Université de Montréal

Implantation d'ECDSA sur une carte à puce

par

Monique Dion

Département d'Informatique et de Recherche Opérationnelle
Faculté des Arts et des Sciences

Mémoire présenté au Département d'Informatique et de Recherche Opérationnelle
en vue de l'obtention du grade de
Matre ès sciences (M.Sc)
en Informatique

mai 1999

Sommaire

Ce mémoire traite de la mise en œuvre d'une signature numérique nommée ECDSA (Elliptic Curve Digital Signature Algorithm) sur une carte á puce. Ce travail fut réalisé pendant un stage à Gemplus Canada, de janvier à septembre 1998.

ECDSA est un procédé de signature basé sur les courbes elliptiques dont la securité repose sur la difficulté présumée du problème du logarithme discret dans le groupe des courbes elliptiques. Même si une courbe elliptique peut être définie sur un corps quelconque, deux corps finis en particulier sont dignes d'intérêt : le corps fini $\mathbf{F}_p$, où p est un nombre premier et le corps fini $\mathbf{F}_{2^m}$, où $m > 1$. Le corps $\mathbf{F}_p$ est un choix propice car la carte à puce utilisée pour la mise en œuvre possédait un cryptoprocesseur qui effectuait des opérations dans ce corps. Le corps fini $\mathbf{F}_{2^m}$ se distingue par sa représentation interne binaire qui permet une mise en œuvre efficace sur une carte à puce.

Le but de ce projet était de générer des signatures numériques selon le procédé ECDSA sur les corps finis $\mathbf{F}_p$ et $\mathbf{F}_{2^m}$, tout en minimisant le temps d'exécution d'une signature. Grâce à de techniques puissantes, le temps d'exécution d'une signature sur chacun des corps $\mathbf{F}_p$ et $\mathbf{F}_{2^m}$ a pu être réduit de 64%.

Mots clés : Cryptologie, signature numérique, ECDSA, courbes elliptiques, corps finis, coordonnées projectives.

Table des matières

Sommaire **i**

Mots clés **i**

Remerciements **vii**

1 Introduction **1**

1.1 Organisation des chapitres 2

2 Notions élémentaires **3**

2.1 Théorie des nombres 3

2.2 Structures algébriques 6

2.2.1 Groupes 6

2.2.2 Anneaux 7

2.2.3 Corps 7

2.2.4 Espaces vectoriels 8

2.2.5 Corps finis 8

2.2.6 Le corps fini $\mathbf{F}_p$, avec p premier 9

2.2.7 Le corps fini $\mathbf{F}_{2^m}$, avec m entier 9

3 Courbes elliptiques **14**

3.1 Les courbes elliptiques sur les nombres réels 14

3.2 Les courbes elliptiques sur $\mathbf{F}_p$ 17

3.3 Les courbes elliptiques sur $\mathbf{F}_{2^m}$ 19

3.4 Le théorème de Hasse 21

3.5 Algorithme d'exponentiation 21

3.6 Signatures numériques 22

3.7 DSA 23

3.8 ECDSA 24

4 Mise en œuvre d'ECDSA sur $\mathbf{F}_p$ **28**

4.1 Les opérations du corps $\mathbf{F}_p$ 29

4.2 Les opérations du groupe $E(\mathbf{F}_p)$ 29
4.2.1 Coordonnées affines 29
4.2.2 Coordonnées projectives 30
4.3 Exponentiation 38
4.3.1 La technique des rateaux 39
4.3.2 La technique de bits condensés 40
4.3.3 La technique de rateaux imbriqués 41
4.4 Résultats 43

5 Mise en œuvre d'ECDSA sur $\mathbf{F}_{2^m}$ **45**
5.1 Base polynomiale 46
5.1.1 Les opérations du corps $\mathbf{F}_{2^m}$ 46
5.2 Base polynomiale composée 50
5.2.1 Les opérations du sous-corps $\mathbf{F}_{2^8}$ 50
5.2.2 Les opérations du sur-corps $\mathbf{F}_{2^{8 \cdot r}}$ 51
5.3 Base normale optimale 55
5.3.1 Les opérations du corps $\mathbf{F}_{2^m}$ 55
5.4 Résumé des représentations des éléments de $\mathbf{F}_{2^m}$ 61
5.5 Les coordonnées projectives 64
5.6 Résultats 76

6 Conclusion **77**
6.1 Représentation des données de $\mathbf{F}_{2^m}$ 77
6.2 Sécurité des bases polynomiales composées 77
6.3 Sécurité du logarithme discret sur les courbes elliptiques 78
6.4 Comparaison avec d'autres mises en œuvres 78

Liste des tableaux

2.1 Tableau des puissances de x 13
2.2 Tableau des logarithmes de β 13

3.1 Correspondance entre la notation de DSA et ECDSA 25
3.2 Correspondance entre $\mathbb{Z}_p^*$ et le groupe $E(\mathbf{F}_q)$ 25

4.1 Nombre de multiplications effectuées dans $\mathbf{F}_p$ (addition, méthode 1) . . 33
4.2 Nombre de multiplications effectuées dans $\mathbf{F}_p$ (addition, méthode 2) . . 34
4.3 Nombre de multiplications effectuées dans $\mathbf{F}_p$ (doublement, méthode 1) 37
4.4 Techniques d'exponentiation 43

5.1 Comparaison des représentations des éléments de $\mathbf{F}_{2^m}$ 62
5.2 Nombre d'opérations effectuées dans $\mathbf{F}_{2^m}$ (addition, méthode 1) 68
5.3 Nombre d'opérations effectuées dans $\mathbf{F}_{2^m}$ (doublement, méthode 1) . . . 70
5.4 Nombre d'opérations effectuées dans $\mathbf{F}_{2^m}$ (addition, méthode 2) 73
5.5 Nombre d'opérations effectuées dans $\mathbf{F}_{2^m}$ (doublement, méthode 2) . . . 75
5.6 Comparaison de deux types de coordonnées projectives 75
5.7 Temps d'exécution des opérations de $\mathbf{F}_{2^m}$ avec une base polynomiale composée 75

Table des figures

3.1 $P + Q = R$ 15
3.2 $P + (-P) = \mathcal{O}$ 16
3.3 $2P = R$ 17

4.1 La technique des rateaux 39
4.2 La technique de rateaux imbriqués ($n = 2$ rateaux, $m = 3$ dents) 41

À mon père.

Remerciements

Je tiens à remercier Gemplus Canada et en particulier Jean-Marc Robert pour m'avoir proposé un projet de recherche. Les huits mois que j'ai passés à Gemplus fut fort agréables. Pendant mon séjour, j'ai pu bénéficier de l'expérience de plusieurs experts ce qui a beaucoup facilité mon travail.

Je voudrais aussi remercier mon directeur Claude Crépeau pour son soutien tout au long de mes études.

Je voudrais aussi remercier le CRSNG qui a subventionné mes deux années à l'Université de Montréal et qui, sans eux, je n'aurais jamais songé à entreprendre des études supérieures.

Chapitre 1

Introduction

L'intérêt manifesté pour les cryptosystèmes basés sur des courbes elliptiques a crû pendant ces dernières années. La sécurité de ces systèmes repose sur l'hypothèse suivante : le problème du logarithme discret dans le groupe basé sur les courbes elliptiques est difficile à résoudre. On suppose en fait qu'il soit plus difficile à résoudre que le problème du logarithme discret dans le groupe multiplicatif Z_p^* et le problème de factorisation des grands nombres. En conséquence, un cryptosystème basé sur des courbes elliptiques admet une clé cryptographique de plus petite taille que les systèmes cités tantôt. L'utilisation de moins d'espace mémoire et l'exécution de moins d'opérations mathématiques, tels sont les avantages qu'apporte une clé de petite taille.

Les contraintes de mémoire et la capacité limitée d'une carte à puce à effectuer des opérations complexes posent un défi particulier lors d'une mise en œuvre d'un cryptosystème à clé publique. Les cryptosystèmes basés sur des courbes elliptiques se présentent comme de candidats idéaux, grâce à leur clé de petite taille.

Dans ce rapport, on ne traite que de la mise en œuvre de la génération d'une signature numérique basée sur des courbes elliptiques nommée ECDSA (Elliptic Curve Digital Signature Algorithm). Pour ce faire, on doit implanter non seulement l'opération d'addition sur les points d'une courbe elliptique, mais aussi les opérations d'addition, de multiplication et d'inversion dans le corps fini sur lequel la courbe elliptique est définie. On ne considère que deux types de corps finis : le corps $\mathbf{F}_p$, où p est un nombre premier et le corps $\mathbf{F}_{2^m}$, où m est un nombre entier. Comme l'addition des points d'une courbe est composée d'opérations dans le corps fini sur lequel la courbe est définie, une mise en œuvre efficace des opérations dans le corps s'avère très importante.

Des implantations de l'algorithme ECDSA sur les corps $\mathbf{F}_p$ et $\mathbf{F}_{2^m}$ ont été faites pendant mon stage à Gemplus Canada. La première partie du projet consistait en améliorer le temps d'exécution d'une version existante d'ECDSA sur le corps $\mathbf{F}_p$. La deuxième partie du projet a pris la forme d'une étude de faisabilité : l'objectif de cette étude était d'implanter l'algorithme ECDSA dans le corps $\mathbf{F}_{2^m}$, sans se servir du crytoprocesseur pour effectuer des opérations dans le corps fini. Comme la carte à puce ne possédait aucune mise en œuvre des opérations dans $\mathbf{F}_{2^m}$, on avait l'embarras du choix en ce qui concerne le choix de représentation des données, les algorithmes utilisés, etc.

L'objectif fondamental de ces deux projets était de minimiser le temps d'exécution de la génération d'une signature numérique. On verra par contre qu'un compromis entre l'utilisation d'espace mémoire et le temps d'exécution était souvent nécessaire à faire.

1.1 Organisation des chapitres

Dans le deuxième chapitre, nous présentons des notions mathématiques, jugées essentielles à la compréhension du rapport. On abordera la théorie des nombres, ainsi qu'une étude brève des structures algébriques, y compris les groupes et les corps finis.

Les courbes elliptiques seront présentées dans le troisième chapitre. L'opération d'addition sur les courbes elliptiques sera illustrée en se servant des courbes définies sur les nombres réels, même si ces types de courbes ne portent aucune valeur cryptographique. L'algorithme ECDSA, l'analogue de DSA sur les courbes elliptiques, sera aussi présenté dans ce chapitre.

Le quatrième chapitre traitera de la mise en œuvre d'ECDSA sur $\mathbf{F}_p$ avec p premier. Puisqu'une implantation de cet algorithme existait déjà sur la carte à puce, des techniques pour réduire le temps d'exécution d'ECDSA seront examinées. La représentation des points sur la courbe, ainsi que l'opération d'exponentiation, qui correspond à l'addition répétée dans ce groupe, comptent parmi les techniques étudiées dans ce chapitre.

La mise en œuvre d'ECDSA sur $\mathbf{F}_{2^m}$ sera traitée dans cinquième chapitre. Les avantages et les inconvénients des différentes façons de représenter les éléments de ce corps seront discutés dans ce chapitre. Les algorithmes d'addition, de multiplication et d'inversion associés avec chaque représentation seront présentés. Des techniques semblables à celles exposées au chapitre précédent pour réduire le temps d'exécution de l'algorithme ECDSA seront aussi examinées.

Chapitre 2

Notions élémentaires

Le but de ce chapitre est de réunir certains concepts en mathématiques, telles la théorie des nombres et l'algèbre, qui sont jugés essentiels à la compréhension de la suite du rapport.

2.1 Théorie des nombres

Le principe du bon ordre, qui servira à démontrer plusieurs théorèmes, est tenu pour acquis :

Théorème 1 (Principe du bon ordre). *Tout ensemble non vide* $\mathbf{S} \subset \mathbb{N}$ *contient un plus petit élément.*

Définition 1. Soient $a, b \in \mathbb{Z}$ avec $a \neq 0$. On dit que a **divise** b s'il existe un entier k tel que $b = ak$. Dans ce cas, on écrit $a|b$. Si a ne divise pas b, on écrit $a \nmid b$.

Définition 2. Un entier $p > 1$ est appelé un **nombre premier** si ses seuls diviseurs sont 1 et p.

Définition 3. Soit $n \in \mathbb{Z} \setminus \{0\}$. On dit que a est **congru** à b **modulo** n, et on écrit $a \equiv b \bmod n$, si $n|(a-b)$. Sinon, on écrit $a \not\equiv b \bmod n$.

Définition 4. Soient $a, b \in \mathbb{Z}$ tels que $ab \neq 0$. Le **plus grand commun diviseur** (*p.g.c.d.*) de a et b, noté (a, b), est l'entier positif d qui satisfait aux deux conditions suivantes :

1. $d|a$ et $d|b$;
2. Si $c|a$ et $c|b$, alors $c|d$.

Théorème 2. *Soient* $a, b, c \in \mathbb{Z}$. *Si* $a|b$ *et* $a|c$, *alors* $a|(bx + cy)$, *quels que soient* $x, y \in \mathbb{Z}$.

Démonstration. Si $a|b$ et $a|c$, alors il existe des entiers q et r tels que $b = aq$ et $c = ar$. Il en résulte que

$$bx + cy = aqx + ary = a(qx + ry)$$

et ainsi que $a|(bx + cy)$ quels que soient $x, y \in \mathbb{Z}$. □

Théorème 3 (Division euclidienne). *Soient $a, b \in \mathbb{Z}$ avec $a > 0$. Il existe des entiers q et r tels que $b = aq + r$, où $0 \leq r < a$.*

Démonstration. Considérons l'ensemble $S = \{b - ma \mid m \in \mathbb{Z}, b - ma \geq 0\}$. Il est facile à voir que $S \subset \mathbb{N} \cup \{0\}$ et que $S \neq \varnothing$. On conclut, d'après le principe du bon ordre, que S contient un plus petit élément $r \geq 0$. Soit q l'entier satisfaisant à $r = b - qa$. Ainsi, on a $b = aq + r$. Il reste à démontrer que $r < a$. Supposons le contraire, c'est-à-dire que $r \geq a$. Dans ce cas, on a $b - qa \geq a$, ce qui est équivalent à $b - (q + 1)a \geq 0$. Mais $b - (q + 1)a \in S$ et $b - (q + 1)a < b - qa$, ce qui contredit le fait que $b - qa$ est le plus petit élément de S. On a donc $r < a$. □

Théorème 4. *Soient $a, b \in \mathbb{Z}$ tels que $ab \neq 0$. Il existe des entiers x_0 et y_0 tels que*

$$(a, b) = ax_0 + by_0.$$

Démonstration. Considérons l'ensemble $S = \{ax + by \mid x, y \in \mathbb{Z}, ax + by > 0\}$. Comme $S \subset \mathbb{N}$ et $S \neq \varnothing$, on peut utiliser le principe du bon ordre et conlure que S possède un plus petit élément d. On peut alors écrire $d = ax_0 + by_0$, pour un certain choix $x_0, y_0 \in \mathbb{Z}$. Il suffit de montrer que $d = (a, b)$. Pour ce faire, on montre en premier lieu que $d|a$ et $d|b$. Supposons que $d \nmid a$. D'après la division euclidienne (théorème 3), il existe $q, r \in \mathbb{Z}$ tels que $a = qd + r$, où $0 < r < d$. Mais

$$r = a - qd = a - q(ax_0 + by_0) = a(1 - qx_0) + b(-qy_0)$$

et donc $r \in S$ et $r < d$, ce qui contredit le fait que d est le plus petit élément de S. Donc $d|a$. On montre que $d|b$ de la même façon. D'autre part, si $c|a$ et $c|b$, d'après le théorème 2, on sait que $c|(ax + by)$, quels que soient $x, y \in \mathbb{Z}$; en particulier $c|d$. On a donc $d = (a, b)$. □

Théorème 5. *Soient $d = (a, b)$ et $m \in \mathbb{Z}$; alors $(a, b + ma) = (a, b)$.*

Démonstration. Soit $g = (a, b + ma)$. Comme $d|a$ et $d|b$, alors $d|(b + ma)$ et ainsi $d|a$ et $d|(b + ma)$ entraînent $d|g$. Mais $g|a$ et $g|(b + ma)$ impliquent que $g|a$ et $g|b$, c'est-à-dire $g|(a, b) = d$. On conclut que $d = (a, b) = (a, b + ma) = g$. □

Théorème 6 (Algorithme d'Euclide). *Soient $a, b \in \mathbb{Z}$, où $a > 0$. En appliquant successivement la division euclidienne (théorème 3), on obtient la suite d'équations*

$$\begin{array}{rcll} b &=& aq_1 + r_1, & 0 < r_1 < a \\ a &=& r_1q_2 + r_2, & 0 < r_2 < r_1 \\ r_1 &=& r_2q_3 + r_3, & 0 < r_3 < r_2 \\ & \vdots & & \\ r_{j-3} &=& r_{j-2}q_{j-1} + r_{j-1}, & 0 < r_{j-1} < r_{j-2} \\ r_{j-2} &=& r_{j-1}q_j + r_j, & 0 < r_j < r_{j-1} \\ r_{j-1} &=& r_jq_{j+1}. & \end{array}$$

Si $d = (a, b)$, alors $d = r_j$. Les entiers x_0, y_0 satisfaisant à $d = ax_0 + by_0$ peuvent être obtenus par élimination de $r_1, r_2, \ldots r_{j-1}$ du système d'équations.

Démonstration. On veut d'abord montrer que $r_j = (a, b)$. D'après le théorème 5, on a sucessivement

$$\begin{array}{rcl} (a,b) &=& (a, b - aq_1) = (a, r_1) \\ = \ (r_1, a) &=& (r_1, a - r_1q_2) = (r_1, r_2) \\ = \quad \ldots &=& (r_{j-1}, r_j) = r_j. \end{array}$$

Pour démontrer la deuxième partie du théorème, on écrit l'avant-dernière équation du système sous la forme

$$r_j = r_{j-2} - q_jr_{j-1}.$$

En utilisant l'équation précédant l'avant-dernière équation du système, on obtient

$$r_j = r_{j-2} - q_j(r_{j-3} - q_{j-1}r_{j-2}) = (1 + q_jq_{j-1})r_{j-2} + (-q_j)r_{j-3}.$$

En continuant de cette façon, on arrive à exprimer r_j comme une combinaison linéaire de a et b. □

Théorème 7. *Soient $a, b \in \mathbb{Z}$ tels que $ab \neq 0$. Alors $(a, b) = 1 \iff$ il existe $x, y \in \mathbb{Z}$ tels que $ax + by = 1$.*

Démonstration.
($\Longrightarrow$) Si $(a, b) = 1$, alors il existe, d'après le théorème 4, $x, y \in \mathbb{Z}$ tels que $ax + by = 1$.
($\Longleftarrow$) Soit $d = (a, b)$. D'après le théorème 2, on a que $d | ax + by = 1$ et alors $d = 1$. □

Exemple 1. Utilisons l'algorithme d'Euclide pour calculer le plus grand commun diviseur de 967 et 429 et exprimons ce nombre comme une combinaison linéaire de 967 et 429.

Solution.

$$\begin{aligned}
967 &= 429 \cdot 2 + 109 \\
429 &= 109 \cdot 3 + 102 \\
109 &= 102 \cdot 1 + 7 \\
102 &= 7 \cdot 14 + 4 \\
7 &= 4 \cdot 1 + 3 \\
4 &= 3 \cdot 1 + 1 \\
3 &= 1 \cdot 3
\end{aligned}$$

D'après le théorème 6, on déduit que $(967, 429) = 1$.

De plus, on a

$$\begin{aligned}
1 &= 4 - 3 \\
&= 4 - (7 - 4) = 2 \cdot 4 - 7 \\
&= 2(102 - 7 \cdot 14) - 7 = 2(102) - 7 \cdot 29 \\
&= 2(102) - 29(109 - 102) = 31(102) - 29(109) \\
&= 31(429 - 109 \cdot 3) - 29(109) = 31(429) - 122(109) \\
&= 31(429) - 122(967 - 429 \cdot 2) \\
&= 275(429) - 122(967).
\end{aligned}$$

2.2 Structures algébriques

Nous allons examiner de plus près les groupes, les anneaux et les corps ; en particulier, les corps finis.

2.2.1 Groupes

Définition 5. Un **groupe** $(G, *)$ est formé d'un ensemble G et d'une loi de composition $*$ sur un ensemble G. Le groupe doit posséder les propriétés suivantes :

1. $\forall a, b, c \in G : a * (b * c) = (a * b) * c$ (associativité) ;
2. $\exists 1 \in G$ tel que $\forall a \in G : a * 1 = 1 * a = a$ (existence d'un élément neutre) ;
3. $\forall a \in G, \exists a^{-1} \in G$ tel que $a^{-1} * a = a * a^{-1} = 1$ (existence d'un élément inverse pour tout élément du groupe).

Un groupe G est **abélien** ou **commutatif** si, de plus, $\forall a, b \in G, a * b = b * a$.

On remarque qu'on a utilisé la notation multiplicative. S'il s'agit, au lieu, d'un groupe muni d'une opération additive, on note l'élément neutre par 0 et l'inverse d'un élément a par $-a$.

Exemple 2. L'ensemble des entiers $\mathbb{Z}$ muni de l'opération d'addition forme un groupe. L'élément neture est 0 et l'inverse d'un entier a est $-a$. Comme l'addition est commutative, le groupe $(\mathbb{Z}, +)$ est abélien.

Exemple 3. Les entiers muni de l'addition modulo n forment un groupe additif, noté $\mathbb{Z}_n$.

Définition 6. Le groupe multiplicatif de $\mathbb{Z}_n$ est $\mathbb{Z}_n^* = \{a \in \mathbb{Z}_n \mid (a, n) = 1\}$. En particulier, si n est premier, $\mathbb{Z}_n^* = \{a \mid 1 \leq a \leq n-1\}$.

Définition 7. Soit G un groupe de n éléments. G est dit **fini** si n est fini. Dans ce cas, n désigne **l'ordre** du groupe.

Définition 8. Un sous-ensemble non-vide H d'un groupe G est un sous-groupe de G si H est lui-même un groupe avec l'opération du groupe G.

Définition 9. Un élément a d'un groupe G est **d'ordre** s si s est le plus petit entier positif tel que $a^s = 1$.

Définition 10. Soit G un groupe d'ordre n et soient $\alpha, \beta \in G$. Le problème du **logarithme discret** dans G est de trouver l'entier x, $0 \leq x \leq n-1$, tel que $\alpha^x = \beta$, si un tel x existe.

2.2.2 Anneaux

Définition 11. Un **anneau** est un triplet $(R, +, \times)$ formé d'un ensemble R et deux lois de composition $+$ et $\times$. L'anneau doit posséder les propriétés suivantes :

1. $(R, +)$ est un groupe abélien ;
2. L'opération $\times$ est associative. C'est-à-dire, $\forall a, b, c \in R, a \times (b \times c) = (a \times b) \times c$;
3. L'opération $\times$ admet un élément neutre 1 tel que $\forall a \in R$, $a \times 1 = 1 \times a = a$;
4. La loi de composition $\times$ est distributive par rapport à l'opération $+$. C'est-à-dire, $\forall a, b, c \in R$, $a \times (b + c) = (a \times b) + (a \times c)$ et $(b + c) \times a = (b \times a) + (c \times a)$.

Un anneau est dit **commutatif** si $\forall a, b \in R$, $a \times b = b \times a$.

Exemple 4. L'ensemble des entiers $\mathbb{Z}$ muni des lois d'addition et de multiplication habituelles forme un anneau commutatif.

Exemple 5. L'ensemble $\mathbb{Z}_n$ forme un anneau commutatif avec les opérations d'addition et de multiplication modulo n.

2.2.3 Corps

Définition 12. Un **corps** F est un anneau commutatif où tout élément non nul possède un inverse multiplicatif.

Exemple 6. $\mathbb{Z}_p$ est un corps si et seulement si p est premier.

Exemple 7. L'ensemble des entiers $\mathbb{Z}$ muni des opérations habituelles d'addition et de multiplication ne forment pas un corps, car les seuls éléments non nuls possédant un inverse sont 1 et -1.

2.2.4 Espaces vectoriels

Définition 13. Un **espace vectoriel** V sur un corps F est un groupe abélien $(V, +)$, muni d'un produit défini entre les éléments de F et ceux de V tel que $\forall a, b \in F$ et $\forall v, w \in V$,

1. $a(v + w) = av + aw$;
2. $(a + b)v = av + bv$;
3. $(ab)v = a(bv)$;
4. $1v = v$.

Les éléments de V sont des vecteurs tandis que les éléments de F sont des scalaires.

Définition 14. Soit $S = \{v_1, v_2, \dots, v_n\}$ un sous-ensemble fini d'un espace vectoriel V sur F. Si un vecteur w est tel que $w = c_1v_1 + c_2v_2 + \cdots + c_nv_n$, avec chaque $c_i \in F$, on dit que w est une **combinaison linéaire** des vecteurs $v_1, v_2, \dots, v_n$, ou encore que w est engendré par ces vecteurs. Si chacun des vecteurs d'un espace vectoriel V peut être engendré par un sous-ensemble S de V, on dit que S **engendre** V.

Définition 15. Les vecteurs $v_1, v_2, \dots, v_n$ sont dits **linéairement dépendants** s'il existe des scalaires $c_1, c_2, \dots, c_n$ non tous nuls tels que $c_1v_1 + c_2v_2 + \cdots + c_nv_n = 0$. Sinon, ils sont **linéairement indépendants**.

