Utilisation des Couplages en Cryptographie asymétrique pour la Micro-électronique

Loubna Ghammam

ANNÉE 2016

THÈSE / UNIVERSITÉ DE RENNES 1
sous le sceau de l'Université Bretagne Loire

En Cotutelle Internationale avec

L'université de Monastir

pour le grade de

DOCTEUR DE L'UNIVERSITÉ DE RENNES 1

Mention : Mathématiques et Applications

Ecole doctorale Matisse

présentée par

Loubna Ghammam

préparée à l'unité de recherche UMR 6625 CNRS–IRMAR

Institut de Recherche Mathématique de Rennes
U.F.R. de Mathématiques

Intitulé de la thèse : Utilisation des Couplages en Cryptographie asymétrique pour la micro-électronique.

Thèse soutenue à Rennes
16 Décembre 2016

devant le jury composé de :

Mohamed MKAOUER
Professeur, Université de Sfax, Tunisie / rapporteur

Andreas ENGE
Professeur Université de Bordeaux, France / rapporteur

Nadia EL MRABET
Maître de Conférence, École des mines de Saint-Etienne. / examinateur

John BOXALL
Professeur, Université de Caen / examinateur

Sylvain DUQUESNE
Professeur, Université de Rennes 1 / directeur de thèse

Leila BEN ABDELGHANI
Professeur, Université de Monastir / directeur de thèse

Remerciements

Je fais partie des personnes qui croient qu'il n'y a de force ni de puissance que par Dieu. Cela étant, je commence par Le remercier d'avoir eu la bonté de m'entourer de personnes formidables qui ont contribué à la réalisation de cette thèse de doctorat.

En premier lieu, je souhaite exprimer ma reconnaissance envers mes chers directeurs de thèse Sylvain Duquesne et Leila Ben Abdelghani. Ce fut un réel plaisir de travailler avec vous. C'est grâce à vous que je suis maintenant là. Je vous remercie infiniment pour votre aide et votre soutien, surtout aux moments les plus difficiles.

Je souhaite remercier Andreas Enge et Mohamed Mkaouer pour avoir accepté d'être les rapporteurs de ce manuscrit ; leurs conseils et commentaires m'ont permis de bien améliorer ce manuscrit. Je suis très reconnaissante à John Boxall et Nadia Elmrabet de faire partie de mon jury et d'évaluer mes travaux.
Je remercie aussi mes co-auteurs Emmanuel Fouotsa, Anissa Sghaier, Nadia Elmrabet et Nicolas Méloni. Merci à vous tous !
Je suis très reconnaissante à Patrick Lacharme, mon nouveau directeur en étant en post-doc. Merci beaucoup pour ton aide Patrick pour les répétitions et surtout pour le temps que tu m'as donné pour que je termine ma thèse tranquillement. Je remercie aussi toute ma nouvelle équipe GREYC à l'ENSI Caen pour leur soutien.

J'ai effectué ma thèse en cotutelle entre Rennes 1 et Monastir. Je remercie beaucoup mes deux équipes IRMAR et celui de la FSM.
J'ai plutôt passé plus de temps à Rennes 1 qu'à Monastir pour bien profiter de l'ambiance de travail ainsi de la culture bien développée de la cryptographie à l'IRMAR. Je remercie tous les gens que j'ai rencontré et spécialement je remercie Delphine Boucher, Xhensila, Marie-Aude, Carole, Emmanuelle, Hélène, Nelly et Chantal Hallet. Merci pour les tous les renseignements que vous m'avez donné et merci pour votre magnifique accueil.
Á la Faculté des Sciences de Monastir, je tiens à remercier les membres de laboratoire d'électronique et de micro-électronique et en particulier Mohsen Machhout. Je remercie aussi toute l'équipe administrative de la FSM. Merci pour votre aide. Un remercie spécial à Aam Ennaceur, merci pour ton chaleureux accueil.
Cette thèse n'aura pas eu lieu sans l'aide financière que j'ai eu de la part du Centre Henri Lebesgue (du côté Française) ainsi que de la ministère de l'éducation supérieur et de la Recherche Scientifique Tunisienne. Merci et Chokran.

Je remercie infiniment mon collègue Maher Boudabra. Merci Maher pour ton aide, merci de m'avoir accompagné tout le long de ma thèse. Un grand merci à ton adorable maman.
Je remercie aussi ma grande et éternelle amie Maha Elouni, tu es vraiment une amie formidable, notre amitié de plus de 2 décennies malgré les distances m'ont énormément aidé. Un grand bisou

à ma chère Myssen.
Je remercie également mon amie Ahlem Ben Abdallah et ses adorables enfants, merci d'avoir toujours trouvé le temps pour discuter et rigoler.
Merci à mes chères Sonia, Inès, Marwa, merci de vous êtes retrouvées près de moi dans mes moments de faiblesse. Merci pour nos belles soirées (surtout durant les mois de Ramadan avec Shony). Merci à toi aussi Meryem.
Un remerciement spécial à mes amis depuis la première année de l'ISIMM et même depuis la deuxième année secondaire au lycée (pour certains) et aussi de la FSM. Je commence par mes chères Maroi, Olfa, Rim Jday, Safa, Rim Achour, Ameni Dalila et Rabiaa et aussi mes chers Firas, Nizar, Ashref, Ghazi et Dali. Merci pour une amitié de plus que 10 ans !
Je remercie aussi mes chères amies colocataires , spécialement Smaira, Khouloud et Sawssen. C'est super de rester en contact en se rappelant nos aventures.
Merci à mes supers collègues de l'IRMAR. Je dois commencer par Gwezheneg. Merci beaucoup Gwehzeneg de t'être trouvé toujours à côté de moi. Merci de m'avoir aidé à m'intégrer dans l'équipe. Merci pour les répétitions qu'on a fait ensemble (pour une conférence ou même pour le séminaire d'équipe). Je remercie aussi ma chère Turku, 'benim güzel teşekkür ederim'. Merci aussi à Alexandre, merci pour ces bons gâteaux préparés avec Roselyne. Kodjo, Maxime merci à vous aussi. Merci à tous les doctorants de l'IRMAR.

J'ai laissé ma famille en dernier lieu, c'est parce qu'elle est toute ma vie et j'aime terminer par les choses que j'adore. Merci à mes tantes parisiennes pour votre soutien, merci à mes cousines Bil, Bess, Henda, Imène, Nadia, Faïza, Sonia et aussi à tous mes cousins. Merci pour les soirées à Paris qui m'ont donné beaucoup d'énergie pour poursuivre mes travaux à Rennes en très bonne forme. Mes cousines du côté Papa, merci pour vous aussi, merci d'avoir prises de mes nouvelles et merci pour tous vos vœux d'encouragement.
Merci à mon Tonton Boughrara, depuis le décès de papa, je te considère comme un deuxième papa et tu l'es. Merci aussi à Tonton Adel, merci de m'avoir soutenu.
Merci à mes chères sœurs Ayda, Leila, Hager et Fedwa et à mon cher frère Mohamed. Je sais bien que les mots ne suffisent jamais pour exprimer mon amour envers vous. Très fière de vous avoir dans ma vie. Je commence par ordre croissant (pour éviter la jalousie).
La grande sœur est une deuxième maman. Oui elle l'est. Merci Ayda pour tout, merci à mon beau-frère Mohamed Ali. Merci pour tous ces délices de la mer que tu pêches et que tu m'envoie. Quatre ans en France et je n'ai mangé que tes poissons. J'adore mes neveux Ilèf (Oufa), Hédi (Hedou) et Lyne (Lalouna).
Le Grand et le seul Frère. Merci à mon frère Mohamed de m'avoir accompagné, d'avoir pu passer des week-ends chez toi qui m'ont vraiment fait changer d'humour. Merci à ma chère belle-sœur Zahra plutôt à ma cinquième sœur. La princesse Eyouta bienvenue dans ce monde.
Ma deuxième soeur, Leila, merci ma soeur pour tout. Merci pour les moments qu'on a passé ensemble au téléphone pour ne pas se sentir seule. Merci à mon cher beau-frère Fehmi. Merci pour tes conseils et surtout pour les pâtisseries de Zaghouan. Mes chers neveux Ysra (Sarroura) et Farès (Berkous), je vous adore.
Ma sœur Hajer, je préfère plutôt te dire El Bagra même si tu ne l'es pas. Je te souhaite tout le bonheur du monde. C'est tout ce que tu mérites dans cette vie que du bien. Merci ma chère sœur pour ton soutien surtout dans les moments de ma faiblesse. Un grand bisou à mon cher neveu Ahmed (Atta).

Ma chère sœur Fedwa (Touta), ma petite sœur inchaALLAH bientôt tu seras une excellente ingénieure.
Maman, je ne sais pas quoi te dire tellement que j'ai à dire. Je te dis tout le temps Merci pour tout, mais tu me répondras, je suis une maman et c'est mon devoir d'être à côté de toi, tu me dis aussi toujours quand tu seras maman tu me comprendras. Je ne suis pas encore maman mais je te comprends bien. Tout ce que je souhaite dans cette vie est que tu sois en très bonne santé et que tu sois toujours à nos côtés. Nous avons vraiment besoin de toi.
Finalement je dédie cette thèse à toi PAPA, ton absence me tue, mais aussi ton absence m'a rendue beaucoup plus forte, grande et responsable.

Je remercie cette sacrée ville de Rennes, la ville de mon cœur, Rennes la Reine. Merci à tous.

" La pensée a des ailes, nul ne peut empêcher son envol"
Ibn Rochd (Averroès)

Introduction

Les couplages sont des outils mathématiques introduits par André Weil en 1948 dans [Wei48]. Ils sont un sujet très en vogue depuis une qinzaine d'années en cryptographie asymétrique. Ils permettent en effet de réaliser des opérations cryptographiques impossible à réaliser simplement autrement tel que la signature courte [BLS04] et la cryptographie basée sur l'identité [BBG05].

Ces dernières années, le calcul des couplages est devenu plus facile grâce à l'introduction de nouvelles méthodes de calculs mathématiques particulièrement efficaces sur certaines courbes elliptiques, dites bien adaptées aux couplages.

Aujourd'hui, nous sommes au stade de transfert de cette technologie, de la théorie vers la mise en œuvre pratique, sur des composants électroniques. Ce transfert soulève de nombreuses problématiques qui s'avèrent difficile à surmonter à cause de la différence de culture scientifique entre mathématiciens et micro-électroniciens. Notre but dans cette thèse est de répondre aux deux problématiques suivantes :

1. Problème de l'implémentation du couplage dans des environnement restreints. En effet, le calcul du couplage de Tate, ou aussi de l'une de ses variantes, nécessite plusieurs variables pour être implémenté. Par conséquent, il nécessite une bonne partie de la mémoire du composant électronique sur lequel nous souhaitons implémenter un tel couplage voire dépasse les capacités du composant (carte à puce, carte SIM).
2. Problème de la sécurité des protocoles cryptographiques basés sur les couplages. Il faut en effet protéger les algorithmes de calcul du couplage contre les attaques matérielles.

Le premier problème que nous traitons dans ce manuscrit consiste en l'implémentation matérielle du couplage dans un environnement restreint. Nous désignons par environnement restreint tout composant électronique dont la capacité de la mémoire est limitée. Ceci veut dire que ce n'est pas évident d'implanter facilement des protocoles cryptographiques qui nécessitent une bonne taille de mémoire.
Un couplage est une application bilinéaire qui sert à envoyer (P, Q), un couple de points d'une courbe elliptique, dans le sous-groupe multiplicatif des racines $r-$ièmes de l'unité du corps $\mathbb{F}_{p^k}$. Les couplages ont un rôle important en cryptographie que nous détaillons dans le chapitre 1 de ce manuscrit. Parmi ces rôles, nous citons leur utilisation destructive. Pour atteindre un bon niveau de sécurité p^k doit être de grande taille. On aura donc besoin d'implémenter des éléments de $\mathbb{F}_{p^k}$ qui nécessitent beaucoup de mémoire pour être stockés.
Nous allons voir comment nous pouvons réduire la mémoire utilisée pour implémenter un couplage sur les courbes elliptiques. Dans notre travail, nous nous intéresserons spécifiquement au couplage Optimal Ate, qui est à ce jour le plus efficace. Le calcul de ce couplage nécessite entre autre une exponentiation d'un élément de $\mathbb{F}_{p^k}$ et c'est cette étape qui, d'après la littérature, nécessite un important nombre de variables temporaires dans $\mathbb{F}_{p^k}$ et par conséquent elle consomme beaucoup de

mémoire. Le problème de l'implémentation dans un environnement restreint a été posé dans le cadre du projet ANR SIMPATIC intitulé *SIM and PAiring Theory for Information and Communication security*. Il existe plusieurs travaux dans la littérature qui détaillent le calcul de l'exponentiation finale. Mais peu de travaux ont traité le problème de gestion des ressources mémoire.
Nous reprenons les travaux existants dans la littérature ([DSD07], [SBC+09], [CKH11]) et en proposons des variantes nécessitant nettement moins de ressources mémoires et de complexité équivalente (parfois un plus efficaces, parfois un peu moins). Nous allons ainsi proposer des solutions mathématiques dans le développement de l'exposant intervenant dans cette exponentiation utilisant peu de variables temporaires. Les solutions proposées permettent de rendre possible une implémentation efficace d'un couplage Optimal Ate sur un composant de 3 voire 2 Ko de mémoire alors que les méthodes de la littérature en nécessitaient jusqu'à plus de 4.

La deuxième problématique de cette thèse concerne la sécurité des protocoles cryptographiques à base des couplages, en particulier dans le cadre des attaques matérielles comme l'attaque DPA. Nous présentons deux contre-mesures efficaces afin de protéger le calcul du couplage de cette attaque basée sur une réécriture de l'exposant en utilisant les chaînes d'additions et les nombres de Fibonacci. Par rapport aux contre-mesures de la littérature, les notres sont plus efficaces d'environ **18**%.

Le présent document se décompose en cinq chapitres. La première partie, qui comporte les deux premiers chapitres, concerne la présentation des outils mathématiques nécessaires pour le calcul du couplage sur les courbes elliptiques. Le Chapitre 1 est une introduction à la théorie des corps finis et des courbes elliptiques. Dans ce chapitre, nous présentons les principales propriétés des courbes elliptiques et nous donnons les éléments théoriques nécessaires pour la définition du couplage. Par la suite, nous introduisons la notion de couplage sur les courbes elliptiques. Nous présentons en premier lieu le couplage de Weil [Mil04] et celui de Tate [FMR99a]. Nous présentons également des optimisations du couplage de Tate (Ate [MKHO07], Twisted Ate [MKHO07] et Optimal Ate [Ver10]).
Le Chapitre 2 concerne l'arithmétique sur les corps finis pour les couplages. Nous détaillons les différentes opérations dont nous aurons besoin dans la suite de cette thèse pour le calcul du couplage Optimal Ate sur plusieurs catégories de courbes elliptiques.
Les chapitres suivants sont issus des travaux de recherche effectués durant les années de thèse.

Le Chapitre 3 est une réponse à la première problématique que nous avons présentée. Dans ce chapitre, nous résolvons le problème d'implémentation du couplage Optimal Ate dans un environnement restreint pour les courbes BN. Nous rappelons en premier lieu les méthodes existantes permettant de calculer la partie difficile de l'exponentiation finale des couplages (méthode naïve, méthode de Devegili *et al.* [DSD07], méthode de la chaîne d'addition de Scott *et al.* [SBC+09] et méthode de Fuentes *et al.* basée sur un multiple de l'exposant [CKH11]).
Ensuite nous étudions leurs demandes en ressources mémoire car ce n'est pas fait dans la littérature puis nous présentons nos variantes de ces méthodes qui permettent de gagner en ressources mémoire et parfois en efficacité. La dernière de nos variantes, en jouant sur la parité du paramètre u est plus rapide que tous les résultats présentés dans la littérature et en utilisant peu de ressources mémoire.
Le Chapitre 4 concerne le calcul de l'exponentiation finale du couplage pour de plus hauts niveaux de sécurité. Ce chapitre consiste en deux parties. La première partie concerne le cas des courbes

de Barreto, Lynn et Scott de degrés de plongement respectivement égaux à 12 et 24 notées BLS12 et BLS24. Nous présentons des méthodes de calcul plus efficaces que celles présentées dans la littérature.
Dans la seconde partie, nous déterminons la courbe elliptique la plus adéquate pour le calcul du produit ou du quotient de n couplages (utile pour plusieurs protocoles cryptographiques) pour un haut niveau de sécurité. Un résultat important présenté par Zhang et Lin dans [ZL12] affirme qu'il est conseillé de choisir les courbes elliptiques KSS de degré de plongement k égal à 16 notées KSS16. Nous démontrons que si $n > 2$, la courbe KSS16 est effectivement la meilleure solution pour le calcul de n couplages mais que pour $n = 2$, la courbe BLS12 est préférable. Nous vérifions également que la courbe KSS16 présentée par Zhang *et al.* ne résiste pas aux attaques par sous-groupes et nous proposons une nouvelle courbe plus sécurisée.

Dans le Chapitre 5, nous proposons une contre-mesure pour les attaques Différentielles par Consommation de Courant (DPA) sur l'algorithme de Miller. Elle est basée sur une nouvelle écriture de l'algorithme de Miller. En effet, dans le cas classique, pour calculer la fonction rationnelle $f_{u,Q}$, nous écrivons le paramètre u en base 2. Ici, nous représentons le paramètre u à l'aide d'une chaîne d'additions euclidienne ou à l'aide d'une suite de Fibonacci. Nous donnons ainsi deux nouvelles versions de l'algorithme de Miller qui sont moins efficaces que la version originale du fait d'une représentation plus longue de u mais qui résistent aux attaques DPA. Cela donne finalement un algorithme plus efficace que les autres algorithmes protégés de la littérature.

Table des matières

Chapitre 1

Généralités

Nous présentons dans ce chapitre les éléments théoriques nécessaires à la compréhension des courbes elliptiques et des couplages sur les courbes elliptiques. Nous commençons par introduire les courbes elliptiques en donnant leur définition et leurs propriétés principales.
Ensuite, nous présentons la notion de couplages.

1.1 Définitions et propriétés des courbes elliptiques

Avant de définir les courbes elliptiques et de donner leurs propriétés nous donnons quelques notions sur les corps finis.

1.1.1 Généralités sur les corps finis

Dans ce paragraphe, nous rappelons quelques propriétés importantes des corps finis. Une étude détaillée sur les corps finis est présentée dans [LN94].
Pour p un nombre premier, $\mathbb{F}_p$ désigne le corps fini à p éléments $\mathbb{Z}/p\mathbb{Z}$. Ce corps est de caractéristique p. Rappelons que, la caractéristique d'un corps est le plus petit entier non nul n tel que le produit de tout élément du corps par n soit nul. Si un tel n n'existe pas, alors le corps est dit de caractéristique nulle.

Définition 1.1. *L'ordre d'un élément x de $\mathbb{F}_p^*$ est le plus petit entier non nul α tel que $x^\alpha = 1$.*
Les générateurs de $\mathbb{F}_p^*$ sont les éléments d'ordre $p-1$.
Nous désignons par $\mathbb{F}_p[X]$ l'ensemble des polynômes à coefficients dans $\mathbb{F}_p$. Dans ce travail nous aurons plutôt besoin de la notion de polynôme irréductible.

Définition 1.2. *Un polynôme non nul $P(X) \in \mathbb{F}_p[X]$ de degré n est dit irréductible dans $\mathbb{F}_p[X]$ si $P(X)$ n'est divisible que par ses polynômes associés et par les constantes non nuls.*

Pour définir les couplages sur les courbes elliptiques, nous aurons besoin d'utiliser des extensions de corps de $\mathbb{F}_p$. Une extension de $\mathbb{F}_p$ peut être interprétée comme un corps plus gros englobant $\mathbb{F}_p$.

Définition 1.3. *Une extension de degré k du corps $\mathbb{F}_p$, notée $\mathbb{F}_{p^k}$, est un corps contenant $\mathbb{F}_p$ et tel que la dimension de $\mathbb{F}_{p^k}$ en tant que $\mathbb{F}_p$ espace vectoriel vaut k.*

Exemple 1.1. *Le quotient $\mathbb{F}_p[X]/\langle P(X)\rangle$, où $\langle P(X)\rangle$ est l'idéal engendré par le polynôme irréductible $P(X)$ de degré k est un exemple d'une extension contenant $\mathbb{F}_p$ de degré k.*

Remarquons qu'une extension de degré k de $\mathbb{F}_p$ peut être identifiée à l'ensemble des polynômes de degré strictement inférieur à k et à coefficients dans $\mathbb{F}_p$:

$$\mathbb{F}_{p^k} = \{R(X) \in \mathbb{F}_p[X] \text{ tel que } deg(R) < k\}.$$

Remarque 1.1. *Une extension de degré k de $\mathbb{F}_p$ ne dépend pas du polynôme irréductible $P(X)$. Toutes les extensions sont isomorphes.*

Propriété 1.1. *Le cardinal de $\mathbb{F}_{p^k}$ est noté $\sharp\mathbb{F}_{p^k}$ et est p^k.*

Exemple 1.2. *Dans cet exemple, nous proposons une manière de construire le corps fini $\mathbb{F}_{p^{12}}$ donnée par le théorème suivant.*

Théorème 1.1. *Soit p un nombre premier tel que $p = 36u^4 + 36u^3 + 24u^2 + 6u + 1$ où u est une entier impair.*
Alors $\mathbb{F}_{p^2}$ peut être définit par le polynôme irréductible $X^2 + 1$ dont on notera $\boldsymbol{i}$ une racine et $1 + \boldsymbol{i}$ n'est ni carré, ni un cube dans $\mathbb{F}_{p^2}$ si et seulement si $u \equiv 7[12]$ ou $u \equiv 11[12]$.
Par conséquent, ayant cette condition, on construit $\mathbb{F}_{p^{12}}$ comme suit :

$$\begin{aligned} \mathbb{F}_{p^2} &= \mathbb{F}_p[\boldsymbol{i}] \text{ avec } \boldsymbol{i}^2 = -1 \text{ car } p \equiv 3(4) \\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[\beta] \text{ avec } \beta^3 = 1 + \boldsymbol{i} \\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^2}[\gamma] \text{ avec } \gamma^6 = 1 + \boldsymbol{i}. \end{aligned}$$

Pour la preuve de ce théorème, nous aurons besoin des propositions suivantes.

Proposition 1.1. *Soit p un nombre premier tel que $p \equiv 1[3]$, alors a est un carré ou un cube dans $\mathbb{F}_{p^2}$ si et seulement si sa norme est un carré ou un cube dans $\mathbb{F}_p$.*

Proposition 1.2. *Il est simple de voir que :*

- *-1 est un carré dans $\mathbb{F}_p$ si et seulement si u est pair.*
- *2 est un carré dans $\mathbb{F}_p$ si et seulement si $u = 0$ ou 1 modulo 4.*
- *2 est un cube dans $\mathbb{F}_p$ si et seulement si $u = 0$ modulo 3.*

Pour la preuve des Propositions 1.1 et 1.2 voir [DMHR15]. Revenons maintenant à la démonstration du Théorème 1.1.

Preuve. Nous allons montrer que $(1 + \mathbf{i})$ n'est ni un carré ni un cube dans $\mathbb{F}_{p^2}$ si et seulement si $u = 7$ ou 11 modulo 12.
D'après la Proposition 1.1, $(1+\mathbf{i})$ est un carré dans $\mathbb{F}_{p^2}$ si et seulement si $2 = N_{\mathbb{F}_{p^2}/\mathbb{F}_p}(1+\mathbf{i})$ est un carré dans $\mathbb{F}_p$. Or d'après la Proposition 1.2, 2 est un carré si et seulement si $u = 0$ ou 1 modulo 4. Ainsi, nous obtenons $(1 + \mathbf{i})$ n'est pas un carré dans $\mathbb{F}_{p^2}$ si et seulement si $u \equiv 3[4]$. De même, d'après la Proposition 1.1, $(1 + \mathbf{i})$ n'est pas un cube si et seulement si 2 n'est pas un cube. Or d'après la Proposition 1.2, 2 est un cube si et seulement si $u = 0$ modulo 3. Par conséquent, $(1+\mathbf{i})$ n'est pas un cube si et seulement si $u = 1$ ou 5 modulo 6.
D'où le résultat. □

Définition 1.4. *Une clôture algébrique du corps fini $\mathbb{F}_p$ est une extension de ce corps sur laquelle tout polynôme de $\mathbb{F}_p[X]$ est scindé. Nous la noterons $\overline{\mathbb{F}_p}$.*
Les clôtures algébriques d'un corps ne sont pas uniques mais elles sont toutes isomorphes [LN94].

1.1.2 Généralités sur les courbes elliptiques

Définition 1.5. *Une courbe elliptique E sur le corps $\mathbb{K}$, notée $E(\mathbb{K})$, est définie par une équation de Weierstrass de la forme :*

$$y^2 + a_1xy + a_3y \quad = \quad x^3 + a_2x^2 + a_4x + a_6, \ \textit{avec } a_i \in \mathbb{K}, \tag{1.1}$$

non singulière, c'est-à-dire, les dérivées partielles $(2y + a_1x + a_3)$ et $(3x^2 + 2a_2x + a_4 - a_1y)$ ne s'annulent pas simultanément en un point de la courbe.
Un point P de cette courbe elliptique est représenté par ses coordonnées affines (x, y), $x \in \mathbb{K}$, $y \in \mathbb{K}$ que nous notons $P(x, y)$.

La courbe elliptique $E(\mathbb{K})$ est alors l'ensemble des points $P = (x, y)$ de $\mathbb{K}^2$ qui vérifient l'Équation 1.1 auxquels on rajoute **un point à l'infini** qu'on note P_∞. Ainsi, une courbe elliptique est donné par

$$E(\mathbb{K}) = \{(x, y) \in \mathbb{K} \times \mathbb{K} \text{ tels que } y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\} \cup \{P_\infty\}$$

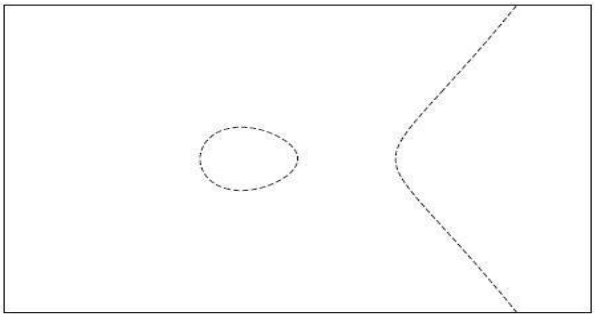
Dans certains cas, l'Équation 1.1 de la courbe elliptique peut être simplifiée. Plus précisément, on a :

Théorème 1.2. *[Sil86] Soit $\mathbb{K}$ un corps fini de caractéristique $\geqslant 5$. Alors pour toute courbe elliptique $E(\mathbb{K})$ il existe un changement de variables pour lequel la courbe elliptique présentée par l'Équation 1.1 admet une équation plus simple dite équation de Weierstrass courte*

$$y^2 = x^3 + ax + b \textit{ avec } a \textit{ et } b \textit{ dans } \mathbb{K}$$

La figure suivante est un exemple d'une courbe elliptique définie sur $\mathbb{R}$ par l'équation

$$E : \ y^2 = x^3 - x$$



$E \ : y^2 = x^3 - x$

Toute courbe elliptique admet deux invariants : le discriminant et le $j-$invariant.

Définition 1.6. *Pour toute courbe elliptique d'équation de Weierstrass courte (dite aussi réduite) $E : y^2 = x^3 + ax + b$, nous associons :*

- *Le discriminant que nous notons Δ et qui est égal à $-16(4a^3 + 27b^2)$,*
- *le $j-$invariant j qui est défini par la quantité $-1728\dfrac{(4a)^3}{\Delta}$*

Remarquons que le discriminant non nul assure la non singularité de la courbe elliptique.

Remarque 1.2. *Toute courbe elliptique définie sur un corps $\mathbb{K}$ de caractéristique 2 ou 3 admet une équation de Weierstrass réduite comme suit.*

- *Si $carac(\mathbb{K}) = 2$ alors $E : y^2 + xy = x^3 + ax^2 + b$. Dans ce cas, le discriminant Δ est égal à b.*
- *Si $carac(\mathbb{K}) = 3$ alors $E : y^2 = x^3 + ax^2 + b$. Dans ce cas, le discriminant Δ est égal à a^3b.*

1.1.3 Loi de groupe

Soit $E(\mathbb{K})$ une courbe elliptique définie sur le corps $\mathbb{K}$. D'après [Sil86], il existe une loi de groupe notée $+$ sur $E(\mathbb{K})$. Une telle loi existe en toute caractéristique du corps $\mathbb{K}$. Dans cette thèse, nous ne considérons que les courbes elliptiques définies sur les corps de grande caractéristique (différente de 2 et de 3). Nous allons donc considérer la courbe elliptique donnée par l'équation de Weierstrass courte $E : y^2 = x^3 + ax + b$.
La loi de groupe repose essentiellement sur le théorème suivant :

Théorème 1.3. *[Sil86] Soit E une courbe elliptique et D une droite du plan. Si D est tangente en un point à la courbe ou bien si D coupe E en deux point distincts alors D coupe E en un unique autre point.*
Si on est dans le cas d'une droite verticale alors le troisième point d'intersection est bien le point à l'infini P_∞.

Géométriquement, soient P et Q deux points de la courbe elliptique $E(\mathbb{K})$. Le point $P + Q$ est obtenu par la construction décrite dans la figure suivante.

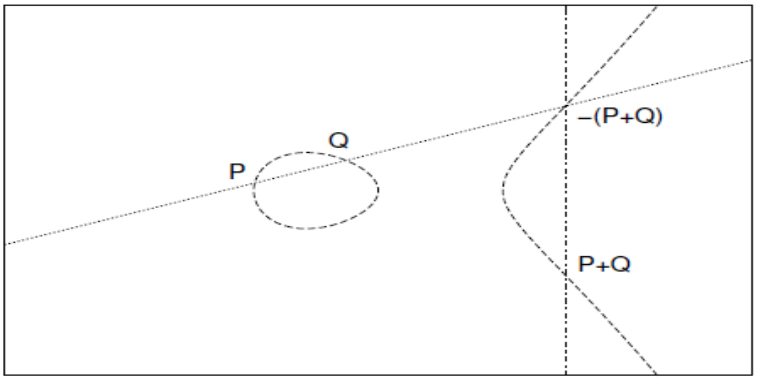


FIGURE 1.1 – Somme de deux points

Nous traçons la droite (PQ). Cette droite coupe obligatoirement la courbe $E(\mathbb{K})$ en un troisième point que nous notons $-(P+Q)$. Le point $P+Q$ est alors l'intersection de la verticale passant par ce troisième point avec $E(\mathbb{K})$.
Si nous sommes dans le cas où $P = Q$, c'est-à-dire si nous voulons traçer le double du point P, il suffit de tracer la tangente passant par P qui va aussi couper la courbe en un deuxième point. Le point $2P$ est donc l'intersection de la courbe E avec la verticale passant par ce point d'intersection comme c'est indiqué dans la figure suivante.

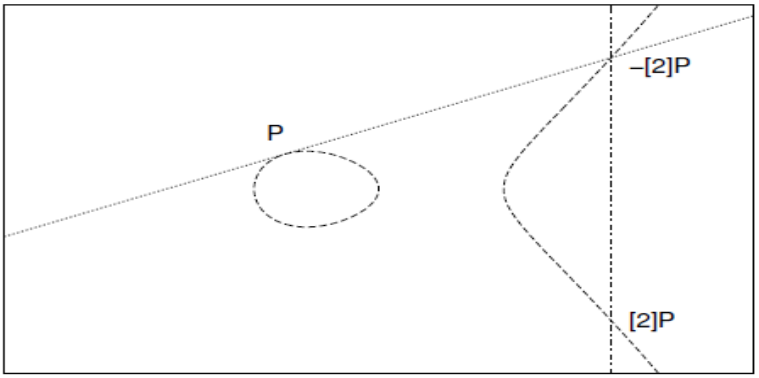


FIGURE 1.2 – Le double d'un point

Remarque 1.3. *Nous pouvons refaire cette opération r fois afin de trouver* $[r]P$ *en additionnant r fois le point* P*.*

$$[r]P = \underbrace{P + P + \cdots + P}_{\boldsymbol{r\ fois}}$$

Théorème 1.4. *L'ensemble des points d'une courbe elliptique* E *muni de la loi* $+$ *est un groupe abélien dont l'élément neutre est* P_∞*.*

Graphiquement, comme nous l'avons décrit, il est évident de trouver la somme de deux points d'une courbe elliptique. Algébriquement aussi, il est simple de décrire la loi de groupe. Soient P et Q deux points de la courbe elliptique dont nous souhaitons calculer la somme. Nous nous plaçons dans le plan affine pour écrire les équations des droites qui permettent de trouver explicitement les expressions de la somme de deux points (pareil pour le double d'un point de E).
Soient P et Q deux points de E de coordonnées respectives (x_P, y_P) et (x_Q, y_Q). Nous décrivons dans les deux paragraphes suivants comment calculer la somme de ces deux points de E ainsi que le double du point P.

Addition de deux points

Soient $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ deux points distincts de $E(\mathbb{K})$ tels que $Q \neq (-P)$. Nous allons déterminer les coordonnées du point $R = (x_R, y_R)$ où $R = P+Q$. Nous devons alors résoudre le système à deux équations formé par l'équation de la courbe elliptique E et l'équation de la droite (PQ). La solution de ce système est bien le point $-(P+Q)$ qui est le troisième point d'intersection de la courbe avec la droite (PQ). Le point R est donné par les formules suivantes où $\lambda_{P,Q}$ est la pente de la droite (PQ).

$$\begin{cases} \lambda_{P,Q} = \frac{y_P - y_Q}{x_P - x_Q} \\ x_R = \lambda_{P,Q}^2 - x_P - x_Q \\ y_R = \lambda_{P,Q}(x_P - x_R) - y_P \end{cases} \tag{1.2}$$

En utilisant ces coordonnées, les coordonnées affines, le coût de l'addition de deux point sur la courbe elliptique est $1S + 2M + I$. Où nous désignons par S une élévation au carré, M une multiplication et I une inversion. Ces opérations sont toutes dans le corps $\mathbb{K}$. Nous ne comptons pas les additions et les soustractions.

Remarque 1.4. *Si $Q = -P$, la droite $(P(-P))$ est une droite verticale. Le troisième point d'intersection de la courbe elliptique et de cette droite est le point à l'infini P_∞.*

Doublement d'un point

Pour le calcul des coordonnées du point $R = [2]P$, nous utilisons le même principe que pour le calcul de la somme de deux points. Dans ce cas, l'équation de la courbe elliptique et la tangente à E au point P forment un système dont la solution est le point $-R$. Il est facile par la suite de déduire les coordonnées de R qui sont donnés par les formules suivantes. $\lambda_{P,P}$ est la pente de la tangente à E au point P.

$$\begin{cases} \lambda_{P,P} = \frac{3x_P^2+a}{2y_P} \\ x_R = \lambda_{P,P}^2 - 2x_P \\ y_R = \lambda_{P,P}(x_P - x_R) - y_P \end{cases} \tag{1.3}$$

Le calcul de double d'un point P sur la courbe elliptique coûte $2S + 2M + 1I$.

Exemple 1.3. *Soit E la courbe elliptique définie sur $\mathbb{F}_p$ par l'équation $y^2 = x^3 + 5x + 8$ où $p = 13$. Soient $P = (9,\ 7)$ et $Q = (1,\ 8)$ deux points de E. Sur cette courbe, nous pouvons facilement calculer la somme de P et Q qui est donnée par le point $P + Q$ de coordonnées $(2, 10)$. De même, le double de point P est $[2]P = (9,\ 6)$.*

L'opposé d'un point

Les coordonnées du point $-P$, l'inverse du point P pour la loi $+$, peuvent s'exprimer en fonction des coordonnées de P selon la caractéristique de $\mathbb{F}_p$. Si par exemple le corps est de caractéristique supérieure ou égale à 5 alors $-P$ est donné par $(x_P, -y_P)$.

1.2 Systèmes de coordonnées

Les formules d'addition et de doublement font intervenir des inversions dans $\mathbb{F}_p$. Malheureusement, c'est une opération compliquée et coûteuse sur les corps finis, il est mieux de l'éviter. Nous devons alors utiliser d'autres systèmes de coordonnées pour éviter ces inversions Parmi ces systèmes, nous considérerons les systèmes de coordonnées projectives et jacobiennes.
Nous détaillons dans cette section ces deux systèmes de coordonnées et nous donnons à chaque fois la complexité du doublement et d'addition des points sur les courbes elliptiques définies sur tout corps de caractéristique ≥ 5.

1.2.1 Les coordonnées projectives

Les coordonnées projectives du point P sont représentées par le triplet $(X_P : Y_P : Z_P)$, correspondant au point affine $(X_P/Z_P : Y_P/Z_P)$ si Z_P est non nul et au point à l'infini P_∞ sinon. Ces coordonnées sont homogènes, c'est-à-dire que pour tout α non nul, le point (X_P, Y_P, Z_P) est équivalent à $(\alpha X_P : \alpha Y_P : \alpha Z_P)$.
L'équation de la courbe elliptique E devient

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

L'opposé du point $P = (X_P : Y_P : Z_P)$ est le point $-P = (X_P : -Y_P : Z_P)$ et le point à l'infini est de coordonnées $(0 : 1 : 0)$.
Soit $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ deux points de E en coordonnées affines. Nous voulons déterminer les formules de l'addition et de doublement pour les coordonnées projectives.

Nous remplaçons dans la formule de doublement (1.3) x_P et y_P par leurs équivalents en coordonnées projectives $\frac{X_P}{Z_P}$ et $\frac{Y_P}{Z_P}$. La pente de la tangente à la courbe au point P devient :

$$\lambda_{P,P} = \frac{3X_P^2 + aZ_P^2}{2Y_P Z_P}$$

Les formules de doublement d'un point P en cordonnées projectives sont données par les formules suivantes :

$$\begin{cases} X_R = 2Y_P Z_P \left((aZ_P^2 + 3X_P^2)^2 - 8X_P Y_A^2 Z_P\right), \\ Y_R = (aZ_P^2 + 3X_P^2)\left(4X_P Y_P^2 Z_P - (aZ_P^2 + 3X_P^2)^2 - 8X_P Y_P^2 Z_P\right) - 8Y_P^4 Z_P^2, \\ Z_R = 8Y_P^3 Z_P^3. \end{cases}$$

Ainsi pour calculer le double d'un point P de la courbe elliptique, il n'y a pas d'inversion et nous devons effectuer 7 multiplications et 5 élévations au carré dans $\mathbb{F}_p$, soit donc $7M + 5S$.
Les coordonnées du point $R = P + Q$ en cordonnées projectives sont données par les formules suivantes :

$$\begin{cases} X_R = (X_Q Z_P - X_P Z_Q)\, C, \\ Y_R = (Y_Q Z_P - Y_P Z_Q)\left((X_Q Z_P - X_P Z_Q)^2 X_P Z_Q - C\right) - (X_Q Z_P - X_P Z_Q)^3 Y_P Z_P, \\ Z_R = (X_Q Z_P - X_P Z_Q)^3 Z_P Z_Q. \end{cases}$$

$$\text{où } C = \left((Y_Q Z_P - Y_P Z_Q)^2 Z_P Z_Q - (X_Q Z_P - X_P Z_Q)^3 - 2(X_Q Z_P - X_P Z_Q)^2 X_P Z_Q\right).$$

De même, le calcul de ce point R ne fait pas intervenir l'inversion et requiert 12 multiplications et 2 élévations au carré dans $\mathbb{F}_p$, soit $12M + 2S$.

1.2.2 Les coordonnées Jacobiennes

En coordonnées Jacobiennes, le point P est représenté par le triplet $(X_P : Y_P : Z_P)$ correspondant au point affine $(X_P/Z_P^2 : Y_P/Z_P^3)$.
Ces coordonnées sont homogènes, en effet, pour tout α non nul, le point (X_P, Y_P, Z_P) est équivalent à $(\alpha X_P : \alpha Y_P : \alpha Z_P)$.
L'équation de la courbe elliptique E devient

$$Y^2 = X^3 + aXZ^4 + bZ^6$$

De la même manière que pour les coordonnées projectives, pour le doublement de P, nous obtenons les formules suivantes :

$$\begin{cases} X_R = -2A + B^2 \\ Y_R = -8Y_P^4 + B(A - X_R) \\ Z_R = 2Y_P Z_P \end{cases}$$

$$\text{Avec } A = 4X_P Y_P^2 \text{ et } B = 3X_P^2 + aZ_P^4,$$

Cette opération de doublement d'un point de la courbe elliptique en coordonnées jacobiennes s'effectue en 4 multiplications et 6 élévations au carré c'est-à-dire $4M + 6S$.

En coordonnées Jacobiennes, la somme R de deux points P et Q est donnée par les formules suivantes :

$$\begin{cases} X_R = -E^2 - 2AE^2 + F^2 \\ Y_R = -CE^3 + F(AE^2 - X_R) \\ Z_R = Z_P Z_Q E, \end{cases}$$

avec,

$$A = X_P, \qquad B = X_Q Z_P^2, \qquad C = Y_P Z_Q^3$$

$$D = Y_Q Z_P^3, \qquad E = B - A, \qquad F = D - C$$

L'étape d'addition de deux points sur la courbe elliptique en utilisant les coordonnées jacobiennes nécessite 12 multiplications et 4 élévations au carré $\mathbb{F}_p$, soit $12M + 4S$.
Dans la suite de cette thèse, lors du calcul de couplages sur les courbes elliptiques, nous allons considérer le système des coordonnées le plus performant. Nous avons déjà présenté le système des coordonnées affines, projectives et jacobiennes. Nous récapitulons dans le tableau suivant le coût du doublement d'un point et celui de la somme de deux points en utilisant les trois systèmes des coordonnées cités.

Système des Coordonnées	Doublement	Addition
Affines	$I + 2M + 2S$	$I + 2M + S$
Projectives	$7M + 5S$	$12M + 2S$
Jacobiennes	$4M + 6S$	$12M + 4S$

TABLE 1.1 – Coûts des différents types des coordonnées dans $\mathbb{F}_p$

1.3 Cardinalité et Torsion

1.3.1 Cardinal d'une courbe elliptique

Toute courbe elliptique définie sur un corps fini admet un nombre fini de points. Ce nombre est appelé le cardinal de la courbe E.
Une approximation du cardinal d'une courbe elliptique est donnée par le théorème de Hasse :

Théorème 1.5. *[Sil86] (**Théorème de Hasse**).*
Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_p$. Le cardinal de E noté $\sharp E$, vérifie

$$p + 1 - 2\sqrt{p} \leq \sharp E \leq p + 1 + 2\sqrt{p}$$

$$\textit{Ou encore } \mid p + 1 - \sharp E(\mathbb{F}_p) \mid \leq 2\sqrt{p}$$

Cette formule de cardinal peut être ré-écrite en utilisant la trace du Frobenius sur la courbe elliptique. Nous rappelons alors la définition du Frobenius.

Définition 1.7. *Le Frobenius π_p est une application qui va de la courbe elliptique $E(\mathbb{F}_q)$, avec $q = p^s$, d'équation $y^2 = x^3 + ax + b$ vers la courbe elliptique $E^{(p)}(\mathbb{F}_q)$ d'équation $y^2 = x^3 + a^p x + b^p$*

donné par

$$\begin{aligned} \pi_p : E(\overline{\mathbb{F}_q}) &\longrightarrow E^{(p)}(\overline{\mathbb{F}_q}) \\ (x_P, y_P) &\longmapsto (x_P^p, y_P^p) \\ P_\infty &\longmapsto P_\infty \end{aligned}$$

Le Frobenius est un morphisme de la courbe elliptique [Was03], il vérifie alors :

Propriété 1.2. *Soient P_1 et P_2 deux points d'une courbe elliptique $E(\mathbb{F}_q)$, alors nous avons :*

$$\pi_p(P_1 + P_2) = \pi_p(P_1) + \pi_p(P_2).$$

Lemme 1.1. *Soit E une courbe elliptique définie sur $\mathbb{F}_p$ et $(x, y) \in E(\overline{\mathbb{F}_p})$, alors nous avons les assertions suivantes :*

1. *$\pi_p(x, y) \in E(\overline{\mathbb{F}_p})$*
2. *$(x, y) \in E(\mathbb{F}_p)$ si et seulement si $\pi_p(x, y) = (x, y)$*
3. *π_p est un endomorphisme.*

Le Frobenius induit donc un endomorphisme sur la courbe elliptique $E(\mathbb{F}_p)$ dont nous pouvons calculer le polynôme caractéristique [CFA+05]. Ce polynôme est l'endomorphisme nul sur la courbe elliptique. Il est défini par $X^2 - tX + d$, où t est la trace du Frobenius et d son déterminant.

Théorème 1.6. *Soit E une courbe elliptique définie sur $\mathbb{F}_p$ et t la trace du Frobenius sur E. Le cardinal de E est donné par la formule suivante*

$$\sharp E(\mathbb{F}_p) = p + 1 - t.$$

Remarque 1.5. *Nous pouvons donc donner le théorème de Hasse sous la forme $\mid t \mid \leqslant 2\sqrt{p}$.*

Théorème 1.7. *[Was03] Pour tout t tel que $\mid t \mid \leqslant 2\sqrt{p}$, il existe au moins une courbe elliptique $E(\mathbb{F}_p)$ de cardinal $p + 1 - t$.*

Il existe des algorithmes permettant de calculer le cardinal d'une courbe elliptique donnée, tel que l'algorithme de Schoof [R.S95] et ses variantes.

1.3.2 Sous groupe de torsion

Soit E une courbe elliptique définie sur $\mathbb{F}_p$ et r un entier. Nous définissons la notion de points de $r-$torsion de la courbe elliptique définie sur la clôture algébrique de $\mathbb{F}_p$ comme suit :

Définition 1.8. *Un point P de la courbe elliptique $E(\overline{\mathbb{F}_p})$ est dit un point de $r-$torsion s'il vérifie la condition $[r]P = P_\infty$.*

Cette définition ainsi que les propriétés des points de $r-$torsion sont décrites dans [Sil86](Chapitre 3, Sections 4 et 6).
Remarquons que l'ensemble $E[r]$ admet la structure de groupe. En effet c'est un sous groupe de $(E(\overline{\mathbb{F}_p}), +)$.

Exemple 1.4. *Soit E la courbe elliptique d'équation $y^2 = x^3 + 1$ sur $\mathbb{F}_{257}$. Cette courbe est de cardinal $258 = 2 \times 129$. La courbe admet donc deux sous groupes de torsion non triviaux, un d'ordre 2 et un d'ordre 129. Le point $P = (8, 16)$ est un point de E de $129-$torsion et $Q = (-1, 0)$ est un point de $2-$torsion.*

Remarque 1.6. *En utilisant les formules données dans la Section 1.1.3, nous pouvons décrire les coordonnées affines de $[r]P$ comme une fraction de polynômes en les coordonnées de P. Le point $[r]P$ est le point à l'infini si son dénominateur est nul. Or, nous savons qu'un polynôme à coefficients dans un corps fini $\mathbb{F}_p$ n'a pas forcement toutes ses racines dans $\mathbb{F}_p$. C'est pour cela que nous devons considérer la courbe elliptique définie sur la clôture algébrique de $\mathbb{F}_p$ pour décrire tous les points de $r-$torsion. Nous avons alors :*

$$E[r] = \{P \in E(\overline{\mathbb{F}_p}), \ \ tel\ que\ [r]P = P_\infty\}$$

Pour plus de détails sur la structure des points de $r-$torsion, nous renvoyons le lecteur au [Sil86, Chapitre 3].

Théorème 1.8. *Soit E une courbe elliptique définie sur le corps premier $\mathbb{F}_p$ de caractéristique p et soit r un entier positif non nul.*
- Si p ne divise pas r alors :

$$E[r] \simeq \mathbb{Z}_r \times \mathbb{Z}_r$$

-Si p divise r alors r s'écrit sous la forme $r = p^\alpha n$ avec p ne divisant pas n, et on a

$$E[r] \simeq \mathbb{Z}_n \times \mathbb{Z}_n \ ou\ E[r] \simeq \mathbb{Z}_r \times \mathbb{Z}_n$$

Corollaire 1.1. *D'après le théorème précédant, le Théorème 1.8, nous constatons que si $r \wedge p = 1$ alors $\sharp E[r] = r^2$.*

Remarque 1.7. *Il est simple de remarquer que les points de $129-$torsion de la courbe elliptique donnée dans l'exemple 1.4 ne sont pas tous dans $E(\mathbb{F}_{257})$ car $16641 = 129 \times 129 > 257 + \sqrt{257} + 1$.*

1.3.3 Degré de plongement d'une courbe elliptique

Nous avons défini les points de $r-$torsion d'une courbe elliptique. Pour connaître l'ensemble dans lequel vivent ces points, nous définissons le degré de plongement de la courbe elliptique [BSS05, Chapitre IX.5].

Définition 1.9. *Soit E une courbe elliptique définie sur $\mathbb{F}_p$ et r un diviseur premier de $\sharp E(\mathbb{F}_p)$. Le degré de plongement d'une telle courbe E relativement à r est le plus petit entier k tel que r divise $p^k - 1$. Autrement dit, k est l'ordre de p dans $\mathbb{F}_r$.*

Ce paramètre permet de savoir dans quel groupe sont les points de r-torsion comme le montre le théorème suivant.

Théorème 1.9. *[BK98] Soit $E(\mathbb{F}_p)$ une courbe elliptique définie sur un corps fini $\mathbb{F}_p$, et r un diviseur de $\sharp E$. Si r est premier avec p, et si r ne divise pas $(p-1)$ alors $E[r] \subset E(\mathbb{F}_{p^k})$ où k est le degré de plongement de E relativement à r et $E[r]$ n'est pas inclus dans une extension de $\mathbb{F}_p$ de plus petit degré.*

Remarque 1.8. *Le degré de plongement d'une courbe elliptique est un paramètre spécifique à toute courbe et à un diviseur r de son cardinal.*

Exemple 1.5. *Reprenons la courbe elliptique de l'Exemple 1.4.*
Il est facile de voir que le sous groupe de $2-$torsion est de degré de plongement $k = 1$ puisque 2 divise $257 - 1 = 256$.
Par contre, le sous groupe de $129-$torsion admet un degré de plongement égal à 2. En effet, nous pouvons vérifier que 129 divise $(257^2 - 1)$ mais pas $257 - 1$.

1.3.4 Twist d'une courbe elliptique

La notion de twist ou tordue d'une courbe elliptique est une courbe elliptique telle que nous pouvons construire un isomorphisme entre ces deux courbes sur une extension du corps de base. Dans cette partie, nous n'allons donner que les éléments dont nous aurons besoin dans la suite de cette thèse pour le calcul des couplages. La théorie complète des courbes tordues est détaillée dans [Sil86, Chapitre X, Sections **2** et **5**].

Définition 1.10. *Soient E et E' deux courbes elliptiques définies sur $\mathbb{F}_q$. On dit que E' est une courbe tordue, ou twist, de degré d de E s'il existe un isomorphisme Ψ_d défini sur $\mathbb{F}_{q^d}$ (et pas sur une extension plus petite) de E' dans E. Nous notons :*

$$\Psi_d : E'(\mathbb{F}_{q^d}) \longrightarrow E(\mathbb{F}_{q^d})$$

Il n'existe qu'un nombre limité de twists de courbe elliptique. Le Théorème 1.10 donne la classification des différents twists possibles [Sil86].

Théorème 1.10. *Soit E une courbe elliptique définie sur un corps fini, les degrés possibles de d sont* 2, 3, 4 *et* 6.

En pratique, dans notre cas d'étude des couplages sur les courbes elliptiques, nous considérons que la courbe E est définie sur l'extension $\mathbb{F}_{p^{k/d}}$ qui est un sous corps de $\mathbb{F}_{p^k}$, pour d diviseur de k. Une twist E' est également définie sur $\mathbb{F}_{p^{k/d}}$. Nous donnons maintenant l'équation d'une courbe tordue de E pour chaque degré d.

