

# Criptografia FIB

## 6. Algoritmes aritmètics

Anna Rio

Departament de Matemàtica Aplicada II • Universitat Politècnica de Catalunya



# Entrades i sortides són **nombres enters**

El **tamany** (longitud) d'un nombre enter és el nombre de dígit (en una certa base)

$$10^{k-1} \leq n < 10^k \Rightarrow n \text{ té } k \text{ xifres decimals} \Rightarrow \ell(n) = k$$
$$\Downarrow$$
$$k - 1 \leq \log_{10} n < k \Rightarrow \ell(n) = \lfloor \log_{10} n \rfloor + 1 \approx \log n$$

La **complexitat** dels algorismes aritmètics s'ha de mirar en funció del tamany de l'entrada, és a dir, com a **funció de  $\log n$**



## Canvi de base

$$\log_2 x = \frac{\log_{10} x}{\log_{10} 2} \approx 3.32 \log_{10} x$$

El nombre de bits és aproximadament 3.32 vegades el nombre de xifres decimals

512 bits  $\Rightarrow$  154 o 155 xifres decimals

\_\_\_\_\_  $10^{153}$  \_\_\_\_\_  $2^{511}$  \_\_\_\_\_  $10^{154}$  \_\_\_\_\_  $2^{512}$  \_\_\_\_\_  $10^{155}$  \_\_\_\_\_

1024 bits  $\Rightarrow$  308 o 309 xifres decimals

...



# PROPIETATS DE LA FUNCIÓ LONGITUD

- Longitud de la suma:

$$\max\{\ell(u), \ell(v)\} \leq \ell(u + v) \leq \max\{\ell(u), \ell(v)\} + 1$$

- Longitud del producte:

$$\ell(u) + \ell(v) - 1 \leq \ell(uv) \leq \ell(u) + \ell(v),$$

- Longitud del quocient de la divisió euclidiana:

$$u = qv + r \quad (0 \leq r < v)$$

$$\ell(u) - \ell(v) - 1 \leq \ell(q) \leq \ell(u) - \ell(v) + 1,$$



## Sumes, restes i productes d'enters d'un sol dígit

- Els resultats són enters d'un dígit, més un "carry", que pot ser 0 o 1 en el cas de la suma i la resta i pot ser un dígit qualsevol en el cas del producte.
- A partir d'aquestes operacions elementals s'implementen les operacions aritmètiques amb enters de longitud arbitrària.

# SUMA/RESTA

**Algoritme de la suma:** per sumar 2 enters de  $l$  dígits cal fer  $l$  sumes elementals

$$\begin{array}{r} 685629 \\ + 472281 \\ \hline 1157910 \end{array}$$

**Algoritme de la resta:** per restar 2 enters de  $l$  dígits cal fer  $l$  restes elementals

El nombre d'operacions és igual a la longitud de les entrades.





**Algoritme de la divisió euclidiana:** per dividir un enter de  $k$  dígits per un de  $\ell$  dígits

- els dígits del quocient es van trobant per tempteig
- el candidat es multiplica pel divisor ( $\ell$  productes elementals) i el resultat es resta dels dígits més significatius del dividend ( $\ell + 1$  restes elementals)
- el resultat és positiu i menor que el divisor  $\rightarrow$  dígit correcte

# DIVISIÓ

- El quocient té com a molt  $k - \ell + 1$  dígits
- En cada pas el nombre de provatures és  $\leq b$  (la base en la que treballem)

Per tant, farem com a molt

$$\begin{aligned} &\leq b\ell(k - \ell + 1) && \text{productes elementals} \\ &\leq b(\ell + 1)(k - \ell + 1) && \text{restes elementals} \end{aligned}$$

$$O(\ell(k - \ell - 1))$$

En base  $b$ , per dividir un enter de fins a  $2\ell$  xifres per un enter de  $\ell$  xifres farem **menys de  $2b(\ell + 1)^2$  operacions elementals**

$$O(\ell^2)$$



La complexitat dels algorismes aritmètics es mesura comptant el nombre d'operacions elementals en funció de la longitud de l'entrada.

$$f_A(\ell) = \max\{nop_A(n) \mid n \text{ de longitud } \ell\}.$$

Normalment ens interessa només l'ordre de la funció  $f_A$ .  
 $f$  és de l'ordre de  $g$ , escrivim

$$f = O(g),$$

si existeix una constant positiva  $C$  tal que

$$f(\ell) \leq Cg(\ell)$$

per a tot  $\ell$  prou gran.

