Cifrario RSA

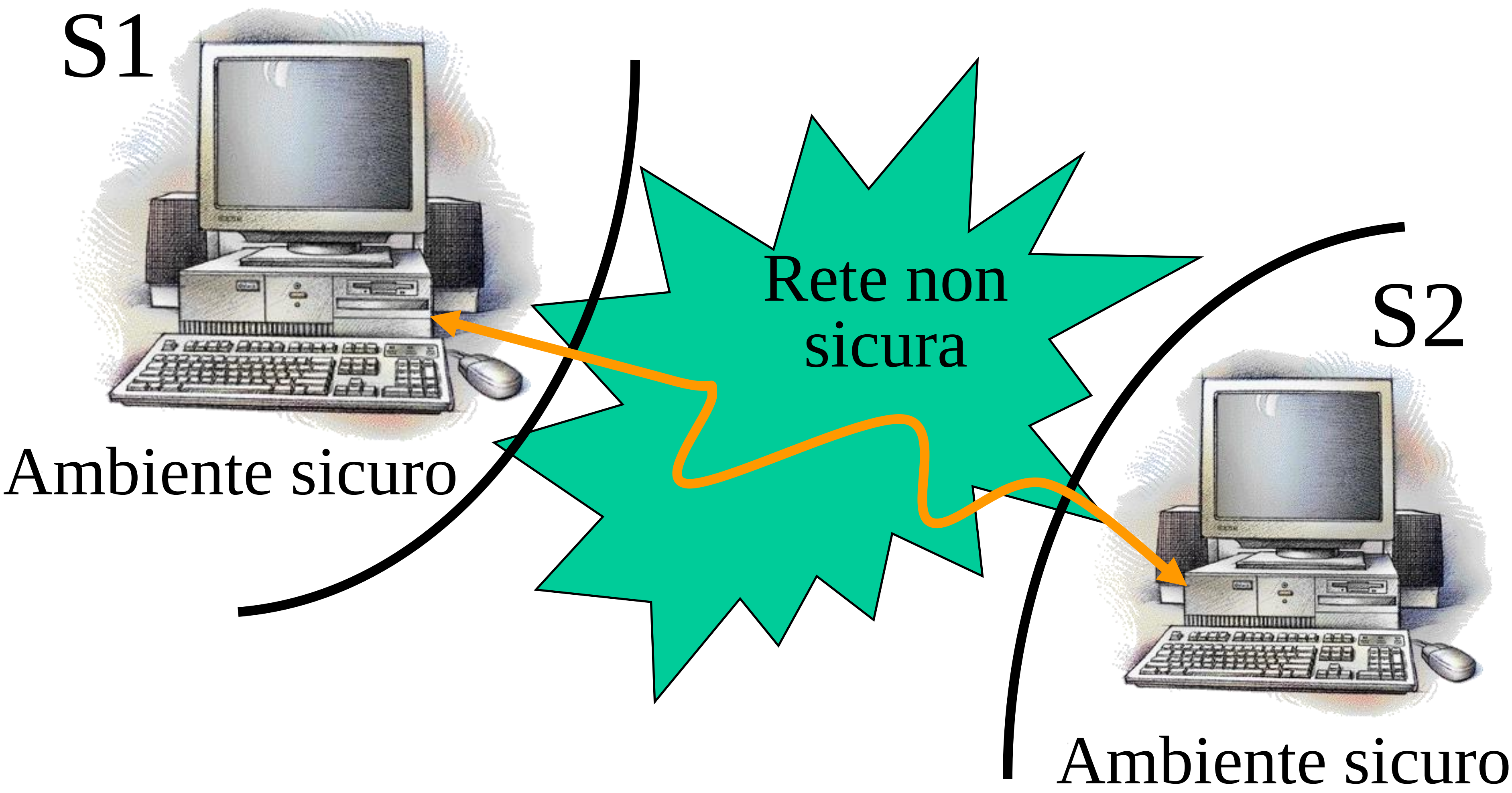
Prof. Francesco Bergadano

Dipartimento di Informatica
Università di Torino

Cifrario RSA

Rivest, Shamir, Adelman 1977

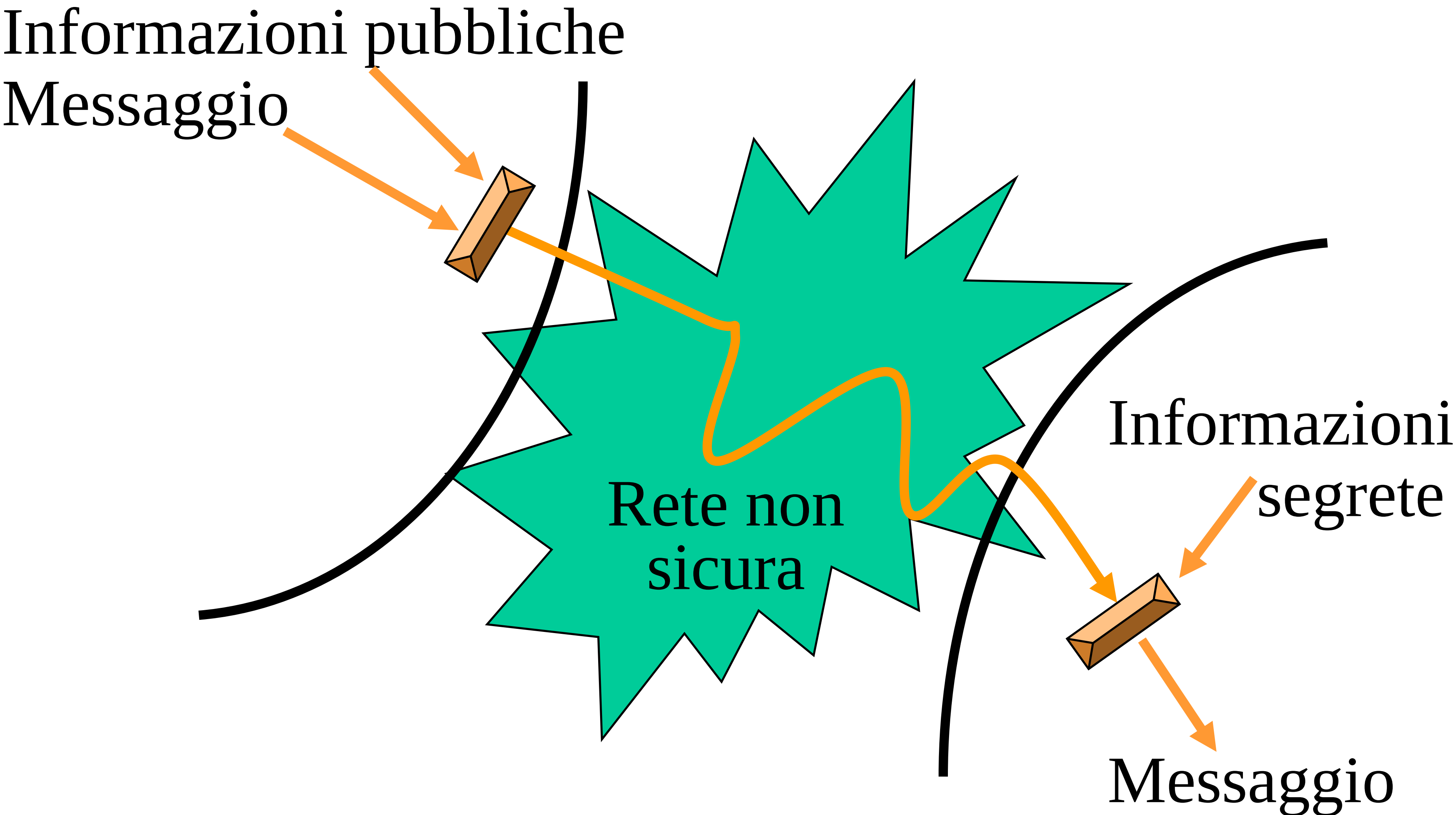
primo cifrario a chiave pubblica noto alla comunità scientifica internazionale



