Useful Arithmetic for Cryptography

Jean Claude Bajard

LIP6 - UPMC (Paris 6)/ CNRS
France

8 Jul. 2013

Introduction

Modern Public Key Cryptography

- In 1985, Victor S. Miller [1] and Neal Koblitz [2] introduced Elliptic Curve Cryptography.
- Gödel Prize 2013: Dan Boneh, Matthew K. Franklin [3] and Antoine Joux [4] for Pairing Cryptography.
- Group operations on points of elliptic curve defined on finite fields.
- Basic finite field operations: addition, multiplication, inversion...

Content

Finite Fields Representations

General Principles [5]

A finite field $F(+, \times)$ is a finite set F such that:

- $F(+)$ is an Abelian Group
- $F(+, \times)$ is a Ring where every element (excepted 0 for $\times$) has an inverse

Elementary Finite Fields have an order equal to a prime p.
Example of a such finite prime field $\mathbb{Z}/p\mathbb{Z}$

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

Calculus are based on modular arithmetic.

Splitting Finite Field

More generally, Finite Field has an order equal to a power of a prime, we note $GF(p^m)$ or $\mathbb{F}_{p^m}$ with p prime.
p is the caracteristic, if $u \in GF(p^m)$ then $p \times u = 0$.

- as a set of polynomial residues modulo an irreducible polynomial $P(X)$ of degree m in $\mathbb{F}_p[x]$
- as a set of the powers of a primitive element g, $GF(p^m) = \{0, g^0, g^1, ..., g^{p^m-2}\}$
- as a set of linear combinations of base elements :

canonical $\{1, \alpha, \alpha^2, ..., \alpha^{m-1}\}$ ou normal $\{\alpha, \alpha^p, \alpha^{p^2} ..., \alpha^{p^{m-1}}\}$ (α root of $P(X)$)

Example in $GF(2^2)$ (notice $GF(2^2) \neq \mathbb{Z}/2^2\mathbb{Z}$)

- Polynomials in $GF(2)[X]$: $0, 1, X, 1+X$.
- Addition on $GF(2)$: $1 + (1+X) = X$.
- Product with a modular reduction in function of an irreducible one.
  - $X^2 + X + 1$ is irreducible over $GF(2)$, $GF(4)$ can be represented by $GF(2)[X]/X^2 + X + 1$.
  - Multiplication modulo $X^2 + X + 1$:
    $X * (1+X) = (X + X^2) \bmod (X^2 + X + 1) = 1$
  - The choice of the irreducible polynomial impacts the complexity.

Multiplication in $GF(p)$

Multiplication of two values

Multiplication of two values

Product of two numbers

via polynomials

- Let $A = \sum_{i=0}^{k-1} a_i \beta^i$ and $B = \sum_{i=0}^{k-1} b_i \beta^i$ be two numbers in base β
- Let $A(X) = \sum_{i=0}^{k-1} a_i X^i$ and $B(X) = \sum_{i=0}^{k-1} b_i X^i$ be the associated polynomials
- Evaluation of the product $P = A \times B$:
  1. Polynomial Evaluation: $P(X) = A(X) \times B(X)$
  2. Calculus of the value: $P(\beta) = A(\beta) \times B(\beta)$

Product of two numbers

via polynomials: Remarks

- Step 1, the p_i are lower than $k \times \beta^2$
- Step 2, the calculus of $P(\beta)$ becomes a reduction of the p_i by carry propagation.

Polynomial representations

- A polynomial of degree $k-1$ can be defined:
  - by its k coefficients a_i

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

  - or by k values in different points e_i

$$\text{for } \; i = 0..k-1, \quad A(e_i) = \sum_{j=0}^{k-1} a_j e_i^j$$

e_i are chosen, in respect to two criteria: easy evaluation and small size for the $A(e_i)$.

Polynomial Product

defined by coefficients

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

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

k^2 products.

Polynomial Product

defined by points

- $P(X) = A(X) \times B(X) = (\sum_{i=0}^{k-1} a_i X^i) \times (\sum_{i=0}^{k-1} b_i X^i) = \sum_{i=0}^{2k-2} p_i X^i$
  is computed at $2k-1$ differents points:

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

$2k-1$ products.

Coefficient reconstruction

Lagrange approach

- Use of a sum of k polynomials, such that the $i-$th one is equal to $P(e_i)$ for e_i, and 0 for all other e_j with $j \neq i$.

