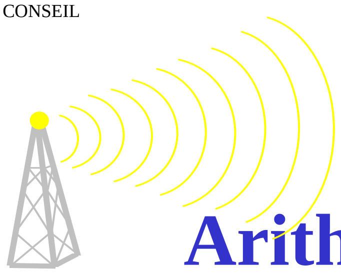
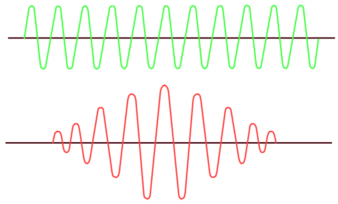
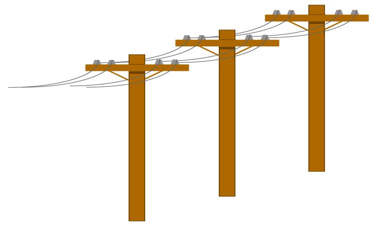






Arithmétique Modulaire

Cours de

Bases mathématiques pour la sécurité informatique

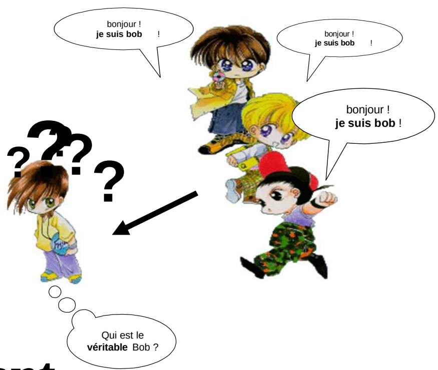
Jean-Luc Stehlé

EPITA

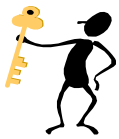
30 mai 2013

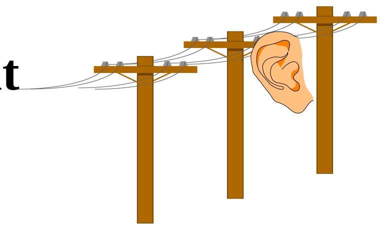
Jean-Luc.Stehle@NormaleSup.org

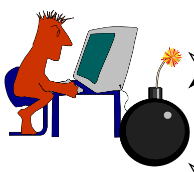
Révisions



- Les éléments de la sécurité
  - ➢Authentification
  - ➢Partage d'une clé de chiffrement
  - ➢Chiffrement/déchiffrement
  - ➢Scellement
  - ➢Non répudiation et signature électronique







JLS CONSEIL

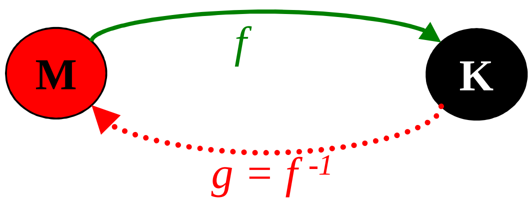
Révisions

- **Deux type de chiffrement**
  - **Chiffrement symétrique**
    - **Même clé pour chiffrer et pour déchiffrer**
    - **Vigénère, DES, AES**
  - **Chiffrement asymétrique**
    - **Clé de chiffrement ≠ Clé de déchiffrement**
    - **Systèmes à clé publique / clé privée**
      - Authentification,
      - Non répudiation,
      - Signature électronique

JLS CONSEIL

Notion de fonction à sens unique

- **Fonction à sens unique**

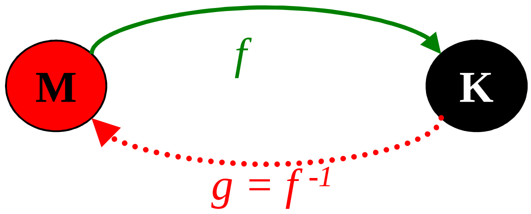


- **Exemple :**
  - **Exponentiation modulaire**
  - **Logarithme discret**

JLS CONSEIL

Fonction à sens unique avec clé

- Le calcul inverse est facile si on connaît la clé



  - Exemple : Système de chiffrement asymétrique

JLS CONSEIL

Arithmétique modulaire

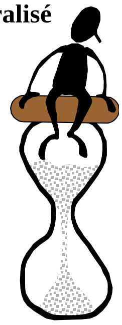
- De nombreux systèmes cryptographiques sont basés sur l'arithmétique modulaire
  - Logarithme discret
  - *Théorème de Bezout*
  - Théorèmes de Fermat et d'Euler
  - RSA
    - Propriétés des nombres premiers
    - Tests de primalité
- Développer des algorithmes efficaces en arithmétique modulaire

Arithmétique modulo N

JLS CONSEIL

N est très grand : 1024 bits ($\cong 10^{300}$) , 2048, …, 4096 bits

- **Addition** — **Facile (Temps en Log N)**
- **Multiplication** — **Facile mais plus long (Log^2 N)**
- **Division** — **Plus difficile (Log^3 N)** avec Bezout/Euclide généralisé
- **Réduction modulo N** — **Il existe des algorithmes de complexité équivalente à celle de la multiplication**
- **Puissance a^b** — **Facile (Log^3 N)** ***(écrire b en binaire)***



- **Problème du logarithme Discret**
- ***Pas d'algorithme rapide***

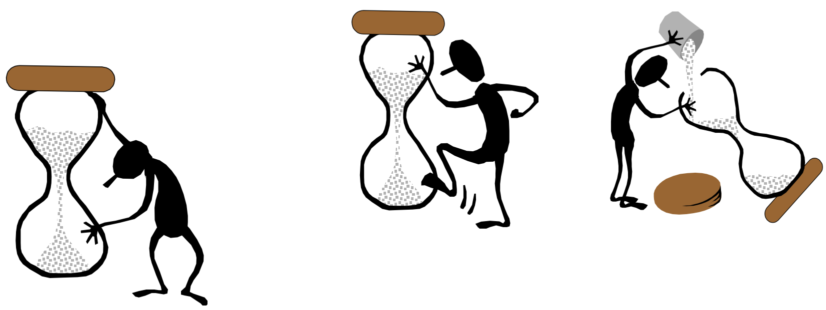
$a^x \equiv b\ [N]$?

Problème du logarithme Discret

JLS CONSEIL

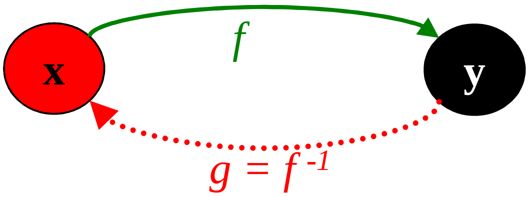
- $a^x \equiv b\ [N]$

- **Pour N grand (10^{300}) , le calcul de x connaissant a et b nécessite un temps supérieur à l'âge de l'univers**



JLS CONSEIL

L'exponentiation modulaire est une fonction à sens unique



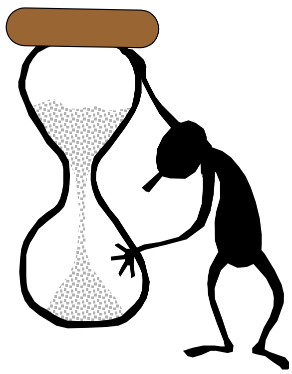
a et N sont connus et publics

- **y = f(x) = a^x (Mod N)**
- **x = g(y) est la solution, en arithmétique modulo N de l'équation $a^x = y$ (Logarithme discret en base a)**

JLS CONSEIL

Complexité algorithmique

Temps de calcul en fonction du nombre de bits de N



- **Algorithmes polynomiaux**
  - **Linéaires (addition)**
  - **Quadratiques (multiplication)**
  - **Cubiques (exponentiation)**
- **Algorithmes exponentiels**
  - **Casser le Log discret par attaque brutale**
    (essayer successivement tous les exposants)
- **Algorithmes subexponentiels**
  - **$L_n(\gamma,c) = O(\exp(c\ \ n^{\gamma}\ \ \ln(n)^{1-\gamma}))$**
  - **γ=1 : exponentiels** **γ=0 : polynomiaux**

Log discret

γ=1/2 et depuis les théories du crible numérique de Lenstra (1993) γ=1/3

Applications du Logarithme Discret

JLS CONSEIL

- **Il est impossible d'inverser en un temps raisonnable l'exponentiation modulaire**
  - **Protocoles d'échanges de clés (Diffie Hellman)**
  - **Chiffrement asymétrique (El Gamal)**
  - **Protocoles d'authentification**

Protocole de Défi / Réponse

JLS CONSEIL

- **Pour authentifier A, on lui envoie un défi**
- **Seul quelqu'un connaissant la clé secrète de A peut calculer rapidement la réponse**
  - Le calcul de la réponse nécessite une exponentiation modulaire
- **Un pirate ne connaissant pas la clé secrète doit faire un calcul très long**
  - Il faut résoudre le logarithme discret
- **A dispose d'une puissance de calcul limitée**
  - **carte à puce, carte SIM de téléphone mobile**
- **Le pirate dispose de moyens très importants**
  - **organisation criminelle puissante**

Bonnes fonctions à sens unique ?