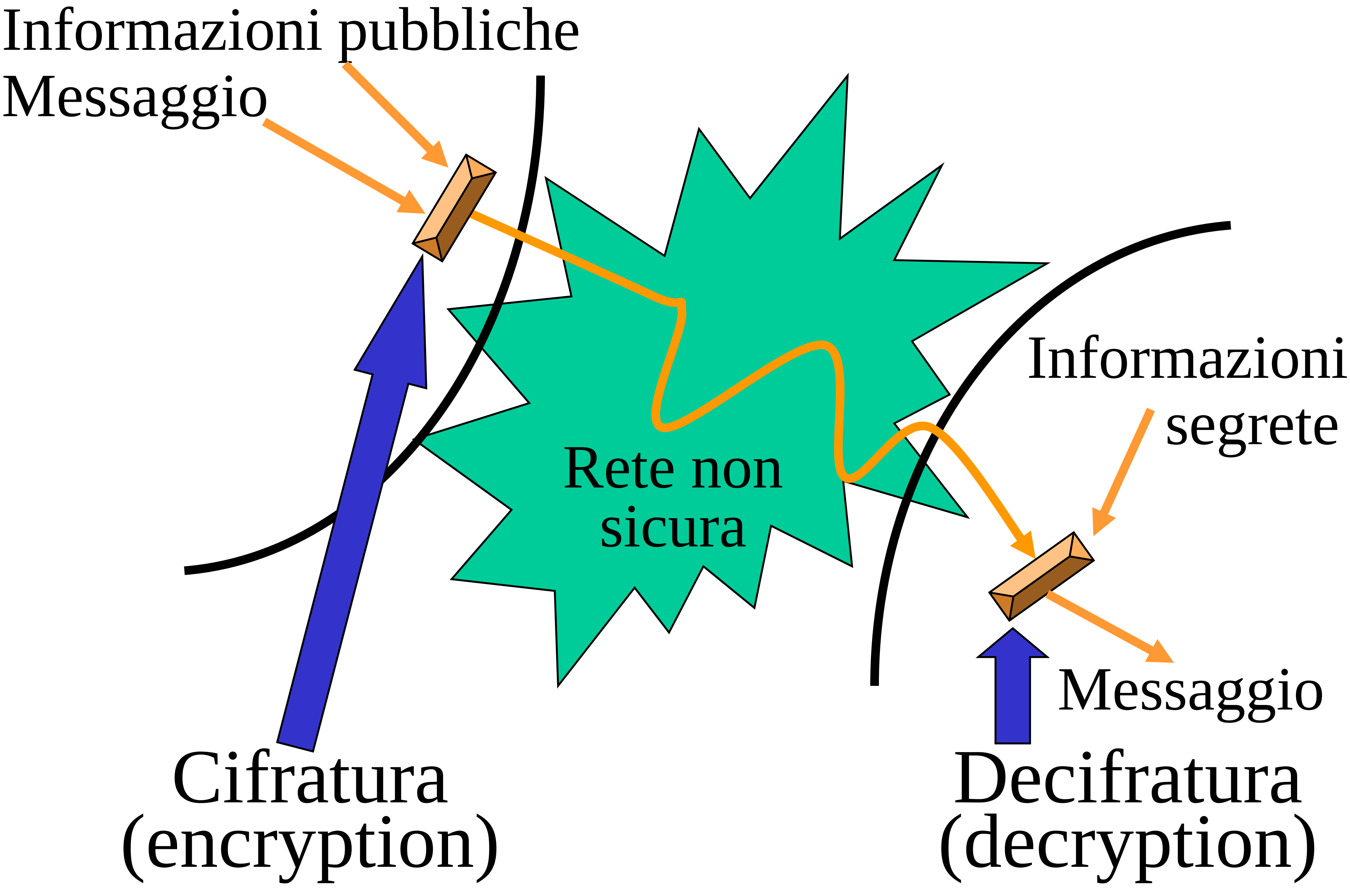
Cifrari Asimmetrici

Caratteristiche di RSA in quanto cifrario asimmetrico

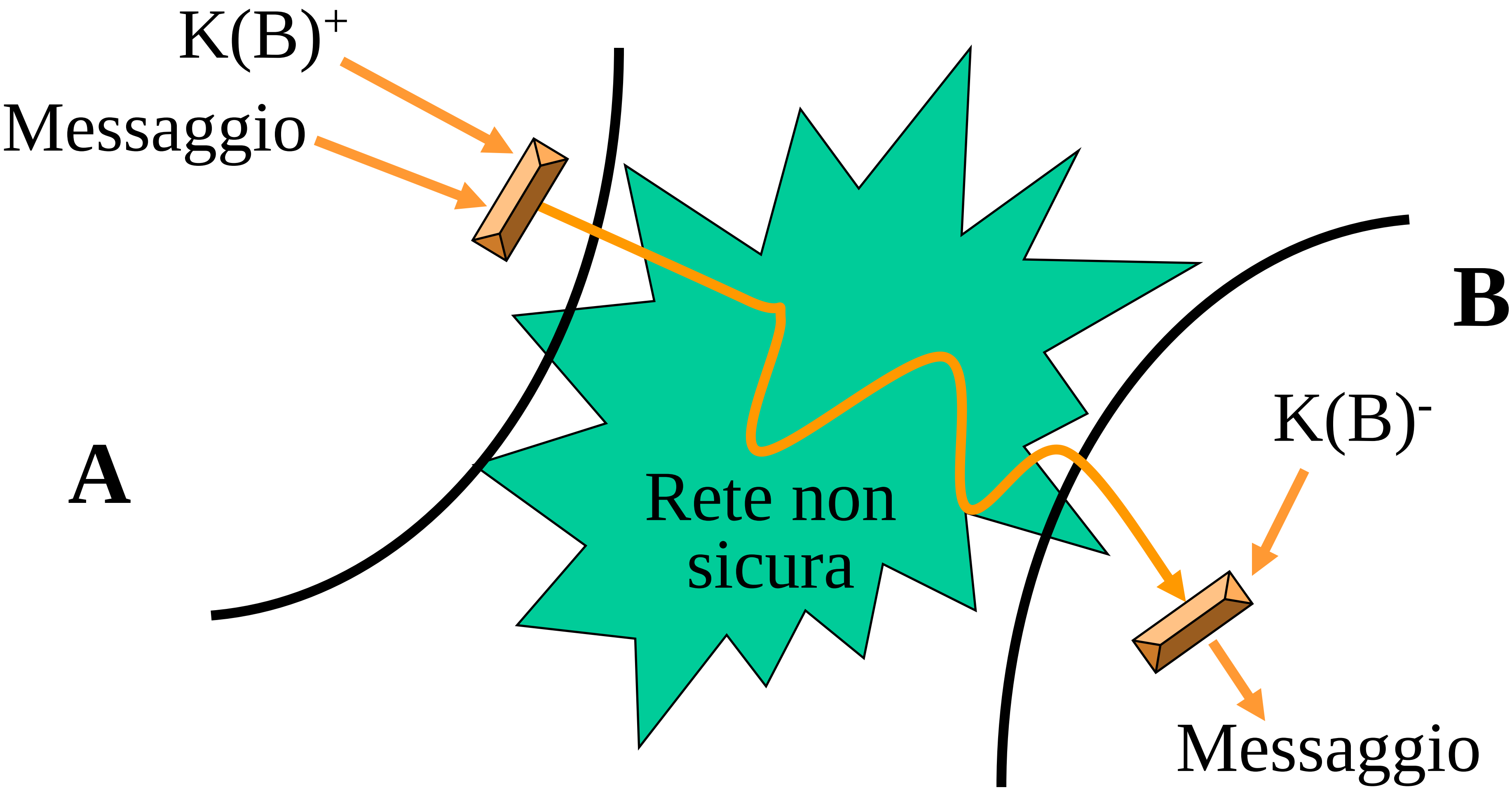
- Mittente e ricevente non condividono chiavi
- Per cifrare e decifrare si usano chiavi diverse
- Cifratura e decifratura sono relativamente inefficienti
- E’ difficile o praticamente impossibile decifrare senza conoscere la chiave, perché questo richiede eccessive risorse computazionali



Cifratura asimmetrica



Informazioni pubbliche
Messaggio
Rete non sicura
Informazioni segrete
Messaggio
Cifratura (encryption)
Decifratura (decryption)



Cifratura asimmetrica

Caratteristiche specifiche di RSA

- Basato sulla difficoltà di calcolare la fattorizzazione di numeri ottenuti moltiplicando numeri primi molto grandi
- Testo da cifrare (plaintext) e testo cifrato (ciphertext) sono numeri minori di un determinato numero **n**, che viene detto **modulo**
- Cifrario asimmetrico più usato

Schema generale di RSA

- Generazione delle chiavi per ogni utente
- Algoritmo per cifrare
- Algoritmo per decifrare

Generazione delle chiavi

- Scegli p e q primi
- calcola il modulo $n = pq$
- scegli $e < (p-1)(q-1)$ tale che $\gcd(e,(p-1)(q-1))=1$
- calcola $d = e^{-1} \bmod (p-1)(q-1)$
- $K^+ = \langle e,n \rangle$, $K^- = \langle d,n \rangle$

