# fvA3qe!408\$q 2C yg\*6Xp

w6840%f 74g68&m q0ci8.&.a3 yyC3\* a-2

# Mathématiques du secret

Lancelot PECQUET Université Paris XII

#### Plan

- 1. Définitions de base;
- 2. Cryptologie antique et moderne, notion de clé;
- 3. Cassage par force brute;
- 4. Quelques rappels d'arithmétique;
- 5. Fonction à sens unique, trappes et cryptographie à clé publique;
- 6. L'algorithme RSA (chiffrement, cryptanalyse, signature);
- 7. Les cartes « yes »;
- 8. Autres cryptosystèmes.

# Cryptologie : science (lógos) du secret (kruptós)

```
Concepts cryptologiques:
1. cryptographie : écriture (graphè) secrète ;
2. signature;
3. stéganographie : écriture discrète (steganós);
4. tatouage;
5. argent électronique;
Terminologie de base :
chiffrement : Alice écrit d'un message secret à Bob;
déchiffrement : Bob lit le message qui lui est destiné ;
décryptement : Charlie casse le secret et lit le message.
(cryptanalyse)
```

# Cryptologie "antique"

Chiffrement de César (Ier siècle av. J.C.) :

$$\mathsf{A} \longmapsto \mathsf{C}, \mathsf{B} \longmapsto \mathsf{D}, \mathsf{C} \longrightarrow \mathsf{E} \dots$$

Chiffrement Battista (1568) et Vigenère (1586) :

- 0. choix d'un mot-clé (e.g. TERMINALE)
- 1. à  $t_1$ , on décale A  $\rightarrow$  T, B  $\rightarrow$  U,... (TERMINALE)
- 2. à  $t_2$ , on décale A  $\rightarrow$  E, B  $\rightarrow$  F, . . . (TERMINALE)
- 3. à  $t_3$ , on décale A  $\rightarrow$  R, B  $\rightarrow$  M, . . (TERMINALE)

: . . .

10. à  $t_{10}$ , on recommence comme en  $t_1$ .

#### Cryptanalyse de Kasiski, (1863):

Des substitutions polyalphabétiques par statistique de fréquences.

# Quelques étapes de la cryptographie moderne

1883	principes généraux de la cryptographie moderne(Kerkhoffs)
1917	invention du masque jetable (OTP) (MAUBORGNE, VERNAM)
1939	cryptanalyse d' <i>Enigma</i> (Turing)
1949	principes mathématiques de la cryptographie (Shannon)
1976	principes de la crypto à clef publique (DIFFIE, HELLMAN)
1978	cryptosystème RSA (RIVEST, SHAMIR, ADELMAN)
1982	factorisation par crible quadratique (Pomerance)
1987	factorisation par courbes elliptiques (Lenstra)
1990	factorisation par crible algébrique (LENSTRA et al.)
1990	premiers essais de crypto quantique (Bennett et al.)
1994	factorisation quantique(Schor)
1998	première machine quantique 4 qubits ( $\simeq$ 4 bits)(Bell Labs)
2002	test rapide de la primalité(AGRAWAL, KAYAL, SAXENA)

### Algorithme publié utilisant une clé

```
Principe fondamental de la cryptographie moderne (Kerkhoffs, 1863) :
cryptosystème expliqué publiquement
une clé est choisie et gardée secrète par l'utilisateur
Cryptographie à clé secrète : (coffre-fort)
Principe : clé secrète partagée pour chiffrer/déchiffrer;
Avantages: algo rapides (70Mo/s par AES en C++ sur un Pentium 200);
Inconvénients : secret partagé "hors ligne" ;
               autant de secrets que d'interlocuteurs;
Cryptographie à clé publique : (boîte aux lettres)
Principe : un clé sert à chiffrer (poster), l'autre à déchiffrer (ouvrir la boîte) ;
Avantages : pas de secret préalable ; signature possible ;
Inconvénients: Algorithmes lents (RSA est 1000 fois plus lent que DES);
En pratique : méthodes hybrides.
```

### Cryptanalyse par force brute

Énumération de toutes les clés possibles (complexité exponentielle)!

