MT10

Mathématiques pour la cryptographie

Première partie

Primalité et factorisation

Walter SCHÖN

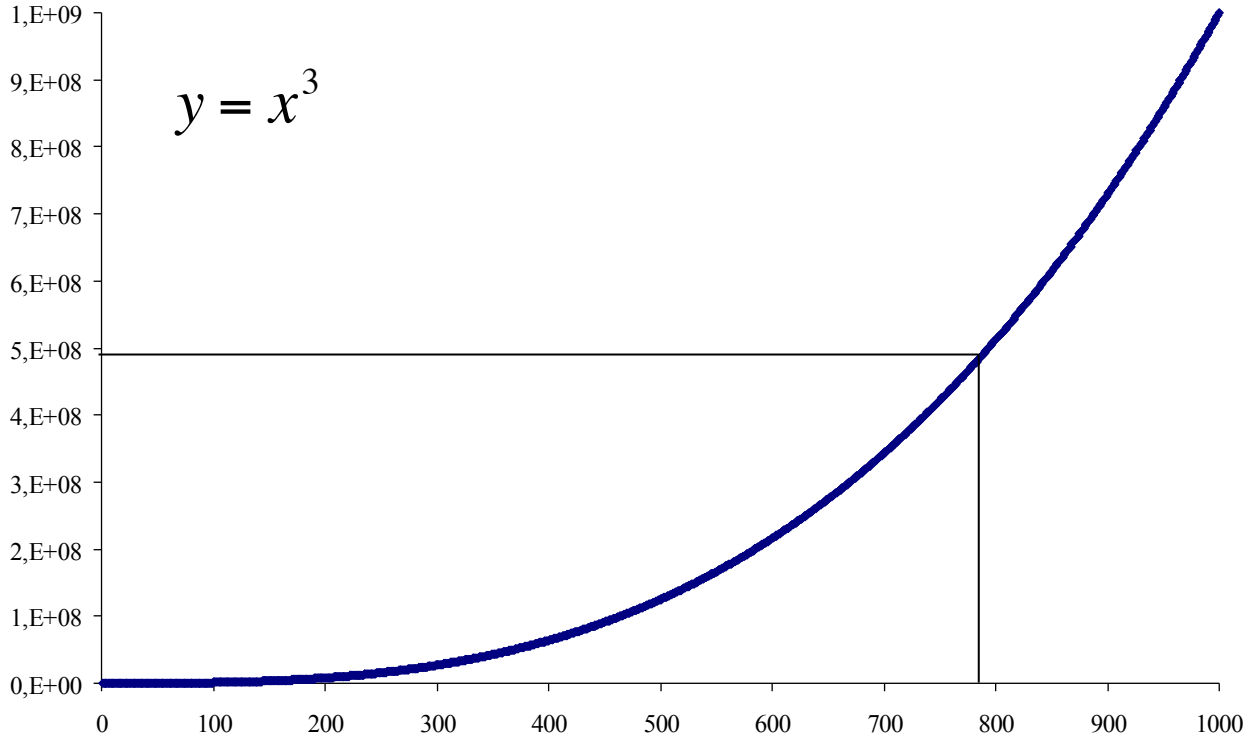
Chiffrement par blocs à clé publique

- Tous les algorithmes à clé publique (algorithmes asymétriques) reposent sur la difficulté *supposée* de certains problèmes mathématiques pour lesquels l'opération inverse est simple :

Il est facile de	Il est difficile de
•Calculer le produit pq (2 grands entiers premiers)	•Trouver les facteurs pq, en ne connaissant que n=pq (problème de la factorisation)
•Calculer a^x mod n (a, x et n grands entiers)	•Trouver x / a^x mod n = b en ne connaissant que b, a et n (problème du logarithme discret)

