Calculer vite et sans traces
un défi pour les cryptographes

Laurent Imbert

Équipe ECO

6 janvier 2016

ECO : *Exact Computing*

Algorithmes et implémentations efficaces en Arithmétique exacte et Calcul formel

Données

Nombres

Polynômes

Matrices

Corps-finis

Corps de nombres

Courbes elliptiques

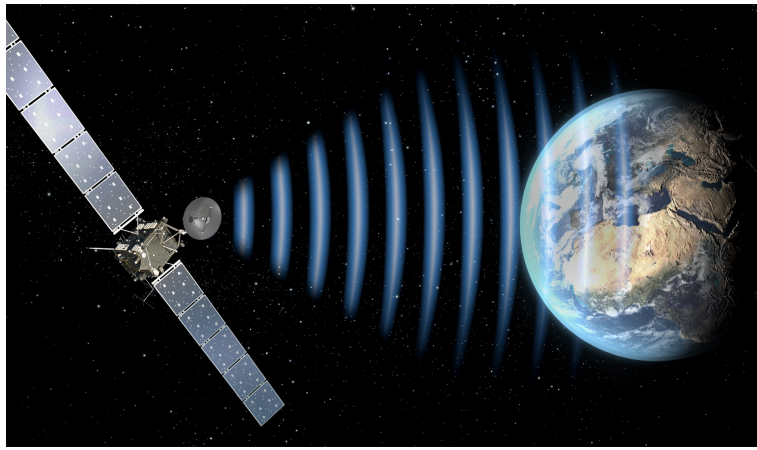
...

ECO : *Exact Computing*

Algorithmes et implémentations efficaces en Arithmétique exacte et Calcul formel

Données	**Problèmes et algorithmes**
Nombres	Multiplication/division rapide
Polynômes	pgcd
Matrices	Algèbre linéaire
Corps-finis	Systèmes polynomiaux
Corps de nombres	Factorisation
Courbes elliptiques	Représentations des éléments et opérations arithmétiques
...	...

Applications



Codes correcteurs d'erreurs

- Recherche de codes optimaux
- Complexité théorique et pratique des algorithmes de décodage

Cryptologie

- Arithmétique pour la cryptographie asymétrique
- Factorisation, calcul de logarithme discret
- Contre-mesures algorithmiques aux attaques physiques

Cryptographie asymétrique

Tout le monde peut chiffrer un avec une clé publique ; seul le destinataire peut déchiffrer avec la clé privée correspondante (ex : RSA)

Retrouver la clé secrète à partir de la clé publique doit être "impossible"

En pratique, on utilise des problèmes réputés difficiles, principalement issus de la théorie des nombres

- factorisation entière : RSA

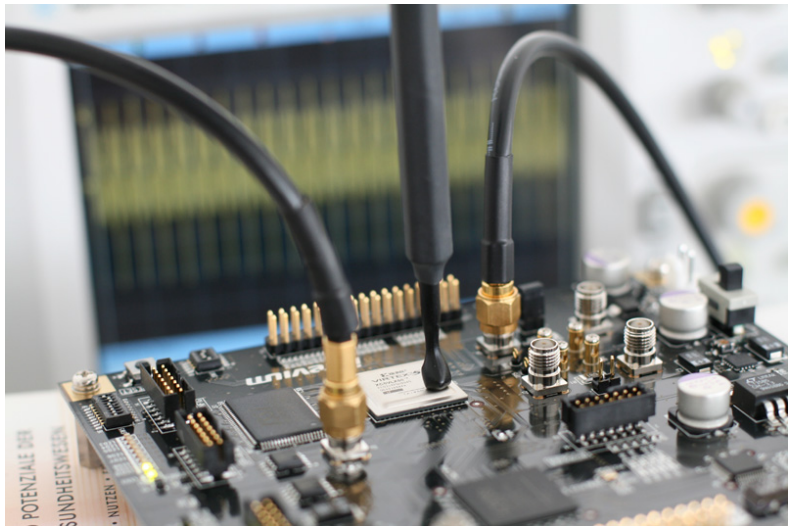
  étant donné $N(= pq)$, trouver p (et q)

- logarithme discret dans un groupe cyclique : ElGamal, Diffie-Hellman, crypto sur courbes elliptiques