DES 56 bits (1998) : 39 jours sur 10 000 Pentium ; 56 h sur une machine à 1500 circuits dédiés ; Triple DES à 2 clés (112 bits) prendrait  $2^{56} \simeq 10^{17}$  fois plus de temps.

temps écoulé depuis le Big Bang			secondes
temps restant avant le Big Crunch	(Univers fermé)	$10^{18}$	secondes
temps restant avant liquéfaction à $0^{\circ}$	K (Univers ouvert)	$10^{72}$	secondes

l'Univers a existé et existera au plus  $T=10^{72}\simeq 2^{239}$  secondes ; l'Univers contient au plus  $N=10^{77}\simeq 2^{256}$  atomes ;

#processeurs à 10GHz	#opérations par s	$ imes 10^9$ (par siècle)	en $T$ s
1	$10^{10} \simeq 2^{35}$	$10^{19} \simeq 2^{63}$	$10^{82} \simeq 2^{272}$
$10^9 \simeq 2^{29}$	$10^{19} \simeq 2^{63}$	$10^{28} \simeq 2^{93}$	$10^{91} \simeq 2^{302}$
N	$10^{87} \simeq 2^{272}$	$10^{96} \simeq 2^{319}$	$10^{159} \simeq 2^{528}$

### Division euclidienne et congruences

Th: pour tout  $(a,b) \in \mathbb{N}^2$  avec  $b \neq 0$ , il existe un unique  $(q,r) \in \mathbb{N}^2$  tel que:

$$a = b \times q + r$$
 et  $r < b$ .

Def : on dit alors que a est congru à r modulo b et on note :

$$a \equiv r \bmod b$$
.

Si le reste r est nul, et on dit aussi que b divise a.

Ex : si a = 15 et b = 6, on a q = 2 et r = 3 :

$$15 = 6 \times 2 + 3$$
 ou encore  $15 \equiv 3 \mod 6$ .

si 
$$a = 21$$
,  $b = 7$ , on a  $q = 3$  et  $r = 0$ :

$$21 = 7 \times 3 + 0$$
 ou encore  $21 \equiv 0 \mod 7$  *i.e.* 7 divise  $21$ .

### pgcd et Algorithme d'Euclide

Th: si a=bq+r est la division euclidienne de a par b, pour tout entier c: c divise à la fois a et b ssi c divise à la fois b et r.

Th: il existe un unique plus grand entier g qui divise à la fois a et b. On le note  $g = \operatorname{pgcd}(a, b)$  et on le calcule grâce à la formule d'Euclide :

$$\operatorname{pgcd}(a,b) = \begin{cases} a & \text{si } b = 0 \ ; \\ \operatorname{pgcd}(b,r) & \text{sinon, où } r \stackrel{\scriptscriptstyle \mathrm{def}}{=} \text{ le reste de la division de } a \text{ par } b \ . \end{cases}$$

Ex: 
$$pgcd(126, 35) = pgcd(35, 21)$$
 car  $126 = 3 \times 35 + 21$   
=  $pgcd(21, 14)$  car  $35 = 1 \times 21 + 14$   
=  $pgcd(14, 7)$  car  $21 = 1 \times 14 + 7$   
=  $pgcd(7, 0)$  car  $14 = 2 \times 7 + 0$   
= 7.

Def : si pgcd(a, b) = 1, on dit qu'ils sont premiers entre-eux.