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

Coefficient reconstruction

Newton approach

- The main idea is to use polynomials of increasing degrees

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

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

with, $p'_i = P(e_i)$

Product of two numbers

Karatsuba Algorithm(1)

- Select points $e_0 = 0$, $e_1 = -1$ and $e_2 = \infty$
- We have:
$$A = \sum_{i=0}^{k-1} a_i\beta^i = (\sum_{i=0}^{k/2-1} a_{k/2+i}\beta^i)\beta^{k/2} + \sum_{i=0}^{k./2-1} a_i\beta^i = A_1\beta^{k/2} + A_0$$
- Polynomial view: $A(X) = A_1X + A_0$

$$\begin{cases} A(0) = A_0 \\ A(-1) = A_0 - A_1 \\ A(\infty) = \lim_{X\to\infty} A_1X \end{cases}$$

Product of two numbers

Karatsuba Algorithm (2)

- Values of the product polynomials

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

- Newton interpolation

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

Product of two numbers

Karatsuba Algorithm(3)

- Reconstruction

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

- Final evaluation

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

Product of two numbers

Karatsuba Algorithm (4) : Complexity

- Let denote $K(k)$ as the number of elementary operations
- By recurrence $K(k) = 3K(k/2) + \alpha k$, we suppose that the addition is linear
- We obtain $K(k) = O(k^{\log_2(3)})$

Product of two numbers

Toom Cook Algorithm (1)

The Karatsuba approach can be generalized:

- Select points $e_0 = 0$, $e_1 = -1$, $e_2 = 1$, $e_3 = 2$ and $e_4 = \infty$
- We have:
  $A = A_2\beta^{2k/3} + A_1\beta^{k/3} + A_0$
- Polynomial view: $A(X) = A_2X^2 + A_1X + A_0$

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

Product of two numbers

Toom Cook Algorithm (2)

- With Newton

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

- We notice a division by 3 $\rightarrow$ limits of this approach
- Reconstruction by computing $P(\beta^{k/3})$:
  $P(X) = \hat{p}_0 + X(\hat{p}_1 + (X+1)(\hat{p}_2 + (X-1)(\hat{p}_3 + \hat{p}_4(X-2))))$

Product of two numbers

Toom Cook Algorithm (3)

- Let denote $T_3(k)$ as the number of elementary operations
- By recurrence $T_3(k) = 5T_3(k/3) + \alpha k$, assuming that addition is linear
- We obtain $T_3(k) = O(k^{\log_3(5)})$

Product of two numbers

Toom Cook Algorithm (4), asymptotic point of view

- Splitting by n
- With $T_n(k)$ he number of elementary operations
- By recurrence $T_n(k) = (2n-1)T_n(k/n) + \alpha k$, assuming that addition is linear
- We obtain $T_n(k) = O(k^{\log_n(2n-1)})$
- Then the complexity of the multiplication can reach $O(k^{1+\epsilon})$

Fourier Transform

Complexité Algorithme FFT

- Select points: the n^{th}roots of unity, $\omega^n = 1$, ω primitive.
- Properties: ω^{2k} is a $\frac{n}{2}$th root, $(\omega^k)^{n/2} = -1$ (assuming n even)

$$A(\omega^k) = \sum_{i=0}^{\frac{n}{2}-1} a_{2i}\omega^{2ik} + \omega^k \sum_{i=0}^{\frac{n}{2}-1} a_{2i+1}\omega^{2ik} = A_0(\omega^{2k}) + \omega^k A_1(\omega^{2k})$$

- $F(n)$ number of elementary op. for a FFT of dimension n
- We have $F(n) = 2F(n/2) + \alpha n$, then, $F(n) = O(n \log_2 n)$

Multiplication in $GF(p)$

Modular Reduction

Modular Reduction

p fixed

Two options:

- Specific p allowing an easy reduction

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

- Common $p \rightarrow$ generic algorithms

Modular Reduction

$p = \beta^n - \xi$ with $0 \leq \xi < \beta^{n/2}$ and $\xi^2 \leq \beta^n - 2\beta^{n/2} + 1$

We have $C = A \times B \leq (p-1)^2$

- We write $C = C_1\beta^n + C_0$
- First reduction pass: $C \equiv C_1\xi + C_0 \ (= C') \pmod{p}$
- Second reduction pass: $C' \equiv C'_1\xi + C'_0 \ (= C'') \pmod{p}$
- Final touch:
  If $C'' + \xi \geq \beta^n$ Then $R = C'' + \xi - \beta^n$, Else $R = C''$

Modular Reduction

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

- This reduction uses two multiplications by ξ, two options
  - Choose a very small ξ, for example, $\xi < \beta \rightarrow$ digit$\times$number
  - Choose a very sparce $\xi \rightarrow$ shift and add approach
- If $\xi > \beta^{n/2}$, then the number of passes increases

Modular Reduction with $p = \beta^n - 1$

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

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

Modular Reduction with $p = \beta^n - 1$

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

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

Modular Reduction with $p = \beta^n - \beta^t - 1$

If $t < n/2$ then M is obtained with one matrix addition.

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

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

Multiplication in $GF(p)$

Generic Modular Reduction

Generic Modular Reduction

Barrett Algorithm [7]

Reduction of A modulo P via the approximation of the quotient.

