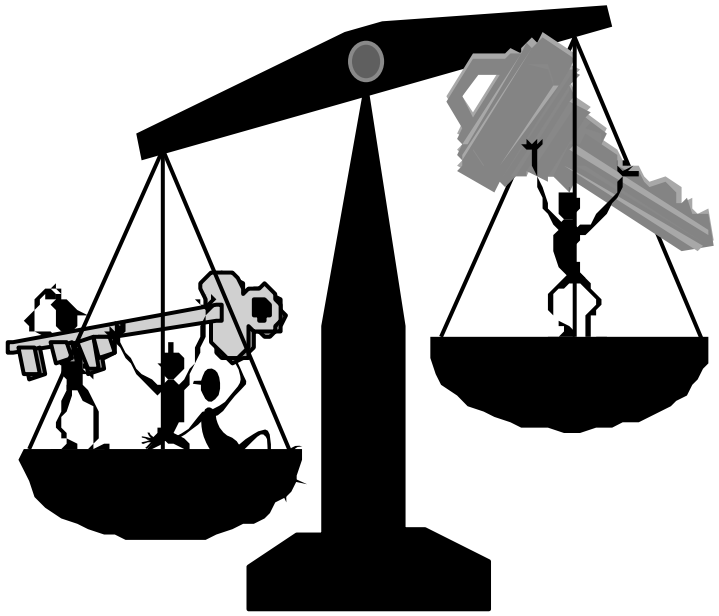
Meccanismi asimmetrici

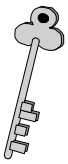


Crittografia a chiave pubblica

- **Cifrari**
- **Generatori di bit pseudocasuali**
- **Schemi di firma**
- **Protocolli d'identificazione attiva**

Ogni utente ha una chiave segreta SU

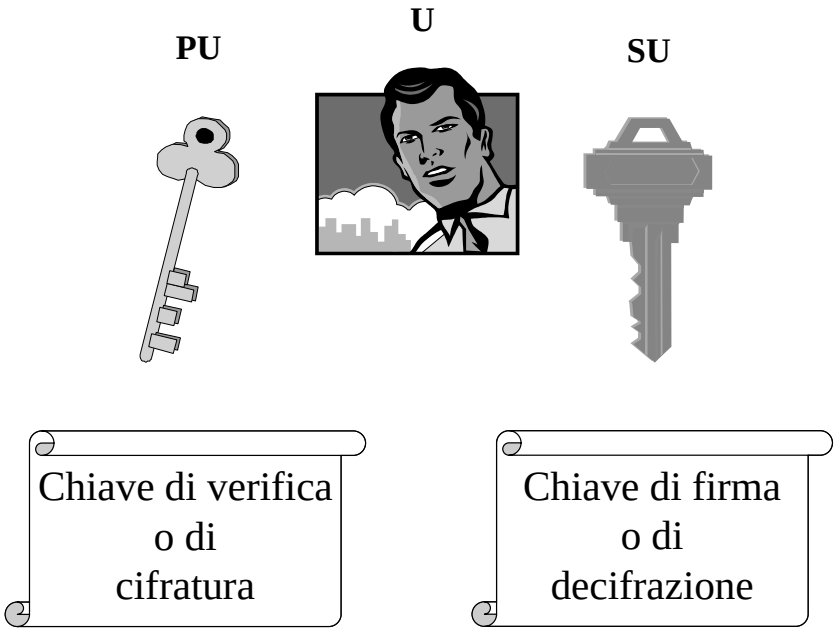
..e rende pubblica una chiave PU



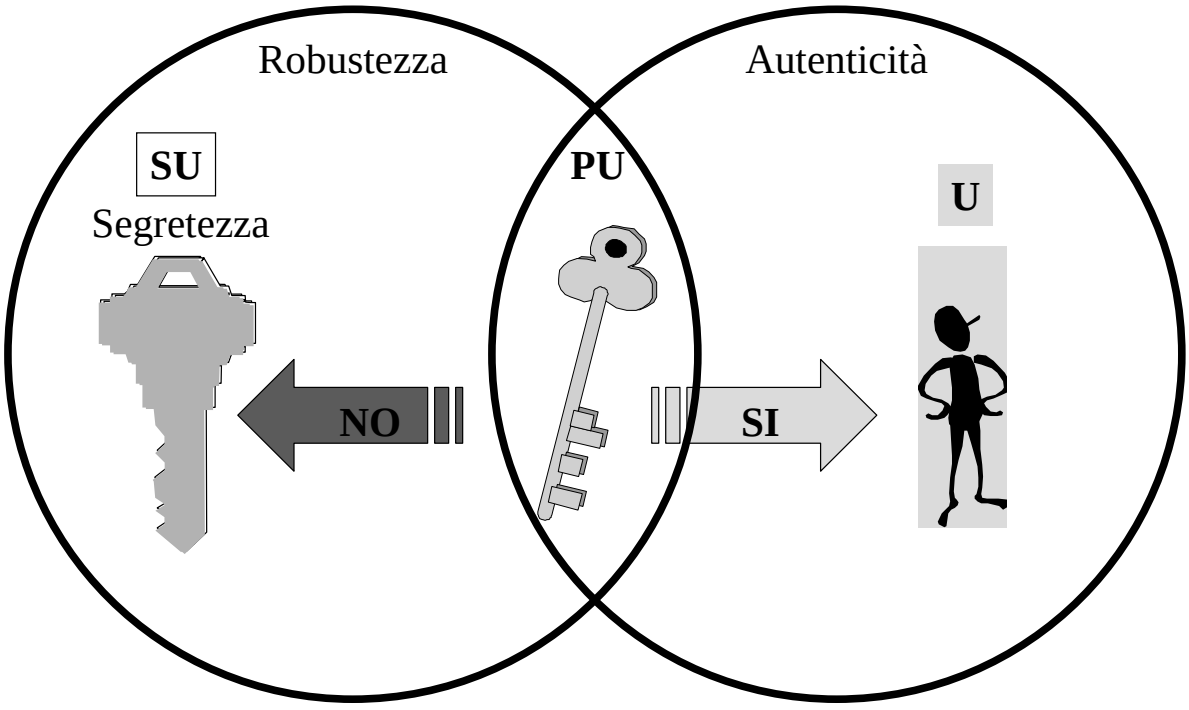
- Incontro
- Elenco
- e-mail
- Sito
- Database

PLaschi:
www.lia.deis.unibo.it
ldap://certserver.pgp.com

Le due chiavi ed il proprietario



Proprietà della chiave pubblica



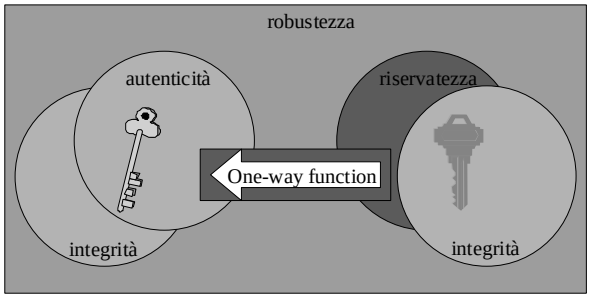
Robustezza degli algoritmi asimmetrici

- **funzione unidirezionale**
- **funzione pseudounidirezionale**
- **problemi difficili**

La relazione tra le chiavi

Una funzione **f** è unidirezionale se è invertibile, se il suo calcolo è facile e se per quasi tutti gli **x** appartenenti al dominio di **f** è difficile risolvere per **x** il problema **y = f (x)**.

