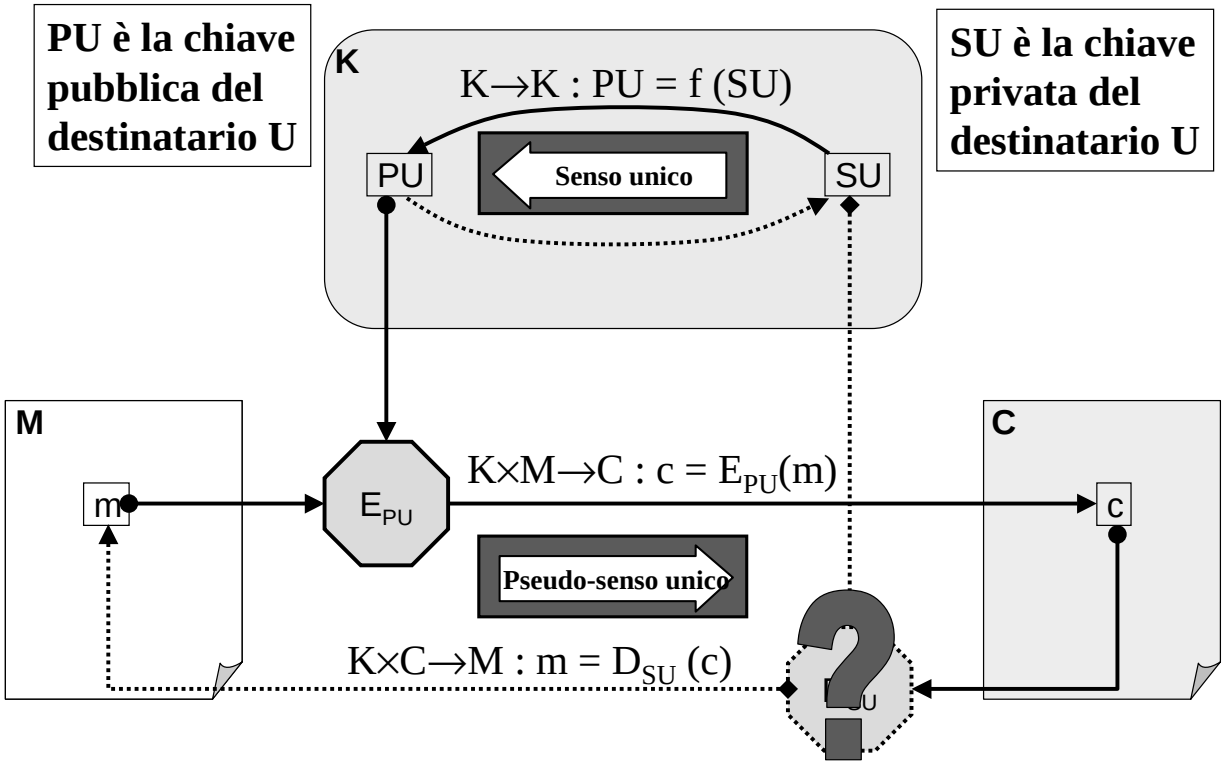
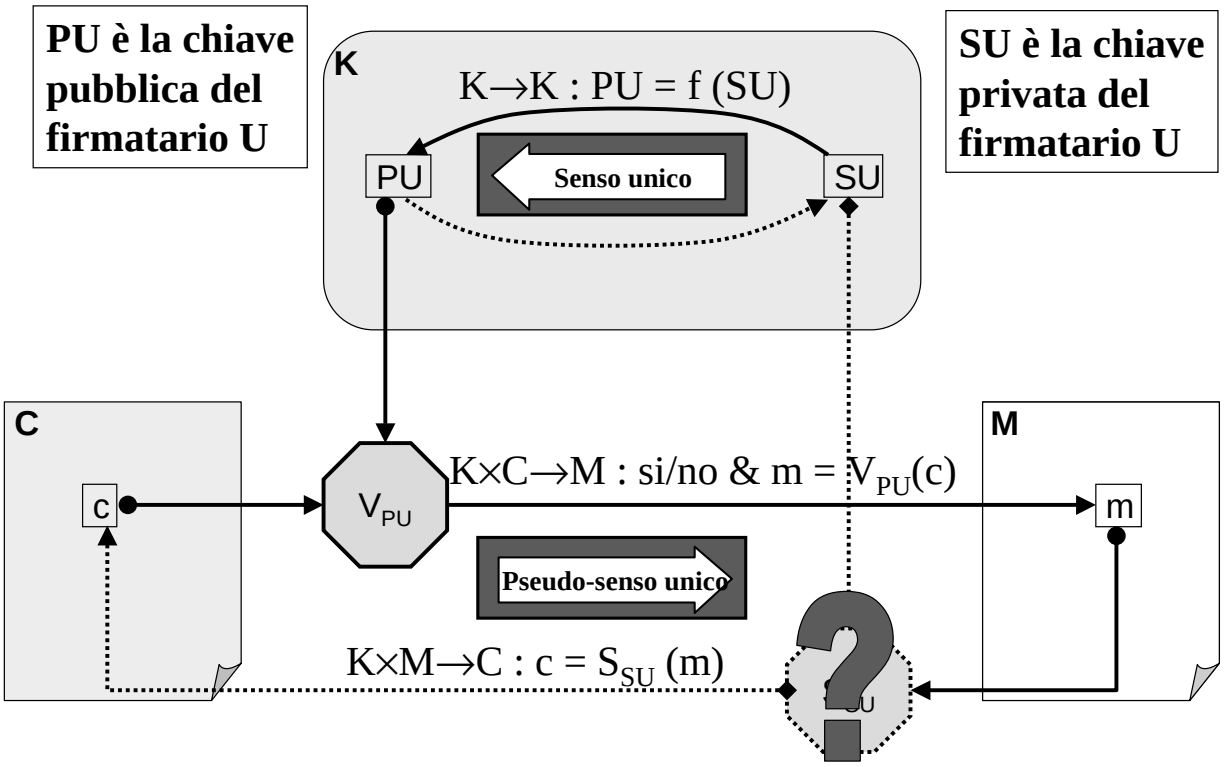
Crittografia asimmetrica

