Opérations sur les corps finis

Journée Sécurité Numérique du GDR SoC-SiP

JC Bajard - LIP6 UPMC (Paris 6)
France

16 Nov. 2011

Contexte

- Koblitz en 1987 [Kob87] introduit l'utilisation des courbes elliptiques sur des corps finis en cryptographie.
- Groupe des Points d'une courbe à coordonnées dans un corps finis
- Addition : Intersection d'une droite passant par deux points de la courbe avec la courbe
- Opérations sur les corps finis : addition, multiplication, division.

Contenu

Représentations sur les Corps Finis

Des généralités [LN85]

Un corps fini $F(+,\times)$ est un ensemble fini F muni de deux lois internes

- $F(+)$ est un groupe abélien
- $F(+,\times)$ est un anneau où tout élément (sauf 0 pour $\times$) admet un inverse

Les corps finis les plus élémentaires sont ceux dont le cardinal est un nombre premier p.
Le représentant le plus connu est le corps $\mathbb{Z}/p\mathbb{Z}$

$$\mathbb{Z}/p\mathbb{Z} = \{0, 1, 2, ..., p-1\}$$

Le calcul sur ces corps utilise l'arithmétique modulaire classique.

Extension de corps finis

Les autres corps finis sont de la forme $GF(p^m)$ avec p premier. p est la caractéristique, si $u \in GF(p^m)$ alors $p \times u = 0$.

- soit l'ensemble des restes des polynômes à coefficients dans $GF(p)$ modulo un polynôme irréductible $P(X)$ de degré m (opérations modulaires sur les polynômes)
- soit l'ensemble des puissances d'un élément primitif g, $GF(p^m) = \{0, g^0, g^1, ..., g^{p^m-2}\}$
- soit l'ensemble des combinaisons linéaires d'une base : canonique $\{1, \alpha, \alpha^2, ..., \alpha^{m-1}\}$ ou normale $\{\alpha, \alpha^p, \alpha^{p^2} ..., \alpha^{p^{m-1}}\}$ où α racine du polynôme irréductible

Quelle que soit l'approche un élément de $GF(p^m)$ est donc représenté par un $m-$uplet d'éléments de $GF(p)$.

UPMC SORBONNE UNIVERSITÉS

Exemple dans $GF(4)$ (Attention $GF(4) \neq \mathbb{Z}/4\mathbb{Z}$)

- Représentation polynomiale à coefficient dans $GF(2)$: 0, 1, X, $1+X$.
- L'addition se fait simplement sur $GF(2)$: $1+(1+X)=X$.
- Par contre pour le produit nous devons faire une réduction liée au polynôme irréductible utilisé pour la représentation des éléments du corps.
  - X^2+X+1 est irréductible sur $GF(2)$, $GF(4)$ peut être assimilé à $GF(2)[X]/X^2+X+1$.
  - multiplication modulo le polynôme irréductible X^2+X+1, exemple :
    $X*(1+X)=(X+X^2) \bmod (X^2+X+1)=1$
  - Le choix du polynôme irréductible n'est donc pas sans conséquence sur la complexité de la multiplication.

La multiplication dans $GF(p)$

Multiplication de deux entiers

Produit de deux nombres

via les polynômes

- Soient $A = \sum_{i=0}^{k-1} a_i \beta^i$ et $B = \sum_{i=0}^{k-1} b_i \beta^i$ deux nombres en base β
- Soient $A(X) = \sum_{i=0}^{k-1} a_i X^i$ et $B(X) = \sum_{i=0}^{k-1} b_i X^i$ les polynômes associés
- Calcul du produit $P = A \times B$:
  1. Evaluation du produit polynomial : $P(X) = A(X) \times B(X)$
  2. Evaluation de la valeur: $P(\beta) = A(\beta) \times B(\beta)$

Produit de deux nombres

via les polynômes : remarques

- A l'étape 1, les coefficients p_i obtenus sont inférieurs à $k \times \beta^2$
- A l'étape 2, le calcul de $P(\beta)$ revient à réduire les p_i en propageant des retenues.

Représentation d'un polynôme

- Un polynôme de degré $k-1$ peut être défini par la donnée :
  - de ses k coefficients a_i

$$A(X) = \sum_{i=0}^{k-1} a_i X^i$$

  - de ses valeurs en k points différents e_i

$$\text{pour, } i = 0, k \quad A(e_i) = \sum_{i=0}^{k-1} a_i e_i^i$$

les e_i sont choisis suivant deux critères : un calcul simple des $A(e_i)$ et une taille respectable des $A(e_i)$.

Produit de polynômes

définis par leurs coefficients

- $P(X) = A(X) \times B(X) = (\sum_{i=0}^{k-1} a_i X^i) \times (\sum_{i=0}^{k-1} b_i X^i) = \sum_{i=0}^{2k-2} p_i X^i$

$$\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{k-1} \\ \vdots \\ p_{2k-3} \\ p_{2k-2} \end{pmatrix} = \begin{pmatrix} a_0 & 0 & 0 & \dots & 0 \\ a_1 & a_0 & 0 & \dots & 0 \\ \vdots & \vdots & \dots & & \vdots \\ a_{k-1} & a_{k-2} & a_{k-3} & \dots & a_0 \\ 0 & a_{k-1} & a_{k-2} & \dots & a_1 \\ \vdots & \vdots & \dots & & \vdots \\ 0 & 0 & \dots & 0 & a_{k-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{k-2} \\ b_{k-1} \end{pmatrix}$$

Soit k^2 produits.

Produit de polynômes

définis par leurs valeurs en des points

- $P(X) = A(X) \times B(X) = (\sum_{i=0}^{k-1} a_i X^i) \times (\sum_{i=0}^{k-1} b_i X^i) = \sum_{i=0}^{2k-2} p_i X^i$

  est évalué en $2k-1$ différents points :

$$\begin{cases} P(e_0) = A(e_0) \times B(e_0) \\ P(e_1) = A(e_1) \times B(e_1) \\ \vdots \\ P(e_{2k-3}) = A(e_{2k-3}) \times B(e_{2k-3}) \\ P(e_{2k-2}) = A(e_{2k-2}) \times B(e_{2k-2}) \end{cases}$$

Soit $2k-1$ produits.

Reconstruction des coefficients

Méthode de Lagrange

- La construction passe par une somme de k polynômes tels que le $i-$ième vaut $P(e_i)$ en e_i et 0 pour tous les autres e_j, $j \neq i$.

$$P(X) = \sum_{i=0}^{k-1} P(e_i) \frac{\prod_{j \neq i}(X - e_j)}{\prod_{j \neq i}(e_i - e_j)}$$

Reconstruction des coefficients

Méthode de Newton

- L'idée est d'écrire le polynôme comme dans un système de numération en augmentant le degré des polynômes sommés.

$$P(X) = \sum_{i=0}^{k-1} \hat{p}_i \prod_{j=0}^{i-1} (X - e_j) = \hat{p}_0 + \hat{p}_1(X - e_0) + \hat{p}_2(X - e_0)(X - e_1) + \dots$$

$$\begin{cases} \hat{p}_0 = p'_0 \\ \hat{p}_1 = (p'_1 - \hat{p}_0)/(e_1 - e_0) \\ \dots \\ \hat{p}_i = (\dots(p'_i - \hat{p}_0)/(e_i - e_0) - \hat{p}_1)/(e_i - e_1) - \dots - \hat{p}_{i-1})/(e_i - e_{i-1}) \\ \dots \\ \hat{p}_{k-1} = (\dots(p'_{k-1} - \hat{p}_0)/(e_{k-1} - e_0) - \hat{p}_1)/(e_{k-1} - e_1) \dots - \hat{p}_{k-2})/(e_{k-1} - e_{k-2}) \end{cases}$$

Où, $p'_i = P(e_i)$

Produit de deux nombres

Algorithme Karatsuba (1)

- Choix des points $e_0 = 0$, $e_1 = -1$ et $e_2 = \infty$
- Nous avons :

$$A = \sum_{i=0}^{k-1} a_i \beta^i = \left(\sum_{i=0}^{k/2-1} a_{k/2+i} \beta^i \right) \beta^{k/2} + \sum_{i=0}^{k./2-1} a_i \beta^i = A_1 \beta^{k/2} + A_0$$

- Point de vue polynomial : $A(X) = A_1 X + A_0$

$$\begin{cases} A(0) = A_0 \\ A(-1) = A_0 - A_1 \\ A(\infty) = \lim_{X \to \infty} A_1 X \end{cases}$$

Produit de deux nombres

Algorithme Karatsuba (2)

- Valeurs du polynôme produit

$$\begin{cases} P(0) = A_0 B_0 \\ P(-1) = (A_0 - A_1)(B_0 - B_1) \\ P(\infty) = \lim_{X \to \infty} A_1 B_1 X^2 \end{cases}$$

- Application de Newton

$$\begin{cases} \hat{p}_0 = P(0) = A_0 B_0 \\ \hat{p}_1 = (P(-1) - \hat{p}_0)/(-1) = (A_1 - A_0)(B_0 - B_1) + A_0 B_0 \\ \hat{p}_\infty = \lim_{X \to \infty} ((P(\infty) - \hat{p}_0)/X - \hat{p}_1)/(X + 1) = A_1 B_1 \end{cases}$$

Produit de deux nombres

Algorithme Karatsuba (3)

- Reconstruction

$$\begin{cases} P(X) & = & \hat{p}_0 + \hat{p}_1 X + \hat{p}_\infty X (X+1) \\ & = & A_0 B_0 \\ & & +((A_1 - A_0)(B_0 - B_1) + A_0 B_0 + A_1 B_1)X \\ & & +A_1 B_1 X^2 \end{cases}$$

- Calcul

$$\begin{cases} P(\beta^{k/2}) & = & A_0 B_0 \\ & & +((A_1 - A_0)(B_0 - B_1) + A_0 B_0 + A_1 B_1)\beta^{k/2} \\ & & +A_1 B_1 \beta^k \end{cases}$$

Produit de deux nombres

Algorithme Karatsuba (4) : Complexité

- On note $K(k)$ le nombre d'opérations élémentaires
- On a la formule de récurrence $K(k) = 3K(k/2) + \alpha k$ en supposant les additions linéaires
- Nous obtenons, $K(k) = O(k^{\log_2(3)})$

Produit de deux nombres

Algorithme Toom Cook (1)

L'approche de Karatsuba se généralise, par exemple avec une découpe en trois et donc 5 points différents

- Choix des points $e_0 = 0$, $e_1 = -1$, $e_2 = 1$, $e_3 = 2$ et $e_4 = \infty$
- Nous avons :
  $A = A_2\beta^{2k/3} + A_1\beta^{k/3} + A_0$
- Point de vue polynomial : $A(X) = A_2X^2 + A_1X + A_0$

$$\begin{cases} A(0) = A_0 \\ A(-1) = A_2 - A_1 + A_0 \\ A(1) = A_2 + A_1 + A_0 \\ A(2) = 4A_2 + 2A_1 + A_0 \\ A(\infty) = \lim\limits_{X\to\infty} A_2X^2 \end{cases}$$

Produit de deux nombres

Algorithme Toom Cook (2)

- Comme pour Karatsuba nous appliquons Newton...

$$\begin{cases} \hat{p}_0 = P(0) = A_0 B_0 \\ \hat{p}_1 = (P(-1) - \hat{p}_0)/(-1) \\ \hat{p}_2 = ((P(1) - \hat{p}_0)/(1) - \hat{p}_1)/(2) \\ \hat{p}_3 = (((P(2) - \hat{p}_0)/(2) - \hat{p}_1)/(3) - \hat{p}_2)/(1) \\ \hat{p}_4 = \lim_{X \to \infty} ((((P(\infty) - \hat{p}_0)/X - \hat{p}_1)/(X+1) - \hat{p}_2)/(X-1) - \hat{p}_3)/(X-2) \\ \quad = A_2 B_2 \end{cases}$$

- Ici nous remarquons une division par 3 qui montre un peu les limites de cette approche lorsque l'on généralise
- Reconstruction en calculant $P(\beta^{k/3})$ avec :
  $P(X) = \hat{p}_0 + X(\hat{p}_1 + (X+1)(\hat{p}_2 + (X-1)(\hat{p}_3 + \hat{p}_4(X-2))))$