Un algoritme  $A$  és  $O(g)$  quan  $f_A$  ho és.

- Algoritme lineal:  $O(\ell)$  (Ex: la **suma d'enters mòdul  $n$** )
- Algoritme quadràtic:  $O(\ell^2)$  (Ex: el **producte mòdul  $n$** )
- **Algoritme polinòmic**:  $O(\ell^k)$  per a algun  $k$  (**eficient**)
- **Algoritme exponencial**:  $O(a^\ell)$  per a algun  $a > 1$  (**ineficient**)

**Problemes fàcils (classe P)**: problemes que es poden resoldre amb un algoritme polinòmic

**Problemes computacionalment intractables**: els que no pertanyen a la classe P (no existeix un algoritme polinòmic per resoldre'ls)



# MÀXIM COMÚ DIVISOR

$d$  és màxim comú divisor de  $a$  i  $b$  si els divideix a tots dos i qualsevol altre divisor comú de  $a$  i  $b$  és un divisor de  $d$

- 1  $mcd(a, b) = mcd(|a|, |b|)$
- 2  $mcd(a, 0) = a$
- 3  $mcd(a, b) = mcd(b, a - tb)$  per a tot enter  $t$

## Algoritme d'Euclides

$a$  i  $b$  enters positius

- Inicialitzem  $r_0 = a, r_1 = b$
- Mentre  $r_k \neq 0$ , fem la divisió euclidiana  $r_{k-1} = q_k r_k + r_{k+1}$
- El màxim comú divisor de  $a$  i  $b$  és l'última resta no nul.la  $r_n$



# Algoritme d'Euclides

$$178 \bmod 47 = 37$$

$$47 \bmod 37 = 10$$

$$37 \bmod 10 = 7$$

$$10 \bmod 7 = 3$$

$$7 \bmod 3 = 1$$

$$3 \bmod 1 = 0$$

L'algoritme d'Euclides es basa en  $\gcd(a, b) = \gcd(b, a \bmod b)$

While  $b \neq 0$

$$r = a \bmod b$$

$$a \leftarrow b$$

$$b \leftarrow r$$



# Algoritme d'Euclides: complexitat

- Es compleix  $r_{k+1} \leq \frac{r_{k-1}}{2}$
- Amb això s'afita el nombre de passos de l'algoritme:

$$n \leq 2 \log_2 a + 1$$

El nombre de passos és de l'ordre de la longitud de  $a$  :

$$n = O(\ell(a))$$

- En cada pas, la divisió euclidiana és  $O(\ell(r_k)\ell(q_k))$ . Afitant la suma  $\sum_{k=1}^n \ell(r_k)\ell(q_k)$ , s'obté que el nombre d'operacions que fa l'algoritme és

$$O(\ell(a)\ell(b))$$



# IDENTITAT DE BÉZOUT

Si  $d = \gcd(a, b)$ , existeixen enters  $x, y$  tals que  $ax + by = d$

De fet, hi ha infinites solucions i si  $\tilde{x}, \tilde{y}$  és una d'elles, les altres són

$$x = \tilde{x} + t b/d \quad y = \tilde{y} - t a/d \quad (t \in \mathbf{Z})$$

## Algoritme d'Euclides estès

$$\begin{aligned}x_0 &= 1, & y_0 &= 0, \\x_1 &= 0, & y_1 &= 1, \\x_{k+1} &= x_{k-1} - q_k x_k, & y_{k+1} &= y_{k-1} - q_k y_k \\ & & a x_n + b y_n &= r_n = \gcd(a, b)\end{aligned}$$



# Algoritme d'Euclides estès: complexitat

L'algoritme estès té la mateixa complexitat que l'algoritme d'Euclides simple:

$$O(\ell(a)\ell(b))$$

Proporciona les solucions **òptimes en tamany**, ja que

$$|x_n| \leq \frac{b}{2d}, \quad |y_n| \leq \frac{a}{2d}$$



# INVERSOS MODULARS

Si  $a$  és un enter tal que  $\gcd(a, n) = 1$ , aleshores existeix  $x$  tal que

$$ax \equiv 1 \pmod{n}$$

és a dir,  $a$  té invers mòdul  $n$  ( i recíprocament)

**Algoritme:**

Resoldre la identitat de Bézout  $aX + nY = 1$ .