— $\mathbf{d = 2}$: Soit $\zeta \in \mathbb{F}_{p^{k/2}}$ tel que le polynôme $X^2 - \zeta$ soit irréductible dans $\mathbb{F}_{p^{k/2}}$. L'équation de la twist E' de E définie sur $\mathbb{F}_{p^{k/2}}$ est :

$$\zeta y^2 = x^3 + ax + b.$$

L'isomorphisme Ψ_2 est défini par :

$$\begin{array}{rcl} \Psi_2 : E'\left(\mathbb{F}_{p^k}\right) & \to & E(\mathbb{F}_{p^k}) \\ (x, y) & \mapsto & (x, y\zeta^{1/2}) \end{array}$$

— $\mathbf{d = 4}$: La courbe elliptique E admet une twist de degré 4 si et seulement si E est définie sur $\mathbb{F}_{p^{k/4}}$, avec $p^{k/4} \equiv 1(4)$, par l'équation $y^2 = x^3 + ax$ (autrement dit $b = 0$). Soit $\zeta \in \mathbb{F}_{p^{k/4}}$ tel que le polynôme $X^4 - \zeta$ soit irréductible dans $\mathbb{F}_{p^{k/4}}$. E' est définie sur $\mathbb{F}_{p^{k/4}}$ par l'équation

$$y^2 = x^3 + \frac{a}{\zeta}x.$$

L'isomorphisme Ψ_4 est :

$$\begin{array}{rcl} \Psi_4 : E'\left(\mathbb{F}_{p^k}\right) & \to & E(\mathbb{F}_{p^k}) \\ (x, y) & \mapsto & (x\zeta^{1/2}, y\zeta^{3/4}) \end{array}$$

— $\mathbf{d = 3}$: La courbe elliptique E admet une twist de degré 3 si et seulement si E est définie sur $\mathbb{F}_{p^{k/3}}$, avec $p^{k/3} \equiv 1(3)$, par l'équation $y^2 = x^3 + b$, (autrement dit $a = 0$). Soit $\zeta \in \mathbb{F}_{p^{k/3}}$ tel que le polynôme $X^3 - \zeta$ soit irréductible dans $\mathbb{F}_{p^{k/3}}$. E' est définie sur $\mathbb{F}_{p^{k/3}}$ par l'équation :

$$y^2 = x^3 + \frac{b}{\zeta}.$$

L'isomorphisme Ψ_3 est :

$$\begin{aligned} \Psi_3 : E'\left(\mathbb{F}_{p^k}\right) &\rightarrow E(\mathbb{F}_{p^k}) \\ (x,y) &\mapsto (x\zeta^{1/3}, y\zeta^{1/2}) \end{aligned}$$

— $\mathbf{d = 6}$, La courbe elliptique E admet une twist de degré 6 si et seulement si E est définie sur $\mathbb{F}_{p^{k/6}}$ par l'équation $y^2 = x^3 + b$, (autrement dit si et seulement si $a = 0$). Soit $\zeta \in \mathbb{F}_{p^{k/6}}$ tel que le polynôme $X^6 - \zeta$ soit irréductible dans $\mathbb{F}_{p^{k/6}}$. E' est définie par l'équation

$$y^2 = x^3 + \frac{b}{\zeta}.$$

L'isomorphisme Ψ_6 est alors défini par :

$$\begin{aligned} \Psi_6 : E'\left(\mathbb{F}_{p^k}\right) &\rightarrow E(\mathbb{F}_{p^k}) \\ (x,y) &\mapsto (x\zeta^{1/3}, y\zeta^{1/2}) \end{aligned}$$

Remarque 1.9. *L'utilisation de la courbe tordue lors de calcul des couplages est avantageuse. Nous reviendrons avec plus de détails sur le rôle d'optimisation des twists dans le calcul des couplages dans le Chapitre 2.*

Le $j-$invariant de la courbe elliptique est un moyen très simple de vérifier si les deux courbes E et E' sont tordues l'une de l'autre [Sil86]. En effet :

Propriété 1.3. *Deux courbes elliptiques E et E' sont tordues l'une de l'autre si et seulement si elles ont le même $j-$invariant.*

1.4 Le Problème du Logarithme discret

La sécurité de plusieurs protocoles cryptographiques, tels que l'échange de clés de Diffie-Hellman [DH76], repose essentiellement sur la difficulté du problème de logarithme discret sur des groupes utilisés en cryptographie. Nous nous intéressons dans cette section au problème de logarithme discret dans les groupes intervenant en cryptographie à base de couplages.
La sécurité des couplages repose sur la difficulté de résoudre le problème de logarithme discret sur les courbes elliptiques $E(\mathbb{F}_p)$ d'une part et d'autre part sur le corps fini $\mathbb{F}_{p^k}$.
Nous présentons dans le Paragraphe 1.4.1 une méthode de résolution du problème de logarithme discret sur une courbe elliptique $E(\mathbb{F}_p)$ dite la méthode de **Pollard Rho**. Puis, dans le deuxième Paragraphe 1.4.2, nous présentons la méthode de **calcul d'indice** pour le calcul de logarithme discret sur les corps finis de type $\mathbb{F}_{p^k}$.

1.4.1 Pollard Rho

Rappelons que le problème de logarithme discret sur les courbes elliptiques est le suivant.

Définition 1.11. *Soit G un groupe cyclique noté additivement, P un générateur de G et Q un élément de G, alors il existe un entier α tel que*

$$Q = \alpha \times P.$$

Le problème de logarithme discret consiste à retrouver α à partir des données de P et de Q.

Le principe de la méthode de Pollard-ρ, proposée par Pollard en 1978 et détaillée dans [Pol78], est décrit ci-dessous.
Soit r le cardinal du groupe cyclique G et $f : G \longrightarrow G$ une fonction qui sert à construire une marche aléatoire sur G. Elle peut être définie comme suit :

$$\begin{cases} R_0 \in G \\ \forall i \in \mathbb{N}, R_{i+1} = f(R_i) \end{cases}$$

G est un ensemble fini, alors, il existe deux entiers μ et τ vérifiant $R_i = R_{i+\tau}$, pour tout $i \geqslant \mu$. Notons que μ est appelé la pré-période et τ la période de f. La représentation graphique de la suite R_i est formée d'une première suite de taille μ, suivie d'un cycle de taille τ, d'où la forme ρ qui a donné son nom à cette méthode [Pol78]. Il est démontré dans [Har60] que τ et μ valent environ $\sqrt{\frac{\pi r}{8}}$ si f a un bon comportement aléatoire.
Le but de cette marche aléatoire est de trouver une collision, ce qui se traduit par trouver des entiers i et j distincts tels que $R_i = R_j$. Cela peut se faire à l'aide des algorithmes de recherche de cycles, par exemple nous citons celui de Floyd détaillé dans [CFA+05, Chapitre 19].
Soit $R_i = u_iP + v_iQ$ avec P et Q les deux points pour lesquels nous souhaitons résoudre le logarithme discret. Une fois que nous obtenons une collision,

$$\begin{aligned} R_i = R_j &\Leftrightarrow u_iP + v_iQ = u_jP + v_jQ \\ &\Leftrightarrow (u_i + \alpha v_i)P = (u_j + \alpha v_j)P \\ &\Leftrightarrow (u_i + \alpha v_i) \equiv (u_j + \alpha v_j) \mod r \\ &\Leftrightarrow u_i - u_j \equiv \alpha(v_j - v_i) \mod r \end{aligned}$$

Ainsi, $\alpha \equiv (u_i - u_j)/(v_j - v_i) \mod r$. Ce qui résout le problème de logarithme discret sur les groupes additifs. Notons que la complexité de la méthode de Pollard Rho est de l'ordre de $O(\sqrt{r})$. Pour plus de détails sur cette méthode nous renvoyons le lecteur à [BLS11] et [Pol78].

Remarque 1.10. *En pratique, il est très difficile de résoudre le problème de logarithme discret sur les courbes elliptiques si r est suffisamment grand car il n'existe pas à ce jour de meilleur algorithme que Pollard rho dans le cas général.*

1.4.2 Calcul d'indice

Le calcul d'indice est, à ce jour, la méthode la plus efficace pour calculer le logarithme discret sur les corps finis utilisés en cryptographie.
Posons G un sous groupe multiplicatif d'ordre r de $\mathbb{F}^*_{p^k}$ et g un générateur de G. Dans ce cas, le problème de logarithme discret consiste à trouver α tel que $x = g^\alpha$, pour x un élément de G. α est alors noté $\log_g(x)$.
Cette méthode est de complexité sous exponentielle donnée par :

$$L_p(1/3, c) = \exp\left(c(\log(p))^{1/3}(\log(\log(p)))^{2/3}\right)$$

où c est une constante.
Nous décrivons le principe de la méthode du calcul d'indice pour les sous groupes multiplicatifs d'un corps fini de la forme $\mathbb{F}^*_p$. Nous commençons par construire une base de facteurs qu'on note $B = \{p_1, p_2, \ldots, p_n\}$. Cette base est composées des petits nombres premiers. Puis nous tirons aléatoirement des entiers h et calculons $g^h \mod p$. Nous testons si la décomposition en facteurs

premiers du représentant entier du résultat n'admet que des éléments de la base B. Si c'est bien le cas, nous stockons la relation :

$$g^h = \prod_i p_i^{\alpha_i} \mod p,$$

qui est équivalente à la relation :

$$h \equiv \sum_i \alpha_i \times \log_g(p_i) \mod r,$$

où, nous connaissons h et les α_i, nous cherchons alors les $\log_g(p_i)$. Nous réitérons ce procédé jusqu'à obtenir n relations indépendantes qui forment un système linéaire en les $\log_g(p_i)$, $i = \{1, \dots n\}$. La résolution de ce système linéaire nous permet d'obtenir le logarithme discret des éléments de la base B.
Pour terminer la résolution du problème de logarithme discret sur un sous groupe multiplicatif $\mathbb{F}_p^*$, nous cherchons aléatoirement un entier l vérifiant une relation du type :

$$g^l x \equiv \prod_i p_i^{\alpha_i} \mod r.$$

Sachant que $x = g^\alpha$, cette relation permet de calculer α :

$$\alpha = \sum_i \alpha_i \times \log_g(p_i) - l \mod r.$$

Dans ce cas, la constante c intervenant dans la complexité vaut 64/9. On peut toutefois obtenir de meilleures constantes (48/9 voire 32/9) dans le cas de $\mathbb{F}_{p^k}^*$ [KB16]. Pour plus de détails sur cette méthode, nous renvoyons le lecteur aux références [JL07], [KB16].

1.5 Construction des Courbes elliptiques par Multiplication Complexe

Dans cette section, nous rappelons les méthodes utilisées pour construire des courbes elliptiques adaptées au calcul des couplages qui sont particulières. En fait, nous choisissons une courbe elliptique définie $\mathbb{F}_p$ dont le cardinal possède un grand facteur premier noté r. Ainsi que le degré de plongement k doit vérifier $6 < k \leqslant 32$.
Une méthode efficace qui nous permet d'obtenir des courbes satisfaisants ces conditions est la **Multiplication Complexe** (Voir chapitre 10 de [Was03]).
La construction d'une courbe elliptique en utilisant cette méthode se décompose en deux étapes. La première étape consiste à la résolution du système suivant :

$$\begin{cases} r & | & p+1-t \\ r & | & p^k - 1 \\ Dy^2 & = & 4p - t^2 \end{cases} \tag{1.4}$$

Les inconnues de ce système sont p, définissant le corps de base, r, le facteur premier du cardinal de E, t la trace de Frobenius sur $E(\mathbb{F}_p)$ et l'entier D. Cet entier D est appelé le discriminant de la multiplication complexe et il doit être choisi petit pour que la méthode soit efficace.
La deuxième étape consiste à la construction d'une courbe elliptique admettant les paramètres p, r et t trouvés lors de la résolution du Système (1.4). Il existe plusieurs méthodes permettant la résolution de ce système qui sont décrites en détails dans [FST10]. L'idée générale de ces méthodes

est de fixer deux paramètres, puis, déduire le deux restantes. Voici deux exemples d'application de cette méthode qui seront utilisés dans la suite.

1. La première méthode permet de construire des courbes elliptiques avec un degré de plongement fixé à l'avance. Initiée par Miyaji, Nakabayashi et Takano [MNT00], cette méthode a été généralisée par Barreto et Naehrig (pour plus de détails sur la construction voir [BLS02]) pour construire des courbes de degré de plongement $k = 12$. Ces courbes sont obtenues à l'aide de la paramétrisation suivante :

$$\begin{cases} k = 12 \\ r = 36u^4 + 36u^3 + 18u^2 + 6u + 1 \\ p = 36u^4 + 36u^3 + 24u^2 + 6u + 1 \\ t = 6u^2 + 1. \end{cases}$$

   Pour déterminer ces paramètres, nous cherchons des entiers u tel que p et r soient premiers et vérifient un niveau de sécurité précis.
   Cette catégorie de courbes elliptiques est donnée par l'équation de la forme :

$$E : y^2 = x^3 + b \text{ avec } b \neq 0 \text{ dans } \mathbb{F}_p,$$

   et possèdent donc une twist de degré 6.

2. La deuxième méthode consiste à construire des courbes dont le degré de plongement k est arbitraire. Il s'agit de la méthode de **Cocks-Pinch** [CP01]. Pour cette méthode, le paramètre r est arbitraire, mais satisfait l'approximation $\log(p)/\log(r) \simeq 2$.
   Cette méthode a été généralisée par Brezing-Weng dans [BW05] pour obtenir $\log(p)/\log(r) < 2$. Dans ce contexte, une famille de courbes elliptiques de degré de plongement égal à 16 a été générée par la méthode de Brezing-Weng. Sa paramétrisation est donnée comme suit :

$$\begin{cases} t &= 1/35\left(2u^5 + 41u + 35\right) \\ r &= u^8 + 48u^4 + 625 \\ p &= \frac{1}{980}\left(u^{10} + 2u^9 + 5u^8 + 48u^6 + 152u^5 + 240u^4 + 625u^2 + 2398u + 3125\right) \end{cases}$$

1.6 Fonctions rationnelles et diviseurs

Pour définir la notion de couplages, nous avons besoin d'introduire les notions de fonctions rationnelles et diviseurs.

1.6.1 Les fonctions rationnelles

Définition 1.12. *Une fonction rationnelle f sur $E(\mathbb{F}_p)$ est une fonction de la forme*

$$\begin{array}{rcl} f : E(\mathbb{F}_p) & \longrightarrow & \mathbb{F}_p \\ P & \longmapsto & f(P) \end{array}$$

avec $f(P)$ est de la forme $f(P) = \frac{u(P)}{v(P)}$ avec u, $v \in \mathbb{F}_p[x, y]$.

Exemple 1.6. *Soit E la courbe elliptique définie sur $\mathbb{F}_p$ d'équation $y^2 = x^3 + ax + b$. L'application suivante est une fonction rationnelle sur $E(\mathbb{F}_p)$*

$$\begin{array}{rcl} f : E(\mathbb{F}_p) & \to & \mathbb{F}_p \\ (x, y) & \mapsto & \dfrac{(x - \alpha_1)^2(\alpha_2 - x)}{(y - \beta_1)(\alpha_3 - x)^3} \end{array}$$

où α_i, β_i sont des éléments non nuls de $\mathbb{F}_p$. Nous avons donc

— $u(x, y) = (x - \alpha_1)^2(\alpha_2 - x)$

— $v(x, y) = (y - \beta_1)(\alpha_3 - x)^3$.

Dans la suite, nous aurons besoin des définitions suivantes.

Définition 1.13. *: Soit $P \neq P_\infty$ un point d'une courbe elliptique définie sur $\mathbb{F}_p$ et f une fonction rationnelle. On dit que P est un zéro de f si $f(P) = 0$ et on dit que P est un pôle de f si la fonction f n'est pas définie en P, autrement dit si $u(P) \neq 0$ et $v(P) = 0$.*

Définition 1.14. *: Soit P un point de $E(\mathbb{F}_p)$. Il existe une fonction u_P, appelée uniformisante, qui s'annule en P et telle que pour toute fonction rationnelle f sur $E(\mathbb{F}_p)$, il existe un entier α tel que $f = u_P^\alpha \times g$ où g une fonction rationnelle dont P n'est ni un zéro ni un pôle. L'entier α, noté $ord_P(f)$, est appelé l'ordre de f au point P.*

Remarque 1.11. *Il est facile de voir que l'ordre de $f = u/v$ au point $P \in E(\mathbb{F}_p)$ est :*

$$ord_P(f) = ord_P(u) - ord_P(v).$$

— *Si $ord_P(f) < 0$, le point P est un pôle pour la fonction f de multiplicité $-ord_P(f)$.*

— *Sinon, si $ord_P(f) > 0$, le point P est un zéro pour la fonction f de multiplicité $ord_P(f)$.*

Théorème 1.11. *Une fonction a autant de zéros que de pôles comptés avec leurs multiplicités.*

Corollaire 1.2. *L'ordre de f au point P_∞ est égal à :*

$$ord_{P_\infty}(f) = \sum_{P \text{ zéro de } v} ord_P(v) - \sum_{P \text{ zéro de } u} ord_P(u).$$

Exemple 1.7. *Reprenons la fonction rationnelle donnée dans l'Exemple 1.6. Soit $P = (\alpha_1, \beta_1)$, $Q = (\alpha_3, y_Q)$ avec $y_Q \neq \beta_1$ et $R = (x_R, y_R)$ avec $x_R \neq \{\alpha_1, \alpha_2, \alpha_3\}$ et $y_R \neq \beta_1$. Nous supposons que la courbe E n'admet pas de point dont l'abscisse est α_2, alors il est simple de voir que :*

$ord_P(f) = 2 - 1 = 1,$
$ord_Q(f) = 0 - 3 = -3,$
$ord_R(f) = 0 - 0 = 0,$
$ord_{P_\infty}(f) = (3 + 3 + 1 + 1) - (2 + 2) = 4.$

1.6.2 Les diviseurs

La définition, ainsi que la construction des couplages, est basée essentiellement sur l'existence des diviseurs associées à toute fonction rationnelle sur une courbe elliptique. Dans cette section, nous définissons la notion de diviseurs et nous donnons leurs importantes propriétés.

Définition 1.15. *: Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_p$. Un diviseur de E est une somme formelle de points :*

$$D = \sum_{P \in E, n_P \in \mathbb{Z}} n_P(P),$$

pour laquelle un nombre fini de n_p sont non nuls.
Pour un diviseur D, nous définissons :

— ***Le support du diviseur*** D *est* $supp(D) := \{P \in E, n_P \neq 0\}$.

— ***L'ordre du diviseur*** D *en un point* P *est* $ord_P(D) := n_P$,

— ***Le degré du diviseur*** D *est* $deg(D) := \sum_{P \in E} n_P$.

Remarque 1.12. *Pour faire la différence entre une somme formelle pour construire un diviseur et une somme réelle de points, nous noterons (P) le diviseur associé au point P et dont nous ne ferons que des sommes formelles. Par conséquent, $n_P(P)$ est le diviseur représentant n_P fois le point P. Nous noterons $[n_P]P$ la multiplication scalaire du point P par n_P.*

Propriété 1.4. *L'ensemble des diviseurs est un groupe additif pour la loi définie comme suit : pour D et D' deux diviseurs d'une même courbe $E(\mathbb{F}_p)$:*

$$D = \sum_{P \in E, n_P \in \mathbb{Z}} n_P(P) \text{ et } D' = \sum_{P \in E, n'_P \in \mathbb{Z}} n'_P(P),$$

on pose

$$D + D' = \sum_{P \in E, (n_P, n'_P) \in \mathbb{Z}^2} (n_P + n'_P)(P).$$

Pour définir les couplages, nous aurons également besoin de la notion de diviseurs associés à une fonction rationnelle.

Définition 1.16. *Soit f une fonction rationnelle sur la courbe elliptique $E(\mathbb{F}_p)$. Le diviseur de f, noté par $Div(f)$, est défini par :*

$$Div(f) := \sum_{P \in E} ord_P(f)(P).$$

Exemple 1.8. *Revenons à la fonction rationnelle définie dans l'Exemple 1.6 par*

$$f(x, y) = \frac{(x - \alpha_1)^2(\alpha_2 - x)}{(y - \beta_1)(\alpha_3 - x)^3}$$

Nous pouvons construire le diviseur associé à la fonction f comme suit :

$$Div(f) = (P) + (-P) - 3(Q) - 3(-Q) + 4P_\infty$$

Propriété 1.5. *Soient f_1 et f_2 deux fonctions rationnelles. Alors,*

— $Div(f_1 \times f_2) = Div(f_1) + Div(f_2)$

— $Div\left(\frac{f_1}{f_2}\right) = Div(f_1) - Div(f_2)$.

Théorème 1.12. *Soient f_1, f_2 deux fonctions rationnelles définies sur une courbe $E(\mathbb{F}_p)$. Alors :*

$$div(f_1) = div(f_2) \text{ si et seulement si } f_1 = cf_2, \text{ où } c \text{ est une constante quelconque .}$$

Corollaire 1.3. *Soit f une fonction rationnelle, si $div(f) = 0$, alors f est une fonction constante. Bien sûr la réciproque est vraie : $div(c) = 0$.*

Définition 1.17. *Un diviseur principal sur une courbe elliptique $E(\mathbb{F}_p)$ est un diviseur D tel qu'il existe une fonction rationnelle f qui vérifie $D = Div(f) = \sum ord_P(f)(P)$. Nous notons D_{prin} leur ensemble.*

Propriété 1.6. *Un diviseur $D = \sum n_P(P)$ est principal si et seulement si :*

— $deg(D) = 0$,

— $\sum_{n_P \in \mathbb{Z}} [n_P]P = P_\infty$.

Nous pouvons vérifier que le diviseur associé à la fonction rationnelle citée dans l'Exemple 1.6 est un diviseur principal.

1.7 Couplages sur les courbes elliptiques

1.7.1 Introduction aux couplages

Les couplages sont une notion mathématique très en vogue depuis une quinzaine d'année en cryptographie asymétrique. Les premiers à être utilisés en cryptographie (dans les années 90) sont les couplages de Weil et de Tate existants depuis longtemps dans la littérature.

1.7.2 Généralités sur les couplages

Définition 1.18. *Soient G_1, G_2 et G_3 des groupes abéliens de même ordre r. Les groupes G_1 et G_2 sont notés additivement et le groupe G_3 multiplicativement.*
Un couplage est une application

$$e : G_1 \times G_2 \to G_3$$

qui vérifie la bilinéarité et la non dégénérescence qui sont définies comme suit :

- ***Non dégénérescence :***
  - *Pour tout $P \in G_1$, $\exists\ Q \in G_2$ tel que $e(P, Q) \neq 1$,*
  - *Pour tout $Q \in G_2$, $\exists\ P \in G_1$ tel que $e(P, Q) \neq 1$*
- ***Bilinéarité :*** *Pour tout $P_1, P_2 \in G_1$ et pour tout $Q_1, Q_2 \in G_2$, nous avons :*
  - $e(P_1 + P_2, Q_1) = e(P_1, Q_1) \cdot e(P_2, Q_1)$,
  - *et* $e(P_1, Q_1 + Q_2) = e(P_1, Q_1) \cdot e(P_1, Q_2)$.

Si $G_1 = G_2$, le couplage est dit symétrique.

Propriétés 1.1. *Soit e un couplage, $P \in G_1$ et $Q \in G_2$. P_∞^1 l'élément neutre de G_1 et P_∞^2 celui de G_2. Pour tous $P \in G_1$ et $Q \in G_2$,*

1. $e(P, P_\infty^2) = e(P_\infty^1, Q) = 1$,
2. $e(-P, Q) = e(P, Q)^{-1} = e(P, -Q)$,
3. $\forall\ i \in \mathbb{Z}$, $\forall j \in \mathbb{Z}$, $e([j]P, [i]Q) = e(P, Q)^{ij} = e([i]P, [j]Q)$.

Remarque 1.13. *Si on élève un couplage à une puissance non multiple de r, cela reste un couplage.*

1.7.3 Rôle des couplages en Cryptographie

Les couplages ont un rôle très important en cryptographie. Ce rôle peut être destructif, simplificatif ou créatif. Dans cette section, nous allons rappeler ces différents rôles des couplages.

Rôle destructif

Nous avons déjà vu dans la Section 1.4 que le problème de logarithme discret sur les courbes elliptiques est plus difficile à résoudre que sur les corps finis car on dispose d'un algorithme de complexité sous-exponentielle dans ce dernier cas.
Soient P et Q deux éléments de G_1 (un groupe cyclique). Il existe un entier α tel que $Q = \alpha P$. Le problème du logarithme discret sur G_1 consiste à retrouver α. Supposons qu'on ait un couplage symétrique

$$e : G_1 \times G_1 \to G_3$$

où G_3 désigne le groupe multiplicatif de $\mathbb{F}_{p^k}$. En calculant le couplage entre les points P et Q, nous obtenons,

$$e(P,Q) = e(P,[\alpha]P) = e(P,P)^{\alpha}$$

Notons qu'avec $h = e(P,Q)$ et $g = e(P,P)$, où $g \neq 1$, le nouveau problème consiste alors à la recherche de l'entier α tel que $h = g^{\alpha}$.
La propriété de la bilinéarité des couplages nous a permis alors de transférer le problème de logarithme discret dans G_1 vers un problème de logarithme discret dans G_3. Dans le cas des couplages de Weil et Tate, G_1 est une courbe elliptique et G_3 est un corps fini sur lequel on dispose d'algorithmes efficaces pour peu que le corps fini G_3 ne soit pas trop gros.

Rôle de simplification

Les couplages ont aussi un rôle de simplification des protocoles existants. Nous rappelons en premier lieu le protocole d'échange de clé de Diffie Hellman [DH76] à 3. Nous considérons les trois protagonistes Alice, Bob et Charlie qui veulent communiquer de manière sécurisée. Ils veulent partager une clé sécrète commune afin de pouvoir chiffrer et déchiffrer leurs messages. Pour le partage de la clé sécrète, Alice, Bob et Charlie doivent être d'accord sur un groupe G cyclique d'ordre r et de générateur P. Puis,

1. Ils choisissent leurs clés sécrètes. Alice choisit sa clé notée a, Bob choisit b et Charlie c. $a, b, c \in \{1, 2, \ldots, r-1\}$.
2. Premier tour d'échange :

   Alice envoie $[a]P$ à Bob,
   Bob envoie $[b]P$ à Charlie,
   Charlie envoie $[c]P$ à Alice.
3. Deuxième tour d'échange :

   Alice envoie $[a]([c]P)$ à Bob,
   Bob envoie $[b]([a]P)$ à Charlie,
   Charlie envoie $[c]([b]P)$ à Alice.
4. Le partage de la clé sécrète : Maintenant, après ces deux tours d'échanges, les trois protagonistes partagent une même clé sécrète $[abc]P$.

L'échange de clé de Diffie Hellman nécessite alors deux tours d'échange d'information.
En 2001, Antoine Joux [Jou00] a proposé un schéma plus simple d'échange de clé à partir d'un seul tour d'échange en utilisant un couplage symétrique e. Le schéma d'échange de Joux est le suivant :

1. Alice, Bob et Charlie choisissent leurs clés sécrètes, comme dans le premier cas du schéma de Diffie Hellman, respectivement a, b et c dans $\{1, 2, \ldots, r-1\}$.
2. Premier tour d'échange :

   Alice envoie $[a]P$ à Bob et à Charlie,
   Bob envoie $[b]P$ à Alice et à Charlie,
   Charlie envoie $[c]P$ à Alice et à Bob.
3. À cette étape, Alice, Bob et Charlie peuvent partager une même clé sécrète comme suit :

   Alice calcule $e([b]P,[c]P)^a$, Bob calcule $e([a]P,[c]P)^b$, Charlie calcule $e([a]P,[b]P)^c$.

   Grâce à la bilinéarité des couplages, les trois protagonistes partagent une clé $e(P,P)^{abc}$.

Ainsi, l'échange de clé tri-partie d'Antoine Joux ne nécessite qu'un tour d'échange de données au lieu de deux si nous considérons le schéma original de Diffie Hellman. Cet exemple montre bien le rôle de simplification des couplages en cryptographie.

Construction de nouveaux protocoles

Plusieurs protocoles cryptographiques ont pu être réalisés grâce aux couplages sur les courbes elliptiques. Nous pouvons citer dans ce contexte la cryptographie basée sur l'identité [BF03] et aussi les schémas de signatures courtes [BLS02].

1.7.4 Sécurité des Couplages

La sécurité des couplages sur les courbes elliptiques repose essentiellement sur la difficulté de la résolution du problème du logarithme discret sur les courbes elliptiques (G_1 et G_2) et sur les corps finis (G_3). Le niveau de sécurité est défini par le nombre d'opérations nécessaires sur le corps de base pour résoudre le problème de logarithme discret. Le niveau de sécurité est donné en bits et nous utilisons la notion de sécurité équivalente pour le définir. Par exemple, le niveau de sécurité 128 bits signifie qu'il nous faut au minimum 2^{128} opérations pour résoudre le problème du logarithme discret sur les groupes en considération.

En cryptographie basée sur les couplages, pour définir le niveau de sécurité, nous devons définir deux bornes. La première est la taille du sous groupe de la courbe elliptique dans lequel nous travaillons qui vaut $\log_2(r)$. La deuxième borne est la taille du corps fini où vivent les résultats d'un couplage c'est-à-dire la taille binaire de $\mathbb{F}_{p^k}^*$ qui vaut $\log_2(p^k)$.
Par conséquent, lorsque le niveau de sécurité augmente les deux bornes $\log_2(r)$ et $k\log_2(p)$ augmentent aussi mais avec des rythmes différents. Il est clair que la deuxième borne, $k\log_2(p)$ augmente d'une manière plus rapide que la première. Nous donnons dans le tableau les tailles de r et p^k en bits selon les recommandations du NIST [oST] pour les niveaux standards de sécurité . Il faut toutefois prendre en compte les résultats récents sur le calcul du logarithme discret sur les corps finis [KB16]. Les valeurs de la dernière colonne devraient donc être plus élevées.

Niveau de sécurité en bits	Taille de r en bits	Taille de p^k en bits
80	160	1024
128	256	3072
192	384	7680
256	512	15360

TABLE 1.2 – Niveau de sécurité selon 1.2

D'après ce tableau, pour assurer un niveau de sécurité de 80 bits par exemple, il faut que choisir p et k tels que p^k soit de taille 1024 bits. Les tailles des paramètres sont fixées les unes en fonction des autres et on note

$$\rho = \frac{\log_2(p)}{\log_2(r)}$$

Ayant la taille de r et de p, nous pouvons déduire le degré de plongement idéal de la courbe k.

Remarque 1.14. *Les courbes elliptiques pour lesquelles le paramètre ρ est proche de 1 sont favorisées pour les applications cryptographiques [FST10].*

Lors du calcul du couplage, nous nous ramenons à des opérations dans le corps de base $\mathbb{F}_p$. Cela signifie que l'implémentation efficace des couplages dépend directement du corps de base utilisé, c'est-à-dire de la taille de p en bits. Donc, pour avoir un couplage efficace, il vaut mieux bien choisir k afin d'atteindre le niveau de sécurité souhaité que d'augmenter la taille de p.
Dans le tableau suivant, nous présentons les valeurs de k pour atteindre un niveau de sécurité précis en fonction de la taille de r et de celle de p (pour plus de détails voir [FST10]). Nous avons choisi ici d'utiliser un intervalle pour la taille de p^k tenant compte des résultats de [KB16] mais ces données sont approximatives car elles n'ont pas encore été étudiées sérieusement par la communauté.

Niveau de sécurité en bits	Taille de r en bits	Taille de p^k en bits	k pour $\rho \approx 1$	k pour $\rho \approx 2$
80	160	$900 - 1280$	$6 - 8$	$2 - 4$
128	256	$3000 - 5000$	$12 - 20$	$6 - 10$
192	384	$7800 - 10000$	$20 - 26$	$10 - 13$
256	512	$14000 - 18000$	$28 - 36$	$14 - 18$

TABLE 1.3 – Niveau de sécurité

1.8 Le Calcul des couplages

Plusieurs types de couplages sont à considérer en cryptographie. Dans cette section, nous rappelons en premier lieu les premiers couplages apparus en cryptographie qui sont les couplages de Weil [Mil04], [MOV93] et Tate [FMR99a], [adv05]. Nous définissons ces deux couplages ainsi que leurs propriétés.
Ces deux couplages sont basés sur le calcul d'une fonction rationnelle évaluée en un point de la courbe elliptique. Cette fonction se calcule à l'aide de l'algorithme de Miller.

Soit E une courbe elliptique définie sur $\mathbb{F}_p$ par l'équation $y^2 = x^3 + ax + b$. Soit r un facteur premier du cardinal de E. De plus nous supposons que r^2 ne divise pas $\sharp E(\mathbb{F}_p)$ pour éviter que toute la $r-$torsion soit dans $E(\mathbb{F}_p)$. Soit :

— $G_1 = E(\mathbb{F}_p)[r]$;
— G_2 un sous groupe de $E(\mathbb{F}_{p^k})[r]$,
— $G_3 = \{\mu \in \mathbb{F}_{p^k}$ tel que $\mu^r = 1\}$: le sous groupe des racines $r-$ième de l'unité de $\mathbb{F}^*_{p^k}$.

Pour tout point R d'une courbe elliptique définie sur n'importe quelle extension de $\mathbb{F}_p$, et pour tout entier s, soit la fonction rationnelle $f_{s,R}$ qui admet le point R comme zéro d'ordre s et le point $[s]R$ comme pôle. Elle vérifie :

$$Div(f_{s,R}) = s(R) - ([s]R) - (s-1)(P_\infty).$$

Remarquons que la fonction $f_{s,R}$ est définie modulo une constante dans l'extension où le point R est défini. Nous donnons maintenant la définition des couplages de Weil et de Tate.

Définition 1.19. *Le couplage de Weil, noté e_W, est l'application*

$$\begin{array}{rcl} e_W : G_1 \times G_2 & \longrightarrow & G_3 \\ (P,Q) & \longmapsto & (-1)^r \dfrac{f_{r,P}(Q)}{f_{r,Q}(P)} \end{array}$$

Définition 1.20. *Le couplage de Tate réduit, qu'on note e_T, est l'application suivante :*

$$\begin{array}{rcl} e_T : G_1 \times G_2 & \longrightarrow & G_3 \\ (P,Q) & \longmapsto & (f_{r,P}(Q))^{\frac{p^k-1}{r}} \end{array}$$

Remarque 1.15. *L'élévation de $f_{r,P}(Q)$ à la puissance $\frac{p^k-1}{r}$ assure l'unicité de couplage entre P et Q. Le calcul de couplage de Tate original correspond au calcul de $f_{r,P}(Q)$. Les propriétés des diviseurs ont pour conséquence que $f_{r,P}(Q)$ n'est pas définie de manière unique. Cette fonction est à valeurs dans les classes d'équivalences modulo les puissances r-ièmes dans $\mathbb{F}^*_{p^k}$.*

Nous pouvons vérifier la bilinéarité des couplages de Tate et Weil [Sil86]. Pour la non dégénérescence, il suffit de prendre le point Q n'appartenant pas au sous-groupe de $E[r]$ engendré par le point P. Autrement dit, Q appartient à l'ensemble des points de $r-$torsion dans $E(\mathbb{F}_{p^k})$ privé des points de $E(\mathbb{F}_p)$.

1.8.1 L'algorithme de Miller

Les couplages de Weil et de Tate nécessitent le calcul des fonctions rationnelles $f_{r,P}$ et $f_{r,Q}$. Grâce à l'algorithme de Miller, ces fonctions sont faciles à calculer. Le calcul de cette fonction repose sur la fameuse égalité de Miller [Mil85] qui est la suivante :

$$f_{i+j,P} = f_{i,P} f_{j,P} \frac{l_{[i]P,[j]P}}{v_{[i+j]P}}$$

où

— $l_{[i]P,[j]P}$ est l'équation de la droite passant par les points $[i]P$ et $[j]P$, et donc par le point $-[i+j]P$. Son diviseur est

$$Div(l_{[i]P,[j]P}) = ([i]P) + ([j]P) + (-[i+j]P) - 3(P_\infty)$$

— $v_{[i+j]P}$ désigne l'équation la verticale à la courbe au point $[i+j]P$. Son diviseur est

$$Div(v_{[i+j]P}) = (-[i+j]P) + ([i+j]P) - 2(P_\infty)$$

Si $i = 1$ alors la fonction $f_{1,P}$ est la fonction constante et est égale à 1.
La preuve de l'égalité de Miller est basée sur l'étude des diviseurs. En effet

$$\begin{array}{rcl} Div(f_{i,P} f_{j,P} \frac{l_{[i]P,[j]P}}{v_{[i+j]P}}) & = & Div(f_{i,P}) + Div(f_{j,P}) + Div(l_{[i]P,[j]P}) - Div(v_{[i+j]P}) \\ & = & i(P) - ([i]P) - (i-1)(P_\infty) + j(P) - ([j]P) - (j-1)(P_\infty) \\ & + & ([i]P) + ([j]P) + (-[i+j]P) - 3(P_\infty) - (-[i+j]P) - ([i+j]P) + 2(P_\infty) \\ & = & (i+j)(P) - ([i+j]P) - (i+j-1)(P_\infty) \\ & = & Div(f_{i+j,P}) \end{array}$$

Exemple 1.9. *Dans cet exemple, nous calculons $f_{5,P}$ en utilisant l'égalité de Miller. Nous suivons les étapes suivantes :*

1. *Écrivons $5 = 4 + 1$, puis appliquons l'égalité de Miller. Nous obtenons :*

$$f_{5,P} = f_{1,P} \times f_{4,P} \times \frac{l_{[4]P,P}}{v_{[5P]}}$$

Comme nous avons initialisé $f_{1,P}$ à 1, nous obtenons :

$$f_{5,P} = f_{4,P} \times \frac{l_{[4]P,P}}{v_{[5P]}}.$$

2. *De nouveau, décomposons $4 = 2 + 2$, et appliquons une deuxième fois l'égalité de Miller à la fonction $f_{4,P}$ alors,*

$$f_{4,P} = f_{2,P}^2 \times \frac{l_{[2]P,[2]P}}{v_{[4P]}}.$$

D'où

$$f_{5,P} = f_{2,P}^2 \times \frac{l_{[2]P,[2]P}}{v_{[4P]}} \times \frac{l_{[4]P,P}}{v_{[5P]}}.$$

3. *Nous recommençons en écrivant $2 = 1 + 1$:*

$$f_{2,P} = f_{1,P} \times f_{1,P} \times \frac{l_{P,P}}{v_{[2P]}} = \frac{l_{P,P}}{v_{[2P]}}$$

Cela implique que :

$$f_{5,P} = \left(\frac{l_{P,P}}{v_{[2P]}}\right)^2 \times \frac{l_{[2]P,[2]P}}{v_{[4P]}} \times \frac{l_{[4]P,P}}{v_{[5P]}}.$$

Après plusieurs applications (3 exactement) de l'égalité de Miller, nous avons réussi à calculer la fonction rationnelle $f_{5,P}$. C'est une évaluation simple d'équations de droites et de tangentes. Nous remarquons bien qu'en remontant la succession d'égalité obtenues en commençant par la dernière, le calcul de $f_{5,P}$ s'effectue en parallèle du calcul de $[5]P$.

D'après cet exemple, nous remarquons que le calcul de toute fonction rationnelle $f_{r,P}$ évaluée au point Q se fait de façon analogue au schéma de l'exponentiation rapide à base de doublements et d'additions. Ce calcul s'effectue en $\lfloor \log_2(r) \rfloor$ étapes. Nous présentons dans l'algorithme suivant la méthode générale pour calculer $f_{r,P}$ en utilisant l'égalité de Miller.

Algorithm 1 : Miller [Mil85]

Require: $P \in G_1, Q \in G_2$, $r = (r_{n-1}, \ldots, \ldots r_0)$: la représentation binaire de r avec $r_{n-1} = 1$,
Ensure: $f_{r,P}(Q) \in {\mathbb{F}_{p^k}}^*$

1: $f_1 \leftarrow 1$
2: $T \leftarrow P$
3: **for** $i = n - 2$ **down to** 0 **do**
4: $\quad T \leftarrow [2]T$
5: $\quad f_1 \leftarrow f_1^2 \cdot \frac{l_1(Q)}{v_1(Q)}$
6: $\quad$ **if** $r_i = 1$ **then**
7: $\quad\quad T \leftarrow T + P$
8: $\quad\quad f_1 \leftarrow f_1 \cdot \frac{l_2(Q)}{v_2(Q)}$
9: $\quad$ **end if**
10: $\quad$ **return** f_1
11: **end for**

On a utilisé les notations suivantes :

— l_1 est l'équation de la tangente à la courbe en T.

— v_1 est l'équation de la verticale à la courbe au point $[2]T$.

— l_2 est l'équation de la droite (TP).

— v_2 est l'équation de la verticale à la courbe au point $T + P$.

Dans chaque itération de l'algorithme de Miller, nous avons deux étapes de calcul. La première appelée étape de doublement décrite dans le slignes 3 et 5. La deuxième étape est celle d'addition représentée dans l'Algorithme 1 par les lignes 6 et 8.
Nous remarquons que pour cet algorithme nous devons faire :

$l_r - 1$ étapes de doublements (où l_r désigne la longueur binaire de r),
$w_r - 1$ étapes d'additions (où w_r désigne le poids de Hamming de r).

La complexité de l'Algorithme 1 de Miller est liée à celle du calcul des équations l_1, l_2, v_1 et v_2 qui dépend du système des coordonnées choisi.
Le couplage de Weil requiert deux exécutions de l'algorithme de Miller, le calcul de $f_{r,P}(Q)$ en premier lieu puis le calcul de $f_{r,Q}(P)$. La première fonction à calculer est dite Miller Lite car l'arithmétique de la courbe elliptique est effectuée dans le corps finis $\mathbb{F}_p$. Par contre, la deuxième fonction à calculer $f_{r,Q}(P)$ est dite Miller Full vu que les équations des droites à évaluer sont dans la courbe définie sur l'extension $\mathbb{F}_{p^k}$. Une fois calculées ces deux fonctions, nous devons effectuer à la fin une inversion dans $\mathbb{F}_{p^k}$.
Le couplage de Tate ne nécessite qu'une seule fonction rationnelle à calculer et il s'agit bien de Miller Lite. Puis il faut élever le résultat à la puissance $\frac{p^k-1}{r}$. Cette étape est appelée exponentiation finale.

On peut toutefois définir des couplages qui sont en général plus facile à calculer.

1.9 Les Couplages Optimaux

Dans cette section nous rappelons les optimisations principales sur les couplages : le couplage Ate, le couplage Twisted Ate et enfin le couplage Optimal Ate.

1.9.1 Les couplages Ate et Twisted Ate

Le couplage de Ate est une optimisation du couplage de Tate introduite par Hess, Smart et Vercauteren [HSV06a]. Le principe est de réduire le nombre d'itérations dans l'algorithme de Miller. En utilisant les propriétés des diviseurs, Hess *et al.* dans [HSV06a] ont réussi à réduire ce nombre d'itérations de $\lfloor \log_2(r) \rfloor$ à $\lfloor \log_2(T) \rfloor$ et $T = t - 1$, où t est la trace du Frobenius sur $E(\mathbb{F}_p)$.
Le couplage de Ate est construit en utilisant les groupes G_1 et G_2 définis par l'intermédiaire des espaces propres du Frobenius :

— $G_1 = E[r] \cap \mathrm{Ker}(\pi_p - [1]) = E(\mathbb{F}_p)[r]$

— $G_2 = E[r] \cap \mathrm{Ker}(\pi_p - [p])$

— $G_3 = \{\mu \in \mathbb{F}_{p^k} \text{ tel que } \mu^r = 1 \}$

Définition 1.21. *Le couplage de Ate qu'on note e_A est l'application suivante :*

$$\begin{array}{rcl} e_A : G_2 \times G_1 & \longrightarrow & G_3 \\ (Q, P) & \longmapsto & (f_{T,Q}(P))^{\frac{p^k-1}{r}} \end{array}$$

D'après le théorème de Hasse, nous savons que $-2\sqrt{p} \leqslant t \leqslant 2\sqrt{p}$. Si nous supposons que nous sommes dans le cas où $\rho \simeq 1$, c'est-à-dire $\log_2(r) \simeq \log_2(p)$, nous déduisons que $\log_2(t) \leqslant \frac{\log_2(r)}{2} + 1$.

Cela implique que l'Algorithme 1 de Miller nécessite deux fois moins d'itérations pour le couplage de Ate que pour le couplage de Tate, ce qui est un gain non négligeable. Mais malheureusement pour ce couplage nous calculons la fonction rationnelle $f_{T,Q}(P)$, qui est Miller Full. Cela est un inconvénient puisque les opérations seront faites dans l'extension $\mathbb{F}_{p^k}$ et elles seront beaucoup plus complexes que pour le couplage de Tate.
Il serait intéressant de ramener le couplage de Ate sur $G_1 \times G_2$ afin d'avoir des étapes de Miller Lite, ce qui diminuerait la complexité de l'algorithme de Miller.

Le couplage Twisted Ate répond à cette problématique en utilisant une twist E' de E de degré d et en choisissant $G_2 = \Psi(E'[r]) \cap \mathrm{Ker}(\pi_p - [p])$. Le couplage de Twisted Ate est défini comme suit :

Définition 1.22. *Le couplage de Twisted Ate qu'on note e_{TA} est l'application suivante :*

$$\begin{aligned} e_{TA} : G_1 \times G_2 &\longrightarrow G_3 \\ (P,Q) &\longmapsto (f_{T^e,P}(Q))^{\frac{p^k-1}{r}} \end{aligned}$$

où $T = t-1$, t est la trace de Frobenius sur E, $m = pgcd(k,d)$ et $e = k/m$.

Remarque 1.16. *[HSV06a] Pour prouver la non dégénérescence du couplage de Ate et du Twisted Ate, il suffit de voir que ces deux couplages sont en fait une exponentiation fixe du couplage de Tate.*

Pour le couplage de Twisted Ate, il s'agit bien du calcul de Miller Lite qui se calcule en $e\log_2(t-1)$ itérations dans l'algorithme de Miller.Le couplage de Twisted Ate est donc plus performant que Tate si et seulement si $e\log_2(t-1) \leqslant \log_2(r)$

1.9.2 Le Couplage Optimal Ate

Le couplage Optimal Ate a été proposé par Vercauteren dans [Ver10].

Définition 1.23. *Soit $e : G_1 \times G_2 \longmapsto G_3$ un couplage $\mid G_1 \mid = \mid G_2 \mid = \mid G_3 \mid = r$. e est dit un couplage optimal s'il peut être calculé en exactement $\log_2(r)/\phi(k) + \epsilon(k)$ itérations de l'algorithme de Miller, où k est le degré de plongement relativement à r et $\epsilon(k) \leqslant \log_2(k)$.*

Remarque 1.17. *La définition du couplage Optimal Ate que nous venons de donner ne spécifie pas que e soit calculé en évaluant une seule fonction de Miller $f_{\lambda,Q}$ comme le cas du couplage de Ate. Par contre, ce couplage peut être défini par le produits des $f_{\lambda_i,Q}$ ou autres combinaisons sous la contrainte que le calcul à faire ne dépasse pas $\log_2(r)/\phi(k)$ itérations au total.*

Dans les Chapitres 3 et 4 , nous allons étudier le couplage Optimal Ate sur des catégories différentes de courbes elliptiques. Nous allons définir le groupe G_2 en faisant intervenir la courbe tordue que nous avons défini dans la Section 1.3.4. L'utilisation d'une telle courbe optimise le calcul des couplages au niveau de l'algorithme de Miller.
Dans notre travail, nous sommes intéressés aux courbes tordues pour un twist de degré pair, c'est-à-dire pour $d = 2,\ 4,$ et 6. Pour montrer l'importance de l'utilisation de la courbe tordue lors de calcul des couplages, nous traitons le cas de $d = 2$ (les autres cas se font de la même manière).

Soit $\zeta \in \mathbb{F}_{p^{k/2}}$ tel que ζ n'est pas un carré dans $\mathbb{F}_{p^{k/2}}$. La tordue E' de E est définie sur $\mathbb{F}_{p^{k/2}}$ par l'équation $\zeta y^2 = x^3 + ax + b$. L'isomorphisme envoyant un point de $\mathbb{F}_{p^k}$ sur un point de $E(\mathbb{F}_{p^k})$

est donné par :

$$\begin{aligned} \Psi_2 : E'\left(\mathbb{F}_{p^k}\right) &\rightarrow E(\mathbb{F}_{p^k}) \\ (x,y) &\mapsto (x, y\zeta^{1/2}) \end{aligned}$$

On choisit alors G_2 dans l'image de $E'\left(\mathbb{F}_{p^{k/2}}\right)$ par ψ_2. Le point $Q \in E(\mathbb{F}_{p^k})$ peut alors s'écrire $(x_Q, y_Q\zeta)$ avec $x_Q, y_Q \in \mathbb{F}_{p^{k/2}}$.

Remarque 1.18. *La non non-dégénérescence du couplage défini en utilisant la courbe tordue est prouvée dans [BLS03].*

En utilisant les coordonnées Jacobiennes présentées dans le Paragraphe 1.2.2, les équations de droites intervenant dans le calcul de l'algorithme de Miller sont [CLN10] :

$$\begin{aligned} &l_1(x_Q, y_Q\zeta) = Z_P^2\left(Z_{2T}Dy_Q\zeta - B(Dx_Q - X_T) - 2Y_T\right), \\ &v_1(x_Q, y_Q\zeta) = Z_{2T}^2 Z_P x_Q + 4Y_P^2(X_P D + X_T Z_P^2) - 9Z_P^2(X_T^2 - Z_T^4)^2, \\ &l_2(x_Q, y_Q\zeta) = Z_{T+P}^2\left(Z_T^3 E y_Q\zeta - Z_T F(Z_T^2 x_Q) - Y_T E\right), \\ &v_2(x_Q, y_Q\zeta) = Z_T^3 E\left(Z_{T+P}^3 x_Q + E(A+B) - Z_T^2 Z_P^2 F\right) \end{aligned}$$

D'après ces équations, nous remarquons que les additions et les multiplications intervenant dans ces formules s'effectuent dans le corps $\mathbb{F}_p$ et dans $\mathbb{F}_{p^{k/2}}$ au lieu d'être effectuées dans le corps $\mathbb{F}_{p^k}$. Ce résultat est important car l'arithmétique sur le corps fini $\mathbb{F}_{p^{k/2}}$ est moins coûteuse que celle dans $\mathbb{F}_{p^k}$. (Voir Chapitre 2).
De plus, nous remarquons aussi que les droites verticales v_1 et v_2 ne font pas intervenir l'ordonnée du point Q, et donc ζ non plus. Il est clair que $v_1(X_Q, y_Q\zeta)$ et $v_2(x_Q, y_Q\zeta)$ sont des éléments de $\mathbb{F}_{p^{k/2}}$. Ces expressions seront donc éliminées par l'exponentiation finale et il n'est donc pas nécessaire de les calculer. Cela est dû au résultat suivant

Proposition 1.3. *Soit r un diviseur premier de $\sharp E(\mathbb{F}_p)$ pour E une courbe elliptique de degré de plongement k pair et relativement à r. Alors $\frac{p^k-1}{r}$ est un multiple de $p^{k/2}-1$.*

Preuve. Nous avons,

$$\begin{cases} r | p^k - 1, \\ r | (p^{k/2}-1)(p^{k/2}+1), \\ r \nmid (p^{k/2}-1) \text{ (par définition de } k.) \\ r \text{ est premier} \end{cases}$$

D'après le théorème de Gauss

$$\frac{p^k-1}{r} = (p^{k/2}-1)\frac{p^{k/2}+1}{r}$$

□

En effet, puisque $v_1(Q)$ et $v_2(Q)$ sont deux éléments de $\mathbb{F}_{p^{k/2}}$, d'après la Proposition 1.3, $\frac{p^k-1}{r}$ est un multiple de $p^{k/2}-1$, ainsi $f_2^{\frac{p^k-1}{r}} = 1$ avec $f_2 = \{v_1(Q),\ \text{ou}\ \ v_2(Q)\}$.

Nous présentons maintenant la version optimisé de l'algorithme de Miller dans ce contexte.

Algorithm 2 : Miller (optimisé) pour k pair et twist de degré $\{2,\ 4,\ 6\}$

Require: $P \in G_1, Q \in G_2$, $r = (r_{n-1}, \ldots, \ldots r_0)$: la représentation binaire de r avec $r_{n-1} = 1$,

Ensure: $f_{r,Q}(P) \in \mathbb{F}_{p^k}^*$ à un facteur multiplicatif près appartenant à $\mathbb{F}_{p^{k/2}}$.

1: $f_1 \leftarrow 1$
2: $T \leftarrow Q$
3: **for** $i = n-2$ **down to** 0 **do**
4: $\quad T \leftarrow [2]T$
5: $\quad f_1 \leftarrow f_1^2 \cdot l_1(P)$
6: $\quad$ **if** $r_i = 1$ **then**
7: $\quad\quad T \leftarrow T + P$
8: $\quad\quad f_1 \leftarrow f_1 \cdot l_2(P)$
9: $\quad$ **end if**
10: $\quad$ **return** f_1
11: **end for**

Nous donnons un exemple de couplage Optimal Ate que nous allons étudier dans le Chapitre 3, c'est le couplage Optimal Ate sur les courbes de Barreto et Naehrig qui possède un degré de plongement $k = 12$ et une twist de degré $d = 6$.

Le couplage Optimal Ate est défini par l'application bilinéaire et non dégénérée suivante

$$\begin{aligned} E(\mathbb{F}_p)[r] \times \Psi_6\left(E'(\mathbb{F}_{p^2})[r]\right) &\longrightarrow \mu_r \\ (P,Q) &\longmapsto \left((f_{6u+2,Q}(P) l_{[6u+2]Q,\pi(Q)}(P) l_{[6u+2]Q,\pi^2(Q)}(P))\right)^{\frac{p^{12}-1}{r}} \end{aligned}$$

avec le morphisme Ψ_6 est alors défini par :

$$\begin{aligned} \Psi_6 : E'\left(\mathbb{F}_{p^2}\right) &\to E(\mathbb{F}_{p^{12}}) \\ (x,y) &\mapsto (x\zeta^{1/3}, y\zeta^{1/2}). \end{aligned}$$

Chapitre 2

Arithmétique sur les corps finis pour les couplages

Dans la suite de cette thèse, nous utilisons les notations suivantes pour décrire les opérations dans le corps fini $\mathbb{F}_{p^k}$.