Produit de deux nombres

Algorithme Toom Cook (3)

- On note $T_3(k)$ le nombre d'opérations élémentaires
- On a la formule de récurrence $T_3(k) = 5\,T_3(k/3) + \alpha k$ en supposant les additions linéaires
- Nous obtenons, $T_3(k) = O(k^{\log_3(5)})$

Produit de deux nombres

Algorithme Toom Cook (4)

- On généralise en découpant en n
- On note $T_n(k)$ le nombre d'opérations élémentaires
- On a la formule de récurrence $T_n(k) = (2n-1)T_n(k/n) + \alpha k$ en supposant les additions linéaires
- Nous obtenons, $T_n(k) = O(k^{\log_n(2n-1)})$
- Nous pouvons donc dire que la multiplication peut se faire pour tout ϵ en $O(k^{1+\epsilon})$

Transformée de Fourier (Annexe 19)

Complexité Algorithme FFT

- Les points considérés sont des racines n^{ieme} de l'unité : $\omega^n = 1$, ω racine primitive.
- Soit $F(n)$ le nombre d'opérations élémentaires pour une FFT de dimension n
- Nous avons la récurrence : $F(n) = 2F(n/2) + \alpha n$
- D'où, $F(n) = O(n \log_2 n)$

La multiplication dans $GF(p)$

Réduction modulaire sur les entiers

Réduction modulaire

p fixé

Deux options :

- p est choisi de façon à rendre la réduction simple

$$p = \beta^n - \xi \quad \text{avec} \quad \xi < \beta^{n/2}$$

- p quelconque, choisi sur d'autres critères, algorithmes généraux

Réduction modulaire

$p = \beta^n - \xi \quad$ avec $\quad 0 \leq \xi < \beta^{n/2}$

Nous avons $C = A \times B \leq (p-1)^2$

- On peut écrire $C = C_1\beta^n + C_0$
  - donc $C = C_1\beta^n + C_0 \leq (p-1)^2 = \beta^n(\beta^n - 2(\xi+1)) + (\xi+1)^2$
  - comme $0 \leq \xi < \beta^{n/2}$ nous avons $C_1 \leq \beta^n - 2$ et $C_0 \leq \beta^n - 1$
- Première passe de réduction : $C \equiv C_1\xi + C_0 \pmod p$
  - $C' = C_1\xi + C_0 \leq (\beta^n - 2)\xi + (\beta^n - 1) \leq \beta^n\xi + (\beta^n - 1)$
  - donc $C' = C'_1\beta^n + C'_0$ avec $C'_1 \leq \xi$ et $C'_0 \leq \beta^n - 1$
- Seconde passe de réduction : $C' \equiv C'_1\xi + C'_0 \pmod p$
  - $C'' = C'_1\xi + C'_0 \leq \xi^2 + (\beta^n - 1) = \beta^n + (\xi^2 - 1)$
  - nous avons $C'' + \xi < \beta^n + p$
- Dernière passe : si $C'' + \xi \geq \beta^n$ alors $R = C'' + \xi - \beta^n$ sinon $R = C''$

Réduction modulaire

$p = \beta^n - \xi \quad$ avec $\quad 0 \leq \xi < \beta^{n/2}$

- Cette réduction effectue deux produits par ξ, pour réduire le coût nous avons deux options
  - Choisir ξ très petit par exemple $\xi < \beta$ pour avoir des produits chiffre×nombre
  - Choisir ξ très creux (peu de chiffres non nul et égaux à un) pour remplacer les produits par des additions et des décalages
- On peut prendre $\xi > \beta^{n/2}$ mais dans ce cas le nombre de passes de réduction augmente
- On peut inclure une partie de la réduction dans le produit initial

Réduction modulaire

p fixé de la forme $\beta^n - 1$

Le produit du point de vue matriciel le produit $C = A \times B$ s'écrit :

$$\begin{pmatrix} 1, & \beta, & \beta^2, & \dots & \beta^{2n-2} \end{pmatrix} \begin{pmatrix} a_0 & 0 & 0 & \dots & 0 \\ a_1 & a_0 & 0 & \dots & 0 \\ \vdots & \vdots & & \dots & \vdots \\ a_{n-1} & a_{n-2} & a_{n-3} & \dots & a_0 \\ 0 & a_{n-1} & a_{n-2} & \dots & a_1 \\ \vdots & \vdots & & \dots & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1} \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix}$$

$$C \equiv \begin{pmatrix} 1, & \beta, & \beta^2, & \dots & \beta^{n-1} \end{pmatrix} .M. \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-2} \\ b_{n-1} \end{pmatrix} \pmod p$$

avec $\beta^n \equiv 1 \pmod p$

Réduction modulaire

p fixé de la forme $\beta^n - 1$

Le produit du point de vue matriciel avec $\beta^n \equiv 1 \pmod{p}$, on a :

$$M = \begin{pmatrix} a_0 & 0 & \dots & 0 & 0 \\ a_1 & a_0 & \dots & 0 & 0 \\ \vdots & \vdots & \dots & & \vdots \\ a_{n-2} & a_{n-3} & \dots & a_0 & 0 \\ a_{n-1} & a_{n-2} & \dots & a_1 & a_0 \end{pmatrix} + \begin{pmatrix} 0 & a_{n-1} & a_{n-2} & \dots & a_1 \\ 0 & 0 & a_{n-1} & \dots & a_2 \\ \vdots & \vdots & \dots & & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1} \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix}$$

$$M = \begin{pmatrix} a_0 & a_{n-1} & a_{n-2} & \dots & a_1 \\ a_1 & a_0 & a_{n-1} & \dots & a_2 \\ \vdots & \vdots & \dots & & \vdots \\ a_{n-2} & a_{n-3} & \dots & a_0 & a_{n-1} \\ a_{n-1} & a_{n-2} & \dots & a_1 & a_0 \end{pmatrix}$$

Réduction modulaire

p fixé de la forme $\beta^n - 1$

- Le point de vue matricielle pour $p = \beta^n - 1$, donne un résultat $C < p \times n(\beta - 1)^2$ pour le coût d'un produit en $O(n^2)$
- On reprend la réduction faite avec $\xi = 1$
  - nous avons $C = C_1\beta^n + C_0$ avec $C_1 < n(\beta - 1)^2$ et $C_0 \leq \beta^n - 1$
  - on pose $C' = C_1 + C_0 < \beta^n + n(\beta - 1)^2 - 1$
  - si $C' + 1 \geq \beta^n$ alors $R = C' + 1 - \beta^n$ sinon $R = C'$

Réduction modulaire

p fixé de la forme $\beta^n - \beta^t - 1$

Le produit du point de vue matriciel avec $\beta^n \equiv \beta^t + 1 \pmod{p}$

Si $t < n/2$ alors la construction de M revient à une addition de deux matrices.

$$M = \begin{pmatrix} a_0 & a_{n-1} & a_{n-2} & \dots & a_1 \\ a_1 & a_0 & a_{n-1} & \dots & a_2 \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \dots & & \vdots \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \dots & & \vdots \\ a_{n-2} & a_{n-3} & \dots & a_0 & a_{n-1} \\ a_{n-1} & a_{n-2} & \dots & a_1 & a_0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \dots & & \vdots \\ 0 & 0 & 0 & \dots & 0 \\ 0 & a_{n-1} & a_{n-2} & \dots & a_1 \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \dots & & \vdots \\ 0 & \dots & a_{n-1} & \dots & a_{n-t} \end{pmatrix}$$

$$+ \begin{pmatrix} 0 & \dots & a_{n-1} & \dots & a_{n-t+1} \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \dots & & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1} \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \dots & & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \dots & & \vdots \\ 0 & 0 & 0 & \dots & 0 \\ 0 & \dots & a_{n-1} & \dots & a_{n-t+1} \\ \vdots & \vdots & & & \vdots \\ \vdots & \vdots & \dots & & \vdots \\ 0 & 0 & \dots & 0 & a_{n-1} \end{pmatrix}$$

Réduction modulaire

p fixé de la forme $\beta^n - \beta^t - 1$

- L'approche matricielle pour $p = \beta^n - \beta^t - 1$, donne un résultat $C < 2p \times n(\beta - 1)^2$ pour le coût d'un produit en $O(n^2)$
- On reprend la réduction faite avec $\xi = \beta^t + 1$
  - $C = C_1\beta^n + C_0$ avec $C_1 < 2n(\beta - 1)^2$ et $C_0 \leq \beta^n - 1$
  - on pose $C' = C_1(\beta^t + 1) + C_0 < \beta^n + n(\beta - 1)^2(\beta^t + 1) - 1$
  - si $C' + 1 + \beta^t \geq \beta^n$ alors $R = C' + 1 + \beta^t - \beta^n$ sinon $R = C'$

Ces approches s'inspirent de celles développées pour $GF(2^k)$ dans la Thèse de E. Mastrovito "VLSI Architectures for Computation in Galois Fields." PhD thesis, Linkoping University, Dept. Electr. Eng., 1991.

La multiplication dans $GF(p)$

Réduction modulaire sur les entiers
Cas plus général

Réduction modulaire

Algorithme de P.Barrett

Nous désirons réduire A modulo P en évaluant une approximation du quotient.

- Conditions : $\beta^{n-1} \leq P < \beta^n$ et $A < P^2 < \beta^{2n}$
- On peut écrire que : $A - P \times \frac{A}{P} = 0$
  Ainsi, $\beta^{u+v} A - P \times \frac{\beta^{n+u}}{P} \times \frac{A}{\beta^{n-v}} = 0$
- $\beta^{u+v} A - P \times \lfloor \frac{\beta^{n+u}}{P} \rfloor \times \lfloor \frac{A}{\beta^{n-v}} \rfloor =$
  $P \left(\lfloor \frac{\beta^{n+u}}{P} \rfloor f(\frac{A}{\beta^{n-v}}) + \lfloor \frac{A}{\beta^{n-v}} \rfloor f(\frac{\beta^{n+u}}{P}) + f(\frac{A}{\beta^{n-1}}) f(\frac{\beta^{2n}}{P}) \right) < P(\beta^{u+1} + (\beta^{n+v} - 1) + 1)$
  où $f(.)$ la partie fractionnaire

  SI $u \geq n+1$ et $v \geq 2$ alors $(\beta^{u+1} + \beta^{n+v})/\beta^{u+v} < 1$
- On en déduit que $A - P \times \left\lfloor \frac{\lfloor \frac{\beta^{2n+1}}{P} \rfloor \times \lfloor \frac{A}{\beta^{n-2}} \rfloor}{\beta^{n+3}} \right\rfloor < 2P$

Réduction modulaire

Algorithme de P.Barrett

Barrett(A P)

Entrées $\beta^{n-1} \leq P < \beta^n$ et $A < P^2 < \beta^{2n}$

Sortie , $R = A \pmod P$ et $Q = \lfloor \frac{A}{P} \rfloor$

Corps $Q \leftarrow \left\lfloor \frac{\lfloor \frac{\beta^{2n+1}}{P} \rfloor \times \lfloor \frac{A}{\beta^{n-2}} \rfloor}{\beta^{n+3}} \right\rfloor$

$R \leftarrow A - Q \times P$

Si $R \geq P$ Alors $R \leftarrow R - P$et $Q \leftarrow Q + 1$

Complexité : 2 produits sur $n + 1$ chiffres

Réduction modulaire

Algorithme de P. Montgomery

Nous désirons réduire A modulo P.

