UNIVERSITÉ MONTPELLIER II

SCIENCES ET TECHNIQUES DU LANGUEDOC

THÈSE

pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ MONTPELLIER II

Discipline : **Informatique**

École Doctorale : **Information, Structures et Systèmes**

présentée et soutenue publiquement par

Nadia EL MRABET

le 7 décembre 2009

Arithmétique des couplages, performance et résistance aux attaques par canaux cachés

JURY

Jean Claude Bajard	Directeur de thèse
Sylvain Duquesne	CoDirecteur de thèse
Andreas Enge	Rapporteur
Viktor Fisher	Rapporteur
Bruno Martin	Examinateur
Lionel Torres	Examinateur

Table des matières

Remerciements

Durant les trois dernières années j'ai beaucoup appris sur moi même, sur le métier de chercheur et sur la nature humaine. J'ai l'occasion ici d'exprimer ma gratitude envers chacune des personnes qui de près ou de loin ont contribué à faire de cette thèse un succès.

En premier lieu, je souhaite exprimer ma reconnaissance envers Jean Claude Bajard et Sylvain Duquesne qui m'ont permis d'effectuer mon doctorat au sein du LIRMM et de l'I3M. Ce fut un réel plaisir de travailler sous la direction de personnes aussi patientes et attentionnées. Ce travail doit beaucoup à la clarté de leurs explications et à leurs disponibilités parfaitement complémentaires. Je tiens également à les remercier pour m'avoir apporter toute leur aide et tout leur soutien, surtout aux moments les plus difficiles.

Je souhaite ici remercier Andreas Enge et Viktor Fischer pour avoir accepté d'être les rapporteurs de ce manuscrit ; leurs conseils m'ont été très précieux pour améliorer mon mémoire. Enfin, je suis très reconnaissante à Bruno Martin et Lionel Torres pour avoir accepté de faire partie de mon jury, et ainsi d'évaluer mes travaux de thèse.

Ces travaux de thèse n'auraient pas été possibles sans l'aide de mes coauteurs ; Christophe Nègre et sa gentillesse, Sorina Ionica et sa disponibilité permanente, Nicolas Guillermin et son efficacité, Marie Lise Flottes et Giorgio Di Natale pour leur aide en attaque par consommation de courant. Merci à Philippe Ravache, ainsi qu'à Michael Tunstall, Victor Lomne et Alexandre Venelli pour m'aider à construire mes travaux futurs. Je souhaite remercier vivement Jean Luc Beuchat pour ses relectures minutieuses, et ses indications très pertinentes concernant mes articles ou mon mémoire ; ainsi que Jérémie Detrey. Leur aide m'a beaucoup apporté pour cette thèse.

J'ai effectué celle-ci en étant rattaché à deux équipes de recherche l'équipe Arith au LIRMM et l'équipe TNM à l'I3M. Je tiens à remercier chacun des membres de ces équipes. En particulier, Louise Nyssens pour les soirées autour de ces délicieuses tartes salées ou sucrées ; Bill Allombert pour son

aide immédiate concernant mes problèmes de programmation ; Christophe Fiorio pour sa bonne humeur constante, Fabrice pour son humour décapant ou encore Thierry Monteil pour ses exposés si vivants et passionnants.

J'ai une pensée toute particulière pour Valérie Berthé, surement la meilleure cheffe d'équipe qu'il m'ait été donnée de rencontrer. Sa patience et sa douceur ont su apporter un souffle de tranquillité au sein de l'équipe Arith.

Je tiens à remercier Florence Raffray, Michel Robert, Evelyne Sanders et Cécile Lukasik pour leur aide, leur soutien et surtout leur très grande diligence ; ainsi que Damien et Christophe pour avoir su répondre présent.

Merci aux Elizabeth et à tous les membres du service administratif pour leur gentillesse et leur aptitude à régler rapidement le moindre souci pratique. Merci au GRM IM pour m'avoir permis de participer au programme d'échange de doctorants et en particulier à Brigitte Vallée et Claude Carlet. Sans oublier pour les grands moments d'émotion, les trois mousquetaires, Laurel et Hardy ou encore PO (en hommage à mes années taupines).

Je n'oublie pas mes collègues enseignants qui ont accepté de me confier leurs étudiants, Annie Chateau, Stefano Cerri, Hervé Dicky, Marianne Huchard, Séverine Bérard, Thérèse Libourel, Michel Leclère, Christian Mercat et Pascal Giorgi. Ni mes nouveaux collègues caennais, qui ont eu la patience d'assister à une soutenance avant l'heure, merci à Ali, Ayoub, Brigitte, Fabien, Jean Marie, Julien, Loïc, Nicolas et Marya pour leur accueil chaleureux au pays du temps normand où il peut faire beau et pleuvoir en même temps.

Je ne pourrai surement pas remercier toutes les personnes qui ont croisé ma route d'apprenti chercheuse, mais qu'elles sachent toutes que leur soutien me va droit au coeur. Parmi ces personnes, je pense tout particulièrement à Romain, Annie, Xavier, Sorina, Jean Luc, Christophe pour leur relecture patiente de mon manuscrit et mes parents pour m'avoir supporté avec sérénité à ces moments là, et aux autres aussi.

Je tiens malgré tout à accorder ici une place toute particulière à mes camarades de promotion, thésards ou post doc de divers horizons que j'ai rencontré au cours de ces trois années. Par ordre alphabétique (et j'espère sans oubli), merci à Afaf et son courage de battante, Amine pour ses pas de danse, Annie et nos longues discussions (et le sport avec Valérie !), Benoit (M) le gendre idéal, Benoit (V) et sa tranquillité, Bruno pour sa capouera, Christian ou Spider Man en thèse, Chady le plus Rady de mes amis, Chloé et sa force de caractère, Clément et ses livres de poches, Frédéric et cette partie de billard jamais commencée, Guillaume et HIMYM ou BSG ou encore BBT, Gwladys la reine des anniversaires réussis, Ioan fan de la Corée du Sud, Julien et sa zen attitude, Junior et sa patience, Hilde et sa glamour

attitude, Linda la fervente fan des lamas, Lisa la spécialiste des motos, Michael et sa disponibilité, Olivier Ze cobureau, Petrutza pour son sourire de tous les instants, Rémi et sa simplicité, Radouane le plus grand adepte de l'énergie noire, Raluca et sa joie de vivre, Romain, parce que c'était lui, parce c'était moi, Sean pour sa nonchalance, Star et ses paillettes, Thomas Cro Magnon et son franc parler, Thomas (I) et son aide très sage, Valéry et son ingénuité, Véronique pour ses patisseries, Xavier pour ses hamburgers, et tant d'autres amis (Coralie, Eve, Imad, Julie,...). Beaucoup de personnes et encore plus d'excellents moments !!

Pour terminer, je tiens à remercier les personnes dont la confiance et le soutien indéfectible m'ont permis d'arriver jusque là : mes parents, mes frères, ma famille ; sans qui ce bonheur aurait moins d'éclat.

Introduction

Les couplages sont des outils mathématiques introduits par André Weil en 1948 [Wei48]. Ils sont utilisés depuis peu en cryptographie. Nous étudions dans ce manuscrit leur arithmétique, ainsi que les faiblesses des protocoles à base de cet outil lorsqu'ils sont soumis à des attaques par canaux cachés.

D'après un dictionnaire usuel la cryptographie est : « l'ensemble des principes, méthodes et techniques dont l'application assure le chiffrement et le déchiffrement des données afin d'en préserver la confidentialité et l'authenticité ». Autrement dit, tout moyen permettant de communiquer avec quelqu'un sans que des personnes étrangères (pouvant être des espions, ou tout simplement des curieux) ne puissent récupérer et surtout comprendre les messages. Un protocole cryptographique se décompose en deux parties différentes : le chiffrement et le déchiffrement du message. Historiquement, la cryptographie était surtout utilisée à des fins militaires. Le plus ancien protocole connu est celui des esclaves du roi de Babylone, Nabuchodonosor. Il était utilisé pour communiquer avec ses généraux. Il rasait un esclave, et lui faisait tatouer sur le crâne le message à transmettre. Ensuite, il attendait que les cheveux repoussent avant d'envoyer l'esclave à ses généraux. Cette méthode était non seulement très lente, mais aussi très facile à décrypter. En effet, il suffisait d'intercepter l'esclave et de lui raser la tête pour déchiffrer le message. De plus, le messager ne pouvait « servir » qu'une fois. Un des autres inconvénients de ce protocole est que la méthode de chiffrement est elle même secrète.

De nos jours, une grande partie des cryptosystèmes est basée sur des problèmes mathématiques difficiles à résoudre. Le principe d'un protocole cryptographique est qu'il doit être simple de chiffrer un message, mais qu'il doit être très difficile voire impossible à déchiffrer sans posséder la clé du système.

Selon le principe de Kerckhoffs : « **La sécurité d'un protocole cryptographique doit être garantie par les clés de chiffrement et déchiffrement utilisées, et non par la méthode de chiffrement.** »

Ce principe a donné naissance à la cryptographie moderne. Celle-ci se caractérise par le fait que les algorithmes de chiffrement et méthodes de déchiffrement sont connus de tous, mais les clés utilisées sont partiellement,

ou entièrement secrètes.

Les protocoles cryptographiques se divisent principalement en deux catégories. Les systèmes symétriques pour lesquels la clé servant au chiffrement et au déchiffrement est secrète ; et les protocoles asymétriques pour lesquels la clé de chiffrement est publique, et la clé de déchiffrement privée.

En cryptographie asymétrique, la clé publique permet de chiffrer un message, elle est connue de tous. Le déchiffrement d'un message ne peut se faire sans la clé secrète associée à la clé publique. Le couple (clé privée, clé publique) est construit de sorte qu'un protocole donné soit difficile à casser, c'est à dire retrouver la clé secrète à partir de la clé publique.

La sécurité des protocoles asymétriques repose sur des problèmes considérés difficiles. La sécurité de nombreux protocoles actuels repose sur le problème du logarithme discret [Was03, Chap. 5]. L'un des premiers protocoles asymétriques est l'échange de clé de Diffie Hellman [DH76], sa sécurité repose sur la difficulté de résoudre le problème du logarithme discret.

De nombreux protocoles asymétriques sont utilisés actuellement. Le plus célèbre est sûrement le protocole RSA, créé par Rivest, Shamir et Adelmann en 1978 [RSA78]. Le couple (clé privée, clé publique) du protocole RSA est construit à partir des propriétés des anneaux de la forme $\mathbb{Z}/n\mathbb{Z}$. La sécurité de RSA repose sur la factorisation d'entiers. Le record de factorisation en date de mai 2007, est la factorisation d'un nombre de 1020 bits après onze mois de calculs [1]. Ils ont utilisés un nombre premier de forme particulière. Les clés utilisées aujourd'hui sont des nombres de taille 1024 bits. La taille des clés de RSA doit donc croître au fur et à mesure que les capacités de calcul des ordinateurs augmentent, et que des progrès sont faits pour le calcul de la factorisation des grands entiers. La cryptographie à base de courbes elliptiques pourrait être une alternative à RSA. Pour un niveau de sécurité égal, les tailles de clés pour la cryptographie à base de courbes elliptiques sont plus petites que pour RSA. Par exemple, la sécurité RSA de 1 024 bits est atteinte avec des clés de taille 160 bits sur les courbes elliptiques.

Le développement de la cryptographie basée sur les courbes elliptiques fut initié par V. Miller[Mil86b] et N. Koblitz [Kob87]. A l'heure actuelle, de nombreux protocoles à base de courbes elliptiques sont utilisés [X9.99, Cra92, DH76]. Les protocoles à base de couplage sont encore en développement.

Un couplage est une application bilinéaire $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, avec $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_3$ des groupes d'ordre premier r. Lorsqu'il est utilisé en cryptographie, il envoie (P, Q) un couple de points d'une courbe elliptique dans le sous groupe multiplicatif des racines r-ième de l'unité du corps $\mathbb{F}_{p^k}$ à p^k éléments, où k est le degré de plongement de la courbe. L'entier r est le plus petit entier

[1] `http://actualites.epfl.ch/presseinfo-com?id=439`

tel que toute les racines r-ième de l'unité de $\mathbb{F}_p$ soient incluses dans $\mathbb{F}_{p^k}$, l'extension de degré k de $\mathbb{F}_p$. Leur première utilisation fut destructrice. La propriété de bilinéarité des couplages permet de transférer le problème du logarithme discret sur une courbe elliptique en un problème de logarithme discret sur un corps fini. En 1993, l'attaque MOV [MOV91] du nom de ses auteurs Menezes, Okamoto et Vanstone utilise le couplage de Weil pour faire ce transfert, et en 1994 l'attaque de Frey et Rück [FR94] utilise le couplage de Tate. La construction des algorithmes de couplage repose sur l'algorithme de Miller, datant de 1986 [Mil86a, Mil04], et dont les performances dépendent de l'arithmétique utilisée dans le corps de base et dans ses extensions.

En 2000, Joux [Jou00] proposa un schéma d'échange de clé tri-partite à la Diffie Hellman utilisant la propriété de bilinéarité des couplages. Ce fut le premier d'une longue série d'articles proposant des protocoles cryptographiques asymétriques basés sur les couplages. L'une des utilisation les plus intéressantes des couplages pourrait être la cryptographie basée sur l'identité. En 1984, A. Shamir avait lancé un défi à la communauté des cryptographes, ce défi consistait a créer un protocole cryptographique basé sur l'identité [Sha84]. Le principe devait être que la clé publique de l'utilisateur soit son identité, et la clé privée devait être construite de manière uniforme pour tous les utilisateurs. Seize ans plus tard, en 2001, Boneh et Franklin ont proposé le premier protocole basé sur l'identité, construit autour de la propriété de bilinéarité des couplages [BF01]. Ce fut le premier schéma cryptographique basé sur l'identité, et sa construction fut rendue possible par les propriétés des couplages. Depuis, de nombreux articles proposent des protocoles utilisant les couplages [BLS01, BS04], d'autres améliorent les calculs des couplages [Sco05, KM05, EN09, EGI09] et certains proposent des réalisations matérielles pour calculer les couplages [AHM07, GPS05, BBS+07, BDE+09].

Cette thèse comporte cinq chapitres. Le premier se subdivise en deux parties. La première est une introduction à la théorie des courbes elliptiques, nous donnons les éléments théoriques dont nous aurons besoin pour définir les couplages. Dans la seconde partie nous introduisons la notion de couplage. Ce Chapitre I nous permettra d'introduire la cryptographie à base de couplage, et la notion de sécurité relative à un protocole cryptographique. Nous donnons les propriétés des couplages de Weil et Tate et nous présentons l'algorithme de Miller permettant de les calculer. L'algorithme de Miller est construit sur le principe de l'algorithme d'exponentiation rapide par doublement et addition. Nous conclurons ce chapitre d'introduction par la présentation des derniers nés de la famille des couplages, les couplages Ate et Twisted Ate. Les chapitres suivants sont issus des travaux de recherche effectués durant ces trois années de thèse.

Les problèmes d'implémentation vont de pair avec le développement des

protocoles utilisant les couplages. En effet, un algorithme innovant théoriquement, se doit d'être aussi facile et efficace à calculer. Le coût de calcul d'un couplage dépend de nombreux paramètres, les articles [BKLS02, BLS04a] ou encore les livres [Gal05, CF06] présentent les tenants et aboutissants de la recherche d'implémentations efficaces. Dans le Chapitre II, nous nous concentrons sur la complexité de l'implémentation telle qu'elle est présentée par les articles [KM05] et [GPS06]. Le premier repose sur l'utilisation de « corps adaptés au calcul des couplages » pour des courbes elliptiques dont le degré de plongement k est de la forme 2^i3^j. Quant au second, il utilise une représentation à l'aide de polynômes cyclotomiques pour représenter le corps fini. Ces deux articles comparent l'efficacité du couplage de Weil à celle du couplage de Tate. A priori, le couplage de Weil est plus coûteux en terme d'opérations élémentaires sur les corps que le couplage de Tate. En 2005 dans [KM05], Koblitz et Menezes montrent dans leur article que ce n'est plus le cas lorsque le niveau de sécurité requis devient important. En 2006 dans [GPS06], Granger, Page et Smart ont montré que le couplage de Tate reste toujours plus performant quel que soit le niveau de sécurité.
Leurs conclusions étant contradictoires, nous avons étudié les arithmétiques utilisées afin de réaliser une comparaison la plus objective possible entre les deux approches. Ces travaux nous permettront d'introduire les améliorations les plus récentes pour l'exécution de l'algorithme de Miller.

La cryptographie a base de couplage nécessite une arithmétique efficace dans les corps de la forme $\mathbb{F}_{p^k}$. Les recommandations de sécurité fixent la taille de p et de k. Pour correspondre aux niveaux de sécurité actuels, p est un entier de taille minimalle 160 bits, et k varie entre 6 et 32. Nous étudions dans le Chapitre III l'utilisation d'une représentation originale des éléments du corps $\mathbb{F}_p$ pour améliorer les calculs dans l'extension $\mathbb{F}_{p^k}$ [EN09]. Pour des valeurs de k s'écrivant 2^i3^j, la multiplication sur $\mathbb{F}_{p^k}$ est optimisée par l'utilisation des multiplications de Karatsuba et Toom Cook. Ces multiplications sont taillées sur mesure pour des extensions quadratiques ou cubiques. Néanmoins, les valeurs de k de cette forme ne sont pas très nombreuses, et il serait intéressant d'avoir une multiplication efficace pour d'autre formes de degré de plongement. Nous avons choisi d'utiliser une représentation du corps fini à l'aide d'une base adaptée. Il s'agit de la représentation AMNS, introduite par Thomas Plantard durant sa thèse [Pla05]. L'originalité de son approche est l'utilisation d'une représentation polynomiale des éléments du corps fini. Cette écriture permet de représenter un élément du corps fini par un vecteur dont les coefficients sont de taille inférieure à celle d'un mot machine. Les opérations sur le corps fini s'en trouvent donc améliorées, ce qui accélère l'arithmétique sur le corps fini. Une représentation AMNS bien calibrée permet de construire des racines de l'unité ayant des représentations creuses en base AMNS.

La multiplication par ces racines de l'unité dans $\mathbb{F}_p$ est alors très efficace. L'utilisation de l'approche DFT/FFT pour faire la multiplication dans l'extension de $\mathbb{F}_{p^k}$ devient intéressante. Notre approche consiste à combiner la représentation AMNS et la multiplication par FFT. Nous comparons la complexité de notre approche aux approches classiques, en utilisant la multiplication de Karatsuba et Toom Cook pour des corps dont le degré d'extension k vérifie que $k \geq 12$ [KM05].

Après avoir amélioré l'arithmétique des corps finis pour une valeur de k quelconque, nous nous intéressons au cas spécifique des courbes de degré de plongement $k = 15$ [EGI09]. Les dernières recherches pour un calcul efficace des couplages se sont focalisées sur la réduction du nombre d'itérations durant l'algorithme de Miller. Vercauteren a prouvé dans [Ver08] que pour la plupart des courbes elliptiques ordinaires construites par multiplication complexe, le nombre d'itérations de l'algorithme de Miller ne peut être inférieur à $O(r^{1/\varphi(k)})$, où k représente le degré de plongement de la courbe et φ la fonction d'Euler. Cette fonction associe a tout entier k le nombre de nombre premiers avec k et inférieurs à k. Par conséquent, pour un niveau de sécurité fixé et donc une taille en bit de r fixée, le calcul des couplages sera plus efficace en terme d'itérations de l'algorithme de Miller pour des courbes telles que l'indicatrice d'Euler du degré de plongement k est grande.

L'efficacité du calcul des couplages met donc en balance un degré de plongement k pas trop grand pour que l'arithmétique dans les extensions de corps soit efficace, et un degré de plongement k qui possède une grande indicatrice d'Euler. A ces conditions d'efficacité s'ajoute une condition sur le rapport : $\rho = \frac{\log p}{\log r}$ dépendant de p, la caractéristique du corps de base, et de r la taille du sous groupe de la courbe elliptique. Ce rapport ρ traduit le surcoût induit par l'arithmétique dans $\mathbb{F}_p$. Si ρ est proche de 1, alors p et r sont approximativement de même taille en bit, et les calculs sur la courbe elliptique et sur le corps fini seront beaucoup plus efficaces que pour des courbes avec ρ proche de 2.

Nous nous sommes intéressés aux courbes de degré de plongement $k = 15$. En prenant en compte les paramètres de sécurité, les courbes dont le rapport $\rho \approx 2$ doivent être envisagées pour des hauts niveaux de sécurité ; et les courbes de degré $k = 15$ et de rapport $\rho \approx 1$ doivent être envisagées pour des niveaux moyens de sécurité. Par conséquent, au delà de l'aspect purement théorique, l'étude du calcul des couplages sur ces courbes se justifie pleinement par des critères de sécurité dans le cadre de l'utilisation des couplages en cryptographie.
Le Chapitre IV présente la première description d'une arithmétique et d'un calcul de couplage efficace pour les courbe de degré $k = 15$. Nous

présentons les multiples avantages de ces courbes. Dans un premier temps, nous démontrons qu'il est possible de calculer le couplage Ate en $\log_2(r^{1/8})$ itérations de l'algorithme de Miller ; et le couplage Twisted Ate en $\log_2(r^{1/2})$ itérations. Dans un second temps, nous montrons que l'utilisation d'un twist de degré 3 nous permet d'effectuer la majorité des opérations sur $\mathbb{F}_p$ et $\mathbb{F}_{p^5}$, mais aussi, que ce twist nous évite le calcul des dénominateurs durant l'algorithme de Miller et l'inversion finale qui en résulte. Enfin, dans un troisième temps, nous introduisons une arithmétique originale et efficace sur $\mathbb{F}_{p^5}$, cette arithmétique repose sur une méthode de multiplication par évaluation-interpolation. Nos résultats sont qu'en utilisant notre arithmétique optimisée dans $\mathbb{F}_{p^5}$ et $\mathbb{F}_{p^{15}}$, les courbes de degré 15 sont compétitives avec les courbes de Barreto-Naehrig de degré 12, et pourraient donc être prises en compte pour des applications cryptographiques.

Les chapitres précédents nous auront permis de mieux saisir les subtilités de l'implémentation des couplages. Nous pouvons maintenant étudier le sujet encore peu exploré de la résistance face à des attaques à canaux cachés des protocoles à base de couplage.

Initialement, l'utilisation des couplages n'impliquait aucun secret. Par conséquent, les attaques à canaux cachés n'étaient pas une menace pour la cryptographie à base de couplage. Cependant, lors d'un protocole de cryptographie basé sur l'identité, l'un des arguments pour lequel on calcule le couplage est secret. Nous décrivons dans le Chapitre V le protocole de cryptographie basé sur l'identité de Boneh et Franklin.

Le point important à souligner est que le déchiffrement d'un message transmis par un protocole basé sur l'identité nécessite un calcul de couplage entre la clé privée et le message chiffré reçu. Le message chiffré est par définition publique puisqu'il transite par un canal non sécurisé. Il devient alors envisageable d'appliquer une attaque par canaux cachés.

Un attaquant potentiel aura la connaissance de l'algorithme utilisé, en l'occurrence il s'agit de l'algorithme de Miller, et des paramètres nécessaires à son exécution, en particulier l'ordre r des points qui définit le nombre d'itérations faites durant l'algorithme de Miller. La clé privée n'influence ni le temps d'exécution, ni le nombre d'itérations de l'algorithme, ce qui est différent d'une attaque à canaux cachés contre un protocole RSA.

Une attaque par canaux cachés vise à casser un procédé de chiffrement non pas de manière frontale, mais en exploitant des fuites d'informations occasionnées par la mise en oeuvre du procédé de chiffrement : par exemple, dans le cas d'une carte à puce, nous pouvons mesurer précisément la consommation électrique et en déduire des informations sur les données manipulées, et même sur le secret utilisé. Casser un protocole cryptographique signifie pouvoir retrouver le message

clair à partir du message chiffré. Nous nous intéressons dans le Chapitre V à deux types d'attaques particuliers. Les attaques différentielles par consommation de courant et les attaques par injections de fautes. Nous menons nos attaques sur l'algorithme de Miller. Cet algorithme est l'étape centrale lors des calculs des couplages de Weil, de Tate et du récent couplage Ate.

Le principe des attaques par analyse de consommation de courant est d'étudier la courbe de consommation électrique d'un algorithme cryptographique pour en déduire des informations sur le secret utilisé. Pour mener à bien une attaque différentielle nous devons relever plusieurs traces de consommation de courant correspondant à plusieurs exécutions de l'algorithme cible, chaque exécution de l'algorithme étant faite avec une entrée différente et connue. La connaissance de l'algorithme utilisé permet un traitement statistique des courbes de consommations, ceci afin de retrouver le secret utilisé. Nous décrivons le principe d'une attaque différentielle par analyse de courant, et nous l'appliquons à l'algorithme de Miller. Nous présentons une validation expérimentale de notre théorie. Nous montrons que la contre mesure consistant à placer le secret du couplage en premier argument n'est pas efficace, et que sans contre mesure, l'algorithme de Miller, ainsi que tous les couplages basés sur celui ci sont vulnérables à une attaque différentielle par consommation de courant.

Les attaques par injection de fautes consistent à perturber le circuit pendant une exécution de l'algorithme. L'analyse du résultat faussé permet de retrouver des informations sur le secret utilisé. Les fautes auxquelles nous nous intéressons perturbent le nombre d'itérations de l'algorithme de Miller ; le but de l'attaque est d'obtenir deux résultats faussés correspondants à τ et τ+1 itérations consécutives. Nous montrons que l'algorithme de Miller est sensible à une telle attaque, et ce quel que soit le système de coordonnées utilisé. Cette vulnérabilité implique celle du couplage de Weil, mais pas directement celle du couplage de Tate. En effet, la dernière étape du couplage de Tate est une exponentiation qui pourrait protéger ce couplage d'une attaque par faute. La même remarque s'applique aux couplages Ate et Twisted Ate. Nous présentons des méthodes issues de la littérature en micro-électronique pour pouvoir malgré tout réussir l'attaque par injection de fautes contre ces couplages.

Chapitre I

État de l'art

Sommaire

I.1 Présentation

Nous introduisons dans ce chapitre les éléments théoriques dont nous nous servirons tout au long de ce manuscrit. Nous commençons par introduire les courbes elliptiques en donnant leur définition ainsi que leurs principales propriétés. Ensuite, nous présentons les couplages, outil mathématique basé sur ces courbes et dont l'utilisation en cryptographie ne cesse de se développer.

Les courbes elliptiques sont des objets mathématiques connus depuis le XVIII ième siècle à travers les travaux de Fagnano [CGCDF50]. Leur date de naissance serait le 23 décembre 1751. Elles sont apparues la première fois lors de l'étude du périmètre d'ellipse, d'où leur nom. Elles trouvent aujourd'hui de nombreuses applications dans des domaines assez divers comme par exemple la théorie des nombres ou la cryptographie.

Nous nous intéressons à l'utilisation cryptographique de ces courbes, et nous nous contentons d'étudier leur arithmétique lorsqu'elles sont définies sur des corps finis, en la présentant de la manière la plus simple. Nous verrons dans la partie I.2 qu'une courbe elliptique est l'ensemble des points vérifiant une équation algébrique, plus un point à l'infini. Nous donnons leurs principales propriétés dont nous aurons besoin dans la partie I.3. La sécurité des protocoles cryptographiques repose souvent sur la difficulté de résoudre le problème du logarithme discret. Nous introduisons le problème du logarithme discret dans la partie I.4. Puis dans la partie I.5 nous présentons la multiplication complexe, une méthode de construction de courbes elliptiques.

Nous découvrirons ensuite la notion de couplages définis sur des sous groupes de courbes elliptiques. Les couplages ont permis la simplification et la création de protocoles cryptographiques. Nous motivons l'intérêt pour leur étude par une présentation de la cryptographie à base de couplage dans la partie I.7.3. Nous présentons ensuite dans la partie I.8 les couplages de Weil et de Tate ainsi que l'algorithme de Miller permettant de les calculer. Nous

terminons ce chapitre par l'introduction, dans la partie I.9, des derniers nés de la famille des couplages : les couplages Ate et Twisted Ate. Nous n'entrons pas dans les détails mathématiques, et nous simplifions autant que possibles les notions introduites. Nous ne présentons que la théorie nécessaire à ce manuscrit. Pour des définitions et démonstrations plus complètes de la théorie des courbes elliptiques et des couplages, nous renvoyons le lecteur aux références [Sil92, Was03] et [BSS05].

I.2 Définition et propriétés des courbes elliptiques

I.2.1 Quelques mots à propos des corps finis

Rappelons tout d'abord très succinctement la définition et les principales propriétés d'un corps fini, dont une description détaillée est faite dans [LN94].

Soit p un nombre premier, nous noterons $\mathbb{F}_p$ le **corps fini** à p éléments. Il est isomorphe à $\mathbb{Z}/p\mathbb{Z} = \{0, 1, 2, \ldots, p-2, p-1\}$. $\mathbb{F}_p$ est un corps premier de caractéristique p. La **caractéristique** d'un corps est le plus entier n non nul tel que le produit de n'importe quel élément du corps par n soit nul, si un tel n n'existe pas alors le corps est de caractéristique 0 [LN94]. Nous notons $\mathbb{F}_p^\star$ le sous ensemble de $\mathbb{F}_p$ composé des éléments inversibles pour la loi $\times$ dans $\mathbb{F}_p$, il s'agit du sous groupe multiplicatif de $\mathbb{F}_p$. Le sous groupe $\mathbb{F}_p^\star$ est cyclique [LN94, Chap. 2 Th. 2.8], il admet donc un élément g tel que n'importe lequel des éléments de $\mathbb{F}_p^\star$ soit une puissance de g ; g est appelé un générateur de $\mathbb{F}_p^\star$.

Le calcul des couplages repose sur l'arithmétique des corps finis. En particulier, nous verrons que de nombreuses démonstrations de théorèmes reposent sur le petit théorème de Fermat :

Théorème I.2.1 (Petit théorème de Fermat)**.** *Soit p un nombre premier et x un entier non nul dans $\mathbb{F}_p$. Alors les égalités suivantes sont vraies :*

$$\begin{aligned} x^p &\equiv x \ mod\ p, \\ si\ x \neq 0 \ mod\ p,\ x^{(p-1)} &\equiv 1 \ mod\ p. \end{aligned}$$

L'**ordre d'un élément** x est le plus petit entier α tel que $x^\alpha \equiv 1 \bmod(p)$. Les générateurs de $\mathbb{F}_p^\star$ sont les éléments de $\mathbb{F}_p$ d'ordre $p-1$.

Nous notons $\mathbb{F}_p[X]$ l'ensemble des polynômes à coefficients dans $\mathbb{F}_p$; nous aurons en particulier besoin de la notion de polynômes irréductibles :

Définition I.2.2. Un **polynôme** $P(X)$ de degré n est dit **irréductible** sur $\mathbb{F}_p$ si $P(X)$ n'est divisible par aucun polynôme $Q(X)$ de degré compris entre 1 et $(n-1)$.

***Exemple* I.2.3.** Soit $p = 257$, le corps fini $\mathbb{F}_{257}$ est défini comme étant l'ensemble des entiers compris entre 0 et 256. Le polynôme $P(X) = X^4 - 3$ est un polynôme irréductible sur $\mathbb{F}_{257}$. En effet, P est de degré 4. Donc si P était réductible, P s'écrirait soit comme le produit d'un polynôme de degré 1 et d'un polynôme de degré 3 soit comme le produit de deux polynômes de degré 2. Or,

1. si P est le produit d'un polynôme de degré 1 et d'un polynôme de degré 3, cela signifie que P admet une racine dans $\mathbb{F}_{257}$, c'est-à-dire qu'il existe un élément $x \in \mathbb{F}_{257}$ tel que $x^4 = 3$. Si un tel élément existait, nous aurions, en utilisant le petit théorème de Fermat I.2.1, $1 = x^{257-1} = 3^{\frac{257-1}{4}} = 3^{64} = -16 \bmod 257$, ce qui est absurde.

2. si P est le produit de deux polynômes de degré 2, nous pouvons écrire

$$X^4 - 3 = (X^2 + aX + b)(X^2 + cX + d),$$

pour certains coefficients $a, b, c, d \in \mathbb{F}_{257}$. En développant, nous construisons par identification des termes un système en les a, b, c, d. De même que pour le premier cas, nous montrons par l'absurde que ce système n'admet pas de solutions.

L'arithmétique des couplages nécessite de construire des extensions de $\mathbb{F}_p$. Une extension de $\mathbb{F}_p$ peut être vue comme un corps plus gros englobant $\mathbb{F}_p$, la définition suivante donne une représentation classique d'une telle extension de corps.

Une **extension de degré k du corps fini** $\mathbb{F}_p$, notée $\mathbb{F}_{p^k}$, est un corps fini que nous pouvons construire par le quotient $\mathbb{F}_p[X]/(P(X)\mathbb{F}_p[X])$ où $\mathbb{F}_p[X]$ est l'ensemble des polynômes à coefficients dans $\mathbb{F}_p$, $P(X)$ est un polynôme irréductible sur $\mathbb{F}_p$ de degré k et $P(X)\mathbb{F}_p[X]$ est l'ensemble des polynômes admettant $P(X)$ comme facteur (ou divisible par $P(X)$).
Une extension de degré k de $\mathbb{F}_p$ peut donc être décrite comme l'ensemble des polynômes à coefficients dans $\mathbb{F}_p$ et de degré strictement inférieur à k.

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

Il est possible de construire une **base** de $\mathbb{F}_{p^k}$. Notons γ une racine dans $\mathbb{F}_{p^k}$ de $P(X)$, polynôme irréductible de degré k, γ est une classe de X dans $\mathbb{F}_{p^k}$. Alors une base de $\mathbb{F}_{p^k}$ est composée des puissances de γ inférieures à k : $\{1, \gamma, \gamma^2, \ldots, \gamma^{k-1}\}$. Cette représentation permet de déduire directement le cardinal d'une extension de degré k du corps $\mathbb{F}_p$:

Propriété I.2.4. *Le cardinal de $\mathbb{F}_{p^k}$, noté $\#\mathbb{F}_{p^k}$, est p^k.*

Les racines des polynômes de $\mathbb{F}_p[X]$ ne sont pas toutes contenues dans $\mathbb{F}_p$. Afin de décrire l'espace dans lequel elles évoluent nous devons introduire la notion de clôture algébrique d'un corps :

Définition I.2.5. La clôture algébrique d'un corps fini $\mathbb{F}_p$ est l'extension de ce corps qui contient les racines de tous les polynômes de $\mathbb{F}_p[X]$. Nous la noterons $\overline{\mathbb{F}_p}$.

***Remarque* I.2.6.** Il n'y pas unicité de la clôture algébrique associée à un corps, mais elles sont toutes isomorphes [LN94].

***Exemple* I.2.7.** Le polynôme $P(X) = X^4 - 3$ est un polynôme irréductible sur $\mathbb{F}_{257}$. Comme il s'agit d'un polynôme de degré 4 il nous permet de construire une extension $\mathbb{F}_{257^4}$ de degré 4 du corps fini $\mathbb{F}_{257}$. Notons γ une racine dans $\mathbb{F}_{257^4}$ du polynôme $P(X) = X^4 - 3$. Nous pouvons alors décrire les éléments de $\mathbb{F}_{257^4}$ comme des polynômes en γ à coefficient dans $\mathbb{F}_{257^4}$:

$$\mathbb{F}_{257^4} \cong \{a_0 + a_1\gamma + a_2\gamma^2 + a_3\gamma^3, \quad a_i \in \mathbb{F}_{257}\}.$$

L'arithmétique dans $\mathbb{F}_{p^k}$ combine l'arithmétique des polynômes et celle de $\mathbb{F}_p$, elle est faite modulo un polynôme irréductible $P(X)$ de degré k. Nous privilégions les polynômes $P(X)$ creux de la forme $X^k - \beta$ pour un élément $\beta \in \mathbb{F}_p$ « petit » [KM05, BE07]. La probabilité qu'un élément pris au hasard dans $\mathbb{F}_p$ soit une racine k-ième dans $\mathbb{F}_p$ dépend fortement de k. Par exemple, si $p \equiv 1 \bmod(k)$ nous pouvons déterminer la probabilité qu'un élément pris au hasard dans $\mathbb{F}_p$ soit une racine k-ième dans $\mathbb{F}_p$.

Propriété I.2.8. *Soit p un nombre premier et k un entier positif. Si $p \equiv 1$ mod(k), alors un élément α pris au hasard dans $\mathbb{F}_p$ a une chance sur k d'être une racine k-ième dans $\mathbb{F}_p$.*

Démonstration. Soit α une puissance k-ième dans $\mathbb{F}_p$. Il existe donc $x \in \mathbb{F}_p$ tel que $x^k \equiv \alpha \bmod(p)$. Comme $p \equiv 1 \bmod(k)$, nous savons que k divise $(p-1)$. Nous pouvons donc élever la congruence reliant x et α à la puissance $\frac{(p-1)}{k}$. Ce qui nous donne les congruences suivantes :

$$\begin{aligned} x^{k\frac{(p-1)}{k}} &\equiv \alpha^{\frac{(p-1)}{k}} \bmod(p), \\ \alpha^{\frac{(p-1)}{k}} &\equiv x^{(p-1)} \bmod(p), \\ \alpha^{\frac{(p-1)}{k}} &\equiv 1 \bmod(p). \end{aligned}$$

Ainsi, α est une puissance k-ième dans $\mathbb{F}_p$ si et seulement si α est un élément du sous groupe d'ordre $\frac{(p-1)}{k}$ de $\mathbb{F}_p$. Or, ce sous groupe compte exactement $\frac{(p-1)}{k}$ éléments. □

Corollaire I.2.9. *Pour un élément β pris au hasard dans $\mathbb{F}_p$, le polynôme $X^k - \beta$ est irréductible avec une probabilité de $\frac{p-1}{k}$, si $p \equiv 1 \ mod(k)$.*

Pour un polynôme de réduction de la forme $X^k - \beta$, la multiplication de deux éléments dans cette représentation

$$U(X) = \sum_{i=0}^{k-1} u_i X^i \quad \text{et } V(X) = \sum_{i=0}^{k-1} v_i X^i$$

s'effectue tout d'abord en calculant le produit polynomial $W = U \times V$, puis en réduisant $W(X)$ modulo $P(X)$. La forme binomiale de $P(X)$ rend la réduction modulo $P(X)$ très simple, comme nous le montrons ci-dessous.

Nous séparons W en deux morceaux $W = \underline{W} + X^k \overline{W}$ tels que $\deg \underline{W} < k$ et $\deg \overline{W} < k$. La réduction consiste alors à calculer la somme $\underline{W} + \beta \overline{W}$, puisque $X^k \equiv \beta \mod P$. Le problème principal est donc d'effectuer la multiplication des polynômes U et V de la manière la plus efficace possible. Ce problème sera traité dans la partie II.3.

***Exemple* I.2.10.** Soient $a = 3 + \gamma^3$ et $b = 234\gamma + 36\gamma^2$ deux éléments de $\mathbb{F}_{257^4}$. Pour calculer $c = a \times b \in \mathbb{F}_{257^4}$, nous calculons d'abord le produit dans $\mathbb{F}_p[\gamma]$:
$c = 702\gamma + 108\gamma^2 + 234\gamma^4 + 36\gamma^5$,
soit $c = 188\gamma + 108\gamma^2 + 234\gamma^4 + 36\gamma^5$.
Puis nous calculons la réduction modulo $P(X) = X^4 - 3$.
Sachant que dans $\mathbb{F}_{257^4}$ nous avons $\gamma^4 = 3$, nous pouvons réduire c comme suit : $c = 3 \times 234 + (188 + 3 \times 36)\gamma + 108\gamma^2 = 702 + 296\gamma + 108\gamma^2$, il ne reste plus qu'à réduire les coefficients modulo 257 pour obtenir le résultat de $a \times b$: $c = 188 + 39\gamma + 108\gamma^2$.

I.2.2 Généralités sur les courbes elliptiques

Nous commençons par donner la représentation algébrique d'une courbe elliptique définie sur un corps quelconque [Sil92, Chap III].

Une **courbe elliptique** E sur le corps $\mathbb{K}$ (notée $E(\mathbb{K})$) est définie par une équation de Weierstrass du type

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6, \text{ pour } a_i \in \mathbb{K}. \tag{I.2.1}$$

Une équation de Weierstrass définit bien une courbe elliptique s'il n'existe pas de point de $E(\overline{\mathbb{K}})$ pour lequel les dérivées partielles $(2y + a_1 x + a_3)$ et $(3x^2 + 2a_2 x + a_4 - a_1 y)$ s'annulent simultanément. Les coordonnées (x, y) sont appelées **coordonnées affines** sur E.

La courbe elliptique $E(\mathbb{K})$ est l'ensemble des points $P = (x, y)$ de $\mathbb{K}^2$ vérifiant l'équation I.2.1 plus un **point à l'infini** noté P_∞.

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

Le point à l'infini peut symboliquement être vu comme le point d'intersection des droites verticales. Cette idée grossière se formalise en considérant l'espace projectif, ce qui est fait dans [Sil92, Chap III §1].

***Remarque* I.2.11.** Dans le cas où $\mathbb{K} = \mathbb{R}$, la non singularité d'une courbe se traduit graphiquement par l'absence de point multiple sur son tracé. Le cas réel est pratique pour visualiser le tracé d'une courbe elliptique, ce que nous illustrons dans la Figure I.1.

Il est possible de décrire plus simplement une courbe elliptique, en utilisant une équation de Weierstrass réduite comme nous le montre le théorème suivant de [Sil92, Chap III §1] :

Théorème I.2.12. *Soit $\mathbb{K}$ un corps fini de caractéristique différente de 2 et 3. Alors pour toute courbe elliptique $E(\mathbb{K})$ il existe un changement de variables tel que $E(\mathbb{K})$ admette une équation de Weierstrass réduite*

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

Il est possible d'associer à une courbe elliptique deux invariants de la courbe elliptique qui sont le discriminant et le j-invariant.

Définition I.2.13. A une courbe elliptique d'équation de Weierstrass $E : y^2 = x^3 + ax + b$, nous associons les invariants suivants :

$$\text{le discriminant : } \Delta = -16(4a^3 + 27b^2)$$

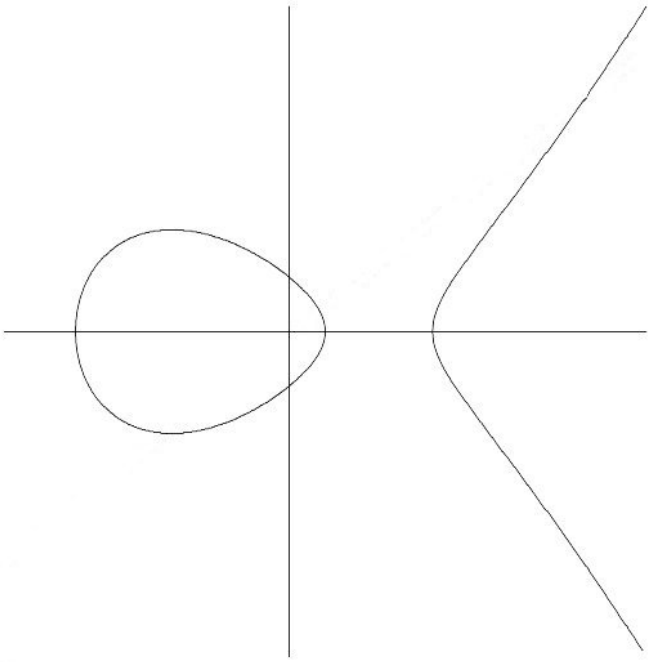
$$\text{et le } j\text{-invariant } \quad j = -1728\frac{(4a)^3}{\Delta}.$$

Un discriminant non nul assure la non singularité d'une courbe elliptique. Ces deux invariants liés à une courbe elliptique permettent de construire une classification des courbes elliptiques, ce qui est décrit dans [Sil92, Chap III Prop. 1.4].

I.2.3 Loi de groupe

Soit $E(\mathbb{K})$ une courbe elliptique sur un corps $\mathbb{K}$. Il existe une loi de groupe notée + sur $E(\mathbb{K})$ [Sil92, Chap III §2]. Cette loi existe en toute caractéristique, mais nous ne considérons que les courbes elliptiques définies sur un corps de caractéristique différente de 2 et 3 et données par une équation réduite de Weierstrass. La loi de groupe repose sur le théorème suivant :

FIGURE I.1 – Exemple de courbe elliptique sur $\mathbb{R}$.



Théorème I.2.14. *Soit E un courbe elliptique et D une droite du plan. Si D est tangente en un point à la courbe E ou bien si D coupe E en deux points distincts ; alors D coupe E en un unique autre point.*

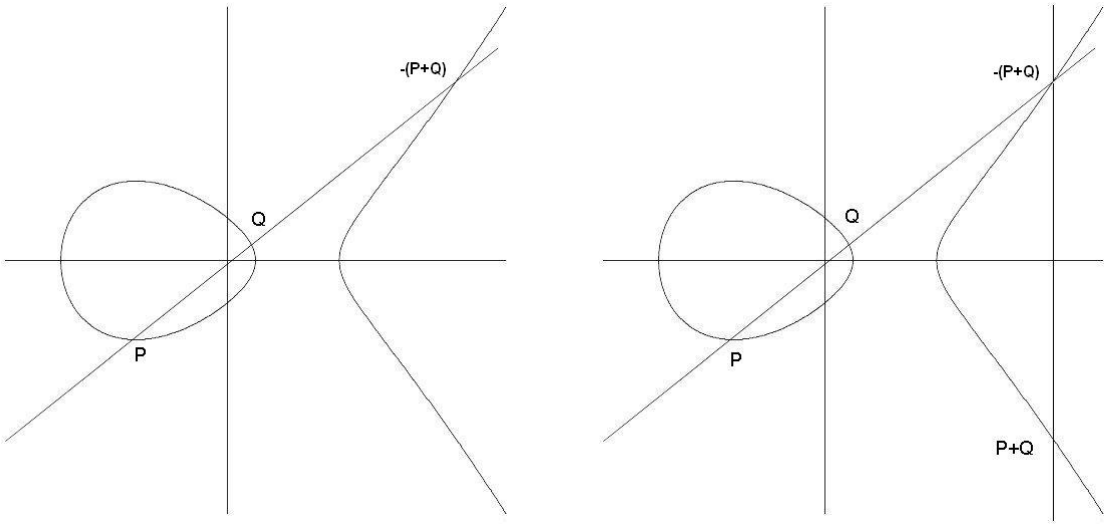
Nous commençons par donner une interprétation géométrique de la loi. Soient P et Q deux points $\mathbb{K}$ rationnels de E, nous appelons **somme des points** P et Q le point R obtenu par la construction suivante schématisée dans la Figure I.2. Tout d'abord nous traçons la droite (PQ). La droite (PQ) coupe la courbe elliptique E en un troisième point, le point $P+Q$ est le symétrique par rapport à l'axe des abscisses de ce troisième point d'intersection.

Pour P un point de la courbe, il est possible de calculer le point $[2]P$ en utilisant la tangente à la courbe au point P. Nous noterons $[r]P$ le point obtenu en additionnant r fois le point P, $[r]P = \underbrace{P+P+\ldots+P}_{r \text{ fois}}$. Les schémas de la Figure I.2 résument cette construction.

Cette construction nous permet d'assurer que le théorème suivant est vrai.

Théorème I.2.15. *L'ensemble des points d'une courbe elliptique E muni de la loi $+$ vérifie les propriétés suivantes :*

(i) *La loi $+$ est interne : $\forall P, Q \in E, P+Q \in E$.*



(a) Tracé de la droite (PQ) (b) Illustration de la construction de $P+Q$

FIGURE I.2 – Loi de groupe sur les courbes elliptiques.

(ii) *Existence d'un élément neutre :* $\forall P \in E,\ P + P_\infty = P$.

(iii) *Commutativité de la loi +* : $\forall P, Q \in E, P + Q = Q + P$.

(iv) *La loi + est associative :* $\forall P, Q, R \in E, (P + Q) + R = P + (Q + R)$.

(v) *La loi + est symétrique : $\forall P \in E$, il existe un point noté $-P$ tel que* $P + (-P) = P_\infty$.

$(E(\mathbb{K}), +)$ *est donc un groupe abélien additif d'élément neutre* P_∞.

I.2.4 Aspects algébriques de la loi de groupe

Nous pouvons décrire algébriquement la loi de groupe que nous venons de présenter graphiquement. Soient P et Q deux points d'une même courbe elliptique. Nous nous plaçons dans le plan affine sur $\mathbb{K} = \mathbb{R}$ pour écrire les équations de droites intervenant dans les constructions des points $P + Q$ et $[2]P$. Ces équations de droites vont nous permettre de décrire par des formules explicites la loi de groupe d'une courbe elliptique.