La solució  $x \pmod{n}$  és l'invers buscat

- El càlcul d'inversos mod  $n$  es pot fer amb un algoritme  $O(\ell(n)^2)$
- L'algoritme d'Euclides estès retorna una **solució  $x$**  tal que  $|x| \leq n/2$ . Per tant,

$$a^{-1} \pmod{n} = x \pmod{n} = \begin{cases} x & \text{si } x \text{ és positiva} \\ x + n & \text{si } x \text{ és negativa} \end{cases}$$



## MCD i inversos modulars d'enters $< n$

Algoritme d'Euclides (estès)

$\mathcal{O}(\ell(n)^2)$

$\gcd(\varphi(n), e) = 1$	$\varphi(n)x + ed = 1$	
$\text{mcd}(19872, 343)$	1	0
$\text{mcd}(343, 19872 \bmod 343)$	0	1
$\text{mcd}(321, 343 \bmod 321)$	1	-57
$\text{mcd}(22, 321 \bmod 22)$	-1	58
$\text{mcd}(13, 22 \bmod 13)$	15	-869
$\text{mcd}(9, 13 \bmod 9)$	-16	927
$\text{mcd}(4, 9 \bmod 4)$	31	-1796
$\text{mcd}(1, 4 \bmod 1) = \text{mcd}(1, 0) = 1$	$x = -78$	$d = 4519$

# Claus RSA

p 849971724893325784903844427454709335666480217862558028431141  
291063388757700841813153830452688959514142719205381021252842  
3419135275391648150563020216378773

q 124987075137699682010130344784719887401171766209184429254814  
007267157805668200318061133138657163417406550755050927232487  
27898445465024365419631859460644767

$\varphi(n)$  106235479844162313112623622376706062835748388959562335916243  
821142638516264459989006286900617903692802910893730009874328  
562357789700487670277809790099753292864400477004858054996486  
514501525379223148853477167143782522741079840793414048591209  
151578767926677719276681573275451293763044439503821558324216  
795307352

e 65537

d 175943344710398362226592123117745335614876026331246409514459  
074214901270356355756393218795383787277672581113042331381656  
501797679083432743823389453551844338427178964159288543712996  
418572646972868464784112970105225369155085004191787247418860  
205873925732969157121794069965324678045086645158380639036124  
49399057

# Claus RSA

$\varphi(n) \bmod 65537$	$=$	14123	$=$	$r_2$	$q_1$
65537 mod 14123	$=$	9045	$=$	$r_3$	$q_2 = 4$
14123 mod 9045	$=$	5078	$=$	$r_4$	$q_3 = 1$
9045 mod 5078	$=$	3967	$=$	$r_5$	$q_4 = 1$
5078 mod 3967	$=$	1111	$=$	$r_6$	$q_5 = 1$
3967 mod 1111	$=$	634	$=$	$r_7$	$q_6 = 3$
1111 mod 634	$=$	477	$=$	$r_8$	$q_7 = 1$
634 mod 477	$=$	157	$=$	$r_9$	$q_8 = 1$
477 mod 157	$=$	6	$=$	$r_{10}$	$q_9 = 3$
157 mod 6	$=$	1	$=$	$r_{11}$	$q_{10} = 26$



$$y_0 = 0$$

$$y_1 = 1$$

$$y_2 = -q_1$$

$$\begin{aligned} &= -1621000043397810597259923743483926069788796 \\ &9995508237471389264254182906795315621558247539 \\ &6520902227448480848574102986600793990859667802 \\ &4173792053192849127864632199780350917115468918 \\ &5422208123536803462696975318336591961957343301 \\ &2518193678699286782684478504843487925253330868 \\ &5075042257648409624461361014317 \end{aligned}$$

$$y_3 = 1 - q_2 y_2 = 1 - 4y_2$$

$$y_4 = y_2 - y_3$$

...

$$y_{11} = y_9 - q_{10} y_{10} = y_9 - 26y_{10} \quad \text{És positiu. Per tant, } d = y_{11}$$