JLS CONSEIL

- **C'est un des enjeux des recherches actuelles**
  - **Fonction directe rapide à calculer sur des processeurs à très faible puissance**
  - **Fonction inverse impossible à calculer même avec des ressources puissantes**

- **En tenant compte des possibles évolutions de la technologie et des puissances de calcul**

- ***Utilisation de courbes elliptiques sur un corps fini***
- ***Groupes de Jacobi des courbes hyperelliptiques***

Bases mathématiques : Arithmétique dans $\mathbb{Z}$

JLS CONSEIL

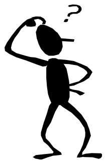
Division euclidienne

a = b q + r 0 ≤ r < b

q = a DIV b

r = a MOD b

a MOD b = a – (b × a DIV b)



L'écriture de programmes DIV et MOD en grands nombres n'est pas élémentaire

Temps de calcul proportionnel au carré du nombre de bits

Bases mathématiques : Arithmétique dans Z

JLS CONSEIL

Notion d'Idéal

$$\left.\begin{array}{l} a \in \mathcal{I} \\ b \in \mathcal{I} \\ \lambda \in \mathbb{Z} \end{array}\right\} \Rightarrow \begin{array}{l} a + b \in \mathcal{I} \\ \lambda a \in \mathcal{I} \end{array}$$

- **Dans Z, tout idéal est principal, c'est-à-dire égal à l'ensemble des multiples d'un d unique**

- **Démonstration : Division euclidienne et notion d'ordre**

Arithmétique dans Z : Le pgcd

JLS CONSEIL

- **Z : Ensemble des entiers relatifs. Muni de + et de ×, Z est un anneau.**
- **Un idéal $\mathcal{I}$ dans Z est un sous-ensemble stable par + et par $\times \lambda$ avec $\lambda \in Z$**
- **Dans Z, tout idéal $\mathcal{I}$ est principal, c'est-à-dire formé des multiples d'un élément d $\in \mathcal{I}$**
- **Démonstration : par la division euclidienne :**
  - Soit d le plus petit élément positif dans $\mathcal{I}$
  - Pour tout x positif dans $\mathcal{I}$, on peut faire une division euclidienne x=dq+r avec r<d.
  - On a r$\in\mathcal{I}$, r<d donc r=0, donc x=dq. Pour y<0, même raisonnement avec x=-y $\in\mathcal{I}$
- **Exemple d'idéal : pour a,b$\in$ Z : $\mathcal{A}$(a,b)={λa+μb; $\lambda,\mu\in$ Z}**
  - **$\exists$d$\in\mathcal{A}$(a,b) tel que tout élément de $\mathcal{A}$(a,b) soit multiple de d**
    - **d divise a et b.**
    - **$\exists\,\lambda\,\mu$ tels que d= λa+μb**
    - **Tout diviseur commun de a et b divise d**

d = pgcd(a,b)

JLS CONSEIL

Arithmétique dans Z : Th de Bezout

- **a et b premiers entre eux : pgcd(a,b)=1**

- **Théorème de Bezout :**
  Il existe λ μ tels que $\lambda a+\mu b = 1$

- **Si d divise a et b, d divise 1, donc d=1**
- **$\mathscr{I}$(a,b)={λa+μb; λ, $\mu \in$ Z} = Z tout entier**

Attention : λ et μ ne sont pas uniques

$\lambda \rightarrow \lambda + k\,\mathbf{b}$

$\mu \rightarrow \mu - k\,\mathbf{a}$

JLS CONSEIL

Bases mathématiques : Arithmétique dans Z

Idéal maximal

- **Tout idéal strictement plus grand est égal à $\mathbb{Z}$ tout entier**
- **Égal à l'ensemble des multiples d'un d unique**
- **d n'a pas de diviseurs autres que lui-même et l'unité**

$\Rightarrow$ d premier !

Arithmétique modulaire

JLS CONSEIL

Quotient d'un anneau par un idéal

- $N \in Z$
- $a \cong b \quad \Leftrightarrow \quad$ a-b est multiple de N

  $\Leftrightarrow \quad a\text{-}b \in \mathcal{I}(N) = NZ$

- On note Z/NZ

  l'ensemble des classes d'équivalence

- Un élément de Z/NZ peut être représenté par un entier entre 0 et N-1
- Z/NZ a N éléments
- Z/NZ est un anneau;
- Z/NZ est un corps si et seulement si N est premier

Bases mathématiques : La fonction d'Euler

JLS CONSEIL

- **Théorème :** a est premier à N si et seulement s'il est inversible dans $Z/_{NZ}$
  - Bezout : a inversible : $\exists b$ vérifiant $ab+\lambda N=1$
- **Définition :** U_N est le groupe multiplicatif des éléments inversibles de $Z/_{NZ}$
- **Définition :** $\Phi(N)$: nombre d'éléments (cardinal) de U_N.

  = *nombre d'entiers inférieurs à N et premiers à N .*

Exemple $U_{12} = \{ 1 , 5 , 7 , 11 \} \quad \Phi(12) = 4$

Attention : ici tous les éléments sont racines carrées de l'unité

Bases mathématiques : La fonction d'Euler

JLS CONSEIL

$\Phi(N)$ = Indicateur d'Euler de N
$\Phi(N)$ = nombre d'entiers compris entre 1 et N-1 et premiers à N
$\Phi(N)$ = nombre d'éléments du groupe multiplicatif U_N
des éléments inversibles dans Z /NZ

$N = p$ premier : $\Phi(p) = p-1$
$N = p^{\alpha}$, p premier : $\Phi(p^{\alpha}) = p^{\alpha}-p^{\alpha-1} = p^{\alpha-1}(p-1)$

__Cas général :__ $N = \prod p_i^{\alpha_i}$, p_i premiers

$$\Phi(\prod p_i^{\alpha_i}) = \prod (p_i^{\alpha_i} - p_i^{\alpha_i-1}) = \prod p_i^{\alpha_i-1}(p_i - 1) = N \prod(1-1/p_i)$$

Se démontre par le théorème des restes chinois
- __Exemple__ 12 = 3.2.2 $\Phi(12) = (3-1).2.(2-1) = 4$

Bases mathématiques : Le théorème des restes chinois

JLS CONSEIL

- $n = n_1 n_2 \dots n_k$ avec n_j premiers entre eux deux à deux

- $a \in Z/nZ \leftrightarrow (a_1, a_2,\dots,a_k) \in \prod Z/n_jZ$ avec $a_j = a \bmod n_j$
  est une bijection
  compatible avec les structures de groupes additif et multiplicatif

$a \leftrightarrow (a_1, a_2,\dots,a_k)$ $b \leftrightarrow (b_1, b_2,\dots,b_k)$ $\Rightarrow$	$a+b \leftrightarrow (a_1+b_1, a_2+b_2,\dots,a_k+b_k)$ $a.b \leftrightarrow (a_1.b_1, a_2.b_2,\dots,a_k.b_k)$

- Détermination de la fonction réciproque *(se ramène à Bezout pour k=2)*
  - $m_i = n / n_i$ (donc $m_i \equiv 0$ [mod n_j] pour $j \neq i$)
  - $c_i = m_i (m_i^{-1} \bmod n_i)$ (car m_i et n_i premiers entre eux)
  - $c_i = 1 \bmod n_i$ $c_i = 0 \bmod n_j$ pour $j \neq i$
  - $a = (a_1c_1 + a_2c_2 + \dots + a_kc_k) \bmod n$

- La famille d'équations $x \equiv a_i$ [mod n_i] a une solution unique modulo n
- $x \equiv a$ [mod n_i] $\forall i$ $\Leftrightarrow$ $x \equiv a$ [mod n]

JLS CONSEIL

Bases mathématiques :

Le théorème des restes chinois

Cas particulier : k=2

- $n = n_1 n_2$ avec n_1 et n_2 premiers entre eux
- Par Bezout, il existe λ μ vérifiant $\lambda n_1 + \mu n_2 = 1$
- Etant donnés a_1 et a_2 en posant $a = \lambda n_1 a_2 + \mu n_2 a_1$ on aura
  $a \bmod n_1 = a_1$ et $a \bmod n_2 = a_2$

JLS CONSEIL

Bases mathématiques :