- Conditions: $\beta^{n-1} \leq P < \beta^n$ et $A < P^2 < \beta^{2n}$
- We can write that: $\beta^{u+v} A - P \times \frac{\beta^{n+u}}{P} \times \frac{A}{\beta^{n-v}} = 0$
- $\beta^{u+v} A - P \times \lfloor \frac{\beta^{n+u}}{P} \rfloor \times \lfloor \frac{A}{\beta^{n-v}} \rfloor =$
  $P\left(\lfloor \frac{\beta^{n+u}}{P} \rfloor f(\frac{A}{\beta^{n-v}}) + \lfloor \frac{A}{\beta^{n-v}} \rfloor f(\frac{\beta^{n+u}}{P}) + f(\frac{A}{\beta^{n-1}}) f(\frac{\beta^{2n}}{P}) \right) < P(\beta^{u+1} + (\beta^{n+v} - 1) + 1)$
  with $f(.)$ the fractional part function

  If $u \geq n+1$ and $v \geq 2$ then $(\beta^{u+1} + \beta^{n+v})/\beta^{u+v} < 1$
- We deduce: $A \bmod P \equiv A - P \times \left\lfloor \frac{\lfloor \frac{\beta^{2n+1}}{P} \rfloor \times \lfloor \frac{A}{\beta^{n-2}} \rfloor}{\beta^{n+3}} \right\rfloor < 2P$

Generic Modular Reduction

Barrett Algorithm [7]

Barrett(A ,P)

Inputs $\beta^{n-1} \leq P < \beta^n$ and $A < P^2 < \beta^{2n}$

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

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

$R \leftarrow A - Q \times P$

If $R \geq P$, Then $R \leftarrow R - P$ and $Q \leftarrow Q + 1$

Complexity: 2 products of $n + 1$ digits

Generic Modular Reduction

Montgomery Algorithm [8]

Reduction of A modulo P via a multiple of P.

- Conditions : $\beta^{n-1} \leq P < \beta^n$ and $A < P\beta^n$
- The scheme is to add a multiple of P to A such that the result is a multiple of β^n
- The division by β^n in base β is a shift.
- The output of this approach is $A \times \beta^{-n} \bmod P$

Generic Modular Reduction

Montgomery Algorithm [8]

Montgomery(A, P)

Inputs $\beta^{n-1} \leq P < \beta^n$ and $A < P\beta^n < \beta^{2n}$

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

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

$R \leftarrow (A + Q \times P)$ *R is a multiple of β^n*

$R \leftarrow R \div \beta^n$ *division by β^n is a shift, ($R < 2P$)*

If $R \geq P$ Then $R \leftarrow R - P$ *(optional)*

Complexity: 2 products of n digits (in fact close to two half products)

Generic Modular Reduction

Montgomery Representation

- To avoid the accumulation of factors β^{-n} mod P, we note: $\widetilde{A} = A \times \beta^n \bmod P$
- Thee construction $\widetilde{A} = \text{Montgomery}(A \times |\beta^{2n}|_P, P)$
- Stable for addition and multiplication using Montgomery reduction: $\widetilde{A} + \widetilde{B} = \widetilde{A + B}$ and $\widetilde{AB} = \text{Montgomery}(\widetilde{A} \times \widetilde{B}, P)$
- Reconversion to standard: $A = \text{Montgomery}(\widetilde{A}, P)$
- **It is the most used algorithm in cryptography**

Interleaved Modular Multiplication

Montgomery Algorithm

MontgomeryI(A, B, P)

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

Output $R = A \times B \times \beta^{-n} \bmod P$

Core $R \leftarrow 0$

For $i = 0$ **to** $i = n - 1$ **do**

$R \leftarrow (R + b_i \times A)$
$q_i \leftarrow r_0 \times |-p_0^{-1}|_\beta \bmod \beta$
$R \leftarrow (R + q_i \times P)$ *multiple of* β
$R \leftarrow R \div \beta$ *at the end* $(R < 2P)$

If $R \geq P$, Then $R \leftarrow R - P$ *(optional)*

Binary Interleaved Modular Multiplication

Montgomery Algorithm

MontgomeryB(A , B, P)

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

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

Core $R \leftarrow 0$

For $i = 0$ **to** $i = n - 1$ **do**

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

$q_i \leftarrow r_0$ *In fact* $|-p_0^{-1}|_2 = 1$ *if* P *odd*

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

$R \leftarrow R >> 1$ *at the end* $(R < 2P)$

If $R \geq P$, Then $R \leftarrow R - P$ *(optional)*

Bipartite Modular Multiplication [9]

- This approach is based on:

  We define $*$ as: $X * Y = (X \times Y) \times R^{-1} \bmod P$

  We split *y*: $Y = Y_h \times R + Y_l$ *for example* $R = \beta^{n/2}$

  thus $X * Y = (X \times Y_h \bmod P + X \times Y_l \times R^{-1} \bmod P) \bmod P$
- $X \times Y_h \bmod P$ is computed using Barret.
- $X \times Y_l \times R^{-1} \bmod P$ is computed via Montgomery.
- These two operations can be done in parallelel