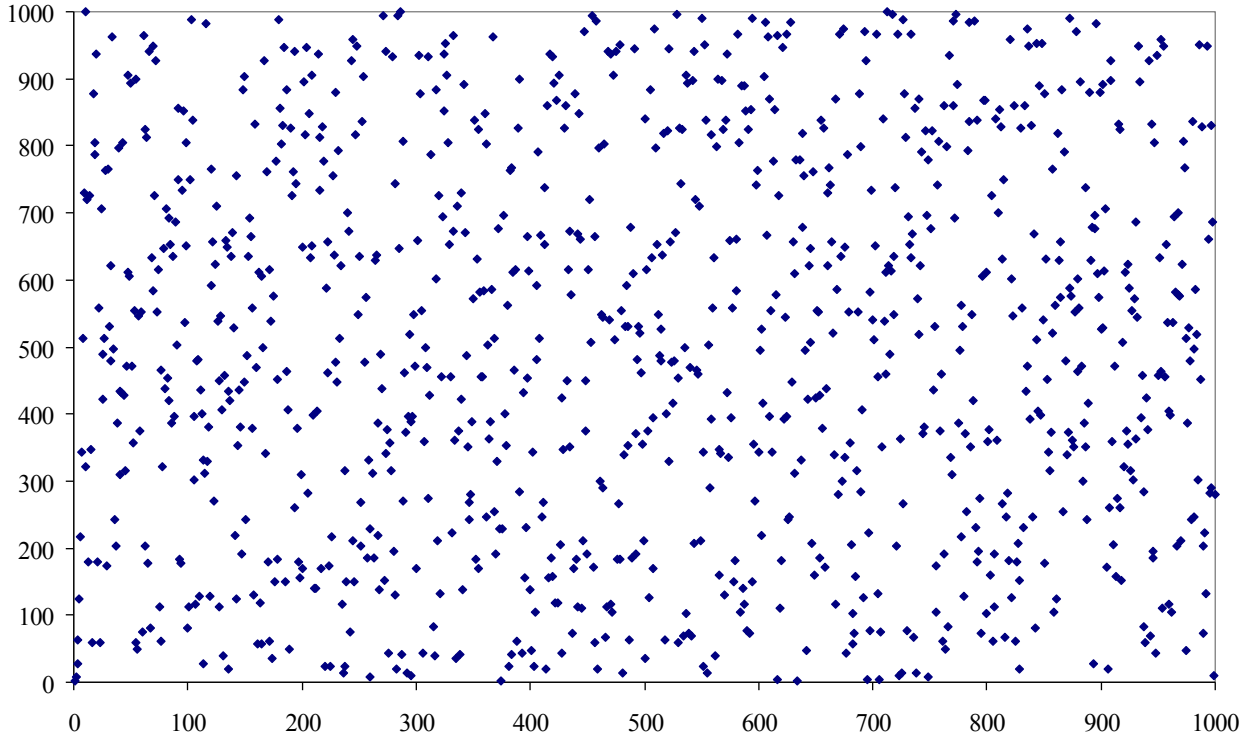
Un exemple de problème facile dans Z...

- Recherche de la racine cubique d'un entier



Et son équivalent dans Z/pZ...

- Très compliqué si p est grand (ici p=1009)



Fonctions à sens unique à porte d'évitement secrète

- De telles fonctions sont supposées être des fonctions à sens unique à porte d'évitement secrète (trapdoor one-way functions) :
- La fonction est facile à calculer (calcul d'un produit),
- L'inversion de la fonction (factorisation) est un problème difficile qui prendrait un temps prohibitif, même compte tenu des progrès supposés des algorithmes et de la puissance informatique (aspect sens unique),
- **Mais** : si l'on connaît un secret (l'un des deux facteurs par exemple) l'inversion redevient facile (aspect porte d'évitement secrète)

Le premier algorithme a clé publique : Diffie-Hellmann (1/2)

- Diffie-Hellmann permet d'échanger une clé secrète sur un canal non sûr. Il fut à l'origine des concepts de la cryptographie à clé publique. Il est basé sur la difficulté supposée du calcul de logarithmes discrets sur un corps fini :
- Les deux protagonistes choisissent deux entiers p et g : p doit être grand et premier, et g un générateur du groupe multiplicatif $(Z/pZ)^*$: la suite des g^k $1 \leq k \leq p-1$ doit parcourir tous les éléments de $(Z/pZ)^*$: on verra dans la partie sur les corps fini qu'un tel générateur existe toujours (i.e le sous-groupe multiplicatif d'un corps fini est cyclique)
- Le premier choisit un grand nombre entier x et calcule $X=g^x \bmod p$ qu'il transmet au second
- Le second choisit un grand nombre entier y et calcule $Y=g^y \bmod p$ qu'il transmet au premier

Le premier algorithme a clé publique : Diffie-Hellmann (2/2)

- Celui qui a choisi x et reçu Y calcule $Y^x \bmod p$
- Celui qui a choisi y et reçu X calcule $X^y \bmod p$
- Les deux résultats sont égaux à $g^{xy} \bmod p$ qui devient la clé secrète partagée par les deux
- Un indiscret éventuel ne connaît que g, p, X et Y
- Ignorant x et y, il n'a aucun moyen simple de calculer $g^{xy} \bmod p$, à moins de les retrouver, ce qui revient à calculer le logarithme discret à base g dans GF(p)

Primalité et factorisation : introduction

- Les algorithmes de cryptographie à clé publique (RSA…) sont grands utilisateurs de grands nombres premiers
- Il est donc important de disposer d'algorithmes de génération de grands nombres premiers
- Par ailleurs RSA repose sur la difficulté de la décomposition en facteurs premiers : la recherche d'algorithmes de factorisation efficaces permet donc d'estimer la robustesse du chiffrage dans l'état actuel des connaissances scientifiques et technologiques