A_k est le coût d'une addition dans $\mathbb{F}_{p^k}$,
A'_k est le coût d'une multiplication par 2 dans $\mathbb{F}_{p^k}$,
M_k est le coût d'une multiplication dans $\mathbb{F}_{p^k}$,
sM_k est le coût d'une multiplication creuse *'sparse multiplication'* dans $\mathbb{F}_{p^k}$ (voir la Définition 2.2.3),
$m_{k,c}$ désigne le coût de la multiplication par une constante c dans $\mathbb{F}_{p^k}$.
S_k est le coût d'une élévation au carré dans $\mathbb{F}_{p^k}$,
F_k est le coût d'un Frobenius dans $\mathbb{F}_{p^k}$ (voir la Définition 2.3),
I_k est le coût d'une inversion dans $\mathbb{F}_{p^k}$

Pour faciliter les notations et comme nous l'avons noté dans le chapitre 1, la multiplication dans $\mathbb{F}_p$ est notée par M, le carré par S et l'inversion par I.

Dans ce chapitre, nous allons présenter l'arithmétique sur les corps finis pour les couplages. Plus précisément, nous nous intéressons dans notre travail aux couplages optimaux et nous allons présenter l'arithmétique sur les corps finis utilisés pour le calcul du couplage Optimal Ate.

Pour calculer le couplage Optimal Ate sur les courbes elliptiques nous avons besoin de l'arithmétique dans les corps $\mathbb{F}_p$, $\mathbb{F}_{p^{k/d}}$ et $\mathbb{F}_{p^k}$. Dans notre travail, nous allons plutôt étudier les courbes elliptiques de degré de plongement $k = 12$, c'est le cas des courbes de Barreto et Naehrig [BN05] et aussi les courbes de Barreto, Lynn et Scott [BLS02]. Nous allons également considérer les courbes elliptiques de degré de plongement égal à 16 qui sont les courbes de Kachisa, Schafer et Scott [KSS07]. Pour cette raison, nous allons présenter deux types d'extensions de $\mathbb{F}_p$ qui sont $\mathbb{F}_{p^{2i}}$ et $\mathbb{F}_{p^{3i}}$ pour qu'on puisse définir l'arithmétique nécessaire pour le calcul du couplage sur ces catégories de courbes. Nous supposons que toutes les opérations : la multiplication, le carré, l'inversion et l'addition sur $\mathbb{F}_p$ sont déjà implémentées.

2.1 Arithmétique sur les extensions de la forme $\mathbb{F}_{p^{2i}}/\mathbb{F}_{p^i}$

Dans cette section nous présentons l'arithmétique sur les extensions quadratiques. Théoriquement, pour construire le corps fini $\mathbb{F}_{p^{2i}}$ à partir du $\mathbb{F}_{p^i}$, nous avons besoin d'un polynôme quadratique irréductible. De plus, nous choisissons ce polynôme de telle sorte qu'il possède des coefficients nuls. Car, sinon, des opérations supplémentaires seront impliquées. Pour cette raison, $\mathbb{F}_{p^{2i}}$ est toujours construit à partir d'un polynôme de la forme $X^2 - \mu$ sous la condition que μ ne soit pas un carré dans $\mathbb{F}_{p^i}$ (pour assurer l'irréductibilité du polynôme $X^2 - \mu$). Nous avons alors :

$$\mathbb{F}_{p^{2i}} = \mathbb{F}_{p^i}[\alpha] \text{ avec } \alpha^2 = \mu.$$

Exemple 2.1. *Soit $p \equiv 3 \mod 4$, il est clair que (-1) n'est pas un carré dans $\mathbb{F}_p$. Nous déduisons alors que :*

$$\mathbb{F}_{p^2} = \mathbb{F}_p[\boldsymbol{i}], \text{ avec } \boldsymbol{i}^2 = -1.$$

2.1.1 Addition dans $\mathbb{F}_{p^{2i}}$

L'addition sur les corps finis est l'opération la plus simple et la moins coûteuse (pareil aussi pour la soustraction).
Soient $x = x_0 + x_1\alpha$ et $y = y_0 + y_1\alpha$ deux éléments de $\mathbb{F}_{p^{2i}}$ avec $x_0,\ x_1,\ y_0, y_1 \in \mathbb{F}_{p^i}$. Alors,

$$(x_0 + x_1\alpha) + (y_0 + y_1\alpha) = (x_0 + y_0) + (x_1 + y_1)\alpha.$$

Ainsi, une addition dans $\mathbb{F}_{p^{2i}}$ vaut 2 additions dans $\mathbb{F}_{p^i}$.

$$A_{2i} = 2A_i \text{ et } A'_{2i} = 2A'_i.$$

2.1.2 Multiplication dans $\mathbb{F}_{p^{2i}}$

La multiplication est l'opération la plus importante dans le calcul du couplage sur les courbes elliptiques et les coûts des différentes opérations que nous étudions dans ce chapitre sont évaluées en termes de multiplications.
Plusieurs méthodes de multiplications sont présentées. Nous allons citer en premier lieu la multiplication **schoolbook** puis celle de **Karatsuba**. Puisque nous allons citer deux méthodes différentes, nous allons noter par M^1 le coût de la première méthode et par M^2 pour le coût de la deuxième.
La méthode de Schoolbook :
C'est la méthode classique du livre scolaire. c'est une méthode très simple. Soient x et y les éléments de $\mathbb{F}_{p^{2i}}$ dont nous souhaitons calculer le produit. Nous avons :

$$(x_0 + x_1\alpha)(y_0 + y_1\alpha) = x_0y_0 + \mu x_1y_1 + (x_0y_1 + x_1y_0)\alpha.$$

Cette méthode nécessite 4 multiplications, deux additions dans $\mathbb{F}_{p^i}$ et une multiplication par μ. Explicitement, le coût de cette méthode est alors :

$$M^1_{2i} = 4M_i + m_{i,\mu} + 2A_i.$$

La méthode de Karatsuba :
Cette méthode est une variante de la multiplication schoolbook et consiste à calculer le produit de x et y comme suit :

$$(x_0 + x_1\alpha)(y_0 + y_1\alpha) = x_0y_0 + \mu x_1y_1 + ((x_0 + x_1)(y_0 + y_1) - x_0y_0 - x_1y_1)\,\alpha.$$

L'idée de cette méthode est de remarquer que :

$$(x_0y_1 + x_1y_0) = ((x_0 + x_1)(y_0 + y_1) - x_0y_0 - x_1y_1)$$

L'avantage de cette nouvelle écriture est que nous réutilisons les termes x_0y_0 et x_1y_1 qui sont déjà pré-calculés. Du coup nous n'aurons pas besoin de les ré-calculer. Par conséquent la multiplication de Karatsuba coûte :

$$M_{2i}^2 = 3M_i + m_{i,\mu} + 5A_i.$$

En utilisant cette méthode, d'une part nous gagnons une multiplication dans $\mathbb{F}_{p^i}$ et d'autre part nous effectuons 3 additions de plus dans $\mathbb{F}_{p^i}$ par rapport à la méthode schoolbook.

Remarque 2.1. *Selon le coût d'une addition par rapport à une multiplication dans $\mathbb{F}_{p^i}$ nous pouvons déduire la méthode de multiplication la plus efficace.*
Si $M_i > 3A_i$, il est intéressant de considérer la multiplication de Karatsuba plutôt que la multiplication schoolbook. Nous supposerons dans la suite que nous serons toujours dans ce cas.

Nous récapitulons dans le tableau suivant le coût d'une multiplication dans $\mathbb{F}_{p^{2i}}$ en utilisant les deux méthodes citées.

Méthode	Coût
Schoolbook	$M_{2i}^1 = 4M_i + m_{i,\mu} + 2A_i.$
Karatsuba	$M_{2i}^2 = 3M_i + m_{i,\mu} + 5A_i.$

TABLE 2.1 – Les coûts de la multiplication dans $\mathbb{F}_{p^{2i}}$

2.1.3 Carré dans $\mathbb{F}_{p^{2i}}$

Nous présentons trois méthodes qui nous permettent de calculer un carré dans $\mathbb{F}_{p^{2i}}$: la méthode schoolbook, de complexité S_{2i}^1, celle de Karatsuba de complexité S_{2i}^2 et enfin la méthode complexe de complexité S_{2i}^3.

La méthode de Schoolbook :
C'est la méthode classique de calcul du carré. Étant donné x un élément de $\mathbb{F}_{p^{2i}}$ dont nous souhaitons calculer le carré, nous écrivons :

$$(x_0 + x_1\alpha)^2 = x_0^2 + \mu x_1^2 + 2x_0x_1\alpha.$$

Cette méthode nécessite une multiplication, deux carrés, une addition, une multiplication par 2 et une multiplication par μ dans $\mathbb{F}_{p^i}$. Ainsi

$$S_{2i}^1 = M_i + 2S_i + A_i + A_i' + m_{i,\mu}.$$

La méthode de Karatsuba :
En utilisant la méthode de Karatsuba, le calcul du carré dans $\mathbb{F}_{p^{2i}}$ se fait comme suit :

$$(x_0 + x_1\alpha)^2 = x_0^2 + \mu x_1^2 + \left((x_0 + x_1)^2 - x_0^2 - x_1^2\right)\alpha.$$

Comme dans le cas de la multiplication de Karatsuba, pour l'élévation au carré, l'idée est de réutiliser des termes déjà calculés afin de gagner sur le nombre des multiplications. Cette méthode requiert 3 carrés, une multiplication par μ et 4 additions dans $\mathbb{F}_{p^i}$. Ainsi

$$S_{2i}^2 = 3S_i + 4A_i + m_{i,\mu}.$$

La méthode complexe :
Cette méthode est très efficace si μ vaut $-1(\alpha = \sqrt{-1})$ d'où son nom.
En utilisant cette méthode, nous calculons le carré de x comme suit :

$$(x_0 + x_1\alpha)^2 = (x_0 + \mu x_1)(x_0 + x_1) - (\mu + 1)x_0x_1 + 2x_0x_1\alpha.$$

Cette méthode nécessite deux multiplications, une multiplication par μ, une multiplication par $(\mu + 1)$, trois additions et une multiplication par 2 dans $\mathbb{F}_{p^i}$. Le coût d'un carré en utilisant la méthode complexe est :

$$S^3_{2i} = 2M_i + m_{i,\mu} + m_{i,\mu+1} + 3A_i + A'_i.$$

Remarque 2.2. *Dans le cas où nous utilisons la méthode complexe et lorsque nous avons $\mu = -1$, le coût du carré devient particulièrement intéressant :*

$$S^3_{2i} = 2M_i + 2A_i + A'_i.$$

Nous résumons dans le tableau suivant le coût d'une élévation au carré dans $\mathbb{F}_{p^{2i}}$ en utilisant les trois méthodes citées : la méthode schoolbook, celle de Karatsuba et la méthode complexe.

Méthode	Coût
Schoolbook	$S^1_{2i} = M_i + 2S_i + A_i + A'_i + m_{i,\mu}$.
Karatsuba	$S^2_{2i} = 3S_i + 4A_i + m_{i,\mu}$.
Complexe	$S^3_{2i} = 2M_i + m_{i,\mu} + m_{i,\mu+1} + 3A_i + A'_i$.

TABLE 2.2 – Les coûts du carré dans $\mathbb{F}_{p^{2i}}$

2.1.4 Inversion dans $\mathbb{F}_{p^{2i}}$

D'une manière générale, l'inversion est l'opération la plus coûteuse et la plus compliquée dans $\mathbb{F}_p$ et dans toute extension de $\mathbb{F}_p$.
L'inversion de tout élément de $\mathbb{F}_{p^{2i}}$ est déduite grâce à la formule suivante :

$$N = (x_0 + x_1\alpha)(x_0 - x_1\alpha) = x_0^2 - \mu x_1^2 \ \in \mathbb{F}_p$$

Cette égalité nous permet de déduire que :

$$(x_0 + x_1\alpha)^{-1} = \frac{x_0}{N} - \frac{x_1}{N}\alpha.$$

C'est-à-dire, l'inverse de $x \in \mathbb{F}_{p^{2i}}$ est :

$$(x_0 + x_1\alpha)^{-1} = \frac{x_0}{x_0^2 - \mu x_1^2} - \frac{x_1}{x_0^2 - \mu x_1^2}\alpha$$

qui coûte une inversion dans $\mathbb{F}_{p^i}$, deux multiplications, deux carrés, une multiplication par μ et une addition dans $\mathbb{F}_{p^i}$. Nous aurons alors :

$$I_{2i} = I_i + 2M_i + 2S_i + A_i + m_{i,\mu}.$$

2.2 Arithmétique sur les extensions de la forme $\mathbb{F}_{p^{3i}}/\mathbb{F}_{p^i}$

Dans cette section nous nous intéressons à l'arithmétique des extensions de la forme $\mathbb{F}_{p^{3i}}/\mathbb{F}_{p^i}$. Nous détaillons le coût d'une addition, d'une multiplication creuse, d'une multiplication et d'une élévation au carré $\mathbb{F}_{p^{3i}}$.
La construction de $\mathbb{F}_{p^{3i}}$ se fait de la même manière que celle de $\mathbb{F}_{p^{2i}}$. Nous choisissons un polynôme irréductible bien particulier, défini que par son coefficient dominant et la constante, pour minimiser les coûts des opérations sur ce corps fini.
Ainsi, si $p^i \equiv 1(3)$, $\mathbb{F}_{p^{3i}}$ est construit comme suit :

$$\mathbb{F}_{p^{3i}} = \mathbb{F}_{p^i}[\alpha] \text{ avec } \alpha^3 = \xi, \text{ où } \xi \text{ n'est pas un cube dans } \mathbb{F}_{p^i}.$$

Exemple 2.2. *Nous donnons ici une manière de construire le corps fini $\mathbb{F}_{p^6}$ comme extension de degré 3 de $\mathbb{F}_{p^2} = \mathbb{F}_p[\boldsymbol{i}]$. On suppose que $1 + \boldsymbol{i}$ n'est pas un cube de $\mathbb{F}_p[\boldsymbol{i}]$ (on a vu dans le Chapitre 1, Théorème 1.1, les conditions dans lesquelles cette hypothèse est vérifiée) et on construit*

$$\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[\beta] \text{ avec } \beta^3 = 1 + \boldsymbol{i}.$$

Remarque 2.3. *L'arithmétique sur le corps fini $\mathbb{F}_{p^{3i}}$ implique des multiplications par ξ. Un bon choix de ξ est donc important pour atteindre une complexité optimale.*

2.2.1 Addition dans $\mathbb{F}_{p^{3i}}$

Comme nous l'avons déjà mentionné, l'addition dans les corps finis est l'opération la plus simple. Étant donné x, y deux éléments de $\mathbb{F}_{p^{3i}}$ avec $x = x_0 + x_1\alpha + x_2\alpha^2$ et $y = y_0 + y_1\alpha + y_2\alpha^2$, leur somme est :

$$(x_0 + x_1\alpha + x_2\alpha^2) + (y_0 + y_1\alpha + y_2\alpha^2) = (x_0 + y_0) + (x_1 + y_1)\alpha + (x_2 + y_2)\alpha^2.$$

L'addition de deux éléments de $\mathbb{F}_{p^{3i}}$ coûte trois additions dans $\mathbb{F}_{p^i}$. Ainsi

$$A_{3i} = 3A_i.$$

Remarquons que, la multiplication par deux dans $\mathbb{F}_{p^{3i}}$, que nous notons A'_i, vaut 3 multiplications par 2 dans $\mathbb{F}_{p^i}$.

2.2.2 Multiplication dans $\mathbb{F}_{p^{3i}}$

Pour la multiplications dans $\mathbb{F}_{p^{3i}}$ nous allons présenter aussi deux méthodes comme dans le cas de la multiplication dans $\mathbb{F}_{p^{2i}}$, la méthode Schoolbook de complexité notée M^1_{3i} et la méthode de Karatsuba de complexité M^2_{3i}.
La méthode de Schoolbook :
C'est la méthode directe et simple de la multiplication de deux éléments de $\mathbb{F}_{p^{3i}}$. En effet, le produit de x par y se calcule comme suit :

$$\begin{aligned}(x_0 + x_1\alpha + x_2\alpha^2)(y_0 + y_1\alpha + y_2\alpha^2) &= x_0y_0 + \xi(x_1y_2 + x_2y_1) \\ &+ [x_0y_1 + x_1y_0 + \xi x_2y_2]\,\alpha \\ &+ [x_0y_2 + y_2x_0 + x_1y_1]\,\alpha^2.\end{aligned}$$

Ce calcul de $x \times y$ nécessite 9 multiplications dans $\mathbb{F}_{p^{3i}}$, deux multiplications par ξ et 6 additions dans $\mathbb{F}_{p^{3i}}$. Nous avons ainsi :

$$M_{3i}^1 = 9M_i + 2m_{i,\xi} + 6A_i.$$

La méthode de Karatsuba :
Comme dans $\mathbb{F}_{p^{2i}}$, nous calculons le terme $x_1y_2 + x_2y_1$ en utilisant une seule multiplication dans $\mathbb{F}_{p^i}$ au lieu de deux en utilisant le fait que x_1y_1 et x_2y_2 sont de toutes façons calculés :

$$\begin{aligned}(x_0 + x_1\alpha + x_2\alpha^2)(y_0 + y_1\alpha + y_2\alpha^2) &= x_0y_0 + \xi\left((x_1 + x_2)(y_1 + y_2) - x_1y_1 - x_2y_2\right)\\ &+ \left[(x_0 + x_1)(y_0 + y_1) - x_0y_0 - x_1y_1 + \xi x_2y_2\right]\alpha\\ &+ \left[(x_0 + x_2)(y_0 + y_2) - x_0y_0 - x_2y_2 + x_1y_1\right]\alpha^2.\end{aligned}$$

Cette multiplication requiert 6 multiplications dans $\mathbb{F}_{p^{3i}}$, deux multiplications par ξ et 15 additions.

$$M_{3i}^2 = 6M_i + 15A_i + 2m_{i,\xi}.$$

Remarquons, qu'en utilisant cette méthode de Karatsuba, d'une part nous gagnons trois multiplications dans $\mathbb{F}_{p^{3i}}$. Mais d'autre part, nous avons 9 additions de plus à effectuer.
Dans ce cas, nous pouvons déduire que la multiplication de Karatsuba n'est plus efficace que celle de schoolbook que lorsque le coût d'une multiplication dans $\mathbb{F}_{p^i}$ est supérieur ou égal à celui de 3 additions dans $\mathbb{F}_{p^i}$, c'est-à-dire $M_i \geqslant 3A_i$.

Remarque 2.4. *Il existe d'autres méthodes qui permettent de multiplier deux éléments de $\mathbb{F}_{p^{3i}}$ telle que la méthode de Toom-Cook. Nous n'allons pas les considérer car elles nécessitent encore plus d'additions dans le corps fini $\mathbb{F}_{p^i}$ ainsi que des divisions par 2 ou par 3.*

2.2.3 Multiplication creuse dans $\mathbb{F}_{p^{3i}}$

Lors du calcul du couplage sur les courbes elliptiques et plus précisément dans le calcul de l'algorithme de Miller, nous devons effectuer des multiplications creuses dans $\mathbb{F}_{p^{3i}}$ qu'on note sM_{3i}.

Définition 2.1. *Nous désignons par multiplication creuse, une multiplication de deux éléments dont l'un possède au moins un coefficient nul.*

Nous présentons maintenant la multiplication creuse, le cas où $y_2 = 0$, en utilisant la méthode de Karatsuba qui est la plus efficace. Nous calculons alors :

$$\begin{aligned}(x_0 + x_1\alpha + x_2\alpha^2) \times (y_0 + y_1\alpha) &= x_0y_0 + \xi x_2y_1\\ &+ \left[(x_0 + x_1)(y_0 + y_1) - x_0y_0 - x_1y_1\right]\alpha\\ &+ \left[x_2y_0 + x_1y_1\right]\alpha^2.\end{aligned}$$

Cette multiplication creuse coûte 5 multiplications dans $\mathbb{F}_{p^i}$, une multiplication par ξ et 6 additions dans $\mathbb{F}_{p^i}$.

$$sM_{3i}^2 = 5M_i + 6A_i + m_{i,\xi}.$$

Remarquons que le coût de la multiplication creuse dépend du coefficient nul. En effet, si par exemple $y_1 = 0$ alors $sM_{3i}^2 = 5M_i + 6A_i + 2m_{i,\xi}$ et si $y_0 = 0$ alors $sM_{3i}^2 = 6M_i + 3A_i + 2m_{i,\xi}$.

Remarque 2.5. *Une multiplication creuse en utilisant la méthode de Karatsuba est importante si un seul coefficient est nul. Sinon, si les deux coefficients d'un même facteur sont nuls, nous utilisons plutôt la multiplication classique schoolbook.*

Nous résumons les résultats obtenus dans le tableau suivant.

Méthode	Coût
Schoolbook	$M_{3i}^1 = 9M_i + 6A_i + 2m_{i,\xi}$.
Karatsuba	$M_{3i}^2 = 6M_i + 15A_i + 2m_{i,\xi}$.
Multiplication creuse ($y_2 = 0$)	$sM_{3i}^2 = 5M_i + 6A_i + m_{i,\xi}$.
Multiplication creuse ($y_1 = 0$)	$sM_{3i}^2 = 5M_i + 6A_i + 2m_{i,\xi}$.
Multiplication creuse ($y_0 = 0$)	$sM_{3i}^2 = 6M_i + 3A_i + 2m_{i,\xi}$.

TABLE 2.3 – Les coûts de la multiplication dans $\mathbb{F}_{p^{3i}}$

2.2.4 Carré dans $\mathbb{F}_{p^{3i}}$

Pour le calcul du carré dans $\mathbb{F}_{p^{3i}}$, nous allons présenter trois méthodes, la méthode de Schoolbook de complexité S_{3i}^1, la méthode de Karatsuba de complexité S_{3i}^2 et la méthode de Chung-Hasan de complexité S_{3i}^4.

La méthode de Schoolbook :

Cette méthode calcule le carré de $x \in \mathbb{F}_{p^{3i}}$ comme suit :

$$(x_0 + x_1\alpha + x_2\alpha^2)^2 = x_0^2 + 2\xi x_1x_2 + \left[2x_0x_1 + \xi x_2^2\right]\alpha + \left[x_1^2 + 2x_0x_2\right]\alpha^2.$$

Elle nécessite 3 multiplications, 3 carrés, 3 additions, deux multiplications par 2 et une multiplication par ξ dans $\mathbb{F}_{p^i}$.

$$S_{3i}^1 = 3M_i + 3S_i + 3A_i + 2A_i' + 2m_{i,\xi}.$$

La méthode de Karatsuba :

Nous calculons le carré dans $\mathbb{F}_{p^{3i}}$ en utilisant la méthode de Karatsuba comme suit :

$$\begin{aligned}(x_0 + x_1\alpha + x_2\alpha^2)^2 &= x_0^2 + \xi\left((x_0 + x_1)^2 - x_0^2 - x_1^2\right)\\ &+ \left[(x_0 + x_1) - x_0^2 - x_1^2 + \xi x_2^2\right]\alpha\\ &+ \left[x_1^2 + (x_0 + x_2) - x_0^2 - x_2^2\right]\alpha^2.\end{aligned}$$

Cette méthode requiert 6 carrés, 12 additions et deux multiplications par ξ.

$$S_{3i}^2 = 6S_i + 12A_i + 2m_{i,\xi}.$$

La méthode de Chung-Hasan :

Pour les extensions de degré 3 il est recommandé d'utiliser le carré de Chung-Hasan présentée dans [CH07]. Nous calculons le carré comme suit :

$$\begin{aligned}(x_0 + x_1\alpha + x_2\alpha^2)^2 &= x_0^2 + 2\xi x_1x_2 + \left[2x_0x_1 + \xi x_2^2\right]\alpha\\ &+ \left[(x_0 + x_1 + x_2)^2 - (2x_0x_1 + 2x_1x_2 + x_0^2 + x_2^2)\right]\alpha^2.\end{aligned}$$

Cette méthode nécessite 2 multiplications, 3 élévations au carré, 8 additions, une multiplication par deux et une multiplication par ξ.

$$S_{3i}^4 = 2M_i + 3S_i + 8A_i + A_i' + 2m_{i,\xi}.$$

Nous résumons dans le tableau suivant le coût d'une élévation au carré dans $\mathbb{F}_{p^{3i}}$.

Méthode	Coût
Schoolbook	$S^1_{3i} = 3M_i + 3S_i + 3A_i + 2A'_i + 2m_{i,\xi}$.
Karatsuba	$S^2_{3i} = 6S_i + 12A_i + 2m_{i,\xi}$.
Chung-Hasan	$S^4_{3i} = 2M_i + 3S_i + 8A_i + A'_i + 2m_{i,\xi}$.

TABLE 2.4 – Les coûts du carré dans $\mathbb{F}_{p^{3i}}$

2.3 Exponentiation finale des Couplages

Le calcul du couplage de Tate, ainsi que ses dérivés, se fait en deux étapes. D'abord le calcul de la fonction de Miller (Lite si nous calculons le couplage de Tate ou Twisted Ate et Full si nous traitons le couplage Optimal Ate). Puis, le résultat de l'algorithme de Miller, qui est un élément de l'extension $\mathbb{F}_{p^k}$, est élevé à la puissance $\frac{p^k-1}{r}$. La complexité de l'exponentiation finale dépend alors de cet exposant qui est grand. Afin de le simplifier, Kolbitz et Menzez [KM05] l'ont divisé en deux parties, et nous allons démontrer par la suite qu'elles sont bien définies. Ainsi

$$\frac{p^k-1}{r} = \frac{(p^k-1)}{\phi_k(p)} \times \frac{\phi_k(p)}{r},$$

où $\phi_k(p)$ représente l'évaluation du $k-$ième polynôme cyclotomique en p. Ce polynôme est défini comme suit :

Définition 2.2. *Le $k-$ième polynôme cyclotomique est le polynôme unitaire admettant pour racine exactement les racines $k-$ième primitive de l'unité dans $\mathbb{C}$. C'est-à-dire,*

$$\phi_k(X) = \prod_{\alpha^k=1,\ ord(\alpha)=k} (X-\alpha).$$

Proposition 2.1. *Soit k un entier, alors le polynôme X^k-1 se factorise à l'aide des polynômes cyclotomiques, en effet :*

$$X^k - 1 = \prod_{k'|k} \phi_{k'}(X).$$

Propriétés 2.1. *Nous avons les propriétés suivantes :*

- *le polynôme cyclotomique admet des racines simples,*
- *le degré du polynôme cyclotomique vaut $\varphi(k)$ où φ est l'indicatrice d'Euler.*

Exemple 2.3. *Soit $k = 2^i 3^j$ pour $i, j \geq 1$, le $k-$ième polynôme cyclotomique est le polynôme*

$$\phi_k(X) = X^{k/3} - X^{k/6} + 1.$$

La factorisation de $\frac{p^k-1}{r}$ à l'aide du polynôme cyclotomique est bien définie comme nous l'avons vu dans le Chapitre 1 pour les raisons suivantes

- le fait que $\phi_k(p)$ divise p^k-1 est une conséquence directe de la Proposition 2.1.
- Nous supposons que r divise $\phi_{k'}(p)$ pour k' diviseur strict de k, alors r doit diviser $p^{k'}-1$ avec $k' \leq k$, ce qui contredit la définition de k (rappelons que k, le degré de plongement de la courbe elliptique, est le plus petit entier tel que r divise (p^k-1)). Par la suite, et puisque r est premier, nous justifions que r divise $\phi_k(p)$.

$$\text{Ainsi , } \frac{p^k-1}{r} = \frac{(p^k-1)}{\phi_k(p)} \times \frac{\phi_k(p)}{r} \text{ est bien défini.}$$

Dans le calcul de l'exponentiation finale, nous distinguons deux parties. En premier lieu, nous devons élever le résultat de l'algorithme de Miller à la puissance $\frac{(p^k-1)}{\phi_k(p)}$. Cette partie est appelée la partie facile de l'exponentiation finale. Nous devons ensuite élever le résultat à la puissance $\frac{\phi_k(p)}{r}$. Cette deuxième partie est dite la partie difficile de l'exponentiation finale. Une grande partie de cette thèse est consacrée à étudier cette partie qui est la plus coûteuse.

2.3.1 Partie facile de l'exponentiation finale

Nous commençons par étudier la partie facile de l'exponentiation finale. Supposons que f_1 est le résultat de l'algorithme de Miller, qui est un élément de $(\mathbb{F}^*_{p^k})$, et que nous souhaitons l'élever à la puissance $\frac{(p^k-1)}{\phi_k(p)}$. Nous traitons dans la majorité de notre travail le cas où $k = 2^i3^j$ alors, le polynôme cyclotomique est $\phi_k(p) = p^{k/3} - p^{k/6} + 1$. Par conséquent,

$$\frac{(p^k-1)}{\phi_k(p)} = \left(p^{k/2}-1\right)\left(p^{k/6}+1\right)$$

Pour cette première partie de l'exponentiation finale, nous devons

— Élever f_1 à la puissance $\left(p^{k/2}-1\right)$ en effectuant un Frobenius (élévation à la puissance d'un exposant de p) pour calculer $f^{p^{k/2}}$, une inversion dans le corps fini $\mathbb{F}_{p^k}$ afin d'avoir f_1^{-1}, puis une multiplication dans $\mathbb{F}_{p^k}$ pour avoir $f_1^{\left(p^{k/2}-1\right)}$,

— élever le résultat de $f_1^{\left(p^{k/2}-1\right)}$ à la puissance $\left(p^{k/6}+1\right)$. Cette opération nous coûte un Frobenius dans le calcul de $f^{p^{k/6}}$ et une multiplication dans $\mathbb{F}_{p^k}$.

Comme un Frobenius est une opération simple à effectuer (cf Section 2.4), le calcul de la partie facile de l'exponentiation finale est simple à traiter. De plus, l'élévation à la puissance $p^{k/2}-1$ facilite les calculs ultérieurs comme les inversions ou les élévations au carré.

Lemme 2.1. *Soit $\alpha \in \mathbb{F}^*_{p^k}$ tel que $\alpha = f_1^{p^{k/2}-1}$ avec $f_1 \in \mathbb{F}^*_{p^k}$, alors nous avons :*

$$\alpha^{-1} = \alpha^{p^{k/2}}.$$

Preuve. Il suffit de calculer $\alpha^{p^{k/2}+1}$

$$\alpha^{p^{k/2}+1} = f_1^{(p^{k/2}-1)(p^{k/2}+1)} = f_1^{p^k-1} = 1.$$

La dernière égalité n'est autre que le théorème de Lagrange dans le groupe multiplicatif $\mathbb{F}^*_{p^k}$.

Nous aurons alors $\alpha^{p^{k/2}+1} = \alpha^{p^{k/2}} \times \alpha$ qui est égal à 1. Nous déduisons que,

$$\alpha^{p^{k/2}} = \alpha^{-1}.$$

□

Par conséquent, une fois nous avons calculé la première partie de la partie facile de l'exponentiation finale, toute prochaine inversion ne coûte qu'un Frobenius. De plus, ce Frobenius est une exponentiation par $p^{k/2}$ qui est une simple conjugaison dans $\mathbb{F}_{p^k}/\mathbb{F}_{p^{k/2}}$. De la même manière, le coût de tout carré dans $\mathbb{F}_{p^k}$ peut être optimisé mais nous y reviendrons dans la Section 2.5.

2.3.2 Partie difficile de l'exponentiation finale

Une fois calculée la première partie de l'exponentiaton finale, $f_1^{\frac{p^k-1}{\phi_k(p)}}$, dite partie facile, il reste à évaluer ce résultat à la puissance $\frac{\phi_k(p)}{r}$. Cette deuxième étape de calcul est dite la partie difficile de l'exponentiation finale vu qu'elle est coûteuse.
Soit f le résultat de $f_1^{\frac{p^k-1}{\phi_k(p)}}$. Le calcul de $f^{\frac{\phi_k(p)}{r}}$ peut se faire de différentes manières possibles.
Les premières méthodes apparues sont les algorithmes des exponentiations rapides. Dans ce contexte, nous citons deux algorithmes. Le premier est dit *square and multiply*. C'est une méthode classique de calcul apparue il y a plus de 2000 ans. Nous traitons cette méthode en détail dans le Chapitre 3. Le deuxième algorithme de multi-exponentiations est celui de la fenêtre glissante. C'est une optimisationde l'algorithme *square and multiply*.
La méthode des suites de Lucas est une alternative remarquable à ces deux algorithmes. L'idée est d'effectuer les opérations dans le corps intermédiaire $\mathbb{F}_{p^{k/2}}$ au lieu de $\mathbb{F}_{p^k}$. Ceci est un avantage important puisque l'arithmétique y sera bien plus simple. La meilleure méthode à ce jour pour le calcul de la partie difficile de l'exponentiation finale est celle proposée par Scott *et al.* dans [Sco05]. Le principe de cette méthode est de développer la partie difficile de l'exponentiation finale en base p, c'est-à-dire :

$$\frac{\phi_k(p)}{r} = \sum_{i=0}^{\varphi(k)-1} \lambda_i \times p^i,$$

avec $\varphi(k)$ est l'indicatrice d'Euler de k (et donc le degré de ϕ_k). Ce développement permet de calculer la partie difficile de l'exponentiation finale d'une manière plus simple, (voir Chapitre 3), car il fait intervenir des exposants qui sont des puissances de p, c'est à dire des Frobenius, dont nous allons maintenant étudier la complexité. Nous reprendrons en détail l'étude de l'exponentiation finale dans le chapitre suivant où nous détaillerons toutes les méthodes permettant le calcul de la partie coûteuse de l'exponentiation finale ainsi que leurs complexités.

2.4 Le calcul du Frobenius

Le Frobenius que nous avons défini dans le Chapitre 1 est une opération simple à calculer.

Définition 2.3. *Nous désignons par l'opération* ***Frobenius*** *toute élévation à la puissance de p^i où i est un entier naturel vérifiant $i < \varphi(k)$.*

Nous allons nous restreindre ici au calcul du Frobenius sur le corps fini $\mathbb{F}_{p^{12}}$ car c'est celui qui nous sera le plus utile dans la suite mais les résultats sont similaires dans les autres cas. Nous supposons de plus que nous utilisons la tour d'extension suivante qui est recommandée dans la littérature :

$$\begin{aligned}
\mathbb{F}_{p^2} &= \mathbb{F}_p[\mathbf{i}] \text{ avec } \mathbf{i}^2 = -1, (\text{ si } p \equiv 3 \mod 4) \\
\mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[\beta] \text{ avec } \beta^3 = \xi = 1 + \mathbf{i}, \ p^2 \equiv 1[3], \\
\mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[\gamma] \text{ avec } \gamma^2 = \beta.
\end{aligned}$$

Ainsi, un élément arbitraire de $\mathbb{F}_{p^{12}}$ s'écrit sous la forme suivante :

$$a = b_0 + b_1\gamma + b_2\gamma^2 + b_3\gamma^3 + b_4\gamma^4 + b_5\gamma^5, \tag{2.1}$$

avec $b_j \in \mathbb{F}_{p^2}$ pour $0 \leqslant j \leqslant 5$ et $\gamma^6 = \xi = 1 + \mathbf{i} \in \mathbb{F}_{p^2}$ n'est ni un carré ni un cube dans $\mathbb{F}_{p^2}$.

Dans cette section, nous allons donner deux exemples de calcul de Frobenius, celui de a^p et celui de a^{p^2}. Nous donnons également le coût pour chaque opération. Nous donnons ensuite une généralisation du calcul pour toute autre puissance de p.

2.4.1 Le calcul de a^p

Soit $a \in \mathbb{F}_{p^{12}}$ donné par l'expression 4.11. Alors, le calcul de a^p se fait comme suit :

$$a^p = b_0^p + b_1^p\gamma^p + b_2^p(\gamma^2)^p + b_3^p(\gamma^3)^p + b_4^p(\gamma^4)^p + b_5^p(\gamma^5)^p.$$

Le calcul de b_j^p est simple, en effet, b_j^p est une simple conjugaison de b_j. C'est-à-dire, $b_j^p = \overline{b_j}$.
En effet, puisque b_j est un élément de $\mathbb{F}_{p^2}$, alors $b_j = b_{0,j} + b_{1,j}\mathbf{i}$ avec $b_{0,j}$, $b_{1,j} \in \mathbb{F}_p$, alors :

$$\begin{aligned} b_j^p &= (b_{0,j} + b_{1,j}\mathbf{i})^p \\ &= b_{0,j} + b_{1,j}\mathbf{i}(\mathbf{i}^2)^{\frac{p-1}{2}} \\ &= b_{0,j} - b_{1,j}\mathbf{i} \quad \text{car } p \equiv 3 \mod 4. \end{aligned}$$

Nous devons étudier maintenant le calcul de $(\gamma^j)^p$. Définissons tout d'abord δ par $\delta = \xi^{\frac{p-1}{6}}$. C'est un élément de $\mathbb{F}_{p^2}$ et nous avons :

$$\gamma^p = (\gamma^6)^{\frac{p-1}{6}}\gamma = (\xi)^{\frac{p-1}{6}}\gamma = \delta\gamma.$$

Ainsi, nous obtenons :

$$a^p = \overline{b_0} + \overline{b_1}\delta\gamma + \overline{b_2}\delta^2\gamma^2 + \overline{b_3}\delta^3\gamma^3 + \overline{b_4}\delta^4\gamma^4 + \overline{b_5}\delta^5\gamma^5.$$

Pour calculer a^p et sous l'hypothèse que les δ^j sont pré-calculés, il nous faut $5M_2$. En utilisant la multiplication de Karatsuba, cette opération nécessite 15 multiplications dans $\mathbb{F}_p$ (sans compter les additions).

2.4.2 Le calcul de a^{p^2}

Le calcul de a^{p^2}, avec a un élément de $\mathbb{F}_{p^{12}}$ se fait comme suit :

$$a^{p^2} = b_0^{p^2} + b_1^{p^2}\gamma^{p^2} + b_2^{p^2}(\gamma^2)^{p^2} + b_3^{p^2}(\gamma^3)^{p^2} + b_4^{p^2}(\gamma^4)^{p^2} + b_5^{p^2}(\gamma^5)^{p^2}.$$

De la même manière que pour le calcul de b_j^p, il est simple de vérifier que $b_j^{p^2} = b_j$. Ainsi,

$$a^{p^2} = b_0 + b_1\gamma^{p^2} + b_2(\gamma^2)^{p^2} + b_3(\gamma^3)^{p^2} + b_4(\gamma^4)^{p^2} + b_5(\gamma^5)^{p^2}.$$

Pour le calcul de γ^{p^2}, remarquons que $\gamma^{p^2} = (\xi^{\frac{p^2-1}{6}})\gamma$. On pose donc $w = \xi^{\frac{p^2-1}{6}} = N_{\mathbb{F}_{p^2}/\mathbb{F}_p}(\delta) \in \mathbb{F}_p$.
Nous avons alors,

$$\gamma^{p^2} = (\xi^{\frac{p^2-1}{6}})\gamma = w\gamma$$

Notons que w est une racine primitive 6-ième de l'unité ($w^6 = \xi^{p^2-1} = 1$) puisque ξ n'est ni un carré ni un cube dans $\mathbb{F}_{p^2}$. En particulier, nous avons :

$$w^2 - w + 1 = 0. \tag{2.2}$$

Alors,

$$a^{p^2} = b_0 + b_1 w\gamma + b_2 w^2\gamma^2 + b_3 w^3\gamma^3 + b_4 w^4\gamma^4 + b_5 w^5\gamma^5.$$

Or, $w^2 = w - 1$, $w^3 = -1$, $w^4 = -w$ et $w^5 = 1 - w$, nous obtenons :

$$a^{p^2} = b_0 + b_1 w\gamma + b_2(w-1)\gamma^2 - b_3\gamma^3 - b_4 w\gamma^4 - b_5(w-1)\gamma^5.$$

En supposant que w est pré-calculé, le calcul de a^{p^2} nécessite 4 multiplications d'un élément de $\mathbb{F}_{p^2}$ par un élément de $\mathbb{F}_p$ soit 8 multiplications dans $\mathbb{F}_p$.

2.4.3 Généralisation du calcul de a^{p^i}

Le résultat que nous avons présenté dans les Paragraphes 2.4.1 et 2.4.2 peut être généralisé comme suit :
Pour $i = 1, \ldots\ldots, 11$, $j = 0, \ldots\ldots, 5$, posons $c_{i,j} = w^{j\lfloor i/2 \rfloor}$. Remarquons que d'après l'équation 2.2, nous avons :

$$\begin{cases} c_{i,j} = 1 \text{ si } j\lfloor i/2 \rfloor = 0 \mod 6 \\ c_{i,j} = w \text{ si } j\lfloor i/2 \rfloor = 1 \mod 6 \\ c_{i,j} = w - 1 \text{ si } j\lfloor i/2 \rfloor = 2 \mod 6 \\ c_{i,j} = -1 \text{ si } j\lfloor i/2 \rfloor = 3 \mod 6 \\ c_{i,j} = -w \text{ si } j\lfloor i/2 \rfloor = 4 \mod 6 \\ c_{i,j} = -w + 1 \text{ si } j\lfloor i/2 \rfloor = 5 \mod 6 \end{cases}$$

Nous obtenons selon la parité de i ce deux cas :

— Si i est impair alors :

$$a^{p^i} = \overline{b_0} + \overline{b_1} c_{i,1}\delta\gamma + \overline{b_2} c_{i,2}\delta^2\gamma^2 + \overline{b_3} c_{i,3}\delta^3\gamma^3 + \overline{b_4} c_{i,4}\delta^4\gamma^4 + \overline{b_5} c_{i,5}\delta^5\gamma^5.$$

— Si i est pair alors :

$$a^{p^i} = b_0 + b_1 c_{i,1}\gamma + b_2 c_{i,2}\gamma^2 + b_3 c_{i,3}\gamma^3 + b_4 c_{i,4}\gamma^4 + b_5 c_{i,5}\gamma^5.$$

Remarque 2.6. *Remarquons que le calcul de a^{p^6} est simple et ne coûte rien. En effet, il ne s'agit que d'une simple conjugaison dans $\mathbb{F}_{p^{12}}/\mathbb{F}_{p^6}$.*

En supposant que les pré-calculs sont faits. Alors, le coût du calcul de a^{p^i} est $15M$ si i est impair et $8M$ sinon.

2.5 Optimisation dans le calcul des carrés

Comme nous l'avons mentionné dans la Section 2.3, une fois calculée la première partie de l'exponentiation finale, toute autre opération sera faite dans un sous groupe particulier, le sous groupe cyclotomique $G_{\phi_k(p)}$. Nous avons déjà vu que cela simplifiait les inversions, mais cela simplifie également les élévations au carré. Ceci est intéressant vu que la deuxième partie de l'exponentiation finale nécessite un important nombre de carrés. Toute optimisation sur le coût du carré a un effet important sur le coût total du couplage.
Dans cette section, nous allons présenter deux méthodes d'optimisation du calcul de carré dans ce groupe, la méthode de Karabina [Kar13] puis celle de Granger et Scott [GS10]. Nous donnons un exemple de calcul du carré pour chaque méthode. De même que pour le Frobenius, nous considérons que $k = 12$ et nous gardons la même tour d'extension que dans la Section 2.4.

2.5.1 La méthode de Karabina

Cette méthode de calcul du carré dans le sous-groupe cyclotomique a été présentée par Karabina dans [Kar13] et a été optimisée par Karabina *et al.* dans [AKL+11a]. Soit a un élément de $G_{\phi_{12}}(p)$ dont nous voulons calculer le carré en utilisant la méthode de Karabina.

$$a = b_0 + b_1\gamma + b_2\gamma^2 + b_3\gamma^3 + b_4\gamma^4 + b_5\gamma^5, \text{ avec } b_i \in \mathbb{F}_{p^2} \text{ où } 0 \leqslant i \leqslant 5.$$

D'après Karabina *et al.* , tout élément a peut être représenté d'une manière plus simple. En effet :

Théorème 2.1. *[Kar13] La représentation compressée de l'élément a est* $[b_1,\ b_2,\ b_4,\ b_5]$. *Ayant les coefficients* $b_1,\ b_2,\ b_4,\ b_5$ *et pour retrouver la représentation complète de a (ou représentation décompressée de a) on utilise les formules*

— *Si* $b_1 \neq 0$ *alors,* $\begin{cases} b_3 = \frac{b_5^2\xi + 3b_2^2 - 2b_4}{4b_1} \\ b_0 = (2b_3^2 + b_1b_5 - 3b_4b_5)\xi + 1 \end{cases}$

— *Si* $b_1 = 0$ *alors,* $\begin{cases} b_3 = \frac{2b_2b_5}{b_4} \\ b_0 = (2b_3^2 - 3b_4b_2)\xi + 1 \end{cases}$

Ainsi, pour retrouver les coefficients b_0 et b_3, on a à effectuer une inversion, 3 multiplications, 3 élévations au carré, 4 multiplications par deux, 6 additions, 2 multiplications par ξ dans $\mathbb{F}_{p^2}$ et une addition dans $\mathbb{F}_p$ si $b_1 \neq 0$. C'est-à-dire, nous effectuons :

$$I_2 + 3M_2 + 3S_2 + 4A_2' + 6A_2 + 2m_{2,\xi} + A.$$

Si nous sommes dans le deuxième cas où $b_1 = 0$, la décompression de l'élément a nécessite une inversion, 2 multiplications, une élévation au carré, 3 multiplications par deux, 2 additions, une multiplication par ξ dans $\mathbb{F}_{p^2}$ et une addition dans $\mathbb{F}_p$. C'est à dire :

$$I_2 + 3M_2 + S_2 + 2A_2' + 2A_2 + m_{2,\xi} + A.$$

L'idée de cette méthode est de calculer le carré de a en n'utilisant que sa représentation compressée. En effet, nous avons le théorème suivant :

Théorème 2.2. *[Kar13] Le carré de l'élément a, noté par* a^2 *sera présenté par* $[B_1,\ B_2,\ B_4,\ B_5]$ *avec :*

$$\begin{aligned} B_1 &= 2b_1 + 3\left((b_2+b_5)^2 - b_2^2 - b_5^2\right)\xi, \\ B_2 &= 3\left(b_1^2 + b_4^2\xi\right) - 2b_2, \\ B_3 &= 3\left(b_2^2 + b_5^2\xi\right) - 2b_4, \\ B_4 &= 2b_5 + 3\left((b_1+b_4)^2 - b_1^2 - b_4^2\right). \end{aligned}$$

Le calcul de a^2 compressé requiert alors 6 carrés, 4 multiplications par deux, 16 additions et 2 multiplications par ξ dans $\mathbb{F}_{p^2}$. C'est à dire :

$$6S_2 + 4A_2' + 16A_2 + 3m_{2,\xi}.$$

Remarque 2.7. *Nous avons compté une multiplication par* 3 *comme suit :*

$$3\left(b_1^2 + b_4^2\xi\right) = 2\left(b_1^2 + b_4^2\xi\right) + \left(b_1^2 + b_4^2\xi\right).$$

2.5.2 La méthode de Granger et Scott

Cette méthode a été présentée par Granger et Scott dans [GS10]. Dans leur papier, ils ont présenté le détail du calcul du carré pour toute extension de type (2, 2, 3). Leurs résultats ont été adaptés à toute extension de type (2, 3, 2) [DMHR15].
Nous ne présentons donc ici que le premier cas et nous supposons que le corps fini $\mathbb{F}_{p^{12}}$ est construit à partir de $\mathbb{F}_{p^4}$ comme suit :

$$\mathbb{F}_{p^4} = \mathbb{F}_{p^2}[\beta] \text{ avec } \beta^2 = \xi \text{ et } \mathbb{F}_{p^{12}} = \mathbb{F}_{p^4}[\gamma] \text{ où } \gamma^3 = \beta.$$

Soit a l'élément du sous groupe cyclotomique de $\mathbb{F}_{p^{12}}$ dont nous souhaitons calculer le carré.
$a = c_0 + c_1\gamma + c_2\gamma^2$ avec c_0, c_1 et $c_3 \in \mathbb{F}_{p^4}$. Le carré de a dans le sous-groupe cyclotomique $G_{\phi_6(p^2)}$ est donné par le théorème suivant :

Théorème 2.3. *[GS10]*

$$a^2 = C_0 + C_1\gamma + C_2\gamma^2 \text{ avec :}$$

$$\begin{aligned} C_0 &= 3c_0^2 - 2\overline{c_0}, \\ C_1 &= 3\beta c_2^2 + 2\overline{c_1}, \\ C_2 &= 3c_1^2 - 2\overline{c_2}. \end{aligned}$$

Ici $\overline{c_0}$ désigne le conjuguée de c_0, c'est-à-dire, si $c_0 = u + v\beta \in \mathbb{F}_{p^4}$, alors, $\overline{c_0} = \overline{u + v\beta} = u - v\beta$ pour tout u et v dans $\mathbb{F}_{p^2}$.

Ainsi, le coût d'un carré dans $\mathbb{F}_{p^{12}}$ nécessite 3 carrés, 3 multiplications par deux, 6 additions et une multiplication par β dans $\mathbb{F}_{p^4}$. Soit :

$$3S_4 + 3A'_4 + 6A_4 + m_{4,\beta} = 9S_2 + 6A'_2 + 24A_2 + 4m_{2,\xi}.$$

Remarque 2.8. *Pour expliciter le coût d'un carré en utilisant la méthode de Granger et Scott, nous utilisons le fait que :*

$$3a + 2b = 2(a + b) + a.$$

Remarque 2.9. *Nous ne pouvons pas déduire directement quelle méthode de calcul de carré est la plus efficace. En effet la méthode de Karabina [Kar13] est très efficace pour les carrés mais elle nécessite de faire une décompression avant chaque multiplication. Ainsi son efficacité globale sur une exponentiation dépendra de la forme de l'exposant (et en particulier de son poids de Hamming). Nous illustrons cela avec un exemple.*

Exemple 2.4. *Soit a un élément de $G_{\phi_{12}(p)}$, nous allons calculer deux exponentiations de a. Dans notre comparaison, nous n'allons considérer que les inversions, multiplications et élévations aux carrés.*
Cas 1 : calcul de $\mathbf{a^7}$ ***:*** *Comme le développement binaire de a n'est composé que de 1, il faut faire une multiplication à chaque étape de l'exponentiation et donc une décompression après chaque carré. La complexité totale est donné dans le tableau ci-dessous.*

Méthode	Karabina	Granger et Scott
Coût du calcul de a^7	$2I_2 + 40M_2 + 18S_2$ $\simeq 164\,M + 2\,I$	$\mathbf{36M_2 + 18S_2}$ $\simeq \mathbf{144\,M}$

TABLE 2.5 – Exemple 1 du calcul d'un carré dans le sous groupe cyclotomique

D'après ce tableau, il est clair que pour calculer a^7 avec a un élément de $G_{\phi_{12}(p)}$ c'est mieux d'utiliser la méthode de carré de Scott et Granger que celle de Karabina.
Cas 2 : calcul de $\mathbf{a^{64}}$: *Dans ce cas, le développement binaire de 64 est creux et on peut donc enchainer les carrés compressés. La complexité totale est donné dans le tableau ci-dessous :*

Méthode	Karabina	Granger et Scott
Coût du calcul de a^8	$\mathbf{I_2 + 2M_2 + 39S_2}$ $\simeq \mathbf{84\,M + I}$	$54S_2$ $\simeq 128\,M$

TABLE 2.6 – Exemple 2 du calcul d'un carré dans le sous groupe cyclotomique

D'après ce dernier tableau, il est clair que le calcul de a^{64}, en utilisant la méthode de Karabina est plus efficace qu'en utilisant celle de Granger et Scott.

Dans le calcul des complexités dans les chapitres suivants, nous n'allons pas considérer le nombre d'additions à faire pour chaque opération. En effet, nous considérons que le coût d'une addition est négligeable.