- Conditions : $\beta^{n-1} \leq P < \beta^n$ et $A < P\beta^n$
- Le principe de cette approche est d'ajouter à A un multiple de P tel que le résultat soit un multiple de β^n
- Ainsi, ce résultat est divisible par β^n qui représente dans la base β un simple décalage.
- Cet algorithme donne en sortie $A \times \beta^{-n} \bmod P$

Réduction modulaire

Algorithme de P. Montgomery

Montgomery(A P)

Entrées $\beta^{n-1} \leq P < \beta^n$ et $A < P\beta^n < \beta^{2n}$

Sortie , $R = A \times \beta^{-n} \bmod P$

Corps $Q \leftarrow A \times |-P^{-1}|_{\beta^n} \bmod \beta^n$

$R \leftarrow (A + Q \times P)$ *R est un multiple de* β^n

$R \leftarrow R \div \beta^n$ *la division par* β^n *est un simple décalage, on a* $(R < 2P)$

Si $R \geq P$ Alors $R \leftarrow R - P$ *(opération facultative)*

Complexité : 2 produits sur n chiffres (en fait deux demi-produits)

Réduction modulaire

Notation de P. Montgomery

- Pour éviter l'accumulation de facteurs β^{-n} mod P, nous utilisons la notation : $\widetilde{A} = A \times \beta^n \bmod P$
- Elle s'obtient par $\widetilde{A} = \text{Montgomery}(A \times |\beta^{2n}|_P, P)$
- Elle est stable pour l'addition et la multiplication réduite par Montgomery :
  $\widetilde{A} + \widetilde{B} = \widetilde{A + B}$ et $\widetilde{AB} = \text{Montgomery}(\widetilde{A} \times \widetilde{B}, P)$
- Retour à la notation standard : $A = \text{Montgomery}(\widetilde{A}, P)$
- **C'est l'algorithme le plus utilisé en cryptographie.**

Multiplication modulaire interlacée

Algorithme de P. Montgomery

Montgomery(A , B, P)

Entrées $\beta^{n-1} \leq P < \beta^n$ et $AB < P\beta^n < \beta^{2n}$ et $B = \sum_{i=0}^{n-1} b_i \beta^i$

Sortie , $R = A \times B \times \beta^{-n} \bmod P$ avec $(R < 2P)$

Corps $R \leftarrow 0$

Pour $i = 0$ **à** $i = n - 1$ **faire**

$R \leftarrow (R + b_i \times A)$

$q_i \leftarrow r_0 \times |-p_0^{-1}|_\beta \bmod \beta$

$R \leftarrow (R + q_i \times P)$ *multiple de* β

$R \leftarrow R \div \beta$ *on a* $(R < 2P)$ *en fin de boucle*

Si $R \geq P$ Alors $R \leftarrow R - P$ *(opération facultative)*

Multiplication modulaire interlacée en base deux

Algorithme de P. Montgomery

Montgomery(A , B, P)

Entrées $2^{n-1} \leq P < 2^n$ et $AB < 2^n P < 2^{2n}$ et $B = \sum_{i=0}^{n-1} b_i 2^i$

Sortie , $R = A \times B \times 2^{-n} \bmod P$

Corps $R \leftarrow 0$

Pour $i = 0$ **à** $i = n - 1$ **faire**

$R \leftarrow (R + b_i \bullet A)$

$q_i \leftarrow r_0$ *En fait* $| - p_0^{-1}|_2 = 1$ *si* P *impair*

$R \leftarrow (R + q_i \bullet P)$ *multiple de* 2

$R \leftarrow R >> 1$ *on a* $(R < 2P)$ *en fin de boucle*

Si $R \geq P$ Alors $R \leftarrow R - P$ *(opération facultative)*

Multiplication modulaire Bipartite

M. Kaihara et N. Takagi 2007 IEEETC

- Cet algorithme est basé sur la remarque suivante :

  On définit $*$ comme : $X * Y = (X \times Y) \times R^{-1} \bmod P$

  On décompose $Y = Y_h \times R + Y_l$ *par exemple* $R = \beta^{n/2}$

  donc $X * Y = (X \times Y_h \bmod P + X \times Y_l \times R^{-1} \bmod P) \bmod P$
- $X \times Y_h \bmod P$ est calculé via par exemple Barret.
- $X \times Y_l \times R^{-1} \bmod P$ est calculé via Montgomery.
- Ces deux calculs peuvent être faits en parallèle.

Multiplication dans $GF(2^n)$

La multiplication dans $GF(2^m)$

Dans la pratique la grande majorité des implantations utilisent des corps de la forme $GF(2^m)$ où les opérations sur le corps de base $GF(2)$ se réduisent à des "et" et des "ou exclusif".
Le produit sur un corps fini comprend une réduction modulaire qui donne lieu à plusieurs types d'approches :

- celles qui ne dépendent pas du corps choisi
- celles qui au contraire exploitent les particularités de ce dernier
- celles utilisant des bases non canoniques...

Multiplication dans $GF(2^n)$

Approches polynomiales

La multiplication dans $GF(2^m)$

Le calcul de $C(X) = A(X) \times B(X) \bmod P(X)$ peut se faire en deux temps :

1. un produit de polynômes $C'(X) = A(X) \times B(X)$,

$$\begin{pmatrix} c'_0 \\ c'_1 \\ \dots \\ c'_{m-1} \\ c'_m \\ \dots \\ c'_{2m-2} \end{pmatrix} = \begin{pmatrix} a_0 & 0 & & \dots & 0 & 0 \\ a_1 & a_0 & 0 & & 0 & 0 \\ & & & \dots & & \\ a_{m-1} & & & & a_1 & a_0 \\ 0 & a_{m-1} & & & & a_1 \\ & & & \dots & & \\ 0 & 0 & & & 0 & a_{m-1} \end{pmatrix} \times \begin{pmatrix} b_0 \\ b_1 \\ \dots \\ b_{m-1} \end{pmatrix}$$

2. une réduction modulo $P(X)$: $C(X) = C'(X) \bmod P(X)$

Algorithme de Montgomery

- Nous voulons le produit $A(X) * B(X)$ dans $GF(2^m)$ définie par $P(X)$ un polynôme irréductible de degré m,
- L'algorithme de Montgomery évalue $A(X) * B(X) * R^{-1}(X) mod P(X)$ où $R(X)$ est un élément fixé et $R^{-1}(X)$ représente son inverse mod $P(X)$.
  Connaissant $R(X)$, comme $P(X)$ est irréductible il est possible de déterminer par avance $R^{-1}(X)$ et $P'(X)$ tels que :

$$R^{-1}(X) * R(X) + P'(X) * P(X) = 1$$

Algorithme de Montgomery (cas général)

Entrées : $A(X)$ *et* $B(X)$ *deux polynômes de degrés inférieurs à* m
Résultat : $T(X) = A(X) * B(X) * R^{-1}(X) \bmod P(X)$
Connues : $P'(X)$, $R(X)$

Produit : $C(X) = A(X) * B(X)$
Réduction : $Q(X) = -C(X) * P'(X) \bmod R(X)$
$T(X) = \big(C(X) + Q(X) * P(X)\big) \mathit{div}\ R(X)$

- La complexité de cet algorithme est due aux trois produits.
- La réduction modulo $R(X)$ et la division par $R(X)$ sont très simples si $R(X) = X^m$.

Algorithme de Montgomery (détails d'exécution)

- Nous considérons les représentations polynomiales,

$$\begin{aligned} A(X) &= a_0 + a_1X + a_2X^2 + ... + a_{m-1}X^{m-1} \\ B(X) &= b_0 + b_1X + b_2X^2 + ... + b_{m-1}X^{m-1} \\ P(X) &= p_0 + p_1X + p_2X^2 + ... + p_{m-1}X^{m-1} + X^m \\ P'(X) &= p'_0 + p'_1X + p'_2X^2 + ... + p'_{m-1}X^{m-1} \end{aligned}$$

- Nous décomposons le calcul en calcul matriciel avec
  - pour les parties basses, le calcul de $Q(X)$
  - pour les parties hautes, le calcul du résultat $T(X)$

Algorithme de Montgomery (détails d'exécution)

Décomposition du calcul de $Q(X)$: *(on intègre la partie basse du produit)*

$$Q(X) = -\begin{pmatrix} p'_0 & 0 & \dots & 0 & 0 \\ p'_1 & p'_0 & \dots & 0 & 0 \\ & & & & \\ p'_{m-2} & p'_{m-3} & \dots & p'_0 & 0 \\ p'_{m-1} & p'_{m-2} & \dots & p'_1 & p'_0 \end{pmatrix} \begin{pmatrix} a_0 & 0 & \dots & 0 & 0 \\ a_1 & a_0 & \dots & 0 & 0 \\ & & & & \\ a_{m-2} & a_{m-3} & \dots & a_0 & 0 \\ a_{m-1} & a_{m-2} & \dots & a_1 & a_0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \dots \\ b_{m-2} \\ b_{m-1} \end{pmatrix}$$

Décomposition du calcul de $T(X)$:*(on intègre la partie haute du produit)*

$$\begin{pmatrix} 0 & a_{m-1} & \dots & a_2 & a_1 \\ 0 & 0 & \dots & a_2 & a_1 \\ & & & & \\ 0 & 0 & \dots & a_{m-1} & a_{m-2} \\ 0 & 0 & \dots & 0 & a_{m-1} \\ 0 & 0 & \dots & 0 & 0 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \dots \\ b_{m-3} \\ b_{m-2} \\ b_{m-1} \end{pmatrix} + \begin{pmatrix} 1 & p_{m-1} & \dots & p_2 & p_1 \\ 0 & 1 & \dots & p_2 & p_1 \\ & & & & \\ 0 & 0 & \dots & p_{m-1} & p_{m-2} \\ 0 & 0 & \dots & 1 & p_{m-1} \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \\ \dots \\ q_{m-3} \\ q_{m-2} \\ q_{m-1} \end{pmatrix}$$

Algorithme de Montgomery (complexité du cas général)

- Complexité en nombre d'opérations élémentaires sur $GF(2)$:
  - $m^2 + (m-1)^2$ multiplications (AND)
  - $(m-1)^2 + (m-2)^2 + m$ additions (XOR).
- Cet algorithme s'utilise généralement en représentation de Montgomery : $\widetilde{A}(X) = A(X) \times R(X) \pmod{P}(X)$
- Cet algorithme se généralise facilement à $GF(p^k)$

Montgomery itératif dans $GF(2^m)$ avec $R(X) = X^m$

Entrées : $A(X)$ et $B(X)$ *deux polynômes de degrés inférieurs à* m

Résultat : $T(X) = A(X) * B(X) * R^{-1}(X) \bmod P(X)$

Connues : $P'(X)$, $R(X)$

Initialisation $T(X) = 0$

Boucle pour $i = 0$ à $m - 1$

$$T(X) = T(X) + a_i * B(X)$$
$$T(X) = (T(X) + t_0 * P(X))/X$$

Montgomery itératif dans $GF(2^m)$ avec $R(X) = X^m$

- A chaque itération réduction par X d'où au final réduction par $R(X) = X^m$.
- De plus, $P(X)$ est irréductible donc son terme de degré 0 vaut 1, idem pour $P'(X)$.
- La complexité en nombre de portes logiques :
  - $2m^2$ *XOR* (pour les sommes)
  - et $2m^2$ *AND* (pour les produits)

Méthode de Mastrovito

Principe de l'approche

- $GF(2^m)$ est définie par une racine α d'un polynôme irréductible $P(X)$ de degré m.
- Les éléments de $GF(2^m)$ sont écrit dans la base canonique $\{1, \alpha, \alpha^2, ..., \alpha^{m-1}\}$: $A = \sum_{i=0}^{m-1} a_i \alpha^i$ et $B = \sum_{i=0}^{m-1} b_i \alpha^i$.
- Le produit $A \times B$ dans $GF(2^m)$ est noté $C = \sum_{i=0}^{m-1} c_i \alpha^i$.

Mastrovito propose de construire Z, une matrice carrée $m \times m$ dépendant de A, telle que :

$$C = Z \times B$$

Méthode de Mastrovito

Construction de l'approche

Mastrovito donne un algorithme de construction de Z :

1. on construit la matrice $(m-1) \times m$, Q correspondant à l'écriture des X^k pour $k \geq m$ modulo $P(X)$:

$$\begin{pmatrix} X^m \\ X^{m+1} \\ \dots \\ X^{2m-2} \end{pmatrix} = Q \times \begin{pmatrix} X^0 \\ X^1 \\ \dots \\ X^{m-1} \end{pmatrix}$$

2. puis la matrice Z où :

$$z_{i,j} = \begin{cases} a_i \text{ pour } j = 0,\ i = 0 \dots m-1 \\ u(i-j) * a_{i-j} + \sum_{t=0}^{j-1} q_{j-1-t,i} * a_{m-1-t}, \text{ sinon , où } u(t) = \begin{cases} 1 \text{ si } t \geq 0 \\ 0 \text{ sinon} \end{cases} \end{cases}$$

Méthode de Mastrovito

Coût de l'approche

- La complexité de cette méthode vient en partie de la construction de la matrice Z qui peut nécessiter $m^3/2$ *And* et *Xor*, le choix du polynôme est donc fondamental.
- Avec des polynômes de la forme $X^m + X + 1$ le produit peut se faire avec $m^2 - 1$ XOR et m^2 AND.
- Il existe des variantes pour des polynômes
  - composés que de 1 (all-one polynomial)

$P(X) = 1 + X + X^2 + ... + X^m$, *dans ce cas* $X^{m+1} \equiv 1 \pmod{P(X)}$

  - ou encore régulièrement espacés

$P(X) = 1 + X^\Delta + X^{2\Delta} + ... + X^{k\Delta=m}$, ici $X^{(k+1)\Delta} \equiv 1 \pmod{P(X)}$.