Arithmétique modulaire et primalité : rappels

- Un entier a est inversible modulo un autre entier b <=> a et b premiers entre eux
- Inverse efficacement déterminé par l'algorithme d'Euclide étendu (qui donne pgcd(a,b) et deux entiers u et v tels que au+bv=pgcd). L'inverse de a modulo b et donc u mod b
- (Petit) théorème de Fermat : Si p est premier, alors pour tout entier a non multiple de p : $a^{p-1} \bmod p = 1$
- Algorithme d'exponentiation rapide : Calcul de $a^p \bmod m$ pour des grands entiers : calculer de proche en proche la suite des $a^{2^i} \bmod m$, représenter p en binaire, p : $c_n c_{n-1} \ldots c_0$, calculer enfin de proche en proche : $\prod_{i=0}^{n} (a^{2^i} \bmod m)^{c_i}$

Petit théorème de Fermat : rappel

- a inversible modulo b <=> a et b premiers entre eux : démonstration de => et <= très facile.
- Si p est premier, tout entier non multiple de p est donc inversible modulo p.
- $((Z/pZ)^*,*,1)$ est donc un groupe
- Pour tout élément a de ce groupe, son ordre (ordre du groupe engendré par a, qui est le plus petit entier m tel que $a^m=1$) divise l'ordre du groupe (Lagrange) qui vaut ici p-1
- Par conséquent pour tout entier a non multiple de p

$$a^{p-1} \bmod p = 1$$

Petit théorème de Fermat : démonstration élémentaire

- Soit p premier et a non multiple de p,
- Les p-1 premiers multiples de a : a, 2a, …, (p-1)a sont tous distincts et non nuls modulo p.
- En effet si na=ma mod p alors n=m mod p (car p ne divise pas a). D'autre part si p divisait ka (1<k<p-1) il diviserait a
- Ces p-1 entiers valent donc 1, 2, …, p-1 dans un certain ordre.
- Par multiplication, on obtient $a^{p-1}(p-1)! = (p-1)! \bmod p$
- Par simplification par (p-1)! on obtient

$$a^{p-1} \bmod p = 1$$

Rappel sur les (petits) groupes

- Il résulte de ce qui précède :
- Il n'y a qu'un seul groupe (à un isomorphisme près) à 1 (trivial), 2 et 3 éléments (le groupe engendré par n'importe lequel de ses éléments différent de 1)
- $((Z/5Z)^*,*,1)$ est l'un des deux groupes à 4 éléments dont deux éléments sont d'ordre 4 (2->4->3->1, 3->4->2->1), et le dernier élément différent de 1, est d'ordre 2 (son propre inverse) 4->1, ce groupe est donc cyclique
- Mais il y a un autre groupe à 4 éléments que l'on peut identifier à $((Z/2Z)*(Z/2Z),+, (0,0))$ (ou bien encore au groupe des rotations du matelas !) où tout élément différent de l'élément neutre est son propre inverse (d'ordre 2), ce groupe n'est pas cyclique

Rappel sur les (petits) groupes

- Exercice : déterminer l'ordre de tous les éléments de $((Z/7Z)^*,*,1)$
- 2->4->1, 3->2->6->4->5->1, 4->2->1, 5->4->6->2->3->1, 6->1)
- Il y a donc des éléments d'ordre 2 et d'ordre 3, mais il y en a aussi d'ordre 6, générateurs du groupe qui est donc cyclique
- D'une manière générale comme rappelé plus haut et utilisé plus bas si p est premier Z/pZ est un corps fini, dont le sous groupe multiplicatif est cyclique.

Primalité d'un nombre : test de Fermat

- Soit p un nombre candidat premier
- Soit a un nombre test non multiple de p
  (pris en général $< p$)
- Si $a^{(p-1)}$ modulo p ne vaut pas 1, alors le nombre p n'est certainement pas premier
- On calcule $a^{(p-1)}$ par l'algorithme d'exponentiation rapide

Test de Fermat : Exemple

- 21 n'est pas premier :
- Avec a=2 comme nombre test, on calcule 2^{20} mod 21 : 20_2=10100

2^1 mod 21 = 2	0	1
2^2 mod 21 = 4	0	1
2^4 mod 21 = 4^2 mod 21 = 16	1	16
2^8 mod 21 = 16^2 mod 21 = 256 mod 21 = 4	0	16
2^{16} mod 21 = 4^2 mod 21 = 16	1	16*16 mod 21=4

2^{20} mod 21= 4

21 n'est donc pas premier

Exemple (suite)