# Théorème de Bézout et Algorithme d'Euclide étendu

Th de Bézout : il existe une infinité de  $(x,y) \in \mathbb{Z}^2$  tels que  $ax+by = \operatorname{pgcd}(a,b)$ . On note  $\operatorname{bez}(a,b)$  le couple (u,v) que l'on peut calculer grâce à la formule :

$$\mathrm{bez}(a,b) = \begin{cases} (0,1) & \text{si } a = bq + r \text{ avec } r = 0 \\ (v',u'-v'q) & \text{sinon, où } (u',v') = \mathrm{bez}(b,r) \end{cases}$$

Démo de la formule : si  $(u',v')=\mathrm{bez}(b,r)$  on a u=v' et v=(u'-v'q) car  $\mathrm{pgcd}(b,r)=bu'+rv'=bu'+(a-bq)v'=av'+b(u'-v'q)=\mathrm{pgcd}(a,b).$  Autres solutions :  $(x,y)=\left((u-kb/g),v+(ka/g)\right)$  où  $g=\mathrm{pgcd}(a,b)$  et  $k\in\mathbb{Z}.$  On a essentiellement les mêmes étapes que pour l'Algo. d'Euclide :

$$126 = 3 \times 35 + 21 \qquad \text{bez}(35, 21) = (2, -1 - 2 \times 3) = (2, -7)$$
et  $35 = 1 \times 21 + 14 \qquad \text{bez}(35, 21) = (-1, 1 - (-1) \times 1) = (-1, 2)$ 

$$\downarrow \text{ et } 21 = 1 \times 14 + 7 \qquad \text{bez}(21, 14) = (1, 0 - 1 \times 1) = (1, -1) \qquad \uparrow$$
et  $14 = 2 \times 7 + 0 \qquad \text{donc} \quad \text{bez}(14, 7) = (0, 1)$ 

On sait résoudre  $ax \equiv \operatorname{pgcd}(a, b) \mod b$ 

# Rapidité d'exécution de l'Algorithme d'Euclide

Th. de Lamé (1845) : si  $a,b \leq n$ , alors le nombre de divisions à effectuer dans l'algorithme d'Euclide (étendu ou pas) est au plus  $\lceil \log_{\varphi}(\sqrt{5}n) \rceil - 2$  où  $\varphi$  est le nombre d'or  $\frac{1+\sqrt{5}}{2}$ .

Pour  $n = 10^r$ , on a :

$$\log_{\varphi}(\sqrt{5}n) = \frac{\ln(\sqrt{5}) + \ln(10) \cdot \log_{10}(n)}{\ln(\varphi)} \le \frac{0.81 + 2.31r}{0.48} \le 1.7 + 4.82r ,$$

et  $\lceil \log_{\varphi}(\sqrt{5}n) \rceil - 2 \le \lceil 4.82r \rceil$  donc le pgcd de deux entiers de r=100 chiffres prend au plus 482 divisions.

Coût linéaire en le nombre de chiffres : algorithme rapide!

### Rappel sur les nombres premiers

Def: Un entier  $p \in \mathbb{N}$  est premier ssi il a exactement deux diviseurs.

Th: Pour tout entier naturel n, il existe un unique  $(p_1, \ldots, p_m)$  de nombres premiers et un unique  $(e_1, \ldots, e_m)$  d'entiers naturels tels que :

 $n=p_1^{e_1}\cdots p_m^{e_m}$  (décomposition de n en facteurs premiers)

```
Ex: 9847451597917500 = 2^2 \times 3^{14} \times 5^4 \times 7^7; 9847451597917403 = 227 \times 282677 \times 153464357; 9847451597917457 = 9847451597917457.
```

# Fonction à sens unique et factorisation

Déf : Une fonction  $f: x \longmapsto f(x) = y$  est à sens unique ssi :

- 1. Si x est donné, il est facile de calculer y;
- 2. Si y est donné, il est *infaisable* de calculer x.

Soient p, q deux grands entiers premiers tels que n = pq:

- 1. si p, q sont connus alors il est facile de calculer n;
- 2. si on ne connaît que n alors il est très difficile de trouver p,q.