Méthode de Mastrovito I

Exemple avec un trinôme

Considérons $GF(2^7)$ muni d'une base canonique $\{1, \alpha, \alpha^2, ..., \alpha^6\}$ où α est une racine du polynôme irréductible $P(X) = X^7 + X + 1$. Ainsi,

$$\begin{array}{lll} \alpha^7 & = \alpha + 1 & \rightarrow (1,1,0,0,0,0,0) \\ \alpha^8 & = \alpha^2 + \alpha & \rightarrow (0,1,1,0,0,0,0) \\ \alpha^9 & = \alpha^3 + \alpha^2 & \rightarrow (0,0,1,1,0,0,0) \\ \alpha^{10} & = \alpha^4 + \alpha^3 & \rightarrow (0,0,0,1,1,0,0) \\ \alpha^{11} & = \alpha^5 + \alpha^4 & \rightarrow (0,0,0,0,1,1,0) \\ \alpha^{11} & = \alpha^6 + \alpha^5 & \rightarrow (0,0,0,0,0,1,1) \end{array}$$

$$Q = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Méthode de Mastrovito II

Exemple avec un trinôme

$$Z = \begin{pmatrix} a_0 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \\ a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 \\ a_4 & a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 + a_6 \end{pmatrix}$$

Méthode de Mastrovito

Exemple avec un All-One

Pour $P(X) = 1 + X + X^2 + ... + X^m$ la matrice Z peut se décomposer sous la forme $Z = Z_1 + Z_2$ avec :

$$Z_1 = \begin{pmatrix} a_0 & 0 & a_{m-1} & & \dots & a_3 & a_2 \\ a_1 & a_0 & 0 & a_{m-1} & & a_4 & a_3 \\ & & & & \dots & & \\ & & & & \dots & & \\ a_{m-2} & a_{m-3} & & & & a_0 & 0 \\ a_{m-1} & a_{m-2} & & & & a_1 & a_0 \end{pmatrix}$$

et

$$Z_2 = \begin{pmatrix} 0 & a_{m-1} & a_{m-2} & & a_1 \\ 0 & a_{m-1} & a_{m-2} & & a_1 \\ & & & \dots & \\ 0 & a_{m-1} & a_{m-2} & & a_1 \end{pmatrix} \text{ (ie ligne } X^m\text{)}$$

Matrices de Toeplitz

Definition

Une $n \times n$ matrice de Toeplitz est de la forme $[t_{i,j}]_{1 \leq i,j \leq n}$ tels que $t_{i,j} = t_{i-1,j-1}$ pour $i, j \geq 1$.

$$T = \begin{bmatrix} t_n & t_{n+1} & t_{n+2} & \cdots & t_{2n-1} \\ t_{n-1} & t_n & t_{n+1} & & \vdots \\ t_{n-2} & t_{n-1} & t_n & & \vdots \\ \vdots & & & & \vdots \\ t_1 & & & t_{n-1} & t_n \end{bmatrix}$$

Note. Une addition de 2 Toeplitz demande seulement $2n - 1$ additions.

Matrices de Toeplitz

Definition

Une $n \times n$ matrice de Toeplitz est de la forme $[t_{i,j}]_{1 \leq i,j \leq n}$ tels que $t_{i,j} = t_{i-1,j-1}$ pour $i, j \geq 1$.

$$T = \begin{bmatrix} t_n & t_{n+1} & t_{n+2} & \cdots & t_{2n-1} \\ t_{n-1} & t_n & t_{n+1} & & \vdots \\ t_{n-2} & t_{n-1} & t_n & & \vdots \\ \vdots & & & & \vdots \\ t_1 & & & t_{n-1} & t_n \end{bmatrix}$$

Note. Une addition de 2 Toeplitz demande seulement $2n - 1$ additions.

Matrices de Toeplitz

Definition

Une $n \times n$ matrice de Toeplitz est de la forme $[t_{i,j}]_{1 \leq i,j \leq n}$ tels que $t_{i,j} = t_{i-1,j-1}$ pour $i, j \geq 1$.

$$T = \begin{bmatrix} t_n & t_{n+1} & t_{n+2} & \cdots & t_{2n-1} \\ t_{n-1} & t_n & t_{n+1} & & \vdots \\ t_{n-2} & t_{n-1} & t_n & & \vdots \\ \vdots & & & & \vdots \\ t_1 & & & t_{n-1} & t_n \end{bmatrix}$$

Note. Une addition de 2 Toeplitz demande seulement $2n - 1$ additions.

Le Produit matrice de Toeplitz - vecteur *(Fan-Hasan 2007)*

Si T une Toeplitz $n \times n$ avec $2|n$

$$T = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix}$$

Alors $T \cdot V$ est égale à

$$T \cdot V = \begin{bmatrix} P_0 + P_2 \\ P_1 + P_2 \end{bmatrix}$$

où P_0, P_1, P_2 peuvent être calculés récursivement

$$\begin{aligned} P_0 &= (T_0 + T_1) \cdot V_1, \\ P_1 &= (T_1 + T_2) \cdot V_0, \\ P_2 &= T_1 \cdot (V_0 + V_1), \end{aligned}$$

Complexité du produit Toeplitz - vecteur

Fan et Hasan ont aussi proposé une 3-way split method.

	Two-way split method	Three-way split method
# AND	$n^{\log_2(3)}$	$n^{\log_3(6)}$
# XOR	$5.5n^{\log_2(3)} - 6n + 0.5$	$\frac{24}{5}n^{\log_3(6)} - 5n + \frac{1}{5}$
Delay	$T_A + 2\log_2(n)D_X$	$D_A + 3\log_3(n)D_X$

D_A est le délai d'un AND , D_X celui d'un XOR.

Application du produit Toeplitz - vecteur

- Nous avons vu que $C(X) = A(X) \times B(X) \bmod P(X)$ peut s'écrire $C(X) = Z \times B(X)$ où Z est une matrice $m \times m$
- On s'aperçoit que via des permutations circulaires de lignes voire aussi de colonnes nous pouvons transformer Z en une Toeplitz.
- C'est ce qu'on fait Fan-Hasan avec des trinômes, des pentanômes (2006) et des All-One (2007), puis Hasan-Nègre (2010) avec des quadrinômes (où $Q(X) = (X+1)P(X)$)

Application du produit Toeplitz - vecteur

Exemple

$GF(2^7)$ avec comme polynôme irréductible $P(X) = X^7 + X + 1$

$$Z = \begin{pmatrix} a_0 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \\ a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 \\ a_4 & a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 + a_6 \end{pmatrix}$$

se transforme en Toeplitz

$$Z' = \begin{pmatrix} a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 & a_3 + a_2 & a_2 + a_1 \\ a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 & a_3 + a_2 \\ a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 & a_4 + a_3 \\ a_4 & a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 & a_5 + a_4 \\ a_5 & a_4 & a_3 & a_2 & a_1 & a_0 + a_6 & a_6 + a_5 \\ a_6 & a_5 & a_4 & a_3 & a_2 & a_1 & a_0 + a_6 \\ a_0 & a_6 & a_5 & a_4 & a_3 & a_2 & a_1 \end{pmatrix}$$

Ici une permutation 1ere ligne $->$ dernière ligne permet d'avoir une Toeplitz, si on veut une cohérence des bases alors on fait de même sur les colonnes

Multiplication dans $GF(2^n)$

Approches liées aux bases

Base normale de $GF(2^m)$

- Une base normale de $GF(2^m)$ est de la forme $\{\alpha, \alpha^2, \alpha^{2^2} ..., \alpha^{2^{m-1}}\}$ où α racine de $P(X)$ irréductible de degré m. *(les α^{2^i} sont racines de $P(X)$, propriété du frobenius, $P(X)^{2^i} = P(X^{2^i})$)*
- A de $GF(2^m)$: $A = (a_0, a_1, ..., a_{m-1}) = \sum_{i=0}^{m-1} a_i \alpha^{2^i}$.
- L'élévation au carré sous cette représentation se réduit à une simple permutation circulaire :

nous avons $A^2 = \sum_{i=0}^{m-1} a_i \alpha^{2^{i+1}}$ *or* $\alpha^{2^m} = \alpha$,

donc, $A^2 = a_{m-1}\alpha + \sum_{i=1}^{m-1} a_{i-1} \alpha^{2^i}$ *autrement dit* $A^2 = (a_{m-1}, a_0, ..., a_{m-2})$.

Base normale : Multiplication de Massey-Omura 1986

- Produit dans $GF(2^m)$: $D = A \times B = A \times M \times B^t$ où :

$$M = \begin{pmatrix} \alpha^{2^0+2^0} & \alpha^{2^0+2^1} & \dots & \alpha^{2^0+2^j} & \dots & \alpha^{2^0+2^{m-2}} & \alpha^{2^0+2^{m-1}} \\ \alpha^{2^1+2^0} & \alpha^{2^1+2^1} & \dots & \alpha^{2^1+2^j} & \dots & \alpha^{2^1+2^{m-2}} & \alpha^{2^1+2^{m-1}} \\ \alpha^{2^i+2^0} & \alpha^{2^i+2^1} & \dots & \alpha^{2^i+2^j} & \dots & \alpha^{2^i+2^{m-2}} & \alpha^{2^i+2^{m-1}} \\ \alpha^{2^{m-1}+2^0} & \alpha^{2^{m-1}+2^1} & \dots & \alpha^{2^{m-1}+2^j} & \dots & \alpha^{2^{m-1}+2^{m-2}} & \alpha^{2^{m-1}+2^{m-1}} \end{pmatrix}$$

- $M = M_0\ \alpha + M_1\ \alpha^2 + ... + M_{m-1}\ \alpha^{2^{m-1}}$ où les M_i sont composées de 0 et de 1.
- Ainsi $D = A \times B$ s'obtient en évaluant ses coordonnées $d_{m-1-k} = A \times M_{m-1-k} \times B^t$ pour $k = 0, ..., m-1$.

Base normale : Multiplication de Massey-Omura 1986

Stockage d'une seule matrice

- $D^{2^k} = A^{2^k} \times B^{2^k}$ et l'élévation à l'exposant 2^k revient à k décalages circulaires :
  $d_{m-1-k} = A^{2^k} \times M_{m-1} \times (B^{2^k})^t$ pour $k = 0, ..., m-1$
- La complexité dépend du nombre de 1 dans M_{m-1}, donc du choix de m et de $P(X)$.
- Ce nombre est minoré par $2m-1$. Lorsque cette borne est atteinte la base est dite optimale
- Si $P(X)$ a tous ses coefficients à 1 (All-One) , cette borne est atteinte et la complexité est m^2 AND et $2m^2 - 2m$ XOR.

