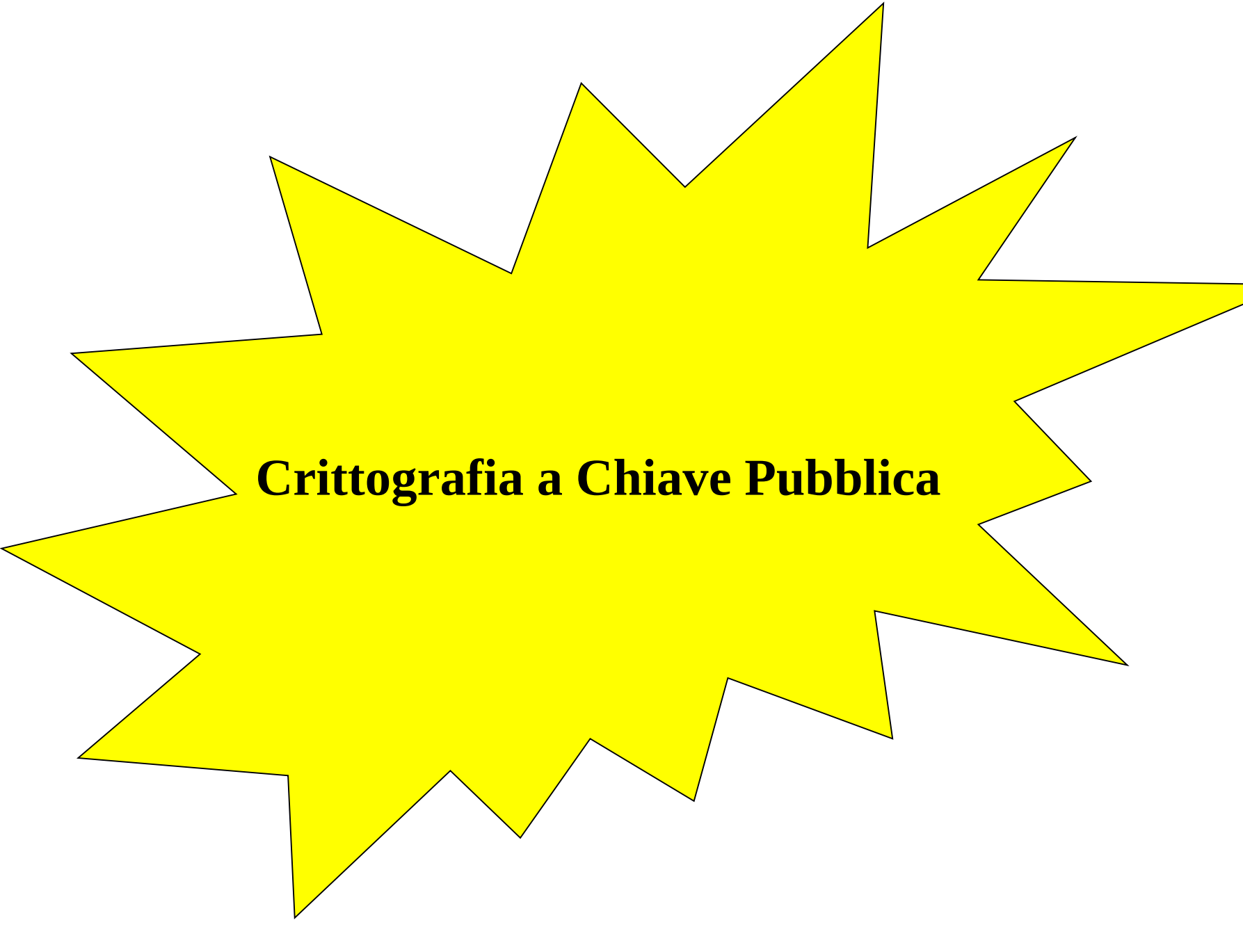


A large, bright yellow starburst shape with multiple sharp points, centered on the page. It has a black outline and serves as a background for the text.

Crittografia 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 $\text{GF}(p^n)$)

Scambio DH, Cifrario ElGamal, Firma DSS

Safe prime: $p = 2q+1$ con q primo di Sophie Germain

q si ottiene dalla progressione $6k-1$

P2: radice e-esima - Dato un n prodotto di due primi, un e coprimo con $\Phi(n)$ ed un elemento $c \in \mathbb{Z}_n$ trovare un intero m tale che

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

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

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

$$n = p_1^{e_1} \times \dots \times p_k^{e_k} \text{ (Crittografia asimmetrica: } n = p_1 \times p_2 \text{)}$$

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: TWINCLE, setaccio “optoelettronico” (2-3 ordini di grandezza)
- ❖ 2004 (prev.): **300** cifre decimali (1024 bit)
- ❖ 2014 (prev.): **450** cifre decimali (1500 bit)

Teoria dei numeri (1)

Utilizzata per: DH

Firma digitale

Cifratura a chiave pubblica

- **b divisore di a:**

$b \mid a$ se $a = m \times b$ con m intero

- **p numero primo:**

se i suoi divisori sono solo 1 e p

- **fattorizzazione di n:**

$n = p_1^{e_1} \times p_2^{e_2} \dots \times p_k^{e_k}$ con $p_1 \dots p_k$ primi, e_i interi ≥ 0

- **massimo comun divisore di a,b:**

$\text{MCD}(a,b) = k$ con k più grande divisore di a e di b

- **a, b coprimi (o relativamente primi): non hanno fattori primi in comune**

$\text{MCD}(a,b) = 1$

- **a mod n:**

resto della divisione di $a \geq 0$ per $n > 0$

- **a congruo a b modulo n:**

$a \equiv b \pmod{n}$ se $a \bmod n = b \bmod n$

Teoria dei numeri (2)

- Dato un intero positivo n si indica con $Z_n = \{0, 1, \dots, n-1\}$ l'insieme di tutti gli interi non negativi $< n$ (Z_n denota un anello in cui sono definite operazioni di addizione e moltiplicazione mod n)
 $Z_n^* = \{ a \in Z_n \mid \text{MCD}(a, n) = 1 \}$ insieme dei coprimi con n
 $Z_p^* = \{1, \dots, p-1\}$ insieme di tutti gli elementi non nulli di Z_p quando p è primo ($\text{gcd}(0, p) = p$)
- **funzione totiente di Eulero:** n° di interi $< n$ e coprimi con n
 $\Phi(n) = n \times (1 - 1/p_1) \times \dots \times (1 - 1/p_k)$ dove i p_k sono i primi che compongono la fattorizzazione di n
 1. $\Phi(n) = n-1$ se n è primo
 2. $\Phi(n) = (p-1)(q-1)$ se n è il prodotto di due primi p e q
 $\Phi(6) = 2$ (1 e 5)

Crittografia asimmetrica

x segreto

One-way

y pubblico

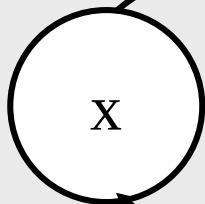
Scambio DH (nel campo finito $GF(p)$)

$Z_p = \{1, \dots, p-1\}$ $a+b \bmod p, a.b \bmod p$

$y = g^x \bmod p$, con p primo

•facile

•tempo polinomiale: $O((\log p)^n)$



$x = \log_g y$ in $GF(p)$

•difficile

•tempo esponenziale $O(\exp(\log p))$

•tempo subesponenziale

Crittografia asimmetrica

x segreto

Teoria dei
numeri

y pubblico

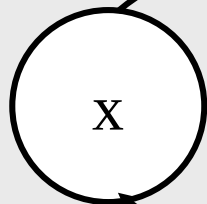
Scambio DH (nel campo finito $GF(p)$)

$Z_p = \{1, \dots, p-1\}$ $a+b \bmod p, a.b \bmod p$

$y = g^x \bmod p$, con p primo

•*facile*

•*tempo polinomiale: $O((\log p)^n)$*



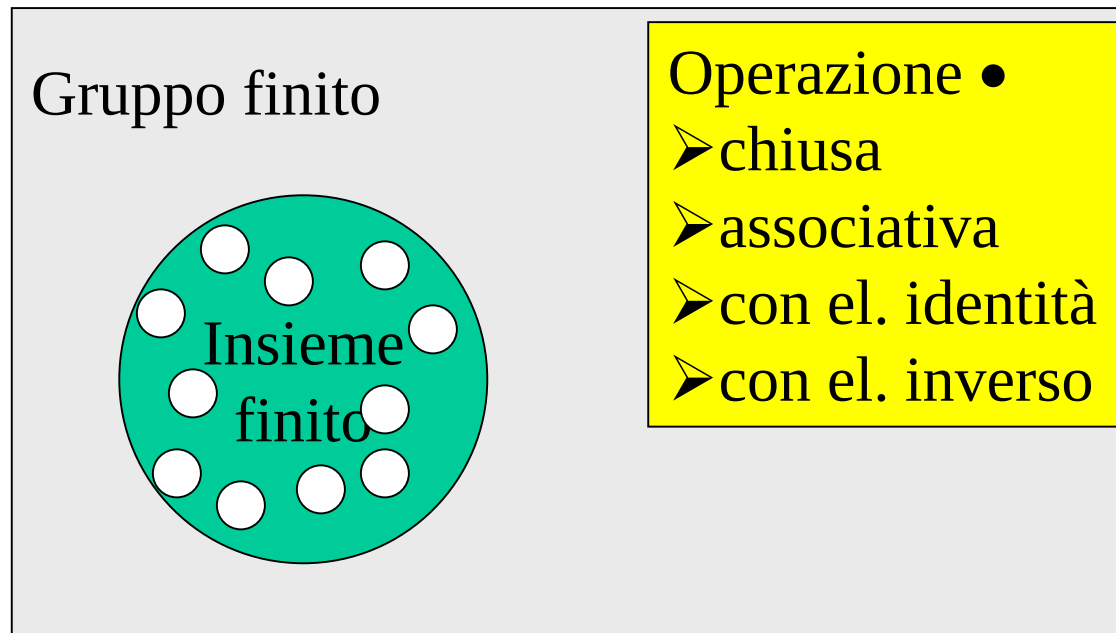
$x = \log_g y$ in $GF(p)$

•*difficile*

•*tempo esponenziale $O(\exp(\log p))$*

•*tempo subesponenziale*

Insieme finito di elementi affiancato da un'operazione binaria che associa a ciascuna coppia (a,b) di elementi di G un elemento $(a \bullet b)$ sempre di G con le proprietà sotto elencate



Chiusura: se a e b appartengono a G , anche $a \bullet b$ appartiene a G

Associatività: $a \bullet (b \bullet c) = (a \bullet b) \bullet c$ per ogni, a, b, c di G

El. Identità: esiste un elemento di G tale per cui: $a \bullet e = e \bullet a = a$ per ogni a

El. Inverso: per ogni a di G esiste un b di G tale che $a \bullet b = b \bullet a = e$

Gruppo abeliano

Gruppo finito



Operazione •

- chiusa
- associativa
- con el. identità
- con el. inverso

Operazione •

- Commutativa

Commutativa: $a \bullet b = b \bullet a$

Gruppo ciclico

Gruppo abeliano

Gruppo finito



Operazione •

- chiusa
- associativa
- con el. identità
- con el. inverso

Operazione •

- Commutativa

➤ generatore

Ogni el. g^k è una potenza con esp. intero di g , con g fisso

Anello

Gruppo ciclico

Gruppo abeliano

Gruppo finito



Operazione

- chiusa
- associativa
- commutativa
- con el. Identità
- distributiva rispetto
- alla somma

Operazione +

- chiusa
- associativa
- con el. identità
- con el. inverso

➤ generatore

Ogni el. è una potenza con esp. Intero di $g \in I$

Insieme finito di elementi con due operazioni binarie, somma e moltiplicazione con proprietà sopra elencate.

Un anello è un insieme i cui si possono eseguire somma, sottrazione ($a-b = a+(-b)$) e moltiplicazione senza uscire dall'insieme

Campo

Anello

Gruppo ciclico

Gruppo abeliano

Gruppo finito



Operazione •

- chiusa
- associativa
- con el. identità
- con el. inverso

Operazione •

- Commutativa

Operazione +

- chiusa
- associativa
- con el. identità
- con el. inverso

- generatore

Ogni el. è una potenza con esp. Intero di $g \neq I$

- distributiva

➤ Ogni elemento $\neq 0$ ha il suo inverso moltiplicativo (per ogni a eccetto 0, esiste un a^{-1} tale che $a(a^{-1}) = (a^{-1})a = 1$)

GF(p): elementi, operazioni e proprietà

Campo finito (Campo di Galois) è definito su $\mathbb{Z}_p$
(insieme con un numero finito di elementi su cui sono definite somma e molt. modulo p)

$$n = 1: \text{GF}(p)$$

$$\mathbb{Z}_p = \{0, 1, \dots, p-1\}$$

$$a, b \in \mathbb{Z}_p$$

$$a, b \in \mathbb{Z}_p^*$$

Addizione: $(a+b) \bmod p$

Moltiplicazione: $(a \times b) \bmod p$

$O(\log p)$ operazioni su bit

$O((\log p)^2)$ operazioni su bit

Proprietà notevole dell'Aritmetica modulare

Sia **m** primo o composto e siano **a, b** interi qualsiasi , allora

$$(a+b) \bmod m = ((a \bmod m) + (b \bmod m)) \bmod m$$

$$(a \times b) \bmod m = ((a \bmod m) \times (b \bmod m)) \bmod m$$

(conviene ricondursi a due numeri congruenti con quelli assegnati, eseguire le operazioni e poi dividere il risultato per m)

$$p = 11$$

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

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

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

GF(11): addizione

$$(3+1)+2 = 3+(1+2)$$

chiusura

associativa

commutativa

elemento
identità

inverso
additivo

	y										
x	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

$(x + y) \bmod 11$

Inverso
additivo:
Per ogni
w in
GF(11)
esiste z
tale che
w+z
congruo
con
0mod11

Tutti i ris. appartengono a Z_{11} (**chiusura**)

Ogni riga e colonna contiene una permutazione di Z_{11}

Scambiando l'ordine degli addendi si ottiene lo stesso risultato (commut.)

