 \bmod 6, 6) &= \text{mcd}(0, 6) = 6 \end{aligned}$$

Due numeri sono primi tra loro se

$$\begin{aligned} \text{mcd}(n, m) &= 1 \\ n \perp m ; m \perp n \end{aligned}$$

---

# NUMERI INVERSI

---

Dato  $n > 0$  intero, l'inverso è il numero (in generale non intero)  $m$  tale che

$$m = \frac{1}{n}.$$

Ad esempio:

$$n = 4; m = \frac{1}{4},$$

cioè  $m$  è tale che

$$n \times m = 1.$$

Nell'aritmetica modulo  $m$  ( $m > 0$ ), lo spazio dei residui  $\mathbb{Z}_m$ , l'operazione di inversione è definita analogamente  
Dato

$$a \bmod m \qquad a \geq 0$$

trovare l'intero positivo  $x > 0$  tale che

$$(x \cdot a) \bmod m = 1$$

che si può scrivere come

$$a^{-1} \equiv x \pmod{m}$$

$a^{-1}$  congruente con  $x \bmod m$ .



---

L'equazione

$$a^{-1} \text{ congruente con } x \text{ mod } m$$

ha una sola soluzione se e solo se

$$a \text{ e } m$$

sono primi relativi

Se  $m$  è primo  $m = p$

allora ogni numero da 1 a  $m - 1$  è primo relativo con  $m$   
ed ha esattamente un inverso corrispondente

Due numeri sono primi tra loro

$m$  è primo di  $n$

$$m \perp n \equiv m, n \text{ sono interi}$$

$$\text{e } \text{mcd}(m, n) = 1$$

Ad esempio:

$$m = 3 = p$$

$$a = 7 = \text{primo, è certo che } a \perp m$$

allora esiste  $a^{-1}$

$$a^{-1} = 1$$

infatti

$$\begin{aligned} (a^{-1} \cdot a) \text{ mod } m &= & \begin{cases} a &= 7 \\ a^{-1} &= 1 \end{cases} \\ &= (1 \times 7) \text{ mod } 3 = 1 \end{aligned}$$

---

# NUMERI PRIMI

---

Dato

$$m > 0$$

il numero dei numeri primi  $< m$

è 
$$\Pi(m) \rightarrow \frac{m}{l_n(m)}$$

Esempio:

$$m = 2^{256} = 1,1 \cdot 10^{77}$$

$$\Pi(2^{256}) \cong \frac{1,1 \cdot 10^{77}}{l_n(1,1 \cdot 10^{77})} \cong \frac{1,1 \cdot 10^{77}}{177,4} \cong 6,2 \cdot 10^{74}$$

---

A giugno 1999 il numero primo più grande scoperto era:

$$2^{6,972.593} - 1$$

[www.utm.edu/research/prime](http://www.utm.edu/research/prime)

corrispondente a un numero decimale con 2.098.960 cifre,

numero primo di Mersenne della forma  $2^p - 1$  con  $p$  numero primo.

Breve storia della scoperta dei numeri primi più grandi:

1588 Cataldi  $2^{17} - 1$

1772 Eulero  $2^{31} - 1$

1951 Ferrier  $\frac{2^{148} + 1}{17}$  (44 cifre decimali)

Non tutti i numeri di Mersenne sono primi.

Fermat e poi 1876 - Test di Lucas, per verificare se il numero è primo.

---

In generale  $\mathbb{Z}_m$  l'insieme di tutti i residui  $(0 \div m - 1)$ ,  $m > 0$ , comprende numeri interi composti e numeri primi

$$|\mathbb{Z}_m| = m.$$

L'insieme ridotto

$$\mathbb{Z}_m^* \in \mathbb{Z}_m$$

comprende residui che sono numeri primi

$$p_i, 1 \leq i \leq \varphi(m), \quad p_i \perp m$$

e cioè la cardinalità

$$|\mathbb{Z}_m^*| = \varphi(m)$$

dell'insieme 'ridotto' dei residui modulo  $m$  è uguale alla funzione  $\varphi$  di Eulero (Funzione 'Toziente')

Si sa che:

- se  $m = p$ , cioè è primo, allora

$$\varphi(p) = p - 1$$

e cioè comprende tutti i residui tranne lo 0, infatti

$$\text{mcd}(0, p) = p.$$

- se  $m = p \times q$ , ove  $p$  e  $q$  sono primi

allora

$$\varphi(p \times q) = (p - 1)(q - 1).$$

più in generale se

$$m = \prod_{i=1}^n p_i^{l_i} \quad \begin{matrix} l_i > 0 \\ 1 \leq i \leq n \end{matrix}$$
$$\varphi(m) = \prod_{i=1}^n (p_i^{l_i} - p_i^{l_i-1})$$

Ad esempio:

$$m = 60; \quad 60 = 2^2 \cdot 3 \cdot 5$$

$$\varphi(60) = (2^2 - 2^1)(3 - 1)(5 - 1) =$$

$$= 2 \times 2 \times 4 = 16 \quad \begin{cases} p_1 = 2, l_1 = 2 \\ p_2 = 3, l_2 = 1 \\ p_3 = 5, l_3 = 1 \end{cases}$$

$$98 = 2 \times 7^2$$

$$\varphi(98) = (2 - 1)(7^2 - 7) = 42$$

---

# PICCOLO TEOREMA DI FERMAT

---

Dato  $m > 0$  intero e primo  $m = p$ , se  $a$  non è multiplo di  $p$  allora

$$(1) \quad a^{p-1} \equiv 1 \pmod{p}, \quad a \not\equiv 0 \pmod{p};$$

$$\text{e cioè anche } \rightarrow \quad a^p \equiv a \pmod{p}, \quad a \not\equiv 0 \pmod{p};$$

dalla (1) si ha

$$a^{p-1} \bmod p = 1 = 1 \bmod p$$

$$\text{e quindi} \quad (a \cdot a^{p-2}) \bmod p = 1$$

$$a^{p-2} \equiv a^{-1} \pmod{p}$$

$$\text{e cioè} \quad a^{-1} = a^{p-2} \bmod p.$$

---

# TEOREMA DI LAGRANGE

---

afferma che in  $\mathbb{Z}_m^*$  gli elementi  $a \not\equiv 0 \pmod{m}$  elevati a  $\varphi(m)$  danno 1:

Dato  $m > 0$  e  $a \not\equiv 0 \pmod{m}$ ,

$m$  qualsiasi intero, se:  $a \not\equiv 0 \pmod{m}$ , allora  $\rightarrow a^{\varphi(m)} \equiv 1 \pmod{m}$

$$\text{cioè} \quad a^{\varphi(m)} \bmod m = 1 = 1 \bmod m$$

$$\text{quindi} \quad a^{-1} = a^{\varphi(m)-1} \bmod m, \quad a \not\equiv 0 \pmod{m}$$

$$a \in \mathbb{Z}_m^*$$

detta anche espressione di Eulero.

---

• Ad esempio:

$$m = 3 = p$$

$$a = 7 \text{ primo}$$

$$7 \perp 3, a \perp p$$

$$a^{-1} = a^{p-2} \bmod p = 7 \bmod 3 = 1 \bmod 3$$

infatti

$$(1 \times 7) \bmod 3 = 1.$$

• Ad esempio:

$$m = 15$$

$$15 = 3 \times 5, 7 \perp 15$$

$$a = 7$$

$$\varphi(15) = 2 \times 4 = 8$$

$$a^{-1} = 7^7 \bmod 15 = 13 \bmod 15$$

$$7^7 = 823543$$

$$\left\lfloor \frac{7^7}{15} \right\rfloor = 54902$$

$$54902 \times 15 = 823530$$

$$\Delta = a^{-1} = 13$$

infatti

$$(13 \times 7) \bmod 15 = 91 \bmod 15 = 1.$$

• Esempio:

$$\begin{cases} m = 26 = 2 \times 13 \\ a = 7 \end{cases}$$

$$26 \perp 7$$

$$\varphi(26) = 1 \times 12 = 12$$

$$a^{-1} = 7^{11} \bmod 26 = 15$$

$$7^{11} = 1.977.326.743$$

$$\left\lfloor \frac{7^{11}}{26} \right\rfloor = 76.051.028$$

$$76.051.028 \times 26 = 1.977.326.728$$

$$\Delta = 15$$

Verifica:

$$(7 \times 15) \bmod 26 = 105 \bmod 26 = 1$$

$$105 - 4 \times 26 = 1.$$

---

• Esempio:

$$\begin{cases} m = 75 \\ a = 28 \end{cases} \qquad \begin{matrix} 75 = 3 \times 5^2 \\ 28 = 2^2 \times 7 \end{matrix}$$

$$75 \perp 28$$

$$a^{-1} = 28^{39} \bmod 75 =$$

$$\varphi(75) = 2 \cdot (5^2 - 5) = 2 \cdot (25 - 5) = 40$$

$$\left\{ \begin{array}{l} \lfloor \frac{28^{39}}{75} \rfloor \text{ non c'è modo di controllare,} \\ \text{col calcolatore 55 cifre decimali} \end{array} \right\}$$

$$a^{-1} = 67$$

$$(67 \times 28) \bmod 75 = 1$$

$$1876 \bmod 75 = 1$$

$$(\text{infatti: } 25 \times 75 = 1875)$$

Metodo di Euclide Esteso

solo se

$$r_0 = m$$

$$r_1 = a$$

$$a < m$$

$$m > a$$

$$r_0 > r_1.$$

---

# ALGORITMO DI EUCLIDE ESTESO

## EUCLIDEAN ALGORITHM

---

- $\mathbb{Z}_m$  è un anello per ogni intero positivo  $m$

se  $a \in \mathbb{Z}_m$  ha un  $a^{-1}$  moltiplicatore inverso; se, e solo se

$$\text{mcd}(a, m) = 1;$$

inoltre, il numero di interi positivi minori di  $m$   
e primi relativi a  $m$  è  $\varphi(m)$ .

- L'insieme dei residui modulo  $m$  che sono primi relativi a  $m$  è  $\mathbb{Z}_m^*$ .

Ogni elemento di  $\mathbb{Z}_m^*$  ha un moltiplicatore inverso (anche lui in  $\mathbb{Z}_m^*$ ).

$$a \in \mathbb{Z}_m \text{ ha un } a^{-1}.$$

Due numeri  $r_0$  e  $r_1$  positivi,

$$r_0 > r_1$$

$$\begin{array}{l} r_1, 0 \div \max \\ q_1, 1 \div \max \end{array} \left\{ \begin{array}{l} r_0 = q_1 r_1 + r_2 \\ r_1 = q_2 r_2 + r_3 \\ \vdots \\ r_{\max-1} = q_{\max} r_{\max} \end{array} \right. \begin{array}{l} 0 < r_2 < r_1 \\ 0 < r_3 < r_2 \end{array}$$

---

allora

$$\begin{aligned} \text{mcd}(r_0, r_1) &= \text{mcd}(r_1, r_2) \dots = \text{mcd}(r_{\max-1}, r_{\max}) = r_{\max} \\ \text{mcd}(r_0, r_1) &= r_{\max} \end{aligned}$$

Si può usare l'ALGORITMO ESTESO

$$\begin{cases} \text{se} & a < m \\ \text{ha un} & a^{-1} \text{ modulo } m \end{cases}$$

comincio con

$$\begin{aligned} r_0 &= m \\ r_1 &= a \end{aligned}$$

tuttavia - NON calcola  $b^{-1}$

Numeri  $t_0, t_1, \dots, t_{\max}$

con i  $q_j$  definiti come sopra

$$t_0 = 0$$

$$t_1 = 1$$

$$t_j = (t_{j-1} - q_{j-1}t_{j-1}) \bmod r_0 \quad \text{per } j \geq 2$$

si ha

$$r_j \equiv t_j r_1 \pmod{r_0} \quad 0 \leq j \leq \max$$

ove  $q_j$  e  $r_j$  sono definiti dall'algoritmo e  $t_j$  come detto



---

Allora, se

$$\text{mcd}(r_0, r_1) = 1$$

si ha

$$t_{\max} = r_1^{-1} \bmod r_0$$

Allora si calcola la sequenza

$$t_0, t_1, \dots, t_{\max}$$

insieme a

$$q_1, q_2, \dots, q_{\max}$$

$$r_1, r_2, \dots, r_{\max}$$

algoritmo di Euclide

---

Per controllo deve essere

$$t_i \cdot r_1 \equiv r_i \pmod{r_0}$$

per

$$0 \leq i \leq \max$$

$$t_0 \cdot a \equiv m \pmod{m}$$

$$r_0 = m$$

$$0 \equiv 0$$

$$t_1 \cdot a \equiv a$$

$$r_1 = a$$

$$a \equiv a$$

$$t_2 \cdot a \equiv r_2 \pmod{r_0}$$

---

# ESEMPIO

---

$$\begin{cases} m = 75 \\ a = 28 \end{cases}$$

$$\begin{aligned} r_0 &= 75 \\ r_1 &= 28 \\ r_2 &= 19 \\ r_3 &= 9 \\ r_4 &= 1 \end{aligned}$$

$$4 = \max$$

$$a^{-1} = t_{\max} \rightarrow$$

per verifica

$$75 = 2 \times 28 + 19$$

$$r_0 = q_1 r_1 + r_2$$

$$28 = 1 \times 19 + 9$$

$$r_1 = q_2 r_2 + r_3$$

$$19 = 2 \times 9 + 1$$

$$r_2 = q_3 r_3 + r_4$$

$$9 = 9 \times 1$$

$$r_3 = q_4 r_4$$

$$q_1 = 2$$

$$q_2 = 1$$

$$q_3 = 2$$

$$q_4 = 9$$

$$t_0 = 0$$

$$t_1 = 1$$

$$t_2 = t_0 - q_1 t_1 = 0 - 2 \times 1 = -2 \bmod 75 = 73$$

$$t_3 = t_1 - q_2 t_2 = 1 + 1 \times 2 = 3$$

$$t_4 = t_2 - q_3 t_3 = -2 - 2 \times 3 = -8 \bmod 75 = 67$$

$$a^{-1} = t_4 = 67$$

$$\left\{ \begin{array}{l} \bullet t_2 \cdot r_1 = r_2 \bmod 75 \\ 73 \times 28 = 19 \bmod 75 \\ \bullet t_3 \cdot r_1 = r_3 \bmod 75 \\ 3 \times 28 = 9 \bmod 75 \\ 84 \bmod 75 = 9 \\ \bullet t_4 \cdot r_1 = r_4 \bmod 75 \\ 67 \times 28 = 1 \bmod 75 \end{array} \right.$$

---

# ESEMPIO

---

$$\begin{cases} m = 26 \\ a = 7 \end{cases}$$

$$\begin{aligned} r_0 &= 26 \\ r_1 &= 7 \\ r_2 &= 5 \\ r_3 &= 2 \\ r_4 &= 1 \end{aligned}$$

$$26 = 3 \times 7 + 5$$

$$r_0 = q_1 r_1 + r_2$$

$$7 = 1 \times 5 + 2$$

$$r_1 = q_2 r_2 + r_3$$

$$5 = 2 \times 2 + 1$$

$$r_2 = q_3 r_3 + r_4$$

$$2 = 2 \times 1$$

$$r_3 = q_4 r_4$$

$$q_1 = 3$$

$$q_2 = 1$$

$$q_3 = 2$$

$$q_4 = 2$$

$$t_0 = 0$$

$$t_1 = 1$$

$$t_2 = t_0 - q_1 t_1 = 0 - 3 \times 1 = -3, \quad -3 \equiv 23$$

$$t_3 = t_1 - q_2 t_2 = 1 + 1 \times 3 = 4$$

$$t_4 = t_2 - q_3 t_3 = -3 - 2 \times 4 = -11, \quad -11 \equiv 15$$

$$a^{-1} \equiv 15.$$

per verifica

$$\left\{ \begin{array}{l} \bullet t_2 \cdot r_1 = r_2 \pmod{26} \\ \quad 23 \times 7 = 5 \pmod{26} \\ \bullet t_3 \cdot r_1 = r_3 \pmod{26} \\ \quad 4 \times 7 = 2 \pmod{26} \\ \bullet t_4 \cdot r_1 = r_4 \pmod{26} \\ \quad 15 \times 7 = 1 \pmod{26} \end{array} \right.$$

---

Algoritmo esteso di Euclide

Complessità  $O(l_n(m))$

il numero medio di divisioni è

$$0,843 l_n(m) + 1,47$$

(Knuth)

---

# ELEMENTI GENERATORI

---

Se  $m = p$  è primo allora  $\mathbb{Z}_p^*$  è un gruppo di ordine  $p - 1$   $p > 0$

$$|\mathbb{Z}_p^*| = p - 1.$$

Se  $p$  è primo c'è un elemento

$$\alpha \in \mathbb{Z}_p^* \quad \alpha < p$$

che è di *ordine*  $(p - 1)$ :

$$\alpha^{p-1} \equiv 1 \quad \text{cioè } \alpha^{p-1} \bmod p = 1,$$

$\alpha$  è detto elemento primitivo ( o generatore) modulo  $p$

Ogni elemento

$$\beta \in \mathbb{Z}_p^* \quad \beta \perp p$$
$$\text{mcd}(\beta, p) = 1$$

può essere scritto come

$$\beta > 0 \quad \beta = \alpha^i \quad 0 \leq i \leq p-2$$

Il numero di elementi primitivi modulo  $p$  è

$$\varphi(p - 1)$$

# AD ESEMPIO

$$\begin{cases} p = 5 \\ \alpha = 2 \end{cases} \quad \beta = \alpha^i \quad 0 \leq i \leq p-2 \quad \beta = 1, 2, 3, 4 \quad \begin{matrix} \text{da 1 a } p-1 \\ p-1 \\ \beta > 0 \end{matrix}$$

$$\begin{matrix} \text{ciclico} \\ 2^0 \bmod 5 = 1 \\ 2^1 \bmod 5 = 2 \\ 2^2 \bmod 5 = 4 \\ 2^3 \bmod 5 = 3 \end{matrix} \quad \begin{cases} \beta_1 = \alpha^0 \\ \beta_2 = \alpha^2 \\ \beta_{p-1} = \alpha^{p-2} \end{cases}$$

$$\begin{matrix} 2^4 \bmod 5 = 1 \\ 2^5 \bmod 5 = 2 \\ 2^6 \bmod 5 = 4 \\ 2^7 \bmod 5 = 3 \end{matrix}$$

notare che  
 $\alpha = 2$   
è di ordine  
 $p-1$

$$\begin{matrix} 2^8 \bmod 5 = 1 \\ 2^9 \bmod 5 = 2 \\ 2^{10} \bmod 5 = 4 \\ 2^{11} \bmod 5 = 3 \end{matrix} \quad \text{infatti} \quad 2^4 = 16 \bmod 5 = 1$$

ecc.

quanti  $\alpha$  ci sono?

$$\varphi(4)$$

$$4 = 2^2$$

$$\varphi(4) = 2^2 - 2^1 = 2$$

c'è  $\alpha = 2$  e  $\alpha = 3$

2 e 3

---

Infatti per

$$p = 5$$
$$\alpha = 3$$

$$3^0 = 1 \bmod 5 = 1$$

$$3^1 = 3 \bmod 5 = 3$$

$$3^2 = 9 \bmod 5 = 4$$

$$3^3 = 27 \bmod 5 = 2$$

---

$$3^4 = 81 \bmod 5 = 1$$

$$3^5 = 243 \bmod 5 = 3$$

$$3^6 = 729 \bmod 5 = 4$$

$$3^7 = 2187 \bmod 5 = 2$$

---

$$3^4 = 81 \bmod 5 = 1$$

anche  $\alpha = 3$  è di ordine  $p - 1$

---

# QUANTI ELEMENTI PRIMITIVI CI SONO?

$\varphi(p-1)$ .