  étant donnés $G =< g >$ et $h(= g^a) \in G$, trouver a

Problèmes difficiles en général, i.e. pas d'algorithme polynomial connu

Attaques par canaux cachés [Kocher'96]



Principe d'une attaque sur l'exponentiation rapide

Exponentiation rapide

Input: $x, k = (1k_{\ell-2} \dots k_0)_2$
Output: x^k

$z \leftarrow x$
for $i = \ell - 2, \dots, 0$ **do**
  $z \leftarrow z^2$
  if $k_i \neq 0$ **then**
    $z \leftarrow z \cdot x$
return z

Principe d'une attaque sur l'exponentiation rapide

Exponentiation rapide

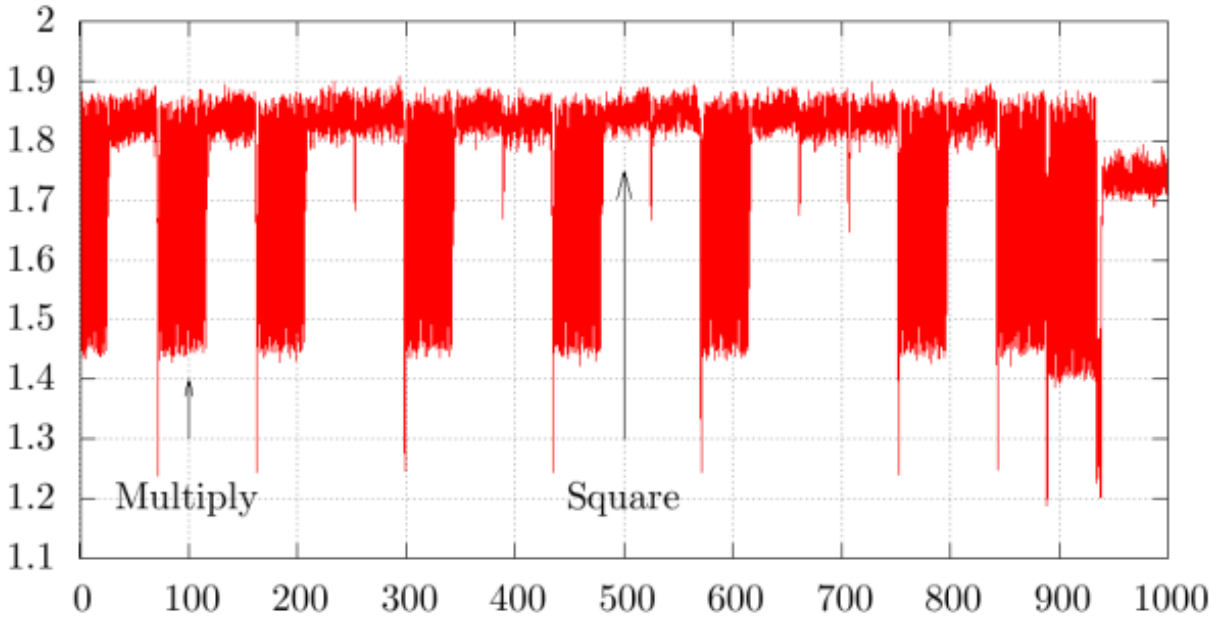
Input: $x, k = (1 k_{\ell-2} \dots k_0)_2$
Output: x^k

$z \leftarrow x$
for $i = \ell - 2, \dots, 0$ **do**
  $z \leftarrow z^2$
  if $k_i \neq 0$ **then**
    $z \leftarrow z \cdot x$
return z