Avec un bon algo sur Athlon XP1800+ 512Mo, on trouve en 1.7s que :

```
\underbrace{189800197355371869102891045449272876576189}_{\text{42 chiffres} \,\simeq\, 138 \text{ bits}} = \underbrace{230498230498230498299}_{\text{20 chiffres} \,\simeq\, 68 \text{ bits}} \times \underbrace{823434509432509235111}_{\text{21 chiffres} \,\simeq\, 70 \text{ bits}}
```

 $\rightarrow$  ajustement de la taille de (p,q) pour que le problème soit *infaisable*.

### Rappel sur le Théorème de Fermat-Euler

Def : la fonction indicatrice d'Euler est la fonction  $\varphi: \mathbb{N} \longrightarrow \mathbb{N}$  qui, à tout entier n associe le nombre  $\varphi(n)$  d'entiers k premiers avec n.

**E**x : pour n = 10, on a  $\varphi(n) = |\{1, 3, 7, 9\}| = 4$ .

Th. de Fermat-Euler : si a est premier avec n, alors  $a^{\varphi(n)} \equiv 1 \mod n$ .

Ex: si a = 3 et n = 10, on a pgcd(3, 10) = 1 et  $3^{\varphi(n)} = 3^4 = 81 \equiv 1 \mod 10$ .

Th : si n=pq avec p et q premiers, alors  $\varphi(n)=(p-1)(q-1)$ .

**E**x : en effet,  $\varphi(10) = (2-1) \times (5-1) = 4$ .

### Exponentiation modulaire rapide

Soit  $m \in \mathbb{N}$ , et  $l \geq 1$ , pour calculer  $m^l \mod n$ , on utilise la formule :

$$m^l \bmod n = \begin{cases} m \bmod n & \text{si } l = 1 \\ \left(m^{l/2}\right)^2 \bmod n & \text{si } l \text{ est pair} \\ \left(m^{(l-1)/2}\right)^2 \times m \bmod n & \text{si } l \text{ est impair} \end{cases}$$

#### Avantages:

les nombres manipulés sont toujours inférieurs à n; coût linéaire en le nombre de chiffres de l (rapide!).

Ex: m = 324, n = 531, l = 9. Comparaison classique/rapide:

$$\underbrace{m = 324 \longrightarrow m^2 = 104976 \longrightarrow \cdots \longrightarrow m^9 = 39346408075296537575424}_{9 \text{ opérations sur des entiers de taille croissante : lent!} \longrightarrow m^9 \bmod n = 441 \ .$$

$$m^{2} \bmod n = 369$$

$$m^{4} \bmod n = (m^{2})^{2} \bmod n = 225$$

$$m^{9} \bmod n = (m^{4})^{2} \times m \bmod n = 441$$

$$3 opérations sur des entiers bornés : rapide!$$

### Fonction à trappe et principe de RSA

**Déf**: Une fonction  $f: x \longmapsto f(x) = y$  est à trappe ssi :

- 1. Si x est donné, il est facile de calculer y;
- 2. Si y est donné seul, il est *infaisable* de calculer x.
- 3. Si y est donné et qu'on connait la trappe, il est facile de calculer x.

Soient p,q premiers,  $n \stackrel{\text{def}}{=} pq$ , e premier avec  $\varphi(n)$ . On suppose (n,e) public.

- 1. si m est donné alors il est facile de calculer  $c \stackrel{\text{def}}{=} m^e \mod n$ ;
- 2. si c est donné alors il est *infaisable* de retrouver m;
- 3. si c est donné et qu'on connaît (p,q), alors il est facile de calculer m :
  - (a) on calcule  $\varphi(n) = (p-1)(q-1)$ ;
  - (b) on calcule d (Bézout), de telle sorte que  $ed \equiv 1 \mod \varphi(n)$ ;
  - (c) on calcule  $c^d=m^{ed}=m^{1+k\varphi(n)}=m\cdot (m^{\varphi(n)})^k\equiv m \bmod n$  ;
  - (d) si  $pgcd(m, n) \neq 1$ , la méthode marche toujours (théorème).