Le théorème des restes chinois

Applications

☞ **Le problème de Sun-Tsu (Mesure des champs)**

☞ **Le partage du sac de pièces d'or par les pirates**

☞ **Le calcul de $\Phi(\prod p_i^{\alpha_i})$**

- ☞ Si a et b sont premiers entre eux, alors $\Phi(ab)=\Phi(a) \times \Phi(b)$

JLS CONSEIL

Bases mathématiques : Le théorème d'Euler

- **Pour $a \in U_n$, on a $a^{\Phi(n)} \equiv 1$ [mod n]**

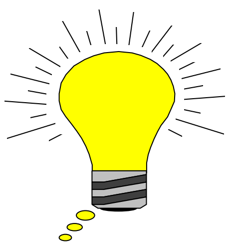
- **Démonstration**

  - Les puissances successives de a dans U_n : $1\ a\ a^2\ a^3 \ldots\ a^n\ \ldots\ a^\nu \equiv 1$ [mod n]
    forment un sous-groupe multiplicatif de U_n
    formé de ν éléments, où ν est le premier exposant vérifiant $a^\nu \equiv 1$ [mod n]
  - **L'ordre ν du sous-groupe divise l'ordre $\Phi(n)$ du groupe.**
    Donc $a^{\Phi(n)} \equiv 1$ [mod n]
  - Pour $x,y \in U_n$ les ensembles
    $\{x, xa, xa^2, xa^3, \ldots, xa^n \ldots, xa^{\nu-1}\}$
    $\{y, ya, ya^2, ya^3, \ldots, ya^n \ldots, ya^{\nu-1}\}$
    sont disjoints ou confondus
  - On réalise ainsi une partition de U_n en sous-ensembles ayant tous ν éléments

JLS CONSEIL

Bases mathématiques : Le théorème d'Euler

Cas particulier : n = p premier



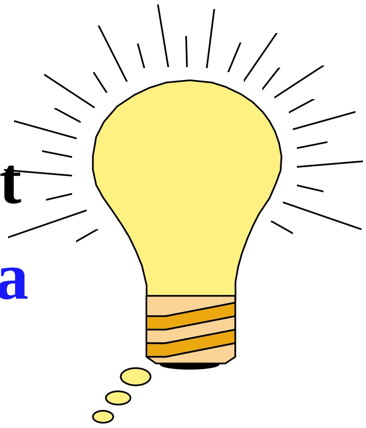
- **$\Phi(p) = p-1$**

- **Pour a non multiple de p $a^{p-1} \equiv 1$ [mod p]**

- **C'est le « petit » théorème de Fermat**

JLS CONSEIL

Théorème de Fermat



« Petit » théorème de Fermat

Pour p **premier**, $a \neq 0$ [p] **, on a**

$$a^{p-1} \equiv 1 \ [p]$$

JLS CONSEIL

Application du théorème de Fermat

- **Pour** c **premier à** p-1, **on calcule** d **inverse de** c **modulo** p-1 **(Bezout)**
- **Pour** $cd \equiv 1 \ [p-1]$ **on a, pour tout a** (y compris 0 [p])

  $$(a^c)^d \equiv (a^d)^c \equiv a \ \ [p]$$

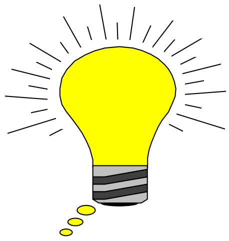
- ***Deux exponentiations modulaires réciproques l'une de l'autre***

$$\begin{tikzcd} x \arrow[r, bend left, "f"] & y \arrow[l, bend left, dotted, "g = f^{-1}"] \end{tikzcd}$$

f : élever à la puissance c

g : élever à la puissance d

Bases mathématiques :
Le théorème d'Euler



- **Cas particulier: théorème de Fermat :**
  - **Pour $a \neq 0$ [p] , on a $a^{p-1} \equiv 1$ [p] donc $a^p \equiv a$ [p]**
  - **Pour $a \equiv 0$ [p] , on a de toutes façons $a^p \equiv 0$ [p]**
  - **On a donc toujours $a^{\Phi(p)+1} \equiv a$ [p] et de même $\forall\lambda$ $a^{\lambda\Phi(p)+1} \equiv a$ [p]**

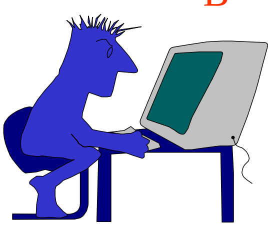
- **Cas général : théorème d'Euler :**
  - Pour $a \in U_n$, on a $a^{\Phi(n)} \equiv 1$ [n]

Mais on ne peut pas en conclure que pour tout $a \in (\mathbb{Z}/_{n\mathbb{Z}})$ $a^{\Phi(n)+1} \equiv a$ [n]

- Contre exemple : Tous les éléments de $U_{12} = \{ 1 , 5 , 7 , 11 \}$ sont racines carrées de l'unité donc leur puissance 5ième est égale à eux mêmes. Mais la suite des puissances successives de 2 ($\notin U_{12}$) est 2, 4, 8, 4, 8, 4, 8, 4, 8, etc., et on ne retrouve jamais 2.

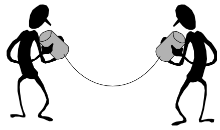
Système de Massey - Omura
Initialisation

- **p premier public très grand (> 10^{120})**
- **A choisit c_A et d_A secrets avec $c_A.d_A \equiv 1$ [p-1]**
- **B choisit c_B et d_B secrets avec $c_B.d_B \equiv 1$ [p-1]**



Système de Massey - Omura

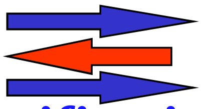
Transmission d'un message M



- A calcule et transmet M^{c_A} ,
- B l'élève à la puissance c_B et renvoie $M^{c_A c_B}$
- A l'élève à la puissance d_A obtient $M^{c_A c_B d_A} \equiv M^{c_B}$ qu'il transmet
- élève à la puissance d_B et retrouve M

N.B. : Tous les calculs sont modulo p

- Trois échanges



- nécessite une *authentification préalable*
- Protocole de valise à deux cadenas

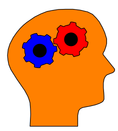
Un cas particulier

- n= p.q, p et q premiers distincts, $\Phi(n) = \Phi(pq) = (p-1)(q-1)$

a premier à p $\Rightarrow a^{p-1} \equiv 1\ [p] \Rightarrow a^{(p-1)(q-1)+1} \equiv a\ [p]$

a multiple de p $\Rightarrow a \equiv 0\ [p] \Rightarrow a^{\text{n'importe quoi}} \equiv a\ [p]$

- Donc pour tout a, $a^{(p-1)(q-1)+1} \equiv a\ [p]$
- De même $a^{\lambda.(p-1)(q-1)+1} \equiv a\ [p]$, $\forall \lambda \in Z$
- Même raisonnement modulo q,
- p et q étant premiers entre eux, une égalité vraie modulo p et modulo q reste vraie modulo pq.



$$a^{\lambda.\Phi(n)+1} \equiv a\ [n]$$

- Même raisonnement dès que n est le produit d'une famille de nombres premiers tous distincts (donc n'a pas de facteur carré)

JLS CONSEIL

Théorème : Si n n'a pas de facteur carré, alors, $\forall\lambda \in Z$, on a $a^{\lambda.\Phi(n)+1} \equiv a\ [n]$

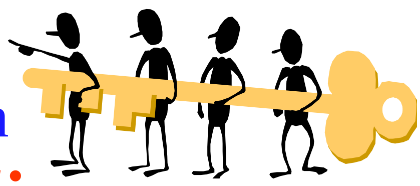
- **Corollaire :** **Si α et β vérifient $\alpha.\beta \equiv 1\ [\Phi(n)]$ alors, pour tout a , on a $a^{\alpha\beta} \equiv a\ [n]$**
- **Applications n=pq p,q très grands (10^{100})**
  - * Connaissant n il est très difficile de calculer p et q **(problème de la factorisation des grands nombres)** ainsi que $\Phi(n) = \Phi(pq) = (p-1)(q-1)$
  - * Connaissant α il est très difficile de calculer β si on ne connaît pas p et q
    Si on les connaît, c'est immédiat (Bezout)
  - * Il est facile de générer des p et q très grands
  - * Si on publie n et α il est très difficile de retrouver p, q et β.