Proprietà delle chiavi asimmetriche



Chiave pubblica **Chiave privata**

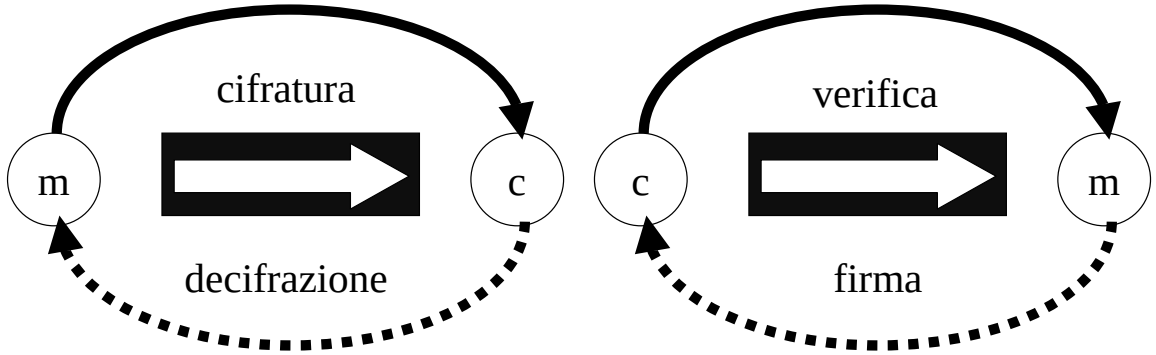
..e inoltre:
•cifra/decifra
•firma/verifica

esistenza?

Problemi difficili:
Teoria dei numeri
Aritmetica modulare

Le due trasformazioni

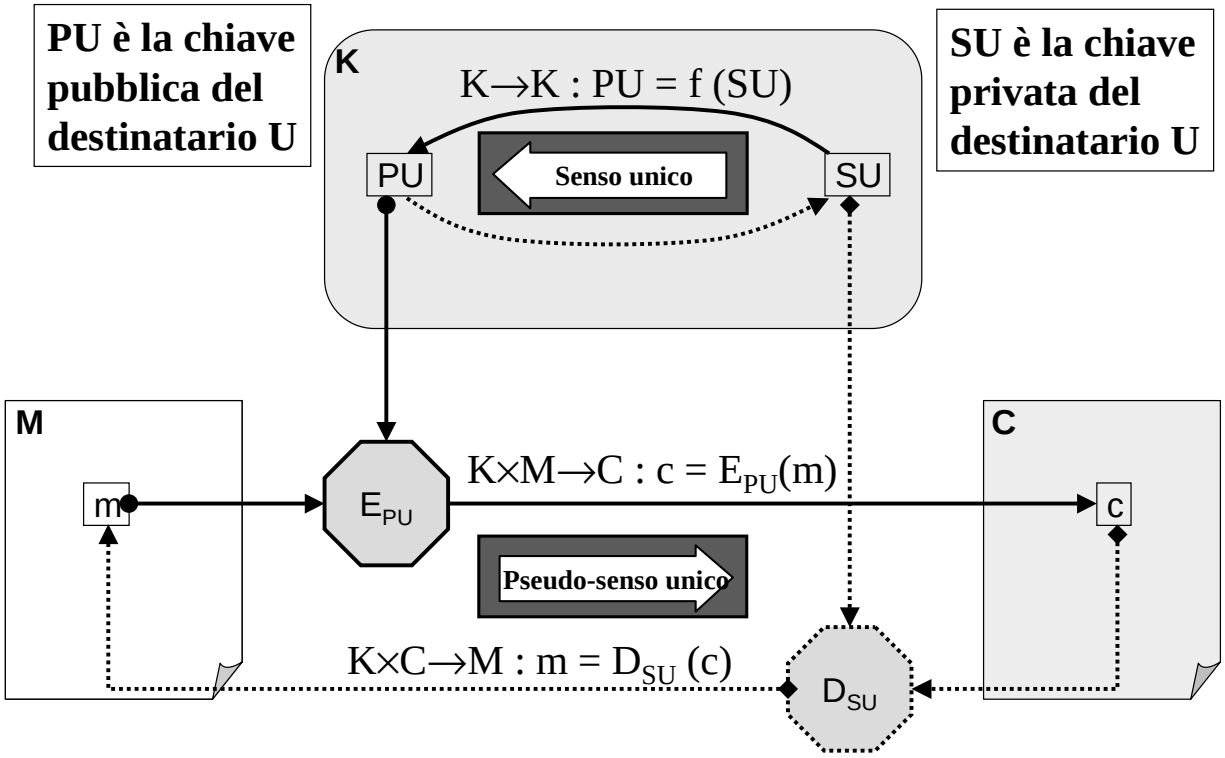
Una funzione **f** è pseudo-unidirezionale se appare come unidirezionale per chiunque non sia in possesso di una particolare informazione sulla sua costruzione.