Teoria ed algoritmi

Il modello del Cifrario a chiave pubblica

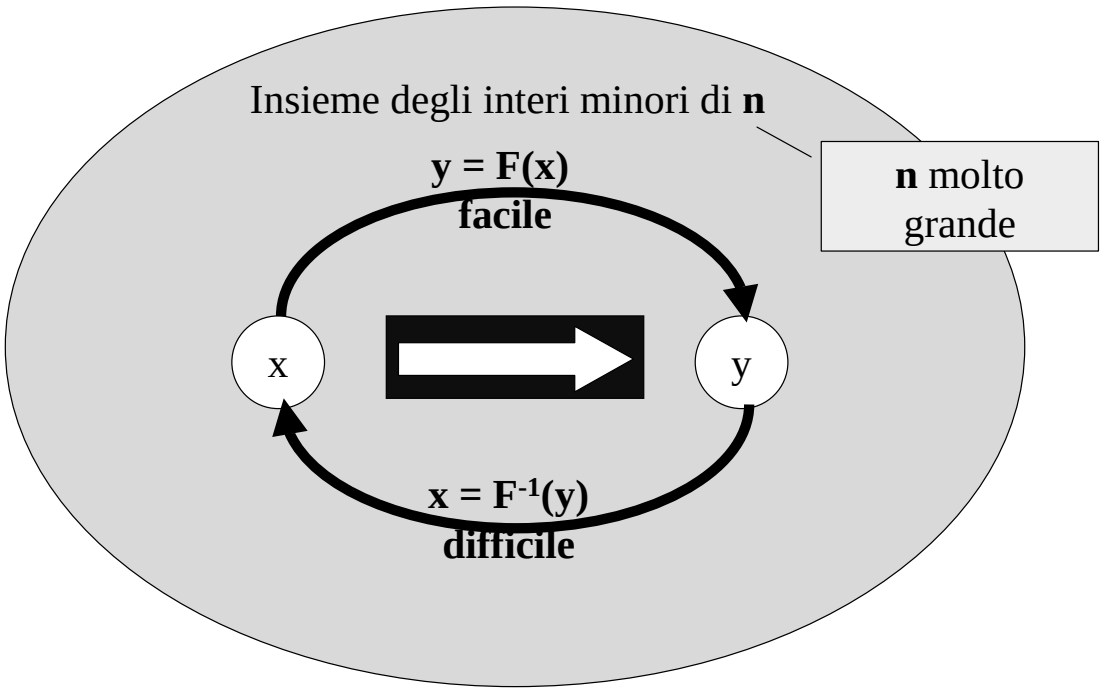


Il modello della Firma digitale con chiave privata



Crittografia asimmetrica e Teoria dei numeri

Messaggio → Stringa di simboli → Stringa di bit → Numero
Trasformazioni sui simboli→ Operazioni sui numeri



Tre importanti funzioni unidirezionali

Algoritmi polinomiali

Esponenziazione modulo un primo
Dati: p primo, **g** generatore, **x < p**
Trovare: $y = g^x \bmod p$

Esponenziazione modulo un composto
Dati: n = p.q, **p**,**q** primi,
MCD(e,Φ(n))=1, **x < n**
Trovare: $y = x^e \bmod n$

Prodotto di due numeri primi
Dati: p,**q** primi
Trovare: $n = p \times q$

Algoritmi subesponenziali

Logaritmo discreto
Dati: p primo, **g** generatore, **y < p**
Trovare: $x = \log_g y$

Radice e-esima
Dati: n composto, **y < n**
Trovare: $x = \sqrt[e]{y}$

Fattorizzazione
Dati: n prodotto di due primi p e q
Trovare: i fattori p e q

Assunzione: nessuno troverà mai algoritmi polinomiali

Teoria dei numeri (1)

- **b divisore di a:**
  b | **a** se a = m × b con m intero
- **p numero primo:**
  se i suoi divisori sono solo 1 e p
- **fattorizzazione di n**:
  $\mathbf{n = p_1^{e1} \times p_2^{e2} .. \times p_k^{ek}}$ con $\mathbf{p_1 \ldots p_k}$ **primi**, **ei** interi **≥0**
- **massimo comun divisore di a,b:**
  MCD(a,b) = k con k più grande divisore di a e di b
- **a, b coprimi (o relativamente primi):**
  MCD(a,b) = 1
- **a mod n:**
  resto della divisione di **a** ≥ 0 per **n** > 0
- **a congruo a b modulo n:**
  a ≡ b (mod n) se **a mod n = b mod n**

Teoria dei numeri (2)

