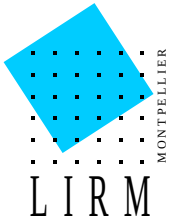
Étude des Changements de Base en RNS

Pascal Giorgi

Directeur de stage : *Jean-Claude Bajard.*
Département : *Informatique Fondamentale et Applications*



Laboratoire d'Informatique, de Robotique
et de Micro-électronique de Montpellier
CNRS - Université Montpellier II

Introduction

- La cryptographie nécessite l'utilisation de grands entiers.
- Le problème est de représenter ces nombres et d'effectuer des opérations.
- Le système modulaire de représentation de nombre est une solution.
- L'opération la plus fréquente en cryptographie est la multiplication modulaire.
- On possède un algorithme efficace pour calculer cette opération en RNS.
- Des progrès sont possibles dans les changements de base inhérents à cet algorithme.

Residue Number System

Soient $m_1, m_2, \ldots, m_{n-1}, m_n$ des nombres tous premier entre eux et $M = \prod_{i=1}^{n} m_i$.

- On peut représenter $X \in [0, M]$ avec $(x_1, x_2, \ldots, x_n)$ tel que:

$$\begin{aligned} x_1 &= X \bmod m_1 \\ x_2 &= X \bmod m_2 \\ &\vdots \\ x_n &= X \bmod m_n \end{aligned}$$

- De plus l'ensemble $(x_1, x_2, \ldots, x_n)$ tel que $x_i \in [0, m_i - 1]$ représente un unique $X \in [0, M]$.
  $(m_1, m_2, \ldots, m_n)$ est appelé base RNS, on la note $\mathcal{B}_n$.

Opérations

$$X_{RNS} = (x_1, x_2, \ldots, x_{n-1}, x_n)_{RNS}$$

$$Y_{RNS} = (y_1, y_2, \ldots, y_{n-1}, y_n)_{RNS}$$

$$X_{RNS} + Y_{RNS} = ((x_1 + y_1) \bmod m_1, \ldots, (x_n + y_n) \bmod m_n)_{RNS}$$
$$X_{RNS} \times Y_{RNS} = ((x_1 \times y_1) \bmod m_1, \ldots, (x_n \times y_n) \bmod m_n)_{RNS}$$

Avantages : Parallélisation des calculs

Inconvénients : Comparaison, division

Cas simple de division

- Si pgcd(Y, M)=1 alors

  $Y_M^{-1} = ((y_1)_{m_1}^{-1}, (y_2)_{m_2}^{-1}, \ldots, (y_{n-1})_{m_{n-1}}^{-1}, (y_n)_{m_n}^{-1})$

  est l'inverse de Y modulo M.

  Si Y est inversible et si X est un multiple de Y, alors: $\frac{X}{Y} = XY_M^{-1}$

Reconstruction avec le théorème des restes chinois

- $X_{RNS} = (x_1, x_2, \ldots, x_n)$ dans $\mathcal{B}_n$ avec $X \in [0, M[$.
  Nous avons

$$X = (\sum_{i=1}^{n} x_i \, |M_i|_{m_i}^{-1} \, M_i) \bmod M$$

  avec $M_i = \frac{M}{m_i}$, and $|M_i|_{m_i}^{-1}$ l'inverse de M_i modulo m_i.

- Remarques:
  $(x_i \, |M_i|_{m_i}^{-1} \, M_i) \bmod m_j = x_i$ si $j = i$
  $(x_i \, |M_i|_{m_i}^{-1} \, M_i) \bmod m_j = 0$ sinon