Soit E une courbe d'équation $y^2 = x^3 + ax + b$, $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ deux éléments de $E(\mathbb{K})$. Les coordonnées du point $-P$ l'inverse du point P pour la loi + peuvent s'exprimer en fonction des coordonnées de P.

Propriété I.2.16. *L'opposé du point P, noté $-P$, admet pour coordonnées* $(x_P, -y_P)$.

Cette propriété est une conséquence directe du fait que la droite $(P(-P))$ est une droite verticale. Le troisième point d'intersection de la courbe elliptique et de cette droite ne peut être que le point à l'infini. Ainsi, $P + (-P) = P_\infty$.

Addition de deux points

Soient $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ deux points distincts d'une courbe elliptique $E(\mathbb{K})$ tels que $P \neq -Q$. Pour trouver les formules donnant les coordonnées du point $R = P+Q$ avec $R = (x_R, y_R)$, nous cherchons à résoudre le système de deux équations formé par l'équation de la droite (PQ) et l'équation de la courbe elliptique. Ce système traduit exactement le fait que $-(P+Q)$ est le troisième point d'intersection de la courbe elliptique et de la droite (PQ). Nous obtenons les formules I.2.2, où λ représente la pente de la droite (PQ) :

$$\begin{cases} \lambda &= \frac{y_P - y_Q}{x_P - x_Q} \\ x_R &= \lambda^2 - x_P - x_Q \\ y_R &= \lambda(x_P - x_R) - y_P \end{cases} \tag{I.2.2}$$

La complexité de l'addition sur la courbe elliptique est de $6A+1S+2M+1I$; où A (respectivement S, M et I) représente une addition (respectivement un carré, une multiplication et une inversion) dans le corps $\mathbb{K}$. Nous n'avons pas compté deux fois la soustraction

Doublement d'un point

Pour trouver les formules donnant les coordonnées du point $R = [2]P$, nous utilisons le fait que $-[2]P$ est le second point d'intersection de la courbe E et de la tangente en E au point P. L'équation de la tangente et celle de la courbe elliptique forment un système dont la résolution nous donne les formules pour obtenir les coordonnées de point $R = [2]P$. Nous obtenons les formules I.2.3, où λ représente la pente de la tangente au point P de la courbe elliptique :

$$\begin{cases} \lambda &= \frac{3x_P^2 + a}{2y_P} \\ x_R &= \lambda^2 - 2x_P \\ y_R &= \lambda(x_P - x_R) - y_P \end{cases} \tag{I.2.3}$$

La complexité doublement sur la courbe elliptique est de $5A+1S+2M+1I$ dans $\mathbb{K}$. Le produit par 3 compte comme une addition, les produits par 2 ne sont en fait que des décalages dans la base binaire.

***Exemple* I.2.17.** Soit $p = 257$ et $\mathbb{F}_p$ le corps finis associé. Soit $E(\mathbb{F}_p)$ la courbe elliptique d'équation $y^2 = x^3+1$. $P = (8, 16)$ et $Q = (19, 101)$ sont deux

points $\mathbb{F}_p$ rationnels de la courbe elliptique E. Les coordonnées des points $-P$, $[2]P$ et $P+Q$ sont $-P = (8, 241)$, $[2]P = (20, 169)$ et $P+Q = (209, 113)$.

I.2.5 Systèmes de coordonnées

Soit $P = (x_P, y_P)$ un point de la courbe elliptique $E : y^2 = x^3 + ax + b$. Nous avons donné dans les parties précédentes les formules d'addition et de doublement de points en coordonnées affines. Par la suite, nous considérons uniquement le cas des courbes elliptiques définies sur un corps fini de type $\mathbb{F}_q$, avec $q = p^m$, pour p premier strictement plus grand que 3 et $m \in \mathbb{N}^\star$. La première constatation est que les formules d'addition et de doublement obtenues dans le cas $\mathbb{K} = \mathbb{R}$ restent valables sur $\mathbb{F}_q$.

Ces formules d'addition et de doublement font intervenir des inversions dans le corps $\mathbb{F}_q$. Or, l'inversion est une opération complexe et donc coûteuse sur les corps finis. Nous préférons utiliser d'autres systèmes de coordonnées, comme les coordonnées projectives ou Jacobiennes. L'avantage de ces coordonnées est qu'il n'y a pas d'inversion à faire durant le calcul d'une addition ou d'un doublement de point. Ces différents systèmes permettent d'optimiser les calculs impliquant des courbes elliptiques, ou comme nous le verrons, de protéger les protocoles basés sur les courbes elliptiques des attaques à canaux cachés. Ces deux points seront développés respectivement dans les Chapitres II et V.

- Les **coordonnées projectives** du point P sont notées $(X_P : Y_P : Z_P)$, Elles sont homogènes, elles vérifient que pour α non nul le point $(X_P : Y_P : Z_P)$ est équivalent au point $(\alpha \times X_P : \alpha \times Y_P : \alpha \times Z_P)$. L'homogénéité nous donne la correspondance suivante avec les coordonnées affines : le point $(X_P : Y_P : Z_P)$ en coordonnées projectives correspond au point $(\frac{X_P}{Z_P}, \frac{Y_P}{Z_P})$ en coordonnées affines, si $Z_P \neq 0$ et au point à l'infini sinon. L'opposé du point $(X_P : Y_P : Z_P)$ est le point $(X_P : -Y_P : Z_P)$.

  L'équation de la courbe elliptique en coordonnées projectives est $E : Y^2Z = X^3 + aXZ^2 + bZ^3$. Le point à l'infini est le point $(0 : 1 : 0)$.

- Les **coordonnées Jacobiennes** du point P sont notées $(X_P; Y_P; Z_P)$,

  Elles vérifient que pour α non nul le point $(X_P; Y_P; Z_P)$ est équivalent au point $(\alpha^2 \times X_P; \alpha^3 \times Y_P; \alpha \times Z_P)$. Le point $(X_P; Y_P; Z_P)$ en coordonnées projectives correspond au point $(\frac{X_P}{Z_P^2}, \frac{Y_P}{Z_P^3})$ en coordonnées affines, si $Z_P \neq 0$ et au point à l'infini sinon. L'opposé du point $(X_P : Y_P : Z_P)$ est le point $(X_P : -Y_P : Z_P)$.

  L'équation de la courbe elliptique en coordonnées Jacobiennes est $E : Y^2 = X^3 + aXZ^4 + bZ^6$. Le point à l'infini est le point $(1; 1; 0)$.

***Remarque* I.2.18.** Les points à l'infini en coordonnées projectives et Jacobiennes sont obtenus en posant $Z = 0$ et en utilisant la propriété d'homogénéité des coordonnées projectives et Jacobiennes.

Soit $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ deux points d'une courbe elliptique en coordonnées affines, nous souhaitons déterminer les formules de l'addition et du doublement pour les coordonnées Jacobiennes (la même méthode s'applique pour les coordonnées projectives). Nous commençons par les formules de **doublement d'un point**, en donnant les formules pour calculer les coordonnées Jacobiennes du point $[2]P$. Nous remplaçons dans la formule du doublement en coordonnées affines les coordonnées x et y par leurs équivalents $\frac{X}{Z^2}$ et $\frac{Y}{Z^3}$. En réduisant les dénominateurs, λ, la pente de la tangente à la courbe au point P, devient une fraction en X_P, Y_P et Z_P :

$$\lambda = \frac{3X_P^2 + aZ_P^4}{2Y_P Z_P}.$$

Soit $R = [2]P = (x_R, y_R)$. L'égalité $x_R = \lambda^2 - 2x_P$ se transforme en

$$\frac{X_R}{Z_R^2} = \frac{(3X_P^3 + aZ_P^4)^2 - 8X_P Y_P^2}{4Y_P^2 Z_P^2},$$

et permet de trouver X_R et Z_R en fonction de X_P, Y_P et Z_P :

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

Ensuite, nous partons de l'égalité $y_R = \lambda(x_P - x_R) - y_P$, pour exprimer Y_R en fonction des coordonnées du point P :

$$Y_R = Z_P^3 \left(\lambda \left(\frac{X_P}{Z_P^2} - \frac{X_R}{Z_R^2} \right) - \frac{Y_P}{Z_P^3} \right).$$

Après calcul, nous obtenons les formules suivantes pour le doublement en coordonnées Jacobiennes :

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

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

L'opération de doublement d'un point sur la courbe elliptique s'effectue en six carrés et trois multiplications dans le corps de base sur lequel est défini la courbe elliptique. Ce que nous noterons $6S_q + 3M_q$, avec S_q qui représente un carré, et M_q une multiplication dans $\mathbb{F}_q$.

En appliquant la même transformation aux formules de **l'addition de point**, nous obtenons les Formules I.2.4 pour l'addition des points $P = (x_P, y_P)$ et $Q = (x_Q, y_Q)$ en coordonnées Jacobiennes, où nous notons $R = P + Q$.

$$\begin{array}{lll} A = X_P Z_Q^2, & B = X_Q Z_P^2, & C = Y_P Z_Q^3, \\ D = Y_Q Z_P^3, & E = B - A, & F = D - C. \end{array}$$

$$\left\{ \begin{array}{rcl} X_R & = & -E^3 - 2AE^2 + F^2 \\ Y_R & = & -CE^3 + F(AE^2 - X_R) \\ Z_R & = & Z_P Z_Q E \end{array} \right. \tag{I.2.4}$$

L'opération d'addition de deux points sur la courbe elliptique s'effectue en quatre carrés et douze multiplications dans $\mathbb{F}_q$, soit $4S_q + 12M_q$. D'autres systèmes de coordonnées existent, par exemple les courbes d'Edwards liées aux courbes homonymes [Edw07], ils sont décrits entre autres dans [CF06, Lan09].

I.3 Cardinalité et torsion

Les courbes elliptiques définies sur un corps fini ont un cardinal fini. Il existe des algorithmes permettant de le calculer. Nous décrivons ici les grandes lignes de la théorie liée à la cardinalité des courbes elliptiques, et décrite plus en détail dans [Sil92, Chap V §1] et [Was03, §4.3].

I.3.1 Cardinal d'une courbe elliptique

Le premier résultat intéressant concernant le cardinal d'une courbe elliptique est le théorème de Hasse I.3.1. Ce théorème donne un intervalle dans lequel se trouvent toutes les valeurs possibles du cardinal d'une courbe elliptique définie sur $\mathbb{F}_q$, pour $q = p^m$ avec p premier strictement supérieur à 3 et $m \in \mathbb{N}^\star$. Nous noterons $\#E(\mathbb{F}_q)$ le cardinal de la courbe elliptique $E(\mathbb{F}_q)$.

Théorème I.3.1 (Théorème de Hasse)**.** *Soit E une courbe elliptique définie sur un corps $\mathbb{F}_q$ alors*

$$|\#E(\mathbb{F}_q) - (q+1)| \leq 2\sqrt{q}.$$

Cette formule peut être ré-écrite en utilisant la trace du Frobenius sur la courbe elliptique.

Définition I.3.2. Le **Frobenius**, noté π_p, est une application de la courbe elliptique $E(\mathbb{F}_q)$ d'équation $y^2 = x^3 + ax + b$ sur la courbe elliptique notée $E^{(p)}(\mathbb{F}_q)$ et d'équation $y^2 = x^3 + a^p x + b^p$. Il est donné par :

$$\begin{array}{rcl} \pi_p : E(\mathbb{F}_q) & \to & E^{(p)}(\mathbb{F}_q) \\ P = (x_P, y_P) & \to & \pi_p(P) = (x_P^p, y_P^p) \end{array}$$

Le Frobenius est un morphisme de la courbe elliptique [Was03, §4.2]. Comme tout morphisme de groupe, il vérifie la propriété suivante.

Propriété I.3.3. *Soient P_1 et P_2 deux points d'une courbe elliptique $E(\mathbb{F}_q)$. Alors*

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

Le Frobenius induit un morphisme sur $E(\mathbb{F}_q)$ dont nous pouvons calculer le polynôme caractéristique [CF06, §5.2.2]. Le polynôme caractéristique du Frobenius est l'endomorphisme nul sur la courbe elliptique. Il est défini par $\Phi_q(X) = X^2 - tX + det(\pi_q)$; où t représente la trace de π_q et $det(\pi_q)$ son déterminant.

Théorème I.3.4. *Soit E une courbe elliptique sur $\mathbb{F}_q$ et t la trace du Frobenius sur E. La formule suivante donne le cardinal de la courbe $E(\mathbb{F}_q)$:*

$$\#E(\mathbb{F}_q) = q - t + 1.$$

***Remarque* I.3.5.** Le théorème de Hasse, se démontre en utilisant l'encadrement suivant de la trace t :

$$|t| \leq 2\sqrt{q}.$$

Cet encadrement est optimal, et le théorème suivant est un résultat important concernant la cardinalité des courbes elliptiques ([Was03, Th 4.3]).

Théorème I.3.6. *Pour chaque entier t premier avec q dans l'intervalle $[-2\sqrt{q}; 2\sqrt{q}]$, il existe au moins une courbe elliptique $E(\mathbb{F}_q)$ de cardinal $q - t + 1$.*

Il existe des algorithmes permettant de calculer le cardinal d'une courbe elliptique donnée. Citons par exemple l'algorithme de Schoof [Sch95]. Il consiste à choisir n nombres premiers $\{l_1, l_2, \ldots, l_n\}$ distincts tels que leur produits est strictement supérieur à $4\sqrt{q}$: $l_1 \times l_2 \times \cdots \times l_n > 4\sqrt{q}$; puis à déterminer les valeurs de la trace du Frobenius sur la courbe elliptique modulo les l_i. Ces valeurs sont trouvées en utilisant les points $P \in E(\mathbb{F}_q)$ tels que $[l_i]P = P_\infty$. La reconstruction de t s'effectue ensuite en appliquant le théorème des restes Chinois. Le principe de l'algorithme de Schoof et ses améliorations sont détaillées dans [Sch95], l'algorithme de Schoof-Elkies-Atkin a une complexité en $O((\log(q))^4)$ opérations. L'algorithme est implémenté dans les logiciels de calculs formels Magma [BCP97] et PariGP [PAR05].

I.3.2 Sous groupe de torsion

Soient E une courbe elliptique définie sur un corps $\mathbb{F}_q$ et r un entier. Nous définissons la notion de points de r-torsion de la courbe elliptique $E(\overline{\mathbb{F}_q})$ comme suit :

Définition I.3.7. Un point P de la courbe elliptique $E(\overline{\mathbb{F}_q})$ est 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 [Sil92, Chap. III §4 et §6].

***Exemple* I.3.8.** Soit E la courbe elliptique d'équation $y^2 = x^3 + 1$ sur $\mathbb{F}_{257}$. Le cardinal de cette courbe est $258 = 2 \times 129$. 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. Le point $Q = (-1, 0)$ est un point de E de 2-torsion.

Si nous devions écrire algébriquement l'expression des coordonnées de $[r]P$ en fonction de celles de P, les formules de doublement et d'addition données dans la partie I.2.4 nous permettraient d'écrire 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 le dénominateur de $[r]P$ est nulle. Or, nous savons qu'un polynôme à coefficients dans un corps fini $\mathbb{F}_q$ n'a pas forcément toutes ses racines dans $\mathbb{F}_q$. Afin de décrire tous les points de r-torsion, nous devons donc considérer la courbe elliptique définie sur la clôture algébrique de $\mathbb{F}_q$, notée $\overline{\mathbb{F}_q}$.

***Remarque* I.3.9.** Soit $E(\mathbb{F}_q)$ une courbe elliptique sur un corps fini $\mathbb{F}_q$, et r un entier.

L'ensemble des points r-torsion appartient à $E(\overline{\mathbb{F}_q})$.

L'ensemble des points de $E(\overline{\mathbb{F}_q})$ de r-torsion est le noyau de l'application $[r] : P \to [r]P$:

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

Nous connaissons la structure de l'ensemble des points de r-torsion [Sil92, Chap III Cor. 6.4] :

Théorème I.3.10. *Soit r un entier premier avec p la caractéristique de $\mathbb{F}_q$. Alors*

$$E[r] \cong \mathbb{Z}/r\mathbb{Z} \times \mathbb{Z}/r\mathbb{Z}.$$

Si p divise r alors r peut s'écrire sous la forme $r = p^\alpha n$, où p ne divise pas n et alors

$$E[r] \cong \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z} \text{ ou } \mathbb{Z}/r\mathbb{Z} \times \mathbb{Z}/n\mathbb{Z}.$$

Corollaire I.3.11. *Pour r premier avec p et divisant $\#E(\mathbb{F}_q)$, nous déduisons du Théorème I.3.10 que $\#E[r] = r^2$.*

***Remarque* I.3.12.** Les points de 129-torsion de la courbe donnée dans l'exemple I.3.8 ne sont pas tous inclus dans $E(\mathbb{F}_{257})$, en effet $16641 = 129 \times 129 > 257$.

Il est possible de déterminer plus précisément l'espace de définition de $E[r]$. Nous introduisons pour cela la notion de degré de plongement de la courbe elliptique.

I.3.3 Degré de plongement d'une courbe elliptique

Le degré de plongement de la courbe elliptique est un élément important pour la détermination de l'espace dans lequel se trouvent tous les points de r-torsion [BSS05, §IX.5].

Définition I.3.13. Soit $E(\mathbb{F}_q)$ une courbe elliptique, et r un diviseur premier de $\#E(\mathbb{F}_q)$. Le **degré de plongement** de la courbe elliptique E relativement à r est le plus petit entier k pour lequel r divise $(q^k - 1)$. Autrement dit, k est l'ordre de q dans $\mathbb{F}_r$.

Ce paramètre de la courbe permet de savoir dans quel groupe vivent les points de r-torsion comme le montre le théorème I.3.14 de Balasubramanian et Koblitz [BK98] :

Théorème I.3.14. *Soit $E(\mathbb{F}_q)$ une courbe elliptique définie sur un corps fini $\mathbb{F}_q$, et r un diviseur de $\#E(\mathbb{F}_q)$. Si r est premier avec q et r ne divise pas $(q-1)$, alors $E[r] \subset E(\mathbb{F}_{q^n})$ pour n un entier positif si et seulement si r divise $(q^n - 1)$.*

***Remarque* I.3.15.** Le degré de plongement k est le degré de l'extension minimale du corps $\mathbb{F}_q$ telle que $E[r] \subset E(\mathbb{F}_{q^k})$. Si $k > 1$, nous savons donc que $E[r] \subset E(\mathbb{F}_{q^k})$. De plus, la structure des points de r-torsion est connue, il existe $P \in E(\mathbb{F}_q)$ et $Q \in E(\mathbb{F}_{q^k})$ tels que

$$E[r] = \{iP + jQ, (i,j) \in \mathbb{Z}^2\}.$$

***Exemple* I.3.16.**

- Le sous groupe de 2-torsion de la courbe E d'équation $y^2 = x^3 + 1$ sur $\mathbb{F}_{257}$ admet un degré de plongement égal à 1. Il suffit de remarquer que 2 divise $257 - 1 = 256$.
- Le sous groupe de 129-torsion de la courbe E d'équation $y^2 = x^3 + 1$ sur $\mathbb{F}_{257}$ admet un degré de plongement égal à 2. En effet, 129 divise $(257^2 - 1)$ mais pas $257 - 1$.

I.3.4 Classification des courbes

Les courbes elliptiques sont divisées en deux catégories, les courbes supersingulières et les courbes ordinaires [Sil92, Chap. V §3 et 4]. La définition suivante permet de différencier ces deux types de courbes :

Définition I.3.17. Soit E une courbe elliptique sur $\mathbb{F}_q$, corps de caractéristique $p > 3$. Une **courbe elliptique supersingulière** est une courbe elliptique E vérifiant une des conditions équivalentes suivantes :

- $\#E(\mathbb{F}_q) \equiv 1 \bmod(p)$, ou $\#E(\mathbb{F}_q) = q + 1 \bmod(p)$, ou $t = 0$,
- E n'admet pas de point d'ordre p sur $\overline{\mathbb{F}_q}$.

Une courbe non supersingulière est dite **courbe ordinaire**.

***Exemple* I.3.18.** La courbe définie sur $\mathbb{F}_{257}$ par l'équation $y^2 = x^3 + b$ pour $b \in \mathbb{F}_{257}^{\star}$ est une courbe supersingulière.

***Remarque* I.3.19.** La structure de groupe des courbes elliptiques supersingulières est bien connue. Le théorème 20 de [BSS05] donne une classification des courbes supersingulières. Toutes les courbes supersingulières ont un degré de plongement k au plus égal à 6.

La notion de sécurité pour un protocole cryptographique à base de courbe elliptique sera introduite dans la partie I.7.4. Nous pouvons toutefois remarquer que pour satisfaire les recommendations de sécurité du NIST [NIS07], il semble préférable de travailler avec des courbes dont le degré de plongement k est supérieur à 6, et donc d'éviter les courbes supersingulières.

I.3.5 Twist d'une courbe elliptique

Nous définissons la notion de twist ou tordue d'une courbe elliptique qui est une courbe elliptique telle que nous pouvons construire un isomorphisme entre ces deux courbes. Nous donnons ici uniquement les éléments dont nous aurons besoin pour les couplages, à savoir la définition d'une courbe tordue ainsi que le théorème décrivant explicitement les équations des twists. La théorie complète des courbes tordues est développée dans [Sil92, Chap. X §2 et §5].

Définition I.3.20. Soient E et $\widetilde{E}$ deux courbes elliptiques définies sur $\mathbb{F}_q$. Alors, la courbe $\widetilde{E}$ est **un twist de degré d de la courbe** E s'il existe un isomorphisme Ψ_d défini sur $\mathbb{F}_{q^d}$ de $\widetilde{E}$ dans E et tel que d soit minimal. Nous noterons :

$$\Psi_d : \widetilde{E}(\mathbb{F}_q) \rightarrow E(\mathbb{F}_{q^d})$$

Le nombre de twists de courbe elliptique possible est limité, il dépend du groupe formé par les morphismes de la courbe elliptique E dans elle même. Le Théorème I.3.21 donne la classification des différents twists possibles. Ce théorème est un résultat important de la théorie des twists de courbes elliptiques, détaillée dans [Sil92].

Théorème I.3.21. *Soit E une courbe elliptique d'équation $y^2 = x^3 + ax + b$ définie sur une extension $\mathbb{F}_{p^k}$ d'un corps fini $\mathbb{F}_p$. Suivant les valeurs de k, les degrés possibles de twists d sont 2, 3, 4 et 6.*

En pratique, nous considérons que E est définie sur $\mathbb{F}_{p^k}$. Nous notons $\widetilde{E}$ la courbe tordue de E définie sur $\mathbb{F}_{p^{k/d}}$ un sous-corps de $\mathbb{F}_{p^k}$, pour d un diviseur de k. Les différents twists possibles sont décrits ci-après. Nous considérons $E(\mathbb{F}_{p^k})$ d'équation $y^2 = x^3 + ax + b$. Pour chaque degré de twist d, nous donnons l'équation d'une courbe tordue de E notée $\widetilde{E}(\mathbb{F}_{p^{k/d}})$. Nous construisons explicitement le morphisme $\Psi_d : \widetilde{E} \to E$.

- $d = 2$, Soit $\nu \in \mathbb{F}_{p^{k/2}}$ tel que le polynôme $X^2 - \nu$ soit irréductible dans $\mathbb{F}_{p^{k/2}}$. L'équation de la courbe $\widetilde{E}$ définie sur $\mathbb{F}_{p^{k/2}}$ est $\widetilde{E} : \nu y^2 = x^3 + ax + b$. Le morphisme Ψ_2 est défini par :
$$\begin{array}{rcl} \Psi_2 : \widetilde{E}(\mathbb{F}_{p^{k/2}}) & \to & E(\mathbb{F}_{p^k}) \\ (x, y) & \to & (x, y\nu^{1/2}). \end{array}$$

- $d = 4$. La courbe elliptique E admet un twist de degré 4 si et seulement si $b = 0$. Soit $\nu \in \mathbb{F}_{p^{k/4}}$ tel que le polynôme $X^4 - \nu$ soit irréductible dans $\mathbb{F}_{p^{k/4}}$. L'équation de $\widetilde{E}$ définie sur $\mathbb{F}_{p^{k/4}}$ est alors $y^2 = x^3 + \frac{a}{\nu}x$. Le morphisme est dans ce cas :
$$\begin{array}{rcl} \Psi_4 : \widetilde{E}(\mathbb{F}_{p^{k/4}}) & \to & E(\mathbb{F}_{p^k}) \\ (x, y) & \to & (x\nu^{1/2}, y\nu^{3/4}). \end{array}$$

- $d = 3$ (respectivement 6), la courbe E admet un twist de degré 3 (respectivement 6) si et seulement $a = 0$. Soit $\nu \in \mathbb{F}_{p^{k/d}}$ tel que le polynôme $X^3 - \nu$ (respectivement $X^6 - \nu$) soit irréductible dans $\mathbb{F}_{p^{k/d}}$. L'équation de $\widetilde{E}$ est alors $y^2 = x^3 + \frac{b}{\nu}$. Le morphisme est :
$$\begin{array}{rcl} \Psi_d : \widetilde{E}(\mathbb{F}_{p^{k/d}}) & \to & E(\mathbb{F}_{p^k}) \\ (x, y) & \to & (x\nu^{1/3}, y\nu^{1/2}). \end{array}$$

Le j-invariant d'une courbe elliptique est un moyen simple de vérifier si deux courbes sont tordues l'une de l'autre [Sil92, Chap. X Cor. 5.4], comme l'indique la propriété suivante :

Propriété I.3.22. *Deux courbes elliptiques distinctes E et $\widetilde{E}$ sont tordues l'une de l'autre si et seulement si elles ont le même j-invariant.*

I.4 Le problème du logarithme discret

La sécurité de nombreux protocoles cryptographiques reposent sur le problème du logarithme discret, par exemple l'échange de clés de Diffie-Hellman [DH76] ou le chiffrement à clé publique d'El Gamal [El 85]. L'étude du logarithme discret a été faite pour chaque type de groupes utilisés en cryptographie. Nous introduisons les deux algorithmes de résolutions du logarithme discret qui influencent la taille des données intervenant en cryptographie à base de couplage. La sécurité des couplages repose sur la difficulté à résoudre le logarithme discret sur une courbe elliptique $E(\mathbb{F}_p)$ d'une part, et d'autre part sur un corps fini de type $\mathbb{F}_{p^k}$. Nous présentons la résolution générique de Pollard-ρ dans la partie I.4.1 qui est une des façons les plus efficaces de résoudre le logarithme discret sur une courbe elliptique $E(\mathbb{F}_p)$; puis dans la partie I.4.2 la méthode par calcul d'index plus spécifique aux corps finis $\mathbb{F}_{p^k}$.

Nous décrivons tout d'abord le problème du logarithme discret.

Soit $\mathbb{G}$ un groupe cyclique noté additivement, P un générateur de $\mathbb{G}$ et Q un élément de $\mathbb{G}$. Il existe un entier α tel que $Q = \alpha P$. Le problème du logarithme discret consiste à retrouver α à partir des données de P et Q.

I.4.1 Pollard Rho

Cette méthode calcule le logarithme discret dans n'importe quel groupe cyclique. Il s'agit de la méthode la plus efficace à l'heure actuelle pour résoudre un problème de logarithme discret sur une courbe elliptique. Elle fut proposée par Pollard en 1978 pour $\mathbb{F}_p^\star$ et s'inspire du paradoxe des anniversaires [Pol78]. Le principe de l'algorithme de Pollard-ρ est décrit ci-après :

Soit $\mathbb{G}$ un groupe cyclique de cardinal r. Choisissons une fonction $f : \mathbb{G} \to \mathbb{G}$ qui servira à construire une marche pseudo aléatoire. A partir d'un point $R_0 \in \mathbb{G}$, nous construisons la suite :

$$\begin{cases} & R_0, \\ \forall i \in \mathbb{N}^\star \; R_{i+1} & = & f(R_i). \end{cases}$$

Comme $\mathbb{G}$ est un ensemble fini, il existe des entiers μ et τ tels que $R_i = R_{i+\tau}$ pour tout $i \geq \mu$. La représentation graphique de la suite R_i des points obtenus à partir de ces itérations est formée d'une première suite de taille μ, suivie d'un cycle de taille τ dont le tracé fait penser à la lettre grecque ρ, d'où le nom de la méthode. Si la marche f est bien aléatoire, les valeurs de μ et τ sont en moyennes égales à $\sqrt{\pi r/8}$ [Har60].

L'efficacité de Pollard-ρ repose sur celle de la marche aléatoire. Nous décrivons une méthode de construction d'une fonction réalisant une marche

aléatoire. Pour cela, nous divisons $\mathbb{G}$ en s sous-ensembles $S_1, S_2, \ldots, S_s$ de tailles semblables, auxquels nous associons s paires d'entiers aléatoires (a_i, b_i). Nous calculons alors les s points Z_i tels que $Z_i = [a_i]P + [b_i]Q$ (où P et Q sont les deux points pour lesquels nous souhaitons résoudre le logarithme discret). La fonction f est définie par $f(R) = R + Z_i$, si $R \in S_i$.

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 $R_i = R_j$. Stocker toutes les valeurs R_i calculées dans une liste L, et chercher pour toute nouvelle valeur si elle se trouve dans la liste L est une méthode qui nécessiterait beaucoup d'espace mémoire. Nous utilisons plutôt un algorithme de recherche de cycle. Par exemple, un des plus simples est l'algorithme de recherche de cycles de Floyd [CF06, Chapitre 19], décrit par l'Algorithme 1. Il consiste à calculer deux suites de points commençant à partir de R_0, mais « avançant » à des rythmes différents. Pour la première, nous appliquons une fois la fonction f, pour la seconde, deux fois.

Algorithme 1 : Recherche de cycles de Floyd

Données : $R_0 \in E$ et f une fonction aléatoire ;
Résultat : Un indice i pour lequel $R_i = R_{2i}$;
$R_1 = f(R_0)$;
$R_2 = f \circ f(R_0)$;
$i = 1$;
tant que $R_1 \neq R_2$ **faire**
  $R_1 = f(R_1)$;
  $R_2 = f \circ f(R_2)$;
  $i = i + 1$;
fin
retourner i

Pendant l'exécution de l'algorithme de recherche de cycles de Floyd, nous ne stockons que deux valeurs R_1 et R_2, ainsi que leurs décompositions (u_1, v_1) et (u_2, v_2) en fonction de P et Q, $R_1 = u_1.P + v_1.Q$ et $R_2 = u_2.P + v_2.Q$. Nous attendons qu'une collision se produise pour déterminer le logarithme discret cherché.

Lorsque une collision se produit :

$$\begin{aligned} R_1 = R_2 &\Leftrightarrow u_1.P + v_1.Q = u_2.P + v_2.Q \\ &\Leftrightarrow (u_1 + \alpha v_1).P = (u_2 + \alpha v_2).P \\ &\Leftrightarrow u_1 + \alpha v_1 \equiv u_2 + \alpha v_2 \ (\bmod r) \\ &\Leftrightarrow u_1 - u_2 \equiv \alpha(v_2 - v_1) \ (\bmod r) \end{aligned}$$

Si $pgcd(r, v_2 - v_1) = 1$, c'est fini, sinon, nous obtenons $pgcd(r, v_2 - v_1)$ possibilités pour α, nous pouvons alors trouver la bonne par une recherche exhaustive. Au final, en considérant que la fonction f est aléatoire, le nombre d'itérations nécessaires pour calculer le logarithme discret est en moyenne de l'ordre $\frac{3}{2}\sqrt{\pi r/8} \approx 0.94\sqrt{r}$, c'est à dire $2.82\sqrt{r}$ appels à la fonction f. Étant donné que tout au long de l'algorithme, seuls deux points et leurs décompositions sont stockés, la complexité en espace n'est pas importante, elle est en $O(1)$. La complexité de la méthode Pollard-ρ est donc en $O(\sqrt{r})$.

Il est intéressant de noter que l'algorithme de Floyd peut être optimisé. Il existe d'autres algorithmes de recherche de cycles pour lesquels la complexité en espace reste faible avec un nombre d'appels à f inférieur, néanmoins pas plus de deux fois plus petit. Pour plus de détails, nous renvoyons le lecteur à [CF06, Chapitre 19].

I.4.2 Calcul d'index

Cette méthode est la plus efficace pour calculer le logarithme discret dans un corps fini. Soient $\mathbb{G} \subset \mathbb{F}_q^\star$ un sous-groupe multiplicatif d'ordre r de $\mathbb{F}_q$ et g un générateur de $\mathbb{G}$. Soit x un élément de $\mathbb{G}$ tel que $x = g^\alpha$, pour $\alpha \in [1, r-1]$. La résolution du logarithme discret par la méthode de calcul d'index est la résolution la plus efficace connue à ce jour pour les corps finis. Elle est en complexité sous exponentielle, s'exprimant en fonction de l'argument q (dans notre cas, cet argument est l'ordre du groupe) par la fonction :

$$L_r(e, c) = \exp\left(c(\log r)^e(\log(\log(r)))^{1-e}\right),$$

où $e \in [0, 1]$ et $c > 0$ est une constante.

Nous décrivons la méthode du calcul d'index pour le sous groupe multiplicatif d'un corps fini de la forme $\mathbb{F}_p^\star$. L'ordre r du sous groupe est alors $(p-1)$. Pour des corps plus généraux de la forme $\mathbb{F}_{p^k}$ avec $k > 1$, nous renvoyons le lecteur à l'article [JL07] qui décrit les différents algorithmes de type calcul d'index.

Soit n un entier, nous commençons par construire une base de facteurs premiers $B = \{p_1, p_2, \ldots, p_n\}$ de $\mathbb{F}_p$. Cette base est librement choisie parmi les petits nombres premiers de $\mathbb{F}_p$, la seule condition est qu'ils doivent être deux à deux différents.

Nous tirons ensuite aléatoirement des entiers h, et calculons $g^h \bmod(p)$. Nous testons si la décomposition en facteurs premiers du résultat admet uniquement des éléments premiers de notre base B. Si c'est le cas, nous stockons la relation :

$$g^h \equiv \prod_i p_i^{\alpha_i} \bmod(p),$$

qui équivaut en fait à la relation :

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

où, nous connaissons h et les α_i et nous cherchons les $\log_g(p_i)$. Nous réitérons ce procédé jusqu'à obtenir n relations indépendantes. Ces n relations forment un système linéaire en les valeurs $\log_g(p_i)$ pour $i = \{1, \ldots, n\}$. La résolution de ce système nous permet d'obtenir le logarithme discret des éléments de la base B.

Pour terminer, nous cherchons aléatoirement un entier l vérifiant une dernière relation de la forme :

$$g^l x \equiv \prod_i p_i^{\alpha_i} \bmod(r),$$

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

$$\alpha \equiv \sum_i \alpha_i \times \log_g(p_i) - l \bmod(r).$$

L'algorithme que nous venons de décrire a une complexité en $L_r(1/2, c)$ [JL07]. Pour les corps finis, les meilleurs résultats sont en $L_r(1/3, c)$ [EGT09]. Le paramètre influençant la complexité des algorithmes de calcul d'index est la constante e (ici 1/2). La valeur de e doit être la plus petite possible, l'influence de la constante c est moins importante.

I.5 Construction de courbes par multiplication complexe

Nous rappelons ici les méthodes utilisées pour construire des courbe elliptiques ordinaires adaptées au calcul des couplages. Les courbes elliptiques dont nous aurons besoin doivent présenter des caractéristiques bien particulières. Le cardinal $\#E(\mathbb{F}_p)$ de la courbe définie sur le corps $\mathbb{F}_p$ doit posséder un grand facteur premier, que nous noterons r et le degré de plongement k doit être tel que $6 < k \leq 32$. Pour obtenir des courbes satisfaisants ces conditions, la méthode la plus efficace est d'utiliser la théorie de la Multiplication Complexe. Nous renvoyons le lecteur au chapitre 10 de [Was03] pour l'aspect théorique de cette approche.

La construction d'une courbe elliptique en utilisant la méthode multiplication complexe se décompose en deux étapes. La première est la résolution du système d'équation (I.5.1), pour lequel les inconnues sont le degré de plongement de la courbe k, le facteur premier r, t la trace du Frobenius sur $E(\mathbb{F}_p)$, p la caractéristique du corps fini et un entier D dont les propriétés arithmétiques déterminent si la courbe construite est une courbe ordinaire ou supersingulière ; cet entier est appelé le discriminant de la multiplication complexe [FST06]. La seconde est la construction d'une courbe elliptique

admettant les paramètres p, r, et t trouvés en utilisant les techniques de multiplication complexe. Le système (I.5.1) est obtenu en écrivant les différentes relations que nous souhaitons voir vérifiées par les paramètres de la courbe elliptique.

$$\begin{cases} r & | & p+t-1, \\ r & | & p^k-1, \text{ pour des premiers } r,p, \\ Dy^2 & = & 4p-t^2 \text{ pour un entier donné } D. \end{cases} \tag{I.5.1}$$

Différentes méthodes permettent de résoudre ce système, un état de l'art de celles-ci est donné dans [FST06]. En général, les méthodes de résolutions fixent les valeurs de deux paramètres sur les quatre. Les deux paramètres restants sont déduits des équations et des conditions qu'ils doivent satisfaire.

Nous rappelons deux méthodes de construction de courbe par multiplication complexe fréquemment utilisées en cryptographie à base de couplage. La première permet de construire des courbes avec un degré de plongement k fixé à l'avance. La seconde méthode permet de construire des courbes pour lesquelles la valeur du degré de plongement n'est pas connue à l'avance.

- La méthode Miyaji-Nakabayashi-Takano (notée MNT) est l'une des premières méthodes de multiplication complexe a avoir été mise en place pour construire des courbes elliptiques adaptées aux attentes des cryptographes [MNT01]. La méthode MNT fût généralisée par Barreto et Naehrig [BN06] pour construire des courbes de degré 12. Les courbes sont obtenues à partir de la paramétrisation suivante :

$$\begin{aligned} k &= 12, \\ p &= 36x^4+36x^3+24x^2+6x+1, \\ r &= 36x^4+36x^3+18x^2+6x+1, \\ t &= 6x^2+1. \end{aligned}$$

  Pour déterminer les paramètres des courbes elliptiques, nous cherchons des entiers x tels que les valeurs de p et r soient des nombres premiers. L'équation de ces courbes est de la forme $E : y^2 = x^3 + b$, pour b non nul dans $\mathbb{F}_p$.

- La seconde méthode permet de construire des courbes de degré de plongement k arbitraire, il s'agit de la méthode Cocks-Pinch [CP01]. Cette méthode génère des courbes pour lesquelles la valeur de r est arbitraire, tout en satisfaisant l'approximation $\log(p)/\log(r) \approx 2$. La méthode Cocks-Pinch fut aussi généralisée pour des valeurs de $\log(p)/\log(r)$ inférieures à 2 par Brezing et Weng dans [BW05]. Par exemple, une

famille de courbes de degré 16 a été générée par la méthode Brezing-Weng, sa paramétrisation est donnée ci-après.

$$\begin{aligned} k &= 16, \text{ pour } x \equiv \pm 25 \bmod(70) \\ p &= (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 \\ &\quad +625x^2 + 2398x + 3125)/980, \\ r &= (x^8 + 4x^4 + 625)/61250, \\ t &= (2x^5 + 41x + 35)/35. \end{aligned}$$

Il est important de souligner qu'avec ces méthodes de construction de courbes le nombre premier p n'est pas connu à l'avance.

I.6 Fonctions rationnelles et diviseurs

Nous venons d'introduire les éléments de la théorie des courbes elliptiques nécessaires à la compréhension des couplages. Nous présentons dans cette partie les prérequis mathématiques nécessaires à la construction et la définition des couplages. Nous introduisons succinctement la notion de fonction rationnelle et de diviseur. Les théories plus complètes se trouvent dans [Sil92, Chap. 2 §1 et §2] pour les fonctions rationnelles, et [Sil92, Chap. 2 §3] pour les diviseurs.

I.6.1 Fonction rationnelle

Nous travaillons sur des courbes elliptiques $E(\mathbb{F}_q)$, pour $q = p^m$, avec p premier supérieur strictement à trois et $m \in \mathbb{N}^\star$. Nous commençons par présenter les fonctions rationnelles sur $E(\mathbb{F}_q)$:

Définition I.6.1. Une fonction rationnelle sur $E(\mathbb{F}_q)$ est une fonction f qui à tout point P de $E(\mathbb{F}_q)$ associe l'entier $f(P) = \frac{u(P)}{v(P)}$, avec u et v deux polynômes de $\mathbb{F}_q[X, Y]$ évalués au point P.

Afin d'illustrer les définitions et propriétés, nous nous servirons de l'exemple suivant.

***Exemple* I.6.2.** Soit E la courbe définie sur $\mathbb{F}_{257}$ d'équation $y^2 = x^3 + 1$, nous définissons la fonction rationnelle suivante :

$$\begin{aligned} f : E(\mathbb{F}_{257}) &\to \mathbb{F}_{257} \\ P = (x_P, y_P) &\to \tfrac{(x_P-8)^2(3-x_P)}{(y_P-16)(19-x_P)^3}. \end{aligned}$$

Nous avons donc $u(x_P, y_P) = (x_P - 8)^2(3 - x_P)$ et $v(x_P, y_P) = (y_P - 16)(19 - x_P)^3$.

Définition I.6.3. Soit P un point d'une courbe elliptique $E(\mathbb{F}_q)$ et f une fonction rationnelle. P est un **zéro** de f si $f(P) = 0$, et P est **pôle** de f si $u(P) \neq 0$ et $v(P) = 0$ n'est pas défini.

Nous avons besoin de définir l'ordre d'une fonction rationnelle en un point. Nous commençons par donner une définition formelle faisant intervenir la notion d'uniformisante en un point [Sil92, Chap. 2 §2].

Définition I.6.4. Soit P un point d'une courbe elliptique $E(\mathbb{F}_q)$. Il existe une fonction u_P appelée **l'uniformisante en** P, s'annulant en P et telle que toute fonction rationnelle f puisse s'écrire $f = u_P^{\alpha} \times g$, pour α un entier naturel et où g est une fonction rationnelle telle que $g(P) \neq 0, \infty$.
L'ordre de f **en** P est défini par $ord_f(P) = \alpha$.

***Remarque* I.6.5.** L'ordre de f au point P_∞ n'est pas facile à exprimer par la notion d'uniformisante. Pour les courbes elliptiques d'équation $y^2 = x^3 + ax + b$, une uniformisante au point P_∞ est $u_\infty = \frac{x}{y}$. [Was03, Chap. 11]

Afin de fixer les idées, nous donnons en remarque une méthode possible pour déterminer l'ordre en un point d'une fonction rationnelle, en commençant par introduire l'ordre d'un polynôme en un point.

***Remarque* I.6.6.**

- L'**ordre d'un polynôme** u **en un point** P **:** $ord_P(u)$ est le degré du polynôme u_0 de plus petit degré permettant d'écrire la décomposition $u = u_0 \times u_1$, avec $u_0(P) = 0$ et $u_1(P) \neq 0$.

  A une fonction $f = \frac{u}{v}$ rationnelle sur E et un point $P \in E$ nous associons l'**ordre de la fonction** f **en un point** $P \in E$ **:**

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

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

  Si $ord_P(f) > 0$, le point P est un **zéro** pour la fonction f.

  L'ordre de la fonction f en le point à l'infini P_∞ est

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

Nous illustrons ces définitions à l'aide de la fonction rationnelle introduite dans l'exemple I.6.2.

***Exemple* I.6.7.** Soit f la fonction rationnelle définie sur $E(\mathbb{F}_{257}) : y^2 = x^3+1$ par $f(x,y) = \frac{(x-8)^2(3-x)}{(y-16)(19-x)^3}$.

Soit $P = (8,16)$, $Q = (19,101)$ et $R = (20,169)$ des points de la courbe elliptique $E(\mathbb{F}_{257})$. Nous cherchons l'ordre de la fonction f en ces points. $ord_P(u) = 2$ car le point P annule le polynôme u avec une multiplicité de degré deux. En effet, $u = u_0 \times u_1$, avec $u_0 = (x_P - 8)^2$ et $u_1 = (3 - x_P)$. $ord_P(v) = 1$ car $(y-16)$ s'annule en P avec une multiplicité de 1, et $(19-x)^3$ ne s'annule pas en P. A partir de ces deux ordres, nous pouvons calculer l'ordre de f en P : $ord_P(f) = 2 - 1 = 1$.

Le cas du point Q se traite de la même façon : $ord_Q(u) = 0$ car $u(Q) = 121 \neq 0$. $ord_Q(v) = 3$, car le polynôme $(19 - x)^3$ s'annule au point Q et comme c'est un polynôme de degré 3, la multiplicité au point Q est 3 ; $(y - 16)$ ne s'annule pas en Q. Comme R n'est un zéro ni pour u ni pour v, nous pouvons calculer l'ordre de la fonction f en Q et R :

$$\begin{aligned} ord_Q(f) &= 0 - 3 = -3 \\ ord_R(f) &= 0 - 0 = 0 \end{aligned}$$

La fonction f admet ainsi P comme zéro de multiplicité 1, et Q est un pôle de multiplicité 3. Nous en déduisons que $f(P) = 0$, et que $f(Q)$ n'est pas défini. Nous pouvons calculer $f(R) = 120$. Les zéro de u sont réduits au point d'abscisse 8, en effet la courbe $E(\mathbb{F}_{257}) : y^2 = x^3 + 1$ n'admet pas de point dont l'abscisse est 3. Ils s'agit des points P et $-P$. Les zéros de v sont les points P et $-P$ avec une multiplicité de 1 et Q et $-Q$ avec une multiplicité 3. Nous pouvons calculer l'ordre de f en le point P_∞ :

$$ord_{P_\infty}(f) = 8 - 4 = 4.$$

I.6.2 Diviseurs

La définition et la construction des couplages reposent essentiellement sur l'existence de diviseurs associés aux points d'une courbe elliptique. Nous donnons les définitions et les propriétés caractéristiques des diviseurs.

Définition I.6.8. Soit E une courbe elliptique, un diviseur de E est une somme formelle de points

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

pour laquelle un nombre fini de n_P sont non nuls.
A un diviseur, nous associons :

- **son support :** $supp(D) = \{P \in E, n_P \neq 0\}$,
- **son ordre en un point** P **:** $ord_P(D) = n_P$,
- **son degré :** $deg(D) = \sum_{P \in E} n_P$.

L'ensemble des diviseurs de degré zéro est noté D_0.

***Remarque* I.6.9.** Afin de 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, $n_P\ (P)$ sera le diviseur représentant n_P fois le point P ; tandis que nous noterons $[n_P]P$ le point obtenu en ajoutant n_P fois le point P sur la courbe elliptique.

Propriété I.6.10. *L'ensemble des diviseurs est un groupe additif pour la loi d'addition suivante. Soit* $D = \sum_{P \in E, n_P \in \mathbb{Z}} n_P(P)$ *et* $D' = \sum_{P \in E, n'_P \in \mathbb{Z}} n'_P(P)$ *deux diviseurs d'une même courbe* E*, alors :*

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

***Remarque* I.6.11.** Dans la Propriété I.6.10, la somme $(n_P + n'_P)$ est une somme d'entiers, alors que la somme globale est une somme formelle de points.

Les diviseurs dont nous aurons besoin pour définir et calculer des couplages sont les diviseurs associés à une fonction rationnelle. Nous savons qu'à toute fonction rationnelle f sur E, nous pouvons associer un diviseur de E noté $Div(f)$.

Définition I.6.12. Soit f une fonction rationnelle sur E, le diviseur $Div(f)$ est construit de la manière suivante :

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

Un **diviseur principal** est un diviseur D tel qu'il existe une fonction rationnelle f vérifiant $D = Div(f)$. Nous notons D_{prin} leur ensemble.

Nous illustrons cette définition à l'aide de la fonction rationnelle introduite dans l'exemple I.6.2 :

***Exemple* I.6.13.** Soit f la fonction rationnelle définie sur $E(\mathbb{F}_{257}) : y^2 = x^3 + 1$ par $f(x, y) = \frac{(x-8)^2(3-x)}{(y-16)(19-x)^3}$.

Cette fonction admet $P = (8, 16)$ et $-P = (8, 241)$ comme zéros de multiplicité 1 et $Q = (19, 101)$, $-Q = (19, 156)$ comme pôles de multiplicité 3 ; l'ordre de f en P_∞ est 4.
Nous pouvons construire le diviseur associée à la fonction f :

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

Nous pouvons construire les relations algébriques suivantes pour les diviseurs de fonctions rationnelles.