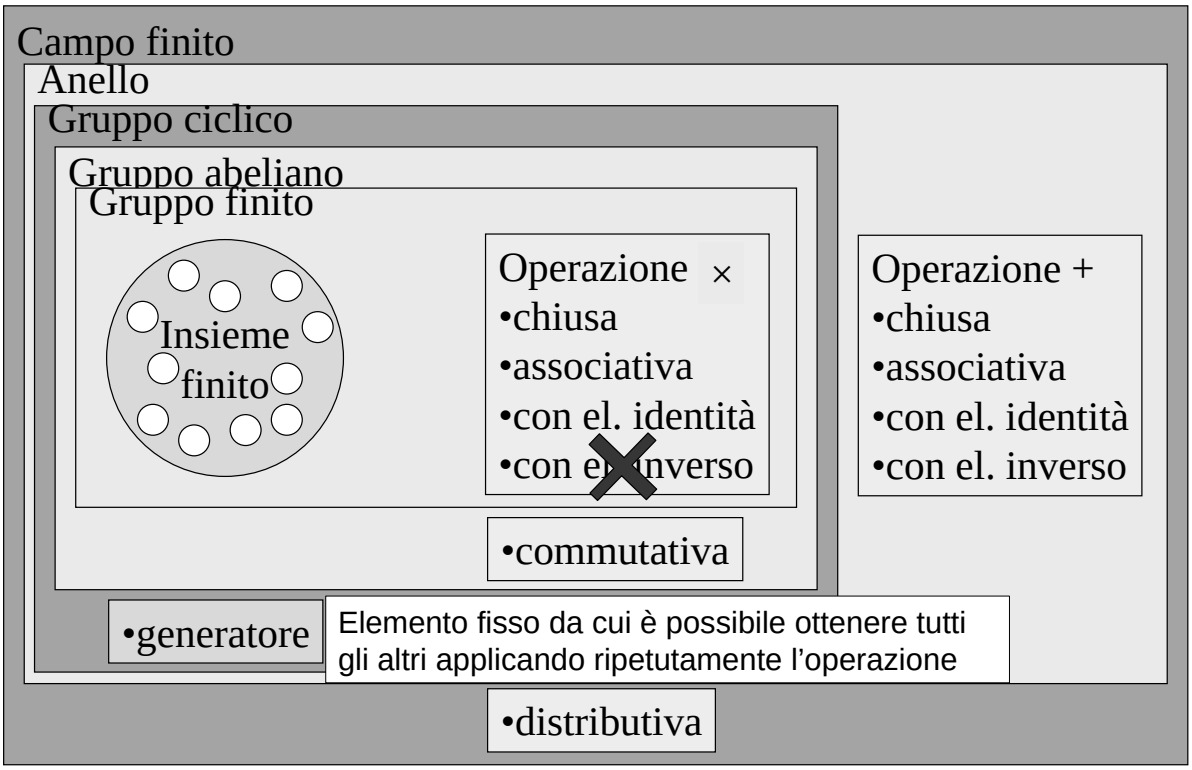
- Dato un intero positivo **n** si indica con
  Z_n = {0,1, …, n-1} l'insieme di tutti gli interi non negativi < **n**
  Z^*_n = { a $\in Z_n$ | MCD(a,n) = 1}insieme dei coprimi con **n**;
  Z^*_p = {1, …, p-1} quando **p** è primo

- **funzione di Eulero:** n° di interi < **n** e coprimi con n

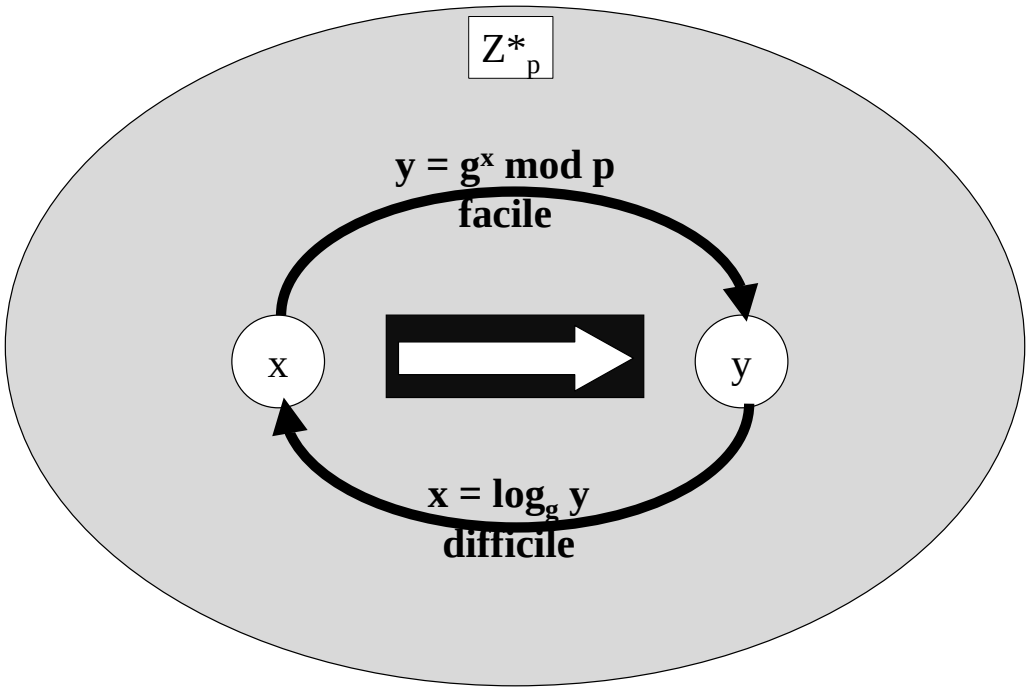
  $$\Phi(n) = n\times(1-1/p_1) \times \ldots \times(1-1/p_k)$$

  1. **Φ(n) = n-1** se **n** è primo
  2. **Φ(n) = (p-1)(q-1)** se **n** è il prodotto di due primi **p** e **q**

Algebra astratta



Esponenziazione e logaritmo discreto su GF(p)



GF(p): elementi, operazioni e proprietà

Campo finito →Campo di Galois: GF(p^n) con p **primo** e n intero

GF(p):
- **$Z^*_p = \{1, \ldots, p-1\}$**
- **Aritmetica modulare**

Addizione: **(a+b) mod p**
Moltiplicazione: **(a × b) mod p**

O(log p) operazioni su bit
O((log p)²) operazioni su bit

Proprietà dell'aritmetica modulare – Per ogni a,b,m si ha:

(a+b) mod m = ((a mod m) + (b mod m)) mod m
(a×b) mod m = ((a mod m) × (b mod m)) mod m