Définition 16. Un ensemble de vecteurs $S = \{v_1, v_2, \dots, v_n\}$ est une **base** de V si les deux conditions suivantes sont vérifiées :

1. $v_1, v_2, \dots, v_n$ sont linéairement indépendants ;
2. $v_1, v_2, \dots, v_n$ engendrent V.

Le nombre d'éléments de S est appelé la **dimension** de V.

Si S est une base de V, chaque élément de V peut être représenté de façon unique par une combinaison linéaire des éléments de S.

2.2.5 Corps finis

Définition 17. Un **corps fini** est un corps F avec un nombre fini d'éléments. Ce nombre est appelé **l'ordre** de F.

Définition 18. La caractéristique d'un corps F est le plus petit entier positif m tel que $\sum_{i=1}^{m} 1 = 0$. Si cette somme n'est jamais égale à 0, on dit que le corps est de caractéristique 0.

Exemple 8. F_p est un corps de caractéristique p et F_{2^m} est un corps de caractéristique 2.

Pour une démonstration des prochains théorèmes, consultez le livre de Lidl and Niederreiter [LN94].

Théorème 8. *Si F est un corps fini, alors F possède p^m éléments, pour un nombre premier p et un entier $m \geq 0$.*

Théorème 9. *Il existe sur tout corps fini F de q éléments un* ***élément primitif****, c'est-à-dire un élément g tel que $g^1, g^2, \ldots, g^{q-1}$ énumère les $q-1$ éléments non nuls du corps.*

Théorème 10. *Soit F un corps de q éléments, alors $a^q = a$, pour tout $a \in F_q$.*

Nous limiterons notre discussion à deux types de corps finis : ceux avec un nombre premier p d'éléments et ceux dont le nombre d'éléments est $q = 2^m$, avec m entier.

2.2.6 Le corps fini $\mathbf{F}_p$, avec p premier

Le corps fini $\mathbf{F}_p$ est tout simplement $\mathbb{Z}_p$ avec les opérations d'addition et de multiplication modulo p. L'inverse multiplicatif d'un élément peut être calculé en utilisant l'algorithme d'Euclide (théorème 6).

Exemple 9. Reprenons l'exemple 1. On suppose qu'on utilise $\mathbb{Z}_{967}$, le corps de 967 éléments. On veut trouver l'inverse multiplicatif de l'élément 429. On sait déjà que $1 = 275(429) - 122(967)$. En réduisant cette équation modulo 967, on obtient $1 \equiv 275 \cdot 429 \mod 967$. L'inverse de 429 modulo 967 est alors 275.

2.2.7 Le corps fini $\mathbf{F}_{2^m}$, avec m entier

Considérons maintenant le corps $\mathbf{F}_{2^m}$. On peut y penser comme étant un espace vectoriel de dimension m sur $\mathbf{F}_2$ dont chaque élément $u \in \mathbf{F}_{2^m}$ peut être exprimé de façon unique :

$$u = a_0 w_0 + a_1 w_1 + \cdots + a_{m-1} w_{m-1},$$

avec $a_i \in \mathbf{F}_2$. L'ensemble des éléments $\{w_0, w_1, \ldots, w_{m-1}\}$ forme une base de $\mathbf{F}_{2^m}$. Plusieurs choix de $\{w_0, w_1, \ldots, w_{m-1}\}$ existent pour une base, chacun utilisant de différents algorithmes pour la multiplication et la division dans $\mathbf{F}_{2^m}$. Nous considérons maintenant une représentation polynomimale du corps $\mathbf{F}_{2^m}$. Une autre représentation, nommée base normale optimale, sera présentée au chapitre 5.

Avant de décrire la construction de ce corps, quelques définitions sont présentées.

Les polynômes sur le corps fini $\mathbf{F}_2$

On décrit d'abord comment effectuer des opérations sur des polynômes définis sur le corps fini $\mathbf{F}_2$.

Définition 19. Un polynôme de l'indéterminé x sur le corps $\mathbf{F}_2$ est une expression de la forme suivante :

$$f(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0,$$

avec $a_i \in \{0, 1\}$, $(0 \leq i \leq n)$ et $n \geq 0$. On désigne l'ensemble de ces polynômes $\mathbf{F}_2[x]$. On définit le degré de $f(x)$, noté $\deg(f)$, comme étant le plus grand exposant d'un terme non nul de $f(x)$.

Soient $f(x) = \sum_{i=0}^{n} a_i x^i$ et $g(x) = \sum_{i=0}^{n} b_i x^i$ deux polynômes sur $\mathbf{F}_2$. La somme de $f(x)$ et $g(x)$ est

$$f(x) + g(x) = \sum_{i=0}^{n} (a_i + b_i) x^i.$$

Soient $f(x) = \sum_{i=0}^{n} a_i x^i$ et $g(x) = \sum_{j=0}^{m} b_j x^j$. Le produit de $f(x)$ et $g(x)$ est

$$f(x)g(x) = \sum_{k=0}^{n+m} c_k x^k, \text{ avec } c_k = \sum_{\substack{i+j=k \\ 0 \leq i \leq n, 0 \leq j \leq m}} a_i b_j$$

La notion de divisibilité s'applique aussi aux polynômes.

Définition 20. Soient $f(x), g(x) \in \mathbf{F}_2[x]$ tels que $f(x) \neq 0$. On dit que $f(x)$ divise $g(x)$ et on note $f(x) \mid g(x)$ s'il existe un polynôme $q(x) \in \mathbf{F}_2[x]$ tel que $g(x) = q(x)f(x)$.

Définition 21 (Algorithme de division pour les polynômes).
Soient $f(x), g(x) \in \mathbf{F}_2[x]$ tels que $f(x) \neq 0$. Si on divise $g(x)$ par $f(x)$, on obtient un unique quotient $q(x)$ et un reste $r(x)$ tels que

$$g(x) = q(x)f(x) + r(x) \text{ avec } \deg(r) < \deg(f).$$

On écrit $g(x) \bmod f(x)$ pour le reste et $g(x) \operatorname{div} f(x)$ pour le quotient.

Définition 22 (Congruence des polynômes). Soient $f(x), g(x), h(x) \in \mathbf{F}_2[x]$. On écrit $g(x) \equiv h(x) \pmod{f(x)}$ si

$$f(x)|(g(x) - h(x)).$$

On voit la ressemblance entre la définition de congruence chez les polynômes et les nombres entiers.

Définition 23. Un polynôme $f(x)$ de $\mathbf{F}_2[x]$ est irréductible s'il n'existe pas de polynômes $g(x), h(x) \in \mathbf{F}_2[x]$ tels que

$$f(x) = g(x)h(x) \text{ avec } \deg(g) > 0 \text{ et } \deg(h) > 0.$$

Le concept du polynôme irréductible est essentiel quant à la construction des corps finis.

Le corps de 2^m éléments

Pour construire un corps de 2^m éléments, on choisit les 2^m polynômes de $\mathbf{F}_2[x]$ de degré au plus $m-1$. L'addition et la multiplication se définissent comme dans $\mathbf{F}_2[x]$, sauf ces opérations sont suivies d'une réduction modulo $f(x)$, où $f(x)$ est un polynôme irréductible de degré m. On note ce corps par $\mathbf{F}_2[x]/f(x)$. Le calcul des inverses dans ce corps se fait à l'aide de l'algorithme d'Euclide, appliqué aux polynômes.

Le polynôme irréductible $f(x)$ joue un rôle semblable à celui du nombre premier p dans la construction de $\mathbb{Z}_p$. Tout entier est congru modulo p à un élément unique dans $\mathbb{Z}_p$; tout polynôme de $\mathbf{F}_2[x]$ est congru modulo $f(x)$ à un polynôme unique de $\mathbf{F}_2[x]$ de degré au plus $m-1$.

Exemple 10. Construisons un corps de 8 éléments. Comme $8 = 2^3$, nous cherchons un polynôme irréductible de degré 3 dans $\mathbf{F}_2[x]$. On choisit le polynôme $f(x) = x^3 + x + 1$. Le corps $\mathbf{F}_2/(x^3 + x + 1)$ comprend les huit polynômes $0, 1, x, x+1, x^2, x^2+1, x^2+x$ et x^2+x+1. Un polynôme $a_2x^2 + a_1x + a_0$ dans $\mathbf{F}_2/(x^3+x+1)$ peut être exprimé sous forme de triplet $a_2a_1a_0$. La somme de deux éléments $a_2a_1a_0$ et $b_2b_1b_0$ est définie ainsi :

$$a_2a_1a_0 + b_2b_1b_0 = c_2c_1c_0, \text{ où } c_i = a_i \oplus b_i,\ i = 1, 2, 3,$$

où $\oplus$ est l'addition dans $\mathbf{F}_2$. Pour calculer le produit de deux éléments du corps, on multiplie les deux polynômes et on réduit le résultat modulo $x^3 + x + 1$. Ceci nous donne un polynôme de degré au plus 2, qui est un élément de $\mathbf{F}_2/(x^3 + x + 1)$.

Multiplions (x^2+1) et (x^2+x+1) dans $\mathbf{F}_2/(x^3+x+1)$. On calcule d'abord le produit de (x^2+1) et (x^2+x+1) dans $\mathbf{F}_2[x]$.

$$(x^2+1)(x^2+x+1) = x^4 + x^3 + x^2 + x^2 + x + 1 = x^4 + x^3 + x + 1$$

Ensuite, on divise le résultat par (x^3+x+1) et on obtient :

$$x^4+x^3+x+1=(x^3+x+1)(x+1)+(x^2+x).$$

On a $(x^2+1)(x^2+x+1)=x^2+1$ dans le corps $\mathbf{F}_2/(x^3+x+1)$.

L'algorithme d'Euclide peut être utilisé pour calculer l'inverse d'un élément de $\mathbf{F}_{2^3}$.

Le produit de deux éléments du corps peut être également calculé à partir des tableaux de logarithmes et antilogarithmes. Pour ce faire, il nous faut un élément primitif. Choisissons l'élément $\alpha = x$ et calculons les puissances successives de cet élément.

$$\begin{array}{lllll}
x & = & 010 & & \\
x^2 & = & 100 & & \\
x^3 & = & x+1 & = & 011 \\
x^4 & = & x^2+x & = & 110 \\
x^5 & = & x^2+x+1 & = & 111 \\
x^6 & = & x^2+1 & = & 101 \\
x^7 & = & 1 & = & 001
\end{array}$$

On voit que les puissances successives de $\alpha = x$ énumèrent les 7 éléments non nuls du corps.

Soient $\beta \in \mathbf{F}_{2^3}$ et $\alpha = x$. On construit un tableau des valeurs de k, $(1 \leq k \leq 7)$ tel que $\alpha^k = \beta$, ainsi qu'un tableau de logarithmes. Pour calculer le produit de 011 et 100, il faut d'abord trouver le logarithme de chaque élément. Le tableau 2.2 nous donne $\log_\alpha(011) = 3$ et $\log_\alpha(100) = 2$. Alors $(011)(100) = \alpha^{3+2} = \alpha^5 = (111)$.

Des tableaux sont aussi utiles pour trouver l'inverse d'un élément lorsque le corps possède peu d'éléments. On verra l'utilité de ces types de tableaux dans le chapitre 5.

k	$\beta = \alpha^{\mathbf{k}}$
1	010
2	100
3	011
4	110
5	111
6	101
7	001

Tab. 2.1: Tableau des puissances de x

β	$\log_\alpha \beta$
001	7
010	1
011	3
100	2
101	6
110	4
111	5

Tab. 2.2: Tableau des logarithmes de β

Chapitre 3

Courbes elliptiques

Même si les courbes elliptiques n'ont été proposées pour de fins cryptographiques qu'en 1986 par Victor Miller [Mil86] and Neal Koblitz [Kob87], l'étude de ces courbes n'est pas un phénomène récent, car elles ont été étudiées depuis plus d'un siècle. À l'heure actuelle, les courbes elliptiques figurent dans de puissants algorithmes de vérification de primalité et de factorisation. Tout récemment, le dernier théorème de Fermat a pu être démontré grâce aux courbes elliptiques.

Nous abordons ce sujet en présentant les courbes elliptiques définies sur les nombres réels. Quoique ces types de courbes n'aient aucune valeur cryptographique, elles sont néanmoins utiles pour démontrer les propriétés géométriques que possèdent les courbes elliptiques.

3.1 Les courbes elliptiques sur les nombres réels

Définition 24. Une courbe elliptique E sur les nombres réels $\mathbb{R}$ est définie comme étant l'ensemble des points $(x, y) \in \mathbb{R} \times \mathbb{R}$ tels que

$$y^2 = x^3 + ax + b, \qquad \text{avec } a, b \in \mathbb{R}.$$

Un groupe additif associé avec la courbe peut être défini si $4a^3 + 27b^2 \neq 0$. Le groupe se compose des points sur la courbe et d'un point spécial, nommé le point à l'infini, l'élément neutre du groupe. La loi de composition est définie de façon géométrique.

Soit $P = (x_1, y_1)$ un point de la courbe E. L'inverse additif du point P est le point $-P = (x_1, -y_1)$.

Soient $P = (x_1, y_1)$ et $Q = (x_2, y_2)$ deux points de E tels que $P \neq Q$. Pour trouver la somme de P et Q, on se sert de la droite qui passe par ces deux points et qui ne coupe la courbe qu'à un troisième point $-R = (x_3, -y_3)$. La somme de P et Q devient le point $R = (x_3, y_3)$, le point symétrique à $-R$ par rapport à l'axe horizontal.

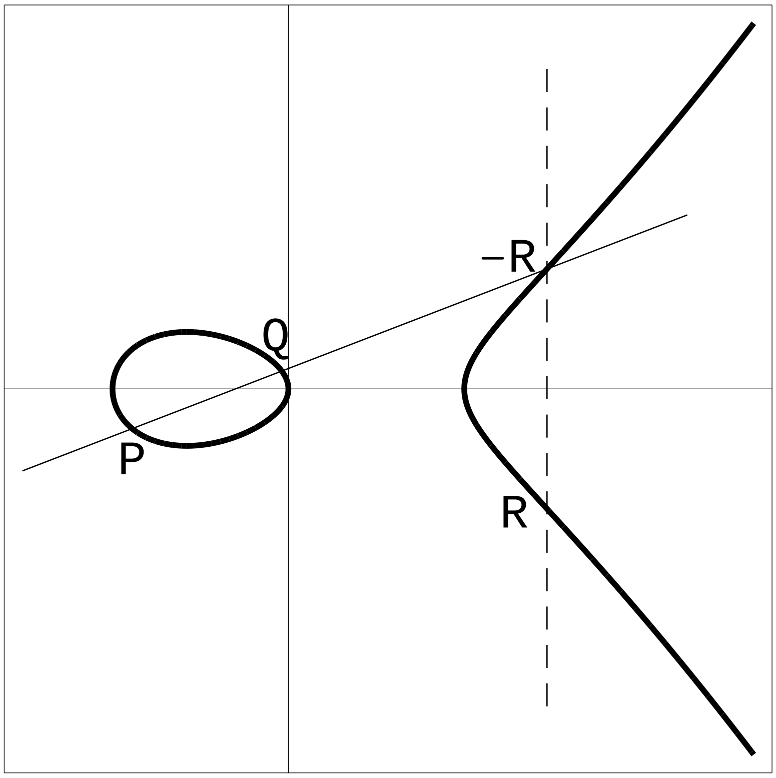

FIG. 3.1: $P + Q = R$

Algébriquement, le point $R = (x_3, y_3)$ est calculé de la façon suivante :

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \\ \text{où} \quad \lambda &= \frac{(y_2 - y_1)}{(x_2 - x_1)}. \end{aligned}$$

Si on essaye d'appliquer la méthode géometrique précédente à la somme de P et Q lorsque $P = -Q$, on voit que la droite qui passe par ces deux points ne coupe pas la courbe à un troisième point. On définit alors le point à l'infini comme étant le point infiniment éloigné dans la direction de l'axe y. Ce point, noté $\mathcal{O}$, devient le troisième point d'intersection avec P et Q.

Le point à l'infini $\mathcal{O}$ représente l'élément neutre du groupe additif. Pour tout point

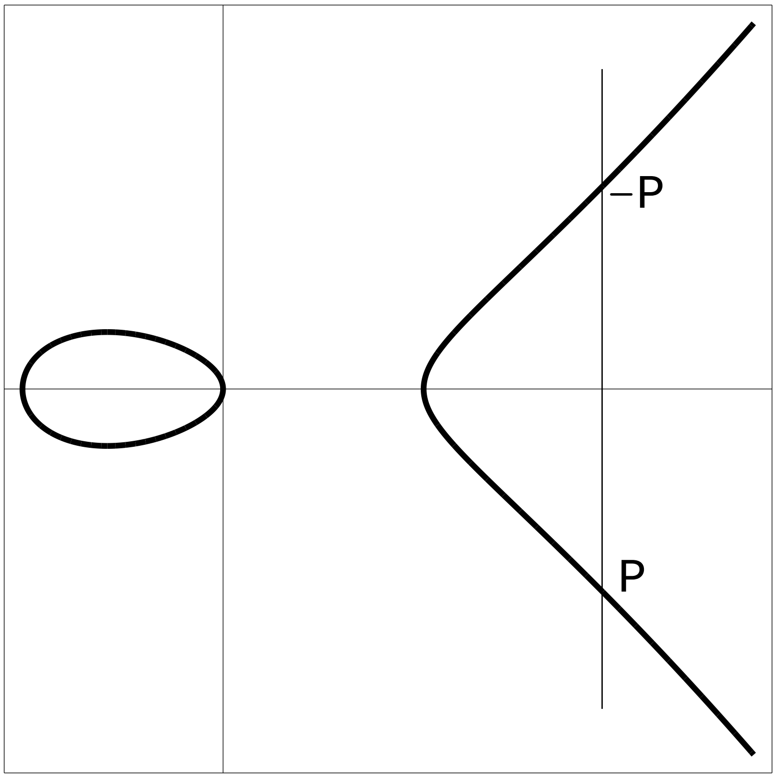

FIG. 3.2: $P + (-P) = \mathcal{O}$

$P \in E$, on a

$$P + \mathcal{O} = \mathcal{O} + P = P.$$

Pour une démonstration de la propriété d'associativité du groupe, consultez le livre de Silverman et Tate [ST92].

Soit $P = (x_1, y_1) \in E$ tel que $y_1 \neq 0$. Pour trouver la somme de P et P, on se sert de la tangente de la courbe au point P. La tangente ne coupe la courbe qu'à un point $-R = (x_3, y_3)$. Le point $P + P = 2P$ devient le point $R = (x_3, y_3)$.

Le point $R = (x_3, y_3)$ est calculé de la manière suivante :

$$\begin{aligned} x_3 &= \lambda^2 - 2x_1 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \\ \text{où} \quad \lambda &= \frac{(3x_1^2 + a)}{2y_1}. \end{aligned}$$

Dans le cas où $y_1 = 0$, la tangente devient une droite verticale qui ne coupe pas la courbe à un autre point. Dans ce cas, $P + P = 2P = \mathcal{O}$.

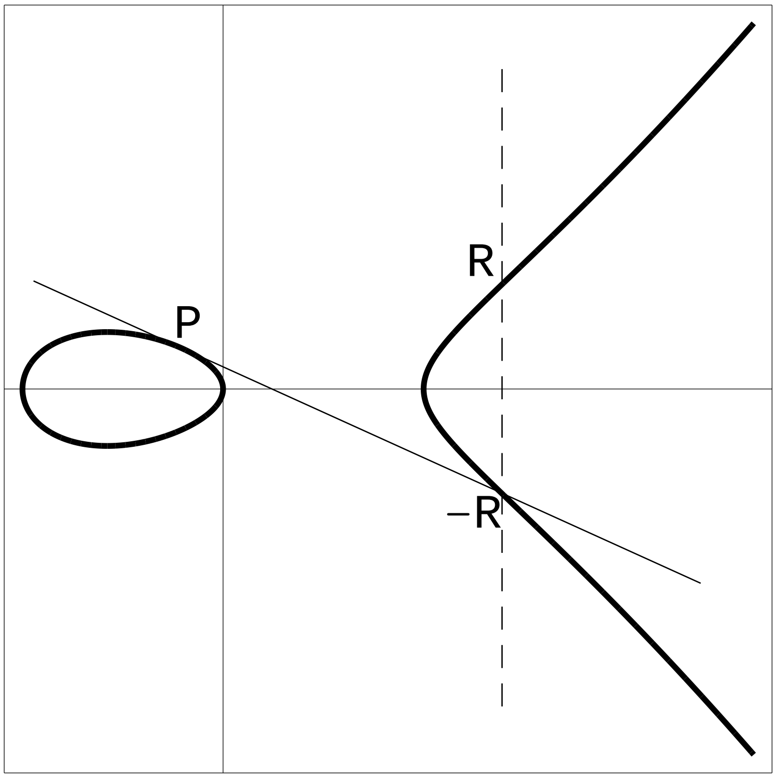

FIG. 3.3: $2P = R$

3.2 Les courbes elliptiques sur $\mathbf{F}_p$

Nous allons maintenant examiner les courbes elliptiques définies sur les corps finis $\mathbf{F}_p$, avec $p > 3$.

Définition 25. Soit un nombre premier $p > 3$. Une courbe elliptique E sur $\mathbf{F}_p$ est définie comme étant l'ensemble des points $(x, y) \in \mathbf{F}_p \times \mathbf{F}_p$ tels que

$$y^2 \equiv x^3 + ax + b \bmod p, \quad \text{avec } a, b \in \mathbf{F}_p.$$

Un groupe additif associé avec la courbe, noté $E(\mathbf{F}_p)$, peut être défini si $4a^3 + 27b^2 \not\equiv 0 \bmod p$. Le groupe se compose des points de la courbe ainsi que le point à l'infini $\mathcal{O}$. La loi de composition du groupe est la même que dans le cas des réels sauf que les opérations sont effectuées dans le corps $\mathbf{F}_p$ et non les réels.

Soit $P = (x_1, y_1) \in E(\mathbf{F}_p)$. L'inverse additif de P est $-P = (x_1, -y_1 \bmod p)$.

Soient P et Q deux points de la courbe tels que $P \neq Q$. Le point $R = P + Q$ est calculé de la manière suivante :

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \\ \text{où} \quad \lambda &= \frac{(y_2 - y_1)}{(x_2 - x_1)}. \end{aligned}$$

Si $P = -Q,\ P + Q = \mathcal{O}$.

Soit $P = (x_1, y_1)$ tel que $y_1 \neq 0$. On calcule le point $P + P = 2P$ de la façon suivante :

$$\begin{aligned} x_3 &= \lambda^2 - 2x_1 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \\ \text{où} \quad \lambda &= \frac{(3x_1^2 + a)}{2y_1}. \end{aligned}$$

Si $y_1 = 0, \quad 2P = \mathcal{O}$.

Exemple 11. Soit la courbe elliptique $y^2 = x^3 + x + 6$ définie sur $\mathbf{F}_{11}$. Ici, $a = 1$ et $b = 6$ et on voit que $4a^3 + 27b^2 \not\equiv 0 \bmod 11$. Les 12 points de cette courbe sont

$$\begin{array}{cccccc} (2,7), & (5,2), & (8,3), & (10,2), & (3,6), & (7,9) \\ (7,2), & (3,5), & (10,9), & (8,8), & (5,9), & (2,4) \end{array}$$

plus $\mathcal{O}$, le point à l'infini.

Prenons $P = (2,7)$ et $Q = (7,9)$ et calculons $R = P + Q$. La formule d'addition donne

$$\begin{aligned} \lambda &= \frac{9-7}{7-2} = \frac{2}{5} = 2 \cdot 9 \equiv 7 \bmod 11 \\ x_3 &= 7^2 - 2 - 7 \equiv 7 \bmod 11 \\ y_3 &= 7(2-7) - 7 \equiv 2 \bmod 11. \end{aligned}$$

On a alors $R = (x_3, y_3) = (7, 2)$.

Calculons maintenant $R = 2P$. La formule de doublement donne

$$\begin{aligned} \lambda &= \frac{3 \cdot 2^2 + 1}{2(7)} = \frac{2}{3} = 2 \cdot 4 \equiv 8 \bmod 11 \\ x_3 &= 8^2 - 2 \cdot 2 \equiv 5 \bmod 11 \\ y_3 &= 8(2-5) - 7 \equiv 2 \bmod 11. \end{aligned}$$

On a alors $2(2,7) = (5,2)$. Le point $(2,7)$ est en fait un générateur du groupe $E(\mathbf{F}_{11})$ car il engendre les 13 points de la courbe.

Passons maintenant aux courbes elliptiques définies sur $\mathbf{F}_{2^m}$ pour un m entier.

3.3 Les courbes elliptiques sur $\mathbf{F}_{2^m}$

Définition 26. Une courbe elliptique E sur $\mathbf{F}_{2^m}$ pour un m entier est définie comme étant l'ensemble des points $(x, y) \in \mathbf{F}_{2^m} \times \mathbf{F}_{2^m}$ tels que

$$y^2 + xy = x^3 + ax^2 + b, \quad \text{avec } a, b \in \mathbf{F}_{2^m}.$$

Un groupe additif associé avec la courbe, noté $E(\mathbf{F}_{2^m})$, peut être défini si $b \neq 0$. Le groupe se compose des points de la courbe ainsi que le point à l'infini $\mathcal{O}$. La loi de composition du groupe diffère des deux autres groupes déjà étudiés.

Soit $P = (x_1, y_1) \in E(\mathbf{F}_{2^m})$. L'inverse additif de P est $-P = (x_1, x_1 + y_1)$.