Propriété I.6.14. *Soient f_1 et f_2 deux fonctions rationnelles, alors*

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

et

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

Propriété I.6.15. *Un diviseur $D = \sum n_P(P)$ est principal si et seulement si $deg(D) = 0$ et $\sum_{n_P \in \mathbb{Z}} [n_P]P = P_\infty$.*

Nous pouvons vérifier que le diviseur associé à la fonction rationnelle de l'exemple I.6.2 vérifie bien cette propriété. Cette dernière nous permet de construire un isomorphisme entre la courbe elliptique et l'ensemble des diviseurs de degré zéro quotienté par les diviseurs principaux [Sil92, Chap. 3 Prop. 3.4].

Corollaire I.6.16. *Nous savons que*

$$E \cong D_0/D_{prin}.$$

***Remarque* I.6.17.** Il n'y a pas unicité de la fonction rationnelle f associée à un diviseur, en effet la fonction rationnelle associée à un diviseur est définie à multiplication par une constante près.

I.7 Définition et propriétés des couplages

I.7.1 Introduction aux couplages

Les couplages sont une notion mathématique qui apparaît dans les années 90 en cryptographie. Les premiers introduits en cryptographie sont les couplages de Weil et de Tate. Cette utilisation était destructice, elle consistait à attaquer le logarithme discret sur une courbe elliptique. En effet, nous avons vu dans la partie I.4 que le logarithme discret sur une courbe elliptique est plus difficile à résoudre que le logarithme discret sur un corps fini.

Or, une des propriétés des couplages, la bilinéarité, permet de transporter ce problème sur un corps fini. La première attaque de ce type est l'attaque MOV [MOV91] du nom de ses auteurs Menezes, Okamoto et Vanstone, elle utilisait le couplage de Weil. L'attaque Frey Rück [FR94] est construite sur le même principe et utilise celui de Tate. Ensuite, la propriété de bilinéarité des couplages a permis la construction de protocoles originaux et l'amélioration des protocoles existants. Citons par exemple l'échange de clé tri-partite à la Diffie Hellman [Jou00], la cryptographie basée sur l'identité [BF01] ou encore des schémas de signatures courtes [BLS01]. Nous introduisons dans cette partie leurs définitions et principales propriétés, qui reposent sur les notions de diviseurs et fonctions rationnelles que nous venons d'introduire [BSS05]. Dans les parties I.8 et I.9 nous expliquons comment calculer un couplage, et quelles sont les dernières optimisations théoriques.

I.7.2 Généralités sur les couplages

Définition I.7.1. Soient $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_3$ trois groupes abéliens de même ordre r. Les groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ sont additifs, $\mathbb{G}_3$ est un groupe multiplicatif.

Un **couplage** est une application **bilinéaire** et **non dégénérée** notée e du produit cartésien de $\mathbb{G}_1$ et $\mathbb{G}_2$ à valeur dans $\mathbb{G}_3$:

$$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$$

Un **couplage symétrique** est un couplage pour lequel $\mathbb{G}_1 = \mathbb{G}_2$.

Nous rappelons la définition de la non-dégénérescence et de la bilinéarité dans la propriété suivante.

Propriété I.7.2. *Un couplage doit vérifier deux propriétés essentielles, la non dégénérescence et la bilinéarité :*

- *Non dégénérescence :*

$$\forall P \in \mathbb{G}_1,\ \exists Q \in \mathbb{G}_2 \text{ tel que } e(P,Q) \neq 1.$$

$$\forall Q \in \mathbb{G}_2,\quad \exists P \in \mathbb{G}_1 \text{ tel que } e(P,Q) \neq 1.$$

- *Bilinéarité :* $\forall P, P' \in \mathbb{G}_1, \forall Q, Q' \in \mathbb{G}_2$:

$$e(P + P', Q) = e(P,Q) \times e(P',Q),$$

$$e(P, Q + Q') = e(P,Q) \times e(P,Q').$$

Corollaire I.7.3. *Soient e un couplage, $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$. Nous notons P^1_∞ l'élément neutre de $\mathbb{G}_1$ et P^2_∞ celui de $\mathbb{G}_2$.*

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

I.7.3 Protocoles cryptographiques basés sur les couplages

Nous présentons dans cette partie les utilisations des couplages en cryptographie à base de courbes elliptiques.

Cryptanalyse à base de couplage

La cryptanalyse s'oppose, en quelque sorte, à la cryptographie. Cryptanalyser un protocole consiste à être capable de déchiffrer un message sans en être le destinataire, donc a priori sans connaître la clé de déchiffrement. La recherche de la clé permettant de déchiffrer un message codé par un protocole cryptographique est une méthode de cryptanalyse. Soit $E(\mathbb{F}_p)$ une courbe elliptique, P et Q deux points appartenant au même sous groupe $\mathbb{G}_1$ d'ordre premier r de $E(\mathbb{F}_p)$. Il existe un entier α tel que $Q = \alpha P$. Le problème du logarithme discret sur la courbe elliptique consiste à retrouver α avec la donnée de P et Q. Nous posons $\mathbb{G}_3 = \mathbb{F}^\star_{p^k}$ afin de construire un couplage symétrique $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_3$. En calculant $e(P, Q)$, nous transposons le problème du logarithme discret sur la courbe elliptique, sur un corps fini. En effet, la propriété de bilinéarité du couplage e implique :

$$\begin{aligned} e(P, Q) &= e(P, \alpha P) \\ &= e(P, P)^\alpha. \end{aligned}$$

Notons $g = e(P, P)$ et $h = e(P, Q)$. A condition que $e(P, P) \neq 1$, nous nous retrouvons avec le problème du logarithme discret consistant à retrouver α connaissant g et $h = g^\alpha$. Ce problème du logarithme discret dans le corps fini $\mathbb{F}_{p^k}$ peut se résoudre à l'aide d'algorithmes plus efficaces en $O(\exp\,(c(\log r)^e(\log(\log(r)))^{1-e}))$ que les algorithmes génériques en $O(\sqrt{r})$. L'attaque MOV [MOV91] repose sur ce schéma et utilise le couplage de Weil.

Cryptographie à base de couplage

Plus récemment, les couplages ont permis la création et la réalisation de protocoles cryptographiques originaux. En 2000, A. Joux [Jou00] propose

un schéma d'échange de clé tri-partite à la Diffie Hellman utilisant la propriété de bilinéarité des couplages. Ce fut le premier d'une longue série d'articles proposant des protocoles cryptographiques asymétriques basés sur les couplages. Depuis, de nombreux articles proposent des protocoles utilisant les couplages, par exemple [BLS01, CP01], d'autres améliorent les calculs [Sco05, BCF+05] et certains proposent des réalisations d'architectures matérielles pour le calcul des couplages [BDE+09, BBS+07]. Une utilisation intéressante des couplages est la cryptographie basée sur l'identité. En 2001 dans [BF01], D. Boneh et M. Franklin proposent une solution basée sur les couplages au challenge proposé par Shamir en 1984 [Sha84]. Ce challenge consistait à créer un protocole cryptographique basé sur le principe où la clé publique d'un utilisateur est son identité. La clé privée de l'utilisateur lui est fournie par une autorité de confiance. L'autorité de confiance construit les clés privées à l'aide des clés publiques des utilisateurs et les leur transmet par un canal sécurisé. Ce premier schéma cryptographique basé sur l'identité fut réalisable principalement grâce à la propriété de bilinéarité des couplages. Nous le détaillons dans la partie V.2.2. Afin d'illustrer la simplification d'algorithmes existants en utilisant la cryptographie basée sur les couplages, nous décrivons le protocole d'échange de clé tri-partite de Diffie Hellman [Jou00].

Échange de clé de Diffie Hellman : Afin de décrire le protocole d'échange de clé tripartite de Diffie Hellman, nous introduisons Alice, Bob et Charlie, personnages récurrents des situations cryptographiques qui veulent communiquer de manière sécurisée par un canal non sécurisé. Ils désirent créer une clé secrète commune pour chiffrer et déchiffrer les messages qui transiteront via le canal non sécurisé. Nous commençons par décrire l'échange de clé de Diffie Hellman [DH76] pour un groupe $\mathbb{G}$ cyclique noté additivement. Tout d'abord, Alice, Bob et Charlie se mettent d'accord sur un groupe $\mathbb{G}$ cyclique d'ordre r et de générateur g. Puis, ils procèdent aux étapes suivantes.

- Choix des entiers aléatoires

  Alice choisit $a \in \{1, 2, \ldots, r-1\}$,

  Bob choisit $b \in \{1, 2, \ldots, r-1\}$,

  Charlie choisit $c \in \{1, 2, \ldots, r-1\}$.

- Premier tour d'échange

  Alice envoie $a \times g$ à Bob,

  Bob envoie $b \times g$ à Charlie,

  Charlie envoie $c \times g$ à Alice.

- Deuxième tour d'échange

  Alice envoie $a \times (cg)$ à Bob,

  Bob envoie $b \times (ag)$ à Charlie,

  Charlie envoie $c \times (gb)$ à Alice.

- Création de la clé commune

  chacun des trois protagonistes peut alors calculer la même clé $(abc) \times g$.

L'échange de clé Diffie Hellman nécessite deux tours d'échanges d'information. Nous décrivons ci-dessous le protocole d'Antoine Joux. Soit $\mathbb{G}$ un groupe additif d'ordre r et $\mathbb{G}_3$ un groupe multiplicatif cyclique de même ordre. Nous construisons un couplage symétrique $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_3$. Alice, Bob et Charlie suivent la démarche suivant :

- Choix des entiers aléatoires

  Alice choisit $a \in \{1, 2, \ldots, r-1\}$,

  Bob choisit $b \in \{1, 2, \ldots, r-1\}$,

  Charlie choisit $c \in \{1, 2, \ldots, r-1\}$.

- Premier tour d'échange

  Alice envoie $a \times g$ à Bob et Charlie,

  Bob envoie $b \times g$ à Charlie et Alice,

  Charlie envoie $c \times g$ à Alice et Bob.

- Calcul de la clé commune

  Alice calcule $e(bg, cg)^a$,

  Bob calcule $e(cg, ag)^b$,

  Charlie calcule $e(ag, bg)^c$,

- Par bilinéarité du couplage la clé commune est $e(g, g)^{abc}$.

Ainsi, l'échange de clé tri-partite d'Antoine Joux ne nécessite qu'un échange de données, au lieu de deux pour le protocole original de Diffie Hellman. La simplification réside dans le nombre d'échanges à faire pour construire une clé commune. Cet exemple illustre de manière simple l'apport des couplages à la cryptographie.

I.7.4 La sécurité des couplages

La sécurité d'un protocole cryptographique repose souvent sur la difficulté de résolution du problème du logarithme discret. Le niveau de sécurité est défini par le nombre d'opérations nécessaires sur le corps de base pour réussir une attaque de logarithme discret. Il est donné en bits, et nous utilisons pour le définir la notion de sécurité équivalente. Par exemple, nous parlons d'un niveau de sécurité de 80 bits s'il faut au minimum 2^{80} opérations pour résoudre le problème du logarithme discret sur les groupes en considération.

En cryptographie basée sur les couplages, deux bornes sont à prendre en compte : le nombre de bits $\log_2(r)$ de la taille du sous groupe de la courbe elliptique dans lequel nous travaillons, et le nombre de bits $\log_2(p^k)$ de la taille du corps fini où vivent les résultats d'un couplage. Nous avons vu dans la partie I.4 que la complexité du problème du logarithme discret est différente suivant que nous voulons le résoudre sur la courbe elliptique $E(\mathbb{F}_p)$ ou sur le corps fini $\mathbb{F}_{p^k}$. A taille de corps égale, le logarithme discret se résout plus facilement sur un corps fini que sur une courbe elliptique. La complexité de résolution du logarithme discret dans $\mathbb{F}^\star_{p_k}$ est $O(\exp\,(c(k\log_2(p))^e(\log(k\log_2(p)))^{1-e}))$, celle du logarithme discret sur la courbe est $O(\sqrt{r})$. Par conséquent, lorsque le niveau de sécurité va augmenter les bornes $\log_2(r)$ et $k\log_2(p)$ vont aussi devoir augmenter mais à des rythmes différents. La borne $k\log_2(p)$, dont dépend le nombre d'opérations nécessaires sur le corps fini pour résoudre le logarithme discret, augmentera plus rapidement que la borne $\log_2(r)$ dont dépend la complexité du logarithme discret sur la courbe elliptique. Les paramètres utilisés en cryptographie à base de couplage tiennent compte de cette différence. Dans le Tableau I.1, nous donnons la taille minimal en bits de r et p^k pour travailler à un niveau de sécurité équivalent par exemple au niveau de sécurité du protocole AES. Ces niveaux de sécurité sont les recommandations du NIST [NIS07] en matière de cryptographie à base de couplage.

Niveau de sécurité en bits	80	128	192	256
Nombre minimal de bits de r	160	256	384	512
Nombre minimal de bits de p^k	1 024	3 072	7 680	15 360

TABLE I.1 – Niveau de sécurité

Ici, les groupes sont les corps finis $\mathbb{F}_{p^k}$, et pour assurer une sécurité de 80 bits, il faut que p^k soit de taille 1 024 bits. Les tailles des paramètres sont fixées les unes en fonction des autres. La taille de p est en pratique égale à $\rho \times r$. Une fois ces deux tailles fixées, nous en déduisons k.

De plus, les courbes pour lesquelles le rapport

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

est proche de 1 sont privilégiées pour les applications cryptographiques [FST06].

En effet, plus ρ est proche de 1, plus les calculs sur le corps fini seront efficaces. Ceci est une conséquence directe du fait que pour un niveau de sécurité fixé (i.e. une taille de r en bits fixées), l'implémentation efficace des couplages dépend directement de la taille du corps de base utilisé, i.e. la taille en bits de p. Par conséquent, prendre des valeurs de k de plus en plus grandes est une solution plus intéressante qu'augmenter la taille de p pour garantir de hauts niveaux de sécurités (soit un niveau de sécurité supérieur ou équivalent à du AES 128 bits). Le tableau suivant est issu de [FST06] et donne les tailles de r, p^k et k pour des niveaux de sécurité fixés, en fonction de ρ. Les tailles sont données en bits.

Niveau de sécurité	Taille de r	Taille de p^k p^k	Degré de plongement $\rho \approx 1$	$\rho \approx 2$
80	160	960-1 280	6 - 8	2-4
128	256	3 000 - 5 000	12 - 20	6 - 10
192	384	8 000 - 10 000	20 - 26	10 - 13
256	512	14 000 - 18 000	28 - 36	14 - 18

TABLE I.2 – Niveau de sécurité

Ce tableau donne les degrés de plongement à considérer pour un niveau de sécurité fixé et suivant la valeur de ρ. Les courbes vérifiant $\rho \approx 2$ ont des degrés de plongement de taille moitié par rapport aux courbes telles que $\rho \approx 1$. Les rapports sont inversés pour la taille des corps finis $\mathbb{F}_p$ sur lesquels la courbe elliptique est définie, pour $\rho \approx 1$ le corps fini sur lequel est défini la courbe elliptique est de taille moitié que le corps pour $\rho \approx 2$. Pour des raisons d'efficacité, il semble préférable de travailler sur des corps de taille moitié, ainsi les valeurs de ρ proches de 1 devraient être privilégiées, et donc des degrés de plongement plus grands.

I.8 Réalisation pratique des couplages

Nous travaillons avec un nombre p de taille fixée pour assurer un niveau de sécurité suffisant à la fois sur la courbe elliptique $E(\mathbb{F}_p)$ et sur le corps fini $\mathbb{F}_{p^k}$. Nous définissons et décrivons les couplages de Weil et de Tate.

I.8.1 Couplage sur des courbes elliptiques

A l'heure actuelle, les couplages sur les courbes elliptiques sont les plus naturels à réaliser. Cette partie présente les données nécessaires pour réaliser un couplage sur une courbe elliptique.

Les premiers couplages apparus en cryptographie sont les couplages de Weil et de Tate. Ces deux couplages prennent en argument deux points d'une courbe elliptique définie sur un corps fini $\mathbb{F}_p$ et renvoient un élément d'une extension $\mathbb{F}_{p^k}$ de ce corps fini. Tout ce qui suit s'adapte aisément au cas des courbes elliptiques sur un corps fini $\mathbb{F}_{p^m}$. Dans le cadre de cette thèse nous travaillerons uniquement sur des couplages définis sur des courbes elliptiques $E(\mathbb{F}_p)$.

Soient $E : y^2 = x^3 + ax + b$ une courbe elliptique définie sur $\mathbb{F}_p$ et r un facteur premier du cardinal de $E(\mathbb{F}_p)$. Afin d'éviter que toute la r-torsion soit dans $E(\mathbb{F}_p)$, nous demandons de plus que r^2 ne divise pas le cardinal de $E(\mathbb{F}_p)$, noté $\#E(\mathbb{F}_p)$. Cette condition est facilement réalisable en pratique. Nous savons que les points de r-torsion de la courbe elliptique vérifient que $E[r] \subset E(\mathbb{F}_{p^k})$, où k est le degré de plongement de la courbe relativement au facteur r. Nous noterons $\mathbb{G}_1 = E(\mathbb{F}_p)[r]$, $\mathbb{G}_2 \subset E(\mathbb{F}_{p^k})[r]$ et $\mathbb{G}_3 = \{\mu \in \mathbb{F}_{p^k}$ tel que $\mu^r = 1$ dans $\mathbb{F}_{p^k}\}$, il s'agit du sous-groupe des racines r-ième de l'unité.

Soient $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$ deux points de la courbe elliptique. A tout point R de $E(\mathbb{F}_{p^e})$, pour e un entier quelconque, et tout entier s, nous associons $f_{s,R}$ une fonction rationnelle vérifiant que $Div(f_{s,R}) = s(R) - ([s]R) - (s-1)(P_\infty)$. Une telle fonction existe d'après la Propriété I.6.15 des diviseurs principaux.

Nous donnons les définitions des couplages de Weil et de Tate adaptées à notre utilisation des couplages.

Définition I.8.1. Le **couplage de Weil** noté e_W est l'application :

$$\begin{array}{rcl} e_W : \mathbb{G}_1 \times \mathbb{G}_2 & \to & \mathbb{G}_3, \\ (P, Q) & \to & (-1)^r \frac{f_{r,P}(Q)}{f_{r,Q}(P)} \end{array}$$

***Remarque* I.8.2.** Une définition plus précise du couplage de Weil se trouve dans le livre [Wei48].

Nous présentons ci-dessous la définition du couplage de Tate réduit. La définition complète du couplage de Tate est donnée dans [BSS05, §IX.5].

Définition I.8.3. Le **couplage de Tate** réduit noté e_T est l'application :

$$\begin{aligned} e_T : \mathbb{G}_1 \times \mathbb{G}_2 &\rightarrow \mathbb{G}_3, \\ (P,Q) &\rightarrow f_{r,P}(Q)^{\frac{p^k-1}{r}} \end{aligned}$$

***Remarque* I.8.4.** Nous venons de définir le couplage de Tate réduit. Le couplage de Tate entre les points P et Q correspond par définition au calcul de $f_{r,P}(Q)$. Les propriétés mathématiques des diviseurs ont pour conséquence que $f_{r,P}(Q)$ n'est pas défini de manière unique. Cette fonction est à valeur dans les classes d'équivalence modulo les puissances de r dans $\mathbb{F}^{\star}_{p^k}$. Elever $f_{r,P}(Q)$ à la puissance $\frac{p^k-1}{r}$ assure l'unicité du couplage de Tate entre P et Q.

Ces deux couplages reposent sur le calcul de fonction rationnelle du type $f_{r,P}$. Le calcul de ces fonctions est effectué par l'algorithme de Miller [Mil86b].

***Remarque* I.8.5.** Les couplages de Weil et de Tate sont bien définis et non dégénérés à condition que le point Q n'appartienne pas au sous groupe de $E[r]$ engendré par le point P. Pour être bien sûr que ce soit le cas, il suffit de considérer que $\mathbb{G}_2$ est l'ensemble des points de r-torsion dans $E(\mathbb{F}_{p^k})$ privé des points de $E(\mathbb{F}_p)$. L'ensemble $\mathbb{G}_2$ n'est alors plus un sous groupe de la courbe elliptique, mais les éléments de $\mathbb{G}_2$ sont d'ordre r, cela suffit pour la définition et le calcul des couplages.

***Remarque* I.8.6.** La condition $Q \notin < P >$ n'est pas remplie a priori lors de l'utilisation de couplages symétriques pour lesquels $\mathbb{G}_1 = \mathbb{G}_2$. Dans ces cas, le couplage est défini à l'aide d'une fonction de distorsion. Une application de distorsion ϕ est une fonction qui transforme un point d'une courbe elliptique en un point de cette même courbe elliptique mais défini sur une extension de corps du corps de base : $\phi : \mathbb{G}_1 \subset E(\mathbb{F}_p) \rightarrow \mathbb{G}_2 \subset E(\mathbb{F}_{p^k})$. Par exemple, e_{Tsym} le couplage de Tate symétrique est défini par :

$$\begin{aligned} e_{Tsym} : \mathbb{G}_1 \times \mathbb{G}_1 &\rightarrow \mathbb{G}_3 \\ (P,Q) &\rightarrow e_{Tsym}(P,Q) = e_T(P,\phi(Q)). \end{aligned}$$

Des exemples sont détaillés dans [BSS05, Chap. IX.13].

I.8.2 Algorithme de Miller

L'algorithme de Miller est un algorithme permettant de calculer les couplages de Weil et de Tate. Il construit la fonction $f_{r,P}$ tout en l'évaluant au point Q, il repose sur l'égalité de Miller [Mil86b] qui est la suivante :

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

où i et j sont des entiers, $l_{[i]P,[j]P}$ désigne l'équation de la droite $([i]P[j]P)$ et $v_{[i+j]P}$ celle de la verticale au point $[i+j]P$. De plus, pour $i = 1$, la fonction $f_{1,P}$ est la fonction constante égale à 1.

Nous pouvons traduire cette égalité en terme de diviseurs :

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

dont nous déduisons :

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

Nous montrons sur un exemple comment cette égalité est utilisée pour construire $f_{r,P}$ tout en l'évaluant au point Q.

***Exemple* I.8.7.** Supposons $r = 7$, nous décrivons ce qui se passe en déroulant les équations pour $f_{7,P}$, où P est un point d'une courbe elliptique donnée. Nous décomposons 7 en le lisant comme si nous voulions calculer une exponentiation rapide d'exposant 7.

- Nous écrivons $7 = 6 + 1$ pour ensuite appliquer l'égalité de Miller :

$\Rightarrow f_{7,P} = f_{6,P} \times f_{1,P} \times \frac{l_{[6]P,P}}{v_{[7]P}}$,

sachant que $f_{1,P} = 1$ nous obtenons alors que

$\Rightarrow f_{7,P} = f_{6,P} \times \frac{l_{[6]P,P}}{v_{[7]P}}$.

- En décomposant 6 en $6 = 3 \times 2$, nous pouvons décomposer la fonction $f_{6,P}$ pour en déduire $f_{7,P}$:

$f_{6,P} = f_{3,P} \times f_{3,P} \times \frac{l_{[3]P,[3]P}}{v_{[6]P}}$,

$f_{6,P} = f_{3,P}^2 \times \frac{l_{[3]P,[3]P}}{v_{[6]P}}$,

$\Rightarrow f_{7,P} = f_{3,P}^2 \times \frac{l_{[3]P,[3]P}}{v_{[6]P}} \times \frac{l_{[6]P,P}}{v_{[7]P}}$.

- Nous recommençons en écrivant que $3 = 2 + 1$:

$f_{3,P} = f_{2,P} \times f_{1,P} \times \frac{l_{[2]P,P}}{v_{[3]P}}$,

$f_{3,P} = f_{2,P} \times \frac{l_{[2]P,P}}{v_{[3]P}}$,

$$\Rightarrow f_{7,P} = \left(f_{2,P} \times \frac{l_{[2]P,P}}{v_{[3]P}}\right)^2 \times \frac{l_{[6]P,P}}{v_{[7]P}}.$$

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

$$\Rightarrow f_{7,P} = \left(\frac{l_{P,P}}{v_{[2]P}} \times \frac{l_{[2]P,P}}{v_{[3]P}}\right)^2 \times \frac{l_{[3]P,[3]P}}{v_{[6]P}} \times \frac{l_{[6]P,P}}{v_{[7]P}}.$$

Après plusieurs applications de l'égalité de Miller, nous avons ramené le calcul de $f_{7,P}$ à de simples évaluations d'équations de droites et de tangentes. Il est intéressant de noter qu'en remontant la succession d'égalité obtenues en commençant par la dernière, le calcul de $f_{7,P}$ s'effectue en parallèle du calcul de celui de $[7]P$.

L'égalité de Miller permet de calculer la fonction $f_{r,P}(Q)$ par un procédé itératif construit sur le schéma de l'exponentiation rapide à base de doublement et d'addition.

Ainsi, reprenant le principe de l'exponentiation rapide basée sur l'écriture binaire de r, l'évaluation au point Q de la fonction $f_{r,P}$ s'effectue en $\log_2(r)$ étapes. Pour chaque étape, il suffit d'évaluer deux équations de droites au point Q. La méthode de calcul est résumée dans l'algorithme de Miller ci-après.

Algorithme 2 : Miller(P, Q, r)

Données : $r = (r_n \dots r_0)$(représentation binaire), $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$;
Résultat : $f_{r,P}(Q) \in \mathbb{G}_3 (\subset \mathbb{F}^{\star}_{p^k})$;
$T \leftarrow P$
$f_1 \leftarrow 1$
pour $i = n-1$ **to** 0 **faire**
1 $T \leftarrow [2]T$
$f_1 \longleftarrow {f_1}^2 \times \frac{l_1(Q)}{v_1(Q)}$
où l_1 est l'équation de la tangente au point T
et v_1 est l'équation de la verticale au point $[2]T$
2 **si** $n_i = 1$ **alors**
$T \leftarrow T + P$
$f_1 \longleftarrow f_1 \times \frac{l_2(Q)}{v_2(Q)}$
où l_2 est l'équation de la droite (TP)
et v_2 est l'équation de la verticale au point $T + P$
fin
retourner f_1
fin

La complexité de l'algorithme de Miller est liée à celle des fonctions l_1, l_2, v_1 et v_2. Leurs complexités dépendent du système de coordonnées choisi,

elles seront étudiées en détail dans la partie II.4.

Le couplage de Weil requiert deux exécutions de l'algorithme de Miller plus une inversion et une multiplication dans $\mathbb{F}_{p^k}$; tandis que le couplage de Tate ne nécessite qu'une seule exécution de l'Algorithme 2 suivie par une exponentiation. Nous appelons Miller Lite le calcul de la fonction $f_{r,P}(Q)$, et Miller Full le calcul de $f_{r,Q}(P)$. En effet, durant le calcul de $f_{r,P}(Q)$, l'arithmétique de la courbe elliptique est effectuée dans $\mathbb{F}_p$; et nous évaluons des équations de droites à coefficients dans $\mathbb{F}_p$ en des éléments de $\mathbb{F}_{p^k}$. Le calcul de Miller Full est l'inverse, les doublements et additions de points sur la courbe utilisent des opérations dans $\mathbb{F}_{p^k}$, et nous évaluons des équations de droites à coefficients dans $\mathbb{F}_{p^k}$ en des éléments de $\mathbb{F}_p$. Nous étudierons dans la partie II.2, les complexités des opérations de Miller Lite et de Miller Full. Nous verrons que l'opération Miller Full est plus coûteuse qu'une opération Miller Lite. Ce résultat est prévisible : l'arithmétique dans $\mathbb{F}_{p^k}$ est plus lourde que celle de $\mathbb{F}_p$.

Nous nous intéressons à la complexité de ces calculs dans les chapitres suivants. Nous présenterons les différentes optimisations du calcul des couplages qui existent. Ci-dessous, nous présentons la dernière optimisation théorique des couplages. Il s'agit d'un nouveau couplage, le couplage Ate, qui a été introduit par Hess, Smart et Vercauteren dans l'article [HSV06].

I.9 Les couplages Ate et Twisted Ate

Le couplage Ate est une version améliorée du couplage de Tate, il est basé sur l'algorithme de Miller. L'avantage principal par rapport au couplage de Tate est la réduction du nombre d'itérations faites lors de l'algorithme de Miller.

Pour le calcul du couplage de Tate nous faisons $\log_2(r)$ itérations pendant l'algorithme de Miller, où r est l'ordre des sous groupes utilisés.

Dans l'article [HSV06], Hess, Smart et Vercauteren ont utilisé les propriétés des diviseurs pour réduire le nombre d'itérations nécessaires pour l'algorithme de Miller. Le nombre d'itérations passe de $\log_2(r)$ à l'entier $\log_2(T)$ où $T = t - 1$, et t est la trace du Frobenius sur $E(\mathbb{F}_p)$. Le couplage Twisted Ate est une amélioration du couplage Ate que nous décrivons dans la partie I.9.3.

Nous savons d'après le Théorème de Hasse que la trace du Frobenius sur $E(\mathbb{F}_p)$ est comprise entre $-2\sqrt{p}$ et $2\sqrt{p}$. En supposant que nous travaillons avec des courbes $\rho \approx 1$, alors $\log_2(r) \approx \log_2(p)$, nous en déduisons que la trace du Frobenius est telle que que $\log_2(t) \leq \frac{\log_2(r)}{2}$, la décomposition binaire de t est au moins moitié plus petite que celle de r. Réussir à transformer le

couplage de Tate pour ne plus avoir à faire que la moitié des itérations est une optimisation du temps de calcul non négligeable. Nous pouvons augmenter le gain en temps de calcul en construisant les courbes elliptiques par multiplication complexe. En effet, nous pouvons jouer sur les paramètres des courbes et les construire telles que la taille de t soit la plus petite possible. Les méthodes de construction de courbes par multiplication complexe ont été décrites dans la partie I.5. En utilisant la multiplication complexe pour construire des courbes avec une petite trace, même pour des courbes telles que $\rho \approx 2$, le couplage Ate nécessiterait moins d'itérations que le couplage de Tate. Nous présentons dans cette partie les couplages Ate et Twisted Ate respectivement dans les parties I.9.2 et I.9.3. La partie ci-dessous présente les représentations alternatives de $\mathbb{G}_1$ et $\mathbb{G}_2$ permettant de construire les couplages Ate et Twisted Ate.

I.9.1 Représentation alternative de $\mathbb{G}_1$ et $\mathbb{G}_2$

Soit p un nombre premier, E une courbe elliptique définie sur $\mathbb{F}_p$ et π_p le Frobenius sur la courbe elliptique. Soit r un diviseur premier du cardinal de E suffisamment grand pour assurer la sécurité du logarithme discret sur $E[r]$, et tel que r^2 ne divise pas $\#E(\mathbb{F}_p)$; et k le degré de plongement de la courbe. Nous notons $[\delta]$ la multiplication scalaire par $\delta \in \mathbb{N}$ sur E.

La construction du couplage Ate utilise une définition plus fine des groupes $\mathbb{G}_1$ et $\mathbb{G}_2$. Cette définition fait intervenir les espaces propres du Frobenius $\mathrm{Ker}(\pi_p - [1])$ et $\mathrm{Ker}(\pi_p - [p])$:

- $\mathbb{G}_1 := E[r] \cap \mathrm{Ker}(\pi_p - [1])$.
- $\mathbb{G}_2 := E[r] \cap \mathrm{Ker}(\pi_p - [p])$.

Nous décrivons ci dessous une construction possible des points dans les sous groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ de $E[r]$.

Le sous groupe $\mathbb{G}_1$

Les éléments de $\mathbb{G}_1$ sont les points P de $E[r]$ vecteurs propres du Frobenius pour la valeur propre 1, ce qui se traduit par les égalités :

$$[r]P = P_\infty \qquad \text{et} \qquad \pi_p(P) = P.$$

Or, nous savons que le Frobenius induit l'identité sur $E(\mathbb{F}_p)$. En effet, d'après le petit théorème de Fermat, pour tout $x \in \mathbb{F}_p$, $\pi_p(x) = x$. Par conséquent, pour tout point $P = (x_P, y_P)$ de $E(\mathbb{F}_p)$ nous avons l'égalité suivante :

$$\pi_p(P) = (\pi_p(x_P), \pi_p(y_P)) = (x_P, y_P) = P,$$

car x_P et y_P sont tous deux éléments de $\mathbb{F}_p$.

Les vecteurs propres du Frobenius pour la valeur propre 1 sont dans $E(\mathbb{F}_p)$, le groupe $\mathbb{G}_1$ est alors l'ensemble des points de r-torsion défini sur $E(\mathbb{F}_p)$, $\mathbb{G}_1 = E(\mathbb{F}_p)[r]$.

Pour trouver des points de $\mathbb{G}_1$ nous pouvons suivre la démarche suivante :

- Choisir un point P dans $E(\mathbb{F}_p)$
- Calculer $P' = [\frac{\#E(\mathbb{F}_p)}{r}]P$, P' est un point de r-torsion

  Si $P' = P_\infty$ recommencer avec un autre point P,

  Si $P' \neq P_\infty$, alors P' est un générateur de $\mathbb{G}_1$.

Une fois trouvé le point P', les éléments de $\mathbb{G}_1$ seront des multiples de ce point.

Le sous groupe $\mathbb{G}_2$

Les éléments de $\mathbb{G}_2$ sont les points Q de $E[r]$ et vecteurs propres du Frobenius pour la valeur propre p, ce qui se traduit par :

$$[r]Q = P_\infty \qquad \text{et} \qquad \pi_p(Q) = [p]Q.$$

Trouver un point Q vérifiant que $\pi_p(Q) = [p]Q$ est moins évident que de trouver un vecteur propre du Frobenius pour la valeur propre 1. Nous commençons par nous donner un point de $E(\mathbb{F}_{p^k})$ d'ordre r, puis nous nous servirons de ce point pour construire des vecteurs propres pour la valeur propre p de π_p.

Pour trouver un élément d'ordre r de $E(\mathbb{F}_{p^k})$, nous appliquons la même méthode que pour $\mathbb{G}_1$.

- Choisir un point Q dans $E(\mathbb{F}_{p^k})$.
- Calculer $Q' = [\frac{\#E(\mathbb{F}_{p^k})}{r}]Q$, Q' est un point de r-torsion

(a) Si $Q' \in E(\mathbb{F}_p)$ ou $Q' = P_\infty$ recommencer avec un autre point Q.

(b) Si Q' est un élément de $E(\mathbb{F}_{p^k})$ mais pas de $E(\mathbb{F}_p)$, alors Q' est un point de r-torsion sur $E(\mathbb{F}_{p^k})$ privé de $E(\mathbb{F}_p)$.

Avec la donnée des points P' et Q' trouvés précédemment, nous disposons d'une base (P_1, P_2) de $E[r]$, en posant $P_1 = P'$ et $P_2 = Q'$. Nous pouvons donc construire des points de $E[r]$ valeur propre de p pour π_p en utilisant la base (P_1, P_2) de $E[r]$.

La matrice du Frobenius dans cette base est une matrice de la forme :

$$\begin{bmatrix} 1 & \alpha \\ 0 & p \end{bmatrix}.$$

La première colonne découle du fait que par construction $\pi_p(P_1) = P_1$. L'image de P_2 par le Frobenius est toujours un point d'ordre r de E, il admet donc une décomposition de la forme $\pi_p(P_2) = \alpha P_1 + \theta P_2$, pour α et θ deux entiers dans $\mathbb{F}_p$. Or, d'après le théorème de Hasse, nous savons que $\#E(\mathbb{F}_p) = p+1-t$ où t est la trace du Frobenius, soit que $p+1 \equiv t \bmod(\#E(\mathbb{F}_p))$. Nous pouvons en déduire l'égalité suivante :

$$p + 1 \equiv t \bmod(r).$$

La trace d'une application étant indépendante de la base choisie, nous pouvons déduire de cette égalité que $\theta + 1 = p + 1 \bmod(r)$, c'est à dire que $\theta \equiv p \bmod(r)$. Ce qui justifie l'écriture de la matrice du Frobenius.

Pour déterminer entièrement la matrice nous devons trouver la valeur de α. Nous connaissons les points P_1 et P_2, et nous pouvons calculer $\pi_p(P_2)$. A partir de l'égalité $\pi_p(P_2) = [\alpha]P_1 + [p]P_2$, nous pouvons calculer $[\alpha]P_1 = \pi_p(P_2) - [p]P_2$.

- Si $[\alpha]P_1 = P_\infty$, c'est le cas idéal, où le point P_2 d'ordre r vérifie que $\pi_p(P_2) = [p]P_2$. Nous aurons ainsi trouvé du premier coup un élément de $\mathbb{G}_2$.

- Si $[\alpha]P_1 \neq P_\infty$, retrouver α reviendrait à résoudre un problème de logarithme discret, ce qui est difficile.

Dans le cas où $[\alpha]P_1 \neq P_\infty$, nous contournons le problème en posant $P_1' = [\alpha]P_1$. Nous connaissons l'expression de P_1' et ce point d'ordre r est un élément de $E(\mathbb{F}_p)$. Nous pouvons alors considérer la base (P_1', P_2) de $E[r]$ qui est une famille libre, car (P_1, P_2) est une famille libre.

La matrice du Frobenius dans la base (P_1', P_2) s'écrit alors :

$$\begin{bmatrix} 1 & 1 \\ 0 & p \end{bmatrix}.$$

Afin de ne pas surcharger les notations, nous notons $P_1 = P_1'$.

Nous pouvons maintenant chercher un vecteur propre pour la valeur propre p de π_p. Soit $Q = [c_1]P_1 + [c_2]P_2$ et vérifiant que $\pi_p(Q) = [p]Q$. Les équations ci dessous décrivent ce qui se passe lorsque nous appliquons le Frobenius à Q.

$$\begin{aligned} Q &= [c_1]P_1 + [c_2]P_2. \\ \pi_p(Q) &= \pi_p([c_1]P_1 + [c_2]P_2), \\ (\text{ Propriété I.3.3}) &= [c_1]\pi_p(P_1) + [c_2]\pi_p(P_2), \\ &= [c_1]P_1 + [c_2](P_1 + [p]P_2), \\ &= [c_1 + c_2]P_1 + [c_2p]P_2. \end{aligned}$$

Sachant que $[p]Q = [pc_1]P_1 + [pc_2]P_2$, le point Q est vecteur propre pour la valeur propre p de π_p si et seulement si

$$c_1 + c_2 \equiv c_1 p \bmod(r).$$

Donc en fixant une valeur de c_1, nous pouvons en déterminer une pour c_2 et ainsi construire un point Q d'ordre r, vérifiant que $\pi_p(Q) = [p]Q$. Ce point Q ainsi que ses multiples sont des éléments de $\mathbb{G}_2$.

Une approche plus simple est de considérer le point $\pi_q(Q') - Q'$ pour construire la base.

I.9.2 Le couplage Ate

Nous rappelons que la fonction rationnelle $f_{s,P}$ est une fonction admettant pour diviseur $Div(f_{s,P}) = s(P) - ([s]P) - (s-1)(P_\infty)$.

La construction du couplage Ate repose sur une représentation différente du groupe $\mathbb{G}_2$: $\mathbb{G}_2 := E[r] \cap \mathrm{Ker}(\pi_p - [p])$. Cette construction apporte des propriétés supplémentaires aux points Q, qui permettent de calculer le couplage Ate en réduisant le nombre d'itérations pour l'algorithme de Miller.

Nous définissons le couplage de Ate avant de présenter le théorème permettant de prouver que la définition I.9.1 définit bien un couplage bilinéaire et non dégénéré.

Définition I.9.1. Le **couplage Ate** réduit noté e_A est l'application :

$$\begin{aligned} e_A : \mathbb{G}_2 \times \mathbb{G}_1 &\to \mathbb{G}_3, \\ (Q, P) &\to f_{T,Q}(P)^{\frac{p^k-1}{r}}, \end{aligned}$$

où $T = t - 1$, avec t la trace du Frobenius sur E.

Le Théorème I.9.2 suivant donne une égalité reliant le couplage Ate au couplage de Tate. Ainsi, les propriétés définissant les couplages sont vérifiées pour e_A car elles le sont par le couplage de Tate noté e_T.

Théorème I.9.2. *Soient $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$.*
Soient

$$N = pgcd(T^k - 1, p^k - 1), \qquad L \text{ l'entier tel que } T^k - 1 = NL,$$

$$c = \sum_{i=0}^{k-1} T^{k-1-i} p^i \equiv k p^{k-1} mod(r).$$

Alors les propriétés suivantes sont vraies :

- ◇ *r divise N, soit s l'entier tel que $N = rs$,*
- ◇ *r ne divise pas s, soit $M = Lsc^{-1} \ mod(r)$,*
- ◇ $e_A(Q,P) = e_T(Q,P)^M$.

D'après ce théorème, le couplage Ate est une puissance fixe du couplage de Tate. Il suffit donc de prouver cette égalité pour prouver que l'application e_A que nous venons de définir est bien une application non dégénérée et bilinéaire de $\mathbb{G}_2 \times \mathbb{G}_1 \to \mathbb{G}_3$. La démonstration de cette égalité se trouve dans l'Annexe A.1.

L'exécution du couplage Ate nécessite T itérations de l'algorithme de Miller, plus une exponentiation finale. Le couplage de Tate définie en I.8.3 est exécuté en r itérations de l'algorithme de Miller, plus une exponentiation finale. En terme de nombre d'itérations, le couplage Ate est moins long à calculer que le couplage de Tate, car nous avons vu que T est au plus de la taille $\frac{\log_2(r)}{2}$ lorsque $\rho = 1$; ce qui représente un gain non négligeable. L'inconvénient du couplage Ate est qu'il correspond à une application de Miller Full. Bien que le nombre d'itérations soit inférieur à celui pour le couplage de Tate, les opérations sont plus complexes, car elles sont effectuées dans $\mathbb{F}_{p^k}$. Il serait intéressant de pouvoir ramener le calcul du couplage Ate sur le produit cartésien $\mathbb{G}_1 \times \mathbb{G}_2$, afin de le calculer par une exécution de Miller Lite, ce qui allégerait la complexité des calculs. Le couplage Twisted Ate permet d'effectuer cette transformation.

I.9.3 Le couplage Twisted Ate

Nous ne donnerons ici que les éléments de base permettant de définir le couplage Twisted Ate sans entrer dans les détails mathématiques, ceux ci ainsi que les démonstrations se trouvent dans [HSV06].

Soient $E(\mathbb{F}_p)$ une courbe elliptique admettant un twist de degré d, $\mathbb{G}_1 = E[r] \cap \mathrm{Ker}(\pi_p - [1])$, $\mathbb{G}_2 := E[r] \cap \mathrm{Ker}(\pi_p - [p])$ et $\mathbb{G}_3 = \mathbb{F}^\star_{p^k}$. Soient $m = pgcd(k,d)$ et $e = k/m$. Le couplage Twisted Ate est définie de la manière suivante.

Définition I.9.3. Le **couplage twisted Ate** réduit, noté e_{TA}, est l'application :

$$\begin{aligned} e_{TA} : \mathbb{G}_1 \times \mathbb{G}_2 &\rightarrow \mathbb{G}_3, \\ (P,Q) &\rightarrow f_{T^e,P}(Q)^{\frac{p^k-1}{r}}, \end{aligned}$$

où $T = t - 1$, avec t la trace du Frobenius sur E.

De la même façon que pour le couplage Ate, la démonstration que l'application e_{TA} est bien non dégénérée et bilinéaire est faite en utilisant une relation liant le couplage Twisted Ate au couplage de Tate.

Théorème I.9.4. *Soient $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$.*
Soient $m = pgcd(k,d)$ et $e = k/m$,

$$N = pgcd(T^k - 1, p^k - 1), \qquad L \text{ l'entier tel que } T^k - 1 = NL,$$

$$c \equiv \sum_{i=0}^{m-1} T^{e(m-1-i)} p^{ei} \equiv m p^{e(m-1)} \ mod(r).$$

Alors les propriétés suivantes sont vraies :

- *r divise N, soit s l'entier tel que $N = rs$,*
- *r ne divise pas s, soit $M = Lsc^{-1} \ mod(r)$,*
- *$e_{TA}(P,Q) = e_T(P,Q)^M$.*

Le couplage Twisted Ate est une opération de Miller Lite, et se calcule en $e\log_2(t-1)$ itérations de l'algorithme de Miller. Dès que l'inégalité

$$e\log_2(t-1) < \log_2(r)$$

est vérifiée, le calcul de $e_{TA}(P,Q)$ est plus efficace que le calcul de $e_T(P,Q)$. En effet, les deux couplages se calculent de la même façon, mais en moins d'itérations, donc en moins de temps, pour le Twisted Ate dès que $e\log_2(t-1) < \log_2(r)$. Pour utiliser pleinement cette propriété, les cryptographes recherchent des courbes telles que leur trace du Frobenius sur le corps $\mathbb{F}_p$ soit la plus petite possible. La comparaison des complexités entre les couplages Ate et Twisted Ate dépend fortement de la valeur de T et de $T^e \bmod(r)$.

Des versions optimisées de ces couplages, ainsi qu'une comparaison d'efficacité du calcul des couplages de Tate, Ate et Twisted Ate sont proposées dans [MKHO07]. L'optimisation consiste à jouer sur les puissances de T pour obtenir des couplages minimisant le nombre d'itérations dans l'algorithme de Miller. Il arrive que le couplage de Tate soit plus efficace que le couplage Ate ou Twisted Ate. Dans la partie IV.2.2 nous nous intéresserons à la notion

de couplage optimal, introduite par Vercauteren dans [Ver08] et généralisée par Hess dans [Hes08].

Le chapitre suivant présente l'arithmétique de $\mathbb{F}_{p^k}$ ainsi que les optimisations calculatoires et mathématiques du calcul des couplages.

Chapitre II

Arithmétique des couplages

Sommaire

II.1 Présentation

Nous reprenons l'étude de deux articles aux conclusions contradictoires sur l'efficacité de la cryptographie à base de couplage. Le premier est un travail de Koblitz et Menezes [KM05] publié en 2005, le second est un travail de Granger, Page et Smart [GPS06] publié en 2006. Nous avons prolongé ces deux études en 2007 dans l'article [BE07] publié dans les proceedings de la conférence SPIE. Ces différents travaux comparent l'efficacité du calcul des couplages de Weil et de Tate. Koblitz et Menezes montrent que pour de hauts niveaux de sécurité le couplage de Weil est moins coûteux à calculer que le couplage de Tate. Granger, Page et Smart arrivent à la conclusion opposée dans leur article.

Au delà de la comparaison entre les complexités en nombre d'opérations sur le corps fini du calcul des couplages de Weil et de Tate, cette partie nous permet surtout d'introduire les différentes méthodes qui permettent d'optimiser le calcul des couplages. Nous considérons les arithmétiques intervenant pour ce calcul. Nous reprenons chacune des méthodes de calcul pour les appliquer aux deux couplages et lever les ambiguïtés. Nous faisons une analyse détaillée des corps choisis et de l'arithmétique utilisée afin de comparer leurs complexités d'implémentation. Ce travail préliminaire nous permettra de comparer les deux approches et de comprendre les conclusions contradictoires de ces articles.

Nous détaillons la complexité de l'implémentation conformément à ce qui est décrit dans [KM05] et [GPS06]. Le premier repose sur l'utilisation de « corps adaptés au calcul des couplages » pour des courbes dont le degré de plongement k est de la forme $2^i 3^j$. Quant au second, il utilise une représentation à l'aide de polynômes cyclotomiques pour représenter le corps fini.

Cette comparaison nous permettra de présenter les diverses optimisations possibles du calcul des couplages. Elles se rangent en deux catégories. D'une part, dans la partie II.3 nous présentons les optimisations arithmétiques, se traduisant par l'amélioration de l'arithmétique des corps finis intervenant en cryptographie à base de couplages. Et d'autre part dans la partie II.4, nous présentons les optimisations mathématiques qui consistent à utiliser des propriétés liées à la structure de groupe des courbes elliptiques.

Que ce soit en cryptographie, ou de manière générale en arithmétique, les opérations élémentaires sur le corps de base sont les opérations conditionnant l'efficacité des calculs. En améliorant le calcul du produit de deux polynômes dans une extension de corps, nous améliorons l'efficacité du calcul des couplages.

Nous détaillons dans la partie II.2 les besoins arithmétiques de la cryptographie à base de couplage. Puis, la partie II.3 introduit les différentes multiplications de polynômes citées dans la littérature. Nous nous intéressons

ensuite aux améliorations mathématiques. Les aspects mathématiques utilisent la structure des groupes intervenants, par exemple des isomorphismes de courbes elliptiques, ou encore des choix de corps finis spécifiques pour avoir une multiplication dans $\mathbb{F}_{p^k}$ la plus efficace possible. Chaque approche est détaillée dans la partie II.4. Pour terminer cet état de l'art de l'arithmétique des couplages, nous donnons dans la partie II.5 quelques pistes de réflexion qui permettraient d'améliorer l'efficacité du calcul des couplages.

