

# Arithmétique Modulaire

*Cours de*  
**Bases mathématiques pour la sécurité informatique**

Jean-Luc Stehlé  
EPITA  
14 mai 2014



Jean-Luc.Stehle@NormaleSup.org

## Révisions

### • Les éléments de la sécurité

#### ➤ Authentification



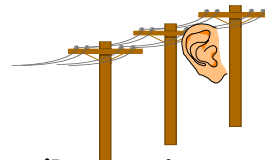
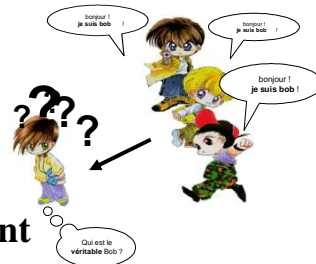
#### Partage d'une clé de chiffrement

#### Chiffrement/déchiffrement

#### ➤ Scellement



#### Non répudiation et signature électronique



## Révisions

### • Deux type de chiffrement

#### ➤ Chiffrement symétrique

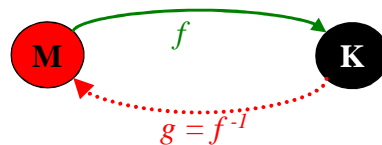
- Même clé pour chiffrer et pour déchiffrer
- Vigenère, DES, AES

#### ➤ Chiffrement asymétrique

- Clé de chiffrement  $\neq$  Clé de déchiffrement
- **Systèmes à clé publique / clé privée**
  - Authentification,
  - Non répudiation,
  - Signature électronique

## Notion de fonction à sens unique

### • Fonction à sens unique

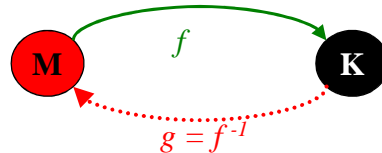


#### ➤ Exemple :

- **Exponentiation modulaire**
- **Logarithme discret**

## Fonction à sens unique avec clé

- Le calcul inverse est facile si on connaît la clé



- Exemple : Système de chiffrement asymétrique

## 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

- $a \equiv b \text{ [Mod } N] \Leftrightarrow \exists \lambda \in \mathbb{Z} \mid (a-b) = \lambda N$   
 $\Leftrightarrow$  a et b ont même reste dans la division euclidienne par N
- Une classe d'équivalence peut être représentée par ce reste par N, nombre compris entre 0 et N-1
- L'ensemble des classes d'équivalence est noté  $\mathbb{Z}/N\mathbb{Z}$



## Propriétés de $\mathbb{Z}/N\mathbb{Z}$

- **Addition : +**  
Commutative, associative, élément neutre 0 (classe des multiples de N)  
Tout nombre a un opposé :  $a + (N-a) \equiv 0 \text{ [Mod } N]$
- **Multiplication :  $\times$**   
Commutative, Associative, élément neutre 1  
Distributive par rapport à l'addition
- **$\mathbb{Z}/N\mathbb{Z}$  est un anneau commutatif unitaire**



## Anneaux commutatifs unitaires

Soit  $A$  un anneau commutatif unitaire

- **Notations habituelles :**

Opérations :  $+$   $\times$

Éléments neutres  $0$   $1$

$-a$  est l'opposé de  $a$

On note habituellement  $2, 3, 4,$  etc

les éléments  $1+1, 1+1+1, 1+1+1+1$  etc

- **Théorème**  $\forall a : a \times 0 = a$

Démonstration :

$a \times b = a \times (0+b) = a \times 0 + a \times b$  (Distributivité)

et on ajoute  $-(a \times b)$  à droite et à gauche...



## Inverse dans un anneau

On dit que  $a$  est inversible s'il existe un élément appelé inverse de  $a$ , noté  $a^{-1}$ , vérifiant

$$a \times a^{-1} = a^{-1} \times a = 1$$

Si tous les éléments de  $A$  autres que  $0$  sont inversibles, alors  $A$  est un corps

Si  $a \times b = 0$  avec  $a \neq 0$  et  $b \neq 0$  (il y a des diviseurs de zéro) alors  $a$  n'est pas inversible.