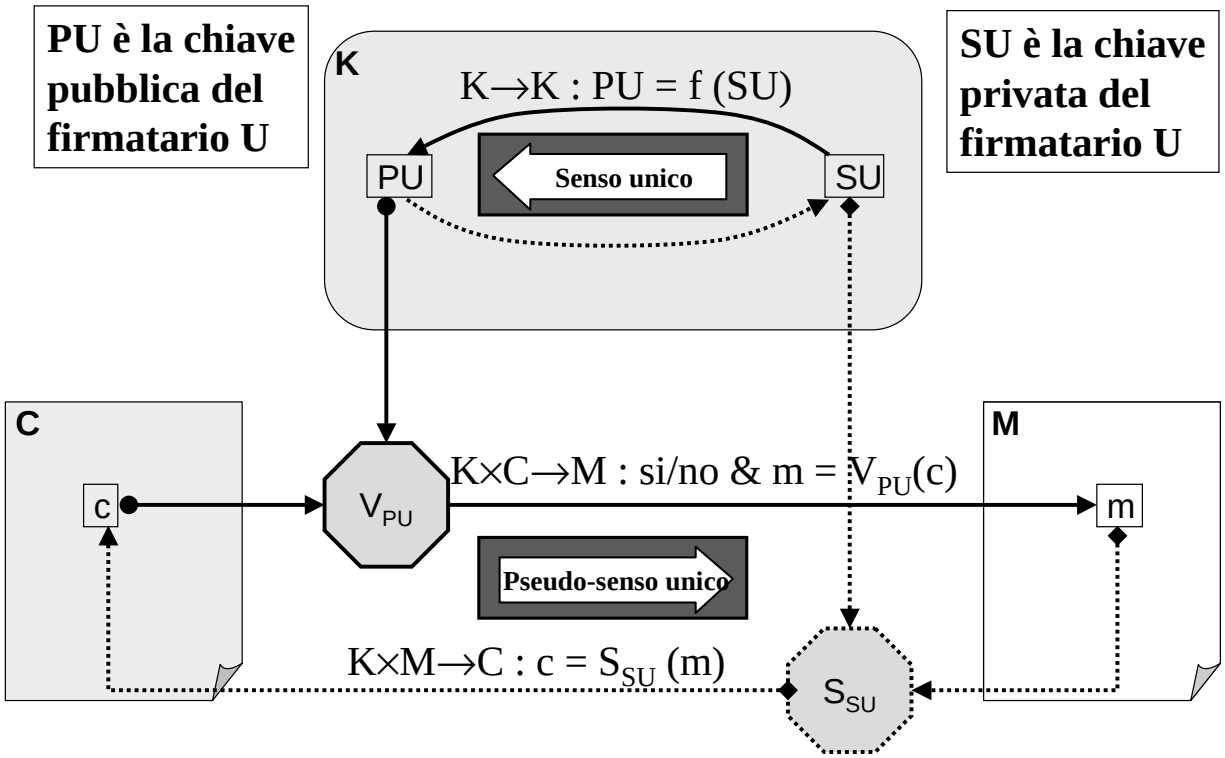
Trapdoor one-way function

Allo stato attuale delle conoscenze la trasformazione segreta degli algoritmi asimmetrici è facile da calcolare solo per chi ha la chiave privata. Probabilmente un altro metodo facile non verrà mai trovato

Il modello del Cifrario a chiave pubblica



Il modello della Firma digitale a chiave pubblica



Problemi difficili

Assunzione: per certi problemi della Teoria dei numeri non si troveranno mai algoritmi con tempo polinomiale

P1: logaritmo discreto (gruppo ciclico o GF(p^n))

Scambio DH, Cifrario ElGamal, Firma DSS
Safe prime: **p = 2q+1** con q primo (SKIP)

P2: radice e-esima - Dato un **n** prodotto di due primi, un **e** coprimo con Φ(n) ed un elemento **c** ∈ $\mathbf{Z_n}$ trovare un intero **m** tale che

$$m \equiv \sqrt[e]{c}$$

Cifrario RSA
Il problema è facile se si conoscono i fattori di n o se n è primo

P3: fattorizzazione

P3: problema della fattorizzazione - Dato un intero positivo **n**, trovare i numeri primi p_i ed i coefficienti interi $e_i \geq 1$ tali che

$n = p_1^{e1} \times ... \times p_k^{ek}$ (Crittografia asimmetrica: $n = p_1 \times p_2$)

Cifrario RSA

- Gauss e Fermat: **20** cifre decimali
- 1970: **41** cifre decimali con un main frame
- 1977, Rivest: **125** cifre decimali è un calcolo impossibile
- 1994: **129** cifre decimali in 8 mesi con 1.600 workstations
- 2000 (stima): **150** cifre decimali in un anno con una macchina parallela da 10 milioni di dollari
- 2004 (prev.): **300** cifre decimali (1024 bit)
- 2014 (prev.): **450** cifre decimali (1500 bit)

Algoritmi di fattorizzazione

Trial division:
1. i=1
2. individua p(i): i-esimo primo
3. calcola n/p(i)
se ***resto*** **≠ 0**, i = i+1 e goto 2
altrimenti **I° fattore =** ***quoziente***

I° fattore < sqrt(n)
O(exp (½ (log n)))

. . . . .

General Number Field Sieve:
$O(\exp (\log n)^{1/3} . (\log(\log n))^{2/3})$

Rivest, 1999
scelta casuale

"p, q *strong primes* : p-1, p+1, q–1, q+1 con grandi fattori primi

"$|p-q| > n^{1/2}$ "

1024 – 2048 bit

Altri problemi difficili

P4: problema del fusto - Dato un insieme di **n** interi positivi

$$\{a_1, a_2, .., a_n\}$$

ed un intero positivo **s** determinare se esiste o meno un sottoinsieme di a_j la cui somma è **s**.

P5: problema della radice quadrata modulare - Dato un **n** composto ed un elemento $a \in Q_n$ (insieme dei residui quadratici modulo n) trovare la sua radice quadrata modulo n, cioè
un intero $x \in Z^*_n$ tale che $x^2 \equiv a \pmod n$.

P6: problema della residuosità quadratica - Dato un **n** composto e dispari ed un elemento $a \in J_p$ (insieme degli interi con simbolo di Jacobi = 1) determinare se **a** è o meno un residuo quadratico modulo **n**.

Logaritmo discreto su curva ellittica

 P7: problema del logaritmo discreto su una curva ellittica - Data la curva ellittica formata da punti le cui coordinate x,y soddisfano l'equazione
$y^2 = x^3 + ax + b \pmod p$, con p primo,
e due suoi punti **P**, **Q** tali che $Q = n \times P$,
determinare **n**.

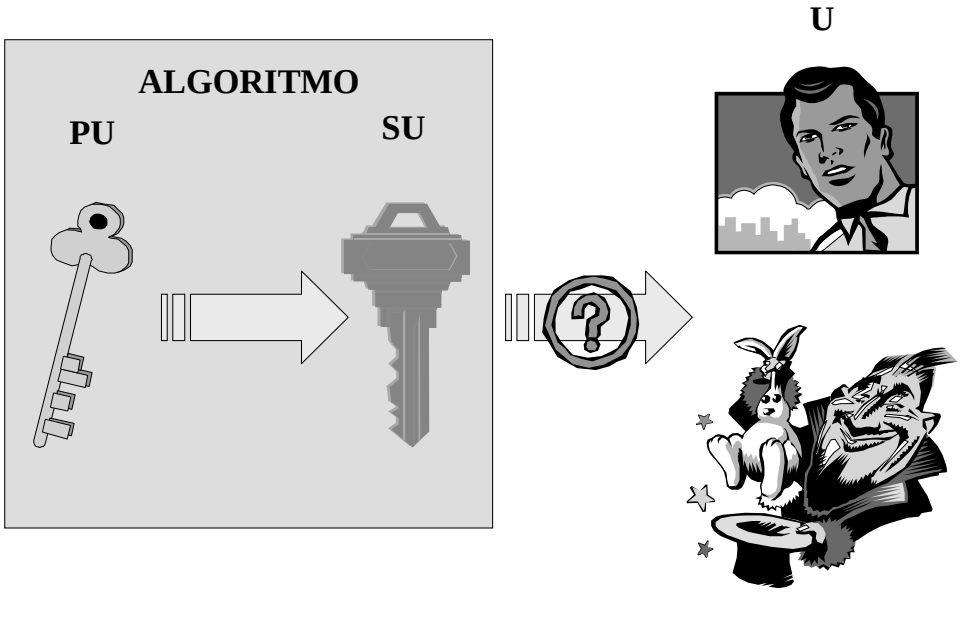
Complessità degli attuali algoritmi di rottura
$O(\exp(\frac{1}{2}(\log p)))$
160-180 bit