II.2 Aspects arithmétiques du calcul des couplages

II.2.1 L'algorithme de Miller

Afin de mieux comprendre à quel niveau interviennent les différentes optimisations, nous analysons les étapes de l'algorithme de Miller.

Soit P un point de $E(\mathbb{F}_p)[r]$ donné en coordonnées affines (X_P, Y_P) (ou Jacobiennes avec $Z_P = 1$). Nous considérons le point Q d'ordre r dans $E(\mathbb{F}_{p^k})$, donné lui aussi en coordonnées affines (x_Q, y_Q). Nous notons $\mathbb{G}_1 =< P >$ le sous groupe d'ordre r de $E(\mathbb{F}_p)$ engendré par le point P et $\mathbb{G}_2 =< Q >$ le sous groupe d'ordre r de $E(\mathbb{F}_{p^k})$ engendré par le point Q. Les groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ seront ceux dans lesquels nous souhaitons calculer un couplage, sous la condition $\mathbb{G}_1 \neq \mathbb{G}_2$. Le groupe $\mathbb{G}_3$ est un sous groupe de $\mathbb{F}^{\star}_{p^k}$ de même ordre r.

Soit T un point de $E(\mathbb{F}_{p^k})$, donné en coordonnées Jacobiennes (X_T, Y_T, Z_T). Les coordonnées Jacobiennes présentent l'avantage de ne pas nécessiter d'inversion lors des opérations arithmétiques sur la courbe elliptique, comme nous l'avons souligné dans la partie I.2.5.

L'algorithme de Miller est le suivant :

Algorithme 3 : Miller(P, Q, r)

Données : $r = (r_n \dots r_0)$(représentation binaire), $P \in \mathbb{G}_1(\subset E(\mathbb{F}_p))$ et $Q \in \mathbb{G}_2(\subset E(\mathbb{F}_{p^k}))$;
Résultat : $f_{r,P}(Q) \in G_3(\subset \mathbb{F}^{\star}_{p^k})$;
1. $T \leftarrow P$
2. $f_1 \leftarrow 1$
3. $f_2 \leftarrow 1$
pour $i = n-1$ **to** 0 **faire**
1 4. $T \leftarrow [2]T$
5. $f_1 \longleftarrow {f_1}^2 \times l_1(Q)$, l_1 est la tangente à la courbe en T.
6. $f_2 \longleftarrow {f_2}^2 \times v_1(Q)$, v_1 est la droite verticale en $[2]T$.
$(\ Div(\frac{l_1}{v_1}) = 2(T) - ([2]T) - P_\infty)$
2 **si** $n_i = 1$ **alors**
7. $T \leftarrow T + P$
8. $f_1 \longleftarrow f_1 \times l_2(Q)$, l_2 est la droite (PT)
9. $f_2 \longleftarrow f_2 \times v_2(Q)$, v_2 est la verticale au point $P + T$
$(\ Div(\frac{l_2}{v_2}) = (T) + D_P - ((T) \oplus D_P) - P_\infty)$
fin
retourner $\frac{f_1}{f_2}$
fin

Les fonctions $l_1(Q)$, $l_2(Q)$, $v_1(Q)$ et $v_2(Q)$ intervenant dans le calcul d'un couplage via l'algorithme de Miller ont pour espace d'arrivée $\mathbb{F}^{\star}_{p^k}$. Les paramètres f_1 et f_2 sont donc des éléments de $\mathbb{F}^{\star}_{p^k}$.

Le choix de l'ordre r du sous groupe de la courbe elliptique dans lequel nous travaillons est très important, il est choisi de manière à avoir une représentation binaire qui est creuse. Ainsi l'étape 2 n'est pas souvent calculée, alors que l'étape 1 est exécutée pour chaque itération. Il est alors admis que la complexité de l'exécution de l'algorithme de Miller est conditionnée par son étape de doublement. Nous suivrons donc la méthode de Koblitz-Menezes, ainsi que celle de Granger-Page-Smart, en considérant uniquement le calcul des fonctions l_1 et v_1 pour évaluer la complexité de calcul d'un couplage.

De manière générale, nous considérons que l'équation de la courbe elliptique E est de la forme $Y^2 = X^3 + aXZ^4 + bZ^6$, avec a et b éléments de $\mathbb{F}_p$. D'après [BJ03] nous pouvons considérer que $a = -3$, mais pour plus de généralité nous conserverons a quelconque. Nous noterons $P = (X_P, Y_P)$, $T = (X_T, Y_T, Z_T)$ le point courant durant l'algorithme de Miller et $2T = (X_{2T}, Y_{2T}, Z_{2T})$ le doublement de ce point.

Les formules de doublement en coordonnées Jacobiennes sont les suivantes [CF06] :

$$C = 2Y_T^2, \quad D = Z_T^2, \quad A = 4X_TY_T^2 = 2X_TC, \quad B = (3X_T^2 + aZ_T^4) \tag{II.2.1}$$

$$X_{2T} = B^2 - 2A, \quad Y_{2T} = B(A - X_{2T}) - 2C^2, \quad Z_{2T} = 2Y_TZ_T. \tag{II.2.2}$$

Dans ce cas, les expressions de l_1, et v_1, avec $Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^k})$, sont données par les formules :

$$l_1(x_Q, y_Q) = Z_P^2(Z_{2T}Dy_Q - B(Dx_Q - X_T) - 2Y_T) \tag{II.2.3}$$

$$v_1(x_Q, y_Q) = Z_{2T}^2Z_Px_Q + 4Y_P^2(X_PD + X_TZ_P^2) - Z_P^2B^2 \tag{II.2.4}$$

Nous pouvons remarquer que certains termes peuvent être calculés une seule et unique fois puis réutilisés, par exemple : Y_T^2, Z_T^2, $4X_TY_T^2$, $(3X_T^2+aZ_T^4)$. Ne pas recalculer deux fois les mêmes valeurs permet de réduire le coût de l'étape de doublement en terme d'opérations sur le corps de base.

Nous noterons A_{p^e} (respectivement S_{p^e} et M_{p^e}) une addition (respectivement un carré et une multiplication) dans le corps fini $\mathbb{F}_{p^e}$, pour e un entier naturel quelconque. Dans le Tableau II.1, nous donnons le coût de chacune des opérations nécessaires lors du calcul des couplages. Chaque coût est exprimé en terme d'opérations de base sur les corps finis. Le coût de l'étape de doublement dans l'Algorithme 3 est obtenue en optimisant l'ordre des calculs.

Doublement point sur E	$4S_p + 4M_p$
Évaluation fonction l_1 seule	$S_P + (3 + 3k)M_P$
Évaluation fonction v_1 seule	$3S_P + (5 + k)M_P$
Étape 1 dans l'Algorithme 3	$8S_p + (12 + 4k)M_p + 2S_{p^k} + 2M_{p^k}$

TABLE II.1 – Coût de l'étape de doublement de l'algorithme de Miller

Nous présenterons des propriétés permettent de réduire la complexité de l'étape 1 dans les parties II.4. Nous rappelons les définitions des couplages de Weil et de Tate avant de comparer leur coût en terme d'opérations sur les corps finis.

II.2.2 Couplages de Weil et de Tate

Définition du couplage de Weil : e_W

Soient $P \in \mathbb{G}_1$, et $Q \in \mathbb{G}_2$, le couplage de Weil e_W est définie par :

$$\begin{aligned} e_W : \mathbb{G}_1 \times \mathbb{G}_2 &\mapsto \mathbb{G}_3 \\ (P, Q) &\mapsto e_W(P, Q) = \frac{f_{r,P}(Q)}{f_{r,Q}(P)} \end{aligned}$$

Pour le calculer, l'algorithme de Miller est appliqué deux fois, mais pas de manière symétrique. Le calcul de $f_{r,P}(Q)$ est appelé opération de Miller Lite, celui de $f_{r,Q}(P)$ est appelée opération de Miller Full. La différence entre ces deux opérations est le corps fini dans lequel sont effectuées les opérations sur la courbe elliptique. L'opération Miller Lite est plus légère que l'opération Miller Full, car l'arithmétique de la courbe elliptique est faite dans le corps $\mathbb{F}_p$. Lors du calcul de $f_{r,Q}(P)$ les opérations sur la courbe elliptique sont effectuées dans le corps fini $\mathbb{F}_{p^k}$, alors que pour Miller Full elles sont effectuées dans $\mathbb{F}_{p^k}$. La complexité de l'étape de doublement 1 durant l'algorithme de Miller est $8S_p + (12 + 4k)M_p + 2S_{p^k} + 2M_{p^k}$ pour Miller Lite et $3kM_p + 10S_{p^k} + 14M_{p^k}$ pour Miller Full. Étant donné que les opérations dans $\mathbb{F}_{p^k}$ font appel à des opérations dans $\mathbb{F}_p$, l'opération Miller Full est plus coûteuse que l'opération Miller Lite.

Le calcul du couplage de Weil nécessite de plus une inversion finale et une multiplication dans $\mathbb{F}_{p^k}$.

Définition du couplage de Tate : e_T

Le couplage de Tate réduit e_T est définie par :

$$\begin{aligned} e_T : \mathbb{G}_1 \times \mathbb{G}_2 &\mapsto \mathbb{G}_3 \\ (P,Q) &\mapsto e_T(P,Q) = f_{r,P}(Q)^{\frac{p^k-1}{r}} \end{aligned}$$

Il nécessite une seule évaluation de Miller Lite dont le résultat est élevé à la puissance $\frac{p^k-1}{r}$. Différentes méthodes améliorant le calcul de cette exponentiation existent, par exemple les suites de Lucas [CF06] ou les fenêtres glissantes [CF06] ; nous rappellerons leurs complexités dans la partie II.4.4.

II.2.3 Comparaison des couplages de Weil et de Tate

Les couplages de Weil et de Tate étant les premiers couplages utilisés, ils ont souvent été comparés du point de vue de l'efficacité calculatoire. La différence majeure entre ces deux couplages est que le couplage de Weil nécessite deux applications de l'algorithme de Miller (Miller Lite et Miller Full), plus une inversion et une multiplication, alors que le couplage de Tate nécessite une application de Miller Lite plus une exponentiation finale. A première vue, une exponentiation semble moins coûteuse qu'un calcul de Miller Full. Par conséquent, le couplage de Weil est a priori le plus coûteux.

Pourtant, en 2005 N. Koblitz et A. Menezes[KM05] ont comparé les coûts de calcul de ces deux couplages. De manière surprenante, leur conclusion a été que pour de hauts niveaux de sécurité, le couplage de Weil se calcule de manière plus efficace que le couplage de Tate. Les hauts niveaux de sécurité

sont atteints en augmentant la taille des corps finis, et par conséquent, la taille de l'exposant $\frac{p^k-1}{r}$ augmente au point de rendre l'exponentiation plus coûteuse qu'une exécution de Miller Full.
En 2006, R. Granger, D. Page et N. Smart[GPS06] répondent à Koblitz et Menezes. En améliorant les calculs dans l'extension $\mathbb{F}_{p^k}$, ils réduisent le coût de calcul de l'exponentiation finale et arrivent à la conclusion opposée, c'est à dire que, quel que soit le niveau de sécurité considéré, le calcul du couplage de Tate est toujours plus efficace que celui de Weil.

Nous allons présenter les différentes optimisations des calculs utilisant la structure des groupes intervenants dans le calcul de couplages pour améliorer l'arithmétique de $\mathbb{F}_{p^k}$ et par conséquent le calcul des couplages. Nous vérifions les complexités avancées dans ces papiers. Puis, nous reprenons l'analyse des coûts de calcul des deux couplages afin de voir si Weil devient plus efficace que celui de Tate pour de hauts niveaux de sécurité.

II.3 Optimisation arithmétique

Une première façon d'optimiser les calculs de couplage est de s'intéresser aux opérations arithmétiques dans les corps et extensions de corps utilisés.

Nous avons introduit dans la partie I.9.1 les dernières optimisations théoriques des couplages, à savoir les couplages Ate et Twisted Ate. Les courbes elliptiques utilisées pour ces derniers sont généralement construites par la méthode de multiplication complexe que nous avons décrite dans la partie I.5. Ces méthodes de constructions de courbes ne laissent pas la liberté de choisir a priori la caractéristique p du corps fini $\mathbb{F}_p$, nous fixons uniquement l'ordre de grandeur de la taille de p. Ainsi, l'arithmétique des couplages est particulière dans le sens où nous avons très peu d'influence sur le nombre premier p définissant le corps de base $\mathbb{F}_p$. Il n'est pas possible de choisir p avec une structure permettant de rendre plus performant les calculs dans $\mathbb{F}_p$. Nous ne traitons donc ici que des améliorations au niveau des opérations dans $\mathbb{F}_{p^k}$ en terme d'opération de base dans $\mathbb{F}_p$. Nous rappelons les différentes méthodes de multiplication utilisées en arithmétique pour effectuer le produit de deux polynômes de même degré.

II.3.1 Problématique

Nous fixons tout d'abord le cadre et les notations de cette partie. Soit $\mathbb{F}_p$ un corps fini de caractéristique p, un nombre premier de grande taille, et $\mathbb{F}_{p^k}$ l'extension de $\mathbb{F}_p$ de degré k. Le polynôme irréductible de degré k définissant $\mathbb{F}_{p^k}$ est noté $P(X)$. Soient A et B deux éléments de $\mathbb{F}_{p^k}$. La représentation des éléments de $\mathbb{F}_{p^k}$ est faite dans la base $\mathcal{B} = (1, \gamma, \gamma^2, \ldots, \gamma^{k-1})$ pour γ une

racine dans $\mathbb{F}_{p^k}$ du polynôme $P(X)$. Les éléments de $\mathbb{F}_{p^k}$ sont représentés par des polynômes en γ à coefficients dans $\mathbb{F}_p$:

$$\mathbb{F}_{p^k} = \{\sum_{i=0}^{k-1} a_i\gamma^i, a_i \in \mathbb{F}_p\}.$$

A est représenté par le polynôme $\sum_{i=1}^{k-1} a_i\gamma^i$, et B par $B = \sum_{i=0}^{k-1} b_i\gamma^i$. Le produit de A et B revient ainsi à effectuer le produit de deux polynômes et se décompose en deux étapes. La première est le produit de ces deux polynômes dont le résultat est un polynôme $C(X)$ de degré $(2k-2)$. La seconde est la réduction polynomiale de ce résultat. La complexité de cette réduction dépend du polynôme $P(X)$. Plus $P(X)$ est creux, et plus la réduction est efficace. Ainsi, le polynôme irréductible définissant $\mathbb{F}_{p^k}$ est choisi de préférence de la forme $X^k - \beta$ pour $\beta \in \mathbb{F}_p$. La réduction polynomiale est alors composé de multiplications par β dans $\mathbb{F}_p$ et d'additions dans $\mathbb{F}_p$:

$$C(X) = C_0(X) + C_1(X)X^k \equiv C_0(X) + \beta C_1(X) \bmod(P(X))$$

Le fait de pouvoir trouver un polynôme irréductible de la forme $X^k - \beta$ dépend des valeurs de k et p, comme nous l'avons souligné dans la Propriété I.2.8. La complexité du produit de A et B dépend fortement de la complexité du produit de deux polynômes. Nous introduisons les différentes multiplications de polynômes existantes dans la littérature.

II.3.2 La multiplication naïve

La multiplication naïve apprise à l'école permet d'obtenir le produit de deux polynômes en utilisant la formule :

$$A(\gamma) \times B(\gamma) = \sum_{i=0}^{2k-1} \left(\sum_{j=0}^{i} (a_j b_{i-j})\right)\gamma^i.$$

Cette méthode simple en apparence est quadratique en le degré des polynômes, donc coûteuse. Elle nécessite k^2 multiplications dans le corps $\mathbb{F}_p$, et $k(2k-1)$ additions, ce qui est noté $k^2M_p + k(2k-1)A_p$.

Les méthodes par interpolation sont une alternative à la multiplication scolaire, elles sont connues pour être plus efficaces à partir d'une certaine valeur de k.

II.3.3 Les multiplications par interpolation

Notons $A(X) = a_0 + a_1X + \ldots + a_{k-1}X^{k-1}$ et $B(X) = b_0 + b_1X + \ldots + b_{k-1}X^{k-1}$, les polynômes obtenus en remplaçant γ par l'inconnue X dans l'expression

de A et B éléments de $\mathbb{F}_{p^k}$. Le résultat $C(X)$ du produit de $A(X)$ et $B(X)$ est un polynôme de degré $2k-2$. Nous savons qu'un polynôme de degré m est déterminé par ses images en $m+1$ points distincts, comme le rappelle le théorème suivant.

Théorème II.3.1. *Soit $P(X)$ un polynôme de degré m, alors le polynôme $P(X)$ est entièrement déterminé par la donnée de l'image de $m+1$ points distincts.*

Les méthodes par interpolation reposent sur ce théorème, elles consistent à trouver $2k-1$ images du polynôme produit $C(X)$, puis d'interpoler C pour retrouver l'expression générale du polynôme. Les méthodes de multiplication par interpolation suivent le mode opératoire suivant :

1. Trouver $2k-1$ points différents deux à deux dans $\mathbb{F}_p$
   nous noterons ces points $\alpha_0, \alpha_1, \ldots, \alpha_{2k-2}$.
2. Évaluer les polynômes $A(X)$ et $B(X)$ en ces $2k-1$ points :
   stocker $A(\alpha_0), \ldots, A(\alpha_{2k-2}), B(\alpha_0), \ldots, B(\alpha_0)$.
3. Calculer les valeurs de C en ces $2k-1$ points,
   sachant que $C(\alpha_i) = A(\alpha_i)B(\alpha_i)$.
4. Reconstruire le polynôme C par une méthode d'interpolation, comme celle de Lagrange ou de Newton, utilisant les $2k-1$ valeurs de ce polynômes.

Les méthodes par interpolation remplacent certaines multiplications dans $\mathbb{F}_p$ par des multiplications par des constantes toujours dans $\mathbb{F}_p$. Les produits utiliseront des propriétés de ces constantes qui dépendent des choix des points d'interpolation. En contrepartie elles utilisent un plus grand nombre d'additions dans $\mathbb{F}_p$. L'addition étant moins coûteuse que la multiplication, pour certains degrés d'extension le gain en nombre de multiplications est plus important que les pertes dues aux additions supplémentaires. Nous présentons deux méthodes classiques d'interpolation, celles de Lagrange et de Newton.

Interpolation de Lagrange

Nous supposons que nous avons évalué les polynômes $A(X)$ et $B(X)$ en $2k-1$ points distincts, notés $(\alpha_0, \alpha_1, \ldots, \alpha_{2k-2})$. Ainsi nous disposons de l'image de $C(X) = A(X) \times B(X)$ en ces $2k-1$ points. Nous désirons reconstruire l'expression de $C(X)$, c'est à dire retrouver ses coefficients. L'interpolation de Lagrange du polynôme $C(X)$ est donnée par la formule suivante :

$$C(X) = \sum_{i=0}^{2k-2} \left(C(\alpha_i) \times \frac{\prod_{j=0, j\neq i}^{2k-2} (X - \alpha_j)}{\prod_{j=0, j\neq i}^{2k-2} (\alpha_i - \alpha_j)} \right) \quad \text{(II.3.1)}$$

La complexité de cette opération dépend de :

1. Les évaluations de A et B en les $(2k-1)$ points d'interpolation α_i.
2. Le calcul des : $C(\alpha_i) = A(\alpha_i) \times B(\alpha_i)$.
3. La reconstruction de $C(X)$ grâce à la Formule II.3.1.

Nous noterons CM_p la multiplication par des constantes dans $\mathbb{F}_p$. Les évaluations en les points $(\alpha_i)_{\{i=0\ldots(2k-1)\}}$ ont un coût de

$$2(2k-1)(k-1)\left(A_p + CM_p\right),$$

lorsqu'elles sont effectuées en utilisant le schéma de Horner :

$$A(\alpha_i) = a_0 + \alpha_i \left(a_1 + \alpha_i(a_2 + \alpha_i[\ldots])\right).$$

Le calcul des $C(\alpha_i) = A(\alpha_i) \times B(\alpha_i)$ s'effectue en $(2k-1)$ produits dans $\mathbb{F}_p$, ce que nous notons $(2k-1)M_p$.

La complexité de la reconstruction des coefficients du polynôme $C(X)$ à l'aide de la Formule II.3.1 est déterminée par les $2k-1$ produits par les $C(\alpha_i)$ et par les additions des résultats des produits. Les éléments $\dfrac{\prod_{j=0, j\neq i}^{2k-1} (X - \alpha_j)}{\prod_{j=0, j\neq i}^{2k-1} (\alpha_i - \alpha_j)}$ dépendent uniquement de l'inconnue X et des points d'interpolation α_i, ils peuvent donc être précalculés une fois pour toute. Ce sont des polynômes de degré $(2k-2)$, car il y a $(2k-1)$ valeurs pour j et que nous faisons le produit sur toutes les valeurs moins une. Le produit par les $C(\alpha_i)$ se fait en $(2k-1)CM_p$ opérations. Nous devons faire $(2k-1)$ produits de cette sorte, les produits par les $C(\alpha_i)$ représentent $(2k-1)^2 CM_p$.

Le nombre d'additions est lui de $(2k-1)(2k-2)A_p$. La reconstruction des coefficients de $C(X)$ nécessite $(2k-1)^2 CM_p + (2k-1)(2k-2)A_p$ opérations.

Au final, la complexité de l'interpolation de Lagrange est :

$$(2k-1)M_p + (2k-1)(4k-3)CM_p + 2(2k-1)(3k-2)A_p \quad \text{(II.3.2)}$$

Interpolation de Newton

Comme pour Lagrange, nous disposons des $C(\alpha_i)$ et nous devons retrouver les coefficients de $C(X)$. L'interpolation de Newton nécessite la construction d'une représentation intermédiaire.

Dans un premier temps, nous devons calculer les valeurs c'_i :

$$\left\{\begin{array}{rcl} c'_0 & = & C(\alpha_0) \\ c'_1 & = & (C(\alpha_1) - c'_0)\frac{1}{(\alpha_1-\alpha_0)} \\ c'_2 & = & \left((C(\alpha_2) - c'_0)\frac{1}{(\alpha_2-\alpha_0)} - c'_1\right)\frac{1}{(\alpha_2-\alpha_1)} \\ \vdots & = & \vdots \\ c'_{2k-2} & = & \left((C(\alpha_{2k-2}) - c'_0)\frac{1}{(\alpha_{2k-2}-\alpha_0)} - c'_1\right)\frac{1}{(\alpha_{2k-2}-\alpha_1)} - \dots \end{array}\right.$$

Ces valeurs permettent d'écrire $C(X)$ sous la forme :

$$\begin{array}{rcl} C(X) & = & c'_0 + c'_1(X-\alpha_0) + c'_2(X-\alpha_0)(X-\alpha_1) \\ & & + \dots + c'_{2k-2}(X-\alpha_0)(X-\alpha_1)\dots(X-\alpha_{2k-2}) \end{array}$$

La reconstruction des coefficients de $C(X)$ peut être faite grâce au schéma de Horner :

$$\begin{array}{rcl} C(X) & = & c'_0 + (X-\alpha_0)[c'_1 + (X-\alpha_1)(c'_2 + (X-\alpha_2)(\dots \\ & & \dots + (X-\alpha_{2k})[c'_{2k-1} + (X-\alpha_{2k-1})c'_{2k-2}]))] \end{array}$$

Le choix des α_i influence l'efficacité de la multiplication par interpolation. L'interpolation de Newton nécessite des divisions exactes par les différences des α_i. En effet, les α_i étant connus a priori, nous pouvons précalculer et stocker les $(\alpha_i - \alpha_j)^{-1}$ dans $\mathbb{F}_p$. Les produits par les $(\alpha_i - \alpha_j)^{-1}$ deviennent alors des multiplications par des constantes. Ces inversions peuvent être transformées en multiplication par des constantes. La complexité de l'interpolation de Newton est la somme des complexités du calcul des $C(\alpha_i)$ et des c'_i, ainsi que de la complexité de la reconstruction des coefficients de $C(X)$.

Le calcul des $C(\alpha_i)$ est le même que pour l'interpolation de Lagrange, sa complexité est

$$2(2k-1)(k-1)\,(A_p + CM_p) + (2k-2)M_p.$$

Le calcul des c'_i nécessite $\left(\sum_{i=0}^{(2k-2)} i\right)(CM_p + A_p)$ opérations soit

$$(k-1)(2k-1)(CM_p + A_p).$$

Le schéma de Horner nécessite $\left(\sum_{i=0}^{(2k-2)} i\right)(CM_p + A_p)$ soit

$$(k-1)(2k-1)A_p + (k-1)(2k-1)CM_p.$$

Au total la complexité de l'interpolation de Newton est de

$$4(2k^2-3k+1)A_p + 4(2k^2-3k+1)CM_p + (2k-1)M_p.$$

Comparaison entre les deux méthodes

Les méthodes de multiplication par interpolation de Lagrange et Newton nécessitent le même nombre de multiplications dans $\mathbb{F}_p$: $(2k-1)$ où $(k-1)$ est le degré des polynômes dont nous voulons calculer le produit. L'interpolation de Lagrange est très intéressante lorsque les calculs peuvent être parallélisés, les calculs des $C(\alpha_i) \times \frac{\prod_{j\neq i}(X-\alpha_j)}{\prod_{j\neq i}(\alpha_i-\alpha_j)}$ sont indépendants les uns des autres. La méthode de Newton nécessite moins d'addition et de multiplications par des constantes que celle de Lagrange, mais du fait de la dépendance des c'_i, la parallélisation des calculs est moins intrinsèque. En effet, le calcul d'un coefficient c'_i pour un entier i fixé dépend des valeurs des c'_j pour $0 \leq j < i$.

Opération \ Méthode	Lagrange	Newton
A_p	$12k^2-14k+4$	$8k^2-12k+4$
CM_p	$8k^2-10k+3$	$8k^2-12k+4$
M_p	$(2k-1)$	$(2k-1)$

TABLE II.2 – Coût interpollation en opération élémentaire

L'interpolation de Lagrange devrait donc être privilégiée lorsque les calculs peuvent être parallélisés, et celle de Newton lorsque la taille du circuit est limitée, typiquement dans le cas de cartes à puce.

II.3.4 La méthode de Karatsuba

La multiplication de Karatsuba est une application directe du produit par interpolation de Newton, pour des polynômes de degré 1. Le polynôme produit est de degré 2, nous avons donc besoin de 2+1 = 3 points d'interpolation, ces points sont $\{0, 1, \infty\}$. Karatsuba permet de faire le produit de deux polynômes de degré 1 en seulement trois produits de coefficients, au lieu de quatre par la multiplication scolaire. Les multiplications par des constantes intervenant durant l'interpolation de Newton disparaissent car il s'agit de divisions par 1, ou par l'infini, ce qui simplifie les calculs.

Soit $A(X) = A_0 + A_1X$ et $B(X) = B_0 + B_1X$ deux polynômes de degré 1 dont nous souhaitons calculer le produit $C(X) = A(X) \times B(X)$.

Nous allons d'abord évaluer le polynôme $C(X)$ en les points $\{0, 1, \infty\}$ à l'aide des équations II.3.3.

$$\begin{array}{rcl} C(0) & = & (A_1X + A_0)(B_1X + B_0) \bmod (X) \\ & = & A_0 \times B_0 \\ C(1) & = & (A_1X + A_0)(B_1X + B_0) \bmod (X-1) \\ & = & (A_0 + A_1) \times (B_0 + B_1) \\ C(\infty) & = & (A_1X + A_0)(B_1X + B_0) \bmod (X-\infty) \\ & = & A_1 \times B_1 \times X^2 \bmod (X-\infty) \end{array} \quad \text{(II.3.3)}$$

L'évaluation du polynôme $C(X)$ en les trois points d'interpolation nécessite trois produits et deux additions dans $\mathbb{F}_p$.

Nous allons appliquer les formules de Newton pour retrouver les coefficients de $C(X)$.

$$\left\{ \begin{array}{rcl} c'_0 & = & C(0) \\ & = & A_0B_0 \\ c'_1 & = & (C(1) - c'_0)\frac{1}{(1-0)} \\ & = & (A_0 + A_1)(B_0 + B_1) - A_0B_0 \\ c'_2 & = & \left((C(\infty) - c'_0)\frac{1}{(\infty-0)} - c'_1\right)\frac{1}{(\infty-1)} \\ & = & \left((A_1B_1X^2 - A_0B_0)\frac{1}{(X-0)} - ((A_0 + A_1)(B_0 + B_1) - A_0B_0)\right)\frac{1}{(X-1)} \bmod (X-\infty) \\ & = & \frac{A_1B_1X^2}{X^2} - \frac{A_0B_0}{X^2} - \frac{((A_0+A_1)(B_0+B_1)-A_0B_0)}{X} \bmod (X-\infty) \\ & = & A_1B_1 \end{array} \right.$$

Puis, à l'aide des coefficients c'_i nous reconstruisons le polynôme $C(X)$.

$$\begin{array}{rcl} C(X) & = & c'_0 + c'_1X + c'_2X(X-1) \\ & = & A_0B_0 + ((A_0 + A_1)(B_0 + B_1) - A_0B_0)X + A_1B_1X(X-1) \\ & = & A_0B_0 + ((A_0 + A_1)(B_0 + B_1) - A_0B_0 - A_1B_1)X + A_1B_1X^2 \end{array}$$

La formule de calcul de la reconstruction du polynôme $C(X)$ par la multiplication de Karatsuba est donnée par l'équation II.3.4.

$$\left\{ \begin{array}{rcl} c_0 & = & A_0 \times B_0 \\ c_1 & = & (A_0 + A_1) \times (B_0 + B_1) \\ c_2 & = & A_1 \times B_1 \\ C(X) & = & c_0 + (c_1 - c_0 - c_2)X + c_2X^2 \end{array} \right. \quad \text{(II.3.4)}$$

Pour des polynômes de degré 1, la complexité de la multiplication de Karatsuba est de $3M_p + 4A_p$.

La multiplication de Karatsuba peut s'appliquer récursivement pour des polynômes de degré supérieur à 1. Soit $A(X) = A_0 + A_1X + \ldots A_mX^m$, nous pouvons séparer $A(X)$ en deux parties de degré inférieur ou égal à $\left\lfloor \frac{m}{2} \right\rfloor$:

$$\begin{aligned} A(X) &= A_0 + A_1X + \dots A_{\lfloor \frac{m}{2} \rfloor - 1} + X^{\lfloor \frac{m}{2} \rfloor}\left(A_{\lfloor \frac{m}{2} \rfloor} + A_{\lfloor \frac{m}{2} \rfloor + 1}X + \dots A_m X^{\lfloor \frac{m}{2} \rfloor}\right), \\ &= \widetilde{A_0} + Y\widetilde{A_1}, \text{ où nous posons } Y = X^{\lfloor \frac{m}{2} \rfloor}. \end{aligned}$$

Ensuite, nous appliquons la multiplication de Karatsuba aux deux morceaux obtenus. Les trois multiplications entre les deux morceaux peuvent elles aussi se faire en appliquant Karatsuba. L'application récursive de cette multiplication est la méthode la plus efficace pour calculer le produit de polynômes dont le degré est une puissance de deux. La complexité asymptotique de la multiplication de Karatsuba est en $O(m^{\log_2(3)})$ multiplications et $O(m)$ additions, où m représente le degré des polynômes dont nous cherchons le produit.

II.3.5 La méthode de Toom Cook 3

De même que la multiplication de Karatsuba, la multiplication de Toom Cook 3 est une application directe de l'interpolation de Newton. Elle permet d'effectuer la multiplication de polynômes de degré 3 en cinq multiplications de coefficients, au lieu de neuf par la multiplication scolaire. Les points pour l'évaluation sont $\{0, 1, -1, 2, \infty\}$. Contrairement à la multiplication de Karatsuba, la multiplication de Toom Cook, et en particulier la partie interpolation, nécessite des opérations comme des multiplications ou divisions par des constantes.

Soit $A(X) = A_0 + A_1X + A_2X^2$ et $B(X) = B_0 + B_1X + B_2X^2$ les polynômes de degré 2 dont nous voulons calculer le produit par la multiplication de Toom Cook.

La partie évaluation de cette multiplication nécessite dix additions des données A_i et B_i, pour $i = 0, 1, 2$. Voici les formules pour l'évaluation de $A(X)$ qui se fait en cinq additions.

$$\begin{cases} A(0) &= A_0 \\ Sp_1 &= A_0 + A_2 \\ A(1) &= Sp_1 + A_1 \\ A(-1) &= Sp_1 - A_1 \\ A(2) &= A_0 + 2A_1 + 4A_2 \\ A(\infty) &= A_2X^2 \bmod (X - \infty) \end{cases}$$

Nous commençons par évaluer $C(X)$ en les α_i pour $i = 0, 1, 2, 3, 4$.

$$\begin{cases} C(0) &= A(0) \times B(0) = A_0 B_0 \\ C(1) &= A(1) \times B(1) \\ C(-1) &= A(-1) \times B(-1) \\ C(2) &= A(2) \times B(2) \\ C(\infty) &= A(\infty) \times B(\infty) = A_2 B_2 X^4 \bmod (X - \infty) \end{cases}$$

Nous effectuons les mêmes calculs que pour la multiplication de Karatsuba, les coefficients c'_i sont :

$$\begin{cases} c'_0 &= C(0) \\ c'_1 &= C(1) - c'_0 \\ c'_2 &= \frac{1}{2}(C(-1) - c'_0 + c'_1) \\ c'_3 &= \frac{1}{6}C(2) - \frac{1}{6}c'_0 - \frac{1}{3}c'_1 - \frac{1}{3}c'_2 \\ c'_4 &= A_2 B_2 \end{cases}$$

La reconstruction de $C(X)$ se fait grâce à l'égalité :

$$\begin{aligned} C(X) = c'_0 + c'_1 X \quad + \quad & c'_2 X(X-1) + c'_3 X(X-1)(X+1) \\ & c'_4 X(X-1)(X+1)(-2), \end{aligned}$$

qui se transforme en :

$$\begin{aligned} C(X) = c'_0 \quad + \quad & (c'_1 - c'_2 - c'_3 - 2c'_4)X + (c'_2 - c'_4)X^2 \\ + \quad & (c'_3 - 2c'_4)X^3 + c'_4 X^4. \end{aligned}$$

d'où

$$\begin{cases} C_0 &= c'_0 \\ C_1 &= c'_1 - c'_2 - c'_3 - 2c'_4 \\ C_2 &= c'_2 - c'_4 \\ C_3 &= c'_3 - 2c'_4 \\ C_4 &= c'_4 \\ C(X) &= C_0 + C_1 X + C_2 X^2 + C_3 X^3 + C_4 X^4 \end{cases}$$

Pour des polynômes de degré 3, la complexité de la multiplication de Toom Cook 3 est de $5M_p + 11CM_p + 11A_p$. De la même façon que la multiplication de Karatsuba, il est possible d'appliquer la multiplication de Toom Cook 3 récursivement pour des polynômes de degré supérieurs à 3. La complexité asymptotique de la multiplication de Toom Cook 3 est en $O(m^{\log_3(5)})$ multiplications et $O(m)$ additions, où m représente le degré des polynômes que l'on multiplie.

II.3.6 La méthode FFT

Principe de la FFT

L'algorithme FFT, de l'acronyme anglais Fast Fourier Transformation, est un algorithme permettant de calculer la transformée de Fourier. D'un point

de vue asymptotique, l'utilisation de la transformée de Fourier permet de calculer le produit de deux polynômes plus rapidement que les algorithmes de multiplications que nous avons présentés. Il s'agit d'une multiplication par interpolation, pour laquelle les points d'interpolation sont des racines k-ième de l'unité.

La multiplication via la transformée de Fourier est très efficace pour des polynômes de degrés des puissances de 2. Nous la décrivons pour le produit de deux polynômes dans un extension de degré $k = 2^m$, pour $m \in \mathbb{N}^\star$. La transformée de Fourier consiste à représenter un polynôme par ses valeurs en les puissances de $\omega \in \mathbb{F}_p$, pour ω une racine k-ième primitive de l'unité. L'opération inverse de la transformée de Fourier consiste à retrouver les coefficients d'un polynôme à partir des valeurs de ce polynôme en les puissances de ω. Ceci permet d'illustrer le fait que la transformée de Fourier est une représentation rappelant l'interpolation.

Soit $A(X) = \sum_{j=0}^{k-1} a_j X^j$, pour tout $i \in \{0, 1, \ldots, (k-1)\}$, $A(\omega^i) = \sum_{j=0}^{k-1} a_j \omega^{ij} = \widehat{a_i}$.La méthode FFT repose sur la remarque suivante.

***Remarque* II.3.2.** Soit k pair et ω une racine k-ième primitive de l'unité dans $\mathbb{F}_p$. Alors $\omega^{k/2} = -1$.

Cette remarque permet de calculer la transformée de Fourier d'un polynôme récursivement. En fait l'égalité $\omega^{k/2} = -1$ permet de séparer le calcul de la transformée de Fourier de A en deux, et d'appliquer une méthode de diviser pour régner, sur laquelle est basé l'algorithme FFT. La séparation est faite suivant la parité des indices, comme nous l'illustrons ci dessous.

$$\begin{cases} A(\omega^i) &= \sum_{j=0}^{k-1} a_j (\omega^i)^j \\ &= \sum_{j=0}^{\frac{k}{2}-1} a_{2j} (\omega^i)^{2j} + \sum_{j=0}^{\frac{k}{2}-1} a_{2j+1} (\omega^i)^{2j+1} \\ &= \sum_{j=0}^{k/2-1} a_{2j} (\omega^2)^{ij} + \omega^i \left(\sum_{j=1}^{k/2-1} a_{2t+1} (\omega^2)^{kj} \right) \\ &= AP(\omega^2) + \omega^i AI(\omega^2) \end{cases}$$

où AP est le polynôme construit en prenant les facteurs de A d'indices pairs, et le polynôme AI les indices impaires. AP et AI sont des polynômes de degré la moitié du degré de A. Le calcul de la transformée de Fourier du polynôme A revient à effectuer le calcul de la transformée de Fourier des

polynômes AP et AI, en considérant les puissances de ω^2 qui est une racine $\frac{k}{2}$-ième de l'unité : $(\omega^2)^{k/2} = 1$.

Pour k divisible par 4, nous pouvons appliquer la même transformation au polynôme AP : retrouver la transformée de Fourier de AP revient à calculer la transformée de deux polynômes de degré moitié, en considérant ω^4 qui est une puissance $\frac{k}{2^2}$-ième de l'unité : $(\omega^{2^2})^{k/2^2} = 1$. Nous réitérons le procédé jusqu'à ce que les polynômes de degré moitié soient des polynômes constants. En remontant les égalités nous construisons ainsi la transformée de Fourier du polynôme A récursivement.

La reconstruction de la transformée de Fourier de A à partir des transformée de Fourier de AP et AI est simplifiée par la propriété suivante.

Propriété II.3.3. *Soit k un entier pair, $A(X)$ un polynôme de degré k et ω une racine k-ième primitive de l'unité ω telle que $\omega^{\frac{k}{2}} = -1$. Alors nous savons que*

$$\begin{cases} A(\omega^i) &= \sum_{j=0}^{k/2-1} a_{2j}(\omega^2)^{ij} + \omega^i \sum_{j=0}^{k/2-1} a_{2j+1}(\omega^2)^{ij} \\ A(\omega^{i+k/2}) &= \sum_{j=0}^{k/2-1} a_{2j}(\omega^2)^{ij} - \omega^i \sum_{j=0}^{k/2-1} a_{2j+1}(\omega^2)^{ij} \end{cases}$$

Algorithme 4 : FFT(A, k, ω)

Données : A un polynôme de degré $(k-1)$ et ω une racine k-ième de l'unité ;
Résultat : La transformée de Fourier de A notée $\hat{A} = [A(\omega^0), A(\omega^1), \ldots, A(\omega^{(k-1)})]$;

Construction des polynômes AP et AI :
pour $j = 0$ **to** $\frac{k}{2} - 1$ **faire**
  $AP_j \leftarrow A_{2j}$
  $AI_j \leftarrow A_{2j+1}$
fin

Appel récursif de l'algorithme FFT :
$\hat{AP} \leftarrow \text{FFT}(AP, \frac{k}{2}, \omega^2)$ construit le vecteur
$\hat{AP} = [AP((\omega^2)^0), AP((\omega^2)^1), \ldots, AP((\omega^2)^{(\frac{k}{2}-1)})]$
$\hat{AI} \leftarrow \text{FFT}(AI, \frac{k}{2}, \omega^2)$,
$\hat{AI} = [AI((\omega^2)^0), AI((\omega^2)^1), \ldots, AI((\omega^2)^{(\frac{k}{2}-1)})]$

Reconstruction de $\hat{A}$ à l'aide des transformées de Fourier de AP et AI.
pour $j = 0$ **to** $\frac{k}{2} - 1$ **faire**
  $\hat{A}_j \leftarrow \hat{AP}_j + \omega^i \hat{AI}_j$
  $\hat{A}_{j+\frac{k}{2}} \leftarrow \hat{AP}_j - \omega^i \hat{AI}_j$
  retourner $\hat{A}$
fin

L'Algorithme 4 construit la transformée de Fourier d'un polynôme récursivement. La complexité de cet algorithme est de $\frac{k}{2}\log_2(k)$ multiplications par des puissances de ω et $k\log_2(k)$ additions dans $\mathbb{F}_p$. Nous devons faire deux fois plus d'additions que de multiplications, car les produits viennent du calcul des $\omega^i \hat{AI}_j$, que nous calculons une seule fois et réutilisons, alors que les additions sont faites dans les deux cas. L'opération inverse de la transformée de Fourier s'effectue avec le même algorithme, en considérant des produits par ω^{-1}. Les deux opérations ont donc la même complexité $\frac{k}{2}\log_2(k)M_\omega + k\log_2(k)A_p$, où nous notons M_ω une multiplication par une puissance de ω, et A_p une addition dans $\mathbb{F}_p$.

***Remarque* II.3.4.** Les multiplications par w sont des multiplications par des constantes dans $\mathbb{F}_p$.

Multiplication de polynôme en utilisant la FFT

La méthode de multiplication en utilisant la transformée de Fourier permet de simplifier la complexité de la multiplication de deux polynômes, en suivant

un principe d'interpolation.

Soient $A(X)$ et $B(X)$ deux polynômes de degré $k = 2^m$ dont nous souhaitons calculer le produit $C(X)$ en utilisant la méthode de multiplication par FFT. Le mode opératoire est le suivant :

1. Trouver la transformée de Fourier de $A(X)$ et $B(X)$ en appliquant l'algorithme FFT.

2. Calculer la transformée de Fourier de $C(X)$ en sachant que $C(\omega^j) = A(\omega^j) \times B(\omega^j)$.

3. Appliquer l'algorithme inverse de la FFT pour retrouver les coefficients de C.

La complexité de la multiplication utilisant la transformée de Fourier est de $3k\log_2(k)CM_p + 6k\log_2(k)A_p + kM_p$. Elle est le résultat de la complexité de trois applications de l'algorithme FFT et des k produits pour trouver la transformée de Fourier du polynôme $C(X)$. Cette multiplication n'est efficace que pour des polynômes de très grand degré.

Chacune des multiplications présentées trouve un intérêt d'utilisation suivant les degrés des polynômes manipulés. La librairie NTL a été conçue par Victor Shoup afin d'effectuer très rapidement tous les calculs d'arithmétique polynomiale. Il s'agit d'une librairie portable en C++ qui fournit des structures de données et des algorithmes pour manipuler des entiers signés de longueur arbitraire et des vecteurs, matrices et polynômes sur les entiers et sur des corps finis. La librairie NTL utilise les multiplications naïve, de Karatsuba, Toom Cook et FFT. Le choix de chacune de ces multiplications est fait de manière heuristique par l'auteur [Sho96]. Par la suite, nous avons privilégié la méthode de multiplication minimisant le nombre d'opérations sur le corps base.

II.4 Optimisations mathématiques

Nous présentons les améliorations du calcul des couplages reposant sur l'utilisation de la structure de groupe des courbes elliptiques.

II.4.1 La tordue d'une courbe elliptique

La définition de la tordue d'une courbe elliptique a été donnée dans la partie I.3.5. Une courbe elliptique peut admettre un twist de degré 2, 3, 4 ou 6. Nous nous intéressons ici aux courbes tordues pour un twist de degré pair. Pour simplifier l'écriture, nous écrivons les formules pour un twist de degré

2. La démarche et les résultats liés à l'utilisation de ce twist de degré 2 se transposent facilement pour les twist de degré 4 et 6. Le cas des twists de degré 3 sera traité dans le Chapitre IV. L'utilisation d'une tordue de la courbe elliptique $E(\mathbb{F}_{p^k})$ permet d'effectuer certaines opérations liées à l'application de l'algorithme de Miller dans un sous-corps de $\mathbb{F}_{p^k}$, plutôt que dans $\mathbb{F}_{p^k}$, et ainsi de simplifier les calculs. Les courbes tordues permettent aussi la simplification des calculs en rendant inutile le calcul des dénominateurs (i.e. la mise à jour de la fonction f_2) durant l'algorithme de Miller décrit dans la partie I.8.2.

Nous noterons $\widetilde{E}(\mathbb{F}_{p^{k/2}})$ la courbe tordue de $E(\mathbb{F}_{p^k})$. Nous pouvons tout d'abord remarquer que la tordue de E est une courbe elliptique définie sur une extension de corps de degré la moitié de celui de l'extension finale $\mathbb{F}_{p^k}$ [BLS04b].

Soit $\nu \in \mathbb{F}_{p^{k/2}}$ un élément qui n'est pas un carré dans $\mathbb{F}_{p^{k/2}}$, alors $\sqrt{\nu}$ est un élément de $\mathbb{F}_{p^k}$ privé de $\mathbb{F}_{p^{k/2}}$. Nous pouvons à ce moment là définir $\widetilde{E}$ la courbe elliptique définie sur le corps $\mathbb{F}_{p^{k/2}}$ et d'équation $\nu y^2 = x^3 - 3x + b$. L'isomorphisme envoyant un point de $\widetilde{E}(\mathbb{F}_{p^{k/2}})$ sur un point $E(\mathbb{F}_{p^k})$ est Ψ_2 défini par

$$\begin{array}{rcl} \Psi_2 : \widetilde{E}(\mathbb{F}_{p^{k/2}}) & \rightarrow & E(\mathbb{F}_{p^k}) \\ (x, y) & \rightarrow & (x, y\sqrt{\nu}) \end{array}$$

La probabilité que le point $Q = (x, y\sqrt{\nu})$ image de $Q' = (x, y)$ par Ψ_2 appartienne au sous-groupe engendré par $P \in E(\mathbb{F}_p)$ est faible, comme le montre [BLS04b]. Ceci nous permet d'assurer la non-dégénérescence du couplage entre les points $P \in E(\mathbb{F}_p)$ et $Q = \Psi_2(Q')$. L'utilisation de ce twist permet de considérer que les coordonnées du point Q sont des éléments de $\mathbb{F}_{p^{k/2}}$, à multiplication par $\sqrt{\nu}$ près.

Nous rappelons les formules de l'algorithme de Miller avant utilisation d'une courbe tordue :

Soient A, B, C, D, E et F les variables intermédiaires utilisées pendant le doublement et l'addition d'un point en coordonnées Jacobiennes, leurs formules sont données dans la partie I.2.5. Ces variables dépendent uniquement des coordonnées du point $P = (X_P; Y_P; Z_P)$, et des multiples de P : $T = (X_T; Y_T; Z_T)$; $2T = (X_{2T}; Y_{2T}; Z_{2T})$ et $T + P = (X_3; Y_3; Z_3)$. Nous rappelons les équations des fonctions l_1, l_2, v_1 et v_2 intervenant dans l'algorithme de Miller.

$$\begin{aligned} l_1(x_Q, y_Q\sqrt{\nu}) &= Z_P^2(Z_{2T}Dy_Q\sqrt{\nu} - B(Dx_Q - X_T) - 2Y_T), \\ v_1(x_Q, y_Q\sqrt{\nu}) &= Z_{2T}^2 Z_P x_Q + 4Y^2(X_P D + X_T Z_P^2) - 9Z_P^2(X_T^2 - Z_T^4)^2, \\ l_2(x_Q, y_Q\sqrt{\nu}) &= Z_{T+P}^2(Z_T^3 E y_Q\sqrt{\nu} - Z_T F(Z_T^2 x_Q) - Y_T E), \\ v_2(x_Q, y_Q\sqrt{\nu}) &= Z_T^3 E(Z_3^3 x_Q + E(A+B) - Z_T^2 Z_P^2 F). \end{aligned} \tag{II.4.1}$$

Les multiplications et additions intervenant dans ces formules s'effectuent dans le corps $\mathbb{F}_p$ et le corps $\mathbb{F}_{p^{k/2}}$. Pour $x_Q \in \mathbb{F}_{p^{k/2}}$, d'après les équations de v_1 et v_2, il est intéressant de noter que les résultats $v_1(x_Q, y_Q\sqrt{\nu})$ et $v_2(x_Q, y_Q\sqrt{\nu})$ sont des éléments de $\mathbb{F}_{p^{k/2}}$. En effet, d'après les formules des équations II.4.1 l'ordonnée de Q n'intervient pas, et donc $\sqrt{\nu}$ non plus.

