RSA

28/03/2023

È il primo CIFRARIO ASIMMETRICO creato ed il più usato.

PREMESSA: I testi da cifrare (e decifrare) sono dei numeri. Più nello specifico si tratta di un numero che deve essere più piccolo di un determinato numero n, noto come MODULO RSA

RSA è composto da:

- ALGORITMO X GENERARE & GESTIRE CHIAVI
- ALGORITMO X CIFRARE
- ALGORITMO X DECIFRARE

La COMPLESSITÀ di RSA risiede nel GENERARE & GESTIRE CHIAVI. Più nello specifico, si basa sulla difficoltà di calcolare la FATTORIZZAZIONE di numeri ottenuti come prodotto di grandi NUMERI PRIMI.

p, q NUMERI PRIMI

$n = pq$ MODULO RSA

m = MESSAGGIO t.c. $m < n$

$e < (p-1)(q-1)$ t.c. $MCD(e, (p-1)(q-1)) = 1$

$d = e^{-1} \ mod \ (p-1)(q-1) = 1$

CHIAVE PUBBLICA $K^+ = \langle e, n \rangle$

CHIAVE PRIVATA $K^- = \langle d, n \rangle$

CIFRARE $c = m^e \ mod \ n$

DECIFRARE $m = c^d \ mod \ n$

Generazione CHIAVI

MODULO RSA (n)

- Scelgo p, q numeri PRIMI molto grandi
- Li moltiplico $\mathbf{n = pq}$ e ne calcolo il modulo — È possibile farlo in complessità Polinomiale (Quadratica)

Componente e (chiave pubblica)

- Genero $\boldsymbol{e < (p-1)(q-1)\ t.c.\ MCD\left(e,(p-1)(q-1)\right) = 1}$

  Genero e in modo casuale tale che sia inferiore e RELATIVAMENTE primo rispetto a $(p-1)(q-1) \sim n$
  Una idea per generare questo numero, ad esempio, può essere generare e primo

Componente d (chiave privata)

- Calcolo $\boldsymbol{d = e^{-1} mod\ (p-1)(q-1) = 1}$

  Calcolo dell'inverso moltiplicativo di e. — È possibile farlo in complessità Polinomiale (Lineare)

Fatto ciò, ho le mie due chiavi, composte dalle seguenti coppie:
Chiave Pubblica: $\boldsymbol{K^{+} = \langle e, n \rangle}$
Chiave PRIVATA: $\boldsymbol{K^{-} = \langle d, n \rangle}$

Ci si potrebbe chiedere: non potrebbe fare la stessa cosa l'avversario?
Problema della FATTORIZZAZIONE: trovare n a partire dal prodotto pq per cui non si pensi esiste un algoritmo in tempo Polinomiale.

CIFRARE e DECIFRARE

CIFRARE: $\mathbf{c = m^{e} \bmod n}$
DECIFRARE: $\mathbf{m = c^{d} \bmod n}$

Abbiamo visto come questa operazione (Esponente Modulare) sia una operazione veloce; vedi Diffie Hellman.

Correttezza di RSA

30/03/2023

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

Detto in termini matematici, devo dimostrare che se $\mathrm{c} = \mathrm{m}^e \bmod \mathrm{n} \Rightarrow \mathrm{c}^d \bmod \mathrm{n} = \mathrm{m}$

Ovvero, devo dimostrare che:

$$\mathrm{m} = \mathrm{c}^d \bmod \mathrm{n} = (\mathrm{m}^e \bmod n)^d \bmod \mathrm{n} \equiv \mathrm{m}^{ed} \bmod n$$

Sapendo che e, d sono Inversi Moltiplicativi:

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

Dimostrazione (lez.30-03-2023):

Passo 1: Dimostro che $\varphi(n) = (p-1)(q+1)$

Caso particolare: p, q PRIMI $\Rightarrow n = pq$ e $\varphi(n) = (p-1)(q+1)$

Poiché:

$\varphi(n) =$ numeri interi $< n$ | numeri interi *relativamente primi* con n

Siccome p, q PRIMI, i multipli di p, q NON sono relativamente primi con n.