GF(11): elementi

p = 11

Z_{11}: {0,1,2,3,4,5,6,7,8,9,10}

Z^*_{11}: {1,2,3,4,5,6,7,8,9,10}

$\Phi(11) = p-1 = 10$

GF(11): addizione

chiusura

associativa

commutativa

elemento identità

inverso additivo

x \ y	0	1	2	3	4	5	6	7	8	9	10
0	0	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10	0
2	2	3	4	5	6	7	8	9	10	0	1
3	3	4	5	6	7	8	9	10	0	1	2
4	4	5	6	7	8	9	10	0	1	2	3
5	5	6	7	8	9	10	0	1	2	3	4
6	6	7	8	9	10	0	1	2	3	4	5
7	7	8	9	10	0	1	2	3	4	5	6
8	8	9	10	0	1	2	3	4	5	6	7
9	9	10	0	1	2	3	4	5	6	7	8
10	10	0	1	2	3	4	5	6	7	8	9

(x + y) mod 11

GF(11): moltiplicazione

chiusura

associativa

commutativa

elemento identità

reciproco

distributiva

		y									
x	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0											
1		1	2	3	4	5	6	7	8	9	10
2		2	4	6	8	10	1	3	5	7	9
3		3	6	9	1	4	7	10	2	5	8
4		4	8	1	5	9	2	6	10	3	7
5		5	10	4	9	3	8	2	7	1	6
6		6	1	7	2	8	3	9	4	10	5
7		7	3	10	6	2	9	5	1	8	4
8		8	5	2	10	7	4	1	9	6	3
9		9	7	5	3	1	10	8	6	4	2
10		10	9	8	7	6	5	4	3	2	1

$(x \times y) \bmod 11$

Esponenziazione modulare

$a = b^e \bmod m$

$a = (b \times b \times \ldots \times b) \bmod m$
e *volte*

$a = [(b \bmod m) \times (b \bmod m) \times \ldots \times (b \bmod m)] \bmod m$
e *volte*

$b \in Z_m$

Algoritmi più efficienti!

GF(11): esponenziazione

colonna 10= **p-1**
"sempre 1"

colonna 9= **p-2**
"b^{-1} mod p"

colonna 5= **(p-1)/2**
"+1, -1 mod p"

	e									
b	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	1	1	1	1	1	1	1	1	1	1
2	2	4	8	5	10	9	7	3	6	1
3	3	9	5	4	1	3	9	5	4	1
4	4	5	9	3	1	4	5	9	3	1
5	5	3	4	9	1	5	3	4	9	1
6	6	3	7	9	10	5	8	4	2	1
7	7	5	2	3	10	4	6	9	8	1
8	8	9	6	4	10	3	2	5	7	1
9	9	4	3	5	1	9	4	3	5	1
10	10	1	10	1	10	1	10	1	10	1

righe2,6,7,8
"permutazioni di Z"
generatori di Z

b^e mod 11

$b,e \in Z^*_{11}$

Piccolo teorema di Fermat

T4 (piccolo teorema di Fermat) : "per ogni primo p e per ogni x di Z^*_p si ha

$$x^{p-1} \bmod p = 1 \text{ , cioè } x^{p-1} \equiv 1 \pmod p$$

Conseguenze :

$$x^p \equiv x \pmod p$$

$$\text{per ogni } c \geq p,\ x^c \equiv x^{(c \bmod p)+1} \pmod p$$

Altre proprietà di GF(p)

Radici quadrate di 1:
$\mathbf{x^{(p-1)/2} \bmod p = +1}$ (se x **non è** una radice primitiva di p)
$\mathbf{= -1}$ (cioè **p-1**, se x **è** un generatore di p)

Reciproco (T5): "l'inverso moltiplicativo di x in GF(p) è
$\mathbf{x^{-1} = x^{p-2} \bmod p}$ (*infatti* $x \cdot x^{p-2} \bmod p = 1$)
N.B.il modo più semplice di calcolarlo è l'algoritmo esteso di Euclide

Generatori modulo p

T6: "per ogni primo **p**, esiste un **g < p**, detto generatore modulo p o radice primitiva di p, le cui potenze
$\mathbf{g^1 \bmod p , g^2 \bmod p , \ldots g^{p-1} \bmod p}$
forniscono tutti gli interi compresi tra **1** e **p-1**"

Generatori di GF(11): 2, 6, 7, 8

T7: "i generatori di **p** sono **Φ(p-1)**; per ogni generatore si ha
$\mathbf{g^{(p-1)/2} \bmod p = -1 = p-1}$".

$\Phi(10) = \Phi(2 \times 5) = 10 \times (1-1/2) \times (1-1/5) = (2-1) \times (5-1) = 4$

Th: **a** è un generatore di Z^*_n se e solo se $\mathbf{a^{\Phi(n)/q} \not\equiv 1}$ per ogni **q** divisore primo di Φ(n)

Generatori mod p

	i									
g	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	2	4	8	5	10	9	7	3	6	1
6	6	3	7	9	10	5	8	4	2	1
7	7	5	2	3	10	4	6	9	8	1
8	8	9	6	4	10	3	2	5	7	1

g^i mod 11

I generatori di p = 11 sono 2, 6, 7,8
In colonna i = 5 si ha sempre $g^5 = 10$, cioè –1
In colonna i = 2 non c'è 1(*per ogni $b \neq 1$ e $\neq p-1 \rightarrow b^2 \bmod p \neq 1$*)

Algoritmi di esponenziazione con modulo primo o composto

$y = b^e \bmod n$

Sia $\mathbf{t = \lceil \log_2 n \rceil}$

La rappresentazione in base 2 di e è:

$$\mathbf{e = e_{n-1}\, 2^{n-1} + e_{n-2}\, 2^{n-2} + \ldots\ldots\ldots + e_1\, 2^1 + e_0\, 2^0}$$

Di conseguenza si ha:

$$\mathbf{y = b \wedge e = (b \wedge (e_{n-1}\, 2^{n-1})) \times (b \wedge (e_{n-2}\, 2^{n-2})) \times .. \times ((b \wedge (e_1\, 2^1)) \times b \wedge e_0}$$

1: si calcola b, b^2, b^4 , b^8 ecc.