La somma di un elemento con 0 fornisce l'elemento stesso (0 è elem. identità)

In ogni riga/colonna lo 0 è presente solo una volta (esistenza dell'inverso additivo)

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

$$(x \times y) \bmod 11$$

Ogni riga della tabella (ad esclusione di $a=0$) è una permutazione in Z_{11}

L'elemento identità nella moltiplicazione è 1

L'inverso moltiplicativo per ogni a diverso da 0 è il valore di b che fornisce $axb \bmod 11 = 1$; tale valore è presente una sola volta (ad esclusione di $a=0$) in ogni riga e in ogni colonna (ad esclusione di $b=0$)

Esponenziazione modulare

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

$$a = (b \times b \times \dots \times b) \bmod m$$

e volte

$$a = [(b \bmod m) \times (b \bmod m) \times \dots \times (b \bmod m)] \bmod m$$

e volte

$$b, e \in \mathbb{Z}_m$$

Caso peggiore $e=m-1$

m moltiplicazioni $\rightarrow 2^{\log m}$ moltiplicazioni

$O(\exp(\log m))$!!

Algoritmi più efficienti

Generatori modulo p

T6: “per ogni primo p , esiste un $g < p$, detto **radice primitiva** di p , o anche generatore modulo p , le cui potenze

$$g^1 \bmod p, g^2 \bmod p, \dots, g^{p-1} \bmod p$$

forniscono tutti gli interi compresi tra 1 e $p-1$ ”

	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

**permutazioni
di Z_{11}^***

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

T6bis: “le radici primitive di p sono $\Phi(p-1)$. $\Phi(10) = (2-1)(5-1) = 4$

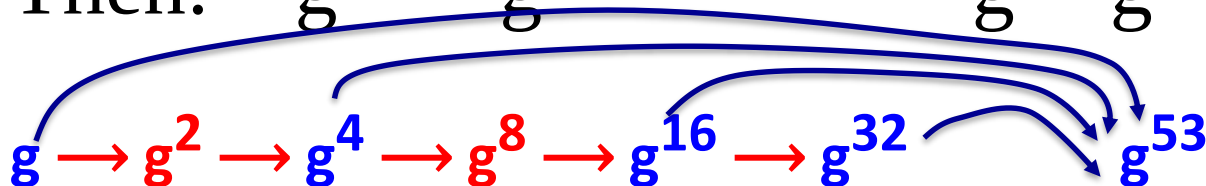
Exponentiation

Finite cyclic group G (for example $G = \mathbb{Z}_p^*$)

Goal: given g in G and x compute g^x

Example: suppose $x = 53 = (110101)_2 = 32+16+4+1$

$$\text{Then: } g^{53} = g^{32+16+4+1} = g^{32} \cdot g^{16} \cdot g^4 \cdot g^1$$



9 moltiplicazioni!!!!

A0: un primo algoritmo di esponenziazione

$b^e \bmod m$

$b, e \in \mathbb{Z}_m$

m primo o composto

Sia $t = \lceil \log_2 m \rceil$

La rappresentazione in base 2 di e è:

$$e = e_{t-1} 2^{t-1} + e_{t-2} 2^{t-2} + \dots + e_1 2^1 + e_0 2^0$$

Di conseguenza si ha:

$$b^e = b^{e_{t-1} 2^{t-1}} \times b^{e_{t-2} 2^{t-2}} \times \dots \times b^{e_1 2^1} \times b^{e_0 2^0}$$

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 m , trattenendo il resto

Caso peggiore: circa $2 \cdot t$ moltiplicazioni (..ma su numeri sempre più grandi!)

The repeated squaring alg.

Input: g in G and $x > 0$; **Output:** g^x

write $x = (x_n x_{n-1} \dots x_2 x_1 x_0)_2$

$y \leftarrow g$, $z \leftarrow 1$

for $i = 0$ to n do:

if $(x[i] == 1)$: $z = z \cdot y$

$y \leftarrow y^2$

output z

example: g^{53}

<u>y</u>	<u>z</u>
g^2	g
g^4	g
g^8	g^5
g^{16}	g^5
g^{32}	g^{21}
g^{64}	g^{53}

Repeated square and multiply: A1

Riducendo in modulo il risultato di ogni moltiplicazione si riesce ad operare sempre e solo su numeri inferiori a m

Non separando il calcolo dei b^i da quello di $\prod (b^i)^{e_i}$ (calcolo elevamenti a quadrato dalle moltiplicazioni) si ottengono tempi di esecuzione più brevi

INPUT: $b \in \mathbb{Z}_m$, $0 \leq e < m$, $e = (e_{t-1} \dots e_0)_2$

OUTPUT: $b^e \bmod m$

1. Set $y \leftarrow 1$. If $e = 0$ then return (y)
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 $A \leftarrow A^2 \bmod m$
 - 4.2 If $e_i = 1$ then set $y \leftarrow A y \bmod m$
5. Return (y)

$m = 11$, $b = 7$, $e = 5 = (101)_2$

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. $7^5 \bmod 11 = 10$

Repeated square and multiply: A2

INPUT: $b \in \mathbb{Z}_m$, $0 \leq e < m$, $e = (e_{t-1} \dots e_0)_2$

OUTPUT: $b^e \bmod m$

1. $y \leftarrow 1$
2. For i from $t-1$ down to 1 do the following
 - 2.1 $y \leftarrow y^2 \bmod m$
 - 2.2 If $e_i = 1$ then $y \leftarrow y \cdot b \bmod m$
3. Return (y)

1. $y = 1$

2. $i = 2$

2.1 $y = 1 \times 1 \bmod 11 = 1$

2.2 $y = 1 \times 7 \bmod 11 = 7$

$i = 1$

2.1 $y = 7 \times 7 \bmod 11 = 5$

$i = 0$

2.1 $y = 5 \times 5 \bmod 11 = 3$

2.2 $y = 3 \times 7 \bmod 11 = 10$

5. $7^5 \bmod 11 = 10$

Proprietà notevole di A1 e di A2:

Sia **t** il numero di bit della rappresentazione binaria dell'esponente **e**

Sia **n** il numero di bit dell'esponente con valore **1**

TEMPO DI ESECUZIONE $\rightarrow$ **t+n** moltiplicazioni

TEMPO MEDIO $\rightarrow$ **3/2 t** moltiplicazioni

Complessità

TEMPO DI ESECUZIONE → **t+n** moltiplicazioni

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

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

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

Robustezza e tempo d'esecuzione

Modulo: da 1024 bit a 2048 bit

Tempo: da 1 a 8

Nota Bene

Se $b=2$, i primi elevamenti al quadrato e moltiplicazioni si fanno con shift.

Se $e = 11$ oppure 100000000000000001 , le moltiplicazioni sono solo 2

Logaritmo discreto

T8: “Dati un primo p , un generatore g ed un qualunque intero $y \in \mathbb{Z}_p^*$, 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

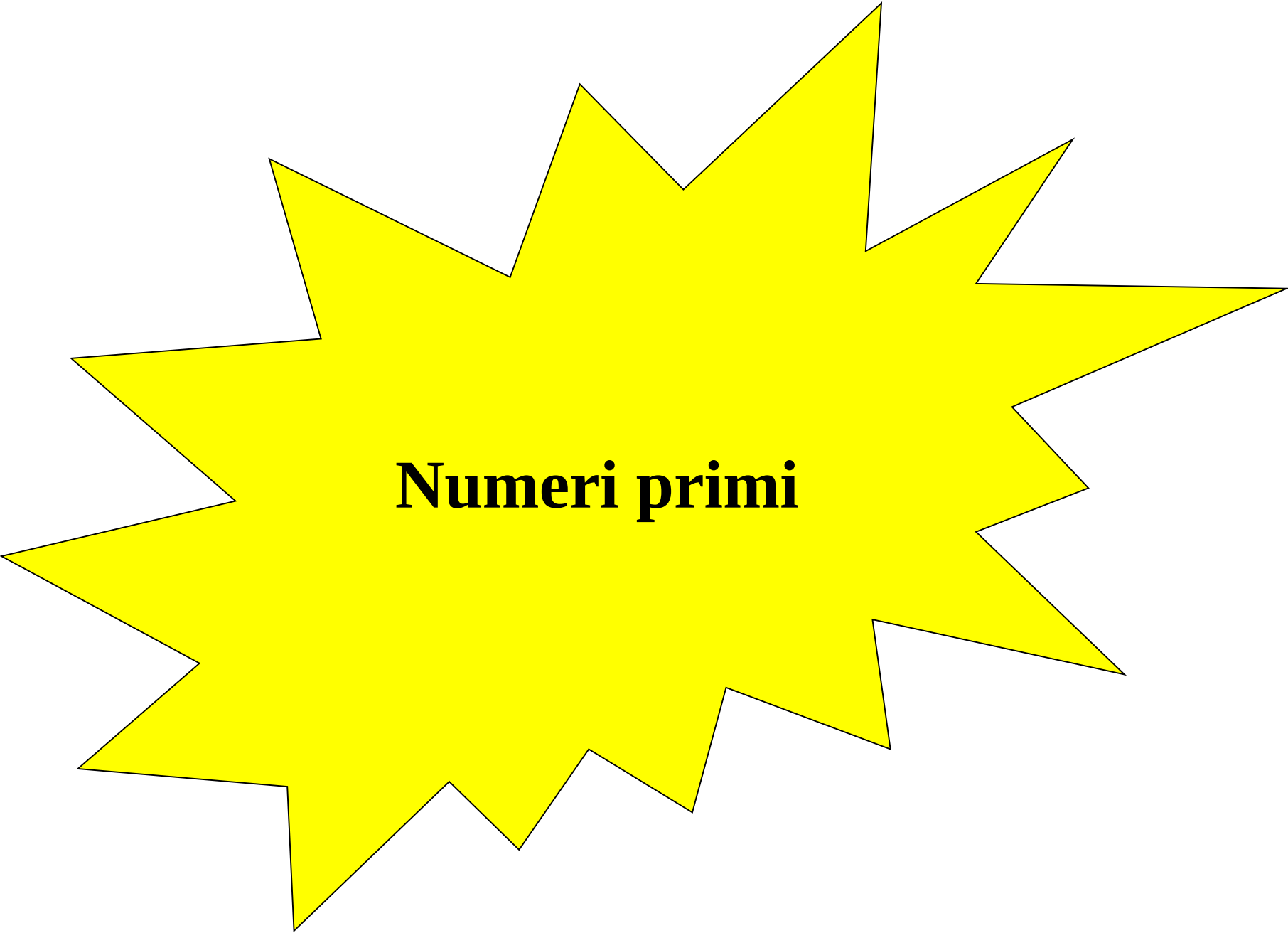
$$g = 2 \rightarrow x = 8$$

$$g = 6 \rightarrow x = 2$$

$$g = 7 \rightarrow x = 4$$

$$g = 8 \rightarrow x = 6$$

$$y = 3$$

A large, bright yellow starburst shape with multiple points, outlined in black, centered on a white background. The text "Numeri primi" is written in the center of the starburst.

Numeri primi

Distribuzione dei numeri primi

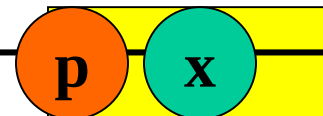
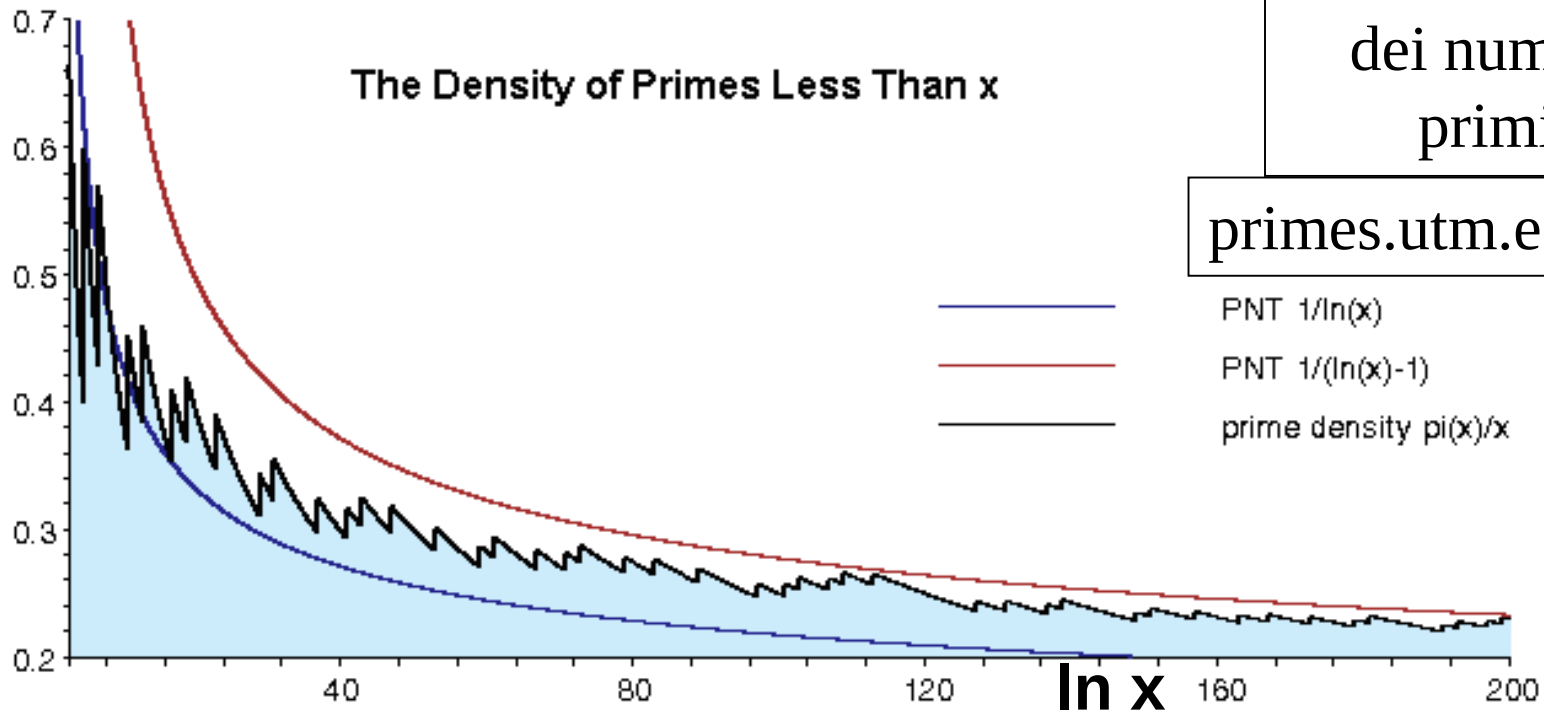
$\pi(x)$: n° di primi
nell'intervallo $2 \div x$