---

Se

$$(p-1) = \prod_{i=1}^n q_i = q_1 q_2 \dots q_n \quad \text{primi}$$

allora si calcola

$$\alpha^{\frac{p-1}{q_i}} \bmod p$$

$$\begin{cases} \text{se } \neq 1 & \text{OK} \\ \text{se } = 1 & \text{allora } \alpha \text{ NON è primitivo} \end{cases}$$

---

Ad esempio

$$\begin{aligned} p &= 5 \\ p-1 &= 4 = 2 \times 2 \\ \alpha &= 3 \\ 3^{\frac{4}{2}} \bmod p \\ 9 \bmod 5 &= 4 \quad \text{OK} \end{aligned}$$

---

Ad esempio

$$\begin{aligned} p &= 11 \\ \alpha &= 3 \end{aligned}$$

$$\begin{aligned} p-1 &= 10 = 2 \times 5 \\ 3^{\frac{10}{2}} \bmod 11 &= 1: \text{NO } \alpha \neq 3. \end{aligned}$$



---

# CRITTOGRAFIA CLASSICA

---

Cifratura	Ciphering
Decifratura	Deciphering

Crittografare	Cifrare
Decrittografare	Decifrare

Crittosistemi	Crittografia
Crittoanalisi	
Crittologia	

---

Canali sicuri
Comunicazioni sicure

---

Principals	Alice & Bob
Opponent	Oscar

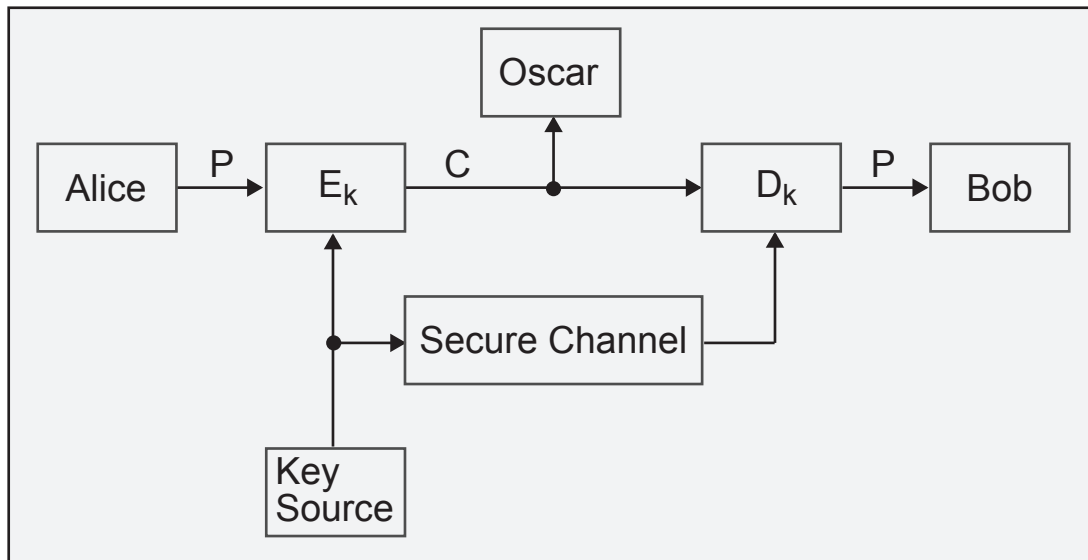
Crittosistema a chiave	
	‘Segreta’ ovvero
	‘Privata’

---

Protocolli di Sicurezza
-------------------------

---

# CRITTOSISTEMA A CHIAVE 'SEGRETA'



'Insecure' Communication Channel

P	Plaintext	Testo in chiaro
C	Ciphertext	Testo cifrato

TEXT\*

testo alfabetico (tipicamente 26 lettere inglesi senza punteggiatura e cifre decimali, ecc.)

$C = E_K(P)$	Algoritmo di cifratura
$P = D_K(C) = D_K[E_K(P)]$	Algoritmo di decifratura
$K = KEY$	Chiave della cifratura

\* A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25  
insieme dei numeri interi da 0 a 25

---

# CRITTOSISTEMA

---

Un CRITTOSISTEMA è una quintupla

$$(\mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{E}, \mathcal{D})$$

- $\mathcal{P}$  insieme dei testi in chiaro,

$$P \in \mathcal{P}$$

- $\mathcal{C}$  insieme dei testi cifrati,

$$C \in \mathcal{C}$$

- $\mathcal{K}$  è lo ‘spazio delle chiavi’, insieme delle possibili chiavi di cifratura,

$$K \in \mathcal{K}$$

- Per ogni  $K \in \mathcal{K}$ , c'è una regola di cifratura

$$E_K \in \mathcal{E}$$

e la corrispondente regola di decifratura

$$D_K \in \mathcal{D}$$

- Per ogni testo in chiaro  $P \in \mathcal{P}$ ,  $E_K$  e  $D_K$  sono funzioni tali che

$$D_K[E_K(P)] = P, \text{ per ogni } P \in \mathcal{P}.$$

---

Il sistema funziona così:

- 1) C'è un modo sicuro con cui Alice e Bob scelgono una 'chiave segreta' a caso

$$K \in \mathcal{K}$$

- 2) Alice vuole mandare un testo in chiaro  $P$  composto dalla sequenza di testi

$$P = P_1 P_2 P_3 \dots P_n, \quad \begin{array}{l} \text{per } n \geq 1 \\ \text{e per } P_i \in \mathcal{P} \\ 1 \leq i \leq n \end{array}$$

Alice calcola

$$C_i = E_k(P_i), \quad 1 \leq i \leq n$$

e manda la sequenza di testi

$$C = C_1 C_2 C_3 \dots C_n$$

$$\left[ \begin{array}{ll} E_k \text{ deve essere una funzione 'iniettiva'} \\ \text{cioè deve essere:} & E_k(P_1) \neq E_k(P_2) \\ \text{se risulta} & P_1 \neq P_2 \\ \\ \text{Se } \mathcal{P} \equiv \mathcal{C} \text{ allora le } E_k \text{ sono PERMUTAZIONI} \end{array} \right]$$

---

# CIFRARIO A SCORRIMENTO

---

Siano  $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$  SHIFT CIPHER  
per  $0 \leq K \leq 25$  definire!

$$P, C \in \mathbb{Z}_{26} \begin{cases} E_K(P) = (P + K) \bmod 26 \\ D_K(C) = (C - K) \bmod 26 \end{cases}$$

26 lettere dell'alfabeto inglese

$$D_K(E_K(P)) = P \text{ per ogni } P \in \mathbb{Z}_{26}$$

---

per  $K = 3$

CAESAR CIPHER  
(cifrario di Giulio Cesare)

A ÷ Z

chiave  $0 \leq K \leq 25$

---

per  $K = 11$

w e w i l l m e e t  
22 4 22 8 11 11 12 4 4 19

aggiungiamo 11 e riduciamo modulo 26

$$\begin{cases} (22 + 11) \bmod 26 = 7 \\ 33 \bmod 26 = 7 \\ (7 - 11) \bmod 26 = 22 \end{cases}$$

$$\begin{aligned} -4 \bmod 26 &= 22 \\ -4 &= K26 + 22 \\ K &= -1 \end{aligned}$$

---

Per rompere il cifrario 'shift' a scorrimento basta esaminare tutte le chiavi,  
sono 26

in media mi bastano  $26/2$  prove per trovare la chiave giusta.

$$26/2 = 13 \text{ prove}$$

è lo schema più 'stupido' possibile per la cifratura dei testi.

---

# CIFRARIO A SOSTITUZIONE

## SUBSTITUTION CIPHER

---

\*  $\mathcal{P} \equiv \mathcal{C} \equiv \mathbb{Z}_{26}$

\*  $\mathcal{K} \equiv$  tutte le permutazioni di 0, 1 ... 25

26 simboli (caratteri alfabetici)

$$|\mathcal{K}| = 26!$$

per ogni permutazione  $\pi \in \mathcal{K}$

$$E_{\pi}(P) = \pi(P)$$

$$D_{\pi}(C) = \pi^{-1}(C)$$

ove  $\pi^{-1}$  è la permutazione inversa di  $\pi$

---

esempio  $\rightarrow$   $a \ b \ c \ \boxed{d} \ e \ f \dots$

permutazione  $\rightarrow$   $X \ N \ Y \ \boxed{A} \ H \ P \dots$

sostituzione lettere  $\rightarrow E_{\pi}(a) \equiv X, E_{\pi}(b) \equiv N, \dots, E_{\pi}(d) \equiv A$

STIRLING

$$m! \simeq \sqrt{2\pi m} \left(\frac{m}{e}\right)^m$$

per  $m \gg 1$

$\boxed{A}$	B	C	D	E	F
$d$	l	r	y	v	o

$$\rightarrow D_{\pi}(A) = d, D_{\pi}(B) = l, \text{ etc.}$$

$$26! \cong 4 \cdot 10^{26}$$

$$\# \text{ medio tentativi} \cong 2 \cdot 10^{26}$$

$$1000 \text{ anni} \cong 3,5 \cdot 10^{10} \text{ sec}$$

$$10^{-9} \text{ sec per tentativo}$$

$$2 \cdot 10^{17} \rightarrow \text{un trilione di anni}$$

---

# CIFRARIO AFFINE

## AFFINE CIPHER

---

$$* \quad \mathcal{P} \equiv \mathcal{C} \equiv \mathbb{Z}_{26}$$

$$* \quad \mathcal{K} = \{(a,b) \in \mathbb{Z}_{26}^* \cdot \mathbb{Z}_{26} : \text{mcd}(a, 26) = 1\}$$

$$\text{Per} \quad K \in \mathcal{K} \quad K = (a, b); a, b > 0$$

$K$  è una coppia di numeri positivi

$$K = (a, b) \in \mathcal{K}$$

si definisce la seguente regola di cifratura

$$C = E_k(P) = (aP + b) \bmod 26$$

e decifratura

$$P = D_k(C) = [a^{-1}(C - b)] \bmod 26,$$

essendo

$$P, C \in \mathbb{Z}_{26} \quad \text{e} \quad a \perp 26$$

infatti

$$C \equiv (aP + b) \pmod{26}$$

$$aP \equiv (C - b) \pmod{26}$$

$$a^{-1}(aP) \equiv [a^{-1}(C - b)] \pmod{26}$$

ma

$$a^{-1}(aP) \equiv (a^{-1}a) P \equiv 1 \cdot P = P$$

quindi

$$P \equiv [a^{-1}(C - b)] \pmod{26}$$

così che

$$D_k(C) = [a^{-1}(C - b)] \bmod 26.$$

---

Residui tutti

$$\mathbb{Z}_{26} \quad 0, 1, 2, \dots, 25 \quad (\text{sono } 26)$$

Primi con 26

$$\text{mcd}(a, 26) = 1$$

sono  $\varphi(26) = 12$

$$1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25$$

$$m = 26 = 2 \times 13$$

$$\varphi(m) = (2-1) \cdot (13-1) = 12$$

Il numero delle chiavi nel cifrario affine è

$$m \cdot \varphi(m)$$

$$26 \times 12 = 312$$

poche!

# ESEMPIO

Ricordo

$$m = \prod_{i=1}^n p_i^{e_i} \quad e_i > 0; 1 \leq i \leq n$$

$$\varphi(m) = \prod_{i=1}^n (p_i^{e_i} - p_i^{e_i-1})$$

$$\text{mcd}(7, 26) = 1$$

$$a = 7 \quad e \quad m = 26$$

$$a \perp 26$$

$$7 \perp 26$$

$$k = (7, 3)$$

$$x = 7^{-1} \text{ mod } 26$$

$$a^{-1} = a^{\varphi(m)-1} \text{ mod } m$$

$$x = 7^{11} \text{ mod } 26$$

$$a \cdot x \equiv 1 \pmod{26}$$

$$7^{11} = 1.977.326.743$$

$$\left\lfloor \frac{7^{11}}{26} \right\rfloor$$

$$76.051.028$$

$$\times 26$$

$$1.977.326.743$$

$$x = a^{-1} = 15$$

$$x = 15$$

infatti

$$7 \times 15 = 105 \equiv 1 \pmod{26}$$

$$\text{mcd}(7, 3) = 1$$

$$\text{mcd}(7 \bmod 3, 3)$$

$$= \text{mcd}(1, 3)$$

$$\text{mcd}(3 \bmod 1, 1) = \text{mcd}(0, 1) = 1$$

cvd



---

# ESEMPIO

---

$$K = (7,3)$$

$$7^{-1} \bmod 26 = 15$$

$$E_k(P) = (7 \times P + 3) \bmod 26$$

$$D_k(C) = [15 (C - 3)]$$

$$= (15C - 45) \bmod 26 =$$

$$= (15C - 45) \bmod 26 =$$

$$= (15C - 19) \bmod 26$$

$$\begin{array}{ccc} \text{h} & , & \text{o} & , & \text{t} \\ \downarrow & & \downarrow & & \downarrow \end{array}$$

$$P = 7 \ , \ 14 \ , \ 19$$

$$(7 \times 7 + 3) \bmod 26 = 52 \bmod 26 = 0$$

$$(7 \times 14 + 3) \bmod 26 = 101 \bmod 26 = 23$$

$$(7 \times 19 + 3) \bmod 26 = 136 \bmod 26 = 6$$

$$\begin{array}{ccc} \downarrow & & \downarrow & & \downarrow \\ C = 0 & , & 23 & , & 6 \\ \downarrow & & \downarrow & & \downarrow \\ A & , & X & , & G \end{array}$$

$$[15 (0 - 3)] \bmod 26 = -45 \bmod 26 = 7$$

$$[15 (23 - 3)] \bmod 26 =$$

$$= 300 \bmod 26 = 14 \qquad \left\lfloor \frac{300}{26} \right\rfloor = 11$$

$$11 \times 26 = 286$$

$$[15 (6 - 3)] \bmod 26 = 45 \bmod 26 = 19.$$

# IL CIFRARIO DI VIGENÈRE

POLIALFABETICO (Blaise de Vigenère)

Shift	}	monoalfabetici	VIGENÈRE CIPHER
Substitution			
Affine			

$n$  è intero positivo

$$\mathcal{P} \equiv \mathcal{C} \equiv \mathcal{K} \equiv (\mathbb{Z}_{26})^n$$

Per una chiave (PAROLA CHIAVE – Keyword) lunga  $n$

$$K = (k_1, k_2 \dots k_n)$$

si definisce:

$$E_k(P_1, P_2 \dots P_n) = (P_1 + k_1, P_2 + k_2 \dots P_n + k_n) \bmod 26$$

e

$$D_k(C_1, C_2 \dots C_n) = (C_1 - k_1, C_2 - k_2 \dots C_n - k_n)$$

$\bmod 26$

operazioni in  $\mathbb{Z}_{26}$ .

Esempio  $n = 6$   $k = \text{CIPHER} \equiv 2, 8, 15, 7, 4, 17$

Parola chiave lunga  $n$

Plaintext

**THIS CRYPTOSYSTEM IS NOT SECURE**

A	0	19	7	8	18	2	17	$n = 6$ parola chiave <b>C I P H E R</b> $k \equiv 2, 8, 15, 7, 4, 17$
B	1	2	8	15	7	4	17	
C	2	21	15	23	25	6	8	
D	3	24	15	19	14	18	24	# of possible keywords $26^n \equiv$ per $n = 6$ $26^6 \equiv 3,1 \times 10^8$
E	4	2	8	15	7	4	17	
F	5	0	23	8	21	22	15	
G	6	18	19	4	12	8	18	per $n = 26$ $26^{26} \equiv 6,1 \times 10^{36}$
H	7	2	8	15	7	4	17	
I	8	20	1	19	19	12	9	
J	9	13	14	19	18	4	2	VPXZGIA XIVWPUBTTMJPWIZITWZT
K	10	2	8	15	7	4	17	
L	11	15	22	8	25	8	19	
M	12	20	17	4				
N	13	2	8	15				
O	14	22	25	19				
P	15							
Q	16							
R	17							
S	18							
T	19							
U	20							
V	21							
W	22							
X	23							
Y	24							
Z	25							

---

Nel cifrario di Vigenère con una parola chiave di  $n$  lettere

Ogni carattere alfabetico

Ad esempio

$T \equiv 19$

Keyword

P A S

15 0 18

può diventare a seconda della sua posizione nel messaggio

$$19 + 15 = 34 \bmod 26 = 8 \rightarrow I$$

$$19 + 0 = 19 \bmod 26 = 19 \rightarrow T$$

$$19 + 18 = 37 \bmod 26 = 11 \rightarrow L$$

 secondo 'P A S'

cifrario Polialfabetico.

# CIFRARIO A PERMUTAZIONE

\*  $n$  è intero positivo  $n > 0$

\*  $\mathcal{P} \equiv C \equiv (\mathbb{Z}_{26})^n$

$K$  consiste in tutte le permutazioni di  $\{1, 2, 3, \dots, n\}$   
(sono  $n!$ )

per  $K = \Pi$  una permutazione si definisce:

$$E_K(P_1, P_2 \dots P_n) = [P_{\Pi(1)}, P_{\Pi(2)} \dots P_{\Pi(n)}]$$

e

$$D_K(C_1, C_2 \dots C_n) = [C_{\Pi^{-1}(1)}, C_{\Pi^{-1}(2)} \dots C_{\Pi^{-1}(n)}]$$

essendo  $\Pi^{-1}$  la permutazione inversa di  $\Pi$ .

→ Usiamo un esempio alfabetico

Ad esempio

$$\begin{array}{c} n = 6 \\ \Pi \rightarrow \end{array} \begin{array}{|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 3 & 5 & 1 & 6 & 4 & 2 \\ \hline \end{array}$$

Testo .....	.....	.....
$\Pi \rightarrow$ S H E S E L	L S S E A S	H E L L S A N D ...
1 2 3 4 5 6		
3 5 1 6 4 2		
E E S L S H	S A L S E S L S	H A L E

$\Pi^{-1} \rightarrow$

$$\begin{array}{|c|c|c|c|c|c|} \hline 1 & 2 & 3 & 4 & 5 & 6 \\ \hline 3 & 6 & 1 & 5 & 2 & 4 \\ \hline \end{array}$$

1 2 3 4 5 6	
3 5 1 6 4 2	
S H E S E L	L S S E A S

la complessità è

$$n!$$

abbiamo visto che

$$26! \cong 4 \times 10^{26}$$

Anche questo è Polialfabetico

a differenza di quello a sostituzione che è Monoalfabetico

---

# CIFRARIO A CATENA

## STREAM CIPHER

---

\* Finora

BLOCK CIPHER

$$C = C_1 C_2 \dots C_i \dots = E_k(P_1) E_k(P_2) \dots E_k(P_i) \dots$$

Cifrari a Blocco

---

Catena di chiavi (Keystream)

$$Z \equiv (Z_1, Z_2, \dots, Z_i \dots) \quad i \rightarrow \infty$$

$$C = C_1 C_2 \dots C_i \dots = E_{Z_1}(P_1) E_{Z_2}(P_2) \dots E_{Z_i}(P_i) \dots$$

Funziona così

$K$  è la 'chiave'  $\in \mathcal{K}$

STREAM CIPHER

La 'generica chiave'  $Z_i$  è generata con una funzione della 'chiave' e dei testi chiari precedenti,  $(i-1)$  testi.

Elemento Keystream

$$Z_i = f_i(K, \underbrace{P_1, P_2, \dots, P_{i-1}}_{(i-1) \text{ testi}}) \quad 1 \leq i \leq \infty$$

la chiave dipende dal passato testo in chiaro e poi è usata per cifrare

$$Z_i \rightarrow E_{Z_i}(P_i) = C_i$$

poi

$$Z_{i+1} \rightarrow C_{i+1}.$$

---

Block cipher = Stream cipher

se

$$Z_i = K, \quad \forall_i$$

---

Esempio

Testo in chiaro a Stream  $P_1, P_2, P_3, P_4$

Calcolo

$$Z_1 = f_1(K) \equiv K$$

e trovo

$$C_1 = E_{Z_1}(P_1)$$

poi calcolo

$$Z_2 = K + P_1$$

e trovo

$$C_2 = E_{Z_2}(P_2)$$

Poi calcolo ancora

$$Z_3 = K + P_1 + P_2, \text{ etc.}$$

In ricezione

$$Z_1 = K$$

calcolo

$$P_1 = D_{Z_1}(C_1)$$

poi calcolo

$$Z_2 = K + P_1$$

e verifico

$$P_2 = D_{Z_2}(C_2)$$

in sequenza, etc.