Chapitre 3

Optimal Ate dans un environnement restreint

Un inconvénient important lors des implémentations des couplages dans les cartes à puces ou dans d'autres environnements restreints est la limite de la mémoire de la carte. Dans ce chapitre, nous nous intéressons au calcul du couplage Optimal Ate sur les courbes de Barreto et Naehrig (courbes BN), pour le niveau de sécurité 128 bits (avant les résultats récents de [KB16]) en pensant à des implémentations dans des environnements restreints. Notre but est alors d'utiliser le minimum de variables temporaires afin de calculer ce couplage. Nous essayons de minimiser la mémoire utilisée et en parallèle nous essayons de ne pas perdre sur la complexité du couplage Optimal Ate sur les courbes BN.
Nous commençons en premier lieu par présenter les courbes de Barreto et Naehrig. Ces courbes sont recommandées pour le calcul du couplage Optimal Ate pour le niveau de sécurité 128 bits. Puis, nous définissons le couplage Optimal Ate sur cette catégorie des courbes elliptiques. Nous nous intéressons plutôt dans ce chapitre au calcul de l'exponentiation finale des couplages qui est la partie la plus gourmande en mémoire en la comparant avec l'étape de l'algorithme de Miller.
Posons $k' = k/2$, nous rappelons (voir chapitre 2) que l'exponentiation finale d'un couplage est donnée par

$$\frac{p^k - 1}{r} = \left(p^{k'} - 1\right)\left(\frac{p^{k'} + 1}{\phi_k(p)}\right)\left(\frac{\phi_k(p)}{r}\right).$$

Dans notre cas, le degré de plongement de la courbe BN est $k = 12$, par conséquent, l'exponentiation finale est :

$$\frac{p^{12} - 1}{r} = \left(p^6 - 1\right)\left(p^2 + 1\right)\frac{p^4 - p^2 + 1}{r}.$$

Dans la suite de ce chapitre, nous désignons par $\left(p^6 - 1\right)\left(p^2 + 1\right)$ la partie facile de l'exponentiation finale. Nous disons que c'est une partie facile vu que nous avons à effectuer que deux Frobenius, p^6 et p^2, deux multiplications et une inversion dans le corps fini $\mathbb{F}_{p^{12}}$.
La deuxième partie de l'exponentiation finale $\frac{p^4-p^2+1}{r}$ est dite la partie difficile. Notre travail dans ce chapitre porte sur le calcul de l'exponentiation correspondante à cette partie en minimisant le nombre de variables temporaires utilisées lors de ce calcul.
Plusieurs méthodes décrites dans la littérature nous permettent d'évaluer le résultat de cette partie. Nous commençons par présenter la méthode naïve. Cette méthode consiste en un calcul direct de l'exposant en utilisant les algorithmes classiques d'exponentiations rapides. Ensuite, nous rappelons

les méthodes de la littérature les plus utilisées pour les implémentations matérielles. En premier lieu, nous citons la méthode de Devigili *et al.* [DSD07], puis celle de Scott *et al.* [SBC+09] et en dernier lieu celle de Fuentes *et al.* [CKH11]. Toutes ces méthodes ne tiennent pas compte du nombre de variables temporaires utilisées dans leurs algorithmes de calcul de la partie difficile de l'exponentiation finale. Nous avons repris leurs algorithmes afin de déterminer la mémoire nécessaire pour chaque algorithme.
Puisque notre but est de gagner sur la mémoire utilisée, nous avons repris ces travaux en faisant des modifications sur le développement de $\frac{p^4-p^2+1}{r}$ afin de rendre la partie difficile de l'exponentiation finale moins gourmande en mémoire.

3.1 Les Courbes de Barreto et Naehrig

Barreto et Naehrig ont présenté dans [BN05] une méthode pour générer des courbes elliptiques, bien adaptées aux couplages, définies sur le corps fini $\mathbb{F}_p$. Cette catégorie de courbes présente un degré de plongement k égal à 12. Rappelons, (voir Chapitre 1), que ces courbes sont dites courbes BN et sont définies par l'équation

$$E:\ y^2 = x^3 + b,$$

avec $b \neq 0$ qui n'est ni un carré ni un cube et par un paramètre u tel que :

$$\begin{aligned} r &= 36u^4 + 36u^3 + 18u^2 + 6u + 1 \\ p &= 36u^4 + 36u^3 + 24u^2 + 6u + 1. \\ t &= 6u^2 + 1 \end{aligned}$$

Le paramètre u est choisi de telle sorte que p et r soient premiers. Pour simplifier le calcul du couplage Optimal Ate sur les courbes BN, il est recommandé de choisir le paramètre u creux. Dans notre travail, nous allons considérer le paramètre donné dans l'exemple suivant :

Exemple 3.1. *Le choix le plus utilisé du paramètre u est celui proposé par Nogami* et al. *dans [NAS+08] et qui est*

$$u = -(4080000000000001)_{16}.$$

Le poids de Hamming de u est noté par w_u et vaut 3 et sa longueur en base deux que nous notons l_u est égale à 63.

Nous nous intéressons dans ce chapitre au calcul de la partie difficile de l'exponentiation finale du couplage Optimal Ate sur les courbes BN. Rappelons (voir Chapitre 1) que le couplage Optimal Ate sur les courbes BN est défini par l'application bilinéaire et non dégénérée suivante :

$$\begin{aligned} E(\mathbb{F}_p)[r] \times \Psi_6\left(E'(\mathbb{F}_{p^2})[r]\right) &\longrightarrow \mathbb{F}_{p^{12}}^{\times} \\ (P,Q) &\longmapsto \left(\left(f_{6u+2,Q}(P) l_{[6u+2]Q,\pi(Q)}(P) l_{[6u+2]Q,\pi^2(Q)}(P)\right)\right)^{\frac{p^{12}-1}{r}} \end{aligned}$$

À partir des expressions de p et r, la partie difficile de l'exponentiation finale peut être écrite à l'aide d'un polynôme en p de degré 3 comme suit :

$$\frac{p^4 - p^2 + 1}{r} = \sum_{i=0}^{\varphi(12)-1} \lambda_i p^i = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \lambda_3 p^3$$

avec

$$\begin{cases} \lambda_0 = -36u^3 - 30u^2 - 18u - 2 \\ \lambda_1 = -36u^3 - 18u^2 - 12u + 1 \\ \lambda_2 = 6u^2 + 1 \\ \lambda_3 = 1 \end{cases}$$

Remarque 3.1. *L'exposant* $d = \frac{p^4-p^2+1}{r}$*, peut être aussi représenté à l'aide d'un polynôme en* u *de degré* 12*.*

3.2 Les Méthodes dans la littérature

Il existe dans la littérature plusieurs méthodes qui nous permettent de calculer la partie difficile de l'exponentiation finale du couplage de Tate et ses variantes [FMR99b]. Dans cette section, nous donnons le coût ainsi le nombre de variables temporaires nécessaires pour ce calcul.

3.2.1 La méthode naïve

Nous désignons par la méthode naïve, la méthode la plus ancienne et la plus simple de calculer toute exponentiation. Il existe plusieurs algorithmes d'exponentiations. Nous présentons dans ce paragraphe l'algorithme suivant appelé *square and multiply*. Le grand avantage de cette méthode

Square and Multiply algorithm
Input : f, $d = (d_{n-1}, d_{n-2}, \dots d_1, d_0)_2$ avec $d_{n-1} = 1$
Output : f^d
$t_0 \leftarrow f$
for $i = n-2$ **down to** 0 **do**
$t_0 \leftarrow t_0^2$
if $d_i = 1$, **then** $t_0 \leftarrow t_0 f$
return t_0

TABLE 3.1 – *Square and Multiply algorithm*

de calcul est qu'elle ne nécessite qu'une seule variable temporaire dans le corps fini $\mathbb{F}_{p^{12}}$ que nous notons t_0, en plus de l'entrée f. Malheureusement cette méthode est très coûteuse au niveau de la complexité. En effet, la complexité de cet algorithme dépend de la longueur de l'exposant $\frac{p^4-p^2+1}{r}$ en base 2 ainsi que de son poids de Hamming.
Pour donner une approximation du poids de Hamming de $\frac{p^4-p^2+1}{r}$, nous donnons le résultat suivant :

Théorème 3.1. *Étant donnés* u*,* v *des entiers de poids de Hamming respectivement* w_u *et* w_v*, alors nous avons :*

1. $w_{u+v} \leqslant w_u + w_v$,
2. $w_{u^2} \leqslant \frac{w_u(w_u+1)}{2}$,
3. $w_{uv} \leqslant w_u w_v$.

Preuve. Le Théorème 3.1 est démontré comme suit :

1. Soit u et v deux entiers de poids de Hamming respectivement $w_u = m$ et $w_u = n$, avec

$$u = (2^{i_1} + \ldots \cdots + 2^{i_m}) \text{ et } v = (2^{j_1} + \ldots \cdots + 2^{j_n}).$$

Nous avons $u + v = (2^{i_1} + \ldots\ldots 2^{i_m} + 2^{j_1} + \ldots\ldots 2^{j_n})$.
Si $i_p \neq i_q$ pour $p \in \{1, \ldots, m\}$ et $q \in \{1, \ldots, n\}$, alors, il est clair que $w_{u+v} = w_u + w_v$.
Sinon, c'est-à-dire il existe des p et q tel que $i_p = i_q$, alors nous aurons des annulations dans l'expression de $u + v$. Par conséquent :

$$w_{u+v} \leqslant w_u + w_v.$$

2. Nous montrons par récurrence sur w_u que $w_{u^2} \leq \frac{w_u(w_u+1)}{2}$. En effet,
pour $w_u = 1$, le résultat est évident.
Supposons que l'hypothèse de récurrence est vrai pour u tel que $w_u = k$, c'est à dire, nous avons $w_{u^2} \leq \frac{k(k+1)}{2}$ et montrons qu'elle l'est pour u tel que $w_u = k + 1$.
Soit u un entier tel que $w_u = k + 1$. On a nécessairement $u = (x + 2^j)$ où x est un entier de poids de Hamming k. Alors,

$$\begin{aligned} u^2 &= (x + 2^j)^2 \\ &= x^2 + (2^{2j}) + 2.2^j x \\ \text{Or d'après 1 , } w_{u^2} &\leqslant w_{x^2} + 1 + w_x \\ &\leq \frac{k(k+1)}{2} + 1 + k = \frac{(k+1)(k+2)}{2}. \end{aligned}$$

Nous avons donc prouvé que

$$w_{u^2} \leqslant \frac{w_u(w_u+1)}{2}.$$

Bien entendu, là encore, cette inégalité est stricte quand il y a des annulations de bits.

3. Pour montrer que $w_{uv} \leqslant w_u w_v$, supposons que u et v deux entiers de poids de Hamming respectivement $w_u = m$ et $w_v = n$ tels que :

$$\begin{cases} v = (2^{i_1} + \cdots + \cdots + 2^{i_n}) \\ u = (2^{j_1} + \cdots + \cdots + 2^{j_m}) \end{cases}$$

Il est clair que

$$uv = 2^{i_1}v + 2^{i_2}v + \cdots + \cdots + 2^{i_m}v$$

$$\text{D'après 1, nous avons } w_{uv} \leqslant \sum_{k=1}^{m} w_v = w_u.w_v.$$

Par conséquent,

$$w_{uv} \leqslant w_u w_v.$$

□

Remarque 3.2. *Nous considérons que les résultats donnés dans le Théorème 3.1 sont des égalités dans la suite de notre travail. Nous négligeons les annulations car le poids de Hamming des entiers que nous utilisons est petit et donc les probabilités d'annulation sont faibles.*

Comme l'exposant $d = \frac{p^4-p^2+1}{r}$ est un polynôme de degré 12 en u, sa longueur en base 2 est d'environ 12 fois la longueur de u en base 2 et on peut utiliser le théorème 3.1 pour calculer explicitement son poids de Hamming en fonction de celui de u. Nous ne donnons toutefois pas

ce résultat car il est compliqué à écrire et présente peu d'interêt. En effet, même si le poids de Hamming de u est petit, celui de d a peu de chances de l'être aussi.

Ainsi, le calcul direct de $f^{\frac{p^4-p^2+1}{r}}$ en utilisant la méthode naïve nécessite $w_{\frac{p^4-p^2+1}{r}}$ multiplications et environ $12l_u$ élévation au carré dans $\mathbb{F}_{p^{12}}$.

Exemple 3.2. *Si nous choisissons la valeur de u, recommandée par Nogami* et al. *et donnée dans l'exemple 3.1, nous devons effectuer* 759 *carrés et* 335 *multiplications dans* $\mathbb{F}_{p^{12}}$ *pour calculer* $f^{\frac{p^4-p^2+1}{r}}$.

Remarque 3.3. *Nous pouvons faire beaucoup mieux que la méthode naïve en choisissant les algorithmes d'exponentiation plus avancés tel que celui de la fenêtre glissante. Cependant ces méthodes nécessitent beaucoup plus de variables temporaires dans* $\mathbb{F}_{p^{12}}$ *pour stocker les précalculs et elles seront de toutes façons moins efficaces que les méthodes qui vont suivre.*

3.2.2 La méthode de Devigili *et al.*

En 2007 Devegili *et al.* ont proposé une méthode efficace pour calculer $f^{\frac{p^4-p^2+1}{r}}$ [DSD07]. Cette méthode consiste à développer l'exposant $\frac{p^4-p^2+1}{r}$ en faisant apparaître les exposants $6u^2+1$ et $6u+5$ plus qu'une fois. Le but étant de les calculer une seule fois.

$$\begin{aligned}
\frac{p^4-p^2+1}{r} &= \lambda_3 p^3 + \lambda_2 p^2 + \lambda_1 p + \lambda_0 \\
&= p^3 + (6u^2+1)p^2 + (-36u^3 - 18u^2 - 12u + 1)p \\
&\quad +(-36u^3 - 30u^2 - 18u - 2) \\
&= p^3 + p^2(6u^2+1) \\
&\quad +p\ \big((-6u-5)(6u^2+1) + 2(6u^2+1) + (-6u-5) + 9\big) \\
&\quad +(-6u-5)(6u^2+1) + (-6u-5) + (-6u-5) + 9 + 4
\end{aligned}$$

Nous avons alors

$$f^{\frac{p^4-p^2+1}{r}} = f^{p^3}\left(f^{p^2}\left(f^p f\right)^{-6u-5}\left(f^p\right)^2\right)^{6u^2+1}\left(f^p f\right)^{-6u-5} f^{-6u-5}\left(f^p f\right)^9 f^4$$

Pour calculer la partie difficile de l'exponentiation finale en utilisant ce développement, Devegili *et al.* ont présenté l'algorithme suivant :

Pour calculer la complexité de cet algorithme, nous présentons d'abord le lemme suivant comme

Méthode de Devegili *et al.* [DSD07]
Input : $f,\ u$
Output : $f^{\frac{p^4-p^2+1}{r}}$
$1 : a \leftarrow f^{-6u-5}$
$2 : b \leftarrow a^p$
$3 : b \leftarrow ab$
$4 :$ Calculer f^p, f^{p^2}, et f^{p^3}
$5 : f \leftarrow f^{p^3}[f^{p^2} b\,(f^p)^2]^{6u^2+1} ba\,(f^p f)^9\, f^4$

TABLE 3.2 – La méthode de Devigili

conséquence immédiate du Théorème 3.1 et en respectant la remarque 3.2.

Lemme 3.1. *Dans cet algorithme, nous avons une exponentiation par* $-(6u+5)$ *et une autre par* $(6u^2+1)$*, afin d'évaluer le coût de ces exponentiations, nous estimons leurs poids de Hamming :*

— $w_{(6u+5)} = 2w_u + 2$,

— $w_{(6u^2+1)} = w_u(w_u+1) + 1$.

Pour la première étape de l'algorithme 3.2, le calcul de f^{-6u-5}, nous aurons donc besoin (au plus) de $2w_u + 1$ multiplications et $l_u + 1$ carrés dans $\mathbb{F}_{p^{12}}$ puisque $-(6u+5)$ est un entier de longueur en base deux $(l_u + 2)$ bits et de poids de Hamming $2w_u + 2$. Les étapes 2 et 4 sont des exponentiations par des puissances de p donc des Frobenius qui sont simples à calculer. Il y a 4 Frobenius dans $\mathbb{F}_{p^{12}}$ à effectuer. La troisième étape de cet algorithme, c'est-à-dire le calcul de $b \leftarrow ab$, nécessite une multiplication dans $\mathbb{F}_{p^{12}}$. Pour la dernière étape, nous devons calculer en premier lieu $f^{p^2} b \left(f^p\right)^2$, qui nécessite deux multiplications, et une élévation de ce terme à la puissance $6u^2+1$. Pour cette exponentiation, il nous faut donc $2l_u S_{12}$ et $w_u(w_u+1)M_{12}$ vu que la longueur de $6u^2+1$ en base deux est $2l_u + 1$ et son poids de Hamming vaut $w_u(w_u+1)+1$. Dans cette étape de l'algorithme, nous calculons aussi $\left(f^p f\right)^9 a f^4$ qui exige 3 multiplications et 5 carrés dans $\mathbb{F}_{p^{12}}$. De plus on a besoin de 3 multiplications pour multiplier les termes ensemble. Au total, le calcul de la partie difficile de l'exponentiation finale en utilisant l'algorithme de Devigili *et al.* exige $(3l_u+7)S_{12} + 4F_{12}$ et $(w_u^2 + 3w_u + 10)M_{12}$.
Dans ce chapitre, nous nous intéressons plutôt à la mémoire utilisée lors du calcul de la partie difficile de l'exponentiation finale. Malheureusement, Devigili *et al.* dans leur papier [DSD07] n'ont pas considéré le problème de l'implémentation du couplage dans un environnement restreint. C'est pour cette raison que nous avons réécrit leur algorithme afin de compter le nombre de variables temporaires utilisées (cf Table 3.3). Appliquer l'algorithme de Devegili et al. nécessite donc 4

Méthode de Devegili détaillée	**Termes calculés**
Input : f, u	
Var. Temp. : t_0, t_1, t_2, t_3	
Output : $f^{\frac{p^4-p^2+1}{r}}$	
$t_0 \leftarrow f^{-6u-5}$	a
$t_1 \leftarrow t_0^p$	b
$t_0 \leftarrow t_0 t_1$	ba
$t_2 \leftarrow f^p$; $t_3 \leftarrow t_2 f$	$f^p f$
$t_3 \leftarrow {t_3}^9$; $t_0 \leftarrow t_0 t_3$	$b(f^p f)^9 a$
$t_3 \leftarrow f^4$; $t_0 \leftarrow t_0 t_3$	$b(f^p f)^9 a f^4$
$t_2 \leftarrow t_2^2$; $t_2 \leftarrow t_2 t_1$	$b(f^p)^2$
$t_1 \leftarrow f^{p^2}$; $t_1 \leftarrow t_1 t_2$	$f^{p^2} b \left(f^p\right)^2$
$t_2 \leftarrow t_1^{6u^2+1}$	
$t_0 \leftarrow t_2 t_0$	$[f^{p^2} b \left(f^p\right)^2]^{6u^2+1} b \left(f^p f\right)^9 a f^4$
$t_1 \leftarrow f^{p^3}$; $t_1 \leftarrow t_1 t_0$	$f^{\frac{p^4-p^2+1}{r}}$
return t_1	

TABLE 3.3 – La méthode de Devigili détaillée

variables temporaires dans $\mathbb{F}_{p^{12}}$ pour calculer $f^{\frac{p^4-p^2+1}{r}}$.

Exemple 3.3. *Si le paramètre u est celui choisi de l'exemple 3.1, alors* $(-6u-5)$ *est un entier de* 65

bits et son poids de Hamming devrait valoir 7 *(d'après le lemme 3.1) mais grâce aux simplifications, il est égal à 5. D'autre part,* $6u^2+1$ *est un entier de 127 bits et de poids de Hamming 13.*
La première étape de l'algorithme coûte alors $64S_{12}+4M_{12}$*. Nous aurons besoin de* $126S_{12}+12M_{12}$ *pour élever à la puissance* $6u^2+1$*. Donc, le coût total du calcul de la partie difficile de l'exponentiation finale en utilisant le développement de Devigili* et al. *exige exactement* $196S_{12}+26M_{12}+4F_{12}$.

3.2.3 La chaîne d'addition

En 2009, Scott *et al.* ont présenté une nouvelle approche pour calculer la partie difficile de l'exponentiation finale. Leur approche consiste à écrire l'exposant $\frac{p^4-p^2+1}{r}$ à l'aide de groupes d'éléments ayant les mêmes exposants. Soit donc,

$$\begin{aligned} f^d &= f^{\lambda_3 p^3} f^{\lambda_2 p^2} f^{\lambda_1 p} f^{\lambda_0} \\ &= f^{p^3} f^{\left(6u^2+1\right)p^2} f^{\left(-36u^3-18u^2-12u+1\right)p} f^{\left(-36u^3-30u^2-18u-2\right)} \\ &= f^{p^3} f^{p^2} \left(f^{p^2}\right)^{6u^2} \left(f^{p}\right)^{-36u^3} \left(f^{p}\right)^{-18u^2} \left(f^{p}\right)^{-12u} f^{p} f^{-36u^3} f^{-30u^2} f^{-18u} f^{-2} \\ &= f^{p} f^{p^2} f^{p^3} \frac{1}{f^2} \left[\left(f^{u^2}\right)^{p^2}\right]^6 \left[\frac{1}{\left(f^{u}\right)^{p}}\right]^{12} \left[\frac{1}{f^{u}\left(\left(f^{u^2}\right)^{p}\right)}\right]^{18} \left[\frac{1}{\left(f^{u^2}\right)}\right]^{30} \left[\frac{1}{f^{u^3}\left(f^{u^3}\right)^{p}}\right]^{36} \end{aligned}$$

$$\text{Alors} \qquad f^{\frac{p^4-p^2+1}{r}} = y_0 y_1^2 y_2^6 y_3^{12} y_4^{18} y_5^{30} y_6^{36} \tag{3.1}$$

$$\text{avec} \quad \begin{cases} y_0 = f^p f^{p^2} f^{p^3}; & y_1 = 1/f; \quad y_2 = \left(f^{u^2}\right)^{p^2} \\ y_3 = \left(f^{u}\right)^{p}; & y_4 = 1/f^{u}\left(\left(f^{u^2}\right)^{p}\right) \\ y_5 = 1/\left(f^{u^2}\right); & y_6 = 1/f^{u^3}\left(f^{u^3}\right)^{p} \end{cases}$$

L'algorithme d'Olivos nous aide à évaluer d'une manière très efficace toute expression de la forme 3.1 ([Oli81], [CFA+06] chapitre 9.2). La première étape consiste à trouver une chaîne d'addition qui inclut tous les exposants qui apparaissent dans l'équation (3.1). Dans notre cas, la meilleure chaîne d'addition est donnée par

$$\{1,\ 2,\ \overline{3},\ 6,\ 12,\ 18,\ 30,\ 36\}$$

Nous remarquons que 3 apparaît dans la chaîne d'addition choisie mais n'est pas l'un des exposants de l'équation (3.1). Comme mentionné dans [SBC+09], c'est un avantage puisqu'il signifie que nous aurons moins d'opérations à effectuer par la suite.
Rappelons que l'algorithme d'Olivos consiste à évaluer toute expression présentée à l'aide d'une chaîne d'addition en utilisant une complexité optimale. L'expression (3.1) peut ainsi être calculée grâce à l'algorithme suivant :

Pour cet algorithme présenté dans le tableau 3.4, nous n'aurons besoin que de deux variables temporaires, t_0 et t_1, 9 multiplications et 4 carrés dans $\mathbb{F}_{p^{12}}$. Cependant, nous aurons besoin de pré-calculer les y_i avec $0 \leqslant i \leqslant 6$. Le stockage des y_i nécessite alors 7 variables temporaires de plus dans $\mathbb{F}_{p^{12}}$, comme c'est indiqué dans [SBC+09]. Scott dit qu'il faut stocker les y_i et rajouter deux variables temporaires t_0 et t_1 pour calculer la partie difficile de l'exponentiation finale. Au total, il nous faut 9 variables temporaires dans $\mathbb{F}_{p^{12}}$.
On peut faire mieux, mais pas moins que 8. En effet, une fois calculé $t_0 \leftarrow y_6^2$, nous n'aurons plus

Méthode de la chaîne d'addition [Oli81], [SBC+09]
Input : $f,\ u$
Var. Temp : t_0, t_1
Output : $f^{\frac{p^4-p^2+1}{r}}$
$t_0 \leftarrow y_6^2$
$t_0 \leftarrow t_0 y_4$
$t_0 \leftarrow t_0 y_5$
$t_1 \leftarrow y_3 y_5$
$t_1 \leftarrow t_0 t_1$
$t_0 \leftarrow t_0 y_2$
$t_1 \leftarrow t_1^2$
$t_1 \leftarrow t_1 t_0$
$t_1 \leftarrow t_1^2$
$t_0 \leftarrow t_1 y_1$
$t_1 \leftarrow t_1 y_0$
$t_0 \leftarrow t_0^2$
$t_0 \leftarrow t_0 t_1$
return t_0

TABLE 3.4 – La méthode de la chaîne d'addition de Scott

besoin de y_6. Donc, au lieu d'utiliser une nouvelle variable temporaire t_1, nous pouvons ré-utiliser y_6.
Le coût du calcul des y_i est comme suit : $3(w_u - 1)$ multiplications et $3(l_u - 1)$ carrés pour calculer $f^{-u},\ f^{u^2}$ et f^{-u^3}, (car, des simples exponentiations par u), 7 Frobenius et à la fin 4 multiplications pour multiplier les termes ensemble. (Rappelons que, comme nous l'avons déjà mentionné dans le Chapitre 2, les inversions, qui sont des exponentiations par p^6, sont négligeables).
Ainsi, le coût total de la partie difficile de l'exponentiation finale est bien celui des y_i auquel nous rajoutons la complexité de l'algorithme d'Olivos. Nous obtenons alors $7F_{12}$, $(3w_u + 10)M_{12}$ et $(3l_u + 1)S_{12}$.

Exemple 3.4. *Si nous choisissons le paramètre u de l'Exemple 3.1, le calcul de la partie difficile de l'exponentiation finale en utilisant la méthode de la chaîne d'addition de Scott* et al. *coûte exactement* $190S_{12} + 19M_{12} + 7F_{12}$.

Remarque 3.4. *Nos estimations sur la complexité des deux méthodes précédentes, celles de Devigili* et al. *et de Scott* et al. *, nous permettent de confirmer les résultats d'implémentations pratiques de la littérature, à savoir que la méthode de calcul de la partie difficile de l'exponentiation finale présentée par Scott* et al. *est plus rapide d'environ* 4% *que la méthode de Devigili* et al.*. Par contre, nous remarquons que la méthode la plus rapide est beaucoup plus gourmande en mémoire. Pour le niveau de sécurité 128 bits, les 9 variables temporaires (sans compter les supplémentaires que nous allons détailler dans la Section 3.5) utilisées lors de la méthode de Scott représentent* 3.4 *Ko.*

3.2.4 La méthode de Fuentes *et al.*

Plus récemment, Fuentes *et al.* ont présenté une nouvelle manière de calculer la partie difficile de l'exponentiation finale dans [CKH11]. Leur méthode est basée sur le fait d'utiliser un multiple d' de $d = \frac{p^4-p^2+1}{r}$, (avec r ne divise pas d'). Ils calculent $f^{d'}$ au lieu de calculer f^d en utilisant le fait qu'une puissance fixe d'un couplage est un couplage dans le cas où $d'(u) = s(u) \times d(u)$ avec $s(u)$ n'est pas un multiple de $r(u)$.
Ils ont présenté une méthode basée sur les réseaux pour déterminer d' de telle sorte que le calcul de $f^{d'}$ est beaucoup plus simple et plus efficace que celui de f^d.
Grâce à l'algorithme LLL [LLL82] appliqué au réseau enegendré par les $u^i d(u)$, le meilleur vecteur, que nous notons d', qu'ils ont pu trouver est donné par :

$$d'(u) = \alpha_0 + \alpha_1 p + \alpha_2 p^2 + \alpha_3 p^3 = sd\,(u)$$

$$\text{avec, } \begin{cases} s = 2u\left(6u^2 + 3u + 1\right) \\ \alpha_0 = 1 + 6u + 12u^2 + 12u^3 \\ \alpha_1 = 4u + 6u^2 + 12u^3 \\ \alpha_2 = 6u + 6u^2 + 12u^3 \\ \alpha_3 = -1 + 4u + 6u^2 + 12u^3 \end{cases}$$

On peut remarquer que LLL a fourni un multiple faisant intervenir de plus petits coefficients dans la décomposition de d' que dans celle de d. Ensuite, ils reprennent la méthode de Devigili pour calculer $f^{d'}$ comme suit

$$f^{d'} = f^{\alpha_0} f^{\alpha_1 p} f^{\alpha_2 p^2} f^{\alpha_3 p^3} = \left(a f^{6u^2} f\right) b^p a^{p^2} \left(b f^{-1}\right)^{p^3}$$

avec $a = f^{6u} f^{6u^2} f^{12u^3}$ et $b = a f^{-2u}$.
Fuentes *et al.* n'ont pas donné un algorithme explicite du calcul de la partie difficile de l'exponentiation finale où on peut compter le nombre des variables temporaires nécessaires. Pour cette raison, nous avons détaillé leur développement dans l'algorithme 3.5

Méthode de Fuentes *et al.* [CKH11]	**Termes calculés**	**Coût**
Input : f, u		
Output : $f^{s\frac{p^4-p^2+1}{r}}$		
Var. Temp t_0, t_1, t_2, t_3, t_4		
$t_0 \leftarrow f^{-u}$		$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_0 \leftarrow t_0^2$		S_{12}
$t_1 \leftarrow t_0^2$		S_{12}
$t_1 \leftarrow t_0 t_1$		M_{12}
$t_2 \leftarrow t_1^{-u}$	f^{6u^2}	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_3 \leftarrow t_1^{-1}$	f^{6u}	
$t_1 \leftarrow t_2 t_3$		M_{12}
$t_3 \leftarrow t_2^2$		S_{12}
$t_4 \leftarrow t_3^{-u}$		$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_4 \leftarrow t_4^{-1}$		
$t_4 \leftarrow t_4 t_1$	$f^{6u}f^{6u^2}f^{12u^3}=f^{\alpha_2}$	M_{12}
$t_3 \leftarrow t_4 t_0$	$f^{4u}f^{6u^2}f^{12u^3}=f^{\alpha_1}$	M_{12}
$t_0 \leftarrow t_2 t_4$		M_{12}
$t_0 \leftarrow t_0 f$	$af^{6u^2}f=f^{\alpha_0}$	M_{12}
$t_2 \leftarrow t_3^p$		F_{12}
$t_0 \leftarrow t_2 t_0$		M_{12}
$t_2 \leftarrow t_4^{p^2}$		F_{12}
$t_0 \leftarrow t_2 t_0$		M_{12}
$t_2 \leftarrow f^{-1}$		
$t_2 \leftarrow t_2 t_3$	f^{α_3}	M_{12}
$t_2 \leftarrow t_2^{p^3}$		F_{12}
$t_0 \leftarrow t_2 t_0$	$f^{s\frac{p^4-p^2+1}{r}}$	M_{12}
return t_0		

TABLE 3.5 – La méthode de Fuentes *et al.*

Pour cet algorithme, nous avons utilisé 5 variables temporaires dans $\mathbb{F}_{p^{12}}$. Le coût de cet algorithme est bien détaillé dans le Tableau 3.5. Ainsi, le coût total du calcul de $f^{d'}$ est $3l_uS_{12} + (3w_u+7)M_{12} + 3F_{12}$.

Exemple 3.5. *Avec la même valeur du paramètre u choisi dans l'Exemple 3.1, le coût total du calcul du multiple de la partie difficile de l'exponentiation finale d' est* $189S_{12} + 16M_{12} + 3F_{12}$.

Remarque 3.5. *Il est clair qu'en utilisant la méthode de Fuentes* et al. *nous calculons une puissance du couplage et non pas le couplage lui même. Cela n'est pas un problème vu qu'une puissance d'un couplage est toujours un couplage (sous bien sûr certaines conditions). Malheureusement, cela peut être un inconvénient dans plusieurs situations. Par exemple, dans le cas où nous voulons implémenter un couplage standard tel que le couplage Optimal Ate [Ver10] pour des raisons d'interopérabilité et de compatibilité. Dans un cas pareil, cette méthode est à éviter.*

3.3 Nos Variantes

Dans cette section, nous présentons nos nouvelles approches de calcul de la partie difficile de l'exponentiation finale. Dans nos variantes, nous tenons compte de l'importance de la réduction du nombre des variables temporaires utilisées afin de pouvoir implémenter le couplage Optimal Ate défini sur les courbes BN dans des environnements restreints.
Nous allons présenter quatre variantes où nous gagnons sur la mémoire utilisée ce qui est important en pratique. Dans certains cas, nous sommes plus rapides que les résultats de la littérature et dans d'autres cas, nous sommes un peu moins efficaces mais notre perte est négligeable.
Plus précisément, en premier lieu, nous allons présenter un nouveau développement de la partie difficile de l'exponentiation finale, pour obtenir une variante de la méthode de Devigili. Cette variante n'est pas seulement moins gourmande en mémoire, elle est aussi plus efficace, elle présente un gain de 5% sur la complexité. Ensuite nous donnons une nouvelle chaîne d'addition qui présente une variante de celle de Scott *et al.*. Notre nouvelle chaîne d'addition nécessite moins de variables temporaires que la première. En dernier lieu, nous présentons 3 variantes de la méthode de Fuentes *et al.*. Nous sommes toujours efficaces sur la mémoire, nous gagnons dans chaque nouvelle variante un bon pourcentage sur la mémoire utilisée. Dans la première variante que nous allons présenter, nous gardons la même complexité (pour le u proposé dans la littérature), pour la deuxième, où nous utilisons un nouveau multiple de la partie difficile de l'exponentiation finale, nous sommes un peu plus lents et pour la troisième variante, nous somme plus efficaces.

3.3.1 Nouveau développement de $f^{\frac{p^4-p^2+1}{r}}$

Nous allons présenter une nouvelle manière de développer $\frac{p^4-p^2+1}{r}$. Dans ce développement, nous avons choisi de faire apparaître des termes plusieurs fois. Par exemple, nous avons $6u^2+1$, $-6u-1$ et $-6u-5$ dans λ_0, λ_1, λ_2 et λ_3. Cela est un avantage vu que nous allons les calculer une seule fois puis nous les réutilisons.
Notre développement est le suivant :

$$\begin{aligned}
\lambda_0 &= -36u^3 - 30u^2 - 18u - 2 = (-6u-5)(6u^2+1) + 2(-6u-1) + 5\\
\lambda_1 &= -36u^3 - 30u^2 - 12u + 1 = (-6u-5)(6u^2+1) + (-6u-1) + 12u^2 + 7\\
\lambda_2 &= 6u^2 + 1\\
\lambda_3 &= 1.
\end{aligned}$$

Par conséquent $f^{\frac{p^4-p^2+1}{r}} = f^{\lambda_3 p^3} f^{\lambda_2 p^2} f^{\lambda_1 p} f^{\lambda_0}$ peut être évaluée grâce à notre développement comme suit :

$$\begin{aligned}
f^{\frac{p^4-p^2+1}{r}} = {} & f^{p^3} \left(f^{6u^2+1}\right)^{p^2} \left(f^{-6u-1}\right)^{p} \left(\left(f^{6u^2+1}\right)^{-6u-5}\right)^{p} \left(f^{7}\right)^{p} \left(f^{12u^2}\right)^{p}\\
& \times \left(f^{6u^2+1}\right)^{-6u-5} \left(f^{-6u-1}\right)^{2} f^{5}
\end{aligned}$$

Pour calculer la partie difficile de l'exponentiation finale c'est-à-dire $\frac{p^4-p^2+1}{r}$, en utilisant notre développement, nous utilisons l'algorithme suivant présenté dans le Tableau 3.6.

Nouveau développement	**Termes calculés**	**Coût**
Input : $f,\ u$		
Output : $f^{\frac{p^4-p^2+1}{r}}$		
Var. Temp : t_0, t_1, t_2		
$t_0 \leftarrow f^{-2u}$		$l_u S_{12} + (w_u - 1)M_{12}$
$t_1 \leftarrow t_0^2;\ t_0 \leftarrow t_0 t_1$	f^{-6u}	$S_{12} + M_{12}$
$t_1 \leftarrow f^{-1};\ t_1 \leftarrow t_0 t_1$	f^{-6u-1}	M_{12}
$t_2 \leftarrow t_1^p$	$\left(f^{-6u-1}\right)^p$	F_{12}
$t_1 \leftarrow t_1^2$	$\left(f^{-6u-1}\right)^2$	S_{12}
$t_1 \leftarrow t_1 t_2$		M_{12}
$t_2 \leftarrow t_0^{-u}$	f^{6u^2}	$(l_u - 1)S_{12} + (w_u - 1)M_{12}$
$t_0 \leftarrow t_2^2;\ t_0 \leftarrow t_0^p$	$\left(f^{12u^2}\right)^p$	$S_{12} + F_{12}$
$t_1 \leftarrow t_1 t_0$		M_{12}
$t_2 \leftarrow t_2 f$		M_{12}
$t_0 \leftarrow t_2^{p^2}$	$\left(f^{6u^2+1}\right)^{p^2}$	F_{12}
$t_1 \leftarrow t_0 t_1$		M_{12}
$t_0 \leftarrow t_2^{-6u-5}$	$\left(f^{6u^2+1}\right)^{-6u-5}$	$(l_u + 1)S_{12} + (2w_u + 1)M_{12}$
$t_1 \leftarrow t_1 t_0$		M_{12}
$t_0 \leftarrow t_0^p$		F_{12}
$t_1 \leftarrow t_1 t_0$		M_{12}
$t_0 \leftarrow f^2;\ t_2 \leftarrow t_0^2;\ t_2 \leftarrow f t_2$	f^5	$2S_{12} + M_{12}$
$t_1 \leftarrow t_2 t_1$		M_{12}
$t_2 \leftarrow t_2 t_0;\ t_2 \leftarrow t_2^p$	$\left(f^7\right)^p$	$M_{12} + F_{12}$
$t_1 \leftarrow t_2 t_1$		M_{12}
$t_2 \leftarrow f^{p^3}$		F_{12}
$t_1 \leftarrow t_2 t_1$	$f^{\frac{p^4-p^2+1}{r}}$	M_{12}
return t_1		

TABLE 3.6 – Notre nouveau développement.

Pour cet algorithme, nous n'avons utilisé que trois variables temporaires dans $\mathbb{F}_{p^{12}}$. Nous avons donné le coût étape par étape dans le Tableau 3.6. En fait, le coût total du calcul de la partie difficile de l'exponentiation finale en utilisant notre nouveau développement est $(3l_u+5)S_{12}+6F_{12}$ et (au plus) $(4w_u + 12)M_{12}$.

Exemple 3.6. *En choisissant le paramètre u proposé par Nogami* et al. *dans l'Exemple 3.1, le coût du calcul de* $f^{\frac{p^4-p^2+1}{r}}$ *en utilisant notre développement est* $194S_{12} + 21M_{12} + 6F_{12}$.

Nous avons présenté en premier lieu dans la première section la méthode de Devigili *et al.* , et dans cette section, nous venons de présenter notre nouvelle variante de la méthode de Devigili *et al.*. Nous pouvons alors comparer ces deux méthodes dans le Tableau 3.7 suivant :

Méthode	Algorithme	Complexité			Var.
		S_{12}	M_{12}	F_{12}	Temp.
Devegili *et al.*	3.3	$3l_u + 7$	$w_u^2 + 3w_u + 10$	4	4
Notre nouveau développement	3.6	$3l_u + 5$	$4w_u + 12$	6	3

TABLE 3.7 – Comparaison entre la méthode de Devigili *et al.* et notre variante.

Exemple 3.7. *Pour avoir une comparaison complète, nous choisissons le paramètre u proposé dans l'Exemple 3.1. Nous obtenons alors le tableau suivant.*

Méthode	Complexité	Var. Tempo
Devegili *et al.*	$196S_{12} + 26M_{12} + 4F_{12}$	4
Notre nouveau développement	$194S_{12} + 21M_{12} + 6F_{12}$	3

TABLE 3.8 – Exemple pour la comparaison de 3.7

Si nous comparons la complexité de notre développement de la partie difficile de l'exponentiation finale avec le développement de Devigili *et al.* dans [DSD07], nous constatons que nous sommes plus rapides. Nous avons 5 multiplications et 2 carrés de moins que Devegili *et al.* dans le cas de l'exemple 3.1. Par contre, nous avons deux Frobenius de plus cependant cela n'est pas un problème puisque le coût d'un Frobenius est bien inférieur à celui d'une multiplication dans $\mathbb{F}_{p^{12}}$ comme nous l'avons montré dans le Chapitre 2. Remarquons également qu'en utilisant notre développement nous sommes moins gourmands en mémoire que Devegili *et al.* . Nous n'utilisons que 3 variables temporaires dans $\mathbb{F}_{p^{12}}$ au lieu de 4. Cela est très important vu que notre but était de réduire la mémoire utilisée pour le calcul de l'exponentiation finale du couplage et nous y arrivons pour notre première variante.

3.3.2 Notre nouvelle chaîne d'addition

Dans cette partie, nous nous intéressons au calcul de la partie difficile de l'exponentiation finale en utilisant les chaînes d'additions. Notre but est alors d'optimiser la méthode de calcul de $f^{\frac{p^4-p^2+1}{r}}$ en utilisant la méthode de Scott *et al.* dans [SBC+09]. Nous voulons minimiser le nombre de variables temporaires, pour cela, nous choisissons le développement suivant :

$$\begin{aligned}
f^{\frac{p^4-p^2+1}{r}} &= f^{\lambda_3 p^3} f^{\lambda_2 p^2} f^{\lambda_1 p} f^{\lambda_0} \\
&= f^{p^3} f^{\left(6u^2+1\right)p^2} f^{\left(-36u^3-18u^2-12u+1\right)p} f^{-36u^3-30u^2-18u-2} \\
&= f^{p^3} f^{p^2} \left((f^{u^2})^{p^2}\right)^6 (f^p)^{-36u^3} (f^p)^{-18u^2} (f^p)^{-12u} f^p f^{-36u^3} \\
&\quad f^{-30u^2} f^{-18u} f^{-2} \\
&= \left[f^p f^{p^2} f^{p^3}\right] \left[\left(f^{3u^2}\right)^{p^2} f^{-1} f^{-6u^2}\right]^2 \left[(f^p f)^{-4u} f^{-2u}\right]^3 \\
&\quad \left[(f^p f)^{-6u^3} (f^p f)^{-3u^2}\right]^6 . \qquad (3.2)
\end{aligned}$$

Nous obtenons alors une chaîne d'additions plus courte :

$$f^{\frac{p^4-p^2+1}{r}} = y_0 y_1^2 y_2^3 y_3^6 \tag{3.3}$$
$$\begin{aligned} \text{avec } y_0 &= f^p f^{p^2} f^{p^3} \\ y_1 &= \left(f^{3u^2}\right)^{p^2} f^{-1} f^{-6u^2} \\ y_2 &= (f^p f)^{-4u} f^{-2u} \\ y_3 &= (f^p f)^{-6u^3} (f^p f)^{-3u^2}. \end{aligned}$$

La première étape consiste à pré-calculer y_0, y_1, y_2 et y_3. Nous détaillons ces calculs dans le tableau suivant et donnons le coût de chaque opération que nous réalisons.

Les Précalculs	**Termes calculés**	**Coût**
Input : $f,\ u$		
Temp. var. : y_0, y_1, y_2, y_3, t_0		
Output : y_0, y_1, y_2, y_3		
$t_0 \leftarrow f^{-u}$		$(l_u-1)S_{12} + (w_u-1)M_{12}$
$y_3 \leftarrow t_0^2$	f^{-2u}	S_{12}
$y_2 \leftarrow y_3^p;\ y_2 \leftarrow y_3 y_2;\ y_2 \leftarrow y_2^2$	$(f^p f)^{-4u}$	$M_{12} + S_{12} + F_{12}$
$y_2 \leftarrow y_3 y_2$	y_2	M_{12}
$y_0 \leftarrow t_0 y_3$	f^{-3u}	M_{12}
$t_0 \leftarrow y_0^{-u}$	f^{3u^2}	$(l_u-1)S_{12} + (w_u-1)M_{12}$
$y_0 \leftarrow f^{-1};\ y_1 \leftarrow t_0^{p^2};\ y_1 \leftarrow y_0 y_1$	$(f^{3u^2})^{p^2} f^{-1}$	$F_{12} + M_{12}$
$t_0 \leftarrow t_0^{-1};\ y_3 \leftarrow t_0^p;\ y_3 \leftarrow t_0 y_3$	$(f^p f)^{-3u^2}$	$F_{12} + M_{12}$
$t_0 \leftarrow t_0^2$	f^{-6u^2}	S_{12}
$y_1 \leftarrow t_0 y_1$	y_1	M_{12}
$t_0 \leftarrow y_3^{-u}$	$(f^p f)^{3u^3}$	$(l_u-1)S_{12} + (w_u-1)M_{12}$
$t_0 \leftarrow t_0^2;\ t_0 \leftarrow t_0^{-1}$	$(f^p f)^{-6u^3}$	S_{12}
$y_3 \leftarrow t_0 y_3$	y_3	M_{12}
$t_0 \leftarrow f^p;\ y_0 \leftarrow f^{p^2};\ y_0 \leftarrow t_0 y_0$	$f^p f^{p^2}$	$2F_{12} + M_{12}$
$t_0 \leftarrow f^{p^3};\ y_0 \leftarrow t_0 y_0$	y_0	$F_{12} + M_{12}$
return y_0, y_1, y_2, y_3		

TABLE 3.9 – Les pré-calculs des y_0, y_1, y_2, et y_3

.

Dans cet algorithme, nous n'avons utilisé que 5 variables temporaires dans $\mathbb{F}_{p^{12}}$. Le coût de ces pré-calculs est donné étape par étape dans l'algorithme présenté dans le Tableau 3.9 et est exactement $(3l_u+1)S_{12} + (3w_u+6)M_{12} + 6F_{12}$.
Dans cette partie, nous avons besoin de 4 variables temporaires pour stocker nos résultats présentés dans y_0, y_1, y_2 et y_3 auxquels nous rajoutons la variable t_0 qui intervient dans le calcul intermédiaire des y_i, $0 \leqslant i \leqslant 3$.
Selon notre développement de la deuxième partie de l'exponentiation finale, il est clair que la chaîne d'addition que nous allons considérer est donnée par :

$$\{1,\ 2,\ 3,\ 6\}.$$

Afin de calculer $f^{\frac{p^4-p^2+1}{r}}$ en utilisant notre nouvelle chaîne d'additions, nous appliquons l'algorithme d'Olivos [Oli81] pour évaluer l'expression $y_0\ y_1^2\ y_2^3\ y_3^6$ en utilisant le minimum possible d'opérations grâce à l'algorithme suivant.

Notre Nouvelle chaîne d'addition
Input : f, y_0, y_1, y_2, y_3
Output : $f^{\frac{p^4-p^2+1}{r}}$
Temp. var. : t_0
$t_0 \leftarrow y_3^2$
$t_0 \leftarrow t_0 y_2$
$y_3 \leftarrow t_0 y_0$
$t_0 \leftarrow t_0 y_1$
$t_0 \leftarrow t_0^2$
$t_0 \leftarrow t_0 y_3$
return t_0

TABLE 3.10 – Algorithme d'Olivos appliqué à notre chaîne d'addition

Pour cet algorithme, nous devons effectuer 4 multiplications et deux carrés dans $\mathbb{F}_{p^{12}}$ pour calculer le résultat final de l'exponentiation finale du couplage Optimal Ate sur les courbes BN.
Remarquons que dans cet algorithme, nous n'avons pas utilisé de nouvelles variables temporaires dans $\mathbb{F}_{p^{12}}$. Nous avons réutilisé, en premier lieu, la variable t_0, déjà libre depuis l'algorithme présenté dans le Tableau 3.9. Puis, nous avons réutilisé la variable temporaire y_3 vu qu'à partir de la première affectation, y_3 est libre.
Ainsi, le coût total de la partie difficile de l'exponentiation finale en utilisant notre nouvelle chaîne d'addition est 6 Frobenius, $3w_u + 10$ multiplications et $3l_u + 3$ carrés dans $\mathbb{F}_{p^{12}}$.
Pour la mémoire totale utilisée, nous aurons besoin que de 5 variables temporaires dans $\mathbb{F}_{p^{12}}$ pour calculer la partie difficile de l'exponentiation finale en utilisant notre nouvelle chaîne d'addition.

Exemple 3.8. *En considérant la même valeur de u choisi dans l'Exemple 3.1, le coût de calcul de notre nouvelle chaîne d'addition est* $19M_{12} + 192S_{12} + 6F_{12}$. *Cependant, nous n'utilisons que 5 variables temporaires dans* $\mathbb{F}_{p^{12}}$.

Nous venons de présenter notre variante de la chaîne d'addition de Scott *et al.* , nous pouvons alors les comparer ensemble dans le tableau suivant.

Méthode	Complexité			Var. Temp
	S_{12}	M_{12}	F_{12}	
Chaîne d'addition de Scott [SBC+09]	$3l_u + 1$	$3w_u + 10$	7	9
Notre chaîne d'addition	$3l_u + 3$	$3w_u + 10$	6	5

TABLE 3.11 – Comparaison entre la méthode de Scott *et al.* et notre variante.

Exemple 3.9. *Pour avoir une comparaison concrète, nous choisissons de comparer les deux méthodes de la chaîne d'additions en utilisant le paramètre u donné dans l'exemple 3.1. Nous obtenons le Tableau 3.12 suivant.*

Méthode	Complexité	Temp. var.
Chaîne d'addition de Scott [SBC+09]	190 S_{12}+ 19M_{12} + 7F_{12}	9
Notre chaîne d'addition	192 S_{12}+ 19M_{12} + 6F_{12}	5

TABLE 3.12 – Exemple pour la comparaison du Tableau 3.11

Remarquons que nous devons effectuer 2 carrés de plus que la méthode de Scott *et al.* [SBC+09] mais aussi nous gagnons un Frobenius. Cela signifie que notre nouvelle version de la chaîne d'addition est légèrement plus coûteuse que l'originale.
Cependant, nous n'avons utilisé dans notre algorithme que 5 variables temporaires au lieu des 9 qui sont utilisées par Scott *et al.* . Nous gagnons alors 4 variables temporaires dans $\mathbb{F}_{p^{12}}$, ce qui est un gain très important pour les implémentations dans des environnements restreints.

3.3.3 Variante de la méthode de Fuentes

Dans cette section, nous allons présenter notre première variante de la méthode de Fuentes *et al.* présentée dans [CKH11]. Nous allons reprendre le multiple de la partie difficile de l'exponentiation finale $d'(u)$ et nous allons le présenter d'une manière différente de celle de Fuentes *et al.* afin de réduire le nombre de variables temporaires nécessaires pour ce calcul.
L'idée principale de notre développement est de faire apparaître $6u^2+1$ dans l'expression de α_2 et de α_0, et par la suite d'écrire α_0, α_1 et α_3 en fonction de α_2.
Nous représentons les α_i avec $0 \leqslant i \leqslant 3$ comme suit :

$$\begin{aligned}
\alpha_2 &= 12u^3 + 6u^2 + 6u \\
&= \left(6u^2+1\right)(2u+1) + 4u - 1 \\
\alpha_1 &= \alpha_2 - 2u \\
\alpha_0 &= \alpha_2 + 6u^2 + 1 \\
\alpha_3 &= \alpha_1 - 1.
\end{aligned}$$

Nous n'allons pas évaluer $f^{d'(u)}$ directement comme dans le cas de Fuentes *et al.*. Notre algorithme de calcul de la variante de Fuentes *et al.* est donné dans le tableau 3.13 :

Notre variante de Fuentes *et al.*	**Termes calculés**	**Coût**
Input : f, u		
Output : $f^{s\frac{p^4-p^2+1}{r}}$		
Var. Temp : t_0, t_1, t_2, t_3		
$t_0 \leftarrow f^{-u}$		$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_0 \leftarrow t_0^2$		S_{12}
$t_2 \leftarrow t_0^{-u}$	f^{2u^2}	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_1 \leftarrow t_2^2$		S_{12}
$t_2 \leftarrow t_2 t_1$	f^{6u^2}	M_{12}
$t_2 \leftarrow t_2 f$	f^{6u^2+1}	M_{12}
$t_1 \leftarrow t_2^{-2u-1}$	$\left(f^{6u^2+1}\right)^{-2u-1}$	$l_u S_{12}+w_u M_{12}$
$t_3 \leftarrow t_1^{-1}$		
$t_1 \leftarrow t_0^2$	f^{-4u}	
$t_1 \leftarrow t_1 f$		M_{12}
$t_1 \leftarrow t_1^{-1}$	f^{4u-1}	
$t_1 \leftarrow t_1 t_3$	f^{α_2}	M_{12}
$t_0 \leftarrow t_0 t_1$	$f^{\alpha_1}=f^{\alpha_2-2u}$	M_{12}
$t_2 \leftarrow t_2 t_1$	$f^{\alpha_0}=f^{\alpha_2+6u^2+1}$	M_{12}
$t_3 \leftarrow t_1^{p^2}$		F_{12}
$t_2 \leftarrow t_2 t_3$	$(f^{\alpha_2})^{p^2}\, f^{\alpha_0}$	M_{12}
$t_3 \leftarrow f^{-1}$		
$t_3 \leftarrow t_0 t_3$	$f^{\alpha_3}=f^{\alpha_1-1}$	M_{12}
$t_1 \leftarrow t_3^{p^3}$	$(f^{\alpha_3})^{p^3}$	F_{12}
$t_2 \leftarrow t_2 t_1$		M_{12}
$t_1 \leftarrow t_0^p$	$(f^{\alpha_1})^p$	F_{12}
$t_1 \leftarrow t_2 t_1$	$f^{d'}$	M_{12}
return t_1		

TABLE 3.13 – Notre Variante de la méthode de Fuentes *et al.*

Pour cet algorithme présenté dans le Tableau 3.13, nous n'avons utilisé que 4 variables temporaires dans $\mathbb{F}_{p^{12}}$.
Puisque $-2u-1$ est un entier de longueur l_u+1 bits et de poids de Hamming au plus égal à w_u+1 d'après le Théorème 3.1, nous aurons besoin au plus de w_u multiplications et l_u carrés sur $\mathbb{F}_{p^{12}}$ pour calculer le terme t_2^{-2u-1}. D'une manière générale, le coût du calcul de $f^{d'}$ est donné et détaillé étape par étape dans le Tableau 3.13. Le coût total de cet algorithme est alors, $3l_uS_{12}$ et $3F_{12}$ et au plus nous avons $(3w_u+8)M_{12}$.