- Mais attention il y a des menteurs…
- Avec a=8 comme nombre test, on calcule 8^{20} mod 21

8^1 mod 21 = 8	0	1
8^2 mod 21 = 64 mod 21 = 1	0	1
8^4 mod 21 = 1	1	1
8^8 mod 21 = 1	0	1
8^{16} mod 21 = 1	1	1

8^{20} mod 21= 1

- Les menteurs peuvent même être très nombreux dans le cas des nombres de Carmichael (tous les nombres tests sont menteurs sauf ceux qui ne sont pas premiers avec le nombre candidat premier, appelé dans ce cas fortement pseudo premier)

Propriétés de Z/pZ pour p premier

- Si p est premier tout entier est inversible modulo p et Z/pZ est donc un corps
- Le sous groupe multiplicatif de ce corps est d'ordre p-1 ce qui prouve directement $a^{p-1} \bmod p = 1$ d'après les propriétés des groupes vues en cours
- Le corps Z/pZ est de caractéristique et de cardinalité égales à p, nous y reviendrons également
- Il est bien sûr commutatif mais on sait que tout corps fini est commutatif (propriété non triviale…)
- Dans Z/pZ comme dans tout corps l'équation $x^2=1$ soit $(x-1)(x+1)=0$ n'a que deux racines : 1 et -1 (-1 s'écrivant p-1 dans Z/pZ)
- En effet $x^2=1 \iff (x-1)(x+1)=0$ et comme Z/pZ est un corps pour p premier, si l'un des éléments x-1 ou x+1 est non nul il suffit de multiplier par son inverse pour conclure que l'autre est nul

Génération des grands nombres premiers

- Pour des grands nombres premiers les méthodes classiques (type crible d'Eratosthène) ne sont pas envisageables
- On utilise alors des tests de primalité qui rendent un résultat type : «p n'est certainement pas premier» ou «p est probablement premier»
- En répétant le test plusieurs fois, on parvient à générer des nombres pour lesquels les chances qu'ils ne soient pas premiers sont très faibles
- L'un des plus utilisés est le test de Rabin-Miller décrit ci-après

Génération des grands nombres premiers : Rabin-Miller (1/5)

Rabin-Miller est basé sur deux propriétés des nombres premiers :

- ✓ Pour tout a non nul et < p : a^{p-1} mod p = 1 (théorème de Fermat)
- ✓ Dans Z/pZ l'équation $x^2=1$ n'a que les deux solutions triviales x=1 et x=-1 (autre écriture de l'élément p-1)
- ✓ L'utilisation du premier critère seulement donne le test de Fermat pour lequel on sait qu'il existe des nombres non premiers qui passent le test avec succès pour tous les nombres test (les nombres de Carmichael) : le test de Rabin Miller est plus fort.

Génération des grands nombres premiers : Rabin-Miller (2/5)

Étant donné un candidat nombre premier p et un nombre test non nul a<p on effectue les actions suivantes :

- Trouver b : nombre de fois que 2 divise p-1 (p-1 est pair donc b vaut au moins 1).
- Déterminer m tel que $p-1=2^b.m$
  - Envisager la suite des $\left(a^m\right)^{2^j}$ (modulo p) pour 0<=j<=b (obtenue par élévations au carré successives à partir d'un élément initial a^m). Le dernier élément de la suite vaut $\left(a^m\right)^{2^b} = a^{2^b.m} = a^{p-1} = 1$ (modulo p)

    d'après le théorème de Fermat

Génération des grands nombres premiers : Rabin-Miller (3/5)

Si p est effectivement premier, deux cas sont alors possibles :

- Soit le premier terme a^m vaut 1 ou -1 (modulo p) et auquel cas tous les autres termes valent 1
- Sinon les élévations au carré successives doivent obligatoirement faire tomber sur -1 pour un j<b

Il est en effet impossible de tomber sur 1 sans que la valeur précédente soit -1 car sinon on aurait trouvé une racine carrée de l'unité différente de 1 ou -1

On peut donc coder l'algorithme sous la forme :

RabinMiller(p : Candidat premier, a : Entier test<p) :
(NonPremier, ProbablementPremier)

Génération des grands nombres premiers : Rabin-Miller (4/5)

RabinMiller(p,a) : NonPremier, ProbablementPpremier

Diviser p-1 par 2 autant de fois que possible
b=Nombre de fois que 2 divise p-1
$m=(p-1)/2^b$
$z=a^m \bmod p$ (algorithme d'exponentiation rapide)
Si z=1 ou p-1 retourner ProbablementPremier
Pour 1=<j<b Boucle
    z=z*z mod p
    Si z=p-1 retourner ProbablementPremier
    Si z=1 retourner NonPremier
    j=j+1
FinBoucle
Retourner NonPremier