Démonstration par l'absurde :

S'il existe  $a^{-1}$  alors  $a^{-1} \times (a \times b) = 0$  donc  $b = (a^{-1} \times a) \times b = 0$



## Racines carrées de l'unité

### Solutions de l'équation $x^2=1$

- $(x+1)(x-1)=0$
- deux racines évidentes :  $x=1$  et  $x=-1$   
Appelées racines triviales.
- S'il y a d'autres racines, il y a des diviseurs de 0.

**Si dans un anneau il y a des racines carrées non triviales de l'unité, l'anneau n'est pas un corps**



## Équation $x^2=x$

$$x(x-1) = 0$$

**Deux solutions triviales :  $x=0$  et  $x=1$**

**S'il y a d'autres solutions,  
alors il y a des diviseurs de 0**

**Si dans un anneau il y a des éléments égaux à leur carré et autres que 0 ou 1, alors l'anneau n'est pas un corps**



JLS CONSEIL

## Éléments inversibles dans $\mathbb{Z}/N\mathbb{Z}$

### **a inversible**

$$\Leftrightarrow \exists b : a.b \equiv 1 \pmod{N}$$

$$\Leftrightarrow \exists \lambda : a.b + \lambda N = 1$$

$$\Leftrightarrow a \text{ premier à } N \quad (\text{Bezout})$$

**Théorème :  $\mathbb{Z}/N\mathbb{Z}$  est un corps  
si et seulement si  $N$  est premier**

Si  $N = u.v$  avec  $0 < u < N$  et  $0 < v < N$ , alors  $u.v \equiv 0$ , il y a des diviseurs de 0, et  $\mathbb{Z}/N\mathbb{Z}$  n'est pas un corps.

Plus généralement, si avec  $0 < u < N$  n'est pas premier à  $N$ , il y a un diviseur  $d$ ,  $0 < d < N$  commun à  $u$  et à  $N$ .  
 $\exists a, b : u = a.d \quad N = b.d \Rightarrow a.b.d = a.N = b.d \Rightarrow b.d \equiv 0 \pmod{N} :$   
 il y a des diviseurs de 0, et  $\mathbb{Z}/N\mathbb{Z}$  n'est pas un corps.



JLS CONSEIL

## Tests de primalité

- S'il y a dans  $\mathbb{Z}/N\mathbb{Z}$  une racine carrée non triviale de l'unité,  $N$  n'est pas premier.
- S'il y a dans  $\mathbb{Z}/N\mathbb{Z}$  un élément autre que 0 et 1 égal à son carré,  $N$  n'est pas premier.

### Exemples :

$$5^2 = 25 \equiv 5 \pmod{10}$$

$$6^2 = 36 \equiv 6 \pmod{10}$$

10 n'est pas premier

$$5^2 = 25 \equiv 1 \pmod{12}$$

$$7^2 = 49 \equiv 1 \pmod{12}$$

12 n'est pas premier



## Le groupe $U_N$

JLS CONSEIL

$U_N$  est le groupe multiplicatif des éléments inversibles dans  $\mathbb{Z}/N\mathbb{Z}$

$\Phi[N]$  = Nombre d'éléments de  $U_N = \text{card}(U_N)$

Nombre d'entiers inférieurs à  $N$  et premiers à  $N$

$\Phi$  est la fonction d'Euler

$\Phi[N]$  est appelé indicateur d'Euler de  $N$

Exemple :  $N=12$   $U_{12} = \{1, 5, 7, 11\}$

Remarque : Tous les éléments de  $U_{12}$  sont racines carrées de l'unité



## Le théorème d'Euler

JLS CONSEIL

**Théorème :** Si  $a$  est inversible modulo  $N$ ,  
alors  $a^{\Phi[N]} \equiv 1 \pmod{N}$

**Corollaire :** Si  $a$  est inversible modulo  $N$ ,  
et  $e \equiv 1 \pmod{\Phi[N]}$   
alors  $a^e \equiv a \pmod{N}$

Démonstration :

$\{1, a, a^2, a^3, \dots, a^e, \dots, a^v, \dots, a^{v-1}\}$  sous-groupe de  $U_N$  formé des puissances successives de  $a$ .