T9:

$$\lim_{x \rightarrow \infty} \frac{\pi(x)}{x / \ln x} = 1$$

Fornisce
indicazione
sulla
distribuzione
dei numeri
primi

primes.utm.edu/



Dato x grande è probabile trovare un primo p in un intorno di ampiezza $\ln x$

Ricerca di numeri primi grandi

1- Ricerca di un numero primo

1. x dispari, generato a caso nel desiderato intervallo
2. If $(x \text{ primo?}) = \text{false}$ —————
 then repeat 1.
3. Return x

Test di primalità

2- Ricerca di un numero primo

1. x dispari, generato a caso nel desiderato intervallo
2. If $(x \text{ primo?}) = \text{false}$ —————
 then $x = x + 2$ and repeat 2.
3. Return x

Test di primalità

Test di primalità

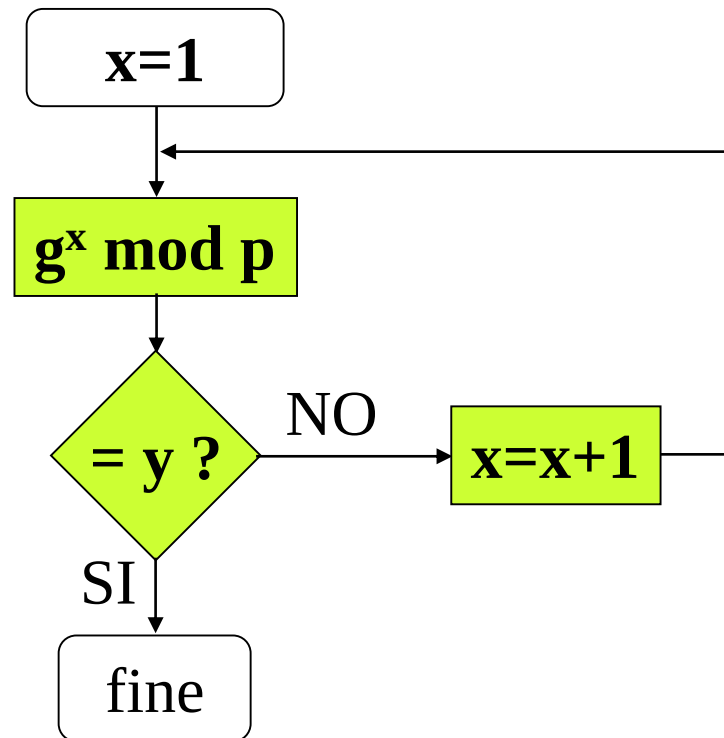
1. Test **deterministici**: se n non lo supera è **composto**,
se lo supera è **primo**
2. Test **probabilistici**: se n fallisce il test è **composto**;
se lo supera è **probabilmente primo**

I test deterministici sono computazionalmente più onerosi dei test probabilistici.

I test probabilistici (Miller Rabin) sono polinomiali, ma devono essere poi ripetuti più e più volte per far tendere a 1 la probabilità di avere realmente individuato un primo.

Ricerca esauriente

Si calcola $g^x \bmod p$ per $x=1,2,\dots$ fino a quando non si trova y



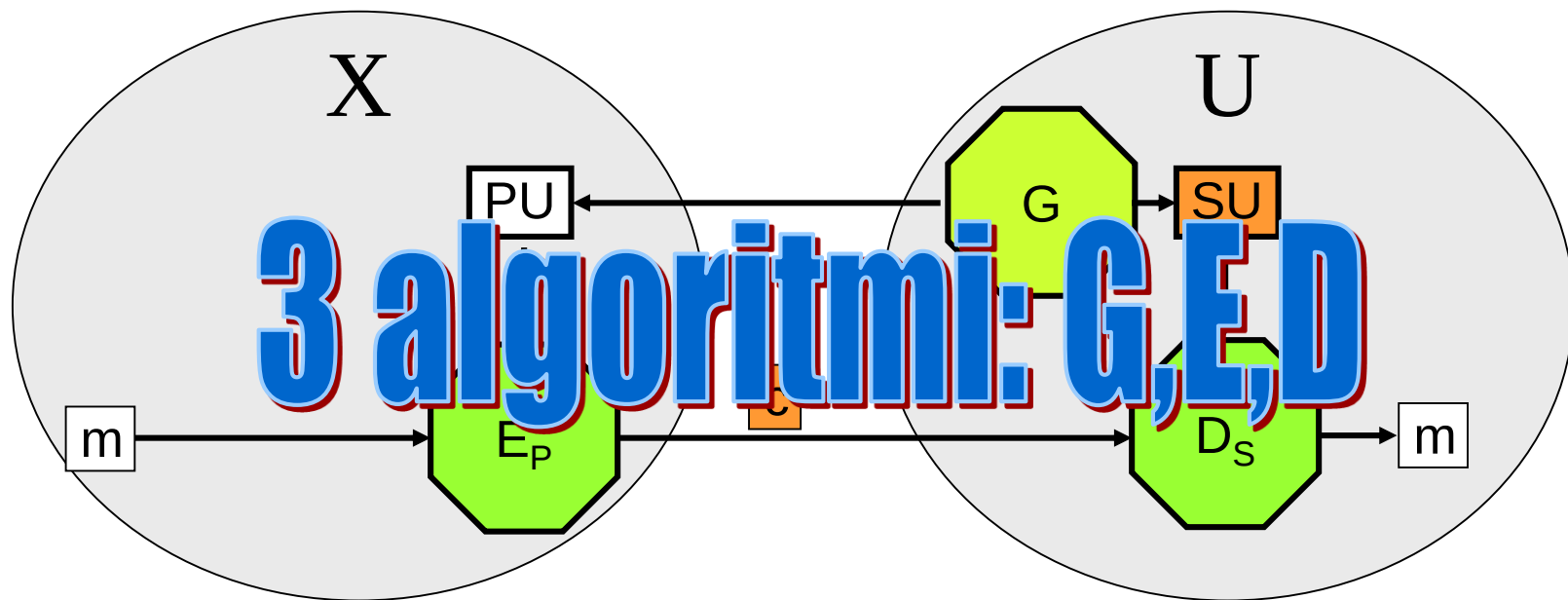
$$n = \lceil \log_2 p \rceil$$
$$T = O(\exp(n)).$$

Altri algoritmi più pericolosi: passo del bambino-passo del gigante

A large, bright yellow starburst shape with multiple sharp points, centered on the page. It has a black outline and a solid yellow fill.

Cifrari asimmetrici

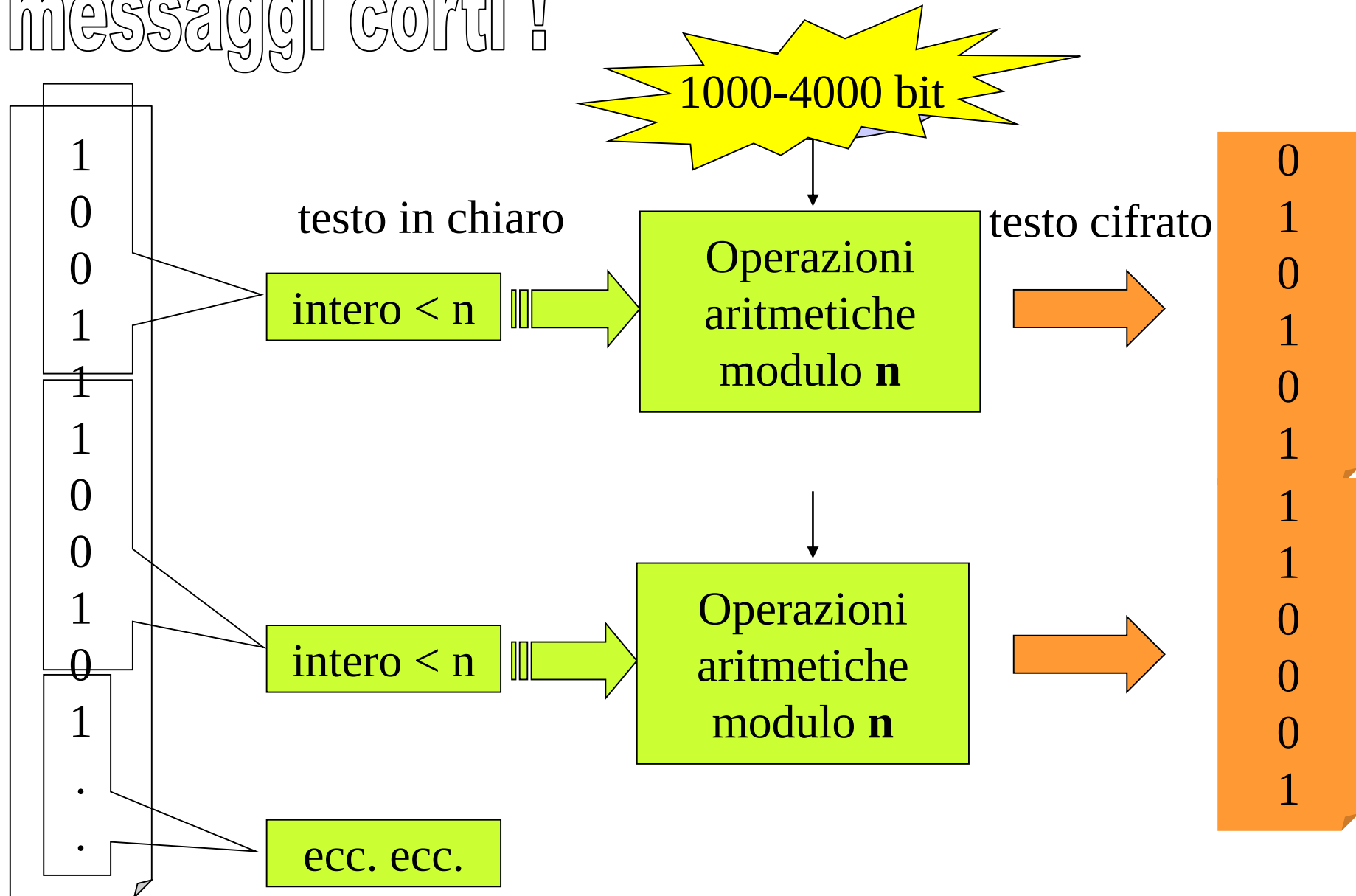
Aspetti caratteristici



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

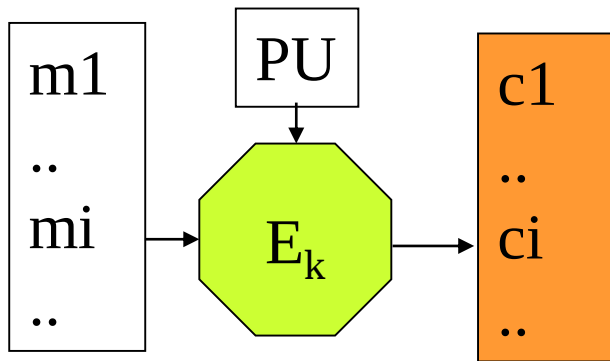
1 - Frammentazione del testo in chiaro

messaggi corti !



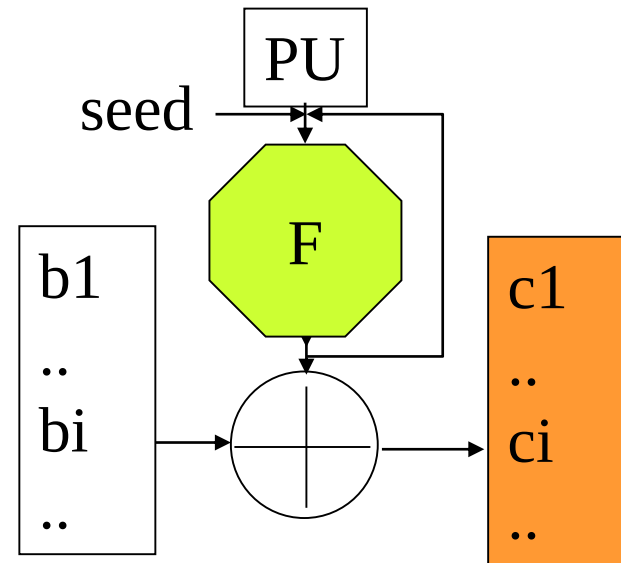
Cifratura a blocchi ed a flusso

Cifratura a blocchi



$$c_i = E_{PU}(m_i)$$

Cifratura a flusso



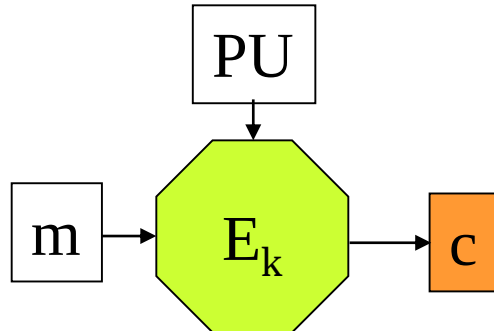
$$\begin{aligned} c_i &= b_i \oplus k(i) \\ k(i+1) &= F(k(i), PU) \\ k(0) &= \text{seed} \& PU \end{aligned}$$

I bit di chiave sono ottenuti con moltiplicazioni ed esponenziazioni => molto lenti