Base normale : Multiplication de Massey-Omura 1986

Exemple

Considérons le corps fini $GF(2^4)$ muni d'une base normale $(\alpha^{2^0}, \alpha^{2^1}, \alpha^{2^2}, \alpha^{2^3})$ où α est une racine du polynôme irréductible $P(X) = X^4 + X^3 + 1$

$$M = \begin{pmatrix} \alpha^2 & \alpha + \alpha^2 + \alpha^8 & \alpha + \alpha^4 & \alpha + \alpha^4 + \alpha^8 \\ \alpha + \alpha^2 + \alpha^8 & \alpha^4 & \alpha + \alpha^2 + \alpha^4 & \alpha^2 + \alpha^8 \\ \alpha + \alpha^4 & \alpha + \alpha^2 + \alpha^4 & \alpha^8 & \alpha^2 + \alpha^4 + \alpha^8 \\ \alpha + \alpha^4 + \alpha^8 & \alpha^2 + \alpha^8 & \alpha^2 + \alpha^4 + \alpha^8 & \alpha \end{pmatrix}$$

Ainsi nous avons

$$M_3 = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$

Base normale : Massey et Omura modifié HWB 1993

- Si $P(X)$ est un All-One, la complexité peut être réduite à m^2 portes AND et $m^2 - 1$ portes XOR, en décomposant la matrice M_{m-1}
- $M_{m-1} = (P + Q) \pmod 2$

$$\text{avec } P_{i,j} = \begin{cases} 1 & \text{si } i = (m/2 + j) \bmod m \\ 0 & \text{sinon} \end{cases}$$

- Dans ce cas, si on pose $T^{(k)}$ telle que : $B^{2^k} = BT^{(k)}$, nous remarquons que $T^{(k)} P T^{(k)t} = P$. Nous obtenons ainsi

$$d_{m-1-k} = A \times P \times B^t + A^{2^k} \times Q \times (B^{2^k})^t \text{ pour } k = 0, ..., m-1$$

Base normale : Massey et Omura modifié HWB 1993

Exemple

Considérons $GF(2^4)$ muni d'une base normale $(\alpha^{2^0}, \alpha^{2^1}, \alpha^{2^2}, \alpha^{2^3})$ où α est une racine du polynôme irréductible $P(X) = X^4 + X^3 + X^2 + X + 1$. Nous obtenons , en posant $\gamma = \alpha + \alpha^2 + \alpha^4 + \alpha^8$:

$$M = \begin{pmatrix} \alpha^2 & \alpha^8 & \gamma & \alpha^4 \\ \alpha^8 & \alpha^4 & \alpha & \gamma \\ \gamma & \alpha & \alpha^8 & \alpha^2 \\ \alpha^4 & \gamma & \alpha^2 & \alpha \end{pmatrix}$$

Ainsi, nous avons :

$$M_3 = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} = P + Q = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Bases duales dans $GF(2^m)$

Définition

- **Fonction Trace** : forme linéaire $Tr(u) = \sum_{i=0}^{m-1} u^{2^i} \in GF(2)$

  avec $u \in GF(2^m)$ *(polynôme minimal de* α, $P(X) = \prod_{i=0}^{m-1}(X - \alpha^{2^i}) \in GF(2)[X]$*)*
- **Bases duales** : deux bases $\{\lambda_i, i = 0..m-1\}$ et $\{\nu_j, j = 0..m-1\}$ sont duales si $Tr(\lambda_i.\nu_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
- **Changement de base** :

  $Tr(\nu_j.x) = x_j$ où x_j avec $x = \sum_{j=0}^{m-1} x_j \lambda_j$

Bases duales dans $GF(2^m)$ (suite)

Définition plus générale

- **Autre forme linéaire** : $f(u) = Tr(\beta.u)$ où $\beta \in GF(2^k)$
- **Bases duales** si $Tr(\beta.\lambda_i.\nu_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
- **Changement de base** :

$$Tr(\beta.\lambda_j.x) = x_j \text{ où } x_j \text{ avec } x = \sum_{j=0}^{m-1} x_j \lambda_j$$

Multiplication avec les Bases duales dans $GF(2^m)$

- On considère ici la base canonique $\{\alpha^i, i = 0..m-1\}$ et une base duale par (f, β)
- Soient a, b et c dans $GF(2^m)$: $c = a \times b$

$$\begin{pmatrix} Tr(b\beta) & Tr(b\beta\alpha) & .. & Tr(b\beta\alpha^{m-1}) \\ Tr(b\beta\alpha) & Tr(b\beta\alpha^2) & .. & Tr(b\beta\alpha^m) \\ \\ Tr(b\beta\alpha^{m-1}) & Tr(b\beta\alpha^2) & .. & Tr(b\beta\alpha^{2m-2}) \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \\ a_{m-1} \end{pmatrix} = \begin{pmatrix} Tr(c\beta) \\ Tr(c\beta\alpha) \\ \\ Tr(c\beta\alpha^{m-1}) \end{pmatrix}$$

  - la première ligne correspond aux coordonnées de b dans la base duale,
  - les coordonnées de a sont dans la base canonique,
  - c est obtenu dan la base duale.

- **but** : trouver une fonction f simple où la base duale est une simple permutation de la base canonique
[Ber82, STFT96, HWB98, Gol02]

Multiplication avec les Bases duales dans $GF(p^m)$

Exemple

Dans le corps fini $GF(2^4)$, considérons la base canonique $(1, \alpha, \alpha^2, \alpha^3)$ où α est une racine du polynôme irréductible $P(X) = X^4 + X^3 + 1$.
Considérons la base

$$(\alpha^{12} = \alpha + 1, \alpha^{11} = \alpha^3 + \alpha^2 + 1, \alpha^{10} = \alpha^3 + \alpha, \alpha^{13} = \alpha^2 + \alpha)$$

Elle vérifie $Tr(\alpha^{10}) = Tr(\alpha^{11}) = Tr(\alpha^{13}) = Tr(\alpha^{14}) = Tr(1) = 0$, et $Tr(\alpha^{12}) = Tr(\alpha) = 1$.
Ainsi les bases $(1, \alpha, \alpha^2, \alpha^3)$ et $(\alpha^{12}, \alpha^{11}, \alpha^{10}, \alpha^{13})$ sont duales.
Soient $A = \alpha^{12} = (1, 1, 0, 0)$ et $B = \alpha^7 = (0, 1, 1, 1)$ dans la base canonique, et dans la base duale $A = \alpha^{12} = (1, 0, 0, 0)$ et $B = \alpha^7 = (0, 1, 1, 0)$. Nous avons,

$$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

Nous pouvons vérifier que $C = \alpha^4 = (1, 0, 1, 0)$ dans la base duale et $C = (1, 0, 0, 1)$ dans la base canonique.

Dans le corps fini $GF(2^4)$, considérons la base canonique $(1, \alpha, \alpha^2, \alpha^3)$ où α est une racine du polynôme irréductible $P(X) = X^4 + X^3 + 1$.
Ici, nous ne prenons plus la fonction trace pour construire une base duale, nous considérons la fonction linéaire $Tr(\alpha^{10}u)$ qui est simplement la projection de la troisième coordonnée. La base duale obtenue est une simple permutation de la base canonique $(\alpha^2, \alpha, 1, \alpha^3)$.
Dans ce cas les changement de base sont immédiats et le produit de $A = \alpha^{12}$ par $B = \alpha^7$ devient :

$$\begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix}$$

Nous vérifions là encore que $C = \alpha^4$.

Inversion dans un corps fini

Algorithme d'Euclide Etendu

Construction de l'algorithme

- Calcul de l'inverse de a modulo b en utilisant l'égalité de Bezout $b.u_1 + a.u_2 = \gcd(a, b)$.
- Nous définissons deux triplets $U = (u_1, u_2, u_3)$ et $V = (v_1, v_2, v_3)$ tels que :

$$\begin{aligned} u_1 b + u_2 a &= u_3 \\ v_1 b + v_2 a &= v_3 \end{aligned}$$

- On initialise $(u_1, u_2, u_3) = (1, 0, b)$ et $(v_1, v_2, v_3) = (0, 1, a)$
- On applique l'algorithme du pgcd d'Euclide sur u_3 et v_3 en maintenant les égalités précédentes.

En réalité les termes d'indice 2 sont inutiles pour le calcul de l'inverse et ne sont donc pas implantés.

Algorithme d'Euclide Etendu

L'algorithme dans $GF(p)$

Initialisation $u_1 \leftarrow 1 \quad u_2 \leftarrow 0 \quad u_3 \leftarrow p$
$v_1 \leftarrow 0 \quad v_2 \leftarrow 1 \quad v_3 \leftarrow a$

Boucle while $v_3 \neq 0$

$$q = \lfloor u_3 / v_3 \rfloor$$
$$t_1 \leftarrow u_1 - q.v_1 \quad t_2 \leftarrow u_2 - q.v_2 \quad t_3 \leftarrow u_3 - q.v_3$$
$$u_1 \leftarrow v_1 \quad u_2 \leftarrow v_2 \quad u_3 \leftarrow v_3$$
$$v_1 \leftarrow t_1 \quad v_2 \leftarrow t_2 \quad v_3 \leftarrow t_3$$

Résultat $u_2 \equiv a^{-1} \bmod p$

Algorithme d'Euclide Etendu

L'algorithme dans $GF(2^m)$ version polynomiale

Initialisation $U_1 \leftarrow 1 \quad U_2 \leftarrow 0 \quad U_3 \leftarrow P(X)$
$V_1 \leftarrow 0 \quad V_2 \leftarrow 1 \quad V_3 \leftarrow A(X)$

Boucle while $V_3 \neq 0$

$n = deg(U_3) - deg(V_3)$
$T_1 \leftarrow U_1 - X^n.V_1 \qquad t_2 \leftarrow U_2 - X^n.V_2 \quad T_3 \leftarrow U_3 - X^n.V_3$

If $deg(t_3) \geq deg(v_3)$

$U_1 \leftarrow T_1 \quad U_2 \leftarrow T_2 \quad U_3 \leftarrow T_3$

then

$U_1 \leftarrow V_1 \quad U_2 \leftarrow V_2 \quad U_3 \leftarrow V_3$
$V_1 \leftarrow T_1 \quad V_2 \leftarrow T_2 \quad V_3 \leftarrow T_3$

Résultat $U_2 \equiv A^{-1} \bmod P(X)$

Sur $GF(2^m)$, cet algorithme est en $O(k)$, (à chaque pas le degré le plus haut diminue au moins de un)

Algorithme d'Euclide Etendu

Exemple dans $GF(2^4)$

Polynôme irréductible $P(X) = X^4 + X^3 + 1$. *Considérons* $A(X) = X^2 + 1$

$u_1(X) = 1 \quad u_2(X) = 0 \quad u_3(X) = P(X) = X^4 + X^3 + 1$
$v_1(X) = 0 \quad v_2(X) = 1 \quad v_3(X) = A(X) = X^2 + 1$

$n = 2 \quad u_1(X) = 1 \quad u_2(X) = X^2 \quad u_3(X) = X^3 + X^2 + 1$
$\quad\quad v_1(X) = 0 \quad v_2(X) = 1 \quad v_3(X) = X^2 + 1$

$n = 1 \quad u_1(X) = 1 \quad u_2(X) = X^2 + X \quad u_3(X) = X^2 + X + 1$
$\quad\quad v_1(X) = 0 \quad v_2(X) = 1 \quad v_3(X) = X^2 + 1$

$n = 0 \quad u_1(X) = 0 \quad u_2(X) = 1 \quad u_3(X) = X^2 + 1$
$\quad\quad v_1(X) = 1 \quad v_2(X) = X^2 + X + 1 \quad v_3(X) = X$

$n = 1 \quad u_1(X) = 1 \quad u_2(X) = X^2 + X + 1 \quad u_3(X) = x$
$\quad\quad v_1(X) = X \quad v_2(X) = X^2 + X^3 + X + 1 \quad v_3(X) = 1$

$n = 1 \quad u_1(X) = X \quad u_2(X) = X^2 + X^3 + X + 1 \quad u_3(X) = 1$
$\quad\quad v_1(X) = 1 + X^2 \quad v_2(X) = X^4 + X^3 + 1 \quad v_3(X) = 0$

Nous vérifions que $(X^2 + X^3 + X + 1)(X^2 + 1) = 1 \bmod (X^4 + X^3 + 1)$ *et donc que* $X^2 + X^3 + X + 1$ *est l'inverse de* $X^2 + 1$ *modulo* $P(X)$.