---

# CIFRARIO A CATENA

## STREAM CIPHER

---

Settupla  $\{ \mathcal{P}, \mathcal{C}, \mathcal{K}, \mathcal{Z}, \mathcal{F}, \mathcal{E}, \mathcal{D} \}$

- $P \in \mathcal{P}$  testi in chiaro
- $C \in \mathcal{C}$  testi cifrati
- $K \in \mathcal{K}$  spazio delle possibili chiavi
- $Z \in \mathcal{Z}$  alfabeto della catena di chiavi (keystream)
- $\mathcal{F} \equiv (f_1, f_2, \dots)$  è il generatore della catena di chiavi  
ove  $f_i, \quad i \geq 1$

è tale che

$$\mathcal{K} \times \mathcal{P}^{i-1} \Rightarrow \mathcal{Z}$$

Per ogni  $Z_i \in \mathcal{Z}$  c'è una regola di cifratura

$$E_{Z_i} \in \mathcal{E}, \quad i \geq 1$$

e di decifratura

$$D_{Z_i} \in \mathcal{D}.$$

$E_{Z_i}$  è tale che

$$\mathcal{P} \Rightarrow \mathcal{C}$$

$D_{Z_i}$  è tale che

$$\mathcal{C} \Rightarrow \mathcal{P}$$

in modo tale che

$$D_{Z_i}[E_{Z_i}(P)] = P$$

per ogni  $P \in \mathcal{P}$ .

---

Il cifrario a blocco è un caso particolare di quello a catena quando

$$Z_i = K \quad \text{per tutti gli } i \geq 1$$

---

---

Un cifrario a catena si dice

SINCRONO

se la catena di chiavi (Keystream) è indipendente dal testo in chiaro  $P$   
e cioè se il keystream è generato

$$Z_i = f_i(K, i) \quad i \geq 1$$

solo a partire dalla chiave  $K$  e dall'indice ( $i$ ) dello stream.  
 $K$  è il 'SEED' seme della catena.

Un cifrario a catena si dice

PERIODICO

di Periodo =  $d$

per  $d > 0$

se

$$Z_{i+d} = Z_i \quad i \geq 1.$$

Il cifrario di Vigenère è periodico di periodo  $n$  (Parola chiave – keyword)

la  $K = (K_1, K_2, \dots, K_n)$

e si ha  $Z_i = K_i \quad 1 \leq i \leq n$

e poi la catena si ripete periodicamente.

---

I Cifrari a catena spesso si applicano su alfabeti binari

$$\mathcal{P} = \mathcal{C} = \mathcal{Z} = \mathcal{Z}_2$$

si ha

$$\begin{aligned} E_{Z_i}(P_i) &= (P_i + Z_i) \bmod 2 \\ D_{Z_i}(C_i) &= (C_i + Z_i) \bmod 2 \end{aligned} \quad i \geq 1.$$

# ESEMPIO

Ad esempio  
usiamo la regola

$$Z_{i+4} = (Z_i + Z_{i+1}) \bmod 2$$

con vettore di Inizializzazione

$$(1, 0, 0, 0)$$

si ha

$$\begin{array}{c|cccc|cccc|cccc} Z_1 & Z_2 & Z_3 & Z_4 & Z_5 & Z_6 & Z_7 & Z_8 & Z_9 & Z_{10} & Z_{11} & Z_{12} \\ \hline 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{array}$$

$$i = 1, 2, 3, 4, 5, 6, 7, 8,$$

quindi se devo cifrare il testo binario esadecimale

	A	7	B	F	F	1	1	4	0
	1010	0111	1011	1111	1111	0001	0001	0100	0000
$P \oplus$	A	7	B	F	F	1	1	4	0
$Z$	1000	1001	1010						
	0010	1110	0001						
$C$	2	E	1						
$Z \oplus$	1000	1001	1010						
	1010	0111	1011						
$P$	A	7	B						

0000 0  
0001 1  
0010 2  
0011 3  
0100 4  
0101 5  
0110 6  
0111 7  
1000 8  
1001 9  
1010 A  
1011 B  
1100 C  
1101 D  
1110 E  
1111 F



# METODO DI CIFRATURA A CATENA

## ONE TIME PAD

A Catena uso chiavi binarie  $K_i$  per codificare un testo binario

La lunghezza è  $n$  bit

Ad esempio, nel GSM

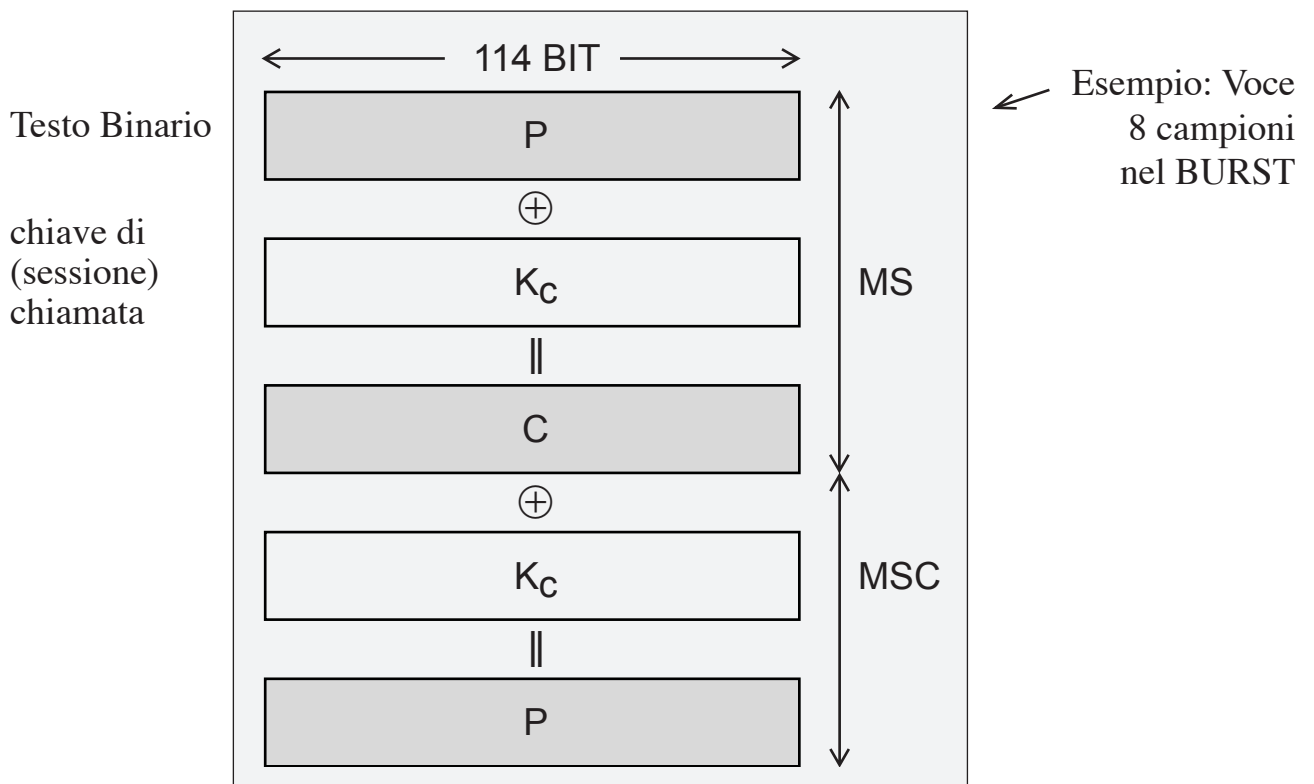
$$K_c = f(K_{\text{SUB}} + \text{SEQNO})$$

Varia BURST PER BURST

$$K_c = 114 \text{ BIT}$$

$$K_{\text{SUB}} = 64 \text{ BIT}$$

$$\text{SEQNO} = 22 \text{ BIT}$$



## ONE TIME PAD

$$2^{114} \cong 2,2 \times 10^{35}$$

tre condizioni per il  
cifrario "perfetto" OTP

- 1 - chiave lunga come il testo
- 2 - chiave "perfettamente" casuale
- 3 - chiave usata una sola volta

---

# CRITTOANALISI DI TESTI CIFRATI

---

Il principio di Kerckhoff dice che Oscar conosce sempre il tipo di algoritmo di cifratura e deve soltanto cercare la chiave per decifrare.

→ Tipi di Attacchi al Criptosistema

(1) Solo Testo Cifrato

Oscar ha solo una stringa di  $C_i$

(2) Testo In Chiaro Noto

Oscar ha una stringa  $P_i$  e la corrispondente stringa  $C_i$

(3) Testo In Chiaro Scelto

Oscar ha accesso temporaneo al CIFRATORE  
quindi sceglie una stringa  $P_i^*$  e la codifica  $C_i^*$

(4) Testo Cifrato Scelto

Oscar ha accesso temporaneo al DECIFRATORE  
quindi sceglie una stringa  $C_i^{**}$  e ottiene la stringa  $P_i^{**}$ .

---

(1) è l'attacco più debole

Basta fare ricorso alle proprietà statistiche del testo (lingua scritta)

---

## ATTACCO SUL 'SOLO TESTO CIFRATO'

$\left\{ \begin{array}{l} \text{Frequenza lettere inglesi singole} \\ \text{digrammi} \\ \text{trigrammi} \end{array} \right\}$  conoscenza a priori  
della statistica  
del messaggio

	lettera	p	monogrammi
0,127	E	$\Rightarrow 0,12$	
0,091	T, A, O, I, N, S, H, R	$\Rightarrow 0,06 \leq p \leq 0,09$	
	D, L	$\Rightarrow 0,04$	
	C, U, M, W, F, G, Y, P, B	$\Rightarrow 0,015 \leq p \leq 0,028$	
	V, K, J, X, Q, Z	$\Rightarrow 0,01$	
			digrammi
	TH		
	HE		
	IN		
	ER		
	AN		
	RE		
	ED, ecc.		
			trigrammi
	THE		
	ING		
	AND		
	HER		
	ERE		
	ENT		
	THA, ecc.		

---

Ad esempio

## ATTACCO AL CIFRARIO AFFINE

Ricevo un testo di 57 caratteri

FMXVEDKA.....RHHRH  
57 ch

con le seguenti occorrenze di monogramma

R - 8 volte  
D - 6 volte  
E, H, K - 5 volte  
F, S, V - 4 volte

Si ipotizza

$\begin{cases} R \rightarrow e \\ D \rightarrow t \end{cases}$

prendo una coppia

$\begin{cases} E_K(4) = 17 \\ E_K(19) = 3 \end{cases}$  ipotesi

$\begin{array}{|l} e = 4 \\ t = 19 \\ \hline R = 17 \\ D = 3 \end{array}$

ma (cifrario affine)

$$E_K(P) \equiv aP + b$$

sempre modulo 26

$$\begin{cases} 4 \cdot a + b \equiv 17 \\ 19 \cdot a + b \equiv 3 \end{cases}$$

$$\begin{cases} a = 6 \\ b = 19 \end{cases}$$

$$\text{mod}(6, 26) = 2$$

ma

$$\text{mcd}(a, 26) = 2 > 1$$

allora errato!

Riprovo

$\begin{aligned} R &\rightarrow e \\ K &\rightarrow t \end{aligned}$

questo porta

$\begin{cases} a = 3 \\ b = 5 \end{cases}$  LEGAL KEY  $\text{mcd}(3, 26) = 1$ .

Allora scegliamo

$\Rightarrow$

$$K = (3, 5)$$

e decrittiamo

$$D_K(C) = 9C - 19$$

ALGORITHM... ..PROCESSES

57 ch

---

Ad esempio

prima lettera è

$$C = 3P + 5$$

F M X V E D  
↓ ↓ ↓ ↓ ↓ ↓  
A L G O R I

$$C = (3P + 5) \bmod 26$$

$$K = 3,5$$

$$a = 3$$

$$b = 5$$

$$a^{-1} = 9$$

$$D_K(C) = a^{-1}(C - b)$$

$$a^{-1} = 9$$

$$a^{-1} = 3^{11} \bmod 26$$

$$P = 9C - 9 \cdot b =$$

$$\left\lfloor \frac{177.147}{26} \right\rfloor =$$

$$= 6813$$

$$= 9C - 45 \bmod 26 =$$

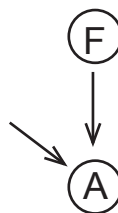
$$6813 \times 26$$

$$= 9C - 19$$

$$\frac{177.138}{9}$$

$$\underline{P = 9 \times 5 - 19 = 0}$$

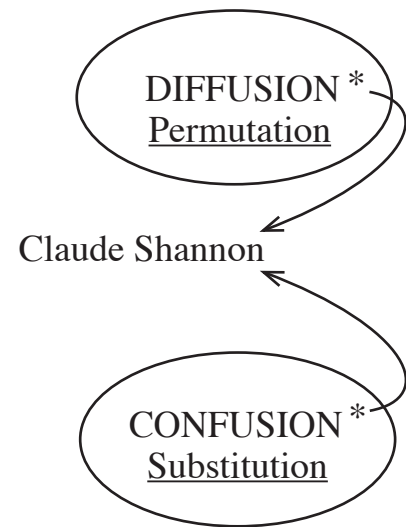
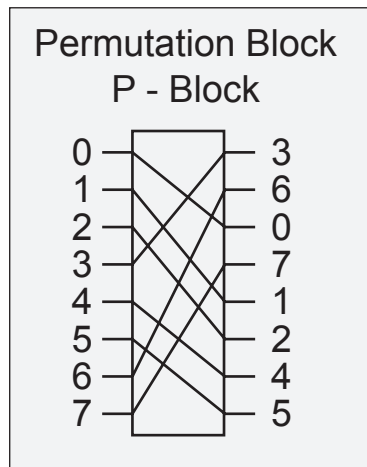
$$A = 0$$



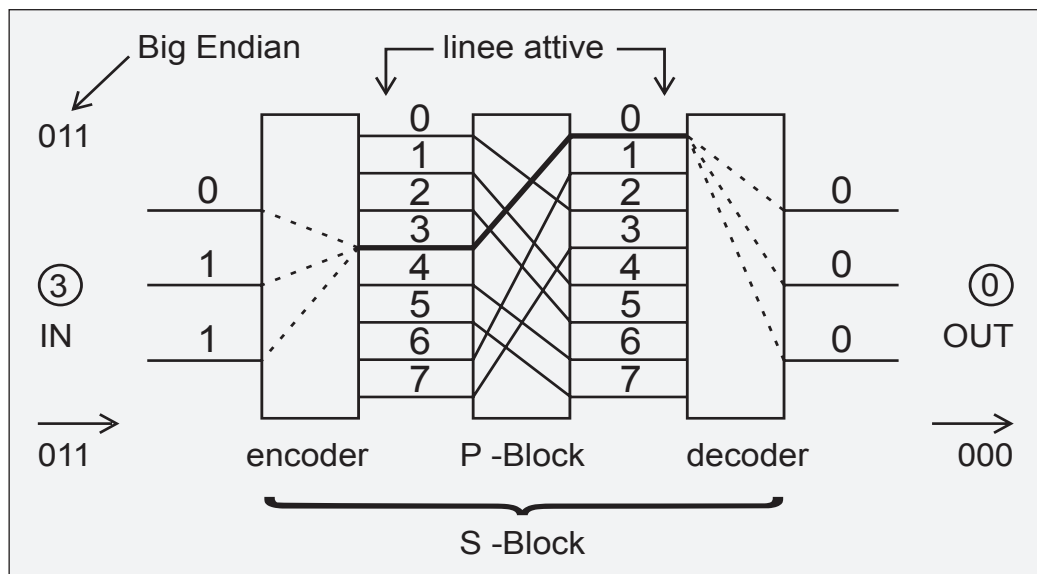
# DIGITAL ENCRYPTION ALGORITHM

## DES

### TRASPOSIZIONI



### SOSTITUZIONI

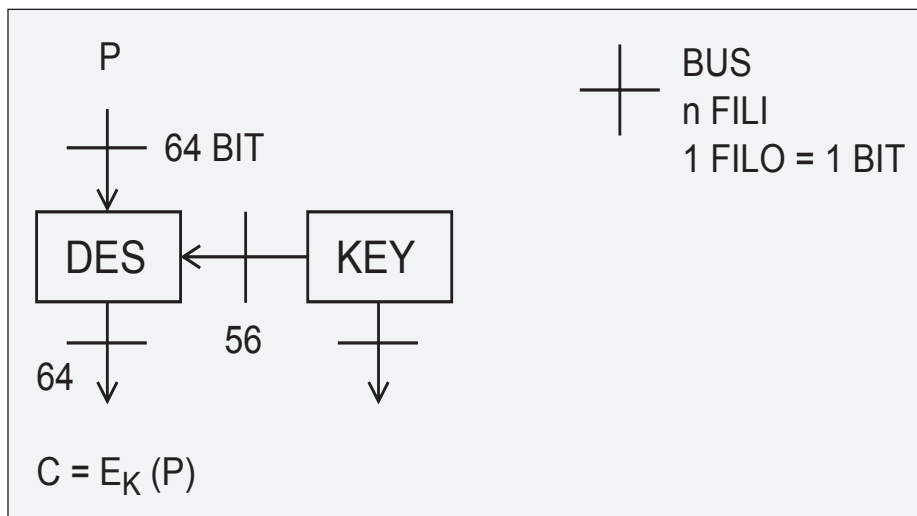


Plain				Chypher
000	0	_____	2	010
001	1	_____	4	100
010	2	_____	5	101
011 →	3	_____	0	→ 000
	4	_____	6	
	5	_____	7	
	6	_____	1	
	7	_____	3	

---

# DES

---



$\frac{2^{56}}{2} = 2^{55}$  numero medio di tentativi per decifrare  
complessità d'attacco

$$2^{55} \cong 10^{16}$$

se per ogni tentativo  $10ns$   $10^{-8}s$

servono per decrittare  $\sim 10^8s$

$10^8s \cong 3$  anni e 2 mesi

---

$p$  = permutazione

$re$  = *rotate left* sulla chiave divisa in due blocchi da 28 bit

rotazioni di 1 o 2 bit secondo la sequenza

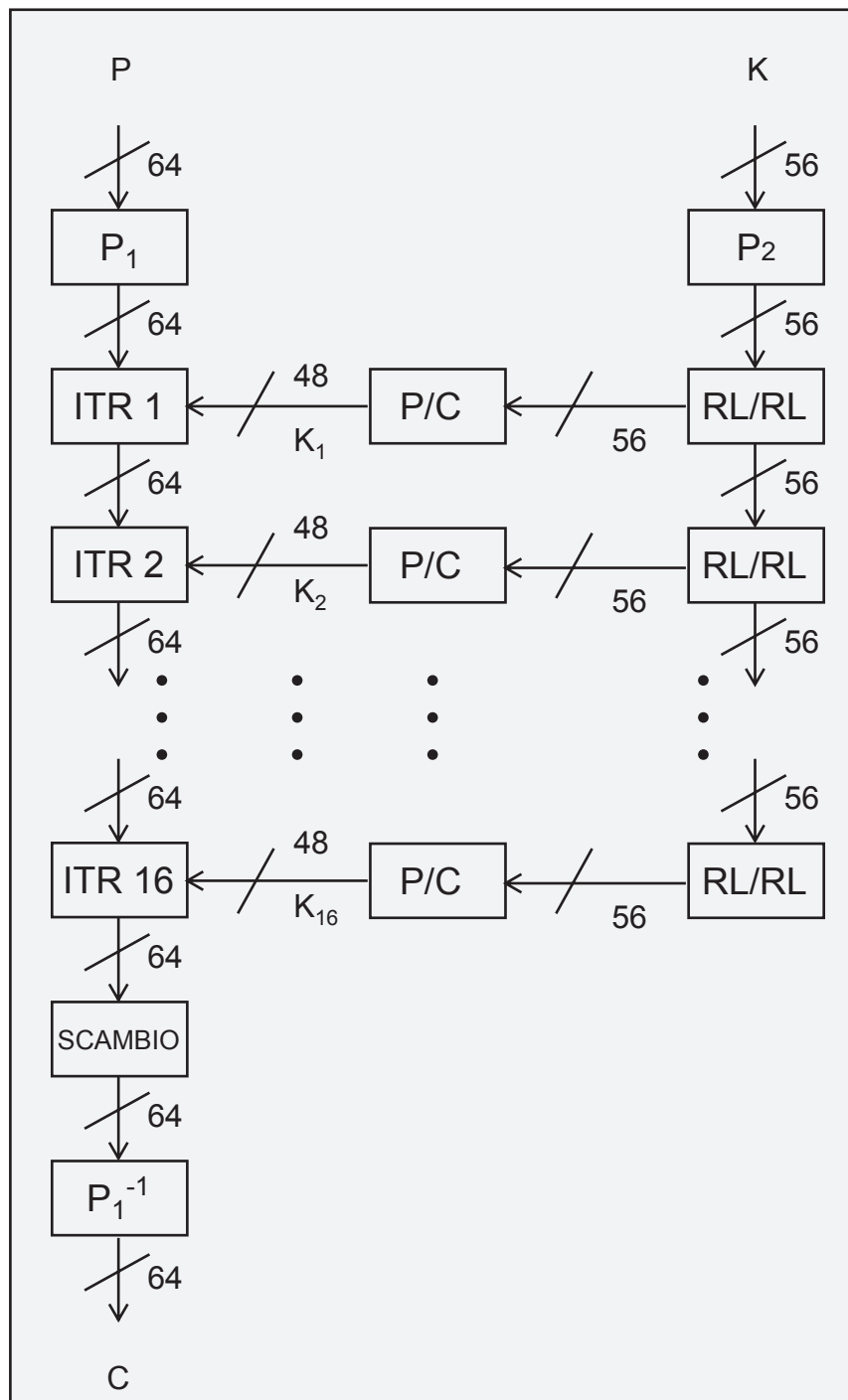
16 ITERAZIONI  $\rightarrow 1\ 1\ 2\ 2\ 2\ 2\ 2\ 2\ 1\ 2\ 2\ 2\ 2\ 2\ 2\ 1 = 28 \rightarrow$  BEGIN AGAIN