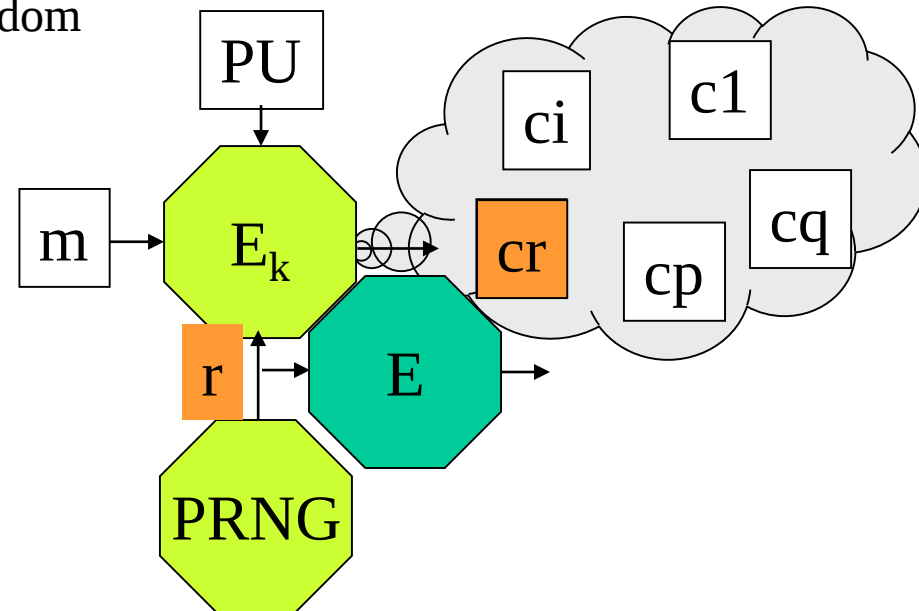
Cifrari deterministici e Cifrari probabilistici

a parità di chiave il testo cifrato è scelto di volta in volta tra molti possibili in base a un numero random

fissati m e chiave il testo cifrato è unico



$$c = E_{PU}(m)$$

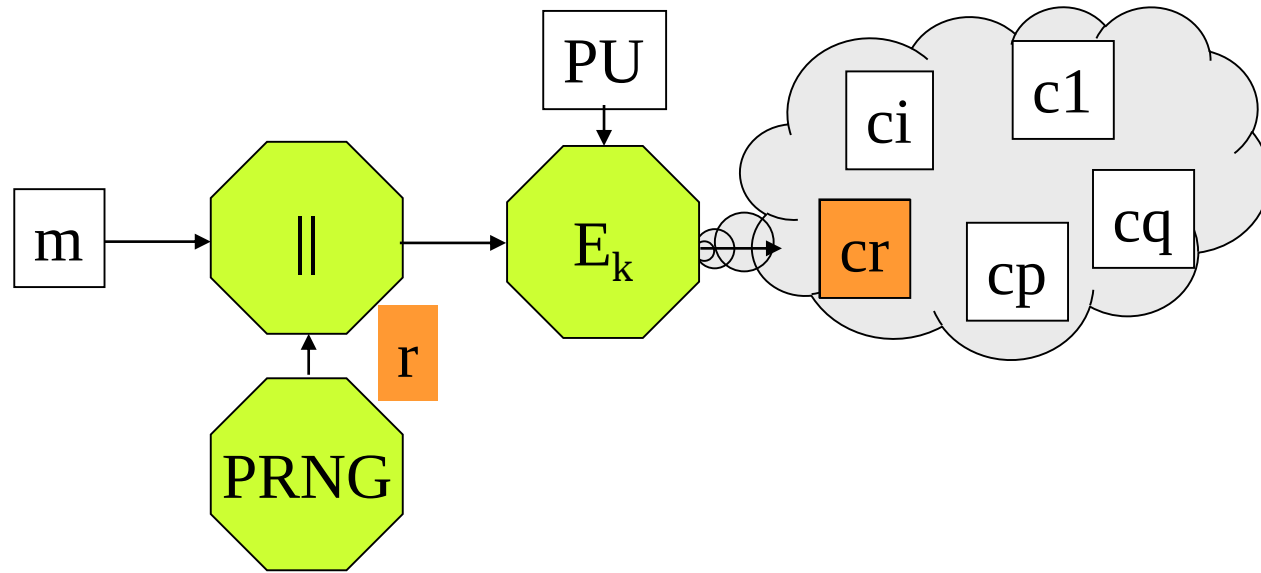


$$c = E_{PU}(m, r) || E(r)$$

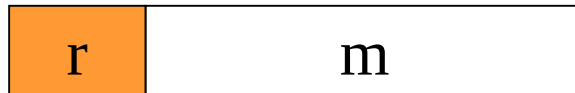
I Cifrari deterministici sono più efficienti, ma ..

- 1: blocchi in chiaro identici producono blocchi cifrati identici
- 2: sono vulnerabili ad attacchi con testo in chiaro noto (es. sì/no in voto elettronico)

Randomizzazione di un Cifrario deterministico



padding



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

Oppure CBC e IV

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



interoperabilità

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.



Il Cifrario RSA

The RSA Algorithm (1978)

Public key: $\{n, e\}$

$n = p \times q$ con p e q primi

e coprimo con $\Phi(n) = (p-1)(q-1)$

Encryption

- Plaintext: $m < n$
con m con dimensione di k bit dove $2^k < n \leq 2^{k+1}$
- 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 (1978)

Public key: $\{n, e\}$

$n = p \times q$ con p e q primi

e coprimo con $\Phi(n) = (p-1)(q-1)$

Private key: $\{n, d\}$

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

d coprimo con $\Phi(n)$

Decryption

- Ciphertext:

$$c < n$$

- Plaintext:

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

anzichè

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

dato che

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

inverso moltiplicativo

si dimostra formalmente che $c^d \bmod n \equiv m^{ed} \pmod n$
 $\equiv m$ se $m < n$ ed e coprimo con $\Phi(n)$ (ad ogni testo
cifrato deve corrispondere un solo testo in chiaro)

L'algoritmo G: RSA Key Pair Generation

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

Tutti gli algoritmi in gioco sono polinomiali

Il calcolo di e

$$\gcd(\Phi(n), e) = 1; 1 < e < \Phi(n)$$

L'algoritmo esteso di Euclide permette di calcolare il gcd di due interi e se $\gcd(a, b) = 1$ l'inverso di uno dei due modulo l'altro. Quindi posso calcolare tra e e $\Phi(n)$ qual è il loro gcd e se $\gcd(e, \Phi(n)) = 1$ trovare poi d

Ora il problema è trovare e. Si può per il calcolo di e scegliere a caso un $r < \Phi(n)$ e poi verificare se r coprimo con $\Phi(n)$. In caso contrario si itera il procedimento. Di solito bastano poche prove

Il calcolo di d: algoritmo esteso di Euclide

$$\text{MCD}(a,b) = c = x a + y b$$

a, b interi positivi; $a \geq b > 0$

Poni $x_2 = 1$, $x_1 = 0$, $y_2 = 0$, $y_1 = 1$

Finché $b > 0$

- calcola $q = \lfloor a/b \rfloor$, $r = a - q \cdot b$
- calcola $x = x_2 - q \cdot x_1$, $y = y_2 - q \cdot y_1$
- poni $a = b$, $b = r$,

$$x_2 = x_1, x_1 = x, y_2 = y_1, y_1 = y$$

Restituisci $c = a$; $x = x_2$; $y = y_2$

$$O((\ln n)^2)$$

Permette di trovare oltre che $\text{MCD}(a,b)$. Se poi $\text{MCD}(a,b)=1$ permette di trovare anche l'inverso moltiplicativo di b

Generazione della chiave

- $n=p \times q$ NOTO $\Rightarrow$ per impedire di trovare p e q con metodi esaustivi p e q scelti in un insieme di sufficientemente esteso (p e q numeri molto estesi)

Anche il metodo per trovare numeri primi di grandi dimensioni deve essere ragionevolmente efficiente



Non esistono ora tecniche utili per fornire numeri primi arbitrariamente estesi. La procedura normalmente scelta è: si sceglie casualmente un numero dispari dell'ordine di grandezza desiderato e si verifica se tale numero è primo tramite test probabilistici (test ad es. Miller Rabin)

Generazione della chiave

- Si sceglie casualmente un intero dispari (con PRNG)
- Si sceglie casualmente $a < n$
- Si esegue un test di primalità con il parametro a . Se n fallisce il test rifiutare il valore e tornare al passo 1
- Se n passa un numero sufficiente di test accettare n altrimenti tornare al passo 2

Quanti numeri rifiutati? Circa $\ln(n)$ ma visto che i numeri pari vanno scartati solo circa $\ln(n)/2$

$$N = 2^{200}$$

Circa 70 tentativi

Generazione della chiave

- Scelti p e q si scelgono e e d . Supponiamo e : con il metodo di euclide esteso si sceglie e coprimo con $\Phi(n)$ e si calcola l'inverso di uno degli interi modulo l'altro. Si generano numeri casuali e si confrontano con $\Phi(n)$, fino a trovare un coprimo (pochi test)

Aspetti Computazionali

- Generazione della chiave
- Cifratura/decifrazione

Per rendere efficiente:

- la cifratura ($x^e \bmod n$) si adottano gli algoritmi “repeated square and multiply” con scelta opportuna di e ; $x < n$
- la decifrazione si ricorre al teorema cinese dei resti (CTR) (impiego efficiente della chiave privata)

La scelta di e per cifrare (impiego efficiente della chiave pubblica)

N.B. Se il numero binario e contiene pochi “uni”
il calcolo di m^e è più efficiente !

$$e = 3$$

$$e = 2^{16} - 1$$

$$e = 2^{32} - 1$$

Se viene scelto $e=65537$ e poi si generano p e q potrebbe accadere
che $\text{MCD}(\Phi(n), e)$ sia diverso da 1, quindi vanno rifiutati perché e
deve essere coprimo con $\Phi(n)$

n	e
-----	-----

n	e
-----	-----

T14 (Teorema cinese dei resti): “Se gli interi $n_1, n_2, \dots, n_k$ sono a due a due coprimi, allora il sistema di congruenze

$$\begin{aligned}x &\equiv a_1 \pmod{n_1}, \\x &\equiv a_2 \pmod{n_2}, \\&\dots, \\x &\equiv a_k \pmod{n_k}\end{aligned}$$

ha un'unica soluzione modulo $n = n_1 \times n_2 \times \dots \times n_k$ ”.

Conseguenze (nel caso $n_1 = p$, $n_2 = q$ con p, q primi):

- Ogni intero $x < n = pq$ ha un'unica **rappresentazione modulare**
$$v(x) = (v_1, v_2) = (x \bmod p, x \bmod q)$$
- Somme e moltiplicazioni modulari possono essere fatte vantaggiosamente sulla rappresentazione modulare degli operandi
- Esistono algoritmi facili (Gauss, Garner) per ripristinare la rappresentazione originaria dell'intero

Decifrazione con CRT

1. *Rappresentazione modulare di c :*

$$\mathbf{v(c) = (c \bmod p, c \bmod q)}$$

2. *Calcolo della rappresentazione modulare di $c^d \bmod n$:*

$$\mathbf{v^d(c) = ((c \bmod p)^d \bmod p, (c \bmod q)^d \bmod q)}$$

N.B: numeri di dimensione più piccola ($\frac{1}{2} \log n$)

3. *Ripristino della rappresentazione usuale di m:*

$$\mathbf{m = c^d \bmod n}$$

$$\mathbf{= \{a \times [(c \bmod p)^d \bmod p] + b \times [(c \bmod q)^d \bmod q]\} \bmod n}$$

Per **T14** **a**, **b** devono soddisfare le congruenze

$$\mathbf{a \equiv 1 \pmod{p}}$$

$$\mathbf{b \equiv 0 \pmod{p}}$$

$$\mathbf{a \equiv 0 \pmod{q}}$$

$$\mathbf{b \equiv 1 \pmod{q}}$$

Circa quattro volte più veloce

Ancora più efficiente se n è il prodotto di più di due primi (PKCS#1)

Sicurezza di RSA

- Forza bruta
- Attacchi matematici
- Attacchi a tempo
- Attacchi a testo cifrato scelto

Attacchi Matematici

- Fattorizzazione di n nei suoi due fattori primi $\Rightarrow$ questo permette di calcolare la funzione totiente di Eulero $(p-1)(q-1)$ e quindi d
- Determinare direttamente la funzione totiente di Eulero $(p-1) \times (q-1)$ senza prima determinare p e q
- Determinare direttamente d senza prima determinare la funzione toziente di Eulero $(p-1) \times (q-1)$

caso più frequente è la fattorizzazione

Algoritmi di Fattorizzazione

Trial division:

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

I° fattore $< \text{sqrt}(n)$
 $O(\exp(1/2 (\log n)))$

.....