Soient P et Q deux points de la courbe tels que $P \neq Q$. Le point $R = P + Q$ est calculé de la manière suivante :

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1, \\ \text{où} \quad \lambda &= \frac{(y_2 + y_1)}{(x_2 + x_1)}. \end{aligned}$$

On note que les opérations sont effectuées dans le corps $\mathbf{F}_{2^m}$.

Si $P = -Q$, $P + Q = \mathcal{O}$.

Soit $P = (x_1, y_1)$ tel que $x_1 \neq 0$. Le point $P + P = 2P$ peut être démontré ainsi :

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + a \\ y_3 &= x_1^2 + (\lambda + 1)x_3, \\ \text{où} \quad \lambda &= x_1 + \frac{y_1}{x_1}. \end{aligned}$$

Si $x_1 = 0$, $2P = \mathcal{O}$.

Exemple 12. Choisissons le corps $\mathbf{F}_{2^3}$, défini par le polynôme irréductible $x^3 + x + 1$. On a déjà vu (chapitre 2) que $\alpha = x$ est un élément primitif qui engendre tout élément non nul du corps.

$$\begin{array}{lllllll} \alpha & = & x & = & (010) & & \\ \alpha^2 & = & x^2 & = & (100) & & \\ \alpha^3 & = & x^3 & = & x+1 & = & (011) \\ \alpha^4 & = & x(x+1) & = & x^2+x & = & (110) \\ \alpha^5 & = & x(x^2+x) & = & x^2+x+1 & = & (111) \\ \alpha^6 & = & x(x^2+x+1) & = & x^2+1 & = & (101) \\ \alpha^7 & = & x(x^2+1) & = & 1 & = & (001) \end{array}$$

Choisissons la courbe $y^2 + xy = x^3 + \alpha^2 x^2 + \alpha^6$. On note que $a = \alpha^2$ et $b = \alpha^6 \neq 0$. Prenons $P = (\alpha^2, \alpha^6)$ et $Q = (\alpha^5, \alpha^5)$. On montre tout d'abord que P et Q sont de véritables points de la courbe. Pour le point P, on voit que

$$\begin{aligned}
(\alpha^6)^2 + \alpha^2\alpha^6 &= (\alpha^2)^3 + \alpha^2(\alpha^2)^2 + \alpha^6 \\
\alpha^5 + \alpha &= \alpha^6 + \alpha^6 + \alpha^6, \\
(111) + (010) &= (101) + (101) + (101) \\
(101) &= (101)
\end{aligned}$$

et pour le point Q,

$$\begin{aligned}
(\alpha^5)^2 + \alpha^5\alpha^5 &= (\alpha^5)^3 + \alpha^2(\alpha^5)^2 + \alpha^6 \\
\alpha^3 + \alpha^3 &= \alpha + \alpha^5 + \alpha^6, \\
(011) + (011) &= (010) + (111) + (101) \\
(000) &= (000).
\end{aligned}$$

Calculons $R = P + Q$. La formule d'addition donne

$$\begin{aligned}
\lambda &= \frac{y_1 + y_2}{x_1 + x_2} \\
&= \frac{\alpha^6 + \alpha^5}{\alpha^2 + \alpha^5} = \frac{(101) + (111)}{(100) + (111)} = \frac{(010)}{(011)} \\
&= \alpha\alpha^{-3} = \alpha\alpha^4 = \alpha^5 \\
x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\
&= (\alpha^5)^2 + \alpha^5 + \alpha^2 + \alpha^5 + \alpha^2 = \alpha^3 \\
y_3 &= \lambda(x_1 + x_3) + x_3 + y_1 \\
&= \alpha^5(\alpha^2 + \alpha^3) + \alpha^3 + \alpha^6 = \alpha^5(\alpha^5) + \alpha^3 + \alpha^6 \\
&= \alpha^3 + \alpha^3 + \alpha^6 = \alpha^6.
\end{aligned}$$

Alors $P + Q = R = (\alpha^3, \alpha^6)$.

Calculons maintenant $R = 2P$. La formule de doublement donne

$$\begin{aligned}
\lambda &= x_1 + \frac{y_1}{x_1} = \alpha^2 + \frac{\alpha^6}{\alpha^2} = \alpha^2 + \alpha^4 \\
&= (100) + (110) = (010) = \alpha \\
x_3 &= \lambda^2 + \lambda + a \\
&= \alpha^2 + \alpha + \alpha^2 = \alpha \\
y_3 &= x_1^2 + (\lambda + 1)x_3 \\
&= (\alpha^2)^2 + (\alpha + \alpha^0)\alpha = \alpha^4 + (\alpha^3)\alpha = 0.
\end{aligned}$$

Alors $2P = R = (\alpha, 0)$.

3.4 Le théorème de Hasse

Soit E une courbe elliptique définie sur un corps $\mathbf{F}_q$ quelconque. On note $\#E(\mathbf{F}_q)$ le nombre de points de cette courbe.

Le **théorème de Hasse** nous dit que

$$\#E(\mathbf{F}_q) = q + 1 - t, \qquad \text{où } |t| \leq 2\sqrt{q}.$$

On peut conclure qu'une courbe E sur $\mathbf{F}_q$ possède environ q points. Le nombre de points de la courbe est appelé **l'ordre** du groupe.

3.5 Algorithme d'exponentiation

Comme une courbe elliptique est munie d'une loi de composition additive, l'exponentiation dans ce groupe correspond à une application repétée d'addition. On note $\underbrace{P + P + \cdots + P}_{k} = kP$ où $P \in E(\mathbf{F}_q)$. Cependant, le calcul de kP se fait non par l'addition repété du même point, mais par la méthode d'addition et de doublement, une technique qui profite de la représentation binaire de l'exposant k.

Algorithme d'exponentiation (addition-doublement)

Entrée : $k = k_{r-1}k_{r-2}\ldots k_1k_0$, $P = (x, y) \in E(\mathbf{F}_q)$
Sortie : $kP \in E(\mathbf{F}_q)$

1. $Q \leftarrow P$
2. **pour** $i = r - 2$ **jusqu'à** 0 **fais**
   - 2.1. $Q \leftarrow 2Q$
   - 2.2. **si** $k_i = 1$ **alors** $Q \leftarrow Q + P$
3. **fin**(Q)

On verra, dans les chapitres qui suivent, d'autres techniques plus efficaces pour calculer le point kP.

Nous sommes maintenant prêt à définir l'ordre d'un point P.

Définition 27. Un point $P \in E(\mathbf{F}_q)$ est **d'ordre** n si n est le plus petit entier positif tel que $nP = \mathcal{O}$.

Définissons maintenant le logarithme discret sur les courbes elliptiques.

Définition 28. Soient E une courbe elliptique définie sur un corps $\mathbf{F}_q$ quelconque, P un point d'ordre n et Q un point de $E(\mathbf{F}_q)$. Le problème du logarithme discret sur une courbe elliptique est de trouver l'entier l $(0 \leq l \leq n-1)$, si un tel l existe, tel que $Q = lP$.

La difficulté présumée de résoudre ce problème forme la base de certains systèmes cryptographiques, tels les procédés de signature, un sujet qui sera maintenant abordé.

3.6 Signatures numériques

Une signature numérique cherche à imiter une signature manuscrite dans le monde électronique. Toute comme une signature traditionelle, une signature numérique doit s'adhérer à un message en question et il doit être facile pour quiconque de vérifier son authenticité. Dans une société informatisée, l'apposition d'une signature prend la forme d'un algorithme de signature gardé secret par le signataire. La vérification de l'authencité de la signature se fait grâce à un algorithme publique de vérification. En voici la définition :

Définition 29. Soient un ensemble fini M de messages, un ensemble fini T de signatures et un ensemble fini K de clés. Un procédé de signature numérique est un quintuplet (M, T, K, S, V) vérifiant :

1. Pour chaque clé $k \in K$, il existe une fonction de signature $sig_k \in S$ et une fonction de vérification $ver_k \in V$ correspondante.
2. Pour chaque message $m \in M$ et chaque signature $t \in T$, les fonctions $sig_K : M \to T$ et $ver_K : M \times T \to \{\text{vrai}, \text{faux}\}$ vérifient

$$ver_K(m,t) = \begin{cases} \text{vrai} & \text{si } t = sig_K(m) \\ \text{faux} & \text{sinon} \end{cases}$$

On désire aussi que les fonctions de signature et de vérification se calculent de façon efficace. En plus, il doit être infaisable, au point de vue calculatoire, que toute autre personne autre que le signataire soit capable, pour un $m \in M$ quelconque, de trouver un $t \in T$ tel que $sig_K(m,t) = vrai$.

En pratique, on ne signe pas le message m mais une empreinte numérique de m, obtenue à partir d'une fonction de hachage, qui permet de transformer un message de longeur arbitraire en une empreinte numérique de taille fixe.

Passons maintenant au DSA (Digital Signature Algorithm) et son analogue sur les courbes elliptiques, ECDSA (Elliptic Curve Digital Signature Algorithm).

3.7 DSA

Adopté comme standard par NIST (National Institute of Standards and Technology) en 1993, le standard de signature numérique DSA [X9.93] est une modification du procédé de signature d'ElGamal [ElG85]. La sécurité de DSA dépend du problème du logarithme discret dans un sous-groupe de $\mathbb{Z}_p^*$. Voici la description de DSA, divisée en trois étapes.

1. Sélection et génération des paramètres
   - 1.1. Choisis un nombre premier p.
   - 1.2. Choisis un nombre premier q qui divise $p-1$.
   - 1.3. Choisis un élément g d'ordre q. Les éléments du sous-groupe sont : $\{g^0, g^1, g^2, \dots, g^{q-1}\}$.
   - 1.4. Choisis une valeur aléatoire x, où $1 \leq x \leq q-1$.
   - 1.5. Calcule $y = g^x \bmod p$.
   - 1.6. La clé secrète est x et la clé publique est y.
2. Génération de la signature
   - 2.1. Choisis une valeur aléatoire k dans $[1, q-1]$.
   - 2.2. Calcule $g^k \bmod p$.
   - 2.3. Calcule $r = (g^k \bmod p) \bmod q$.
   - 2.4. Calcule $e = h(m)$, où m est le message à signer et h est la fonction de hachage Secure Hash Algorithm (SHA-1). [X9.97]
   - 2.5. Calcule $s = k^{-1}(e + xr) \bmod q$.
   - 2.6. La signature du message m est (r, s).
3. Vérification de la signature
   - 3.1. Calcule $e = h(m))$.
   - 3.2. Calcule $s^{-1} \bmod q$.
   - 3.3. Calcule $u_1 = es^{-1} \bmod q$.
   - 3.4. Calcule $u_2 = rs^{-1} \bmod q$.
   - 3.5. Calcule $v' = g^{u_1} y^{u_2} \bmod p$.
   - 3.6. Calcule $v = v' \bmod q$.
   - 3.7. La signature est acceptée si et seulement si $v = r$.

Le Secure Hash Algorithm [X9.97], publié par NIST en 1993, est une fonction de hachage qui prend comme entrée un message arbitraire de longeur $< 2^{61}$ octets et retourne une valeur de 20 octets ou 160 bits.

Si la signature est bien construite, le procédé de vérification l'authentifie, car

$$\begin{aligned} v &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= (g^{es^{-1} \bmod q} g^{xrs^{-1} \bmod q} \bmod p) \bmod q \\ &= (g^{s^{-1}(e+xr) \bmod q} \bmod p) \bmod q \\ &= (g^{k \bmod q} \bmod p) \bmod q \\ &= r \end{aligned}$$

Exemple 13. Dans l'exemple de DSA qui suit, nous omettrons la fonction de hachage.

Soient $p = 7879$ et $q = 101$. On voit bien que q divise $p - 1$. On choisit $g = 170$, un élément d'ordre 101 dans $\mathbb{Z}_{7879}$. Si on choisit $x = 75$, on obtient

$$y = g^x \bmod 7879 = 170^{75} \bmod 7879 = 4567.$$

Supposons que Bob veut signer le message $e = m = 1234$. Il choisit $k = 50$ comme valeur aléatoire et calcule

$$\begin{aligned} r &= (g^k \bmod p) \bmod q = (170^{50} \bmod 7879) \bmod 101 \\ &= (2518 \bmod 7879) \bmod 101 = 94 \bmod 101. \end{aligned}$$

Il calcule ensuite

$$\begin{aligned} s &= 50^{-1}(1234 + 75 \cdot 94) \bmod 101 \\ &= 99(1234 + 75 \cdot 94) \bmod 101 = 97 \bmod 101. \end{aligned}$$

La signature du message $m = 1234$ devient $(r, s) = (94, 97)$. Ceci se vérifie en calculant

$$\begin{aligned} s^{-1} &= 97^{-1} \bmod 101 = 25 \bmod 101 \\ u_1 &= es^{-1} \bmod q = 1234 \cdot 25 \bmod 101 = 45 \bmod 101 \\ u_2 &= rs^{-1} \bmod q = 94 \cdot 25 \bmod 101 = 27 \bmod 101 \\ v &= (g^{u_1} y^{u_2} \bmod p) \bmod q \\ &= (170^{45} 4567^{27} \bmod 7879) \bmod 101 \\ &= 2518 \bmod 101 = 94. \end{aligned}$$

La signature est acceptée car $v = r = 94$.

3.8 ECDSA

ECDSA, l'analogue de DSA sur les courbes elliptiques, est en cours de normalisation par les commités de standardization ANSI X9F1 [Xx97] et IEEE P1363 [P1398]. Avant de

Notation de DSA	Notation d'ECDSA
q	n
g	P
x	d
y	Q

TAB. 3.1: Correspondance entre la notation de DSA et ECDSA

Groupe	$\mathbb{Z}_p^*$	$E(\mathbf{F}_q)$
éléments du groupe	l'ensemble des entiers $\{1, 2, \dots, p-1\}$	les points (x, y) de $E(\mathbf{F}_p)$ et le point à l'infini $\mathcal{O}$
la loi de composition	multiplication modulo p	l'addition des points
notation	éléments : g, h multiplication : $g \times h$ inverse : g^{-1} exponentiation : g^a	éléments : P, Q addition : $P + Q$ négation : $-P$ multiple : aP
problème du log discret	Soient $g \in \mathbb{Z}_p^*$ et $h = g^a \bmod p$, trouvez a.	Soient $P \in E(\mathbf{F}_q)$ et $Q = aP$, trouvez a.

TAB. 3.2: Correspondance entre $\mathbb{Z}_p^*$ et le groupe $E(\mathbf{F}_q)$

présenter ECDSA au complet , on établit dans les tableaux 3.1 et 3.2 la correspondance de notation entre les deux algorithmes, ainsi que la correspondance entre les éléments et les opérations des groupes $\mathbb{Z}_p^*$ et $E(\mathbf{F}_q)$.

Voici l'algorithme d'ECDSA au complet :

1. Sélection et génération des paramètres
   1.1. Choisis une courbe elliptique définie sur le corps $\mathbf{F}_q$. L'ordre de $E(\mathbf{F}_q)$ doit être divisible par un grand nombre premier n.
   1.2. Choisis un élément P de $E(\mathbf{F}_q)$ d'ordre premier n. Les éléments du sous-groupe sont : $\{P, 2P, 3P, \dots, (n-1)P, nP = \mathcal{O}\}$.
   1.3. Choisis une valeur aléatoire d dans $[1, n-1]$.
   1.4. Calcule $Q = dP$.
   1.5. La clé secrète est d et la clé publique est Q.
2. Génération de la signature
   2.1. Choisis une valeur aléatoire k dans $[1, n-1]$.

2.2. Calcule $kP = (x_1, y_1)$.

2.3. Calcule $r = x_1 \bmod n$.

2.4. Calcule $e = h(m)$, où m est le message à signer et h est la fonction de hachage Secure Hash Algorithm (SHA-1).

2.5. Calcule $s = k^{-1}(e + dr) \bmod n$.

2.6. La signature du message m est (r, s).

3. Vérification de la signature

3.1. Calcule $e = h(m)$.

3.2. Calcule $s^{-1} \bmod n$.

3.3. Calcule $u_1 = es^{-1} \bmod n$.

3.4. Calcule $u_2 = rs^{-1} \bmod n$.

3.5. Calcule $u_1P + u_2Q = (x_2, y_2)$.

3.6. Calcule $v = x_2 \bmod n$.

3.7. La signature est acceptée si et seulement si $v = r$.

Si la signature est bien construite, le procédé de vérification l'authentifie, car

$$\begin{aligned} u_1P + u_2Q &= es^{-1}P + rs^{-1}dP \\ &= s^{-1}(e + rd)P \\ &= kP = (x_1, y_1). \end{aligned}$$

On a alors $r = x_1 \bmod n = v$.

Exemple 14. Comme dans l'exemple de DSA, nous omettrons la fonction de hachage.

Soit $y^2 = x^3 + x + 1$ une courbe elliptique définie sur le corps fini de 23 éléments, $\mathbf{F}_{23}$. On choisit un point de la courbe d'ordre 7, le point $P = (13, 7)$. Le signataire choisit sa clé privée $d = 3$ dans $[1, 6]$. Il calcule sa clé publique $Q = dP = 3(13, 7) = (17, 3)$.

Soit $m = 6$ le message à signer. Le signataire choisit la valeur $k = 4$ dans $[1, 6]$. Il calcule le point $kP = 4(13, 7) = (17, 20)$ et il obtient $r \equiv 17 \bmod 7 \equiv 3$. Il calcule ensuite

$$\begin{aligned} s &= k^{-1}(e + dr) \bmod n \\ &= 4^{-1}(6 + 3 \cdot 3) \bmod 7 \\ &= 2(15) \bmod 7 = 2. \end{aligned}$$

La signature du message $m = 6$ devient $(r, s) = (3, 2)$. Ceci se vérifie en calculant

$$\begin{aligned} s^{-1} &= 2^{-1} \bmod 7 = 4 \\ u_1 &= es^{-1} \bmod n = 6 \cdot 4 \bmod 7 = 3 \\ u_2 &= rs^{-1} \bmod n = 3 \cdot 4 \bmod 7 = 5 \\ u_1P + u_2Q &= 3P + 5Q = 3P + 5 \cdot 3P = 18P = 4P = (17, 20) \\ v &= 17 \bmod 7 = 3. \end{aligned}$$

La signature est acceptée car $v = r = 3$.

On remarque que le procédé de signature n'est pas déterministe. C'est-à-dire, il peut exister plusieurs signatures valides pour un message m quelconque. Même si la fonction de vérification accepte toutes signatures valides, on suppose qu'il soit infaisable de façon calculatoire de trouver une autre signature valide dans un groupe de grand ordre. Dans l'exemple précédent, on voit que la signature $(r, s) = (6, 3)$ est aussi valide pour le message $m = 6$. Pour vérifier cela, on calcule

$$\begin{aligned} s^{-1} &= 3^{-1} \bmod 7 = 5 \\ u_1 &= es^{-1} \bmod n = 6 \cdot 5 \bmod 7 = 2 \\ u_2 &= rs^{-1} \bmod n = 6 \cdot 5 \bmod 7 = 2 \\ u_1P + u_2Q &= 2P + 2 \cdot 3P = 8P = P = (13, 7) \\ v &= 13 \bmod 7 = 6 = r. \end{aligned}$$

Comme indiqué auparavant, on suppose que la probabilité de trouver une autre signature valide dans un groupe de grand ordre est infime.

Chapitre 4

Mise en œuvre d'ECDSA sur $\mathbf{F}_p$

Programmer sur une carte à puce présente un défi particulier. La capacité limitée de mémoire et la vitesse lente du processeur sont les facteurs les plus contraignants dans cet environnement. La puce utilisée pour la mise en œuvre d'ECDSA sur $\mathbf{F}_p$ était le ST16CF54B, fabriquée par Thomson.

Le ST16CF54B possède 512 octets de mémoire vive (RAM), 16 octets de mémoire morte (ROM) et 4 KB de mémoire morte programmable et effaçable électriquement (EEPROM). L'unité centrale mémoire est de 8 bits et l'horloge interne fonctionne à 5 Mhz. La puce possède un cryptoprocesseur nommé MAP (Modular Arithmetic Processor) qui effectue des opérations modulaires avec des données de taille maximale 512 bits.

Une mise en œuvre de la génération d'une signature ECDSA existait déjà sur la carte à puce de Thomson avec un temps d'exécution de $2,5$ secondes. Seulement l'algorithme de génération de signature a été implanté, car on suppose que la vérification de la signature soit faite par une entité autre que la carte à puce. L'implantation existante utilisait le corps $\mathbf{F}_p$, où p était un nombre premier de 160 bits. Le cryptoprocesseur MAP exécutait les opérations de multiplication et d'inversion dans $\mathbf{F}_p$. L'objectif était de réduire le temps d'exécution en améliorant la technique d'exponentiation et en trouvant une représentation optimale pour les coordonnées des points de la courbe.

Dans ce chapitre, nous allons examiner les opérations du corps $\mathbf{F}_p$ et du groupe $E(\mathbf{F}_p)$. Les coordonnées projectives et les coordonnées affines, deux types de coordonnées qui peuvent être utilisées pour représenter les points d'une courbe, sont examinées. Des techniques efficaces d'exponentiation dans le groupe $E(\mathbf{F}_p)$ sont aussi présentées.

4.1 Les opérations du corps $\mathbf{F}_p$

L'opération d'addition dans $\mathbf{F}_p$ se fait sans l'aide du cryptoprocesseur si on suppose que la valeur de chacun des opérandes a et b est moins de p. Dans ce cas, on les additionne et on soustrait p du résultat si nécessaire.

Les opérations de multiplication et d'inversion nécessitent pourtant l'emploi du cryptoprocesseur. La technique de rédution de Montgomery [Mon85, HP98, NM96] est utilisée par le cryptoprocesseur pour effectuer de telles opérations.

4.2 Les opérations du groupe $E(\mathbf{F}_p)$

Nous allons maintenant examiner de plus près les opérations d'addition et de doublement du groupe $E(\mathbf{F}_p)$.

4.2.1 Coordonnées affines

Addition

Soient $P = (x_1, y_1), Q = (x_2, y_2) \in E(\mathbf{F}_p)$.

Si $y_2 = -y_1 \bmod p$, $P + Q = \mathcal{O}$. Sinon $P + Q = R = (x_3, y_3)$, où

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \\ \text{et} \quad \lambda &= \frac{(y_2 - y_1)}{(x_2 - x_1)}. \end{aligned}$$

Dans le corps $\mathbf{F}_p$, l'opération de division est la plus coûteuse. La multiplication, même qu'elle soit coûteuse elle aussi, est beaucoup moins chère que la division. En comparaison avec la multiplication, l'addition et la soustraction sont gratuites. Nous distinguons pas la multiplication du carré d'un nombre, puisque le temps d'exécution de chacun est comparable, tel indiqué par le cryptoprocesseur MAP. Pour effectuer une addition dans $E(\mathbf{F}_p)$, on voit que 3 muliplications et une inversion dans $\mathbf{F}_p$ sont nécessaires.

Doublement

Soit $P = (x_1, y_1)$. Si $y_1 = 0$, $2P = \mathcal{O}$. Sinon $2P = R = (x_3, y_3)$, où

$$
\begin{aligned}
x_3 &= \lambda^2 - 2x_1 \\
y_3 &= \lambda(x_1 - x_3) - y_1, \\
\text{et} \quad \lambda &= \frac{(3x_1^2 + a)}{2y_1}.
\end{aligned}
$$

L'opération de doublement nécessite 4 multiplications et 1 inversion. On ne considère pas le calcul des petits multiples de x et y comme des multiplications, mais plutôt des additions.

L'algorithme d'exponentiation dans le groupe $E(\mathbf{F}_p)$ (voir section 3.5) nécessite plusieurs appels aux fonctions d'addition et de doublement du groupe, chaque appel exigeant un appel à la fonction d'inversion dans $\mathbf{F}_p$. Il est possible d'éviter les inversions dans $\mathbf{F}_p$ en utilisant des coordonnées projectives. Il y a cependant un compromis à faire : on évite les inversions dans $\mathbf{F}_p$ mais on effectue plus de multiplications dans $\mathbf{F}_p$ qu'avec les coordonnées affines.

4.2.2 Coordonnées projectives

Le plan projectif P^2 sur $\mathbf{F}_q$ est défini comme étant l'ensemble des classes d'équivalences de triplets (X, Y, Z) qui satisfaient la relation

$$
\begin{gathered}
(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2) \Longleftrightarrow \\
\exists \lambda \in \mathbf{F}_q, \lambda \neq 0, \text{ tel que } X_1 = \lambda X_2, Y_1 = \lambda Y_2 \text{ et } Z_1 = \lambda Z_2.
\end{gathered}
$$

On appelle chaque classe d'équivalence un **point projectif**. Si $P = (X, Y, Z)$ est un point projectif avec $Z \neq 0$, il n'existe qu'un triplet dans sa classe d'équivalence de la forme $(x, y, 1)$: on pose tout simplement $x = X/Z$ et $y = Y/Z$. Le plan projectif correspond alors aux points (x, y) du plan affine plus les points Z tels que $Z = 0$. L'équation de la courbe dans le plan projectif est obtenue en remplaçant x par X/Z et y par Y/Z.

$$\frac{Y^2}{Z^2} = \frac{X^3}{Z^3} + \frac{aX}{Z} + b$$

En multipliant l'équation par Z^3, on obtient

$$Y^2 Z = X^3 + aXZ^2 + bZ^3.$$

Les points projectifs $(X, Y, Z \neq 0)$ qui vérifient l'équation précédente correspondent aux points affines $(x = X/Z, y = Y/Z)$ qui vérifient l'équation $y^2 = x^3 + ax + b$.