Per cifrare $m < n$

$$c = m^e \bmod n$$

Per decifrare $m < n$

$$m = c^d \bmod n$$

Correttezza di RSA

cifrando m otteniamo c, e vogliamo che decifrando c si ottenga lo stesso messaggio m di partenza:

se $C(m) = c$, allora $D(c) = m$

ovvero

se $c = m^e \bmod n$, allora $m = c^d \bmod n$

ovvero

$m = (m^e \bmod n)^d \bmod n$

Correttezza di RSA

Occorre quindi dimostrare che

$$m = (m^e \bmod n)^d \bmod n$$

ovvero

$$m \equiv m^{ed} \pmod{n}$$

sapendo che

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

Correttezza di RSA

Per prima cosa mostriamo che $(p-1)(q-1) = \Phi(n)$,

dove

p e q sono primi, $n=pq$ e

$\Phi(n)$ è il numero di interi minori di n che sono relativamente primi con n

$\Phi(n) = (p-1)(q-1)$

$\Phi(n) = |\{$*numeri da 1 a n-1 che* ***non sono*** *multipli di p o di q*$\}| =$

n-1 - $|\{$*numeri da 1 a n-1 che* ***sono*** *multipli di p o di q*$\}| =$

$= n-1-|\{p, 2p, ..., (q-1)p,$

$q, 2q, ..., (p-1)q\}| =$

$= n-1-(q-1+p-1) = n-q-p+1 =$

$= pq-p-q+1 = (p-1)(q-1)$

Risultato utile per RSA

Teorema (Eulero): se n,m relativamente primi,

$$m^{\Phi(n)} \bmod n = 1$$

$$(\forall k)\ m^{k\,\Phi(n)} \bmod n = (m^{\Phi(n)})^k \bmod n = 1$$

ovvero

se n,m relativamente primi e m < n,

$$(\forall k)\ m^{k\,\Phi(n)+1} \bmod n = m$$

e, per n=pq con p e q primi,

$$(\forall k)\ m^{k\Phi(n)+1} \bmod n = m^{k(p-1)(q-1)+1} \bmod n = m$$

Correttezza di RSA

$a = 1 \bmod b$ allora $a = kb+1$ per qualche k,

es. $a=16$, $b=3$ -> $k=5$

Correttezza di RSA

se m,n relativamente primi, per n=pq con p e q primi, in base al teorema di Eulero,

$$(\forall k)\ m^{k\Phi(n)+1} \bmod n = m^{k(p-1)(q-1)+1} \bmod n = m$$

e, siccome ed = 1 mod (p-1)(q-1), e pertanto ed = k'(p-1)(q-1)+1 per qualche k', abbiamo quindi, per k=k'

$$c^{d} \bmod n = m^{ed} \bmod n = m^{k'(p-1)(q-1)+1} \bmod n = m$$

Correttezza di RSA

Se il messaggio m, tradotto in un numero minore di n, ed n stesso sono relativamente primi, allora cifrando e decifrando si riottiene il messaggio originale m.

e se m non è relativamente primo con n? (ovvero se m=jp oppure m=iq)?

e se m=jp oppure m=iq?

Consideriamo il caso m= jp - il caso m=iq è analogo. Allora $j<q$, altrimenti $m = qp = n$, mentre $m<n$, e quindi q,m sono relativamente primi, quindi, per il teorema di Eulero:

$m^{\Phi(q)} \equiv 1 \bmod q$, $(m^{\Phi(q)})^{\Phi(p)} \equiv 1 \bmod q$, $m^{\Phi(n)} \equiv 1 \bmod q$

$(\forall i)\ [m^{\Phi(n)}]^i \equiv 1 \bmod q$, quindi $(\forall i)(\exists k)\ m^{i\Phi(n)} = 1 + kq$

Moltiplicando per m=jp:

$(\forall i)(\exists k)\ m^{i\Phi(n)+1} = m + kjpq = m + kjn$, ovvero

$(\forall i)\ m^{i\Phi(n)+1} = m^{i(p-1)(q-1)+1} \equiv m \pmod n$

Ma sappiamo che $(\exists k')\ ed = k'(p-1)(q-1)+1$, quindi

per $i=k'$, $m^{k'(p-1)(q-1)+1} \equiv m^{ed} \equiv m \pmod n$

Correttezza di RSA

Se il messaggio m può essere espresso come numero minore di n, allora cifrando e decifrando si riottiene il messaggio originale m.

Sicurezza di RSA

Si ritiene computazionalmente difficile il problema di decifrare un messaggio cifrato con RSA senza conoscere la corrispondente chiave d

Si ritiene computazionalmente difficile calcolare d a partire da e ed n senza conoscere p e q

Si ritiene computazionalmente difficile calcolare p e q a partire da n, per n grande

Efficienza di RSA

- Gli algoritmi per cifrare e decifrare, conoscendo le corrispondenti chiavi, hanno complessità logaritmica rispetto ad n, ma sono relativamente inefficienti se confrontati con i cifrari simmetrici
- L'algoritmo per generare le chiavi (Euclide generalizzato) è relativamente inefficiente e può causare problemi per n grande o con hardware limitato (es. smartcard).

Per realizzare RSA, occorre

- Algoritmo efficiente per calcolare $a^b \bmod q$ (per cifrare e decifrare)

- Algoritmo efficiente per generare p e q primi (per generazione chiavi)

- Algoritmo efficiente per calcolare l'inverso moltiplicativo di e

Per realizzare RSA, occorre

Vedi discussione per Diffie-Hellman

- ~~Algoritmo efficiente per calcolare a^b mod q (per cifrare e decifrare)~~
- Algoritmo efficiente per generare p e q primi (per generazione chiavi)
- Algoritmo efficiente per calcolare l'inverso moltiplicativo di e

Per realizzare RSA, occorre

- Algoritmo efficiente per calcolare a^bmod q (per cifrare e decifrare)
- Algoritmo efficiente per generare p e q primi (per generazione chiavi)