$p/c$  = permutazione e concentrazione

che da 56 bit genera chiave di stadio a 48 bit

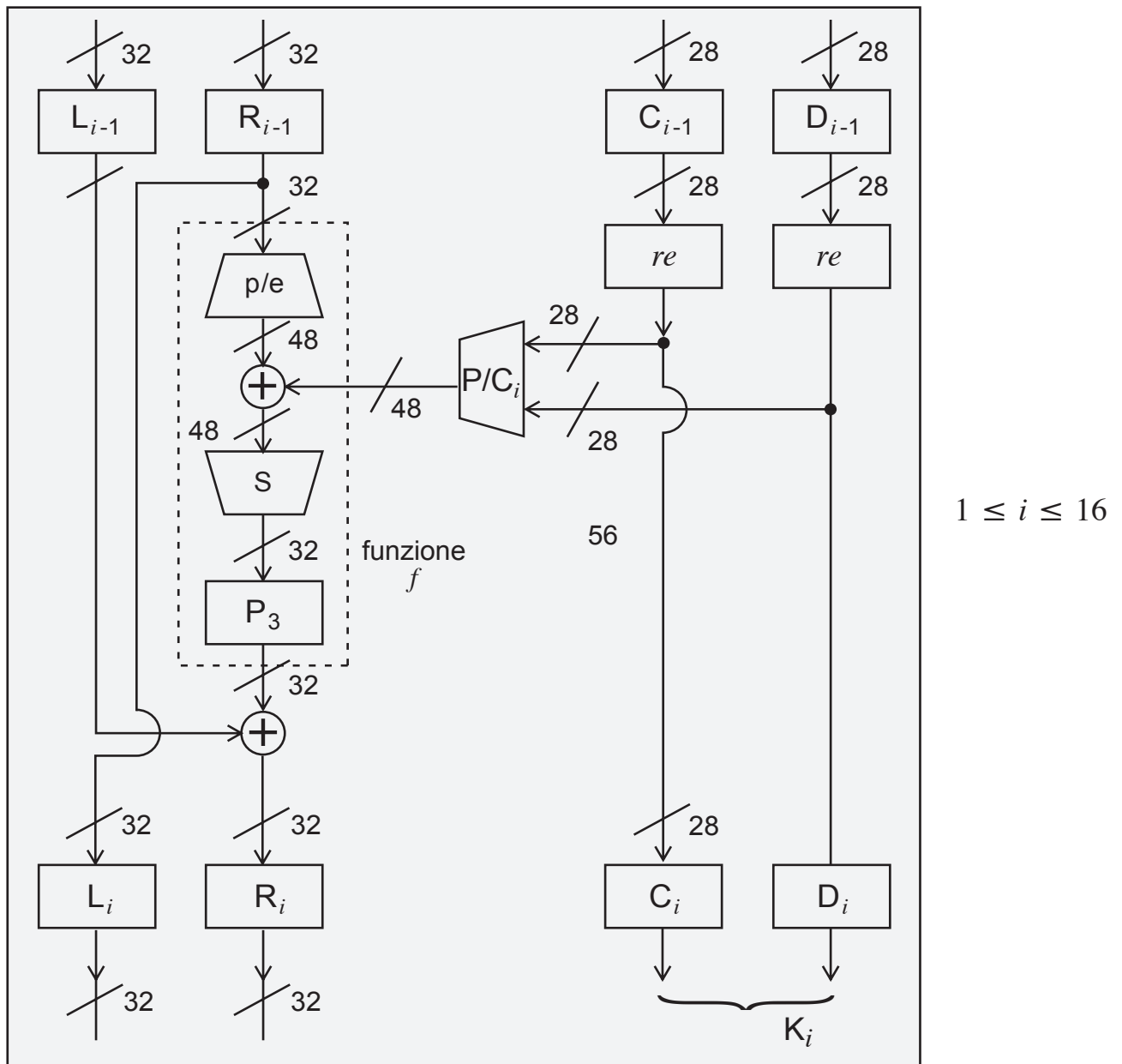
$p/e$  = permutazione e espansione

che da 32 bit genera codice a 48 bit





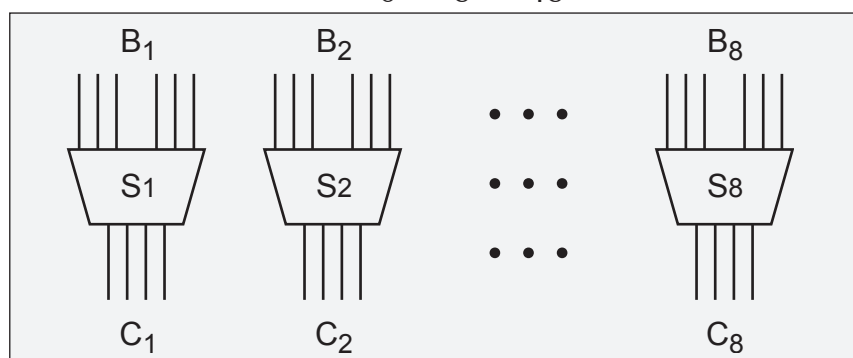
# OGNI ITERAZIONE



BLOCCO S

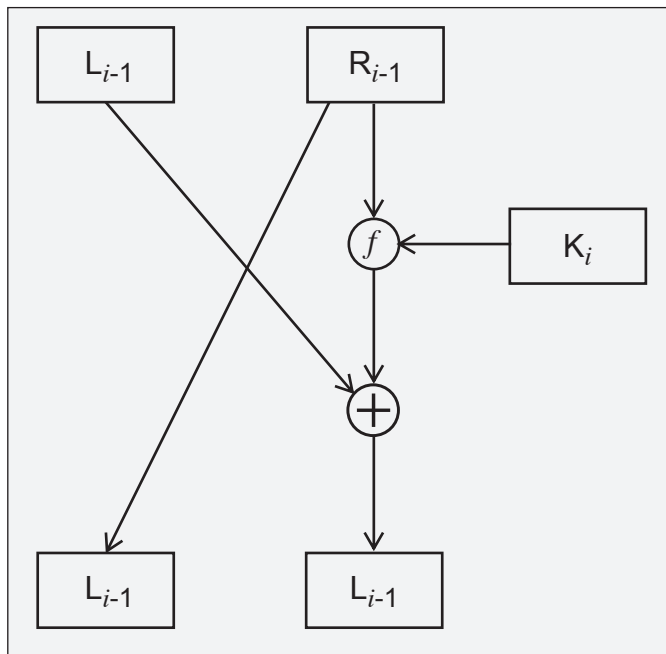
$$6 \times 8 = 48$$

8  
BLOCCHI  
 $6 \times 4$

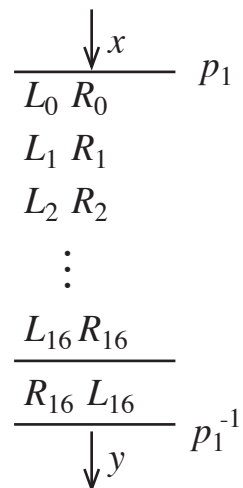


$$4 \times 8 = 32$$

## ITERAZIONE



$i$  da 1 a 16



- ❶ initial permutation  $p_1$  on plain text  $x$

$$x_0 = p_1(x) = L_0 R_0$$

$L_0$  primi 32 bit di  $x_0$   
 $R_0$  ultimi 32 bit di  $x_0$

- ❷ 16 iterazioni  $1 \leq i \leq 16$

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$$

$K_i = K_1, K_2 \dots K_{16}$  chiavi di 48 bit

calcolate in funzione della chiave  $K$  di 56 bit

Ogni  $K_i$  è una permutazione selezionata di  $K$

- ❸ inverse permutation  $p_1^{-1}$

si prende

$$L_{16} R_{16}$$

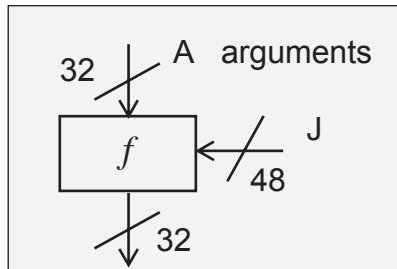
e si scambia

$$R_{16} L_{16}$$

poi si fa:

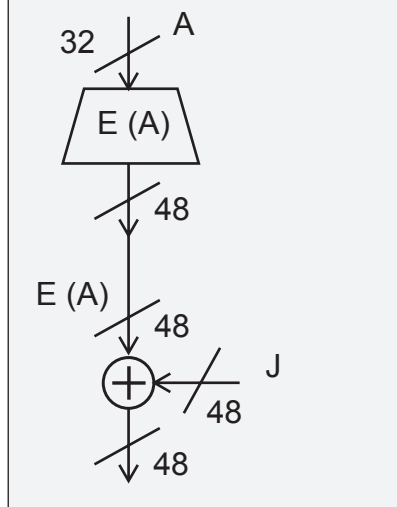
$$y = p_1^{-1}(R_{16} L_{16})$$

Funzione  $f$



Passo 1

$p/e$



← permutazione di A  
e con 16 bit  
che appaiono due volte

Passo 2

calcola

$$E(A) \oplus J$$

e spilla il risultato a 48 bit in otto stringhe da 6 bit

$$B \equiv B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$$

$$A \rightarrow E(A) \rightarrow \oplus \leftarrow J$$

↓ 48

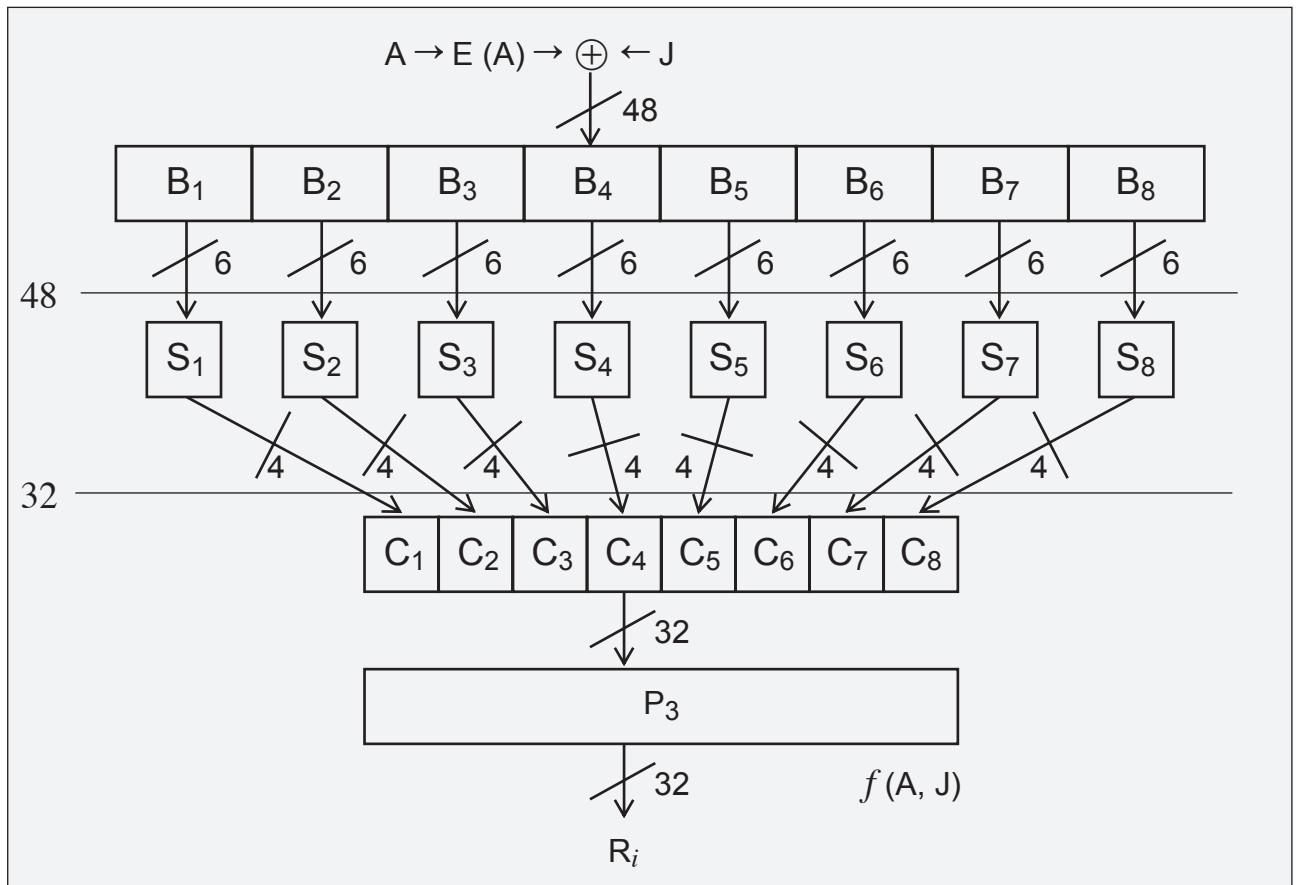
## Passo 2

calcola

$$E(A) \oplus J$$

e spilla il risultato a 48 bit in otto stringhe da 6 bit

$$B \equiv B_1 B_2 B_3 B_4 B_5 B_6 B_7 B_8$$



Ogni  $S_i$  ( $1 \leq i \leq 8$ )

ha una matrice  $4 \times 16$  con elementi compresi tra 0 e 15

$$B_j = b_1 b_2 b_3 b_4 b_5 b_6$$

$$S_j(B_j)$$

$$b_1 b_6$$

row bits -  $r$

$$0 \leq r \leq 3$$

$$b_2 b_3 b_4 b_5$$

column bits -  $c$

$$0 \leq c \leq 15$$

$$\text{entry } S_j(B_j) = S_j(r, c) = c_1 c_2 c_3 c_4$$

$$\text{scritto in binario a 4 BIT} = C_j \quad C_j = S_j(r, c) = S_j(B_j)$$

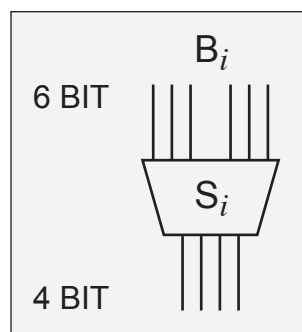
$$1 \leq j \leq 8$$

## S – BOX

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

Box rows & columns

$$1 \leq i \leq 8$$



RIGA a b c d e f COLONNA

0 ÷ 3 0 ÷ 15

a f b c d e

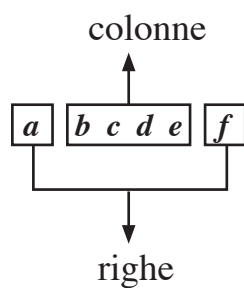
$R_1, C_1 \rightarrow 12$

Box entry

12 → 11 00

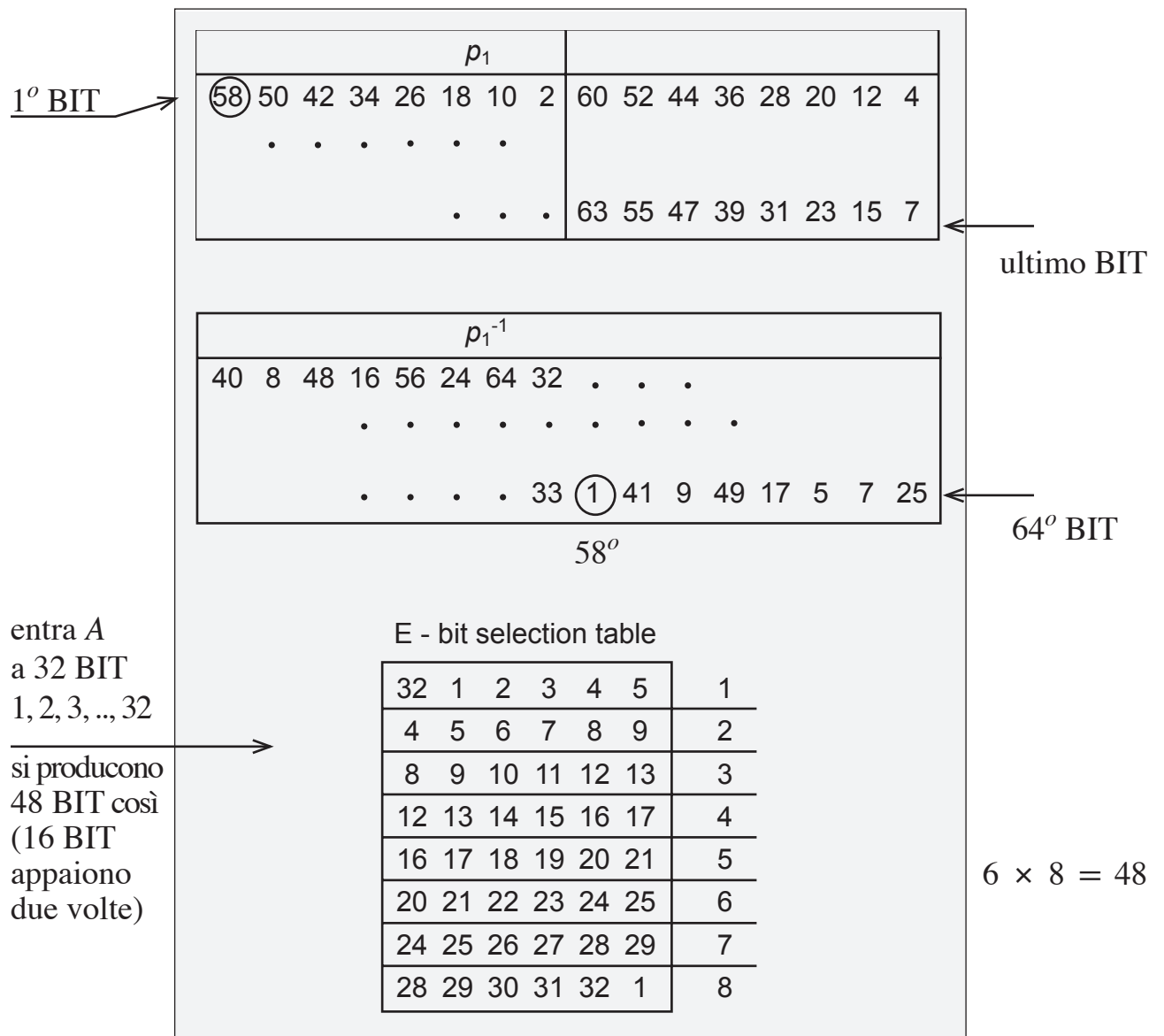
Ogni riga è una permutazione di 16 valori 8 S – box