Fin RabinMiller

Génération des grands nombres premiers : Rabin-Miller (5/5)

- On reprend l'exemple de 21 : $20=2*2*5=2^b*m$: b=2, m=5
- Avec a=8 comme nombre test, on calcule 8^5 mod 21

$8^1 \bmod 21 = 8$

$8^2 \bmod 21 = 64 \bmod 21 = 1$

$8^4 \bmod 21 = 1$

1	8
0	8
1	8

$8^5 \bmod 21 = 8$

- On procède alors aux élévations au carré :

$(8^5)^2 \bmod 21 = 8^2 \bmod 21 = 64 \bmod 21 = 1$

- On a trouvé une racine carrée non triviale de l'unité (en fait déjà rencontrée en cours d'exponentiation rapide dans cet exemple)… 21 n'est donc pas premier

Génération des grands nombres premiers : Rabin-Miller (5/5)

On peut ainsi générer des nombres sur N bits très probablement premiers en faisant :

Générer un nombre aléatoire sur N bits

Mettre le premier bit à 1 (pour avoir la bonne taille)

Mettre le dernier bit à 1 (pour avoir un nombre impair)

Tester par prudence la divisibilité par tous les premiers <256

Faire 5 tests de RabinMiller pour 5 valeurs tests a aléatoires mais relativement petites (pour des raisons de rapidité des calculs)

Si le nombre candidat passe avec succès toutes ces étapes, la probabilité qu'il ne soit pas premier est très faible

Le plus grand nombre premier connu (25/01/2013)

- Un projet informatique de longue haleine appelé Great Internet Mersenne Prime Search (GIMPS : http://www.mersenne.org/) permet de déterminer des grands nombres dits de Mersenne, du nom d'un mathématicien français du XVIIe siècle, dont certains sont premiers. Ce type de nombre, très rare, s'exprime sous la forme 2 puissance p moins 1, avec p également nombre premier. Ces nombres qui nécessiteraient des millions de digits (17425170 dans ce cas) pour être écrits sous forme développée, peuvent cependant être désignés par leur forme plus condensée de nombre de Mersenne :

$$2^{57885161} - 1$$

Nota : il s'agit bien ici d'un nombre *certainement* premier

Factorisation : introduction

- Pour de grands nombres composés de deux facteurs premiers (cas du modulo des clés RSA) la factorisation par des méthodes « naïves » n'est pas envisageable
- Heureusement car factoriser le modulo revient à casser la clé
- Par contre certaines méthodes peuvent permettre dans certains cas d'aboutir beaucoup plus rapidement
- La plus connue et l'une des plus efficace est l'utilisation de l'algorithme p-1 de Pollard décrit ci-après

Algorithme p-1 de Pollard : définitions

- L'algorithme p-1 de Pollard est efficace pour factoriser un entier ayant un facteur premier p tel que p-1 soit suffisamment « smooth » (traduit approximativement par « lisse » ou « friable »)
- Un entier K est dit B-smooth ssi tous ses facteurs premiers sont inférieurs ou égaux à B.
- Tout nombre K est donc K-smooth et n'est pas B-smooth pour B<K que dans les cas où K est premier
- En toute généralité le plus petit B pour lequel K est B-smooth est son plus grand facteur premier.

Algorithme p-1 de Pollard : définitions

- Un entier K est dit B-supersmooth ssi toutes les puissances de nombres premiers qui le divisent sont inférieures ou égales à B.
- Tout nombre K est donc K-supersmooth et n'est pas B-supersmooth pour B<K que dans les cas où K est une puissance d'un nombre premier.
- En toute généralité le plus petit B pour lequel K est B-supersmooth est son plus grand facteur qui est une puissance de nombre premier

Algorithme p-1 de Pollard : définitions

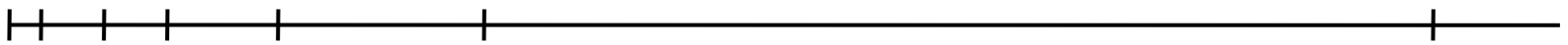
- Exemples : Prenons le nombre 72… quel est le plus petit B pour lequel 72 soit B-smooth ? Même question pour B-supersmooth ?
- Réponses : $72 = 2^3 * 3^2$, par conséquent :
- Le plus grand facteur premier de 72 est 3, donc 72 est 3-smooth
- La plus grande puissance de nombre premier divisant 72 est 9, donc 72 est 9-supersmooth