Vedi discussione per Diffie-Hellman

- Algoritmo efficiente per calcolare l'inverso moltiplicativo di e

Per realizzare RSA, occorre

- Algoritmo efficiente per calcolare $a^b \bmod q$ (per cifrare e decifrare)
- Algoritmo efficiente per generare p e q primi (per generazione chiavi)
- Algoritmo efficiente per calcolare l’inverso moltiplicativo di e

Generazione delle chiavi
Algoritmo per calcolare l’inverso moltiplicativo di e

Basato sull’algoritmo di Euclide per calcolare il Massimo Comun Divisore (MCD)

Algoritmo di Euclide per calcolare il Massimo Comun Divisore

```
// calcolo di mcd(a,b) (con a≥b, altrimenti invertiamo)
Z = a; W = b; // mcd(W,Z) = mcd(Z,W) = mcd(a,b)
while (TRUE) {
    if (W = = 0) return Z; // Z = mcd(a,b)
    R = Z mod W;
    Z = W; W = R; }
```

Algoritmo di Euclide per calcolare il Massimo Comun Divisore

Esempio: mcd(99,81)

i	a	b	Z	W	R	
0	99	81	99	81	18	(99=1*81+18=81+18)

Algoritmo di Euclide per calcolare il Massimo Comun Divisore

Esempio: mcd(99,81)

i	a	b	Z	W	R	
0	99	81	99	81	18	(99=1*81+18=81+18)
1	99	81	81	18	9	(81=4*18+9=72+9)

Algoritmo di Euclide per calcolare il Massimo Comun Divisore

Esempio: mcd(99,81)

i	a	b	Z	W	R	
0	99	81	99	81	18	(99=1*81+18=81+18)
1	99	81	81	18	9	(81=4*18+9=72+9)
2	99	81	18	9	0	(18=2*9+0=18+0)

Algoritmo di Euclide per calcolare il Massimo Comun Divisore

Esempio: mcd(99,81)

i	a	b	Z	W	R	
0	99	81	99	81	18	(99=1*81+18=81+18)
1	99	81	81	18	9	(81=4*18+9=72+9)
2	99	81	18	9	0	(18=2*9+0=18+0)
3	99	81	9	0		MCD(99,81)=9

Algoritmo di Euclide per calcolare il Massimo Comun Divisore

Esempio: mcd(99,81)

i	$a=r_0$ $\quad b=r_1$	
0	$a=r_0=q_1*b+r_2=q_1*r_1+r_2$	(99=1*81+18=81+18)
1	$b=r_1=q_2*r_2+r_3$	(81=4*18+9=72+9)
2	$r_2=q_3*r_3+r_4=q_3*r_3+0$	(18=2*9+0=18+0)
3	$r_3=MCD(a,b)=MCD(r_0,r_1)$	MCD(99,81)=9

Algoritmo di Euclide per calcolare il Massimo Comun Divisore

Esempio: mcd(99,81)

q_1 r_2

$a=r_0$ $b=r_1$

i	$a=r_0$ $b=r_1$	
0	$a=r_0=q_1*b+r_2=q_1*r_1+r_2$	(99=1*81+18=81+18)
1	$b=r_1=q_2*r_2+r_3$	(81=4*18+9=72+9)
2	$r_2=q_3*r_3+r_4=q_3*r_3+0$	(18=2*9+0=18+0)
3	$r_3=MCD(a,b)=MCD(r_0,r_1)$	MCD(99,81)=9

Algoritmo di Euclide per calcolare il Massimo Comun Divisore

Esempio: mcd(99,81)

q_2 r_3

$b=r_1$ r_2

i	$a=r_0$ $\quad b=r_1$	
0	$a=r_0=q_1*b+r_2=q_1*r_1+r_2$	(99=1*81+18=81+18)
1	$b=r_1=q_2*r_2+r_3$	(81=4*18+9=72+9)
2	$r_2=q_3*r_3+r_4=q_3*r_3+0$	(18=2*9+0=18+0)
3	$r_3=MCD(a,b)=MCD(r_0,r_1)$	MCD(99,81)=9

In generale … $r_k=q_{k+1}*r_{k+1}+r_{k+2}$

Correttezza di MCD

0 $a=r_0=q_1*b+r_2=q_1*r_1+r_2$

1 $b=r_1=q_2*r_2+r_3$

2 $r_2=q_3*r_3+r_4=q_3*r_3+0$

3 $r_3=MCD(a,b)=MCD(r_0,r_1)$ - in generale, $r_k=q_{k+1}*r_{k+1}+r_{k+2}$

se d|a e d|b, allora d|$\mathbf{r_i}$, per ogni i, infatti:

$d|a=r_0$ e $d|b=r_1$, inoltre $r_k=q_{k+1}*r_{k+1}+r_{k+2}$

Per induzione, supponiamo che $d|r_k$ e $d|r_{k+1}$

allora, $r_k=dm$ e $r_{k+1}=dn$, per qualche m,n

Pertanto $r_{k+2}=r_k-q_{k+1}*r_{k+1}=dm-q_{k+1}*dn=d*(m-q_{k+1}*n)$, ovvero $d|r_{k+2}$

Correttezza di MCD

0 $a=r_0=q_1*b+r_2=q_1*r_1+r_2$

1 $b=r_1=q_2*r_2+r_3$

2 $r_2=q_3*r_3+r_4=q_3*r_3+0$

ultimo step =k=2,
$r_4=r_{k+2}=0$,
$r_3=r_{k+1}=MCD(a,b)$

3 $r_3=MCD(a,b)=MCD(r_0,r_1)$ - in generale, $r_k=q_{k+1}*r_{k+1}+r_{k+2}$

$\mathbf{r_{k+1}|r_i}$, per ogni i, dove k è l'ultimo step (ovvero $r_{k+2}=0$), infatti:

$r_k=q_{k+1}*r_{k+1}+0$, quindi $r_{k+1}|r_k$. Inoltre, $r_{k+1}|r_{k+1}$.