Detto in altre parole: $\text{MCD}(n, i \cdot p) = p$, $\text{MCD}(n, j \cdot q) = p$

segue che i multipli di p, q NON sono relativamente primi con n; tutti gli altri si

$\varphi(n) = \{$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 $q\}$

$= n - 1 - \{p, 2p, \dots, (q-1)p, q, 2q, \dots, (p-1)q\}$ Mi fermo a p-1 e q-1; Non vado oltre altrimenti supero n

$= n - 1 - \{(q-1) + (p-1)\}$ Conto i multipli di p o di q (non coprimi)

$= n - 1 - \{q - 1 + p - 1\}$

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

$= n - q - p + 1$

$= pq - q - p + 1$ Sostituisco $n = pq$

$= (p-1)(q+1)$ 😊

Passo 2: uso il Teorema di Eulero $m^{\varphi(n)} \bmod n = 1$ se m, n RELATIVAMENTE PRIMI

Assumo che m, n RELATIVAMENTE PRIMI

$m^{k \cdot \varphi(n)} \bmod n = 1^k$ Elevo entrambi i membri per un qualsiasi valore k

$\left(m^{\varphi(n)}\right)^k \bmod n = 1$

$m\left(m^{(p-1)(q-1)}\right)^k \bmod n = 1 * m$ Moltiplico entrambi i membri per m

$m^{k \cdot \varphi(n)+1} \bmod n = m$

$m^{k \cdot (p-1)(q-1)+1} \bmod n = m$ Uso la nostra ipotesi

Dire che $a = 1 \bmod b$ equivale a dire (per la definizione di modulo) che $a = kb + 1$

Ad Esempio: $a = 16, b = 3 \rightarrow k = 5$

Noi sappiamo che e, d sono Inversi Moltiplicativi e quindi: $ed \equiv 1 \pmod{(p-1)(q-1)}$

Applichiamo il concetto sopra descritto:

$$a = 1 \bmod b \rightarrow ed = 1 \bmod (p-1)(q-1) \rightarrow ed = k(p-1)(q-1) + 1$$

$m^{k' \cdot (p-1)(q-1)+1} \bmod n$ Sscelgo un k'

$= m^{ed} \bmod n$ Il testo cifrato (m^e)è uguale al testo decifrato (m^e)

$= m$ 😊 RSA funziona!

Non possiamo però ritenerci ancora soddisfatti: se m, n NON sono *relativamente primi*?!

m, n NON sono *relativamente primi*; questo accade ad esempio quando $m = p$ (testo da cifrare=p)...ma più in generale (come avevamo visto sopra) per multipli di p o di q.

Il Teo. di Eulero non vale! Tuttavia, RSA funziona ancora!

Caso: $m = j \cdot p$ →Teo. di Eulero non vale ma RSA funziona!

Dimostrazione:

Assumo che j<q (poiché altrimenti m=p*q=n)

$m^{\varphi(q)} \equiv 1 \; mod \; q$ Teo. di Eulero; vero poiché *m,q* sono relativamente primi (e lo sono visto che q è primo)

$\left(m^{\varphi(q)}\right)^{\varphi(p)} \equiv 1^{\varphi(p)} \; mod \; q$ Elevo entrambi i membri per $\varphi(p)$

$m^{\varphi(n)} \equiv 1 \; mod \; q$ Essendo q e p primi: $\varphi(q) = q - 1, \varphi(p) = p - 1$

$\left(m^{\varphi(n)}\right)^{i} \equiv 1^{i} \; mod \; q$ Elevo entrambi i membri per un generico i

$m^{\varphi(n)*i} = kq + 1$ Uso che $a = 1 \; mod \; b \rightarrow a = kb + 1$

$m \cdot m^{\varphi(n)*i} = m \cdot kq + m \cdot 1$ Moltiplico entrambi i membri per $m = jp$

$m^{i \cdot \varphi(n)+1} = kjpq + m$

$m^{i \cdot \varphi(n)+1} = kjn + m$ Sapendo che $n = pq$

$m^{i \cdot \varphi(n)+1} = m^{i \cdot (p-1)(q-1)+1} \equiv m \bmod n$

$m^{k' \cdot (p-1)(q-1)+1} \; mod \; n$ Uso il fatto che e,d sono Inversi moltiplicativi; scelgo un k'

$= \mathrm{m}^{ed} \bmod \; n$ Il testo cifrato (m^e)è uguale al testo decifrato (m^e)

$= m$ 😊 RSA funziona!