ECC: Certicom.com

Autenticità della chiave pubblica

- **l'attacco dell'uomo in mezzo**

Chiavi e Proprietario

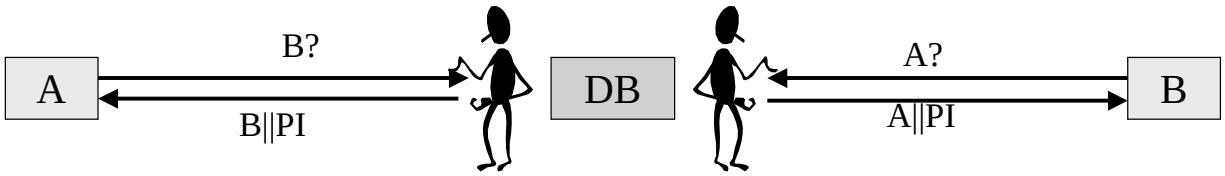


R28: "*prima d'impiegare una chiave pubblica bisogna o essere certi dell'identità del suo proprietario o poterla verificare*"

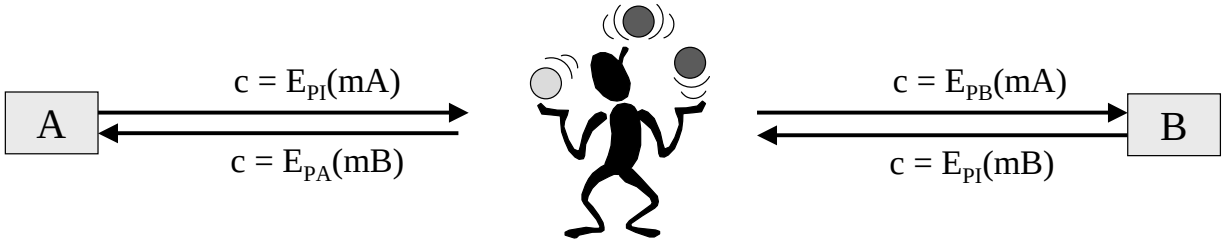
Attacco dell'uomo in mezzo

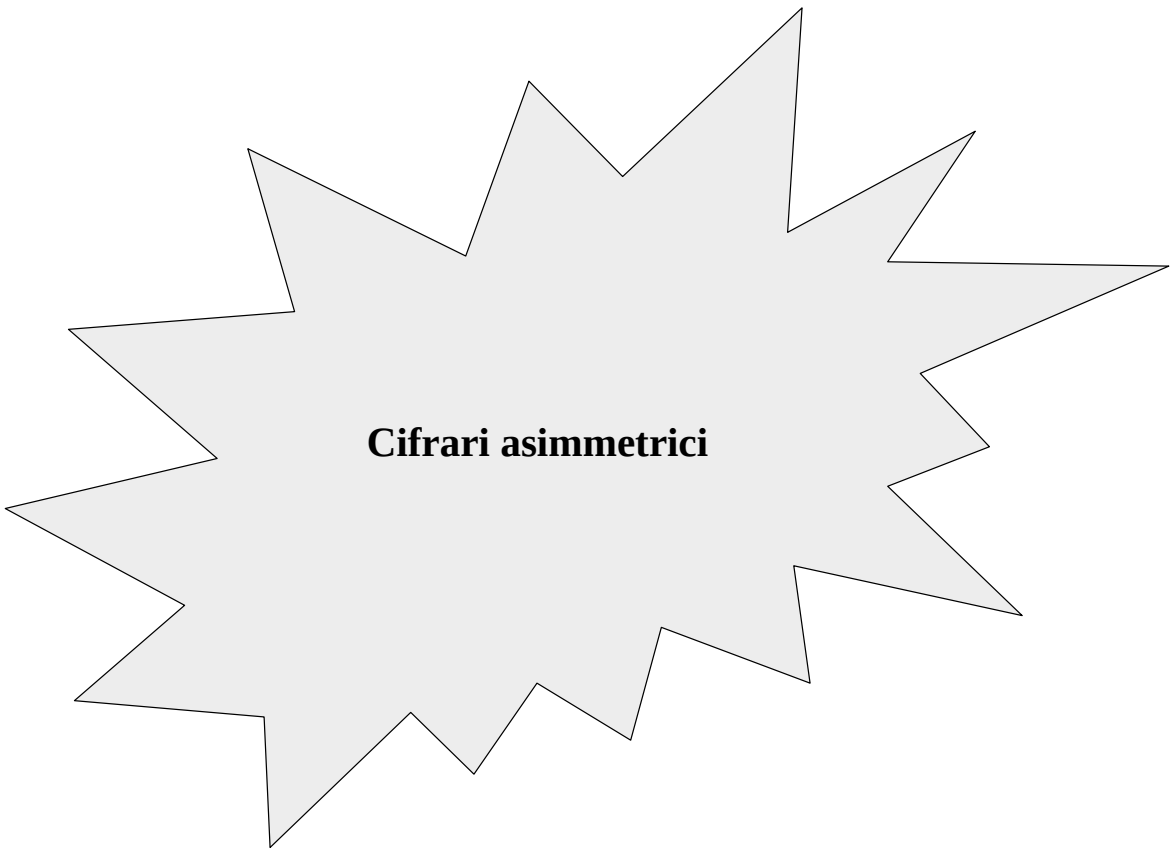
1 - Registrazione

2 - Intercettazione delle interrogazioni e falsificazione delle risposte

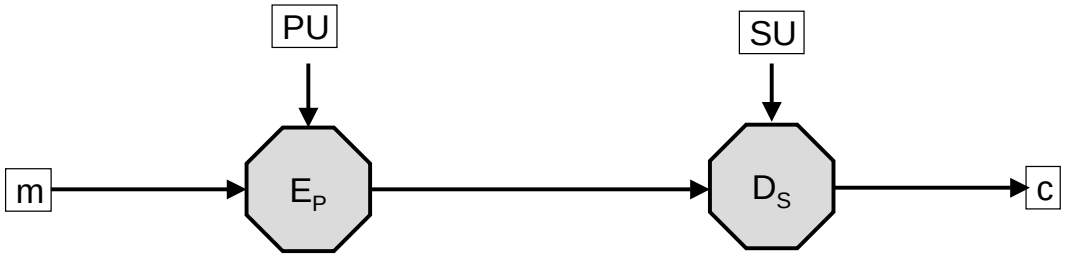


3 - Intercettazione, decifrazione, cifratura ed inoltro.