#### Chiffrement et déchiffrement RSA

#### Phase de création de clé :

- 1. Alice choisit p, q, deux grands entiers premiers au hasard;
- 2. Alice calcule n = pq et  $\varphi(n) = (p-1)(q-1)$ ;
- 3. Alice choisit e premier avec  $\varphi(n)$  et calcule d pour que  $ed \equiv 1 \mod n$ ;
- 4. Alice publie (n, e).

#### Phase de chiffrement :

- 1. Bob convertit son message en une suite d'entiers m < n;
- 2. Pour chaque message m, Bob calcule  $c \stackrel{\text{def}}{=} m^e \mod n$ ;
- 3. Bob envoie c à Alice.

#### Phase de déchiffrement :

- 1. Alice reçoit c;
- 2. Alice calcule  $m \equiv c^d \mod n$ .
- 3. Alice lit le message de Bob.

## Exemple de chiffrement/déchiffrement RSA

#### Phase de création de clé :

- 1. Bob choisit p = 32409827, q = 92341019
- 2. Bob calcule  $n \stackrel{\text{def}}{=} pq = 2992756450793713$  et  $\varphi(n) = (p-1)(q-1) = 2992756326042868$ ;
- 3. Bob choisit e=1814008226250739 qui est premier avec  $\varphi(n)$  (algo d'Euclide) et calcule d=324056296185433 (algo d'Euclide étendu) pour que  $ed\equiv 1 \bmod \varphi(n)$ ;
- 4. Bob publie (n, e).

#### Phase de chiffrement :

1. Alice traduit en ASCII et en base 256 :

$$m = \text{love} = \underbrace{108}_{1} + \underbrace{111}_{0} \cdot 256 + \underbrace{118}_{v} \cdot 256^{2} + \underbrace{101}_{e} \cdot 256^{3} = 1702260588$$

- 2. Alice calcule  $c \stackrel{\text{def}}{=} m^e \mod n = 1947230704349401$ ;
- 3. Alice envoie c à Alice.

#### Phase de déchiffrement :

- 1. Bob reçoit c = 1947230704349401;
- 2. Bob calcule  $m \equiv c^d \mod n = 1702260588$  et retraduit vers l'ASCII : love

### Autres primitives de calcul dans RSA

- 1. Comment choisir p, q premiers au hasard?
  - (a) Générer p, q au hasard :
    - i. Utiliser le hasard physique;
    - ii. L'amplifier avec des méthodes cryptographiquement sûres;
    - iii. Théorème densité de Hadamard et de la Vallée Poussin (1896) : environ un entier de l chiffres sur l est premier.
  - (b) Tester s'ils sont premiers :
    - i. Crible d'Erathosthènes (impraticable);
    - ii. Tests (Miller-Rabin, Solovay-Strassen, ECPP, AKS,...).
- 2. Comment choisir e premier avec  $\varphi(n)$ ?
  - (a) choisir e au hasard, puis tester si  $\operatorname{pgcd}(e, \varphi(n)) = 1$ .

### Cryptanalyses de RSA

Décryptement : Charlie connaît (n, e) et c et doit trouver m, voire (p, q).

- 1. si  $\varphi(n)$  est connu alors on retrouve immédiatement (p,q);
- 2. connaître d revient à savoir factoriser n (théorème);
- 3. la factorisation de n permet d'inverser vite  $m \longmapsto m^e \mod n$ ;
- 4. la réciproque de  $m \longmapsto m^e \mod n$  permet-elle de factoriser vite n?