Par le théorème de Fermat-Euler

- **Théorème** : Tout élément non nul β d'un corps fini d'ordre q satisfait : $\beta^q = \beta$, autrement dit β est racine de $X^q = X$
- **Corollaire** : Tout élément non nul β d'un corps fini d'ordre q satisfait : $\beta^{q-2} = \beta^{-1}$,
- Dans $GF(p)$ cet approche est coûteuse demandant une exponentiation à la puissance $p - 2$
- Dans $GF(2^m)$ que l'inverse est égal à $\beta^{-1} = \beta^{2^m-2}$. *L'algorithme d'exponentiation utilise une stratégie de produits et carrés liée à l'écriture binaire de l'exposant. Vu la forme de ce dernier dans le cas présent, de nombreuses astuces permettent d'accélérer le calcul [TYT01]. La complexité reste malgré tout en $O(m)$.*

Par le théorème de Fermat-Euler

Exemple dans $GF(2^4)$

Soit le corps fini $GF(2^4)$ défini par la base canonique $(1, \alpha, \alpha^2, \alpha^3)$ où α est une racine du polynôme irréductible $P(X) = X^4 + X^3 + 1$. Nous avons $2^4 - 2 = 14$. Soit $A(X) = X^2 + 1$, son inverse est donc

$$A^{-1}(X) = A^{14}(X) = (X^2 + 1)^{14} \bmod (X^4 + X^3 + 1)$$

L'écriture binaire de 14 est 1110 d'où

$$(X^2 + 1)^{14} = ((((X^2 + 1)^2)(X^2 + 1))^2)(X^2 + 1))^2 \bmod (X^4 + X^3 + 1)$$

Le calcul pas à pas donne,

$$\begin{array}{lll}
(X^2+1)^2 & = X^3 & \\
((X^2+1)^2)(X^2+1) & = (X^2+1)^3 & = X+1 \\
(((X^2+1)^2)(X^2+1))^2 & = (X^2+1)^6 & = X^2+1 \\
((((X^2+1)^2)(X^2+1))^2)(X^2+1) & = (X^2+1)^7 & = X^3 \\
(((((X^2+1)^2)(X^2+1))^2)(X^2+1))^2 & = (X^2+1)^{14} & = X^3+X^2+X+1
\end{array}$$

Par le théorème de Fermat-Euler

Exemple dans $GF(2^{31})$

Dans le cas de $GF(2^{31})$ nous devons calculer $\beta^{2^{31}-2}$, or $2^{31}-2 = 2147483646$ s'écrit en binaire 11111111111111111111111111111110. Si nous avons une base normale, nous pouvons calculer $\beta^{2^{31}-2}$ de la façon suivante :

calcul	*valeur*	*exposant*
β^2	$= \beta^2$	10
$\beta^2\beta$	$= \beta^3$	11
$(\beta^3)^{2^2}$	$= \beta^{12}$	1100
$\beta^{12}\beta^3$	$= \beta^{15}$	1111
$(\beta^{15})^{2^4}$	$= \beta^{240}$	11110000
$\beta^{240}\beta^{15}$	$= \beta^{255}$	11111111
$(\beta^{255})^{2^8}$	$= \beta^{65280}$	1111111100000000
$\beta^{65280}\beta^{255}$	$= \beta^{65535}$	1111111111111111
$(\beta^{65535})^{2^{15}}$	$= \beta^{2147450880}$	11111111111111110000000000000000
$(\beta^{255})^{2^7}$	$= \beta^{32640}$	111111110000000
$(\beta^{15})^{2^3}$	$= \beta^{120}$	1111000
$(\beta^3)^{2^1}$	$= \beta^6$	110
$\beta^{2147450880}\beta^{32640}$	$= \beta^{2147483520}$	11111111111111111111111110000000
$\beta^{120}\beta^6$	$= \beta^{126}$	1111110
$\beta^{2147483520}\beta^{126}$	$= \beta^{2147483646}$	11111111111111111111111111111110

Another Approach: Residue Systems
Introduction to Residue Systems

Introduction to Residue Systems

- In some applications, like cryptography, we use finite field arithmetics on huge numbers or large polynomials.
- Residue systems are a way to distribute the calculus on small arithmetic units.
- Are these systems suitable for finite field arithmetics?

Residue Number Systems in $\mathbb{F}_p$, p prime

- Modular arithmetic mod p, elements are considered as integers.
- Residue Number System
  - RNS base: a set of coprime numbers $(m_1, ..., m_k)$
  - RNS representation: $(a_1, ..., a_k)$ with $a_i = |A|_{m_i}$
  - Full parallel operations mod M with $M = \prod_{i=1}^{k} m_i$
    $(|a_1 \otimes b_1|_{m_1}, \ldots, |a_n \otimes b_n|_{m_n}) \rightarrow A \otimes B \pmod{M}$
- Very fast product, but an extension of the base could be necessary and a reduction modulo p is needed.

Residue Number Systems in $\mathbb{F}_p$, p prime

- $\Phi(m) = \sum_{\substack{p \leq m \\ p \text{ prime}}} \log p = \log \prod_{\substack{p \leq m \\ p \text{ prime}}} p \sim m$
- If $2^{m-1} \leq M < 2^m$ then the size moduli is of order $\mathcal{O}(\log m)$.
- In other words, if addition and multiplication have complexities of order $\Theta(f(m))$ then in RNS the complexities become $\Theta(f(\log m))$.

Lagrange representations in $\mathbb{F}_{p^k}$ with $p > 2k$

- Arithmetic modulo $I(X)$, an irreducible $\mathbb{F}_p$ polynomial of degree k. Elements of $\mathbb{F}_{p^k}$ are considered as $\mathbb{F}_p$ polynomials of degree lower than k.
- Lagrange representation
  - is defined by k different points $e_1, ...e_k$ in $\mathbb{F}_p$. ($k \leq p$.)
  - A polynomial $A(X) = \alpha_0 + \alpha_1 X + ... + \alpha_{k-1} X^{k-1}$ over $\mathbb{F}_p$ is given in Lagrange representation by:

$$(a_1 = A(e_1), ..., a_k = A(e_k)).$$

  - Remark: $a_i = A(e_i) = A(X) \bmod (X - e_i)$. If we note $m_i(X) = (X - e_i)$, we obtain a similar representation as RNS.
- Operations are made independently on each $A(e_i)$ (like in FFT or Tom-Cook approaches). We need to extend to $2k$ points for the product.

Trinomial residue in $\mathbb{F}_{2^n}$

- Arithmetic modulo $I(X)$, an irreducible $\mathbb{F}_2$ polynomial of degree n. Elements of $\mathbb{F}_{2^n}$ are considered as $\mathbb{F}_2$ polynomials of degree lower than n.
- Trinomial representation
  - is defined by a set of k coprime trinomials $m_i(X) = X^d + X^{t_i} + 1$, with $k \times d \geq n$,
  - an element $A(X)$ is represented by $(a_1(X), ...a_k(X))$ with $a_i(X) = A(X) \bmod m_i(X)$.
  - This representation is equivalent to RNS.
- Operations are made independently for each $m_i(X)$

Residue Systems

- Residue systems could be an issue for computing efficiently the product.
- The main operation is now the modular reduction for constructing the finite field elements.
- The choice of the residue system base is important, it gives the complexity of the basic operations.

Modular reduction in Residue Systems

Reduction of Montgomery on $\mathbb{F}_p$

- The most used reduction algorithm is due to Montgomery (1985)[9]
- For reducing A modulo p,
  one evaluates $q = -(Ap^{-1}) \bmod 2^s$,
  then one constructs $R = (A + qp)/2^s$.
  The obtained value satisfies: $R \equiv A \times 2^{-s} \pmod{p}$ and $R < 2p$ if $A < p2^s$.
  We note $\text{Montg}(A, 2^s, p) = R$.
- Montgomery notation: $A' = A \times 2^s \bmod p$
  $\text{Montg}(A' \times B', 2^s, p) \equiv (A \times B) \times 2^s \pmod{p}$

Residue version of Montgomery Reduction

- The residue base is such that $p < M$
  (or $\deg M(X) \geq \deg I(X)$)
- We use an auxiliary base such that $p < M'$
  (or $\deg M'(X) \geq \deg I(X)$), M' and M coprime.
  (Exact product, and existence of M^{-1})
- Steps of the algorithm
  1. $Q = -(Ap^{-1}) \bmod M$ (calculus in base M)
  2. Extension of the representation of Q to the base M'
  3. $R = (A + Qp) \times M^{-1}$ (calculus in base M')
  4. Extension of the representation of R to the base M
- The values are represented in the two bases.

Extension of Residue System Bases (from M to M')

The extension comes from the Lagrange interpolation.
If $(a_1, ..., a_k)$ is the residue representation in the base M, then

$$A = \sum_{i=1}^{k} \left| a_i \times \left[\frac{M}{m_i} \right]_{m_i}^{-1} \right|_{m_i} \times \frac{M}{m_i} - \alpha M$$

The factor α can be, in certain cases, neglected or computed.[1]
Another approach consists in the Newton interpolation where A is correctly reconstructed. [4]
In the polynomial case, the term $-\alpha M$ vanishes.

Extension for Q

By the CRT

$$\widehat{Q} = \sum_{i=1}^{n} \left| q_i \, |M_i|_{m_i}^{-1} \right|_{m_i} M_i = Q + \alpha M$$

where $0 \leq \alpha < n$.

When $\widehat{Q}$ has been computed it is possible to compute $\widehat{R}$ as

$$\begin{aligned} \widehat{R} = (AB + \widehat{Q}p)M^{-1} &= (AB + Qp + \alpha Mp)M^{-1} \\ &= (AB + Qp)M^{-1} + \alpha p \end{aligned}$$

so that $\widehat{R} \equiv R \equiv ABM^{-1} \pmod{p}$, which is sufficient for our purpose. Also, assuming that $AB < pM$ we find that $\widehat{R} < (n+2)p$ since $\alpha < n$.

Extension R

Shenoy et Kumaresan (1989):

we have $(\sum_{i=1}^{n} M_i \left| |M_i|_{m_i}^{-1} r_i \right|_{m_i}) = R + \alpha \times M$

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

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

Extension of Residue System Bases

We first translate in an intermediate representation (MRS):

$$\begin{cases} \zeta_1 = a_1 \\ \zeta_2 = (a_2 - \zeta_1)\, m_1^{-1} \bmod m_2 \\ \zeta_3 = ((a_3 - \zeta_1)\, m_1^{-1} - \zeta_2)\, m_2^{-1} \bmod m_3 \\ \quad \vdots \\ \zeta_n = (\ldots((a_n - \zeta_1)\, m_1^{-1} - \zeta_2)\, m_2^{-1} - \cdots - \zeta_{n-1})\, m_{n-1}^{-1} \bmod m_n. \end{cases}$$

We evaluate A, with Horner's rule, as

$$A = (\ldots((\zeta_n\, m_{n-1} + \zeta_{n-1})\, m_{n-2} + \cdots + \zeta_3)\, m_2 + \zeta_2)\, m_1 + \zeta_1.$$

Features of the residue systems

- Efficient multiplication, the cost being the cost of one multiplication on one residue.
- Costly reduction: $O(k^{1.6})$ for trinomials [4] (annexe 11), $2k^2 + 3k \rightarrow O(k)$? for RNS [1] (annexe 6), $O(k^2) \rightarrow O(k)$ for Lagrange representation [5] (annexe 14).
- If we take into account that most of the operations are multiplications by a constant, the cost can be considerably smaller.

Applications to Cryptography

Elliptic curve cryptography

- The main idea comes from the efficiency of the product and the cost of the reduction in Residue Systems.
- We try to minimize the number of reductions. A reduction is not necessary after each operation. Clearly, for a formula like $A \times B + C \times D$, only one reduction is needed.
- Elliptic Curve Cryptography is based on points addition. We use appropriate forms (Hessian, Jacobi, Montgomery...) and coordinates: projective, Jacobian or Chudnowski...
- For 512 bits values Residues Systems for curves defined over a prime field, are more efficient than classical representations.[2]

Pairings

- To summarize we define a pairing as follows: let G_1 and G_2 be two additive abelian groups of cardinal n and G_3 a multiplicative group of cardinal n.
- A pairing is a function $e : G_1 \times G_2 \longrightarrow G_3$ which verifies the following properties: Bilinearity, Non-degeneracy.
- For pairings defined on an elliptic curve E over a finite field $\mathbb{F}_p$, we have $G_1 \subset E(\mathbb{F}_p)$, $G_2 \subset E(\mathbb{F}_{p^k})$ and $G_3 \subset \mathbb{F}p^k$, where k is the smallest integer such that n divides $p^k - 1$, k is called the embedded degree of the curve.