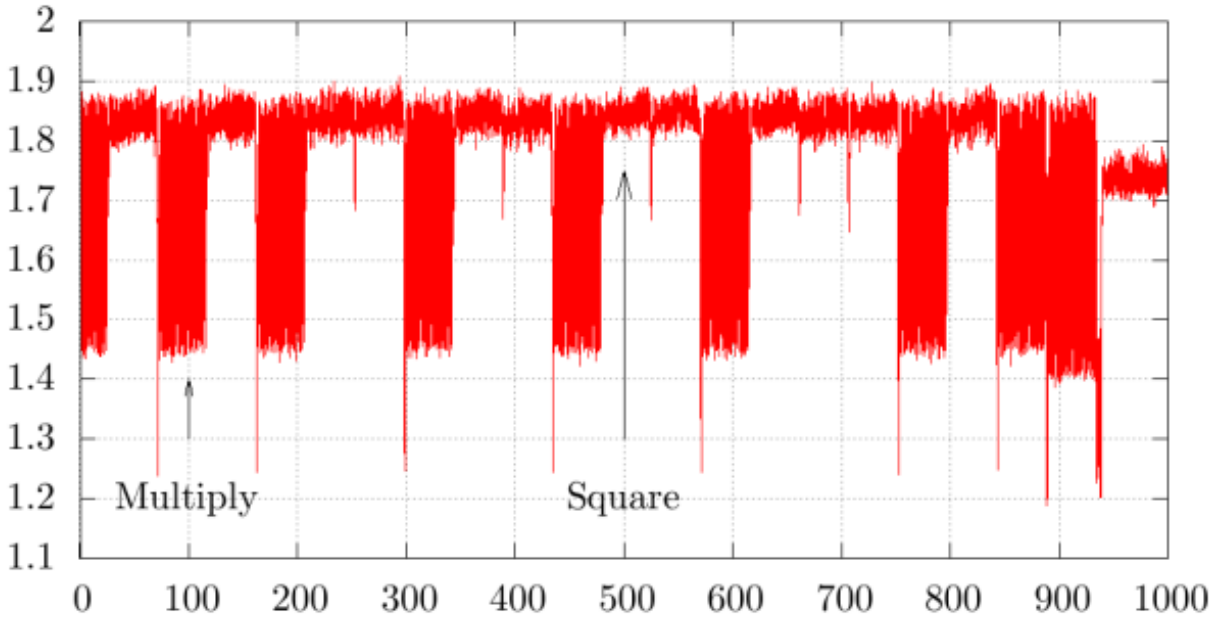
si bit 0 : *Square*

si bit 1 : *Square & Multiply*

Attaques simples



Attaques simples



1 1 1 0 1 0 1 0 1 0 0 1 1

Une seule trace permet de récupérer tous les bits la clé

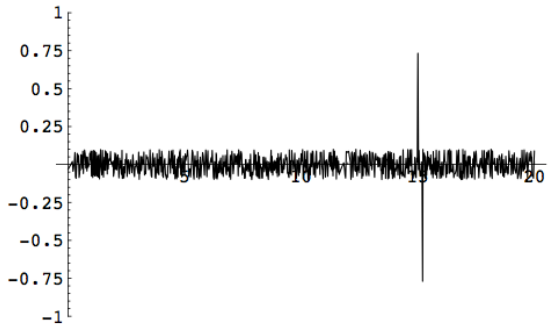
Attaques différentielles

Exponentiation rapide

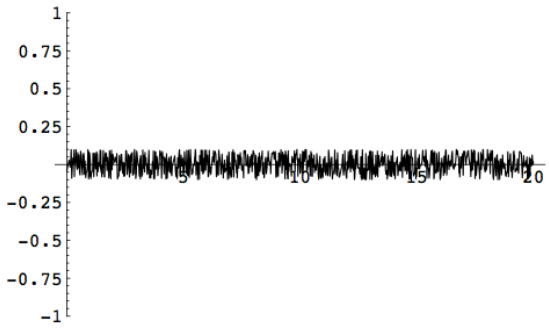
Input: $x, k = (1k_{\ell-2} \dots k_0)_2$
Output: x^k

$z \leftarrow x$
for $i = \ell - 2, \dots, 0$ **do**
  $z \leftarrow z^2$
  if $k_i \neq 0$ **then**
    $z \leftarrow z \cdot x$
return z

$k_{\ell-2} = 0 : x, x^2, x^4, \dots$



$k_{\ell-2} = 1 : x, x^2, x^3, x^6, \dots$



Si $k_{\ell-2} = 0$, il existe une correlation entre l'information émise par le circuit et n'importe quel bit de x^4. Sinon, x^4 n'est jamais calculé et aucune correlation n'est observée.