8 tabelle di 64 possibili uscite



$C = C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8$   
 è permutata con la permutazione  $p_3$   
 $f(A, J) = p_3(C)$

BIT di  $x$      $1^o, \dots, 64^o$



## S1 BOX

1÷32  
32 BIT  
permutazione  
 $P_3$

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

$p_3$								
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	

= CALCOLO DELLE CHIAVI  $K_1 \div K_{16}$   
KEY SCHEDULE

64 BIT =  $K$  = 56 BIT – CHIAVE + 8 PARITY CHECK

64 BIT  
8 PARITY  
56 KEY

BYTE

1	1	2	3	4	5	6	7	8
2	9	10	11	12	13	14	15	16
3	17	18	19	20	21	22	23	24
4	25	26	27	28	29	30	31	32
5								40
6								48
7								56
8								64

→ ogni byte ha un  
numero dispari di 1

← ignorati

---

### Key Schedule

- ① Dati  $K$  64 BIT buttare i parity-check  
permutare con la permutazione  $p_2$

$$p_2(K) = C_o D_o$$

$C_o$  – primi 28 bit di  $p_2(K)$

$D_o$  – ultimi 28 bit di  $p_2(K)$

- ② Calcolare per  $1 \leq i \leq 16$

$$\begin{cases} C_i = LS_i(C_{i-1}) \\ D_i = LS_i(D_{i-1}) \end{cases}$$

e quindi

$$- K_i = p/c(C_i, D_i)$$

ove

$LS_i$  è uno shift a sinistra di 1 o 2 bit dipendente da  $i$

①	—	1	$p/c$
②	—	1	permutazione fissa
3	—	2	
4	—	2	
5	—	2	
6	—	2	
7	—	2	
8	—	2	
⑨	—	1	
10	—	2	
11	—	2	
12	—	2	
13	—	2	
14	—	2	
15	—	2	
⑬	—	1	



	1			$p_2$			7
1	57	49	41	33	25	17	9
2	1	58	50	42	34	26	18
		•	•	•	•	•	
8	21	13	5	28	20	12	4

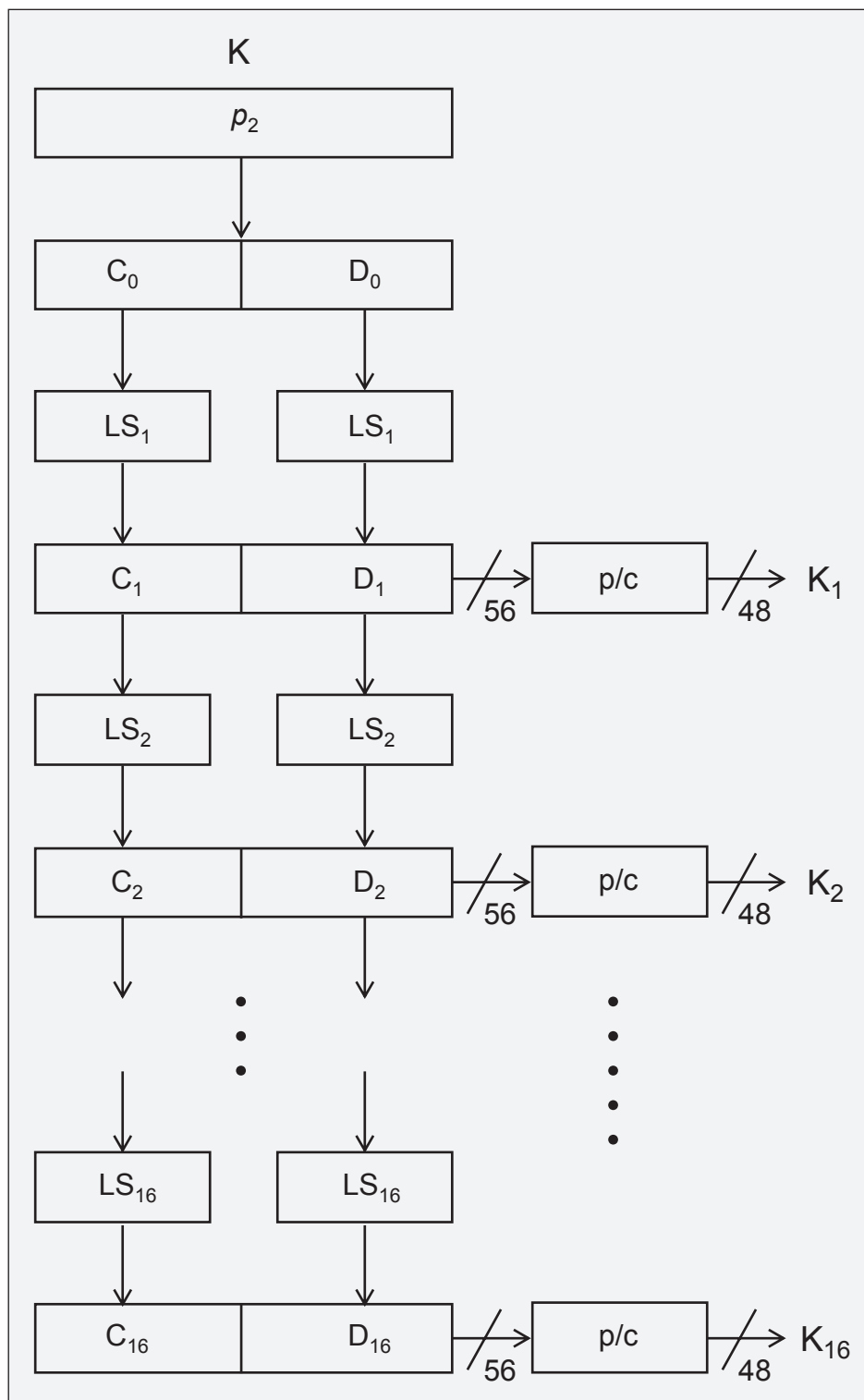
  

							6
—	14	17	11	24	1	5	
—	3	28	15	6	21	10	
—	23	19	12	4	26	8	
—	16	7	27	20	13	2	
—	41	52	31	37	47	55	
—	30	40	51	45	33	48	
—	44	49	39	56	34	53	
8	46	42	50	36	29	32	

7 × 8  
56 BIT

entrano 56 BIT  
escono 48 BIT

6 × 8  
perdo 8 BIT



---

dei 56 bit di  $K$

$K_1$											
10	51	34	60	49	17	33	57	2	9	19	42
3											41
22											29
61	21	38	63	15	20	45	14	13	62	55	31

$12 \times 4 = 48 \text{ BIT}$   
ROUND 1  
 $K_{16}$

---

Decrittazione

→ si usa  $C$  come input  
le chiavi in ordine inverso

$K_{16}, K_{15}, \dots, K_1$

l'uscita è  $P$ .

---

# EXTENDED DES

---

DES è usato per crittografare i PIN degli ATM  
è usato nelle CHIPS per autenticare transazioni  
tra Clearing Houses Interbank Payment System

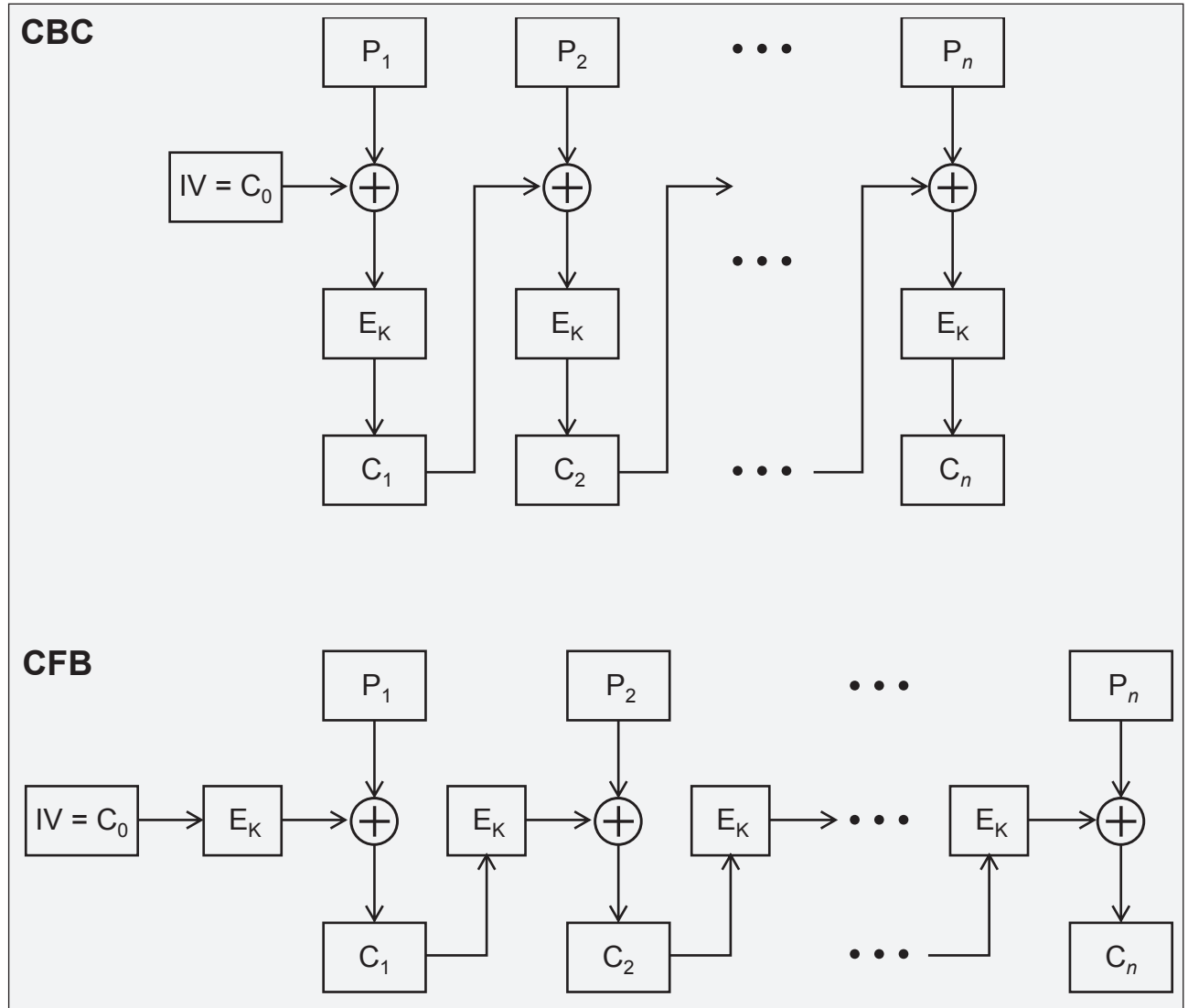
(1)	E C B	Electronic CodeBook
(3)	C F B	Cipher FeedBack mode
(2)	C B C	Cipher Block Chaining mode
(4)	O F B	Output FeedBack mode

ECB

$P_1$	$P_2$	$\dots P_n$	di 64 bit di $P$	$\leftarrow$ plaintext
$\downarrow$	$\downarrow$	$\downarrow$		
$k$	$k$	$k$	stessa chiave	
$\downarrow$	$\downarrow$	$\downarrow$		
$C_1$	$C_2$	$\dots C_n$	di 64 bit di $C$	$\leftarrow$ ciphertext

# ENCRYPT

## ENCRYPT



CBC

64 BIT Initialization Vector IV

$C_0 = IV$

$$\{ C_i = E_k = (C_{i-1} \oplus P_i), \quad 1 \leq i, \quad i = 1, 2, \dots, n$$

CFB

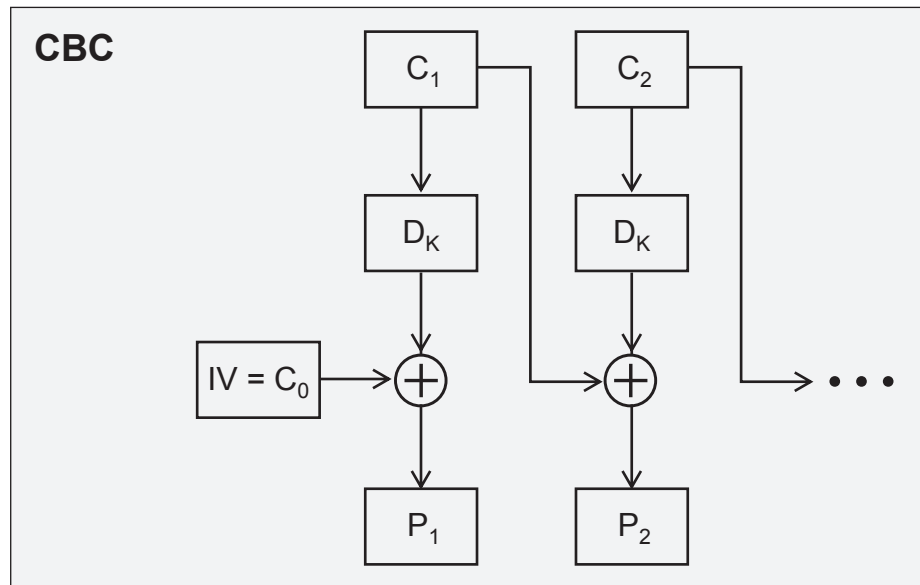
$C_0 = IV$

Keystream element  $Z_i$

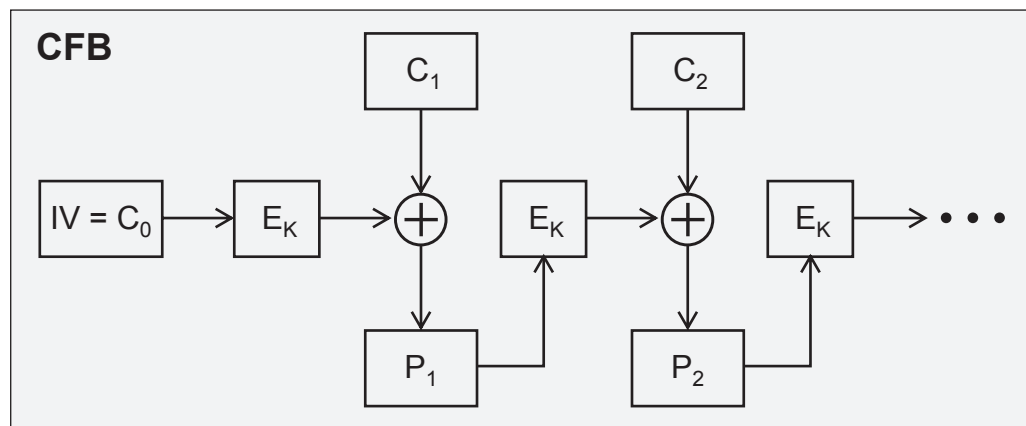
$$\begin{cases} Z_i \equiv E_k(C_{i-1}), & i \geq 1, & i = 1, 2, \dots, n \\ C_i \equiv P_i \oplus Z_i, & i \geq 1, & i = 1, 2, \dots, n \end{cases}$$

# DECRYPT

## DECRYPT



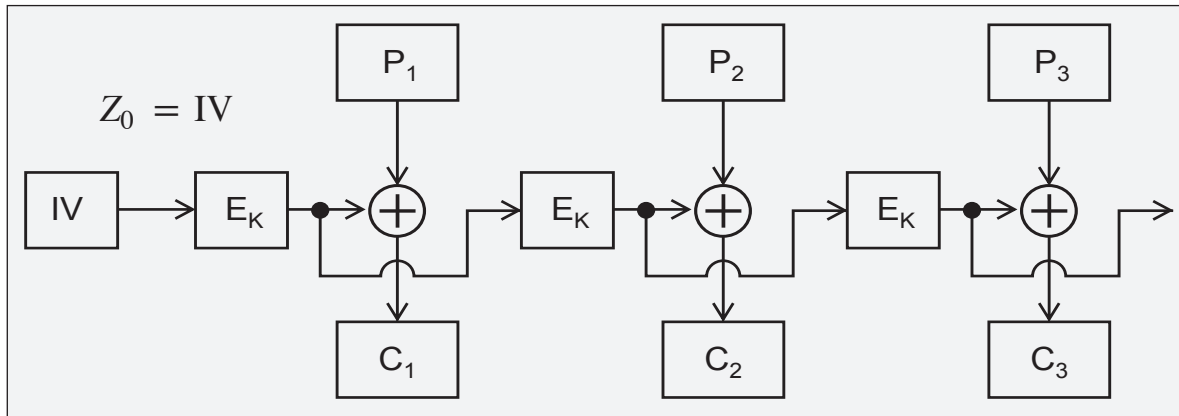
$$P_i = C_{i-1} \oplus D_k(C_i) \quad i \geq 1 \quad i = 1, 2, \dots n$$
$$C_0 = IV$$



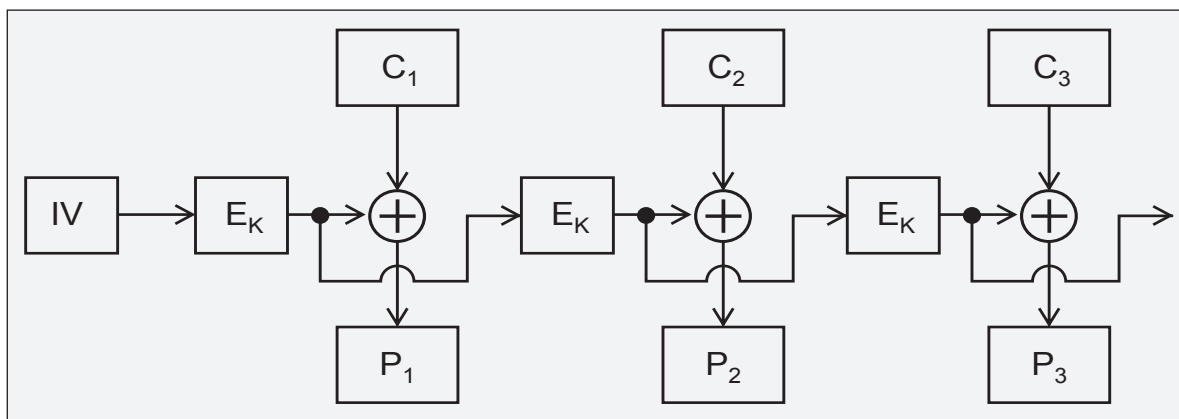
$$P_i = E_k(C_{i-1}) \oplus C_i \quad i \geq 1 \quad i = 1, 2, \dots n$$
$$C_0 = IV$$

# OUTPUT FEEDBACK MODE

$$\begin{cases} Z_i = E_k(Z_{i-1}) & i = 1, 2, \dots, n \\ \text{per } i = 1, & Z_0 = IV \\ C_i = P_i \oplus Z_i \end{cases}$$



$$\begin{cases} P_i = E_k(Z_{i-1}) \oplus C_i & i = 1, 2, \dots, n \\ Z_0 = IV \end{cases}$$



---

# DES / INTEGRITÀ DEL MESSAGGIO A CHIAVE SEGRETA

---

Message Authentication Code

MAC

Senza segretezza

- ❶  $\boxed{\text{Bob}} \rightarrow (P, \text{MAC}) \rightarrow \boxed{\text{Alice}}$  controlla  
Bob usa  $\text{MAC} = E_k(P) \quad E_k(P) = \text{MAC}$

one-time-DES

OK!