Pour transformer un point affine (x, y) en un point projectif, on pose $X = x, Y = y$ et $Z = 1$. Un point projectif (X, Y, Z) se transforme en point affine en posant $x = X/Z$ et $y = Y/Z$.

Nous allons maintenant transformer les équations d'addition et de doublement du plan affine au plan projectif.

Addition

On souligne d'abord que le point affine $P = (x, y)$ est additionné au point Q dans l'algorithme d'exponentiation (section 3.5) lorsque le i$^{\text{ème}}$ bit de k est 0. Comme la valeur de P est constante, on peut représenter ce point affine en coordonnées projectives comme $P = (X_2, Y_2, Z_2 = 1)$.

Nous allons présenter deux méthodes pour additionner des points de la courbe dans le plan projectif. (Les méthodes se différencient seulement par la façon dont les termes semblables sont regroupés).

Revoyons la formule d'addition dans le plan affine.

Soient $Q = (x_1, y_1), P = (x_2, y_2) \in E(\mathbf{F}_p)$.

Si $y_2 = -y_1 \bmod p,\ P + Q = \mathcal{O}$. Sinon $P + Q = R = (x_3, y_3)$, où

$$\begin{aligned} x_3 &= \lambda^2 - x_1 - x_2 \\ y_3 &= \lambda(x_1 - x_3) - y_1 \\ \text{et} \quad \lambda &= \frac{(y_2 - y_1)}{(x_2 - x_1)}. \end{aligned}$$

On suppose que le point P est exprimé sous forme projectif $P = (X_2 = x_2, Y_2 = y_2, Z_2 = 1)$. Les points affines $Q = (x_1, y_1)$ et $R = (x_3, y_3)$ doivent chacun être transformé en points projectifs. Pour ce faire, on remplace y_1 par Y_1/Z_1 et x_1 par X_1/Z_1 dans λ pour obtenir

$$\Lambda = \left[\frac{Y_2 - Y_1/Z_1}{X_2 - X_1/Z_1}\right] \frac{Z_1}{Z_1} = \frac{Y_2 Z_1 - Y_1}{X_2 Z_1 - X_1}.$$

x_3 devient alors

$$X_3 = \frac{(Y_2 Z_1 - Y_1)^2}{(X_2 Z_1 - X_1)^2} - \frac{X_1}{Z_1} - X_2.$$

Soit $Z_3 = Z_1(X_2Z_1 - X_1)^3$. On multiplie X_3 par Z_3 pour obtenir

$$\begin{aligned} X_3 &= (Y_2Z_1 - Y_1)^2(X_2Z_1 - X_1)Z_1 - X_1(X_2Z_1 - X_1)^3 - X_2Z_1(X_2Z_1 - X_1)^3 \\ &= (Y_2Z_1 - Y_1)^2(X_2Z_1 - X_1)Z_1 - (X_2Z_1 - X_1)^3(X_1 + X_2Z_1). \end{aligned}$$

On procède de la même façon pour trouver Y_3.

$$Y_3 = \left(\frac{Y_2Z_1 - Y_1}{X_2Z_1 - X_1}\right)\left[\frac{X_1}{Z_1} - \frac{(Y_2Z_1 - Y_1)^2}{(X_2Z_1 - X_1)^2} + \frac{X_1}{Z_1} + X_2\right] - \frac{Y_1}{Z_1}.$$

On multiplie Y_3 par Z_3 pour obtenir

$$\begin{aligned} Y_3 &= (Y_2Z_1 - Y_1)(X_2Z_1 - X_1)^2Z_1\left[\frac{2X_1}{Z_1} - \frac{(Y_2Z_1 - Y_1)^2}{(X_2Z_1 - X_1)^2} + X_2\right] - (X_2Z_1 - X_1)^3Y_1 \\ &= (Y_2Z_1 - Y_1)(X_2Z_1 - X_1)^2\,2X_1 - (Y_2Z_1 - Y_1)^3Z_1 \\ &\quad + (Y_2Z_1 - Y_1)(X_2Z_1 - X_1)^2X_2Z_1 - (X_2Z_1 - X_1)^3Y_1. \end{aligned}$$

On veut trouver un moyen de regrouper les mêmes termes dans les équations de X_3, Y_3 et Z_3 afin de minimiser le nombre de multiplications dans $\mathbf{F}_p$.

Soient $U = Y_2Z_1 - Y_1$ et $R = X_2Z_1 - X_1$.

$$\begin{aligned} X_3 &= U^2RZ_1 - R^3(X_1 + X_2Z_1) \\ &= U^2RZ_1 - R^3(X_1 + X_1 - X_1 + X_2Z_1) \\ &= U^2RZ_1 - R^3(2X_1 + R). \end{aligned}$$

Soit $W = -R^3(2X_1 + R)$.

$$X_3 = U^2RZ_1 + W.$$

Y_3 devient, en utilisant U et R,

$$\begin{aligned} Y_3 &= UR^22X_1 - U^3Z_1 + UR^2X_2Z_1 - R^3Y_1 \\ &= -R^3Y_1 - U^3Z_1 + UR^2(2X_1 + X_2Z_1) \\ &= -(R^3Y_1 + U^3Z_1) + UR^2(2X_1 + X_2Z_1 - X_1 + X_1) \\ &= -(R^3Y_1 + U^3Z_1) + UR^2(3X_1 + R) \end{aligned}$$

et Z_3 devient

$$Z_3 = R^3Z_1.$$

Nous avons

$$\begin{cases} X_3 = U^2RZ_1 + W \\ Y_3 = -(R^3Y_1 + U^3Z_1) + UR^2(3X_1 + R) \\ Z_3 = R^3Z_1. \end{cases}$$

Variable	Nombre de multiplications effectuées dans $\mathbf{F}_p$
$U = Y_2Z_1 - Y_1$	1
$R = X_2Z_1 - X_1$	1
$W = -(R(RR))(2X_1 + R)$	3
$X_3 = (UU)RZ_1 + W$	3
$Y_3 = -(R^3Y_1 + UU^2Z_1) + UR^2(3X_1 + R)$	5
$Z_3 = R^3Z_1$	1
	14

TAB. 4.1: Nombre de multiplications effectuées dans $\mathbf{F}_p$ (addition, méthode 1)

Le nombre de multiplications effectuées dans $\mathbf{F}_p$ est calculé dans le tableau 4.1.

Si on suppose qu'on peut mémoriser les valeurs intermédiaires R^2, R^3 et U^2, le calcul de (X_3, Y_3, Z_3) exige 14 multiplications dans $\mathbf{F}_p$ (12 multiplications et 2 mises aux carrés). Les coordonnées affines, à leur tour, n'exigent que 3 multiplications dans $\mathbf{F}_p$, mais elles nécessitent une inversion, une opération fort coûteuse.

La mise en œuvre existante d'ECDSA utilisait des coordonnées projectives au lieu des coordonnées affines, car l'opération d'inversion dans $\mathbf{F}_p$ était trop coûteuse : le MAP effectuait une multiplication en seulement $160\,\mu s$; une inversion en $60\,ms$. Malheureusement, avec seulement 4 registres-tampons disponibles pour mémoriser des valeurs intermédiares, R^2, R^3 et U^2 n'ont pas pu être sauvegardées. Ceci a entraîné deux mises aux carrés et une multiplication supplémentaires. Le nombre d'opérations effectuées est alors 13 multiplications et 4 mises aux carrés, soient 17 multiplications.

Nous présentons une variante de la méthode précédente, qui consiste à regrouper de façon différente les termes semblables. Cette méthode, due à Atsuko Mijaji [Miy93], n'utilise que 12 multiplications.

Reprenons les équations de X_3, Y_3 et Z_3.

$$\left\{\begin{array}{lcl} X_3 & = & (Y_2Z_1 - Y_1)^2(X_2Z_1 - X_1)Z_1 - (X_2Z_1 - X_1)^3(X_1 + X_2Z_1) \\ Y_3 & = & (Y_2Z_1 - Y_1)(X_2Z_1 - X_1)^2\,2X_1 - (Y_2Z_1 - Y_1)^3Z_1 \\ & & +(Y_2Z_1 - Y_1)(X_2Z_1 - X_1)^2X_2Z_1 - (X_2Z_1 - X_1)^3Y_1 \\ Z_3 & = & Z_1(X_2Z_1 - X_1)^3 \end{array}\right.$$

Soient $U = Y_2Z_1 - Y_1$ et $R = X_2Z_1 - X_1$.

$$\begin{aligned} X_3 &= U^2RZ_1 - R^3(X_1 + X_2Z_1) \\ &= R\left[U^2Z_1 - R^2(X_1 + X_2Z_1)\right]. \end{aligned}$$

Soient $T = X_1 + X_2Z_1$ et $A = U^2Z_1 - R^2T$. On a

$$X_3 = RA.$$

Avec U et R, Y_3 devient

$$\begin{aligned} Y_3 &= UR^2 2X_1 - U^3Z_1 + UR^2X_2Z_1 - R^3Y_1 \\ &= UR^2(X_1 + X_1 + X_2Z_1) - R^3Y_1 - U^3Z_1 \\ &= U\left[R^2X_1 + R^2(X_1 + X_2Z_1) - U^2Z_1\right] - R^3Y_1 \\ &= U\left[R^2X_1 - (U^2Z_1 - R^2T)\right] - R^3Y_1 \\ &= U(R^2X_1 - A) - R^3Y_1 \end{aligned}$$

et Z_3 devient

$$Z_3 = R^3Z_1.$$

En resumé, nous avons

$$\begin{cases} X_3 &= RA \\ Y_3 &= U(R^2X_1 - A) - R^3Y_1 \\ Z_3 &= R^3Z_1. \end{cases}$$

Le nombre de multiplications effectuées dans $\mathbf{F}_p$ est calculé dans le tableau 4.2.

Variable	Nombre de multiplications effectuées dans $\mathbf{F}_p$
$U = Y_2Z_1 - Y_1$	1
$R = X_2Z_1 - X_1$	1
$T = X_1 + X_2Z_1$	0
$A = U^2Z_1 - (RR)T$	4
$X_3 = RA$	1
$Y_3 = U(R^2X_1 - A) - (RR^2)Y_1$	4
$Z_3 = R^3Z_1$	1
	12

TAB. 4.2: Nombre de multiplications effectuées dans $\mathbf{F}_p$ (addition, méthode 2)

Si nous pouvons stocker les valeurs intermédiaires X_2Z_1, R^2 et R^3, seulement 12 multiplications dans $\mathbf{F}_p$ sont nécessaires pour le calcul de (X_3, Y_3, Z_3) (10 multiplications et

2 mises aux carrés). Cependant, même si cette technique permet moins de multiplications, elle est irréalisable avec le petit nombre de registres-tampons à notre disposition. De plus, la vitesse de transfert entre les registres du cryptoprocesseur MAP, où sont effectuées les multiplications et les mises aux carrés dans $\mathbf{F}_p$, et les registres du processeur est péniblement lente : le temps de transfert entre les processeurs est plus grand que le temps d'exécution d'une simple multiplication dans $\mathbf{F}_p$! En conséquence, un nombre réduit de multiplications ne garanti aucunement une performance élevée. En fait, une mise en œuvre de la technique de Atsuko Mijaji avec 13 multiplications n'a relevé qu'une légère amélioration de quelques millisecondes, par rapport à une implantation avec 14 multiplications.

Ce qui peut influencer le temps d'exécution cependant est l'ordre d'exécution des opérations. Si on cherche à minimiser le nombre de transferts entre les processeurs, on pourrait peut-être obtenir une amélioration du temps d'exécution. Malheureusement, quatre mises en œuvres différentes de la méthode d'Atsuko Mijaji ont toutes donné presque le même temps d'exécution. En essayant de réduire le nombre de transferts, le nombre de multiplications augmente... En resumé, les deux méthodes présentées pour calculer la somme de deux points de la courbe sont presque équivalentes au niveau du temps d'exécution.

Doublement

Nous allons maintenant considérer l'opération de doublement. Examinons les équations de doublement dans le plan affine.

Soit $Q = (x_1, y_1)$. Si $y_1 = 0,\ 2Q = \mathcal{O}$. Sinon

$$\begin{aligned} x_3 &= \lambda^2 - 2x_1 \\ y_3 &= \lambda(x_1 - x_3) - y_1, \\ \text{et} \quad \lambda &= \frac{(3x_1^2 + a)}{2y_1}. \end{aligned}$$

Comme dans le cas de l'addition, les points affines $Q = (x_1, y_1)$ et $R = (x_3, y_3)$ doivent chacun être transformé en points projectifs. Pour ce faire, on remplace y_1 par Y_1/Z_1 et x_1 par X_1/Z_1 dans λ pour obtenir

$$\Lambda = \left[\frac{3\,(X_1/Z_1)^2 + a}{2(Y_1/Z_1)}\right] \frac{Z_1^2}{Z_1^2} = \frac{3X_1^2 + aZ_1^2}{2Y_1Z_1}.$$

x_3 devient alors

$$X_3 = \left(\frac{3X_1^2 + aZ_1^2}{2Y_1Z_1}\right)^2 - \frac{2X_1}{Z_1}.$$

Soit $Z_3 = (2Y_1Z_1)^3Z_1$. On multiplie X_3 par Z_3 pour obtenir

$$\begin{aligned}
X_3 &= (2Y_1Z_1)^3Z_1\left[\frac{(3X_1^2+aZ_1^2)^2}{(2Y_1Z_1)^2} - 2\frac{X_1}{Z_1}\right] \\
&= 2Y_1Z_1^2(3X_1^2+aZ_1^2)^2 - 2X_1(2Y_1Z_1)^3 \\
&= Z_1\left[2Y_1Z_1(3X_1^2+aZ_1^2)^2 - 2X_1(2Y_1)^3Z_1^2\right].
\end{aligned}$$

On procède de la même façon pour trouver Y_3.

$$Y_3 = \left(\frac{3X_1^2+aZ_1^2}{2Y_1Z_1}\right)\left[\frac{X_1}{Z_1} - \frac{(3X_1^2+aZ_1^2)^2}{(2Y_1Z_1)^2} + 2\frac{X_1}{Z_1}\right] - \frac{Y_1}{Z_1}.$$

On multiplie Y_3 par Z_3 pour obtenir

$$\begin{aligned}
Y_3 &= (2Y_1Z_1)^3Z_1\left[\frac{3X_1^2+aZ_1^2}{2Y_1Z_1}\left(\frac{X_1}{Z_1} - \frac{(3X_1^2+aZ_1^2)^2}{(2Y_1Z_1)^2} + 2\frac{X_1}{Z_1}\right) - \frac{Y_1}{Z_1}\right] \\
&= (2Y_1Z_1)^2Z_1(3X_1^2+aZ_1^2)\left[3\frac{X_1}{Z_1} - \frac{(3X_1^2+aZ_1^2)^2}{(2Y_1Z_1)^2}\right] - (2Y_1Z_1)^3Y_1 \\
&= 3X_1\left(2Y_1Z_1\right)^2\left(3X_1^2+aZ_1^2\right) - \left(3X_1^2+aZ_1^2\right)^3Z_1 - \left(2Y_1Z_1\right)^3Y_1 \\
&= Z_1\left[3X_1\left(2Y_1\right)^2Z_1\left(3X_1^2+aZ_1^2\right) - \left(3X_1^2+aZ_1^2\right)^3 - \left(2Y_1\right)^3Z_1^2Y_1\right].
\end{aligned}$$

Comme avant, on veut trouver un moyen de regrouper les mêmes termes dans les équations de X_3, Y_3 et Z_3 afin de minimiser le nombre de multiplications dans $\mathbf{F}_p$.

On remarque que la variable Z_1 apparaît en évidence dans chacun des équations de X_3, Y_3 et Z_3. On peut alors l'omettre dans chacun des équations, puisqu'on obtient un point dans la même classe d'équivalence.

Soit $U = 3X_1^2 + aZ_1^2$.

$$\begin{aligned}
X_3 &= 2Y_1Z_1U^2 - 2X_18Y_1^3Z_1^2 \\
&= 2Y_1Z_1(U^2 - 8X_1Y_1^2Z_1) \\
&= 2Y_1Z_1\left[U^2 - 4X_1Y_1(2Y_1Z_1)\right].
\end{aligned}$$

Soient $V = 2Y_1Z_1$, $T = Y_1V$, $W = -2X_1T$ et $R = U^2 + 2W$. On a

$$\begin{aligned}
X_3 &= V(U^2 - 4X_1Y_1V) \\
&= V(U^2 - 4X_1T) \\
&= V\left[U^2 + 2(-2X_1T)\right] \\
&= V(U^2 + 2W) \\
&= VR
\end{aligned}$$

et Y_3 devient

$$\begin{aligned}
Y_3 &= 12X_1Y_1^2Z_1U - U^3 - 8Y_1^4Z_1^2 \\
&= -U(U^2 - 12X_1Y_1^2Z_1) - 8Y_1^4Z_1^2 \\
&= -U\left[U^2 - 6X_1Y_1(2Y_1Z_1)\right] - 2(2Y_1^2Z_1)^2 \\
&= -U\left[U^2 - 6X_1(Y_1V)\right] - 2\left(Y_1(2Y_1Z_1)\right)^2 \\
&= -U\left[U^2 + 3(-2X_1T)\right] - 2(Y_1V)^2 \\
&= -U(U^2 + 3W) - 2T^2 \\
&= -U(R + W) - 2T^2.
\end{aligned}$$

et Z_3 devient

$$Z_3 = (2Y_1Z_1)^3 = V^3.$$

En resumé, nous avons

$$\begin{cases} X_3 &= VR \\ Y_3 &= -U(R+W) - 2T^2 \\ Z_3 &= V^3. \end{cases}$$

Le nombre de multiplications effectuées dans $\mathbf{F}_p$ est calculé dans le tableau 4.3.

Variable	Nombre de multiplications effectuées dans $\mathbf{F}_p$
$U = 3X_1^2 + aZ_1^2$	3
$V = 2Y_1Z_1$	1
$T = Y_1V$	1
$W = -2X_1T$	1
$R = U^2 + 2W$	1
$X_3 = RV$	1
$Y_3 = -U(R+W) - 2T^2$	2
$Z_3 = VV^2$	2
	12

TAB. 4.3: Nombre de multiplications effectuées dans $\mathbf{F}_p$ (doublement, méthode 1)

Nous avons un total de 12 multiplications. Si on pose $a = 1$, on obtient 11 multiplications. Avec les coordonnées affines, seulement 4 multiplications sont nécessaires mais une inversion est aussi effectuée. La mise en œuvre existante d'ECDSA utilisait la technique précédente pour doubler un point de la courbe. La valeur de a a été fixée à 1, afin de profiter du nombre réduit de multiplications. Un choix astucieux quant à l'ordre des opérations effectuées a permis de garder le nombre de multiplications à 11. Malheureusement, aucun autre changement dans l'ordre des opérations n'a pu être effectué afin de réduire d'avantage le nombre de transferts entre les processeurs. Ceci est attribuable non seulement au petit nombre de registres-tampons disponibles, mais aussi au nombre important de calculs de petits multiples (p.ex. $3X_1^2, 2Y_1Z_1, -2X_1T, 2W, 2T^2$) qui nécessitent l'usage des registres-tampons.

Une autre méthode de regroupement due à Atsuko Mijaji [Miy93] nécessite aussi 12 multiplications. Par contre, cette méthode exige qu'un grand nombre de valeurs soit mémorisées. De plus, un grand nombre de petits multiples doivent être calculés. Puisque cette méthode n'offre aucun avantage par rapport à la précédente, elle ne sera pas présentée.

D'après les méthodes exposées pour effectuer une addition et un doublement, on voit qu'on ne peut réduire le temps d'exécution actuel de ces opérations que par quelques millisecondes.

4.3 Exponentiation

Même si on ne peut pas diminuer de façon significative le temps d'exécution d'un doublement ou d'une addition de deux points, on peut néanmoins diminuer le temps d'exécution d'une signature d'ECDSA en réduisant le temps pour calculer le multiple d'un point. Nous allons maintenant examiner des algorithmes efficaces d'exponentiation. La mise en œuvre originelle d'ECDSA sur la carte Thomson utilisait la technique simple d'addition-doublement.

Algorithme d'exponentiation (addition-doublement)

Entrée : $k = k_{r-1}k_{r-2}\dots k_1k_0$, $P = (x, y) \in E(\mathbf{F}_p)$
Sortie : $kP \in E(\mathbf{F}_q)$

1. $Q \leftarrow P$
2. **pour** $i = r-2$ **jusqu'à** 0 **fais**
   - 2.1. $Q \leftarrow 2Q$
   - 2.2. **si** $k_i = 1$ **alors** $Q \leftarrow Q + P$
3. **fin**(Q)

Cette méthode nécessite $r - 1$ doublements et $\omega(k) - 1$ additions, où $\omega(k)$ représente le nombre de 1 dans la représentation binaire de k. En moyenne, on fait $r/2-1$ additions.

Au lieu de scruter les bits de k de gauche à droite, on pourrait scruter plusieurs bits de k simultanément. Cette technique permettrait de réduire le nombre d'additions et de doublements. Cependant, cette méthode n'est pas sans aucun inconvénient : il faut quand même mémoriser quelques points projectifs. Nous décrivons l'algorithme, nommé la ≪méthode des rateaux≫ [CHJI98, GN98], en plus grand détail.

4.3.1 La technique des rateaux

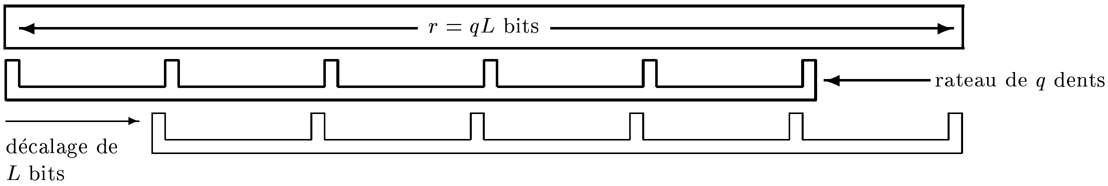

FIG. 4.1: La technique des rateaux

Soit k un entier de r bits. Au lieu de scruter k bits, un à la fois de gauche à droite, on scrute q bits simultanément. On peut imaginer un rateau de q dents, séparées par r/q espaces. On scrute k en glissant le rateau de gauche à droite $L = r/q$ fois.

Algorithme de balayage avec rateau simple

Entrée : $k = k_{r-1}k_{r-2}\ldots k_1k_0$, $P = (x,y) \in E(\mathbf{F}_p)$, $q =$ le nombre de dents, $L = r/q$
Sortie : $kP \in E(\mathbf{F}_p)$

1. **pour** $i = 0$ **jusqu'à** $q-1$ **fais**
   1.1. $p_i = 2^{iL}P$
2. $Q \leftarrow 2^{(q-1)L}P$
3. **pour** $i = 0$ **jusqu'à** $q-2$ **fais**
   3.1. $Q \leftarrow Q + k_{iL+L-1}p_i$
4. **pour** $j = L-2$ **jusqu'à** 0 **fais**
   4.1. $Q \leftarrow 2Q$
   4.2. **pour** $i = 0$ **jusqu'à** $q-1$ **fais**
      4.2.1. $Q \leftarrow Q + k_{iL+j}p_i$
5. **fin**(Q)

Cet algorithme nécessite le calcul de q points p_i qui doivent être sauvegardés. En moyenne, $r/q-1$ doublements et $(r-1)/2$ additions sont effectués.

Exemple 15. Soit k un entier de 4 bits et posons $q = 2$. On a $r = 4$ et $L = r/q = 2$.

$$\begin{array}{rcl} p_0 & \leftarrow & P \\ p_1 & \leftarrow & 2^2 P \\ Q & \leftarrow & 2^2 P \\ Q & \leftarrow & 2^2 P + k_1 P \\ Q & \leftarrow & 2Q = (2^3 + 2k_1)P \\ Q & \leftarrow & Q + k_0 P + k_2 2^2 P = (k_0 + 2k_1 + 2^2 k_2 + 2^3)P \end{array}$$

$r/q - 1 = 1$ doublements et $(r-1)/2 = 3/2$ en moyenne ont été effectués.

4.3.2 La technique de bits condensés

On peut réduire le nombre d'additions en augmentant le nombre de valeurs précalculées. On remplace les q valeurs scrutées par les q dents du rateaux par une seule valeur. Il y a $2^q - 1$ différentes combinaisons de q bits (occultant la combinaison de q zéros).

Soit $[\lambda] = [\lambda_{q-1}\lambda_{q-2} \dots \lambda_1 \lambda_0]$ une chaîne de q bits et soit $A[\lambda] = \sum_{i=0}^{q-1} \lambda_i p_i$, avec $0 < \lambda < 2^q$. Chaque $A[\lambda]$ représente la somme des q bits balayés par le rateau de q dents.