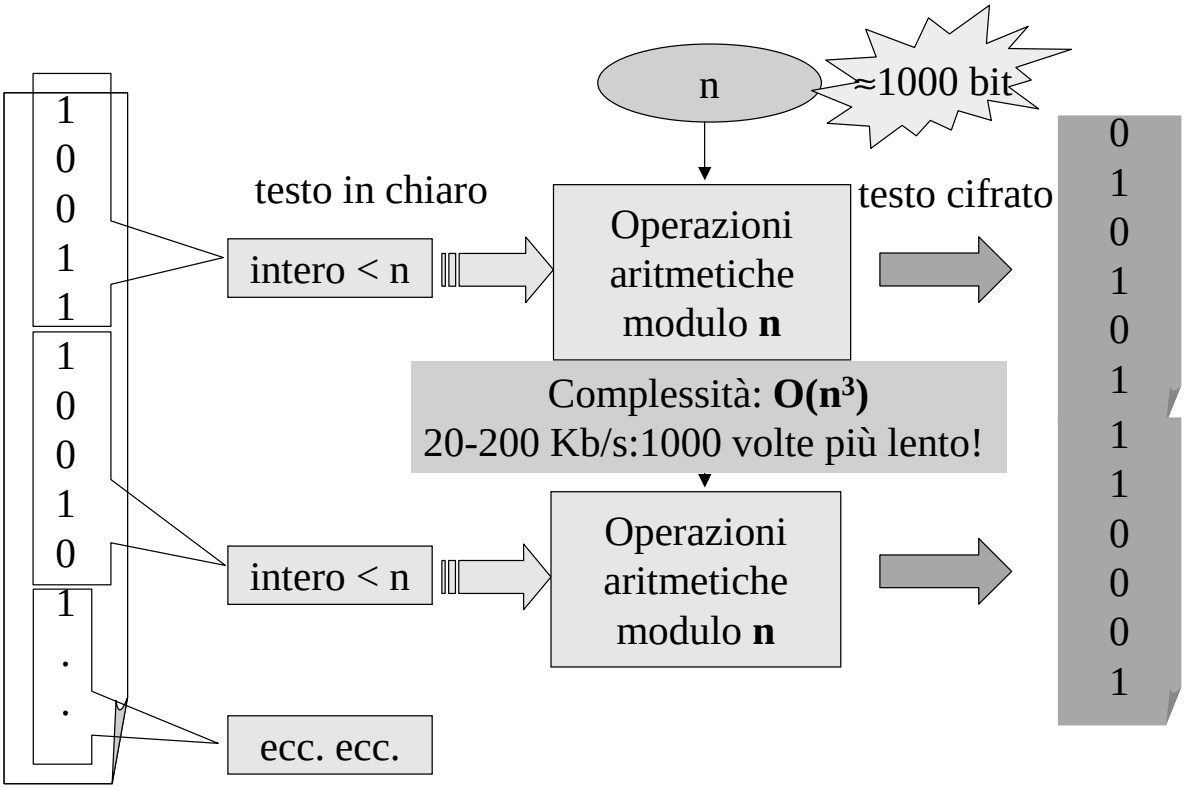


Aspetti caratteristici



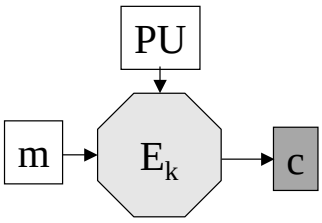
1. Frammentazione del testo in chiaro
2. Aleatorietà del testo cifrato
3. Variabilità della trasformazione
4. Problema difficile su cui si basa la sicurezza
5. Modalità d'impiego

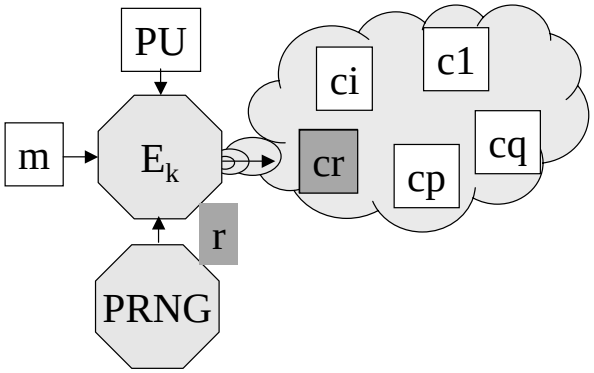
1 - Frammentazione del testo in chiaro



2 - Aleatorietà del testo cifrato

Cifrari deterministici e Cifrari probabilistici





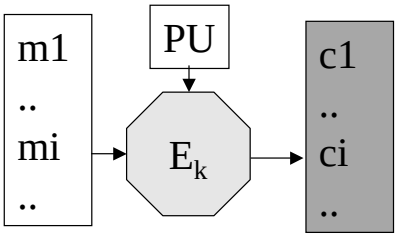
- Blocchi identici
- Primo blocco
- Attacco con testo noto

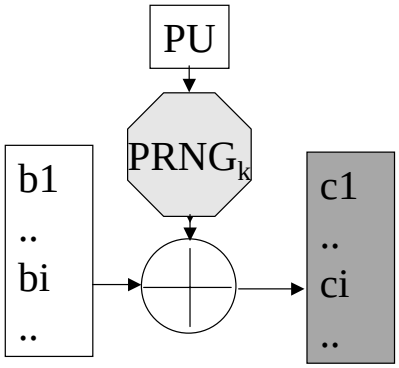
Casualità:
- IV & CBC
- c = E(m||r)

Ridondanza:
c = E(m)||E(r)

3 – Variabilità della trasformazione

Cifrari a blocchi e Cifrari a flusso





4 – I problemi difficili dei cifrari asimmetrici

Cifrario	Proprietà	Tipo	Sicurezza
RSA	deterministico	a blocchi	P2, P3
ElGamal	probabilistico	a blocchi	P1
Rabin	deterministico	a blocchi	P3, P5
Golwasser-Micali	probabilistico	a flusso	P6
Blum-Goldwasser	probabilistico	a flusso	P3, P5
Chor-Rivest	deterministico	a blocchi	P4

5 - Public Key Cryptography Standards

PKCS # 1 - The RSA encryption standard
PKCS # 3 - The Diffie-Hellman key-agreement standard
PKCS # 5 - The password-based encryption standard (PBE)
PKCS # 6 - The extended-certificate syntax standard (X509)
PKCS # 7 - The cryptographic message syntax standard
PKCS # 8 - The private-key information syntax standard
PKCS # 9 - This defines selected attribute types for use in other PKCS standards.
PKCS # 10 - The certification request syntax standard
PKCS # 11 - The cryptographic token interface standard
PKCS # 12 - The personal information exchange syntax standard.
PKCS # 13 - The elliptic curve cryptography standard
PKCS # 14 - This covers pseudo random number generation (PRNG).
PKCS # 15 - The cryptographic token information format standard.