2: si moltiplicano tra loro i b^i per cui $e_i = 1$

3: si divide per n, trattenendo il resto

Caso peggiore: **2.t** moltiplicazioni (..ma su numeri sempre più grandi!)

Complessità

1 - Riducendo in modulo il risultato di ogni moltiplicazione si riesce ad operare sempre e solo su numeri di Z^*_p
2 – Non separando il calcolo dei $\mathbf{g^i}$ da quello dei $(\mathbf{g^i})^{bi}$ si ottengono tempi di esecuzione più brevi

Successive quadrature e moltiplicazioni

INPUT: $b \in Z^*_n$, $0 \le e < n$, $e = (e_{t-1} .. e_0)_2$

OUTPUT: $b^e \bmod n$

1. Set $y \leftarrow 1$. If $e = 0$ then return (b)
2. Set $A \leftarrow b$
3. If $e_0 = 1$ then set $y \leftarrow b$
4. For i from 1 to t-1 do the following:
   4.1 Set $A \leftarrow A^2 \bmod n$
   4.2 If $e_i = 1$ then set $y \leftarrow A\ b \bmod n$.
5. Return (y)

n = 11, b = 7, e = 5 = $(101)_2$
1. **y = 1**
2. **A = 7**
3. **y = 7**
4 **i = 1**
   4.1 **$A = 7 \times 7 \bmod 11 = 5$**
   i = 2
   4.1 **$A = 5 \times 5 \bmod 11 = 3$**
   4.2 **$y = 3 \times 7 \bmod 11 = 10$**
5. **y = 10**

INPUT: $a \in Z^*_n$, $0 \le k < n$, $k = (k_t k_{t-1} .. k_0)_2$

OUTPUT: $a^k \bmod n$

1. $A \leftarrow 1$
2. For i from t down to 1 do the following:
   2.1 $A \leftarrow A^2 \bmod n$
   2.2 If $k_i = 1$ then $A \leftarrow A \times b \bmod n$.
3. Return (A)

1. **A = 1**
2. **i = 2**
   2.1 **$A = 1 \times 1 \bmod 11 = 1$**
   2.2 **$A = 1 \times 7 \bmod 11 = 7$**
   i = 1
   2.1 **$A = 7 \times 7 \bmod 11 = 5$**
   i = 0
   2.1 **$A = 5 \times 5 \bmod 11 = 3$**
   2.2 **$A = 3 \times 7 \bmod 11 = 10$**
5. **$7^5 \bmod 11 = 10$**

Complessità

$O(\lceil \log_2 p \rceil)$ moltiplicazioni

Moltiplicazione e divisione: $O((\log p)^2)$ operazioni su bit

Esponenziazione: $O((\log p)^3)$ operazioni binarie.

Robustezza e tempo d'esecuzione
Modulo: da 1024 bit a 2048 bit
Tempo: da 1 a 8

Logaritmo discreto

T8: " Dati un primo **p**, un generatore **g** ed un qualunque intero **y** $\in Z^*_p$, esiste un unico intero **x** , $1 \leq x \leq p-1$, tale che $g^x \bmod p = y$.

	i									
g	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	2	4	8	5	10	9	7	3	6	1
6	6	3	7	9	10	5	8	4	2	1
7	7	5	2	3	10	4	6	9	8	1
8	8	9	6	4	10	3	2	5	7	1

$g = 2 \rightarrow x = 8$

$g = 6 \rightarrow x = 2$

$g = 7 \rightarrow x = 4$

$g = 8 \rightarrow x = 6$

$y = 3$

Algoritmi per il logaritmo discreto

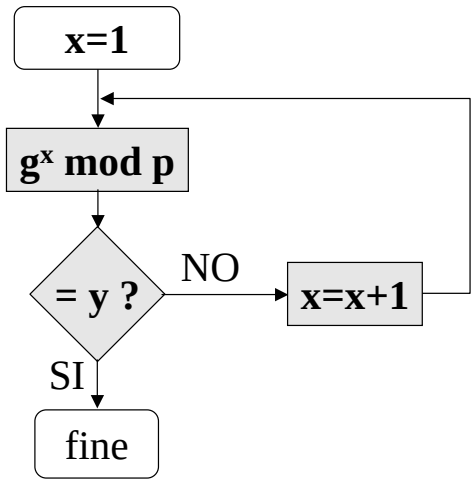
Ricerca esauriente
polinomiale nel valore del dato **y** $\leq p-1$,
esponenziale nel numero di bit $\mathbf{n = \lceil \log_2 p \rceil}$:
$\mathbf{T = O(\exp(n))}$.

Baby-step Giant-step
$\mathbf{T = O(\exp(n^{1/2}))}$

Index Calculus(**sub-esponenziale**)
$\mathbf{T = O(\exp((n)^{1/2} (\ln(n)^{1/2}))}$

Number field sieve (**sub-esponenziale**)
$\mathbf{T = O(\exp((n)^{1/3} (\ln(n)^{2/3}))}$

Ricerca esauriente



$\mathbf{n = \lceil \log_2 p \rceil}$
$\mathbf{T = O(\exp(n))}$.