Algorithme de balayage avec bits condensés

Entrée : $k = k_{r-1}k_{r-2} \dots k_1 k_0$, $P = (x, y) \in E(\mathbf{F}_p)$, $q =$ le nombre de dents, $L = r/q$
Sortie : $kP \in E(\mathbf{F}_p)$

1. **pour** $i = 0$ **jusqu'à** $q - 1$ **fais**
   1.1. $p_i = 2^{iL} P$
2. $[\lambda] \leftarrow [k_{(q-2)L+L-1} k_{(q-3)L+L-1} \dots k_{L+L-1} k_{L-1}]$
3. $Q \leftarrow A[\lambda]$
4. **pour** $j = L - 2$ **jusqu'à** 0 **fais**
   4.1. $Q \leftarrow 2Q$
   4.2. $[\lambda] = [k_{(q-1)L+j} k_{(q-2)L+j} \dots k_{L+j} k_j]$
   4.3. $Q \leftarrow Q + A[\lambda]$
5. **fin**(Q)

On effectue encore $L - 1$ doublements, mais grâce aux $2^q - 1$ valeurs précalculées, le nombre d'additions décroît à $(L-1)(1 - \frac{1}{2^q})$. On note que la probabilité d'avoir q zéros est $1/2^q$, ce qui correspond à aucune addition.

Exemple 16. Soient k un entier de 4 bits et $q = 2$. On a $r = 4$ et $L = r/q = 2$.

$$\begin{array}{lcl}
[\lambda] & \leftarrow & [k_3 k_1] \\
Q & \leftarrow & Q = (k_1 + 2^2 k_3)P \\
Q & \leftarrow & 2Q = (2k_1 + 2^3 k_3)P \\
[\lambda] & \leftarrow & [k_2 k_0] \\
Q & \leftarrow & Q + A[k_2 k_0] = (2k_1 + 2^3 k_3)P + (k_0 + 2^2 k_2)P \\
 & = & (k_0 + 2k_1 + 2^2 k_2 + 2^3 k_3)P
\end{array}$$

On souligne que même si on peut réduire le nombre d'additions de façon significative, un nombre exponentiel en q de précalculs doivent être sauvegardés. De plus, chaque point précalculé correspond à une paire d'éléments (X, Y) $(Z = 1)$. Si on suppose que les éléments du corps fini sont de taille t bits, $2t(2^q - 1)$ bits sont nécessaires pour représenter les $2^q - 1$ précalculs.

Une autre technique existe néanmoins qui consiste en un compromis entre le nombre de doublements, d'additions et de précalculs. La technique de rateaux imbriqués utilise n rateaux, chacun muni de m dents. Cette méthode permet de réduire le nombre de doublements à $r/(mn) - 1$ et le nombre d'additions en moyenne à $(\frac{r}{m} - 1)(1 - \frac{1}{2^m})$ avec seulement $n(2^m - 1)$ précalculs, ce qui est exponentiel en m au lieu de q.

4.3.3 La technique de rateaux imbriqués

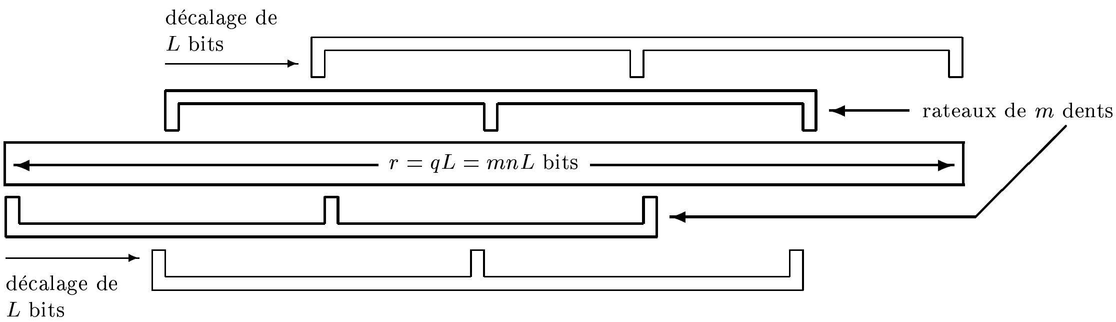

FIG. 4.2: La technique de rateaux imbriqués ($n = 2$ rateaux, $m = 3$ dents)

Soit $\lambda = [\lambda_{m-1}\lambda_{m-2} \ldots \lambda_1\lambda_0]$ une chaîne de m bits qui représente les m bits balayés par un rateau de m dents.

Soit $A_i[\lambda] = \sum_{h=0}^{m-1} \lambda_h 2^{iL} 2^{hnL} P$, pour $0 \leq i \leq n - 1$. Comme il y a $2^m - 1$ chaînes de bits λ (moins la chaîne de zéros) et n rateaux, nous avons $n(2^m - 1)$ valeurs à calculer.

Algorithme de balayage avec rateaux imbriqués

Entrée : $k = k_{r-1}k_{r-2}\dots k_1k_0$, $P = (x, y) \in E(\mathbf{F}_p)$, n rateaux de m dents chacun, $L = r/(mn)$
Sortie : $kP \in E(\mathbf{F}_p)$

1. **pour** $i = 0$ **jusqu'à** $n-1$ **fais**
   1.1. **pour** $\lambda = 1$ **jusqu'à** $2^m - 1$ **fais**
      1.1.1. $A_i[\lambda] = A_i[\lambda_{m-1}\lambda_{m-2}\dots\lambda_1\lambda_0] = \sum_{h=0}^{m-1} \lambda_h 2^{iL}2^{hnL}P$
2. $Q \leftarrow \sum_{i=0}^{n-1} A_i[k_{(m-1)nL+iL+L-1}k_{(m-2)nL+iL+L-1}\dots k_{iL+L-1}]$
3. **pour** $j = L-2$ **jusqu'à** 0 **fais**
   3.1. $Q \leftarrow 2Q$
   3.2. $Q \leftarrow Q + \sum_{i=0}^{n-1} A_i[k_{(m-1)nL+iL+j}k_{(m-2)nL+iL+j}\dots k_{iL+j}]$
4. **fin**(Q)

$L-1$ doublements et $(\frac{r}{m}-1)(1-\frac{1}{2^m})$ additions en moyenne sont effectués. La valeur $(1-\frac{1}{2^m})$ représente la probabilité avec laquelle la chaîne de m bits balayés par un rateau n'est pas la chaîne de m zéros.

Exemple 17. Soit k un entier de 8 bits. Supposons qu'on a $n = 2$ rateaux de $m = 2$ dents chacun. On a $r = 8$ et $L = \frac{r}{mn} = 2$.

$$\begin{array}{lcl}
A_0[01] & \leftarrow & 1P + 0 \cdot 2^4 P = P \\
A_0[10] & \leftarrow & 0P + 1 \cdot 2^4 P = 2^4 P \\
A_0[11] & \leftarrow & 1P + 1 \cdot 2^4 P = (1 + 2^4)P \\
A_1[01] & \leftarrow & 1 \cdot 2^2 P + 0 \cdot 2^6 P = 2^2 P \\
A_1[10] & \leftarrow & 0 \cdot 2^2 P + 1 \cdot 2^6 P = 2^6 P \\
A_1[11] & \leftarrow & 1 \cdot 2^2 P + 1 \cdot 2^6 P = (2^2 + 2^6)P \\
Q & \leftarrow & A_0[k_5k_1] + A_1[k_7k_3] = (k_1 + k_5 2^4)P + (2^2k_3 + 2^6k_7)P \\
Q & \leftarrow & 2Q = (2k_1 + 2^5k_5)P + (2^3k_3 + 2^7k_7)P \\
Q & \leftarrow & Q + A_0[k_4k_0] + A_1[k_6k_2] = Q + (k_0 + 2^4k_4)P + (2^2k_2 + 2^6k_6)P \\
 & = & (k_0 + 2k_1 + 2^2k_2 + 2^3k_3 + 2^4k_4 + 2^5k_5 + 2^6k_6 + 2^7k_7)P
\end{array}$$

Le calcul de kP a nécessité $n(2^m - 1) = 2(2^2 - 1) = 6$ précalculs, $L - 1 = 1$ doublements et environ $(\frac{r}{m} - 1)(1 - \frac{1}{2^m}) = (\frac{8}{2} - 1)(1 - \frac{1}{4}) = 9/4 \approx 2$ additions.

L'algorithme d'exponentiation consomme la plus grande partie du temps d'exécution du calcul de la signature. La version originelle de la signature, qui possédait un temps d'exécution de $2,5$ secondes, utilisait la technique simple d'addition et doublement.

Avec une clé k de 160 bits, 159 doublements et environ 80 additions ont été effectués. Avec un rateau simple muni de deux dents, le nombre de doublements descend à 79 (le nombre d'additions demeure environ 80). Il faut, cependant, mémoriser 2 points projectifs, le point P et le point $2^{80}P$. Au lieu de sauvegarder $2^{80}P = (X, Y, Z)$, on ne sauvegarde que X/Z et Y/Z $(Z = 1)$. Les 2 points projectifs occupent alors $2 \cdot 2 \cdot 160 = 640$ bits $= 80$ octets d'espace. Ces valeurs précalculées sont conservées dans le programme principal.

Le tableau 4.4 résume la complexité des différentes techniques d'exponentiation.

Technique	Nombre de rateaux	Nombre de dents	Nombre de doublements	Nombre d'additions	Nombre de précalculs
addition-doublement	1	1	$r-1$	$(r-1)/2$	0
rateau simple	1	q	$r/q-1$	$(r-1)/2$	q
rateau avec bits condensés	1	q	$r/q-1$	$(r/q-1)(1-\frac{1}{2^q})$	2^q-1
rateaux imbriqués	n	m	$r/(mn)-1$	$(r/m-1)(1-\frac{1}{2^m})$	$n(2^m-1)$

TAB. 4.4: Techniques d'exponentiation

4.4 Résultats

La technique de rateau simple avec bits condensés a été implantée, ainsi que la technique de rateaux imbriqués. Avec un seul rateau muni de 2 dents, 79 doublements et 59 additions en moyenne ont été effectués, avec un temps d'exécution de $1,7$ secondes. 3 points projectifs ont été mémorisés, ce qui représente $3 \cdot 2 \cdot 160$ bits $= 960$ bits $= 120$ octets d'espace dans le programme principal. Un rateau muni de 3 dents a permis de réduire de nouveau le temps d'exécution à $1,3$ secondes. 52 doublements et 45 additions ont été effectués et 7 points projectifs ont été sauvegardés, ce qui correspond à $7 \cdot 2 \cdot 160$ bits $= 2240$ bits $= 280$ octets. L'utilisation d'un rateau de 4 dents était irréalisable, car ceci nécessitait $15 \cdot 2 \cdot 160$ bits $= 4800$ bits $= 600$ octets d'espace.

Une amélioration supplémentaire du temps d'exécution a été possible grâce à la technique de rateaux imbriqués. Avec 3 rateaux, chacun muni de 2 dents, seulement 26 doublements et 59 additions ont été nécessaires, qui a réduit le temps d'exécution à $1,1$ secondes. Cette technique a nécessité la mémorisation de 9 points projectifs, soient

$9 \cdot 2 \cdot 160$ bits $= 2880$ bits ou 360 octets d'espace dans le programme principal. Afin de franchir la barrière d'une seconde, 2 rateaux de 3 dents chacun ont été utilisés, ce qui a pu réduire le temps d'exécution à $0,9$ secondes. 26 doublements et 47 additions ont été effectués, mais 14 points projectifs ont été sauvegardés, ce qui correspond à $14 \cdot 2 \cdot 160$ bits $= 4480$ bits, soient 560 octets d'espace dans le programme principal. Avec cette dernière technique, on a pu réduire le temps d'exécution de la génération de la signature ECDSA de 2,5 à 0,9 secondes, une amélioration de 64 %.

Chapitre 5

Mise en œuvre d'ECDSA sur $\mathbf{F}_{2^m}$

La carte à puce utilisée pour la mise en œuvre d'ECDSA sur le corps $\mathbf{F}_{2^m}$ était le SLE66CX160S, fabriquée par Siemens. Cette puce possède 1280 octets de mémoire vive, 32 KB de mémoire morte et 16 KB de mémoire morte programmable et effaçable électriquement. L'unité centrale mémoire est de 8 bits. La puce possède également un cryptoprocesseur, nommé ACE (Advanced Crypto Engine), qui effectue des opérations modulaires sur des données de taille maximale 1100 bits.

Aucune mise en œuvre sur $\mathbf{F}_{2^m}$ existait sur la composante de Siemens. À part du choix du corps, de nombreux autres choix se présentaient : comment choisit-on la représentation des éléments du corps et les coordonnées des points de la courbe ? Quelle technique d'exponentiation utilise-t-on ? Et encore, comment choisit-on les paramètres de la courbe ?

Comme dans le cas de $\mathbf{F}_p$, seulement l'algorithme de génération de signature a été implanté. On suppose que la vérification de la signature soit faite par une entité autre que la carte à puce.

Comme mentionné plus haut, la carte à puce SLE66CX160S de Siemens ne possédait aucune implantation d'ECDSA. Il fallait tout d'abord choisir une représentation des éléments du corps $\mathbf{F}_{2^m}$. Une base polynomiale, une base polynomiale composée et une base normale optimale sont les trois représentations qui seront examinées dans ce chapitre. Comme dans le cas de $\mathbf{F}_p$, des coordonnées projectives peuvent être utilisées afin d'éliminer les opérations d'inversion dans $\mathbf{F}_{2^m}$. Nous examinerons deux différents types de coordonnées projectives.

Comme dans le cas de $\mathbf{F}_p$, des techniques efficaces d'exponentiation s'avèrent très utiles dans le calcul de kP. Puisque ce thème a déjà été abordé dans le chapitre précédent, il ne sera pas présenté de nouveau. Il suffit tout simplement de noter que les mêmes techniques sont applicables.

5.1 Base polynomiale

Une représentation polynomiale des éléments du corps fini $\mathbf{F}_{2^m}$ a déjà été examinée dans le chapitre 3. Nous faisons un résumé de cette méthode.

Soit $f(x) = x^m + a_{m-1}x^{m-1} + \cdots + a_1x + a_0$ un polynôme de degré m sur $\mathbf{F}_2$. Les éléments du corps $\mathbf{F}_{2^m}$ peuvent être représentés comme l'ensemble des 2^m polynômes de degré au plus $m-1$. Un élément de $\mathbf{F}_{2^m}$ prend la forme $b_{m-1}x^{m-1} + b_{m-2}x^{m-2} + \cdots + b_1x + b_0$ où $b_i \in \mathbf{F}_2$ et $0 \leq i \leq m-1$. La base consiste en l'ensemble des m éléments : $\{x^{m-1}, x^{m-2}, \ldots, x^2, x^1, x^0\}$. En forme compacte, un élément de $\mathbf{F}_{2^m}$ devient une chaîne de m bits : $(b_{m-1}b_{m-2}\ldots b_1b_0)$. Avec cette représentation, l'élément neutre multiplicative est $(00\ldots01)$ et l'élément neutre additif est $(00\ldots00)$.

Le choix du polynôme irréductible peut avoir un grand impact sur les opérations de réduction. Au lieu de choisir un polynôme irréductible aléatoire, on utilise un trinôme irréductible de la forme $x^m + x^k + 1$ avec $1 \leq k \leq m-1$ ou un polynôme irréductible de 5 termes de la forme $x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$ avec $1 \leq k_3, k_2, k_1 \leq m-1$. Ces polynômes sont préferables aux polynômes aléatoires car un petit nombre de OU-exclusif sont effectués dans l'algorithme de réduction. Un polynôme irréductible aléatoire exigerait beaucoup plus de ces opérations. Malheureusement, les trinômes irréductibles n'existent que pour certaines valeurs de m. Néanmoins, des polynômes irréductibles de 5 termes existent pour toutes valeurs de m.

5.1.1 Les opérations du corps $\mathbf{F}_{2^m}$

Examinons maintenant les opérations dans le corps $\mathbf{F}_{2^m}$. On utilise la représentation binaire d'un polynôme pour les opérations d'addition, multiplication, mise au carré et réduction.

Addition

L'addition n'est que le OU-exclusif de la représentation binaire des opérandes.

Algorithme d'addition

Entrée : $a, b \in \mathbf{F}_{2^m}$
Sortie : $r = a \oplus b \in \mathbf{F}_{2^m}$

1. **pour** $i = 0$ **jusqu'à** $m - 1$ **fais**
   1.1. $r_i = a_i \oplus b_i$
2. **fin**(r)

Multiplication

L'opération de multiplication consiste en multipliant les représentations polynomiales des opérandes, suivie d'une réduction modulo le polynôme irréductible. (voir la section Réduction)

Algorithme de multiplication

Entrée : $a, b \in \mathbf{F}_{2^m}$
Sortie : $r = a \cdot b \in \mathbf{F}_{2^{2m-1}}$

1. $r \leftarrow \underbrace{(00\ldots00)}_{2m-1}$
2. **pour** $i = 0$ **jusqu'à** $m - 1$ **fais**
   2.1. **pour** $j = 0$ **jusqu'à** $m - 1$ **fais**
      2.1.1. $r_{i+j} = r_{i+j} \oplus (a_i \cdot b_j)$
3. **fin**(r)

On note que les opérations $\cdot$ et $\oplus$ sont des opérations binaires.

Mise au carré

On note que la mise au carré d'une somme est la somme des carrés lorsque le corps est de caractéristique 2. C'est-à-dire, $\left(\sum_{i=0}^{m-1} \alpha_i x^i\right)^2 = \sum_{i=0}^{m-1} \alpha_i^2 x^{2i}$. Cette opération produit un polynôme de degré au plus $2(m-1)$. La mise au carré s'effectue en intercalant le bit 0 entre chaque bit de la représentation binaire du polynôme.

Algorithme de mise au carré

Entrée : $a \in \mathbf{F}_{2^m}$
Sortie : $b = a^2 \in \mathbf{F}_{2^{2m-1}}$

1. $b \leftarrow \underbrace{(00\ldots00)}_{2m-1}$
2. **pour** $i = 0$ **jusqu'à** $m-1$ **fais**
   2.1. $b_{2i} = a_i$
3. **fin**(b)

Réduction

Comme les opérations de multiplication et de mise au carré produisent des polynômes de degré au plus $2(m-1)$, il faut les réduire à des polynômes de degré au plus $m-1$. On suppose qu'il existe un trinôme de degré m irréductible sur $\mathbf{F}_2$. Sinon, on choisit un polynôme de 5 termes de degré m irréductible sur $\mathbf{F}_2$.

Algorithme de réduction

Entrée : $a \in \mathbf{F}_{2^{2m-1}}$, trinôme $f = x^m + x^t + 1$ irréductible sur $\mathbf{F}_2$
Sortie : $a \bmod f \in \mathbf{F}_{2^m}$

1. **pour** $i = 2(m-1)$ **jusqu'à** $m+1$ **en descendant fais**
   1.1. $a_{i-m} = a_{i-m} \oplus a_i$
   1.2. $a_{i-m+t} = a_{i-m+t} \oplus a_i$
   1.3. $a_i = 0$
2. **fin**(a)

Comme le trinôme f ne possède que 3 termes non nuls, on n'effectue que deux OU-exclusif à chaque itération. On remarque que le degré de a est réduit de 1 à chaque tour de boucle.

Division

Il existe plusieurs algorithmes pour calculer l'inverse d'un élément de $\mathbf{F}_{2^m}$. L'algorithme étendu d'Euclide, le «Almost inverse algorithm» de R. Schroeppel et al. [SOOS85] et la technique de Montgomery [KAK96] résolvent chacun le problème de division dans $\mathbf{F}_{2^m}$. Nous ne présentons que la méthode d'Euclide.

Algorithme de division

Entrée : $g(x) \in \mathbf{F}_{2^m}$ tel que $g(x) \neq$ polynôme nul, polynôme irréductible $f(x)$ sur $\mathbf{F}_2$ de degré m
Sortie : polynôme $s(x) \in \mathbf{F}_{2^m}$ tel que $s(x)g(x) = 1$

1. $s_2(x) \leftarrow 1, s_1(x) \leftarrow 0$
2. **tant que** $f(x) \neq$ polynôme nul **fais**
   - 2.1. $q(x) \leftarrow g(x)$ div $f(x)$
   - 2.2. $r(x) \leftarrow g(x) - f(x)q(x)$
   - 2.3. $s(x) \leftarrow s_2(x) - q(x)s_1(x)$
   - 2.4. $g(x) \leftarrow f(x)$
   - 2.5. $f(x) \leftarrow r(x)$
   - 2.6. $s_2(x) \leftarrow s_1(x)$
   - 2.7. $s_1(x) \leftarrow s(x)$
3. $s(x) \leftarrow s_2(x)$
4. **fin**$(s(x))$

Une représentation polynomiale des éléments de $\mathbf{F}_{2^m}$ est caractérisée par un grand nombre d'opérations binaires. Les algorithmes de multiplication et de réduction nécessitent des opérations OU-exclusif sur des bits, ce qui n'est pas optimal avec un processeur de 8 bits.

Nous présentons maintenant une variation de la forme standard de la représentation polynomiale, où les éléments sont représentés par des polynômes sur $\mathbf{F}_{2^8}$ au lieu de $\mathbf{F}_2$. On appelle cette représentation base polynomiale composée [HMV92, WBV$^+$96, GP97].

5.2 Base polynomiale composée

Soit $m = 8 \cdot r$, avec r un nombre impair. Au lieu de représenter le corps $\mathbf{F}_{2^m}$ comme un espace vectoriel sur $\mathbf{F}_2$ de dimension m, on choisit un espace vectoriel sur le corps $\mathbf{F}_{2^8}$ de dimension $r = m/8$. Chaque élément de $\mathbf{F}_{2^{8 \cdot r}}$ peut être représenté par un polynôme $\alpha_{r-1}x^{r-1} + \cdots + \alpha_1 x + \alpha_0$ avec $\alpha_i \in \mathbf{F}_{2^8}$. L'avantage de cette méthode est que chaque coéfficient est représenté par 8 bits et les opérations OU-exclusif se font sur des mots et non des bits.

On a besoin de définir les opérations mathématiques dans le sous-corps $\mathbf{F}_{2^8}$ ainsi que dans le sur-corps $\mathbf{F}_{2^{8 \cdot r}}$. Commençons par le sous-corps $\mathbf{F}_{2^8}$.

5.2.1 Les opérations du sous-corps $\mathbf{F}_{2^8}$

Comme il n'y a que $2^8 = 256$ éléments dans le corps $\mathbf{F}_{2^8}$, les opérations d'addition, de multiplication et de division se font aisement à l'aide des tableaux de logarithmes et antilogarithmes. Pour ce faire, on choisit tout d'abord un polynôme irréductible sur $\mathbf{F}_{2^8}$. On choisit ensuite un générateur γ et on calcule les paires (α, i) tel que $\alpha = \gamma^i$ où $\alpha \in \mathbf{F}_{2^8} \setminus \{0\}$ et $0 \leq i < 2^8 - 1$. A partir de ces paires, on crée 2 tableaux : un tableau de logarithmes trié sur α et un tableau de antilogarithme trié sur i. Ces tableaux nécessitent un espace total de $2 \cdot 2^8 \cdot 8 = 512$ bits. Regardons maintenant les opérations de $\mathbf{F}_{2^8}$ en détail.

Addition

Cette opération n'est que le OU-exclusif des deux opérandes.

Algorithme d'addition

Entrée : $x, y \in \mathbf{F}_{2^8}$
Sortie : $z = x + y \in \mathbf{F}_{2^8}$

1. $z = x \oplus y$
2. $\mathbf{fin}(z)$

Multiplication

Le produit de deux opérandes s'effectue à l'aide des tableaux de logarithmes et antilogarithmes, ainsi qu'avec une réduction modulo 255. Trois consultations de tables sont aussi effectuées.

Algorithme de multiplication

Entrée : $x, y \in \mathbf{F}_{2^8}$
Sortie : $z = x \cdot y \in \mathbf{F}_{2^8}$

1. $z = \log^{-1}[(\log x + \log y) \bmod 255]$
2. **fin**(z)

On souligne que l'opération de réduction modulo 255 ne nécessite pas l'usage d'un cryptoprocesseur. On inspecte la retenue du résultat de $\log x + \log y$ pour déterminer si une réduction aurait lieu.

Inverse

Cette opération ne demande que deux consultations de tables et une réduction modulo 255.

Algorithme d'inversion

Entrée : $x \in \mathbf{F}_{2^8}$
Sortie : $z = x^{-1} \in \mathbf{F}_{2^8}$

1. $z = \log^{-1}[-\log x \bmod 255]$
2. **fin**(z)

Passons maintenant aux opérations dans le sur-corps $\mathbf{F}_{2^{8 \cdot r}}$.

5.2.2 Les opérations du sur-corps $\mathbf{F}_{2^{8 \cdot r}}$

Si on veut représenter les éléments du corps $\mathbf{F}_{2^{8 \cdot r}}$ par un polynôme de degré moins de r avec des coéfficients dans $\mathbf{F}_{2^8}$, il nous faut un polynôme irréductible de degré r. On peut choisir un polynôme sur $\mathbf{F}_{2^8}$, mais on verra que ce choix entraîne de nombreuses consultations de table lors de l'opération de réduction. Il serait préférable de choisir un polynôme irréductible sur $\mathbf{F}_2$. Pour ce faire, on utilise le fait qu'un polynôme irréductible de degré r sur $\mathbf{F}_2$ est irréductible sur $\mathbf{F}_{2^8}$ si le $\text{pgcd}(r, 2^8) = 1$. [LN94] Le choix de r est donc limité aux nombres impairs.

Les opérations du sur-corps $\mathbf{F}_{2^{8\cdot r}}$ utilisent les opérations du sous-corps $\mathbf{F}_{2^8}$. Elles sont tirées de [WBV$^+$96]. Regardons-les de plus près.

Addition

L'addition dans le sur-corps comprend, comme dans le cas du sous-corps, le OU-exclusif des opérandes.