The RSA Algorithm

Public key: {n,e}
n = p×q con p e q primi
e coprimo con **Φ(n)**

Encryption

- Plaintext: **M < n**
- Ciphertext: **$C = M^e \bmod n$**

Private key: {n,d}
$d = e^{-1} \bmod \Phi(n)$

Decryption

- Ciphertext: **C < n**
- Plaintext: **$M = C^d \bmod n$**

The RSA Algorithm (1998)

$\lambda(n) = mCM(p-1, q-1)$
$= \Phi(n)/MCD(p-1, q-1)$

Public key: {n,e}
n = p.q con p e q primi
e coprimo con **$\lambda(n)$**

Private key: {n,d}
$d = e^{-1} \bmod \lambda(n)$

La decrittazione di un testo cifrato con RSA

$C = M^e \bmod n$ ⇒ **$M = \sqrt[e]{C} \bmod n$**

P2: noti C, e, n calcolare M
☹

Altri attacchi a RSA

- **Fattorizzazione**: noti p e q, P2 diventa facile
  - NFS
  - Twinkle
- **Cycling attack**: se $(c^e)^k = c$, allora $(c^e)^{k-1} = m$
- **Timing attack**
- **Message unconcealed:** $m^e = m$

La trapdoor di RSA

$\mathbf{d = e^{-1} \bmod \Phi(n)}$

n = 3 × 11 e=3 Φ(n) = 20

y = 3x mod 20

x:																		
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19
y:																		
03	06	09	12	15	18	01	04	07	10	13	16	19	02	05	04	08	14	17

m	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16
$c=m^3$ mod 33	00	01	08	27	31	26	18	13	17	03	10	11	12	19	05	09	04
c^7 mod 33	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16

m	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
$c=m^3$ mod 33	29	24	28	14	21	22	23	30	16	20	15	07	02	06	25	32
c^7 mod 33	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

p,q segreti: per calcolare **Φ(n)** bisogna saper fattorizzare **n**

Key Generation

- Select *p,q* — ***p* and *q* both prime**
- Calculate *n* — ***n* = *p* x *q***
- Calculate Φ(*n*) — **Φ(*n*) = (*p*-1)(*q*-1)**
- Select integer *e* — **gcd(Φ(*n*),*e*) = 1; 1<e<Φ(*n*)**
- Calculate *d* — $\boldsymbol{d = e^{-1} \bmod \Phi(n)}$
- Public Key — **k[pub] = {e,n}**
- Private key — **k[priv] = {d,n}**

Example 8.5 (da [6], pag.117)

Chiave pubblica di B (destinatario)

p = 197; g = 2; cB = 82;

N.B. “cB” equivale al “g^u”precedente

Chiave privata di B

mB = 111;

N.B. “mB” equivale al “u” precedente

Cifratura (eseguita dal mittente)

r = Random[Integer, {0, p-2}] = 191

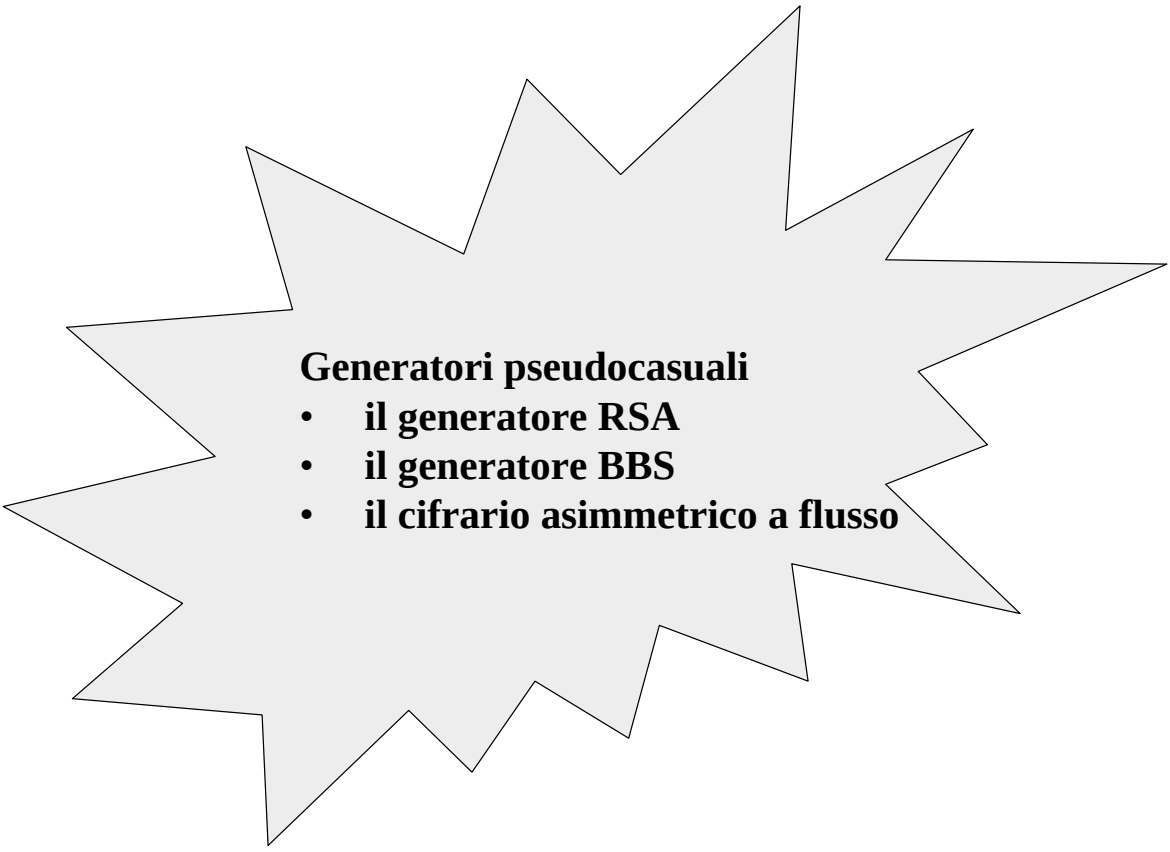
m = 123 elemento di {0,1,…,p-1}

R = PowerMod[g, r, p] = 117

S = Mod[PowerMod[cB, r, p] $\times$ **m, p]** = 175

Decifrazione (eseguita dal destinatario: $S \times R^{-mB}$ mod p)