Passi del bambino, passi del gigante

$g^x \equiv y \pmod p$

$g^{qm+r} \equiv y$

$g^r \equiv y.g^{-qm}$

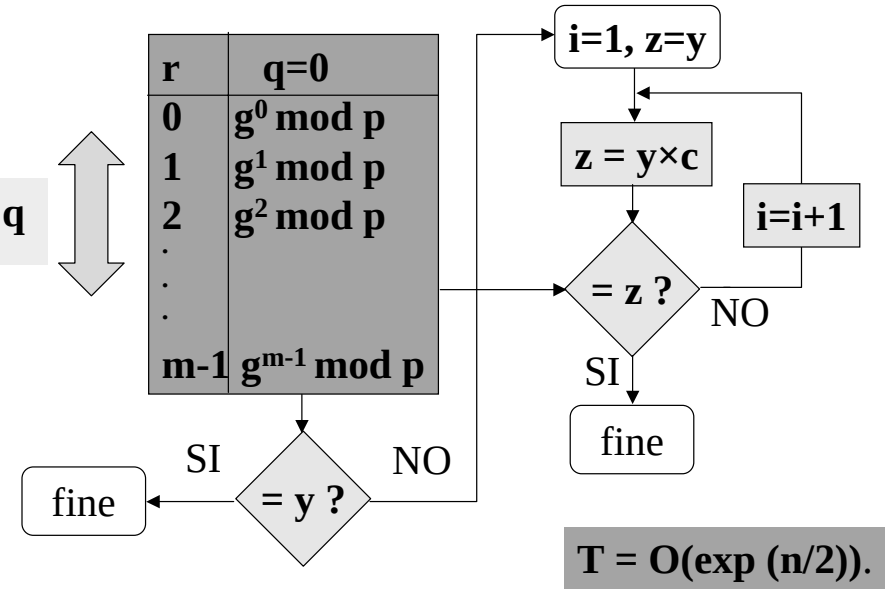
$g^r \equiv y.((g^{-m})^q$

$g^r \equiv y.c^q$

$m = \lceil sqrt(p) \rceil$

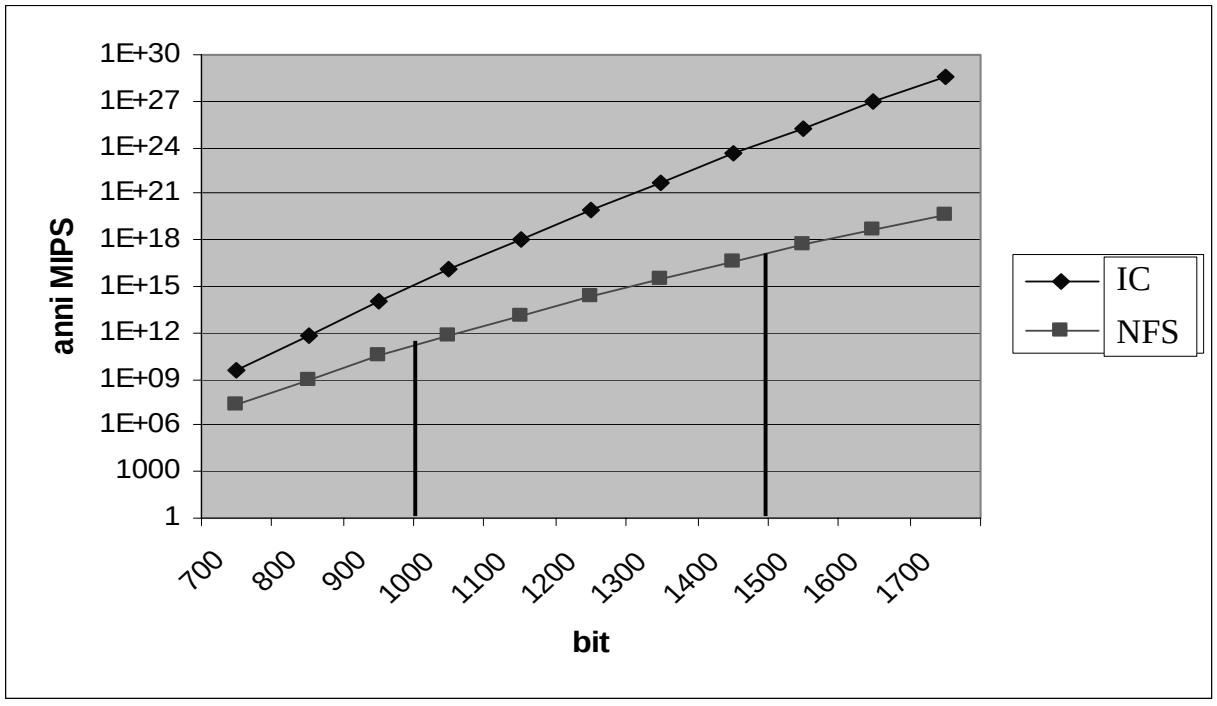
$x = q.m + r$

r	q=0
0	g^0 mod p
1	g^1 mod p
2	g^2 mod p
.	
.	
.	
m-1	g^{m-1} mod p



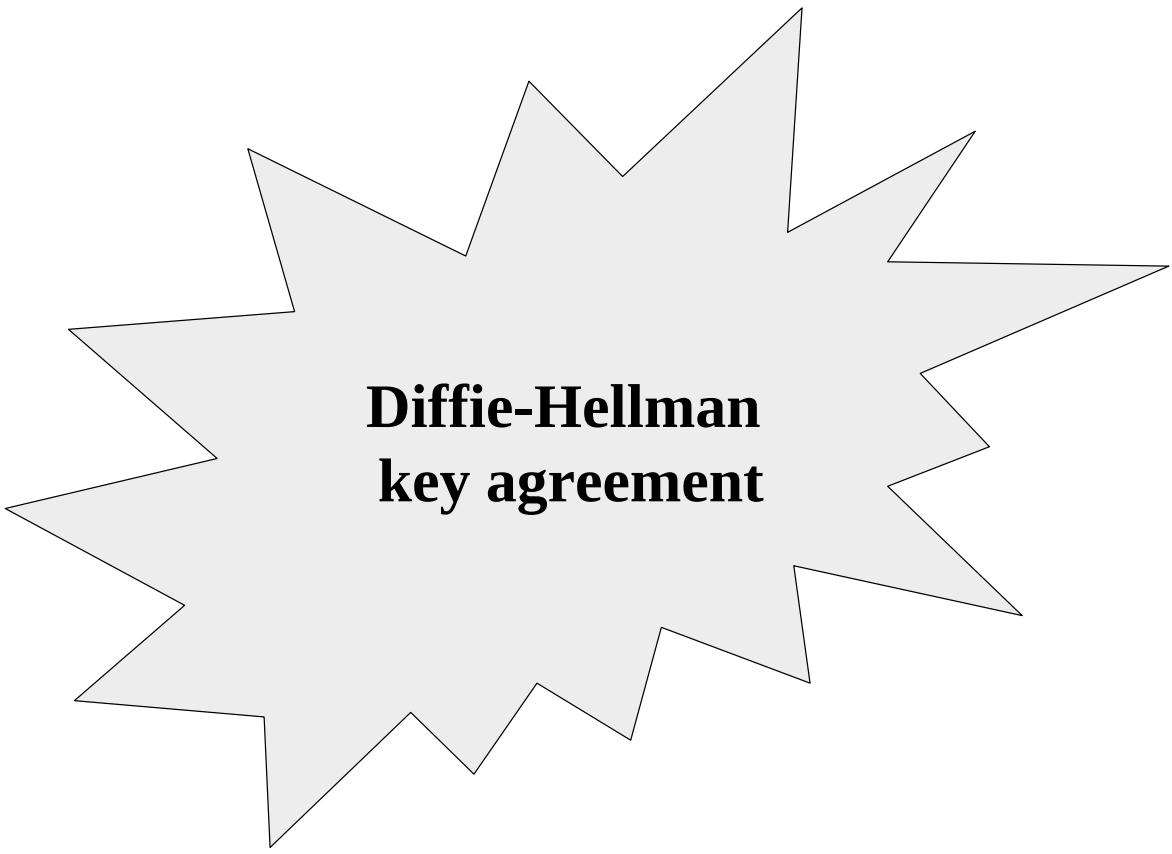
T = O(exp (n/2)).