General Number Field Sieve:

$O(\exp(\log n)^{1/3} \cdot (\log(\log n))^{2/3})$

Per ridurre la possibilità di attacco si possono anche imporre vincoli su p e q :

“ p, q *strong primes* : $p-1, p+1, q-1, q+1$ con grandi fattori primi

“ $|p-q| > n^{1/2}$ ”

1024 – 2048 bit

Attacchi a Tempo

- Si basano unicamente sul testo cifrato
- Si è dimostrato che si può determinare una chiave privata analizzando il tempo impiegato dai computer per decifrare i messaggi
- Sono analoghi a un ladro che cerca di indovinare la combinazione di sicurezza di una cassaforte osservando il tempo impiegato per ruotare la manopola

Contromisure: ad es. tecnica di blinding



- Si genera numero casuale segreto r compreso tra 0 e $n-1$
- Si calcola $c' = c(r^e)$
- Si calcola $m' = (c')^d \bmod n$
- Si calcola $m = m' r^{-1} \bmod n$ con r^{-1} inverso moltiplicativo di r modulo n

Attacchi a Testo Cifrato Scelto

Contromisura: padding OAEP

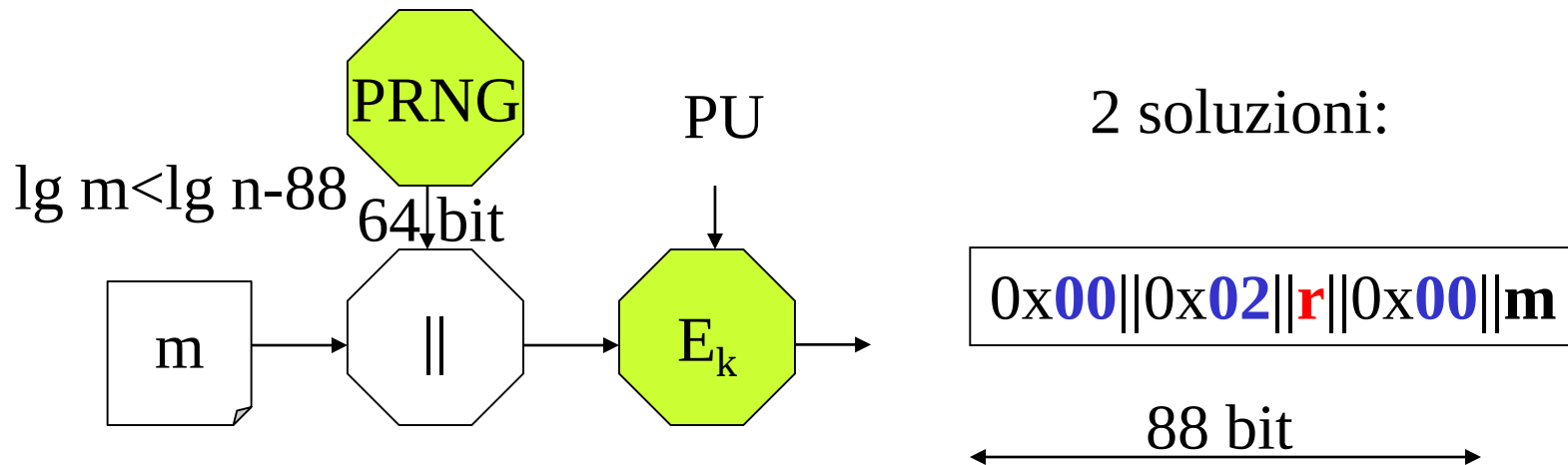
Randomizzazione di RSA

RSA esegue una sostituzione semplice di blocchi ed è deterministico:

1- **m** identici generano **c** identici

2 - per certi **m** si ha **c = m**

Standard PKCS#1v2



2 soluzioni:

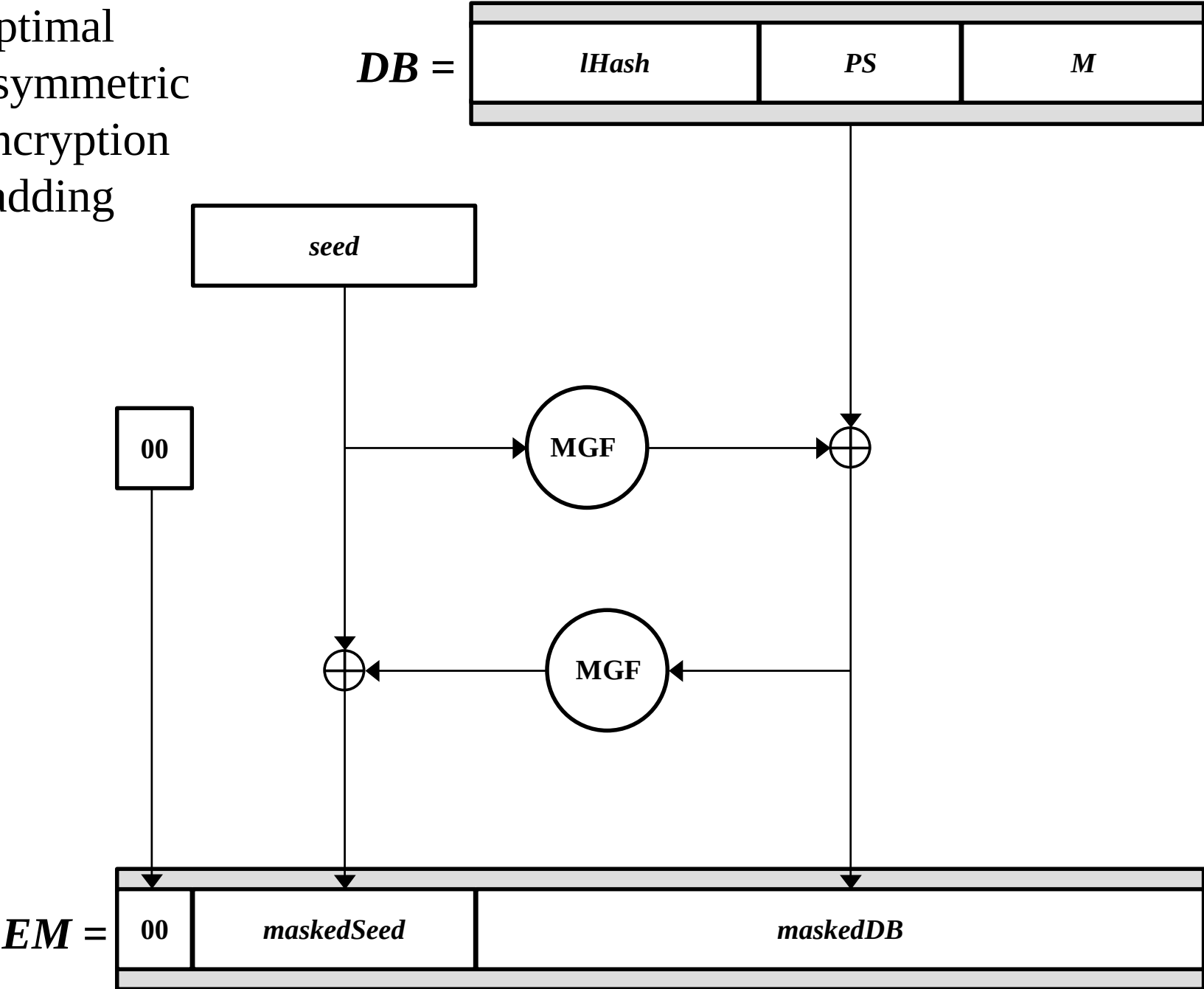
OAEP: Optimal Asymmetric Encryption Padding

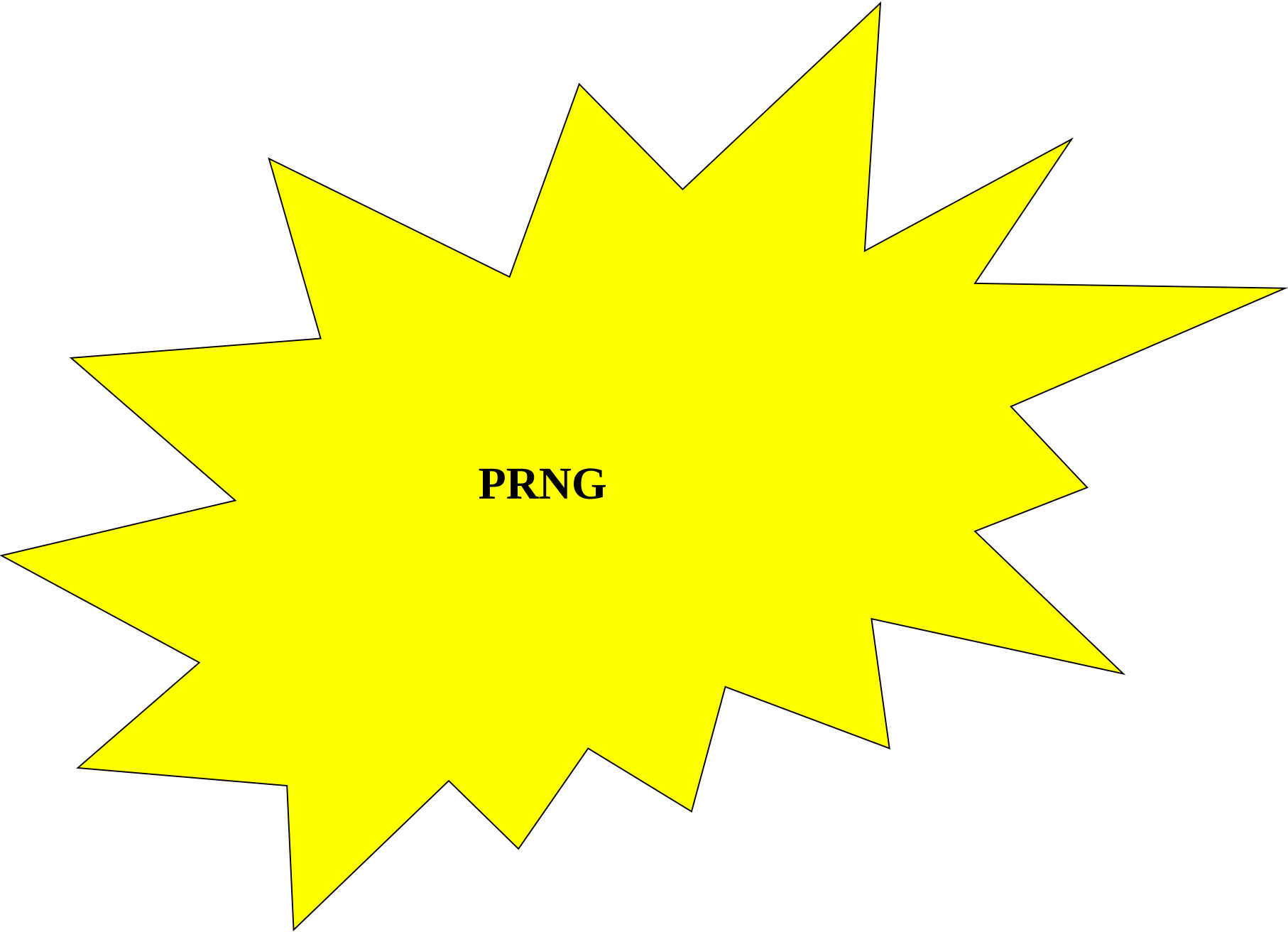
The resulting padded ciphertext structure for OAEP is shown as:

$0x00 || \text{maskedSeed} || \text{maskedDB}$

The total length of the padded ciphertext is n bit.

Optimal
Asymmetric
Encryption
Padding

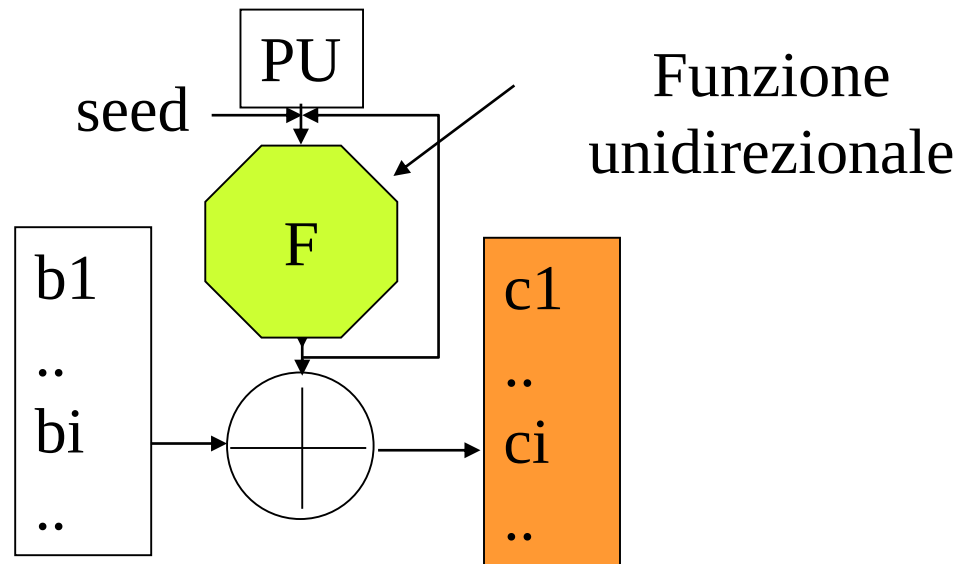


A bright yellow, multi-pointed starburst shape with a black outline. The shape is centered on the page. Inside the starburst, the text "PRNG" is written in a bold, black, serif font.