Per induzione, supponiamo che $r_{k+1}|r_j$ e $r_{k+1}|r_{j+1}$

(ovvero $mr_{k+1}=r_j$ e $nr_{k+1}=r_{j+1}$, per qualche m,n)

$r_{j-1}=q_j*r_j+r_{j+1}=q_j*mr_{k+1}+nr_{k+1}=(q_j*m+n)r_{k+1}$, ovvero $r_{k+1}|r_{j-1}$.

Correttezza di MCD

0 $a=r_0=q_1*b+r_2=q_1*r_1+r_2$

1 $b=r_1=q_2*r_2+r_3$

2 $r_2=q_3*r_3+r_4=q_3*r_3+0$

3 $r_3=MCD(a,b)=MCD(r_0,r_1)$ - in generale, $r_k=q_{k+1}*r_{k+1}+r_{k+2}$

ultimo step =k=2,
$r_4=r_{k+2}=0$,
$r_3=r_{k+1}=MCD(a,b)$

Uniamo i risultati precedenti:

$\mathbf{r_{k+1}|r_i}$, per ogni i, dove k è l'ultimo step. Quindi $\mathbf{r_{k+1}|r_0=a, r_{k+1}|r_1=b}$.

Se d|a e d|b, allora $\mathbf{d|r_i}$, per ogni i, quindi $d|r_{k+1}$. Pertanto $\mathbf{d \leq r_{k+1}}$.

Ovvero r_{k+1} e' un divisore di a e b, e ogni altro divisore d è minore di r_{k+1}, ovvero $r_{k+1}=MCD(a,b)$.

Revisione Algoritmo di Euclide per calcolare il MCD, con invarianti

```
// calcolo di mcd(a,b) (con a≥b, altrimenti invertiamo)
Z=a; W=b; i=0 // a=r_0, b=r_1, Z=r_i, W=r_{i+1}
while (TRUE) { // a=r_0, b=r_1, Z=r_i, W=r_{i+1}
    if (W == 0) return Z; // Z = r_i = mcd(a,b), i=k+1
    Q = Z div W; R = Z mod W; // Q = q_{i+1}, R= r_{i+2}
    Z = W; W = R; // Q = q_{i+1}, Z=r_{i+1}, W=R= r_{i+2}
    i = i+1; // Q = q_i, Z=r_i, W=R= r_{i+1}
}
```

Teorema I: $(\exists x,y)\ ax+by=MCD(a,b)$

Dimostriamo che $(\exists x_i,y_i)\ ax_i+by_i=r_i$ (da cui la tesi per i=k+1)

$r_0=a=1*a+0*b$ $\qquad$ $x_0=1,\ y_0=0$

$r_1=b=0*a+1*b$ $\qquad$ $x_1=0,\ y_1=1$

$a=r_0=q_1*b+r_2=q_1*r_1+r_2$

quindi $r_2=r_0-q_1*r_1=a-q_1*b$ $\qquad$ $x_2=1,\ y_2=-q_1$

…

Supponiamo che $ax_i+by_i=r_i$ fino a i-1, quindi anche per i-1 e i-2, allora, dato che $r_{i-2}=q_{i-1}*r_{i-1}+r_i$, abbiamo

$$r_i=r_{i-2}-q_{i-1}*r_{i-1}=(ax_{i-2}+by_{i-2})-q_{i-1}*(ax_{i-1}+by_{i-1})=$$

$$=a(\underbrace{x_{i-2}-q_{i-1}*x_{i-1}}_{x_i})+b(\underbrace{y_{i-2}-q_{i-1}y_{i-1}}_{y_i}).$$

Algoritmo di Euclide (con spazi vuoti)

// calcolo di r_{k+1}=mcd(a,b) e x,y t.c. ax+by=mcd(a,b)

```
Z=a; W=b; i=0,


while (T) {
    if (W == 0) return Z;


    Q = Z div W; R = Z mod W;  Z = W; W = R;
                                     i=i+1; }
```

Algoritmo di Euclide esteso (con x_i, y_i)

// calcolo di r_{k+1}=mcd(a,b) e x,y t.c. ax+by=mcd(a,b)

Z=a; W=b; i=0, X2=1, Y2=0

Z=b; W=a mod b; Q=a div b; i=1; X1=0; Y1=1;

while (T) { X=X1; Y=Y1;

if (W == 0) return Z,X,Y;

X=X2-Q*X1; Y=Y2-Q*Y1;

Q = Z div W; R = Z mod W; Z = W; W = R;
X2,Y2←X1,Y1; X1,Y1←X,Y; i=i+1; }

Algoritmo di Euclide esteso (con x_i,y_i) (con invarianti)

// calcolo di r_{k+1}=mcd(a,b) e x,y t.c. ax+by=mcd(a,b)

Z=a; W=b; i=0, X2=1, Y2=0

// i=0, a=r_0, b=r_1, Z=r_i, W=r_{i+1}, X2=x_i,Y2=y_i

Z=b; W=a mod b; Q=a div b; i=1; X1=0; Y1=1;

// i=1, Z=r_i, W=r_{i+1}, Q=q_i, X1=x_i,Y1=y_i, X2=x_{i-1},Y2=y_{i-1}

Algoritmo di Euclide esteso (con x_i,y_i)

while (T) { // $Z=r_i$, $W=r_{i+1}$, $Q=q_i$, $X1=x_i$,$Y1=y_i$,$X2=x_{i-1}$,$Y2=y_{i-1}$

