D34: Méthodes de calcul efficaces et sécurisées

Nicolas Méloni

Master 2: 1er semestre
(2014/2015)

Représentation des nombres

Données du problème:

- La cryptographie nécessite l'utilisation de grands nombres (192 bits avec ECC, plus de 1024 bits avec RSA)
- Les machines actuelles manipulent, généralement, des entiers de 32/64 bits (128 bits plus rarement)
- Possibilité d'implanter directement des algorithmes ad hoc sur des matériels dédiés
- Obligation de passer par une implantation logicielle pour les machines standards

Représentation des nombres

Tableau d'entiers

Pour représenter de grands entiers, on utilise des tableaux d'entiers machines. Cela revient à considérer leur écriture en base β (avec $\beta = 2^{64}$ par exemple):

$$a = \sum_{i=0}^{n-1} a_i \times \beta^i, 0 \leq a_i \leq \beta - 1.$$

$$a = \underbrace{\boxed{10\ldots01}}_{a_{n-1}} \underbrace{\boxed{10\ldots00}}_{a_{n-2}} \ldots\ldots \underbrace{\boxed{00\ldots10}}_{a_0}$$

Addition multi-précision

Addition machine

- Les processeurs possèdent une instruction permettant de calculer la somme de 2 entiers machine, a et b. Le résultat est un mot machine c plus une éventuelle retenue $r \in \{0, 1\}$.
- La retenue peut également être une entrée de la fonction d'addition
- En résumé, cette instruction effectue:

$$a + b + r = r'\beta + c,$$

où r' est la nouvelle retenue et $c = a + b + r \mod \beta$.

Addition multi-précision

Addition de tableaux d'entiers

- L'instruction d'addition est appelée pour faire la somme de chaque paires de coefficients ainsi que de l'éventuelle retenue précédente.

		a_{n-1}	$\cdots\cdots$	a_1	a_0
$+$		b_{n-1}	$\cdots\cdots$	b_1	b_0
$=$	$\boxed{r}$	c_{n-1}	$\cdots\cdots$	c_1	c_0

Addition multi-précision

Algorithm 1 Addition d'entiers multi-précision

Require: $a = \sum_{i=0}^{n-1} a_i \beta^i$, $b = \sum_{i=0}^{n-1} b_i \beta^i$
Ensure: $c = \sum_{i=0}^{n} c_i \beta^i = a + b$

1: $r \leftarrow 0$
2: **for** i=0,..,n-1 **do**
3: $\quad c_i \leftarrow a_i + b_i + r, r \leftarrow 0$
4: $\quad$ **if** $c_i \geq \beta$ **then**
5: $\quad\quad c_i \leftarrow c_i - \beta, r \leftarrow 1$
6: $\quad$ **end if**
7: **end for**
8: $c_n \leftarrow r$
9: **return** $c_n, \ldots, c_0$

Addition multi-précision

Analyse

- Opération principale: addition d'entiers *machine* avec retenue
- nombre total d'additions: n
- Compléxité: $O(n)$

Multiplication multi-précision

Instruction de multiplication

- Prend en entrée deux entiers machine a et b,
- renvoie deux entiers machines d_l et d_h tels que:

$$a \times b = d_h \times \beta + d_l$$

multiplication multiprécision

- Si $a = \sum_{i=0}^{n-1} a_i \beta^i$ et $b = \sum_{i=0}^{n-1} b_i \beta^i$ alors

$$c = a \times b = \sum_{i=0}^{2n-2} c_i \beta^i = \sum_{i=0}^{n-1} (a_i b) \beta^i$$

Multiplication multi-précision

Algorithme des livres d'école

L'algorithme fait appel à deux fonctions:

- l'une calculant le produit entre un entier multi-précision et un entier machine
- l'autre calculant effectuant la multiplication d'un entier mutli-précision par une puissance de la base (simple décalage d'indices).