Algorithme d'addition

Entrée : $x, y \in \mathbf{F}_{2^{8\cdot r}}$
Sortie : $z = x + y \in \mathbf{F}_{2^{8\cdot r}}$

1. **pour** $i = 0$ **jusqu'à** $r - 1$ **fais**
   1.1. $z_i = x_i \oplus y_i$
2. **fin**(z)

Multiplication

Le produit de deux polynômes se calcule grâce à la méthode de «décalage et addition» où l'addition est remplacée par un OU-exclusif. On souligne que la sortie z contient $2r - 1$ octets. L'opération de réduction permet de réduire le polynôme à r octets.

Algorithme de multiplication

Entrée : $x, y \in \mathbf{F}_{2^{8\cdot r}}$
Sortie : $z = x \cdot y \in \mathbf{F}_{2^{8(2r-1)}}$

1. $z \leftarrow \underbrace{(00\ldots00)}_{8(2r-1)}$
2. **pour** $i = 0$ **jusqu'à** $r - 1$ **fais**

2.1. **si** $x_i \neq (00000000)$ **alors**

2.1.2. **pour** $j = 0$ **jusqu'à** $r - 1$ **fais**

2.1.2.1. **si** $y_j \neq (00000000)$ **alors**

$$z_{i+j} = z_{i+j} \oplus \log^{-1}[(\log x_i + \log y_j) \bmod 255]$$

3. **fin**(z)

Puisque le log de zéro n'est pas défini, il faut vérifier qu'aucun des octets de x et y n'est zéro. On note que le calcul de z_{i+j} utilise l'opération de multiplication dans $\mathbf{F}_{2^8}$. Aucune opération ne manipule des bits, seulement des octets.

Mise au carré

Le carré d'un polynôme peut être calculé de façon plus efficace qu'en le multipliant avec lui-même. On sait déjà que dans un corps de caractéristique 2, le carré d'une somme est égal à la somme des carrés. On obtient un polynôme de degré $2(r-1)$ qu'on réduit ensuite à un polynôme de degré $r-1$ grâce à l'algorithme de réduction.

Algorithme mise au carré

Entrée : $x \in \mathbf{F}_{2^{8 \cdot r}}$
Sortie : $z = x^2 \in \mathbf{F}_{2^{8(2r-1)}}$

1. $z \leftarrow \underbrace{(00\ldots00)}_{8(2r-1)}$
2. **pour** $i = 0$ **jusqu'à** $r - 1$ **fais**

2.1 **si** $x_i \neq (00000000)$ **alors**

$$z_{2i} = \log^{-1}[(2 \log x_i) \bmod 255]$$

3. **fin**(z)

On voit que cet algorithme est beaucoup plus efficace que celle de multiplication.

Réduction

Soit $A = a_{2r-2}x^{2r-2} + a_{2r-3}x^{2r-3} + \cdots + a_1 x + a_0$ un polynôme à réduire et $B = x^m + b_t x^t + b_0$ un trinôme irréductible avec $b_t, b_0 \in \mathbf{F}_{2^8}$. Pour annuler le coéfficient a_{2r-2}, on multiplie le polynôme B par $a_{2r-2}x^{r-2}$ et on additionne le résultat avec A. Le coéfficient a_{t+r-2} devient $a_{t+r-2} \oplus a_{2r-2} \cdot b_t$ et le coéfficient a_{r-2} devient $a_{r-2} \oplus a_{2r-2} \cdot b_0$. Ces deux calculs nécessitent chacun une multiplication dans $\mathbf{F}_{2^8}$ et un OU-exclusif des octets. On peut éviter la multiplication dans $\mathbf{F}_{2^8}$ en choissisant un trinôme irréductible de degré r avec des coéfficients dans $\mathbf{F}_2$. Comme indiqué tantôt, un trinôme irréductible de degré r est irréductible sur $\mathbf{F}_{2^8}$ si le pgcd$(r, 2^8) = 1$. Ceci limite le choix de r aux nombres impairs. L'algorithme de réduction est le suivant :

Algorithme de réduction

Entrée : $A \in \mathbf{F}_{2^{2(r-1)}}$, trinôme $F = x^r + x^t + 1$ irréductible sur $\mathbf{F}_{2^8}$
Sortie : $A \bmod F \in \mathbf{F}_{2^{8 \cdot r}}$

1. **pour** $i = 2(r-1)$ **jusqu'à** $r+1$ **en descendant fais**
   - 2.1. $a_{i-r} = a_{i-r} \oplus a_i$
   - 2.2. $a_{i-r+t} = a_{i-r+t} \oplus a_i$
   - 2.3. $a_i = 0$
2. **fin**(a)

Passons maintenant à la dernière opération sur $\mathbf{F}_{2^{8 \cdot r}}$, celle de l'inverse.

Division

Pour calculer l'inverse d'un élément de $\mathbf{F}_{2^{8 \cdot r}}$, on utilise encore une fois l'algorithme d'Euclide.

Algorithme de division

Entrée : $A \in \mathbf{F}_{2^{8 \cdot r}}$, M = polynôme irréductible sur $\mathbf{F}_{2^8}$ de degré r
Sortie : $B = A^{-1} \bmod F \in \mathbf{F}_{2^{8 \cdot r}}$
Soit deg(p) le degré d'un polynôme p.

1. initialise les polynômes :
   - 1.1. $B \leftarrow 1, C \leftarrow 0, F \leftarrow A, G \leftarrow M$
2. **tant que** $\deg(F) \neq 0$ **fais**
   - 2.1. **si** $\deg(F) < \deg(G)$ **échange** F et G, **échange** B et C
   - 2.2. $j = \deg(F) - \deg(G)$
   - 2.3. α = coéfficient dominant de F/coéfficient dominant de G

2.4. $F = F + \alpha x^j G$

2.5. $B = B + \alpha x^j C$

3. **fin**(B/terme constant de F)

$A^{-1} = B \bmod M$ si et seulement si il existe un X tel que $BA + XM = 1$. L'algorithme étendu d'Euclide préserve, à chaque itération, les deux relations $F = BA + XM$ et $G = CA + YM$ (il n'est pas nécessaire de sauvegarder X et Y).

Passons maintenant à la dernière méthode de représentation des éléments de $\mathbf{F}_{2^m}$, nommée base normale optimale.

5.3 Base normale optimale

Une base normale [ABMV93, ABV89, MOVW89, AMOV91] de $\mathbf{F}_{2^m}$ sur $\mathbf{F}_2$ est une base de la forme $B = \{\beta, \beta^2, \beta^{2^2}, \ldots, \beta^{2^{m-1}}\}$, pour un β dans $\mathbf{F}_{2^m}$. Tout $\alpha \in \mathbf{F}_{2^m}$ peut être représenté de la façon suivante :

$$\alpha = \sum_{i=0}^{m-1} \alpha_i \beta^{2^i}, \text{avec } a_i \in \mathbf{F}_2.$$

Selon l'usage, on note $\alpha = (\alpha_0\alpha_1 \ldots \alpha_{m-2}\alpha_{m-1})$. L'élément neutre multiplicatif est représenté par la chaîne $(11\ldots11)$, l'élément neutre additif par la chaîne $(00\ldots00)$.

Une base normale existe pour chaque corps fini [LN94]. Un des avantages de cette représentation est que la mise au carré d'un élément équivaut à un décalage circulaire de sa représentation binaire.

Poursuivons avec les opérations dans le corps $\mathbf{F}_{2^m}$.

5.3.1 Les opérations du corps $\mathbf{F}_{2^m}$

Addition

L'addition avec une base normale se fait de la même façon qu'avec une base polynomiale.

Algorithme d'addition

Entrée : $x, y \in \mathbf{F}_{2^m}$
Sortie : $z = x + y \in \mathbf{F}_{2^m}$

1. **pour** $i = 0$ à $m - 1$ **fais**
   1.1. $z_i = x_i \oplus y_i$
2. **fin**(z)

Mise au carré

On sait déjà que dans un corps de caractéristique 2, le carré d'une somme est égal à la somme des carrés. Soit

$$\begin{aligned}
\alpha &= \sum_{i=0}^{m-1} a_i \beta^{2^i} \\
\alpha^2 &= \sum_{i=0}^{m-1} a_i^2 \beta^{2^{i+1}} = \sum_{i=0}^{m-2} a_i \beta^{2^{i+1}} + a_{m-1}\beta^{2^m} \\
&= \sum_{i=0}^{m-2} a_i \beta^{2^{i+1}} + a_{m-1}\beta \quad \text{puisque } \beta^{2^m} = \beta \\
&= \sum_{i=0}^{m-1} a_{i-1} \beta^{2^i} \quad \text{avec les indices réduits modulo } m.
\end{aligned}$$

L'algorithme de mise au carré n'est qu'un simple décalage circulaire à droite d'un bit.

Algorithme mise au carré

Entrée : $\alpha \in \mathbf{F}_{2^m}$
Sortie : $z = \alpha^2 \in \mathbf{F}_{2^m}$
Soit $\alpha = (\alpha_0\alpha_1\alpha_2 \ldots \alpha_{m-2}\alpha_{m-1})$

1. $z = \alpha^2 = (\alpha_{m-1}\alpha_0\alpha_1\alpha_2 \ldots \alpha_{m-2})$
2. **fin**(z)

Multiplication

Le calcul du produit de deux éléments est plus complexe qu'avec une représentation polynomiale, car il nécessite un certain nombre de précalculs.

Soient $a = (a_0 a_1 a_2 \ldots a_{m-1})$ et $b = (b_0 b_1 b_2 \ldots b_{m-1}) \in \mathbf{F}_{2^m}$. Avec une base normale, le produit $c = ab = (c_0 c_1 \ldots c_{m-1})$ peut être exprimé de la façon suivante :

$$c = \sum_{k=0}^{m-1} c_k \beta^{2^k} = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \beta^{2^i} \beta^{2^j}. \tag{5.1}$$

Puisque B est une base pour $\mathbf{F}_{2^m}$,

$$\beta^{2^i} \beta^{2^j} = \sum_{k=0}^{m-1} \lambda_{i,j}^{(k)} \beta^{2^k}, \quad \text{avec } \lambda_{i,j}^{(k)} \in \mathbf{F}_2. \tag{5.2}$$

En remplaçant $\beta^{2^i} \beta^{2^j}$ de l'équation 5.1 avec l'équation 5.2, on obtient

$$c = \sum_{k=0}^{m-1} c_k \beta^{2^k} = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \sum_{k=0}^{m-1} \lambda_{i,j}^{(k)} \beta^{2^k}. \tag{5.3}$$

En comparant les coéfficients de β^{2^k} de l'équation 5.3, on a

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \lambda_{i,j}^{(k)}, \quad \text{pour } 0 \leq k \leq m-1. \tag{5.4}$$

On élève ensuite chaque côté de l'équation 5.2 à la puissance 2^{-l}. L'expression de gauche devient

$$\beta^{2^{i-l}} \beta^{2^{j-l}} = \sum_{k=0}^{m-1} \lambda_{i-l,j-l}^{(k)} \beta^{2^k} \tag{5.5}$$

et l'expression de droite devient

$$\sum_{k=0}^{m-1} \lambda_{i,j}^{(k)} \beta^{2^{k-l}}. \tag{5.6}$$

Si on égalise les coéfficients de β^{2^0} dans les expressions 5.5 et 5.6, on obtient

$$\lambda_{i-l,j-l}^{(0)} = \lambda_{i,j}^{(l)}, \quad \text{pour } 0 \leq i, j, l \leq m-1. \tag{5.7}$$

On peut maintenant réécrire c_k de l'équation 5.4

$$c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_i b_j \lambda_{i-k,j-k}^{(0)} = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{i+k} b_{j+k} \lambda_{i,j}^{(0)}. \tag{5.8}$$

On souligne que $c_0 = \sum_{i=0}^{m-1}\sum_{j=0}^{m-1} a_i b_j \lambda_{i,j}^{(0)}$ et que le calcul de c_1 se fait en ajoutant 1 modulo m aux indices inférieurs de a et b, ce qui équivaut à un décalage circulaire à gauche des vecteurs a et b.

Le nombre de termes non nuls $\lambda_{i,j}^{(0)}$ est au plus m^2. Une base normale est dite optimale si ce nombre est égal à la borne inférieure de $2m-1$. [MOVW89]. Pour une valeur de m quelconque, il peut exister soit un ou deux types de bases normales optimales ou aucune base normale optimale. Consultez l'annexe E du document X9.62-199x [Xx97] pour une liste des corps $\mathbf{F}_{2^m}$ qui possèdent des bases normales optimales.

On présente maintenant une méthode pour déterminer les valeurs de $\lambda_{i,j}^{(0)}$, qu'on notera dorénavant par $\lambda_{i,j}$. Pour ce faire, on suppose tout d'abord qu'une base normale optimale existe pour $\mathbf{F}_{2^m}$. On désigne type I et type II les deux genres de bases normales optimales [Xx97]. Les formules de multiplication sont légèrement différentes dans chacun des cas.

Désignons l'élément $a \in \mathbf{F}_{2^m}$ par la chaîne de m bits $(a_0 a_1 \dots a_{m-2} a_{m-1})$ et le polynôme $g(x) = g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + \cdots + g_1 x + g_0$ par la chaîne de m bits $(g_{m-1} g_{m-2} \dots g_1 g_0)$. Voici les étapes à suivre pour calculer les valeurs $\lambda_{i,j}$.

1. Si $\mathbf{F}_{2^m}$ ne possède qu'une base normale optimale de type I, on pose alors $f(x) = x^m + x^{m-1} + \cdots + x^2 + x + 1$. Sinon, une base de type II existe et on calcule $f(x) = f_m(x)$ de manière récursive :

   1.1. $f_0(x) = 1$

   1.2. $f_1(x) = x + 1$

   1.3. $f_{i+1}(x) = x f_i(x) + f_{i-1}(x), \quad i \geq 1.$

   On réduit les coéfficients de $f_i(x)$ modulo 2 à chaque étape et on obtient un polynôme $f(x)$ de degré m sur $\mathbf{F}_2$.
   L'ensemble des polynômes $\left\{x, x^2 \bmod f(x), x^{2^2} \bmod f(x), \dots, x^{2^{m-1}} \bmod f(x)\right\}$ forment une base normale pour $\mathbf{F}_{2^m}$.

2. On construit une $m \times m$ matrice A, telle que la i$^{\text{ème}}$ ligne correspond à la representation binaire du polynôme $x^{2^i} \bmod f(x)$.

3. La matrice A^{-1} est ensuite calculée.

4. On construit une $m \times m$ matrice T, telle que la i$^{\text{ème}}$ ligne correspond à la représentation binaire du produit vA^{-1}, où v est la représentation binaire du polynôme $x \cdot x^{2^i} \bmod f(x)$.

5. On pose $\lambda_{i,j} = T(j-i, -i)$, $(0 \leq i, j \leq m-1)$ avec les indices réduits modulo m.

On note que $\lambda_{0,j} = 1$ pour exactement une valeur de j $(0 \leq j \leq m-1)$ et que pour chaque i $(1 \leq i \leq m-1)$, $\lambda_{i,j} = 1$ pour exactement deux valeurs distinctes de

j, $(0 \leq j \leq m-1)$ [MOVW89]. Des m^2 coéfficients de T, il n'y a que $2m-1$ qui sont non nuls. Comme on a atteint la borne inférieure, on appelle cette base optimale. De toute les bases normales, les bases optimales utilisent le plus petit nombre d'opérations ET et OU-exclusif pour calculer un produit dans $\mathbf{F}_{2^m}$.

On souligne qu'on ne sauvegarde que les valeurs de $\lambda_{i,j}$ pour effectuer les opérations de multiplications dans $\mathbf{F}_{2^m}$. Résumons ainsi l'algorithme de multiplication.

Algorithme de multiplication

Entrée : $a, b \in \mathbf{F}_{2^m}$
Sortie : $c = a \cdot b \in \mathbf{F}_{2^m}$

1. **pour** $k = 0$ **jusqu'à** $m-1$ **fais**
   1.1. $c_k = \sum_{i=0}^{m-1} \sum_{j=0}^{m-1} a_{i+k} \cdot b_{j+k} \cdot \lambda_{i,j}$
2. **fin**(c)

On note que les indices sont réduits modulo m.

Passons maintenant à la dernière opération dans $\mathbf{F}_{2^m}$: la division.

Division

Nombre de méthodes peuvent être utilisées afin de calculer l'inverse d'un élément dans $\mathbf{F}_{2^m}$. On pourrait se servir, une fois de plus, de l'algorithme d'Euclide, mais ceci entraîne des changements de bases : la transformation d'une base normale à une base polynomiale, suivie d'une transformation à la base normale après l'application de l'algorithme d'Euclide. Une des techniques les plus efficaces est celle d'Itoh et al. [IT88, ABMV93] qui tire profit de l'opération peu coûteuse de mise au carré dans $\mathbf{F}_{2^m}$. La méthode nécessite $m-1$ mises aux carrés et exactement $\lfloor \log_2(m-1) \rfloor + \omega(m-1) - 1$ multiplications dans $\mathbf{F}_{2^m}$, où $\omega(m-1)$ désigne le nombre de 1 dans la représentation binaire de $m-1$. Malheureusement, cette technique exige la mémorisation de quelques valeurs précalculées.

L'algoritme d'Itoh

La méthode d'Itoh est fondée sur les observations suivantes :

Soit $\alpha \in \mathbf{F}_{2^m}, \alpha \neq 0$. Comme $\alpha = \alpha^{2^m}$,

$$\alpha^{-1} = \alpha^{2^m-2} = \alpha^{2(2^{m-1}-1)} = \left(\alpha^{2^{m-1}-1}\right)^2.$$

Si m est impair,

$$2^{m-1} - 1 = \left(2^{(m-1)/2} - 1\right)\left(2^{(m-1)/2} + 1\right)$$

et on obtient

$$\begin{aligned}
\alpha^{2^{m-1}-1} &= \alpha^{\left(2^{(m-1)/2}-1\right)\left(2^{(m-1)/2}+1\right)} \\
&= \left(\alpha^{2^{(m-1)/2}-1}\right)^{2^{(m-1)/2}+1} \\
&= \left(\alpha^{2^{(m-1)/2}-1}\right)^{2^{(m-1)/2}}\left(\alpha^{2^{(m-1)/2}-1}\right).
\end{aligned}$$

Si on ne tient pas compte des mises aux carrés, seulement une multiplication est nécessaire afin de calculer $\alpha^{2^{m-1}-1}$, une fois la valeur $\alpha^{2^{(m-1)/2}-1}$ est calculée.

Si m est pair,

$$2^{m-1} - 1 = 2\left(2^{(m-2)/2} - 1\right)\left(2^{(m-2)/2} + 1\right) + 1$$

et on obtient

$$\begin{aligned}
\alpha^{2^{m-1}-1} &= \alpha^{2\left(2^{(m-2)/2}-1\right)\left(2^{(m-2)/2}+1\right)+1} \\
&= \left(\alpha^{2^{(m-2)/2}-1}\right)^{2^{m/2}+2}\alpha \\
&= \left(\alpha^{2^{(m-2)/2}-1}\right)^{2^{m/2}}\left(\alpha^{2^{(m-2)/2}-1}\right)^2\alpha.
\end{aligned}$$

Si on ne tient pas compte des mises aux carrés, seulement deux multiplications sont nécessaires afin de calculer $\alpha^{2^{m-1}-1}$, une fois la valeur $\alpha^{2^{(m-2)/2}-1}$ est calculée. On applique de façon récursive cette méthode afin d'obtenir α^{-1}.

Exemple 18. Soit $\alpha \in \mathbf{F}_{2^{155}}$. On montre que seulement $7 + 4 - 1 = 10$ multiplications et 154 mises aux carrés sont effectuées dans le calcul de α^{-1}.

$$\begin{aligned}
2^{155} - 2 &= 2(2^{154} - 1) = 2(2^{77} - 1)(2^{77} + 1) \\
2^{77} - 1 &= 2(2^{38} - 1)(2^{38} + 1) + 1 \\
2^{38} - 1 &= (2^{19} - 1)(2^{19} + 1) \\
2^{19} - 1 &= 2(2^9 - 1)(2^9 + 1) + 1 \\
2^9 - 1 &= 2(2^4 - 1)(2^4 + 1) + 1 \\
2^4 - 1 &= (2^2 - 1)(2^2 + 1) \\
2^2 - 1 &= (2 + 1)
\end{aligned}$$

Nous avons

$$\begin{array}{lcll}
\alpha^{2^2-1} & = & \alpha^{2+1} & \\
& = & \alpha^2 \cdot \alpha & \text{(1 multiplication, 1 mise au carré)} \\
& & & \\
\alpha^{2^4-1} & = & \alpha^{(2^2-1)(2^2+1)} & \\
& = & \left(\alpha^{2^2-1}\right)^{2^2} \cdot \left(\alpha^{2^2-1}\right) & \text{(1 multiplication, 2 mises aux carrés)} \\
& & & \\
\alpha^{2^9-1} & = & \alpha^{2(2^4-1)(2^4+1)+1} & \\
& = & \left(\alpha^{2^4-1}\right)^{2^5} \cdot \left(\alpha^{2^4-1}\right)^2 \cdot \alpha & \text{(2 multiplications, 5 mises aux carrés)} \\
& & & \\
\alpha^{2^{19}-1} & = & \alpha^{2(2^9-1)(2^9+1)+1} & \\
& = & \left(\alpha^{2^9-1}\right)^{2^{10}} \cdot \left(\alpha^{2^9-1}\right)^2 \cdot \alpha & \text{(2 multiplications, 10 mises aux carrés)} \\
& & & \\
\alpha^{2^{38}-1} & = & \alpha^{(2^{19}-1)(2^{19}+1)} & \\
& = & \left(\alpha^{2^{19}-1}\right)^{2^{19}} \cdot \left(\alpha^{2^{19}-1}\right) & \text{(1 multiplication, 19 mises aux carrés)} \\
& & & \\
\alpha^{2^{77}-1} & = & \alpha^{2(2^{38}-1)(2^{38}+1)+1} & \\
& = & \left(\alpha^{2^{38}-1}\right)^{2^{39}} \cdot \left(\alpha^{2^{38}-1}\right)^2 \cdot \alpha & \text{(2 multiplications, 39 mises aux carrés)} \\
& & & \\
\alpha^{2(2^{154}-1)} & = & \alpha^{2(2^{77}-1)(2^{77}+1)} & \\
& = & \left(\alpha^{2^{77}-1}\right)^{2^{78}} \cdot \left(\alpha^{2^{77}-1}\right)^2 & \text{(1 multiplication, 78 mises aux carrés)}
\end{array}$$

On effectue 10 multiplications et 154 mises aux carrés. On voit aussi qu'il est nécessaire de conserver le résultat d'une étape pour calculer le résultat de la prochaine étape.

5.4 Résumé des représentations des éléments de $\mathbf{F}_{2^m}$

Nous comparons dans le tableau 5.1 les opérations de multiplication, mise au carré, réduction et division des différentes méthodes de représentations des éléments dans $\mathbf{F}_{2^m}$. On note que l'addition se fait de la même façon, peu importe la représentation choisie.

Représentation	Opération	Technique	Avantages	Désavantages
base polynomiale	multiplication	multiplication de 2 polynômes modulo polynôme irréductible		opérations sur des bits, réduction modulaire
	mise au carré	intercaler des bits 0 dans représentation binaire	plus efficace que la multiplication	opérations sur des bits, réduction modulaire
	réduction modulaire	OU-exclusif des bits		opérations sur des bits
	inverse	algorithme d'Euclide		multiplication utilise opérations sur des bits
base polynomiale composée	multiplication	consultations des tableaux du sous-corps $\mathbf{F}_{2^8}$	consultations des tableaux, opérations sur des octets	réduction modulaire
	mise au carré	intercaler des octets de 0 dans représentation binaire	plus efficace que multiplication	réduction modulaire
	réduction modulaire	OU-exclusif des octets	opérations sur des octets et non des bits	
	inverse	algorithme d'Euclide	opérations sur des octets	
base normale optimale	multiplication	opérations sur des bits calcul de $\lambda_{i,j}$	décalage des opérandes pour calcul	opérations ET et OU-exclusif sur des bits, sauvegarder $\lambda_{i,j}$
	mise au carré	décalage circulaire à la droite	très efficace	
	inverse	algorithme d'Itoh	mise au carré peut coûteuse, petit nombre de multiplications	

TAB. 5.1: Comparaison des représentations des éléments de $\mathbf{F}_{2^m}$

Il est clair que la façon dont on représente les éléments de $\mathbf{F}_{2^m}$ peut influencer le temps d'exécution de la signature numérique. Il est aussi évident qu'on doit tenir compte des proprietés du langage d'assembleur afin d'effectuer le meilleur choix de représentation. Si on compare les deux représentations polynomiales, on voit qu'une base polynomiale composée est nettement supérieure à une base polynomiale simple, pour la simple raison que la première effectue des opérations sur des octets et non des bits. Un processeur de 8 bits éffectue des opérations sur des mots de façon plus efficace que sur des bits. De plus, des tableaux de logarithmes et antilogarithmes sur $\mathbf{F}_{2^8}$ ont été sauvegardés en mémoire, afin d'augmenter la rapidité avec laquelle on effectue des opérations dans $\mathbf{F}_{2^{8 \cdot r}}$. Même si une représentation normale optimale s'avère efficace au point de vue de l'opération de mise au carré, de nombreuses opérations binaires sont néanmoins effectuées dans l'opération de multiplication. Le meilleur choix de représentation est alors la base polynomiale composée.