Algorithme p-1 de Pollard : définitions

B	B-smooth	B-supersmooth
2	2, 4, 8, 16…(2^i)	2
3	Id + 3, 6, 9, 12, 18… (2^i*3^j)	Id + 3, 6
4	Id	Id + 4, 12
5	Id + 5, 10, 15, 20… ($2^i*3^j*5^k$)	Id + 5, 10, 15, 20, 30, 60
6	Id	Id
7	Id+7, 14, 21…($2^i*3^j*5^k*7^l$)	Id + 7, 14, 21, 28, 35, 42
8	Id	Id + 8, 24, 40, 56
9	Id	Id + 9, 18, 36, 45, 63, 72
10	Id	Id

Algorithme p-1 de Pollard : principe

- Si un entier K (dont la décomposition canonique est $K=p^iq^jr^k\ldots$, p, q, r… premiers, i, j, k… entiers) est B-supersmooth, alors le produit B! de tous les nombres inférieurs ou égaux à B contient (par définition) toutes les puissances p^i, q^j r^l de nombres premiers divisant K, par conséquent B! est multiple de K



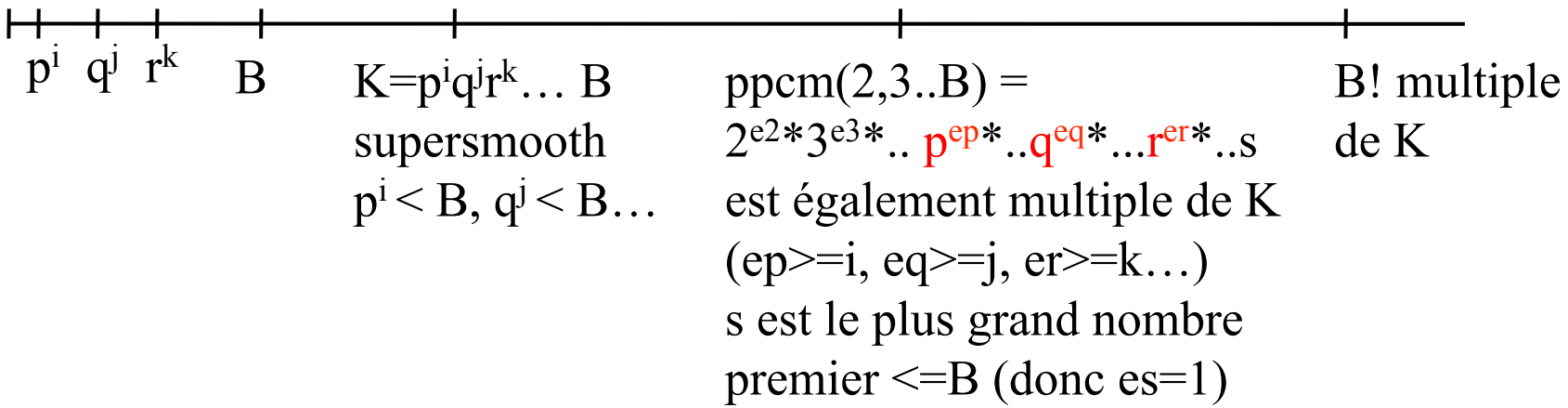
$K=p^iq^jr^k\ldots$ B supersmooth
$p^i < B$, $q^j < B$…

B!=2*3*… B est multiple de K car est le produit de tous les nombres inférieurs ou égaux à B, donc contient tous les facteurs p^i, q^j, r^k…

- Par exemple 72 étant 9-supersmooth, 9!=362880 est multiple de 72 (=5040*72)

Algorithme p-1 de Pollard : principe

- Il en est d'ailleurs de même de ppcm(2,3…B) qui est le produit de toutes les puissances q^{eq} de nombres premiers q compris entre 2 et B, en prenant l'exposant eq le plus grand possible compatible avec le fait que q^{eq} reste inférieur ou égal à B



- Par exemple 72 étant 9-supersmooth, ppcm(2,3,4,5,6,7,8,9) = 8*9*5*7=2520 (produit des plus grandes puissances possibles de 2, 3, 5 et 7) est multiple de 72 (=35*72)

Algorithme p-1 de Pollard : principe

- Ce qui précède montre qu'avec un peu de chance un nombre K peut être B-supersmooth pour une valeur de B beaucoup plus petite que K
- L'algorithme p-1 de Pollard fonctionne si l'un des facteurs p du nombre N à factoriser est tel que p-1 est B-supersmooth pour une valeur de B très petite devant N (sinon la méthode n'a aucun intérêt)
- On va donc faire cette hypothèse pour une valeur de B choisie arbitrairement (en espérant que cela marche, le résultat n'est pas garanti…)