Mixed Radix : Bases de Cantor

- Soit $X = a_1 + a_2 m_1 + a_3 m_1 m_2 + \cdots + a_n m_1 \cdots m_{n-1}$
- Représentation MRS de $X = (a_1, a_2, a_3, \cdots, a_n)$
- $$\begin{array}{l} a_1 = x_1 \bmod m_1 \\ a_2 = (x_2 - a_1) m_{1,2}^{-1} \bmod m_2 \\ a_3 = ((x_3 - a_1) m_{1,3}^{-1} - a_2) m_{2,3}^{-1} \bmod m_3 \\ a_4 = (((x_4 - a_1) m_{1,4}^{-1} - a_2) m_{2,4}^{-1}) - a_3) m_{3,4}^{-1} \bmod m_4 \\ \vdots \\ a_n = (\cdots (x_n - a_1) m_{1,n}^{-1} - a_2) m_{2,n}^{-1}) - \cdots - a_{n-1}) m_{n-1,n}^{-1} \bmod m_n \end{array}$$

  où $m_{i,j}^{-1}$ est l'inverse de m_i modulo m_j et $(x_1, x_2, \ldots, x_n)$ la représentation RND de X.

Algorithme 1 (Multiplication modulaire en RNS)

Fonction: *RNS_Modular_Montgomery_Multiplication*

Résultat: *un entier* $\mathbf{R < 2N}$ *représenté dans les deux bases tel que* $\mathbf{R \equiv ABM^{-1} (mod N)}$

Méthode: $\mathbf{Q \leftarrow (-A \times_{RNS} B) \times_{RNS} N^{-1}}$ *dans* $\mathcal{B}_\mathbf{n}$

Conversion de représentation de $\mathbf{Q}$ *de* $\mathcal{B}_\mathbf{n}$ *vers* $\widetilde{\mathcal{B}}_{\tilde{\mathbf{n}}}$

$\mathbf{R \leftarrow (A \times_{RNS} B \ +_{RNS} \ Q \times_{RNS} N) \times_{RNS} M^{-1}}$ *dans* $\widetilde{\mathcal{B}}_{\tilde{\mathbf{n}}}$

Conversion de représentation de $\mathbf{R}$ *de* $\widetilde{\mathcal{B}}_{\tilde{\mathbf{n}}}$ *vers* $\mathcal{B}_\mathbf{n}$

Changement de base en RNS

- Szabo et Tanaka : Pour chaque $\tilde{m}_i$ on évalue :

  $$\tilde{x}_i = |a_1 + a_2 m_1 + a_3 m_1 m_2 + \cdots + a_n m_1 \cdots m_{n-1}|_{\tilde{m}_i}$$

- Shenoy et Kumaresan :

  Nous avons, $(\sum_{i=1}^{n} M_i \left| |M_i|_{m_i}^{-1} x_i \right|_{m_i}) = X + \alpha \times M$

  $$\alpha = \left| |M|_{m_{n+1}}^{-1} \left(\sum_{i=1}^{n} \left| M_i \left| |M_i|_{m_i}^{-1} x_i \right|_{m_i} \right|_{m_{n+1}} - |X|_{m_{n+1}} \right) \right|_{m_{n+1}}$$

  $$\tilde{x}_j = \left| \sum_{i=1}^{n} \left| M_i \left| |M_i|_{m_i}^{-1} x_i \right|_{m_i} \right|_{\widetilde{m_j}} - |\alpha M|_{\widetilde{m_j}} \right|_{\widetilde{m_j}}$$

Amélioration du changement de base

- Recherche de bases particulières.
- Recherche d'opérateurs modulaires sur des nombres particuliers.

Recherche de Bases intéressantes

- Soient $\mathcal{B}_n = (m_1, m_2, \ldots, m_n)$ et $\tilde{\mathcal{B}}_t = (\tilde{m}_1, \tilde{m}_2, \ldots, \tilde{m}_t)$ deux base RNS.
- On cherche à améliorer le changement de base de $\mathcal{B}_n$ vers $\tilde{\mathcal{B}}_t$ selon la méthode de *Shenoy et Kumaresan*