Il a été signalé dans la section 5.2 que la valeur de m doit être divisible par r, un nombre impair. La valeur de m choisie pour l'implantation était $184 = 8 \cdot 23$. Les éléments de $\mathbf{F}_{2^{8 \cdot r}}$ étaient des polynômes de degré au plus 22, avec des coéfficients de $\mathbf{F}_{2^8}$. Un tableau de 23 octets formait la représentation interne de chaque élément du corps. L'octet de poids faible représentait le terme constant du polynôme. Puisque le degré du polynôme n'était pas sauvegardé explicitement dans un octet supplémentaire, il était calculé en vérifiant l'octet de poids le plus fort. Les opérations dans les corps $\mathbf{F}_{2^8}$ et $\mathbf{F}_{2^{8 \cdot r}}$ ont été implantées suivant les sections 5.2.1 et 5.2.2. Le polynôme irréductible choisi pour $\mathbf{F}_{2^8}$ était $x^8 + x^5 + x^3 + x^2 + 1$ et celui de $\mathbf{F}_{2^{8 \cdot r}}$ était $x^{23} + x^5 + 1$.

Le choix des paramètres de la courbe demeure un sujet qui, jusqu'à présent, n'a pas encore été abordé. Un choix judicieux des paramètres doit être fait, afin d'éviter certaines types d'attaques. Le meilleur algorithme connu pour résoudre le problème du logarithme discret sur les courbes elliptiques est le méthode Pollard-ρ [Pol78], qui possède un temps d'exécution proportionel à la racine carré du plus grand diviseur premier de l'ordre du groupe $E(\mathbf{F}_q)$. Cette méthode a été parallelisée par van Oorschot and Weiner [vOW94]. Une autre attaque, due à Menezes, Okamoto et Vanstone [MOV93], réduit le problème du logarithme discret dans le groupe $E(\mathbf{F}_q)$ au problème du logarithme discret dans le corps fini $\mathbf{F}_{q^k}$, pour un entier k quelconque.

Afin d'éviter ces genres d'attaques, on choisit une courbe avec les proprietés suivantes :

1. L'ordre du groupe $E(\mathbf{F}_q)$ est grand.
2. L'ordre du groupe est divisible par un grand facteur premier.
3. Le plus grand facteur premier r de $\#E(\mathbf{F}_q)$ ne divise pas $q^v - 1, v = 1, 2, 3, \ldots, t$, où t est un petit entier.

Les deux premières conditions sont choisies afin d'éviter l'attaque de Pollard-ρ, la dernière rend la réduction de MOV infaisable.

Afin de choisir une courbe elliptique avec de bonnes proprietés, on utilise une technique de Beth et Schaefer [BS91] et Beauregard [Bea96] qui calcule le nombre de points d'une courbe sur un sur-corps à partir du nombre de points de la courbe sur un des ces sous-corps. Pour un exposé sur cette méthode, consultez Beauregard [Bea96]. La courbe choisie avait comme paramètres $a = 0$ (polynôme nul) et $b = (00001111)$ (polynôme avec terme constant (00001111)).

L'ordre du groupe $E(\mathbf{F}_q)$ était un nombre composé de 5 facteurs premiers, dont le plus grand de 150 bits. On a choisit un point P d'ordre n, où n était le plus grand facteur premier de l'ordre du groupe.

5.5 Les coordonnées projectives

Comme il a déjà été signalé au chapitre précédent, des coordonnées projectives peuvent être utilisées afin déliminer les appels à la fonction inversion dans $\mathbf{F}_{2^m}$. Nous allons examiner deux types de coordonnées projectives. Faisons d'abord un rappel bref des opérations d'addition et de doublement sur $E(\mathbf{F}_{2^m})$.

Addition

Soient $P = (x_1, y_1),\ Q = (x_2, y_2) \in E(\mathbf{F}_{2^m})$.

Si $x_1 = x_2$ et $y_2 = x_1 + y_1,\ P + Q = \mathcal{O}$. Sinon $P + Q = R = (x_3, y_3)$, où

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1 \\ \text{et} \quad \lambda &= \frac{(y_2 + y_1)}{(x_2 + x_1)}. \end{aligned}$$

On note que la valeur de a est un des paramètres de l'équation de la courbe. On compte 2 multiplications, 1 mise au carré et 1 inversion. On fait la distinction entre une multiplication et une mise au carré, puisque ce dernier est plus efficace que le premier, peu importe la représentation choisie.

Doublement

Soit $P = (x_1, y_1) \in E(\mathbf{F}_{2^m})$.

Si $x_1 = 0$, $2P = \mathcal{O}$. Sinon $2P = R = (x_3, y_3)$, où

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + a \\ y_3 &= x_1^2 + (\lambda + 1)x_3 \\ \text{et} \quad \lambda &= x_1 + \frac{y_1}{x_1}. \end{aligned}$$

On compte 2 multiplications, 2 mises aux carrés et 1 inversion.

Méthode 1

On définit deux types de coordonnées projectives. Les formules obtenues à partir du premier type se trouvent dans le document de IEEE P1363 [P1398].

Le plan projectif P^2 sur $\mathbf{F}_{2^m}$ est défini comme étant l'ensemble des classes d'équivalences de triplets (X, Y, Z) qui satisfaient la relation

$$(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2) \Longleftrightarrow$$
$$\exists \lambda \neq 0 \in \mathbf{F}_{2^m} \text{ tel que } X_1 = \lambda^2 X_2, Y_1 = \lambda^3 Y_2 \text{ et } Z_1 = \lambda Z_2.$$

L'équation de la courbe dans le plan projectif est obtenue en remplaçant x par X/Z^2 et y par Y/Z^3. On obtient

$$\frac{Y^2}{Z^6} + \frac{XY}{Z^5} = \frac{X^3}{Z^6} + \frac{aX^2}{Z^4} + b$$

En multipliant l'équation par Z^6, on obtient

$$Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6$$

Les points projectifs $(X, Y, Z \neq 0)$ qui vérifient l'équation précédente correspondent au points affines $(x = X/Z^2, y = Y/Z^3)$ qui vérifient l'équation $y^2 + xy = x^3 + ax^2 + b$.

Pour transformer un point affine (x, y) en un point projectif, on pose $X = x, Y = y$ et $Z = 1$. Un point projectif (X, Y, Z) se transforme en point affine en posant $x = X/Z^2$ et $y = Y/Z^3$. Nous allons maintenant réexaminer les opérations d'addition et de doublement en se servant des coordonnées projectives.

Addition

Reprenons les équations pour l'addition.

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1, \\ \text{où} \quad \lambda &= \frac{(y_2 + y_1)}{(x_2 + x_1)}. \end{aligned}$$

On suppose que le point P est exprimé sous forme projectif $P = (X_2 = x_2, Y_2 = y_2, Z_2 = 1)$. Les points affines $Q = (x_1, y_1)$ et $R = (x_3, y_3)$ doivent chacun être transformé en points projectifs. Pour ce faire, on remplace y_1 par Y_1/Z_1^3 et x_1 par X_1/Z_1^2 dans λ pour obtenir

$$\Lambda = \left[\frac{Y_1/Z_1^3 + Y_2}{X_1/Z_1^2 + X_2}\right]\frac{Z_1^3}{Z_1^2 Z_1} = \frac{Y_1 + Y_2 Z_1^3}{(X_1 + X_2 Z_1^2)Z_1}.$$

x_3 devient alors

$$X_3 = \frac{(Y_1 + Y_2 Z_1^3)^2}{(X_1 + X_2 Z_1^2)^2 Z_1^2} + \frac{Y_1 + Y_2 Z_1^3}{(X_1 + X_2 Z_1^2)Z_1} + \frac{X_1}{Z_1^2} + X_2 + a.$$

Soit $Z_3 = (X_1 + X_2 Z_1^2)Z_1$. On multiplie X_3 par $Z_3^2 = (X_1 + X_2 Z_1^2)^2 Z_1^2$ pour obtenir

$$\begin{aligned}
X_3 &= (Y_1 + Y_2 Z_1^3)^2 + (Y_1 + Y_2 Z_1^3)(X_1 + X_2 Z_1^2)Z_1 + (X_1 + X_2 Z_1^2)^2 X_1 \\
&\quad + (X_1 + X_2 Z_1^2)^2 X_2 Z_1^2 + a(X_1 + X_2 Z_1^2)^2 Z_1^2 \\
&= aZ_1^2(X_1 + X_2 Z_1^2)^2 + (Y_1 + Y_2 Z_1^3)\left[(Y_1 + Y_2 Z_1^3) + (X_1 + X_2 Z_1^2)Z_1\right] \\
&\quad + (X_1 + X_2 Z_1^2)^3.
\end{aligned}$$

On procède de la même façon pour trouver Y_3.

$$\begin{aligned}
Y_3 &= \frac{Y_1 + Y_2 Z_1^3}{(X_1 + X_2 Z_1^2)Z_1}\left[\frac{X_1}{Z_1^2} + \frac{(Y_1 + Y_2 Z_1^3)^2}{(X_1 + X_2 Z_1^2)^2 Z_1^2} + \frac{Y_1 + Y_2 Z_1^3}{(X_1 + X_2 Z_1^2)Z_1} + \frac{X_1}{Z_1^2} + X_2 + a\right] \\
&\quad + \frac{(Y_1 + Y_2 Z_1^3)^2}{(X_1 + X_2 Z_1^2)^2 Z_1^2} + \frac{Y_1 + Y_2 Z_1^3}{(X_1 + X_2 Z_1^2)Z_1} + \frac{X_1}{Z_1^2} + X_2 + a + \frac{Y_1}{Z_1^3}.
\end{aligned}$$

On multiplie Y_3 par $Z_3^3 = Z_1^3(X_1 + X_2 Z_1^2)^3$ pour obtenir

$$\begin{aligned}
&= (Y_1 + Y_2 Z_1^3)\left[(X_1 + X_2 Z_1^2)^2 X_1 + (Y_1 + Y_2 Z_1^3)^2 + (Y_1 + Y_2 Z_1^3)(X_1 + X_2 Z_1^2)Z_1\right. \\
&\quad \left. + (X_1 + X_2 Z_1^2)^2 X_1 + (X_1 + X_2 Z_1^2)^2 X_2 Z_1^2 + (X_1 + X_2 Z_1^2)^2 a Z_1^2\right] \\
&\quad + Z_1(X_1 + X_2 Z_1^2)\left[(Y_1 + Y_2 Z_1^3)^2 + (Y_1 + Y_2 Z_1^3)(X_1 + X_2 Z_1^2)Z_1 + (X_1 + X_2 Z_1^2)^2 X_1\right. \\
&\quad \left. + (X_1 + X_2 Z_1^2)^2 X_2 Z_1^2 + (X_1 + X_2 Z_1^2)^2 a Z_1^2\right] + (X_1 + X_2 Z_1^2)^3 Y_1
\end{aligned}$$

$$\begin{aligned}
&= (Y_1 + Y_2 Z_1^3)\left[(X_1 + X_2 Z_1^2)^3 + (Y_1 + Y_2 Z_1^3)^2 + aZ_1^2(X_1 + X_2 Z_1^2)^2\right.\\
&\quad \left.+ (Y_1 + Y_2 Z_1^3)(X_1 + X_2 Z_1^2)Z_1\right] + (Y_1 + Y_2 Z_1^3)(X_1 + X_2 Z_1^2)^2 X_1\\
&\quad + Z_1(X_1 + X_2 Z_1^2)^2\left[(X_1 + X_2 Z_1^2)^3 + (Y_1 + Y_2 Z_1^3)^2 + aZ_1^2(X_1 + X_2 Z_1^2)^2\right.\\
&\quad \left.+ (Y_1 + Y_2 Z_1^3)(X_1 + X_2 Z_1^2)Z_1\right] + (X_1 + X_2 Z_1^2)^3 Y_1\\
&= \left[(Y_1 + Y_2 Z_1^3) + Z_1(X_1 + X_2 Z_1^2)\right]\\
&\quad \left[(X_1 + X_2 Z_1^2)^3 + (Y_1 + Y_2 Z_1^3)^2 + aZ_1^2(X_1 + X_2 Z_1^2)^2 + (Y_1 + Y_2 Z_1^3)(X_1 + X_2 Z_1^2)Z_1\right]\\
&\quad + (X_1 + X_1 Z_1^2)^2\left[(Y_1 + Y_2 Z_1^3)X_1 + (X_1 + X_2 Z_1^2)Y_1\right].
\end{aligned}$$

On veut trouver un moyen de regrouper les mêmes termes dans les équations de X_3, Y_3 et Z_3 afin de minimiser le nombre de multiplications et de mises aux carrés dans $\mathbf{F}_{2^m}$.

Soient $W = X_1 + X_2 Z_1^2$ et $R = Y_1 + Y_2 Z_1^3$.

$$\begin{aligned}
Z_3 &= Z_1(X_1 + X_2 Z_1^2) = Z_1 W \text{ et}\\
X_3 &= aZ_1^2 W^2 + R[R + Z_1 W] + W^3\\
&= aZ_3^2 + R[R + Z_3] + W^3.
\end{aligned}$$

Si on pose $T = R + Z_3$, X_3 devient

$$X_3 = aZ_3^2 + RT + W^3.$$

En utilisant W, R, T et Z_3, Y_3 devient

$$\begin{aligned}
Y_3 &= [R + Z_1 W][W^3 + R^2 + aZ_1^2 W^2 + RWZ_1] + W^2[RX_1 + WY_1]\\
&= [R + Z_3][aZ_3^2 + R(R + Z_3) + W^3]\\
&= TX_3 + W^2[RX_1 + WY_1].
\end{aligned}$$

Nous avons

$$\begin{cases}
X_3 = aZ_3^2 + RT + W^3\\
Y_3 = TX_3 + W^2[RX_1 + WY_1]\\
Z_3 = Z_1 W.
\end{cases}$$

Le nombre de multiplications et mises aux carrés effectuées dans $\mathbf{F}_{2^m}$ est calculé dans le tableau 5.2.

Si on suppose qu'on peut mémoriser les valeurs intermédiares Z_1^2 et W^2, le calcul de (X_3, Y_3, Z_3) exige 11 multiplications et 3 mises aux carrés dans $\mathbf{F}_{2^m}$. Si on suppose que $a = 0$ (un des paramètres de la courbe), le nombre de multiplications est réduit à 10 et le nombre de mises aux carrés à 2. Les coordonnées affines, à leur tour, n'exigent que 2 multiplications et 1 mise au carré dans $\mathbf{F}_{2^m}$, mais elles nécessitent une inversion.

Variable	Nombre de multiplications	Nombre de mises aux carrés
$W = X_1 + X_2(Z_1Z_1)$	1	1
$R = Y_1 + Y_2Z_1Z_1^2$	2	0
$T = R + Z_3$	0	0
$X_3 = aZ_3^2 + RT + W(WW)$	3	2
$Y_3 = TX_3 + W^2[RX_1 + WY_1]$	4	0
$Z_3 = Z_1W$	1	0
	11	3

TAB. 5.2: Nombre d'opérations effectuées dans $\mathbf{F}_{2^m}$ (addition, méthode 1)

Doublement

Comme l'addition et la soustraction sont équivalentes dans un corps de caractéristique 2, on peut réécrire l'équation de la courbe sur $\mathbf{F}_{2^m}$ comme $y^2 + xy + x^3 + ax^2 = b$. Divisant chaque terme par x^2 donne

$$\frac{y^2}{x^2} + \frac{y}{x} + x + a = \frac{b}{x^2}$$

Or,

$$\begin{aligned}
x_3 &= \lambda^2 + \lambda + a \\
&= \left(x_1 + \frac{y_1}{x_1}\right)^2 + \left(x_1 + \frac{y_1}{x_1}\right) + a \\
&= x_1^2 + \left(\frac{y_1^2}{x_1^2} + x_1 + \frac{y_1}{x_1} + a\right) \\
&= x_1^2 + \frac{b}{x_1^2}
\end{aligned}$$

Nous allons utiliser cette nouvelle formule pour x_3.

Comme dans le cas de l'addition, les points affines $Q = (x_1, y_1)$ et $R = (x_3, y_3)$ doivent chacun être transformé en points projectifs. Pour ce faire, on remplace x_1 par X_1/Z_1^2 dans x_3 pour obtenir

$$X_3 = \frac{X_1^2}{Z_1^4} + \frac{bZ_1^4}{X_1^2}.$$

On remplace y_1 par Y_1/Z_1^3 dans λ pour obtenir

$$\Lambda = \frac{X_1}{Z_1^2} + \frac{Y_1}{X_1Z_1}.$$

Soit $Z_3 = X_1 Z_1^2$. On multiplie X_3 par $Z_3^2 = (X_1 Z_1^2)^2$ pour obtenir

$$X_3 = X_1^2 Z_1^4 \left[\frac{X_1^2}{Z_1^4} + \frac{bZ_1^4}{X_1^2}\right] = X_1^4 + bZ_1^8.$$

On procède de la même façon pour trouver Y_3.

$$Y_3 = \frac{X_1^2}{Z_1^4} + \left[\frac{X_1}{Z_1^2} + \frac{Y_1}{X_1 Z_1} + 1\right]\left[\frac{X_1^2}{Z_1^4} + \frac{bZ_1^4}{X_1^2}\right].$$

On multiplie Y_3 par $Z_3^3 = (X_1 Z_1^2)^3$ pour obtenir

$$Y_3 = X_1^5 Z_1^2 + X_1^6 + bZ_1^8 X_1^2 + Y_1 X_1^4 Z_1 + bZ_1^9 Y_1 + X_1^5 Z_1^2 + bX_1 Z_1^{10}$$

On veut trouver un moyen de regrouper les mêmes termes dans les équations de X_3, Y_3 et Z_3 afin de minimiser le nombre de multiplications et de mises aux carrés dans $\mathbf{F}_{2^m}$.

En se servant de $X_3 = X_1^4 + bZ_1^8$, Y_3 devient

$$\begin{aligned} Y_3 &= bZ_1^8(X_1^2 + Z_1 Y_1 + X_1 Z_1^2) + X_1^4(X_1^2 + Y_1 Z_1 + X_1 Z_1^2) + X_1 Z_1^2 X_1^4 \\ &= (bZ_1^8 + X_1^4)(X_1^2 + Y_1 Z_1 + X_1 Z_1^2) + X_1 Z_1^2 X_1^4 \\ &= X_3(X_1^2 + Y_1 Z_1 + X_1 Z_1^2) + X_1 Z_1^2 X_1^4. \end{aligned}$$

Comme $Z_3 = X_1 Z_1^2$,

$$Y_3 = X_3(X_1^2 + Y_1 Z_1 + Z_3) + Z_3 X_1^4.$$

Soit $U = X_1^2 + Z_1 Y_1 + Z_3$.

$$Y_3 = UX_3 + Z_3 X_1^4.$$

Nous avons

$$\begin{cases} X_3 &= X_1^4 + bZ_1^8 \\ Y_3 &= X_3 U + X_1^4 Z_3 \\ Z_3 &= X_1 Z_1^2. \end{cases}$$

Le nombre de multiplications et mises aux carrés effectuées dans $\mathbf{F}_{2^m}$ est calculé dans le tableau 5.3.

Si on suppose qu'on peut mémoriser les valeurs intermédiares Z_1^2, X_1^2 et X_1^4, le calcul de (X_3, Y_3, Z_3) exige 5 multiplications et 5 mises aux carrés dans $\mathbf{F}_{2^m}$. Les coordonnées affines, à leur tour, n'exigent que 2 multiplications et 2 mises aux carrés dans $\mathbf{F}_{2^m}$, mais elles nécessitent une inversion.

Variable	Nombre de multiplications	Nombre de mises aux carrés
$Z_3 = X_1(Z_1 Z_1)$	1	1
$X_3 = (X_1 X_1)^2 + b((Z_1^2)^2)^2$	1	4
$U = Z_3 + X_1^2 + Y_1 Z_1$	1	0
$Y_3 = X_1^4 Z_3 + U X_3$	2	0
	5	5

TAB. 5.3: Nombre d'opérations effectuées dans $\mathbf{F}_{2^m}$ (doublement, méthode 1)

Méthode 2

Nous présentons une dernière variante des coordonnées projectives, due à J. López et R. Dahab [LD98].

Le plan projectif P^2 sur $\mathbf{F}_{2^m}$ est défini comme étant l'ensemble des classes d'équivalences de triplets (X, Y, Z) qui satisfaient la relation

$$\begin{gathered}(X_1, Y_1, Z_1) \sim (X_2, Y_2, Z_2) \Longleftrightarrow \\ \exists \lambda \neq 0 \in \mathbf{F}_{2^m} \text{ tel que } X_1 = \lambda X_2, Y_1 = \lambda^2 Y_2 \text{ et } Z_1 = \lambda Z_2.\end{gathered}$$

L'équation de la courbe dans le plan projectif est obtenue en remplaçant x par X/Z et y par Y/Z^2. On obtient

$$\frac{Y^2}{Z^4} + \frac{XY}{Z^3} = \frac{X^3}{Z^3} + \frac{aX^2}{Z^2} + b$$

En multipliant l'équation par Z^4, on obtient

$$Y^2 + XYZ = X^3 Z + aX^2 Z^2 + bZ^4$$

Les points projectifs $(X, Y, Z \neq 0)$ qui vérifient l'équation précédente correspondent au points affines $(x = X/Z, y = Y/Z^2)$ qui vérifient l'équation $y^2 + xy = x^3 + ax^2 + b$.

Nous allons maintenant revoir les opérations d'addition et de doublement en se servant des coordonnées projectives.

Addition

On suppose d'avance que $a = 0$.

$$\begin{aligned} x_3 &= \lambda^2 + \lambda + x_1 + x_2 \\ y_3 &= \lambda(x_1 + x_3) + x_3 + y_1, \\ \text{où} \quad \lambda &= \frac{(y_2 + y_1)}{(x_2 + x_1)}. \end{aligned}$$

On suppose que le point P est exprimé sous forme projectif $P = (X_2 = x_2, Y_2 = y_2, Z_2 = 1)$. Les points affines $Q = (x_1, y_1)$ et $R = (x_3, y_3)$ doivent chacun être transformé en points projectifs. Pour ce faire, on remplace y_1 par Y_1/Z_1^2 et x_1 par X_1/Z_1 dans λ pour obtenir

$$\Lambda = \left[\frac{Y_1/Z_1^2 + Y_2}{X_1/Z_1 + X_2}\right] \frac{Z_1^2}{Z_1 Z_1} = \frac{Y_1 + Y_2 Z_1^2}{(X_1 + X_2 Z_1) Z_1}.$$

x_3 devient alors

$$X_3 = \frac{(Y_1 + Y_2 Z_1^2)^2}{(X_1 + X_2 Z_1)^2 Z_1^2} + \frac{Y_1 + Y_2 Z_1^2}{X_1 + X_2 Z_1 Z_1} + \frac{X_1}{Z_1} + X_2.$$

Soit $Z_3 = (X_1 + X_2 Z_1)^2 Z_1^2$. On multiplie X_3 par Z_3 pour obtenir

$$\begin{aligned} X_3 &= (Y_1 + Y_2 Z_1^2)^2 + (Y_1 + Y_2 Z_1^2)(X_1 + X_2 Z_1) Z_1 + (X_1 + X_2 Z_1)^2 X_1 Z_1 \\ &\quad + (X_1 + X_2 Z_1)^2 X_2 Z_1^2 \\ &= (Y_1 + Y_2 Z_1^2)^2 + (Y_1 + Y_2 Z_1^2)(X_1 + X_2 Z_1) Z_1 + Z_1 (X_1 + X_2 Z_1)^3. \end{aligned}$$

On procède de la même façon pour trouver Y_3.

$$\begin{aligned} Y_3 &= \frac{Y_1 + Y_2 Z_1^2}{Z_1(X_1 + X_2 Z_1)} \left[\frac{X_1}{Z_1} + \frac{(Y_1 + Y_2 Z_1^2)^2}{Z_1^2 (X_1 + X_2 Z_1)^2} + \frac{Y_1 + Y_2 Z_1^2}{Z_1(X_1 + X_2 Z_1)} + \frac{X_1}{Z_1} + X_2\right] \\ &\quad + \frac{(Y_1 + Y_2 Z_1^2)^2}{Z_1^2 (X_1 + X_2 Z_1)^2} + \frac{Y_1 + Y_2 Z_1^2}{Z_1(X_1 + X_2 Z_1)} + \frac{X_1}{Z_1} + X_2 + \frac{Y_1}{Z_1^2}. \end{aligned}$$

On multiplie Y_3 par $Z_3^2 = Z_1^4 (X_1 + X_2 Z_1)^4$ pour obtenir

$$\begin{aligned} Y_3 &= (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1) \left[(Y_1 + Y_2 Z_1^2)^2 + (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1)\right. \\ &\quad \left. + X_2 Z_1^2 (X_1 + X_2 Z_1)^2\right] + Z_1^2 (X_1 + X_2 Z_1)^2 \left[(Y_1 + Y_2 Z_1^2)^2\right. \\ &\quad + (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1) + X_1 Z_1 (X_1 + X_2 Z_1)^2 + X_2 Z_1^2 (X_1 + X_2 Z_1)^2 \\ &\quad \left. + Y_1 (X_1 + X_2 Z_1)^2\right] \\ &= (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1) \left[(Y_1 + Y_2 Z_1^2)^2 + (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1)\right. \\ &\quad \left. + X_2 Z_1^2 (X_1 + X_2 Z_1)^2\right] + Z_1^2 (X_1 + X_2 Z_1)^2 \left[(Y_1 + Y_2 Z_1^2)^2\right. \\ &\quad \left. + (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1) + Z_1 (X_1 + X_2 Z_1)^3 + Y_1 (X_1 + X_2 Z_1)^2\right]. \end{aligned}$$

En se servant de X_3 et Z_3, on peut réecrire Y_3 comme

$$
\begin{aligned}
Y_3 &= (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1) \left[(X_3 + Z_1 (X_1 + X_2 Z_1)^3 + X_2 Z_3 \right] \\
&\quad + Z_3 \left[X_3 + Y_1 (X_1 + X_2 Z_1)^2 \right] \\
&= (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1)(X_3 + X_2 Z_3) \\
&\quad + Z_1^2 (X_1 + X_2 Z_1)^2 (X_1 + X_2 Z_1)^2 (Y_1 + Y_2 Z_1^2) + Z_3 \left[X_3 + Y_1 (X_1 + X_2 Z_1)^2 \right] \\
&= (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1)(X_3 + X_2 Z_3) \\
&\quad + Z_3 \left[X_3 + (X_1 + X_2 Z_1)^2 (Y_1 + Y_2 Z_1^2) + Y_1 (X_1 + X_2 Z_1)^2 \right] \\
&= (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1)(X_3 + X_2 Z_3) \\
&\quad + Z_3 \left[X_3 + (X_1 + X_2 Z_1)^2 (Y_1 + Y_1 + Y_2 Z_1^2) \right] \\
&= (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1)(X_3 + X_2 Z_3) + Z_3 \left[X_3 + Y_2 Z_3) \right].
\end{aligned}
$$

On veut trouver un moyen de regrouper les mêmes termes dans les équations de X_3, Y_3 et Z_3 afin de minimiser le nombre de multiplications et de mises aux carrés dans $\mathbf{F}_{2^m}$.