Multiplication in $GF(2^m)$

Multiplication in $GF(2^m)$

Most of the hardware implementations use $GF(2^m)$ where basic operators are AND and XOR.
The different approaches for the modular reduction needed in the multiplication over $GF(2^m)$ are:

- The ones depending of the finite field
- The generic ones
- Those using specific bases

Multiplication in $GF(2^m)$

Polynomial Approaches

Multiplication in $GF(2^m)$

The calculus of $C(X) = A(X) \times B(X) \bmod P(X)$ can be executed in two steps:

1. a polynomial product $C'(X) = A(X) \times B(X)$,

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

2. a modular reduction $P(X)$: $C(X) = C'(X) \bmod P(X)$

Montgomery Algorithm

- $A(X) * B(X)$ is computed in $GF(2^m)$ defined by $P(X)$ a degree m irreducible polynomial
- Montgomery compute $A(X) * B(X) * R^{-1}(X) mod P(X)$ where $R(X)$ is a fixed element and $R^{-1}(X)$ is its inverse mod $P(X)$.
  We know $R(X)$ and $P(X)$ (irreducible), we can precompute $R^{-1}(X)$ and $P'(X)$ such that:

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

Montgomery Algorithm (generic case)

Inputs: $A(X)$ *and* $B(X)$ *of degrees lower than m*
Outputs: $T(X) = A(X) * B(X) * R^{-1}(X) \bmod P(X)$
Precomputed: $P'(X)$, $R(X)$

Product: $C(X) = A(X) * B(X)$
Reduction: $Q(X) = -C(X) * P'(X) \bmod R(X)$
$T(X) = \big(C(X) + Q(X) * P(X)\big)\, div\ R(X)$

- The complexity is due to the three products.
- The reduction modulo $R(X)$ and the division by $R(X)$ are easy if $R(X) = X^m$.

Montgomery Algorithm (execution)

- Polynomial representations:

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

- We decompose the evaluation using matrices, into two parts:
  - The first lines for the computation of $Q(X)$
  - The last lines for the result $T(X)$

Montgomery Algorithm (execution)

Decomposition of the calculus for $Q(X)$: *(the lower degrees)*

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

Then for $T(X)$: *(the upper degrees)*

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

Montgomery Algorithm (complexity of the general case)

- Complexity counting the number of elementary operations over $GF(2)$:
  - $m^2 + (m-1)^2$ multiplications (AND)
  - $(m-1)^2 + (m-2)^2 + m$ additions (XOR).
- For this approach we can use the Montgomery representation: $\widetilde{A}(X) = A(X) \times R(X) \pmod{P}(X)$
- It can be generalized to $GF(p^k)$

Iterative Montgomery in $GF(2^m)$ with $R(X) = X^m$

Inputs: $A(X)$ and $B(X)$ *of degrees lower than m*
Output: $T(X) = A(X) * B(X) * R^{-1}(X) \bmod P(X)$
Precomputed: $P'(X)$, $R(X)$

Initialisation $T(X) = 0$
Loop For $i = 0$ to $m - 1$ do
$$T(X) = T(X) + a_i * B(X)$$
$$T(X) = (T(X) + t_0 * P(X))/X$$

Iterative Montgomery in $GF(2^m)$ with $R(X) = X^m$

- At each step a division by X, hence at the end it is equivalent to $R(X) = X^m$.
- Moreover $P(X)$ is irreducible, thus its constant term is 1, idem for $P'(X)$.
- The complexity given in logical gates:
  - $2m^2$ *XOR* (for the additions)
  - and $2m^2$ *AND* (for the products)

Method of Mastrovito [10]

Approach Idea

- $GF(2^m)$ is defined by a root α of the irreducible $P(X)$ of degree m.
- The elements of $GF(2^m)$ are given in the canonical $\{1, \alpha, \alpha^2, ..., \alpha^{m-1}\}$:

$$A = \sum_{i=0}^{m-1} a_i \alpha^i \quad \text{and} \quad B = \sum_{i=0}^{m-1} b_i \alpha^i.$$

- We note $C = A \times B$ in $GF(2^m)$, $C = \sum_{i=0}^{m-1} c_i \alpha^i$.