Algorithme p-1 de Pollard : principe

- Soit un entier N dont on suppose sait qu'il est factorisable. Supposons qu'un de ses facteurs premiers p soit tel que p-1 soit B-supersmooth (B choisi arbitrairement) alors :
- ppcm(2,3..B)=k(p-1) d'après ce qui précède
- Par conséquent étant donné un nombre test a (supposé premier avec N donc en particulier non multiple de p mais si cela était le cas on aurait aussi trouvé un facteur de N) :
- $a^{ppcm(2,3..B)} \bmod p = a^{k(p-1)} \bmod p = 1$ (Fermat)
- Autrement dit $a^{ppcm(2,3..B)} - 1 \bmod p = 0$ donc $a^{ppcm(2,3..B)} - 1$ est un nombre divisible par p comme N.
- Ce nombre est peut être grand mais comme N est divisible par p $a^{ppcm(2,3..B)} - 1 \bmod N$ est aussi divisible par p (donc on peut toujours au besoin recaler modulo N dans le calcul de $a^{ppcm(2,3..B)} - 1$)

Algorithme p-1 de Pollard : principe

- On a donc deux nombres : $a^{ppcm(2,3..B)}$ -1 mod N et N tous deux divisibles par p
- Il peut arriver que l'on tombe malencontreusement sur un $a^{ppcm(2,3..B)}$ -1 multiple de N (donc $a^{ppcm(2,3..B)}$ -1 mod N vaut 0) : dans ce cas il faut changer la valeur du nombre test a
- Sinon le calcul de pgcd($a^{ppcm(2,3..B)}$ -1 mod N, N) par l'algorithme d'Euclide, avec un choix judicieux de B peut donc permettre de trouver un facteur premier (si tel n'est pas le cas, on a été trop « gourmand » sur la valeur de B qu'il faut donc augmenter)
- Les considérations qui précèdent sont aussi valables pour pgcd($a^{B!}$ -1 mod N, N), plus grand mais dont l'algorithme de calcul est plus simple (peser le pour et le contre…)
- Exemple : factoriser 91 en supposant qu'un des facteurs est tel que p-1 soit 3-supersmooth (B=3) et avec le nombre test 2 :
- Ppcm(2,3)=3!=6 avec le nombre test 2 : $2^6=[2^2]^3=64$, le calcul du pgcd de 91 et 63 (91[63]=28, 63[28]=7) donne le facteur 7 (91=7*13) ce qui s'explique par le fait que 6 (2*3) est 3-supersmooth (12 est 4-supersmooth)

Algorithme p-1 de Pollard : implantation

- Calcul de ppcm(2,3..B) :
- La plus grande puissance d'un nombre premier p ($p \leq B$) restant inférieure ou égale à B est le plus grand entier ep tel que $p^{ep} \leq B$, c'est donc la partie entière de log(B)/log(p)
- Le ppcm recherché est par conséquent le produit pour tous les nombres premiers p inférieurs ou égaux à B des p^{ep} où ep est la partie entière de log(B)/log(p)
- $a^{ppcm(2,3..B)}$ s'obtient donc par élévation aux puissances successives p^{ep} (où p balaie tous les nombres premiers inférieurs ou égaux à B et ep est la partie entière de log(B)/log(p)), d'une variable initialisée à a, le tout étant en permanence recalé modulo N (algorithme d'exponentiation rapide)

Algorithme p-1 de Pollard : implantation

- Soit le nombre N à factoriser
- Choisir un seuil de friabilité B et un nombre test a entre 2 et N-1
- Calculer (Euclide), D = pgcd(a,N) si D différent de 1, retourner D
- Pour tous les entiers premiers p compris entre 2 et B faire
  - ✓ Calculer ep=partie entière de log(B)/log(p)
  - ✓ Remplacer a par (a puissance p^{ep}) mod N : utiliser l'exponentiation rapide
- Fin faire : la valeur finale de a vaut donc en fonction du a initial : $a^{ppcm(2,3..B)}$ mod N
- Si a-1 vaut 0, changer la valeur de a
- Calculer (Euclide) : D = pgcd(a-1,N)
- Si D=1 augmenter la valeur de B
- Sinon retourner D Fin Si
- Fin

Algorithme p-1 de Pollard : exemple plus conséquent

- Soit le nombre N à factoriser : 194923
- On choisit B=7 et a=2 (2 est bien premier avec 194923)
- ppcm(2,3,4,5,6,7)=4*3*5*7=210 (4,3,5,7 sont les p^{ep} successifs), mais a^{210} mod 194923 n'est pas commode à calculer… on applique donc l'algorithme (en balayant tous les nombres premiers inférieurs ou égaux à 7), a valant initialement 2
  - ✓ La plus grande puissance de 2 inférieure ou égale à 7 est 2^2 (e2=2) donc il faut remplacer 2 par 2^4 mod N =16
  - ✓ La plus grande puissance de 3 inférieure ou égale à 7 est 3 donc il faut remplacer 16 par 16^3 mod N = 4096
  - ✓ La plus grande puissance de 5 inférieure ou égale à 7 est 5 donc il faut remplacer 4096 par 4096^5 mod N calculé par exponentiation rapide :
    - ✓ 4096 mod N = 4096
    - ✓ 4096^2 mod N = 13838
    - ✓ 4096^4 mod N = 13838^2 mod N = 75858
    - ✓ Par conséquent 5 s'écrivant $(101)_2$ 4096^5 mod N =4096*75858 mod N = 7106