X=X1; Y=Y1; // $X=x_i$,$Y=y_i$

if (W == 0) return Z,X,Y; // $Z=r_i=mcd(a,b)=aX+bY$

X=X2-Q*X1; // $X=x_{i-1}-q_i*x_i=x_{i+1}$

Y=Y2-Q*Y1; // $Y=y_{i-1}-q_i*y_i=y_{i+1}$

Q = Z div W; R = Z mod W; // $Q = q_{i+1}$, $R= r_{i+2}$

Z = W; W = R; // $Q = q_{i+1}$, $Z=r_{i+1}$, $W=R= r_{i+2}$

X2,Y2←X1,Y1; X1,Y1←X,Y;//$X2=x_i$,$Y2=y_i$,$X1=x_{i+1}$;$Y1=y_{i+1}$

i=i+1;// $Q=q_i$,$Z=r_i$,$W=R=r_{i+1}$,$X2=x_{i-1}$,$Y2=y_{i-1}$,$X1=x_i$,$Y1=y_i$ }

Come usare Euclide Generalizzato

Problema: dati n,s t.c. MCD(n,s)=1

Calcolare t t.c. st mod n = 1

Soluzione

Dati a=n,b=s,

usare Euclide generalizzato per ottenere X e Y t.c. (X*a+Y*b)=MCD(a,b)=1

Avremo che X*n+Y*s = 1, quindi Y*s = 1 – X*n

Ovvero Y*s mod n = 1

Algoritmo per calcolare IM

Esempio a=60, b=13

X2	X1	Z	Y2	Y1	W	Q	X	Y	R
1		60	0		13	4	1	-4	8
0	1	13	1	-4	8	1	-1	5	5
1	-4	8	-1	5	5	1	2	-9	3
-1	5	5	2	-9	3	1	-3	14	2
2	-9	3	-3	14	2	1	5	-23	1
-3	14	2	5	-23	1				

5*60+(-23)*13=1 (X*a+Y*b)=MCD(a,b)=1

Esempio per RSA (a)

Siano p=11, q=7, n=77, (p-1)(q-1)=60, e=13

calcolare l'inverso moltiplicativo di e, ovvero il numero d tale che $d*e \equiv 1 \pmod{60}$, supponendo che MCD(e,60)=1

Allora, per il teorema visto, esistono X,Y t.c. X*e+Y*60=1,

e possono essere trovati con l'algoritmo di Euclide esteso.

Da quanto appena visto, risulta

5*60+(-23)*13=1, quindi $(-23)*13 \equiv 1 \pmod{60}$

Siccome -23 = 37 mod 60, anche $37*13 \equiv 1 \pmod{60}$,

infatti 37*13 = 481 = 8*60+1.

Quindi d=37 è l'inverso moltiplicativo di 13 (mod 60).

Esempio per RSA (b) – MCD(m,n)=1

Siano p=11, q=7, n=77, (p-1)(q-1)=60, e=13, allora d=37

Sia m=2, allora $c=m^{13} \bmod 77 = 2^{13} \bmod 77 =$

$= 8192 \bmod 77 = 106*77+30 \bmod 77 = 30$

Ora, $c^d \bmod n = 30^{37} \bmod 77 = (30^3)^{12}*30 \bmod 77 =$

$(27000 \bmod 77)^{12}*30 \bmod 77 = 50^{12}*30 \bmod 77 =$

$(50^2)^6*30 \bmod 77 = (2500 \bmod 77)^6*30 \bmod 77 =$

$36^6*30 \bmod 77 = (36^6 \bmod 77)*30 \bmod 77 =$

$36*30 \bmod 77 = 1080 \bmod 77 = (14*77+2) \bmod 77 = 2 = m$

Esempio per RSA (c) – MCD(m,n)≠1

Siano p=11, q=7, n=77, (p-1)(q-1)=60, e=13, allora d=37

Sia m=7, allora $c=m^{13} \bmod 77 = 7^{13} \bmod 77 = (7^4)^3 * 7 \bmod 77 =$

$= 14^3 * 7 \bmod 77 = (2744 \bmod 77) * 7 \bmod 77 =$

$= 49 * 7 \bmod 77 = 343 \bmod 77 = (4*77+35) \bmod 77 = 35$

Ora, $c^d \bmod n = 35^{37} \bmod 77 = (35^4 \bmod 77)^9 * 35 \bmod 77 =$

$= 49^9 * 35 \bmod 77 = (49^2 \bmod 77)^4 * 49 * 35 \bmod 77 =$

$= (14^4 \bmod 77) * 49 * 35 \bmod 77 = 70 * 49 * 35 \bmod 77 = 7 = m$

Generazione delle chiavi RSA

Miller-Rabin

- Scegli p e q primi
- calcola il modulo $n = pq$
- scegli $e < (p-1)(q-1)$ tale che $\gcd(e,(p-1)(q-1))=1$
- calcola $d = e^{-1} \bmod (p-1)(q-1)$ $K^{+} = <e,n>$, $K^{-} = <d,n>$

Euclide esteso

Per cifrare $m < n$

$c = m^e \bmod n$

algoritmo efficiente per Exp mod n

Per decifrare $m < n$

$m = c^d \bmod n$

Operazioni con bit vector

Esempio: per calcolare p*q dove p = AB, q = CD

```
                AB
                CD
                ==
          A*D,B*D
   A*C,B*C   -----
==============
XXXXXXXXXXXX
```

Operazioni con bit vector - esempio

11*14=154, in binario 154 = 10011010, 11=1011, 14 = 1110

```
     1011  (A||B=2||3)
     1110  (C||D=3||2)
=========
      110  (B*D=3*2=6)
    100==  (A*D=2*2=4)
   1001==  (B*C=3*3=9)
  110====  (A*C=2*3=6)
=========
 10011010  (somma colonna per colonna con riporti = 154 = 11*14)
```

Lunghezza del modulo RSA

n = pq deve essere abbastanza grande:

- perché m < n (il messaggio m è più corto)
- perché deve essere difficile fattorizzare n (quindi occorre n molto grande, si consiglia normalmente n di almeno 1024 bit)

RSA challenge

Il 18 marzo 1991, la RSA labs pubblicò una serie di sfide, con premi in denaro, con numeri n da fattorizzare. Tra queste la RSA-100:

n = 15226050279225333605356183781326374297180681149613
80688657908494580122963258952897654000350692006139

p*q = 37975227936943673922808872755445627854565536638199 ×
40094690950920881030683735292761468389214899724061

Questo n, di circa 330 bit, fu fattorizzato il primo aprile 1991 da Arjen Lenstra, vincitore di 1000$.