Mastrovito proposed to construct Z, a matrix $m \times m$ using the coefficients of A, such that:

$$C = Z \times B$$

Method of Mastrovito

Construction of Z

Z is obtained by:

1. constructing the matrix $(m-1) \times m$, Q which is the representations of X^k for $k \geq m$ 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. and then, the matrix Z is obtained with:

$$z_{i,j} = \begin{cases} a_i \text{ for } j = 0, \; i = 0 \dots m-1 \\ u(i-j) * a_{i-j} + \sum_{t=0}^{j-1} q_{j-1-t,i} * a_{m-1-t}, \text{ else }, \text{ with } u(t) = \begin{cases} 1 \text{ if } t \geq 0 \\ 0 \text{ else} \end{cases} \end{cases}$$

Method of Mastrovito

Cost of the approach

- The complexity is due to the construction of Z which can need $m^3/2$ *And* and *Xor*, the choice of the irreducible polynomial is fundamental.
- With trinomials like $X^m + X + 1$ the multiplication is done with $m^2 - 1$ XOR and m^2 AND.
- There are some variants
  - if all the coefficients are 1 (all-one polynomial)

$P(X) = 1 + X + X^2 + ... + X^m$, *in this case* $X^{m+1} \equiv 1 \pmod{P(X)}$

  - or for regular sparced polynomials

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

Method of Mastrovito I

Example with a trinomial

We consider $GF(2^7)$ with the canoical base $\{1, \alpha, \alpha^2, ..., \alpha^6\}$ where α is a root of the irreducible $P(X) = X^7 + X + 1$. Thus,

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

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

Method of Mastrovito II

Example with a trinomial

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

Méthode de Mastrovito

Exemple avec un All-One

If $P(X) = 1 + X + X^2 + ... + X^m$, the matrix Z can be written as $Z = Z_1 + Z_2$ with:

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

and

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

Toeplitz Matrices

Definition

A $n \times n$ matrix is Toeplitz if $[t_{i,j}]_{1 \leq i,j \leq n}$ are such that $t_{i,j} = t_{i-1,j-1}$ for $i, j \geq 1$.

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

Remark: An addition of 2 Toeplitz requires only $2n-1$ additions.

Toeplitz Matrices

Definition

A $n \times n$ matrix is Toeplitz if $[t_{i,j}]_{1 \leq i,j \leq n}$ are such that $t_{i,j} = t_{i-1,j-1}$ for $i, j \geq 1$.

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

Remark: An addition of 2 Toeplitz requires only $2n - 1$ additions.

Toeplitz Matrices

Definition

A $n \times n$ matrix is Toeplitz if $[t_{i,j}]_{1 \leq i,j \leq n}$ are such that $t_{i,j} = t_{i-1,j-1}$ for $i, j \geq 1$.

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

Remark: An addition of 2 Toeplitz requires only $2n - 1$ additions.

Product matrix-vector with a Toeplitz [11]

If T is Toeplitz $n \times n$ with $2|n$ then:

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

is such that:

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

with

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

Complexity of the Toeplitz - vector product

Fan and Hasan proposed also a 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 is the delay of one AND and D_X the one for one XOR.

Application of Toeplitz - vector approach

- We have seen that $C(X) = A(X) \times B(X) \bmod P(X)$ can be obtained with $C(X) = Z \times B(X)$, where Z is a $m \times m$ matrix
- Using circular permutations of rows or columns, Z can be transformed into a Toeplitz.
- Fan-Hasan did it with trinomials, pentanomials (2006) and All-One (2007), then Hasan-Nègre (2010) used quadrinomals (with $Q(X) = (X+1)P(X)$)

Application of Toeplitz - vector approach

Example

We consider $GF(2^6)$ with $P(X) = X^6 + X + 1$

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

is transformed in Toeplitz with a rotation of the 1st row to the last one

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

Multiplication in $GF(2^n)$

Approaches using specific bases

Normal Base for $GF(2^m)$

- We call normal base of $GF(2^m)$, the base $\{\alpha, \alpha^2, \alpha^{2^2} \ldots, \alpha^{2^{m-1}}\}$ where α is a root of $P(X)$ (irreducible of degree m) *(α^{2^i} are roots of $P(X)$, Frobenius property, $P(X)^{2^i} = P(X^{2^i})$)*
- A in $GF(2^m)$: $A = (a_0, a_1, \ldots, a_{m-1}) = \sum_{i=0}^{m-1} a_i \alpha^{2^i}$.
- The square operation is a left rotation:

  we have $A^2 = \sum_{i=0}^{m-1} a_i \alpha^{2^{i+1}}$ *but* $\alpha^{2^m} = \alpha$,

  thus, $A^2 = a_{m-1}\alpha + \sum_{i=1}^{m-1} a_{i-1} \alpha^{2^i}$ *in other words* $A^2 = (a_{m-1}, a_0, \ldots, a_{m-2})$.

Normal Base: Multiplication of Massey-Omura [13]

- We have $D = A \times B = A \times M \times B^t$ with:

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

- $M = M_0\,\alpha + M_1\,\alpha^2 + \dots + M_{m-1}\,\alpha^{2^{m-1}}$ where M_i are composed of 0 and 1.
- Thus, $D = A \times B$ is obtained coordinate by coordinate with $d_{m-1-k} = A \times M_{m-1-k} \times B^t$ for $k = 0, \dots, m-1$.