PRNG

RSA Pseudorandom Bit Generator

Cifratura a flusso



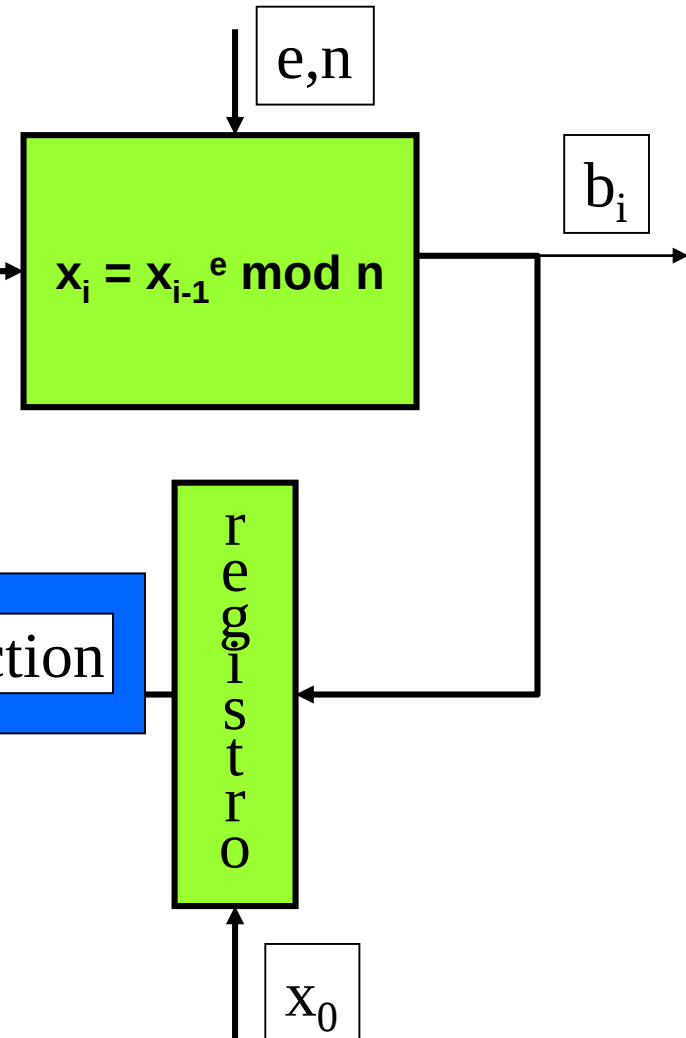
$$\begin{aligned} c_i &= b_i \oplus k(i) \\ k(i+1) &= F(k(i), PU) \\ k(0) &= \text{seed} \& PU \end{aligned}$$

RSA Pseudorandom Bit Generator

Spesso $e = 3$
N.B. $a^3 = (a)^2 \times a$

1. Si sceglie a caso due grandi primi p e q
2. Si calcola
 - 2.1 $n = pq$
 - 2.2 $\Phi = (p-1)(q-1)$
 - 2.3 e tale che $\text{MCD}(e, \Phi) = 1$
3. Si sceglie a caso un seme $x_0 < n-1$
(N.B. Intel generator consigliato)
4. Si itera quanto serve
 - 4.1 $x_i = x_{i-1}^e \bmod n$,
 - 4.2 $b_i = x_i \bmod 2$(N.B. b_i è il bit meno significativo di $(x_i)_2$)

One-way function



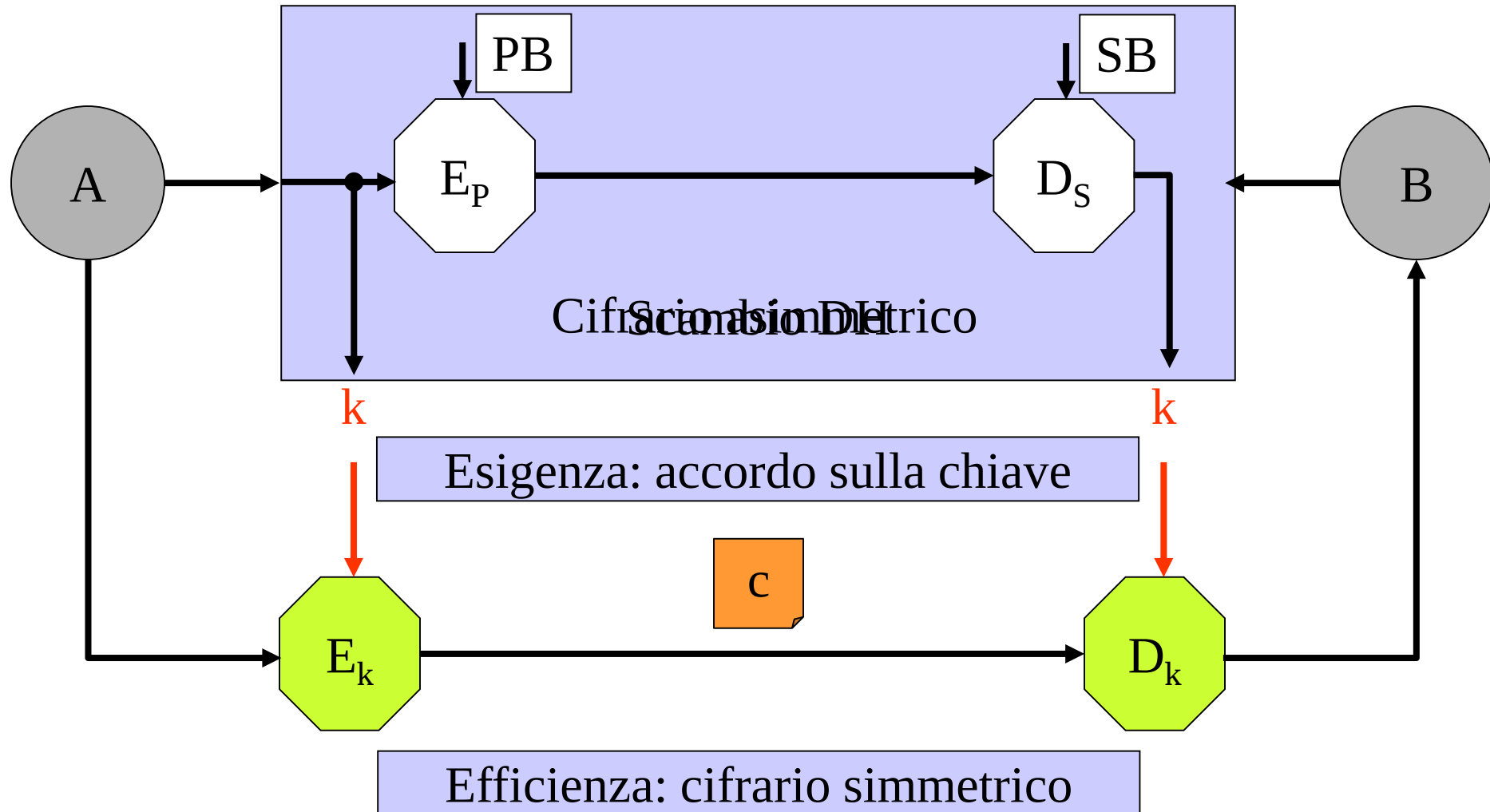
imprevedibilità dimostrabile
assumendo che la fattorizzazione
sia un problema difficile

A large, bright yellow starburst shape with multiple sharp points, centered on the page. It has a thin black outline.

Il Cifrario ibrido

KA con Cifrario asimmetrico

Problema: A e B vogliono scambiarsi informazioni riservate in assenza di accordi precedenti



Il cifrario ibrido

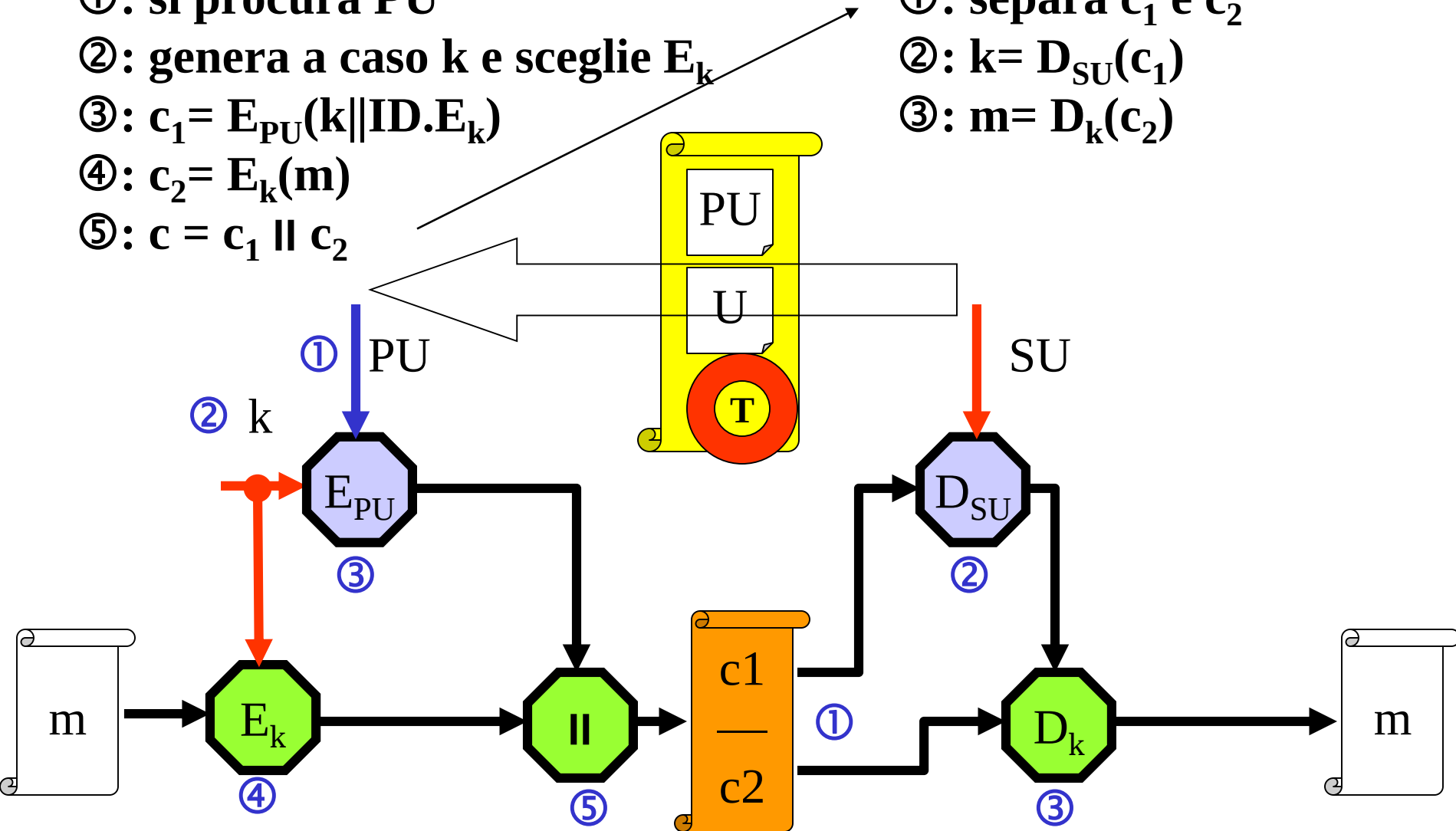
(asimmetrico e simmetrico)

Sorgente X

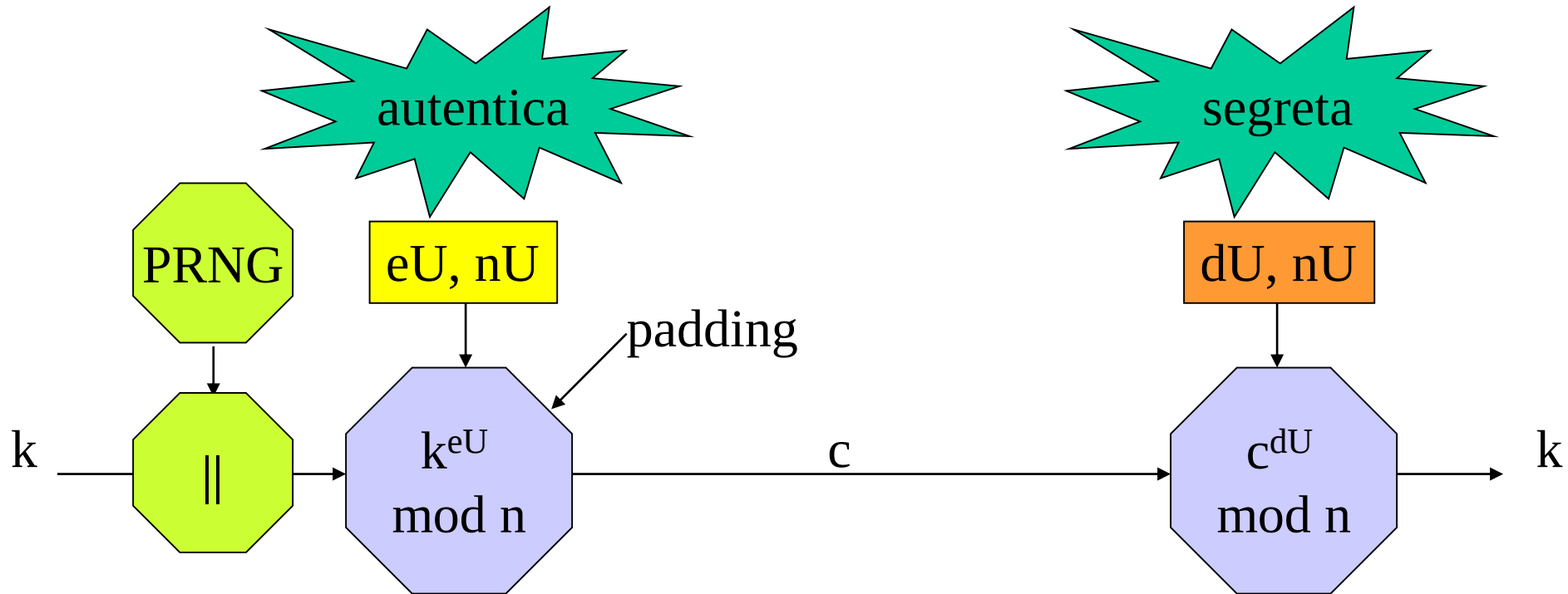
- ①: si procura PU
- ②: genera a caso k e sceglie E_k
- ③: $c_1 = E_{PU}(k || ID.E_k)$
- ④: $c_2 = E_k(m)$
- ⑤: $c = c_1 || c_2$

Destinazione U

- ①: separa c_1 e c_2
- ②: $k = D_{SU}(c_1)$
- ③: $m = D_k(c_2)$



Chiave di sessione con RSA



Vulnerabilità: $k^{eU} < n$ con k ed e numeri piccoli

Contromisura PKCS#1v2

- 1: padding con r di 64 bit
- 2: padding OAEP

A large, bright yellow starburst shape with multiple sharp points, centered on the page. The text "Firma digitale" is written in a bold, black, serif font in the center of the starburst.