La dimostrazione è analoga per $m = i \cdot q$

CALCOLO INVERSO MOLTIPLICATIVO

Tale ALGORITMO prende il nome di **EUCLIDE GENERALIZZATO** e si basa sul **TEOREMA DI EUCLIDE** per il calcolo dell'MCD:

$$
\begin{aligned}
&Z = a; W = b;\\
&While\,(TRUE)\,\{\\
&\quad if(W == 0)\\
&\quad\quad return\;\; Z;\\
&\quad R = Z\%W;\\
&\quad Z = W;\;\; W = R;\\
&\}
\end{aligned}
$$

$Z = MCD\,(a,b)$

$$
MCD\,(a,b) = \begin{cases} a \;\; se \;\; b = 0 \\ MCD(\,b, a \;\; mod \;\; b) \;\; se \;\; b \neq 0 \end{cases}
$$

Resto

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

A ogni passo shifto a sx le variabili.

Mi fermo!

Soffermiamoci ora sulla sua COMPLESSITÀ.
Il resto va da un minimo di 0 ad un massimo di $W-1$.

CASO PEGGIORE: Il resto è uguale a poco meno di quello che avevo prima. Segue che devo fare W iterazioni...

☹

ESPONENZIALE

CASO MEDIO: Il resto è circa la metà. Segue che dimezzo il numero ad ogni iterazione ...

☺

LOGARITMICA

Correttezza dell'MCD

Possiamo vedere i passi dell'algoritmo come una sequenza di resti e quozienti...che possono enumerare:

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, $r_k=q_{k+1}*r_{k+1}+r_{k+2}$

Il Massimo Comun Divisore è il più grande divisore comune tra i due input (a, b), che appunto divide sia a che b.

Dimostrazione (x Induzione):

Dimostro che se esiste un divisore d di a, b, allora dividerà anche tutti i resti della sequenza che sto calcolando.

Se $d|a$ e $d|b$, allora $d|r_i$, per ogni i
Infatti: $d|a = r_0$ e $d|b = r_1$, inoltre $r_k = q_{k+1} \cdot r_{k+1} + r_{k+2}$

Per induzione, supponiamo che il divisore d divide sia r_k che r_{k+1}.
Se d divide r_k allora quest'ultimo sarà della forma d per un certo moltiplicatore m. Analogamente r_{k+1}:
$d|r_k$ e $d|r_{k+1}$ allora, $r_k = dm$ e $r_{k+1} = dn$,per qualche m, n.

Noi sappiamo che $r_k = q_{k+1} \cdot r_{k+1} + r_{k+2} \Rightarrow r_{k+2} = r_k - q_{k+1} \cdot r_{k+1}$. Pertanto:

$r_{k+2} = r_k - q_{k+1} \cdot r_{k+1}$ Sostituisco $r_k = dm$ e $r_{k+1} = dn$
$= dm - q_{k+1} \cdot dn$ Raccolgo d
$= d(m - q_{k+1} \cdot n)$ Ottengo come prima che d moltiplica un certo moltiplicatore...
$= d|r_{k+2}$ Segue che d divide r_{k+2}

Ho così dimostrato che: se $d|r_k$ e $d|r_{k+1}$ allora dividerà anche r_{k+2}(and so on...)
Detto in altre parole: Se c'è un divisore, dividerà anche tutti i resti che seguono

Dimostro che ogni resto divide tutti i resti precedenti.

$r_{k+1}|r_i$ per ogni i, dove k è l'ultimo step (ovvero $r_{k+2} = 0$)
Infatti: $r_k = q_{k+1} \cdot r_{k+1} + 0$. Quindi: $r_{k+1}|r_k$. Inoltre: $r_{k+1}|r_{k+1}$
Per Induzione supponiamo che $r_{k+1}|r_j$ e $r_{k+1}|r_{j+1}$. Dimostriamo che $r_{k+1}|r_{j-1}$ (resto precedente)
Ovvero $r_j = mr_{k+1}$ e $r_{j+1} = nr_{k+1}$ per qualche m,n
$r_{j-1} = q_j \cdot r_j + r_{j+1}$
$= q_j \cdot mr_{k+1} + nr_{k+1}$
$= (q_j \cdot m + n)r_{k+1}$
$= r_{k+1}|r_{j-1}$

Ho così dimostrato che l'ultimo resto r_{k+1} divide tutti i resti precedenti (e quindi anche a,b)

Infine, mi resta da dimostrare che sia il MASSIMO comun divisore:

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

Se $d|a$ e $d|b$ Se esiste un altro divisore che divide a,b
allora $d|r_i$ per ogni i Allora divide tutti i resti precedenti
Quindi: $d|r_{k+1}$ e quindi divide anche l'ultimo resto

Pertanto: $d \leq r_{k+1}$ Se un numero divide un altro numero, allora è minore di quest'ultimo
Ovvero: r_{k+1} è un divisore di a, b e ogni altro divisore d è minore di r_{k+1} ovvero $r_{k+1} = MCD(a, b)$

IDENTITÀ DI BEZOUT: $(\exists x, y) \quad ax + by = MCD\,(a, b)$

Ci dice che l'MCD di due numeri qualsiasi interi è esprimibile come COMBINAZIONE LINEARE.