Pairings

- The construction of the pairing involves values over $\mathbb{F}_p$ and $\mathbb{F}_{p^k}$ into the formulas. An approach with Residue Systems, similar to the one made on ECC could be interesting.[3]
- k is most of the time chosen as a small power of 2 and 3 for algorithmic reasons. Residue arithmetics allow to pass over this restriction.
- With pairings, we can also imagine two levels of Residue Systems: one over $\mathbb{F}_p$ and one over $\mathbb{F}p^k$.

ANNEXES

Détails d'implémentations Residue Systems

Annexe $\mathbb{F}_p$

Table: Hamming weight $w(m_{i,j}^{-1})$ of the inverse of m_i modulo m_j.

	m_j					
m_i	2^k	2^k-1	$2^k-2^{t_1}-1$	$2^k-2^{t_2}-1$	$2^k-2^{t_1}+1$	$2^k-2^{t_2}+1$
2^k		1				
2^k-1	1		2	2		
$2^k-2^{t_1}-1$	$\left\lceil \frac{k}{t_1} \right\rceil$	1		$\frac{k-t_2}{t_1-t_2}$	2	
$2^k-2^{t_2}-1$	$\left\lceil \frac{k}{t_2} \right\rceil$	1	$\frac{k-t_1}{t_1-t_2}$			2
$2^k-2^{t_1}+1$	$\left\lceil \frac{k}{t_1} \right\rceil$	$\frac{k-1}{t_1-1}$	2			$\frac{k-t_1}{t_1-t_2}$
$2^k-2^{t_2}+1$	$\left\lceil \frac{k}{t_2} \right\rceil$	$\frac{k-1}{t_2-1}$		2	$\frac{k-t_1}{t_1-t_2}$	

Back to 8

Table: Hamming weight $w(m_{i,j}^{-1})$ of the inverse of m_i modulo m_j.

	m_j					
m_i	2^k	2^k-1	$2^k-2^{t+1}-1$	2^k-2^t-1	$2^k-2^{t+1}+1$	2^k-2^t+1
2^k		1				
2^k-1	1		2	2		
$2^k-2^{t+1}-1$	$\left\lceil \frac{k}{t+1} \right\rceil$	1		2	2	$\frac{k-t}{t-1}$
2^k-2^t-1	$\left\lceil \frac{k}{t} \right\rceil$	1	2		$\frac{k-t-1}{t-1}$	2
$2^k-2^{t+1}+1$	$\left\lceil \frac{k}{t+1} \right\rceil$	$\frac{k-1}{t}$	2	$\frac{k-t}{t-1}$		2
2^k-2^t+1	$\left\lceil \frac{k}{t} \right\rceil$	$\frac{k-1}{t-1}$	$\frac{k-t-1}{t-1}$	2	2	

Back to 8

Pair of 5 Moduli - Parallel mode

The dynamic range is
$M = 2^{320} - 2^{267} - 2^{265} - 2^{258} - 2^{256} + 2^{213} + 2^{206} - 2^{204} + 2^{195} - 2^{193} - 2^{157} - 2^{151} - 2^{148} - 2^{142} + 2^{138} + 2^{129} + 2^{95} + 2^{87} + 2^{85} + 2^{76} - 2^{67} + 2^{64} - 2^{31} + 2^{29} - 2^{22} + 2^{20} + 2^{11} - 2^{9} + 2^{2} - 1$ and
$M < M'$.

RNS bases for 5 moduli (P)	$m_1 = 2^{64} - 2^8 - 1$	3	$m'_1 = 2^{64} - 2^{10} + 1$	3
	$m_2 = 2^{64} - 2^{16} - 1$	3	$m'_2 = 2^{64} - 2^9 - 1$	3
	$m_3 = 2^{64} - 2^{22} - 1$	3	$m'_3 = 2^{64} - 2^2 + 1$	3
	$m_4 = 2^{64} - 2^{28} - 1$	3	$m'_4 = 2^{64} - 1$	2
	$m_5 = 2^{64}$	1	$m'_5 = 2^{64} - 2^{10} - 1$	3

Back to 8

Inverses $m_{i,j}^{-1}$ in basis $\mathcal{B}_5$	$\omega(m_{i,j}^{-1})$
$m_{1,2}^{-1} = 2^{48} + 2^{40} + 2^{32} + 2^{24} + 2^{16} + 2^8$	6
$m_{1,3}^{-1} = 2^{42} + 2^{28} + 2^{14}$	3
$m_{1,4}^{-1} = 2^{60} - 2^{56} - 2^{52} + 2^{44} + 2^{40} - 2^{32} + 2^{21} + 2^{16} - 2^{12} - 2^8 + 1$	11
$m_{1,5}^{-1} = 2^{56} - 2^{48} + 2^{40} - 2^{32} + 2^{24} - 2^{16} + 2^8 - 1$	8
$m_{2,3}^{-1} = 2^{42} + 2^{36} + 2^{30} + 2^{24} + 2^{18} + 2^{12} + 2^6$	7
$m_{2,4}^{-1} = 2^{36} + 2^{24} + 2^{12}$	3
$m_{2,5}^{-1} = 2^{48} - 2^{32} + 2^{16} - 1$	4
$m_{3,4}^{-1} = 2^{36} + 2^{30} + 2^{24} + 2^{18} + 2^{12} + 2^6$	6
$m_{3,5}^{-1} = 2^{64} - 2^{44} + 2^{22} - 1$	4
$m_{4,5}^{-1} = 2^{64} - 2^{56} + 2^{28} - 1$	4

Back to 8

Inverses ${m'}_{i,j}^{-1}$ in basis $\mathcal{B}'_5$	$\omega({m'}_{i,j}^{-1})$
${m'}_{1,2}^{-1} = 2^{62} - 2^{54} - 2^{46} - 2^{38} - 2^{30} - 2^{22} - 2^{14} - 2^8 + 2^6$	9
${m'}_{1,3}^{-1} = 2^{63} + 2^{61} - 2^{53} - 2^{45} - 2^{37} - 2^{29} - 2^{21} - 2^{13} - 2^5 - 2$	10
${m'}_{1,4}^{-1} = 2^{54} + 2^{45} + 2^{36} + 2^{27} + 2^{18} + 2^9 + 1$	7
${m'}_{1,5}^{-1} = 2^{63} - 2^9$	2
${m'}_{2,3}^{-1} = 2^{62} - 2^{54} - 2^{46} - 2^{38} - 2^{30} - 2^{22} - 2^{14} - 2^6 - 1$	9
${m'}_{2,4}^{-1} = 2^{64} - 2^{55} - 1$	3
${m'}_{2,5}^{-1} = 2^{55} - 2$	2
${m'}_{3,4}^{-1} = 2^{63} - 1$	2
${m'}_{3,5}^{-1} = 2^{54} + 2^{45} + 2^{36} + 2^{27} + 2^{18} + 2^9$	6
${m'}_{4,5}^{-1} = 2^{54} - 1$	2

Back to 8

Annexe $\mathbb{F}_{2^n}$

To compute

$$\psi = F \times T_j^{-1} \bmod T_i. \tag{1}$$

We note , $B_{j,i}(X) = T_j \bmod T_i$. Thus, (1)becomes

$$\psi = F \times B_{j,i}^{-1} \bmod T_i. \tag{2}$$

We evaluate (2) like a Montgomery reduction, where $B_{j,i}$ is the Montgomery factor:

1. $\phi = F \times T_i^{-1} \bmod B_{j,i}$,
   $(F + \phi.T_i$ multiple of $B_{j,i})$.
2. $\psi = (F + \phi\, T_i)/B_{j,i}$
   (with a division by $B_{j,i}$).

Back to 8

We remark that $B_{j,i}(X) = X^{t_j}(X^{t_i - t_j} + 1)$, (if $t_j < t_i$).
In order to evaluate (2), we compute into

$$\psi = \left(F \times (X^a)^{-1} \bmod T_i\right) \times \left(X^b + 1\right)^{-1} \bmod T_i. \qquad (3)$$

we evaluate $F \times (X^a)^{-1} \bmod T_i$ in two steps:

$$\phi = F \times T_i^{-1} \bmod X^a \qquad (4)$$
$$\psi = (F + \phi \times T_i)/X^a \qquad (5)$$

Back to 8

To end (3), we compute $F \times (X^b + 1)^{-1} \bmod T_i$. (degree of F is at most $d - 1$.) in four steps:

$$F = F \bmod (X^b + 1) \tag{6}$$
$$\phi = F \times T_i^{-1} \bmod (X^b + 1) \tag{7}$$
$$\rho = F + \phi \times T_i \tag{8}$$
$$\psi = \rho/(X^b + 1) \text{ (We have } \rho = \psi X^b + \psi \text{ thus } \rho \bmod X^b = \psi \bmod X^b\text{)} \tag{9}$$

Back to 8

Annexe $\mathbb{F}_{p^k}$

Let us consider the first $2k$ integers: we define $E = \{0, \ldots, k-1\}$ and $E' = \{k, \ldots, 2k-1\}$.
we can precompute $k-1$ constants
$C_j = ((e_j - e_1)(e_j - e_2)\ldots(e_j - e_{j-1}))^{-1} \bmod p$, for $2 \leq j \leq k$
and we can evaluate $(\hat{q}_1, \ldots, \hat{q}_k)$

$$\begin{cases} \hat{q}_1 = q_1 \bmod p, \\ \hat{q}_2 = (q_2 - \hat{q}_1)C_2 \bmod p, \\ \hat{q}_3 = (q_3 - (\hat{q}_1 + 2\hat{q}_2))C_3 \bmod p, \\ \quad \vdots \\ \hat{q}_k = (q_k - (\hat{q}_1 + (k-1)(\hat{q}_2 + (k-2)(\hat{q}_3 + \ldots \\ \qquad + 2\hat{q}_{k-1})\ldots))) C_k \bmod p. \end{cases} \tag{10}$$

Back to 8

$$q'_i = \big((\ldots(\hat{q}_k(e'_i - e_{k-1}) + \hat{q}_{k-1})(e'_i - e_{k-2}) + \cdots + \hat{q}_2)(e'_i - e_1) + \hat{q}_1\big) \bmod p. \quad (11)$$

$$\begin{cases} q'_1 = ((\ldots(\hat{q}_k \times 2 + \hat{q}_{k-1}) \times 3 + \cdots + \hat{q}_2) \times k + \hat{q}_1) \bmod p, \\ q'_2 = ((\ldots(\hat{q}_k \times 3 + \hat{q}_{k-1}) \times 4 + \cdots + \hat{q}_2) \times (k+1) + \hat{q}_1) \bmod p, \\ \quad \vdots \\ q'_k = ((\ldots(\hat{q}_k \times (k+1) + \hat{q}_{k-1}) \times (k+2) + \cdots + \hat{q}_2) \times (2k-1) + \hat{q}_1) \bmod p, \end{cases} \quad (12)$$

Back to 8

for example the multiplication by $45 = (10\bar{1}0\bar{1}01)_2$ three additions if one considers the NAF, or with only two if one considers its factorization $45 = 9 \times 5$.

c	$\#A$	c	$\#A$	c	$\#A$
1	0	16	0	31	1
2	0	17	1	32	0
3	1	18	1	33	1
4	0	19	2	34	1
5	1	20	1	35	2
6	1	21	2	36	1
7	1	22	2	37	2
8	0	23	2	38	2
9	1	24	1	39	2
10	1	25	2	40	1
11	2	26	2	41	2
12	1	27	2	42	2
13	2	28	1	43	3
14	1	29	2	44	2
15	1	30	1	45	2

Table: Number of addition ($\#A$) required in the multiplication by some small constants c