Multiplication des livres d'école

Algorithm 2 Multiplication entier mp/entier machine

Require: $0 \leq a_i < \beta$, $b = \sum_{j=0}^{n-1} b_j \beta^j$

Ensure: $c = \sum_{j=0}^{n} c_j \beta^j = a_i b$

1. $c_0 \leftarrow 0$
2. **for** j=0,..,n-1 **do**
3. $\quad (d_l, d_h) \leftarrow a_i b_j$
4. $\quad c_j \leftarrow c_j + d_l, c_{j+1} \leftarrow d_h$
5. $\quad$ **if** $c_j \geq \beta$ **then**
6. $\quad\quad c_j \leftarrow c_j - \beta$
7. $\quad\quad c_{j+1} \leftarrow c_{j+1} + 1$
8. $\quad$ **end if**
9. **end for**
10. **return** $c_n, \ldots, c_0$

Multiplication des livres d'école

- On peut alors effectuer la multiplication grâce à l'algorithme suivant:

Algorithm 3 Multiplication d'entiers mp

Require: $a = \sum_{i=0}^{n-1} a_i \beta^i$, $b = \sum_{i=0}^{n-1} b_i \beta^i$
Ensure: $c = \sum_{i=0}^{2n-2} c_i \beta^i = ab$

1. $c \leftarrow 0$
2. **for** i=0,..,n-1 **do**
3. $\quad temp \leftarrow a_i \times b$ `(Alg. 2)`
4. $\quad temp \leftarrow temp \times \beta$
5. $\quad c \leftarrow c + temp$ `(Alg. 1)`
6. **end for**
7. **return** $c_{2n-2}, \ldots, c_0$

Multiplication des livres d'école

Analyse

- L'algorithme 2 est appelé n fois, celui-ci effectue n multiplications et n additions d'entiers machine
- L'algorithme 1 est appelé n fois avec des entiers de $n+1$ entiers machine

Au total l'algorithme 3 effectue $O(n^2)$ multiplications et $O(n^2)$ additions de mots machine.

Multiplication rapide

Préambule

- La multiplication modulaire est l'opération centrale des algorithmes de chiffrement à clé publique
- Combinaison d'un algorithme de multiplication rapide suivi d'un réduction modulaire
- On décrira les algorithmes dans le cadre de la multipication de polynomes dans $\mathbb{Z}[X]$ pour simplifier la description

Multiplication rapide: Karastuba

Données

- On considère deux polynomes de $\mathbb{Z}[X]$:
  $A[X] = \sum_{i=0}^{n-1} a_i X^i$ et $B[X] = \sum_{i=0}^{n-1} b_i X^i$
- Objectif: on cherche à calculer le produit
  $C[X] = A[X]B[X]$ de manière sous quadratique

Multiplication rapide: Karastuba

Astuce de Karatsuba

- Dans le cas $n = 2$, on a $A[X] = A_1X + A_0$ et $B[X] = B_1X + B_0$. Le produit $C[X] = A[X] * B[X]$ se calcule traditionelement en 4 multiplications:

$$C[X] = A_1B_1X^2 + (A_1B_0 + A_0B_1)X + A_0B_0$$

- Remarque:
  $A_1B_0 + A_0B_1 = (A_1 + A_0)(B_1 + B_0) - A_1B_1 - A_0B_0$
- A_1B_1 et A_0B_0 sont déjà calculés, le produit est obtenue en 3 multiplications et 4 additions.

Multiplication rapide: Karastuba

Idée générale

- Étant donnés deux polynômes de degré $n-1$ (on prendra $n = 2^t$ pour simplifier), on découpe chaque polynome en deux polynomes de degré $n/2 - 1$ et on applique récursivement l'astuce précédente.