Contre-mesures algorithmiques – attaques simples

Contre-mesures algorithmiques – attaques simples

S&M always

Input: $x, k = (k_{\ell-1} \dots k_0)_2$
Output: x^k

$z_0 \leftarrow x$
for $i = \ell - 2, \dots, 0$ **do**
$z_0 \leftarrow z_0^2$
$z_1 \leftarrow z_0 \cdot x$
$z_0 \leftarrow z_{k_i}$
return z_0

Contre-mesures algorithmiques – attaques simples

S&M always

Input: $x, k = (k_{\ell-1} \dots k_0)_2$
Output: x^k

$z_0 \leftarrow x$
for $i = \ell - 2, \dots, 0$ **do**
  $z_0 \leftarrow z_0^2$
  $z_1 \leftarrow z_0 \cdot x$
  $z_0 \leftarrow z_{k_i}$
return z_0

Échelle de Montgomery

Input: $x, k = (k_{\ell-2} \dots k_0)_2$
Output: x^k

$z_1 \leftarrow x$
$z_2 \leftarrow x^2$
for $i = \ell - 1, \dots, 0$ **do**
  if $k_i = 0$ **then**
    $z_1 \leftarrow z_1^2, \quad z_2 \leftarrow z_1 \cdot z_2$
  else
    $z_1 \leftarrow z_1 \cdot z_2, \quad z_2 \leftarrow z_2^2$
return z_1

$k = 19 = (10011)_2$

$z_1 : x, x^2, x^4, x^9, x^{19}$

$z_2 : x^2, x^3, x^5, x^{10}, x^{20}$

Contre-mesures algorithmiques – attaques différentielles

Contre-mesures algorithmiques – attaques différentielles

Algorithmes randomisés

- en masquant les données secrètes avec une valeur aléatoire

  Si $x \in G$ d'ordre N, calculer $x^{(e+\tilde{r}N)}$ pour $\tilde{r}$ (petit) aléatoire

- en variant aléatoirement les représentations de x

  Les éléments x des groupes utilisés en crypto sont des classes d'équivalence

- en randomisant l'algorithme d'exponentiation

  construire aléatoirement des chaînes d'additions pour calculer k

  $k = 19 = (10011)_2 \quad x, x^2, x^4, x^8, x^9, x^{18}, x^{19}$

  chaîne d'addition binaire pour 19 : $1, 2, 4, 8, 9, 18, 19$

Représentation à chiffres signés

$k = \sum_{i=0}^{\ell-1} k_i 2^i$, avec $k_i \in \{\overline{1}, 0, 1\}$

Redundance: # entiers $(2^{\ell+1} - 1) \leq$ # mots (3^ℓ)

Calculer aléatoirement une des representations possibles de k

$$a = k + f(\tilde{r}), \quad \sum d_i 2^i = a - f(\tilde{r}) \text{ avec } d_i \in \{\overline{1}, 0, 1\}$$

Example: $k = 478$

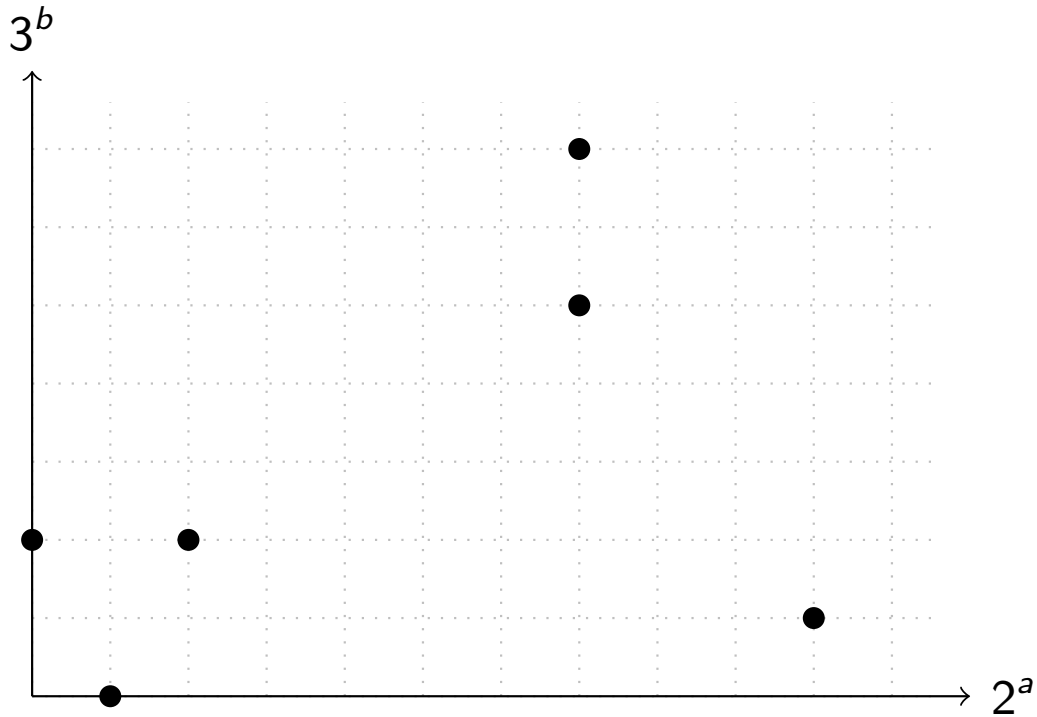
$0111011110_2 : 1, 3, 7, 14, 29, 59, 119, 239, 478$