Formé de  $v$  éléments où  $v$  est le premier exposant tel que  $a^v = 1$

$v$ , nombre d'éléments du sous groupe divise  $\Phi[N]$  nombre d'éléments du groupe  $U_N$





## Le théorème de Fermat

Cas particulier du théorème d'Euler :

$N=p$  premier

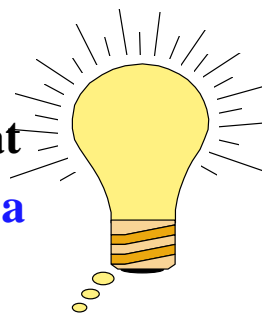
- $\forall a : a^p = a \text{ [Mod } p]$
- Pour  $e \equiv 1 \text{ [Mod } (p-1)]$   $a^e = a \text{ [Mod } p]$
- Attention : Pour  $N$  non premier, il n'existe pas toujours d'exposant pour  $a^e = a \text{ [Mod } p]$
- Contre exemple : Tous les éléments de  $U_{12} = \{1, 5, 7, 11\}$  sont racines carrées de l'unité donc leur puissance 5<sup>ème</sup> 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.



## Théorème de Fermat

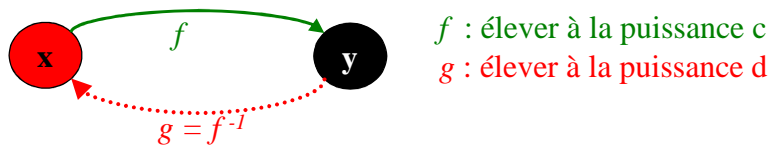
« Petit » théorème de Fermat  
Pour  $p$  premier,  $a \neq 0 \text{ [p]}$ , on a

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



## 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



## Système de Massey - Omura

### Initialisation

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





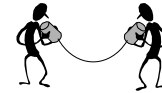
## Système de Massey - Omura

### 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 



## Bases mathématiques :

### Arithmétique dans $\mathbb{Z}$

#### Division euclidienne

$$a = b q + r \quad 0 \leq r < b$$

$$q = a \text{ DIV } b$$

$$r = a \text{ MOD } b$$

$$a \text{ MOD } b = a - (b \times a \text{ DIV } b)$$



*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 $\mathbb{Z}$

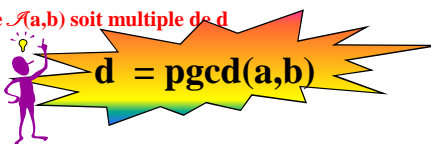
## Notion d'Idéal

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

- Dans  $\mathbb{Z}$ , 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 $\mathbb{Z}$ : Le pgcd

- $\mathbb{Z}$  : Ensemble des entiers relatifs. Muni de  $+$  et de  $\times$ ,  $\mathbb{Z}$  est un anneau.
- Un idéal  $\mathcal{I}$  dans  $\mathbb{Z}$  est un sous-ensemble stable par  $+$  et par  $\times \lambda$  avec  $\lambda \in \mathbb{Z}$
- Dans  $\mathbb{Z}$ , 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  $a,b \in \mathbb{Z}$  :  $\mathcal{A}(a,b)=\{\lambda a+\mu b; \lambda,\mu \in \mathbb{Z}\}$** 
  - $\exists d \in \mathcal{A}(a,b)$  tel que tout élément de  $\mathcal{A}(a,b)$  soit multiple de  $d$ 
    - $d$  divise  $a$  et  $b$ .
    - $\exists \lambda, \mu$  tels que  $d=\lambda a+\mu b$
    - Tout diviseur commun de  $a$  et  $b$  divise  $d$





## Arithmétique dans $\mathbb{Z}$ : Th de Bezout

- a et b premiers entre eux :  $\text{pgcd}(a,b)=1$

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

- Si d divise a et b, d divise 1, donc  $d=1$
- $\mathcal{A}(a,b) = \{\lambda a + \mu b; \lambda, \mu \in \mathbb{Z}\} = \mathbb{Z}$  tout entier