Dalla dimostrazione (↓) ricaviamo anche come calcolare questi due numeri in maniera efficiente: **LOGARITMICA** ☺
Questo poiché segue lo stesso identico schema di EUCLIDE.

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

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

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

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

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

...

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

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

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

Dimostrazione per Induzione

MOLTIPLICATORI COMBINAZIONE LINEARE

$$x_i = x_{i-2} - q_{i-1} \cdot x_{i-1}$$
$$y_i = y_{i-2} - q_{i-1} \cdot y_{i-1}$$

```
//Euclide: MCD(a,b)
Z = a; W = b;
while (TRUE) {
    if (W = = 0)
        return Z; // Z = MCD(a,b)
Q = Z / W;//Quoziente
R = Z mod W;//Resto
Z = W; //Faccio shift a sx
W = R; //Faccio shift a sx
}
```

L'idea è usare EUCLIDE per il calcolo dell'MCD, e contestualmente usare BEZOUT per calcolare x_i e y_i
Questo è quello che prende il nome di:
ALGORITMO DI **EUCLIDE ESTESO**

↓ ☺

COMPLESSITÀ
LOGARITMICA

```
//Calcolo i moltiplicatori ad ogni passo
Z = a; W = b;
i = 0; X2 = 1; Y2 = 0;
Z = b; W= a mod b; Q=a/b; i=1;
X1=0; Y1=1;//Valori Iniziali di X1 e Y1
while (TRUE) {
    X=X1; Y=Y1;
    if (W = = 0)
        return Z,X,Y;
    X=X2-Q*X1; //Xi=Xi-2-Q*Xi-1
    Y=Y2-Q*Y1; //Yi=Yi-2-Q*Yi-1
Q = Z / W;//Quoziente
R = Z mod W;//Resto
Z = W; //Faccio shift a sx
W = R; //Faccio shift a sx
X2=X1; Y2=Y1; //Faccio shift a sx
X1=X; Y1=Y; //Faccio shift a sx
i++;
}
```

L'idea è usare EUCLIDE GENERALIZZATO per il calcolo dell'INVERSO MOLTIPLICATIVO di RSA: $ed \equiv 1 \mod (p-1)(q-1)$

Im generale:
Dati a, b, me voglio calcolare l'INVERSO MOLTIPLICATIVO sapendo che $MCD(a,b) = 1$

↓ Implica che

$\exists$ COMBINAZIONE LINEARE
$t.c. \quad x \cdot a + y \cdot b = 1$
$\Rightarrow y \cdot b = 1 - x \cdot a$
$\Rightarrow y \cdot b \mod a = 1$

Nel caso di RSA abbiamo generato la CHIAVE PUBBLICA e in modo che fosse COPRIMO con $(p-1)(q-1)$:

↓ ovvero

$MCD = 1$

$x \cdot e = y \cdot (p-1)(q-1) = 1$

$x \cdot e = 1 - y \cdot (p-1)(q-1)$

$x \cdot e = 1 \mod (p-1)(q-1)$

↳ $x =$ INVERSO MOLTIPLICATIVO

ESEMPIO: RSA

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				

MCD

MOLTIPLICATORI BEZOUT

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

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 ≡ 1 (mod 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 ≡ 1 (mod 60)
Siccome -23 = 37 mod 60, anche 37*13 ≡ 1 (mod 60),
infatti 37*13 = 481 = 8*60+1.
Quindi d=37 è l'inverso moltiplicativo di 13 (mod 60).