Alice si convince che il testo in chiaro  $P$  è ‘integro’  
e non è stato modificato da Oscar  
troppe coppie ‘ $P$ ’  $\leftrightarrow$  ‘ $C$ ’  
per Oscar

- ❷ Bob usa ora una CBC-DES  
Initialization Vector  $\text{IV} \equiv \text{tutti } 0$   
Bob ha  $P_1, P_2, \dots P_n$   
e calcola  $C_1, C_2, \dots C_n$   
con chiave  $K$  e modalità CBC

$\text{MAC} = C_n$   
Bob manda  $\{(P_1, P_2, \dots P_n), \text{MAC}\}$

Alice verifica  
Riceve  $\downarrow \downarrow \downarrow$   
 $(P_1, P_2, \dots P_n)$

$\text{MAC} = C_n$

costruisce  
 $C_i = E_k(P_i) \quad C_1, C_2, \dots C_n$   
e verifica  $C_n = \text{MAC ricevuto}$

Oscar/Trudy non può produrre il MAC perché non conosce  
la chiave segreta  $K$  di Alice e Bob.

Se Trudy intercetta  $P_1, P_2, \dots P_n + \text{MAC}$

se combina un  $P_i$  ( $1 \leq i \leq n$ ) non sa combinare il MAC  
e Bob se ne accorge.



---

# MAC / AUTENTICAZIONE DEI MESSAGGI CON SEGRETEZZA

---

Alice e Bob hanno due chiavi segrete  $K_1$  e  $K_2$

Alice usa  $K_1$  per produrre MAC da  $P_1, P_2, \dots P_n$

poi dice che MAC calcolato

$$\text{MAC} = C_n = E_{k1}(P_n) = P_{n+1}$$

$$\boxed{C_n = \text{MAC} = P_{n+1}}$$

Alice manda poi

$$E_{k2}(P_1, P_2, \dots P_n, P_{n+1})$$

Bob usa  $K_2$  e trova

$$P_1 \div P_n \text{ e } P_{n+1}$$

usa  $K_1$  per trovare  $C_n = \text{MAC}$ :

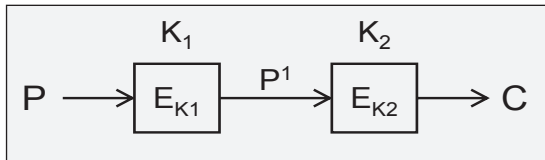
$$P_{n+1} = C_n = E_{k1}(P_n)$$

check OK!

Alice può anche scambiare l'ordine delle chiavi  $K_1$  e  $K_2$

---

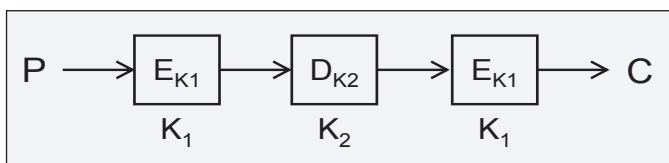
## DES IN CASCATA



DOPPIA CON DUE DIVERSE CHIAVI  $K_1$  e  $K_2$

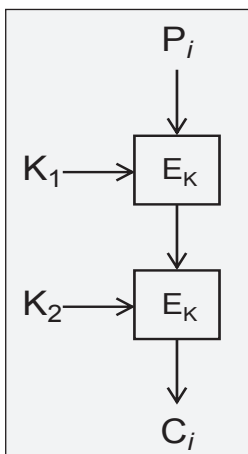
soggetta a Meet-in-the-Middle

## TRIPLA CON DUE CHIAVI



Meet-in-the-Middle Attack

Modalità ECB



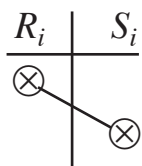
$$C_i = E_{k_2} [E_{k_1} (P_i)]$$

$$D_{k_2}(C_i) = E_{k_1} (P_i)$$

# Meet-in-the-Middle Attack

❶ Calcolo  $R_i = E_i(P_i)$   
per tutti i  $2^{56}$  valori di  $i$   
e costruisco una tabella ascendente per i valori di  $R_i$

❷ Calcolo  $S_j = D_j(C_j)$   
per tutti i  $2^{56}$  valori di  $j$   
e costruisco una tabella ascendente per i valori di  $R_j$



❸ Cerco l'equivalenza in coppie  $i-j$  di chiavi  $\begin{cases} i = K_1 \\ j = K_2 \end{cases}$  che rende:

$$D_j(C_j) = E_i(P_i)$$

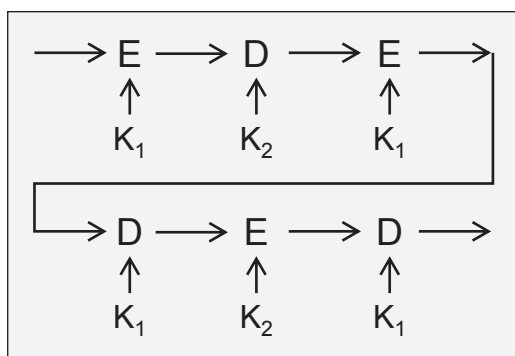
❹ Controllo se dati  $P_1, P_2 \rightarrow C_1, C_2$   
 $E_j[E_i(P_2)]$  è equivalente a  $C_2$   
se lo è prova per tutte le coppie chiaro-cifrato  
tale attacco richiede

$$2 \times 2^{56}$$

operazioni di codifica decodifica  
e  $2^{60}$  BYTE di memoria per le due tabelle

A tre in cascata (TRIPLE DES)

← Il Bancomat  
usa questo



Triple DES  
{ si dice a  
112 BIT  
di sicurezza }

può essere  $K_1 = K_2 = K$   
e dà compatibilità con SINGLE STAGE DES

---

# PUBLIC KEY CRYPTOGRAPHY

---

I sistemi a chiave segreta o privata (simmetrica)

Alice e Bob scelgono segretamente la chiave  $K$  e quindi

$$E_k \text{ e } D_k$$

$D_k$  è lo stesso di  $E_k$  o molto semplicemente derivabile da  $E_k$ .

---

Alice e Bob si scambiano l'informazione sulla chiave  $K$  da usare, prima della comunicazione, sul canale sicuro.

⇒ Avendo esposto  $E_k$  trovare  $D_k$  è possibile  
e quindi rende il sistema poco sicuro.

---

I sistemi a chiave pubblica (asimmetrica)

sono basati sul fatto di ideare crittosistemi ove è computazionalmente impossibile determinare  $D_k$  dato  $E_k$ .

Allora  $E_k$  è disponibile al PUBBLICO IN UN ELENCO (DIRECTORY)

Alice usa  $E_k$  e Bob è l'unico che può decrittare perchè conosce  $D_k$ .

---

# CRYPTOSYSTEM

---

DIFFIE e HELLMAN (1976)

RIVEST (1977)

SHAMIR

ADLEMAN

RSA

Basato sulla difficoltà di fattorizzare numeri interi molto grandi  
(100 – 200 – 300 cifre decimali)

ElGamal

è basato sulla difficoltà di trovare logaritmi discreti

Sistema a chiave pubblica

Oscar osserva il testo  $C^*$  cifrato

Prova a codificare ogni possibile testo in chiaro  $P$

usando la chiave pubblica  $E_k, P_1, P_2, \dots$

finché trova

$$C^* = E_k(P^*)$$

cioè decifra  $C^*$  in  $P^*$ .

---

La sicurezza computazionale di questi crittosistemi va verificata!

---

Si pensi a questo sistema come una

(Scappatoia)      TRAPDOOR      (one-way function)

Funzione unidirezionale con porta di uscita dalla trappola

L'algoritmo  $E_k$  di Bob deve essere facile da calcolare.

Calcolare l'inversa  $D_k$  deve essere HARD per chiunque tranne Bob.

EASY TO COMPUTE	}	Proprietà ONE-WAY (senso unico)
HARD TO INVERT		

Non esiste oggi prova di esistenza di funzioni a senso unico.

---

# ALGORITMO RSA

---

$$m = p \times q$$

$$p, q \text{ primi } > 0$$

$$\mathcal{P} = \mathcal{C} = \mathbb{Z}_m$$

---

$$\mathcal{K} = \{(m, p, q, a, b) : m = pq; p, q \text{ primi}$$

$$ab \equiv 1 \pmod{\varphi(m)}\}$$

$$\text{interi } a, b > 0$$

$$\text{e } b \perp \varphi(m)$$

---

$$\text{Per } K = (m, p, q, a, b)$$

$$E_k(P) = P^b \pmod{m}$$

$$D_k(C) = C^a \pmod{m}$$

$$P, C \in \mathbb{Z}_m$$

$$m, b \longrightarrow \text{PUBLIC}$$

$$p, q, a \longrightarrow \text{SECRET}$$

TRAP DOOR

$$m = p \cdot q$$

e quindi

$$\varphi(m) = (p-1)(q-1)$$

calcola  $a$  noti  $b$  e  $m$  essendo  $b \perp \varphi(m)$ .

---

# ESEMPIO

---

Bob sceglie  $\begin{cases} p = 3 \\ q = 11 \end{cases}$  numeri primi

→  $m = 3 \times 11 = 33$

$\varphi(m) = 2 \times 10 = 20 = 2^2 \times 5$

\* SCELGO  $b = 7$  (primo con 2 e 5)  $b \perp \varphi(m)$   
 $\text{mcd}(7, 20) = 1$

allora devo codificare con  $b$

Bob trova  $b^{-1} \Rightarrow a$

$b^{-1} = a = b^{\varphi(20)-1} \bmod 20 =$

$\varphi(20) = \varphi(\varphi(m)) = 8$

$b^{-1} = a = 7^8 \bmod 20 = 3$

$\lfloor \frac{823543}{20} \rfloor = 41177$

823540  $\boxed{b^{-1} = 3} \longrightarrow \boxed{a = 3}$  ← chiave segreta di Bob

$3 \times 7 = 21 \rightarrow 21 \bmod 20 = 1$

---

BOB PUBBLICA  $m = 33$   
 $b = 7$

---

quindi chiunque può mandare  $C = P^7 \bmod 33$

---

SOLO BOB DECODIFICA  $P = C^3 \bmod 33$

---

Alice manda, per esempio, la lettera  $S \rightarrow 19$

$P = 19$

$P^7 = 19^7 = \lfloor \frac{893871739}{20} \rfloor = 27087022 \times 33$

$= 893871726$

messaggio di Alice  $C = 13$

Bob decodifica usando  $a = 3$

$13^3 = \lfloor \frac{2197}{33} \rfloor = 66 \times 33$

$\overline{2178}$

$19 \rightarrow S$

---

# RSA

---

Bob • genera due primi grandi (200 cifre decimali almeno)

$$p \text{ e } q$$

$$\bullet \text{ calcola } m = p \times q \text{ e } \varphi(m) = (p-1)(q-1)$$

$$\bullet \text{ sceglie a caso } b$$

$$0 < b < \varphi(m)$$

• in modo che sia

$$\text{mcd}[b, \varphi(m)] = 1$$

• calcola

$$a = b^{-1} \bmod \varphi(m)$$

con l'algoritmo di Euclide esteso

• pubblica

$$m \text{ e } b.$$

---

ATTACCO del crittoanalista

Ha  $m \rightarrow$  fattorizzazione in numeri primi;

Quanti primi ci sono tra 0 e  $m$ ?

$$\Pi(m) \rightarrow \frac{m}{l_n m}$$

$$\frac{10^{200}}{460} \cong 10^{197} \quad \text{sono tanti!}$$

$$\begin{cases} m = 10^{200} \\ l_n m \cong 460,5 \end{cases}$$

La fattorizzazione in DUE primi

molto difficile da calcolare

$$p \text{ e } q \quad \text{a 150 cifre decimali}$$

$$m = p \times q \quad \text{a 300 cifre decimali}$$

$$p, q \cong 80 \text{ cifre}$$

$$m \cong 160 \text{ cifre}$$

troppo piccoli

$$512 \text{ BIT RSA} \quad 2^{512} \cong 1,3 \cdot 10^{154}$$

$$1024 \text{ BIT RSA} \quad 2^{1024} \cong 1,7 \cdot 10^{308}$$

2048 BIT RSA = OK



---

# ALTRO ESEMPIO

## CRITTOSISTEMA RSA

---

Bob sceglie

$$\begin{cases} p = 101 \\ q = 113 \end{cases}$$

$$m = 11413$$

$$\varphi(m) = 100 \times 112 = 11200$$

$$11200 = 2^6 \cdot 5^2 \cdot 7$$

posso scegliere  $b$  primo relativo a  $\varphi(m)$ ,  
cioè  $b$  non deve essere divisibile per 2, 5 o 7.

$$b \perp \varphi(m)$$

Verifica

$$\text{mcd}[\varphi(m), b] = 1$$

Bob sceglie

$$b = 3533$$

quindi

$$b^{-1} = 6597 \bmod 11200$$

e quindi la chiave segreta di Bob è

$$a = 6597 = b^{-1}$$

Bob pubblica

$$\begin{cases} m = 11413 \\ b = 3533. \end{cases}$$

Alice manda

9726 a Bob

$$9726 \equiv P$$

calcola

$$9726^{3533} \bmod 11413 = 5761$$

$$5761 = C$$

Bob riceve e calcola

$$5761^{6597} \bmod 11413 = 9726.$$

---

## SICUREZZA

Bob ha una TRAPDOOR  
e quindi  
e calcola “ $a$ ” il decrittore

$$E_k(P) = P^b \bmod m$$

funzione a senso unico

$$\boxed{m = p \cdot q}$$
$$\varphi(m) = (p-1)(q-1)$$
$$a = b^{-1} \bmod [\varphi(m)]$$

Controlliamo che  $E$  e  $D$  siano operazioni inverse.

$$a \equiv b^{-1} \pmod{\varphi(m)}$$
$$a b \equiv 1 \pmod{\varphi(m)}$$
$$a b = t \varphi(m) + 1, \quad b \perp \varphi(m)$$

$t$  intero  $\geq 1$   
per ogni  
si ha

$$P \in Z_m^*$$
$$(P^b)^a \equiv [P^{t\varphi(m)+1}] \pmod{m}$$
$$\equiv [(P^{\varphi(m)})^t P] \pmod{m}$$
$$\equiv [1^t P] \pmod{m}$$
$$\equiv P \pmod{m}.$$

cioè $0 < P < m$ è primo con $m$ $P \perp m$
---

{ Teorema di Lagrange: se  
{ allora

$$P \in Z_m^*$$
$$P^{\varphi(m)} \bmod m = 1$$

---

# SCHEMI DI FIRMA DIGITALE

---

Firma convenzionale

Firma elettronica

Messaggio  $M$

Firma  $\rightarrow$  Algoritmo di firma

SIG

É SEGRETO

$A = \text{SIG}(M)$

$M\|A$  vengono spediti

al destinatario “separatamente”

la verifica è elettronica

Verifica  $\rightarrow$  Algoritmo di verifica

VER

É PUBBLICO

$\text{VER}(M\|A) = \begin{cases} \rightarrow \text{TRUE} \\ \rightarrow \text{FALSE} \end{cases}$

tutti possono esattamente verificare.

La copia elettronica è come l'originale per evitare di essere riusata (reply attack)  
deve contenere un **TIMESTAMP**, cioè la data, l'ora, ad esempio.

---

# SCHEMA DI FIRMA

---

⊕ Algoritmo di firma

SEGRETO

$P, \text{ SIG } (P)$

$A = \text{SIG } (P)$

Spedisco

$P \| A$

⊕ Algoritmo di verifica

PUBBLICO

$$\text{VER } (P, A) = \begin{cases} \text{TRUE} \\ \text{FALSE} \end{cases}$$

se la firma è vera o falsa

## DEFINIZIONE

Lo schema di firma è una quintupla

$(\mathcal{P}, \mathcal{A}, \mathcal{K}, S, \mathcal{V})$

- $\mathcal{P}$  = insieme finito dei messaggi  $P \in \mathcal{P}$
- $\mathcal{A}$  = insieme finito delle firme  $A \in \mathcal{A}$
- $\mathcal{K}$  = insieme finito delle chiavi  $K \in \mathcal{K}$

Keyspace

Per ogni  $K \in \mathcal{K}$  c'è un algoritmo di firma

$\text{SIG}_K \in S$

ed un corrispondente algoritmo di verifica

$\text{VER}_K \in \mathcal{V}$

---

Ogni

$\text{SIG}_K$  comporta  $\mathcal{P} \longrightarrow \mathcal{A}$

Ogni

$\text{VER}_K$  comporta  $\mathcal{P} \times \mathcal{A} \longrightarrow \{\text{vero}, \text{falso}\}$

---

Sono funzioni tali che per ogni messaggio  $P \in \mathcal{P}$  e per ogni firma  $A \in \mathcal{A}$

$$\text{VER}_K (P, A) = \begin{cases} \text{vero} & \text{se } A = \text{SIG}_K(P) \\ \text{falso} & \text{se } A \neq \text{SIG}_K(P) \end{cases}$$

---

# SCHEMA DI FIRMA RSA

---

Sia	$m = p \cdot q$ ,	$p$ e $q$ primi
	$\mathcal{P} \equiv \mathcal{A} \equiv \mathbb{Z}_m$	
	$\mathcal{K}, \equiv \{(m, p, q, a, b) : m = p \cdot q; p, q \text{ primi}$	
	$a \cdot b \equiv 1 \pmod{\varphi(m)}\}$ , $b \perp \varphi(m)$	$a, b > 0$
I valori $m$ e $b$ sono pubblici,		
e $p, q$ e $a$ sono segreti.		
Per	$k = m, p, q, a, b$	
(segreta)	$\text{SIG}_K(P) = P^a \bmod m = A$	decifratura RSA
e		
(pubblica)	$\text{VER}_K(P, A) = \text{true} \leftrightarrow$	
	vero se	
	$P \equiv A^b \pmod{m}$	cifratura RSA
$(P, A \in \mathbb{Z}_m)$		

---

## FIRMA RSA

Quindi Bob firma il messaggio  $P$

con la regola RSA di decrittaggio  $D_K$

$$D_K = \text{SIG}_K \text{ è segreto } (a \text{ segreto: } a = b^{-1})$$

La verifica usa RSA crittaggio  $E_K$

Chiunque può verificare la firma dato che  $E_K$  è PUBBLICA. ( $m$  e  $b$  pubblici)

---

# SEGRETARE TESTO E FIRMA

---

Alice vuole combinare FIRMA e TESTO CIFRATO con un RSA

Dato un testo in chiaro  $P$  Alice calcola la sua firma

(segreta)

$$A = \text{SIG}_{\text{ALICE}}(P)$$

poi Alice cifra il tutto con  $m$  e  $b$

(pubblica)

$$E_{\text{BOB}}(P, A) = Z$$

FIRMARE PRIMA DI CRITTOGRAFARE!

{ è la decodifica  
di RSA con  $a$  e  $m$

{ è la codifica di RSA.

Alice quindi spedisce

→  $Z$  va a Bob

Prima Bob

(segreta)

$$D_{\text{BOB}}(Z)$$

ottiene

$P$  e  $A$

Poi Bob fa

(pubblica)

$$\text{VER}_{\text{ALICE}}(P, A) = \text{true}$$

---

che succede se Alice prima crittografa e poi firma?

Alice calcola

$$C = E_{\text{BOB}}(P)$$

$$A = \text{SIG}_{\text{ALICE}}[E_{\text{BOB}}(P)]$$

$$A = \text{SIG}_{\text{ALICE}}(C)$$

e manda  $(A, C)$ .

Bob riceve  $A$  e  $C$

calcola

$$P = D_{\text{BOB}}(C)$$

poi

$$\text{VER}_{\text{ALICE}}(C, A) = \text{true}$$

---

Ma Oscar ottiene  $A$  e  $C$  e li blocca  
ora fa così, rimpiazza  $A$  con la sua

e manda  $A' = \text{SIG}_{\text{OSCAR}} [E_{\text{BOB}}(P)]$  (segreta)  
 $(A', C)$  a Bob

Bob decrittà  $P = D_{\text{BOB}}(C)$   
e verifica  $\text{VER}_{\text{OSCAR}}(C, A') = \text{true}$  (pubblica)  
e crede che il messaggio lo ha mandato Oscar

---

PLEASE  
SIGNING BEFORE ENCRYPTING!