Attention :  $\lambda$  et  $\mu$  ne sont pas uniques

$$\lambda \rightarrow \lambda + k b$$

$$\mu \rightarrow \mu - k a$$



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

### 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 !**

## Bases mathématiques : La fonction d'Euler

$\Phi(N)$  = 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  $\mathbb{Z}/N\mathbb{Z}$

$N = p$  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  $12 = 3 \cdot 2 \cdot 2$   $\Phi(12) = (3-1) \cdot 2 \cdot (2-1) = 4$

## Bases mathématiques : Le théorème des restes chinois

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

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

$\begin{aligned} a &\leftrightarrow (a_1, a_2, \dots, a_k) \\ b &\leftrightarrow (b_1, b_2, \dots, b_k) \end{aligned}$	$\Rightarrow$	$\begin{aligned} a+b &\leftrightarrow (a_1+b_1, a_2+b_2, \dots, a_k+b_k) \\ a \cdot b &\leftrightarrow (a_1 \cdot b_1, a_2 \cdot b_2, \dots, a_k \cdot b_k) \end{aligned}$
--	---------------	--

- Détermination de la fonction réciproque** (se ramène à Bezout pour  $k=2$ )
  - $m_i = n / n_i$  (donc  $m_i \equiv 0 \pmod{n_j}$  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  $j \neq i$
  - $a = (a_1 c_1 + a_2 c_2 + \dots + a_k c_k) \bmod n$
- La famille d'équations  $x \equiv a_i \pmod{n_i}$  a une solution unique modulo  $n$**
- $x \equiv a \pmod{n_i} \forall i \Leftrightarrow x \equiv a \pmod{n}$**



## 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  $\lambda, \mu$  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  $a \bmod n_2 = a_2$




## 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  $\Phi(ab) = \Phi(a) \times \Phi(b)$

## Un cas particulier

➤  $n = p \cdot q$ ,  $p$  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  $p \Rightarrow a \equiv 0 [p] \Rightarrow a^{n'importe\ quoi} \equiv a [p]$

➤ Donc pour tout  $a$ ,  $a^{(p-1)(q-1)+1} \equiv a [p]$

➤ De même  $a^{\lambda \cdot (p-1)(q-1)+1} \equiv a [p]$ ,  $\forall \lambda \in \mathbb{Z}$

➤ Même raisonnement modulo  $q$ ,

➤  $p$  et  $q$  étant premiers entre eux, une égalité vraie modulo  $p$  et modulo  $q$  reste vraie modulo  $pq$ .

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

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



## Théorème : Si $n$ n'a pas de facteur carré,

alors,  $\forall \lambda \in \mathbb{Z}$ , on a  $a^{\lambda \cdot \Phi(n) + 1} \equiv a [n]$

• **Corollaire :** Si  $\alpha$  et  $\beta$  vérifient  $\alpha \cdot \beta \equiv 1 [\Phi(n)]$  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  $\alpha$  il est **très difficile** de calculer  $\beta$  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  $\alpha$  il est **très difficile** de retrouver  $p$ ,  $q$  et  $\beta$ .

Quelques contraintes sur le choix de  $p$  et  $q$

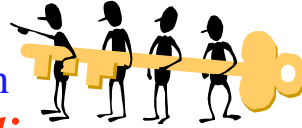
➤ (**Très difficile** = temps > âge de l'univers)





# Le système RSA

Rivest Shamir Adleman



## Systeme à clé publique

- Chaque utilisateur choisit  $n$ ,  $\alpha$  et  $\beta$  vérifiant  $\alpha\beta \equiv 1 \pmod{\Phi(n)}$  et publie  $n$  et  $\alpha$  et garde le reste confidentiel.

- $\alpha$  est la clé publique,  $\beta$  est la clé secrète.

- codage public :  $M \rightarrow \mu \equiv M^\alpha \pmod{n}$



- décodage privé :  $\mu \rightarrow M \equiv \mu^\beta \pmod{n}$




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

# Le système RSA

Rivest Shamir Adleman



## Systeme à clés publiques

- Tout le monde peut envoyer un message secret que seul le destinataire saura decoder. 