Back to 8

p	form of p	k	l
59	$2^6 - 2^2 - 1$	29	170
67	$2^6 + 3$	29 . . . 31	175 . . . 188
73	$2^6 + 2^3 + 1$	29 . . . 31	179 . . . 191
127	$2^7 - 1$	23 . . . 61	160 . . . 426
257	$2^8 + 1$	23 . . . 73	184 . . . 584
503	$2^9 - 2^3 - 1$	19 . . . 61	170 . . . 547
521	$2^9 + 2^3 + 1$	19 . . . 61	171 . . . 550
8191	$2^{13} - 1$	13 . . . 43	168 . . . 558
65537	$2^{16} + 1$	11 . . . 37	176 . . . 592
131071	$2^{17} - 1$	11 . . . 31	186 . . . 526
524287	$2^{19} - 1$	11 . . . 31	208 . . . 588
2147483647	$2^{31} - 1$	7 . . . 19	216 . . . 588
2305843009213693951	$2^{61} - 1$	3 . . . 7	182 . . . 426

Table: Good candidates for p and k suitable for elliptic curve cryptography and the corresponding key lengths

Back to 8

ANNEXES

Transformée de Fourier

Transformée de Fourier

Le principe

- Les points considérés sont des racines n^{ieme} de l'unité : $\omega^n = 1$, ω racine primitive.
- On calcule $\begin{cases} \widehat{A}^n & = & (A(\omega^0), A(\omega^1), \ldots, A(\omega^{n-1})) \\ \widehat{B}^n & = & (B(\omega^0), B(\omega^1), \ldots, B(\omega^{n-1})) \end{cases}$
- On en déduit $\widehat{P}^n = (P(\omega^0), P(\omega^1), \ldots, P(\omega^{n-1}))$
- On reconstruit $P(X)$

(Retour 16)

Transformée de Fourier

Propriétés

- $A(X) = \sum_{i=0}^{n-1} a_i X^i$
- Nous avons $A(\omega^k) = \sum_{i=0}^{n-1} a_i \omega^{ik}$
- Comme ω^k est une racine n^{ieme} de l'unité, ω^{2k} est une racine $\frac{n}{2}^{\text{ieme}}$. De plus, $(\omega^k)^{n/2} = -1$ (on suppose n pair)
- Ainsi pour
$$A(\omega^k) = \sum_{i=0}^{\frac{n}{2}-1} a_{2i}\omega^{2ik} + \omega^k \sum_{i=0}^{\frac{n}{2}-1} a_{2i+1}\omega^{2ik} = A_0(\omega^{2k}) + \omega^k A_1(\omega^{2k})$$

(Retour 16)

Transformée de Fourier

Algorithme FFT

FFT(A, ω, n)

Si $n = 1$ **alors** *retourne* (a_0) **Sinon**

$$\begin{cases} A_0(X) & \leftarrow & \sum_{i=0}^{\frac{n}{2}-1} a_{2i} X^i \\ A_1(X) & \leftarrow & \sum_{i=0}^{\frac{n}{2}-1} a_{2i+1} X^i \end{cases}$$

$$\begin{cases} \widehat{A_0}^{n/2} & \leftarrow & \text{FFT}(A_0, \omega^2, n/2) \\ \widehat{A_1}^{n/2} & \leftarrow & \text{FFT}(A_1, \omega^2, n/2) \end{cases}$$

Pour $k = 0$ **à** $\frac{n}{2} - 1$ **faire**

$$\begin{cases} A(\omega^k) & \leftarrow & A_0(\omega^{2k}) + \omega^k A_1(\omega^{2k}) \\ A(\omega^{k+n/2}) & \leftarrow & A_0(\omega^{2k}) - \omega^k A_1(\omega^{2k}) \end{cases}$$

Retourner $\widehat{A}^n$

(Retour 16)

Transformée de Fourier

Complexité Algorithme FFT

- Soit $F(n)$ le nombre d'opérations élémentaires pour une FFT de dimension n
- Nous avons la récurrence : $F(n) = 2F(n/2) + \alpha n$
- D'où, $F(n) = O(n \log_2 n)$

(Retour 16)

Transformée Inverse de Fourier

le retour

- Considérons $\widehat{A}^n(\omega^{-t})$
- $$\begin{aligned} \widehat{A}^n(\omega^{-t}) &= \sum_{k'=0}^{n-1} \left(\sum_{k=0}^{n-1} a_k \omega^{kk'} \right) \omega^{-tk'} \\ &= \sum_{k=0}^{n-1} a_k \left(\sum_{k'=0}^{n-1} \omega^{(k-t)k'} \right) \end{aligned}$$
- Or, si ω est une racine n^{ieme} de l'unité alors $1 + \omega + \omega^2 + \cdots + \omega^{n-1} = 0$
  D'où, $\sum_{k'=0}^{n-1} \omega^{(k-t)k'} = 0$ si $t \neq k$ et $= n$ sinon.
- Ainsi $A = \frac{1}{n}\text{FFT}(\widehat{A}^n, \omega^{-1}, n)$

(Retour 16)

Transformée de Fourier

Application à la multiplication

- Nous n'entrerons pas dans les détails. Pour en savoir plus : G. Brassard, S. Monet et D. Zuffellato, Algorithmes pour l'arithmétique des très grands entiers, TSI: Technique et Science Informatiques, Vol. 5, no. 2, pp. 89 - 102, mars 1986
- L'idée est de travaillé sur l'anneau $\mathbb{Z}/(2^n+1)\mathbb{Z}$ où 2 est racine $2n$ de l'unité.

(Retour 16)

REFERENCES


E.R. Berlekamp.
Bit-serial reed-solomon encoders.
IEEE Trans. Information Theory, 1982.


Daniel V. Bailey and Christof Paar.
Public-key cryptography with arbitrary finite fields.
Submission to IEEE P1363a, February 2000.


Joachim Von Zur Gathen and Mark Giesbrecht.
Constructing normal bases in finite fields.
Journal of Symbolic Computation, 10(6):547–570, December 1990.


Shuhong Gao and Hendrik W. Lenstra, Jr.
Optimal normal bases.
Designs, Codes, and Cryptography, 2(4):315–323, 1992.


Geiselmann and Lukhaub.
Redundant representation of finite fields.
In *PKC: International Workshop on Practice and Theory in Public Key Cryptography*. LNCS, 2001.


Dieter Gollmann.
Equally spaced polynomials, dual bases,and multiplication in $\mathbb{F}_{2^n}$.
IEEE Transactions on Computers, 2002.


Shuhong Gao and Scott A. Vanstone.
On orders of optimal normal basis generators.
Mathematics of Computation, 64(211):1227–1233, July 1995.


A. Halbutogullari and Cetin Koya Koc.
Mastrovito multiplier for general irreducible polynomials.
IEEE Transactions on Computers, 49(5):503–518, 2000.


Hasan, Wang, and Bhargava.
A modified massey-omura parallel multiplier for a class of finite fields.
IEEETC: IEEE Transactions on Computers, 42, 1993.


M. Anwarul Hasan Huapeng Wu and Ian F. Blake.
New low-complexity bit-parallel finite field multipliers using weakly dual bases.
IEEE Transactions on Computers, 1998.


Koc and Acar.
Montgomery multiplication in $GF(2^k)$.
IJDCC: Designs, Codes and Cryptography, 14, 1998.


Cetin Kaya Koc, Golga Acar, and Burton S. Kaliski, Jr.
Analyzing and comparing montgomery multiplication algorithms.
IEEE Micro, June 1996.
26–33.


Neal Koblitz.
Elliptic curve cryptosystems.
Mathematics of Computation, 48(177):203–209, January 1987.


Koc and Sunar.
Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields.
IEEETC: IEEE Transactions on Computers, 47, 1998.


R. Lidl and H. Niederreiter.
Finite Fields.
Addison-Wesley, Reading, 1985.


Rudolf Lidl and Harald Niederreiter.
Introduction to Finite Fields and Their Applications.
Cambridge University Press, revised edition edition, 1994.


E. D. Mastrovito.
VLSI designs for multiplication over finite fields GF(2).
In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 6th International Conference, (AAECC-6)*, pages 297–309, Berlin - Heidelberg - New York, July 1989. Springer.


E. Mastrovito.
VLSI Architectures for Computation in Galois Fields.
PhD thesis, Linkoping University, Dept. Electr. Eng., 1991.


Peter L. Montgomery.
Modular multiplication without trial division.
Mathematics of Computation, 44(170):519–521, April 1985.


D. Page and N. P. Smart.
Hardware implementation of finite fields of characteristic three.

In B. S. Kaliski Jr., ?. K. Ko?, and C. Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 529–539. Springer-Verlag, February 2003.


S.A. Vanstone R.C. Mullin, I.M. Onyszchuk and R. Wilson.
Optimal normal basis in $gf(p^m)$.
Discrete Applied Mathematics, 1989.


Silverman.
Fast multiplication in finite fields $GF(2^N)$.
In *CHES: International Workshop on Cryptographic Hardware and Embedded Systems, CHES, LNCS*, 1999.


B. Sunar and C. K. Koc.
Mastrovito multiplier for all trinomials.
IEEE Trans. Comput., 1999.


B. Sunar and C. K. Koc.
An eficient optimal normal basis type ii multiplier.
IEEE Transactions on Computers, 2001.


N. P. Smart.
Elliptic curves over small fields of odd characteristic.
J. Cryptology, 12(2):141–151, August 1999.

UPMC SORBONNE UNIVERSITÉS


N. P. Smart.
A comparison of different finite fields for elliptic curve cryptosystems.
Computers and Mathematics with Applications, 42(1-2):91–100, 2001.


Mohammed Benaissa Sebastian T.J. Fenn and David Taylor.
$gf(2^m)$ multiplication and division over the dual basis.
IEEE Transactions on Computers, 1996.


Erkay Savas, Alexandre F. Tenca, and Çetin K. Koç.
A scalable and unified multiplier architecture for finite fields GF (p) and GF (2 m).
Lecture Notes in Computer Science, 1965:277–??, 2001.


Takagi, Yoshiki, and Takagi.
A fast algorithm for multiplicative inversion in $GF(2^m)$ using normal basis.
IEEETC: IEEE Transactions on Computers, 50, 2001.


Wu.
Bit-parallel finite field multiplier and squarer using polynomial basis.
IEEETC: IEEE Transactions on Computers, 51, 2002.

 Bajard, J.C., Didier, L.S., Kornerup, P.: Modular multiplication and base extension in residue number systems. *15th IEEE Symposium on Computer Arithmetic, 2001 Vail Colorado USA pp. 59–65*

 Bajard, J.C., Duquesne, S., Ercegovac M. and Meloni N.: Residue systems efficiency for modular products summation: Application to Elliptic Curves Cryptography, *in Advanced Signal Processing Algorithms, Architectures, and Implementations XVI, SPIE 2006, San Diego, USA.*

 Bajard, J.C. and ElMrabet N.: Pairing in cryptography: an arithmetic point of view, *Advanced Signal Processing Algorithms, Architectures, and Implementations XVII, part of the SPIE Optics & Photonics 2007 Symposium. August 2007 San Diego, USA.*

 J.C. Bajard, L. Imbert, and G. A. Jullien: Parallel Montgomery Multiplication in $GF(2^k)$ using Trinomial Residue Arithmetic, *17th IEEE symposium on Computer Arithmetic, 2005, Cape Cod, MA, USA.pp. 164-171*

 J.C. Bajard, L. Imbert et Ch. Negre, Arithmetic Operations in Finite Fields of Medium Prime Characteristic Using the Lagrange Representation, *journal IEEE Transactions on Computers, September 2006 (Vol. 55, No. 9) p p. 1167-1177*

 Bajard, J.C., Meloni, N., Plantard, T.: Efficient RNS bases for Cryptography *IMACS'05, Applied Mathematics and Simulation, (2005)*

 Garner, H.L.: The residue number system. *IRE Transactions on Electronic Computers, EL* **8:6** *(1959) 140–147*

 Knuth, D.: Seminumerical Algorithms. The Art of Computer Programming, vol. 2. *Addison-Wesley (1981)*

 Montgomery, P.L.: Modular multiplication without trial division. *Math. Comp.* **44:170** *(1985) 519–521*

 Svoboda, A. and Valach, M.: Operational Circuits. *Stroje na Zpracovani Informaci, Sbornik III, Nakl. CSAV, Prague, 1955, pp.247-295.*

 Szabo, N.S., Tanaka, R.I.: Residue Arithmetic and its Applications to Computer Technology. *McGraw-Hill (1967)*