---

Oppure  
FUNZIONI HASH

$$P \rightarrow MD$$
$$\downarrow$$
$$\text{SIG}(MD)$$
$$P \parallel \text{SIG}[H(P)] \text{ nudo } P, \text{ SIG}(MP)$$

---

# METODO DI FIRMA

## PUBLIC KEY

---

ElGamal

DSA Digital Signature Algorithm

- messaggio 160 BIT
- firma da 320 BIT

RSA-Fattorizzazione numeri primi

DSA-Calcolo dei logaritmi discreti

---

Sia  $p$  primo tale che il problema di trovare i logaritmi discreti in  $\mathbb{Z}_p^*$  sia intrattabile.

Sia  $\alpha \in \mathbb{Z}_p^*$  un elemento primitivo

Sia  $\mathcal{P} = \mathbb{Z}_p^*$ ,  $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$

e sia

$\mathcal{K} = \{(p, \alpha, a, \beta) \text{ tale che } \beta \equiv \alpha^a \pmod{p}\}$

•  $p, \alpha$  e  $\beta$  PUBBLICI

•  $a$  SEGRETO

$$a = \log_{\alpha}^D \beta$$

Allora per  $\mathcal{K} = (p, \alpha, a, \beta)$  e per un numero segreto  $k$

$k \in \mathbb{Z}_{p-1}^*$  casuale

(da usare one time)

$k \perp (p-1)$ , sia:



---


$$\Rightarrow \left. \begin{array}{l} \text{ove} \\ \text{SIG}_K(P, k) = (\gamma, \delta) \\ \bullet \gamma = \alpha^k \bmod p \\ \bullet \delta = [(P - a\gamma)k^{-1}] \bmod (p-1) \end{array} \right\} \begin{array}{ll} a \text{ e } k & \text{SEGRETI} \\ P, \alpha, \beta & \text{PUBBLICI} \end{array}$$

per

$$P, \gamma \in \mathbb{Z}_p^*$$

$$\delta \in \mathbb{Z}_{p-1}$$

sarà

$$\text{VER}(P, \gamma, \delta) = \begin{cases} \text{vero} & \text{se } \beta^\gamma \gamma^\delta \equiv \alpha^P \pmod{p} \\ \text{falso, altrimenti} & \end{cases}$$


---

infatti

$$\begin{aligned} \beta^\gamma \gamma^\delta &\equiv \alpha^{a\gamma} \alpha^{k\delta} \pmod{p} \\ &\equiv \alpha^{(a\gamma + k\delta)} \pmod{p} \\ &\equiv \alpha^P \pmod{p} \end{aligned}$$

ove

$$a\gamma + k\delta \equiv P \pmod{p-1}.$$

Bob calcola la firma usando

$a$  = parte della chiave  
 $k$  = numero casuale usato  
per un SINGOLO messaggio

---

Esempio:

$$\begin{array}{ll} p = 467 \\ \alpha = 2 \\ a = 127 \end{array} \quad \text{PUBBLICI} \quad \begin{cases} p = 467 \\ \alpha = 2 \\ \gamma = 132 \end{cases}$$

Bob ora  
e  
allora

$$\begin{array}{ll} \beta = \alpha^a \bmod p = 2^{127} \bmod 467 = 132 \\ P = 100 \\ k = 213 \end{array} \quad \text{SEGRETI} \quad \begin{cases} a = 127 \\ k = 213 \end{cases}$$

$$\begin{aligned} \gamma &= 2^{213} \bmod 467 = 29 \\ \delta &= [(100 - 127 \times 29) 431] \bmod 466 = 51 \end{aligned}$$

Chiunque verifica

e quindi

$$\begin{aligned} 132^{29} 29^{51} &\equiv 189 \pmod{467} \\ 2^{100} &\equiv 189 \pmod{467} \end{aligned}$$

# DSA

## DIGITAL SIGNATURE ALGORITHM

Digital Signature Algorithm usa ElGamal

messaggio  $P$  lungo 160 BIT

$p, \alpha$  e  $\beta$  pubblici  
 $a$  segreto

scelgo  $k$  casuale  $\in Z_{p-1}^*$  segreto

$$\text{SIG}_K(P, k) = (\gamma, \delta)$$

ove  $\gamma = \alpha^k \text{mod } p$   $k$  segreto

$$\delta = (P - a\gamma)k^{-1} \text{mod}(p-1) \quad k \text{ segreto, } a \text{ segreto}$$

$$\text{VER}(P, \gamma, \delta) = \beta^\gamma \gamma^\delta \equiv \alpha^P \text{(mod } p)$$

Richiede

una firma DSA  $(\gamma, \delta)$  di 320 BIT.

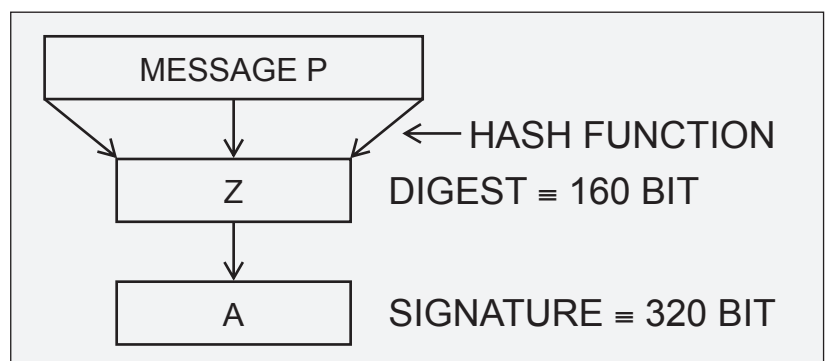
Allora per messaggi lunghi si fa così

$P$  – lunghezza arbitraria

Lungo messaggio

‘digest’ del messaggio

Firma



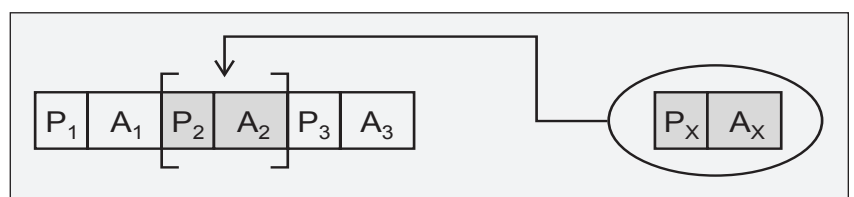
To hash = tagliare (la carne) in dadini

uguali

Problema con le firme

perché è facile cancellare o variare l'ordine.

INTEGRITÀ



---

# DSA

---

Digital Signature Algorithm usa ElGamal Public Key System

messaggio corto di 160 BIT      $P$  – 160 BIT  
usa una Firma da 320 BIT      $(\gamma, \delta)$  320 BIT  
    $SIG_K(P, k) = (\gamma, \delta)$

---

messaggi corti

FUNZIONE HASH PUBBLICA

---

messaggio $P$	$P$ – lunghezza arbitraria
compute	
message DIGEST $z = h(P)$	160 BIT
then apply	
SIGNATURE	$A = SIG(z)$ 320 BIT
ricordi	$\gamma$ e $\delta \equiv SIG(z)^k$ $160 \times 2 = 320$ BIT

---

	$z = h(P)$	FUNZIONE HASH UNIDIREZIONALE	PUBBLICA
Bob	vuole formare un messaggio	$P$	
	costruisce	$z = h(P)$	
	poi calcola	$A = SIG_k(z)$	
	manda	$(P, A) =$	
Chiunque	calcola	$z = h(P)$	
	e poi	$VER_K(z, A) = TRUE$	

---

Una funzione hash

$$z = h(P)$$

è unidirezionale se dato un message digest  $z$   
è impossibile calcolare  $P$  tale che

$$h(P) = z$$

---

messaggio  $P$

Bob vuole firmare il messaggio  $P$

calcola

- ❶  $z = h(P)$       message digest
- ❷  $A = \text{SIG}_k(z)$     la firma segreta
- ❸ manda  $P$  e  $A$

$P \parallel A$

Alice

calcola

❶  $z = h(P)$

e poi

❷  $\text{VER}_K(z, A) = \text{vero se } z \equiv A^b \pmod{m}$   
ad es. cifrato RSA

$$\text{PUBBLICHE} \quad \begin{cases} y = h(x) & \text{HASH} \\ b, m & \text{RSA} \end{cases}$$

---

## ATTACCO DI OSCAR

alle funzioni hash

intercetta

$P \parallel A$  valido di Bob

$A = \text{SIG}_{\text{BOB}}[h(P)]$

---

poi calcola

$z = h(P)$

PUBBLICA

e cerca

$P' \neq P$

tale che

$h(P') = h(P).$

Allora Oscar manda

$(P', A)$

è valido firmato da Bob!  
è una FALSIFICAZIONE.

---

❶  $h$  deve soddisfare la proprietà detta SENZA COLLISIONI (collision free)

Dato  $P$  trovare  $P'$

$h$  è debolmente senza collisioni se è computazionalmente intrattabile

il calcolo di  $P' \neq P$  tale che  $h(P') = h(P).$

---

## ALTRO ATTACCO

Oscar trova due messaggi       $P'$  e  $P$   
    $P' \neq P$   
tali che                               $h(P') = h(P)$

---

Oscar manda  $P$  a Bob e lo convince a firmare

Bob lo firma

calcola                               $Z = h(P)$   
lo firma                             $A_{\text{BOB}} = \text{SIG}_{\text{BOB}}(Z)$   
riceve                                $P, A_{\text{BOB}}$

---

successivamente

Oscar manda                       $P', A_{\text{BOB}}$       è valida!  
e falsifica la firma di Bob.

- 
- ② Una funzione hash è fortemente senza collisioni  
se è computazionalmente intrattabile trovare due messaggi

$$P \text{ e } P'$$

tali che  $P \neq P'$

e  $h(P) = h(P')$ .

Trovare  $P$  e  $P'$  tali che.

---

Infine

- ③ Una funzione hash è unidirezionale (one-way)  
se dato un digest di messaggio  $z$   
è computazionalmente impossibile trovare un messaggio  $P$   
tale che  $h(P) = z$ .

Message digest ( $P = 512$  BIT FISSI)

- MD4 64 BIT hash function
- MD5 128 BIT

Message digest ( $P < 2^{64}$  BIT)

SHA 160 BIT Secure Hash Algorithm.


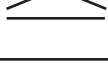
---

# LEGGE ITALIANA

---

DIGEST → IMPRONTA

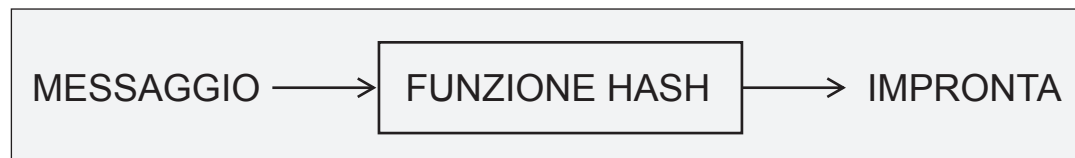
Chiavi a 1024 BIT

Generazione	firme digitali		RSA
Verifica	firme digitali		DSA

---

Algoritmi di hash	RIPEMD – 160	160 BIT
$P$ variabile fino a $2^{64}$ BIT	SHA – 1	Secure Hash Algorithm

---



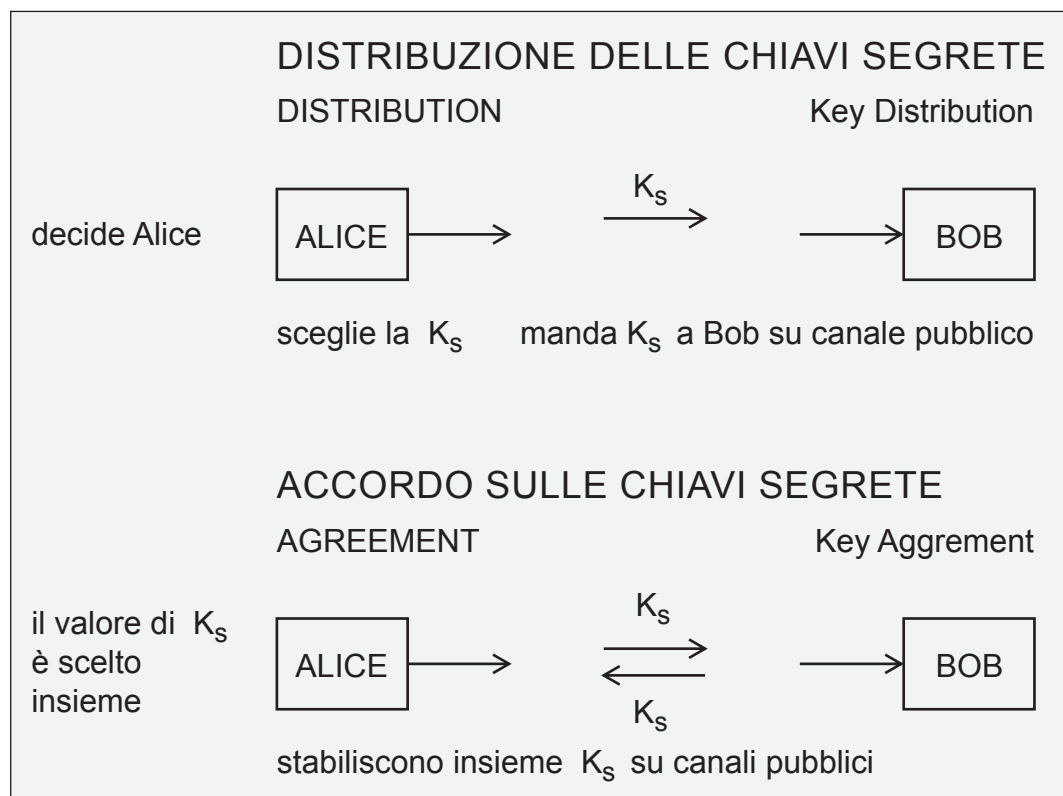
Digest	→	sommario	To digest	digerire
↓	→	riassunto		
↓	→	selezione		

IMPRONTA del MESSAGGIO



# DISTRIBUZIONE DELLE CHIAVI

- I sistemi a chiave pubblica non richiedono canali sicuri per scambiare chiavi segrete come nei sistemi a chiave privata
- I sistemi pubblici sono lenti. RSA rispetto a DES.
- Per cifrare messaggi lunghi bisogna usare chiavi private, quindi bisogna trovare dei metodi per lo scambio delle chiavi segrete



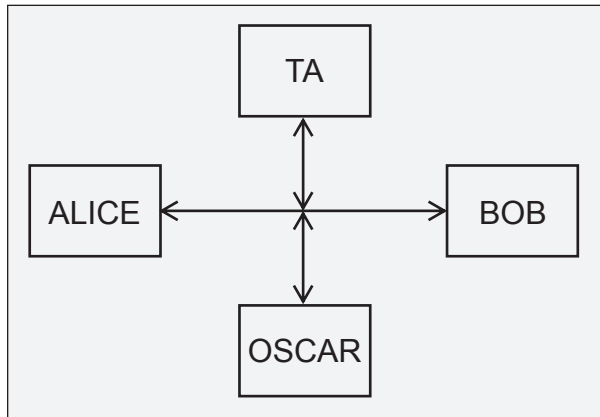
---

# TRUSTED AUTHORITY

TA Trusted Authority

$n$  Utenti

- Verifica l'identità dell'utente
- Sceglie e trasmette chiavi agli utenti



PASSIVO (Oscar)

SPIA

- spia i messaggi

ATTIVO (Oscar = Trudy)

INTRUSO

- altera i messaggi
- conserva i messaggi per uso futuro
- si maschera, al posto di un certo utente si traveste: è un impostore

---

## ❶ Obiettivi dell'intruso

- imbrogliare  $A$  e  $B$  facendogli accettare una “chiave” falsa (vecchia chiave o chiave inventata da Trudy)
- imbrogliare  $A$  e  $B$  facendogli credere che hanno scambiato una chiave tra loro, mentre non l'hanno fatto.

## ❷ Obiettivo della distribuzione delle chiavi o dell'accordo sulle chiavi

è che alla fine del protocollo  $A$  e  $B$  possiedano la stessa chiave  $K$ .

# PREDISTRIBUZIONE DELLE CHIAVI

Per ogni coppia di utenti  $U$  e  $V$

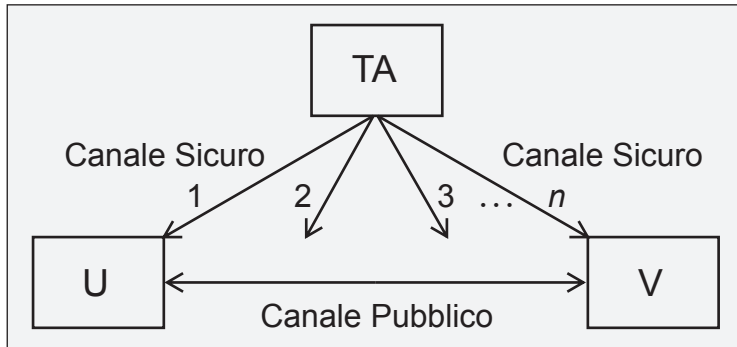
$$\{U, V\}$$

$n$  Utenti

il TA sceglie una chiave a caso

$$K_{u,v} = K_{v,u}$$

e la manda su un canale sicuro



solo  $n$  canali sicuri,  
e non  $\binom{n}{2}$  tra tutti gli utenti

TA genera e trasmette

$\frac{n(n-1)}{2} = \binom{n}{2}$  chiavi e dà ogni chiave ad un'unica coppia di utenti  $U, V$ .

Ogni utente

$$U \longrightarrow \begin{matrix} K_{u,v} \\ K_{u,k} \\ K_{u,h} \end{matrix}$$

ha  $(n-1)$  chiavi

È complicato.

---

# DISTRIBUZIONE ON LINE DELLE CHIAVI DA PARTE DEL TA

---

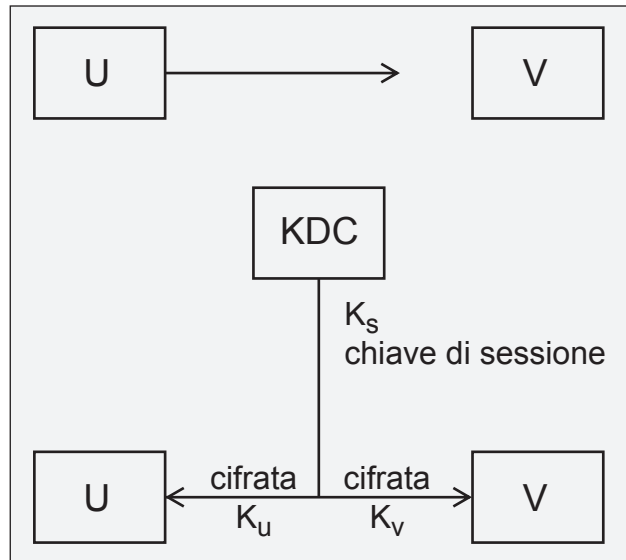
TA è un Key Distribution Center      KDC

KDC ha  $n$  chiavi segrete, una per ciascuno degli utenti della rete

$K_u$       chiave segreta dell'utente  $u$

U vuole parlare con V  
e chiede  
una chiave di sessione

KDC manda a U e V  
la chiave di sessione  
cifrata  $K_u$  e  $K_v$



## KERBEROS

---

oppure l'altro schema è

### PROTOCOLLO D'ACCORDO SULLE CHIAVI

{ Diffie-Hellman  
  Merkle

DISTRIBUITA: senza TA.

---

# KERBEROS KDC

---

KERBEROS è basato su sistemi a chiave privata

- Ogni utente ha una chiave segreta

DES con TA (KERBEROS)  $U \longrightarrow K_u$   
 $V \longrightarrow K_v$

DES – CBC MODE e usa DES per segretezza

ID (U) informazione pubblica di identificazione dell'utente U  
(Nome/Cognome/Indirizzo/Nascita/Luogo/E-mail/Telefono)

TA  $T$  – Timestamp  
 $L$  – Lifetime

genera Session Key  $K$  valida nell'intervallo  
 $(T \div T + L)$ .