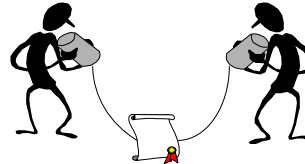
- On peut authentifier un message en codant un checksum avec la clé secrète de l'émetteur

➤ Scellement, non répudiation

- On peut authentifier une liaison 

## Le système RSA

### Authentification d'une liaison



➤ **Publics :**

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

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

➤ **Connu de A seul :**

$\beta_A$

$g_A : M \rightarrow M^{\beta_A} [n_A]$

➤ **Connu de B seul :**

$\beta_B$

$g_B : M \rightarrow M^{\beta_B} [n_B]$

- A génère un nombre aléatoire  $\xi$  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  $\xi$ , 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.

## RSA : Comment l'installer ?

### • Routines de calcul en arithmétique modulaire

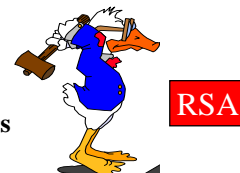
- Addition, Multiplication
- Division [ mod  $\Phi(n)$  ] pour le calcul des clés
- Réduction modulo N
  - Algorithme « de proche en proche »
  - Méthode de Montgomery
- Exponentiation

### • 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  $\gamma = 1/3$

## Attaques sur RSA

- **Rappels mathématiques**

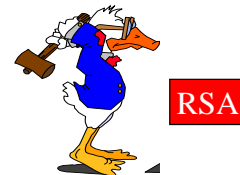
- **p, q premiers secrets très grands** **N = pq public**
- **α secret β public** avec  $\alpha\beta \equiv 1 \pmod{(p-1)(q-1)}$
- $M \xrightarrow{\text{rouge}} (M^\alpha) \pmod N \xrightarrow{\text{vert}} ((M^\alpha)^\beta) \pmod N$
- $M \xrightarrow{\text{vert}} (M^\beta) \pmod N \xrightarrow{\text{rouge}} ((M^\beta)^\alpha) \pmod N$

- **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  $\gamma = 1/3$*

## 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  $t=1,2,3,\dots$  : 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  $p-q$



## 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



## 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  $R \equiv \delta \pmod{N}$ 
  - On a intérêt à prendre  $\delta$  petit
  - Exemple des groupes d'Oakley  
On commence par 64 bits à 1  $\Rightarrow \delta < 2^{-64} R$

## 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

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

*On travaille sur une arithmétique en base  $B$*

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

## Arithmétique modulaire

- **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  $\mathbb{Z}$  requiert  $q^2$  multiplications élémentaires (plus les contrôles de boucles, calculs d'indices,...) et il faudra ensuite faire la réduction modulo  $N$



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

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

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

## Calcul du pgcd : Algorithme d'Euclide

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

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

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

### Programme récursif

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

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

## 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 :  $\phi = (1+\sqrt{5})/2 = 1.61803\dots$   $\phi^{-1} = (-1+\sqrt{5})/2 = \phi - 1 = 0.61803\dots$   $\phi^2 - \phi - 1 = 0$

$F_k = (\phi^k - (-1)^k \phi^{-k}) / \sqrt{5} \equiv \phi^k / \sqrt{5}$  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  $b + (a \bmod b) = b + a - (a \text{ DIV } b) b \leq a$   
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

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

$$\text{Nombre d'appels} < (\log_2 b + \log_2(\sqrt{5}) - \log_2(\phi)) / \log_2(\phi) \sim 1.44 \log_2 b + 0.67$$





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

### Calcul de l'inverse d'un nombre modulo $N$

*Trouver un  $x$  vérifiant  $ax + \mu N = 1$*

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

$\Phi(N)$  = 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  $\mathbb{Z}/N\mathbb{Z}$

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



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

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

#### Phase 1 : On descend

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

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

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

.....