Multiplication rapide: Karastuba

Algorithm 4 Algorithme de Karatsuba

Require: $A[X] = \sum_{i=0}^{n-1} a_i X^i$, $B[X] = \sum_{i=0}^{n-1} b_i X^i$
Ensure: $C[X] = A[X]B[X]$

1. **procedure** KARATSUBA(A, B)
2. **if** $n = 1$ **then**
3. **return** $A_0 B_0$
4. **else**
5. $P_2 \leftarrow$KARATSUBA(A_1, B_1)
6. $P_0 \leftarrow$KARATSUBA(A_0, B_0)
7. $P_1 \leftarrow$KARATSUBA$(A_1 + A_0, B_1 + B_0)$
8. $P_1 \leftarrow P_1 - P_2 - P_0$
9. **return** $P_2 X^n + P_1 X^{\frac{n}{2}} + P_0$
10. **end if**
11. **end procedure**

Multiplication rapide: Karastuba

Analyse

Soient $M(n)$ la compléxité de multiplication 2 polynomes de degré $n-1$ (avec la convention $M(1)=1$) et $A(n)$ celle de l'addition de 2 polynomes. En choisissant $n=2^t$, on a:

$$M(n) = 3M\left(\frac{n}{2}\right) + 2A(n) + 2A\left(\frac{n}{2}\right).$$

L'addition étant de compléxité linéaire, on en déduit alors que

$$T(n) = O(3^t) = O\left(n^{\log_2(3)}\right).$$

Multiplication rapide: Toom-Cook

Interpolation de polynomes

- Un polynome de degré $n-1$ est parfaitement déterminé par ses valeurs en n points
- Certains problèmes, comme la multiplication, sont plus faciles à résoudre dans cette représentation
- Idée: choisir des points pour lesquels il est facile d'évaluer un polynome, effectuer la multiplication et enfin revenir dans le système de représentation initial

Multiplication rapide: Toom-Cook

Toom-3

- Extension de l'idée de Karatsuba. On découpe les polynomes en 3 qu'on évalue en 5 points (on prendra $n = 3^t$ pour simplifier).
- On pose $A[X] = A_2X^2 + A_1X + A_0$, $B[X] = B_2X^2 + B_1X + B_0$ et $C[X] = A[X]B[X] = C_4X^4 + C_3X^3 + \cdots + C_0$ où les A_i, B_i et C_i sont des polynomes de degré $n/3 - 1$.
- On les évalue en les points: $0, 1, -1, 2, \infty$

Multiplication rapide: Toom-Cook

- On calcule les coefficient comme suit:

$$
\begin{array}{lll}
C[0] & = A[0]B[0] & = A_0B_0 \\
C[1] & = A[1]B[1] & = (A_2 + A_1 + A_0)(B_2 + B_1 + B_0) \\
C[-1] & = A[-1]B[-1] & = (A_2 - A_1 + A_0)(B_2 - B_1 + B_0) \\
C[2] & = A[2]B[2] & = (4A_2 + 2A_1 + A_0)(4B_2 + 2B_1 + B_0) \\
C[\infty] & = A[\infty]B[\infty] & = A_2B_2
\end{array}
$$

Multiplication rapide: Toom-Cook

- On écrit les équations correspondantes:

$$
\begin{aligned}
C[0] &= C_0 \\
C[1] &= C_4 + C_3 + C_2 + C_1 + C_0 \\
C[-1] &= C_4 - C_3 + C_2 - C_1 + C_0 \\
C[2] &= 16C_4 + 8C_3 + 4C_2 + 2C_1 + C_0 \\
C[\infty] &= C_4
\end{aligned}
$$

Multiplication rapide: Toom-Cook

- On se retrouve avec un système de 5 équations à 5 inconnues que l'on résoud facilement grâce à l'algèbre linéaire:

$$\begin{pmatrix} C[0] \\ C[1] \\ C[-1] \\ C[2] \\ C[\infty] \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 2 & 4 & 8 & 16 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathtt{TC}_3} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{pmatrix}$$

Multiplication rapide: Toom-Cook

- Pour retrouver les coefficients du polynome produit, il suffit alors d'inverser la matrice $\mathtt{TC}_3$:

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \\ C_4 \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{3} & -\frac{1}{6} & 2 \\ -1 & \frac{1}{2} & \frac{1}{2} & 0 & -1 \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{6} & \frac{1}{6} & -2 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}}_{\mathtt{TC}_3^{-1}} \begin{pmatrix} C[0] \\ C[1] \\ C[-1] \\ C[2] \\ C[\infty] \end{pmatrix}$$

Multiplication rapide: Toom-Cook

Analyse

- Évaluation des points: $5M\left(\frac{n}{3}\right) + 12A\left(\frac{n}{3}\right)$
- Reconstitution des coefficients: $11A\left(\frac{n}{3}\right)$
- Les multiplications et divisions par des constantes sont ignorées

Conclusion

- On obtient $M(n) = 5M\left(\frac{n}{3}\right) + 23A\left(\frac{n}{3}\right)$ d'où:

$$M(n) = O(5^t) = O(n^{\log_3(5)}).$$

Multiplication rapide: Toom-Cook

Algorithme général

Généralisation de Cook à un découpage en k parties:

- On considère que $n = k^t$,
- $A[X] = A_{k-1}X^{k-1} + \cdots + A_1X + A_0$ où les A_i sont des polynomes de degré $\frac{n/k}{-}1$.
- Le produit $C[X] = A[X]B[X] = \sum_{i=0}^{2k-2} C_iX^i$ doit être évalué en $2k-1$ points (on choisit par exemple $\{-k, -k+1, \ldots 0, \ldots, k-1, k\}$).
- Les $C[j] = A[j]B[j]$ sont calculé par appels récursifs.
- Les C_i sont reconstruit à partir des $C[j]$ grâce à l'algèbre linéaire.

Multiplication rapide: Toom-Cook

Calcul de $A[j]$

$$A[j] = \sum_{i=0}^{k-1} A_i \times j^i$$

- On multiplie n/k polynômes avec des entiers de taille $i \log_2(k)$: $O(n \times c(k))$ où $c(k) = k \log_2(k)$.

Multiplication rapide: Toom-Cook

Calcul dese C_i

- On pose $C[X] = \theta_0 + \theta_1 X + \theta_2 X(X-1) + \cdots + \theta_{2k-2} X(X-1)...(X-2k-3)$
- On calcule des θ_i comme suit:

$$\begin{aligned}
\theta_0 &= C[0] \\
\theta_1 &= C[1] - \theta_0 \\
\theta_2 &= (C[2] - \theta_0)/2 - \theta_1 \\
&\vdots \\
\theta_{2k-2} &= (\ldots(C[2k-2] - \theta_0)/(2k-2) \\
&\quad -\theta_1)/(2k-1)\ldots)/1 - \theta_{2k-3}
\end{aligned}$$

Multiplication rapide: Toom-Cook

Compléxité

- On réalise $O(k^2)$ opérations sur des polynomes de degré $\frac{n}{k} - 1$ pour calculer les $theta_i$ puis pour reconstruire les C_i
- Complexité globale: $O(n \times k)$

compléxité totale

- $T(n) = (2k-1)T(n/k) + c \times n \times klog(k)$
- Ainsi $T(n) = O\left(c(k) n^{\log_k(2k-1)}\right)$
- En particulier, pour $k = 2$ on retrouve la complexité de l'algo de Karatsuba.

Algorithme Schonhage et Strassen

Description générale

- Repose sur la transformée de Fourier discrete
- Complexité: $O(n \log n \log \log n)$
- Vraiment efficace uniquement pour de très grands nombres

Seuils de `gmp` ($\mathrm{MUL\_*\_THRESHOLD}$)

Méthode	Seuil (mots machines)
TOOM22 (Karatsuba)	10-37
TOOM33 (Toom-3)	33-132
FFT (Schonhage et Strassen)	1728-11720