En 1999, avec le crible algébrique :

- phase 1 : 300 ordinateurs pendant 3 mois ( $\simeq$  8000 années-MIPS);
- phase 2 : 224 heures sur le Cray-C916 du SARA à Amsterdam ;
   ont factorisé l'entier de 512 bits (155 chiffres décimaux) :

 $10941738641570527421809707322040357612003732945449205990913842131476349984288 \\ 934784717997257891267332497625752899781833797076537244027146743531593354333897 = \\ 102639592829741105772054196573991675900716567808038066803341933521790711307779 * \\ 106603488380168454820927220360012878679207958575989291522270608237193062808643$ 

En 2001, une machine quantique IBM 7 qubits a factorisé  $15 = 3 \times 5$ .

### Signature RSA

#### Phase de création des clés de Bob identique à celle du chiffrement RSA :

- 1. privés :  $\varphi(n)$ , d (on n'a plus besoin de p, q);
- 2. publics : *n*, *e*.

#### Phase de signature :

- 1. Bob convertit le message à signer en entier m < n;
- 2. Bob calcule  $s \equiv m^d \mod n$ ;
- 3. Bob envoie m et s.

#### Phase de vérification :

- 1. Alice reçoit m et s;
- 2. Alice calcule  $m' \equiv s^e \mod n$  et vérifie que m' = m;
- 3. Alice est alors sûr que m a été signé par la clé privée de Bob.

Problème : comment être sûr que (n, e) appartient bien à Bob?

Réponse : autorité de certification (tiers de confiance).

### Exemple de signature RSA

#### Bob utilise sa clé habituelle :

- 1. privés :  $\varphi(n) = 2992756326042868$ , d = 324056296185433;
- 2. publics : n = 2992756450793713, e = 1814008226250739.

#### Phase de signature :

1. Bob traduit en ASCII et en base 256 :

$$m = \text{me too} = \overbrace{109}^{\text{m}} + \overbrace{101}^{\text{e}} \cdot 256 + \overbrace{32}^{\text{e}} \cdot 256^2 + \overbrace{116}^{\text{t}} \cdot 256^3 + \overbrace{111}^{\text{o}} \cdot 256^4 + \overbrace{111}^{\text{o}} \cdot 256^5 \\ = 128022038472045$$

- 2. Bob calcule  $s = m^d = 2575691833347216$ ;
- 3. Bob envoie s et m à Alice.

#### Phase de déchiffrement :

- 1. Alice reçoit s=2575691833347216 et m=128022038472045;
- 2. Alice calcule  $m' \equiv s^e \mod n = 128022038472045 = m$ ;
- 3. Alice est alors sûre que m a été signé par la clé privée de Bob.

# Cartes "yes"

Dans une carte à puce : numéro de compte m signé par une banque B; un programme qui teste le code confidentiel.

Paiement d'une petite somme : vérification de la signature de B sur m; vérification du code confidentiel;

Pour fabriquer une carte « yes », achetez une carte vierge puis :

- 1. programmez la pour qu'elle réponde toujours « code correct »;
- 2. obtenez une clé publique (n, e) de banque à 320 bits (terminaux);
- 3. factorisez n = pq en quelques jours sur un PC; calculez d;
- 4. choisissez au hasard un numéro bancaire m;
- 5. signez m et inscrivez le tout dans la carte;
- 6. allez en prison... (affaire HUMPICH).

Les clés font maintenant au moins 792 bits.

#### Autres cryptosystèmes

#### Chiffrement à clé secrète :

- 1. DES (1977) à 56 bits  $\rightarrow$  triple DES à deux clés : 112 bits ;
- 2. IDEA (1992): 128 bits;
- 3. AES (2000): 128, 192, 256 bits, au choix.

#### Chiffrement à clé publique :

- 1. logarithme discrets sur les corps finis (1978);
- 2. log discret sur les courbes elliptiques (ECC) et hyperelliptiques (1985);
- 3. cryptosystème à décodage de McEliece (1978).

```
ECC 160 \simeq RSA 1024
108 bit cassé (Harley, 2000) 512 bit cassé, 1999
```

Signature: RSA  $1024 \simeq DSA$  (log. discret)  $320 \simeq McEliece$  111.

#### J'ai fini

Merci de votre attention, avez-vous des questions?