---

❶ U chiede a TA di avere una chiave di sessione  $K$  per l'utente V

❷ TA sceglie  $K$  a caso con  $T$  e  $L$

❸ TA calcola

$$\bullet m_1 = E_{K_U}(K, \text{ID}(V), T, L)$$

$$\bullet m_2 = E_{K_V}(K, \text{ID}(U), T, L) \longleftarrow \text{ticket per V}$$

e manda  $m_1$  e  $m_2$  a U

❹ U usa la decrittazione

$$D_{K_u}(m_1) = K, \text{ID}(V), T, L$$

e calcola

$$\bullet m_3 = E_K[\text{ID}(U), T]$$

e lo manda a V insieme al Ticket per V ricevuto da TA :  $m_2$ .

❺ V usa  $D_{K_v}(m_2) = K, \text{ID}(U), T, L$

poi calcola

$$D_K(m_3) = T, \text{ID}(U)$$

Verifica che i due valori di  $T$  coincidono e che i due valori di  $\text{ID}(U)$  coincidono,

e calcola

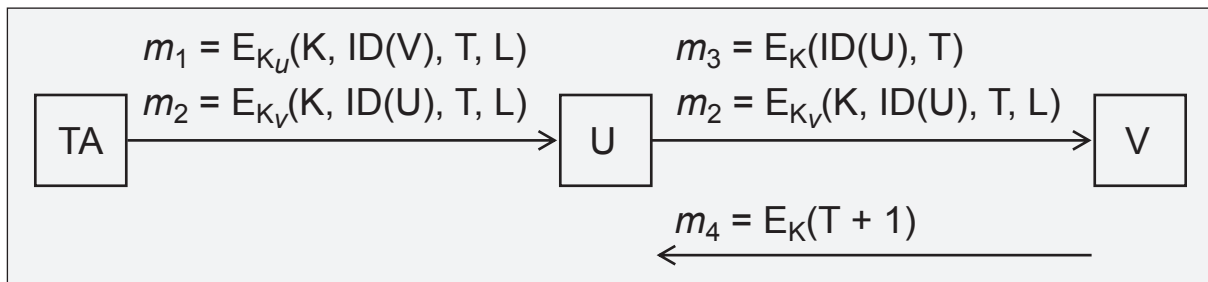
$$\bullet m_4 = E_K(T + 1)$$

e lo manda a U.

---

⑥  $U$  decifra

e verifica  $D_K(m_4) = T + 1$   
 $T + 1$ .



---

SYNCHRONIZED CLOCK (or quasi synchronized)

$T, L$

TO AVOID REPLAY ATTACK

---

Key Transmission Secrecy

$m_1$  &  $m_2$  forniscono segretezza  
nella trasmissione di  $K$

Key Confirmation

$m_3$  &  $m_4$  forniscono riscontro  
della ricezione delle chiavi

---

# SCAMBIO DELLE CHIAVI DI DIFFIE & HELLMAN

---

## KEY AGREEMENT PROTOCOL

Complessità di calcolo dei logaritmi discreti

- ❶  $U$  sceglie  $a_u$  a caso

$$0 \leq a_u \leq p-2$$

- ❷  $U$  calcola

$$b_u = \alpha^{a_u} \bmod p$$

e lo manda a  $V$

- ❸  $V$  sceglie  $a_v$  a caso

$$0 \leq a_v \leq p-2$$

- ❹  $V$  calcola

$$b_v = \alpha^{a_v} \bmod p$$

e lo manda a  $U$

- ❺  $U$  calcola

$$K = (\alpha^{a_v})^{a_u} \bmod p$$

$V$  calcola

$$K = (\alpha^{a_u})^{a_v} \bmod p$$

$$\left\{ \begin{array}{l} p \text{ e } \alpha \\ \text{noti, numeri primi} \\ \text{con } \frac{p-1}{2} \text{ primo:} \\ \alpha \text{ elemento} \\ \text{primitivo di } \mathbb{Z}_p^* \end{array} \right.$$

---

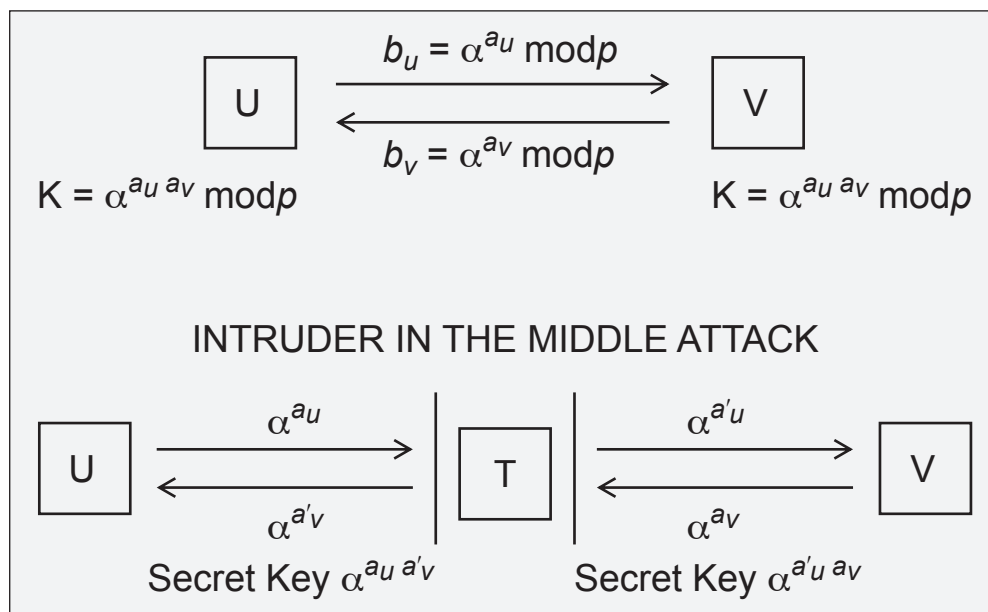
$p$  è numero primo

$\alpha$  è elemento primitivo di  $\mathbb{Z}_p$

Valori noti, pubblicamente

$U$  e  $V$  alla fine hanno calcolato la chiave di sessione

$$K = \alpha^{a_u a_v} \bmod p.$$





---

# PROTOCOLLO DI ACCORDO SULLE CHIAVI AUTENTICATO

---

## Authenticated Key Agreement Protocol

Per lo scambio delle chiavi D&H è quindi necessario che  $U$  e  $V$  siano sicuri della loro identità

### IDENTIFICAZIONE

si usano  $p$ ,  $\alpha$  e i certificati rilasciati dalla TA.

Ogni utente  $U$  ha uno schema di firma elettronica

$SIG_u$	segreto	ad esempio $a_u$
$VER_u$	pubblico	$b_u = \alpha^{a_u}$

mod  $p$   
TA ha il suo

$SIG_{TA}$	e
$VER_{TA}$	pubblico

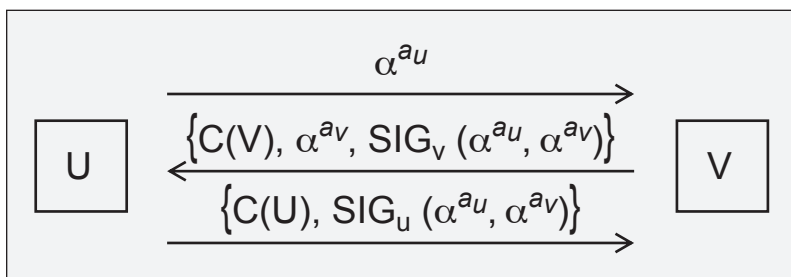
Ogni utente ha un certificato

$$C(U) = \{ID(U); VER_u; SIG_{TA} [ID(U); VER_u]\}$$

$ID(U)$  è ID di  $U$ .

# SIMPLIFIED STATION-TO-STATION PROTOCOL

- ❶  $U$  sceglie  $a_u, \quad 0 \leq a_u \leq p-2$
- ❷  $U$  calcola  $\alpha^{a_u} \bmod p$   
e lo manda a  $V$
- ❸  $V$  sceglie  $a_v, \quad 0 \leq a_v \leq p-2$
- ❹  $V$  calcola  $\alpha^{a_v} \bmod p$   
poi calcola  $K = (\alpha^{a_u})^{a_v} \bmod p$   
e  $y_v = \text{SIG}_v(a^{a_v}, a^{a_u})$
- ❺  $V$  manda  $\{C(V), \alpha^{a_v}, y_v\}$  a  $U$
- ❻  $U$  calcola  $K = (\alpha^{a_v})^{a_u} \bmod p$   
verifica  $y_v$  usando  $\text{VER}_v$   
 $C(V)$  usando  $\text{VER}_{\text{TA}}$   
 $U$  calcola  $y_u = \text{SIG}_u(a^{a_u}, a^{a_v})$   
e manda  $\{C(U), y_u\}$  a  $V$
- ❼  $V$  verifica  $y_u$  usando  $\text{VER}_u$   
e  $C(U)$  usando  $\text{VER}_{\text{TA}}$



L'uso dei certificati evita "impostori"

$$K = \alpha^{a_v a_u} \bmod p$$

---

$p, \alpha$   
 $\text{VER}_v$   
 $\text{VER}_u$   
 $\text{VER}_{\text{TA}}$

---

PUBBLICI

---

$\text{SIG}_v a_v$   
 $\text{SIG}_u a_u$

---

SEGRETI

---

## SCHEMI DI IDENTIFICAZIONE

(Autenticazione dell'identità di una persona)

## CODICI DI AUTENTICAZIONE

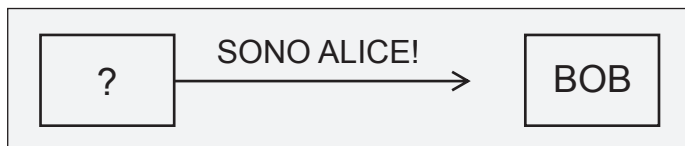
(Autenticare l'integrità di un messaggio e la provenienza)

cioè – non alterato

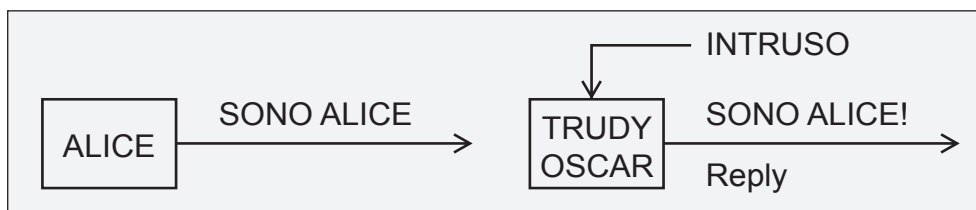
– è stato originato dal presunto trasmettore

## SCHEMI DI IDENTIFICAZIONE

Obiettivi



❶ Come fa Bob a sapere se è veramente Alice?



❷ Alice vuole provare che è Alice, ma evitare che, così facendo, consenta di essere 'impersonata' da Bob più tardi.

AUTORIZZARE: dare il permesso di un'azione a qualcuno già identificato

---

Semplice metodo da realizzare su

SMART CARD

Carta con un chip che fa operazioni aritmetiche

Piccola memoria

Poca potenza di calcolo

Poichè OSCAR può rubare la card, occorre anche un PIN

---

## SISTEMA BASATO SU CRITTOSISTEMI A CHIAVE PRIVATA (DES)

SFIDA E RISPOSTA

Challenge-and-Response

Richiede una chiave segreta condivisa

$K$        $E_k$

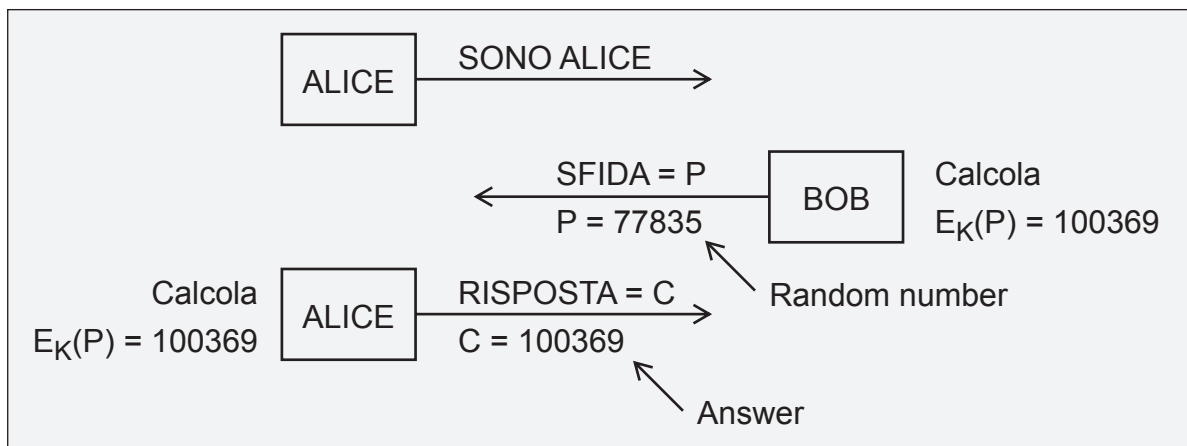
---

Esempio

$$C = E_k(P) = P^{101379} \bmod 167653$$

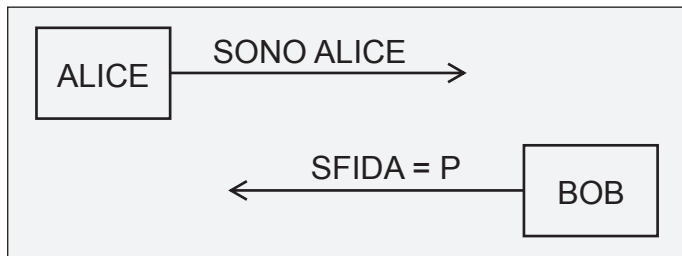
$$K_s(b, m) \quad K_s = (\overset{b}{101379}, \overset{m}{167653})$$

chiave



---

## Challenge-and-Response con chiave segreta



$P$  è un numero casuale a 64 BIT

es.  $P = 77835$   
 $b = 101379$   
 $m = 167653$   
 $K_s = (b, m)$

Bob manda	$P$	
e calcola	$E_{K_s}(P) = P^b \bmod m$	
es. se	$K_s = (b, m) = (101379, 167653)$	
Bob calcola	$x = E_{K_s}(P) = 100369$	
riceve	$P$	
	Risposta = $x'$	
Alice manda	$x' = 100369$	
calcola	$E_{K_s}(P) = 100369$	
		Bob confronta se $x = x'$ è Alice

---

# SCHEMA DI IDENTIFICAZIONE DI SCHNORR

---

C'è la TA che fa così

- 1  $p$  è primo grande  
(il problema di trovare il logaritmo discreto è intrattabile)
- 2  $q$  è un divisore primo grande di  $p$   $0 < q < p - 1, q \in \mathbb{Z}_p^*$
- 3  $\alpha \in \mathbb{Z}_p^*$  primitivo elemento generatore
- 4 un parametro  $t$  tale che  $q > 2^t$  ad esempio  $t = 40$
- 5 TA decide  

$\text{SIG}_{\text{TA}}$

e

$\text{VER}_{\text{TA}}$
- 6 TA decide  

$h(p)$  hash function

$p, q, \alpha$  e  $\text{VER}_{\text{TA}}$  e  $h(p)$  PUBBLICI

---

# CERTIFICAZIONE

---

❶ TA vede Alice e passaporto, crea una stringa  $ID(Alice)$  con le sue informazioni di identità

❷ Alice sceglie  $0 \leq a \leq q-1$  e calcola  $v = \alpha^{-a} \bmod p$   
e dà  $v$  al TA.

❸ Il TA genera la firma

$$s = \text{SIG}_{TA}[ID(A), v]$$

e dà ad Alice il

CERTIFICATO

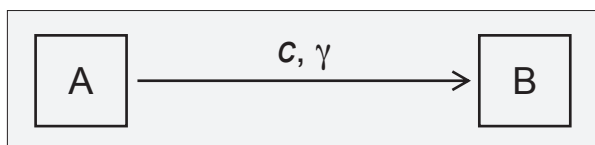
$$C(Alice) = [ID(Alice), v, s]$$

$$\left\{ \begin{array}{l} p, q, \alpha \\ \text{PUBBLICI} \\ a \\ \text{SEGRETO} \\ \text{di Alice} \end{array} \right.$$

# PROVA DELL'IDENTITÀ

- ① Alice sceglie un numero  $k$   $0 \leq k \leq q-1$   
e calcola

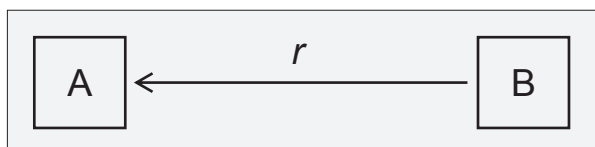
- ② Alice manda a Bob  $C(\text{Alice}) = [\text{ID}(\text{Alice}), v, s]$   
e  $\gamma$
- $$\begin{cases} p, q, \alpha \\ \text{VER}_{\text{TA}} \\ \text{e } h(p) \\ \text{PUBBLICI} \end{cases}$$



- ③ Bob verifica la firma del TA

$$\text{VER}_{\text{TA}}[\text{ID}(\text{Alice}), v, s] = \text{true}$$

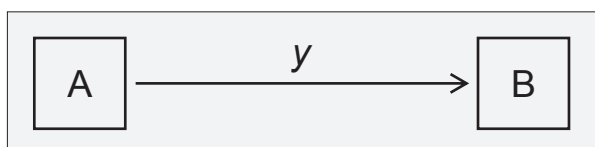
- ④ Bob sceglie  $r$  e lo manda ad Alice numero casuale a  $t$  BIT  
 $1 \leq r \leq 2^t$



- ⑤ Alice calcola

$$y = (k + ar) \bmod q$$

e lo manda



- ⑥ Bob verifica che

$$\gamma \equiv \alpha^y v^r \pmod{p}.$$

Infatti

$$\begin{aligned} \alpha^y v^r &\equiv \alpha^{k+ar} v^r \pmod{p} \\ &\equiv \alpha^{k+ar} \alpha^{-ar} \pmod{p} \\ &\equiv \alpha^k \pmod{p} \\ &\equiv \gamma \pmod{p} \end{aligned}$$



---

# ESEMPIO

---

$$p = 88667$$

$$p \perp q$$

$$q = 1031$$

$$t = 10$$

$\alpha = 70322$  è primitivo in  $\mathbb{Z}_p^*$  di ordine 1031

$$\alpha^{1031} \equiv 1 \pmod{p}$$

ALICE SEGRETO

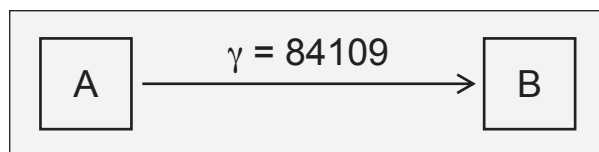
$$a = 755$$

allora

$$\begin{aligned} v &= \alpha^{-a} \pmod{p} = \\ &= 70322^{1031-755} \pmod{88667} = \\ &= 13136 \end{aligned}$$

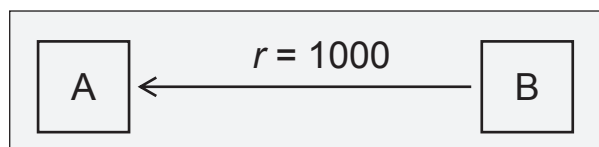
Alice ora sceglie  $k = 543$

$$\begin{aligned} \gamma &= \alpha^k \pmod{p} \\ &= 70322^{543} \pmod{88667} = \\ &= 84109 \end{aligned}$$



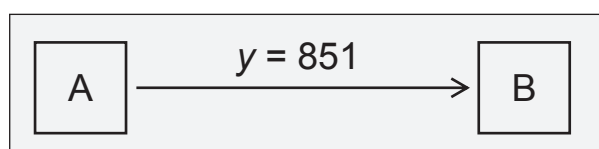
Bob sceglie

$$r = 1000 < 2^{10}$$



Alice calcola

$$\begin{aligned} y &= (k + ar) \pmod{q} \\ &= (543 + 755 + 1000) \pmod{1031} = \\ &= 851 \end{aligned}$$



Bob calcola che

$$\begin{aligned} \gamma &= \alpha^y v^r \pmod{p}. \\ 84109 &\equiv 70322^{851} 13136^{1000} \pmod{88667} \end{aligned}$$