Quelques contraintes sur le choix de p et q

➢ **(Très difficile = temps > âge de l'univers)**

JLS CONSEIL

Le système RSA
Rivest Shamir Adleman
Système à clé publique

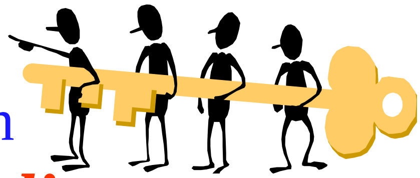


- **Chaque utilisateur choisit** n, α **et** β **vérifiant** $\alpha.\beta \equiv 1\ [\Phi(n)]$ **et publie** n **et** α **et garde le reste confidentiel.**
- **α est la clé publique, β est la clé secrète.**
- **codage public : M** $\rightarrow\rightarrow\rightarrow$ **$\mu \equiv M^{\alpha}$ [n]**
- **décodage privé : μ** $\rightarrow\rightarrow\rightarrow$ **M $\equiv \mu^{\beta}$ [n]**

Similaire à Massey Omura, mais l'un des exposants est public et ici on ne peut pas calculer l'autre

JLS CONSEIL

Le système RSA



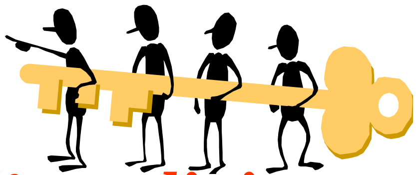
Rivest Shamir Adleman

Système à clés publiques

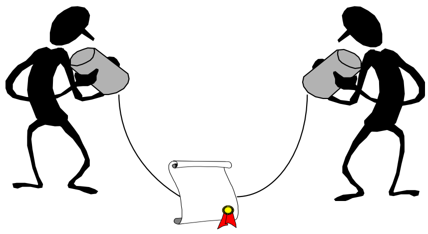
- **Tout le monde peut envoyer un message secret que seul le destinataire saura décoder.**
- **On peut authentifier un message en codant un checksum avec la clé secrète de l'émetteur**
  - ➢ **Scellement, non répudiation**
- **On peut authentifier une liaison**

JLS CONSEIL

Le système RSA



Authentification d'une liaison



➢ ***Publics :***

➢ n_A , α_A f_A : $M \rightarrow\rightarrow M^{\alpha_A}$ $[n_A]$

➢ n_B , α_B f_B : $M \rightarrow\rightarrow M^{\alpha_B}$ $[n_B]$

➢ ***Connu de A seul :*** β_A g_A : $M \rightarrow\rightarrow M^{\beta_A}$ $[n_A]$

➢ ***Connu de B seul :*** β_B g_B : $M \rightarrow\rightarrow M^{\beta_B}$ $[n_B]$

- **A génère un nombre aléatoire ξ et l'envoie à B**
- **B applique g_B et renvoie $g_B(\xi)$ à A**
- **A vérifie que $f_B(g_B(\xi))$ redonne bien ξ , ce qui authentifie B. Il applique g_A et envoie $f_B(g_B(\xi))$ à B.**
- **B vérifie que f_A de ce qu'il a reçu redonne bien $g_B(\xi)$ ce qui authentifie A.**

JLS CONSEIL

RSA : Comment l'installer ?

- **Routines de calcul en arithmétique modulaire**
  - **Addition, Multiplication**
  - ***Division [mod $\Phi(n)$] pour le calcul des clés)***
  - **Réduction modulo N**
    - **Algorithme « de proche en proche »**
    - **Méthode de Montgomery**
  - **Exponentiation**

- **Recherche de nombres premiers**
  - **Générateurs aléatoires**
  - **Tests de primalité**

- **Peut on casser le RSA ?**
  - **Précautions à prendre sur les nombres premiers**
  - **Casser le RSA par le Log Discret**
  - **Casser le RSA par factorisation**

Il existe des algorithmes en $\gamma = 1/3$

JLS CONSEIL

Attaques sur RSA

- **Rappels mathématiques**
  - **p, q premiers secrets très grands N = pq public**
  - **α secret β public avec $\alpha\beta \equiv 1$ [mod (p-1)(q-1)]**
  - **M $\rightarrow$ (M^α) [mod N] $\rightarrow$ ((M^α) $^\beta$) [mod N]**
  - **M $\rightarrow$ (M^β) [mod N] $\rightarrow$ ((M^β) $^\alpha$) [mod N]**

- **Attaque par Log discret**
  - **Retrouve l'exposant secret sans factoriser**

- **Attaque par factorisation**
  - **Faiblesse si p ou q ou (p-q) petits**

Il existe des algorithmes en $\gamma = 1/3$

JLS CONSEIL

Factorisation : n=pq

- **Si l'un des facteurs est petit : Essais successifs**

- **Si les facteurs sont voisins : Test de Fermat**

On pose $a = (p+q)/2$ $b = (p-q)/2$ $r = \lfloor\sqrt{n}\rfloor$

Donc $p = (a-b)$ $q = (a+b)$ $n = a^2-b^2 = (a+b)(a-b)$ $a \geq r+1$

$a(t) = r+t$ $f(t) = a(t)^2-n = t^2+2tr+r^2-n$

On calcule donc f(t) pour t=1,2,3,… : fonction croissante de t, *incrément 2t+2r+1*

jusqu'à obtenir un carré parfait b^2 pour $t=t_0$.

On écrit $f(t_0) = b^2 = a^2-n$, d'où la factorisation

Temps de calcul proportionnel à b donc à p-q

JLS CONSEIL

Recommandations DCSSI pour l'utilisation du Log Discret

1. Dans GF(p), p premier

- **Niveau standard**
  - **< 2010 :** **p : Minimum 1536 bits**
    Sous groupe engendré par g : ordre multiple d'un premier à au moins 160 bits
  - **< 2020 :** **p : Minimum 2048 bits**
    Sous groupe engendré par g : ordre multiple d'un premier à au moins 256 bits

- **Niveau Renforcé**
  - **< 2010 :** **p : Minimum 2048 bits**
    Sous groupe engendré par g : ordre multiple d'un premier à au moins 256 bits
  - **< 2020 :** **p : Minimum 4096 bits**
    Sous groupe engendré par g : ordre multiple d'un premier à au moins 256 bits

Recommandations DCSSI pour l'utilisation du Log Discret

2. Dans $GF(2^n)$

- **Niveau standard**
  - < 2010 : n > 2048 bits
    Sous groupe engendré par g : ordre multiple d'un premier à au moins 160 bits
  - < 2020 : n > 2048 bits
    Sous groupe engendré par g : ordre multiple d'un premier à au moins 256 bits

Mais il est recommandé d'utiliser plutôt GF(p), plus sûr à taille de clé égale

- **Niveau Renforcé**
  - Déconseillé

Recommandations DCSSI pour l'utilisation de RSA

Problème de la factorisation

- **Niveau standard**
  - < 2010 : Modules > 1536 bits ; 2048 bits conseillés
  - < 2020 : Modules > 2048 bits
  - Exposant secret de même ordre de grandeur que module
  - Exposant public $> 2^{16}$

N.B. : $2^{16} + 1$ = x10001 est un exposant public souvent utilisé

- **Niveau Renforcé**
  - < 2010 : Modules > 2048 bits
  - < 2020 : Modules > 4096 bits
  - Exposant secret de même ordre de grandeur que module

JLS CONSEIL

Arithmétique modulaire

- **Arithmétique modulo N avec N très grand, nombre à n bits**
- **On travaille toujours en entiers non signés**
- **On pose R = 2^n**
  - **n : première puissance de 2 supérieure à N**
  - **R : premier nombre qui ne puisse pas s'écrire sur n bits**
- **$\delta = R - N \leq R/2$ Remarquer $R \equiv \delta$ [mod N]**
  - **On a intérêt à prendre δ petit**
  - **Exemple des groupes d'Oakley**

    On commence par 64 bits à 1 $\Rightarrow \delta < 2^{-64}$ R

JLS CONSEIL

Arithmétique modulaire

- **Il est souhaitable que le nombre de bits n soit un multiple du nombre de bits du processeur**
- **Exemple : processeur à 32 bits**
  - ➢ **Une multiplication élémentaire :**
    - Multiplie deux nombres à 32 bits,
    - Résultat à 64 bits (sur 2 registres pouvant fonctionner en accumulateur)
  - ➢ **Réalisée en une instruction machine**

$n=32\,q$ $\quad B = 2^{32}$

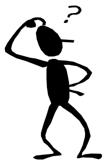
On travaille sur une arithmétique en base B

$$x = x_0 + x_1 B + x_2 B^2 + x_3 B^3 + \ldots + x_{q-1} B^{q-1}$$

Arithmétique modulaire

JLS CONSEIL

- **Représentation machine du grand nombre x : q mots (par exemple entiers 32 bits non signés)**
  - 1024 bits : q=32
  - 2048 bits : q=64