Exemple 3.10. *Avec la valeur de u choisie dans l'Exemple 3.1, le coût de notre première variante de la méthode de Fuentes, le nouveau développement de $d'(u)$, est $16M_{12}+189S_{12}+3F_{12}$.*
Pour pouvoir calculer $f^{d'(u)}$, nous n'aurons besoin que de 4 variables temporaires dans $\mathbb{F}_{p^{12}}$.

3.3.4 Notre nouveau multiple

Comme nous l'avons indiqué dans l'introduction de ce chapitre, notre but initial est de réduire le nombre de variables temporaires utilisées pour calculer la partie difficile de l'exponentiation finale. Nous avons donc cherché d'autres multiples de l'exposant intervenant dans la partie difficile de l'exponentiation finale. Nous imposons une nouvelle contrainte lors de notre recherche dans la base LLL [LLL82] : au moins l'une des expressions des α_i avec $0 \leqslant i \leqslant 3$ est constante. Avec cette contrainte, nous n'aurons pas besoin de calculer et stocker tous les α_i et nous n'aurons ainsi besoin de moins de variables temporaires.
Notre recherche nous a permis de trouver 4 multiples où l'une des expressions des α_i est constante. Suite à une étude comparative, nous considérons celui des 4, noté d_1, qui coûte le moins cher en termes de complexité :

$$d_1 = \alpha_0 + \alpha_1 p + \alpha_2 p^2 + \alpha_3 p^3 = s_1 d,$$

avec

$$\begin{cases} s_1 = 36u^3 + 18u^2 + 6u + 1 \\ \alpha_0 = 6u^2 + 1 \\ \alpha_1 = 1 \\ \alpha_2 = 36u^3 + 24u^2 + 18u + 1 \\ \alpha_3 = 36u^3 + 18u^2 + 12u - 2. \end{cases}$$

Le développement suivant est choisi car les trois exposants $6u^2+1$, $6u-1$ et $6u+4$ sont utilisés plusieurs fois ce qui fait que nous allons les calculer juste une fois et les réutiliser quand c'est nécessaire. Nous écrivons :

$$\begin{aligned} \alpha_2 &= 36u^3 + 24u^2 + 18u + 2 \\ &= (6u+4)\left(6u^2+1\right) + 2\,(6u-1) - 1 \\ \alpha_3 &= 36u^3 + 18u^2 + 12u - 2 \\ &= (6u+4)\left(6u^2+1\right) - \left(6u^2+1\right) + (6u-1) - 4 \\ \alpha_0 &= 6u^2 + 1 \\ \alpha_1 &= 1. \end{aligned}$$

Ainsi, f^{d_1} devient

$$\begin{aligned} f^{s_1 \frac{p^4-p^2+1}{r}} = f^{d_1} &= f^{\alpha_0} f^{\alpha_1 p} f^{\alpha_2 p^2} f^{\alpha_3 p^3} \\ &= f^{6u^2+1} f^{p} \left(f^{(6u^2+1)(6u+4)}\right)^{p^2} \left(f^{2(6u-1)} f^{-1}\right)^{p^2} \left(f^{(6u^2+1)(6u+4)}\right)^{p^3} \\ &\quad \left(f^{-(6u^2+1)}\right)^{p^3} \left(f^{6u-1} f^{-4}\right)^{p^3}. \end{aligned} \tag{3.4}$$

Les étapes de calcul du multiple de la partie difficile de l'exponentiation finale sont alors détaillées dans l'algorithme présenté dans le Tableau 3.14 :

Nouveau multiple de d	**Termes calculés**	**Coût**
Input : $f,\ u$		
Output : f^{d_1}		
Var.Temp t_0, t_1, t_2		
$t_0 \leftarrow f^{-u}$		$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_0 \leftarrow t_0^2;\ t_1 \leftarrow t_0^2;\ t_0 \leftarrow t_0 t_1$	f^{-6u}	$2S_{12}+M_{12}$
$t_1 \leftarrow f t_0$		M_{12}
$t_2 \leftarrow f^2;\ t_2 \leftarrow t_2^2$		$2S_{12}$
$t_2 \leftarrow t_2 t_1;\ t_2 \leftarrow t_2^{-1}$	$f^{6u-1}f^{-4}$	M_{12}
$t_2 \leftarrow t_2^{p^3}$	$\left(f^{6u-1}f^{-4}\right)^{p^3}$	F_{12}
$t_1 \leftarrow t_1^2;\ t_1 \leftarrow t_1 f;\ t_1 \leftarrow t_1^{-1}$	$f^{2(6u-1)}f^{-1}$	$S_{12}+M_{12}$
$t_1 \leftarrow t_1^{p^2}$	$\left(f^{2(6u-1)}f^{-1}\right)^{p^2}$	F_{12}
$t_1 \leftarrow t_1 t_2$		M_{12}
$t_2 \leftarrow t_0^{-u}$		$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_2 \leftarrow f t_2$	f^{6u^2+1}	M_{12}
$t_1 \leftarrow t_2 t_1$		M_{12}
$t_0 \leftarrow t_2^{-1};\ t_2 \leftarrow t_0^{p^3}$	$\left(f^{-(6u^2+1)}\right)^{p^3}$	F_{12}
$t_1 \leftarrow t_1 t_2$		M_{12}
$t_2 \leftarrow t_0^{-6u-4}$		$(l_u+1)S_{12}+2w_uM_{12}$
$t_0 \leftarrow (t_2)^{p^2}$	$\left(f^{(6u^2+1)(6u+4)}\right)^{p^2}$	F_{12}
$t_1 \leftarrow t_1 t_0$		M_{12}
$t_2 \leftarrow t_0^{p}$	$\left(f^{(6u^2+1)(6u+4)}\right)^{p^3}$	F_{12}
$t_1 \leftarrow t_1 t_2$		M_{12}
$t_0 \leftarrow f^{p}$		F_{12}
$t_1 \leftarrow t_1 t_0$	f^{d_1}	M_{12}
return t_1		

TABLE 3.14 – Nouveau multiple de la partie difficile de l'exponentiation finale

Dans cet algorithme, nous n'avons utilisé que 3 variables temporaires dans $\mathbb{F}_{p^{12}}$.
Le coût de cet algorithme est donné étape par étape dans le Tableau 3.14. Comme $-6u-4$ est de longueur l_u+2 en base 2 et de poids de Hamming égal à $2w_u+1$ d'après le Théorème 3.1, nous aurons besoin alors de $2w_u$ multiplications et de l_u+1 carrés dans $\mathbb{F}_{p^{12}}$ pour calculer t_0^{-6u-4}.
Le coût total de cet algorithme est alors $3l_u+4$ carrés, $4w_u+9$ multiplications et 6 Frobenius dans $\mathbb{F}_{p^{12}}$.

Exemple 3.11. *Avec toujours la même valeur du paramètre u recommandée par Nogami* et al. *dans [NAS+08] et présenté dans l'Exemple 3.1, le coût total de calcul de la partie difficile de l'exponentiation finale avec notre nouveau multiple est* $19M_{12}$ *(grâce aux simplifications sur le poids de Hamming de* $-(6u+4)$*),* $193S_{12}$ *et* $6F_{12}$ *et nous n'avons utilisé que 3 variables temporaires dans* $\mathbb{F}_{p^{12}}$.

Nous avons présenté deux variantes qui sont basées sur la méthode de Fuentes *et al.*. La première consiste à développer le vecteur proposé dans [CKH11] d'une manière plus efficace. Quant à la

deuxième, elle consiste à trouver un nouveau multiple de $d(u)$. Pour ces deux variantes, nous visons de gagner sur la mémoire, ce qui est une contrainte importante pour des implémentations dans des environnements restreints.
Nous comparons alors la méthode de Fuentes et ses deux variantes dans le Tableau 3.15.

Méthode	Algorithme	Complexité			Var.Temp
		S_{12}	M_{12}	F_{12}	
Fuentes *et al.*	3.5	$3l_u$	$3w_u + 7$	3	5
Nouveau développement d'	3.13	$3l_u$	$3w_u + 8$	3	4
Nouveau multiple de d	3.14	$3l_u + 4$	$4w_u + 9$	6	3

TABLE 3.15 – Comparaison de la méthode de Fuentes et nos variantes

Exemple 3.12. *Pour avoir une comparaison complète, nous choisissons le paramètre u donné dans l'Exemple 3.1. Nous obtenons le Tableau 3.16.*

Méthode	Complexité	Var. Temp
La Méthode de Fuentes	$16M_{12} + 189S_{12} + 3F_{12}$	5
Notre nouveau développement de d'	$16M_{12} + 189S_{12} + 3F_{12}$	4
Notre nouveau multiple de d	$19M_{12} + 193S_{12} + 6F_{12}$	3

TABLE 3.16 – Exemple de la Table 3.15

À partir de ces tableaux, nous remarquons qu'en utilisant le nouveau développement du multiple $d'(u)$, nous avons la même complexité que la méthode de Fuentes *et al.* mais nous gagnons sur la mémoire, en effet nous avons une variable temporaire en moins.
Nous pouvons gagner plus sur le nombre des variables temporaires si nous utilisons notre nouveau multiple de la partie difficile de l'exponentiation finale que nous notons d_1. Mais malheureusement nous avons quelques opérations en plus pour cette méthode.

3.4 Écriture en $u + 1$

Jusqu'à maintenant, nous avons présenté les méthodes les plus connues dans la littérature et nos variantes pour chaque méthode de calcul de la partie difficile de l'exponentiation finale.
En général, dans les variantes que nous avons proposé, le parité du paramètre u n'influe pas sur le calcul. Cependant, nous remarquons que les valeurs de u données dans la littérature sont négatives et impaires. C'est par exemple le cas de l'exemple 3.1. Le but de cette section est d'utiliser cette caractéristique de u pour optimiser le calcul de l'exponentiation finale. Pour cela, nous allons écrire l'exposant de la partie difficile de l'exponentiation finale sous forme de développement en fonction de $u + 1$ au lieu de u.

Corollaire 3.1. *Si le paramètre u est négatif et impair, alors le poids de Hamming de $|u+1|$ vaut celui de $|u|$ moins 1. Autrement dit, dans ce cas nous avons :*

$$w_{u+1} = w_u - 1$$

Cela signifie qu'une exponentiation par $u+1$, que nous notons E_{u+1} coûte moins cher qu'une exponentiation par u notée par E_u.

3.4.1 Sur le calcul de $d'(u)$

Nous allons écrire le vecteur $d'(u)$, le multiple de la partie difficile de l'exponentiation finale présenté par Fuentes *et al.* dans [CKH11], en termes de $u+1$ au lieu de u. Nous obtenons

$$\begin{aligned} d'(u) &= (12u^2(u+1) - 6u^2 + 4u - 1)p^3 + (12u^2(u+1) - 6u^2 + 6u)p^2 \\ &\quad + (12u^2(u+1) - 6u^2 + 4u)p + (12u^2(u+1) + 6u + 1), \\ &= \alpha_3 p^3 + \alpha_2 p^2 + \alpha_1 p + \alpha_0. \end{aligned}$$

Ensuite, nous présentons les α_i avec $0 \leqslant i \leqslant 3$ d'une manière plus simple. Nous choisissons l'écriture suivante :

$$\begin{cases} \alpha_0 = (12u^2(u+1) + 6u) + 1 = c + 1 \\ \alpha_1 = \alpha_2 - 2u \\ \alpha_2 = c - 6u^2 \\ \alpha_3 = \alpha_1 - 1 \end{cases}$$

Ce développement admet deux avantages importants. Le premier est que nous avons à effectuer une exponentiation par $u+1$. Cette exponentiation nous permet d'économiser deux multiplications dans $\mathbb{F}_{p^{12}}$. En effet, si nous voulons calculer f^{u+1} à partir de f^u, nous calculons $f^{u+1} = f^u \times f$. Cette exécution coûte une exponentiation par u et une multiplication dans $\mathbb{F}_{p^{12}}$. Par contre, le calcul direct de f^{u+1} ne nous coûte qu'une exponentiation par $u+1$. Nous savons qu'une exponentiation par $u+1$, que nous notons E_{u+1}, n'est autre qu'une exponentiation par u, notée par E_u, moins une multiplication. De ce fait, nous économisons deux multiplications dans le corps fini $\mathbb{F}_{p^{12}}$.

Exemple 3.13. *Si nous reprenons le paramètre u donné dans l'exemple 3.1, nous déduisons que :*

- *Le calcul direct de f^{u+1} coûte exactement 1 multiplication et 62 carrés dans $\mathbb{F}_{p^{12}}$.*
- *Par contre, si nous calculons f^{u+1} en passant par f^u, c'est-à-dire $f^{u+1} = f^u \times f$, il nous faut 3 multiplications et 62 carrés dans $\mathbb{F}_{p^{12}}$.*

Le deuxième avantage pour notre développement est que nous avons réussi à écrire les variables α_i, $0 \leqslant i \leqslant 3$ l'une en fonction de l'autre. Par exemple, pour calculer α_3, il suffit de la déduire à partir de α_1 en la multipliant par f^{-1}. Ce développement nous permet de réduire le nombre des variables temporaires nécessaires.
Nous avons détaillé notre méthode de calcul de $f^{d'(u)}$ dans le Tableau 3.17.

Écriture en $u+1$	**Termes calculés**	**Coût**
Input : f, u, p		
Output : $f^{d'(u)}$		
Var. Tempt_0, t_1, t_2, t_3		
$t_0 \leftarrow f^{-u}$		E_u
$t_1 \leftarrow t_0^2$	f^{-2u}	S_{12}
$t_0 \leftarrow t_1^2$	f^{-4u}	S_{12}
$t_0 \leftarrow t_1 t_0$	f^{-6u}	M_{12}
$t_2 \leftarrow t_0^{-u}$	f^{6u^2}	E_u
$t_3 \leftarrow t_2^{u+1}$	$f^{6u^3+6u^2}$	E_{u+1}
$t_3 \leftarrow t_3^2$	$f^{12u^3+12u^2}$	S_{12}
$t_0 \leftarrow t_0^{-1}$	f^{6u}	
$t_3 \leftarrow t_3 t_0$	$f^{12u^3+12u^2+6u}$	M_{12}
$t_2 \leftarrow t_2^{-1}$	f^{-6u^2}	
$t_2 \leftarrow t_3 t_2$	$f^{12u^3+6u^2+6u} = f^{\lambda_2}$	M_{12}
$t_3 \leftarrow t_3 f$	$f^{12u^3+12u^2+6u+1} = f^{\lambda_0}$	M_{12}
$t_1 \leftarrow t_1 t_2$	f^{λ_1}	M_{12}
$t_0 \leftarrow f^{-1}$		
$t_0 \leftarrow t_1 t_0$	f^{λ_3}	M_{12}
$t_0 \leftarrow t_0^{p^3}$	$f^{\lambda_3 p^3}$	F_{12}
$t_0 \leftarrow t_0 t_3$		M_{12}
$t_1 \leftarrow t_1^p$	$f^{\lambda_1 p}$	F_{12}
$t_0 \leftarrow t_0 t_1$		M_{12}
$t_2 \leftarrow t_2^{p^2}$	$f^{\lambda_2 p^2}$	F_{12}
$t_0 \leftarrow t_0 t_2$	$f^{\lambda_0} f^{\lambda_1 p} f^{\lambda_2 p^2} f^{\lambda_3 p^3}$	M_{12}
returnt_0	$f^{d'(u)}$	$2E_u + E_{u+1} + 9M_{12}$
		$+3S_{12} + 3F_{12}$

TABLE 3.17 – Écriture de $d'(u)$ en $u+1$

Le coût de chaque opération de cet algorithme est présenté dans le Tableau 3.17. Par conséquent le coût total du calcul de la partie difficile de l'exponentiation finale, en utilisant notre développement en $u+1$, est $2E_u + E_{u+1} + 9M_{12} + 3S_{12} + 3F_{12}$. Pour cet algorithme, nous avons utilisé 4 variables temporaires dans $\mathbb{F}_{p^{12}}$.

Exemple 3.14. *En utilisant la même valeur du paramètre u donné dans l'Exemple 3.1, le coût de notre méthode de calcul de* $f^{d'(u)}$ *est* $14M_{12} + 189S_{12} + 3F_{12}$.

Remarque 3.6. *Nous remarquons que dans notre développement du multiple de d, nous n'avons effectué qu'une seule exponentiation par* $u+1$. *Nous avons essayé de développer* $d'(u)$ *afin d'avoir plus d'une exponentiation par* $u+1$ *mais cela n'était pas possible vu que les* α_i, $0 \leqslant i \leqslant 3$, *n'ont pas les mêmes coefficients.*
Nous avons aussi essayé de trouver un autre multiple de la partie difficile de l'exponentiation finale $\frac{p^4-p^2+1}{r}$ *où nous pouvons appliquer d'avantage d'exponentiations par* $u+1$ *mais sans plus de succès. Dans notre recherche, nous avons imposé le fait que les coefficients de chaque puissance de u dans*

les α_i soient les mêmes. Une recherche par force brute des combinaisons linéaires dans la base LLL [LLL82] n'a rien donné. C'est-à-dire qu'il n'existe aucun multiple qui répond à nos exigences. Il parait donc difficile d'effectuer plus d'une exponentiation par $u+1$

Nous avons présenté dans cette section une nouvelle variante de la méthode de Fuentes. Nous pouvons alors comparer les deux méthodes dans le tableau suivant :

Méthode	Complexité	Var.Temp.
Méthode de Fuentes *et al.* [CKH11]	$3E_u + 3S_{12} + 10M_{12} + 3F_{12}$	5
Écriture en $u+1$	$2E_u + E_{u+1} + 3S_{12} + 9M_{12} + 3F_{12}$	4

TABLE 3.18 – Comparaison de notre écriture en $u+1$ avec la méthode de Fuentes *et al.*

Pour avoir une comparaison explicite, nous considérons le paramètre u présenté dans l'Exemple 3.1. Nous obtenons la comparaison suivante :

Méthode	Complexité	Var. Temp.
Méthode de Fuentes *et al.* [CKH11]	$16M_{12} + 189S_{12} + 3F_{12}$	5
Écriture en $u+1$	$14M_{12} + 189S_{12} + 3F_{12}$	4

TABLE 3.19 – Exemple du tableau 3.18

3.4.2 Remarque sur l'algorithme de Miller

Une question qui peut se poser est la suivante : pourquoi nous n'utilisons pas le développement par $u+1$ dans le calcul de l'algorithme de Miller.
Dans cette Section, nous sommes intéressés par le calcul de la boucle de Miller pour le couplage Optimal Ate recommandé sur les courbes BN pour le niveau de sécurité 128 bits.
Rappelons que la boucle de Miller d'un couplage Optimal Ate sur les courbes BN est donné par le calcul de la fonction

$$\left((f_{6u+2,Q}(P) l_{[6u+2]Q,[p]Q}(P) l_{[6u+2]Q,[-p^2]Q}(P))\right)$$

En premier lieu, nous devons calculer la fonction rationnelle $f_{6u+2,Q}$ évaluée au point P. Ce calcul se fait directement grâce à l'algorithme de Miller (voir Chapitre 1).
Nous souhaitons dans cette section écrire le paramètre de la boucle de Miller $6u+2$ à l'aide du $u+1$.

$$6u + 2 = 6(u + 1) - 4$$

On aura alors

$$f_{6u+2} = f_{6(u+1)} \times f_{-4} \times \ell_{[6(u+1)]Q,[-4]Q}$$

Remarque 3.7. *Dans notre contexte, le cas du couplage Optimal Ate sur les BN, les droites verticales seront simplifiés par l'exponentiation finale.*

Malheureusement, en fonction du poids de Hamming de $6u+2$ ainsi que sa longueur en base 2, nous avons réalisé qu'il vaut mieux calculer directement f_{6u+2} plutôt que de le calculer en utilisant les développements par $u+1$. (Nous rappelons que la complexité de l'algorithme de Miller est basée

essentiellement sur le poids de Hamming et la longueur en base 2 du paramètre de la boucle de Miller.)
En effet nous présentons l'exemple suivant :

Exemple 3.15. *Si nous utilisons le paramètre u, donné dans l'Exemple 3.1, le calcul de f_{6u+2} nécessite 4 multiplications creuses et 64 carrés dans $\mathbb{F}_{p^{12}}$ (puisque $6u+2$ est un entier de 65 bits et de poids de Hamming égal à 5). Par contre le calcul de $f_{6(u+1)}.f_{-4}$ requiert 3 multiplications creuses, , 66 carrés et une multiplication dans le corps fini $\mathbb{F}_{p^{12}}$ et de plus l'évaluation de $\ell_{[6(u+1)]Q,[-4]Q}(P)$ et aussi une multiplication creuse supplémentaire.*

3.5 Comparaison

Dans ce chapitre, nous avons rappelé les méthodes existantes dans la littérature permettant de calculer la partie difficile de l'exponentiation finale. Pour chaque méthode nous avons rappelé la complexité du calcul et nous avons aussi compté le nombre de variables temporaires nécessaires. Puis, nous avons présenté nos variantes qui sont toutes beaucoup moins gourmandes en mémoire que les anciennes méthodes et qui, dans plusieurs cas, sont aussi plus efficaces.

Nous allons maintenant récapituler toutes ces méthodes dans le tableau comparatif 3.20 où nous donnons le coût du calcul de la partie difficile de l'exponentiation finale ainsi que le nombre de variables temporaires dans $\mathbb{F}_{p^{12}}$ utilisées.
Cette comparaison est donnée en terme d'opérations dans le corps fini $\mathbb{F}_{p^{12}}$. Nous donnons, pour chaque méthode et sa variante, le coût pour tout paramètre u en général, puis nous donnons la complexité en considérant l'exemple 3.1 de Nogami *et al.*.

La Méthode	Algorithme dans la table	Complexité			Complexité en utilisant u de l'exemple 3.1	Var.Temp dans $\mathbb{F}_{p^{12}}$
		S_{12}	M_{12}	F_{12}		
Naïve	3.1	$12l_u$	$w(\frac{p^4-p^2+1}{r})$		$759S_{12}+335M_{12}$	1
Devigili *et al.*	3.2	$3l_u+7$	$w_u^2+3w_u+10$	4	$196S_{12}+26M_{12}$ $+4F_{12}$	4
Notre variante	3.6	$\mathbf{3l_u+5}$	$\mathbf{4w_u+12}$	**6**	$\mathbf{194S_{12}+21M_{12}}$ $\mathbf{+6F_{12}}$	**3**
Chaîne d'addition de Scott	3.4	$3l_u+1$	$3w_u+10$	7	$190S_{12}+19M_{12}$ $+7F_{12}$	9
Notre variante	3.10	$\mathbf{3l_u+3}$	$\mathbf{3w_u+10}$	**6**	$\mathbf{192S_{12}+19M_{12}}$ $\mathbf{+6F_{12}}$	**5**
Fuentes *et al.*	3.5	$3l_u$	$3w_u+7$	3	$189S_{12}+16M_{12}$ $+3F_{12}$	5
Notre variante	3.13	$\mathbf{3l_u}$	$\mathbf{3w_u+8}$	**3**	$\mathbf{189S_{12}+16M_{12}}$ $\mathbf{+3F_{12}}$	**4**
Notre nouveau multiple	3.14	$\mathbf{3l_u+4}$	$\mathbf{4w_u+9}$	**6**	$\mathbf{193S_{12}+19M_{12}}$ $\mathbf{+6F_{12}}$	**3**
Écriture en $u+1$	3.17	$\mathbf{3l_u}$	$\mathbf{3w_u+5}$	**3**	$\mathbf{189S_{12}+14M_{12}}$ $\mathbf{+3F_{12}}$	**4**

TABLE 3.20 – Comparaison des méthodes de calcul de la partie difficile de l'exponentiation finale.

Pour que la comparaison soit plus explicite et précise, nous allons la présenter en termes de variables temporaires et de complexité dans le corps fini $\mathbb{F}_p$. Pour cela, nous choisissons le paramètre u de l'exemple 3.1 et la courbe elliptique E définie sur le corps fini $\mathbb{F}_p$ par l'équation :

$$E : y^2 = x^3 + 2.$$

Dans ce cas, il est facile de vérifier que -1 n'est pas un carré et $(1+i)$ n'est ni un carré ni un cube. Alors l'extension $\mathbb{F}_{p^{12}}$ est construite en utilisant la tour d'extension suivante :

$$\begin{aligned} \mathbb{F}_{p^2} &= \mathbb{F}_p[\mathbf{i}] \text{ avec } \mathbf{i}^2 = -1 \\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[\beta] \text{ avec } \beta^3 = 1+\mathbf{i} \\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[\gamma] \text{ avec } \gamma^2 = \beta. \end{aligned}$$

Dans ce cas, les coûts des opérations arithmétiques dans les corps finis $\mathbb{F}_p$, $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^6}$ et $\mathbb{F}_{p^{12}}$ sont donnés dans le chapitre 2 et nous les rappelons dans le tableau 3.21.

Opération	Notation	Coût dans $\mathbb{F}_p$
Multiplication dans $\mathbb{F}_p$	M	M
Carré dans $\mathbb{F}_p$	S	S
Inversion dans $\mathbb{F}_p$	I	I
Multiplication dans $\mathbb{F}_{p^2}$	M_2	$3\,M$
Carré dans $\mathbb{F}_{p^2}$	S_2	$2\,M$
Multiplication dans $\mathbb{F}_{p^6}$	M_6	$18\,M$
Carré dans $\mathbb{F}_{p^6}$	S_6	$12\,M$
Inversion dans $\mathbb{F}_{p^6}$	I_6	$37\,M + I$
Multiplication dans $\mathbb{F}_{p^{12}}$	M_{12}	$54\,M$
Carré cyclotomique dans $\mathbb{F}_{p^{12}}$	S_{12}	$18\,M$
Compression du carré dans $\mathbb{F}_{p^{12}}$	S'_{12}	$12\,M$
Décompression du carré dans $\mathbb{F}_{p^{12}}$	D'_{12}	$12\,M + I$
p, p^3 dans $\mathbb{F}_{p^{12}}$	F_{12}	$15\,M$
p^2 dans $\mathbb{F}_{p^{12}}$	F_{12}	$8\,M$

TABLE 3.21 – Les coûts des opérations pour la tour d'extension choisie

Rappelons que nous ne sommes pas intéressés que par la complexité du calcul de l'exponentiation finale des couplages mais aussi par les ressources mémoire utilisées pour l'implémentation matérielle dans un environnement restreint. C'est pour cela que nous devons également présenter le nombre des variables temporaires nécessaires dans $\mathbb{F}_p$ pour calculer la partie difficile de l'exponentiation finale pour chaque algorithme. Notre estimation est basée sur le fait que toute variable temporaire dans le corps fini $\mathbb{F}_{p^{12}}$ est représentée par 12 variables temporaires dans $\mathbb{F}_p$. De plus, nous avons besoin de rajouter d'autres variables temporaires intermédiaires. En effet, lorsque nous calculons par exemple le produit de deux éléments de $\mathbb{F}_{p^{12}}$, nous aurons besoin de variables temporaires dans $\mathbb{F}_p$ en plus des 24 utilisées pour les 2 opérandes. Par exemple, dans notre travail, nous avons considéré la multiplication de Karatsuba. Dans ce cas, nous devons rajouter 10 variables temporaires dans $\mathbb{F}_p$ afin de multiplier deux éléments de $\mathbb{F}_{p^{12}}$. De la même manière, pour le calcul d'un carré ou d'un Frobenius dans $\mathbb{F}_{p^{12}}$, nous devons rajouter des variables temporaires supplémentaires, mais dans tous les cas moins que 10. Cela nous permet d'affirmer qu'au plus, nous rajoutons au nombre total de variables temporaires 10 variables dans $\mathbb{F}_p$.

Remarque 3.8. *L'algorithme de la multiplication dans $\mathbb{F}_{p^{12}}$ présenté dans [BGM+10] nécessite* 14 *variables temporaires supplémentaires dans $\mathbb{F}_p$. Nous l'avons modifié un peu de façon à en économiser 4.*

Bien entendu, nous devons aussi stocker l'entrée de l'exponentiation finale (le résultat de la boucle de Miller f_1), qui est un élément de $\mathbb{F}_{p^{12}}$ et nécessite donc 12 variables temporaires dans $\mathbb{F}_p$. Cela signifie que nous devons rajouter en tout 22 variables temporaires dans $\mathbb{F}_p$ au nombre total des variables temporaires utilisées dans chaque algorithme de calcul de la partie difficile de l'exponentiation finale pour connaître le coût total en ressources mémoires.
Autrement dit, pour calculer le nombre total de variables temporaires dans le corps fini $\mathbb{F}_p$, nous devons en premier lieu multiplier le nombre de variables temporaires utilisées dans $\mathbb{F}_{p^{12}}$ par 12, puis nous rajouter à ce nombre les 22 variables temporaires supplémentaires (12 pour f_1 et 10 intervenant dans le calcul intermédiaire lors d'une multiplication).

Pour présenter une comparaison explicite entre les méthodes existantes et nos variantes, nous allons distinguer deux cas.

— **Premier cas :** nous supposons que les carrés dans $\mathbb{F}_{p^{12}}$ sont effectués par la méthode de Granger et Scott [Sco05]. Nous obtenons alors le tableau comparatif 3.22 :

Méthode	Algorithme présenté dans	Coût dans $\mathbb{F}_p$	Gain sur le coût	Var. Temp dans $\mathbb{F}_p$	Gain sur la mémoire
Naïve	3.1	$31752\,M$		34	
Devigili *et al.*	3.2	$4992\,M$		70	
Notre variante	3.6	$\mathbf{4716\,M}$	**5.5**%	**58**	**17**%
Chaîne de Scott	3.4	$4551\,M$		130	
Notre variante	3.10	$\mathbf{4572\,M}$	**−0.4**%	**82**	**37**%
Fuentes *et al.*	3.5	$4311\,M$		82	
Notre variante	3.13	$\mathbf{4311\,M}$	**0**%	**70**	**15**%
Notre nouveau multiple	3.14	$\mathbf{4590\,M}$	**−6.4**%	**58**	**29**%
Écriture en $u+1$	3.17	$\mathbf{4203\,M}$	**2.5**%	**70**	**15**%

TABLE 3.22 – Comparaison en utilisant le carré de Granger et Scott

— **Deuxième cas :** nous allons plutôt considérer la technique du calcul du carré présentée par Karabina [Kar13]. Nous présentons alors le Tableau comparatif 3.23 :

Méthode	Algorithme présenté dans	Coût dans $\mathbb{F}_p$	Gain sur le coût	Var. Temp dans $\mathbb{F}_p$	Gain sur la mémoire
Naïve	3.1	$25671\,M$		34	
Devigili *et al.*	3.2	$3938\,M + 3\,I$		70	
Notre variante	3.6	$\mathbf{3711\,M + 3\,I}$	**5.8**%	**58**	**17**%
Chaîne de Scott	3.4	$3366\,M + 3\,I$		130	
Notre variante	3.10	$\mathbf{3363\,M + 3\,I}$	**0.1**%	**82**	**37**%
Fuentes *et al.*	3.5	$3324\,M + 3\,I$		82	
Notre variante	3.13	$\mathbf{3318\,M + 3\,I}$	**0.2**%	**70**	**15**%
Notre nouveau multiple	3.14	$\mathbf{3591\,M + 3\,I}$	**−8**%	**58**	**29**%
Écriture en $u+1$	3.17	$\mathbf{3212\,M + 3\,I}$	**3.4**%	**70**	**15**%

TABLE 3.23 – Comparaison en utilisant le carré de Karabina

Ces deux Tableaux 3.22 et 3.23 illustrent bien le fait que nos variantes de calcul de la partie difficile de l'exponentiation finale sont moins gourmandes en mémoire, ce qui est très important dans le cas des implémentations dans des environnement restreints. Nous sommes aussi dans plusieurs cas plus efficaces en termes de complexité. Si nous prenons par exemple la première méthode optimisée dans la littérature, celle de Barreto *et al.* [BLS02], notre gain sur la complexité est de 5.5% si nous utilisons le carré cyclotomique. Si nous utilisons la méthode de Karabina pour évaluer

les carrés, notre gain est légèrement plus important. Pour cette première variante notre gain sur la mémoire est de 17%. Puis, la méthode la plus utilisée en pratique est celle basée sur les chaînes d'additions présentée par Scott *et al.* dans [Sco05] qui nécessite un important nombre de variables temporaires. Grâce à notre variante, ce n'est plus le cas. Nous arrivons à gagner 37% sur la mémoire pour presque la même complexité. Pour cela, notre chaîne d'addition est un bon candidat pour être utilisée lors des implémentations matérielles des couplages dans des environnements restreints.
Récemment la méthode de Fuentes *et al.* est devenue très utilisée pour calculer l'exponentiation finale des couplages. Ce choix est dû au fait que cette méthode présente une complexité meilleure que les autres algorithmes même si son principe est plutôt de calculer une puissance de l'exponentiation finale et donc du couplage. Le résultat est toujours un couplage, mais dans certains cas, cela peut être un inconvénient, notemment en terme d'interopérabilité. Nous avons présenté des variantes de cette méthode. Comme le montrent les Tableaux 3.22 et 3.23 nous utilisons toujours moins des variables temporaires dans $\mathbb{F}_p$. Quant à la complexité, pour notre première variante, où nous avons utilisé le même vecteur que Fuentes *et al.*, nous avons gardé une complexité similaire. Dans la deuxième méthode, où nous avons choisit un nouveau multiple de l'exposant de la partie difficile de l'exponentiation finale, notre gain est de 29% en terme de ressources mémoire. Mais malheureusement nous sommes moins efficaces puisque nous sommes plus lents dans le calcul de l'exponentiation finale. Par contre, en écrivant les exposants en terme de $(u+1)$, nous devenons plus rapides que toutes les méthodes décrites dans la littérature tout en consommant moins de ressources mémoire et ce quelque soit l'algorithme choisi pour les carrés cyclotomiques.

Le problème des ressources mémoire devient très important dès lors que nous souhaitons implémenter le couplage Optimal Ate dans un environnement restreint tel que les cartes à puce ou les cartes SIM. Les variantes que nous venons de présenter sont toutes une réponse adéquate à ce problème. Nous illustrons ce propos avec l'exemple suivant provenant d'une problématique réelle soulevée à l'occasion du projet ANR SIMPATIC :

Exemple 3.16. *Si nous souhaitons implémenter le couplage Optimal Ate, défini sur une courbe BN de 256 bits en utilisant la méthode de Scott* et al. *, il nous faut exactement* 130 *variables temporaires dans* $\mathbb{F}_p$ *(Voir Tableau 3.22). Or toute variable temporaire dans* $\mathbb{F}_p$ *nécessite* $\frac{256}{8} = 32$ *octets. Donc pour implémenter cette méthode de calcul de la partie difficile de l'exponentiation finale il nous faut au moins* $32 \times 130 = 4.16Ko$ *de mémoire. Cela signifie que nous ne pouvons pas implémenter le couplage Optimal Ate dans un environnement restreint de capacité mémoire 2 ou 4Ko (qui sont des capacités classiques pour une carte à puce) Par contre, si nous utilisons notre nouvelle chaîne d'addition, nous ne consommons que* $32 \times 82 = 2.6Ko$. *Nous passons donc du non implémentable à l'implémentable.*
De plus nous avons présenté d'autres variantes qui nécessitent environ 2 Ko de mémoire. C'est le cas de la variante de Devigili et al. *ainsi que notre nouveau multiple de la partie difficile de l'exponentiation finale.*

Le problème des ressources mémoires a été récemment étudié dans [UW14]. Unterluggauer *et al.* y proposent une variante de la méthode de Fuentes *et al.* optimisée en terme de ressources mémoire. Leur algorithme nécessite une multiplication et un Frobenius dans $\mathbb{F}_{p^{12}}$ de plus que Fuentes *et al.* mais ils n'ont besoin que de 3 variables temporaires dans $\mathbb{F}_{p^{12}}$. Cependant ce résultat a été obtenu en supposant que nous pouvons faire une exponentiation par u en utilisant une seule variable temporaire qui est celle d'entrée. Malheureusement, ce n'est pas le cas en pratique (car

une exponentiation par u se fait en utilisant les algorithmes d'exponentiations rapides où il est nécessaire d'utiliser au moins deux variables temporaires dans le corps fini $\mathbb{F}_{p^{12}}$ en comptant la variable temporaire l'entrée qui est dans notre cas f). Dans ce contexte, Unterluggauer *et al.* doivent utiliser en pratique plutôt 4 variables temporaires dans $\mathbb{F}_{p^{12}}$.
Ainsi, notre variante de Fuentes *et al.* est plus efficace que leur méthode proposée dans [UW14].

Remarque 3.9. *Dans ce chapitre, nous réduisons la mémoire utilisée pour calculer la partie difficile de l'exponentiation finale afin d'implémenter le couplage Optimal Ate dans un environnement restreint sans avoir parlé de l'algorithme de Miller. En effet, l'algorithme de Miller ne nécessite pas plus que 3 variables temporaires dans $\mathbb{F}_{p^{12}}$. Une variable pour notre entrée f_1, une deuxième pour le calcul des droites et la mise à jour des points. En effet, le calcul de la droite d'addition de deux points ainsi que de celle de doublement d'un point ne nécessite que* 6 *variables dans $\mathbb{F}_p$ pour être représenté et* 6 *variables pour le calcul intermédiaire. Ces* 6 *variables temporaires qui interviennent dans le calcul intermédiaire des droites sont réutilisables pour la mise à jour des points $T \longleftarrow [2]T$ et $T \longleftarrow T + Q$. Une troisième variable temporaire dans $\mathbb{F}_{p^{12}}$ est nécessaire (en réalité 10 variables dans $\mathbb{F}_p$ suffisent) pour le calcul intermédiaire d'un carré et d'une multiplication creuse dans $\mathbb{F}_{p^{12}}$. Nous réutilisons ces variables temporaires dans l'exponentiation finale, le résultat de la boucle de Miller étant stocké dans la variable f_1. Nous pouvons affirmer alors que l'exponentiation finale, plus précisément la partie difficile de l'exponentiation finale, est la plus gourmande en mémoire. Pour cette raison, notre travail est consacré à cette partie.*

3.6 Implémentation matérielle des couplages

Suite à nos résultats dans la réduction de la mémoire ainsi que la complexité du calcul du couplage Optimal Ate sur les courbes de Barreto et Naehrig pour un niveau de sécurité de 128 bits, nous avons essayé de faire une implémentation matérielle de nos variantes.
Dans cette section nous donnons les résultats de l'implémentation matérielle que nous avons effectuée au sein du Laboratoire d'Électronique et de Microélectronique de la Faculté des Sciences de Monastir sous la direction de Mohsen Machhout et avec la doctorante Anissa Sghaier. Faute de moyens, nous nous sommes contentés de faire notre implémentation sur une carte FPGA (Field-Programmable Gate Array).
Dans cette section, nous n'allons pas détailler les techniques utilisées lors de l'optimisation matérielle faite, mais nous allons juste nous intéresser au résultat final présenté dans le tableau suivant :

Designs	FPGA	Area Slices ; DSP	Freq. (MHz)	Cycle ($\times 10^3$)
Notre Design	xc6vlx240t-3	9767	105	89
Notre Design	xc6vlx240t-3	5276 ; 30	145	80
[GMC10]	xc4vlx200-12	52K	50	821
[CDF+11]	xc6vlx240t-2	7032 ; 32	250	143
[FVV12]	xc6vlx240t-3	4014 ; 42	210	245
[GVC12]	xc6vlx240t-3	5163 ; 144	166	62

TABLE 3.24 – Résultats de l'implémentation matérielle

3.7 Conclusion

Dans ce chapitre, nous avons présenté le problème de l'implémentation des couplages dans un environnement restreint. Ce problème est très important actuellement vu que les couplages sont devenus de plus en plus utilisés en cryptographie et dans ce contexte nous devons trouver un équilibre entre l'efficacité et les ressources mémoire utilisées.

Dans ce chapitre, nous avons suggéré 5 nouvelles méthodes de calcul de la partie difficile de l'exponentiation finale du couplage de Tate (et ses dérivés) sur les courbes BN. Nos méthodes sont plus efficaces dans la majorité des cas, et dans certains, légèrement moins efficaces. Elles sont toujours une meilleure solution pour les implémentations dans des environnements restreints en raisons de leur moindre utilisation de mémoire.

Nous avons présenté en premier lieu un nouveau développement de l'exposant $d = \frac{p^4-p^2+1}{r}$ qui est une variante de la méthode de Devegili *et al.* . En utilisant notre nouveau développement, nous sommes plus rapides dans le calcul de l'exponentiation finale des couplages. En outre, les ressources mémoires requises sont bien améliorées.

Par la suite, nous avons présenté la chaîne d'addition de Scott *et al.* [SBC+09]. À nouveau, nous avons gagné un important nombre de variables temporaires et nous avons une légère perte de quelques multiplications, (exactement 3 multiplications dans $\mathbb{F}_p$ ce qui est une perte négligeable, de l'ordre de 0.1%).

Ensuite, nous avons présenté une nouvelle manière d'écrire l'exposant d' proposé par Fuentes *et al.* dans [CKH11] qui nécessite moins des variables temporaires pour calculer $f^{d'}$. Nous avons aussi présenté un nouveau multiple de d que nous avons noté d_1 de telle sorte que le calcul de f^{d_1} requiert très peu des variables temporaires dans $\mathbb{F}_{p^{12}}$. Malheureusement ce nouveau multiple nécessite quelques opérations de plus que celui de Fuentes. Finalement, en jouant sur la parité du paramètre du couplage u, nous avons pu trouver un nouveau développement du vecteur de Fuentes *et al.* . Notre développement est basé sur l'utilisation des exponentiations par $(u+1)$ au lieu de u. Cette méthode est la plus efficace des méthodes existantes dans la littérature ainsi que de nos variantes. En termes de variables temporaires, cette méthode est une bonne solution pour des implémentations dans des environnements restreints.

Nous avons implémenté nos algorithmes sur Sage [S+] pour pouvoir facilement vérifier leurs exactitude [DG].

Rappelons finalement que l'implémentation hardware que nous avons effectuée, confirme l'apport

des algorithmes que nous avons présentés dans ce chapitre tant au niveau efficacité qu'au niveau consommation mémoire. Une étude détaillée de notre implémentation matérielle est présentée dans le papier [SGM+15].

Chapitre 4

Calcul des couplages pour les hauts niveaux de sécurité

Dans ce chapitre, nous nous intéressons au calcul du couplage Optimal Ate pour un niveau de sécurité élevé recommandé par le NIST [oST].
Dans la littérature, et en particulier dans [AFK+12], Aranha *et al.* ont recommandé l'utilisation de la courbe de Barreto, Lynn et Scott [BLS02] que nous notons dans la suite par BLS12, (12 étant le degré de plongement de cette courbe), pour calculer le couplage Optimal Ate pour un haut niveau de sécurité (environ 192 bits niveau de sécurité). Cependant, récemment, Kim et Barbulescu dans [KB16], ont prouvé que cette courbe elliptique ne possède pas le niveau de sécurité 192 bits, mais un niveau moindre que celui-ci grâce aux optimisations sur la résolution du problème de logarithme discret sur les corps finis.
Dans ce chapitre, nous allons présenter deux travaux qui ont donné lieu à deux publications. Dans un premier temps, nous calculons le couplage Optimal Ate sur les courbes BLS12.
Nous reprenons l'exponentiation finale de ce couplage et nous simplifions son calcul. Grâce à notre nouveau développement de la partie difficile de l'exponentiation finale, nous sommes plus efficaces que les méthodes présentées dans la littérature sur le calcul de l'exponentiation finale, par conséquent, dans le calcul du couplage Optimal Ate sur les courbes BLS12. Ensuite, et dans le même contexte, nous présentons un nouveau paramètre du couplage. Ce paramètre possède certaines caractéristiques qui favorisent l'efficacité de notre manière de calculer l'exponentiation finale. Le paramètre que nous présentons réduit également la complexité de l'algorithme de Miller.
D'autre part, nous avons optimisé le calcul du couplage Optimal Ate pour ce niveau de sécurité sur une autre catégorie des courbes elliptiques à savoir les courbes BLS24. Ce choix des courbes est aussi recommandé en pratique.

Dans un deuxième temps, nous nous intéressons au calcul du produit ou/et du quotient de n couplages. Nous étions intéressés par ce problème car il existe plusieurs protocoles cryptographiques qui nécessitent ce calcul tels que 'Identity-Based Encryption Gone Wild' [ACD+06], le *non interactive proofs system* proposé par Groth and Sahai dans [GS08] et autres.
Tout d'abord, nous présentons le dernier résultat dans la littérature qui traite ce problème. Ce résultat est présenté par Zhang *et al.* dans [ZL12] où ils affirment que la courbe elliptique la plus adéquate pour ce calcul est celle de Kachisa, Schafer et Scott [KSS07] dite la courbe KSS16 qui possède un degré de plongement k égal à 16.

Nous avons vérifié les formules données dans leur papier et nous avons constaté un souci dans le calcul de l'exponentiation finale, plus précisément la partie difficile de l'exponentiation finale. Nous avons corrigé leurs formules et nous avons donné une version correcte des formules présentées ainsi que l'algorithme permettant de calculer l'exponentiation finale du couplage Optimal Ate sur les courbes KSS16. De plus, nous avons déterminé un nouveau multiple de la partie difficile de l'exponentiation finale qui rend le calcul du couplage plus efficace en termes de consommation mémoire.

4.1 Optimal Ate sur les BLS 12

4.1.1 Les courbes de Barreto, Lynn et Scott

Barreto, Lynn et Scott ont introduit en 2001 une méthode pour générer des courbes elliptiques définies sur le corps premier $\mathbb{F}_p$ et de degré de plongement $k = 12$ [BLS02].

Définition 4.1. *La courbe BLS12 est définie sur $\mathbb{F}_p$ par l'équation :*

$$E : y^2 = x^3 + b$$

et par le paramètre $u \in \mathbb{Z}$ tel que :

$$\begin{cases} p &= (u-1)^2(u^4 - u^2 + 1)/3 + u \\ r &= u^4 - u^2 + 1 \\ t &= u + 1 \end{cases} \tag{4.1}$$

où t est la trace de Frobenius de la courbe. Le paramètre u est choisi de telle sorte que p et r soient premiers et ont une taille correspondant au niveau de sécurité désiré selon les recommandations du [oST].

En gardant les définitions de G_1, G_2 et G_3 de la Section 1.9.2, nous définissons le couplage Optimal Ate sur les courbes BLS12 comme suit :

Proposition 4.1. *[Ver10] Le couplage Optimal Ate défini sur les courbes BLS12 est l'application bilinéaire et non dégenerée suivante :*

$$\begin{array}{rccc} e_{opt} : G_1 \times G_2 & \to & G_3 \\ (P, Q) & \longmapsto & f_{u,Q}(P)^{\frac{p^{12}-1}{r}} \end{array}$$

Nous avons deux étapes à effectuer pour calculer le couplage Optimal Ate. Tout d'abord, nous calculons la boucle de Miller donc la fonction $f_{u,Q}(P)$. Puis nous élevons ce résultat à la puissance $\frac{p^{12}-1}{r}$, d'où l'exponentiation finale.
La taille et le poids de Hamming du paramètre u sont très importants dans le calcul de la complexité du couplage. Il faut considérer un u de faible poids de Hamming. Dans ce contexte, Aranha *et al.* dans [AFK+12] on proposé le paramètre de l'exemple 4.1.

Exemple 4.1. *Le paramètre u choisi par Aranha* et al. *est un entier de taille 108 bits et dont le poids de Hamming est* 4 *:*

$$u = -2^{107} + 2^{105} + 2^{93} + 2^5.$$

Ce paramètre vérifie bien un haut niveau de sécurité. En effet, p et r sont premiers et de tailles respectives 638 et 427 bits.

Remarque 4.1. *Remarquons que l'écriture du paramètre u en base deux est* $u = u_n 2^n + ... + u_1 2 + u_0$ *avec* $u_i \in \{-1, 0, 1\}$. *Il est simple d'appliquer l'algorithme de Miller pour calculer* $f_{u,Q}(P)$. *Il suffit de rajouter un deuxième test sur le bit* u_i *comme il est indiqué dans l'algorithme suivant [EM14] (le cas* $u_i = -1$ *fait intervenir la droite verticale passant par Q mais nous ne l'avons pas écrite car elle est annulée par l'exponentiation finale) :*

Algorithm 3 L'algorithme de Miller avec u signé

Require: $u = (u_n, \ldots, u_0)$ avec $u_n = 1$, P, Q
Ensure: $(f_{u,Q}(P))^{\frac{p^{12}-1}{r}}$

$f_1 \leftarrow 1$ and $R \leftarrow Q$
for $i = n - 1$ **down to** 0 **do**
  $f \leftarrow f_1^2 \ell_{R,R}(P)$
  $R \leftarrow 2R$ — Étape de doublement.
  if $u_i = 1$ **then**
    $f \leftarrow f\ell_{R,Q}(P)$
    $R \leftarrow R + Q$ — Étape d'addition.
  else if $u_i = -1$ **then**
    $f \leftarrow f\ell_{R,-Q}(P)$
    $R \leftarrow R - Q$ — Étape d'addition.
  end if
end for
return $f_1 = f^{\frac{p^k-1}{r}}$

Pour calculer $f_{u,Q}(P)$ *dans le cas de l'exemple 4.1, il est facile de vérifier que nous devons effectuer 107 étapes de doublements et 3 étapes d'additions.*

4.1.2 Le calcul de l'exponentiation finale

Nous nous intéressons dans cette partie au calcul de l'exponentiation finale du couplage Optimal Ate sur les courbes BLS12. Nous rappelons tout d'abord dans cette section la méthode de calcul de la partie difficile de l'exponentiation finale présentée par Aranha *et al.* dans [AFK+12]. Nous considérons également la mémoire utilisée pour le calcul du couplage, ce qui n'était pas étudié avant dans la littérature.
Rappelons que l'exponentiation finale du couplage défini sur une courbe elliptique, de degré de plongement égal à 12, est simplifié grâce au polynôme cyclotomique comme suit :

$$\frac{p^{12}-1}{r} = \left(p^6 - 1\right)\left(p^2 + 1\right)\frac{p^4 - p^2 + 1}{r}.$$

Nous avons deux parties à traiter : la partie facile de l'exponentiation finale en premier lieu qui consiste à élever le résultat de l'algorithme de Miller à la puissance $\left(p^6 - 1\right)\left(p^2 + 1\right)$. Puis la deuxième partie dite difficile qui consiste à élever le résultat de la première partie à la puissance $\frac{p^4-p^2+1}{r}$. Nous nous intéressons dans notre travail à cette deuxième partie.
L'exposant $\frac{p^4-p^2+1}{r}$ peut être écrit comme étant un polynôme de degré 3 en p. Nous avons donc :

$$\frac{p^4 - p^2 + 1}{r} = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \lambda_3 p^3$$

avec

$$\begin{cases} \lambda_0 &= u^5 - 2u^4 + 2u^2 - u + 3 \\ \lambda_1 &= u^4 - 2u^3 + 2u - 1 \\ \lambda_2 &= u^3 - 2u^2 + u \\ \lambda_3 &= u^2 - 2u + 1. \end{cases} \tag{4.2}$$

La méthode décrite dans la litérature

Dans leur papier, Aranha *et al.* ont repris la méthode de Devegili *et al.* [DSD07] afin de calculer f^d avec $d = \frac{p^4-p^2+1}{r}$. Ils ont calculé la partie difficile de l'exponentiation finale en deux étapes. Tout d'abord, ils ont calculé, dans l'ordre indiqué, les termes suivants :

$$f \longrightarrow f^{-2} \longrightarrow f^u \longrightarrow f^{2u} \longrightarrow f^{u-2} \longrightarrow f^{u^2-2u} \longrightarrow$$
$$f^{u^3-2u^2} \longrightarrow f^{u^4-2u^3} \longrightarrow f^{u^4-2u^3+2u} \longrightarrow f^{u^5-2u^4+2u^2}.$$

Ces calculs ont nécessité deux multiplications dans $\mathbb{F}_{p^{12}}$ pour calculer $f^{u-2} = f^u.f^{-2}$ et $f^{u^4-2u^3+2u} = f^{u^4-2u^3}.f^{2u}$, aussi que deux carrés cyclotomiques pour avoir f^{-2} et f^{2u}. De plus, il a fallu 5 exponentiations par u comme suit :

- f^u,
- $f^{u^2-2u} = \left(f^{(u-2)}\right)^u$,
- $f^{u^3-2u^2} = \left(f^{(u^2-2u)}\right)^u$,
- $f^{u^4-2u^3} = \left(f^{(u^3-2u^2)}\right)^u$,
- $f^{u^5-2u^4+2u^2} = \left(f^{(u^4-2u^3+2u)}\right)^u$.

Une fois ces expressions préparées, Aranha *et al.* ont appliqué le Frobenius et multiplié les termes entre eux afin de trouver le résultat souhaité :

$$f^d = f^{u^5-2u^4+2u^2}\left(f^{u-2}\right)^{-1} f \left(f^{u^4-2u^3+2u} f^{-1}\right)^p \left(f^{u^3-2u^2} f^u\right)^{p^2} \left(f^{u^2-2u} f\right)^{p^3}.$$

Ce calcul supplémentaire nécessite 3 multiplications pour avoir les expressions $\left(f^{u^4-2u^3+2u} f^{-1}\right)$, $\left(f^{u^3-2u^2} f^u\right)$ et $\left(f^{u^2-2u} f\right)$, trois Frobenius appliqués à ces expressions et 5 multiplications.
Au total, Aranha *et al.* ont calculé la partie difficile de l'exponentiation finale en utilisant 5 exponentiations par u, 10 multiplications dans $\mathbb{F}_{p^{12}}$, 2 carrés dans le sous groupe cyclotomique et 3 Frobenius.