Normal Base: Multiplication of Massey-Omura [13]

Storage of one matrix

- We have $D^{2^k} = A^{2^k} \times B^{2^k}$ and the power to 2^k is given by k left rotations:
  $d_{m-1-k} = A^{2^k} \times M_{m-1} \times (B^{2^k})^t$ for $k = 0, ..., m-1$
- The complexity is given by the number of $1's$ in M_{m-1} which depends on m and on $P(X)$.
- The lower bound is $2m-1$. When this bound is reached, the base is said "optimal"[12]
- If all the coefficients of $P(X)$ are 1 (All-One), it is reached and the complexity is m^2 AND and $2m^2 - 2m$ XOR.

Normal Base: Multiplication of Massey-Omura [13]

Example

We consider $GF(2^4)$ and the normal base $(\alpha^{2^0}, \alpha^{2^1}, \alpha^{2^2}, \alpha^{2^3})$ where α is a root of $P(X) = X^4 + X^3 + 1$ (irreducible)

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

Thus,

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

Normal Base: Modified Massey-Omura [14]

- If $P(X)$ is All-One, the complexity can be decreased to m^2 AND and m^2-1 XOR, by decomposing M_{m-1}
- $M_{m-1} = (P+Q) \pmod 2$

  with $P_{i,j} = \begin{cases} 1 & \text{if } i = (m/2+j) \bmod m \\ 0 & \text{else} \end{cases}$
- Let $T^{(k)}$ be such that: $B^{2^k} = BT^{(k)}$,

  we have $T^{(k)}PT^{(k)t} = P$,

  and

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

for $k = 0, \ldots, m-1$

Normal Base: Modified Massey-Omura [14]

Example

We consider $GF(2^4)$ and the normal base $(\alpha^{2^0}, \alpha^{2^1}, \alpha^{2^2}, \alpha^{2^3})$ where α is a root of $P(X) = X^4 + X^3 + X^2 + X + 1$ (irreducible). With $\gamma = \alpha + \alpha^2 + \alpha^4 + \alpha^8$, we obtain:

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

Thus:

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

Dual Bases in $GF(2^m)$

Definition

- **Trace Function**: linear form $Tr(u) = \sum_{i=0}^{m-1} u^{2^i} \in GF(2)$ with $u \in GF(2^m)$ *(minimal polynomial of α, $P(X) = \prod_{i=0}^{m-1}(X - \alpha^{2^i}) \in GF(2)[X]$)*
- **Dual Bases**: two bases $\{\lambda_i, i = 0..m-1\}$ and $\{\nu_j, j = 0..m-1\}$ are dual if $Tr(\lambda_i.\nu_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
- **Base conversion** :

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

Dual Bases in $GF(2^m)$

General Definition

- **An other linear form**: $f(u) = Tr(\beta.u)$ where $\beta \in GF(2^k)$
- **Dual bases** if $Tr(\beta.\lambda_i.\nu_j) = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$
- **Base conversion**:

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

Multiplication avec les Bases duales dans $GF(2^m)$ [15]

- We consider the canonical base $\{\alpha^i, i = 0..m-1\}$ and a dual base with (f, β)
- Be a, b et c in $GF(2^m)$: $c = a \times b$

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

  - first line, we find the coordinates of b in the dual base,
  - coordinates of a are in the canonical one,
  - c is obtained in the dual base.
- **Goal**: find f such that the dual base is a permutation of the canonical one [16]

Dual Bases in $GF(2^m)$: example 1

In $GF(2^4)$, we consider the canonical base $(1, \alpha, \alpha^2, \alpha^3)$ where α is a root of $P(X) = X^4 + X^3 + 1$ (irreducible)
Consider the base,

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

which satisfies $Tr(\alpha^{10}) = Tr(\alpha^{11}) = Tr(\alpha^{13}) = Tr(\alpha^{14}) = Tr(1) = 0$, et $Tr(\alpha^{12}) = Tr(\alpha) = 1$.
Thus bases $(1, \alpha, \alpha^2, \alpha^3)$ and $(\alpha^{12}, \alpha^{11}, \alpha^{10}, \alpha^{13})$ are dual.
Let $A = \alpha^{12} = (1, 1, 0, 0)$ and $B = \alpha^7 = (0, 1, 1, 1)$ in the canonical base, and $A = \alpha^{12} = (1, 0, 0, 0)$ and $B = \alpha^7 = (0, 1, 1, 0)$ in the dual one. We have,

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

We verify that $C = \alpha^4 = (1, 0, 1, 0)$ in the dual base and $C = (1, 0, 0, 1)$ in the canonical one.

Dual Bases in $GF(2^m)$: example 2

We consider $GF(2^4)$ and the canonical base $(1, \alpha, \alpha^2, \alpha^3)$ with α root of $P(X) = X^4 + X^3 + 1$.
We consider the linear form $Tr(\alpha^{10} u)$. In this case, the dual base is a permutation of the canonical one. $(\alpha^2, \alpha, 1, \alpha^3)$.
Base conversion is trivial and the product of $A = \alpha^{12}$ and $B = \alpha^7$ becomes:

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

We verify that $C = \alpha^4$.