Mod[S $\times$ **PowerMod[PowerMod[R, mB, p], -1, p], p]** = **123**



RSA pseudorandom bit generator

1. Si sceglie a caso due grandi primi **p** e **q**

2. Si calcola
   - 2.1 **n = pq**
   - 2.2 **Φ = (p-1)(q-1)**
   - 2.3 **e** tale che **MCD(e,Φ) = 1**

3. Si sceglie a caso un seme $x_0 < n-1$
(N.B. Intel generator consigliato)

4. Si itera quanto serve
   - 4.1 $\mathbf{x_i = x_{i-1}^e \bmod n}$,
   - 4.2 $\mathbf{b_i = x_i \bmod 2}$

(N.B. $\mathbf{b_i}$ è il bit meno significativo di $(\mathbf{x_i})_2$

```
#define  MAX_LUNGHEZZA 256
void RSAGenerator(intero
z[MAX_LUNGHEZZA],intero l){
intero n,p,q;
intero phi;
intero x[MAX_LUNGHEZZA];
p=prime(1000,5000);
q=prime(1000,5000);
n=p*q;
phi = (p-1)*(q-1);
intero e = random(1,phi);
while(mcd(e,phi)!=1) e =
random(1,phi);
x[0]=random(1,n-1); //seme
for (intero i=1;i<=l;i++) {
x[i]=powermod(x[i-1],e,n);
z[i-1]=x[i]%2;
printf("%d",x[i]%2);
}
}
```

Firma digitale

La firma digitale di un documento informatico deve:

1- consentire a **chiunque** di identificare **univocamente** il firmatario,

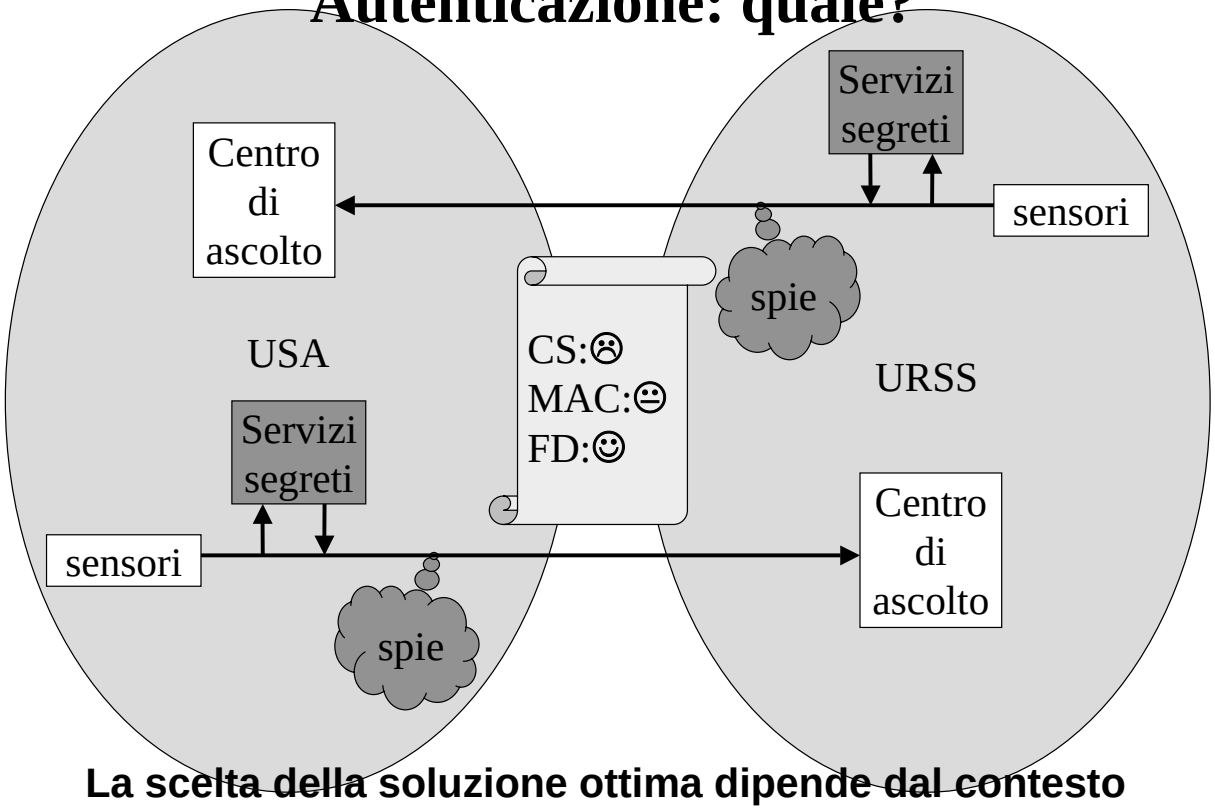
2- non poter essere **imitata** da un impostore,

3- non poter essere **trasportata** da un documento ad un altro,

4- non poter essere **ripudiata** dall'autore,

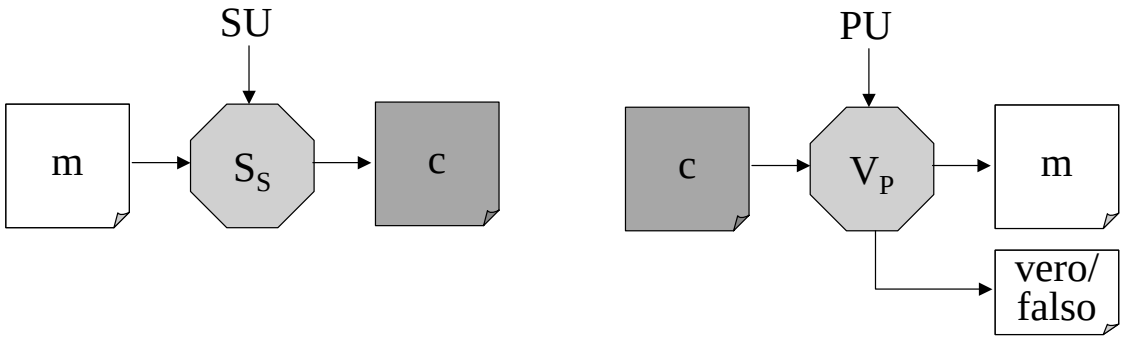
5- rendere **inalterabile** il documento in cui è stata apposta.

Autenticazione: quale?