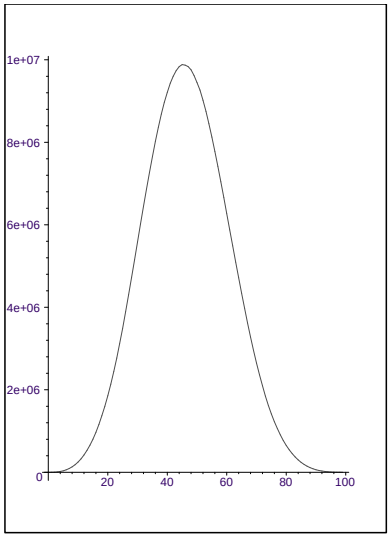
$$\text{pour j}=1 \ldots t \text{ on a } \tilde{x}_j = \left| \sum_{i=1}^{n} \left| \left| \mu_i^j \right|_{\tilde{m}_j} \times x_i \right|_{\tilde{m}_j} - |\alpha M|_{\tilde{m}_j} \right|_{\tilde{m}_j}$$

$$\text{avec } \mu_i^j = \left| M_i (M_i)_{m_i}^{-1} \right|_{\tilde{m}_j}$$

- L'ensemble des μ_i^j dépend des deux bases choisies.
- On s'intéresse à trouver deux bases qui donnent des valeurs faibles à ces constantes.

Étude statistique

- Étude exhaustive sur des bases de 4 moduli de 10 bits
  On prend l'ensemble des nombres premiers de 10 bits pour constituer l'ensemble des moduli possibles
- Calcul de la distribution des valeurs moyennes des constantes multiplicatives dans de telles bases

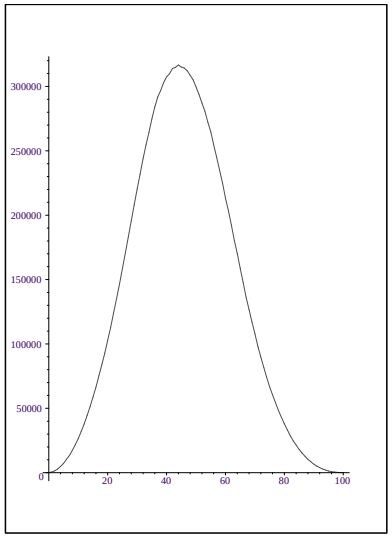


Résultats

- La distribution suit une loi gaussienne (Théorème de la limite centrale).
- Les valeurs moyennes sont concentrés sur $\frac{\mathbf{valeur_{max}}}{\mathbf{2}}$ (Loi des grands nombres).
- Les deux bases fournissant les plus petites valeurs pour les constantes multiplicatives sont :
  $\mathcal{B}_\mathbf{n} = (1091, 1117, 1187, 1193)$ et $\tilde{\mathcal{B}}_\mathbf{t} = (1307, 1367, 1439, 1481)$
  pour un ensemble de constantes ,
  $\mu^{1307} = (163, 8, 179, 17)$, $\mu^{1367} = (69, 29, 311, 133)$,
  $\mu^{1439} = (34, 64, 360, 128)$, $\mu^{1481} = (345, 25, 224, 39)$
- On cherche à augmenter la combinatoire du nombre de bases possibles.

Étude statistique sur des moduli plus grands

- Étude sur des échantillons aléatoires de 10 moduli de 32 bits.
  On prend aléatoirement 10 nombres premier de 32 bits pour constituer une base et on regarde les extension possibles de cette base avec des moduli de 32 bits.
- calcul de la distribution des valeurs moyennes des constantes multiplicatives par extension.



Résultats

- La courbe obtenue possède toujours un maximum de valeur au niveau de $\frac{\mathbf{valeur_{max}}}{2}$.
- La base et le modulo fournissant les plus petites valeurs pour les constantes multiplicatives obtenus pour ces test sont :

$\mathcal{B}_n =$
$(\ 7298095309, 4656564179, 6183656711, 4625970883, 6634405093,$
$4487632993, 6765757589, 5513626369, 6216182647, 5188670603\ )$

et $\tilde{m}_j = (4312403201)$

pour un ensemble de constantes ,

$\mu^{m_j} = (1505105449, 49, 628053581, 597329546, 1804107405,$
$3512147959, 271150152, 1145794824, 392376117, 1936184159)$