Nous allons voir que cette remarque permet l'élimination des dénominateurs lors de l'exécution de l'algorithme de Miller pour le calcul du couplage de Tate.

Propriété II.4.1. *Lors de l'exécution de Miller Lite pour le couplage de Tate, l'évaluation de la fonction f_2 et donc des lignes v_1 et v_2 peut être omise [BLS04b].*

En effet, en utilisant une courbe tordue, les équations de v_1 et v_2 montrent que $v_1(Q)$, $v_2(Q) \in \mathbb{F}_{p^{k/2}}$, et donc que $f_2 \in \mathbb{F}_{p^{k/2}}$. Par définition du degré de plongement k de la courbe elliptique, $\frac{p^k-1}{r}$ est un multiple de $p^{k/2}-1$, ainsi $f_2^{\frac{p^k-1}{r}} = 1$, comme le montre la proposition suivante.

Propriété II.4.2. *Soit r un diviseur premier de $\#E(\mathbb{F}_p)$ pour E une courbe elliptique de degré de plongement k. Alors $\frac{p^k-1}{r}$ est un multiple de $p^{k/2}-1$.*

Démonstration. La démonstration est une conséquence directe de la construction de k comme étant le plus petit entier pour lequel r divise $p^k - 1$. Nous savons que, pour k pair, $p^k - 1 = (p^{k/2} - 1)(p^{k/2} + 1)$. r est premier et divise $p^k - 1$ par construction de k. D'après le théorème de Gauss, r divise donc $(p^{k/2} - 1)$ ou $(p^{k/2} + 1)$. Si r divisait $(p^{k/2} - 1)$ alors la définition de k serait mise en défaut. La seule possibilité est donc que r divise $(p^{k/2} + 1)$. □

Pour tout élément $\xi \in \mathbb{F}_{p^{k/2}}$, nous savons que $\xi^{p^{k/2}-1} \equiv 1$ (d'après le petit théorème de Fermat). Par conséquent, l'exponentiation finale à la puissance $\frac{p^k-1}{r}$ neutralise n'importe quel facteur du résultat appartenant à un sous-corps propre de $\mathbb{F}_{p^k}$.

Ainsi, les formules de l'étape de doublement peuvent se simplifier en tenant compte du fait que les éléments appartenant à un sous corps propre de $\mathbb{F}_{p^k}$ seront éliminés par l'exponentiation finale. Il n'est alors plus nécessaire de calculer les dénominateurs v_1 et v_2. De même, les facteurs Z_P^2 et Z_{T+P}^2 des

fonctions l_1 et l_2 appartiennent à $\mathbb{F}_p$ et sont neutralisés par l'exponentiation finale. Les formules des fonctions $l_1(x_Q, y_Q)$ et $l_2(x_Q, y_Q)$ se simplifient en les Formules II.4.2.

$$\begin{aligned} l_1(x_Q, y_Q\sqrt{\nu}) &= Z_{2T} D y_Q \sqrt{\nu} - B(Dx_Q - X_T) - 2Y_T \\ l_2(x_Q, y_Q\sqrt{\nu}) &= Z_T^3 E y_Q \sqrt{\nu} - Z_T F(Z_T^2 x_Q) - Y_T E \end{aligned} \tag{II.4.2}$$

L'utilisation des courbes tordues permet d'effectuer la majorité des calculs durant l'algorithme de Miller dans les corps $\mathbb{F}_p$ et $\mathbb{F}_{p^{k/2}}$ plutôt que dans $\mathbb{F}_{p^k}$. De plus, elle simplifie les calculs en permettant l'élimination des dénominateurs dans l'algorithme de Miller.

Cette propriété semble ne pouvoir s'appliquer que pour le couplage de Tate, car le couplage de Weil ne nécessite pas d'exponentiation finale. Mais, élever le résultat du couplage de Weil à une puissance fixée non nulle qui ne soit pas un multiple de r ne modifiera pas ses propriétés. Au prix d'une exponentiation supplémentaire à la fin du calcul du couplage de Weil il est possible de s'affranchir du calcul des dénominateurs durant les opérations de Miller Lite et de Miller Full. Nous pouvons par exemple choisir d'élever le résultat du couplage de Weil à la puissance $p^{k/2} - 1$, et analyser la complexité du couplage $e_w(P, Q)^{p^{k/2}-1}$. Le coût de l'exponentiation sera étudié dans la partie II.4.4.

L'utilisation d'une courbe tordue permet de ramener le coût de l'étape de doublement pour le calcul de Miller Lite à $4S_p + (7+k)M_p + S_{p^k} + M_{p^k}$ opérations dans $\mathbb{F}_p$ et $\mathbb{F}_{p^k}$.
Le calcul de Miller Full revient lui à $kM_p + 5S_{p^k} + 7M_{p^k}$.

Le Tableau II.3 compare les coûts des opérations Miller Lite et Miller Full, avant et après utilisation des twists.

Miller	Sans twist	Avec twist
Lite	$8S_p + (12+4k)M_p + 2S_{p^k} + 2M_{p^k}$	$4S_p + (7+k)M_p + S_{p^k} + M_{p^k}$
Full	$3kM_p + 10S_{p^k} + 14M_{p^k}$	$kM_p + 5S_{p^k} + 7M_{p^k}$

TABLE II.3 – Coût de Miller Lite et Miller Full

II.4.2 Les corps amis

Les couplages nécessitent des calculs dans des extensions de corps de la forme $\mathbb{F}_{p^k}$. Si le degré de plongement k de la courbe est *friable*, alors l'arithmétique dans $\mathbb{F}_{p^k}$ peut se faire par étape. Nous donnons une définition intuitive de la friabilité d'un nombre, une définition plus précise peut être trouvée dans [LN94].

Définition II.4.3. Un ***entier friable*** est un entier dont la décomposition en facteurs premiers est composée uniquement de petits facteurs.

***Exemple* II.4.4.** Un nombre premier de la forme 2^i3^j est donc considéré friable.

Nous allons illustrer l'utilisation de la friabilité d'un nombre pour construire une extension de corps fini par étape.

***Exemple* II.4.5.** Soit l un nombre premier et m un entier quelconque tels que $k = lm$. L'extension $\mathbb{F}_{p^k}$ peut être construite comme étant une extension de degré l du corps $\mathbb{F}_{p^m}$. Nous supposons que nous avons construit $\mathbb{F}_{p^m}$. Soit $P(X)$ un polynôme irréductible de degré l dans $\mathbb{F}_{p^m}[X]$. Nous construisons $\mathbb{F}_{(p^m)^l}$ en considérant le quotient :

$$\mathbb{F}_{p^{lm}} = \mathbb{F}_{p^m}[X]/(P(X))$$

L'application de la multiplication la plus adaptée à chaque étape permet d'optimiser le produit. Par exemple, pour des tours extensions de degré 2 et 3, les multiplications de Karatsuba et Toom Cook permettent d'utiliser la friabilité de k en construisant l'extension $\mathbb{F}_{p^k}$ par une succession d'extensions de plus petits degrés. Cette tour d'extension permet de diminuer le nombre d'opérations nécessaires sur les corps de base $\mathbb{F}_p$ pour faire une multiplication dans $\mathbb{F}_{p^k}$ [PFSR99].

A.Menezes et N.Koblitz [KM05] proposent d'utiliser des « **corps amis** » pour améliorer le calcul des couplages. Les corps amis, dont la définition suit, se caractérisent par un degré d'extension k très friable.

Définition II.4.6. Un **corps ami** $\mathbb{F}_{p^k}$ est une extension de corps fini $\mathbb{F}_p$ possédant les caractéristiques suivantes :

- p la caractéristique du corps vérifie que $p \equiv 1 \ mod(12)$,
- k le degré de plongement de la courbe est tel que $k = 2^i3^j$.

Les corps adaptés au calcul des couplages possèdent une propriété intéressante rendant la réduction polynomiale dans l'extension $\mathbb{F}_{p^k}$ très facile à réaliser, comme le montre le théorème suivant [LN94, Theorem 3.75].

Théorème II.4.7. *Soit $\beta \in \mathbb{F}_p$ qui ne soit ni un carré ni un cube dans $\mathbb{F}_p$, et $\mathbb{F}_{p^k}$ un corps adapté au calcul des couplagesdonc $k = 2^i3^j$. Alors le polynôme $X^k - \beta$ est irréductible dans $\mathbb{F}_p$.*

D'après la définition et la propriété précédente, nous pouvons construire l'extension $\mathbb{F}_{p^k} = \mathbb{F}_p[X]/(X^k - \beta)$, en utilisant des extensions successives de degré 2 (extension quadratique) et 3 (extension cubique). La construction se fait pas à pas en considérant des racines cubiques ou carrés de β et des

résultats. β étant choisi librement, nous pouvons le choisir comme étant une petite valeur dans $\mathbb{F}_p$, de sorte que la multiplication par β dans $\mathbb{F}_p$ se retrouve réduite à quelques additions.

***Exemple* II.4.8.** Exemple de tours de corps possible pour $k = 2^2 3^1$:

$$\begin{aligned} \mathbb{F}_p \overset{2}{\to} L &= \mathbb{F}_p[T]/(T^2 - \beta) \\ K \overset{3}{\to} M &= L[U]/(U^3 - T) \\ L \overset{2}{\to} N &= M[V]/(V^2 - U) \end{aligned}$$

Les représentations des corps L, M et N sont :

$$\begin{aligned} L &= \{l_0 + l_1\beta, \text{ avec } l_0, l_1 \in \mathbb{F}_p\} \\ M &= \{m_0 + m_1 T + m_2 T^2, \text{ avec } m_0, m_1, m_2 \in L\} \\ N &= \{n_0 + n_1 U, \text{ avec } n_0, n_1 \in M\} \end{aligned}$$

L'arithmétique dans $\mathbb{F}_{p^k}$ se fait alors étape par étape dans chacune des petites extensions. Étant donné que k n'admet que 2 et 3 en facteurs premiers, les méthodes de multiplications de Karatsuba et de Toom Cook optimisent la multiplication dans l'extension $\mathbb{F}_{p^k}$. Nous pouvons alors estimer qu'une multiplication dans $\mathbb{F}_{p^k}$ avec $k = 2^i 3^j$ nécessite $3^i 5^j$ multiplications dans $\mathbb{F}_p$. Ce qui est noté $M_{p^k} = 3^i 5^j M_p$.

II.4.3 Sous-groupe cyclotomique et carré

A. Lenstra et M. Stam introduisent dans [LS03] une méthode originale et très efficace pour améliorer le calcul d'un carré dans un sous-groupe cyclotomique. Leur méthode repose sur la construction d'une extension de degré 6 à l'aide d'un trinôme particulier plutôt qu'un polynôme de la forme $X^6 - \beta$. Le **sous-groupe cyclotomique** $G_{\phi_k(p)}$ est le sous-groupe d'ordre $\phi_k(p)$ de $\mathbb{F}^\star_{p^k}$, où $\phi_k(p)$ représente le k-ième polynôme cyclotomique évalué en p. Les « **polynômes cyclotomiques** » sont les polynômes construits de telle sorte que leurs racines sont les racines primitives de l'unité.

Définition II.4.9. Le k-ième ***polynôme cyclotomique*** est le polynôme admettant pour racine exactement les racines k-ème primitives de l'unité dans $\mathbb{C}$. Autrement dit, le k-ième polynôme cyclotomique est le polynôme minimal des racines k-ième de l'unité dans $\mathbb{C}$.

Les racines primitives de l'unité sont les éléments ω tels que $\omega^k = 1$, mais quel que soit d un diviseur strict de k, $\omega^d \neq 1$.

***Exemple* II.4.10.** 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 multiplication de Lenstra et Stam est intéressante pour effectuer les carrés dans les extensions de $\mathbb{F}_p$ de degré 6 ou un multiple de 6. Pour l'instant elle n'a pas été généralisée pour des degrés d'extensions plus généraux.

Leur méthode repose sur la construction de l'extension de degré 6 à l'aide du polynôme cyclotomique $\phi_k(X) = X^{k/3} - X^{k/6} + 1$. Leur méthode est applicable pour toutes les extensions dont le degré est un multiple de six. Nous garderons donc des notations générales, pour k multiple de six et développerons les exemples pour $k = 6$.

Soit $\alpha \in G_{\phi_k(p)}$, $\alpha = \sum_{i=0}^{k-1} a_i \gamma^i$, où les $a_i \in \mathbb{F}_p$, et $\mathcal{B} = (1, \gamma, \gamma^2, \ldots, \gamma^{k-1})$ est une base de $\mathbb{F}_{p^k}$.

Nous voulons déterminer l'expression générale d'un élément de $G_{\phi_k(p)}$. Pour cela, nous considérons que α est un polynôme à plusieurs variables, dont les coefficients sont les puissances de γ dans $\mathbb{F}_{p^k}$, et dont les variables sont les a_i dans $\mathbb{F}_p$.

Sachant que α appartient au sous-groupe cyclotomique $G_{\phi_k(p)}$, il est d'ordre divisant le le cardinal de $G_{\phi_k(p)}$. Or le cardinal de $G_{\phi_k(p)}$ est $\phi_k(p)$. Nous savons donc que $\alpha^{p^{k/3}-p^{k/6}+1} = 1$ dans $G_{\phi_k(p)}$. Cette égalité peut être transformée en la suivante $\alpha^{p^{k/3}+1} = \alpha^{p^{k/6}}$.

Nous pouvons alors calculer formellement les puissances $\alpha^{p^{k/3}}$, $\alpha^{p^{k/6}}$ de α, pour retrouver la décomposition de $\alpha \times \alpha^{p^{k/3}} - \alpha^{p^{k/6}}$ en fonction des a_i :

$$\alpha \times \alpha^{p^{k/3}} - \alpha^{p^{k/6}} = \sum_{i=0}^{k-1} v_i \gamma^i.$$

Où

$$\begin{aligned}
v_0 &= a_1^2 - a_0 a_2 - a_4 - a_4^2 + a_3 a_5, \\
v_1 &= -a_0 + a_1 a_2 + a_3 - 2a_0 a_3 + a_3^2 - a_2 a_4 - a_1 a_5, \\
v_2 &= -a_0 a_1 + a_3 a_4 - a_5 - 2a_2 a_5 + a_5^2, \\
v_3 &= -a_1 - a_2 a_3 + 2a_1 a_4 - a_4^2 - a_0 a_5 + a_3 a_5, \\
v_4 &= a_0^2 + a_1 a_2 + a_3 - 2a_0 a_3 - a_4 a_5, \\
v_5 &= -a_2 + a_2^2 - a_1 a_3 - a_0 a_4 + a_3 a_4 - 2a_2 a_5.
\end{aligned}$$

α est un élément de $G_{\phi_k(p)}$, ce qui impose que $\sum_{i=0}^{k-1} v_i\gamma^i = 0$. Cette condition nous permet de construire un système en les α_i, dont la résolution nous donnera la forme générale d'un élément de $G_{\phi_k(p)}$.

Le sous-groupe $G_{\phi_k(p)}$ correspond à l'ensemble déterminé par les α tels que $\forall i,\ v_i = 0$. Nous pouvons donc écrire symboliquement l'égalité suivante $\alpha^2 = \alpha^2 + \mathcal{B}.\Gamma.{}^t v$, avec $\mathcal{B} = (1, \gamma, \gamma^2, ..., \gamma^{k-1})$ et où Γ est une matrice choisie de façon à minimiser le coût d'un carré. Cette écriture nous permet de réduire le coût d'un carré, sachant que v est un représentant de zéro dans $\mathbb{F}_p$.

En notant $\alpha^2 = \sum_{i=1}^{k} s_i\gamma^i$, nous obtenons l'égalité

$$\sum_{i=1}^{k} s_i\gamma^i = (\sum_{i=1}^{k} a_i\gamma^i)^2 + \mathcal{B}.\Gamma^t.v.$$

Nous pouvons développer formellement le terme de droite, et pour une matrice Γ bien choisie, les formules pour calculer un carré dans $\mathbb{F}_{p^k}$ seront simplifiées. Cette écriture nous permet de réduire le coût d'un carré.

Par exemple, pour $k = 6$ [LS03] :

$$\alpha^2 = \mathcal{B}.\begin{pmatrix} 2a_1 + 3a_4(a_4 - 2a_1) \\ 2a_0 + 3(a_0 + a_3)(a_0 - a_3) \\ -2a_5 + 3a_5(a_5 - 2a_2) \\ 2(a_2 - a_4) + 3a_1(a_1 - 2a_4) \\ 2(a_0 - a_3) + 3a_3(2a_0 - a_3) \\ -2a_2 + 3a_2(a_2 - 2a_5) \end{pmatrix} \tag{II.4.3}$$

Granger, Page et Smart dans [GPS06] ont appliqué cette méthode pour obtenir les coûts donnés dans le Tableau II.4.

Degré k de l'extension	coût d'un carré dans $\mathbb{F}_{p^k}$
6	$4,5M_p$
12	$18M_p + 12S_p$
24	$84M_p + 24S_p$

TABLE II.4 – Coût d'un carré en nombre d'opérations élémentaires dans $\mathbb{F}_p$

La remarque importante est que dans le cas ou $k = 6$ et $p \equiv 2 \pmod 9$, l'élévation au carré se calcule efficacement en utilisant la méthode de Lenstra et Stam, et pour un coût inférieur à $0,75M_{p^k}$, qui est souvent considéré comme le coût d'un carré par rapport à un produit [BL07].

***Exemple* II.4.11.** Dans $\mathbb{F}_{p^6}$, un carré avec la méthode de Lenstra et Stam se calcule en $6 \times 0,75M_p \approx 4,5M_p$. Alors qu'avec l'approximation classique un carré dans $\mathbb{F}_{p^6}$ s'effectuerait en $15 \times 0,75M_p \approx 10M_p$.

II.4.4 Améliorer l'efficacité du calcul de l'exponentiation finale

Le couplage de Tate peut se diviser en deux étapes, d'abord le calcul de la fonction de Miller Lite, puis l'exponentiation de ce résultat. Cette exponentiation finale est faite dans l'extension $\mathbb{F}_{p^k}$, sa complexité dépend de l'exposant $\frac{p^k-1}{r}$ qui est grand. Afin de simplifier le calcul de cette exponentiation, Koblitz et Menezes [KM05] l'ont divisé en deux parties en remarquant que

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

où $\phi_k(p)$ représente l'évaluation en p du k-ième polynôme cyclotomique.

Cette ré-écriture de l'exposant est possible car $\phi_k(p)$ divise (p^k-1), ce que nous rappelons dans la Propriété II.4.12, qui permet aussi de démontrer que r divise $\phi_k(p)$. Nous rappelons que les polynômes cyclotomiques permettent de factoriser les polynômes de la forme X^k-1.

Propriété II.4.12. *Soit k un entier, alors le polynôme X^k-1 se factorise à l'aide des polynômes cyclotomiques :*

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

Le fait que $\phi_k(p)$ divise (p^k-1) est une conséquence directe de la Propriété II.4.12. Cette propriété nous permet aussi de conclure facilement que r divise $\phi_k(p)$. En effet,

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

Or, nous savons que k est le plus petit entier tel que r divise (p^k-1). Si r divisait $\phi_{k'}(p)$ pour k' un diviseur strict de k, alors r diviserait $p^{k'}-1$ avec $k' < k$, ce qui contredirait la définition de k. Ainsi, r divise $\phi_k(p)$.

Le calcul de la première partie utilise les propriétés des courbes tordues et des corps amis, cette partie de l'exponentiation revient à calculer des Frobenius d'éléments de $\mathbb{F}_{p^k}$; la seconde partie est une exponentiation réduite dans l'extension $\mathbb{F}_{p^k}$ qui ne peut être faite qu'en employant des méthodes d'exponentiation classiques.

Nous allons étudier plus en détails le calcul de chaque étape de l'exponentiation, mais avant cela, nous faisons une remarque sur l'inversion d'un élément de $\mathbb{F}_{p^k}$.

Une remarque sur l'inversion

Soit r l'ordre des sous-groupes intervenants dans le calcul des couplages. Nous avons précédemment remarqué grâce à la Propriété II.4.12 que pour

k pair r divise $p^{k/2}+1$ (ceci est une conséquence du fait que k est la plus petite valeur entière pour laquelle r divise p^k-1).

Notons α le résultat de l'algorithme de Miller, α est un élément d'ordre r dans $\mathbb{F}^\star_{p^k}$. Il est alors intéressant de souligner que $\alpha^{-1}=\alpha^{p^{k/2}}$, car :

$$\begin{cases} \alpha\times\alpha^{p^{k/2}} &= \alpha^{p^{k/2}+1} \\ &= \alpha^{r.\frac{p^{k/2}+1}{r}} \\ &= 1 \end{cases}$$

De ce fait, l'inversion finale lors du calcul du couplage de Weil est réduite à une exponentiation à la puissance $p^{k/2}$, ce qui revient au coût de l'application d'un Frobenius, pouvant se calculer comme nous l'indiquons ci-dessous.

Première partie de l'exponentiation :

Nous analysons la complexité de l'exponentiation à la puissance $\frac{p^k-1}{\phi_k(p)}$. Nous pouvons tout d'abord remarquer que si $k=2^i3^j$, alors $\phi_k(p)=p^{k/3}-p^{k/6}+1$ et $\frac{p^k-1}{\phi_k(p)}=(p^{k/2}-1)(p^{k/6}+1)$.

L'utilisation d'une courbe tordue, que nous avons introduite dans la partie II.4.1 nous permet d'écrire le résultat de l'algorithme de Miller sous la forme $(X+Y\sqrt{\nu})$ avec $X,Y\in\mathbb{F}_{p^{k/2}}$.

Le calcul de $(X+Y\sqrt{\nu})^{p^{k/2}-1}$ peut se décomposer en $(X+Y\sqrt{\nu})^{p^{k/2}}\times(X+Y\sqrt{\nu})^{-1}$.

Nous venons de voir que $(X+Y\sqrt{\nu})^{-1}=(X+Y\sqrt{\nu})^{p^{k/2}}$, ainsi,

$$(X+Y\sqrt{\nu})^{p^{k/2}-1}=(X+Y\sqrt{\nu})^{2p^{k/2}}.$$

L'élévation à la puissance $p^{k/2}$ est une opération de Frobenius, nous détaillons ci-après comment l'exécuter en effectuant essentiellement des décalages. Au total l'élévation à la puissance $(p^{k/2}-1)$ du résultat de l'algorithme de Miller coûte un carré dans $\mathbb{F}_{p^k}$, et une application du Frobenius. Nous notons $(X'+Y'\sqrt{\nu})$ le résultat de $(X+Y\sqrt{\nu})^{p^{k/2}-1}$.

Il ne reste plus qu'à évaluer $(X'+Y'\sqrt{\nu})^{p^{k/6}+1}$ ce qui correspond aussi à une application du Frobenius.

Soit γ une racine dans $\mathbb{F}_{p^k}$ du polynôme $X^k-\beta$. L'élément $a\in\mathbb{F}_{p^k}$ peut s'écrire sous la forme $a=\sum_{i=0}^{k-1}a_i\gamma^i$, avec $a_i\in\mathbb{F}_p$. Les propriétés des corps finis

nous permettent d'obtenir l'égalité $a^p = \sum_{i=0}^{k-1} a_i \gamma^{ip}$, et par récurrence que

$$a^{p^j} = \sum_{i=0}^{k-1} a_i \gamma^{ip^j}.$$

Pour i et j deux entiers fixés, notons q_{ij} et r_{ij} le quotient et le reste de la division de ip^j par k, nous savons alors que

$$\gamma^{ip^j} = \beta^{q_{ij} \bmod(p)} \gamma^{r_{ij}}.$$

Ces valeurs peuvent être précalculées une fois pour toutes, en effet elles ne dépendent que des éléments de la base.

Le calcul de $(X' + Y'\sqrt{\nu})^{p^{k/6}+1}$ peut donc se décomposer comme suit.

$$(X' + Y'\sqrt{\nu})^{p^{k/6}+1} = (X'^{p^{k/6}} + Y'^{p^{k/6}} \sqrt{\nu}^{(p^{k/6})}) \times (X' + Y'\sqrt{\nu})$$

Par exemple, si nous écrivons ce qui se passe pour la variable X' élevée à la puissance $p^{k/6}$, nous obtenons les étapes de calculs suivantes.

$$\begin{cases} X' &= \sum_{i=0}^{k/2-1} x_i \gamma^i, \\ X'^{p^{k/6}} &= \sum_{i=0}^{k/2-1} x_i \gamma^{ip^{(k/6)}}, \\ X'^{p^{k/6}} &= \sum_{i=0}^{k/2-1} (x_i \beta^{q_{i(k/6)}} \mod (p)) \gamma^{r_{i(k/6)}}. \end{cases}$$

Nous devons donc calculer les $\frac{k}{2}$ produits $(x_i \beta^{q_{i(k/6)} \bmod(p)})$, or x_i et $\beta^{q_{i(k/6)} \bmod(p)}$ sont des éléments de $\mathbb{F}_p$, ainsi ces produits sont des produits dans $\mathbb{F}_p$. Le coût du calcul de $(X' + Y'\sqrt{\nu})^{p^{k/6}+1}$ est au final de $kM_p + M_{p^k}$. Au total, la première partie de l'exponentiation à une complexité de $2kM_p + S_{p^k} + M_{p^k}$ à laquelle s'ajoutent des décalages et multiplications par des puissances de β. Cette complexité est négligeable par rapport aux opérations nécessaires pour calculer l'algorithme de Miller, ou encore par rapport à la seconde partie de l'exponentiation. Nous ne prendrons pas en compte le coût de cette première partie dans nos comparaisons. Cette omission ne changera pas le résultat final, car cette première partie est commune au couplage de Weil et de Tate.

Deuxième partie : Le second morceau de l'exponentiation est le plus délicat à calculer. Nous donnons les coûts de l'élévation à la puissance $\frac{\phi_k(p)}{r}$ en la calculant à travers deux méthodes classiques : celle des suites de Lucas [SB04] et celle des fenêtres glissantes [GPS05].

La méthode des suites de Lucas permet d'effectuer l'exponentiation avec un coût d'une multiplication et d'un carré dans le corps intermédiaire $\mathbb{F}_{p^{k/2}}$ pour chaque bit de l'exposant.

L'exponentiation à l'aide des fenêtres glissantes présente l'avantage d'effectuer les carrés dans le sous-groupe cyclotomique ; donc nous permet d'utiliser la méthode d'élévation au carré décrite dans la partie II.4.3. Les complexités de ces deux méthodes dépendent linéairement du nombre de bits de l'exposant. Nous ne détaillerons pas ces méthodes d'exponentiation, nous rappellerons leurs complexités, nous renvoyons le lecteur au livre [CF06] décrivant ces méthodes.

Soit b_r le nombre de bits de r, qui est l'ordre des sous-groupes dans lesquels nous effectuons nos calculs. Soit b_{p^k} le nombre de bits de p^k. Les tailles en bit b_r et b_{p^k} de r et p^k sont fixées par les considérations de sécurité que nous avons énoncé dans la partie I.7.4. Nous les rappelons ci-dessous dans le Tableau II.5. Le nombre d'entiers positifs strictement inférieurs à k et premiers avec k est $\varphi(k)$, où $\varphi(x)$ représente la fonction d'Euler appliquée à l'entier x. L'entier $\varphi(k)$ est aussi le nombre de racines primitives k-ième de l'unité, cet entier est donc le degré du polynôme $\phi_k(p)$. L'exposant de la deuxième partie de l'exponentiation comporte alors $(\frac{\varphi(k)}{k} b_{p^k} - b_r)$ bits.

Le nombre de multiplications et carrés nécessaires pour le calcul de l'exponentiation dépend de $\frac{\varphi(k)}{k} b_{p^k} - b_r = (\tau_k \gamma - 1) b_r$, où

$$\gamma = \frac{b_{p^k}}{b_r},$$

$$\tau_k = \frac{\varphi(k)}{k} = \begin{cases} 1/2 & \text{si } k = 2^i,\ i \geqslant 1 \\ 1/3 & \text{si } k = 2^i 3^j,\ i,\ j \geqslant 1 \end{cases}$$

Le nombre γ est lié aux niveaux de sécurité donnés dans le Tableau II.5 et permet de donner une idée approximative de la complexité attendue pour le calcul de l'exponentiation.

Niveau de sécurité en bits	80	128	192	256
Nombre minimal de bits de r	160	256	384	512
Nombre minimal de bits de p^k	1 024	3 072	7 680	15 360
$\gamma = \frac{b_{p^k}}{b_r}$	6,4	12	20	30

TABLE II.5 – Niveau de sécurité

La méthode des suites de Lucas [SB04] est de complexité

$$C_{Luc} = (M_{p^{k/2}} + S_{p^{k/2}}) \log_2 \left(\frac{\phi_k(p)}{r} \right).$$

La méthode des fenêtres glissantes [GPS05] est de complexité

$$C_{sw} = \left(\frac{\log_2(e)}{\log_2(p)} + \log_2(p)\right) S_{G_{\phi_k(p)}} + \left(\frac{\log_2(e)}{\log_2(p)}\left(2^{n-1}-1\right) + \frac{\log_2(e)}{n+2} - 1\right) M_{p^k},$$

où $e = \frac{\phi_k(p)}{r}$, et n est l'entier donnant la taille de la fenêtre en bits, généralement $n = 4$.

Afin de simplifier la complexité de l'exponentiation finale, nous l'avons séparé en deux parties dont les complexités sont différentes. La première nécessite au total $2kM_p + S_{p^k} + M_{p^k}$ opérations, alors que la seconde nécessite au moins une multiplication et un carré dans $\mathbb{F}_{p^{k/2}}$ par bit de $\frac{\phi_k(p)}{r}$. De plus, la première partie est commune au couplage de Weil et au couplage de Tate. Ainsi, sa complexité n'influencera pas les résultats de nos comparaisons d'efficacité. Nous négligerons donc la première partie de l'exponentiation dans nos comparaison de complexité.

II.4.5 Weil ou Tate ?

Nous allons utiliser les diverses optimisations arithmétiques et mathématiques introduites précédemment pour comparer les complexités de calcul des couplages de Weil et de Tate. Nous nous sommes restreint à l'étude du coût de l'étape du doublement dans l'algorithme de Miller.

Nous avons travaillé dans les corps adaptés au calcul des couplages décrits par Koblitz et Menezes. Nous choisissons $f(X) = X^k - \beta$ comme polynôme définissant l'extension de degré k de $\mathbb{F}_p$ et notée $\mathbb{F}_{p^k}$, pour $k = 2^i 3^j$. L'entier β est choisi de sorte qu'il ne soit ni un carré, ni un cube dans $\mathbb{F}_p$, et aussi de telle façon que la multiplication par β dans $\mathbb{F}_p$ consiste en des opérations très simples comme des décalages ou des additions. Les points P et Q dont nous voulons calculer le couplage sont des points d'ordre r de la même courbe elliptique, mais définie sur des corps différents. Le point P appartient à la courbe $E(\mathbb{F}_p)$ et le point $Q \in E(\mathbb{F}_{p^k})$ est représenté comme l'image d'un point de la courbe tordue $\widetilde{E}(\mathbb{F}_{p^{k/2}})$: $Q = (x, y\sqrt{\nu})$ avec $x,\ y,\ \nu \in \mathbb{F}_{p^{k/2}}$ et $\sqrt{\nu} \in \mathbb{F}_{p^k}$. Pour éviter les inversions dues aux doublement et additions sur la courbe elliptique lors de l'exécution de Miller, nous utilisons les coordonnées Jacobiennes. Dans le Tableau II.6, nous présentons les complexités des exécutions de Miller Lite et Miller Full que nous avons calculées, elles sont exprimées en nombre d'opérations dans $\mathbb{F}_p$.

Complexité des couplages de Weil et de Tate

Le calcul du **couplage de Weil** est composé d'une exécution de Miller Lite et d'une exécution de Miller Full, auxquelles s'ajoutent une inversion

Miller Lite C_{Lite}	$(4S_p + (7+k)M_p + S_{p^k} + M_{p^k})\log_2(r)$
Miller Full C_{Full}	$(kM_p + 4S_{p^{k/2}} + 6M_{p^{k/2}} + S_{p^k} + M_{p^k})\log_2(r)$

TABLE II.6 – Complexité globale de l'algorithme de Miller

et un produit dans $\mathbb{F}_{p^k}$ et une exponentiation optionnelle à la puissance $(p^{k/2}-1)$. Cette exponentiation permet de simplifier les calculs en permettant l'élimination du dénominateur durant l'exécution de l'algorithme de Miller. Sa complexité est de $2kM_p + S_{p^k} + M_{p^k}$. La mise à jour de la fonction f_2 de l'algorithme de Miller décrit par l'Algorithme 2, a un coût de $S_{p^{k/2}} + M_{p^k}$ par bit de r. Élever le résultat du couplage de Miller à la puissance $(p^{k/2}-1)$ est plus rentable que de calculer les dénominateurs. Cette exponentiation étant commune aux couplages de Tate et de Weil, nous ne la considérerons pas dans les comparaisons de complexité. Au final, le calcul du couplage de Weil a une complexité C_W qui vaut : $C_W = C_{Lite} + C_{Full}$.

Le **couplage de Tate** se compose d'une exécution de Miller Lite, suivie d'une élévation à la puissance $(p^k - 1)/r$. Nous avons vu dans la partie II.4.4 que le coût de l'exponentiation lors du calcul du couplage de Tate se réduit à quelques multiplications dans $\mathbb{F}_p$ près, au coût de l'exponentiation à la puissance $\frac{\phi_k(p)}{r}$. Pour calculer cette élévation à la puissance $\frac{\phi_k(p)}{r}$, deux possibilités s'offrent à nous, la méthode des suites de Lucas ou la méthode des fenêtres glissantes. En pratique, la méthode des fenêtres glissantes est plus efficace que les suites de Lucas, mis à part pour le cas où $k = 2$ pour lequel la complexité de l'exponentiation est plus faible en utilisant les suites de Lucas. Au final, le calcul du couplage de Tate a une complexité C_T qui vaut : $C_T = C_{Lite} + C_{exp}$, où C_{exp} représente le coût de l'exponentiation à la puissance $\frac{\phi_k(p)}{r}$.

Les corps adaptés au calcul des couplages sont construits à l'aide de tours d'extension. Chaque étage de la tour est une extension de degré deux ou trois. Pour ces degrés d'extension, les méthodes de Karatsuba et Toom Cook s'appliquent. Pour $k = 2^i 3^j$, nous savons que $M_{p^{k/2}} = 3^{i-1}5^j M_p$ et que $M_{p^k} = 3^i 5^j M_p$. Nous avons les mêmes égalités pour les carrés. Nous présentons par la suite la comparaison du coût de l'évaluation du couplage de Weil par rapport au couplage de Tate.

Coût des couplages

Cas $k = 2$: Pour calculer l'exponentiation, la méthode des suites de Lucas est la plus efficace [GPS06]. Sa complexité est de $C_{Luc} = (M_p + S_p)\log_2(\frac{\phi_2(p)}{r})$, où $\log_2(\frac{\phi_2(p)}{r}) = (\frac{\gamma}{2} - 1)b_r$. La complexité de l'exponentiation est donc : $C_{Luc} = (\frac{\gamma}{2} - 1)(M_p + S_p)b_r$, où γ représente le ratio entre la taille de r, l'ordre des sous-groupes, et la taille de p^k.

L'étape de doublement pour le couplage de Weil nécessite $C_W = 36.b_r M_p$ opérations dans $\mathbb{F}_p$; alors que, pour le couplage de Tate, elle en nécessite $C_T = (16 + \gamma).b_r M_p$. La comparaison de ces deux complexités nous apprend que pour les valeurs de γ telles que $\gamma < 20$ (autrement dit pour un niveau de sécurité de 128 bits) le couplage de Tate est plus efficace que le couplage de Weil. Nous rappelons que γ représente le rapport entre les tailles de r et de p^k dans le Tableau II.5. Mais pour des valeurs de γ plus grandes, le couplage de Weil devient plus efficace à calculer en nombre d'opérations dans $\mathbb{F}_p$.

Cas $k > 4$: Pour $k = 6$ et $p \equiv 2 \pmod 9$ nous utilisons les sous-groupes cyclotomiques des corps finis pour accélérer le calcul d'un carré comme nous l'avons expliqué dans la partie II.4.3.

Pour $k = 12$ et $k = 24$ nous utiliserons la méthode d'élévation au carré de la partie II.4.3. Nous construisons donc les tours d'extension en considérant les polynômes cyclotomiques. Ce faisant, nous pouvons appliquer les améliorations de Lenstra et Stam pour le calcul des carrés dans $\mathbb{F}_{p^k}$, que nous combinons à l'utilisation d'une courbe tordue. Les multiplications sont faites en utilisant les méthodes de Karatsuba et Toom Cook. L'exponentiation du couplage de Tate est calculée en utilisant la méthode des fenêtres glissantes qui est plus efficace pour ces degrés d'extension que les suites de Lucas.

La complexité pour l'exponentiation à l'aide des fenêtres glissantes est donnée de manière globale, en prenant en compte tous les bits de l'exposant. Nous avons donc considéré le nombre total des opérations à effectuer pour calculer le couplage.

Nous noterons $S_{\phi_k(p)}$ la complexité d'un carré dans le corps cyclotomique $G_{\phi_k(p)}$, et $S_{\phi_{k/2}(p)}$ celle d'un carré dans $G_{\phi_{k/2}(p)}$.

Les complexités des couplages de Weil et de Tate sont alors :

$$\begin{aligned} C_W &= ((2k+4)M_p + 4S_p + 6M_{p^{k/2}} + 4S_{p^{k/2}} + \\ &\quad 2M_{p^k} + 2S_{\phi_k(p)})\log_2(r) + M_{p^k} \\ C_T &= \left((k+4)M_p + 4S_p + M_{p^k} + S_{\phi_k(p)}\right)\log_2(r) + C_{sw} \end{aligned}$$

où, C_{sw} représente la complexité de la méthode des fenêtres glissantes,

$$C_{sw} = \left(\frac{k}{\gamma}\left(\frac{\gamma}{3} - 1\right) + \frac{\gamma}{k}b_r\right)S_{\phi_k(p)} + \left(\frac{\gamma/3 - 1}{r+2}b_r + \frac{k}{\gamma}(2^{n-1} - 1) - 1\right)M_{p^k}.$$

k	$M_{p^{k/2}}$	M_{p^k}	$S_{\phi_k(p)}$
6	$5M_p$	$15M_p$	$9M_p$
12	$15M_p$	$45M_p$	$30M_p$
24	$45M_p$	$135M_p$	$108M_p$

k	Niveau de sécurité	Nb op. pour Weil	Nb op. pour Tate
6	80	18 880	8 180
-	128	30 208	16 428
-	192	45 312	31 592
-	256	60 416	52 456
12	80	49 920	19 600
-	128	79 872	39 320
-	192	119 808	73 480
-	256	159 744	118 640
24	80	158 720	56 000
-	128	253 952	109 400
-	192	380 928	202 600
-	256	507 904	320 800

TABLE II.7 – Comparaison des complexités des couplages en nombre d'opérations dans $\mathbb{F}_p$

Dans le Tableau II.7, nous présentons une comparaison entre les complexités des couplages de Weil et de Tate à différents niveaux de sécurité. Nous avons utilisé les diverses méthodes de calcul présentées dans les parties précédentes afin de réduire le coût global en opération de l'exécution des algorithmes. Nous avons privilégié le système de représentation minimisant le nombre d'opérations pour le calcul des couplages. Nous avons comparé les complexités des calculs en coordonnées affines, projectives et Jacobiennes. Pour $k = 6$ et 12 les coordonnées Jacobiennes minimisent le nombre d'opération, pour $k = 24$ les coordonnées affines le font. Pour simplifier les comparaisons, nous avons exprimé les complexités des couplages en nombre de multiplications dans $\mathbb{F}_p$. Les équivalences des multiplications dans les diverses extensions utilisées sont données dans le Tableau II.7. Excepté pour les carrés dans le sous-groupe cyclotomique, nous avons pris la convention selon laquelle le coût d'un carré S se calcule comme $S = 0,75M$ quel que soit le degré d'extension du corps. Lorsque des inversions non négligeables doivent être calculées, nous avons utilisé les complexités données par Granger, Page et Smart dans [GPS06]. A savoir que $I_{p^{24}} = 133S_p + 141M_p + I_p \approx 314M_p$, où $I_{p^{24}}$ (respectivement I_p) représente une inversion dans $\mathbb{F}_{p^{24}}$ (respectivement $\mathbb{F}_p$).

L'analyse des résultats montre que pour $k \geq 6$ et les niveaux de sécurité actuels et futurs, le couplage de Tate est toujours plus efficace que le cou-

plage de Weil. Il est, malgré tout, intéressant de noter que pour $k = 6$ l'écart se réduit au fur et à mesure que le niveau de sécurité augmente. Par exemple à un niveau de sécurité de 80 bits, le couplage de Weil nécessite 130% d'opérations en plus que le couplage de Tate, alors que pour un niveau de sécurité de 256 bits le surcoût du calcul du couplage de Weil n'est plus que de 15%.

Pour les deux autres valeurs de k (12 et 24) le couplage de Tate est toujours plus efficace. Il faut toutefois souligner que si le calcul de Miller Lite a été bien optimisé, ce n'est pas le cas du calcul de Miller Full. Il n'est pas improbable que d'autres améliorations de cette opération puissent être trouvées. Ainsi, la comparaison entre les couplages de Weil et de Tate n'est pas vraiment terminée. Néanmoins, ces études ont été menées avant que les couplages Ate et Twisted Ate ne soient découverts. Comme nous l'avons vu dans la partie I.9 ces couplages se calculent de la même manière que le couplage de Tate (une exécution de Miller suivie d'une exponentiation) ; mais avec moins d'itérations. Ils sont donc naturellement moins coûteux à calculer que le couplage de Tate, et par conséquent, bien plus efficace que le couplage de Weil pour les niveaux de sécurité envisagés aujourd'hui.

II.5 Quelques perspectives du point de vue arithmétique

Nous revenons sur les diverses optimisations utilisées et nous analysons les pistes de recherche présentées dans [GPS06].

Pour faire la multiplication, Granger, Page et Smart suggèrent dans l'article [GPS06] d'utiliser une version améliorée de l'algorithme de Montgomery en $2\tau^2+\tau$ opérations sur des mots machines de taille w, avec $\tau = \log_2 p/w$. Cet algorithme calcule simultanément le produit et la réduction dans $\mathbb{F}_p$. Étant donné les tailles des mots utilisés, il semblerait plus judicieux d'évaluer dans un premier temps le produit, puis dans un second temps d'effectuer la réduction séparément en opérations machines. Cette approche laisse le choix de l'algorithme de multiplication, qui peut être l'algorithme de Karatsuba ou Toom Cook, et de réduire ainsi le coût à $\tau^2+\tau+\tau^{\log_2(3)}$ où $\tau^2+\tau+\tau^{\log_3(5)}$ opérations.

De plus, si nous voulons donner les complexités réelles des calculs de couplage, il nous faut descendre au niveau d'un mot machine et prendre en compte les additions.

Une simplification possible des calculs pourrait être de ne pas faire la réduction modulaire après chaque produit. En effet, les réductions modulo p pourraient être faites après plusieurs opérations, en particulier les additions de produits, dont le résultat est un nombre dont la taille ne varie pas

beaucoup. Par exemple, si nous devons calculer la somme du résultat de deux produits, la démarche classique est d'effectuer les deux produits, de les réduire modulo p, puis d'additionner les résultats que nous réduirons une fois de plus modulo p. Ce qui représente trois réductions modulaires. Nous pourrions nous contenter de n'effectuer qu'une réduction après l'addition des résultats non réduits des produits. Ainsi, nous passerions de trois à une réduction modulaire, ce qui permettrait de réduire les coûts de calcul et donc les complexités des évaluations. L'article [BDEM06] présentent des travaux allant dans ce sens.

De nombreux articles ont pour sujet le calcul des couplages dans des corps finis en caractéristiques 2, 3 ou $p > 3$ [AHM07, GPS05, BBS+07, BDE+09]. Il serait intéressant de comparer l'efficacité de ces couplages en différentes caractéristiques, pour des niveaux de sécurité équivalents.

Nous avons vu que la méthode de Lenstra et Stam pour calculer un carré dans les extensions de $\mathbb{F}_p$ de degré un multiple de 6 est plus performante que les méthodes classiques. Leur méthode n'est pour l'instant pas encore généralisée à des degrés d'extension plus généraux.

De nouveau dans la référence [GPS06], les auteurs arguent que les meilleures méthodes pour effectuer un carré et une multiplication sont des combinaisons de multiplication de Karatsuba et Toom Cook. Ceci semble vrai pour des valeurs de k de la forme $k = 2^i 3^j$; en effet une multiplication dans $\mathbb{F}_{p^k}$ avec $k = 2^i 3^j$ se fait en $3^i 5^j$ multiplication dans $\mathbb{F}_p$. L'inconvénient de cette écriture est que les valeurs de k ne sont pas très nombreuses. Il serait donc intéressant d'analyser comment l'arithmétique pour des extensions F_{p^k} avec un k quelconque peut être améliorée. Que se passe-t-il pour les valeurs de k moins adaptées aux multiplications de Karatsuba et Toom Cook ?

Des éléments de réponses sont apportés dans les deux chapitres suivants.

Le Chapitre III présente une adaptation à l'arithmétique des couplages de l'arithmétique des bases adaptées proposée par Thomas Plantard dans sa thèse [Pla05]. La représentation en base adaptée permet de construire la base de $\mathbb{F}_p$ de telle sorte que le coût d'une multiplication par une racine de l'unité soit négligeable. Cette propriété nous permettra d'améliorer l'arithmétique de $\mathbb{F}_{p^k}$ en effectuant une multiplication à l'aide de la méthode de Fourier, pour des valeurs de k quelconques.

Le Chapitre IV propose une arithmétique originale et efficace pour les extensions de degré 5. Nous l'avons utilisé pour calculer les couplages Ate et Twisted Ate sur des courbes de degré de plongement 15. Nous avons comparé nos résultats avec la complexité du couplage sur les courbes Barreto-Naehrig qui sont considérées comme les courbes les plus adaptées à un calcul de couplage efficace.

Chapitre III

Une arithmétique plus efficace fondée sur les bases adaptées

Sommaire

III.1 Présentation

Le calcul des couplages implique d'effectuer des opérations dans des extensions de corps finis, de la forme $\mathbb{F}_{p^k}$ pour p un nombre premier.

Les corps finis $\mathbb{F}_{p^k}$ utilisés en cryptographie à base de couplages sont assez spécifiques. Les tailles de p et k sont fixées par les critères de sécurité introduits dans la partie I.7.4. Les méthodes privilégiées de multiplications dans les corps $\mathbb{F}_{p^k}$ sont une combinaison des méthodes de Karatsuba et Toom Cook, que nous avons décrites dans la partie II.3. Par conséquent, de nombreux articles se sont concentrés sur des degrés de plongement k de la forme $k = 2^i 3^j$. Les corps finis vérifiant cette condition sont appelés les *corps amis*, nous les avons présentés dans la partie II.4.2. Dans ce cas, le coût d'une multiplication entre deux éléments de $\mathbb{F}_{p^k}$ est de $3^i 5^j$ multiplications dans $\mathbb{F}_p$.

Récemment, une méthode de multiplication basée sur la transformée de Fourier discrète a été proposée dans [GPS07] pour réaliser une multiplication dans le corps $\mathbb{F}_{q^6}$ avec $q = 3^n$. En pratique, la transformée de Fourier discrète (notée par la suite DFT pour l'acronyme anglais Discrete Fourier Transform) est intéressante pour des degrés d'extension k assez grands. Nous travaillons dans des extensions de petits degrés, mais les corps finis sous-jacents sont de taille importante, donc même si l'utilisation de la DFT ne permet d'économiser que quelques multiplications dans $\mathbb{F}_p$ alors elle est avantageuse. Nous rappelons dans la partie III.2 le principe d'une multiplication utilisant la DFT, dont nous rappellons ci-dessous les caractéristiques.