- **Une grande multiplication dans Z requiert q^2 multiplications élémentaires (plus les contrôles de boucles, calculs d'indices,…) et il faudra ensuite faire la réduction modulo N**

On écrit $x = (x_0, x_1, x_2, x_3, \ldots, x_{q-1})_B$

Attention à l'ordre des mots (Big endian, Little endian)

Bases mathématiques : Arithmétique dans Z

JLS CONSEIL

Calcul du pgcd : Algorithme d'Euclide

$\mathcal{I} = \mathcal{I}(a,b) = \{ \lambda a + \mu b , \ \lambda, \mu \in Z \} \ (a>b)$

$$
\begin{array}{lll}
a = b\, q_1 + r_1 & 0 \le r_1 < b & r_1 \in \mathcal{I} \\
b = r_1\, q_2 + r_2 & 0 \le r_2 < r_1 & r_2 \in \mathcal{I} \\
r_1 = r_2\, q_3 + r_3 & 0 \le r_3 < r_2 & r_3 \in \mathcal{I} \\
\ldots \ \ldots \ \ldots & & \\
r_{n-1} = r_n\, q_{n+1} + 0 & &
\end{array}
$$

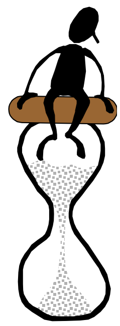
a, b et tous les r sont multiples de r_n et $r_n \in \mathcal{I}$ donc $r_n = \mathrm{pgcd}(a,b)$

Programme récursif

```
Euclide(a,b)
Si b=0
   Alors retourner a
   Sinon retourner Euclide(b,a mod b)
```

JLS CONSEIL

Bases mathématiques : Arithmétique dans Z



Calcul du pgcd : Algorithme d'Euclide

Estimation du temps de calcul

Suite de Fibonacci $F_0=0$; $F_1=1$; $F_k = F_{k-1} + F_{k-2}$ 0,1, 1, 2, 3, 5, 8, 13, 21, 34, 55

Nombre d'or : $\Phi = (1+\sqrt{5})/2 = 1.61803\ldots$ $\Phi^{-1} = (-1+\sqrt{5})/2 = \Phi - 1 = 0.61803\ldots$ $\Phi^2 - \Phi - 1 = 0$

$F_k = (\Phi^k - (-1)^k \Phi^{-k}) / \sqrt{5} \cong \Phi^k / \sqrt{5}$ à démontrer par récurrence sur k

Théorème : Si Euclide nécessite plus de k appels récursifs, alors $a \geq F_{k+2}$ et $b \geq F_{k+1}$.

Démonstration : par récurrence sur k
Supposons $b \geq F_{k+1}$ et $(a \bmod b) \geq F_k$
Or $b+(a \text{ MOD } b) = b+a-(a \text{ DIV } b)\, b \leq a$
Donc $a \geq F_k + F_{k+1}$

Corollaire : $b \leq F_{k+1}$ et $a>b>0$ alors il faut moins de k appels récursifs

Si $b \sim \Phi^{k+1} / \sqrt{5}$ alors moins de k appels. Le cas le pire est (F_{k+2}, F_{k+1}) qui nécessite exactement k appels

$b\sqrt{5}/\Phi \sim \Phi^k$

Nombre d'appels < $(\text{Log}_2\, b + \text{Log}_2(\sqrt{5}) - \text{Log}_2(\Phi))/\text{Log}_2(\Phi) \sim 1.44\ \text{Log}_2\, b + 0.67$

JLS CONSEIL

Bases mathématiques : Arithmétique dans Z

Calcul de l'inverse d'un nombre modulo N

Trouver un x vérifiant $a\,x + \mu\, N = 1$

$\exists\, x \Leftrightarrow$ pgcd(a,N) = 1 $\Leftrightarrow$ a premier à N

Φ(N) = Indicateur d'Euler de N
Φ(N) = Nombre d'entiers compris entre 1 et N-1 et premiers à N
Φ(N) = Nombre d'éléments du groupe multiplicatif U_N des éléments inversibles dans Z /NZ

Algorithme d'Euclide généralisé dans $\mathcal{G} = \mathcal{G}(a,N)$

Algorithme d'Euclide généralisé dans $\mathcal{G} = \mathcal{G}(a,N)$

Phase 1 : On descend

$N = a\,q_1 + r_1$ $\quad 0 \le r_1 < a$ $\quad r_1 \in \mathcal{G}$

$a = r_1\,q_2 + r_2$ $\quad 0 \le r_2 < r_1$ $\quad r_2 \in \mathcal{G}$

$r_1 = r_2\,q_3 + r_3$ $\quad 0 \le r_3 < r_2$ $\quad r_3 \in \mathcal{G}$

……

$r_{n-3} = r_{n-2}\,q_{n-1} + r_{n-1}$ $\quad 0 \le r_{n-1} < r_{n-2}$ $\quad r_{n-1} \in \mathcal{G}$

$r_{n-2} = r_{n-1}\,q_n + r_n$ $\quad 0 \le r_n < r_{n-1}$ $\quad r_n \in \mathcal{G}$

$r_{n-1} = r_n\,q_{n+1} + 0$

Tous les r_j , a et N sont multiples de $r_n \in \mathcal{G}$ donc $r_n = 1$

Algorithme d'Euclide généralisé dans $\mathcal{G} = \mathcal{G}(a,N)$

Phase 2 : On remonte et on applique des multiplicateurs

$r_{n-2} = r_{n-1}\,q_n + r_n \Rightarrow$ $\quad r_{n-2} - r_{n-1}\,q_n = 1$

$r_{n-3} = r_{n-2}\,q_{n-1} + r_{n-1} \Rightarrow$ $\quad r_{n-3} - r_{n-2}\,q_{n-1} - r_{n-1} = 0$ $\quad$ ($\times$ $-q_n$ et addition, tue le terme en r_{n-1})

$r_{n-4} = r_{n-3}\,q_{n-2} + r_{n-2} \Rightarrow$ $\quad r_{n-4} - r_{n-3}\,q_{n-2} - r_{n-2} = 0$ $\quad$ ($\times$ $q_n q_{n-1}$ et addition, tue le terme en r_{n-2})

$r_1 = r_2\,q_3 + r_3 \Rightarrow$ $\quad r_1 - r_2\,q_3 - r_3$ ($\times$ ce qu'il faut puis + pour tuer le terme en r_3)

$a = r_1\,q_2 + r_2 \Rightarrow$ $\quad a - r_1\,q_2 - r_2$ ($\times$ ce qu'il faut puis + pour tuer le terme en r_2)

$N = a\,q_1 + r_1 \Rightarrow$ $\quad N - a\,q_1 - r_1$ ($\times$ ce qu'il faut puis + pour tuer le terme en r_1)

Il reste au final *(un terme en a)* + *(un terme en N)* = 1

Exercice : Écrire un programme récursif pour inverser a

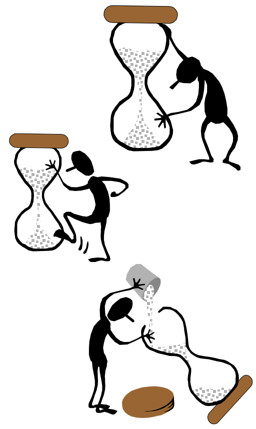
JLS CONSEIL

Bases mathématiques : Arithmétique dans Z

Algorithme d'Euclide généralisé dans $\mathcal{I}=\mathcal{I}(a,N)$

Phase 3 : Écrire cela dans un programme récursif

```
EuclideEtendu(a,b)  renvoie (d,x,y) tel que d = ax + by
Si b=0
   Alors retourner (a,1,0)
   Sinon (d',x',y') := EuclideEtendu(b,a MOD b)
         (d,x,y) := (d',y',x' – (a DIV b) y')
         retourner (d,x,y)
```



Inverser **a** revient à calculer **EuclideEtendu(N,a)**
Borne supérieure du nombre d'appels ~ $1.44 \text{ Log}_2 a + 0.67$

a équiréparti entre 1 et N-1, et pas forcément Fibonacci
Estimation de Knuth : moyenne du nombre d'appel = $0.843 \text{ Log}_2(N) + 1.47$

Nombre d'étapes proportionnel au nombre de bits de N
Temps proportionnel au cube du nombre de bits

JLS CONSEIL

Bases mathématiques : Arithmétique dans Z

Réduction modulo N

- **$R = 2^n$ $N = R-\delta$ avec δ petit**
  - On suppose toujours $\delta < N/2$, et en général on aura $\delta << N/2$
  - Cas des groupes d'Oakley : $\delta \sim 2^{-64}$ N
- **Le nombre de bits n est un multiple entier du nombre de bits du processeur**
  (n bits représentent un nombre entier de mots)
- **$R = 2^n = B^q$ avec $n=32q$**
  un nombre de n bits est stocké sur q mots de 32 bits
- **Un élément de $\mathbb{Z}/N\mathbb{Z}$ est représenté par un grand entier entre 0 et N-1 stocké sur q mots.**
- **Chaque fois qu'un calcul intermédiaire donne un résultat supérieur à N, on réduit modulo N.**