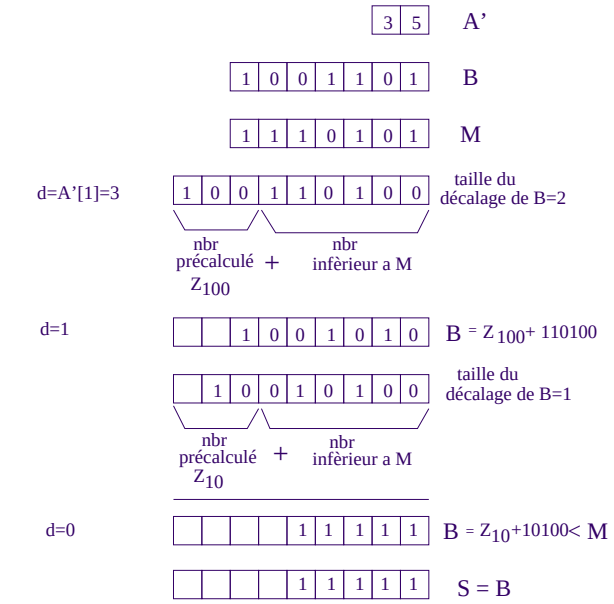
Multiplication modulaire avec précalcul

- On borne nos décalages : nombre décalé au plus $\mathbf{n_m + 2}$ bits, avec $\mathbf{n_m}$ le nombre de bits de $\mathbf{M}$.
- On découpe le nombre décalé en deux partie : partie haute (3 bits de poids fort), partie basse $< \mathbf{M}$
- On additionne la partie basse avec le reste modulaire de la partie haute (lu dans une table).
- Il suffit de réitérer cette méthode sur le résultat qui est strictement inférieur à $\mathbf{2M}$.

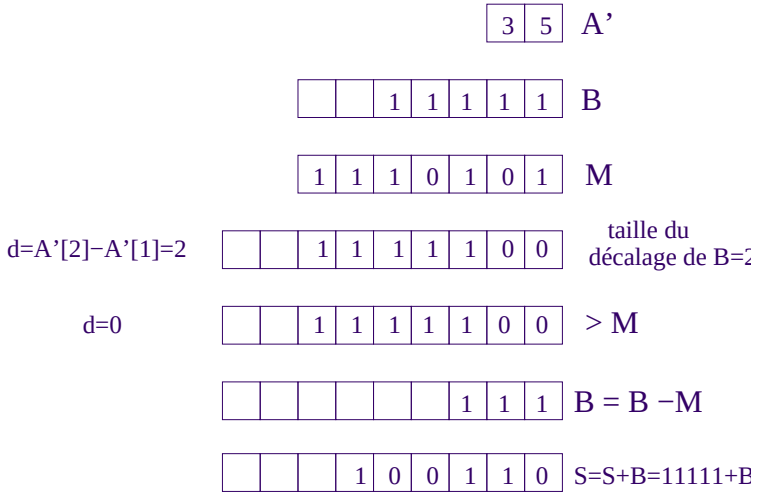
Exemple

- **A = 40, B = 77, M = 117** et **AB mod M = 38**.

i = 1



i = 2



Méthode sur des moduli de 32 bits

- Soit une base RNS $\mathcal{B}_n = (m_1, m_2, \ldots, m_n)$ on représente alors X de la manière suivante :
  $([x_1]_{m_1}, [x_2]_{m_2}, \ldots, [x_n]_{m_n})$ tel que $[x_i]_{m_i} = (x_i \times 2^{32}) \bmod m_i$.
- Dans une telle représentation on a :

$$[x_i + y_i]_{m_i} = [x_i]_{m_i} + [y_i]_{m_i} \bmod m_i$$

$$[x_i \times y_i]_{m_j} = \mathbf{M\_reduce}([x_i]_{m_j} \times [y_i]_{m_j})$$

Multiplication modulaire 32 bits

Algorithme 2 *Réduction de Montgomery M_reduce(*$\mathbf{t}$*)*

Résultat: *un entier* $\mathbf{u}$, $\mathbf{u} = (\mathbf{t}\mathbf{r}^{-1}) \bmod \mathbf{m}$.