$100\overline{1}1000\overline{1}0_2 : 1, 2, 4, 7, 15, 30, 60, 120, 239, 478$

Représentation *double-base*

$k = \sum_i d_i p^{a_i} q^{b_i}$, avec $a_i, b_i \geq 0$ et $d_i \in \{\text{chiffres}\}$

Exemple avec $(p, q) = (2, 3)$ et $d_i \in \{1\}$:

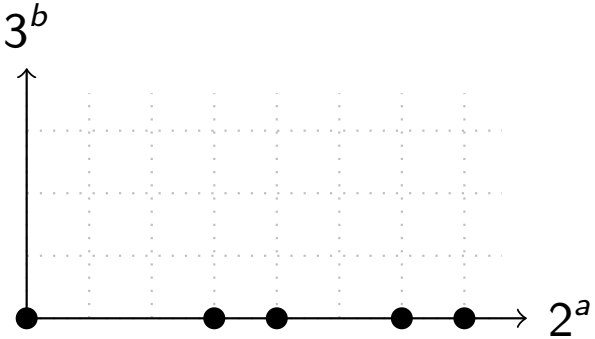


$$314159 = 2^7 3^7 + 2^7 3^5 + 2^{10} 3^1 + 2^2 3^2 + 2^0 3^2 + 2^1 3^0$$

Exponentiation rapide *double-base*

$$g^{217} = (((g^2 \cdot g)^{2^2} \cdot g)^2 \cdot g)^{2^{2^2}} \cdot g$$

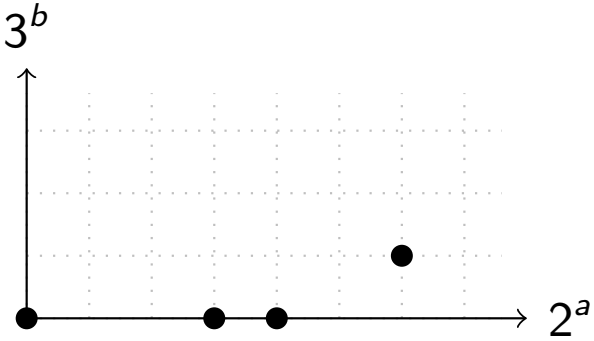
coût : 4 mult., 7 carrés



Exponentiation rapide *double-base*

$$g^{217} = (((g^2 \cdot g)^{2^2} \cdot g)^2 \cdot g)^{2^{2^2}} \cdot g$$

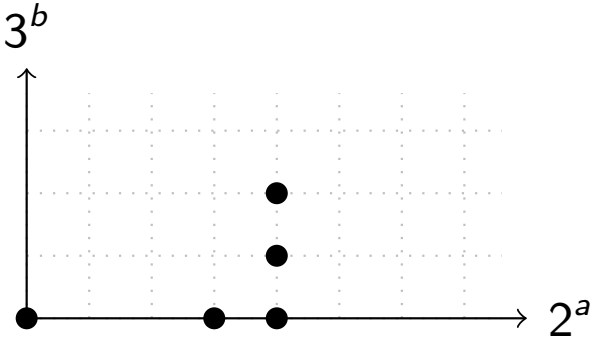
coût : 4 mult., 7 carrés



Exponentiation rapide *double-base*

$$g^{217} = (((g^2 \cdot g)^{2^2} \cdot g)^2 \cdot g)^{2^{2^2}} \cdot g$$

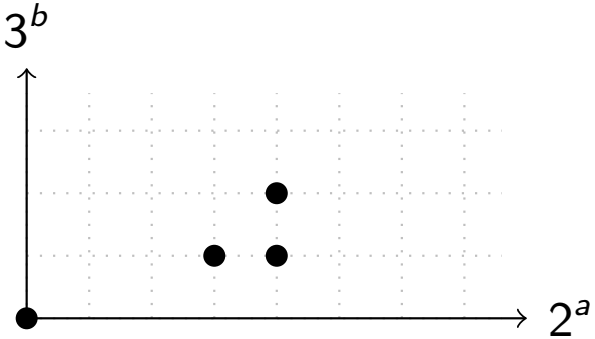
coût : 4 mult., 7 carrés



Exponentiation rapide *double-base*

$$g^{217} = (((g^2 \cdot g)^{2^2} \cdot g)^2 \cdot g)^{2^{2^2}} \cdot g$$

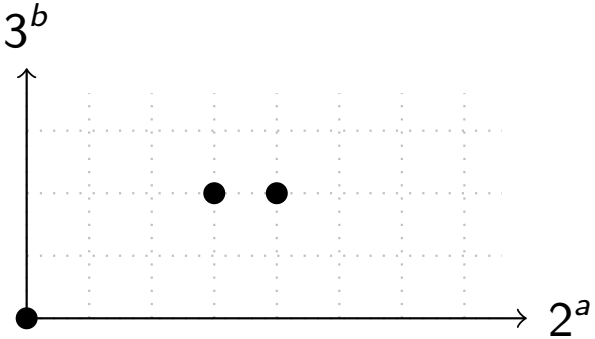