Une importante contrainte qui n'était pas bien étudiée dans la littérature est la mémoire utilisée pour l'implémentation du couplage. Dans cette partie nous nous intéressons à ce problème puisque nous visons l'implémentation matérielle de couplage dans un environnement restreint comme dans le cas du Chapitre 3.
Dans leur calcul du couplage Optimal Ate sur les courbes BLS12, Aranha *et al.* n'ont pas précisé le nombre de variables temporaires utilisées pour aboutir au résultat de la partie difficile de l'exponentiation finale et par conséquent dans le calcul de couplage.
Nous avons écrit l'algorithme qui permet de calculer $f^{\frac{p^4-p^2+1}{r}}$ en utilisant le développement de Aranha *et al.* afin de compter les variables temporaires nécessaires. De ce fait, nous pouvons déterminer la mémoire utilisée dans le calcul de l'exponentiation finale par conséquent la mémoire utilisée pour implémenter le couplage Optimal Ate tout entier. Dans ce contexte, nous présentons l'algorithme présenté dans le Tableau 4.1 :

Méthode de Aranha *et al.* détaillée	**Termes calculés**	**Coût**
Input : f, u		
Output : $f^{\frac{p^4-p^2+1}{r}}$		
Var. Temp t_0, t_1, t_2, t_3, t_4, t_5		
$t_0 \leftarrow f^{-2}$	f^{-2}	S_{12}
$t_5 \leftarrow f^u$		$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_1 \leftarrow t_5^2$	f^{2u}	S_{12}
$t_3 \leftarrow t_0 t_5$	f^{u-2}	M_{12}
$t_0 \leftarrow t_3^u$	f^{u^2-2u}	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_2 \leftarrow t_0^u$	$f^{u^3-2u^2}$	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_4 \leftarrow t_2^u$	$f^{u^4-2u^3}$	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_4 \leftarrow t_1 t_4$		M_{12}
$t_1 \leftarrow t_4^u$	$f^{u^5-2u^4+2u^2}$	$(l_u-1)S_{12}+(w_u-1)M_{12}$
$t_3 \leftarrow t_3^{-1}$		
$t_1 \leftarrow t_3 t_1$		M_{12}
$t_1 \leftarrow t_1 f$	f^{λ_0}	M_{12}
$t_3 \leftarrow f^{-1}$		
$t_0 \leftarrow t_0 f$	f^{λ_3}	M_{12}
$t_0 \leftarrow t_0^{p^3}$		F_{12}
$t_4 \leftarrow t_3 t_4$	f^{λ_1}	M_{12}
$t_4 \leftarrow t_4^p$		F_{12}
$t_5 \leftarrow t_2 t_5$	f^{λ_2}	M_{12}
$t_5 \leftarrow t_5^{p^2}$		F_{12}
$t_5 \leftarrow t_5 t_0$		M_{12}
$t_5 \leftarrow t_5 t_4$		M_{12}
$t_5 \leftarrow t_5 t_1$	$f^{\frac{p^4-p^2+1}{r}}$	M_{12}
return t_5		

TABLE 4.1 – La méthode de Aranha *et al.* [AFK$^+$12]

Pour cet algorithme nous remarquons que nous devons utiliser 6 variables temporaires dans $\mathbb{F}_{p^{12}}$ pour calculer la partie difficile de l'exponentiation finale.
Cette partie, gourmande en mémoire, est aussi gourmande en terme de complexité car, elle coûte $(5l_u-3)S_{12}+(5w_u+5)M_{12}+3F_{12}$.

Exemple 4.2. *Si nous considérons le paramètre u présenté dans l'Exemple 4.1, par un simple calcul, nous constatons que pour calculer la partie difficile de l'exponentiation finale il nous faut exactement* $537S_{12}+25M_{12}+3F_{12}$.

Nouveau développement de l'exponentiation finale

Notre but dans ce paragraphe est de présenter une nouvelle manière de calculer la partie difficile de l'exponentiation finale plus efficace que celle présentée par Aranha *et al.* dans [AFK$^+$12]. Pour cette raison nous allons développer autrement les λ_i avec $0 \leqslant i \leqslant 3$ données dans l'expression

$d = \frac{p^4-p^2+1}{r} = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \lambda_3 p^3$. Nous allons reformuler les λ_i comme suit :

$$\begin{cases} \lambda_0 &= \lambda_1 u + 3 \\ \lambda_1 &= \lambda_2 u - \lambda_3 \\ \lambda_2 &= \lambda_3 u \\ \lambda_3 &= u^2 - 2u + 1 \end{cases} \tag{4.3}$$

Remarquons que nous avons écrit les λ_i, $0 \leqslant i \leqslant 3$, l'une en fonction de l'autre. Une fois calculé λ_3, il devient simple de déduire les autres. Par exemple, pour avoir λ_2, on a juste une exponentiation de λ_3 par u à faire etc. Nous n'allons pas suivre ce développement tel qu'il est comme dans le cas de Aranha *et al.* car notre but est de réduire le nombre de variables temporaires utlisées pour le calcul de la partie difficile de l'exponentiation finale. En fait, nous allons apporter quelques modifications sans changer les expressions afin de trouver un algorithme moins gourmand en mémoire et en complexité pour calculer $f^{\frac{p^4-p^2+1}{r}}$.
Dans ce contexte, nous proposons :

Notre nouvelle variante	**Termes calculés**	**Coût**
Input : $f,\ u$		
Output : $f^{\frac{p^4-p^2+1}{r}}$		
Var. Temp : t_0, t_1, t_2, t_3, t_4		
$t_0 \leftarrow f^2$		S_{12}
$t_1 \leftarrow t_0^u$		$(l_u - 1)S_{12} + (w_u - 1)M_{12}$
$t_2 \leftarrow t_1^{u/2}$	f^{u^2}	$(l_u - 2)S_{12} + (w_u - 1)M_{12}$
$t_3 \leftarrow f^{-1}$		
$t_1 \leftarrow t_3 t_1$	f^{2u-1}	M_{12}
$t_1 \leftarrow t_1^{-1}$	f^{-2u+1}	
$t_1 \leftarrow t_1 t_2$	f^{λ_3}	M_{12}
$t_2 \leftarrow t_1^u$	f^{λ_2}	$(l_u - 1)S_{12} + (w_u - 1)M_{12}$
$t_3 \leftarrow (t_2)^u$	$f^{\lambda_2 u}$	$(l_u - 1)S_{12} + (w_u - 1)M_{12}$
$t_1 \leftarrow t_1^{-1}$	$f^{-\lambda_3}$	
$t_3 \leftarrow t_1 t_3$	f^{λ_1}	M_{12}
$t_1 \leftarrow t_1^{-1}$	f^{λ_3}	
$t_1 \leftarrow t_1^{p^3}$	$f^{\lambda_3 p^3}$	F_{12}
$t_2 \leftarrow t_2^{p^2}$	$f^{\lambda_2 p^2}$	F_{12}
$t_1 \leftarrow t_1 t_2$	$f^{\lambda_3 p^3} f^{\lambda_2 p^2}$	M_{12}
$t_2 \leftarrow t_3^u$	$f^{\lambda_1 u}$	$(l_u - 1)S_{12} + (w_u - 1)M_{12}$
$t_2 \leftarrow t_2 t_0$		M_{12}
$t_2 \leftarrow t_2 f$	f^{λ_0}	M_{12}
$t_1 \leftarrow t_1 t_2$	$f^{\lambda_3 p^3} f^{\lambda_2 p^2} f^{\lambda_0}$	M_{12}
$t_2 \leftarrow t_3^p$	$f^{\lambda_1 p}$	F_{12}
$t_1 \leftarrow t_1 t_2$	$f^{\lambda_3 p^3} f^{\lambda_2 p^2} f^{\lambda_1 p} f^{\lambda_0}$	M_{12}
return t_1		

TABLE 4.2 – Notre nouveau développement

Pour cet algorithme, nous avons effectué une exponentiation par $\frac{u}{2}$ au lieu d'une exponentia-

tion par u. Nous avons donc $l_{u/2} = l_u - 1$. Cela signifie que, en appliquant l'algorithme 'square and multiply' pour calculer $t_1^{\frac{u}{2}}$, nous aurons à effectuer un carré de moins que pour calculer t_1^u. L'avantage est de déduire directement f^{u^2}. En effet, d'après le développement de Aranha *et al.* il faut 2 exponentiations par u et deux carrés pour y parvenir :

$$f \longrightarrow f^{-2} \longrightarrow f^u \longrightarrow f^{2u} \longrightarrow (f^u)^u :$$

Dans notre développement, nous n'aurons pas besoin de calculer f^u puisque nous calculons simplement :

$$f \longrightarrow f^{-2} \longrightarrow f^{2u} \longrightarrow (f^{2u})^{u/2} = f^{u^2}$$

qui coûte 1 carré dans $\mathbb{F}_{p^{12}}$, 1 exponentiation par u et 1 exponentiation par $u/2$.

Nous avons ainsi gagné deux carrés dans $\mathbb{F}_{p^{12}}$. De plus, puisque nous déduisons les λ_i, $0 \leqslant i \leqslant 2$, en fonction des autres, nous avons réussi à réduire le nombre des multiplications dans $\mathbb{F}_{p^{12}}$. Dans notre algorithme, nous effectuons 8 multiplications dans $\mathbb{F}_{p^{12}}$ au lieu des 10 qui sont faites dans le développement de Aranha *et al.* .
Ainsi, pour notre nouveau développement de la partie difficile de l'exponentiation finale, nous sommes plus rapides que dans [AFK$^+$12]. Nous avons à effectuer 4 exponentiations par u, une exponentiation par $u/2$, un carré dans le sous-groupe cyclotomique, 8 multiplications et 3 Frobenius dans $\mathbb{F}_{p^{12}}$. En utilisant un paramètre u proposé par Aranha *et al.*, nous obtenons la complexité suivante.

Exemple 4.3. *En choisissant le paramètre u de Aranha* et al. *présenté dans l'Exemple 4.1, la complexité de la partie difficile de l'exponentiation finale est* $535S_{12} + 23M_{12} + 3F_{12}$.

Le grand avantage de notre développement est qu'il ne nécessite pas beaucoup de variables temporaires dans $\mathbb{F}_{p^{12}}$. Nous sommes alors moins gourmands en mémoire que Aranha *et al.* .
Nous n'avons utilisé que 4 variables temporaires au lieu des 6 variables qui doivent être utilisées dans le cas de l'Algorithme 4.1.
Nous présentons dans le tableau ci-dessus une comparaison entre notre méthode de calcul et celle de [AFK$^+$12] en précisant la complexité ainsi que la mémoire utilisée.

Méthode	Complexité			Var.temp
	S_{12}	M_{12}	F_{12}	
Aranha et al.[AFK$^+$12]	$5l_u - 3$	$5w_u + 5$	3	6
Notre développement	$5l_u - 5$	$5w_u + 3$	3	4

TABLE 4.3 – Comparaison entre le développement de [AFK$^+$12] et notre développement

Remarquons que nous n'avons utilisé que 4 variables temporaires au lieu de 6, ce qui représente un gain d'environ 25% dans la mémoire. Ce gain est important si nous pensons à des implémentations dans des environnements restreints. Pour avoir une comparaison complète, nous pouvons considérer le paramètre u donné dans l'Exemple 4.1.

Exemple 4.4. *En utilisant le paramètre u du couplage recommandé dans la littérature pour le calcul de couplage optimal ate sur les courbes BLS12, nous obtenons la comparaison suivante :*

Méthode	Complexité			Var. temp
	S_{12}	M_{12}	F_{12}	
Aranha et *al.*[AFK+12]	537	25	3	6
Notre développement	535	23	3	4

TABLE 4.4 – Exemple

Cet exemple illustre bien que notre méthode de calcul de la partie difficile de l'exponentiation finale du couplage Optimal Ate sur les courbes BLS12 est plus efficace que le meilleur résultat de la littérature que ce soit en terme de complexité ou en terme de consommation mémoire.

Nouveau paramètre u

Nous cherchons à être encore plus efficaces en complexité. Pour cette raison, nous avons cherché un paramètre du couplage u qui possède un poids de Hamming plus faible. En effet, le poids de Hamming du paramètre u influe sur la complexité du couplage Optimal Ate. Ceci peut être observé dans l'algorithme de Miller, puisque le poids de Hamming de u moins 1 représente exactement le nombre d'étapes d'additions dans la boucle de Miller pour calculer $f_{u,Q}(P)$. Nous gagnons également dans l'exponentiation finale puisque chaque exponentiation par u dépend directement de son poids de Hamming.
Nous avons écrit un code sur **Pari/GP** qui permet de trouver tous les paramètres u du couplage qui vérifient un niveau de sécurité élevé et qui sont adaptés aux expressions suivantes :

$$\begin{cases} p &= (u-1)^2(u^4-u^2+1)/3+u \\ r &= u^4-u^2+1 \\ t &= u+1. \end{cases}$$

Parmi ces paramètres, nous choisissons celui qui contient le plus faible poids de Hamming. Notre recherche nous a amené à proposer ce nouveau paramètre.

Proposition 4.2. *La meilleure valeur de u que nous avons pu trouver est :*

$$u = -2^{107} + 2^{84} + 2^{19}.$$

Ce paramètre donne p et r deux nombres premiers de tailles respectives 641 et 428 bits.
Le poids de Hamming de notre nouveau u est $w_u = 3$.

Dans la deuxième partie du calcul du couplage, c'est-à-dire l'exponentiation finale, nous devons effectuer 5 exponentiations par u. Si nous utilisons le paramètre présenté par Aranha *et al.* , il nous faut 3 multiplications dans $\mathbb{F}_{p^{12}}$ pour chaque exponentiation, donc 15 au total. Par contre, si nous considérons notre nouveau paramètre de la Proposition 4.2, nous gagnons une multiplication dans chaque exponentiation par u. Donc au total nous n'effectuons que 10 multiplications dans $\mathbb{F}_{p^{12}}$ au lieu de 15. Ce gain n'est pas négligeable car les multiplications sont les opérations les plus coûteuses intervenant dans l'exponentiation finale. Nous avons maintenant deux paramètres du couplage Optimal Ate, celui donné par Aranha *et al.* et le deuxième paramètre que nous présentons. Nous pouvons donc faire une comparaison de notre nouveau développement avec les

Méthode	Complexité			Nombre de Var. temp.
	S_{12}	M_{12}	F_{12}	utilisées
La méthode de Aranha et *al.*[AFK$^+$12]	537	25	3	6
Notre nouveau développement avec u de Aranha et *al.*[AFK$^+$12]	535	23	3	4
Notre développement avec le nouveau u	535	18	3	4

TABLE 4.5 – Comparaison de la méthode de Aranha avec nos propositions

deux paramètres et le résultat donné dans [AFK$^+$12] dans le tableau suivant :

Comme nous l'avons remarqué précédemment, l'avantage de ce paramètre est qu'il influe aussi sur le coût de l'algorithme de Miller. Nous avons à effectuer une étape d'addition en moins par rapport à Aranha *et al.* .

Pour avoir une comparaison plus explicite, nous considérons les opérations dans $\mathbb{F}_p$. Pour cette raison, nous choisissons un exemple de courbe elliptique sur laquelle nous calculons ce couplage.

Exemple 4.5. *Soit E une courbe elliptique de Barreto, Lynn et Scott (BLS12) définie sur le corps premier $\mathbb{F}_p$ par l'équation*

$$E : y^2 = x^3 + 4$$

Puisque nous voulons calculer le nombre d'operations dans $\mathbb{F}_p$ nécessaires dans le calcul de couplage Optimal Ate sur les BLS12, nous donnons la complexité de chaque opération de $\mathbb{F}_{p^{12}}$, $\mathbb{F}_{p^6}$ et $\mathbb{F}_{p^2}$ dans $\mathbb{F}_p$.
Le corps $\mathbb{F}_{p^{12}}$ peut être construit à partir de la tour d'extension suivante :

$$\begin{aligned} \mathbb{F}_{p^2} &= \mathbb{F}_p[\alpha]/(\alpha^2+1), \\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[\beta]/\big(\beta^3-(\alpha+1)\big)\big) \\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[\gamma]/\big(\gamma^2-\beta\big). \end{aligned}$$

Le coût de chaque opération dans $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^6}$ and $\mathbb{F}_{p^{12}}$ est détaillé dans [GJNB11], [AKL$^+$11b], [AFK$^+$12] ainsi que dans les Chapitre 2 et 3.
Nous évaluons le coût du calcul du couplage Optimal Ate sur la courbe citée dans l'Exemple 4.5 dans $\mathbb{F}_p$. Nous distinguons le coût de l'algorithme de Miller en premier lieu, puis celui de l'exponentiation finale.

— Pour l'algorithme de Miller, avec notre nouveau paramètre u, nous n'avons que 2 étapes d'additions au lieu de 3. Pour toute étape d'addition dans l'algorithme, nous devons effectuer une multiplication creuse 'sparse multiplication' dans $\mathbb{F}_{p^{12}}$. Nous calculons la somme de deux points sur la courbe elliptique et nous évaluons la droite passant par ces deux points au point Q. Cette étape coûte 80 multiplications dans $\mathbb{F}_p$. Donc, nous avons réussi à réduire le nombre de multiplications dans l'algorithme de Miller. L'importance de ce gain se voit beaucoup plus dans la partie implémentation matérielle.

— Dans notre développement, nous avons traité la partie difficile de l'exponentiation finale. Pour évaluer le coût total de l'exponentiation finale, nous rajoutons le calcul de sa partie facile,

autrement dit le coût pour calculer $f^{(p^6-1)(p^2+1)}$. Cette partie ne coûte qu'une inversion, deux Frobenius pour le calcul de f^{p^6} et f^{p^2} et deux multiplications dans $\mathbb{F}_{p^{12}}$ pour calculer $f^{p^6}f^{-1}$ et $\left(f^{p^6-1}\right)^{p^2}\left(f^{p^6-1}\right)$. L'opération la plus coûteuse est bien l'inversion dans $\mathbb{F}_{p^{12}}$ qui nécessite 97 multiplications et une inversion dans $\mathbb{F}_p$. De plus, une multiplication dans $\mathbb{F}_{p^{12}}$ exige 54 multiplications dans le corps fini $\mathbb{F}_p$ donc au total 108 multiplications pour la partie facile de l'exponentiation finale. Nous rajoutons aussi le coût du Frobenius p^2 qui est égal à 10 multiplications dans $\mathbb{F}_p$.
Au total, la partie facile de l'exponentiation finale coûte 215 multiplications et une inversion dans $\mathbb{F}_p$.
Dans la première partie de ce chapitre, nous avons détaillé les méthodes de calcul ainsi que le coût de la partie difficile de l'exponentiation finale qui est égale à $535S_{12} + 18M_{12} + 3F_{12}$ avec l'utilisation de notre nouveau paramètre. Nous explicitons maintenant le coût de cette partie dans $\mathbb{F}_p$.
Le coût total de l'exponentiation finale est une inversion dans $\mathbb{F}_{p^{12}}$, 10 multiplications, 4 Frobenius, un carré, 4 exponentiations par u et une exponentiation par $u/2$. Pour chaque exponentiation par u, nous devons effectuer $107(6S_2)+4(3M_2+3S_2)+3(3M_2)+I_2+2(18M_2) = (57M_2 + 654S_2 + I_2)$ si nous considérons la méthode de Karabina vue à la section 2.5.1.En considérant que $M_2 = 3M$ (méthode de Karatsuba) et $S_2 = 2M$ (méthode complexe), nous avons alors à effectuer $4(57M_2 + 654S_2 + I_2) + (57M_2 + 648S_2 + I_2) + 4(15M) + 10(18M_2) + (23M_2 + 11S_2 + I_2) + 9S_2 = 8116M + 6I$.

Remarque 4.2. *Pour calculer la complexité de l'exponentiation finale, nous avons utilisé la méthode de Karabina pour le calcul du carré dans le sous-groupe cyclotomique de $\mathbb{F}_{p^{12}}$. Nous pouvons aussi évaluer ce carré en utilisant la méthode de Granger et Scott. Nous rappelons que pour la deuxième méthode, le carré cyclotomique dans $\mathbb{F}_{p^{12}}$ nécessite 9 carrés dans $\mathbb{F}_{p^2}$.*

Nous résumons dans le tableau suivant les complexités que nous avons, en distinguant le cas du carré cyclotomique (de Granger et Scott) du cas du carré compressé (de Karabina).

Méthode	Le développement de Aranha *et al.* [AFK+12]	Notre développement avec le nouveau u
Complexité de l'expo finale (carré cyclotomique)	11271 M + I	**10857 M + I**
Complexité de l'expo finale (carré compressé)	8524 M + 6 I	**8116 M + 6 I**
Complexité de l'algorithme de Miller	10865 M	**10785 M**

TABLE 4.6 – Comparaison des coûts de la partie difficile de l'expo.finale selon la méthode utilisée

Comme nous avons présenté la complexité du calcul du couplage Optimal Ate sur la courbe BLS12 dans $\mathbb{F}_p$, nous présentons aussi le nombre de variables utilisées dans $\mathbb{F}_p$. Nous avons déjà indiqué, dans le chapitre précédent, que toute variable temporaire dans le corps $\mathbb{F}_{p^{12}}$ est représentée dans $\mathbb{F}_p$ par 12 variables temporaires. De plus, nous devons rajouter 10 variables temporaires dans $\mathbb{F}_p$. Cet ajout intervient dans l'algorithme permettant de calculer le produit de deux éléments dans $\mathbb{F}_{p^{12}}$ comme nous l'avons expliqué dans le Chapitre 3. Finalement, pour compter le nombre de

variables temporaires utilisées pour le calcul de l'exponentiation finale, nous multiplions le nombre des variables temporaires dans $\mathbb{F}_{p^{12}}$ par 12 auxquels nous rajoutons 10 variables temporaires ainsi que 12 autres pour présenter notre donnée f qui est un élément de $\mathbb{F}_{p^{12}}$.
En considérant notre nouveau développement de la partie difficile de l'exponentiation finale, nous utilisons exactement $4 \times 12 + 10 + 12$, donc 70 variables temporaires dans $\mathbb{F}_p$. Par contre, Aranha *et al.* doivent utiliser $6 \times 12 + 10 + 12$, soit au total 94 variables temporaires dans $\mathbb{F}_p$ pour qu'ils calculent le couplage Optimal Ate sur les courbes BLS12. Notre optimisation en mémoire est importante puisque nous économisons 24 variables temporaires. Ce gain représente 28%, ce qui est un important pourcentage.
Après avoir donné les coûts du calcul de l'algorithme de Miller ainsi que l'exponentiation finale dans $\mathbb{F}_p$, nous pouvons comparer notre résultat avec celui de Aranha *et al.* . Nous présentons dans le tableau suivant notre gain en complexité ainsi qu'en mémoire.

Méthode	Complexité dans $\mathbb{F}_p$ Expo. finale	Gain	Mémoire utilisée dans $\mathbb{F}_p$	Gain
La méthode de Aranha et *al.*[AFK+12]	8524 M +6 I		90	
Notre nouveau développement	**8116 M+ 6 I**	**+5%**	**70**	**+26.6%**

TABLE 4.7 – Comparaison des deux méthodes citées et du gain.

4.2 Le couplage Optimal Ate sur les courbes BLS24

Les courbes BLS12 aussi bien que les courbes elliptiques de Barreto, Lynn et Scott de degré de plongement $k = 24$, sont adaptées pour l'implémentation du couplage Optimal Ate pour un haut niveau de sécurité. Dans cette section, nous reprenons en premier lieu le calcul de l'exponentiation finale du couplage Optimal Ate sur cette nouvelle catégorie de courbes elliptiques. Nous montrons, dans cette partie, que nous sommes plus efficaces dans ce calcul que Aranha *et al.* dans [AFK+12].
En deuxième lieu et d'une manière similaire à la Section 4.1, nous proposons un nouveau paramètre u du couplage Optimal Ate. Avec ce nouveau paramètre, nous sommes plus efficaces dans le calcul du couplage tout entier.
La courbe BLS24 est définie par les paramètres suivants :

$$\begin{cases} p &= (u-1)^2(u^8-u^4+1)/3+u \\ r &= u^8-u^4+1 \\ t &= u+1 \end{cases} \tag{4.4}$$

avec p et r deux nombres premiers.
Le couplage Optimal Ate sur les courbes BLS24 est donné dans la proposition suivante :

Proposition 4.3. *[Ver10] Le couplage Optimal Ate défini sur les courbes BLS24 est l'application bilinéaire et non dégenerée suivante :*

$$\begin{aligned} e_{opt} : G_1 \times G_2 &\rightarrow G_3 \\ (P,Q) &\longmapsto f_{u,Q}(P)^{\frac{p^{24}-1}{r}} \end{aligned}$$

La partie la plus significative dans le calcul du couplage est l'exponentiation finale. Nous rappelons tout d'abord l'expression de l'exponentiation finale pour une courbe de degré de plongement égal à 24 :

$$\frac{p^{24}-1}{r} = \left(p^{12}-1\right)\left(p^4+1\right)\frac{p^8-p^4+1}{r}$$

Nous avons deux parties à calculer. La première consiste en l'élévation du résultat de l'algorithme de Miller à la puissance $\left(p^{12}-1\right)\left(p^4+1\right)$. Elle est dite la partie facile de l'exponentiation finale. Après avoir calculé cette exponentiation, nous calculons la partie difficile de l'exponentiation finale qui consiste en l'élévation du résultat de $f^{\left(p^{12}-1\right)\left(p^4+1\right)}$ à la puissance $\frac{p^8-p^4+1}{r}$.
L'exposant $\frac{p^8-p^4+1}{r}$ de la partie difficile de l'exponentiation finale peut être présenté d'une manière plus simple [Sco05] comme suit :

$$\frac{p^8-p^4+1}{r} = \sum_{i=0}^{\phi(24)-1} \lambda_i p^i = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \cdots + \lambda_7 p^7$$

Par identification, les λ_i, $0 \leqslant i \leqslant 7$ sont données par :

$$\begin{cases} \lambda_0 &= u^9 - 2u^8 + u^7 - u^5 + 2u^4 - u^3 + 3 \\ \lambda_1 &= u^8 - 2u^7 + u^6 - u^4 + 2u^3 - u^2 \\ \lambda_2 &= u^7 - 2u^6 + u^5 - u^3 + 2u^2 - u \\ \lambda_3 &= u^6 - 2u^5 + u^4 - u^2 + 2u - 1 \\ \lambda_4 &= u^5 - 2u^4 + u^3 \\ \lambda_5 &= u^4 - 2u^3 + u^2 \\ \lambda_6 &= u^3 - 2u^2 + u \\ \lambda_7 &= u^2 - 2u + 1. \end{cases} \tag{4.5}$$

Aranha *et al.* dans [AFK+12] ont étudié la complexité du calcul du couplage Optimal Ate sur la courbe BLS24, en particulier cette partie difficile de l'exponentiation finale. Ils ont choisi un paramètre u bien particulier qu'on présente dans l'exemple suivant :

Exemple 4.6. *Le paramètre u choisi par Aranha* et al. *est un entier de taille 49 bits et de poids de Hamming égal à 3 :*

$$u = -2^{48} + 2^{45} + 2^{31} - 2^7$$

Ce paramètre vérifie un niveau de sécurité élevé similaire à celui vérifié par les courbes BLS12. Dans ce cas, p et r sont premiers et de tailles respectives 479 et 384 bits.

La partie difficile de l'exponentiation finale a été calculée par Aranha *et al.* en appliquant directement l'expression suivante :

$$f^{\frac{p^8-p^4+1}{r}} = f^{\lambda_0} f^{\lambda_1 p} f^{\lambda_2 p^2} f^{\lambda_3 p^3} f^{\lambda_4 p^4} f^{\lambda_5 p^5} f^{\lambda_6 p^6} f^{\lambda_7 p^7}.$$

Cette méthode de calcul leur coûte 9 exponentiations par u, 2 carrés cyclotomiques, 12 multiplications dans $\mathbb{F}_{p^{24}}$ et 7 Frobenius $(p, p^2, \ldots p^7)$. Puisque chaque exponentiation par u nécessite $(l_u - 1)$ carrés cyclotomiques et $(w_u - 1)$ multiplications dans $\mathbb{F}_{p^{24}}$, le coût total de la partie difficile de l'exponentiation finale est $(9(l_u-1)+2)\,S_{24}$, $(9(w_u-1)+12)\,M_{24}$ et 7 Frobenius dans $\mathbb{F}_{p^{24}}$.
En considérant le paramètre donné dans l'Exemple 4.6, Aranha *et al.* doivent effectuer 25412 M+10 I.

Remarque 4.3. *Nous remarquons que le paramètre u du couplage Optimal Ate sur les courbes BLS24 est aussi pair. Cela nous permet de faire des exponentiations par $u/2$.*

4.2.1 Notre nouveau développement de l'exponentiation finale

Afin d'améliorer le coût de la parie difficile de l'exponentiation finale, nous avons cherché à ré-écrire les expressions de λ_i, avec $0 \leqslant i \leqslant 7$ d'une manière plus performante. Dans ce contexte, nous avons réussi à trouver des relations simples entre ces expressions qui facilitent le reste du calcul.

$$\begin{cases} \lambda_0 &= \lambda_1 u + 3 \\ \lambda_1 &= \lambda_2 u \\ \lambda_2 &= \lambda_3 u \\ \lambda_3 &= \lambda_4 u - \lambda_7 \\ \lambda_4 &= \lambda_5 u \\ \lambda_5 &= \lambda_6 u \\ \lambda_6 &= \lambda_7 u \\ \lambda_7 &= u^2 - 2u + 1 \end{cases} \tag{4.6}$$

L'idée générale de notre développement est de présenter chaque λ_i, $0 \leqslant i \leqslant 7$ en fonction des suivants. Cela nous permet de calculer simplement la partie difficile de l'exponentiation finale. Par exemple, pour avoir λ_6, il suffit d'élever λ_7 à la puissance u. De même, nous déduisons le reste des expressions des λ_i.
Le calcul de la partie difficile de l'exponentiation finale, ainsi que son coût détaillé, sont donnés dans l'algorithme présenté dans le Tableau 4.8 :

Nouvau développement	**Termes calculés**	**Coût**
Input : $f,\ u$		
Output : $f^{\frac{p^8-p^4+1}{r}}$		
$t_7 \leftarrow f^2$		S_{24}
$t_1 \leftarrow t_7^u$	f^{2u}	$(l_u-1)S_{24}+(w_u-1)M_{24}$
$t_2 \leftarrow t_1^{u/2}$	f^{u^2}	$(l_u-2)S_{24}+(w_u-1)M_{24}$
$t_3 \leftarrow t_1^{-1}$		
$t_2 \leftarrow t_2 t_3$	f^{u^2-2u}	M_{24}
$t_2 \leftarrow t_2 f$	f^{λ_7}	M_{24}
$t_3 \leftarrow t_2^u$	f^{λ_6}	$(l_u-1)S_{24}+(w_u-1)M_{24}$
$t_4 \leftarrow t_3^u$	f^{λ_5}	$(l_u-1)S_{24}+(w_u-1)M_{24}$
$t_3 \leftarrow t_3^{p^6}$	$f^{\lambda_6 p^6}$	F_{24}
$t_5 \leftarrow t_4^{p^5}$	$f^{\lambda_5 p^5}$	F_{24}
$t_3 \leftarrow t_3 t_5$	$f^{\lambda_6 p^6+\lambda_5 p^5}$	M_{24}
$t_5 \leftarrow t_4^u$	f^{λ_4}	$(l_u-1)S_{24}+(w_u-1)M_{24}$
$t_6 \leftarrow t_5^u$	$f^{\lambda_4 u}$	$(l_u-1)S_{24}+(w_u-1)M_{24}$
$t_5 \leftarrow t_5^{p^4}$		F_{24}
$t_3 \leftarrow t_3 t_5$	$f^{\lambda_6 p^6+\lambda_5 p^5+\lambda_4 p^4}$	M_{24}
$t_0 \leftarrow t_2^{-1}$	$f^{-\lambda_7}$	
$t_6 \leftarrow t_6 t_0$	f^{λ_3}	M_{24}
$t_5 \leftarrow t_6^{p^3}$	$f^{\lambda_3 p^3}$	F_{24}
$t_3 \leftarrow t_3 t_5$	$f^{\lambda_6 p^6+\lambda_5 p^5+\lambda_4 p^4+\lambda_3 p^3}$	M_{24}
$t_5 \leftarrow t_6^u$	f^{λ_2}	$(l_u-1)S_{24}+(w_u-1)M_{24}$
$t_0 \leftarrow t_5^{p^2}$	$f^{\lambda_2 p^2}$	F_{24}
$t_3 \leftarrow t_3 t_0$	$f^{\lambda_6 p^6+\lambda_5 p^5+\lambda_4 p^4+\lambda_3 p^3+\lambda_2 p^2}$	M_{24}
$t_6 \leftarrow t_5^u$	f^{λ_1}	$(l_u-1)S_{24}+(w_u-1)M_{24}$
$t_0 \leftarrow t_6^p$	$f^{\lambda_1 p}$	F_{24}
$t_3 \leftarrow t_3 t_0$	$f^{\lambda_6 p^6+\lambda_5 p^5+\lambda_4 p^4+\lambda_3 p^3+\lambda_2 p^2+\lambda_1 p}$	M_{24}
$t_5 \leftarrow t_6^u$	$f^{\lambda_1 u}$	$(l_u-1)S_{24}+(w_u-1)M_{24}$
$t_5 \leftarrow t_5 t_7$	$f^{\lambda_1 u+2}$	M_{24}
$t_5 \leftarrow t_5 f$	f^{λ_0}	M_{24}
$t_3 \leftarrow t_3 t_5$	$f^{\lambda_6 p^6+\lambda_5 p^5+\lambda_4 p^4+\lambda_3 p^3+\lambda_2 p^2+\lambda_1 p+\lambda_0}$	M_{24}
$t_2 \leftarrow t_2^{p^7}$	$f^{\lambda_7 p^7}$	F_{24}
$t_3 \leftarrow t_3 t_2$		M_{24}
return t_3		

TABLE 4.8 – Notre développement de la partie difficile de l'expo. finale pour les BLS24.

Remarquons que dans cet algorithme, nous avons effectué une exponentiation par $u/2$. Cette opération est permise car le paramètre u est pair. Par cette exponentiation par $u/2$ nous avons gagné un carré cyclotomique par rapport à une exponentiation par u.
De ce fait, notre coût de la partie difficile de l'exponentiation finale est $8(l_u-1)S_{24}$, $(l_u-2)\,S_{24}$, S_{24}, $(9(w_u-1)+12)\,M_{24}$ et 7 Frobenius dans $\mathbb{F}_{p^{24}}$.
Nous sommes alors plus efficaces que Aranha *et al.* dans le calcul de l'exponentiation finale du

couplage Optimal Ate puisque nous avons 2 carrés cyclotomiques de moins.

Remarque 4.4. *Pour notre nouveau développement, nous avons utilisé* 8 *variables temporaires dans* $\mathbb{F}_{p^{24}}$ *qui est également le même nombre de variables qui doivent être utilisées pour le développement de Aranha et* al.*.*

4.2.2 Notre nouveau paramètre de la courbe BLS24

Toujours dans le but d'optimiser le calcul du couplage Optimal Ate sur les courbes BLS24, nous avons cherché un nouveau paramètre u qui possède un poids de Hamming plus faible que celui de [AFK+12].

Proposition 4.4. *Notre recherche nous a permis de trouver le paramètre* u *suivant :*

$$u = 2^{48} - 2^{30} + 2^{26}$$

qui donne un p *premier de taille 479-bit et* r *premier de taille 384-bits.*

Ce nouveau paramètre u est un entier de 49 bits comme celui proposé dans [AFK+12]. Par contre, son poids de Hamming est 3 au lieu de 4. Nous aurons donc moins de multiplications dans $\mathbb{F}_{p^{24}}$. Plus explicitement, avec ce nouveau paramètre, nous gagnons une multiplication dans chaque exponentiation par u. Ainsi, au total nous gagnons 9 multiplications dans $\mathbb{F}_{p^{24}}$.

Comme pour les courbes BLS12, avec ce nouveau paramètre u, nous gagnons une étape d'addition dans l'algorithme de Miller.
Pour une comparaison plus explicite, nous prenons l'exemple suivant d'une courbe BLS24 :

Exemple 4.7. *Soit* E *une courbe BLS24 définie sur* $\mathbb{F}_p$ *par*

$$y^2 = x^3 + 1$$

Nous voulons présenter la complexité du couplage Optimal Ate sur la courbe BLS24 en terme de multiplications dans le corps $\mathbb{F}_p$. Nous utilisons la tour d'extension la plus recommandée dans la littérature [AFK+12] :

$$\begin{aligned}
\mathbb{F}_{p^2} &= \mathbb{F}_p[\alpha]/(\alpha^2+1),\\
\mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[\beta]/\big(\beta^3-(\alpha+2)\big)\big),\\
\mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[\gamma]/\big(\gamma^2-\beta\big),\\
\mathbb{F}_{p^{24}} &= \mathbb{F}_{p^{12}}[\theta]/\big(\theta^2-\gamma\big).
\end{aligned}$$

Le coût de chaque opération dans $\mathbb{F}_{p^{24}}$ en termes d'opérations dans $\mathbb{F}_{p^2}$ est détaillé dans le tableau suivant :

Opération	Le coût
Multiplication dans $\mathbb{F}_{p^{24}}$	89 M_2
Carré dans $\mathbb{F}_{p^{24}}$	36 M_2
Carré Compressé dans $\mathbb{F}_{p^{24}}$	12 M_2
Décompression de carré dans $\mathbb{F}_{p^{24}}$	87 M_2+I_2
Inversion dans $\mathbb{F}_{p^{24}}$	83 M_2+11 S_2+I_2

TABLE 4.9 – Les coûts des opérations dans $\mathbb{F}_{p^{24}}$

D'après ce tableau, le coût total de l'exponentiation finale est :

$$\begin{aligned} Expo.finale &= 8\left(48(12M_2)+89M_2+2S_2+2(54M_2)+I_2\right)+8(45M)+14(54M_2) \\ &+ 18M_2+(47(12M_2)+89M_2+2S_2+2(54M_2)+I_2) \\ &= 7802M_2+29S_2+400M+10I. \end{aligned}$$

En termes d'opérations sur $\mathbb{F}_p$, cela représente $23864M+10I$ (en supposant que nous utilisons le carré compressé). Si nous comparons notre résultat à celui de Aranha *et al.* , il s'avère que nous avons gagné 1548 multiplications dans $\mathbb{F}_p$ dans le calcul de la partie difficile de l'exponentiation finale. C'est un gain important puisqu'il est au tour de 8%. Pour l'algorithme de Miller, nous avons réussi à économiser 353 multiplications dans $\mathbb{F}_p$, ce qui nous donne un gain de $2,5\%$.
Jusqu'à maintenant, nous avons représenté le calcul du couplage Optimal Ate sur deux catégories des courbes elliptiques : les courbes de Barreto, Lynn et Scott de degré de plongement $k=12$ et celles de $k=24$. Ces deux courbes sont recommandées dans la pratique pour les implémentations de ce couplage pour les hauts niveaux de sécurité.
Il existe dans la littérature plusieurs autres courbes elliptiques où nous pouvons calculer le couplage Optimal Ate et qui présentent un niveau de sécurité comparable. La différence est bien sûr la complexité. Dans le tableau suivant, nous rappelons ces résultats en mentionnant également les nôtres (présentés dans les Sections 4.1 et 4.2). Nos résultats ne changent pas (au contraire) la conclusion de Aranha, à savoir que les courbes BLS12 sont les plus adaptées à ce niveau de sécurité.

Courbes	Méthode	Complexité de la boucle de Miller	Complexité de l'expo finale
Les courbes BLS12	Aranha et *al.*[AFK+12]	10865	8524M+6I
	Notre travail	**10785**	**8116M+6I**
Les courbes BLS24	Aranha et *al.*[AFK+12]	14927	25412M+10I
	Notre travail	**14574**	**23864M+10I**
Les courbes BN	Aranha et *al.*[AFK+12]	16553M	7218M+4I
Les courbes KSS18	Aranha et *al.*[AFK+12]	13168M	23821M+8I

TABLE 4.10 – Comparaison d'anciens résultats avec nos travaux

Pour faciliter la vérification de nos résultats présentés dans le calcul du couplage Optimal Ate sur les courbes BLS12 ainsi que sur les courbes BLS24, nous les avons implémenté sur Magma. Dans ce cadre, nous présentons un code magma dans [GF].

4.3 Courbes elliptiques et calcul du produit de n couplages

Il existe plusieurs protocoles cryptographiques qui sont basés sur les couplages sur les courbes elliptiques que nous notons par la suite *pairing-based-cryptography*.
Parmi ces protocoles, il en existe plusieurs qui nécessitent le calcul du produit ou du quotient de n couplages, $n \geqslant 2$. Certains exigent le calcul du produit de deux couplages [CCS06], d'autres nécessitent le calcul du produit de trois couplages [BBS04] et même plus que trois couplages [Wat05, ACD$^+$06].
Dans la littérature, il n'existe pas beaucoup de travaux consacrés au calcul du produit ou de quotient de n couplages. Parmi les travaux existants, nous pouvons citer les travaux de Granger et Smart dans [GS06]. Récemment Zhang *et al.* ont étudié ce problème dans [ZL12]. Ils ont montré que les courbes de Kachisa, Schafer et Scott de degré de plongement $k = 16$, que nous notons par la suite KSS16, sont les courbes elliptiques adéquates pour définir le produit ou le quotient de n couplages pour un haut niveau de sécurité.
Nous avons montré dans la première partie de ce chapitre, que pour calculer un seul couplage, il vaut mieux utiliser les courbes BLS12. Dans cette section, nous allons étudier, en premier lieu, le calcul du couplage Optimal Ate sur les courbes KSS16. Nous calculons d'une manière explicite l'exponentiation finale. Nous avons relevé des inexactitudes dans les étapes de calcul présentées par Zhang *et al.* . Par conséquent, nous avons repris leur travail et nous avons corrigé les formules concernées. Nous donnons également un nouvel algorithme de calcul de l'exponentiation finale. Dans un deuxième lieu, nous cherchons un autre multiple de la partie difficile de l'exponentiation finale. En utilisant ce multiple, nous parvenons à calculer le couplage Optimal Ate sur les courbes KSS16 d'une manière plus efficace que dans la littérature.
En troisième lieu, nous traitons le problème du calcul de produit ou de quotient de n couplages. Nous avons montré que la courbe elliptique la plus adéquate pour ce calcul n'est pas toujours KSS16 comme c'est mentionné dans [ZL12]. Nous montrons que pour $n = 2$, il est préférable d'utiliser BLS12. Sinon, dans le cas où $n \geqslant 3$, KSS16 est la solution.
Dans cette partie, nous montrons également que la courbe KSS16, présentée par Zhang *et al.* , ne résiste pas aux attaques par les sous-groupes '*sub-groups attacks*'. Pour cette raison, nous proposons un nouveau paramètre u qui rend la courbe plus résistante contre ce type d'attaque.

4.3.1 Le couplage Optimal Ate sur les courbes KSS16

Kachisa, Schafer et Scott ont proposé dans [KSS07] une famille de courbes elliptiques adéquates pour le calcul de couplage de degré de plongement $k \in \{16, 18, 32, 36, 40\}$. Les courbes elliptiques de degré de plongement $k = 16$ sont notées KSS16 et sont définies comme suit :

Définition 4.2. *Toute courbe de Kachisa, Schafer et Scott de degré de plongement égal à* 16 *est définie sur* $\mathbb{F}_p$ *par l'équation :*

$$y^2 = x^3 + ax$$

et par le paramètre u *tel que :*

$$\begin{cases} t &= 1/35\left(2u^5 + 41u + 35\right) \\ r &= u^8 + 48u^4 + 625 \\ p &= \frac{1}{980}\left(u^{10} + 2u^9 + 5u^8 + 48u^6 + 152u^5 + 240u^4 + 625u^2 + 2398u + 3125\right) \end{cases} \tag{4.7}$$

où t est la trace de Frobenius de la courbe. Le paramètre u est choisi de telle sorte que p et r sont premiers et ont une taille correspondante au niveau de sécurité désigné. Dans notre cas d'étude p et r sont de tailles respectives 481 *et* 377 *bits.*

Proposition 4.5. *[ZL12][Ver10] Le couplage optimal ate pairing sur les courbes KSS16 est l'application bilinéaire et non dégénérée suivante :*

$$\begin{aligned} e_{opt} : G_1 \times G_2 &\to G_3 \\ (P,Q) &\longmapsto \left(\left(f_{u,Q}(P) l_{[u]Q,[p]Q}(P) \right)^{p^3} l_{Q,Q}(P) \right)^{\frac{p^{16}-1}{r}}. \end{aligned}$$

Exemple 4.8. *Le paramètre u proposé par Zhang et* al. *dans [ZL12] est*

$$u = 2^{49} + 2^{26} + 2^{15} - 2^7 - 1.$$

Ce paramètre est de taille 49 bits. Il possède un poids de Hamming faible qui vaut 5*. Avec ce paramètre, r et p sont bien premiers de tailles respectives de* 377 *et* 481 *bits. Ce paramètre u est impair, donc une exponentiation par $(u+1)$ est moins coûteuse que celle par u.*

Pour calculer le couplage Optimal Ate sur les courbes KSS16, comme sur les autres courbes, nous avons deux étapes à considérer : la boucle de Miller puis l'exponentiation finale.
La boucle de Miller, dans ce cas, consiste à calculer la fonction $\left(\left(f_{u,Q}(P) l_{[u]Q,[p]Q}(P) \right)^{p^3} l_{Q,Q}(P) \right)$. Nous calculons au premier lieu la fonction rationnelle $f_{u,Q}(P)$ en utilisant l'Algorithme 3 puisque notre paramètre u est signé. Ensuite, nous élevons le résultat de l'algorithme de Miller $\left(\left(f_{u,Q}(P) l_{[u]Q,[p]Q}(P) \right)^{p^3} l_{Q,Q}(P) \right)$, que nous notons f_1, à la puissance $\frac{p^{16}-1}{r}$: c'est l'exponentiation finale.

L'exponentiation finale et notre correction

Cette exponentiation est décomposée comme suit :

$$\frac{p^{16}-1}{r} = \left(p^8 - 1\right) \left(\frac{p^8+1}{r} \right).$$

Nous devons donc calculer l'exponentiation finale en deux parties. La première (partie facile) consiste à élever le résultat de l'algorithme de Miller à la puissance (p^8-1). Pour cela, il nous faut calculer le Frobenius p^8, une inversion, f_1^{-1} et une multiplication dans $\mathbb{F}_{p^{16}}$.
Soit f le résultat de la partie facile de l'exponentiation finale. Nous devons maintenant élever f à la puissance $\frac{p^8+1}{r}$ (partie difficile).
Cet exposant peut être présenté de la manière suivante :

$$\frac{p^8+1}{r} = \sum_{i=0}^{\phi(16)-1} c_i p^i = c_0 + c_1 p + c_2 p^2 + \cdots + c_7 p^7$$

avec

$$\begin{cases} c_0 &= -11u^9 - 22u^8 - 55u^7 - 278u^5 - 1172u^4 - 1390u^3 + 1372 \\ c_1 &= 15u^8 + 30u^7 + 75u^6 + 220u^4 + 1280u^3 + 1100u^2 \\ c_2 &= 25u^7 + 50u^6 + 125u^5 + 950u^3 + 3300u^2 + 4750u \\ c_3 &= -125u^6 - 250u^5 - 625u^4 - 3000u^2 - 13000u - 15000 \\ c_4 &= -2u^9 - 4u^8 - 10u^7 + 29u^5 - 54u^4 + 154u^3 + 4704 \\ c_5 &= -20u^8 - 40u^7 - 100u^6 - 585u^4 - 2290u^3 - 2925u^2 \\ c_6 &= 50u^7 + 100u^6 + 250u^5 + 1025u^3 + 4850u^2 + 5125u \\ c_7 &= 875u^2 + 1750u + 4375. \end{cases} \quad (4.8)$$

Remarque 4.5. *En vérifiant ces formules, nous remarquons que nous calculons un multiple de la partie difficile de l'exponentiation finale. En fait, nous considérons plutôt* $857500 \times \frac{p^8+1}{r}$. *Ce choix est important dans le calcul du couplage vu qu'avec ce multiple, tous les coefficients des* c_i, $0 \leqslant i \leqslant 7$ *sont des entiers. De plus, il n'est pas restrictif de considérer une puissance d'un couplage comme nous l'avons vu dans le Chapitre 3.*

Pour simplifier ces expressions, Zhang *et al.* ont récrit les c_i d'une manière plus simple afin de calculer la partie difficile de l'exponentiation finale d'une manière plus efficace. Ils ont proposé tout d'abord :

$$A = u^3.B + 56 \text{ et } B = (u+1)^2 + 4,$$

et ont donné la présentation suivante :

$$\begin{cases} c_0 = -11(u^4A + 27u^3B + 28) + 19A; & c_4 = -(2u^4A + 55u^3B) + 84A \\ c_1 = 5(3u^3A + 44u^2B) = 5c_1'; & c_5 = -5(4u^3A + 117u^2B) = -5c_5' \\ c_2 = 25(u^2A + 38uB) = 25c_2'; & c_6 = 25(2u^2A + 41uB) = 25c_6' \\ c_3 = -125(uA + 24B) = -125c_3'; & c_7 = 125.7B = 125c_7'. \end{cases}$$

Cette présentation facilite bien le calcul de l'exponentiation finale du couplage Optimal Ate sur les courbes KSS16. En vérifiant ces formules, nous avons trouvé des problèmes dans quelques expressions.
Commençant par le premier terme c_0, en effet Zhang *et al.* affirment que

$$\begin{cases} c_0 &= -11u^9 - 22u^8 - 55u^7 - 278u^5 - 1172u^4 - 1390u^3 + 1372 \\ &= -11(u^4A + 27u^3B + 28) + 19A. \end{cases}$$

Par contre, par un calcul simple, nous avons constaté que plutôt, nous devons avoir

$$\begin{cases} c_0 &= -11u^9 - 22u^8 - 55u^7 - 278u^5 - 1172u^4 - 1390u^3 + 1372 \\ &= -11(u^4A + 27u^3B + 28) + 19A + \mathbf{616} \end{cases}$$

Cela implique que, dans leur calcul, Zhang *et al.* n'ont pas calculé l'expression f^{616} qui, dont le coût n'est pas négligeable. Heureusement, ce terme manquant n'influe pas dans le reste de calcul. Il suffit juste de multiplier le résultat final par f^{616}. En utilisant l'algorithme *square-and-multiply*, le calcul supplémentaire de f^{616} exige 8 carrés cyclotomiques et 3 multiplications dans $\mathbb{F}_{p^{16}}$. Mais, pour économiser le coût, et puisque dans l'algorithme original du calcul de l'exponentiation finale de Zhang *et al.* il existe des pré-calculs nécessaires pour le calcul de f^{616}, nous pouvons rajouter à leur algorithme les étapes suivantes :

$$\begin{cases} A0 &\leftarrow T3^8 \\ A1 &\leftarrow A0 \cdot T3 \\ A2 &\leftarrow A1 \cdot T2 \\ A3 &\leftarrow T1^2 \\ A2 &\leftarrow A3 \cdot A2 \end{cases}$$

Remarque 4.6. *Les variables temporaires* T_i *avec* $1 \leqslant i \leqslant 3$ *sont donnés dans l'algorithme original de Zhang et Lin [ZL12] et dans la Table 4.11.*

Dans la variable temporaire $A2$, nous obtiendrons le terme manquant f^{616} que nous devons multiplier à la fin de l'algorithme de Zhang *et al.* par leur résultat. Cette correction nécessite donc

finalement 4 multiplications et 4 carrés cyclotomique dans $\mathbb{F}_{p^{16}}$.

En faisant cette modification à leur algorithme, le résultat final n'est toujours pas correct. Nous étions dans l'obligation de vérifier toutes les étapes de calcul dans leur algorithme. Nous avons trouvé qu'il y avait un soucis dans les expressions de c'_5, c_0, c_4 et c'_7.
En effet, dans l'expression de c'_0, l'algorithme de [ZL12] retourne $-11(u^4A + 55u^3B + 28) + 35A$ au lieu de retourner $-11(u^4A + 55u^3B + 28) + 19A$. De même, l'expression de c'_4 calculée dans l'algorithme est $-(2u^4A + 55u^3B) + 148A$ et non pas celle mentionnée dans le développement des c_i, $-(2u^4A + 55u^3B) + 84A$.
Dans l'algorithme de Zhang *et al.* , l'expression de c'_5 est déduite à partir de la multiplication du terme stocké dans la variable temporaire T_{11} par celui stocké dans F_{25}, alors qu'il faut plutôt multiplier le terme de T_{11} par celui de F_{14}. Enfin pour trouver l'expression de c'_7 il faut effectuer l'opération $\overline{F_5}.T_4$ au lieu de $\overline{F_5}.T_6$.
Ces corrections nécessitent 4 multiplications et 3 carrés cyclotomique supplémentaires dans $\mathbb{F}_{p^{16}}$.
Nous détaillons nos modifications ainsi que le coût ajouté dans l'algorithme présenté dans le Tableau 4.11.