Une multiplication par la méthode DFT nécessite dans le meilleur des cas $2k-1$ multiplications dans $\mathbb{F}_p$ et $O(k^2)$ multiplications par des racines de l'unité. Si de plus la méthode FFT (pour Fast Fourier Transformation ou encore transformée de Fourier rapide) est appliquée, la complexité de la multiplication par l'approche DFT est de $O(k\log(k))$ multiplications par des racines de l'unité dans $\mathbb{F}_p$. Lorsque le corps fini $\mathbb{F}_p$ est représenté de manière classique, l'approche DFT est bien trop coûteuse. En effet, dans un corps $\mathbb{F}_p$ quelconque les racines de l'unité ont une représentation binaire dense, et multiplier par ces valeurs reste aussi coûteux qu'une multiplication quelconque dans $\mathbb{F}_p$. Nous proposons d'utiliser une représentation particulière du corps $\mathbb{F}_p$, la représentation AMNS introduite par Thomas Plantard dans sa thèse [Pla05]. La représentation AMNS permet de construire et représenter des racines de l'unité ayant une décomposition creuse. Ainsi la multiplication par ces racines de l'unité consiste en des décalages et changements de signes. Le coût de ces multiplications peut alors être négligé, et la multiplication dans $\mathbb{F}_{p^k}$ par l'approche DFT ne nécessite plus que $(2k-1)$ multiplications dans $\mathbb{F}_p$.

Nous proposons une multiplication dans $\mathbb{F}_{p^k}$ plus efficace que les multiplications de Karatsuba et Toom Cook. Cette multiplication repose sur la

combinaison de la méthode DFT pour exécuter la multiplication d'éléments d'une extension de corps $\mathbb{F}_{p^k}$ et l'utilisation d'une représentation AMNS. Ces travaux ont été menés en collaboration avec Christophe Nègre, maître de conférence à l'Université de Perpignan, ils ont été publiés dans les proceedings LNCS de la conférence ACISP 2009 [EN09].

L'organisation de ce chapitre est la suivante, la partie III.2 rappelle la multiplication par DFT dans les extensions de corps $\mathbb{F}_{p^k}$. Nous l'adaptons aux cas nous intéressant dans la partie III.2.4. La partie III.3 présente le principe de la représentation AMNS, en donnant la définition et l'arithmétique d'une telle base pour représenter les éléments d'un corps fini $\mathbb{F}_p$. Puis nous nous focalisons sur l'application de la multiplication DFT dans la partie III.4 ; où, en combinant la représentation AMNS et la multiplication DFT nous améliorons la multiplication dans l'extension de corps $\mathbb{F}_{p^k}$. Nous évaluons la complexité de notre approche pour différentes extensions de corps et nous comparons nos coûts à ceux des corps amis. Nous concluons dans la partie III.5.

III.2 Arithmétique dans l'extension de corps

III.2.1 Multiplication de polynômes utilisant la DFT

Nous rappelons ici le principe de la transformée de Fourier discrète (DFT) pour la multiplication de polynômes. Cette approche est un cas spécifique des méthodes de multiplications par interpolation de polynômes [ZG03]. Nous avons décrit dans la partie II.3.3 la multiplication par interpolation lorsque nous pouvions choisir librement le nombre de valeurs d'interpolation nécessaires pour effectuer le produit de deux polynômes de degré $(k-1)$, à savoir $(2k-1)$ valeurs. Nous décrivons ici une approche faisant appel aux racines n-ième de l'unité, avec $n \geq 2k-1$.

Les multiplications par interpolation calculent la multiplication de deux polynômes U et V de degré $(k-1)$ en utilisant l'évaluation de ces polynômes en $n \geq 2k-1$ éléments distincts de $\mathbb{F}_p$. Ces n évaluations permettent de déterminer n valeurs prises par le polynôme $W = U \times V$ en calculant terme à terme le produit des évaluations de U et V. Le polynôme W ainsi déterminé par ses valeurs en n points distincts peut être reconstruit, en retrouvant ces coefficients par une méthode d'interpolation, à l'aide des n valeurs prises par ce polynôme. Dans le cadre de la DFT, où les points sont des racines n-ième de l'unité, l'évaluation des polynômes en ces valeurs ainsi que l'interpolation finale utilisent alors les propriétés des racines de l'unité pour mettre en place des algorithmes de type « divide and conquer », ou en français diviser pour régner.

Soit $\omega \in \mathbb{F}_p$ une racine primitive n-ième de l'unité, la multiplication po-

lynomiale sur le principe de la DFT s'opère de la façon décrite ci dessous.

1. *Évaluation.* Soient U, V deux polynômes dans $\mathbb{F}_p[X]$ de degré $k-1$. Nous évaluons le polynôme U en les puissances de ω. Nous notons $\widehat{U}$ le vecteur dans lequel nous stockons les évaluations du polynôme U.

$$\widehat{U} = DFT_\omega(U) = (U(1), U(\omega), \ldots, U(\omega^{n-1})).$$

   Cette opération est usuellement appelée la transformée de Fourier discrète de U décrite dans la partie II.3.6. La même opération est faite pour V. Ces calculs peuvent être vus comme un produit matrice × vecteur, dont la complexité est en $O(n^2)$ multiplications par des puissances de ω. Pour des entiers n étant des puissances de 2, nous pouvons de surcroit utiliser la transformée de Fourier et l'algorithme FFT au lieu d'un produit matrice×vecteur. La complexité devient alors $O(n\log(n))$ multiplications par des puissances de ω, nous le décrivons dans la partie III.2.3.

$$\widehat{U} = \begin{bmatrix} 1 & \omega & \omega^2 & \cdots & \omega^{k-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{(k-1)2} \\ \vdots & & & & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \cdots & \omega^{(k-1)(n-1)} \end{bmatrix} \cdot \begin{bmatrix} u_0 \\ u_1 \\ \vdots \\ u_{k-1} \end{bmatrix}.$$

2. *Multiplication terme à terme* Ces produits sont effectués sur les coefficients de $\widehat{U}$ et $\widehat{V}$ pour donner

$$\widehat{W} = (\widehat{u}_1 \times \widehat{v}_1, \widehat{u}_2 \times \widehat{v}_2 \ldots, \widehat{u}_n \times \widehat{v}_n),$$

   l'évaluation de W pour $W = U \times V$, où W est un polynôme de degré strictement inférieur à $(2k-1) \leq n$.

3. *Interpolation* La phase d'interpolation consiste à reconstruire le polynôme W en ayant connaissance de ces valeurs en $\vec{\omega} = (1, \omega, \omega^2, \ldots, \omega^{(n-1)})$.

Le lemme suivant décrit la méthode d'interpolation que nous utilisons.

Lemme III.2.1. *soit $\mathbb{F}_p$ un corps fini, ω racine primitive n-ième de l'unité et Ω la matrice :*

$$\Omega = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(n-1)} \\ 1 & \omega^3 & \omega^6 & \cdots & \omega^{3(n-1)} \\ \vdots & & & & \vdots \\ 1 & \omega^{n-2} & \omega^{(n-2)2} & \cdots & \omega^{(n-2)(n-1)} \\ 1 & \omega^{n-1} & \omega^{(n-1)2} & \cdots & \omega^{(n-1)(n-1)} \end{bmatrix} \quad \text{(III.2.1)}$$

Nous notons $\omega' = \omega^{-1} = \omega^{n-1}$. *La matrice inverse de* Ω *est donnée par*

$$\Omega^{-1} = \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega' & \omega'^2 & \cdots & \omega'^{n-1} \\ 1 & \omega'^2 & \omega'^4 & \cdots & \omega'^{(n-1)2} \\ \vdots & & & & \vdots \\ 1 & \omega'^{n-2} & \omega'^{2(n-2)} & \cdots & \omega'^{(n-2)(n-1)} \\ 1 & \omega'^{n-1} & \omega'^{2(n-1)} & \cdots & \omega'^{(n-1)(n-1)} \end{bmatrix} \tag{III.2.2}$$

***Remarque* III.2.2.** Le fait que ω soit une racine n-ième de l'unité nous permet de simplifier les matrices Ω et Ω^{-1}.

$$\Omega = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{(n-2)} \\ 1 & \omega^3 & \omega^6 & \cdots & \omega^{(n-3)} \\ \vdots & & & & \vdots \\ 1 & \omega^{(n-2)} & \omega^{(n-4)} & \cdots & \omega^2 \\ 1 & \omega^{(n-1)} & \omega^{(n-2)} & \cdots & \omega \end{bmatrix} \tag{III.2.3}$$

$$\Omega^{-1} = \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{(n-1)} & \omega^{(n-2)} & \cdots & \omega \\ 1 & \omega^{(n-2)} & \omega^{(n-4)} & \cdots & \omega^2 \\ \vdots & & & & \vdots \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{(n-2)} \\ 1 & \omega & \omega^2 & \cdots & \omega^{(n-1)} \end{bmatrix} \tag{III.2.4}$$

L'interpolation est alors reconstruite en calculant le produit entre la matrice Ω^{-1} et le vecteur $\widehat{W}$ et en ne conservant que les $2k-1$ premiers coefficients. Nous obtenons

$$W = \frac{1}{n}\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega^{(n-1)} & \omega^{(n-2)} & \cdots & \omega \\ 1 & \omega^{(n-2)} & \omega^{(n-4)} & \cdots & \omega^2 \\ \vdots & & & & \vdots \\ 1 & \omega^{(2k-2)(n-1)} & \omega^{(2k-2)(n-2)} & \cdots & \omega^{(2k-2)} \end{bmatrix} \cdot \begin{bmatrix} \widehat{w}_1 \\ \widehat{w}_2 \\ \vdots \\ \widehat{w}_n \end{bmatrix}.$$

III.2.2 Les multiplications par $\frac{1}{n}$

Représentation de Montgomery

Lors de l'étape d'interpolation, nous devons faire un produit matrice×vecteur ainsi que des multiplications par $\frac{1}{n}$. Nous pouvons éviter le surcoût engendré par ces multiplications lors de l'étape d'interpolation. Il suffit pour cela d'utiliser une représentation de Montgomery [Mon85] de U et V. Notons

$$\widetilde{U} = \frac{1}{n}U \text{ et } \widetilde{V} = \frac{1}{n}V.$$

L'étape d'interpolation nécessite n multiplications par $\frac{1}{n}$. Si nous utilisons une méthode de multiplication par DFT en représentation de Montgomery, le produit de $\widetilde{U}$ et $\widetilde{V}$ devient :

$$n\widetilde{U}\widetilde{V} = n(\frac{1}{n}U) \times (\frac{1}{n}V) = \frac{1}{n}W = \widetilde{W},$$

avec $W = U \times V$. Ceci est justifié par le lemme suivant.

Lemme III.2.3. *En représentation de Montgomery,* $\widetilde{\Omega^{-1}} = n\Omega^{-1}$.

En utilisant ce lemme, et en remarquant que $\widehat{\widetilde{W}} = \widehat{\widetilde{U}} \times \widehat{\widetilde{V}}$, nous obtenons que $\widetilde{W} = \widetilde{\Omega^{-1}} \times \widehat{\widetilde{W}}$ soit que $\widetilde{W} = \frac{1}{n}W$.

En d'autres termes, la multiplication par DFT sans division par n est stable en représentation de Montgomery. Nous transformons les multiplications par $1/n$ en des multiplications par n. La représentation de Montgomery est aussi stable pour les additions et réductions modulo le polynôme P. Elle peut donc être utilisée pour une succession d'additions, multiplications ou autres opérations arithmétiques dont nous avons besoin en cryptographie de manière générale, et pour le calcul des couplages en particulier.

Simplification pendant l'opération de Miller Lite

La représentation de Montgomery implique une multiplication par n supplémentaire. Lorsque nous effectuons l'opération de Miller Lite, les multiplications dans $\mathbb{F}_{p^k}$ correspondent à la mise à jour de f_1 pendant les étapes 5 et 8 de l'Algorithme 3, décrit dans la partie II.2. Par exemple, lors de l'étape 5, nous calculons $f_1^2 \times l_1(Q)$, avec f_1 et $l_1(Q)$ qui sont des éléments de $\mathbb{F}_{p^k}$.

Pour calculer le produit $f_1^2 \times l_1(Q)$ en appliquant la méthode DFT en représentation de Montgomery, lors de l'interpolation nous devons multiplier $\widetilde{f_1^2} \times \widetilde{l_1(Q)}$ par n. Or, n appartient à $\mathbb{F}_p$; ainsi, lors de l'exponentiation finale le produit par n sera neutralisé. Nous pouvons donc ne pas le calculer durant les itérations de l'algorithme de Miller. Le même raisonnement s'applique pour l'élévation au carré de f_1 et l'étape 8 de l'Algorithme 3.

Les multiplications par n sont éliminées pendant l'exécution de Miller Lite par l'exponentiation finale. L'implémentation des couplages en utilisant la méthode DFT serait simplifiée, en ne calculant pas ces multiplications par n dans le corps de l'algorithme de Miller.

Remarque **III.2.4.** Cette simplification n'est pas réalisable pour l'exponentiation finale, les multiplications par n doivent être effectuées.

III.2.3 Transformée de Fourier rapide (FFT)

Lorsque l'entier n est une puissance de deux, nous pouvons améliorer l'efficacité des calculs en utilisant un principe de divide and conquer.

Nous rappelons le principe de la FFT introduite dans la partie II.3.6. Soit $U = \sum_{i=0}^{k-1} u_i X^i \in \mathbb{F}_p[X]$ et ω une racine primitive n-ième de l'unité. La FFT est un algorithme qui calcule efficacement l'évaluation du polynôme U en $\vec{\omega} = (1, \omega, \omega^2, \ldots, \omega^{n-1})$.

Le principe de la FFT repose sur une division en deux parties du polynôme U, suivant la parité des coefficients :

$$\begin{array}{rcl} U_1 & = & \sum_{j=0}^{k/2-1} a_{2j} X^{2j}, \\ U_2 & = & \sum_{j=0}^{k/2-1} a_{2j+1} X^{2j+1}, \end{array} \quad \text{tel que } U = U_1 + XU_2.$$

Soit $\widehat{U}[i] = U(\omega^i)$ le i-ième coefficient de $\widehat{U} = DFT_\omega(U)$. Nous notons $\widehat{U}_1[i], \widehat{U}_2[i]$ les coefficients de $DFT_{\omega^2}(U_1)$ et $DFT_{\omega^2}(U_2)$ en $\{1, \omega^2, \omega^4, \ldots, (\omega^2)^{n/2-1}\}$. Nous pouvons exprimer l'évaluation de $U = U_1 + XU_2$ en ω^i et $\omega^{i+n/2}$, pour $i < \frac{n}{2}$ par les formules

$$\begin{array}{rcl} \widehat{U}[i] & = & \widehat{U}_1[i] + \omega^i \widehat{U}_2[i], \\ \widehat{U}[i+n/2] & = & \widehat{U}_1[i] - \omega^i \widehat{U}_2[i]. \end{array}$$

car, $\omega^{i+n/2} = -\omega^i$ et donc $\omega^{n/2} \equiv -1 \bmod(p)$.
Le calcul de $DFT_\omega(U)$ se réduit alors au calcul de $DFT_{\omega^2}(U_1)$ et $DFT_{\omega^2}(U_2)$. Comme n est une puissance de deux, ces calculs sont faits récursivement. Cette méthode peut aussi s'appliquer lorsque n n'est pas une puissance de deux. L'algorithme entier de multiplication par FFT à une complexité en nombre d'opérations effectuées de $\frac{n}{2}\log_2(n)$ multiplications par ω^i dans $\mathbb{F}_p$ et $n\log_2(n)$ additions/soustractions dans $\mathbb{F}_p$.

***Remarque* III.2.5.** Le nombre de multiplications par ω^i est la moitié du nombre d'additions car les calculs de $\omega^i \widehat{U}_2[i]$ apparaissent dans les deux expressions et ne sont calculés qu'une seule fois.

III.2.4 Multiplication par DFT pour $n \leq 2k-2$

Nous décrivons dans cette partie la multiplication par DFT lorsqu'il n'est pas possible d'obtenir de racines primitives n-ième de l'unité pour $n \geq 2k-2$. A priori, pour $n < 2k-2$, la multiplication par DFT ne peut être effectuée car il manquerait des valeurs d'interpolation. Nous décrivons une extension de la multiplication DFT lorsque nous devons nous contenter d'une racine

n-ième primitive de l'unité avec n inférieur à $2k-1$. Nous nous concentrons uniquement sur les deux cas $n = 2k-2$ et $n = 2k-4$, qui correspondent aux cas pratiques décrits dans la partie III.4. L'approche qui suit est une généralisation de la méthode présentée dans [GPS07] pour $k = 3$ et $n = 6$ à d'autres valeurs de k.

La méthode consiste à déterminer a priori certains coefficients du polynôme produit correspondant aux termes de degré le plus grand et au terme constant. Autrement dit, nous calculons a priori les interpolations du polynôme produit en des valeurs fixées à l'avance. Ainsi, nous avons moins de coefficients à déterminer par interpolation et nous pouvons utiliser des matrices d'interpolation de taille plus petite. Les deux lemmes III.2.6 et III.2.7 traitent les cas pour lesquels nous calculons un, puis trois coefficients a priori.

Lemme III.2.6. *Soit $n = 2k-2$, $\mathbb{F}_p$ un corps premier et $\omega \in \mathbb{F}_p$ une racine primitive n-ième de l'unité. Soient $U = \sum_{i=0}^{k-1} u_i X^i$ et $V = \sum_{i=0}^{k-1} v_i X^i$ dans $\mathbb{F}_p[X]$, $W = U \times V$. Les coefficients de W peuvent être calculés par :*

1. *$w_{2k-2} = u_{k-1} \times v_{k-1}$, ce qui correspond à l'évaluation en l'infini,*
2. *Les coefficients w_i pour $i = 0, \ldots, 2k-3$ sont calculés par l'interpolation d'un polynôme de degré $(2k-3)$:*

$$\begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_{2k-3} \end{bmatrix} = \Omega^{-1} \cdot \begin{bmatrix} \widehat{u}_i \times \widehat{v}_i - w_{2k-2} \\ \widehat{u}_i \times \widehat{v}_i - w_{2k-2} \\ \vdots \\ \widehat{u}_i \times \widehat{v}_i - w_{2k-2} \end{bmatrix}. \tag{III.2.5}$$

où Ω^{-1} est définie par l'équation (III.2.2), et les coefficients $\widehat{u}_i$ sont les coefficients de $\widehat{U} = DFT_\omega(U)$ et les $\widehat{v}_i$ ceux de $\widehat{V} = DFT_\omega(V)$.

Démonstration. Nous voulons interpoler le polynôme $W = U \times V$ de degré $(2k-2)$, en utilisant $n = 2k-2$ valeurs d'interpolation. La démonstration de ce lemme repose sur le fait qu'en évaluant nous même le polynôme W en l'infini, nous pouvons construire un polynôme W' de degré $(2k-3)$, dont nous pouvons retrouver l'expression par une l'interpolation à l'aide des $(2k-2)$ valeurs.

Normalement l'évaluation $\widehat{W}$ de W en les n éléments ω^i s'effectue de la manière qui suit :

$$\begin{cases} \widehat{w}_0 = W(1) &= w_0 + w_1 + \ldots + w_{n-1} + w_n \\ \widehat{w}_1 = W(\omega) &= w_0 + w_1\omega + \ldots + w_{n-1}\omega^{n-1} + w_n\omega^n \\ \widehat{w}_2 = W(\omega^2) &= w_0 + w_1(\omega^2) + \ldots + w_{n-1}(\omega^2)^{n-1} + w_n(\omega^2)^n \\ \vdots & \vdots \\ \widehat{w}_{n-1} = W(\omega^{n-1}) &= w_0 + w_1(\omega^{n-1}) + \ldots + w_{n-1}(\omega^{n-1})^{n-1} + w_n(\omega^{n-1})^n \end{cases}$$

Or, comme $\omega^n = 1$, nous savons que $(\omega^i)^n = 1$ pour $i = 1, \ldots, n-1$. La partie à droite de l'équation précédente peut être ré-écrite :

$$\left\{\begin{array}{rcl} \widehat{w}_0 = W(1) & = & w_0 + w_1 + \cdots + w_n \\ \widehat{w}_1 = W(\omega) & = & w_0 + w_1\omega + \cdots + w_n\omega^{n-1} + w_n \\ \widehat{w}_2 = W(\omega^2) & = & w_0 + w_1(\omega^2) + \cdots + w_{n-1}(\omega^2)^{n-1} + w_n \\ \vdots & & \vdots \\ \widehat{w}_{n-1} = W(\omega^{n-1}) & = & w_0 + \cdots + w_{n-1}(\omega^{n-1})^{n-1} + w_n \end{array}\right. \quad \text{(III.2.6)}$$

Le coefficient w_n correspond à l'évaluation à l'infini de W : $w_n = w_{2k-2} = u_{k-1}v_{k-1}$. L'équation (III.2.6) nous permet de remarquer que le vecteur $(\widehat{w}_0 - w_n, \widehat{w}_1 - w_n, \ldots, \widehat{w}_{n-1} - w_n)$ est la DFT du polynôme $W' = \sum_{i=0}^{n-1} w_i X^i = W - w_{2k-2}X^{2k-2}$. Nous obtenons les coefficients de W' en calculant

$$\Omega^{-1} \cdot \left[\begin{array}{cccc} (\widehat{w}_0 - w_n) & (\widehat{w}_1 - w_n) & \cdots & (\widehat{w}_{n-1} - w_n) \end{array} \right]^t.$$

Ce qui correspond à l'équation (III.2.5).

Le polynôme $W(X)$ est reconstruit grâce à l'égalité

$$W(X) = W'(X) + w_{(2k-2)}X^{(2k-2)}.$$

□

Nous allons à présent nous concentrer sur le cas $n = 2k - 4$. Nous travaillons toujours avec des polynômes de degré $(k-1)$, il nous faudrait donc $(2k-1)$ valeurs différentes d'interpolation. Comme $n = 2k-4$, il nous manque trois valeurs d'interpolation. Nous compensons les valeurs d'interpolation manquantes en déterminant a priori trois coefficients du polynôme W.

Lemme III.2.7. *Soit $n = 2k-4$, $\mathbb{F}_p$ un corps premier et $\omega \in \mathbb{F}_p$ une racine primitive n-ième de l'unité. Soient $U = \sum_{i=0}^{k-1} u_i X^i$ et $V = \sum_{i=0}^{k-1} v_i X^i$ dans $\mathbb{F}_p[X]$, $W = U \times V$. Les coefficients de W peuvent être calculés par :*

1. *$w_0 = u_0 \times v_0$, ce qui correspond à l'évaluation en zéro,*
2. *$w_{2k-2} = u_{k-1} \times v_{k-1}$, ce qui correspond à l'évaluation en l'infini,*
3. *$w_{2k-3} = u_{k-1} \times v_{k-2} + u_{k-2} \times v_{k-1}$, ce qui correspond au calcul du coefficient de X^{2k-3} en utilisant les formules de multiplication polynomiale.*

4. *Les coefficients w_i pour $i = 1, \ldots, 2k-4$ sont calculés par l'interpolation d'un polynôme de degré $(2k-5)$:*

$$\begin{bmatrix} w_1 \\ w_2 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \Omega^{-1} \cdot \begin{bmatrix} \widehat{u}_0 \times \widehat{v}_0 - w_0 - w_{n+1} - w_{n+2} \\ (\widehat{u}_1 \times \widehat{v}_1 - w_0 - w_{n+1}\omega - w_{n+2}\omega^2)\omega^{-1} \\ (\widehat{u}_2 \times \widehat{v}_2 - w_0 - w_{n+1}\omega^2 - w_{n+2}(\omega^2)^2)\omega^{-2} \\ \vdots \\ (\widehat{u}_{n-1} \times \widehat{v}_{n-1} - w_0 - w_{n+1}\omega^{n-1} - w_{n+1}(\omega^{n-1})^2)\omega^{-(n-1)} \end{bmatrix} \tag{III.2.7}$$

où Ω^{-1} est définie par l'équation (III.2.2)*, et les coefficients $\widehat{u}_i$ sont les coefficients de $\widehat{U} = DFT_\omega(U)$ et les $\widehat{v}_i$ ceux de $\widehat{V} = DFT_\omega(V)$.*

Démonstration. La preuve est similaire à celle du lemme III.2.6. Comme $n = 2k-4$ et comme U et V sont de degré $(k-1)$ alors $U \times V = W = \sum_{i=0}^{n+2} w_i X^i$ est de degré $n+2 = 2k-2$. L'évaluation $\widehat{W}$ de W en les n éléments ω^i donne

$$\begin{cases} \widehat{w}_0 &= W(1) = w_0 + w_1 + \cdots + w_n + w_{n+1} + w_{n+1} \\ \widehat{w}_1 &= W(\omega) = w_0 + w_1\omega + \cdots + w_n\omega^n + w_{n+1}\omega^{n+1} + w_{n+2}\omega^{n+2} \\ \widehat{w}_2 &= W(\omega^2) = w_0 + w_1(\omega^2) + \cdots + w_n\omega^n + w_n(\omega^2)^n + w_{n+2}(\omega^2)^{n+2} \\ \vdots & \qquad\qquad \vdots \\ \widehat{w}_{n-1} &= W(\omega^{n-1}) = w_0 + w_1(\omega^{n-1}) + \cdots + w_n(\omega^{n-1})^n \\ & \qquad\qquad + w_{n+1}(\omega^{n-1})^{n+1} + w_{n+2}(\omega^{n-1})^{n+2} \end{cases}$$

Puisque $\omega^n = 1$, nous avons que $(\omega^i)^{n+1} = \omega^i$ et $(\omega^i)^{n+2} = (\omega^i)^2$ pour $i = 1, \ldots, n-1$ dans la partie droite de l'équation précédente. Nous obtenons pour $\widehat{w}_i$

$$\begin{cases} \widehat{w}_0 &= w_0 + w_1 + \cdots + w_n + w_{n+1} + w_{n+2} \\ \widehat{w}_1 &= w_0 + w_1\omega + \cdots + w_n\omega^n + w_{n+1}\omega + w_{n+1}\omega^2 \\ \widehat{w}_2 &= w_0 + w_1(\omega^2) + \cdots + w_n(\omega^2)^n + w_{n+1}\omega^2 + w_{n+2}(\omega^2)^2 \\ \vdots & \qquad\qquad \vdots \\ \widehat{w}_{n-1} &= w_0 + \cdots + w_n(\omega^{n-1})^n + w_{n+1}\omega^{n-1} + w_{n+2}(\omega^{n-1})^2 \end{cases}$$

Les trois coefficients w_0, w_{n+2}, w_{n+1} sont calculés a priori en utilisant les formules de multiplication polynomiale. Nous pouvons ré-écrire l'équation de la manière suivante.

$$\begin{cases} \widehat{w}_0 - w_0 - w_{n+1} - w_{n+2} &= w_1 + \cdots + w_{n-1} \\ (\widehat{w}_1 - w_0 - w_{n+1}\omega - w_{n+2}\omega^2)\omega^{-1} &= w_1 + \cdots + w_n\omega^{n-1} \\ (\widehat{w}_2 - w_0 - w_{n+1}\omega^2 - w_{n+2}(\omega^2)^2)\omega^{-2} &= w_1 + \cdots + w_{n-1}(\omega^2)^{n-1} \\ \vdots & \qquad \vdots \\ (\widehat{w}_{n-1} - w_0 - w_{n+1}\omega^{n-1} - w_{n+1}(\omega^{n-1})^2)\omega^{-(n-1)} &= w_1 + w_2 + \cdots \\ & \qquad \cdots + w_{n-1}(\omega^{n-1})^{n-1} \end{cases}$$

Cette ré-écriture permet de transformer ces équations en l'interpolation en $n = 2k-4$ valeurs d'un polynôme de degré $2k-5$.

La partie droite de cette équation correspond à l'évaluation du polynôme

$$W'(X) = \frac{W(X) - w_0 - w_{2k-2}X^{2k-2} - w_{2k-3}X^{2k-3}}{X}.$$

Soit $W' = \sum_{i=0}^{n-1} w_{i+1}X^i$.

Sachant que $\omega^{n+1} = \omega$, et que $\omega^{n+2} = \omega^2$, nous obtenons les évaluations de W' en calculant :

$$W'(\omega^i) = \frac{U(\omega^i) \times V(\omega^i) - w_0 - w_{2k-2}\omega^{2i} - w_{2k-3}\omega^i}{\omega^i}.$$

Par conséquent, les coefficients de W' sont reconstruits grâce au produit matrice×vecteur :

$$\begin{bmatrix} w_1 \\ w_2 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \frac{1}{n}\Omega^{-1} \cdot \begin{bmatrix} \widehat{w}_0 - w_0 - w_{n+1} - w_{n+2} \\ (\widehat{w}_1 - w_0 - w_{n+1}\omega - w_{n+2}\omega^2)\omega^{-1} \\ (\widehat{w}_2 - w_0 - w_{n+1}\omega^2 - w_{n+2}(\omega^2)^2)\omega^{-2} \\ \vdots \\ (\widehat{w}_{n-1} - w_0 - w_{n+1}\omega^{n-1} - w_{n+1}(\omega^{n-1})^2)\omega^{-(n-1)} \end{bmatrix}$$

Ce qui correspond à l'équation (III.2.7).

Le polynôme $W(X)$ est reconstruit par l'égalité

$$W(X) = w_0 + W'(X)X + w_{(2k-3)}X^{(2k-3)} + w_{(2k-2)}X^{(2k-2)}.$$

□

Les lemmes précédents sont des cas particuliers, pouvant être étendus à des valeurs inférieurs de n par l'augmentation du nombre de coefficients w_i précalculés pour les plus petits indices $i = 1, 2, \ldots$ et les plus grands $i = 2k-2, 2k-3, \ldots$. Ces calculs doivent être effectués juste avant l'étape d'interpolation nécessitant l'utilisation de Ω^{-1}.

***Remarque* III.2.8.** Dans le cadre des lemmes III.2.6 et III.2.7 nous pouvons utiliser une représentation de Montgomery pour les polynômes, ce qui permettrait d'éviter la multiplication de Ω^{-1} par $\frac{1}{n}$. Pour ce faire, il suffit d'effectuer selon les cas une ou trois multiplications par $\frac{1}{n}$. Par exemple, si le lemme III.2.6 est appliqué à des polynômes en représentation de Montgomery $\widetilde{U} = \frac{1}{n}U$ et $\widetilde{V} = \frac{1}{n}V$, nous cherchons $\widetilde{W} = \frac{1}{n}U \times V$. Nous devons tout d'abord déterminer le coefficient $\widetilde{w}_{2k-2}$ en calculant
$\widetilde{w}_{2k-2} = \frac{1}{n}w_{2k-2} = n\widetilde{U}_{2k-1} \times \widetilde{V}_{2k-1}$, puis les coefficients $\widetilde{w}_i$ pour $i = 0, \ldots, 2k-1$ sont obtenus par une interpolation polynomiale, en représentation de Montgomery. Dans ce cas nous n'avons qu'à faire une seule multiplication par $\frac{1}{n}$ pour harmoniser la représentation.

La même méthode peut être appliquée au lemme III.2.7 en ajoutant trois multiplications par $\frac{1}{n} \bmod(p)$.

III.2.5 Complexité des différentes méthodes DFT

Nous donnons ici la complexité des différentes approches DFT pour la multiplication. Nous distinguons les cas où la DFT est faite en utilisant un produit matrice×vecteur, et ceux pour lesquels la DFT est calculée en utilisant un algorithme FFT. Nous exprimons les coûts de calcul en nombre d'opérations dans $\mathbb{F}_p$: nous comptons le nombre de multiplication par des racines de l'unité notée M_ω, d'additions et soustractions notées A_p et de multiplications quelconques notées M_p.

Pour l'évaluation et l'interpolation, nous utilisons le fait que les entrées de Ω et Ω^{-1} sont des puissances de ω. Nous supposons que la multiplication est faite en utilisant la représentation de Montgomery, ceci afin d'éviter les n produits par $\frac{1}{n} \bmod(p)$ (pour plus de détails voir la partie III.2.2 et la remarque III.2.8). Nous obtenons alors les complexités données dans le Tableau III.1.

Méthode	M_{ω^i}	M_p
DFT générale	$4nk-3n$	n
FFT générale	$\frac{3n}{2}\log_2(n)$	n
Lemme III.2.6	$3(2k-3)^2$	$2k$
Lemme III.2.6 avec FFT	$3(k-1)\log_2(2k-2)$	$2k$
Lemme III.2.7	$3(2k-5)^2+2(2k-5)$	$2k+3$
Lemme III.2.7 avec FFT	$3(k-2)\log_2(2k-4)+3(2k-5)$	$2k+3$

Méthode	A_p
DFT générale	$4nk-3n$
FFT générale	$3n\log_2(n)$
Lemme III.2.6	$(2k-2)(6k-8)$
Lemme III.2.6 avec FFT	$3(2k-2)\log_2(2k-2)+(2k-2)$
Lemme III.2.7	$3(2k-4)(2k-4-1)+2(2k-4)$
Lemme III.2.7 avec FFT	$3(2k-4)(\log_2(2k-4)+1)$

TABLE III.1 – Complexité de l'approche DFT

De manière générale, la multiplication par DFT est très intéressante lorsque les multiplications par les racines de l'unité peuvent être effectuées efficacement. Ce n'est pas le cas pour des corps finis en représentation quelconque. En effet, pour un nombre premier p choisi de manière aléatoire, les racines de l'unité ont rarement des représentations facilitant la multiplication par ces valeurs. Nous introduisons une représentation particulière des corps finis, la représentation en base adaptée ou AMNS, développée par Thomas Plantard pendant sa thèse [Pla05]. Cette représentation peut être vue comme une généralisation de la famille des nombres de Mersenne [Pla05, Partie 2.4 p. 34]. La construction de cette représentation est telle que les

multiplications par des racines de l'unité consistent juste en des décalages dans la base AMNS. Cette particularité nous permettra d'utiliser la multiplication par DFT que nous venons de présenter, afin d'améliorer l'efficacité des calculs dans les corps $\mathbb{F}_{p^k}$.

III.3 Arithmétique des corps finis en représentation AMNS

III.3.1 Définition d'une représentation AMNS

Les opérations arithmétiques modulo un nombre premier p sont composées d'additions et/ou de multiplications d'entiers compris entre 0 et p, suivies éventuellement de la réduction modulo p du résultat si celui ci est supérieur à p.

L'efficacité de l'arithmétique modulo p dépend fortement du système de représentation utilisé pour représenter un élément de $\mathbb{F}_p$. Nous sommes habitués à utiliser des systèmes de numération simple à position, comme la base décimale ou binaire. Un système de numération simple à position est un système reposant sur l'utilisation d'une base pour écrire les entiers. De manière générale, un entier a est représenté comme étant une somme $a = \sum_{i=0}^{\ell-1} a_i \gamma^i$ avec $0 \leq a_i < \gamma$ et γ^ℓ qui soit supérieur à p. L'élément a est alors représenté dans la base γ, par exemple pour $\gamma = 2$ nous parlons de base binaire, pour $\gamma = 10$ de base décimale ou encore pour $\gamma = 16$ de base hexadécimale. Lors d'implémentation des calculs pour un ordinateur, γ est choisi de la forme 2^w où w est un entier représentant la taille en bit d'un mot machine.

Nous présentons un système de représentation original des éléments de $\mathbb{F}_p$ appelé *système adapté à la représentation modulaire*, en anglais *Adapted Modular Number System* et noté AMNS par la suite, développée par Thomas Plantard durant sa thèse [Pla05]. L'idée principale de la représentation AMNS est de jouer sur le paramètre γ servant à définir la base utilisée. L'élément γ peut être un élément quelconque de l'intervalle $]0, p[$ tel que tout élément a appartenant à $\mathbb{F}_p$ $(0 \leq a < p)$ puisse s'écrire $a = \sum_{i=0}^{\ell-1} a_i \gamma^i \mod p$ avec a_i appartenant à l'intervalle d'entiers $]-\rho, \rho]$, où ρ est un entier dont l'ordre de grandeur est $p^{1/\ell}$. La définition d'une base AMNS est la suivante.

Définition III.3.1 (AMNS [Pla05])**.** Une base AMNS $\mathcal{B}$, est un quintuplet $(p, \ell, \gamma, \rho, \lambda)$, où le polynôme de réduction modulaire est $P_{\mathcal{B}}(\delta) = \delta^\ell - \lambda$ tel que $\gamma^\ell - \lambda \equiv 0 \mod p$ et tel que pour tout entier $0 \leq a < p$ il existe un polynôme

$\mathfrak{a}(\delta) = \sum_{i=0}^{\ell-1} a_i \delta^i$ satisfaisant :

$$\begin{array}{l} \mathfrak{a}(\gamma) \equiv a \bmod p, \\ \deg(\mathfrak{a}(\delta)) < \ell, \\ \|\mathfrak{a}\|_\infty = \max_{i=1\ldots\ell} |a_i| < \rho. \end{array} \tag{III.3.1}$$

Le polynôme $\mathfrak{a}(\delta)$ est une représentation de a dans $\mathcal{B}$.

***Exemple* III.3.2.** Le Tableau III.2 donne une représentation dans la base AMNS $\mathcal{B} = (19, 3, 7, 2, 1)$ de chaque élément modulo $p = 19$. Nous choisissons $\gamma = 7$, ainsi $\gamma^2 \equiv 11 \bmod(p)$ et $\gamma^3 \equiv 1 \bmod(p)$. La racine cubique de γ est strictement inférieur à 3 ; la valeur ρ doit donc être inférieure strictement à 3 aussi, comme il s'agit d'entiers, cela nous impose d'exprimer les éléments de $\mathbb{F}_p$ en des polynômes dont les coefficients seront 0, 1 et −1.

Pour remplir le Tableau III.2, nous commençons par 1 et 18. Puis 7, 6 et 8 car $\gamma = 7$, $6 = 7 - 1$ et $8 = 7 + 1$. Nous faisons de même pour trouver 10, 11 et 12 sachant que $\gamma^2 \equiv 11 \bmod(19)$.

Par symétrie, nous trouvons $13 = 19 - 6$; 3 est obtenu comme étant la différence de 10 et 7, $3 = \delta^2 - \delta - 1$, nous obtenons ensuite 4 car $4 = 3 + 1$, et 5 car $5 = 4 + 1$. Avec la donnée de 5 en base AMNS, nous pouvons trouver la représentation de 14 par symétrie. La valeur 9 est obtenue comme étant 13–4, enfin $2 = 9 - 7$ et $17 = 19 - 2$ nous permettent d'obtenir la représentation de chacun des entiers de $\mathbb{Z}_{19}$ dans la base AMNS $\mathcal{B} = MNS(19, 2, 7, 1)$, décrite dans le Tableau III.2.

1	2	3	4	5	6
1	$-\delta^2 - \delta - 1$	$\delta^2 - \delta - 1$	$\delta^2 - \delta$	$\delta^2 - \delta + 1$	$\delta - 1$

7	8	9	10	11	12
δ	$\delta + 1$	$-\delta^2 + 1$	$\delta^2 - 1$	δ^2	$\delta^2 + 1$

13	14	15	16	17	18
$-\delta + 1$	$-\delta^2 + \delta - 1$	$-\delta^2 + \delta$	$-\delta^2 + \delta + 1$	$\delta^2 + \delta + 1$	-1

TABLE III.2 – Les éléments de $\mathbb{Z}_{19}$ dans la base AMNS $\mathcal{B} = MNS(19, 2, 7, 1)$

En particulier, nous pouvons vérifier que l'évaluation du polynôme $(-1 - \delta + \delta^2)$ en γ donne pour résultat $-1 - \gamma + \gamma^2 = -1 - 7 + 49 = 41 \equiv 3 \bmod 19$; et de plus que $\| - 1 - \delta + \delta^2\|_\infty = 1 < 2$.

***Remarque* III.3.3.** Nous pouvons remarquer que la représentation en base AMNS est redondante, un élément peut admettre plusieurs représentation dans une même base. Par exemple, dans le Tableau III.2, l'entier 8 se décompose comme $\delta + 1$, mais si l'on considère que $8 = 9 - 1$, nous obtenons

alors que $8 = -\delta^2$. Pour le moment cette redondance n'est pas exploitée en tant que telle.

De manière générale, lorsque nous construisons une base AMNS, nous faisons en sorte que les coefficients $|a_i|$ de la décomposition dans la base γ soient petits : $a_i < \rho \cong p^{1/\ell}$. Cette condition facilite la multiplication dans la base adaptée, comme nous le montrons dans la partie suivante.

III.3.2 Multiplication dans une base AMNS

L'intérêt de la représentation AMNS est d'obtenir des propriétés de calcul similaires à celles des pseudo Mersenne [Cra92], en effet le choix de γ tel que $\gamma^\ell = \lambda \mod p$ pour un petit entier λ assure une réduction modulaire très simple dans le corps $\mathbb{F}_p$, comme nous l'illustrons ci-dessous.

Une multiplication quelconque en représentation AMNS pourrait s'effectuer en trois étapes distinctes. Soient $\mathfrak{a}(\delta) = \sum_{i=0}^{\ell-1} a_i\delta^i$ et $\mathfrak{b}(\delta) = \sum_{i=0}^{\ell-1} b_i\delta^i$ deux éléments dans la base $\mathcal{B} = (p, \ell, \gamma, \rho, \lambda)$. Nous souhaitons obtenir l'expression de $\mathfrak{c}$ leur produit.

1. Nous calculons d'abord $C(\delta)$ le résultat du produit des polynômes $\mathfrak{a}(\delta)$ et $\mathfrak{b}(\delta)$:
$$C(\delta) = \sum_{i=0}^{2\ell-2} C_i\delta^i,$$
avec $C_i < \ell\rho^2$.

2. Nous réduisons ensuite le polynôme obtenu modulo $P_\mathcal{B} = \delta^\ell - \lambda$
$$\tilde{c} = \sum_{i=0}^{\ell-2}(C_i + C_{\ell+i})\delta^i + C_{\ell-1}\delta^{\ell-1},$$
avec $\tilde{c} < 2\ell\rho^2|\lambda|$

3. Enfin, la dernière étape est la réduction des coefficients de $\tilde{c}$.

Nous décrirons la méthode de réduction des coefficients par la suite, nous développons d'abord un exemple de produit dans une base AMNS.

***Exemple* III.3.4.** Soit $\mathfrak{a}(\delta) = -\delta^2 + \delta$ et $\mathfrak{b}(\delta) = -\delta^2 + \delta + 1$ deux éléments de $\mathbb{Z}_{19}$ dans la base AMNS $\mathcal{B} = MNS(19, 2, 7, 1)$ décrite dans l'exemple III.3.2. Soit $\mathfrak{c}$ le produit de $\mathfrak{a}$ et $\mathfrak{b}$.

$$\begin{aligned}\mathfrak{c}(\delta) &= \mathfrak{a}(\delta) \times \mathfrak{b}(\delta),\\ &= (\delta + 2) \times (-\delta^2 + \delta + 1),\\ &= \delta^4 - 2\delta^3 + \delta\end{aligned}$$

Sachant que $\gamma^3 \equiv 1 \bmod(19)$, et donc que $\gamma^4 \equiv \gamma \bmod(19)$ la réduction modulaire polynomiale se réduit à une addition d'entier :

$$\mathfrak{c}(\delta) = \delta - 2 + \delta = 2\delta - 2.$$

Afin de retrouver $\mathfrak{c}$ en représentation AMNS nous devons réduire ces coefficients.

Les deux premières étapes sont assez simples à réaliser. La multiplication polynômiale est effectuée à l'aide des multiplications classiques que nous avons introduites dans la partie II.3. Le degré $\ell-1$ des polynômes déterminera si nous utilisons la multiplication scolaire, celle de Karatsuba et/ou Toom Cook ou encore la multiplication par FFT. La construction de la base AMNS permet d'avoir une réduction polynomiale très simple, de complexité au plus $\ell-1$ additions. Il ne reste alors plus qu'à réduire les coefficients du polynôme, la partie délicate de ce produit. Nous ne présentons pas ici les algorithmes de réduction développés par Thomas Plantard dans la partie 3.2.4 de sa thèse [Pla05]. En effet, l'algorithme de réduction présenté dans [Pla05] est efficace pour une classe spécifique de nombres premiers. Or, les corps finis utilisés en cryptographie à base des couplages sont construits avec des nombres premiers p choisis de manière aléatoire. La multiplication modulaire dans $\mathbb{F}_p$ ne peut alors utiliser la forme particulière de p, ce qui est la cas par exemple pour des nombres de Mersenne. Afin d'effectuer efficacement la multiplication dans la représentation AMNS, nous utilisons la multiplication proposée par Christophe Nègre et Thomas Plantard dans [NP08]. Cette multiplication est fondée sur le schéma de la multiplication de Montgomery. Elle est adaptée aux corps finis utilisés en cryptographie à base de couplage. Pour effectuer cette multiplication, nous devons faire intervenir un polynôme $\mathfrak{m}$ admettant γ comme racine modulo p ; ainsi qu'un entier ϕ supérieur à une valeur fixée par les paramètres de la base AMNS $2\ell\lambda\rho$. Le polynôme $\mathfrak{m}$ peut être obtenu en considérant $\mathcal{L}$ le réseau des polynômes de degré inférieur à ℓ et admettant γ comme racine. Ce réseau de rang ℓ est le suivant :

$$\mathcal{L} = \{\mathfrak{a}(\delta) \in \mathbb{Z}[\delta] \text{ tel que } \deg \mathfrak{a}(\delta) < \ell \text{ et } \mathfrak{a}(\gamma) \equiv 0 \mod p\}.$$

Il peut être représenté par la matrice $\mathcal{B}$ construite ligne par ligne :

$$\mathcal{B} = \begin{pmatrix} p & 0 & 0 & 0 & \dots & 0 \\ -\gamma & 1 & 0 & 0 & \dots & 0 \\ -\gamma^2 & 0 & 1 & 0 & \dots & 0 \\ \vdots & & & \ddots & & \vdots \\ -\gamma^{\ell-2} & 0 & 0 & \dots & 1 & 0 \\ -\gamma^{\ell-1} & 0 & 0 & \dots & 0 & 1 \end{pmatrix} \begin{matrix} \leftarrow p \\ \leftarrow \delta - \gamma \\ \leftarrow \delta^2 - \gamma^2 \\ \vdots \\ \leftarrow \delta^{\ell-2} - \gamma^{\ell-2} \\ \leftarrow \delta^{\ell-1} - \gamma^{\ell-1} \end{matrix} . \tag{III.3.2}$$

Le théorème de Minkowski [Min10] nous assure de l'existence d'un polynôme $\mathfrak{m}$ s'annulant en γ et vérifiant que $\|\mathfrak{m}\|_\infty \leq p^{1/\ell}$. Ce polynôme est utilisé

dans l'algorithme de multiplication, décrit par l'Algorithme 5 pour annuler la partie basse des coefficients.

Avant d'introduire l'algorithme général, nous expliquons schématiquement la méthode employée. Le réseau $\mathcal{L}$ peut être représenté par le schéma III.1, nous avons représenté un plus court vecteur $\mathfrak{m}$ de ce réseau.

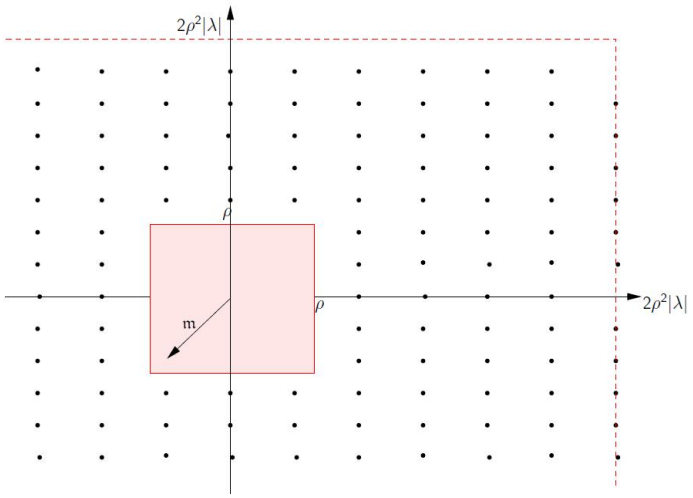


FIGURE III.1 – Réseau de la base AMNS.