coût : 4 mult., 7 carrés



Exponentiation rapide *double-base*

$$g^{217} = (((g^2 \cdot g)^{2^2} \cdot g)^2 \cdot g)^{2^{2^2}} \cdot g$$

coût : 4 mult., 7 carrés



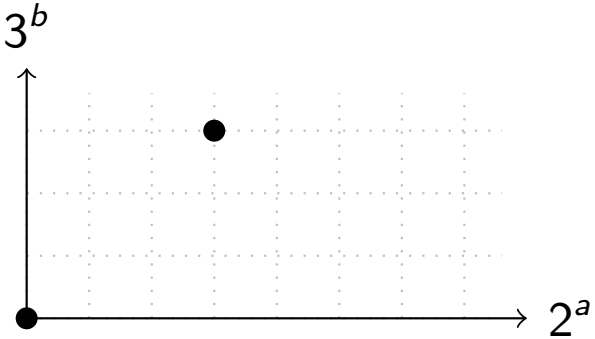
Exponentiation rapide *double-base*

$$g^{217} = (((g^2 \cdot g)^{2^2} \cdot g)^2 \cdot g)^{2^{2^2}} \cdot g$$

coût : 4 mult., 7 carrés

$$g^{217} = g^{2^{2^2} 3^{3^3}} \cdot g$$

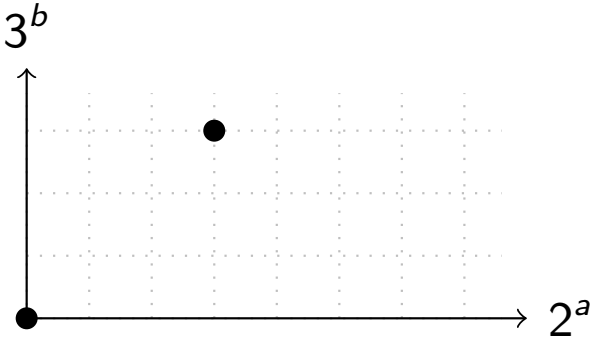
coût : 1 mult., 3 carrés, 3 cubes



Exponentiation rapide *double-base*

$$g^{217} = (((g^2 \cdot g)^{2^2} \cdot g)^2 \cdot g)^{2^{2^2}} \cdot g$$

coût : 4 mult., 7 carrés



$$g^{217} = g^{2^{2^{2^{3^{3^3}}}}} \cdot g$$

coût : 1 mult., 3 carrés, 3 cubes

- rentable si calculer x^3 est plus rapide que calculer $x^2 \cdot x$
  ex : courbes elliptiques, corps de nombres, corps de fonctions
- nécessite des algorithmes de conversion rapides
- problèmes ouverts en combinatoire et théorie des nombres

Pour en savoir plus

www.lirmm.fr/eco

www.lirmm.fr/~imbert

Laurent.Imbert@lirmm.fr

Atelier ECO : “Du Dooble aux codes correcteurs”