JLS CONSEIL

Bases mathématiques : Arithmétique dans Z

Grande addition de 2 termes

- Équivaut à q additions élémentaires
- Le résultat peut atteindre 2N–2 < 2R–2, donc tient sur n+1 bits. On prévoit un bit de retenue.
- Si x≥N,
  - calculer x–N (toujours <N)
  - x–N = x+δ–R
  - On calcule x+δ (qui est toujours ≥R) et on tue le bit de poids fort.
- Temps de calcul pour une grande addition modulo N
  - Une grande addition (q additions élémentaires) si pas de réduction
  - Deux grandes additions (2q additions élémentaires) si réduction nécessaire,
  - Un peu moins si δ est vraiment très petit
  - Un test (quelques tests élémentaires)

JLS CONSEIL

Bases mathématiques : Arithmétique dans Z

Grande addition de plusieurs termes

Par exemple q additions dans une grande multiplication

- Choix 1 : Se ramener à plusieurs grandes additions de 2 termes
- Choix 2 : Travailler dans une arithmétique à q+1 mots
  - (un mot de retenue)

  - x = a + R b (b est la retenue, inférieure au nombre de termes additionnés)
  - On utilise R ≡ δ [mod N]
  - On calcule a + δ b

  - Avec les hypothèses précédentes a<R, δ<<N, b petit on reste <2N–2
  - et on procède comme précédemment si ça dépasse N.

Bases mathématiques : Arithmétique dans Z

JLS CONSEIL

Grande multiplication

Algorithme de l'école communale $z = x \times y$

q^2 multiplications, $q(q-1)$ additions,

arithmétique à $2q$ mots (+un mot de retenue).

Détail des opérations

$x = \sum_{0 \le i < q} x_i B^i \quad y = \sum_{0 \le i < q} y_i B^i \quad z = \sum_{0 \le i < 2q} z_i B^i$

avec $z_i = \sum_{j+k=i} x_j y_k$ (si x = y, temps divisé approximativement par 2)

x_i , $y_i \le B-1$ donc $z_i \le q\,(B^2 - 2B + 1)$ 2 mots principaux + un mot de retenue

Les résultats sont accumulés dans une suite de $2q+1$ mots

Bases mathématiques : Arithmétique dans Z

JLS CONSEIL

Réduction modulo N

Pour $q \le i \le 2q$ on a calculé d'avance B^i [mod N]

On multiplie par le coefficient et on additionne le résultat

Approximativement q^2 multiplications et additions

temps équivalent à une grande multiplication

Calculs préalables :

$B^q = R \equiv \delta$ [mod N] << N

$B^{q+1} \equiv \delta B$ [mod N]

On additionne les $z_i B^i$ [mod N]

On utilise une arithmétique en $q+1$ mots (car retenues).

Méthode équivalente :

Réduction de proche en proche des mots de poids élevé

Bases mathématiques :

Exponentiation dans un groupe multiplicatif

JLS CONSEIL

```
Constantes : UN    Élément neutre du groupe   (Le nombre 1 en « grand nombre »)
             n     entier simple                  nombre de bits de b
Variables :  J     entier simple
Données   :  A     Élément du groupe          (Grand nombre en arithmétique modulaire)
             B     Exposant                    représenté par son développement binaire {B[i]}
Résultat :   R     Élément du groupe          (Grand nombre en arithmétique modulaire)
Fonction :   Mult  fonction opérant sur des éléments du groupe
                    2 arguments 1 résultat      (Grands nombres en arithmétique modulaire)
```

```
BEGIN
  R := UN
  J := n-1
  WHILE ((B[j]=0) AND (j≥0)) DO j := j-1;
  WHILE j≥0 DO
    BEGIN
      R := Mult(R,R)
                   élévation de R au carré
      IF B[j]=1 THEN R := Mult(R,A)
      j := j-1
    END
                   à la fin de la boucle, R contient a^b
END
```

Algorithme
Square and Mult

Exercice ***: une petite amélioration est encore possible…***

Attention : ***si on travaille dans Z/NZ :***

A est défini modulo N, B est défini modulo Φ(N)

Bases mathématiques :

Arithmétique modulo N

JLS CONSEIL

Méthode de Montgomery (1985)

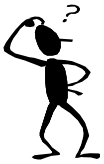
Calculs préalables : $R = B^q$; $N = R-\delta$

Calculer R' et N' vérifiant

Attention : mélange arithmétique modulo N et modulo R

R R' – N N' = 1 Théorème de Bezout
0 < R' < N R' inverse de R modulo N
0 < N' < R N' inverse de -N modulo R

N doit être impair



```
Function REDC(T)     0≤T≤NR
    m  :=  ( T MOD R )* N' MOD R
    t  :=  ( T + mN ) / R
    IF t≥N
           THEN return t-N
           ELSE return t
```

m ≡ T N' [Mod R] < R
mN ≡ T N N' [Mod R] ≡ – T [Mod R] < NR
(T + mN) ≡ 0 [Mod R] < 2NR
(T + mN) est divisible par R ; t < 2N

et tR – T = mN donc tR ≡ T [Mod N]

0 ≤ t < N et tR ≡ T [Mod N]

Réalise une division par R

Bases mathématiques :

Arithmétique modulo N

Méthode de Montgomery (1985)

Calculs préalables : **R R’ – N N’ = 1 ; e = N’ mod B ; δ = R-N**

Brevet

Function $\mathtt{REDC}(T_{2q-1}, T_{2q-2}, \ldots, T_2, T_1, T_0)_B$

$c := 0$

FOR $i := 0$ **TO** $q-1$ **DO**

  $(k, T_{i+q-1}, \ldots, T_{i+1}, T_i)_B := (0, T_{i+q-1}, \ldots, T_{i+1}, T_i)_B + N \times (T_i\ N'\ \mathrm{mod}\ B)$

  $(c, T_{i+q})_B := c + k + T_{i+q}$

« astuces » de calcul brevetées Everbee

Détail du cœur de boucle

$x := T_i$ e [mod B] — *multiplication simple, on laisse tomber la retenue*

$y := x.\delta$ — *donc $y \equiv T_i$ [mod B] car $e\delta \equiv 1$ mod B*

$T := T + x.B^{i+q}$

$T := T - y.B^{i}$ — *tue le mot T_i et $+x\ B^{i+q} - y.B^i$ signifie $+ x(R-\delta)$*

soit +xN

A chaque étape, on tue le mot suivant tout en restant $\equiv T$ [mod N]

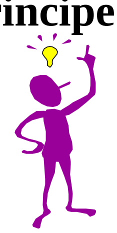
Globalement, temps de calcul d’une grande multiplication

Bases mathématiques :

Arithmétique modulo N

Méthode de Montgomery (1985)

Principe de la méthode



$\xi, \eta, \zeta \in$ Z/NZ représentés en machine par X, Y, Z $\in$ [0..N-1] tels que

$X \equiv \xi R$ [mod N]

$Y \equiv \eta R$ [mod N]

$Z \equiv \zeta R$ [mod N]

Si $\zeta = \xi + \eta$ [mod N] , alors $Z \equiv X+Y$

Si $\zeta = \xi \times \eta$ [mod N] , alors Z.R = X.Y [Mod N] et Z = REDC(X.Y)

- **Les classes d’équivalence modulo N restent codés en machine par leur représentation de Montgomery. Une multiplication équivaut alors à 2 grandes multiplications.**
- **Pour l’exponentiation, on conserve la routine classique** *(Square and Mult)*
- *Attention, les exposants restent stockés en binaire classique*

Application : échange de clé par Diffie Hellman

JLS CONSEIL

Bases mathématiques :

Arithmétique modulo N

Méthode de Montgomery (1985)

Exemple d'implémentation : Mélanger multiplication et réduction

Données

$X = (x_{q-1}x_{q-2} \quad x_1x_0)_B = \sum x_i B^i$

$Y = (y_{q-1}y_{q-2} \quad y_1y_0)_B = \sum y_i B^i$

Variables

$U = (u_{q-1}u_{q-2} \quad u_1u_0)_B = \sum u_i 2^{iw}$

Constantes

h est l'inverse de $-n_0$ modulo B calculé une fois pour toutes

```
U := 0
Pour i := 0 à q-1
    BEGIN LOOP
(1)     P := xi Y               Multiplication scalaire par vecteur
(2)     U := U + P              Addition de deux vecteurs
(3)     m := u0 h   MOD B       Multiplication scalaire sans retenue
(4)     P := m N                Multiplication scalaire par vecteur
(5)     U := U + P              Addition de deux vecteurs
(6)     U := U / B              Simple shift right
    END LOOP
IF U ≥ N THEN U :=U-N
```