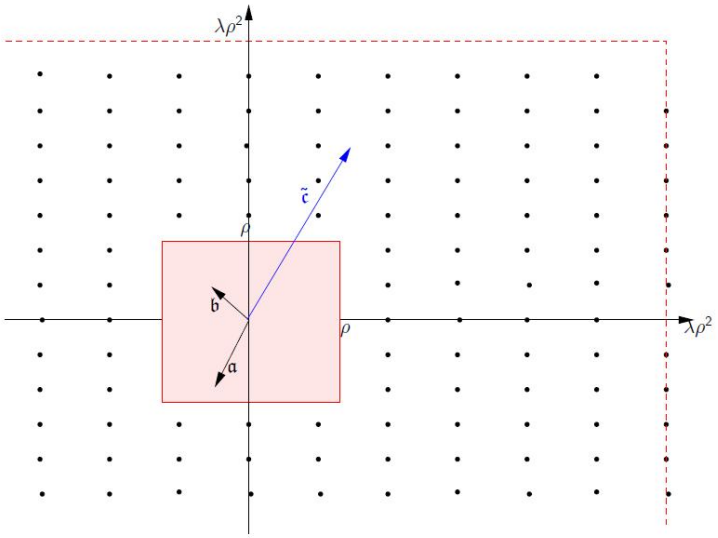


FIGURE III.2 – Vecteur $\tilde{\mathfrak{c}}$ après réduction polynomiale.

Soient $\mathfrak{a}$ et $\mathfrak{b}$ deux éléments représentés en base adaptée. Leur produit $\tilde{\mathfrak{c}}$ est un vecteur dont les coefficients, après réduction modulo $P_{\mathcal{B}}$, sont inférieurs à $2\ell\rho^2|\lambda|$, comme illustré par la Figure III.2.

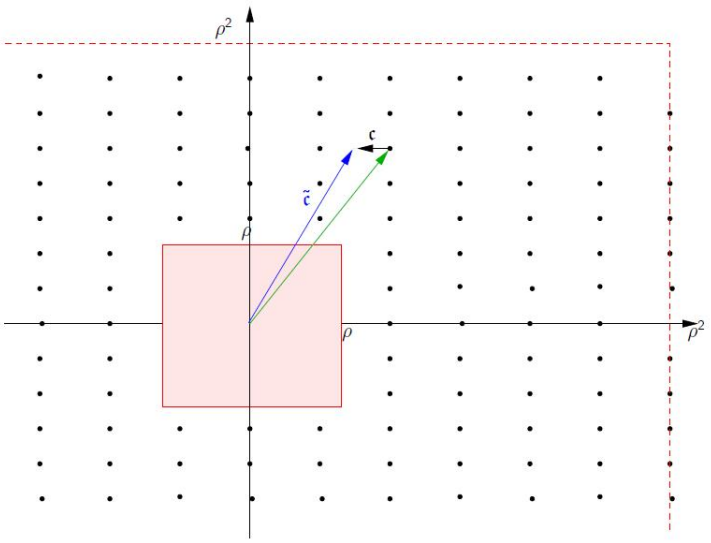


FIGURE III.3 – Vecteur $\mathfrak{c} \equiv \mathfrak{a} \times \mathfrak{b} \bmod(p)$

Pour réduire les coefficients de $\tilde{\mathfrak{c}}$, afin d'obtenir $\mathfrak{c}$ un représentant de $\mathfrak{a} \times \mathfrak{b} \bmod(p)$ en base AMNS, nous cherchons le vecteur du réseau le plus proche de $\tilde{\mathfrak{c}}$ et nous le lui retranchons comme illustré par la Figure III.3.

La méthode de multiplication est résumée dans l'algorithme suivant introduit dans [NP08]. Cet algorithme prend en argument deux éléments $\mathfrak{a}, \mathfrak{b}$ d'un corps fini en représentation AMNS, c'est à dire que $\mathfrak{a}$ et $\mathfrak{b}$ sont des polynômes en δ, et renvoie le polynôme $\mathfrak{r}(\delta)$ qui correspond à un facteur près à $\mathfrak{a} \times \mathfrak{b}$ modulo p. Nous introduisons la notation $\bmod(P_{\mathcal{B}}, \phi)$, qui représente la double réduction modulaire d'un polynôme ; la première réduction est la réduction modulo le polynôme $P_{\mathcal{B}}$ et la seconde réduction consiste en la réduction modulo ϕ des coefficients de ce polynôme. Le polynôme $\mathfrak{m}$ doit être inversible $\bmod(P_{\mathcal{B}}, \phi)$.

Algorithme 5 : AMNS Multiplication

Entrée : $\mathfrak{a}, \mathfrak{b} \in \mathcal{B} = (p, \ell, \gamma, \rho, \lambda)$ avec $P_{\mathcal{B}} = \delta^{\ell} - \lambda$
Donnée : $\mathfrak{m}$ un polynôme tel que $\mathfrak{m}(\gamma) \equiv 0 \pmod{p}$
un entier $\phi \geq 2\ell\lambda\rho$ et $\mathfrak{m}' = -\mathfrak{m}^{-1} \bmod (P_{\mathcal{B}}, \phi)$
Résultat : $\mathfrak{r}(\delta)$ tel que $\mathfrak{r}(\gamma) = \mathfrak{a}(\gamma)\mathfrak{b}(\gamma)\phi^{-1} \bmod p$
début
$\mathfrak{c} \leftarrow \mathfrak{a} \times \mathfrak{b} \bmod P_{\mathcal{B}}$;
$\mathfrak{q} \leftarrow \mathfrak{c} \times \mathfrak{m}' \bmod (P_{\mathcal{B}}, \phi)$;
$\mathfrak{r} \leftarrow (\mathfrak{c} + \mathfrak{q} \times \mathfrak{m} \bmod P_{\mathcal{B}})/\phi$;
fin

Cet algorithme est similaire à l'algorithme de multiplication de Montgomery, nous disposons donc d'une arithmétique efficace pour les corps re-

présentés en base AMNS. L'article [NP08] montre que cet algorithme est correct si $\phi \geq 2\ell\lambda\rho$. L'implémentation de l'Algorithme 5 requiert essentiellement trois multiplications de polynômes $(\mathfrak{a} \times \mathfrak{b}, \mathfrak{c} \times \mathfrak{m}'$ et $\mathfrak{q} \times \mathfrak{m})$, pour des polynômes dont les coefficients sont inférieurs à ρ et ϕ. Il requiert aussi des réductions modulo $P_{\mathcal{B}}$ et ϕ. Nous avons vu que le choix du polynôme $P_{\mathcal{B}}$ réduit la réduction polynomiale à quelques additions. Afin de faciliter la réduction modulo ϕ, l'entier ϕ est choisi de préférence comme la plus petite puissance de deux supérieure à $2\ell\lambda\rho$. Cet algorithme présente l'avantage d'effectuer la multiplication et la réduction des coefficients de deux éléments en représentation AMNS. Le résultat de cette multiplication est un élément représenté dans la base AMNS à multiplication par ϕ^{-1} près. La complexité de cet algorithme repose sur la complexité des trois multiplications polynomiales.

Le choix de l'algorithme de multiplication polynomiale dépend fortement de la taille de ℓ et de la taille de la base. Nous retrouvons les mêmes critères de sélection que pour une arithmétique classique. Pour des entiers ℓ de petites tailles la multiplication scolaire est généralement recommandée, pour de plus grands ℓ les multiplications de Karatsuba ou Toom Cook sont privilégiées. Nous choisissons parmi ces deux méthodes précédemment citées celle qui rend le produit polynomiale le plus efficace dans les cas que nous traitons.

III.3.3 Existence d'une base AMNS

Soit p un nombre premier. Nous décrivons une méthode pour construire une base AMNS pour un corps $\mathbb{F}_p$. La construction d'une base repose sur l'existence d'entiers $(\lambda, \ell) \in \mathbb{Z} \times \mathbb{N}$ tels que le polynôme $P_{\mathcal{B}}(\delta) = \delta^\ell - \lambda$ admette une racine γ dans $\mathbb{F}_p$. Pour pouvoir calculer efficacement la multiplication nous devons construire un polynôme $\mathfrak{m}$ admettant γ comme racine. Ce polynôme est un plus court vecteur du réseau $\mathcal{L}$ décrit par la matrice III.3.2. Le polynôme $\mathfrak{m}$ est obtenu en appliquant à $\mathcal{L}$ l'algorithme de réduction des réseaux LLL [Pau98]. Ainsi, la valeur de ρ se déduit de celle de ℓ et λ comme le montre la proposition III.3.5.

Proposition III.3.5. *Soit p un entier premier et $(\lambda, \ell) \in \mathbb{Z} \times \mathbb{N}$ tels que le polynôme $P_{\mathcal{B}} = \delta^\ell - \lambda$ admette une racine γ dans $\mathbb{F}_p$. Les assertions suivantes sont vraies :*

1. *Il existe un polynôme $\mathfrak{m}(\delta)$ tel que $\mathfrak{m}(\gamma) = 0$ et $\|\mathfrak{m}\|_\infty \leq (\ell!)^{1/\ell} p^{1/\ell}$.*
2. *Soient $\sigma = \|\mathfrak{m}\|_\infty$ et $\rho = 2\ell\sigma|\lambda|$ alors le système $\mathcal{B} = (p, \ell, \gamma, \rho, \lambda)$ est une AMNS pour $\mathbb{F}_p$.*
3. *Étant donné $\phi \geq 2n\lambda\rho$ si $\mathfrak{m}$ admet un inverse $\mathfrak{m}'$ dans $\mathbb{Z}_\phi[\delta]/(\delta^n - \lambda)$, alors l'Algorithme 5 permet d'effectuer la multiplication dans la base $\mathcal{B}$.*

La démonstration de cette proposition est faite dans [NP08]. Ainsi, la construction d'une base AMNS se fait par exemple en utilisant l'algorithme suivant.

Algorithme 6 : Construction d'une base AMNS

Entrée : p un nombre premier.
Résultat : $\mathcal{B} = (p, \ell, \gamma, \rho, \lambda)$ une base adaptée pour le corps $\mathbb{F}_p$.
début
  Choisir des valeurs $(\lambda, \ell) \in \mathbb{Z} \times \mathbb{N}$.
  si $P_{\mathcal{B}} = \delta^\ell - \lambda \ mod(p)$ *admet des racines dans* $\mathbb{F}_p$ **alors**
    Construire la Matrice $\mathcal{B}$ définie par l'équation III.3.2.
    Réduire cette Matrice à l'aide de LLL.
    Choisir un plus court vecteur $\mathfrak{m}$ du réseau réduit.
    si $\|\mathfrak{m}\|_\infty \leq (\ell!)^{1/\ell} p^{1/\ell}$ **alors**
      Poser $\sigma = \|\mathfrak{m}\|_\infty$ et $\rho = 2\ell\sigma|\lambda|$.
      La base $\mathcal{B} = (p, \ell, \gamma, \rho, \lambda)$ est une AMNS pour $\mathbb{F}_p$.
      La recherche a aboutit.
    fin
  sinon
    Changer les valeurs (λ, ℓ) et recommencer.
  fin
fin

III.4 Les corps adaptés au calcul des couplages pour la DFT

Nous venons de présenter la multiplication par DFT nécessitant une multiplication par des racines de l'unité efficace ; ainsi qu'une représentation des corps finis $\mathbb{F}_p$ pour laquelle l'élément définissant la base est choisi comme la racine d'un polynôme de la forme $X^\ell - \lambda$. Nous nous intéressons à des corps spécifiques que nous nommons ***corps adaptés au calcul des couplages pour la DFT***. Ces corps admettent une base AMNS pour laquelle la multiplication par des racines de l'unité est efficace.

III.4.1 Définition des corps adaptés au calcul des couplages pour la DFT

Nous étendons la notion de corps adaptés au calcul des couplages pour les couplages introduits par Koblitz et Menezes [KM05], en définissant des corps adaptés au calcul des couplages pour la DFT. Les corps adaptés au calcul des couplages introduits par Koblitz et Menezes se caractérisent par le fait

que le degré de plongement de la courbe est de la forme $k = 2^i 3^j$. Ceci permet d'utiliser les multiplications de Karatsuba et Toom Cook dans $\mathbb{F}_{p^k}$ qui sont recommandées actuellement pour des degrés d'extension 2 et 3. Nous allons combiner les deux parties III.2.4 et III.3 pour mettre en place une multiplication efficace pour des valeurs de k plus générales. Le but principal est de trouver un corps $\mathbb{F}_p$ possédant des racines n-ième de l'unité pour des valeurs de $n \in \{2k-1, 2k-2, 2k-4\}$: ceci afin de réaliser les interpolations décrites dans la partie III.2.4. Il faut de plus que la multiplication par ces racines de l'unité soit peu coûteuse. Nous avons privilégié de travailler avec des corps $\mathbb{F}_{p^k}$ satisfaisant la définition suivante.

Définition III.4.1 (corps adaptés au calcul des couplages pour la DFT)**.** Nous appelons **corps adaptés au calcul des couplages pour la DFT** une extension de corps $\mathbb{F}_{p^k}$ telle que $\mathbb{F}_p$ admette une base AMNS $\mathcal{B} = (p, \ell, \gamma, \rho, \lambda)$ de longueur ℓ vérifiant que $\gamma^\ell - \lambda \equiv 0 \bmod(p)$ et l'une des propriétés suivantes :

1. $\lambda = 1$, $\ell \in \{2k-1, 2k-2, 2k-4\}$ et γ est une racine ℓ-ème primitive de l'unité.

2. $\lambda = -1$, $\ell \in \{k-1, k-2\}$ et γ est une racine 2ℓ-ème primitive de l'unité.

Cette définition repose sur la construction d'une base AMNS. Dans cette définition, les corps sont dits amis pour la DFT car leur base est construite sur mesure pour que la multiplication par les racines de l'unité soit efficace. En effet, par construction, γ est racine du polynôme $(X^\ell - \lambda) \bmod(p)$, pour $\lambda = 1$ (respectivement $\lambda = -1$), γ est donc une racine ℓ-ème (respectivement 2ℓ-ème) primitive de l'unité. Ainsi, les racines de l'unité sont les éléments $\pm\gamma^i$.

Étant donné que dans une base AMNS, γ est l'élément servant de base de décomposition, la multiplication par ces racines consiste uniquement en des décalages dans la base $\mathcal{B}$ comme le montre le lemme suivant.

Lemme III.4.2. *Soit $\mathcal{B} = (p, \ell, \gamma, \rho, \lambda)$ une base AMNS de $\mathbb{F}_p$ et $\mathfrak{a} = \sum_{i=0}^{n-1} a_i \gamma^i$ un élément dans la base $\mathcal{B}$. Le produit entre $\mathfrak{a}$ et les puissance γ^i de γ s'effectue en appliquant la formule :*

$$\mathfrak{a}\gamma^i = \lambda a_{n-i} + \lambda a_{n-i+1}\gamma + \cdots + \lambda a_{n-1}\gamma^{i-1} + a_0\gamma^i + \cdots + a_{n-i-1}\gamma^{n-1}$$

Démonstration. La preuve est une conséquence directe de la définition d'une représentation AMNS, et repose sur le fait que $\gamma^\ell - \lambda \equiv 0 \bmod(p)$. □

Les cas auxquels nous nous intéressons sont les corps $\mathbb{F}_p$ pour lesquels il existe une représentation AMNS telle que $\lambda = \pm 1$. Si le corps vérifie cette condition, alors la multiplication par $\pm\gamma^i$ dans le corps $\mathbb{F}_p$ sera uniquement constituée de décalages, et éventuellement de changements de signes. La multiplication par $\pm\gamma^i$ se calcule presque sans aucune opération sur le corps fini. Par conséquent, cette multiplication n'ajoute pas d'opérations au décompte total des opérations. Le coût total d'une multiplication AMNS en terme d'opérations sur le corps fini est donné dans le Tableau III.3 dans la partie III.4.3.

L'utilisation des bases adaptées nous permet de construire des racines de l'unité dans $\mathbb{F}_p$ telles que la multiplication par ces racines est très efficace. Nous pouvons donc utiliser la méthode DFT présentée dans la partie III.2 pour effectuer la multiplication d'éléments de $\mathbb{F}_{p^k}$. En effet, la condition sur ℓ pour chacun des cas de la Définition III.4.1 nous autorise à utiliser au moins l'une ou l'autre des stratégies développées dans le paragraphe III.2.1, soit le Lemme III.2.6, soit le Lemme III.2.7.

III.4.2 Corps utilisés en cryptographie pour les couplages

Les méthodes de constructions de courbes ont été décrites dans la partie I.5. Nous rappelons les éléments importants pour construire des courbes elliptiques ordinaires adaptées au calcul des couplages. Le cardinal $\#E(\mathbb{F}_p)$ de la courbe définie sur le corps $\mathbb{F}_p$ doit posséder un grand facteur premier, que nous noterons r et un degré de plongement k tel que $6 < k \leq 32$. Pour obtenir des courbes satisfaisants ces conditions, la méthode la plus efficace est d'utiliser la Multiplication Complexe.

Il est important de souligner qu'avec les méthodes de construction de courbes utilisées, le nombre premier p est construit aléatoirement. La proposition III.3.5 donne la méthode pour construire une base AMNS satisfaisant la définition III.4.1. Il suffit pour cela de pouvoir trouver une racine primitive ℓ-ième ou 2ℓ-ième de l'unité, pour un entier ℓ satisfaisant la définition III.4.1.

Soit p un nombre premier pris au hasard, la probabilité qu'il existe une racine ℓ-ième primitive de l'unité est de $1/(\phi(\ell) - 1)$, comme le montre le théorème suivant.

Théorème III.4.3. *Soit p un nombre premier et ℓ un entier compris dans l'intervalle d'entiers $]0; p[$.*
Il existe dans $\mathbb{F}_p$ une racine ℓ-ième primitive de l'unité si et seulement si $p \equiv 1 \mod \ell$.

Démonstration. La démonstration de ce théorème repose sur le petit théorème de Fermat. Si $p \equiv 1 \mod \ell$, cela signifie que ℓ divise $p-1$. Or d'après le petit théorème de Fermat, quel que soit X non nul, $X^{p-1} \equiv 1 \bmod(p)$. Le polynôme $X^{p-1}-1$ admet $p-1$ racines distinctes dans $\mathbb{F}_p$, il s'écrit donc $X^{p-1}-1 = \prod_{i=1}^{p-1}(X-i)$.

De plus, il peut se factoriser en $(X^\ell-1)(X^{\frac{p-1}{\ell}} + X^{\frac{p-1}{\ell}-1} + \ldots + 1)$. Le facteur $(X^\ell-1)$ admet lui aussi ℓ racines distinctes : les racines ℓ-ième de l'unité, dont au moins une est primitive.

Réciproquement, supposons que γ soit une racine primitive ℓ-ème de l'unité. Si $p-1 \neq 0 \mod \ell$, notons r le reste de la division de $p-1$ par ℓ. Nous obtenons alors l'égalité suivante $\gamma^r \equiv 1 \bmod(p)$, et γ serait une racine r-ième de l'unité. Ceci est en contradiction avec l'hypothèse que γ est une racine ℓ-ième primitive de l'unité. En effet, si γ est une racine primitive ℓ-ième de l'unité, alors pour tout entier $j \in [1, \ell-1]$, $\gamma^j \neq 1 \bmod(p)$. Par conséquent, p doit forcément vérifier l'égalité $p \equiv 1 \mod \ell$.

□

Nous savons que l'ensemble des nombres premiers devrait être équiréparti dans l'ensemble des classes d'équivalence modulo ℓ (d'après le théorème de la progression arithmétique de Dirichlet [1] ; donc pour de petites valeurs de ℓ il est facile de trouver un corps adapté au calcul des couplages $\mathbb{F}_p$ pour la DFT tel que la courbe elliptique permette un calcul de couplage efficace sur ce corps.

Nous illustrons la construction d'une représentation AMNS en trouvant une base associée à une famille de corps finis utilisée en cryptographie à base de couplages. Nous traitons ici le cas des courbes elliptiques dont le degré de plongement est 16.

***Exemple* III.4.4.** Nous utilisons la paramétrisation donnant les éléments caractéristiques de la courbe elliptique de [KSS08] :

$$\begin{aligned} k &= 16, \\ p &= (x^{10} + 2x^9 + 5x^8 + 48x^6 + 152x^5 + 240x^4 + 625x^2 + 2398x + 3125)/980, \\ r &= (x^8 + 48x^4 + 625)/61250, \\ t &= (2x^5 + 41x + 35)/35. \end{aligned}$$

Où, k est le degré de plongement de la courbe, p est la caractéristique de $\mathbb{F}_p$, r l'ordre des sous groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_3$, et t la trace du Frobenius sur la courbe elliptique. Nous choisissons la valeur de l'entier $x \in \mathbb{Z}$ pour que les

[1] Jean-BenoîtBost, PierreColmezetPhilippeBianeLafonctionZêta, Éditionsdel'ÉcolepolytechniqueParis2002

nombres p et r soient des entiers premiers. Par exemple, $x = 74156485$, ce qui donne alors

$$\begin{aligned} p &= 513174771603192518069857791127277415092088396567880595361684047893395993456 1, \\ r &= 149309347072601793039402841900662885259628529084818905369 93, \\ t &= 128146760584932038247348983414439772062, \end{aligned}$$

Pour construire la base AMNS associée à ce p nous commençons par choisir un polynôme $P_{\mathcal{B}}(\delta) = \delta^{\ell} - \lambda$ de degré ℓ et nous calculons γ une racine de $P_{\mathcal{B}}$ dans $\mathbb{F}_p$. Nous avons choisi le polynôme $P_{\mathcal{B}}(\delta) = \delta^{16} + 1$. La racine de l'unité associée est $\gamma = 38696828658217738947551865824060486351009548229977673384131972138697740467 4$.

Puis nous construisons la matrice M suivante.

$$M = \begin{bmatrix} p & 0 & 0 & \cdots & 0 \\ -\gamma & 1 & 0 & \cdots & 0 \\ -\gamma^2 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ -\gamma^{(\ell-1)} & 0 & \cdots & 0 & 1 \end{bmatrix}$$

L'application d'un algorithme de réduction des réseau à cette matrice nous permet d'obtenir un court vecteur $\mathfrak{m}$ satisfaisant l'égalité $\mathfrak{m}(\gamma) = 0 \bmod p$.

$$\begin{aligned} \mathfrak{m}(\delta) = {} & 7400\delta + 49262\delta^2 - 3010\delta^3 - 14335\delta^4 + 34360\delta^5 \\ & + 43021\delta^6 + 6813\delta^7 + 5184\delta^8 + 13206\delta^9 + 10037\delta^{10} \\ & + 2540\delta^{11} - 7384\delta^{12} - 66117\delta^{13} - 57557\delta^{14} + 32450\delta^{15}. \end{aligned}$$

La norme infini du polynôme $\mathfrak{m}$ est 66117, ce qui permet de fixer la valeur de ρ à

$$\rho = 2\ell|\lambda|\|\mathfrak{m}\|_{\infty} = 2115744.$$

III.4.3 Comparaison des complexités

Dans le Tableau III.3, nous présentons la complexité d'une multiplication dans $\mathbb{F}_{p^k}$ un corps adapté au calcul des couplages pour la DFT, et ce pour

différentes valeurs de k. La complexité est donnée en termes de multiplications et d'additions dans $\mathbb{F}_p$. Cette complexité se déduit du Tableau III.1. Comme nous ne tenons pas compte des décalages ; nous négligeons le coût d'une multiplication par une racine de l'unité puisqu'elle consiste en des décalages. De plus nous précisons si la FFT est utilisée ou non.

Afin de pouvoir comparer notre arithmétique aux arithmétiques classiques, le Tableau III.3 donne aussi la complexité de la multiplication dans des corps adaptés au calcul des couplages tels que k soit un produit de puissances de 2 et 3. Pour ces corps, les approches de Karatsuba et Toom Cook sont a priori les plus efficaces II.

Méthode	k	Coût de M_{p^k}	
		A_p	M_p
Karatsuba/Toom-Cook [KM05, BE07]	8	72	27
Partie III.2.1 avec FFT et $P_{\mathcal{B}} = \delta^8 + 1$	8	192	16
Karatsuba/Toom-Cook [KM05, BE07]	9	160	25
Lemme III.2.6 avec FFT et $P_{\mathcal{B}} = \delta^8 + 1$	9	208	18
Lemme III.2.7 avec FFT et $P_{\mathcal{B}} = \delta^8 + 1$	10	240	23
Partie III.2.1 avec $P_{\mathcal{B}} = \sum_{i=0}^{10}(-t)^i$	11	902	22
Karatsuba/Toom-Cook [KM05, BE07]	12	180	45
Lemme III.2.6 avec $P_{\mathcal{B}} = \sum_{i=0}^{10}(-t)^i$	12	1408	24
Lemme III.2.7 avec $P_{\mathcal{B}} = \sum_{i=0}^{10}(-t)^i$	13	1430	28
Karatsuba/Toom-Cook [KM05, BE07]	16	248	81
Partie III.2.1 avec FFT et $P_{\mathcal{B}} = \delta^{16} + 1$	16	480	32
Lemme III.2.6 avec FFT et $P_{\mathcal{B}} = \delta^{16} + 1$	17	512	34
Lemme III.2.7 avec FFT et $P_{\mathcal{B}} = \delta^{16} + 1$	18	576	39
Karatsuba/Toom-Cook [KM05, BE07]	18	480	75
Karatsuba/Toom-Cook [KM05, BE07]	24	588	135

TABLE III.3 – Comparaison des complexités pour des degrés de plongement k donnés

Remarquons que même pour de petites valeurs de k, la méthode de multiplication par DFT avec utilisation de la FFT est compétitive en nombre de multiplications sur les corps finis. Lorsque la méthode FFT ne peut être utilisée, le nombre d'additions augmente de manière significative et dans certains cas bien spécifiques, peut rendre notre approche un peu moins compétitive.

Nous pouvons noter que pour les valeurs de k supérieures ou égales à 16,

notre méthode est clairement plus efficace que les approches de Karatsuba et Toom Cook.

Par exemple, pour $k = 16$, une multiplication dans $\mathbb{F}_{p^{16}}$ s'effectue en 81 multiplications et 248 additions dans $\mathbb{F}_p$. La multiplication basée sur notre méthode se fait en 32 multiplications et 480 additions dans $\mathbb{F}_p$. Notre approche permet un gain de 49 multiplications dans $\mathbb{F}_p$, pour un surcoût de 252 additions.

Le surcoût engendré par les additions est moins important que le coût de 49 multiplications dans $\mathbb{F}_p$. En effet, admettons que la multiplication dans $\mathbb{F}_p$ s'effectue en utilisant la méthode de Karatsuba, la complexité de cette multiplication est

$$5N^{\log_2(3)}A_w + N^{\log_2(3)}M_w,$$

où N est le nombre de mots machines formant les entiers manipulés, et où A_w et M_w représentent une addition et une multiplication d'un mot machine. La complexité d'une addition est NA_w. Considérons qu'une multiplication de mot machine s'effectue en m additions machines $M_w = mA_w$. Nous essayons de déterminer le nombre m à partir duquel 49 multiplications dans $\mathbb{F}_p$ ont une complexité machine plus importante que 252 additions dans $\mathbb{F}_p$.

Les 49 multiplications dans $\mathbb{F}_p$ ont une complexité de

$$49(5N^{\log_2(3)}(1+m))A_w.$$

Les 252 additions dans $\mathbb{F}_p$ ont une complexité de $252NA_w$.

Nous cherchons m telle que l'inégalité suivante soit vraie :

$$49(5N^{\log_2(3)}(1+m))A_w > 252NA_w$$

Nous simplifions cette inégalité en posant $N = 1$, ce qui est loin de la réalité car les nombres que nous manipulons sont au moins de taille 16 mots machine. Avec cette simplification, nous obtenons que m doit être supérieur à $1/35$ pour que 49 multiplications dans $\mathbb{F}_p$ aient une complexité plus importantes que 252 additions dans $\mathbb{F}_p$, cette hypothèse est plus que réaliste considérant les architectures actuelles.

Ainsi, les additions en surplus sont théoriquement largement compensées par les gains en nombre de multiplications. Nous implémentons notre multiplication pour vérifier si les gains théoriques sont vérifiés en pratique.

III.5 Conclusion

Nous venons de présenter une approche originale de la multiplication dans les extensions de corps $\mathbb{F}_{p^k}$ utilisées en cryptographie. Nous avons utilisé la représentation en base AMNS de [Pla05] pour représenter les éléments

de $\mathbb{F}_p$ et la multiplication par une méthode DFT dans les extensions $\mathbb{F}_{p^k}$. Nous avons mis en évidence que certaines représentations AMNS permettent d'améliorer efficacement les méthodes de multiplications par les racines de l'unité et donc d'optimiser l'approche DFT. La méthode de multiplication dans $\mathbb{F}_{p^k}$ que nous venons de présenter permet de diminuer le nombre de multiplications dans le corps fini, mais en contre partie augmente le nombre d'additions. Toutefois, le surcoût engendré par les additions ajoutées ne semble pas si important comparativement au gain en multiplications. La multiplication dans $\mathbb{F}_{p^k}$ obtenue en utilisant une base AMNS pour $\mathbb{F}_p$ nécessite moins de multiplications dans $\mathbb{F}_p$ que les méthodes recommandées dans [KM05, BE07]. Plus particulièrement, pour $k \geq 12$ la combinaison de la multiplication AMNS et l'approche DFT présentée ci-dessus permet de diminuer le nombre de multiplications dans $\mathbb{F}_p$ de 50%.

Chapitre IV

Cas spécifique des courbes de degré de plongement $k = 15$

Sommaire

IV.1 Présentation

L'arithmétique des couplages repose sur l'arithmétique des corps finis et l'efficacité de leurs calculs est souvent liée à celle de l'algorithme de Miller. Les derniers nés de la famille des couplages sont les couplages Ate et Twisted Ate. Ils sont issus de la dernière optimisation théorique des couplages. Cette optimisation consiste à réduire le nombre d'itérations durant l'exécution de l'algorithme de Miller. Dans [Ver08], Vercauteren prouve que, pour la plupart

des familles de courbes paramétrées, le nombre d'itérations de l'algorithme de Miller est $\frac{1}{\varphi(k)}\log_2(r)$, où r est l'ordre des sous-groupes de la courbe elliptique, k le degré de plongement de la courbe et φ la fonction d'Euler. L'**indicatrice d'Euler**, notée φ est la fonction de l'ensemble des entiers naturels non nuls dans lui même, qui a $n \in \mathbb{N}^\star$ associe le nombre de d'entiers positifs inférieurs ou égaux à n et premiers avec n.

Consécutivement, pour un niveau de sécurité fixé, et donc pour une taille en bits de r fixée, le calcul des couplages doit théoriquement être plus rapide pour les courbes elliptiques dont l'image du degré de plongement par la fonction d'Euler est un grand entier. C'est le cas par exemple pour les nombres premiers, mais pas pour les nombres de la forme 2^i3^j. En parallèle, nous cherchons à construire des courbes elliptiques pour lesquelles la valeur

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

est aussi petite que possible. Cette valeur ρ correspond à la perte d'efficacité de l'arithmétique du corps fini $\mathbb{F}_p$ par rapport à la taille de r. Plus ρ est proche de 1, et plus l'arithmétique du corps fini et celle de la courbe seront équilibrées. Par exemple pour un niveau de sécurité de 160 bits sur la courbe elliptique, si ρ vaut 2, cela implique qu'il faut travailler dans un corps fini $\mathbb{F}_p$ de taille 320 bits. Alors que si ρ vaut 1, nous effectuons les calculs dans un corps fini de taille 160 bits. L'arithmétique dans corps de 160 bits étant plus efficace que l'arithmétique dans un corps de 320 bits, construire des courbes elliptiques avec $\rho = 1$ est une méthode permettant l'optimisation des calculs des couplages.

Nous présentons la première étude du calcul des couplages pour des courbes de degré de plongement 15. Ce travail en cours de soumission [EGI09] a été mené en collaboration avec Sorina Ionica, étudiante en thèse au laboratoire PRISM, à l'Université de Versailles ; et Nicolas Guillermin, doctorant au sein de l'équipe IRMAR à l'Université de Rennes 1. Nous étudions l'utilisation des familles paramétrées de courbes elliptiques de degré 15 existantes. Tout d'abord, nous mettons en évidence que le nombre d'itérations de l'algorithme de Miller pour ces courbes est de $\frac{\log_2(r)}{8}$ pour le couplage Ate, et $\frac{\log_2(r)}{2}$ pour le Twisted Ate. Ce qui correspond au nombre minimum d'itérations que nous pouvons effectuer pour les courbes de degré de plongement $k = 15$. Ensuite, en utilisant un twist de degré 3, nous calculons les couplages en effectuant la majorité des opérations dans $\mathbb{F}_p$ et $\mathbb{F}_{p^5}$, et nous éliminons le calcul des dénominateurs durant l'algorithme de Miller. Enfin, nous présentons une amélioration de l'arithmétique sur le corps fini $\mathbb{F}_{p^5}$ afin de rendre les calculs de couplages plus efficaces.

Nos premiers résultats prouvent qu'en travaillant avec notre arithmétique sur $\mathbb{F}_{p^5}$ et $\mathbb{F}_{p^{15}}$, les calculs de couplages sur les courbes de degré $k = 15$ et de valeur $\rho = 1,5$ sont plus efficaces que sur les courbes de Barreto-Naehrig

pour des hauts niveaux de sécurité, c'est à dire pour des niveaux de sécurité de 192 et 256 bits.

L'organisation de ce chapitre est la suivante. La partie IV.2 présente la notion de couplage optimal à travers l'exemple de l'étude des courbes de degré de plongement 15. Nous décrivons dans la partie IV.3 notre multiplication basée sur l'interpolation pour les extensions de degré 5 d'un corps fini. Nous concluons nos travaux dans la partie IV.4 en présentant une évaluation globale du nombre d'opérations nécessaires pour calculer un couplage ; nous comparons les performances de nos résultats avec celles des courbes de Barreto-Naehrig qui sont considérées actuellement comme les courbes les plus performantes pour le calcul des couplages.

IV.2 Couplage optimal pour les courbes de degré de plongement $k = 15$

Nous proposons une optimisation du calcul des couplages pour la famille de courbes de degré de plongement $k = 15$ définies par la paramétrisation suivante :

$$\begin{aligned} p &= \frac{1}{3}\left(x^{12} - 2x^{11} + x^{10} + x^7 - 2x^6 + x^5 + x^2 + x + 1\right), \\ r &= x^8 - x^7 + x^5 - x^4 + x^3 - x + 1, \\ t &= x + 1. \end{aligned}$$

Pour x un entier congru à 1, 4 ou 7 modulo 9, cette paramétrisation détermine p la caractéristique du corps fini sur lequel est définie la courbe, r l'ordre des sous-groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_3$ et t la trace du Frobenius, sous la condition que p et r soient deux nombres premiers. Cette famille de courbes a été générée par Duan, Cui et Chan [DCC06] en utilisant la méthode de Brezing-Weng. Le degré de plongement d'une courbe de cette famille est 15, la valeur du j-invariant est 0 et celle de ρ est $\rho \approx 1,5$. L'équation d'une telle courbe elliptique est de la forme $Y^2 = X^3 + b$, avec $b \in \mathbb{F}_p$. Afin de fixer les idées sur l'efficacité de nos travaux, nous comparons l'efficacité du couplage sur les courbes avec $k = 15$ par rapport au couplage sur les courbes de Barreto-Naehrig de degré de plongement 12 et de j-invariant 0, dont la paramétrisation est :

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

Ces courbes sont prisées en cryptographie car leur ρ est approximativement un, et l'utilisation d'un twist de degré 6 permet d'exécuter la plupart des calculs dans $\mathbb{F}_p$ et $\mathbb{F}_{p^2}$.

IV.2.1 Twists de degré 3

Soit E une courbe elliptique de degré de plongement $k = 15$ de j-invariant 0 définie sur un corps $\mathbb{F}_p$. Nous supposons que son équation est $y^2 = x^3 + b$, avec $b \in \mathbb{F}_p$. La courbe $E(\mathbb{F}_{p^k})$ admet une courbe tordue par un twist de degré 3. En effet, quinze est divisible par 3, nous pouvons donc appliquer le Théorème I.3.21.
Soit $\widetilde{E}$ une courbe elliptique définie sur $\mathbb{F}_{p^{k/3}}$ d'équation $y^2 = x^3 + \frac{b}{\nu}$, avec $\nu \in \mathbb{F}_{p^{k/3}}$ qui ne soit pas un résidu cubique dans $\mathbb{F}_{p^{k/3}}$. Le twist envoyant un point de $\widetilde{E}(\mathbb{F}_{p^{k/3}})$ en un point de $E(\mathbb{F}_{p^k})$ est le morphisme

$$\begin{aligned} \Psi_3 : \widetilde{E}(\mathbb{F}_{p^{k/3}}) &\to E(\mathbb{F}_{p^k}) \\ (x, y) &\to (x\nu^{1/3}, y\nu^{1/2}) \end{aligned}$$

Nous pouvons donc choisir Q, un élément de $\mathbb{G}_2$, comme étant l'image d'un point Q' de r-torsion de $\widetilde{E}$: $Q = \Psi_3(Q')$, avec $Q' \in \widetilde{E}(\mathbb{F}_{p^{k/3}})$[Ver08]. Ainsi, les coordonnées de Q peuvent s'écrire $Q = (\nu^{1/3}x, \nu^{1/2}y)$, avec $x, y \in \mathbb{F}_{p^{k/3}}$.

Cette écriture des coordonnées de Q nous permettra par la suite d'effectuer la plupart des opérations pendant l'exécution de l'algorithme de Miller dans $\mathbb{F}_p$ et $\mathbb{F}_{p^5}$, au lieu de les faire dans $\mathbb{F}_{p^{15}}$.

IV.2.2 Théorie des couplages optimaux

Nous présentons les résultats des articles de Vercauteren [Ver08] et Hess [Hes08] introduisant la notion de couplage optimal, pour le cas des couplages twistés. Les couplages optimaux sont ceux pour lesquels le nombre d'itérations dans l'algorithme de Miller est réduit au minimum. Nous ne donnons pas les démonstrations des théorèmes et résultats.

Nous commençons par introduire les notations dont nous aurons besoin.

Soit s un entier, qui sera en fait une racine primitive k-ième de l'unité dans $\mathbb{F}_p$, et $h = \sum_{i=0}^{m} h_i x^i \in \mathbb{Z}[x]$ un polynôme à coefficients entiers. Nous choisissons h tel que $h(s) \equiv 0 \bmod r$. Soit $R \in E(\mathbb{F}_{p^k})$ et $f_{s,h,R}$ une fonction dont le diviseur est

$$(f_{s,h,R}) = \sum_{i=0}^{m} h_i((s^i R) - (P_\infty))$$

Nous aurons besoin de la norme $\| \|_1$ d'un polynôme qui est définie par :

$$\|h\|_1 = \sum_{i=0}^{m} |h_i|.$$

Le théorème suivant défini la notion de couplage optimal, en précisant les conditions d'existence d'un tel couplage. Il est issu de l'article [Hes08] de Hess et simplifié en utilisant les propriétés des courbes de degré de plongement $k = 15$.

Théorème IV.2.1. *[Hes08] Soit s une racine primitive k-ième de l'unité modulo r^2 et n un entier divisant 24.*

Alors, l'application

$$\begin{array}{rcl} a_{s,h}^{twist} : \mathbb{G}_1 \times \mathbb{G}_2 & \to & \mathbb{G}_3 \\ (P, Q) & \to & (f_{s,h,P}(Q))^{(p^k-1)/r}. \end{array}$$

définie un couplage bilinéaire qui est non dégénéré si et seulement si $h(s) \not\equiv 0 \ mod\ r^2$. Dans les cas où $a_{s,h}^{twist}$ est non dégénéré, cette application peut être liée au couplage de Tate par l'égalité

$$a_{s,h}^{twist}(P, Q) = e_T(P, Q)^{h(s)/r}.$$

Il est toujours possible de trouver un polynôme $h \in \mathbb{Z}[x]$ tel que $h(s) \equiv 0 \ mod\ r$, $deg(h) \leq \varphi(n) - 1$ et $\|h\|_1 = \mathcal{O}(r^{1/\varphi(n)})$, de manière à ce que le couplage $a_{s,h}^{twist}$ soit non dégénéré. La constante intervenant dans le $\mathcal{O}$ dépend uniquement du choix de n.

Le polynôme h est obtenu en réduisant un réseau construit à l'aide des paramètres de la courbe et des relations liant ces paramètres, c'est un plus court vecteur pour le réseau L, présenté dans l'article [Hes08]. Pour les couplages twistés, le réseau est donné par la matrice suivante :

$$L := \begin{pmatrix} r & 0 \\ p^{k/d} & 1 \end{pmatrix}$$

Les lignes de ce réseau correspondent au modulo annulant l'entier s. Il est obtenu en cherchant des entiers λ_1 et λ_2 les plus petits possibles tels que l'égalité $\lambda_1 + \lambda_2 p^{k/d} \equiv 0 \bmod(r)$ soit vraie. Cette égalité nous permettra alors de construire un couplage dont le nombre d'itérations sera $\max(\lambda_1, \lambda_2)$.

Le théorème suivant permet de comprendre comment s'effectue la démonstration du théorème de Hess.

Théorème IV.2.2. *Soit E une courbe elliptique admettant un twist de degré d, r un diviseur premier de $\#E(\mathbb{F}_p)$ et k son degré de plongement relativement à r. Pour tout couple d'entier λ_1 et λ_2 tels que l'égalité*

$$\lambda_1 + \lambda_2 p^{k/d} \equiv 0 \ mod(r) \tag{IV.2.1}$$

soit vraie, nous pouvons construire un couplage défini par la fonction rationnelle $f_{\lambda_1,P} \times f_{\lambda_2,P}^{p^{k/d}}$.

La preuve de ce théorème repose sur les propriétés des diviseurs que nous avons introduits dans la partie I.6.2. Nous pouvons aussi utiliser l'égalité $f_{ab,P} = f_{a,P}^b f_{b,aP}$, pour a et b deux entiers.

Démonstration. Soit P un point de la courbe elliptique, l'égalité IV.2.1 implique l'égalité des diviseurs suivante : $Div(f_{\lambda_1+\lambda_2 p^{k/d},P}) = Div(f_{\lambda_3 r,P})$, où λ_3 est l'entier tel que $\lambda_1 + \lambda_2 p^{k/d} = \lambda_3 r$.

Or

$$\begin{aligned} Div(f_{\lambda_3 r,P}) &= \lambda_3 r(P) - \lambda_3 r(P_\infty) \\ &= \lambda_3 \left(r(P) - r(P_\infty)\right) \\ &= \left(Div(f_{r,P})\right)^{\lambda_3} \end{aligned}$$

et

$$Div(f_{\lambda_1+\lambda_2 p^{k/d},P}) = Div(f_{\lambda_1,P}) + Div(f_{\lambda_2 p^{k/d},P}) + Div(\frac{l_{[\lambda_1]P,[\lambda_2 p^{k/d}]P}}{v_{[\lambda_1]P+[\lambda_2 p^{k/d}]P}}) \tag{IV.2.2}$$

Nous allons simplifier l'expression $Div(f_{\lambda_2 p^{k/d},P})$ en utilisant les résultats de la référence [HSV06].

Nous rappelons qu'ici $k = 15$, $d = 3$ et ainsi $k/d = 5$.

Nous réutilisons une réécriture des groupes $\mathbb{G}_1$ et G_2 introduites dans [HSV06] :

$$G_1 = E[r] \cap \mathrm{Ker}(\xi_3 \circ \pi_p^5 - [p^5])$$

et

$$G_2 = E[r] \cap \mathrm{Ker}(\xi_3 \circ \pi_p^5 - [1]),$$

où ξ_3 est un automorphisme de la courbe elliptique, correspondant à la multiplication par une racine troisième primitive de l'unité.

Avec cette nouvelle définition de $\mathbb{G}_1$, le point $[p^5]P$ est aussi le point $\xi_3 \circ \pi_p^5(P)$.

Nous pouvons alors appliquer le lemme 5 de la référence [HSV06] :

$$f_{\lambda_2,\xi_3(P)} \circ \xi_3 = f_{\lambda_2,P}.$$

Par définitions, les coefficients de la fonction $f_{\lambda_2,P}$ sont des éléments de $\mathbb{F}_p$ et donc :

$$f_{\lambda_2,\xi_3(P)} \circ \xi_3 \circ \pi_p^5 = f_{\lambda_2,P}^{p^5}.$$

Comme $Q \in \mathbb{G}_2$ vérifie que $\xi_3 \circ \pi_p^5(Q) = Q$, et que $P \in \mathbb{G}_1$ vérifie que $\pi_p^5(P) = P$, nous pouvons écrire que :

$$f_{\lambda_2,\xi_3\circ\pi_p^5(P)}(Q) = f_{\lambda_2,\xi_3(P)} \circ \xi_3 \circ \pi_p^5(Q) = f_{\lambda_2,P}(Q)^{p^5}.$$

Nous savons de plus que les fonctions rationnelles $f_{r,P}$ et $f_{p^{k/d},P}$ définissent des couplages, respectivement le couplage de Tate et le couplage Ate.

Nous déduisons de l'égalité IV.2.2 que la fonction $f_{\lambda_1,P} \times f_{\lambda_2,P}$ définit aussi un couplage. □

Ainsi, réussir à trouver des entiers λ_1 et λ_2 les plus petits possibles nous permet de construire un couplage dont le nombre d'itérations sera minimal. Dans la partie suivante, nous appliquons ces résultats aux courbes de degré de plongement $k = 15$.

IV.2.3 Couplage optimal pour $k = 15$

Par construction de la famille paramétrée de degré 15, nous savons que le nombre d'itérations durant l'algorithme de Miller pour le couplage Ate est $\log_2(r^{\frac{1}{8}})$. Pour le Twisted Ate, ce nombre est a priori $\frac{5}{8}\log_2 r$. Cette complexité est obtenue en appliquant la définition du Twisted Ate que nous avons présentée dans la partie I.9.3. En utilisant la notion de couplage optimal, nous réduisons le nombre d'itérations à $\frac{\log_2 r}{2}$. Cette borne est la plus petite que nous puissions espérer trouver. En effet le réseau L_{TA} que nous construisons est réduit à une matrice carré de taille 2.

$$L_{TA} := \begin{pmatrix} r & 0 \\ p^5 & 1 \end{pmatrix}$$

Nous appliquons le Théorème IV.2.1 en fixant le paramètre n égal à 3, et $s = p^5$; la valeur de s est fixée par le twist que nous utilisons. L'égalité $n = 3$ nous permet d'obtenir le polynôme suivant en utilisant l'algorithme de réduction des réseaux LLL [Pau98] :

$$h_{15}(x, v) = (x^3 - x^2 + 1)v - x^4 + x^3 - x + 1$$

Nous devons donc calculer la fonction $f_{s,h,P}$ dont le diviseur est :

$$\begin{aligned} Div(f_{s,h,P}) &= ((x^3-x^2+1)([s]P) - ((x^3-x^2+1)[s]P) - (x^3-x^2)(P_\infty)) \\ &\quad +((-x^4+x^3-x+1)(P) - ((-x^4+x^3-x+1)P) \\ &\quad -(-x^4+x^3-x)(P_\infty)) \\ &\quad +([(x^3-x^2+1)s]P) + ([-x^4+x^3-x+1]P) - 2(P_\infty) \\ &= Div(f_{x^3-x^2+1,[s]P}) + Div(f_{-x^4+x^3-x+1,P}) + Divv_{[(x^3-x^2+1)s]P} \end{aligned}$$

Cette égalité des diviseurs nous permet de construire le couplage optimal Twisted Ate pour la famille de courbes de degré 15 ; sur le même modèle que la démonstration du théorème **??**. Ce couplage est défini par la formule :

$$(f^{p^5}_{x^3-x^2+1,P}(Q) f_{-x^4+x^3-x+1,P}(Q))^{\frac{p^{15}-1}{r}}.$$

Nous avons implanter cet algorithme en Magma pour vérifier que nous obtenons bien un couplage bilinéaire et non dégénéré.

Il est intéressant de noter que $v_{[(x^3-x^2+1)s]P}(Q)$ l'équation de la droite verticale au point $[(x^3-x^2+1)s]P$ évaluée au point Q peut être ignorée car ce facteur disparaîtra au moment de l'exponentiation finale. En effet, nous verrons dans la partie IV.2.4 que $v_{[(x^3-x^2+1)s]P}(Q)$ appartient à $\mathbb{F}_{p^5}$, sous-corps propre de $\mathbb{F}_{p^k}$; ce terme sera neutralisé par l'exponentiation finale. Utilisant l'égalité de Miller introduite dans la partie I.8.2 nous avons donc besoin de calculer les fonctions $f_{x,P}(Q)$, $f_{x^2,P}(Q)$, $f_{x^3,P}(Q)$ et $f_{x^4,P}(Q)$ ainsi que les évaluations au point Q des équations de droites $l_{[x^3]P,[-x^2]P}$, $l_{[-x^4]P,[x^3]P}$, $l_{[-x^4+x^3]P,[-x]P}$, ceci pour une valeur entière de x fixée telle que les entiers $p(x)$ et $r(x)$ soient tous deux premiers. La complexité de l'évaluation du couplage Twisted Ate sur les courbes de degré de plongement $k = 15$ est donc en $\frac{\log_2(r)}{2}$ itérations de l'algorithme de Miller. Bien que le nombre d'itérations soit supérieur au nombre d'itérations pour le couplage Ate, le fait que nous calculons un Miller Lite peut avoir pour conséquence que la complexité du Twisted Ate soit malgré tout inférieure à celle du couplage Ate pour les courbes $k = 15$.