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

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

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

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



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

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

#### Phase 2 : On remonte et on applique des multiplicateurs

$$\begin{array}{ll}
 r_{n-2} = r_{n-1} q_n + r_n & \Rightarrow r_{n-2} - r_{n-1} q_n = 1 \\
 r_{n-3} = r_{n-2} q_{n-1} + r_{n-1} & \Rightarrow r_{n-3} - r_{n-2} q_{n-1} - r_{n-1} = 0 \quad (\times -q_n \text{ et addition, tue le terme en } r_{n-1}) \\
 r_{n-4} = r_{n-3} q_{n-2} + r_{n-2} & \Rightarrow r_{n-4} - r_{n-3} q_{n-2} - r_{n-2} = 0 \quad (\times q_n q_{n-1} \text{ et addition, tue le terme en } r_{n-2}) \\
 \\ 
 r_1 = r_2 q_3 + r_3 & \Rightarrow r_1 - r_2 q_3 - r_3 \quad (\times \text{ ce qu'il faut puis } + \text{ pour tuer le terme en } r_3) \\
 a = r_1 q_2 + r_2 & \Rightarrow a - r_1 q_2 - r_2 \quad (\times \text{ ce qu'il faut puis } + \text{ pour tuer le terme en } r_2) \\
 N = a q_1 + r_1 & \Rightarrow N - a q_1 - r_1 \quad (\times \text{ ce qu'il faut puis } + \text{ pour tuer le terme en } r_1)
 \end{array}$$

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

#### Exercice : Écrire un programme récursif pour inverser $a$

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

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

#### 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)**

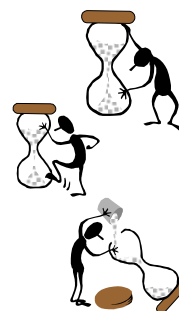
Borne supérieure du nombre d'appels  $\sim 1.44 \log_2 a + 0.67$

$a$  équiréparti entre 1 et  $N-1$ , et pas forcément Fibonacci

Estimation de Knuth : moyenne du nombre d'appel  $= 0.843 \log_2(N) + 1.47$

Nombre d'étapes proportionnel au nombre de bits de  $N$

Temps proportionnel au cube du nombre de bits





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

### Réduction modulo $N$

- $R = 2^n$   $N = R - \delta$  avec  $\delta$  petit
  - On suppose toujours  $\delta < N/2$ , et en général on aura  $\delta \ll N/2$
  - 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  $N$ .



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

### 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  $x \geq N$ ,
  - calculer  $x-N$  (toujours  $< N$ )
  - $x-N = x+\delta-R$
  - On calcule  $x+\delta$  (qui est toujours  $\geq R$ ) 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  $\delta$  est vraiment très petit
  - Un test (quelques tests élémentaires)

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

## 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  $R \equiv \delta \pmod{N}$
  - On calcule  $a + \delta b$
  - Avec les hypothèses précédentes  $a < R$ ,  $\delta < N$ ,  $b$  petit on reste  $< 2N-2$
  - et on procède comme précédemment si ça dépasse  $N$ .

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

## 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

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

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

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

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

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

## Réduction modulo $N$

Pour  $q \leq i \leq 2q$  on a calculé d'avance  $B^i \pmod{N}$

On multiplie par le coefficient et on additionne le résultat

Approximativement  $q^2$  multiplications et additions

*temps équivalent à une grande multiplication*

Calculs préalables :

$$B^q = R \equiv \delta \pmod{N} \ll N$$

$$B^{q+1} \equiv \delta B \pmod{N}$$

On additionne les  $z_i B^i \pmod{N}$

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

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
  END
```

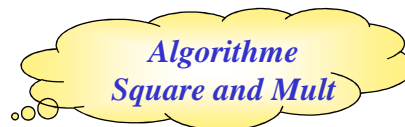
à la fin de la boucle, R contient  $a^b$

END

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

**Attention : si on travaille dans  $\mathbb{Z}/N\mathbb{Z}$  :**

*A est défini modulo  $N$ , B est défini modulo  $\Phi(N)$*





## Bases mathématiques :

### Arithmétique modulo N

#### Méthode de Montgomery (1985)

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

Calculer  $R'$  et  $N'$  vérifiant

$$R R' - N N' = 1$$

$$0 < R' < N$$

$$0 < N' < R$$

Théorème de Bezout

$R'$  inverse de  $R$  modulo  $N$

$N'$  inverse de  $-N$  modulo  $R$

Attention : mélange arithmétique modulo  $N$  et 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
    
```