Inversion in a Finite Field

Extended Euclid Algorithm

- Evaluation of the inverse of a modulo b using Bezout identity $b.u_1 + a.u_2 = \gcd(a, b)$.
- We consider $U = (u_1, u_2, u_3)$ and $V = (v_1, v_2, v_3)$ such that:

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

- Initialization $(u_1, u_2, u_3) = (1, 0, b)$ and $(v_1, v_2, v_3) = (0, 1, a)$
- We apply the Euclid GCD algorithm on u_3 and v_3 keeping the previous identities

In fact terms of index 2 are not useful for the computing of the inverse

Extended Euclide Algorithm in $GF(p)$

Initialization $u_1 \leftarrow 1 \quad u_2 \leftarrow 0 \quad u_3 \leftarrow p$

$v_1 \leftarrow 0 \quad v_2 \leftarrow 1 \quad v_3 \leftarrow a$

Loop while $v_3 \neq 0$

$q = \lfloor u_3 / v_3 \rfloor$

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

$u_1 \leftarrow v_1 \quad u_2 \leftarrow v_2 \quad u_3 \leftarrow v_3$

$v_1 \leftarrow t_1 \quad v_2 \leftarrow t_2 \quad v_3 \leftarrow t_3$

Result $u_2 \equiv a^{-1} \bmod p$

Extended Euclide Algorithm in $GF(2^m)$

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

Loop while $V_3 \neq 0$

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

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

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

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

In $GF(2^m)$, this algorithm is in $O(k)$ (at each step the degree decreases)

Extended Euclide Algorithm in $GF(2^4)$

We consider $A(X) = X^2 + 1$ *and* $P(X) = X^4 + X^3 + 1$ *irreducible.*

$$\begin{array}{lll} u_1(X) = 1 & u_2(X) = 0 & u_3(X) = P(X) = X^4 + X^3 + 1 \\ v_1(X) = 0 & v_2(X) = 1 & v_3(X) = A(X) = X^2 + 1 \end{array}$$

$$\begin{array}{llll} n = 2 & u_1(X) = 1 & u_2(X) = X^2 & u_3(X) = X^3 + X^2 + 1 \\ & v_1(X) = 0 & v_2(X) = 1 & v_3(X) = X^2 + 1 \end{array}$$

$$\begin{array}{llll} n = 1 & u_1(X) = 1 & u_2(X) = X^2 + X & u_3(X) = X^2 + X + 1 \\ & v_1(X) = 0 & v_2(X) = 1 & v_3(X) = X^2 + 1 \end{array}$$

$$\begin{array}{llll} n = 0 & u_1(X) = 0 & u_2(X) = 1 & u_3(X) = X^2 + 1 \\ & v_1(X) = 1 & v_2(X) = X^2 + X + 1 & v_3(X) = X \end{array}$$

$$\begin{array}{llll} n = 1 & u_1(X) = 1 & u_2(X) = X^2 + X + 1 & u_3(X) = x \\ & v_1(X) = X & v_2(X) = X^2 + X^3 + X + 1 & v_3(X) = 1 \end{array}$$

$$\begin{array}{llll} n = 1 & u_1(X) = X & u_2(X) = X^2 + X^3 + X + 1 & u_3(X) = 1 \\ & v_1(X) = 1 + X^2 & v_2(X) = X^4 + X^3 + 1 & v_3(X) = 0 \end{array}$$

We verfify that $(X^2 + X^3 + X + 1)(X^2 + 1) = 1 \bmod (X^4 + X^3 + 1)$ *and* $X^2 + X^3 + X + 1$ *is the inverse of* $X^2 + 1$ *modulo* $P(X)$.

Fermat-Euler Approach

- **Theorem**: If $\beta \neq 0$ in $\mathbb{F}_q$, then $\beta^q = \beta$ in $\mathbb{F}_q$. β is a root of $X^q = X$
- **Corollary**: For $\beta \neq 0$ in $\mathbb{F}_q$: $\beta^{q-2} = \beta^{-1}$
- In $GF(p)$ we need an exponentiation to $p-2$ which can be costly.
- In $GF(2^m)$, we have $\beta^{-1} = \beta^{2^m-2}$. *The exponentiation uses the binary representation of the exponent, we can use a square and multiply strategy, minimizing the multiplications considering that* $2^m - 2 = 111 \ldots 1100$ *[17].*

Fermat-Euler Approach

Example in $GF(2^4)$

We consider $GF(2^4)$ *and the canonical base* $(1, \alpha, \alpha^2, \alpha^3)$ *where* α *is a root of* $P(X) = X^4 + X^3 + 1$ *(irreducible). We have* $2^4 - 2 = 14$.
Let $A(X) = X^2 + 1$*, we have*

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

The binary representation of 14 *is* 1110*, thus,*

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

Step by step:

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

Fermat-Euler Approach

Example in $GF(2^{31})$

We consider $GF(2^{31})$. We want to compute $\beta^{2^{31}-2}$, but $2^{31}-2 = 2147483646$ is 1111111111111111111111111111110 in binary.

operation	*valuer*	*exponent*
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
$(\beta^{65535})^{2^{15}}$	$= \beta^{2147450880}$	1111111111111111000000000000000
$(\beta^{255})^{2^7}$	$= \beta^{32640}$	111111110000000
$(\beta^{15})^{2^3}$	$= \beta^{120}$	1111000
$(\beta^3)^{2^1}$	$= \beta^6$	110
$\beta^{2147450880}\beta^{32640}$	$= \beta^{2147483520}$	1111111111111111111111110000000
$\beta^{120}\beta^6$	$= \beta^{126}$	1111110
$\beta^{2147483520}\beta^{126}$	$= \beta^{2147483646}$	1111111111111111111111111111110

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 $(m_1, \ldots, m_k)$
  - RNS representation: $(a_1, \ldots, a_k)$ 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 of 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 $e_1, \ldots e_k$ in $\mathbb{F}_p$. $(k \leq p.)$
  - A polynomial $A(X) = \alpha_0 + \alpha_1 X + \ldots + \alpha_{k-1} X^{k-1}$ over $\mathbb{F}_p$ is given in Lagrange representation by:

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

  - 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 $(a_1(X), \dots a_k(X))$ 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)[8]
- 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 $\mathrm{Montg}(A, 2^s, p) = R$.
- Montgomery notation: $A' = A \times 2^s \bmod p$
  $\mathrm{Montg}(A' \times B', 2^s, p) \equiv (A \times B) \times 2^s \pmod{p}$

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 [18]
Another approach consists in the Newton interpolation where A is correctly reconstructed. [21]
In the polynomial case, the term $-\alpha M$ vanishes.

Extension for Q

By the CRT

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

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 and Kumaresan (1989):

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

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

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

Extension of Residue System Bases

We first translate into 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 [21] (annexe 109), $2k^2 + 3k \rightarrow \sim O(k)$ for RNS [18] (annexe 104), $O(k^2) \rightarrow O(k)$ for Lagrange representation [22] (annexe 112).
- 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 addition of points . 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 [19]

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 $e : G_1 \times G_2 \to G_3$ 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 $\mathbb{F}_{p^k}$ in the formulas. An approach with Residue Systems, similar to the one made on ECC could be interesting [20]
- k is most of the time chosen as a small power of 2 and 3 for algorithmic reasons. Residue arithmetics allows us 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 of the implementation in 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 98

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

m_i \ m_j	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 98

Pair of 5 Moduli - Parallel mode

The dynamical 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 98

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 98

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

Back to 98

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

To compute

$$\psi = F \times T_j^{-1} \bmod T_i. \qquad (1)$$

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

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

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 98

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

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

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

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

Back to 98

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:

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

(We have $\rho = \psi X^b + \psi$ **thus** $\rho \bmod X^b = \psi \bmod X^b$**)**

Back to 98

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

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

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

Back to 98

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

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

Back to 98

For example the multiplication by $45 = (10\bar{1}0\bar{1}01)_2$ gives 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 98

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 98

REFERENCES

Miller, V. (1985).
"Use of elliptic curves in cryptography".
CRYPTO 85: 417-426.

Koblitz, N. (1987).
"Elliptic curve cryptosystems".
Mathematics of Computation 48 (177): 203-209.

Boneh, Dan; Franklin, Matthew (2003).
"Identity-based encryption from the Weil pairing".
SIAM Journal on Computing 32 (3): 586-615.

Joux, Antoine (2004).
"A one round protocol for tripartite Diffie-Hellman".
Journal of Cryptology 17 (4): 263-276.

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.

Paul Barrett
Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor.
Advances in Cryptology – CRYPTO' 86 Lecture Notes in Computer Science Volume 263, 1987, pp 311-323

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

M. Kaihara and N. Takagi
"Bipartite Modular Multiplication Method"
IEEE Trans. on Computers, vol. 57, No. 2, 157-164, Feb. 2007.

E. Mastrovito
"VLSI Architectures for Computation in Galois Fields."

PhD thesis, Linkoping University, Dept. Electr. Eng., 1991

H. Fan and M. A. Hasan,
"A New Approach to Sub-quadratic Space Complexity Parallel Multipliers for Extended Binary Fields,"
IEEE Trans. Computers, vol. 56, no. 2, pp. 224-233, Feb. 2007.

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

J.L. Massey and J.K. Omura
"Computational Method and Apparatus for Finite Field Arithmetic"
US patent No 4,587,627, 1986.

Hasan, Wang, and Bhargava.
A modified massey-omura parallel multiplier for a class of finite fields.

IEEETC: IEEE Transactions on Computers, 42, 1993.

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

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.

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

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)*