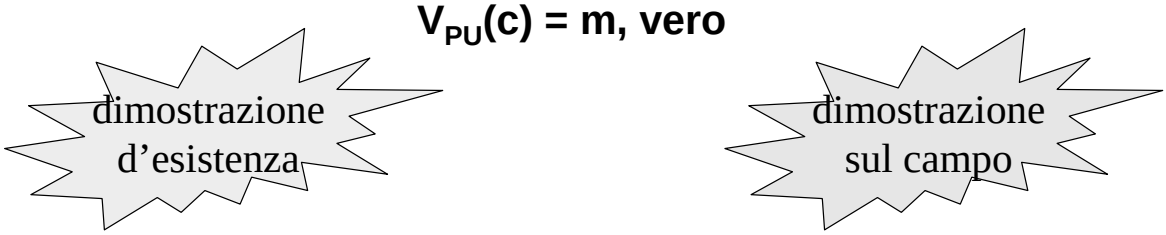


La scelta della soluzione ottima dipende dal contesto

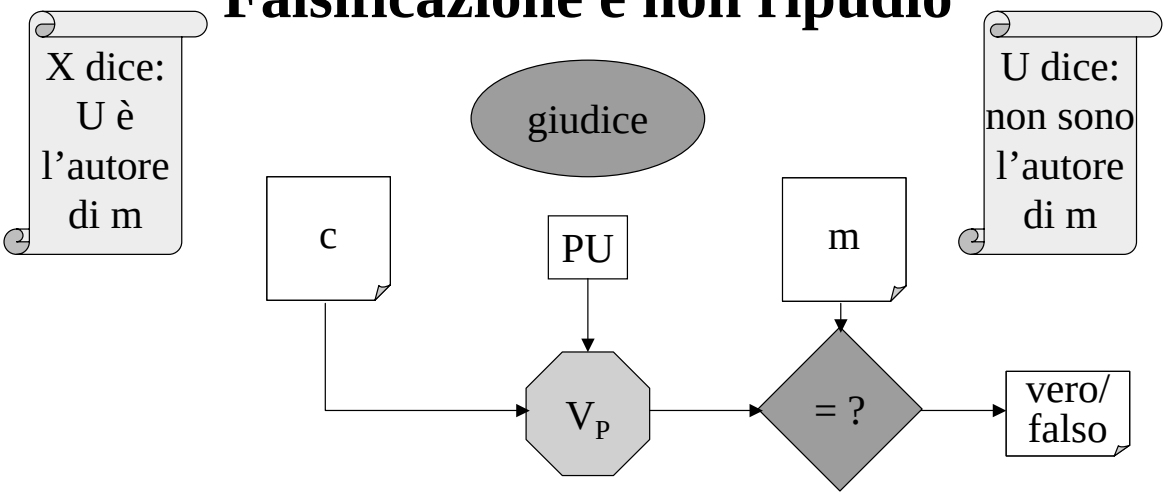
Sicurezza della firma digitale



- Per ogni SU, per ogni m e per $c = S_{SU}(m)$, deve essere
  $V_{PU}(c) = m$, vero
- Per chi non ha SU e per ogni m di sua invenzione deve essere infattibile costruire un c tale che
  $V_{PU}(c) = m$, vero



Falsificazione e non ripudio



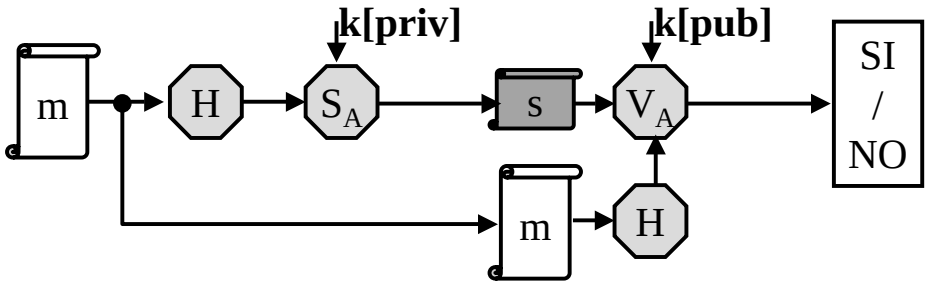
Valore legale della firma digitale
1989: USA e poi ONU
1997: ITALIA
1999: Comunità Europea

Algoritmi di firma a chiave pubblica

Nome	Tipo	Anno
RSA	ricupero	1978
Rabin	ricupero	1979
ElGamal	appendice	1984
DSA	appendice	1991
Schorr	appendice	1991
Nyberg-Rueppel	ricupero	1993

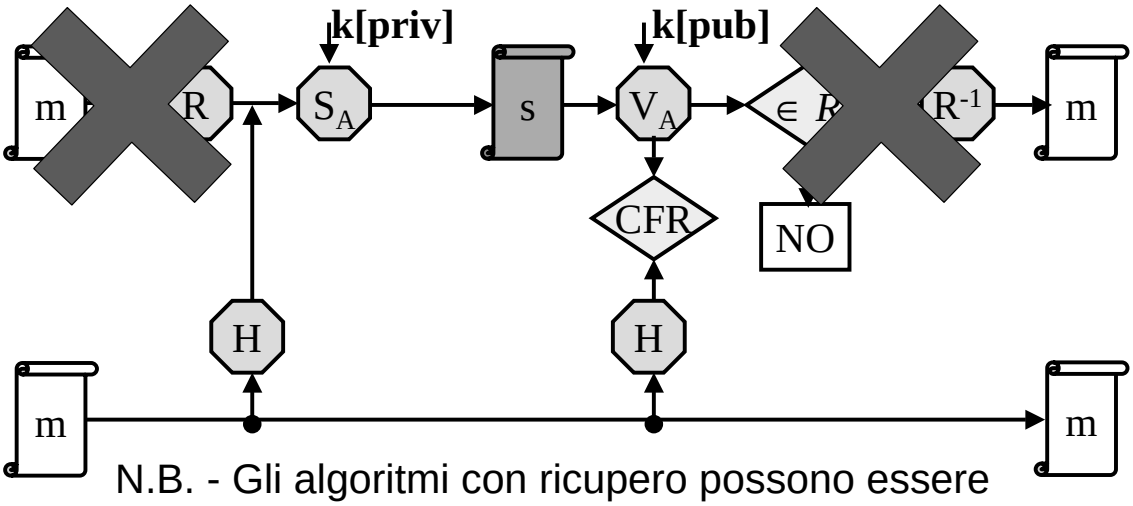
Algoritmi di firma con appendice

Messaggi di lunghezza arbitraria
Appendice: S(H(m),k[privA])



Algoritmi di firma con recupero

Messaggi “corti”: < modulo
Funzione di ridondanza R



N.B. - Gli algoritmi con ricupero possono essere impiegati anche in uno **schema con appendice**

Firma digitale

- **Algoritmo RSA**
- **Firma a occhi chiusi**

Proprietà di reversibilità di RSA

Reversibilità delle chiavi (impiego di SU al posto di PU e viceversa:
$E_{SU}(m) = c = m^{SU} \bmod n$ **messaggio non riservato**
$D_{PU}(c) = (m^{SU})^{PU} \bmod n = m$ **ma con origine verificabile**

<u>Schema di firma con recupero</u> **inefficiente se m > n**

<u>Schema di firma con appendice</u>: $\lceil \log_2 n \rceil$ bit di etichetta

1. FIRMA
$S_{SU}(H(m)) = (H(m))^{SU} \bmod n$ **autenticazione del messaggio**
$m \| S_{SU}(H(m))$ **comunicazione del messaggio**

2: VERIFICA
- **calcolo di H(m)**
- **calcolo di $(S_{SU}(H(m))^{PU} \bmod n$**
- **confronto**

Firma con RSA