Opérations	**Termes calculés**	**Coût**
$E1 = f^{p^8} E2 = E1 \cdot f^{-1}$	$M = f^{p^8-1}$	M_{16}
$T1 = E2^4; T2 = T1^8; T3 = T2^2$		$6S_{16}$
$A0 = T3^8; A1 = A0 \cdot T3; A2 = A3 \cdot A2$		$2M_{16} + 3S_{16}$
$A2 = A3 \cdot A2; F1 = T2 \cdot T1^{-1}; F2 = F1^2$		$2M_{16} + 1S_{16}$
$F3 = E2^{u+1}; F4 = F3^{u+1}$		$2E_{u+1}$
$F5 = F4 \cdot T1; T4 = F5^8$	$F5 = M^B$	$1M_{16} + 3S_{16}$
$F6 = F5^u; F7 = F5^{-1} \cdot T4; F8 = T4^3; T5 = F6^8$	$F7 = M^{c_7'}$	$E_u + 2M_{16} + 4S_{16}$
$F9 = F6^u; F10 = T5 \cdot F6^{-1}; F11 = F10^2$		$1E_u + 1M_{16} + 1S_{16}$
$T6 = F9^8; F12 = F9^u; F13 = T6 \cdot F9^{-1}$		$1E_u + 1M_{16} + 3S_{16}$
$F14 = F13^2; F15 = F12 \cdot F2$	$F15 = M^A$	$1S_{16} + 1M_{16}$
$T7 = F15^2; T8 = T7^4$		$3S_{16}$
$S1 = T8^2; S2 = T7^2$		$2S_{16}$
$S3 = S2 \cdot S1; S4 = S3 \cdot F15^{-1}$		$2M_{16}$
$T9 = S1^4; S5 = S3 \cdot T9$		$1M_{16} + 2S_{16}$
$S6 = F14^2; F16 = F15^u; F22 = F16 \cdot F8$	$F22 = M^{c_3'}$	$1E_u + 1S_{16} + 1M_{16}$
$F23 = F22^u; F24 = F23 \cdot F11$	$F24 = M^{c_2'}$	$1E_u + 1M_{16}$
$T10 = F23^2; F25 = F23^u$		$1E_u + 1S_{16}$
$F26 = T10 \cdot F10^{-1}; T11 = F25^4$	$F26 = M^{c_6'}$	$1M_{16} + 2S_{16}$
$F27 = F25^u; F28 = T11 \cdot F25^{-1}$		$1E_u + 1M_{16}$
$F29 = F13 \cdot F14; F30 = T11 \cdot F29$	$F30 = M^{c_5'}$	$2M_{16}$
$F31 = F28 \cdot S6^{-1}; F32 = F12^2$		$1M_{16} + 1S_{16}$
$F33 = F32 \cdot F12; F34 = F27 \cdot F33$		$2M_{16}$
$F35 = F34^2; F36 = F35 \cdot F12$		$1M_{16} + 1S_{16}$
$F37 = F36^{-1} \cdot S5; F38 = F34 \cdot F1$	$F37 = M^{c_4'}$	$2M_{16}$
$F39 = F38^2; F40 = F39^2$		$2S_{16}$
$F41 = F40^2; F42 = F39 \cdot F38$		$1M_{16} + 1S_{16}$
$F43 = F41 \cdot F42; F44 = F43^{-1} \cdot S4$		$2M_{16}$
$H1 = F7^{p^7}; H2 = F22^{p^3}$		$2(14M)$
$H3 = F24^{p^2}; H4 = F26^{p^6}$		$2(12M)$
$H5 = F30^{p^5}; H6 = F31^p$		$2(14M)$
$H7 = F37^{p^4}; H8 = H1 \cdot H2^{-1}$		$1M_{16} + 1(8M)$
$H9 = H8^2; H10 = H9^2$		$2S_{16}$
$H11 = H10 \cdot H8; H12 = H11 \cdot H3$		$2M_{16}$
$H13 = H12 \cdot H4; H14 = H13^2$		$1M_{16} + 1S_{16}$
$H15 = H14^2; H16 = H15 \cdot H13$		$1M_{16} + 1S_{16}$
$H17 = H16 \cdot H6; H18 = H17 \cdot H5^{-1}$		$2M_{16}$
$H19 = H18^2; H20 = H19^2$		$2S_{16}$
$H21 = H20 \cdot H18; H22 = H21 \cdot H7$		$2M_{16}$
$H23 = H22 \cdot F44$	$H23 = M^{d'}$	$1M_{16}$

TABLE 4.11 – Version corrigée du calcul de l'exponentiation finale de [ZL12]

Le coût rajouté après nos corrections est bien de 8 multiplications et 7 carrés cyclotomiques dans $\mathbb{F}_{p^{16}}$. .
Au total, pour calculer la partie difficile de l'exponentiation finale nous devons faire 7 exponentiations par u, 2 exponentiations par $(u+1)$, 43 carrés cyclotomiques dans $G_{\phi_2(p^8)}$, 37 multiplications dans $\mathbb{F}_{p^{16}}$, 1 cube dans le sous groupe cyclotomique de $\mathbb{F}_{p^{16}}$ et les Frobenius p, p^2, p^3, p^4, p^5, p^6, p^7.

Pour avoir une comparaison complète du coût de l'exponentiation finale présentée par Zhang *et al.* avant et après notre correction, nous considérons la courbe elliptique KSS16 suivante :

Exemple 4.9. *Soit E une courbe elliptique de Kachisa, Schafer et Scott (KSS16) définie sur le corps premier $\mathbb{F}_p$ par l'équation*

$$E : y^2 = x^3 - 3x$$

Nous considérons le tour d'extension recommandé dans la littérature construite comme suit :

$$\begin{aligned} \mathbb{F}_{p^4} &= \mathbb{F}_{p^2}[u]/\big(u^4+3\big) \\ \mathbb{F}_{p^8} &= \mathbb{F}_{p^4}[v]/\big(v^2-u\big), \\ \mathbb{F}_{p^{16}} &= \mathbb{F}_{p^8}[z]/\big(z^2-v\big). \end{aligned}$$

Dans leur papier, Zhang *et al.* ont donné explicitement le coût de chaque opération de $\mathbb{F}_{p^{16}}$ en terme d'arithmétique dans $\mathbb{F}_p$ en utilisant les résultats donnés dans [ZL12]. En utilisant également le Chapitre 2 nous obtenons le tableau suivant

Opération	Coût dans $\mathbb{F}_p$
Multiplication dans $\mathbb{F}_{p^4}$	$9\,M$
Carré dans $\mathbb{F}_{p^4}$	$6\,M$
Multiplication creuse dans $\mathbb{F}_{p^{16}}$	$7\,M_4 = 63\,M$
Multiplication dans $\mathbb{F}_{p^{16}}$	$9\,M_4 = 91\,M$
Carré dans $\mathbb{F}_{p^{16}}$	$9\,S_4 = 54\,M$
Carré Cyclotomique dans $G_{\phi_2(p^8)}$	$4\,M_4 = 32\,M$
Cube Cyclotomique dans $G_{\phi_2(p^8)}$	$8\,M_4 = 72\,M$
Frobenius p^8, Inversion dans $G_{\phi_2(p^8)}$	Conjugaison
Inversion dans $\mathbb{F}_{p^{16}}$	$I + 132\,M + 2\,S$
Formules de doublement	$8\,M + 2\,M_4 + 8\,S_4 = 74\,M$
Formules d'addition	$8\,M + 9\,M_4 + 5\,S_4 = 119\,M$
Frobenius p, p^3, p^5 et p^7	$15\,M$
Frobenius p^2, p^6	$15\,M$
Frobenius p^4	$8\,M$

TABLE 4.12 – Le Coût de chaque opération de $\mathbb{F}_{p^{16}}$ dans $\mathbb{F}_p$

Grâce à ce Tableau 4.12, nous pouvons comparer précisément (en terme d'arithmétique dans $\mathbb{F}_p$) l'algorithme de Zhang et *al.* et notre correction dans le tableau 4.13.

La Méthode	Complexité de la boucle de Miller	Complexité de l'exponentiation finale
La méthode de [ZL12]	10208 M	22330M+I
Notre Correction	10208 M	23662M+I

TABLE 4.13 – Exemple du coût de l'exponentiation finale avant et après notre correction

D'après le Tableau 4.13, nous remarquons bien que la méthode du calcul de la partie difficile de l'exponentiation finale présentée par Zhang et *al.* nécessite 33662 multiplications dans $\mathbb{F}_p$. Il leur manque donc 1332 multiplications dans $\mathbb{F}_p$.

Notre nouveau multiple de l'exponentiation finale

Dans le but de calculer l'exponentiation finale, ou plutôt la partie difficile de l'exponentiation finale, d'une manière plus efficace, nous avons cherché un nouveau multiple de l'exposant final. Cela permet de calculer plus efficacement l'exponentiation finale qu'en utilisant le vecteur de Zhang *et al.* après notre correction.
En nous basant sur le fait qu'une exponentiation d'un couplage est un couplage, nous avons repris la méthode de Fuentes *et al.* [CKH11] afin de déterminer les multiples de $\frac{p^8+1}{r}$. Comme nous l'avons indiqué dans le Chapitre 3, l'idée est d'appliquer l'algorithme LLL [LLL82] à la matrice qui présente la partie difficile de l'exponentiation finale.
Le meilleur vecteur que nous avons trouvé et que nous notons par la suite $d'(u)$ est :

$$d'(u) = m_0 + m_1 p + m_2 p^2 + m_3 p^3 + m_4 p^4 + m_5 p^5 + m_6 p^6 + m_7 p^7 = s(u) d_1$$

avec

$$\begin{cases} s(u) &= u^3/125 \\ m_0 &= 2u^8 + 4u^7 + 10u^6 + 55u^4 + 222u^3 + 275u^2 \\ m_1 &= -4u^7 - 8u^6 - 20u^5 - 75u^3 - 374u^2 - 375u \\ m_2 &= -2u^6 - 4u^5 - 10u^4 - 125u^2 - 362u - 625 \\ m_3 &= -u^9 - 2u^8 - 5u^7 - 24u^5 - 104u^4 - 120u^3 + 196 \\ m_4 &= u^8 + 2u^7 + 5u^6 + 10u^4 + 76u^3 + 50u^2 \\ m_5 &= 3u^7 + 6u^6 + 15u^5 + 100u^3 + 368u^2 + 500u \\ m_6 &= -11u^6 - 22u^5 - 55u^4 - 250u^2 - 1116u - 1250 \\ m_7 &= 7u^5 + 14u^4 + 35u^3 + 392. \end{cases} \quad (4.9)$$

Autrement dit, nous avons : $d'(u) = s(u)d(u)$. Puisque notre but dans cette section est de calculer $f^{d'(u)}$ d'une manière plus efficace, nous devons alors écrire les m_i, avec $0 \leqslant i \leqslant 7$, simplement. Avec le même principe que la simplification de Zhang et *al.*, nous proposons :

$$\begin{cases} A = u^3 B + 56 \\ B = (u+1)^2 + 4. \end{cases}$$

Ainsi, les expressions des m_i s'écrivent simplement en fonction de A et B. Nous avons :

$$\begin{cases} m_0 = 2u^3 A + 55u^2 B; & m_4 = u^3 A + 10u^2 B \\ m_1 = -4u^2 A - 75uB; & m_5 = 3u^2 A + 100uB \\ m_2 = -2uA - 125B; & m_6 = -11uA - 250B \\ m_3 = -u^4 A - 24u^3 B + 196; & m_7 = 7A. \end{cases}$$

Nous détaillons les étapes de calcul qui nous permettent de calculer la partie difficile de l'exponentiation finale dans l'algorithme présenté dans le Tableau 4.14 où nous donnons également le coût de chaque opération.

Opérations	Termes calculés	Coût
$E1 = f^{p^8} E2 = E1 \cdot f^{-1}$	$M = f^{p^8-1}$	
$T0 = M^2; T1 = T0^2$	$M^2; M^4$	$2S_{16}$
$T2 = M^{u+1}; T3 = T2^{u+1}$	$M^{u+1}; M^{(u+1)^2}$	$2E_u$
$T4 = T3 \cdot T1$	$M^{(u+1)^2+4} = M^B$	$1M_{16}$
$T5 = T4^u; T6 = T4^5$	$M^{uB}; M^{5B}$	$1E_u + 1M_{16} + 2S_{16}$
$T7 = T1^8; T8 = T7^2$	$M^{32}; M^{64}$	$4S_{16}$
$T9 = T7 \cdot T1^{-1}; T10 = T9^2$	$M^{28}; M^{56}$	$1M_{16} + 1S_{16}$
$T11 = T5^u; T12 = T11^u$	$M^{u^2B}; M^{u^3B}$	$2E_u$
$T01 = T12 \cdot T10$	$M^{u^3B+56} = M^A$	$1M_{16}$
$T14 = T01^u; T13 = T14^{-2}$	$M^{uA}; M^{-2uA}$	$1E_u + 1S_{16}$
$T00 = T6^5; T15 = T00^5$	$M^{25B}; M^{125B}$	$2M_{16} + 4S_{16}$
$T0 = T13 \cdot T15^{-1}$	$M^{-2uA-125B} = M^{m_2}$	$1M_{16}$
$T16 = T0^2; T17 = T13^4$	$M^{2m_2}; M^{-8uA}$	$3S_{16}$
$T18 = T17 \cdot T14$	M^{-7uA}	$1M_{16}$
$T2 = T16 \cdot T18$	$M^{2m_2-7uA} = M^{m_6}$	$1M_{16}$
$T19 = T14^u; T20 = T19^u$	$M^{u^2A}; M^{u^3A}$	$2E_u$
$T21 = T20^u; T22 = T19^2$	$M^{u^4}; M^{2u^2A}$	$1E_u + 1S_{16}$
$T23 = T5^5; T24 = T23^5$	$M^{5uB}; M^{25uB}$	$2M_{16} + 4S_{16}$
$T25 = T24^3; T26 = T24 \cdot T25$	$M^{75uB}; M^{100uB}$	$1C_{16} + 1M_{16}$
$T27 = T22^2$	M^{4u^2A}	$1S_{16}$
$T37 = (T27 \cdot T25)^{-1}$	$M^{-4u^2A-75uB} = M^{m_1}$	$1M_{16}$
$T28 = T27 \cdot T19^{-1}$	M^{3u^2A}	$1M_{16}$
$T3 = T28 \cdot T26$	$M^{3u^2A+100xB} = M^{m_5}$	$1M_{16}$
$T29 = T11^5; T30 = T29^2$	$M^{5u^2B}; M^{10u^2B}$	$1M_{16} + 3S_{16}$
$T4 = T20 \cdot T30$	$M^{u^3A+10u^2B} = M^{m_4}$	$1M_{16}$
$S0 = T20^2; S1 = T30^5$	$M^{2u^3A}; M^{50u^2B}$	$1M_{16} + 3S_{16}$
$S2 = S1 \cdot T29; S3 = S0 \cdot S2$	$M^{55u^2B}; M^{2u^3A-55u^2B} = M^{m_0}$	$2M_{16}$
$T31 = T12^{24}$	M^{24u^3B}	$1C_{16} + 3S_{16}$
$T5 = T21^{-1} \cdot T31^{-1}$	$M^{-u^4A-24u^3B}$	$1M_{16}$
$T6 = T8^3 \cdot T1$	M^{196}	$1M_{16} + 1C_{16}$
$T7 = T5 \cdot T6$	$M^{-u^4A-24u^3B+196} = M^{m_3}$	$1M_{16}$
$T8 = T1^7$	$M^{7A} = M^{m_7}$	$2M_{16} + 2S_{16}$
$T32 = T37^p \cdot T7^{p^3} \cdot T3^{p^5} \cdot T8^{p^7}$	$M^{m_1 p + m_3 p^3 + cm_5 p^5 + cm_7 p^7}$	$3M_{16} + 4(15M)$
$T33 = T0^{p^2} \cdot T2^{p^6}$	$M^{m_2 p^2 + m_6 p^6}$	$1M_{16} + 2(12M)$
$T = S3 \cdot T32 \cdot T33 \cdot T4^{p^4}$	$M^{\frac{p^8+1}{r}}$	$3M_{16} + 1(8M)$

TABLE 4.14 – Notre nouveau multiple de l'exponentiation finale.

Le coût de notre méthode de calcul est détaillé dans le Tableau 4.14 et comparé avec celle de Zhang et *al.* après les corrections que nous avons apportées.

La Méthode	**Algorithme présenté dans le tableau**	**Complexité**					
		E_u	E_{u+1}	S_{16}	M_{16}	F_{16}	C_{16}
Le développement de Zhang *et al.* après notre correction	4.11	7	2	43	37	7	1
Notre développement avec le nouveau multiple	4.14	7	2	**34**	**32**	**7**	**3**

TABLE 4.15 – Calcul de l'exponentiation finale avec deux vecteurs différents.

D'après ce tableau, nous constatons que notre gain est de 10 carrés cyclotomiques et 5 multiplications dans $\mathbb{F}_{p^{16}}$.
En utilisant l'Exemple 4.9 de la courbe KSS16, le paramètre u proposé par Zhang et *al.* et présenté dans l'Exemple 4.3.1 ainsi que les coûts présentés dans le Tableau 4.12, nous obtenons la comparaison explicite suivante de la complexité du calcul de la partie difficile de l'exponentiation finale.

Le résultat	**Algorithme présenté dans le tableau**	**Complexité de la partie difficile de l'exponentiation finale**
Notre résultat corrigé de [ZL12]	4.11	23537M
Notre nouveau algorithme	4.14	**22673M**

TABLE 4.16 – Le Coût total en utilisant u de [ZL12]

D'après le Tableau 4.16, il est clair que nous sommes plus rapides que Zhang et *al.* dans le calcul de la partie difficile de l'exponentiation finale. Par conséquent nous sommes plus rapide dans le calcul du couplage Optimal Ate sur les courbes KSS16.
Notre résultat est intéressant vu que le couplage Optimal Ate sur les courbes KSS16 pour un niveau de sécurité élevé est utilisé dans des protocoles cryptographiques basés sur les couplages. En pratique un gain de 864 multiplications dans $\mathbb{F}_p$ est important.
Pour la vérification de nos expressions des m_i, avec $0 \leqslant i \leqslant 7$, ainsi que nos deux algorithmes présentés dans cette section, nous avons fait une implémentation sur magma que nous présentons dans [FG].

4.4 Calcul du produit de n couplages

Plusieurs protocoles cryptographiques, tel que le schéma BBG HIBE [BBG05], le schéma de la signature courte (short group signature) BBS [BBS04], le schéma ABE de Waters [Wat05], le *non interactive proofs system* proposé par Groth and Sahai dans [GS08] et d'autres travaux présentés dans [CCS06, ACD+06], nécessitent le calcul du produit ou du quotient de deux couplages ou plus. Il n'existe pas beaucoup d'études sur le calcul du produit ou/et quotient de n couplages dans la littérature. Parmi ces rares travaux, Scott dans [Sco05] et Granger dans [GS06] ont étudié le problème du calcul de n couplages. Soit

$$\begin{aligned} e : G_1 \times G_2 &\rightarrow G_3 \\ (P_i, Q_i) &\longmapsto e(P_i, Q_i), \end{aligned}$$

une application bilinéaire non dégénérée de $G_1 \times G_2$, G_1 et G_2 deux groupes additifs, vers le groupe multiplicatif G_3. L'évaluation du produit de n couplages est de la forme :

$$e_n = \prod_{i=1}^{n} e(P_i, Q_i).$$

Dans cette section, nous allons présenter en premier lieu le calcul détaillé du produit (le quotient se calcule de la même manière) de n couplages Optimal Ate sur n'importe quelle courbe elliptique. Ensuite, nous allons chercher la courbe elliptique la plus adéquate pour ce calcul pour un haut niveau de sécurité. Pour ce niveau de sécurité, comme il est prouvé par Aranha *et al.* dans [AFK+12] et comme nous l'avons montré dans la première section de ce chapitre et dans [GF16b], la courbe BLS12 est la meilleure solution pour le calcul d'un seul couplage Optimal Ate. Mais, nous ne pouvons pas affirmer que cette catégorie de courbes est également adéquate pour le calcul de n couplages. Dans ce contexte, Zhang *et al.* ont affirmé dans [ZL12] que le calcul du produit ou/et du quotient de n couplages est plutôt plus efficace sur les courbes KSS16.
Nous allons rappeler, tout d'abord, dans le tableau, suivant l'expression du couplage Optimal Ate sur différentes catégories des courbes elliptiques recommandées pour le niveau de sécurité que nous étudions dans ce chapitre :

Catégorie des Courbes	le couplage Optimal Ate : $(P, Q) \rightarrow$
KSS16 [ZL12]	$\Big((f_{u,Q}(P) l_{[u]Q,[p]Q}(P))^{p^3} l_{Q,Q}(P)\Big)^{\frac{p^{16}-1}{r}}$
KSS18 [AFK+12]	$\Big(f_{u,Q}(P) f_{3,Q}^{p} l_{[u]Q,[3p]Q}(P)\Big)^{\frac{p^{18}-1}{r}}$
BN [AFK+12]	$\big((f_{6u+2,Q}(P) l_{[6u+2]Q,[p]Q}(P) l_{[6u+2]Q,[-p^2]Q}(P))\big)^{\frac{p^{12}-1}{r}}$
BLS12 [AFK+12]	$(f_{u,Q}(P))^{\frac{p^{12}-1}{r}}$
BLS24 [AFK+12]	$(f_{u,Q}(P))^{\frac{p^{24}-1}{r}}$

TABLE 4.17 – Le Couplage Optimal Ate sur les courbes elliptiques.

Pour le calcul du couplage Optimal Ate, nous avons deux étapes de calcul : La boucle de Miller en premier lieu puis l'exponentiation finale. Il est clair que le calcul du produit de n couplages est équivalent au calcul du produit de n boucles de Miller (pareil pour le quotient). Ensuite, nous avons une seule exponentiation finale à effectuer.
Rappelons que dans les étapes de doublement de l'algorithme de Miller, nous effectuons l'opération suivante :

$$f \leftarrow f^2 l(P) \tag{4.10}$$

où l est la tangente à la courbe E au point T (multiple de Q) Cependant, pour calculer le produit de l'Équation (4.10), chaque fonction de doublement deviendra :

$$f \leftarrow f^2 \prod_{i=1}^{n} l_i(Q_i). \tag{4.11}$$

Cela signifie que, dans l'étape de doublement, nous devons calculer le carré une seule fois. Cela est trivial tant que notre initialisation de chaque fonction rationelle f_i est égale à 1, avec $1 \leqslant i \leqslant n$.

Exemple 4.10. *Pour faciliter le calcul, nous supposons que nous calculons dans cet exemple le produit de* 3 *couplages Optimal Ate sur les courbes BLS12. Ce choix est dû au fait que l'expression*

du couplage Optimal Ate est plus simple sur les courbes BLS12.
Nous présentons donc dans l'algorithme suivant la manière de calculer le produit de 3 *couplages :*

Algorithm 4 Produit de 3 Couplages

Require: $u = (u_n, \ldots, u_0)$,P,Q
Ensure: $(f_{u,Q_1}(P))^{\frac{p^k-1}{r}} \cdot (f_{u,Q_2}(P))^{\frac{p^k-1}{r}} \cdot (f_{u,Q_3}(P))^{\frac{p^k-1}{r}}$

```
f ← 1
R1 ← Q1
R2 ← Q2
R3 ← Q3
for i = n − 1 down to 0 do do
    f ← f^2 · ℓ_{R1,R1}(P) ℓ_{R2,R2}(P) ℓ_{R3,R3}(P)
    R1 ← 2R1
    R2 ← 2R2
    R3 ← 2R3
    if u_i = 1 then
        f ← f ℓ_{R1,Q1}(P) ℓ_{R2,Q2}(P) ℓ_{R3,Q3}(P)
        R1 ← R1 + Q1
        R2 ← R2 + Q2
        R3 ← R3 + Q3
    end if
end for
return f = f^{(p^k−1)/r}
```

Ainsi, le calcul de n couplages nécessite $\log_2(s) - 1$ carrés dans $\mathbb{F}_{p^k}$, (nous désignons par s le paramètre du couplage Optimal Ate puisqu'il est différent d'une courbe à une autre), puis le calcul de n fois le reste des étapes dans la boucle de Miller et enfin une exponentiation finale.
Donc pour évaluer le coût du calcul du produit de n couplages Optimal Ate, nous devons calculer tout d'abord :

- **Coût 1** : Les carrés de la boucle de Miller (les carrés de l'Équation 4.11) .
- **Coût 2** : Le reste des opérations de la boucle de Miller (les évaluations des lignes, les mises à jour des points ainsi que les multiplications par les lignes).
- **Coût 3** : L'exponentiation finale.

Ainsi, nous calculons **Coût 1** auquel nous rajoutons n **Coût 2** et nous rajoutons aussi **Coût 3** afin de trouver le coût du calcul de n couplages Optimal Ate sur toute courbe elliptique.
Nous pouvons calculer le produit ou/et le quotient de n couplages Optimal Ate sur plusieurs courbes elliptiques, pour un niveau de sécurité élevé, que nous avons présenté dans le Tableau 4.17. Nous avons fait ce calcul sur les courbes KSS16 en utilisant le résultat de la Section 4.3 de ce travail et sur les courbes BLS12 en utilisant le résultat de la Section 4.1. Nous avons comparé la complexité du calcul de n couplages sur ces courbes elliptiques. Nous obtenons le tableau suivant :

Coûts	**KSS16 Zhang et Lin corrigé**	**KSS16 Notre nouveau vecteur**	**BLS12** [GF16b]	**BN** [AFK$^+$12]	**KSS18** [AFK$^+$12]
Coût 1	2592M	2592M	5892M	8837M	4158M
Coût 2	7616M	7616M	10760M	16720M	9544M
Coût 3	23662M+I	22888M+I	12574M+6I	11145M+6I	23821M+8I
Coût total pour $n = 1$	33870M +I	33096M +I	29226M +6I	36702M +6I	37523M +8I
Coût Total pour $n = 2$	41486M +I	40712M +I	**39986M** +6I	53422M +6I	47067M +8I
Coût total pour $n = 3$	49102M +I	**48328M** +I	50746M +6I	64567M +6I	56611M +8I
Coût total pour $n = 7$	79656M +I	**78792M** +I	93786M +6I	109147M +6I	94784M +8I

TABLE 4.18 – Le complexité du calcul de produit de n couplages.

Ainsi pour $n = 2$, il faut plutôt utiliser les courbes BLS12 pour calculer le produit de deux couplages Optimal Ate pour un niveau de sécurité élevé. Mais à partir de $n > 2$, nous confirmons le résultat annoncé par Zhang *et al.* dans [ZL12] et nous affirmons que la courbe KSS16, est dans ce cas, la meilleure solution pour effectuer le produit de n couplages.
Donc grâce à nos résultats (présentées dans [GF16b] et [GF16a]), nous avons montré que les courbes KSS16 ne sont pas toujours les plus adéquates pour le calcul de produit et/ou de quotient de n couplages contrairement à ce qui était admis jusque là.

4.5 Résistance de la courbe KSS16 contre les attaques par les sous-groupes

En calculant les couplages sur les courbes elliptiques, nous devons en premier lieu vérifier la sécurité d'une telle courbe. Dans cette section, nous montrons que la courbe KSS16 proposée par Zhang et Lin dans [ZL12] ne résiste pas aux attaques par les sous-groupes. Nous proposons ensuite une nouvelle paramétrisation d'une courbe elliptique sécurisée.
Une étude détaillée sur la sécurité des courbes elliptiques bien adaptées aux couplages a été récemment présentée par Barreto *et al.* dans [BCM$^+$15] dans le cas où les courbes elliptiques possèdent un twist de degré $d = 6$. Dans ce contexte, Barreto *et al.* ont montré que les courbes BN, BLS12, BLS24 et KSS18 recommandées dans la littérature ne résistent pas à cette attaque. Ensuite ils ont présenté de nouveaux paramètres de chaque catégorie des courbes elliptiques pour qu'elles résistent à cette attaque.

Dans cette section, nous prolongeons les mêmes analyses que celles présentées dans [BCM$^+$15] pour les courbes elliptiques possédant un twist de degré $d = 4$. Plus précisément, nous nous intéressons à la courbe elliptique KSS16 et nous cherchons un nouveau paramètre qui la rend plus résistante aux attaques par les sous-groupes.
Soit E une courbe elliptique de degré de plongement k et paramétrée par les polynômes $p(u)$,

$t(u)$, $r(u) \in \mathbb{Q}[u]$ que nous avons définis dans la Partie 4.3.1. Soit $E'(\mathbb{F}_{p^{k/d}})$ la twist de la courbe elliptique E et d son degré.
Pour définir cette attaque, nous considérons les indices des trois groupes où le couplage est défini.

— $h_1(u) = \dfrac{\mid E(\mathbb{F}_p)(u) \mid}{r(u)}$,

— $h_2(u) = \dfrac{\mid E'(\mathbb{F}_{p^{k/d}})(u) \mid}{r(u)}$,

— $h_T = \dfrac{\mid G_{\phi_k}(p(u)) \mid}{r(u)}$.

Définition 4.3. *[BCM+15] Nous disons qu'une courbe elliptique E résiste aux attaques par les sous-groupes si tous les facteurs irréductibles de $h_1(u)$, $h_2(u)$, $h_T(u)$ ne contiennent pas de facteurs premiers inférieur à $r(u_0)$ lorsque $u = u_0$ dont le degré est au moins celui de u.*

Dans le cas des courbes elliptiques KSS16, les indices sont donnés dans la proposition suivante :

Proposition 4.6. *Soit $p(u)$, $t(u)$, $r(u) \in \mathbb{Q}[u]$ les paramètres de la courbe elliptique KSS16. Nous avons $h_1(u) = (125/2)(u^2 + 2u + 5)$ est un polynôme de degré 2, $h_T(u)$ est un polynôme de degré 72 et $h_2(u)$ est un polynôme de degré 32 donné par*
$(1/15059072)(u^{32} + 8u^{31} + 44u^{30} + 152u^{29} + 550u^{28} + 2136u^{27} + 8780u^{26} + 28936u^{25} + 83108u^{24} +$
$236072u^{23} + 754020u^{22} + 2287480u^{21} + 5986066u^{20} + 14139064u^{19} + 35932740u^{18} + 97017000u^{17} +$
$237924870u^{16} + 498534968u^{15} + 1023955620u^{14} + 2353482920u^{13} + 5383092978u^{12} + 10357467880u^{11} +$
$17391227652u^{10} + 31819075896u^9 + 65442538660u^8 + 117077934360u^7 + 162104974700u^6 + 208762740168u^5 +$
$338870825094u^4 + 552745197960u^3 + 632358687500u^2 + 414961135000u + 126854087873)$.

Preuve. Nous savons que l'ordre du groupe $E(\mathbb{F}_{p^4})$ est $\mid E(\mathbb{F}_{p^4}) \mid = p^4 + 1 - t_4$ avec $t_4 = t^4 - 4pt^2 + 2p^2$ (voir [Was08, Theorem 4.12]). L'ordre de la courbe tordue $E'(\mathbb{F}_{p^4})$ est donné par $\mid E'(\mathbb{F}_{p^4}) \mid = p^4 + 1 + v_4$ où $v_4^2 = 4p^4 - t_4^2$ (voir [HSV06b, Proposition 2]). Un calcul direct donne $h_2(u) = \frac{p^4+1+v_4}{r(u)}$. De même un calcul direct de $h_T(u) = \frac{p(u)^8+1}{r(u)}$ donne le polynôme de degré 72. □

Remarque 4.7. *La valeur du paramètre u utilisée dans [ZL12] pour le calcul du couplage Optimal Ate sur les courbes elliptiques KSS16 est $u_0 = 2^{49} + 2^{26} + 2^{15} - 2^7 - 1$. En utilisant cette valeur et en calculant $h_2(u_0)$, nous remarquons que la quantité $h_2(u_0)$ possède la factorisation suivante :*

$$h_2(u_0) = 2 \cdot 1249 \cdot 366593 \cdot c_{1515} \text{ avec } c_{1515} \text{ est un entier composé de } 1515 \text{ bits.}$$

Cela signifie que la courbe elliptique présentée dans [ZL12] avec le choix de u_0 de Zhang et al. *ne résiste pas aux attaques par les sous-groupes.*
Pour cette raison, nous allons chercher dans la suite de notre travail un paramètre u de la courbe elliptique qui la rend résistante contre cette attaque.

Pour un niveau de sécurité élevé, le paramètre u_0 qui donne les bonnes tailles de r et de p doit être un entier de taille au moins égal à 49. Le paramètre u_0 que nous cherchons doit être tel que $p(u_0), r(u_0)$, $h_2(u_0)$ et $h_T(u_0)$ sont tous premiers.
Lors de notre recherche, nous avons remarqué que $h_2(u) \equiv 0 \bmod 2$ et $h_T(u) \equiv 0 \bmod 2$. Par conséquent, nous allons chercher un u_0 de telle sorte que $p(u_0), r(u_0)$, $h_2(u_0)/2$ et $h_T(u_0)/2$ sont tous premiers. Nous pouvons trouver un tel u_0 si et seulement si ces polynômes satisfont la propriété de Bunyakovsky [BCM+15].
En vérifiant ces polynômes, nous constatons que le nombre premier 17 les divise. Cependant, il

suffit de chercher des nombres premiers avec les facteurs 2 et/ou 17.
La conjecture de Batemann-Horn [BCM+15] assure qu'il existe approximativement 24500 valeurs de $u_0 \in [2^{49}, 2^{53}]$ avec $p(u_0), r'(u_0), h'_2(u_0)$ et $h'_T(u_0)$ simultanément premiers, avec $r(u) = 17^{n_1} \cdot r'(u)$, $h_2(u) = 2 \cdot 17^{n_2} \cdot h'_2(u)$ et $h_T = 2 \cdot 17^{n_3} \cdot h'_T(u)$ pour certains entiers positifs ou nuls n_1, n_2 et n_3. Une recherche minutieuse nous a permis, après plusieurs essais en commençant par u_0 de poids de Hamming égal à 5, d'obtenir le paramètre u_0 suivant :

$$u_0 = 2^{50} + 2^{47} - 2^{38} + 2^{32} + 2^{25} - 2^{15} - 2^5 - 1.$$

Ce paramètre donne p premier de taille 492 bits , $r(u_0) = r'(u_0)$, premier de taille 386 bits, $h_2(u_0) = 2 \cdot 17 \cdot h'_2(u_0)$ et $h_T = 2 \cdot 17 \cdot h'_T(u_0)$ avec $h'_2(u_0)$ et $h'_T(u_0)$ sont premiers de tailles respectives 3544 bits et 1577 bits. Pour la valeur de p obtenue, le corps fini $\mathbb{F}_{p^{16}}$ est construit en utilisant la tour d'extension suivante :

$$\begin{aligned}
\mathbb{F}_{p^2} &= \mathbb{F}_p[\alpha]/(\alpha^2 - 11), \\
\mathbb{F}_{p^4} &= \mathbb{F}_{p^2}[\beta]/\big(\beta^2 - \alpha\big), \\
\mathbb{F}_{p^8} &= \mathbb{F}_{p^4}[\gamma]/\big(\gamma^2 - \beta\big), \\
\mathbb{F}_{p^{16}} &= \mathbb{F}_{p^8}[\theta]/\big(\theta^2 - \gamma\big).
\end{aligned}$$

De plus, une courbe elliptique E définie sur $\mathbb{F}_p$ qui satisfait $|E(\mathbb{F}_p)| = p + 1 - t$ est donnée par l'équation

$$E : \quad y^2 = x^3 + 17x.$$

Le twist E' de E de degré 4 définie sur $\mathbb{F}_{p^4}$ d'ordre $|E'(F_{p^4})| = 2 \cdot 17 \cdot h'_2(u_0) \cdot r(u_0)$ par l'équation

$$E' : \quad y^2 = x^3 + 17/\beta x.$$

4.6 Conclusion

Dans ce chapitre, nous nous sommes intéressés au calcul du couplage Optimal Ate sur les courbes elliptiques qui possèdent un haut niveau de sécurité. En premier lieu, nous avons présenté les courbes BLS de degré de plongement k égal à 12, BLS12. Nous avons concentré notre travail sur le calcul de l'exponentiation finale de ce couplage. Nous avons présenté une manière de calcul de la partie difficile de l'exponentiation finale plus efficace que celle présentée dans la littérature. Notre nouveau développement nous a permis de gagner 2 multiplications et 2 élévations au carré dans $\mathbb{F}_{p^{12}}$. Nous avons traité aussi le problème de mémoire dans cette partie. En effet, en pensant à des implémentations du couplage Optimal Ate dans des environnements restreints, nous devons minimiser la mémoire utilisée. Dans ce contexte, nous avons réussi à gagner 26.6% sur la mémoire, ce qui est un gain très important. Dans le même cadre nous avons proposé un nouveau paramètre u d'une courbe BLS12 grâce auquel nous arrivons à réduire la complexité du calcul de l'exponentiation finale ainsi que celle de l'algorithme de Miller. Ainsi, nous avons présenté une manière plus efficace que celle présentée dans la littérature pour le calcul du couplage Optimal Ate.
Nous avons effectué un travail similaire pour les courbes BLS24 utiles pour les hauts niveaux de sécurité.
En deuxième lieu dans ce chapitre, nous avons étudié le problème du calcul de produit ou du quotient de n couplages. Notre but était de trouver la courbe elliptique la plus adéquate pour ce calcul de n couplages pour un haut niveau de sécurité. Il y a peu de travaux dans la littérature qui

ont traité ce problème. Récemment, Zhang et Lin [ZL12] ont affirmé que les courbes KSS16 était le meilleur candidat pour calcul du produit ou du quotient de n couplages. Nous avons repris leurs travaux de calcul de la partie difficile de l'exponentiation finale et avons révisé leurs formules. Puis nous avons présenté une nouvelle méthode du calcul de l'exponentiation finale plus efficace et nous avons fait une étude comparative du choix de la catégorie des courbes elliptiques sur laquelle nous calculons le produit ou/et le quotient de n couplages plus rapidement. Dans ce cadre, nous avons trouvé que la courbe suggérée par Zhang et Lin n'est pas toujours le meilleur candidat. En effet, pour $n < 3$, c'est plutôt mieux de choisir BLS12. Á la fin de cette deuxième partie, nous avons proposé une courbe KSS16 qui résiste aux attaques par les sous-groupes alors que celle proposée par Zhang et Lin n'y résiste pas.

Chapitre 5

Contre-mesures et attaques sur l'algorithme de Miller

5.1 Introduction

Dans ce chapitre, nous proposons deux nouvelles versions de l'algorithme de Miller basées sur la représentation du paramètre du couplage (l'entier qui intervient dans le calcul de la fonction de Miller, généralement u ou par exemple $(6u + 2)$ pour le couplage Optimal Ate sur les courbes BN). Nous écrivons d'abord le paramètre du couplage à l'aide d'une chaîne d'additions euclidienne. Dans ce cas, nous appelons notre nouvel algorithme Miller-Euclide. Le deuxième algorithme, que nous appelons Miller-Fibonacci, est quant à lui basé sur les suites de Fibonacci.

Après avoir présenté ces deux algorithmes, nous calculons leur complexité et nous les comparons. Dans la suite de ce chapitre nous nous concentrons sur le plus efficace des deux, Miller-Fibonacci. Nous étudions les attaques existantes sur l'algorithme de Miller classique ainsi que leurs effets sur nos nouvelles versions. Dans ce contexte, nous montrons que nos versions résistent aux attaques par analyse différentielle par consommation de courant dite **DPA** et constituent donc des contre-mesures pour cette attaque. Nous montrons que l'utilisation de cet algorithme comme contre-mesure est plus efficace que celles proposées dans la littérature avec un gain en vitesse d'environ **10.7**% pour le calcul protégé de la fonction de Miller.

5.2 Nouvelles versions de l'algorithme de Miller

Dans cette section, nous proposons notre nouvel algorithme : Miller-Euclide. Comme nous l'avons dit, cet algorithme est basé sur l'écriture du paramètre du couplage, que nous notons u, à l'aide d'une chaîne d'additions euclidienne. Définissons tout d'abord les chaînes d'additions euclidiennes.

5.2.1 Chaîne d'additions euclidienne et algorithme de Miller

Définition 5.1. *Une chaîne d'additions euclidienne calculant un entier k est une chaîne d'additions telle que :*

1. $v_1 = 1$, $v_2 = 2$ *et* $v_3 = v_2 + v_1 = 3$

2. $\forall\ 3 \leqslant i \leqslant s-1$, *si* $v_i = v_{i-1} + v_j$, *pour un certain* $j < i-1$, *alors nous avons :*

$$v_{i+1} = v_i + v_{i-1} \text{ ou } v_{i+1} = v_i + v_j$$

le premier cas, où nous ajoutons à v_i *le plus grand des 2 entiers précédents, est appelé grand pas. Le deuxième cas est appelé petit pas.*

Exemple 5.1. *[Mel07b] La chaîne d'additions* (1, 2, 3, 4, 7, 11, 15, 19, 34) *est une chaîne d'additions euclidienne qui permet de calculer* 34. *En fait, à l'étape 4 nous calculons* $4 = 3 + 1$. *Pour l'étape 5, nous pouvons calculer* $7 = 4 + 3$ *ou* $5 = 4 + 1$. *Dans cet exemple, nous choisissons de calculer* $7 = 4 + 3$ *(grand pas). Pour l'étape 6, nous calculons* $11 = 7 + 4$ *(grand pas)....*

Nous pouvons trouver une telle chaîne d'additions d'une manière simple. Il suffit de choisir un entier g premier avec k puis d'appliquer l'algorithme d'Euclide additif. Soit l'exemple suivant :

Exemple 5.2. *[Mel07b] Nous voulons présenter* 34 *sous la forme d'une chaîne d'additions euclidienne. Soit* $g = 19$, $\gcd(34,\ 19) = 1$. *L'algorithme d'Euclide additif donne :*

$$\begin{aligned} 34 - 19 &= 15 \\ 19 - 15 &= 4 \\ 15 - 4 &= 11 \\ 11 - 4 &= 7 \\ 7 - 4 &= 3 \\ 4 - 3 &= 1 \\ 3 - 1 &= 2 \\ 2 - 1 &= 1 \\ 1 - 1 &= 0 \end{aligned}$$

Une chaîne d'additions euclidienne associée à 34 *est alors donnée par le premier entier de chaque ligne de l'algorithme d'Euclide additif.*

D'une manière générale, pour simplifier la présentation d'un entier à l'aide d'une chaîne d'additions euclidienne $v = (1,\ 2,\ 3,\ v_4,\ \ldots v_s)$, nous ne considérons cette chaîne qu'à partir du quatrième terme puisqu'il est clair que les trois premiers termes sont toujours 1, 2, et 3. De plus nous notons 1 s'il s'agit d'un petit pas et 0 s'il s'agit d'un grand pas. La chaîne d'additions euclidienne de l'exemple 5.1 sera représentée par (1, 0, 0, 1, 1, 0). Nous représentons ainsi tout entier en chaîne d'additions euclidienne par un vecteur $c = (c_4,\ \ldots\ldots c_s)$ avec $c_i = 0$ ou 1.

Notre démarche consiste à représenter le paramètre u du couplage sous la forme d'une chaîne d'additions euclidienne. Nous devons donc adapter l'algorithme de Miller à une telle représentation. L'algorithme de Miller classique est dérivé de l'algorithme de multiplication scalaire double-and-add. Nous rappelons donc tout d'abord l'algorithme 5 de la multiplication scalaire utilisant les chaînes d'additions euclidiennes.

Algorithm 5 [Mel07b] Calcul de $[k]P$ à l'aide d'une chaîne d'additions euclidienne

Require: $P \in E, c = (c_4, \ldots, \ldots c_s)$ une chaîne d'additions euclidienne calculant k
Ensure: $[k]P \in E$
  $(U_1, U_2) \leftarrow ([2]P, P)$
  for $i = 4$ **to** s **do**
    if $c_i = 0$ **then**
      $(U_1, U_2) \leftarrow (U_1 + U_2, U_1)$
    else
      $(U_1, U_2) \leftarrow (U_1 + U_2, U_2)$
    end if
  end for
  $U_1 \leftarrow U_1 + U_2$
  return U_1

Exemple 5.3. *Une chaîne d'additions euclidienne pour* 34 *est* (1, 0, 0, 1, 1, 0)*. Nous appliquons l'algorithme 5 afin de trouver* $[34]P$*.* U_1 *et* U_2 *sont d'abord initialisés à* $[2]P$ *et* P*. Puis,*

comme $c_4 = 1$ *, nous calculons* $(U_1, U_2) = ([3]P, P)$*,*
comme $c_5 = 0$*, nous calculons* $(U_1, U_2) = ([4]P, [3]P)$*,*
comme $c_6 = 0$*, nous calculons* $(U_1, U_2) = ([7]P, [4]P)$*,*
comme $c_7 = 1$*, nous calculons* $(U_1, U_2) = ([11]P, [4]P)$*,*
comme $c_8 = 1$*, nous calculons* $(U_1, U_2) = ([15]P, [4]P)$*,*
comme $c_9 = 0$ *alors, nous calculons* $(U_1, U_2) = ([19]P, [15]P)$*,*
À la fin nous déduisons $[34]P$ *en calculant* $[19]P + [15]P$*.*

Adaptons maintenant ce principe pour notre nouvel algorithme Miller-Euclide de calcul de $f_{u,Q}(P)$.

Algorithm 6 : Miller-Euclide, Calcul de $f_{u,Q}(P)$ à l'aide d'une chaîne euclidienne.

Require: $P \in G_1$, $Q \in G_2$, $c = (c_4, \ldots, \ldots c_s)$ une chaîne d'additions euclidienne pour u,
Ensure: $f_{u,Q}(P)$
1: $(T_1, T_2) \leftarrow ([2]Q, Q)$
2: $(f_1, f_2) \leftarrow \left(l_{[2]Q,Q}(P), l_{Q,Q}(P)\right)$
3: **for** $i = 4$ **to** s **do**
4: **if** $c_i = 0$ **then**
5: $(f_1, f_2) \leftarrow (f_1 \times f_2 \times \ell_{T_1,T_2}(P), f_1)$
6: $(T_1, T_2) \leftarrow (T_1 + T_2, T_1)$
7: **else**
8: $(f_1, f_2) \leftarrow (f_1 \times f_2 \times \ell_{T_1,T_2}(P), f_2)$
9: $(T_1, T_2) \leftarrow (T_1 + T_2, T_2)$
10: **end if**
11: $f_1 \leftarrow f_1 \times f_2 \times \ell_{T_1+T_2,T_1}(P)$
12: **end for**
13: **return** f_1

Nous remarquons que, une fois l'initialisation effectuée, nous n'avons pas d' étape de doublement à faire comme c'est le cas dans l'Algorithme 2. Nous n'avons que des étapes d'additions. Cela ne nous permet pas de déduire une comparaison facile entre la complexité de l'Algorithme 2 de Miller classique et celle de l'Algorithme 6 de Miller-Euclide. En effet nous ne connaissons pas la longueur de la chaîne d'addition par rapport à la longueur du développement binaire de u. C'est ce à quoi nous allons nous atteler maintenant.

Longueur de la chaîne d'additions euclidienne

Nous pouvons facilement trouver une chaîne d'additions calculant tout entier k grâce à l'algorithme d'Euclide additif. Mais le problème qui se pose est la longueur de cette chaîne.
Pour avoir un algorithme de Miller-Euclide plus efficace, il faut que cette chaîne soit la plus courte possible afin d'avoir le minimum d'opérations à effectuer.
La longueur d'une chaîne d'additions euclidienne est donnée par le théorème suivant démontré en 1975 par Knuth et Yao [KY75].

Théorème 5.1. *Soit $S(k)$ le nombre moyen d'étapes pour calculer le pgcd de k et g avec $1 < g < k$, en utilisant l'algorithme d'Euclide additif. Alors nous avons :*

$$S(k) = 6\pi^{-2}(\ln k)^2 + O\left(\ln k(\ln\ \ln k)^2\right)$$

Ce théorème montre que si nous choisissons l'entier k d'une manière aléatoire, la longueur de la chaîne d'additions euclidienne calculant k est de l'ordre de $(\ln k)^2$ alors que la longueur de son développement binaire est en $\ln k$. Si, par exemple, le paramètre du couplage est de taille 65 bits, la longueur de la chaîne d'additions euclidienne sera d'environ 2500. Ainsi, tandis que la boucle de Miller nécessite 65 itérations, celle de l'Algorithme 6 en nécessitera 2500.

Nous déduisons donc de ce théorème que le calcul de $f_{u,Q}(P)$ est bien plus coûteux en utilisant notre nouvel algorithme 6 Miller-Euclide qu'en utilisant l'algorithme 2 de Miller classique.

Knuth et Yao expliquent que pour limiter la longueur d'une chaîne d'additions euclidienne, il ne faut pas choisir g d'une manière aléatoire.

Théorème 5.2. *[KY75] Pour limiter la longueur d'une chaîne d'additions euclidienne, g doit être choisi assez proche de $\frac{k}{\varphi}$ où $\varphi = \frac{1+\sqrt{5}}{2}$ est le nombre d'or.*

Malheureusement, en pratique ce n'est pas du tout évident de trouver une telle chaîne courte qui facilite le calcul de l'algorithme de Miller. Par exemple, Meloni montre dans [Mel07a] que pour trouver une représentation d'un entier k de taille 160 bits, en une chaîne d'additions euclidienne de longueur 270 bits, il faut au moins 45000 essais pour la recherche d'un g optimal qui satisfait toutes les conditions. L'algorithme de calcul est bien détaillé dans [Mel07a].
Par contre, d'après [Mel07c], il faut 29 essais en moyenne pour avoir une chaîne d'additions euclidienne de taille 320 bits.
Cela peut être gênant pour le calcul de $[k]P$ pour n'importe quelle valeur k, car, en changeant l'entier k, il faut trouver une nouvelle chaîne d'addition euclidienne optimale calculant cet entier. Par contre dans notre cas, puisque le paramètre du couplage est fixé, nous n'aurons besoin que d'une seule représentation de ce paramètre à l'aide d'une chaîne d'additions euclidienne optimale que nous utilisons par la suite pour chaque calcul de couplage.

Complexité de l'algorithme Miller-Euclide

Dans ce paragraphe, nous donnons la complexité détaillée de l'algorithme 6 de Miller-Euclide. Comme nous l'avons déjà dit, cet algorithme n'est basé que sur des étapes d'addition. Il est facile de voir que dans le cas où une tordue de E est utilisée pour G_2, chacune de ces étapes se déroule comme suit :

— Évaluation de la droite passant par les points T_1 et T_2 et mise à jour du point $T_1 \longleftarrow T_1 + T_2$,

— Une multiplication creuse dans $\mathbb{F}_{p^k}$,

— Une multiplication dans $\mathbb{F}_{p^k}$.

Pour évaluer le coût de l'algorithme de Miller-Euclide, nous allons utiliser les coordonnées présentées dans [Mel07b]. En effet Méloni a montré qu'en utilisant les coordonnées projectives avec la condition $Z_1 = Z_2 = Z$, l'addition de deux points sur la courbe elliptique est plus efficace que le doublement d'un point. Ceci est important dans notre cas puisque Miller-Euclide ne comporte que des étapes d'additions.

La somme de deux points T_1 et T_2 de coordonnées respectives (X_1, Y_1, Z) et (X_2, Y_2, Z) est le point $T_1 + T_2$ de coordonnées (X_3, Y_3, Z) données par :

$$\begin{cases} X_3 = (Y_1 - Y_2)^2 - (X_1 + X_2)(X_1 - X_2)^2 \\ Y_3 = (Y_1 - Y_2)\left(X_3 - X_1(X_1 - X_2)^2\right) - Y_1(X_1 - X_2)^3 \end{cases}$$

Ce calcul requiert exactement 4 multiplications et 2 carrés.
L'équation de la droite passant par les points T_1 et T_2 évaluée au point $P = (x_P, y_P)$ est :

$$l_{T_1,T_2}(P) = y_P Z^3 (X_1 - X_2)^3 - \left[Y_1(X_1 - X_2)^3 + (Y_1 - Y_2)\left(x_P Z^2 (X_1 - X_2)^2 - X_1(X_1 - X_2)^2\right)\right].$$

Si on pose $Y_1' = Y_1(X_1 - X_2)^3$, $X_1' = X_1(X_1 - X_2)^2$ et $Z_1' = Z(X_1 - X_2)$, $l_{T_1,T_2}(P)$ est donné par

$$l_{T_1,T_2}(P) = y_P {Z_1'}^3 - Y_1' - (Y_1 - Y_2)\left(x_P {Z_1'}^2 - X_1'\right).$$

Ainsi, l'évaluation de la droite passant par les points T_1 et T_2 et la mise à jour du point $T_1 + T_2$ coûtent au total 7 multiplications et 2 carrés.
Si on suppose que le point Q provient d'une tordue de E, ces calculs seront faits dans un sous corps $\mathbb{F}_{p^l}$ de $\mathbb{F}_{p^k}$. Par conséquent, chaque étape d'addition de l'algorithme 6 nécessite $M_k + sM_k + 7M_l + 2S_l$ opérations, (nous gardons les notations que nous avons utilisé dans le Chapitre 2.
Dans le cas le plus classique où $k = 12$ et $l = 2$, on obtient ainsi une complexité de $M_{12} + sM_{12} + 7M_2 + 2S_2 = 54M + 39M + 7 * 3M + 2 * 2M = 118M$. Nous donnerons un exemple de coût détaillé du calcul de la fonction $f_{u,Q}(P)$ en terme d'arithmétique sur $\mathbb{F}_p$ dans la section 5.2.3.