Firma digitale

Firma digitale

La firma digitale di un documento informatico deve:

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

Crittografia asimmetrica

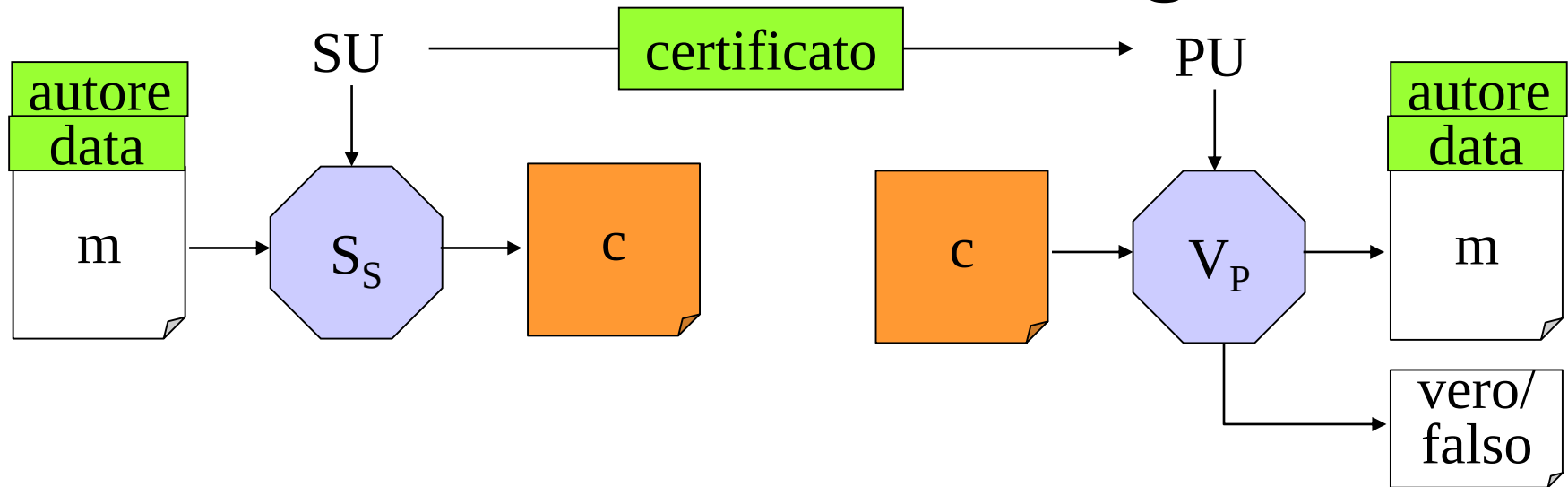
2- non poter essere **imitata** da un impostore,

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.

Sicurezza della firma digitale



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

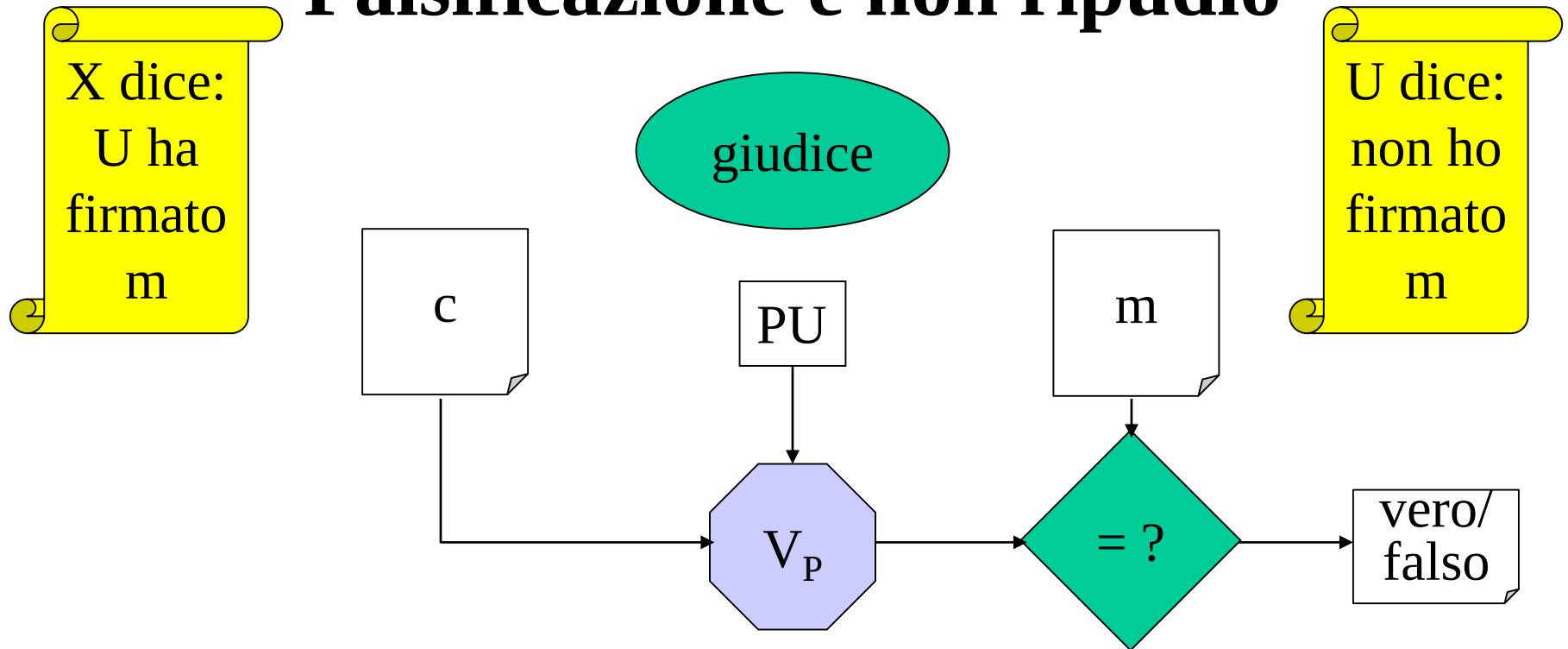
$$V_{PU}(c) = m, \text{ vero}$$

dimostrazione
d'esistenza

dimostrazione
sul campo

messaggi
con formato

Falsificazione e non ripudio



Valore legale della firma digitale

1989: USA e poi ONU

1997: ITALIA

1999: Comunità Europea

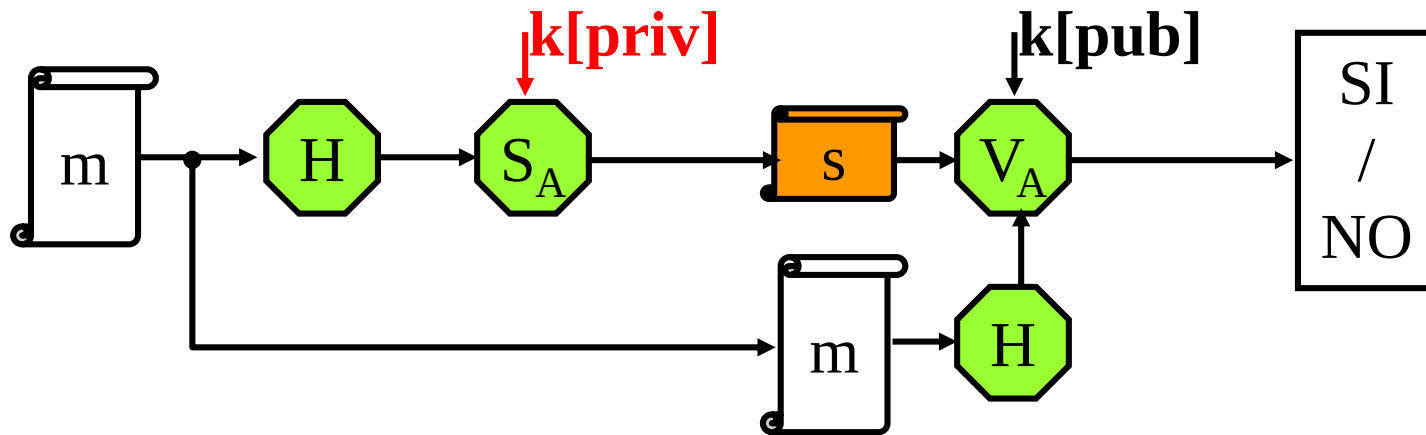
2002: ITALIA

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

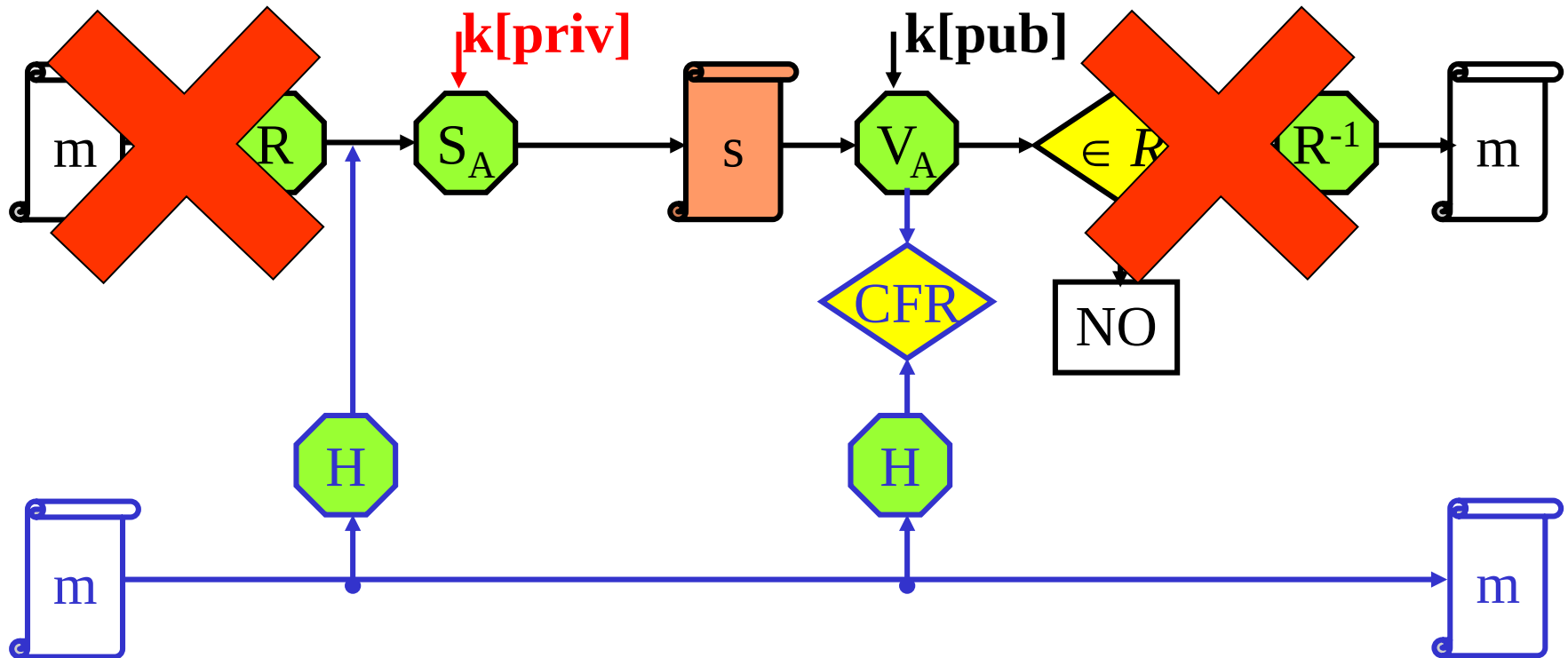
Algoritmi di firma con appendice

Messaggi di lunghezza arbitraria
Appendice: $S(H(m), k[\text{priv}A])$



Algoritmi di firma con recupero

Messaggi “corti”: $< \text{modulo}$
Funzione di ridondanza R



N.B. - Gli algoritmi con recupero possono essere impiegati anche in uno **schema con appendice** (l'hash ha sicuramente una dimensione inferiore a quella del modulo)

Proprietà di reversibilità di RSA

Reversibilità delle chiavi (impiego di SU al posto di PU e viceversa)

$$E_{SU}(m) = c = m^{SU} \bmod n$$

$$D_{PU}(c) = (m^{SU})^{PU} \bmod n = m$$

messaggio non riservato

ma con origine verificabile

Schema di firma con recupero

inefficiente se $m > n$ e insicura

$\lceil \log_2 n \rceil$ bit di etichetta

Schema di firma con appendice:

1. FIRMA

$$S_{SU}(H(m)) = (H(m))^{SU} \bmod n$$

$$m \parallel S_{SU}(H(m))$$

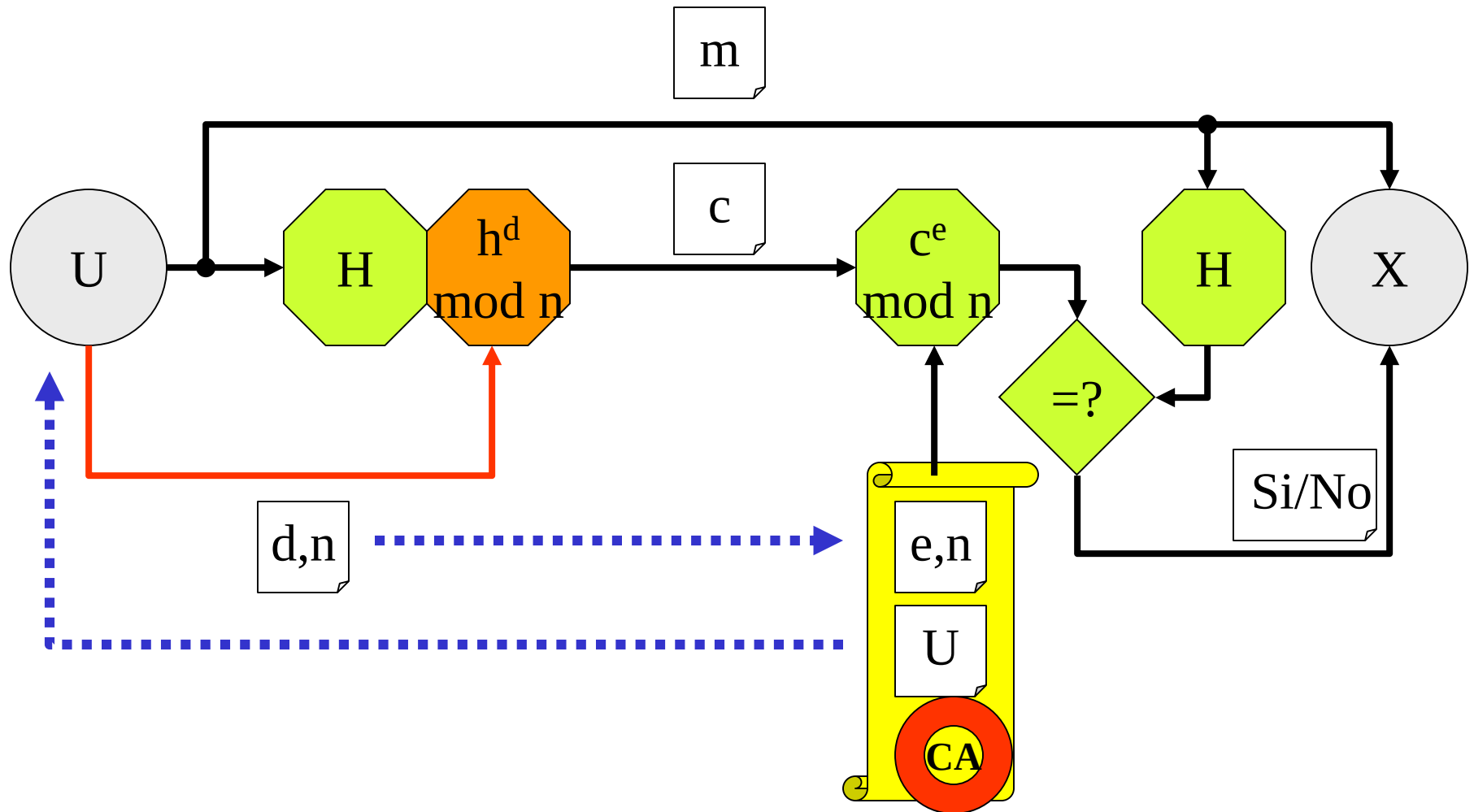
ossia un bit in meno rispetto a n

**autenticazione del messaggio
comunicazione del messaggio**

2: VERIFICA

- calcolo di $H(m')$
- calcolo di $H(m) = (S_{SU}(H(m)))^{PU} \bmod n$
- confronto