Algoritmi subesponenziali

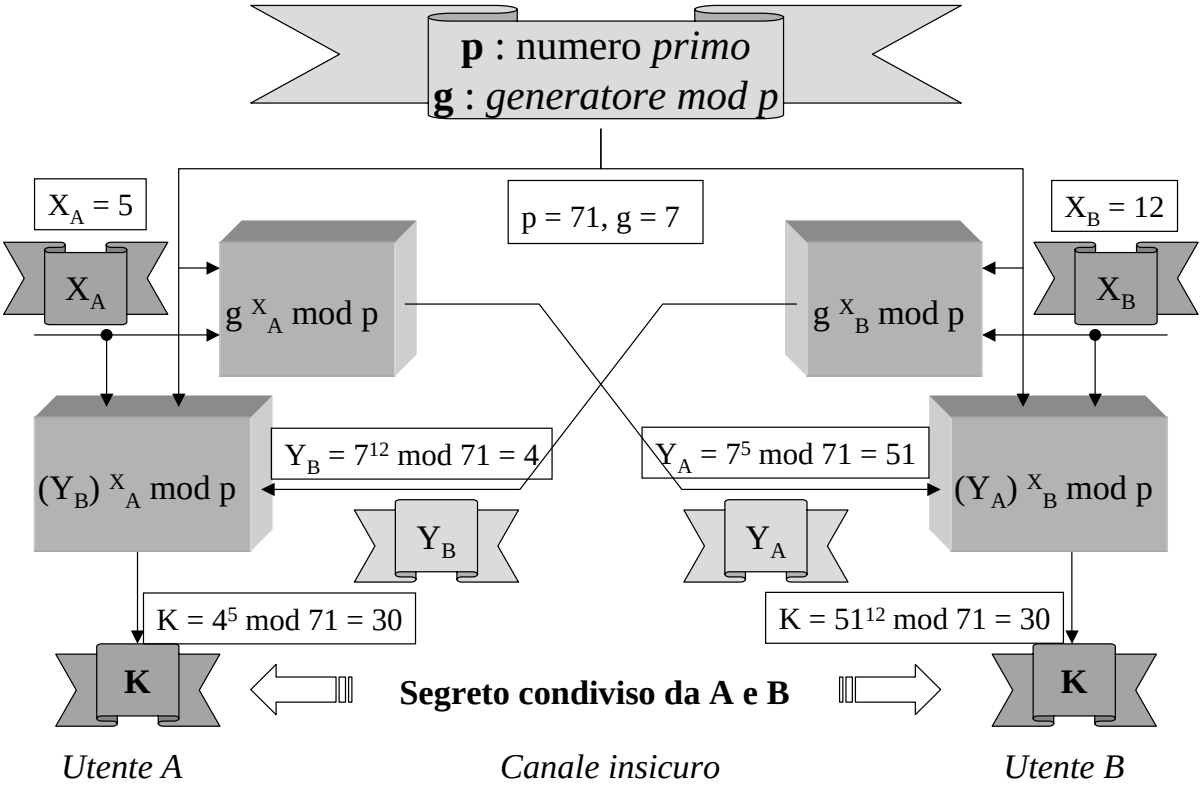


Meccanismi con sicurezza basata sul problema del logaritmo discreto

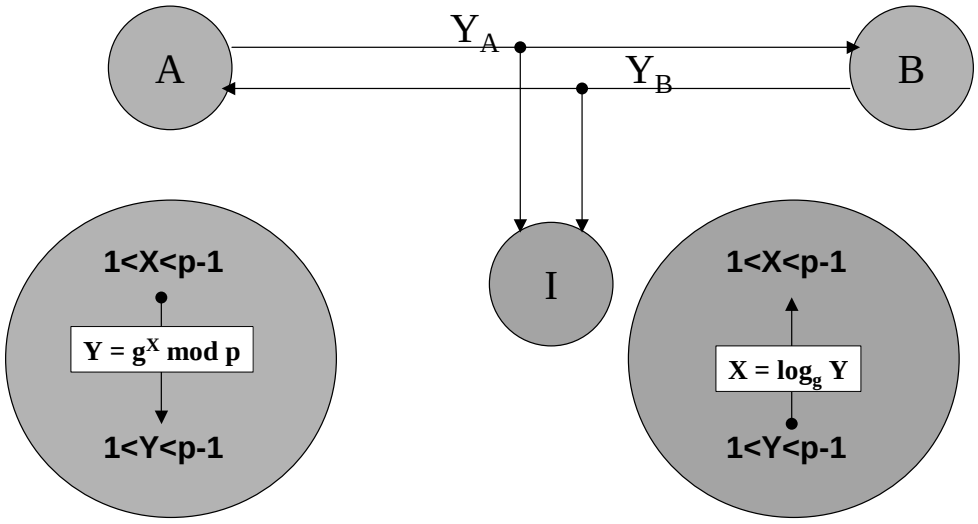
- Scambio di Diffie-Hellman
- Cifrario di ElGamal
- Firma di ElGamal
- Digital Signature Standard



Scambio DH: condivisione di un segreto



Sicurezza dello scambio DH



150 cifre
10^6 op/sec
$0{,}5.10^{-3}$ s

150 cifre
10^{12} op/sec
10^{56} anni

Il Cifrario di ElGamal (1985)

Opera su un gruppo moltiplicativo **ciclico** (caso più semplice: Z^*_p)

E' **probabilistico**

Ogni utente U ha

una chiave privata **SU = <u>** con $1 \leq u \leq p-2$ scelto a caso

una chiave pubblica **PU = <g^u mod p>** con **<p,g>** noti e comuni

• **Cifratura** - Chi deve mandare a U un messaggio $m \in Z_p$