Méthode:

$\mathbf{q} := ((\mathbf{t} \bmod \mathbf{r})\mathbf{m}') \bmod \mathbf{r}$;
$\mathbf{u} := (\mathbf{t} + \mathbf{q}\mathbf{m}) \,\mathbf{div}\, \mathbf{r}$;
Si $\mathbf{u} \geq \mathbf{m}$ *alors* $\mathbf{u} := \mathbf{u} - \mathbf{m}$;

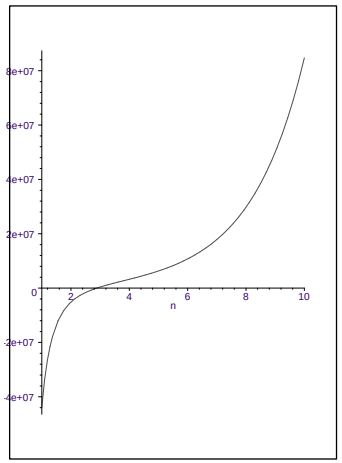
- Dans notre cas $\mathbf{t} = [\mathbf{x_i}]_{\mathbf{m_i}} \times [\mathbf{y_i}]_{\mathbf{m_i}}$, $\mathbf{m} = \mathbf{m_i}$ et $\mathbf{r} = \mathbf{2^{32}}$.
- La multiplication de deux nombres de 32 bits est équivalent à 5 additions dans les calculateurs actuels.

Multiplication modulaire avec tables

- Soient des valeurs résiduelles de $\mathbf{t}$ bits on les découpe en $\mathbf{k}$ mots binaires de $\mathbf{n}$ bits tel que $\mathbf{k} \times \mathbf{n} > \mathbf{t}$.
- Soit $\mathbf{x}$ une valeur résiduelle de $\mathbf{t}$ bits on a :
  $\mathbf{x} = \mathbf{A_1} + \mathbf{A_2} \times \mathbf{2^n} + \mathbf{A_3} \times \mathbf{2^{2n}} + \ldots + \mathbf{A_k} \times \mathbf{2^{(k-1)n}}$
  tel que pour tout $\mathbf{i} = \mathbf{1} \ldots \mathbf{k}$, $\mathbf{A_i}$ est un mot binaire de $\mathbf{n}$ bits.
- On tabule toutes les multiplication des $\mathbf{A_i} \times \mathbf{2^{(i-1)n}}$ par toutes les constantes multiplicatives modulo un modulus donné.
- La multiplication modulaire par une constante multiplicative s'effectue avec $(\mathbf{k} - \mathbf{1})$ additions modulaires et k lectures de tables.

Optimisation du rapport (taille tables×nombre additions)

- La taille des tables pour les constantes et un modulo donnés est de : $\mathbf{k} \times \mathbf{2^n} \times \mathbf{t} \times \mathbf{p} \times \mathbf{2^{-3}}$ Octets.
- Étude du minimum de ce rapport, pour une base RNS de 32 moduli de 32 bits, par la dérivé de cette fonction



- Le minimum est pour des mots binaires de $\mathbf{n = 3}$ bits et il est indépendant de la taille et du nombre de moduli utilisés.

Conclusion

- Aucune base RNS générant de faibles constantes multiplicatives n'a été mise en avant (La distribution suit une loi gaussienne).

  - La meilleure base fournit par l'étude exhaustive n'est pas intéressante.
  - La probabilité de trouver de telles bases pour des modulis de 32 bits est extrêmement faible.

- Les différents opérateurs modulaires définis améliorent la multiplication modulaire suivant les opérandes utilisées.

  - Notre algorithme effectue une mutliplication modulaire rapidement pour des multiplicateurs creux
  - L'algorithme basé sur une nouvelle représentation et sur la réduction de Montgomery est efficace pour des moduli d'au plus 32 bits.
  - L'algorithme basé sur des lectures de tables est intéressant pour des systèmes possédant une certaine capacité mémoire.

Perspectives

- La recherche de constantes multiplicatives creuses serait intéressante dans le cadre de notre algorithme.
- L'implantation de ces opérateurs RNS à des systèmes cryptographiques.
- Étude de nouveaux opérateurs pour les courbes elliptiques dans les corps de Galois.
- Création d'un cryptoprocesseur intégrant de nouveaux systèmes de représentation de nombres .