$m \equiv T N' \pmod{R}$   $< R$   
 $mN \equiv T N N' \pmod{R} \equiv -T \pmod{R}$   $< NR$   
 $(T + mN) \equiv 0 \pmod{R}$   $< 2NR$   
 $(T + mN)$  est divisible par  $R$  ;  $t < 2N$   
 et  $tR - T = mN$  donc  $tR \equiv T \pmod{N}$

$0 \leq t < N$  et  $tR \equiv T \pmod{N}$   
Réalise une division par  $R$

© Jean-Luc Stehlé 1999,2014 Cours ING1 à l'EPITA

Arithmétique Modulaire

14 Mai 2014 Page 57

Document destiné uniquement aux élèves et aux enseignants de l'EPITA. L'auteur vous remercie d'avance de ne pas diffuser ce document



## Bases mathématiques :

### Arithmétique modulo N

#### Méthode de Montgomery (1985)

Calculs préalables :  $R R' - N N' = 1$  ;  $e = N' \bmod B$  ;  $\delta = R - N$

```

Function REDC( $T_{2q-1}, T_{2q-2}, \dots, T_2, T_1, T_0$ )B
  c := 0
  FOR i := 0 TO q-1 DO
    ( $k, T_{i+q-1}, \dots, T_{i+1}, T_i$ )B := ( $0, T_{i+q-1}, \dots, T_{i+1}, T_i$ )B +  $N \times (T_i N' \bmod B)$ 
    ( $c, T_{i+q}$ )B :=  $c + k + T_{i+q}$ 
    
```

« astuces » de calcul brevetées Everbee

Détail du cœur de boucle

$$x := T_i \bmod B$$

$$y := x \cdot \delta$$

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

$$T := T - y \cdot B^i$$

soit  $+xN$

multiplication simple, on laisse tomber la retenue

donc  $y \equiv T_i \pmod{B}$  car  $e\delta \equiv 1 \pmod{B}$

tue le mot  $T_i$  et  $+x B^{i+q} - y \cdot B^i$  signifie  $+x(R - \delta)$

A chaque étape, on tue le mot suivant tout en restant  $\equiv T \pmod{N}$

Globalement, temps de calcul d'une grande multiplication



Brevet

© Jean-Luc Stehlé 1999,2014 Cours ING1 à l'EPITA

Arithmétique Modulaire

14 Mai 2014 Page 58

Document destiné uniquement aux élèves et aux enseignants de l'EPITA. L'auteur vous remercie d'avance de ne pas diffuser ce document

## Bases mathématiques :

### Arithmétique modulo N

#### Méthode de Montgomery (1985)

##### Principe de la méthode



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

$$X \equiv \xi R \pmod{N}$$

$$Y \equiv \eta R \pmod{N}$$

$$Z \equiv \zeta R \pmod{N}$$

Si  $\zeta = \xi + \eta \pmod{N}$ , alors  $Z \equiv X + Y$

Si  $\zeta = \xi \times \eta \pmod{N}$ , alors  $Z.R = X.Y \pmod{N}$  et  $Z = \text{REDC}(X.Y)$