dopo aver scelto a caso r, con $1 \leq r \leq p-2$, calcola:

$R = g^r \bmod p$

$S = [m \times (PU)^r] \bmod p$

$\mathbf{E_{PU}(m) = c = R||S}$

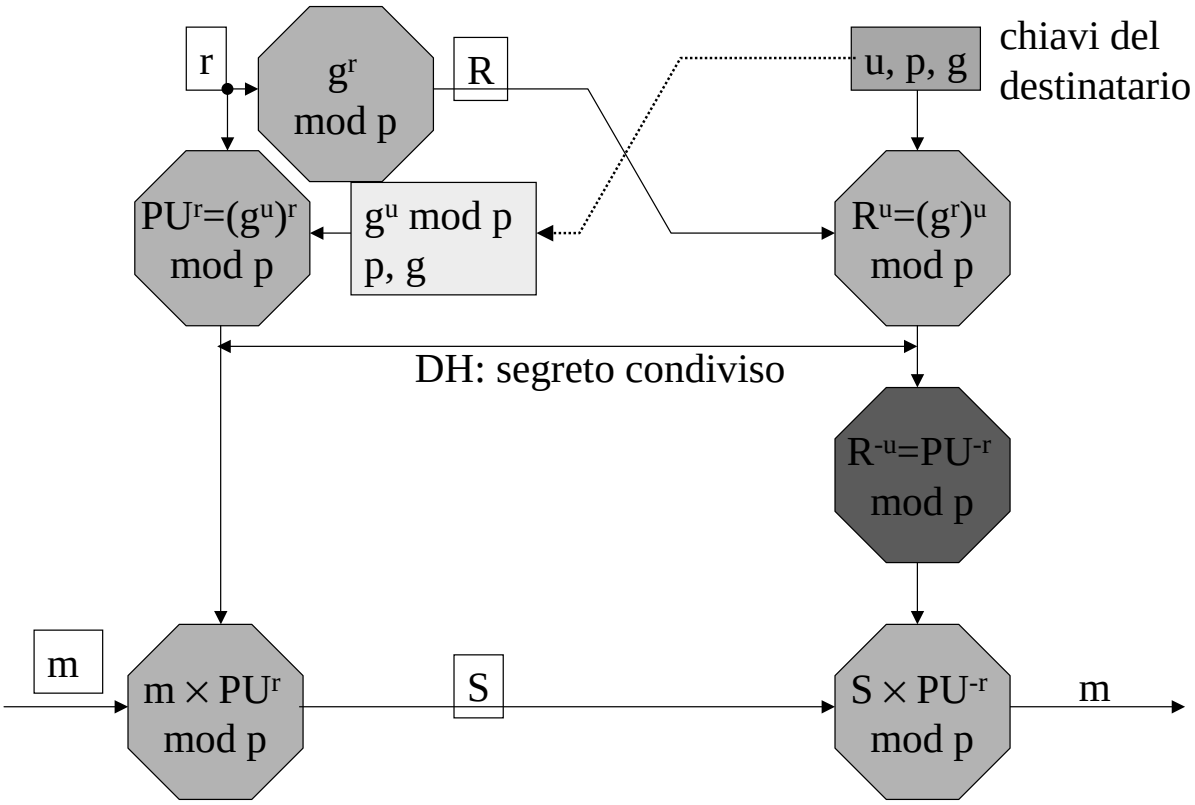
Il testo in chiaro viene espanso di un fattore due

• **Decifrazione** - U rimette in chiaro **c** calcolando:

PU^{-r} (N.B. è l'unico a poterlo fare avendo SU)

$\mathbf{D_{SU}(c) = PU^{-r} \times S \bmod p = m}$

Il Cifrario di ElGamal



Giustificazione

Decifrazione senza conoscere SU = u:

- Si considera R
- **Si estrae r = log R**
- Si calcola PU^{-r} mod p = PU^{p-1-r} mod p
- Si calcola $PU^{-r} \times S$ mod p

Trapdoor per chi conosce SU=u:

- Si considera R
- **Si calcola PU^{-r} mod p = $g^{-u.r}$ mod p**
  = $(g^r)^{-u}$ mod p
  = R^{-u} mod p
  = R^{p-1-u} mod p
- Si calcola $PU^{-r} \times S$ mod p