<u>Astuce algorithmique :</u> (4) (5) s'écrivent U:=U+mR; P:=m δ; U:=U-P si δ=R-N petit, on y gagne du temps

JLS CONSEIL

Bases mathématiques :

Méthode de Montgomery

Principe de la méthode

$\xi,\eta,\zeta \in$ Z/NZ représentés en machine par X, Y, Z $\in$ [0..N-1] tels que

$X \equiv \xi$ R [mod N]

$Y \equiv \eta$ R [mod N]

$Z \equiv \zeta$ R [mod N]

Si $\zeta = \xi + \eta$ [mod N] , alors Z $\equiv$ X+Y [Mod N]

Si $\zeta = \xi \times \eta$ [mod N] , alors Z.R $\equiv$ X.Y [Mod N]

Z = REDC(X.Y)

Calcul de ξ connaissant X : ξ := REDC(X)

Calcul de X connaissant ξ : X := REDC(ξ R^2 MOD N)

Calcul préliminaire de R^2 MOD N :

- **R MOD N = δ : On va q fois multiplier δ par B, modulo N pour obtenir δ R MOD N**
  - **Pour multiplier par B,**
    Décalage d'un mot vers la gauche
    Multiplier par δ le coefficient de B^q
    Ajouter le résultats au grand entier formé des mots entre B^0 et B^{q-1}
- **Réitérer l'opération q fois**

Bases mathématiques : Méthode de Montgomery

JLS CONSEIL

Dans la mesure du possible, on essayera de rester en représentation de Montgomery

Calculs préalables à effectuer pour une nouvelle valeur du module N

- **Les calculs de Bezout :** Déterminer R' et N' vérifiant R R' – N N' = 1
soit un calcul par Euclide généralisé
n'est pas indispensable si on calcule « par boucles »

- **Le calcul de R^2 `MOD` N :** S'il est nécessaire de passer de la représentation classique à la représentation de Montgomery

Utile pour les tests de primalité

Compléments mathématiques : Fonction d'Euler et groupe U_N

JLS CONSEIL

Exemple : N = 12 = 2^2.3 $\Phi(12) = 2\ (2-1)\ (3-1) = 4$ $U_{12} = \{ 1, 5, 7, 11 \}$

Remarque : $\forall\ x \in U_{12} : x^2 = 1$

Théorème d'Euler : $\forall\ x \in U_N\ \ x^{\Phi(N)} \equiv 1$ [mod N]

Remarque : Ne s'applique pas quand $x \in Z/NZ$ n'est pas inversible

Définition : $g \in U_N$:

Ordre de g = $\nu_N(g)$ = premier exposant r tel que $g^r \equiv 1$ [mod N]

<u>Théorème d'Euler :</u> L'ordre de g est un diviseur de $\Phi(N)$

$g \in U_N$ est une racine primitive ou encore un générateur de U_N si $\nu_N(g) = \Phi(N)$.

Les puissances successives de g engendrent alors U_N

Compléments mathématiques : Le groupe U_N

JLS CONSEIL

Si U_N possède un générateur, alors U_N est cyclique

(U_N , $\times$) isomorphe à (Z / $\Phi(N)$ Z , +)

Théorème : Les seules valeurs de N pour lesquelles U_N est cyclique sont 2, 4, p^e , $2p^e$ (p premier impair, $e \geq 1$)

Théorème du logarithme discret

Si g est un générateur, alors $g^x \equiv g^y$ [mod N] $\Leftrightarrow$ $x \equiv y$ [mod $\Phi(N)$]

Démonstration : Bijection entre (U_N , $\times$) et (Z / $\Phi(N)$ Z , +)

Théorème : Si p est premier alors $x^2 \equiv 1$ [mod p] a comme seules racines 1 et -1

Corollaire : Si 1 a une racine carrée [mod n] non triviale, n n'est pas premier

Si $\exists x$, $x \neq -1$ (mod N), $x \neq 1$ (mod N) et $x^2 \equiv 1$ (mod N) alors N n'est pas premier

Ce théorème est utilisé dans les tests de primalité

Compléments mathématiques : Les puissances d'un élément de Z/NZ

JLS CONSEIL

Si g est inversible (donc $g \in U_N$) l'ensemble des puissances successives de g forme un groupe multiplicatif d'ordre $\nu_N(g)$.

C'est faux quand g n'est pas inversible.

Exemple : N = 12 : **g = 5 : { 1, 5, (5^2=1) } : groupe à deux éléments**
g = 2 : { 1, 2, 4, 8, 4, 8,… } n'est pas un groupe

Mais quand N n'a pas de facteurs carrés, $g^a = g$ si $a \equiv 1$ [$\Phi(N)$]

Réciproque vraie si g est un générateur (Théorème du Log discret)

ne s'applique pas si N=pq (cf. RSA) car il n'y a pas de générateurs

Compléments mathématiques :

Les puissances d'un élément de Z/NZ

Exemple : N = 15 : U_{15} = {1,2,4,7,8,11,13,14} Φ[15]= (3-1)(5-1)=8

Tableau des puissances successives de g

g=0	non inversible.	$g^{\text{n'importe quoi}} = g$
g=1	inversible, d'ordre 1	$g^{\text{n'importe quoi}} = g$
g=2	inversible, d'ordre 4	engendre {1, 2, 4, 8, 1,…}
g=3	non inversible	engendre {1, 3, 9, 12, 6, 3, …}
g=4	inversible, d'ordre 2	engendre {1, 4, 1,…}
g=5	non inversible	engendre {1, 5, 10, 5, …}
g=6	non inversible	engendre {1, 6, 6, …}
g=7	inversible, d'ordre 4	engendre {1, 7, 4, 13, 1,…}
g=8	inversible, d'ordre 4	engendre {1, 8, 4, 2, 1,…}
g=9	non inversible	engendre {1, 9, 6, 9, …}
g=10	non inversible	engendre {1, 10, 5, 5, …}
g=11	inversible, d'ordre 2	engendre {1, 11, 1,…}
g=12	non inversible	engendre {1, 12, 9, 3, 6, 12, …}
g=13	inversible, d'ordre 4	engendre {1, 13, 4, 7, 1,…}
g=14	inversible, d'ordre 2	engendre {1, 14, 1…}

Quand g est inversible, l'ordre de g divise toujours Φ[15] =8

Pour tout g, il existe a tel que $g^a = g$, et pour tout k, on a $g^{8k+1} = g$

4, 11 sont des racines non triviales de 1 donc 15 n'est pas premier

Compléments mathématiques :

Les puissances d'un élément de Z/NZ

Problème du logarithme discret : g d'ordre élevé

L'ensemble des puissances successives de g forme un groupe multiplicatif isomorphe à Z/νZ où ν est l'ordre de g.

L'application a → g^a [mod N]

est une permutation non triviale de Z/νZ

Calcul direct facile

Calcul inverse très difficile

Fonction à sens unique

Exemple : N=13, g=6 engendre 1, 6, 10, 8, 9, 2, 12, 7, 3, 5, 4, 11, 1

Bases mathématiques
Les nombres premiers

De grands nombres premiers sont nécessaires pour de nombreux protocoles

- **Diffie Hellman et ses dérivés, basés sur le Log discret**
- **Cryptographie à clé publique RSA**

Pour RSA, il faut trouver deux nombres p et q tels que leur produit N=pq soit difficilement factorisable

Faiblesses
- **Si l'un des deux est petit**
- **Si la différence est petite**

Factorisation est plus difficile
- **Si pgcd((p-1)(q-1)) est petit**
- **Si (p-1) et (q-1) ont de grands facteurs premiers**

On choisira si possible des nombres premiers de Sophie Germain
Nombres premiers de la forme 2r+1 avec r lui-même premier.
La factorisation est alors plus difficile

JLS CONSEIL

Bases mathématiques
Les nombres premiers

Soit p un nombre premier et soit N un nombre >>p

Probabilité que N soit divisible par p : 1/p

Probabilité que N soit premier : $\prod(1-1/p)$ (pour tous les p premier < $\sqrt{N}$) $\cong 1 / \ln(N)$

Pour trouver un grand nombre premier, on tire au hasard un nombre de l'ordre de grandeur souhaité.

Si 1024 bits, environ une chance sur 700 de trouver un nombre premier.