RSA challenge

Furono poi pubblicate altre sfide, alcune mai vinte, altre che portarono alla fattorizzazione di moduli n di varie lunghezze, tra cui un modulo di 768 bit e uno di 704 bit.

La sfida RSA è ora non più attiva, ma i numeri sono sempre pubblicati, è quindi possibile tentare la loro fattorizzazione.

E se il messaggio m è più lungo?

In teoria, è possibile usare RSA come cifrario a blocchi, e gestire messaggi più lunghi con le stesse tecniche usate per i cifrari simmetrici (per esempio CBC - Cipher Block Chaining)

E se il messaggio m è più lungo?

- In pratica, questo non serve, perché n=512 o 1024 bit, mentre
  - per cifrare, si combina RSA con un cifrario simmetrico, e basterà cifrare con RSA una chiave simmetrica K - di solito K è di 56 bit (DES), o di 128 o 112 bit (AES o 3DES)
  - per autenticare, si cifra con RSA un valore generato da una funzione di hash, la cui lunghezza è solitamente 128 bit (MD5) o 160 bit (SHA-1).

Esercizi – approfondimenti

Teorema: $(\exists x,y)\ ax+by=MCD(a,b)$ // già visto in precedenza

Corollario 1: se p è primo e $p|ab$, allora $p|a$ o $p|b$

Dimostrazione:

Se $p|a$ abbiamo concluso, allora si supponga che $\neg(p|a)$.

Siccome p è primo, $MCD(a,p)=1$ oppure $MCD(a,p)=p$.

Ma poiché $\neg(p|a)$, dovrà essere $MCD(a,p)=1$.

Per il Teorema, abbiamo $(\exists x,y)\ px+ay=1$. Moltiplichiamo per b entrambi i termini dell'uguaglianza: $bpx+aby=b$. Siccome $p|p$ e $p|ab$, allora $p|bpx+aby$, ovvero $p|b$.

Esercizi – approfondimenti

Corollario 2: se p è primo e $p|a_1a_2\ldots a_n$, allora p divide uno dei fattori a_i.

Dimostrazione:

Se $p|a_1$ abbiamo concluso, allora si supponga che $\neg(p|a_1)$.

Allora $p|(a_2\ldots a_n)$.

Si procede così finché o si è trovato a_i tale che $p|a_i$,

o si ha che $p|a_{n-1}a_n$, qundi, per il Corollario 1, $p|a_{n-1}$ oppure $p|a_n$.

Esercizi – approfondimenti

Teorema:

(a) ogni intero è un prodotto di fattori primi

(b) questa fattorizzazione è unica, a meno dell'ordine dei fattori

Dimostrazione (a):

L'intero 1 è un prodotto di 0 primi e ogni primo è un prodotto di un solo primo.

Si supponga per assurdo che n sia il più piccolo intero che non è un prodotto di primi. Quindi questo n non può essere il numero 1, né un primo.

Quindi n=ab. Siccome n è il più piccolo intero che non è prodotto di primi, a e b sono prodotti di primi. Ma se n=ab, anche n è un prodotto di primi, ciò che produce una contraddizione.

Esercizi – approfondimenti

Teorema:

(a) ogni intero n è un prodotto di fattori primi

(b) questa fattorizzazione è unica, a meno dell'ordine dei fattori

Dimostrazione (b):

Supponiamo che n si possa scrivere come due diversi prodotti di primi:

$n = p_1^{a1} p_2^{a2} \ldots p_n^{an} = q_1^{b1} q_2^{b2} \ldots q_m^{bm}$, dove gli esponenti a_i e b_i sono non nulli.

Se un primo p compare in entrambe le fattorizzazioni, semplifichiamo entrambe le fattorizzazioni dividendo per p, e ripetiamo. A valle della semplificazione, supponiamo quindi $p_1 \neq q_j$, per ogni j.

Ora, abbiamo che $p_1|n$, quindi $p_1| q_1^{b1} q_2^{b2} \ldots q_m^{bm}$. Per il corollario 2, p_1 deve dividere uno dei fattori $q_1, q_2, \ldots, q_m$, ma questo non è possibile perché i q_i sono primi, e $(\forall i)\ p_1 \neq q_i$.

Proprietà (II) - esercizio

se $n|XY$ e $MCD(X,n)=1$, allora $n|Y$

Dimostrazione (vedi anche esercizio in slide DH):

Per il Teorema I visto in precedenza, abbiamo
$(\exists w,z)\ nw+Xz=1$.
Moltiplichiamo per Y entrambi i termini, ottenendo:
$nwY+XzY=Y$.
Siccome $n|n$ e $n|XY$, allora $n|nwY+XzY$, ovvero $n|Y$.

Proprietà (III)

se MCD(a,n)=1, e $n \neq 0$, allora

$ab \equiv ac \bmod n \rightarrow b \equiv c \bmod n$

Dimostrazione:

Per il Teorema I visto in precedenza, se MCD(a,n)=1, $(\exists x,y)$ $ax+ny=1$. Moltiplicando per (b-c) si ha:

$(b-c)*(ax+ny) = b-c$

$(b-c)*ax+(b-c)*ny = b-c$

$(ab-ac)*x+(b-c)*yn = b-c$

Siccome $ab \equiv ac \bmod n$, $(\exists k)$ $ab-ac = kn$, quindi

$b-c = (ab-ac)*x+(b-c)*yn = knx+(b-c)yn = [kx+(b-c)y]*n$, quindi

b-c è multiplo di n, ovvero $b \equiv c \bmod n$