Firma con RSA



Padding

deterministico

PKCS#1: $0x00\ 0x01\ 0xFF\ \dots 0xFF\ 0x00\ || H(m)$

FDH: $H(c_0.m) || H(c_1.m) || H(c_2.m) || \dots$

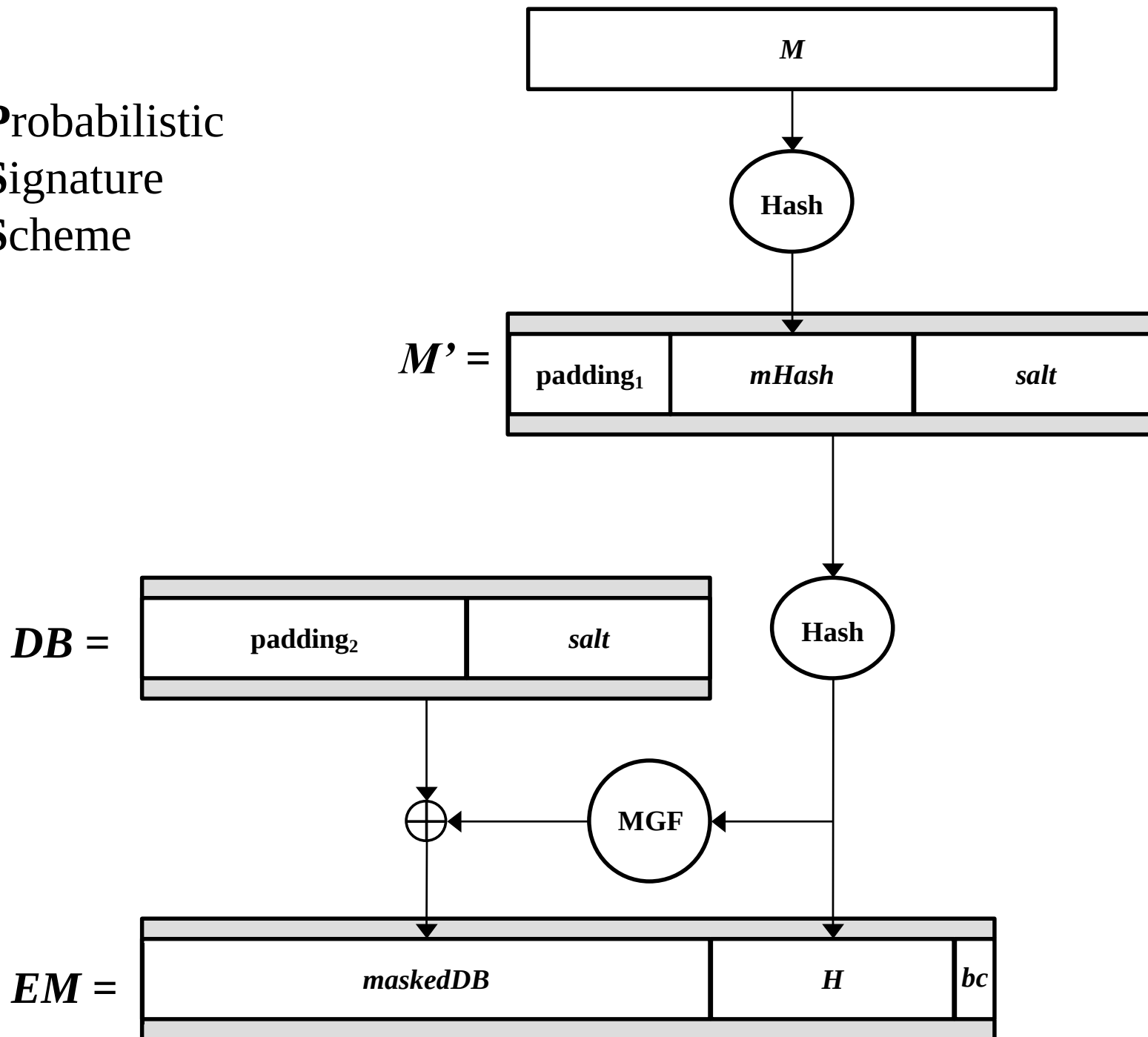
Funzione hash ideale che
genera impronte di $\lceil \log_2 n \rceil$

probabilistico

PKCS#1v2: PPS (Probabilistic Signature Scheme)

“Practical cryptography”

Probabilistic
Signature
Scheme



Proprietà moltiplicativa di RSA

Sia $m = m_1 \times m_2 < n$

Firma di m da parte di U :

$$\begin{aligned} c &= m^{d_U} \bmod n = (m_1 \times m_2)^{d_U} \bmod n \\ &= ((m_1^{d_U} \bmod n) \times (m_2^{d_U} \bmod n)) \bmod n \end{aligned}$$

Proprietà moltiplicativa dell'algoritmo RSA:

*il testo cifrato (con chiave pubblica o con chiave privata)
del prodotto di due testi in chiaro
è congruo (mod n) al prodotto dei due testi cifrati*

Autenticazione di un messaggio oscurato

X vuole farsi autenticare da T un messaggio m senza che T possa conoscerne il contenuto

Autorizzazioni per voto elettronico, commercio elettronico, ecc.

Autenticazione “a occhi chiusi” di un messaggio m

1. **X** sceglie a caso un numero **r** **coprimo con nT**
2. invia a **T** il testo cifrato **$c1 = m \times r^{eT} \bmod nT$**
3. **T** firma **c1** e restituisce a **X**
 $c2 = m^{dT} \times r^{eT \cdot dT} \bmod nT = (m^{dT} \times r) \bmod nT$
4. **X** moltiplica **c2** per r^{-1}
 $c3 = (m^{dT} \times r \times r^{-1}) \bmod nT = m^{dT} \bmod nU$
5. Il destinatario di **m** può verificare che è autenticato da **T**
 $(c3)^{eT} \bmod nT = m$

Attacchi Matematici alla Firma

- Un attaccante può chiedere di firmare qualunque messaggio di sua scelta tranne m di suo interesse. Per ottenere la firma di m genera r , costruisce $m_1 = m \times r$, calcola $m_2 = r^{-1}$ chiede ad X di firmare entrambi i messaggi e moltiplica infine le firme di m_1 e m_2 che gli vengono restituite
- Posso sfruttare questo attacco per intercettare un cifrato RSA destinato a X , l'intruso osserva c , se lo fa firmare da X , elimina il numero a caso ed ottiene il testo in chiaro

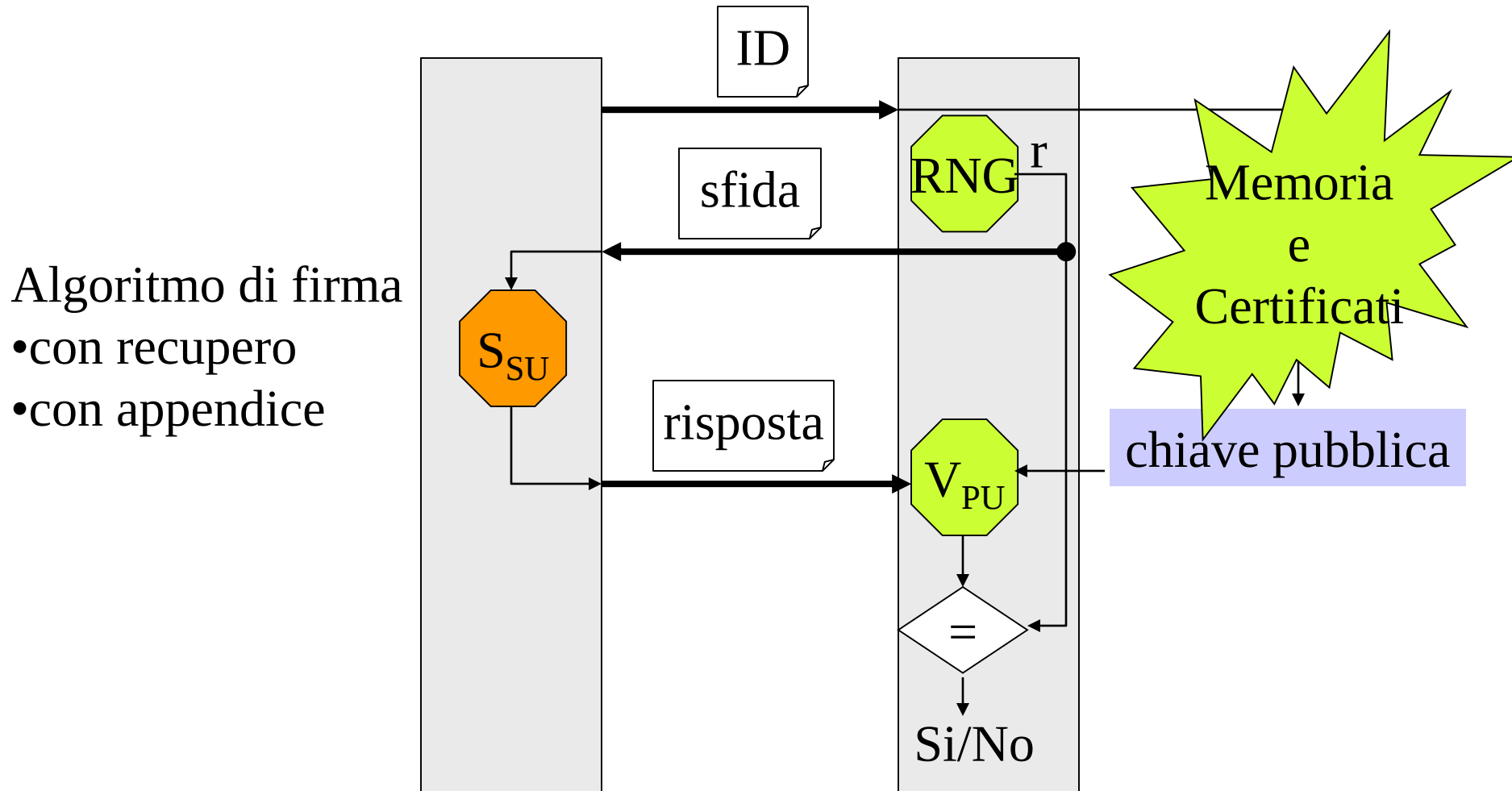
A large, bright yellow starburst shape with multiple sharp points, centered on a white background. The text "Identificazione attiva" is written in bold black font across the middle of the starburst.

Identificazione attiva

Identificazione con C/R

Identificando

Verificatore



Identificazione unilaterale

1 - $A \leftarrow B$: RB

2 - $A \rightarrow B$: $c = S_{SA}(RB)$

3 - $V_{PA}(c) = RB$?

E se RB
testo cifrato
intercettato?

Attacco con testo scelto

2 - $A \rightarrow B$: $RA || S_{SA}(RA || RB)$

Attacco dell'uomo in mezzo

E se RB
fosse un
contratto?

2 - $A \rightarrow B$: $CERT(PA, T) || RA || S_{SA}(RA || RB)$

Il problema dei GM degli scacchi

2 - $A \rightarrow B$: $CERT(PA, T) || RA || B || S_{SA}(RA || RB || B)$

τ

τ



Chiavi
diverse

Identificazione reciproca

1 - A $\leftarrow$ B: τ_{B1}

2 - A $\rightarrow$ B: $\text{CERT}(\text{PA}, \text{T}) || \tau_{\text{A}} || \text{B} || \text{S}_{\text{SA}}(\tau_{\text{A}} || \tau_{\text{B1}} || \text{B})$

3 - A $\leftarrow$ B: $\text{CERT}(\text{PB}, \text{T}') || \tau_{\text{B2}} || \text{S}_{\text{SB}}(\tau_{\text{B2}} || \tau_{\text{A}} || \text{A})$

Attacchi Matematici alla Firma

- Se si impiega firma con recupero, chiunque puo' scegliere un numero a caso r , prendere la chiave pubblica RSA di chiunque altro, calcolare $y=r^e \bmod n$ e produrre r come “firma” inconfutabile del messaggio y (y è difficile che abbia un significato importante per il proprietario della chiave privata)
- Con schema con appendice quasi impossibile, l'attaccante dovrebbe produrre anche $z=H^{-1}(y)$

Regola: il messaggio da firmare deve essere dotato di una struttura ben precisa