Soient

$$
\begin{aligned}
A &= Y_1 + Y_2 Z_1^2 \\
B &= X_1 + X_2 Z_1 \\
C &= Z_1 B = Z_1 (X_1 + X_2 Z_1).
\end{aligned}
$$

Alors $Z_3 = C^2$.

Soient

$$
\begin{aligned}
D &= B^2 C = (X_1 + X_2 Z_1)^2 Z_1 (X_1 + X_2 Z_1) \\
E &= AC = (Y_1 + Y_2 Z_1^2) Z_1 (X_1 + X_2 Z_1).
\end{aligned}
$$

On obtient $X_3 = A^2 + D + E$ et $Y_3 = E(X_3 + X_2 Z_3) + Z_3(X_3 + Y_2 Z_3)$.

Nous avons

$$
\left\{
\begin{aligned}
X_3 &= A^2 + D + E \\
Y_3 &= E(X_3 + X_2 Z_3) + Z_3 (X_3 + Y_2 Z_3) \\
Z_3 &= C^2.
\end{aligned}
\right.
$$

Le nombre de multiplications et mises aux carrés effectuées dans $\mathbf{F}_{2^m}$ est calculé dans le tableau 5.4.

Le calcul de (X_3, Y_3, Z_3) exige 9 multiplications et 4 mises aux carrés dans $\mathbf{F}_{2^m}$. Les coordonnées affines, à leur tour, n'exigent que 2 multiplications et 1 mise au carré dans $\mathbf{F}_{2^m}$, mais elles nécessitent une inversion.

Variable	Nombre de multiplications	Nombre de mises aux carrés
$A = Y_1 + Y_2 Z_1^2$	1	1
$B = X_1 + X_2 Z_1$	1	0
$C = Z_1 B$	1	0
$D = B^2 C$	1	1
$E = AC$	1	0
$X_3 = A^2 + D + E$	0	1
$Y_3 = E(X_3 + X_2 Z_3) + Z_3(X_3 + Y_2 Z_3)$	4	0
$Z_3 = C^2$	0	1
	9	4

TAB. 5.4: Nombre d'opérations effectuées dans $\mathbf{F}_{2^m}$ (addition, méthode 2)

Doublement

On utilise ici la nouvelle formule de x_3.

$$x_3 = x_1^2 + b/x_1^2$$

Comme dans le cas de l'addition, les points affines $Q = (x_1, y_1)$ et $R = (x_3, y_3)$ doivent chacun être transformé en points projectifs. Pour ce faire, on remplace x_1 par X_1/Z_1 dans x_3 pour obtenir

$$X_3 = \frac{X_1^2}{Z_1^2} + \frac{bZ_1^2}{X_1^2}.$$

Soit $Z_3 = X_1^2 Z_1^2$. On multiplie X_3 par Z_3 et on obtient

$$X_3 = X_1^4 + bZ_1^4.$$

On procède de la même façon pour obtenir Y_3. On remplace d'abord y_1 par Y_1/Z_1^2 et x_1 par X_1/Z_1 dans λ pour obtenir

$$\Lambda = \left(\frac{X_1}{Z_1} + \frac{Y_1}{X_1 Z_1}\right).$$

Y_3 devient

$$\begin{aligned} Y_3 &= \frac{X_1^2}{Z_1^2} + \left(\frac{X_1}{Z_1} + \frac{Y_1}{X_1 Z_1} + 1\right)\left(\frac{X_1^2}{Z_1^2} + \frac{bZ_1^2}{X_1^2}\right) \\ &= \frac{X_1^2}{Z_1^2} + \frac{X_1^3}{Z_1^3} + \frac{Y_1 X_1}{Z_1^3} + \frac{X_1^2}{Z_1^2} + \frac{bZ_1}{X_1} + \frac{bY_1 Z_1}{X_1^3} + \frac{bZ_1^2}{X_1^2}. \end{aligned}$$

On multiplie Y_3 par $Z_3^2 = (X_1^2 Z_1^2)^2 = X_1^4 Z_1^4$ pour obtenir

$$\begin{aligned} Y_3 &= X_1^6 Z_1^2 + X_1^7 Z_1 + Y_1 X_1^5 Z_1 + X_1^6 Z_1^2 + bZ_1^5 X_1^3 + Y_1 bZ_1^5 X_1 + bZ_1^6 X_1^2 \\ &= X_1^4 (X_1^3 Z_1 + X_1 Y_1 Z_1) + bZ_1^4 (X_1^3 Z_1 + X_1 Y_1 Z_1) + bZ_1^4 (Z_1^2 X_1^2) \\ &= X_3 (X_1^3 Z_1 + X_1 Y_1 Z_1) + bZ_1^4 Z_3. \end{aligned}$$

L'équation de la courbe $Y^2 + XYZ = X^3 Z + aX^2 Z^2 + bZ^4$ peut s'écrire comme $X^3 Z + XYZ = Y^2 + aX^2 Z^2 + bZ^4$. Si on remplace $X_1^3 Z_1 + X_1 Y_1 Z_1$ par $Y_1^2 + aX_1^2 Z_1^2 + bZ_1^4$ dans l'équation de Y_3, on obtient

$$Y_3 = X_3 (Y_1^2 + aX_1^2 Z_1^2 + bZ_1^4) + bZ_1^4 Z_3.$$

On veut trouver un moyen de regrouper les mêmes termes dans les équations de X_3, Y_3 et Z_3 afin de minimiser le nombre de multiplications et de mises aux carrés dans $\mathbf{F}_{2^m}$.

Au lieu d'utiliser la valeur de b dans les calculs, on calcule et sauvegarde la valeur c telle que $c^2 = b$.

Soient

$$\begin{aligned} A &= Z_1^2 \\ B &= bZ_1^4 = (cA)^2 \\ C &= X_1^2. \end{aligned}$$

On obtient $Z_3 = X_1^2 Z_1^2 = AC$.

Soit $D = C^2$. On obtient $X_3 = D + B$ et $Y_3 = X_3 (Y_1^2 + aZ_3 + B) + BZ_3$.

Le nombre de multiplications et mises aux carrés effectués dans $\mathbf{F}_{2^m}$ est calculé dans le tableau 5.5.

On souligne que le calcul de $c = b^2$ est une valeur précalculée et ne figure par parmi le nombre de mises aux carrés à effectuer. On voit que 5 multiplications et 5 mises aux carrés sont necessaires pour l'opération du doublement. Si on pose $a = 0$, on obtient 4 multiplications et 5 mises aux carrés. Les coordonnées affines, à leur tour, n'exigent que 2 multiplications et 2 mises aux carrés dans $\mathbf{F}_{2^m}$, mais elles nécessitent une inversion.

Le tableau 5.6 récapitule le nombre de multiplications et de mises aux carrés effectuées pour chaque type de coordonnées projectives. (On suppose que $a = 0$).

Afin de choisir le meilleur type de coordonées projectives, il est utile de considérer le temps d'exécution des opérations de multiplication et de mise au carré dans le corps. Avec une représentation polynomiale composée, une mise en œuvre de ces opérations

Variable	Nombre de multiplications	Nombre de mises aux carrés
$A = Z_1^2$	0	1
$B = (cA)^2$	1	1
$C = X_1^2$	0	1
$D = C^2$	0	1
$X_3 = D + B$	0	0
$Z_3 = AC$	1	0
$Y_3 = X_3(Y_1^2 + aZ_3 + B) + BZ_3$	3	1
	5	5

TAB. 5.5: Nombre d'opérations effectuées dans $\mathbf{F}_{2^m}$ (doublement, méthode 2)

Méthode	Opération	Nombre de multiplications dans $\mathbf{F}_{2^m}$	Nombre de mises aux carrés dans $\mathbf{F}_{2^m}$
IEEE P1363	addition	10	2
	doublement	5	5
J. López et al.	addition	9	4
	doublement	4	5

TAB. 5.6: Comparaison de deux types de coordonnées projectives

a permis de déterminer que la mise au carré est environ 13 fois plus rapide qu'une multiplication. En conséquence, les coordonnés projectives avec le plus petit nombre de multiplications en sort gagnantes. La deuxième méthode due à J. López et al. a été choisie, car les opérations d'addition et de doublement possèdent chacun une multiplication de moins par rapport à la méthode de IEEE.

Le tableau 5.7 résume le temps d'exécution des opérations sur $F_{2^{8\cdot 23}}$.

Opération	Temps d'exécution en millisecondes
addition	$0,277$
multiplication	$10,390$
mise au carré	$1,062$
inverse	$272,682$

TAB. 5.7: Temps d'exécution des opérations de $\mathbf{F}_{2^m}$ avec une base polynomiale composée

5.6 Résultats

On souligne que les techniques d'exponentiation développées dans la section 4.3 sont applicables ici car elles dépendent aucunement du corps utilisé. Une clé k de 144 bits a été choisie pour générer la signature numérique. Avec la méthode classique addition-doublement, on effectue 143 doublements et environ 72 additions. Aucun précalcul est nécessaire mais, selon le tableau 5.7, l'algorithme exigerait un temps d'exécution d'environ 13,3 sécondes.

Les techniques de rateau simple avec bits condensés et de rateaux imbriqués ont été implantées. Avec un rateau simple muni de 2 dents, 71 doublements et environ 52 additions ont été effectués, avec un temps d'exécution de 8,62 secondes. 3 points projectifs ont été mémorisés, ce qui représente $3 \times 2 \times 184 = 1104$ bits. Un rateau de 3 dents a permis de réduire de nouveau le temps d'exécution à 6,38 secondes. 47 doublements et 41 additions ont été effectués dans ce cas et 7 points projectifs ont été sauvegardés, ce qui représentent $7 \times 2 \times 184 = 2576$ bits $= 322$ octets.

Une amélioration supplémentaire au temps d'exécution a été possible grâce à la technique de rateaux imbriqués. Avec deux rateaux de trois dents chacun, seulement 23 doublements et 41 additions sont nécessaires, qui a réduit le temps d'exécution à 4,84 secondes. Cette technique à necessité la mémorisation de 14 point projectifs, soient $14 \times 2 \times 184$ bits $= 5152$ bits $= 644$ octets.

Une réduction additionnelle au temps d'exécution n'était pas faisable à cause des grandes exigences de mémoire. 3 rateaux de 3 dents exigeraient la mémorisation de 21 points projectifs, l'équivalent de 966 octets.

Même si les résultats obtenus apparaissent un peu décevants par rapport à la mise en œuvre sur $\mathbf{F}_p$, il ne faut pas oublier que les opérations dans le corps $F_{2^{8 \cdot 23}}$ ont été faites sans l'aide d'un cryptoprocesseur. Il serait peut-être possible d'améliorer le temps d'exécution de l'algorithme d'exponentiation avec d'autres techniques d'optimization. On remarque aussi que la technique de rateaux a pu réduire énormément le temps d'exécution par rapport à la méthode classique d'addition et de doublement, une amélioration quand même de 64 %! Cette technique cependant n'était pas gratuite : on a pu réduire le temps d'exécution mais il fallait mémoriser des valeurs précalculées. Dans le monde de la carte à puce où l'espace est une contrainte importante, il y a certainement un compromis à faire entre l'utilisation de l'espace mémoire et le temps d'exécution.

Chapitre 6

Conclusion

6.1 Représentation des données de $\mathbf{F}_{2^m}$

La capacité d'effectuer des opérations rapides dans un corps fini détermine en grande partie la performance des systèmes basés sur les courbes elliptiques. Dans le cas de $\mathbf{F}_{2^m}$, l'efficacité de ces opérations repose principalement sur la représentation des éléments dans le corps. Les extensions optimales des corps (Optimal Extension Fields) [BP98] et les bases duales circulaires (Circular Dual Basis) [CHJI98] sont deux techniques récentes qui permettent des calculs efficaces dans un corps fini. Les OEF profitent de la multiplication sur les mots pour effectuer des opérations dans le sur-corps. Les bases duales circulaires ont l'avantage que la multiplication se fait avec de simples décalages et des OU-exclusifs. Il serait intéressant de faire la mise en œuvre d'une signature numérique sur $\mathbf{F}_{2^m}$ avec ces techniques de représentation.

6.2 Sécurité des bases polynomiales composées

Une base polynomiale composée a été choisie pour la rapidité avec laquelle les opérations mathématiques sont effectuées, mais ce choix n'est pas sans aucun risque. Il est possible que la structure de ces corps finis, qui sont composés d'un sous-corps, soit plus vunérable à certains types d'attaques [WMPW98]. En particulier, il se peut que le problème du logarithme discret soit plus facile à résoudre avec ces corps. M. Wiener et R.J. Zuccherato [WZ98] ont démontré que cette structure peut effectivement être utilisée pour obtenir des attaques plus rapides. Leur technique est basée sur la méthode de Pollard-ρ. Cette méthode d'attaque connaît une complexité proportionnelle à la racine carré de l'ordre premier du point générateur P. Il a été démontré que le temps d'attaque pour les corps finis $F_{2^{8\cdot r}}$ peut être réduit par un facteur de $\sqrt{2\cdot r}$. Il reste à déterminer si cette méthode, possiblement combinée avec d'autres techniques, pourrait réduire d'avantage le temps d'attaque.

Nombres d'experts ont exprimé leur inquiétude quant à l'usage des bases polynomiales composées. À l'heure actuelle, aucune décision n'a été prise par le comité P1363 de IEEE à ce sujet.

6.3 Sécurité du logarithme discret sur les courbes elliptiques

Il a déjà été signalé que la méthode de Pollard-ρ est une des plus rapides attaques pour le problème du logarithme discret. À l'heure actuelle, plusieurs scientifiques se penchent sur comment améliorer cette technique. E. Teske [Tes98] a démontré comment générer de meilleures marches aléatoires pour cette attaque. La méthode de Pollard-ρ a été parallelisée par P. Oorschot et M. Weiner [vOW94], Gallant et al. [GLV98] et A. Escott [Esc98]. Cette version distribuée demeure l'attaque générique la plus puissante pour le problème du logarithme discret sur les courbes elliptiques.

La méthode des calculs d'indices (Index Calculus Method) est une méthode puissante pour résoudre le problème du logarithme discret dans le groupe multiplicatif $\mathbf{Z}_p^*$. Cependant, cette méthode ne peut pas être appliquée comme telle pour résoudre le problème du logarithme discret sur les courbes elliptiques. Récemment, Silverman [Sil99] a proposé un nouveau algorithme, la méthode de calcul Xedni, qui pourrait être utilisée pour résoudre la problème du logarithme discret sur les courbes elliptiques. Cette méthode inverse les étapes de la technique de calcul d'indice (d'oû vient le nom Xendi). Malheureusement, cette méthode n'a pas encore été mise en pratique ; il serait interessant d'en voir les résultats.

6.4 Comparaison avec d'autres mises en œuvres

Il est intéressant de noter que plusieurs d'autres mises en œuvres efficaces existent sur les courbes elliptiques. Dans le cadre du projet européen CASCADE, le groupe crypto de l'UCL en Belgique a implémenté un ensemble d'opérations élémentaires qui ont pu être utilisées pour calculer le multiple d'un point. Ils ont pu calculer des signatures numériques basées sur les courbes elliptiques en quelques centaines de millisecondes (communication personnelle).

Un collègue de Gemplus à Singapour a pu récemment effectuer une signature sur $F_{2^{167}}$ avec une base polynomiale [HZT99]. La composante utilisée était le Thomson ST16CF54. Grâce à une nouvelle technique, le nombre d'opérations binaires dans l'opération de multiplication à pu être réduit de façon considérable comparée à la méthode classique de décalage et addition.

Michel Gostiaux et Pascal Pallier, également de Gemplus, ont pu obtenir une signature numérique de $300\,ms$ dans le corps $\mathbf{F}_p$, où p est un nombre premier de 160 bits (communication personnelle). La composante utilisée était le SIEMENS SLE 66CX160S.

Bibliographie

[ABMV93] G. B. Agnew, T. Beth, R. C. Mullin, and S. A. Vanstone. Arithmetic operations in GF(2^m). *Journal of Cryptology*, 6 :3–13, 1993.

[ABV89] D. W. Ash, I. F. Blake, and S. A. Vanstone. Low complexity normal bases. *Dicrete Applied Mathematics*, 25 :191–210, 1989.

[AMOV91] G. B. Agnew, R. C. Mullin, I. M. Onyszchuk, and S. A. Vanstone. An implementation for a fast public-key cryptosystem. *Journal of Cryptology*, 3 :63–79, 1991.

[Bea96] D. Beauregard. Efficient algorithms for implementing elliptic curve public-key schemes. Master's thesis, Worcester Polytechnic Institute, May 1996.

[BP98] D. Bailey and C. Paar. Optimal extension fields for fast arithmetic in public-key algorithms. *Advances in Cryptology–CRYTO '98*, pages 472–485, 1998.

[BS91] T. Beth and F. Schaefer. Non supersingular elliptic curves for public key cryptosystems. *Advances in Cryptology–EUROCRYPT '91*, pages 316–327, 1991.

[CHJI98] L. Chang-Hyi and L. Jong-In. A new aspect of dual basis for efficient field arithmetic, 1998. pre-print.

[Cor97] Certicom Corp. Remarks on the security of the elliptic curve cryptosystem, September 1997. Disponible à http ://www.certicom.ca/.

[Cor98] Certicom Corp. The elliptic curve cryptosystem for smart cards, May 1998. Disponible à http ://www.certicom.ca/.

[dKM94] J.-M. de Koninck and A. Mercier. *Introduction à la théorie des nombres*. Modulo Éditeur, 1994.

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31 :469–472, 1985.

[Esc98] A. Escott. Implementing a parallel Pollard rho attack on ECC, 1998. Disponible à http ://www.cacr.math.uwaterloo.ca/.

[GLV98] R. Gallant, R. Lambert, and S. Vanstone. Improving the parallelized Pollard lambda search on binary anomalous curves. *Mathematics of Computation*, 1998. to appear.

[GN98] M. Gostiaux and D. Naccache. Square-and-multiply bit-slice. Gemplus France, 1998.

[GP97] J. Guarjardo and C. Paar. Efficient algorithms for elliptic curve cryptosystems. *Advances in Cryptology–CRYPTO '97*, pages 342–356, 1997.

[HMV92] G. Harper, A. J. Menezes, and S. A. Vanstone. Public-key cryptosystems with very small key lengths. *Advances in Cryptology–Eurocrypt '92*, pages 163–173, 1992.

[HP98] H. Handschuh and P. Paillier. Smart card crypto-coprocessors for public-key cryptography. RSA Laboratories' CryptoBytes, Summer 1998. Disponible à http ://www.rsa.com/.

[HZT99] Yongfei Han, Jiang Zhang, and Peng-Chong Tan. Fast algorithms for elliptic curve cryptosystems on smart cards. In *Cryptographic Techniques and E-Commerce (CrypTEC '99)*, 1999.

[IT88] T. Itoh and S. Tsujii. A fast algorithm for computing multiplicative inverses in in GF(2^m) using normal bases. *Information and Computation*, 78 :171–177, 1988.

[JM97] D. B. Johnson and A. J. Menezes. Elliptic Curve DSA (ESDSA) : An Enhanced DSA. Certicom Corp., August 1997. Disponible à http ://www.certicom.ca.

[Joy95] M. Joye. Introdution élémentaire à la théorie des courbes elliptiques. Technical Report CG-1995/1, UCL Université catholique de Louvain, 1995.

[KAK96] Ç. K. Koç, T. Acar, and B. Kaliski, Jr. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, 16(3) :26–33, June 1996.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177) :203–209, 1987.

[Kob94] N. Koblitz. *A Course in Number Theory and Cryptography*. Springer-Verlag, second edition, 1994.

[LD98] J. López and R. Dahab. Improved algorithms for elliptic curve arithmetic in GF(2^n). In *Proceedings of the Fifth Annual Workshop on Selected Areas in Cryptography*, pages 208–218, 1998.

[LN94] R. Lidl and H. Niederreiter. *Introduction to finite fields and their applications*. Cambridge University Press, Cambridge, 1994.

[Men93] A. J. Menezes. *Elliptic Curve Public Key Cryptosystems*. Kluwer Academic Publishers, Boston, 1993.

[Mil86] V. S. Miller. Use of elliptic curves in cryptography. *Advances in Cryptology–Crypto '85*, pages 417–426, 1986.

[Miy93] A. Miyaji. Elliptic curves over F_p suitable for cryptosystems. *Advances in Cryptology–AUSCRYPT '92*, pages 479–491, 1993.

[Mon85] P. L. Montgomery. Modular multiplication without trial division. *Mathematics of Computation*, 44(170) :519–521, 1985.

[MOV93] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, 39 :1639–1646, 1993.

[MOVW89] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone, and R. M. Wilson. Optimal normal bases in $\mathrm{GF}(p^n)$. *Discrete Applied Mathematics*, 22 :149–161, 1989.

[MvOV97] A. J. Menezes, P. van Oorschot, and S. A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, Boca Raton, Florida, 1997.

[NM96] D. Naccache and D. M'Raïhi. Cryptographic smart cards. *IEEE Micro*, 16(3) :14–24, June 1996.

[P1398] IEEE P1363. Standard Specifications for Public Key Cryptography, September 11, 1998. Disponible à http ://grouper.ieee.org/groups/1363/.

[Pol78] J. Pollard. Monte Carlo methods for index computation mod p. *Mathematics of Computation*, 32 :918–924, 1978.

[RE97] W. Rankl and W. Effing. *Smart Card Handbook.* Gieske and Devrient GmbH, 1997.

[Sae97] M. Saeki. Elliptic curve cryptosystems. Master's thesis, McGill University, February 1997.

[Sil99] J. H. Silverman. The Xedni Calculus and the Elliptic Curve Discrete Logarithm Problem. Technical Report CORR 99-05, University of Waterloo, 1999. Disponible à http ://www.cacr.math.uwaterloo.ca/ (sous Technical Reports).

[SOOS85] R. Schroeppel, H. Orman, S. O'Malley, and O. Spatscheck. Fast key exchange with elliptic curve systems. *Advances in Cryptology, CRYPTO '85*, pages 43–56, 1985.

[ST92] J. H. Silverman and J. Tate. *Rational Points on Elliptic Curves.* Springer-Verlag, 1992.

[Sti95] D. R. Stinson. *Cryptography : Theory and Practice.* CRC Press, Boca Raton, Florida, 1995.

[Tes98] E. Teske. Better random walks for Pollard's Rho Method. Technical Report CORR 98-52, University of Waterloo, 1998. Disponible à http ://www.cacr.math.uwaterloo.ca/ (sous Technical Reports).

[vOW94] P. van Oorschot and M. Weiner. Parallel collision search with applications to hash functions and discrete logarithms. *2nd ACM Conference on Computer and Communications Security*, pages 210–218, 1994. ACM Press.

[WBV+96] E. De Win, A. Bosselaers, S. Vandenberghe, P. De Gersem, and J. Vandewalle. A fast software implementation for arithmetic operations in GF(2^n). *Advances in Cryptology, Proc. Asiacrypt '96 LNCS 1163*, pages 65–76, 1996.

[WMPW98] E. De Win, S. Mister, B. Preenel, and M. Weiner. On the performance of signature schemes based on elliptic curves. *Algorithmic Number Theory Symposium 3, LNCS 1423*, pages 252–266, 1998.

[WZ98] M. J. Wiener and R. Zuccherato. Faster attacks on elliptic curve cryptosystems, April 8, 1998. Disponible à http ://grouper.ieee.org/groups/1363/.

[X9.93] ANSI X9.30 :1. Public Key Cryptography for the Financial Services Industry : Part 1 : The Digital Signature Algorithm (DSA), 1993.

[X9.97] ANSI X9.30 :2. Public Key Cryptography for the Financial Services Industry : Part 2 : The Secure Hash Algorithm (SHA-1), 1997.

[Xx97] ANSI X9.62-199x. Public Key Cryptography for the Financial Services Industry : The Elliptic Curve Digital Signature Algorithm (ECDSA), November 17, 1997.