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

Sia m=2, allora $c=m^{13} \bmod 77 = 2^{13} \bmod 77 =$
$= 8192 \bmod 77 = 106*77+30 \bmod 77 = 30$ CIFRO

Ora, $c^d \bmod n = 30^{37} \bmod 77 = (30^3)^{12}*30 \bmod 77 =$
$(27000 \bmod 77)^{12}*30 \bmod 77 = 50^{12}*30 \bmod 77 =$
$(50^2)^6*30 \bmod 77 = (2500 \bmod 77)^6*30 \bmod 77 =$
$36^6*30 \bmod 77 = (36^6 \bmod 77)*30 \bmod 77 =$
$36*30 \bmod 77 = 1080 \bmod 77 = (14*77+2) \bmod 77 = 2 = m$ DECIFRO

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

Sia m=7, allora $c=m^{13} \bmod 77 = 7^{13} \bmod 77 = (7^4)^3*7 \bmod 77 =$
$= 14^3*7 \bmod 77 = (2744 \bmod 77)*7 \bmod 77 =$
$= 49*7 \bmod 77 = 343 \bmod 77 = (4*77+35) \bmod 77 = 35$ CIFRO

Ora, $c^d \bmod n = 35^{37} \bmod 77 = (35^4 \bmod 77)^9*35 \bmod 77 =$
$= 49^9*35 \bmod 77 = (49^2 \bmod 77)^4*49*35 \bmod 77 =$
$= (14^4 \bmod 77)*49*35 \bmod 77 = 70*49*35 \bmod 77 = 7 = m$ DECIFRO

RIASSUMENDO

p, q NUMERI PRIMI

$n = pq$ MODULO RSA

m = MESSAGGIO $t.c.$ $m < n$

$e < (p-1)(q-1)$ $t.c.$ $MCD(e, (p-1)(q-1)) = 1$

$d = e^{-1} \ mod \ (p-1)(q-1) = 1$

CHIAVE PUBBLICA $K^+ = \langle e, n \rangle$

CHIAVE PRIVATA $K^- = \langle d, n \rangle$

CIFRARE $c = m^e \ mod \ n$

DECIFRARE $m = c^d \ mod \ n$

Per realizzare RSA occorre ALGORITMO per:

- GENERARE p, q NUMERI PRIMI → **TEST DI MILLER-RABIN** (vedi DIFFIE HELLMAN)
- CALCOLARE **INVERSO MOLTIPLICATIVO** → **EUCLIDE ESTESO**
  - ci consente di CALCOLARE la **CHIAVE PRIVATA** a partire dalla **CHIAVE PUBBLICA**
- CALCOLARE $a^b \ mod \ q$ → **ESPONENTE MODULARE** (vedi DIFFIE HELLMAN)

Il **TEOREMA DI EULERO** ci dimostra che CIFRANDO e poi DECIFRANDO, riottengo lo stesso messaggio.

FATTORIZZAZIONE: Trovare n conoscendo $p \cdot q$ (AVVERSARIO)

PROBLEMA: CALCOLARE **PRODOTTI** di **NUMERI** (molto) **GRANDI**

COMPLESSITÀ **QUADRATICA**

Per usare RSA, dobbiamo essere in grado di eseguire PRODOTTI di NUMERI GRANDI; NON è una operazione macchina.
L'idea è eseguire la moltiplicazione così come noi umani siamo abituati a fare:

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

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

Nel nostro ambito si parla di 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)
```

FATTORIZZAZIONE: Trovare n conoscendo $p \cdot q$

↓ ?

COMPLESSITÀ **ESPONENZIALE**

È il problema che si pone l'AVVERSARIO e di cui non si conosce (e si pensa non esserci) una soluzione in tempo polinomiale

Sicuramente il punto cruciale è scegliere dei numeri molto grandi...

Nel corso degli anni sono state proposte e risolte varie challenge
(esempio $RSA-100$ con n di 100 cifre $(\sim 300$ bit$)$)
e altre mai risolte.

PROBLEMA: **TESTI PIU' LUNGHI**

Come abbiamo già detto $m =$ MESSAGGIO e $m < n$. Per cui, potrebbe capitare che tale numero non basti.
È un problema che abbiamo già visto.

SOLUZIONI

→ **SOLUZIONE 1**: vedi CIFRARI A BLOCCHI

→ **SOLUZIONE 2**: vedi DIGITAL ENVELOPE (BUSTA DIGITALE)

→ NON mi pongo il problema! Uso RSA per cifrare la CHIAVE SIMMETRICA

→ che sicuramente rispetta $m < n$

Infine concludiamo dicendo che è possibile usare RSA per fare la FIRMA ELETTRONICA: in questo caso cifrerò un CODICE DI HASH