Algorithme p-1 de Pollard : exemple plus conséquent

- La plus grande puissance de 7 inférieure ou égale à 7 est 7 donc il faut remplacer 7106 par 7106^7 mod N calculé par exponentiation rapide :
  - 7106 mod N = 7106
  - 7106^2 mod N = 10179
  - 7106^4 mod N = 10179^2 mod N = 107928
  - Par conséquent 7 s'écrivant $(111)_2$ 7106^7 mod N s'obtient par 7106*10179 mod N = 15541 puis par 15541*107928 mod N = 191556 qui vaut donc $a^{ppcm(1,2..7)}$ mod N
- Si donc N a un facteur premier p qui est 7 supersmooth alors p divise également 191555, on calcule donc le pgcd(194923,191555) par Euclide
  - pgcd(194923,191555)=
  - pgcd(191555,3368)=
  - pgcd(3368,2947)=
  - pgcd(2947,421)=421 (le reste suivant est nul) ! Victoire !
- On divise maintenant N par 421 et on obtient 194923=421*463
- Ce sont deux facteurs premiers, dont les p-1 respectifs valent
- $420=2^2*3*5*7$ qui est bien 7-Supersmooth (raison de notre succès)
- 462=2*3*7*11 qui est 11 Supersmooth

Chiffrement par blocs à clé publique : RSA

- Pour l'algorithme RSA (Rivest, Shamir, Adleman), le problème posé par la cryptanalyse est *supposé* de même difficulté que la factorisation
- Dans l'état de l'art actuel, on parvient à factoriser des nombres de 200 digits composés d'exactement deux grands facteurs premiers (nombres RSA) en quelques mois avec des équipes internationales qui se partagent le travail pour assurer un effort global de plusieurs années de CPU d'un processeur des technologies actuelles.

Le record actuel (12/12/2009)

- RSA768 (768 bits 232 digits) =
  123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985690214341 3.

- Facteurs =
  - 33478071698956898786044169848212690817704794983713768568912431388982883793878002287614711652531743087737814467999489
  - 36746043666799590428244633799627952632279158164343087642676032283815739666511279233373417143396810270092798736308917
  - A nécessité environ 2000 ans de calculs avec un AMD Opteron mono-cœur à 2,2 GHz

Le challenge actuel

- RSA-240
- Nombre de digits : 240
- Digits :
  124620366781718784065835044608106590434820374651678805754818788883289666801188210855036039570272508747509864768438458621054865537970253930571891217684318286362846948405301614416430468066875699415246993185704183030512549594371372159029236099
- Nota : des nombres RSA plus petits que RSA-768 résistent encore toutefois…
- Nota : des nombres plus grands mais qui ne sont pas des nombres RSA ont été factorisés depuis (actuellement 2^{1061}-1, nombre de 320 digits factorisé en Août 2012 à l'Université de Californie Fullerton)

Factorisation : exemple des cartes bancaires

- La clé publique RSA du GIE cartes bancaires qui sert à faire la signature numérique était jusqu'à récemment e=3 (la puce d'une CB calcule donc le cube d'un grand nombre) et
  n=21359870359209100823950227049996287970510953418264174064425241650085839577464450884050094308659 99
  (nombre sur 320 bits soit 96 digits)
- n fut factorisé et le résultat publié sur un forum :
  p=1113954325148827987925490175477024844070922844843
  q=191748170252450443937578626823086218069693418929 3
- Il ne restait plus qu'à calculer (p-1)(q-1) puis l'inverse de e modulo (p-1)(q-1) pour trouver d, la clé privée
- Cette faille fut l'un des ingrédients de la fabrication des Yescards dites « méthode Humpich »

Factorisation : exemple des cartes bancaires

- La clé publique du GIE est désormais e=3 et n=155088080278376929842392150075130787847102021520671110279311199011387539455345999975760530467173585609159755538979740893817334404367470478098639006990667909672893308140504493596951450867623994249344075058927001573996237452936325182 7 (nombre sur 768 bits soit 231 digits).
- Bien que non cassée à ce jour, il n'est pas à exclure que cette clé puisse l'être dans un futur relativement proche, au vu des progrès technologiques.