# EXPONENCIACIÓ MODULAR

## Exponenciació modular d'enters $< n$

$$m^e \bmod n$$

El mètode més immediat (multiplicacions successives) requereix  $e - 1$  operacions (producte amb reducció mòdul  $n$ )

$$a^{23} = a * a * a * \dots * a \quad (22 \text{ productes})$$

## Algoritme de quadrats successius

$$O(\ell(n)^3)$$

$$\begin{aligned} 23 = 10111 &= 2 \cdot 11 + 1 = 2(2 \cdot 5 + 1) + 1 = \dots = \\ &= 2(2(2(2(0 + 1) + 0) + 1) + 1) + 1 \end{aligned}$$

$$a^{23} = (((((1 * a)^2 * 1)^2 * a)^2 * a)^2 * a \quad (8 \text{ productes})$$



# Algoritme de quadrats successius

1) Escriure l'exponent en base 2

$$\begin{aligned}23 = 10111 &= 2 \cdot 11 + 1 = 2(2 \cdot 5 + 1) + 1 = 2(2(2 \cdot 2 + 1) + 1) + 1 \\ &= 2(2(2(2 \cdot 1 + 0) + 1) + 1) + 1\end{aligned}$$

2) Inicialitzar amb un 1. Recórrer la llista de bits d'esquerra a dreta: si es troba un 1 es multiplica per  $a$  i si es troba un zero no es fa res. Passar al bit següent elevat al quadrat.

$$a^{23} = (((((1 * a)^2 * 1)^2 * a)^2 * a)^2 * a) \quad (8 \text{ productes})$$



# Algoritme de quadrats successius

$$65537 = 2^{16} + 1 = 100000000000000001$$

$$a^{65537} = (((((((((((((((((((((1 * a)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2 * a$$

17 productes

## Cost de $a^k \bmod n$

Suposem que  $a$  i  $k$  són menors que  $n$

- $a^k \bmod n$  es pot fer amb  $2(\ell(k) - 1)$  productes mòdul  $n$ .
- Cada producte (amb reducció mòdul  $n$ ) costa  $O(\ell(n)^2)$ .
- L'algoritme de quadrats successius és  $O(\ell(n)^3)$



# TEOREMA XINÈS DE LES RESTES

Siguin  $n_1, n_2, \dots, n_r$  enters relativament primers dos a dos.  
El sistema de congruències

$$X \equiv a_1 \pmod{n_1}$$

$$X \equiv a_2 \pmod{n_2}$$

...

$$X \equiv a_r \pmod{n_r}$$

té solució (única mòdul  $N = n_1 \cdot n_2 \cdot \dots \cdot n_r$ )



# TEOREMA XINÈS DE LES RESTES

## Algoritme

- 1 Calcular  $N = n_1 \cdot n_2 \cdot \dots \cdot n_r$
- 2 Calcular  $N_i = N/n_i$  per a  $i = 1 \dots r$
- 3 Calcular  $K_i = N_i^{-1} \bmod n_i$  per a  $i = 1 \dots r$
- 4 La solució és  $X = a_1 N_1 K_1 + a_2 N_2 K_2 + \dots + a_r N_r K_r \bmod N$

$$O(\ell(N)^2)$$



# DESXIFRAT RSA

El cost del càlcul  $c^d \bmod n$  es pot reduir fent-ho mòdul  $p$  i mòdul  $q$  i *enganxant* després les solucions via el teorema xinès

- Pre-calcular

$$\begin{array}{ll} d_1 = d \bmod (p-1) & p_1 = p^{-1} \bmod q \\ d_2 = d \bmod (q-1) & q_1 = q^{-1} \bmod p \end{array}$$

La **clau secreta** estarà formada per  $(p, q, p_1, q_1, d_1, d_2)$

Quan arriba el criptograma  $c$ , per calcular el missatge  $m$ :

- 1) Calcular  $c_p = c \bmod p$  i  $c_q = c \bmod q$
- 2) Calcular  $c_1 = c_p^{d_1} \bmod p$  i  $c_2 = c_q^{d_2} \bmod q$
- 3) Resoldre el sistema xinès  $\begin{cases} m \equiv c_1 \pmod{p} \\ m \equiv c_2 \pmod{q} \end{cases}$

és a dir, calcular  $m = c_1 q_1 q + c_2 p_1 p \bmod n$