Tests triviaux :
Éliminer les nombres pairs (en élimine 50%)
Éliminer les nombres multiples de 3 (élimine 33% de ceux qui restent)
Éliminer les nombres multiples de 5 (élimine 20% de ceux qui restent)
Éliminer les nombres multiples de 7 (élimine 14% de ceux qui restent)

Si on élimine tous les nombres ayant un facteur premier <256, il en reste 10.04%
Si on élimine tous les nombres ayant un facteur premier <512, il en reste 8.93%

JLS CONSEIL

Bases mathématiques
Les nombres premiers

Tests triviaux :

P	1 – 1/p	Produit cumulé
2	0.5000	0.5000
3	0.6667	0.3333
5	0.8000	0.2667
7	0.8571	0.2286
11	0.9091	0.2078
13	0.9231	0.1918
17	0.9412	0.1805
19	0.9474	0.1710
23	0.9565	0.1636
29	0.9655	0.1579
31	0.9677	0.1529
37	0.9730	0.1487
41	0.9756	0.1451
43	0.9767	0.1417
47	0.9787	0.1387
53	0.9811	0.1361
...	...	...
251	0.9960	0.1004
...	...	...
509	0.9980	0.0893
...	...	...

Ca décroît de plus en plus lentement

Test de divisibilité par des petits nombres premiers p

On travaille en base B (B = 10, B=256 ou B=2^{32} …),
On veut tester $x = \sum x_i B^i$

On a généré au préalable un tableau r(p,i) avec
$r(p,0) = 1$
$r(p,1) = B \text{ MOD } p$
$r(p,k) = B^k \text{ MOD } p$
Les r(p,k) sont les puissances successives de r(p,1) modulo p

C'est rapidement périodique (car $r(p,1)^{p-1} \equiv 1$ [mod p] d'après Fermat)

On construit la fonction $x \rightarrow \text{red}(p,x) = \sum x_i\, r(p,i) \equiv x$ [mod p]
Rapide à calculer et en principe red(p,x) < B

(pour B=2^{32}, p < quelques centaines de mille et pour une arithmétique à quelques centaines de mots…) Sinon, on réapplique red

On regarde alors si le résultat est divisible par p
(rapide car division sur petits nombres)

JLS CONSEIL

Bases mathématiques
Tests de primalité

- **On utilise les propriétés des nombres premiers : Si N est premier**
  - **Il n'y a pas de racine carrée non triviale de 1 modulo N**
  - **Théorème de Fermat : $\forall a \in U_N \quad a^{N-1} \equiv 1$ [mod N]**
- **Si N ne vérifie pas ces propriétés, c'est qu'il n'est pas premier.**
- **Si N échoue au test, on est sûr qu'il n'est pas premier**
- **Si N réussit un test, il est probablement premier**
- **Si N réussit plusieurs tests, il est presque certainement premier**

Tests probabilistes : si N a une chance sur 2^{50} de ne pas être premier, c'est suffisant pour la plupart des applications

Bases mathématiques
Tests de primalité

- **Définition : N pseudo-premier de base a**

$$a^{N-1} \equiv 1 \ [\bmod N]$$

Sinon, on dit que a est un témoin du caractère non premier de N

- **Il y a très peu de nombres pseudo-premiers non premiers, et il y en a de moins en moins lorsque les nombres augmentent**
- **Nombres de Carmichael : pseudo-premiers pour tout a**
  - **Exemples : 561, 1105, 1729.**
  - **Ils sont très rares (il y en a 255 inférieurs à 10^8)**
- **Test de Rabin-Miller**
  - **Recherche simultanément des témoins et des racines carrées non triviales de l'unité**
  - **Pour un candidat de 256 bits, proba d'erreur après 6 tests < 2^{-50}** **(et ça diminue rapidement quand la taille des nombres augmente)**

Bases mathématiques
Tests de primalité

Test de Rabin-Miller

Déterminer b tel que p = 1 + 2^b m avec m impair (Immédiat si p est écrit en binaire)
Tirer a aléatoire < p
j :=0
z := a^m mod p
Si z = 1 ou p-1 **Alors** **p a des chances d'être premier. EXIT**
Boucle
  Si j>0 et z=1 **Alors** **p n'est pas premier. EXIT**
  Si z = p-1 et j<b **Alors** **p a des chances d'être premier. EXIT**
  Sinon **Si j=b** **Alors p n'est pas premier. EXIT**
  j :=j+1
  z := z^2 mod p *(z vaut toujours $a^{2^j m}$ et on est sûr que j<b et z $\neq$ p-1)*
Fin de boucle

JLS CONSEIL

Bases mathématiques
Tests de primalité

Méthodologie

Générer un nombre p aléatoire à n bits

Forcer à 1 le bit de poids fort et le bit de poids faible

- **Si on veut optimiser certains calculs forcer à 1 les 64 ou 128 bits de poids fort**

Tester si le nombre est divisible par les petits nombres premiers

- **Jusqu'à 2000… optimum à trouver…**

Faire un test de Rabin-Miller avec a aléatoire.

- **Si succès, réitérer 6 fois avec de nouvelles valeurs de a**
- **Si échec, p n'est pas premier, réessayer avec un autre p aléatoire**

JLS CONSEIL

Quelques attaques classiques

- **Générateur aléatoire biaisé : Entropie faible (≅ 40 bits)**
  - ➢**Introduit une backdoor dans les échanges de clés Diffie Hellman**
  - ➢**Introduit une backdoor dans la génération de nombres premiers pour RSA**
- **Clés RSA faibles : si α a ≅ quatre fois moins de bits que N**
  - –p, q premiers secrets très grands N = pq public
  - – α secret β public avec $\alpha\beta \equiv 1$ [mod (p-1)(q-1)]
  - –M $\curvearrowright$ (M^α) [mod N] $\curvearrowright$ $((M^\alpha)^\beta)$ [mod N]
  - –M $\curvearrowright$ (M^β) [mod N] $\curvearrowright$ $((M^\beta)^\alpha)$ [mod N]

Faille découverte en 1990 par Michael Wiener :
algorithme rapide de calcul de α à partir de β

Sachant que $\Phi[N]=(p-1)(q-1)= N+1-(p+q)$ on fournit à l'utilisateur une clé privée égale à $\alpha+r\Phi[N]$

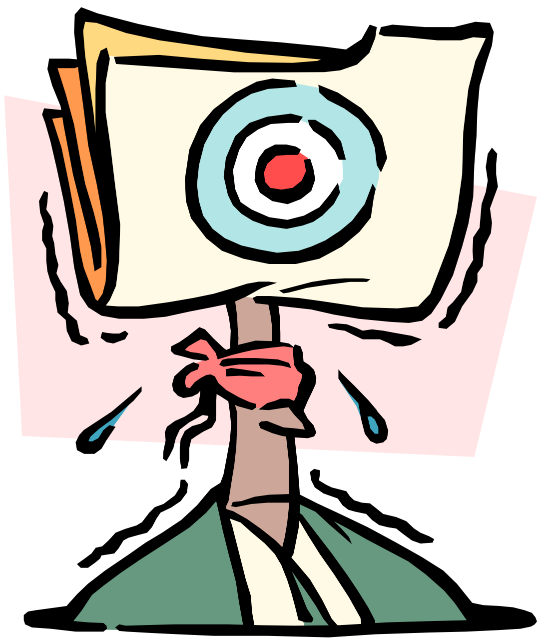
Progrès récents

JLS CONSEIL

- **Comment trouver une bonne fonction à sens unique (avec γ=1) ?**
  – Rappel : temps en $L_n(\gamma,c) = O(\exp(c\, n^{\gamma}\, \ln(n)^{1-\gamma}))$
  – γ=1 : exponentiels γ=0 : polynomiaux
- **Trouver des groupes (G,°) tels que, pour x donné**
  – **$n \rightarrow x^{\circ n} = [n]x$ rapide à calculer**
  – **Réciproque exponentiellement difficile**
  – **Applications : processeurs à faible puissance**
    - Cartes à puce
    - Cartes SIM du GSM,…
- **Théorie des courbes elliptiques**
  - **Clés à 160 bits / 256 bits offrent la sécurité de 1024 / 8192 bits dans Z/pZ**

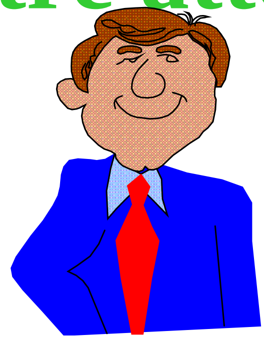
Questions?

JLS CONSEIL



JLS CONSEIL

Nous vous remercions de votre attention



Y a-t-il des questions ?