- 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*

## Bases mathématiques :

### Arithmétique modulo N

#### Méthode de Montgomery (1985)

Exemple d'implémentation : Mélanger multiplication et réduction

Données

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

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

Variables

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

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
  
```

*Astuce algorithmique : (4) s'écrit U:=U+mR; P:=m & U:=U-P si &R-N petit, on y gagne du temps*

## Bases mathématiques : Méthode de Montgomery

### Principe de la méthode



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

$$X \equiv \xi R \pmod{N}$$

$$Y \equiv \eta R \pmod{N}$$

$$Z \equiv \zeta R \pmod{N}$$

Si  $\zeta = \xi + \eta \pmod{N}$ , alors  $Z \equiv X + Y \pmod{N}$

Si  $\zeta = \xi \times \eta \pmod{N}$ , alors  $Z.R \equiv X.Y \pmod{N}$

$$Z = \text{REDC}(X.Y)$$

Calcul de  $\xi$  connaissant  $X$  :  $\xi := \text{REDC}(X)$

Calcul de  $X$  connaissant  $\xi$  :  $X := \text{REDC}(\xi R^2 \text{ MOD } N)$

Calcul préliminaire de  $R^2 \text{ MOD } N$  :

- $R \text{ MOD } N = \delta$  : On va  $q$  fois multiplier  $\delta$  par  $B$ , modulo  $N$  pour obtenir  $\delta R \text{ MOD } N$

- Pour multiplier par  $B$ ,

Décalage d'un mot vers la gauche

Multiplier par  $\delta$  le coefficient de  $B^q$

Ajouter le résultats au grand entier formé des mots entre  $B^0$  et  $B^{q-1}$

- Répéter l'opération  $q$  fois

## Bases mathématiques : Méthode de Montgomery

**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  $R^2 \text{ MOD } N$**  : 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 $U_N$

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

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

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

**Remarque :** Ne s'applique pas quand  $x \in \mathbb{Z}/N\mathbb{Z}$  n'est pas inversible

**Définition :**  $g \in U_N$  :

**Ordre de  $g = v_N(g)$  :** premier exposant  $r$  tel que  $g^r \equiv 1 \pmod{N}$

**Théorème d'Euler :** 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  $v_N(g) = \Phi(N)$ .

Les puissances successives de  $g$  engendrent alors  $U_N$



## Compléments mathématiques : Le groupe $U_N$

Si  $U_N$  possède un générateur, alors  $U_N$  est cyclique

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

**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  $g^x \equiv g^y \pmod{N} \Leftrightarrow x \equiv y \pmod{\Phi(N)}$

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

**Théorème :** Si  $p$  est premier alors  $x^2 \equiv 1 \pmod{p}$  a comme seules racines  $1$  et  $-1$

**Corollaire :** Si  $1$  a une racine carrée  $\pmod{n}$  non triviale,  $n$  n'est pas premier

Si  $\exists x, x \not\equiv -1 \pmod{N}, x^2 \equiv 1 \pmod{N}$  alors  $N$  n'est pas premier

**Ce théorème est utilisé dans les tests de primalité**





## Compléments mathématiques :

### Les puissances d'un élément de $\mathbb{Z}/N\mathbb{Z}$

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

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

Exemple :  $N = 12$  :

$g = 5 : \{ 1, 5, (5^2=1) \}$  : groupe à deux éléments

$g = 2 : \{ 1, 2, 4, 8, 4, 8, \dots \}$  n'est pas un groupe

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

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 $\mathbb{Z}/N\mathbb{Z}$

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

#### Tableau des puissances successives de $g$

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

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

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 $\mathbb{Z}/N\mathbb{Z}$

### Problème du logarithme discret : $g$ d'ordre élevé

L'ensemble des puissances successives de  $g$  forme un groupe multiplicatif isomorphe à  $\mathbb{Z}/v\mathbb{Z}$  où  $v$  est l'ordre de  $g$ .

L'application  $a \rightarrow g^a \pmod{N}$

est une permutation non triviale de  $\mathbb{Z}/v\mathbb{Z}$

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  $\text{pgcd}((p-1)(q-1))$  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



# Bases mathématiques

## Les nombres premiers

Soit  $p$  un nombre premier et soit  $N$  un nombre  $\gg p$

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%



# 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  $r(p,1)^{p-1} \equiv 1 \text{ [mod } p]$  d'après Fermat )

On construit la fonction  $x \rightarrow \text{red}(p,x) = \sum x_i r(p,i) \equiv x \text{ [mod } p]$

Rapide à calculer et en principe  $\text{red}(p,x) < B$

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

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



## 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 :  $\forall a \in \mathbb{U}_N \quad a^{N-1} \equiv 1 \pmod{N}$
- 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  $a$   
$$a^{N-1} \equiv 1 \pmod{N}$$

**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  $< p$

$j := 0$

$z := a^m \bmod p$

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

$j := j + 1$

$z := z^2 \bmod p$  ( $z$  vaut toujours  $a^{2^j m}$  et on est sûr que  $j < b$  et  $z \neq p-1$ )

Fin de boucle



## 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