5.2.2 Suite de Fibonacci et algorithme de Miller

Comme nous l'avons vu, malheureusement, la longueur d'une chaîne d'additions euclidienne est un inconvénient pour le coût de l'algorithme 6 de Miller-Euclide. Il existe certainement une chaîne d'additions euclidienne optimale, c'est à dire dont la longueur est la plus petite possible. C'est le cas des suites de Fibonacci.

Définition 5.2. *On définit la suite de Fibonacci $(\mathcal{F}_n)_{n \geqslant 0}$ de la manière suivante :*

$$\mathcal{F}_n = \begin{cases} 0 \; si \; n = 0 \\ 1 \; si \; n = 1 \\ \mathcal{F}_{n-1} + \mathcal{F}_{n-2} \; si \; n \geqslant 2 \end{cases}$$

Les termes de cette suite sont appelés les nombres de Fibonacci.

La suite de Fibonacci est donc la chaîne d'additions euclidienne qui ne possède que des grands pas. C'est par exemple le cas de la chaîne (1, 2, 3, 5, 8, 13, 21, 34) est une chaîne d'additions euclidienne calculant 34. Nous remarquons que cette chaîne ne possède que des grands pas, c'est donc une suite de Fibonacci, 34 est dit un nombre de Fibonacci. Il est facile de voir que $F_9 = 34$ et la suite de Fibonacci (1, 2, 3, 5, 8, 13, 21, 34) calculant 34 n'est autre que ($F_1 \longrightarrow F_2 \longrightarrow F_3 \longrightarrow F_4 \longrightarrow F_5 \longrightarrow F_6 \longrightarrow F_7 \longrightarrow F_8 \longrightarrow F_9$).
Le calcul de la multiplication scalaire $[k]P$ où k est un nombre de Fibonacci se fait d'une manière simple à l'aide de l'algorithme présenté dans [Mel07b] .
Pour calculer la fonction rationnelle $f_{u,Q}$ évaluée au point P nous devons d'abord représenter le paramètre du couplage u, à l'aide d'une suite de Fibonacci.

Mais u n'est pas forcément un nombre de Fibonacci. Pour cette raison, nous allons représenter tout entier k comme somme de nombres de Fibonacci : nous parlons dans ce cas de la représentation de Zeckendorf [Mel07b]. Pour cela, nous utilisons le théorème suivant :

Théorème 5.3. *[Zec72] Soit k un entier et $(F_i)_{i\geqslant 0}$ la suite de Fibonacci. Alors k peut s'écrire de manière unique sous la forme :*

$$k = \sum_{i=2}^{n} d_i F_i$$

avec $d_i \in \{0, 1\}$ et $d_i d_{i+1} = 0$. Cette écriture est appelée représentation de Zeckendorf de k. On note $k = (d_n, \ldots\ldots d_2)_Z$.

L'inconvénient de cette écriture, comme il est indiqué dans [Zec72] et [Mel07b], est qu'elle requiert 44% de chiffres de plus que la représentation binaire. Cependant, elle est plus courte que la représentation à l'aide d'une chaîne d'additions euclidienne.

Exemple 5.4. *[Mel07a, Page 45] Le plus grand nombre de Fibonacci inférieur à 2^{160} est F_{232}. Cela signifie que la représentation de Zeckendorf d'un entier de 160 bits nécessite au moins 230 chiffres, soit environ 160×1.44.*

Nous rappelons l'algorithme de la multiplication scalaire utilisant les suites de Fibonacci.

Algorithm 7 [Mel07b] Calcul de $[k]P$ à l'aide d'une suite de Fibonacci

Require: $P \in E, k = (d_n, \ldots, \ldots d_2)_Z$ la représentation de Zeckendorf de k avec $d_n = 1$
Ensure: $[k]P \in E$
1: $(U_1, U_2) \leftarrow (P, P)$
2: **for** $i = n - 1$ **to** 2 **do**
3: **if** $d_i = 1$ **then**
4: $(U_1, U_2) \leftarrow (U_1 + P, U_2)$
5: **end if**
6: $(U_1, U_2) \leftarrow (U_1 + U_2, U_1)$
7: **end for**
8: **return** U_1

Exemple 5.5. *Le calcul de* $[25]P$ *avec* $25 = 21 + 3 + 1 = F_8 + F_4 + F_2 = (1\,0\,0\,0\,1\,0\,1)_Z$.
Initialisation : $(U_1, U_2) = (P, P)$,

comme $d_7 = 0$, *nous calculons* $(U_1, U_2) = ([2]P, P)$,
comme $d_6 = 0$, *nous calculons* $(U_1, U_2) = ([3]P, [2]P)$,
comme $d_5 = 0$, *nous calculons* $(U_1, U_2) = ([5]P, [3]P)$,
comme $d_4 = 1$, *alors* $U_1 = [6]P$, *puis nous calculons* $(U_1, U_2) = ([9]P, [6]P)$,
comme $d_3 = 0$, *nous calculons* $(U_1, U_2) = ([15]P, [9]P)$,
comme $d_2 = 1$, *alors* $U_1 = [16]P$, *puis nous calculons* $(U_1, U_2) = ([25]P, [16]P)$.
À la fin, l'algorithme retourne $[25]P$.

Dans ce qui suit, nous allons présenter une nouvelle version de l'algorithme de Miller, que nous appelons Miller-Fibonacci, où nous écrivons u à l'aide des suites de Fibonacci. Adaptons le principe de l'algorithme 7 pour notre nouvel algorithme de calcul de $f_{u,Q}(P)$.

Algorithm 8 : Miller-Fibonacci : Le calcul de $f_{u,Q}(P)$ à l'aide d'une suite de Fibonacci

Require: $P \in G_1$, $Q \in G_2$, $u = (u_n, \ldots, \ldots u_2)$ la représentation de Zeckendorf de u,
Ensure: $f_{u,Q}(P)$
1: $(T_1, T_2) \leftarrow (Q, Q)$
2: $(f_1, f_2) \leftarrow (1, 1)$
3: **for** $i = n - 1$ **down to** 2 **do**
4: **if** $u_i = 1$ **then**
5: $(f_1, f_2) \leftarrow (f_1 \times l_{T_1,T_2}(P), f_2)$
6: $(T_1, T_2) \leftarrow (T_1 + Q, T_2)$
7: **end if**
8: $(f_1, f_2) \leftarrow (f_1 \times f_2 \times l_{T_1,T_2}(P), f_1)$
9: $(T_1, T_2) \leftarrow (T_1 + T_2, T_1)$
10: **end for**
11: **return** f_1

Remarquons que pour cet algorithme, il n'y a pas d'étape de doublement comme dans le cas de la version classique de l'algorithme de Miller. Chaque étape de cet algorithme est similaire à celles de l'algorithme Miller-Euclide et a donc la même complexité (une multiplication creuse dans $\mathbb{F}_{p^k}$, une multiplication dans $\mathbb{F}_{p^k}$ et l'évaluation de la droite passant par deux points de la courbe ainsi

que leurs mises à jour). De plus, si le chiffre que nous traitons vaut 1, nous avons une multiplication creuse supplémentaire et l'évaluation de la droite passant par deux points de la courbe ainsi que leurs mises à jour.
Cette complexité est supérieure à celle d'une étape de l'algorithme de Miller classique. De plus, la longueur de la représentation de Zeckendorf est de l'ordre de 44% plus longue que celle en base 2. Généralement, pour cette représentation, la densité moyenne de 1 est d'environ 27% (voir [Mel07a, Page 45-46], [Mel07b], [Zec72]). Autrement dit, un entier quelconque de l bits est donné à l'aide de la représentation de Zeckendorf avec $1,44 \times l$ chiffres dont la densité de 1 est $0,2764 \times 1,44 \times l = 0,398 \times l$.
Il est donc clair que le coût de cet algorithme Miller-Fibonacci est supérieur à celui de l'Algorithme 2 de Miller classique. De même, il est facile de voir que l'Algorithme 8 Miller-Fibonacci est moins coûteux que celui de Miller-Euclide.
Puisque, pour Miller-Fibonacci, nous avons uniquement des étapes d'additions à effectuer, nous allons utiliser l'équation de la droite $l_{T_1,T_2}(P)$ présentée dans le Paragraphe 5.2.1 et pour la mise à jour du point $T_1 + T_2$, nous allons utiliser les coordonnées présentées dans [Mel07b].
Nous détaillons dans le paragraphe suivant le coût du calcul de $f_{u,Q}$ en utilisant les différentes versions de l'algorithme de Miller que nous proposons dans ce chapitre.

5.2.3 Comparaison des algorithmes de calcul de $f_{u,Q}(P)$

Nous avons maintenant trois versions des algorithmes qui nous permettent de calculer la fonction rationnelle $f_{u,Q}$ évaluée au point P : Miller classique, Miller-Euclide et Miller-Fibonacci. Puisque nous nous intéressons au calcul efficace de la boucle de Miller, nous allons comparer ces algorithmes. Nous rappelons tout d'abord le nombre d'opérations à effectuer pour chaque algorithme dans le cas où une tordue de E est utilisée pour G_2.

1. **Algorithme 2 : Miller classique.**
   Chaque étape de cet algorithme est constitué d'une étape de doublement et éventuellement d'une étape d'addition. Nous avons ainsi l_u étapes de doublements et w_u étapes d'additions. Rappelons que toute étape de doublement consiste à effectuer un carré dans $\mathbb{F}_{p^k}$, une évaluation de la tangente au point T, la mise à jour de ce point T, et une multiplication creuse dans $\mathbb{F}_{p^k}$. L'étape d'addition est moins coûteuse que celle de doublement. Il s'agit d'une évaluation de la droite passant par les points Q et T, de la mise à jour du point T, ainsi que d'une multiplication creuse dans $\mathbb{F}_{p^k}$.
2. **Algorithme 6 : Miller-Euclide.**
   Cet algorithme est basé sur l'écriture de u à l'aide d'une chaîne d'additions euclidienne de longueur le_u (nous désignons par le_u la longueur de u en le présentant à l'aide d'une chaîne d'addition euclidienne). Il consiste donc en le_u étapes d'addition. Si la chaîne d'addition choisie est de longueur $2l_u$ comme le suggère [Mel07c], nous devons alors effectuer environ $2l_u$ étapes dans l'algorithme, chacune constituée d'une multiplication dans $\mathbb{F}_{p^k}$, d'une multiplication creuse dans $\mathbb{F}_{p^k}$, de l'évaluation de la droite passant par T_1 et T_2 et de la mise à jour du point T_1.
3. **Algorithme 8 : Miller-Fibonacci.**
   Comme nous l'avons déjà vu, l'algorithme de Miller-Fibonacci est une optimisation de l'algorithme de Miller-Euclide. Nous aurons moins d'étapes d'additions à effectuer que l'algorithme Miller-Euclide. En effet, d'après le Théorème 5.3, pour notre algorithme 8 nous devons faire

$1.44 \times l_u$ étapes d'additions. Chacune de ces étapes consiste en une multiplication dans $\mathbb{F}_{p^k}$, l'évaluation de la droite passant par deux points ainsi que la mise à jour de ces points et une multiplication creuse dans $\mathbb{F}_{p^k}$. De plus, comme nous l'avons dit, nous avons à effectuer une étape d'addition supplémentaire quand le bit vaut 1. Cette étape nécessite une multiplication creuse, l'évaluation de la droite passant par deux points et la mise à jour de ceux points.

Nous allons donner le coût explicite de chaque opération afin d'avoir une comparaison plus concrète dans le contexte le plus courant, celui du couplage Optimal Ate sur les courbes BN. Nous choisissons la courbe elliptique

$$E(\mathbb{F}_p): \ y^2 = x^3 + 2$$

et l'extension $\mathbb{F}_{p^{12}}$ construite en utilisant la tour d'extensions suivante :

$$\begin{aligned} \mathbb{F}_{p^2} &= \mathbb{F}_p[\mathbf{i}] \text{ avec } \mathbf{i}^2 = -1 \\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[\beta] \text{ avec } \beta^3 = 1 + \mathbf{i} \\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^2}[\gamma] \text{ avec } \gamma^6 = 1 + \mathbf{i}. \end{aligned}$$

Considérons également la courbe elliptique tordue définie par l'équation :

$$E'(\mathbb{F}_{p^2}) : y^2 = x^3 + (1 - i).$$

Et le morphisme

$$\begin{aligned} \Psi : E'\left(\mathbb{F}_{p^2}\right) &\rightarrow E(\mathbb{F}_{p^{12}}) \\ (x_{Q'}, y_{Q'}) &\mapsto (\gamma^2 x_{Q'}, \gamma^3 y_{Q'}) \end{aligned}$$

Rappelons que le couplage Optimal Ate défini sur les courbes BN est l'application suivante :

$$\begin{aligned} E(\mathbb{F}_p)[r] \times \Psi(E'(\mathbb{F}_{p^2}))[r] &\longrightarrow \mathbb{F}^{\star}_{p^{12}} \\ (P, Q) &\longmapsto = \left((f_{6u+2,Q}(P) l_{[6u+2]Q,\pi_p(Q)}(P) l_{[6u+2]Q,\pi_p^2(Q)}(P)) \right)^{\frac{p^{12}-1}{r}}. \end{aligned}$$

Nous avons déjà rappelé dans le chapitre 3 les coûts des opérations dans $\mathbb{F}_{p^{12}}$ pour le calcul de l'exponentiation finale. Dans le tableau suivant, nous présentons les coûts des opérations manquantes pour le calcul de l'algorithme de Miller classique ainsi que pour nos nouvelles versions. Plus précisément, nous donnons le coût de l'évaluation de la droite et la mise à jour du point dans le cas de l'étape de doublement ainsi que l'étape d'addition pour le cas de l'algorithme de Miller classique (pour lequel nous utiliserons le système des coordonnées projectives [DMHR15]). Dans le tableau suivant nous rappelons aussi les coûts de l'évaluation de la droite dans le cas de l'algorithme de Miller-Euclide et de Miller-Fibonacci que nous avons déjà donné dans le Paragraphe 5.2.1.

Opération	Coût dans $\mathbb{F}_p$
Multiplication dans $\mathbb{F}_{p^{12}}$	$54\,M$
Multiplication creuse dans $\mathbb{F}_{p^{12}}$	$39\,M$
Carré dans $\mathbb{F}_{p^{12}}$	$36\,M$
Étape d'addition pour Miller classique (coordonnées Projectives)	$80\,M$
Étape de doublement pour Miller classique (coordonnées Projectives)	$100\,M$
Étape d'addition pour Miller-Euclide ou Miller-Fibonacci	$118\,M$
Étape d'addition pour Miller-Fibonacci (pour $u_i = 1$)	$64\,M$

TABLE 5.1 – Coûts des opérations nécessaires pour Miller

En utilisant les coûts présentés dans le Tableau 5.1, nous obtenons la comparaison présentée dans le Tableau 5.2 entre les différentes versions de l'algorithme de Miller, pour le calcul de $f_{6u+2,Q}(P)$, en termes d'opérations dans $\mathbb{F}_p$. Nous rappelons que le paramètre du couplage dans notre cas est $6u + 2$ avec $u = -(4080000000000001)_{16}$. Ainsi, nous avons :

$$\begin{cases} l_{6u+2} = 65 \\ w_{6u+2} = 5 \end{cases}$$

Méthode	Algorithme	Complexité dans $\mathbb{F}_p$	Gain
Miller classique	2	$6720\,M$	
Miller-Euclide	6	$15104\,M$	$-\mathbf{125}\%$
Miller-Fibonacci	8	$12456\,M$	$-\mathbf{85}\%$

TABLE 5.2 – Coûts des opérations nécessaires pour la boucle de Miller

D'après ce tableau, nous remarquons que l'algorithme de Miller classique est plus efficace que nos deux nouvelles versions. L'algorithme de Miller-Fibonacci est plus efficace que celui de Miller-Euclide mais il reste moins efficace que celui de Miller classique, puisqu'il est plus lent de 85%.
Cependant, le paramètre u utilisé ici a été choisi pour optimiser Miller classique. On pourrait très bien imaginer un autre choix de u spécifiquement adapté à Miller-Fibonacci et minimisant le nombre d'étapes d'addition par Q.
Il faut également penser aux attaques sur l'algorithme de Miller, telles que les attaques par canaux cachés ou les attaques par injection de fautes. Nos nouvelles versions peuvent en effet être une solution efficace pour résister à ces attaques.
Dans la suite de ce chapitre, nous allons nous concentrer sur l'Algorithme 8 de Miller-Fibonacci vu qu'il est plus efficace que l'Algorithme 6 de Miller-Euclide et que les deux algorithmes sont basés sur le même principe.

5.3 L'attaque DPA

D'après les résultats de la littérature, [EDF09], l'algorithme de Miller classique est faible face aux attaques par canaux cachés de type DPA (acronyme anglais signifiant *Differential Power Analysis*).

Nous rappelons tout d'abord le principe de cette attaque. Puis nous montrons que notre nouvelle représentation de l'algorithme de Miller-Fibonacci résiste bien à cette attaque.

5.3.1 Principe de l'attaque DPA sur l'algorithme de Miller classique

Le principe de cette attaque est le suivant

— L'attaquant utilise un grand nombre de courbes de consommation de courant du fonctionnement de l'algorithme de Miller dont la clé sécrète est Q.

— Il cherche alors à attaquer une variable intermédiaire de l'algorithme de Miller qui dépend des coordonnées de Q. L'objectif de cette étape est de déterminer cette variable intermédiaire grâce aux fuites d'informations repérées dans les courbes de consommation. La fuite d'information provient d'opérations faisant intervenir une opérande connue à l'avance (comme les coordonnées de P) et une opérande inconnue (la variable intermédiaire attaquée).

— Une fois connue la variable intermédiaire visée, l'attaquant essaye d'en déduire le point Q.

— Enfin, à l'aide d'un test, il vérifie si le point trouvé est bien le secret cherché de l'algorithme de Miller.

Les variables temporaires visées dans l'attaque présentée dans [EDF09] sont les coordonnées (X, Y, Z) du point de la variable T de l'algorithme de Miller qui est un multiple du secret Q. Cette attaque vise l'étape de doublement de l'algorithme de Miller, plus précisément les opérations effectuées durant le calcul de la tangente à la courbe E au point T évaluée au point public $P = (x_P, y_P)$. Cette tangente a pour équation :

$$l_{T,T}(P) = Z_3 Z^2 y_P - 2Y^2 - (3X^2 - aZ^4)(Z^2 x_P - X)$$

avec, $Z_3 = 2YZ$. Ayant cette expression, le principe de l'attaque DPA est le suivant :

1. Tout d'abord, il faut retrouver la coordonnée Z du point T. Les courbes de consommation le permettent car Z intervient dans la multiplication $Z^2 x_P$ où x_P est connu car public. Dans ce cas, nous trouvons deux valeurs possibles pour Z.
2. Une fois que nous avons déterminé la coordonnée Z, nous pouvons déterminer la coordonnée X grâce à l'opération $(Z^2 x_P - X)$. Si l'analyse de la consommation de courant n'est pas assez précise pour détecter une addition, on peut déterminer Y grâce au calcul du terme $Z_3 = 2YZ$ puisque nous avons déjà calculé Z.
3. Enfin, une fois que nous avons calculé les coordonnées X et Z (ou Y et Z) nous en déduisons 2 (ou 3) valeurs possibles pour Y (ou X) à partir de l'équation de la courbe E : $Y^2 Z = X^3 + aXZ^2 + bZ^3$.

Ainsi, nous avons pu trouver un petit nombre de valeurs possibles pour les coordonnées du point T avec $T = [j]Q$. Pour retrouver Q nous devons connaître j.

Pour cela, comme le paramètre du couplage u est public, le nombre de cycles d'horloges effectués par l'algorithme de Miller avant d'appliquer l'attaque DPA nous permet de déterminer le nombre d'itérations τ de l'algorithme qui ont été effectuées. Ce nombre nous permet de trouver la valeur de j en lisant les bits de u : la représentation binaire de j est donnée par les $\tau + 1$ bits de poids faible de celle de u. Une fois que nous avons déterminé l'entier j, nous calculons l'inverse de j modulo r. Ainsi, nous trouvons le point secret Q. Comme il y avait plusieurs valeurs possibles pour les coordonnées de T, il y a plusieurs candidats pour Q. Le bon candidat est déterminé par un test final. Cette attaque et sa réalisation pratique sont bien détaillées dans [EDF09].

Remarque 5.1. *Nous remarquons que l'attaque DPA sur l'algorithme de Miller de [EDF09] concerne l'étape de doublement.*

5.3.2 Contre-mesures dans la littérature

Pour éviter les attaques par DPA sur l'algorithme de Miller classique, plusieurs contre-mesures ont été proposées dans la littérature :

1. La contre-mesure proposée par Öztürk et al. dans [ÖGS07]. Le principe de cette contre-mesure est d'éviter toute perturbation contre l'algorithme de Miller à l'aide des compteurs résilients. L'inconvénient est que c'est une contre-mesure pratique sans base théorique pour en mesurer la sécurité (pas de protection théorique et dans ce cas elle doit être implémentée).
2. Une deuxième contre-mesure, proposée par Gosh et al. dans [GMC11], consiste à implémenter l'algorithme de Miller modifié. Malheureusement, cette méthode introduit un calcul supplémentaire qui coûte cher et qui, de plus, n'améliore pas vraiment la sécurité.
3. L'utilisation de la propriété de l'homogénéité des coordonnées projectives (ou jacobiennes) du point P ou du point Q, est une contre-mesure contre les attaques par injection de fautes sur l'algorithme de Miller classique. Cette contre-mesure provoque une modification du résultat de l'algorithme de Miller : on en obtient qu'un multiple. Mais cela n'est pas un souci car ce multiple étant dans un sous corps de $\mathbb{F}^*_{p^k}$, il est éliminé par l'exponentiation finale.
   L'avantage de cette contre-mesure est qu'elle n'est pas coûteuse. Mais, malheureusement El Mrabet et al. dans [LPM+14] ont montré qu'elle n'est pas suffisante pour protéger l'algorithme de Miller.
4. Une autre contre-mesure efficace consiste à choisir deux entiers a et b tels que $a \times b = 1 \mod r$. Puis l'idée est de calculer le couplage entre les points aP et bQ au lieu de celui entre les points P et Q. Cette opération est permise, puisque grâce à la bilinéarité, nous avons :
   $$e([a]P,\ [b]Q) = e(P,\ Q)^{ab}.$$
   Ainsi, grâce à l'exponentiation finale, l'exposant ab va être annulé. Nous obtenons donc finalement :
   $$e(P,\ Q)^{ab\times(\frac{p^k-1}{r})} = e(P,\ Q)^{(\frac{p^k-1}{r})}.$$
5. Enfin, nous citons la contre-mesure dite par masquage. Cette contre-mesure consiste à choisir arbitrairement un point R de la courbe elliptique E, puis à calculer le couplage entre les points P et Q en passant par le point R. En effet, en utilisant la propriété de la bilinéarité, nous avons :
   $$e(P,\ Q) = e(P,\ Q+R) \times e(P,\ -R).$$
   Cette contre-mesure consiste alors à calculer deux couplages puis effectuer leur produit.

Les deux dernières contre-mesures sont les plus considérées en pratique. Récemment, dans une étude théorique des différentes contre-mesures existantes, El Mrabet et al. dans [MFGL15] ont suggéré de considérer plutôt la contre-mesure dite par masquage pour protéger l'algorithme de Miller. Ils ont vérifié que cette méthode de protection assure une implémentation sécurisée des couplages même si elle possède une complexité importante vu que nous calculons plutôt deux couplages au lieu d'un seul.

5.4 Résistance de notre algorithme Miller-Fibonacci

Notre nouvelle version de Miller, Miller-Fibonacci, est basée sur l'écriture du paramètre du couplage à l'aide d'une suite de Fibonacci. Dans cet algorithme il n'y a que des étapes d'addition. Nous n'avons donc que des évaluations de droites passant par les deux points T_1 et T_2 (ou T_1 et Q) de coordonnées respectives (X_1, Y_1, Z) et (X_2, Y_2, Z) évaluées au point public P. De ce fait, la seule équation impliquant des données connues qu'une attaque DPA peut utiliser est celle donnée à la fin de la section 5.2.1 :

$$l_{T_1,T_2}(P) = y_P Z^3(X_1 - X_2)^3 - Y_1(X_1 - X_2)^3 - (Y_1 - Y_2)\left(x_P Z^2(X_1 - X_2)^2 - X_1(X_1 - X_2)^2\right).$$

Pour que l'attaque DPA décrite dans [EDF09] fonctionne, il faudrait qu'une opération fasse intervenir une donnée connue et une des variables temporaires X_1, X_2, Y_1, Y_2 ou Z. Comme indiqué dans la section 5.2.1, le calcul de cette équation se fait via la variable intermédiaire $Z'_1 = Z(X_1 - X_2)$. Ainsi les données connues de l'équation (x_P et y_P) ne sont impliquées que dans les opérations $y_P {Z'_1}^3$ et $x_P {Z'_1}^2$. Un attaquant est donc capable de déterminer Z'_1 mais rien d'autre et Z'_1 ne permet pas d'obtenir des informations sur les coordonnées de T_1 et T_2. L'algorithme Miller-Fibonacci ne laisse donc pas fuir d'information sur T_1 et T_2 en tous cas dans le cadre de l'attaque DPA.

Ainsi, nous pouvons conclure que notre variante de l'algorithme de Miller basé sur la représentation de Zeckzndorf de u est une contre-mesure pour l'attaque Différentielle par Consommation de Courant (DPA).

Remarque 5.2. *De la même manière, il est simple de prouver que l'algorithme de Miller-Euclide résiste à l'attaque Différentielle par Consommation de Courant.*

5.5 Comparaison

Dans la Section 5.4, nous avons montré que notre nouvel algorithme de Miller-Fibonacci résiste aux attaques DPA. Pour vérifier l'efficacité de notre contre-mesure, nous allons la comparer avec celle par masquage qui est recommandée dans la littérature [MFGL15]. Nous choisissons de comparer ces deux contre-mesures dans le cadre du calcul du couplage Optimal Ate sur les courbes BN pour le niveau de sécurité 128 bits (avant les résultats de [KB16]).
Nous devons ainsi évaluer $f_{6u+2,Q}(P)$ à l'aide de l'algorithme de Miller où $6u + 2$ est un entier de 65 bits et de poids de Hamming 5. Nous donnons dans le tableau suivant le coût de notre contre-mesure, ainsi que de celle basée par masquage.

Contre-mesure	Naturelle	Masquage	Gain
Algorithme	Miller-Fibonacci	Miller classique	
Coût sur $\mathbb{F}_p$	$12456\,M$	$13944\,M$	**10.7%**

TABLE 5.3 – Comparaison des coûts des contre-mesures

Ce tableau montre que la contre-mesure que nous proposons est plus efficace que celle recommandée dans la littérature pour l'algorithme de Miller classique. En effet, nous sommes ainsi plus rapides d'environ **10.7**% que la contre-mesure par masquage.

Remarque 5.3. *Ce gain est toutefois biaisé par le fait que le paramètre u considéré est choisi de faible poids de Hamming pour minimiser artificiellement le coût de Miller classique. Il serait*

plus à notre avantage si on choisissait un paramètre u spécifiquement adapté à un calcul efficace de Miller-Fibonacci. Si on fait le choix d'un u avec un développement de Zeckendorf creux, disons avec 5 coefficients à 1 par exemple, Miller-Fibonacci nécessiterait moins de $11300M$ *ce qui porterait le gain à* 19%.

5.6 Conclusion

Dans ce chapitre, nous avons proposé deux nouvelles versions de l'algorithme de Miller, appelées respectivement Miller-Euclide et Miller-Fibonacci. L'un de nos algorithmes est plus efficace que l'autre en complexité.
Nous avons montré que nos deux variantes résistent contre l'attaque Différentielle par Consommation de Courant (l'attaque DPA). Nous avons prouvé aussi que notre algorithme de Miller-Fibonacci est une contre-mesure qui se trouve être la plus efficace en complexité lorsqu'on la compare avec des contre-mesures existantes proposées dans la littérature.

Conclusion générale

Les couplages sont des outils mathématiques définis sur les courbes elliptiques. C'est un sujet très en vogue depuis une quinzaine d'années en cryptographie asymétrique.
Dans cette thèse, nous avons étudié deux aspects : les problèmes de gestion des ressources et la faiblesse de la cryptographie basée sur les couplages face aux attaques. Nous avons proposé des méthodes de calcul du couplage moins gourmandes en mémoire que les méthodes existantes et nous avons protégé l'algorithme de Miller contre les attaques DPA.

Nous avons commencé par étudier le couplage Optimal Ate sur les courbes de Barreto et Naehrig pour un niveau de sécurité de 128 bits. Le problème que nous avons résolu, dans cette partie, est la mémoire utilisée. En effet, l'implémentation du couplage de Tate, ou aussi de ses dérivés, nécessite une importante mémoire. Ceci est un problème, si nous souhaitons l'implémenter dans un environnement restreint telle que la carte à puce. Dans ce contexte, nous avons essayé de réduire la mémoire utilisée en minimisant le nombre de variables temporaires nécessaires.
Comme nous l'avons montré dans le Chapitre 3, la partie gourmande en mémoire est celle de l'exponentiation finale. Notre but était de réduire cette mémoire utilisée afin de pouvoir implanter le couplage Optimal Ate, dans le cas de notre étude, dans un environnement restreint. Nous avons rappelé en premier lieu les méthodes existantes dans la littérature. Nous avons essayé d'optimiser la méthode de Devegili sous l'angle de la complexité et de la mémoire. Nous avons pu trouver un nouveau développement de la partie difficile de l'exponentiation finale. Notre nouveau développement représente un gain qui s'élève à **17**% en mémoire. Nous sommes également plus efficaces de **5**%.
Scott *et al.* dans [SBC+09] ont présenté une manière plus efficace que celle de Devegili *et al.* pour le calcul de la partie difficile de l'exponentiation finale. Nous avons également optimisé cette méthode en présentant une nouvelle chaîne d'additions. Cela a permis d'utiliser peu de variables temporaires dans $\mathbb{F}_{p^{12}}$, par conséquent nous avons gagné sur la mémoire utilisée d'environ **37**% pour une complexité comparable. Nous avons rappelé aussi la méthode de Fuentes *et al.* [CKH11], basée sur le calcul d'un exposant du couplage. Pour cette méthode, nous avons présenté trois variantes. La première consiste à garder le même multiple de l'exposant mais avec un nouveau développement. Avec cette méthode, notre gain sur la mémoire est de **15**%. Ensuite, nous avons choisi un nouveau multiple qui nous a permis de gagner **29**% sur la mémoire. Nous avons enfin présenté un développement que nous pouvons considérer comme étant la meilleure manière d'évaluer l'exponentiation finale. Nous avons en effet pu réduire le nombre de variables temporaires et aussi gagner sur la complexité. Notre gain est d'environ **15**% sur la mémoire et $\mathbf{2,5}$% sur la complexité.

Dans la deuxième partie de cette thèse, nous avons étudié le couplage Optimal Ate pour un plus haut niveau de sécurité. Nous avons commençé par étudier le calcul du couplage Optimal Ate sur

les courbes BLS12. Nous avons pu calculer ce couplage de manière plus efficace que celle présentée par Aranha *et al.* dans [AFK+12] et en gagnant sur la mémoire utilisée d'environ **28**%. Dans le même chapitre, nous avons étudié de la même manière, l'aspect complexité du couplage Optimal Ate sur les courbes BLS24 qui sont aussi recommandées en littérature pour des hauts niveaux de sécurité. Nous sommes aussi plus rapides sur le calcul de ce couplage.
L'étude du couplage Optimal Ate est importante pour plusieurs protocoles cryptographiques. Ces protocoles, dans plusieurs cas, nécessitent le calcul du produit ou aussi du quotient de n couplages pour n strictement supérieur à 1. Dans le cadre de la deuxième partie du chapitre 4, nous avons étudié ce problème sur différentes courbes elliptiques et nous avons déduit une comparaison. Cette comparaison nous a amené à déduire que pour $n = 2$, il est préférable de choisir la courbe BLS12 en se référant à nos résultats présentés dans le Chapitre 4 et les papiers [GF16b] et [GF16a]. Dans le cas où $n > 2$, nous revenons au résultat théorique annoncé par Zhang *et al.* où ils suggèrent l'utilisation de la courbe KSS16. Cependant, nous avons constaté que la courbe elliptique présentée par Zhang et Lin ne résiste pas aux attaques par les sous-groupes. Nous avons donc proposé un nouveau paramètre qui rend la courbe elliptique KSS16 résistante contre les attaques par les sous-groupes.

Dans le Chapitre 5 nous avons étudié les attaques sur le calcul des couplages. Dans la littérature, malheureusement, plusieurs attaques ont été appliquées avec succès sur l'algorithme de Miller. Parmi ces attaques, nous citons les attaques par injection des fautes, les attaques par canaux cachés etc ....
Dans ce chapitre, nous avons proposé une contre-mesure pour l'attaque DPA. Nous avons montré que l'écriture de l'algorithme de Miller à l'aide d'une chaîne d'addition euclidienne, ou plus précisément, à l'aide d'une suite de Fibonacci est une contre-mesure plus efficace que celles proposées dans la littérature.

Nos travaux ouvrent plusieurs pistes de recherche. Nous pouvons citer par exemple la recherche de courbes elliptiques adaptées aux couplages assurant un niveau de sécurité 192 bits ou étudier la résistance de nos algorithmes Miller-Fibonacci et Miller-Euclide face aux attaques par injection de fautes et leur réalisation pratique.

Cette thèse a donné lieu à la rédaction de 5 articles dont 2 publiés et 3 soumis. Un autre est en cours de préparation.
Le Chapitre 3 présente trois articles. Le premier article [DG16] concerne la présentation de nos variantes pour le calcul de la partie difficile de l'exponentiation finale. Nous avons publié cet article dans le journal De Gruyter *Groups, Complexity, Cryptograpgy*. Le deuxième article concerne l'optimisation du calcul de l'exponentiation finale en tenant compte des propriétés du paramètre de la courbe elliptique u. Le troisième article du chapitre concerne la partie implémentation matérielle du couplage Optimal Ate. Cette soumission est présentée dans [SGM+15] et a été faite en collaboration avec l'équipe d'Électronique et de Microélectronique de la Faculté des Sciences de Monastir (Tunisie).
Le Chapitre 4 représente deux articles. Le premier concerne le calcul du couplage Optimal Ate pour les hauts niveaux de sécurité et soumis et présenté dans [GF16b]. Le deuxième papier concerne le travail sur le calcul du produit de n couplages et est publié dans le Workshop *WAIFI* [GF16a]. Ces deux travaux ont été fait en collaboration avec Emmanuel Fouotsa.
Le Chapitre 5 a donné lieu à la préparation d'un article en cours. Il concerne la résistance de

l'algorithme de Miller contre les attaques par injection des fautes et les attaques DPA. Ce papier est en collaboration avec Nicolas Méloni et Nadia El Mrabet.

Bibliographie

[ACD+06] Michel Abdalla, Dario Catalano, Alexander W. Dent, John Malone-Lee, Gregory Neven, and Nigel P. Smart. Identity-based encryption gone wild. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 300–311, 2006.

[adv05] Advances in elliptic curve cryptography. 2005.

[AFK+12] Diego F. Aranha, Laura Fuentes-Castañeda, Edward Knapp, Alfred Menezes, and Francisco Rodríguez-Henríquez. Implementing pairings at the 192-bit security level. In *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, pages 177–195, 2012.

[AKL+11a] Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio López. Faster explicit formulas for computing pairings over ordinary curves. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 48–68, 2011.

[AKL+11b] Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio López. Faster explicit formulas for computing pairings over ordinary curves. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 48–68, 2011.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 440–456, 2005.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 41–55, 2004.

[BCM+15] Paulo S. L. M. Barreto, Craig Costello, Rafael Misoczki, Michael Naehrig, Geovandro C. C. F. Pereira, and Gustavo Zanon. Subgroup security in pairing-based cryptography. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 245–265, 2015.

[BF03] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3) :586–615, 2003.

[BGM+10] Jean-Luc Beuchat, Jorge Enrique González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-speed software implementation of the optimal ate pairing over barreto-naehrig curves. In *Pairing-Based Cryptography - Pairing 2010*, pages 21–39, 2010.

[BK98] R. Balasubramanian and Neal Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the menezes - okamoto - vanstone algorithm. *J. Cryptology*, 11(2) :141–145, 1998.

[BLS02] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. Constructing elliptic curves with prescribed embedding degrees. In *Security in Communication Networks, Third International Conference, SCN 2002, Amalfi, Italy, September 11-13, 2002. Revised Papers*, pages 257–267, 2002.

[BLS03] Paulo S. L. M. Barreto, Ben Lynn, and Michael Scott. On the selection of pairing-friendly groups. In *Selected Areas in Cryptography, 10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003, Revised Papers*, pages 17–25, 2003.

[BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. *J. Cryptology*, 17(4) :297–319, 2004.

[BLS11] Daniel J. Bernstein, Tanja Lange, and Peter Schwabe. On the correct use of the negation map in the pollard rho method. In *Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6-9, 2011. Proceedings*, pages 128–146, 2011.

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography - SAC 2005*, pages 319–331, 2005.

[BSS05] Ian F. Blake, Gadiel Seroussi, and Nigel Paul Smart, editors. *Advances in elliptic curve cryptography*. London Mathematical Society lecture note series. Cambridge University Press, New York, 2005.

[BW05] Friederike Brezing and Annegret Weng. Elliptic curves suitable for pairing based cryptography. *Des. Codes Cryptography*, 37(1) :133–141, 2005.

[CCS06] Liqun Chen, Zhaohui Cheng, and Nigel P. Smart. A built-in decisional function and security proof of id-based key agreement protocols from pairings. *IACR Cryptology ePrint Archive*, 2006 :160, 2006.

[CDF+11] Ray C. C. Cheung, Sylvain Duquesne, Junfeng Fan, Nicolas Guillermin, Ingrid Verbauwhede, and Gavin Xiaoxu Yao. FPGA implementation of pairings using residue number system and lazy reduction. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 421–441, 2011.

[CFA+05] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren, editors. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. Chapman and Hall/CRC, 2005.

[CFA+06] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren, editors. *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete Mathematics and its Applications. Chapman & Hall/CRC, Boca Raton, FL, 2006.

[CH07] Jaewook Chung and M. Anwar Hasan. Asymmetric squaring formulae. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18 2007), 25-27 June 2007, Montpellier, France*, pages 113–122, 2007.

[CKH11] Laura Fuentes Castaneda, Edward Knapp, and Francisco Rodrguez Henrquez. Faster hashing to g2. In *Selected Areas in Cryptography - 18th International Workshop, 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, pages 412–430, 2011.

[CLN10] Craig Costello, Tanja Lange, and Michael Naehrig. Faster pairing computations on curves with high-degree twists. In *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, pages 224–242, 2010.

[CP01] C. Cocks and R.G.E Pinch. Identity-based cryptosystems based on the weil pairing. 2001.

[DG] S. Duquesne and L. Ghammam. https ://cloud.sagemath.com/projects/332de229-174f-4d90-ae79-ca9d3b0fc1f7/files/Algorithms.sagews.

[DG16] Sylvain Duquesne and Loubna Ghammam. Memory-saving computation of the pairing final exponentiation on BN curves. *Groups Complexity Cryptology*, 8(1) :75–90, 2016.

[DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6) :644–654, 1976.

[DMHR15] Sylvain Duquesne, Nadia El Mrabet, Safia Haloui, and Franck Rondepierre. Choosing and generating parameters for low level pairing implementation on BN curves. *IACR Cryptology ePrint Archive*, 2015 :1212, 2015.

[DSD07] Augusto Jun Devegili, Michael Scott, and Ricardo Dahab. Implementing cryptographic pairings over barreto-naehrig curves. In *Pairing-Based Cryptography - Pairing 2007*, pages 197–207, 2007.

[EDF09] N. El Mrabet, G. Di Natale, and M.L. Flottes. A practical differential power analysis attack against the miller algorithm. In *PRIME 2009 - 5th Conference on Ph.D. Research in Microelectronics and Electronics, Circuits and Systems Magazine*, IEEE Xplore, 2009.

[EM14] Andreas Enge and Jérôme Milan. Implementing cryptographic pairings at standard security levels. *CoRR*, abs/1407.5953, 2014.

[FG] Emmanuel Fouotsa and Loubna Ghammam. http ://www.camercrypt.org/KSS16-finalexponentiation.

[FMR99a] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Information Theory*, 45(5) :1717–1719, 1999.

[FMR99b] Gerhard Frey, Michael Müller, and Hans-Georg Rück. The tate pairing and the discrete logarithm applied to elliptic curve cryptosystems. *IEEE Trans. Information Theory*, 45(5) :1717–1719, 1999.

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A taxonomy of pairing-friendly elliptic curves. *J. Cryptology*, 23(2) :224–280, 2010.

[FVV12] Junfeng Fan, Frederik Vercauteren, and Ingrid Verbauwhede. Efficient hardware implementation of fp-arithmetic for pairing-friendly curves. *IEEE Trans. Computers*, 61(5) :676–685, 2012.

[GF] L. Ghammam and E. Fouotsa. http ://www.camercrypt.org/BLSformulas.

[GF16a] Loubna Ghammam and Emmanuel Fouotsa. Adequate elliptic curve for computing the product of n pairings. *IACR Cryptology ePrint Archive*, 2016 :472, 2016.

[GF16b] Loubna Ghammam and Emmanuel Fouotsa. On the computation of the optimal ate pairing at the 192-bit security level. *IACR Cryptology ePrint Archive*, 2016 :130, 2016.

[GJNB11] C. C. F. Pereira Geovandro, Marcos A. Simplício Jr., Michael Naehrig, and Paulo S. L. M. Barreto. A family of implementation-friendly BN elliptic curves. *Journal of Systems and Software*, 84(8) :1319–1326, 2011.

[GMC10] Santosh Ghosh, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. High speed flexible pairing cryptoprocessor on FPGA platform. In *Pairing-Based Cryptography - Pairing 2010 - 4th International Conference, Yamanaka Hot Spring, Japan, December 2010. Proceedings*, pages 450–466, 2010.

[GMC11] Santosh Ghosh, Debdeep Mukhopadhyay, and Dipanwita Roy Chowdhury. Fault attack, countermeasures on pairing based cryptography. *I. J. Network Security*, 12(1) :21–28, 2011.

[GS06] Robert Granger and Nigel P. Smart. On computing products of pairings. *IACR Cryptology ePrint Archive*, 2006 :172, 2006.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pages 415–432, 2008.

[GS10] Robert Granger and Michael Scott. Faster squaring in the cyclotomic subgroup of sixth degree extensions. In *Public Key Cryptography - PKC 2010, 13th International Conference on Practice and Theory in Public Key Cryptography, Paris, France, May 26-28, 2010. Proceedings*, pages 209–223, 2010.

[GVC12] Santosh Ghosh, Ingrid Verbauwhede, and Dipanwita Roy Chowdhury. Core based architecture to speed up optimal ate pairing on FPGA platform. In *Pairing-Based Cryptography - Pairing 2012 - 5th International Conference, Cologne, Germany, May 16-18, 2012, Revised Selected Papers*, pages 141–159, 2012.

[Har60] P. Harris. Probability distributions related to random mappings. *In Annals of Math. Statistics,*, 31 :1045–1062, 1960.

[HSV06a] Florian Hess, Nigel P. Smart, and Frederik Vercauteren. The eta pairing revisited. *IEEE Trans. Information Theory*, 52(10) :4595–4602, 2006.

[HSV06b] F. Hesse, N.P. Smart, and F. Vercauteren. The eta pairing revisited. *IEEE Transactions on Information Theory*, 52(10) :4595–4602, 2006.

[JL07] A. Joux and R. Lercier. Algorithmes pour résoudre le problème de logarithme discret dans les corps finis. *In Nouvelles Méthodes Mathématiques en Cryptographie*, Fascicule Journées Annuelles :23–53, Société Mathématiques en Cryptographie, 2007.

[Jou00] Antoine Joux. A one round protocol for tripartite diffie-hellman. In *Algorithmic Number Theory, 4th International Symposium, ANTS-IV, Leiden, The Netherlands, July 2-7, 2000, Proceedings*, pages 385–394, 2000.

[Kar13] Koray Karabina. Squaring in cyclotomic subgroups. *Math. Comput.*, 82(281), 2013.

[KB16] Taechan Kim and Razvan Barbulescu. Extended tower number field sieve : A new complexity for the medium prime case. In *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 543–571, 2016.

[KM05] Neal Koblitz and Alfred Menezes. Pairing-based cryptography at high security levels. In *Cryptography and Coding, 10th IMA International Conference, Cirencester, UK, December 19-21, 2005, Proceedings*, pages 13–36, 2005.

[KSS07] Ezekiel J. Kachisa, Edward F. Schaefer, and Michael Scott. Constructing brezing-weng pairing friendly elliptic curves using elements in the cyclotomic field. *IACR Cryptology ePrint Archive*, 2007 :452, 2007.

[KY75] D. Knuth and A. Yao. Analysis of the subtractive algorithm for greater common divisors. pages 4720–4722, December 1975.

[LLL82] A.K. Lenstra, H.W.jun. Lenstra, and László Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261 :515–534, 1982.

[LN94] R. Lidl and H. Niederreiter. Finite fields. 1994.

[LPM+14] Ronan Lashermes, Marie Paindavoine, Nadia El Mrabet, Jacques J. A. Fournier, and Louis Goubin. Practical validation of several fault attacks against the miller algorithm. In *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography, FDTC 2014, Busan, South Korea, September 23, 2014*, pages 115–122, 2014.

[Mel07a] Nicolas Meloni. *Arithmétique des couplages, performance et résistance aux attaques par cannaux cachés.* 2007.

[Mel07b] Nicolas Meloni. New point addition formulae for ECC applications. In *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007, Proceedings*, pages 189–201, 2007.

[Mel07c] Nicolas Meloni. New point addition formulae for ECC applications. In *Arithmetic of Finite Fields, First International Workshop, WAIFI 2007, Madrid, Spain, June 21-22, 2007, Proceedings*, pages 189–201, 2007.

[MFGL15] Nadia El Mrabet, Jacques J. A. Fournier, Louis Goubin, and Ronan Lashermes. A survey of fault attacks in pairing based cryptography. *Cryptography and Communications*, 7(1) :185–205, 2015.

[Mil85] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology - CRYPTO '85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, pages 417–426, 1985.

[Mil04] Victor S. Miller. The weil pairing, and its efficient calculation. *J. Cryptology*, 17(4) :235–261, 2004.

[MKHO07] Seiichi Matsuda, Naoki Kanayama, Florian Hess, and Eiji Okamoto. Optimised versions of the ate and twisted ate pairings. In *Cryptography and Coding, 11th IMA International Conference, Cirencester, UK, December 18-20, 2007, Proceedings*, pages 302–312, 2007.

[MNT00] Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of elliptic curve traces under fr-reduction. In *Information Security and Cryptology - ICISC 2000, Third International Conference, Seoul, Korea, December 8-9, 2000, Proceedings*, pages 90–108, 2000.

[MOV93] Alfred Menezes, Tatsuaki Okamoto, and Scott A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Trans. Information Theory*, 39(5) :1639–1646, 1993.

[NAS+08] Yasuyuki Nogami, Masataka Akane, Yumi Sakemi, Hidehiro Katou, and Yoshitaka Morikawa. Integer variable chi-based ate pairing. In *Pairing-Based Cryptography - Pairing 2008*, pages 178–191, 2008.

[ÖGS07] Erdinç Öztürk, Gunnar Gaubatz, and Berk Sunar. Tate pairing with strong fault resiliency. In *Fourth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2007, FDTC 2007 : Vienna, Austria, 10 September 2007*, pages 103–111, 2007.

[Oli81] Jorge Olivos. On vectorial addition chains. *J. Algorithms*, 2(1) :13–21, 1981.

[oST] National Institute of Standards and Technology. http ://csrc.nist.gov/publications/PubsSPs.html.

[Pol78] J. M. Pollard. Monte carlo method for index computations (mod p). 1978.

[R.S95] R.Schoof. Counting points on elliptic curves over finite fields. *Journal de Theorie des nombres, Bordeaux*, 1995.

[S+] W. A. Stein et al. *Sage Mathematics Software (Version SageMathCloud)*. The Sage Development Team. `https://cloud.sagemath.com/`, year = 2015,.

[SBC+09] Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J. Dominguez Perez, and Ezekiel J. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves. In *Pairing-Based Cryptography - Pairing 2009, Third International Conference, Palo Alto, CA, USA, August 12-14, 2009, Proceedings*, pages 78–88, 2009.

[Sco05] Michael Scott. Computing the tate pairing. In *Topics in Cryptology - CT-RSA 2005, The Cryptographers' Track at the RSA Conference 2005, San Francisco, CA, USA, February 14-18, 2005, Proceedings*, pages 293–304, 2005.

[SGM+15] Anissa Sghaier, Loubna Ghammam, Zeghid Medien, Sylvain Duquesne, and Mohsen Machhout. Area-efficient hardware implementation of the optimal ate pairing over BN curves. *IACR Cryptology ePrint Archive*, 2015 :1100, 2015.

[Sil86] Joseph H. Silverman. *The arithmetic of elliptic curves*. Graduate texts in mathematics. Springer, New York, Berlin, 1986. 2e tirage corrigé 1992.

[UW14] Thomas Unterluggauer and Erich Wenger. Efficient pairings and ECC for embedded systems. In *Cryptographic Hardware and Embedded Systems - CHES 2014*, pages 298–315, 2014.

[Ver10] Frederik Vercauteren. Optimal pairings. *IEEE Transactions on Information Theory*, 56(1) :455–461, 2010.

[Was03] L. Washington. Elliptic curves, number theory and cryptography. 2003.

[Was08] L.C. Washington. *Elliptic Curves, Number Theory and Cryptography*. Discrete Math .Aplli, Chapman and Hall, 2008.

[Wat05] Brent Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 114–127, 2005.

[Wei48] A. Weil. *Courbes aalgébriques et variétés abéliennes (in french).* 1948.

[Zec72] E. Zeckendof, editor. *Représentations des nombres naturels par une somme de nombre de Fibonacci ou de nombres de Lucas.* Bulletin de la Société Royale des Sciences de Liège. 1972.

[ZL12] Xusheng Zhang and Dongdai Lin. Analysis of optimum pairing products at high security levels. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 412–430, 2012.

Résumé

Les couplages sont des outils mathématiques introduits par André Weil en 1948. Ils sont un sujet très en vogue depuis une quinzaine d'années en cryptographie asymétrique. Ils permettent en effet de réaliser des opérations cryptographiques impossible à réaliser simplement autrement.
Ces dernières années, le calcul des couplages est devenu plus facile grâce à l'introduction de nouvelles méthodes des calculs mathématiques particulièrement efficaces sur les courbes elliptiques dites les courbes bien adaptées aux couplages.
Aujourd'hui, nous sommes au stade de transfert de cette technologie, de la théorie vers la mise en œuvre pratique, sur des composants électroniques. Ce transfert soulève de nombreuses problématiques qui s'avèrent difficile à surmonter à cause de la différence de culture scientifique entre mathématiciens et micro-électroniciens.

Dans le présent document, en premier lieu, nous avons étudié le problème de l'implémentation du couplage dans des environnements à mémoire limitée. Dans ce contexte, en faisant des optimisations mathématiques, nous avons pu implémenté ces couplages dans des environnements retreints, nous avons gagné environ 37% en mémoire.
Le deuxième problème que nous avons traité est la sécurité des protocoles cryptographiques basés sur les couplages. Dans ce contexte, nous avons étudié les attaques sur les couplages et nous avons proposé ue contre-mesure plus efficace que celle présentée dans la littérature.

Abstract

The pairings on elliptic curves were first introduced by André Weil in 1948. They are used in asymmetric cryptography. They have been very popular over the last decades. The pairings on elliptic curves allow to perform some cryptographic operations which are not easily performed without pairings like short signature and the identity based cryptography.
Recently, the calculation of pairings has become easier than before thanks to the introduction of new mathematical optimizations on elliptic curves which are called the pairing-friendly elliptic curves. Nowadays, it is important to transferthis technology from theory to practical implementation which is executed on some electronic components as FPGA or SIM cards.

In the first part of this thesis, we studied the problem of pairing implementation in restricted environment. Indeed, the computation of the Tate pairing, or also one of its variants, requires the implementation of a many temporary variables. Therefore, it requires a lot of memory of the electronic component on which we wish to implement such a pairing. In this context, by doing mathematical optimizations, we have saved 37% of the memory in implementing these pairings in restricted environments.
The second problem is the security of the cryptographic protocols based on pairings. Since pairings on elliptic curves are supposed to be physically attacked, we studied these attacks and we proposed a counter measure.