Pour avoir un ordre d'idée de l'efficacité du calcul des couplages Ate et Twisted Ate sur les courbes de degré de plongement $k = 15$, nous la comparons avec celle du calcul du couplage sur les courbes de Barreto-Naehrig. Le nombre d'itérations du calcul du Twisted Ate sur les courbes de Barreto-Naehrig est a priori de $\frac{\log_2 r}{2}$ itérations de l'algorithme de Miller.

L'application du Théorème IV.2.1 permet d'obtenir par exemple le polynôme h :

$$h_{12}(x,v) = (2x+6x^2)v + 1 + 2x.$$

Ce plus court vecteur ne permet pas de diminuer le nombre d'itérations pour l'algorithme de Miller. Le couplage Twisted Ate sur les courbes de Barreto-Naehrig se calcule bien en $\frac{\log_2 r}{2}$ itérations de l'algorithme de Miller.

Vercauteren propose dans [Ver08] un vecteur pour calculer le couplage Ate optimal sur les courbe de Barreto-Naehrig, ce vecteur réduit le nombre d'itérations pour ce couplage à $\frac{\log\ r}{4}$. nous résumons les différents nombres d'itérations de l'algorithme de Miller pour les couplages Ate et Twisted Ate pour les courbes de degré de plongement 15 et 12 dans le Tableau IV.1.

Degré plongement \ Couplage	Ate	Twisted Ate
15	$\frac{\log_2(r)}{8}$	$\frac{\log_2(r)}{2}$
12	$\frac{\log_2(r)}{4}$	$\frac{\log_2 r}{2}$

TABLE IV.1 – Nombre d'itérations de l'algorithme de Miller

IV.2.4 Élimination des dénominateurs

L'algorithme de Miller introduit dans la partie I.8.2 met à jour à chaque itération les valeurs de f_1 et f_2, avant de calculer le quotient des valeurs obtenues.

Le calcul de la valeur f_2 peut parfois être évité en tirant parti de l'exponentiation finale des couplages Ate et Twisted Ate, comme nous l'avons présenté dans la partie II.4.1. En effet, l'exponentiation à la puissance $\frac{p^{15}-1}{r}$ neutralise les facteurs du résultat intermédiaire appartenant à un sous-corps propre de $\mathbb{F}_{p^{15}}$.

Lorsque le twist est de degré d pair, l'élimination des dénominateurs est quasi automatique. L'ordonnée du point $\Psi_d(Q')$ est un élément de $\mathbb{F}_{p^{k/d}}$. De ce fait, le dénominateur $x_T - x_Q$ est aussi un élément de $\mathbb{F}_{p^{k/d}}$,sous-corps propre de $\mathbb{F}_{p^k}$, le point T représente le point courant dans l'algorithme de Miller (c'est un multiple de P). Le dénominateur étant un élément de $\mathbb{F}_{p^{k/d}}$, il sera donc éliminé lors de l'exponentiation finale.

Ce n'est pas aussi simple pour un twist de degré 3. En effet, par définition du twist $Q = \Psi_3(Q')$ s'écrit $Q = (\nu^{1/3}x, \nu^{1/2}y)$, où l'élément ν n'est pas un cube dans $\mathbb{F}_{p^5}$. La coordonnée de Q qui appartient à $\mathbb{F}_{p^{15}}$ est l'abscisse. Comme le dénominateur $x_T - x_Q$ est un élément de $\mathbb{F}_{p^{15}}$, il influence le résultat après l'exponentiation finale, car il ne sera pas neutralisé par cette dernière. Par contre, nous pouvons choisir $\nu \in \mathbb{F}_{p^5}$ de sorte que le polynôme $X^2 - \nu$ admette une racine dans $\mathbb{F}_{p^5}$. Ainsi, l'ordonnée $\nu^{1/2}y$ de Q restera un élément de $\mathbb{F}_{p^5}$.

La probabilité d'obtenir un élément $\nu \in \mathbb{F}_{p^5}$ qui soit un carré mais pas un cube est non nulle lorsque $p \equiv 1 \bmod(3)$. Un élément de $\mathbb{F}_{p^5}$ a une chance

sur deux d'être un carré, si $p \equiv 1 \bmod(3)$ deux éléments sur trois dans $\mathbb{F}_{p^5}$ ne sont pas des cubes. Au final, la probabilité qu'un élément pris au hasard dans $\mathbb{F}_{p^5}$ ne soit un carré mais pas un cube est de 1/3.

A priori, le dénominateur $x_T - x_Q$ est un élément de $\mathbb{F}_{p^{15}}$ lorsque nous utilisons un twist de degré 3, nous ne pouvons donc l'éliminer. Afin de pouvoir malgré tout procéder à l'élimination du dénominateur, nous utilisons la méthode présentée dans [LZZW08]. Elle consiste à ré-écrire l'équation de la droite v_1 en faisant intervenir l'équation de la courbe elliptique comme décrit ci après.

$$\begin{aligned} x_T - x_Q &= \frac{(x_T - x_Q)(x_T^2 + x_T x_Q + x_Q^2)}{(x_T^2 + x_T x_Q + x_Q^2)} \\ &= \frac{x_T^3 - x_Q^3}{x_T^2 + x_T x_Q + x_Q^2} \\ &= \frac{(y_T^2 - y_Q^2)}{(x_T^2 + x_T x_Q + x_Q^2)}. \end{aligned}$$

Nous avons décomposé le dénominateur en une fraction dont un des termes appartient à $\mathbb{F}_{p^5}$. Une analyse des termes intervenant dans la fraction nous permet d'éliminer le calcul des dénominateurs, et donc l'inversion finale. En effet, la parenthèse $(y_Q^2 - y_T^2)$ est un élément de $\mathbb{F}_{p^5}$ sous-corps propre de $\mathbb{F}_{p^{15}}$ et peut donc être omise lors du calcul du couplage, puisqu'elle sera neutralisée par l'exponentiation finale. L'entier $(p^5 - 1)$ est un facteur de $\frac{p^{15}-1}{r}$. Cette assertion se prouve de la même façon que la Propriété II.4.2, en écrivant $p^{15} - 1 = (p^5 - 1)(p^{10} + p^5 + 1)$. Nous savons que le groupe multiplicatif de $\mathbb{F}_{p^5}$ est d'ordre $(p^5 - 1)$, tout élément de $\mathbb{F}_{p5}$ élevé à la puissance $(p^5 - 1)$ est envoyé sur 1.

L'élimination des dénominateurs est donc réalisable pour les twists de degré 3, moyennant une multiplication supplémentaire par le terme $(x_T^2 + x_T x_Q + x_Q^2)$ à chaque itération de l'algorithme de Miller. Ce calcul ajoute un carré, cinq multiplications dans $\mathbb{F}_p$ et trois multiplications dans $\mathbb{F}_{p^5}$ par itérations, soit un surcoût de $S_p + 5M_p + 3M_{p^5}$. Nous avons comparé ce surcoût, au coût de la mise à jour de la fonction f_2 dans l'algorithme de Miller. La mise à jour de f_2 nécessite un carré et une multiplication dans $\mathbb{F}_{p^{15}}$, soit $S_{p^{15}} + M_{p^{15}}$. Il faut ajouter à ces opérations le coût d'une inversion dans $\mathbb{F}_{p^{15}}$. Il est admis qu'une inversion coûte entre 10 et 40 multiplications dans le corps où elle doit être calculée. En utilisant les complexités données par le Tableau IV.3, nous pouvons conclure que ce surcoût compense la mise à jour de f_1 et l'inversion finale. L'élimination des dénominateurs simplifie la complexité de l'algorithme de Miller.

IV.2.5 Décompte du nombre d'opérations

Les implémentations des couplages en coordonnées Jacobiennes sur les courbes de Weierstrass sont considérées comme les plus efficaces, en terme d'opérations sur le corps de base [CSB04, ALNR09]. Nous avons comparé les coûts du couplage dans notre cas en coordonnées projectives et Jacobiennes, nous avons obtenus de meilleurs résultats en coordonnées projectives, ces différents types de coordonnées ont été introduits dans la partie I.2.5. Nous utilisons les formules de calcul de couplage en coordonnées homogènes. Nous avons adapté à notre cas les résultats de l'article [CHNW09] qui présentent une amélioration du calcul des couplages en coordonnées projectives.

Lors de la i-ième étape de doublement de l'algorithme de Miller décrit par l'Algorithme 3, les équations sont :

$$\begin{aligned} T = [2i]P &\leftarrow 2\cdot[i]P \text{ (étape 4)} \\ f_1 = f_{2i,P} &\leftarrow f_{i,P}^2 \times l_1(Q) \times S_T(Q) \text{ (étape 5)} \end{aligned}$$

avec $f_1 = f_{i,P}$ avant le début de l'itération, $l_1(Q) = 2YZy_Q - 3X^2x_Q + Y^2 - 3cZ^2$ et $S_T(Q) = Z^2x_Q^2 + XZx_Q + X^2$. Nous calculons $(2i)P = (X_3, Y_3, Z_3)$ de la manière suivante :

$$\begin{aligned} X_3 &= 2XY(Y^2 - 9Z^2), \\ Y_3 &= (Y - Z)(Y + 3Z)^3 - 8Y^3Z, \\ Z_3 &= 2Y^3Z. \end{aligned}$$

Afin d'optimiser les calculs, nous effectuons les opérations dans l'ordre donné ci dessous.

$$A = Y^2, \quad B = Z^2, \quad C = (Y+Z)^2 - A - B, \quad E = X^2, \quad F = (X+Y)^2 - E - A,$$

$$X_3 = F\cdot(A - 9B), \qquad Z_3 = 4A\cdot C$$
$$Y_3 = (A - 3B + C)\cdot(A + 9B + 3C) - Z_3.$$

Enfin, nous mettons à jour la variable f_1 en calculant $l_1(Q)\cdot S_T(Q)$ dont l'équation est :

$$\begin{aligned} l_1(Q)S_T(Q) &= (2YZy_Q - 3X^2x_Q + Y^2 - 3cZ^2)(Z^2x_Q^2 + XZx_Q + X^2) \\ &= 2X^2YZy_Q + 2XYZ^2x_Qy_Q + 2YZ^3x_Q^2y_Q - 3X^4x_Q \\ &\quad -3X^3Zx_Q^2 - 3X^2Z^2x_Q^3 + (Y^2 - 3cZ^2)X^2 + \\ &\quad (Y^2 - 3cZ^2)XZx_Q + (Y^2 - 3cZ^2)Z^2x_Q^2 \end{aligned}$$

Nous avons besoin d'effectuer les opérations suivantes :

$$\begin{aligned} G &= B \cdot C \\ H &= B \cdot F \\ I &= E^2 \\ K &= \frac{(X+Z)^2 - E - B}{2} \end{aligned} \qquad \begin{aligned} L &= K^2 \\ M &= F^2 \\ N &= (A - 3cB) \cdot K \\ O &= E \cdot K \\ P &= (A - 3cB) \cdot B \end{aligned}$$

Ce qui nous permet de trouver la valeur de $l_1(Q)S_T(Q)$ en calculant :

$$\begin{aligned} l_1(Q)S_T(Q) &= Fy_Q + Hx_Qy_Q + Gx_Q^2y_Q - (3I - N)x_Q \\ &\quad -(3O - P)x_Q^2 - 3Lx_Q^3 + M - 3cL \end{aligned}$$

Le point Q est fixé pendant toute l'exécution de l'algorithme de Miller, nous pouvons donc supposer que les valeurs x_Q, x_Q^2, y_Q, x_Qy_Q, $x_Q^2y_Q$ dont nous avons besoin pour les calculs sont précalculées une fois pour toute. Ces valeurs sont obtenus après application d'un twist de degré 3 au point $Q' \in E(\mathbb{F}_{p^5})$.

Pour les courbes de degré de plongement $k = 15$, le twist introduit dans la partie IV.2.1 implique qu'une multiplication par ces valeurs précalculées s'effectue en cinq multiplications dans $\mathbb{F}_p$. En effet, x_Q, x_Q^2, y_Q, x_Qy_Q, $x_Q^2y_Q$ sont soit dans le corps $\mathbb{F}_{p^5}$, soit sont égaux à un élément de $\mathbb{F}_{p^5} \times \nu$. Le décompte des opérations pour l'étape du doublement dans l'algorithme de Miller donne le résultat
$9S_p + 38M_p + S_{p^{15}} + M_{p^{15}}$ pour le couplage Twisted Ate. Dans le cas du couplage Ate, les variables $A, B, C, ...L$ sont des éléments de $\mathbb{F}_{p^5}$, la complexité pour ce couplage est donc plus importante : $30M_p + 9S_{p^5} + 8M_{p^5} + S_{p^{15}} + M_{p^{15}}$.

Le décompte des opérations pour les courbes de Barreto-Naehrig de degré de plongement $k = 12$ est donné par le coût du calcul de la fonction l_1, pour le Twisted Ate il est de $5S_p + 15M_p + S_{p^{12}} + M_{p^{12}}$, et pour le couplage Ate de $4M_p + 5S_{p^2} + 3M_{p^2} + S_{p^{12}} + M_{p^{12}}$. Ces complexités sont données par le Tableau IV.2.

IV.2.6 Première comparaison

Nous comparons la complexité du calcul des couplages Ate et Twisted Ate pour les courbes de degré de plongement 12 et 15. Nous utilisons les multiplications de Karatsuba et Toom Cook pour l'arithmétique de $\mathbb{F}_{p^k}$. Le coût d'une multiplication dans chacune des extensions de corps utilisées est donnée dans le Tableau IV.3. Le Tableau IV.4 donne les tailles de r et p pour atteindre un niveau de sécurité fixé pour les courbes de degré de plongement

Degré plongement \ Couplage	Ate
15	$30M_p + 9S_{p^5} + 8M_{p^5} + S_{p^{15}} + M_{p^{15}}$
12	$4M_p + 5S_{p^2} + 3M_{p^2} + S_{p^{12}} + M_{p^{12}}$
Degré plongement \ Couplage	**Twisted Ate**
15	$9S_p + 38M_p + S_{p^{15}} + M_{p^{15}}$
12	$5S_p + 15M_p + S_{p^{12}} + M_{p^{12}}$

TABLE IV.2 – Complexité de l'algorithme de Miller

M_{p^2} [KM05]	M_{p^3} [KM05]	M_{p^5}
$3M_p + 4A_p$	$5M_p + 20A_p$	$13M_p + 58A_p$

M_{p^6} [KM05]	$M_{p^{12}}$ [KM05]	$M_{p^{15}}$
$15M_p + 72A_p$	$45M_p + 180A_p$	$65M_p + 390A_p$

TABLE IV.3 – Évaluation de performance : coût des multiplications dans les extensions de corps

12 et 15.

Il est intéressant de noter dans le Tableau IV.4 que pour un niveau de sécurité fixé, la taille des corps n'est pas la même pour les deux familles de courbes comparées. En effet, la sécurité repose sur les tailles de r et de p^k. Pour les hauts niveaux de sécurité 192 et 256 bits, le Tableau IV.4 nous permet de voir que le corps $\mathbb{F}_p$ pour $k = 15$ est à chaque fois de taille plus petite que pour $k = 12$. Ceci signifie que pour ces niveaux de sécurité, une opération de base dans le corps $\mathbb{F}_p$ pour $k = 15$ sera plus rapide à exécuter que la même opération pour les courbes de Barreto-Naehrig. Inversement, pour les niveaux de sécurité de 80 et 128 bits, la taille des corps est plus petite pour les courbes $k = 12$. Par conséquent, pour les bas niveaux de sécurité, une opération de base sur $\mathbb{F}_p$ sera plus coûteuse pour les courbes avec $k = 15$ que pour les courbes avec $k = 12$.

Nous n'avons pas tenu compte dans nos comparaisons de la réduction polynomiale des résultats dans $\mathbb{F}_{p^k}$. Nous sous-estimons ainsi la complexité des calculs pour les courbes de Barreto-Naehrig. En effet, $\mathbb{F}_{p^{12}}$ est construit comme une extension de $\mathbb{F}_{p^2}$ qui lui même est construit par la tour d'extension $\mathbb{F}_p/(X^2 + \xi)$, où le polynôme $(X^2 + \xi)$ est irréductible dans $\mathbb{F}_p$. Les polynômes utilisés pour construire $\mathbb{F}_{p^{12}}$ à l'aide d'une tour d'extensions ont des termes constants δ dans $\mathbb{F}_{p^2}$. Suivant le choix des valeurs δ qui ne sont ni des carrés, ni des cubes, le coût d'une multiplication par δ varie et influence donc la complexité globale. Pour construire $\mathbb{F}_{p^{15}}$, nous construisons une tour

Niveau de sécurité en bit	Taille des groupes en bits		Taille des groupes $k = 15$		Taille des groupes $k = 12$	
	r	p^k	r	p	r	p
80	160	1024	160	240	160	160
128	256	3072	256	384	256	256
192	384	7680	384	576	640	640
256	512	15360	682	1024	1280	1280

TABLE IV.4 – Évaluation de sécurité : courbes de degré $k = 15$ contre les courbes de Barreto-Naehrig

de deux étages. Nous construisons d'abord $\mathbb{F}_{p^5}$ à l'aide d'un polynôme irréductible de degré 5 de $\mathbb{F}_p[X]$; puis, avec un polynôme irréductible de degré 3 sur $\mathbb{F}_{p^5}[X]$ nous construisons $\mathbb{F}_{p^{15}}$. Nous avons une marge de manoeuvre plus grande pour choisir les constantes des polynômes de réductions comme étant de petits entiers.

La comparaison des complexités pour les couplages Ate et Twisted Ate est donnée dans le Tableau IV.5.

Nous avons choisi de comparer l'efficacité du couplage entre les courbes de degré de plongement $k = 15$ et les courbes de Barreto-Naehrig, car les courbes Barreto-Naehrig sont considérées comme les plus adaptées au calcul des couplages. La combinaison d'un twist de degré 6 et de $\rho = 1$ permet d'avoir une arithmétique très efficace pour le calcul des couplages.

	couplage Ate	
Niveau de sécurité en bits	$k = 15$	$k = 12$
80	$6710M_p$	$4647M_p$
128	$10736M_p$	$7119M_p$
192	$16104M_p$	$17007M_p$
256	$28601M_p$	$33486M_p$
	couplage Twisted Ate	
Niveau de sécurité en bits	$k = 15$	$k = 12$
80	$12680M_p$	$7800M_p$
128	$20288M_p$	$12480M_p$
192	$30432M_p$	$31200M_p$
256	$54048M_p$	$62400M_p$

TABLE IV.5 – Comparaison du coût des couplages pour $k = 15$ et $k = 12$

Il est évident d'après le Tableau IV.5 que pour les niveaux de sécurité de 80 et 128 bits les courbes de Barreto-Naehrig sont les plus efficaces, elles nécessitent moins d'opérations dans $\mathbb{F}_p$ que les courbes avec $k = 15$, pour des corps de tailles plus petites. Par contre, pour les niveaux de sécurité

de 192 et 256 bits, la famille paramétrée de courbes avec $k = 15$ nécessite moins d'opérations que les courbes $k = 12$ durant l'exécution de l'algorithme de Miller. Comme de plus, les corps pour $k = 15$ sont plus petits pour ces niveaux de sécurité, le couplage est plus efficace à calculer sur les courbes $k = 15$ que sur les courbes $k = 12$. Nous pouvons remarquer que ces résultats sont obtenus sans amélioration de l'arithmétique de $\mathbb{F}_{p^5}$. Pour décrire cette arithmétique, nous avons utilisé les multiplications de Karatsuba et Toom Cook, qui ne sont pas adaptées à des polynômes de degré 5. A contrario, pour l'arithmétique des courbes de Barreto-Naehrig, nous avons utilisé l'arithmétique la plus efficace possible consistant à tirer partie des degrés d'extension 2 et 3 en utilisant les multiplications de Karatsuba et Toom Cook. Il semble difficile a priori d'améliorer l'arithmétique de l'extension $\mathbb{F}_{p^{12}}$.

Nous proposons dans la partie suivante une amélioration de l'arithmétique dans $\mathbb{F}_{p^5}$, en introduisant une multiplication par interpolation construite sur mesure pour des éléments d'une extension de degré 5.

IV.3 Arithmétique pour une extension de corps de degré 5

Nous nous assurons dans un premier temps qu'il est toujours possible de construire l'extension de degré 5 à l'aide d'un polynôme irréductible de la forme $X^5 - \beta$, avec $\beta \in \mathbb{F}_p$. Ceci est le cas pour tous les nombres premiers p tels que $p \equiv 1 \bmod(5)$ [LN94, Theorem 3.75].

Théorème IV.3.1. *Soit $\mathbb{F}_{p^5}$ une extension de corps fini, et β un élément de $\mathbb{F}_p$ qui ne soit ni un cube ni une puissance cinquième d'un élément de $\mathbb{F}_p$. Alors le polynôme $X^5 - \beta$ est irréductible dans $\mathbb{F}_p$.*

Nous pouvons donc considérer que l'extension de degré 5 est construite à l'aide d'un polynôme $X^5 - \beta$, avec $\beta \in \mathbb{F}_p$ que nous choisirons autant que possible comme étant une petite constante ou une puissance de 2. La probabilité qu'un élément de $\mathbb{F}_p$ ne soit ni un cube ni une puissance cinquième dans $\mathbb{F}_p$ est de un quinzième, elle est suffisamment grande pour que notre marge de manoeuvre dans le choix de β soit grande. Nous présentons une multiplication par interpolation sur $\mathbb{F}_{p^5}$. L'inconvénient des multiplications par interpolation est qu'elles ont tendance à augmenter le nombre d'additions dans $\mathbb{F}_p$. Nous adaptons l'interpolation de Newton pour une extension de degré 5, et décrivons une multiplication pour laquelle le nombre d'additions ajoutées par l'interpolation est compensé par le gain en nombre de multiplications. Nous comparons notre méthode aux méthodes de Karatsuba et Toom Cook. Nos résultats sont concluants, nous réduisons d'au moins 31% le nombre de multiplications dans $\mathbb{F}_p$, pour le calcul des couplages Ate et Twisted Ate.

Nous avons présenté dans la partie II.3 les méthodes de multiplication par interpolation et l'interpolation de Newton. (Rappelons que les multiplications de Karatsuba et Toom Cook présentées dans la partie II.3 sont aussi des multiplications par interpolation.) Nous adaptons la multiplication par interpolation à une extension de degré 5. Pour celle-ci l'approche de la FFT pour la multiplication n'est pas utilisable de manière efficace. En effet, la FFT implique des multiplications par des racines de l'unité dans $\mathbb{F}_p$. N'ayant aucune maîtrise sur la caractéristique p du corps de base, les racines cinquièmes de l'unité n'ont pas à coup sûr une représentation binaire qui soit creuse, et ce même après recodage. Une multiplication par de telles racines est donc coûteuse. Un degré d'extension 5 est trop petit pour que la combinaison de la représentation AMNS et de la multiplication par DFT présentée au Chapitre III soit applicable. Nous avons vu dans le Chapitre III que notre multiplication en représentation AMNS est plus efficace que les multiplications de Karatsuba et Toom Cook pour des degrés d'extension supérieur à 12. Nous décrivons donc une multiplication générale.

Nous voulons trouver le résultat du produit de $A(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + a_4X^4$ et $B(X) = b_0 + b_1X + b_2X^2 + b_3X^3 + b_4X^4$ deux polynômes de degré 4 à coefficients dans $\mathbb{F}_p$. Nous devons choisir neuf valeurs α_i dans $\mathbb{F}_p$ pour calculer l'interpolation de Newton, dont la complexité présentée dans la partie II.3.3 dépend des étapes suivantes :

1. l'évaluation en les valeurs d'interpolation α_i des polynômes $A(X)$ et $B(X)$, pour $i = 0, 1, 2, \ldots, 8$;
2. les neuf multiplications dans $\mathbb{F}_p$ ($A(\alpha_i) \times B(\alpha_i)$) ;
3. le calcul des c'_i dont les formules sont données dans la partie II.3.3, pour $i = 0, 1, 2, \ldots, 8$;
4. la construction du polynôme de degré 8 $C(X) = A(X) \times B(X)$, via le schéma de Horner.

IV.3.1 Simplification de l'interpolation dans $\mathbb{F}_{p^5}$

Nous travaillons avec des polynômes de degré 4, les $2k - 1 = 9$ valeurs d'interpolations choisies sont :

$$\alpha_0 = 0, \alpha_1 = 1, \alpha_2 = -1, \alpha_3 = 2, \alpha_4 = -2, \alpha_5 = 4, \alpha_6 = -4, \alpha_7 = 3, \alpha_8 = \infty.$$

Ces valeurs ont été choisies afin de minimiser le nombre d'additions et de divisions par les différences des α_i lors des calculs de l'interpolation.

Complexité de l'évaluation en les α_i de $A(X)$ et $B(X)$

Nous devons tout d'abord évaluer les polynômes $A(X)$ et $B(X)$ en les α_i. Pour les valeurs d'interpolation choisies ces évaluations sont uniquement composées de décalages et d'additions dans $\mathbb{F}_p$. En effet, multiplier par une puissance de 2 est une opération qui se réduit à des décalages en base binaire. Les évaluations en les valeurs α_i qui sont des puissances de 2 se fait donc en deux étapes. Nous calculons d'abord les produits $a_i \times (2^j)^i$ qui sont des décalages, puis les additions $\sum_{i=0}^{k-1} a_i (2^j)^i$. Nous calculons les évaluations suivant le schéma de la FFT, en séparant les indices pairs et impairs comme nous l'illustrons par les formules suivantes.

$$\begin{cases} sp1 = a_0 + a_2 + a_4 \\ si1 = a_1 + a_3 \\ A(1) = sp1 + si1 \\ A(-1) = sp1 - si1 \end{cases} \qquad \begin{cases} sp2 = a_0 + 2^2 \times a_2 + 2^4 \times a_4 \\ si2 = 2 \times a_1 + 2^3 \times a_3 \\ A(2) = sp2 + si2 \\ A(-2) = sp2 - si2 \end{cases}$$

$$\begin{cases} sp4 = a_0 + 2^4 \times a_2 + 2^8 \times a_4 \\ si4 = 2^2 \times a_1 + 2^6 \times a_3 \\ A(4) = sp4 + si4 \\ A(-4) = sp4 - si4 \end{cases}$$

L'évaluation en la valeur $\alpha_7 = 3$ consiste elle aussi uniquement en des décalages et des additions. Il suffit pour cela de remarquer que les puissances de 3 peuvent se décomposer en des additions de puissances de 2 comme suit : $3 = 2 + 1$, $3^2 = 2^3 + 1$, $3^3 = 2^5 - 2^2 - 1$ et $3^4 = 2^6 + 2^4 + 1$.

L'évaluation en la valeur 3 s'effectue en calculant :

$$A(3) = sp1 + a_1 + a_3 + 2(a_1 + 2^2 a_2 + 2(2^3 - 1)a_3 + 2^3(2^2 + 1)a_4).$$

La complexité de cette étape d'évaluation est de $23A_p$ pour chaque polynôme, soit un total de $46A_p$ pour les polynômes A et B.

Une fois obtenues les valeurs de A et B en les α_i, nous devons effectuer neuf multiplications dans $\mathbb{F}_p$ pour obtenir les valeurs que prend C le produit de A et B en ces points. La complexité de la première et de la seconde étape de la multiplication par interpolation est donc de $46A_p + 9M_p$.

Complexité du calcul des coefficients c'_j

Afin de déterminer les coefficients c'_j pour l'interpolation de Newton, nous devons effectuer des divisions exactes par les différences $\alpha_i - \alpha_j$, pour $j < i$. Nous appelons une division exacte, une division dont le reste est zéro. Parmi toutes les différences des valeurs α_i choisies, onze ne sont pas des puissances

de 2. Ces différences sont présentées dans le Tableau IV.6. Dans une base binaire, les divisions (exactes ou non) par des puissances de 2 sont constituées de décalages vers la droite de la représentation binaire du dividende. Elles sont donc peu coûteuses. Les divisions par 3, 5 et 7 sont un peu plus complexes à évaluer. Une méthode possible pour les calculer serait de trouver l'inverse des valeurs 3, 5 et 7 modulo p, et de transformer les divisions par ces valeurs en un produit par leur inverse modulo p. L'inconvénient de cette méthode est que comme 3, 5 et 7 sont de petits entiers, leurs inverses, modulo p un grand nombre premier, seront de grands entiers. Une multiplication par de grands entiers est coûteuse. Afin d'améliorer l'efficacité de l'interpolation, nous présentons une division alternative par ces valeurs qui s'effectue en une complexité équivalente à celle d'une soustraction.

$\alpha_3 - \alpha_2 = 3$	$\alpha_4 - \alpha_1 = -3$	$\alpha_5 - \alpha_1 = 3$	$\alpha_5 - \alpha_2 = 5$
$\alpha_5 - \alpha_4 = 6$	$\alpha_6 - \alpha_1 = -5$	$\alpha_6 - \alpha_2 = -3$	$\alpha_6 - \alpha_3 = -6$
$\alpha_7 - \alpha_0 = 3$	$\alpha_7 - \alpha_4 = 5$	$\alpha_7 - \alpha_6 = 7$	

TABLE IV.6 – Les différences problématiques

Nous voulons diviser δ, un multiple de 3, par 3 ; nous savons donc que δ s'écrit $\delta = 3 \times \sigma$, avec σ un entier que nous voulons retrouver. Cette égalité peut se ré-écrire sous la forme $\sigma = \delta - 2 \times \sigma$. En utilisant la décomposition binaire de $\delta = \sum_i \delta_i 2^i$, de $\sigma = \sum_i \sigma_i 2^i$ et cette nouvelle égalité, nous pouvons déterminer σ bit après bit. En effet, $\sigma = \delta - 2 \times \sigma$ nous permet de trouver le premier bit de σ, $\sigma_0 = \delta_0$. Nous pouvons alors trouver le deuxième bit de σ en effectuant la soustraction binaire $\delta_1 - \sigma_0 = \sigma_1$. Nous trouvons ainsi σ_1 ; et nous continuons sur le principe donné dans la Figure IV.1.

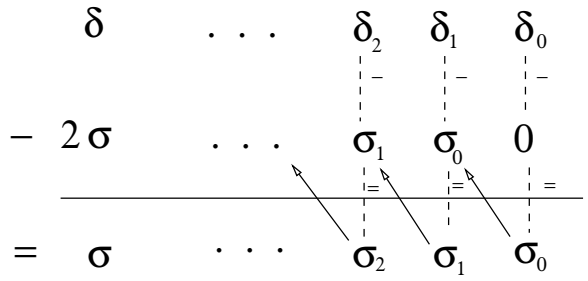


FIGURE IV.1 – Division exacte par 3 en une soustraction

Nous pouvons donc effectuer la division par 3 en une opération équivalente à une soustraction dans $\mathbb{F}_p$. Le même schéma peut s'appliquer à 5. Pour

faire la division exacte de $\chi = 5 \times \kappa$ par 5 nous retrouvons κ grâce à l'égalité $\kappa = \chi - 4 \times \kappa$. Ainsi une division exacte par 5 a une complexité équivalent à celle d'une demi soustraction, ou d'une soustraction en base 4.

La division exacte par 7 s'effectue elle aussi en une soustraction en base 8. Nous voulons diviser par 7 $\mu = 7 \times \nu$, cette égalité se transforme en l'égalité $\nu = 8 \times \nu - \mu$, et la division exacte par 7 s'effectue en une soustraction en base 8.

Nous considérons que la complexité d'une soustraction est celle d'une addition. Ainsi, la complexité de ces onze divisions exactes par 3, 5 et 7 est donc de $11A_p$. Les implémentations de cette méthode sont discutées dans la partie IV.3.2.

La complexité du calcul des c'_i comporte aussi le coût des soustractions faites, elles sont au nombre de 28.

La complexité totale du calcul des c'_i est donc de $39A_p$.

Complexité de la phase finale de l'interpolation polynomiale

Nous utilisons le schéma de Horner pour trouver l'expression du produit $C = A \times B$. Ce schéma consiste à déterminer les coefficients de $C(X)$ à l'aide de la formule suivante.

$$C(X) = ((((c'_8(X - \alpha_7) + c'_7)(X - \alpha_6) + c'_6)(X - \alpha_5) + c'5) \ldots + c'_1)(X - \alpha_1) + c'0$$

Nous commençons à calculer cette représentation en partant de la parenthèse intérieure $(c'_8(X - \alpha_7) + c'_7)$, puis nous passons à la parenthèse qui l'englobe, puis à la suivante, jusqu'à ce que nous arrivons au coefficient c'_0. La complexité du schéma de Horner est donc celle du produit de la i-ième parenthèse par la valeur d'interpolation α_{7-i} et de quelques additions. Les valeurs d'interpolations que nous avons choisie sont telles que les produits par les α_{7-i} consistent uniquement en des décalages. La complexité de cette étape est donc composée uniquement des additions à exécuter, il y en a 29.

La complexité de l'interpolation polynomiale est donc de $29A_p$.

IV.3.2 Premiers résultats et réflexion sur l'implémentation

Nous avons décrit la multiplication par interpolation pour une extension de degré 5 d'un corps fini. Dans le Tableau IV.7, nous donnons les complexités des multiplications en appliquant différentes méthodes, les classiques Karatsuba et Toom Cook dans la ligne KTC, la multiplication par interpolation dans la ligne interpolation et la troisième ligne (Mix) est obtenue en combinant ces deux méthodes. Dans la ligne Mix, pour $k = 15$, nous utilisons

la multiplication par interpolation pour le corps intermédiaire $\mathbb{F}_{p^5}$ puis la méthode de Toom Cook pour l'extension $\mathbb{F}_{p^{15}}$.

Méthode \ Extension	M_{p^2}	M_{p^5}
KTC	$3M_p + 4A_p$	$13M_p + 60A_p$
Interpolation	$--$	$9M_p + 103A_p$
Mix	$--$	$--$
Méthode \ Extension	$M_{p^{12}}$	$M_{p^{15}}$
KTC	$45M_p + 180A_p$	$65M_p + 390A_p$
Interpolation	$23M_p + 2070A_p$	$29M_p + 2136A_p$
Mix	$45M_p + 180A_p$	$45M_p + 615A_p$

TABLE IV.7 – Complexité des différentes méthodes de multiplications

Utilisant ces résultats, notre multiplication par interpolation sur $\mathbb{F}_{p^5}$ est plus intéressante qu'une multiplication par Karatsuba et Toom Cook. Nous sauvons quatre multiplications dans $\mathbb{F}_p$ pour un surcoût de 43 additions. Le surcoût engendré par les additions n'est pas plus important que le coût de quatre multiplications dans $\mathbb{F}_p$. En effet, admettons que la multiplication dans $\mathbb{F}_p$ s'effectue en utilisant la méthode de Karatsuba, la complexité de cette multiplication est

$$5N^{\log_2(3)}A_w + N^{\log_2(3)}M_w,$$

où N est le nombre de mots machines formant les entiers manipulés, et où A_w et M_w représentent une addition et une multiplication d'un mot machine. La complexité d'une addition est de NA_w. Considérons qu'une multiplication de mot machine s'effectue en m additions machines $M_w = mA_w$. Nous allons essayer de déterminer le nombre m à partir duquel quatre multiplications dans $\mathbb{F}_p$ ont une complexité machine plus importante que 43 additions dans $\mathbb{F}_p$.

Les quatre multiplications dans $\mathbb{F}_p$ ont une complexité de

$$4(5N^{1,6}(1+m))A_w.$$

Les 43 additions dans $\mathbb{F}_p$ ont une complexité de $43NA_w$.

Nous cherchons m telle que l'inégalité suivante soit vraie :

$$4(5N^{\log_2(3)}(1+m))A_w > 43NA_w$$

Nous simplifions cette inégalité en posant $N = 1$, ce qui est loin de la réalité car les nombres que nous manipulons sont au moins de taille 16 mots

machine. Avec cette simplification, nous obtenons que m doit être supérieur à $1,15$ pour que quatre multiplications dans $\mathbb{F}_p$ aient une complexité plus importantes que 43 additions dans $\mathbb{F}_p$, cette hypothèse est très réaliste considérant les architectures actuelles.

Ainsi, les additions en surplus sont largement compensées par les gains en nombre de multiplications. Notre méthode est une réelle optimisation des calculs dans une extension de degré 5.

Nous avons programmé notre division exacte par 3 sur un FPGA Stratix (modèle speed grade 3), il s'agit d'un FPGA classique utilisé en arithmétique. Les résultats de notre implémentation sont très positifs, et confirment que nous pouvons effectuer une division par 3 en le temps d'une soustraction. Nous avons calculé la division exacte d'un entier de 240 bits en 13,9 nanosecondes à l'aide de notre méthode. Ce temps de latence permet d'exécuter notre division en un cycle d'horloge. En comparaison, sur le même FPGA, nous avons calculé une addition sur des entiers de même taille en 14,2 nanosecondes. Ces résultats sont concluants, d'autant plus que l'architecture du FPGA contient par défaut une optimisation pour la propagation des retenues dans une addition.

Nous avions calculé la complexité de notre multiplication en faisant l'hypothèse qu'une division exacte par de petits entiers pouvait être effectuée en une complexité équivalente à celle d'une soustraction. L'hypothèse formulée dans la partie IV.3.1 est donc vérifiée ; les complexités que nous avançons sont donc validées.

Par contre, utiliser une multiplication par interpolation pour des extensions de degré 12 et 15 ne semble pas intéressant. Le nombre d'additions ajoutées n'est pas compensé par les multiplications sauvées. La multiplication par interpolation est très intéressante pour une extension de degré 5, car nous pouvons choisir les valeurs d'interpolation de manière à limiter raisonnablement l'augmentation du nombre d'additions par rapport aux multiplications que nous sauvons. Le Tableau IV.8 présente nos comparaisons finales des complexités du calcul des couplages pour différents niveaux de sécurité.

Nous avons utilisé notre arithmétique pour une extension de degré 5. Nous comparons les résultats obtenus pour la complexité du calcul du couplage sur les courbes de degré de plongement 15 par rapport aux courbes de Barreto-Naehrig. En ne tenant compte que du nombre de multiplications, notre arithmétique nous permet de réaliser un gain de 28% pour le calcul du couplage Ate par rapport à une arithmétique classique, et de 22% pour le couplage Twisted Ate pour les courbes $k = 15$. Ce qui représente une réelle amélioration du calcul du couplage sur ces courbes.

Le résultat de nos comparaisons est que la complexité du couplage Ate pour $k = 15$ nécessite 46% de multiplications en moins pour le niveau de sé-

	Couplage Ate	
AES sécurité	k=15	k=12
80	$4830M_p + 52205A_p$	$4300M_p + 13600A_p$
128	$7728M_p + 83528A_p$	$6900M_p + 21760A_p$
192	$11592M_p + 125292A_p$	$17007M_p + 54400A_p$
256	$20587M_p + 172128A_p$	$33486M_p + 108800A_p$
	Couplage Twisted Ate	
AES sécurité	k=15	k=12
80	$9880M_p + 86100A_p$	$7800M_p + 25200A_p$
128	$15808M_p + 137760A_p$	$12480M_p + 40320A_p$
192	$23712M_p + 206640A_p$	$31200M_p + 100800A_p$
256	$42113M_p + 284480A_p$	$62400M_p + 201600A_p$

TABLE IV.8 – Comparaison du coût des couplages pour les courbes $k = 15$ par rapport aux courbes $k = 12$

curité 192 bits, et 62% pour le niveau de sécurité de 256 bits par rapport au couplage Ate sur les courbes de Barreto-Naehrig. Pour le couplage Twisted Ate, nous avons un gain en multiplications de 31% pour 192 bits de sécurité et 48% pour le niveau de sécurité de 256 bits. Nous avions remarqué de plus que pour ces niveaux de sécurité la taille des corps est plus petite pour les courbes $k = 15$ que pour les courbes $k = 12$. Ainsi, nous effectuons moins de multiplications dans des corps qui de plus sont plus petits. Pour les hauts niveaux de sécurité, le couplage sur les courbes $k = 15$ est plus efficace que pour les courbes de Barreto-Naehrig, avec des corps finis de taille plus petite.

Pour les bas niveaux de sécurité, notre arithmétique permet de diminuer le coût des couplages Ate et Twisted Ate sur les courbes de degré de plongement 15 et donc de réduire l'écart par rapport aux courbes de degré de plongement 12. Mais, leur complexité reste plus importante.

Nous avons évalué le coût de l'exponentiation finale pour les couplages en utilisant la technique proposée dans [SBC+09]. Nos calculs montrent que cette opération est d'une complexité équivalente pour notre approche pour les courbes $k = 15$, et les courbes de Barreto-Naehrig $k = 12$ pour les hauts niveaux de sécurité. Par exemple, pour une sécurité de 256 bits, l'exponentiation requiert environ 960 carrés dans $\mathbb{F}_{p^{15}}$ dans le cadre de notre approche pour les courbes $k = 15$ et 940 carrés dans $\mathbb{F}_{p^{12}}$ sur les courbes de Barreto-Naehrig avec $k = 12$.

Pour les bas niveaux de sécurité, l'exponentiation sera plus coûteuse pour les courbes $k = 15$, car sa complexité dépend de la valeur $\rho \approx 1,5$. Il serait intéressant de trouver des courbes de degré de plongement $k = 15$ ayant une valeur ρ plus proche de 1 pour pouvoir utiliser les courbes $k = 15$ à des bas niveaux de sécurité.

IV.4 Conclusion

Nous avons présenté un calcul de couplage efficace pour les courbes de degré de plongement 15. Nous avons montré qu'il existe des familles de courbes de degré de plongement 15 et de j-invariant 0 présentant de nombreux avantages pour le calcul des couplages. Nous avons tout d'abord calculé que les calculs des couplages sur ces courbes se font en $\frac{\log_2(r)}{8}$ itérations de Miller pour le couplage Ate et $\frac{\log_2(r)}{2}$ pour le Twisted Ate. Nous avons ensuite utilisé un twist de degré 3 afin d'effectuer la plupart des opérations dans $\mathbb{F}_p$ et $\mathbb{F}_{p^5}$; et de plus procéder à l'élimination des dénominateurs. Enfin, nous avons amélioré l'arithmétique dans $\mathbb{F}_{p^5}$ en introduisant une multiplication par interpolation pour améliorer les complexités des calculs.

Nos résultats sont concluants, nous sauvons au minimum 22% d'opérations dans $\mathbb{F}_p$ pour calculer le couplage Ate avec notre arithmétique, par rapport à une arithmétique classique. En comparant les complexités des calculs des couplages pour les courbes avec degré de plongement $k = 15$ par rapport aux courbes de Barreto Naehrig, pour les hauts niveaux de sécurité nous sauvons au minimum 31% de multiplications dans $\mathbb{F}_p$.

Chapitre V

Attaques par canaux auxiliaires

Sommaire

V.1 Présentation

Les attaques par canaux cachés, ou par canaux auxiliaires, sont des attaques dont l'objectif est de casser un procédé de chiffrement non pas en l'attaquant de manière frontale, mais en exploitant des fuites d'informations occasionnées par la mise en oeuvre du procédé de chiffrement : par exemple, la consommation électrique d'une carte à puce peut permettre de retrouver des informations sur le secret utilisé. Nous considérons que casser un protocole cryptographique revient à obtenir le secret servant pour le chiffrement ou le déchiffrement afin de pouvoir décrypter des messages qui ne nous seraient pas destinés.

Nous présentons les attaques par canaux auxiliaires appliquées à des cryptosystèmes à clé publique. La notion de cryptographie à clé publique est introduite dans la partie V.2.1, nous l'illustrons à travers l'exemple de la cryptographie basée sur l'identité dans la partie V.2.2.

Initialement, l'utilisation des couplages n'impliquait aucun secret, par conséquent, les attaques à canaux cachés n'étaient pas une menace pour la cryptographie à base de couplage. Cependant, lors d'un protocole de cryptographie basé sur l'identité, l'un des arguments utilisé pendant le calcul du couplage est secret. Il devient alors envisageable d'appliquer une attaque par canaux cachés.

Nous ferons ensuite un bref rappel des caractéristiques des attaques par canaux cachés, et nous présenterons deux types d'attaques en particulier, les attaques par consommation de courant et les attaques par injection de fautes. Les attaques par consommation de courant se subdivisent en deux catégories : les attaques simples et les attaques différentielles. Nous donnons les principes théoriques de ces attaques ; puis, nous illustrons l'attaque différentielle par consommation de courant et l'attaque par fautes en les appliquant à l'algorithme de Miller utilisé durant un protocole de cryptographie basée sur l'identité.

V.2 Cryptographie à clé publique

V.2.1 Principe de la cryptographie à clé publique

Nous faisons de nouveau appel à Alice et Bob et introduisons Eve.

Alice et Bob sont les personnes souhaitant communiquer de manière secrète. Eve est l'espionne voulant comprendre quel terrible secret ils peuvent bien s'échanger. La cryptographie à clé publique repose sur l'utilisation de protocoles asymétriques.

Le principe d'un protocole asymétrique est que les phases de chiffrement et déchiffrement d'un message sont déséquilibrées. Tout le monde peut chiffrer un message, mais seul celui qui possède la clé de déchiffrement peut

déchiffrer le message.

Ceci peut être illustré de la manière suivante, à l'aide de boîtes et de cadenas. Plaçons nous dans le cas où Bob souhaite envoyer un message à Alice. Afin de permettre l'utilisation d'un protocole à clé publique, Alice aura mis à la disposition de Bob, ou de n'importe quel autre utilisateur, des boîtes pouvant être cadenassées, ainsi que des cadenas ouverts. Pour envoyer son message à Alice de manière sécurisée, Bob va récupérer une boîte, il place son message clair à l'intérieur et la verrouille en utilisant le cadenas fourni par Alice. Il ne lui reste plus qu'à envoyer la boîte à Alice. Celle-ci étant théoriquement la seule à posséder la clé, elle seule peut ouvrir la boîte et lire le message.

Dans cet exemple, la clé publique d'Alice est représentée par les cadenas ouverts, et sa clé privée est la clé des cadenas. Chiffrer un message est donc très facile, puisqu'il suffit de cadenasser une boîte. Par contre, il est plus difficile de déchiffrer un message, car pour cela il faut posséder la clé du cadenas ou alors savoir le crocheter. La sécurité de ce protocole repose donc sur la robustesse des cadenas utilisés. Plus ils seront solides, plus les messages seront difficiles à récupérer dans la boîte cadenassée.

Le principe de la cryptographie à clé publique fut défini par Diffie et Hellman en 1976. Le premier protocole à clé publique a été le protocole RSA [RSA78], créé en 1978. Nous illustrons la notion de cryptographie à clé publique en développant l'exemple plus récent de la cryptographie basée sur l'identité, qui est basée sur l'utilisation des couplages.

V.2.2 Cryptographie basée sur l'identité

En 1984, A. Shamir lança un défi à la communauté des cryptographes , ce défi consistait à créer un protocole cryptographique basé sur l'identité [Sha84]. Seize ans plus tard, Boneh et Franklin ont proposé le premier protocole basé sur l'identité, ce protocole utilise la propriété de bilinéarité des couplages [BF01].

La cryptographie basée sur l'identité repose sur le principe simple que dans la paire (clé publique, clé privée), la clé publique d'un utilisateur est son identité. La clé privée est fournie à l'utilisateur par une autorité de confiance qui est chargée de créer toutes les clés relatives au protocole et aux identités des utilisateurs. La cryptographie basée sur l'identité est très bien décrite dans l'article [BSS05], un livre récent lui est entièrement dédié [JN09].

Nous nous plaçons dans le cas où Alice veut envoyer un message chiffré à Bob en utilisant le schéma de Boneh et Franklin [BF01].

Les données publiques sont une courbe elliptique E définie sur un corps fini $\mathbb{F}_p$ de degré de plongement k, deux sous groupes de E notés $\mathbb{G}_1$ et $\mathbb{G}_2$,

un sous groupe $\mathbb{G}_3$ de $\mathbb{F}_{p^k}$, un couplage $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_3$, P_{pub} un générateur de $\mathbb{G}_2$ et deux fonctions de hachage $H_1 : \{0,1\}^\star \to \mathbb{G}_1,\ H_2 : \mathbb{G}_2 \to \{0,1\}^n$, où n représente la longueur du message en bit.

La fonction H_1 associe à un message binaire de longueur quelconque un élément de $\mathbb{G}_1$; la fonction H_2 envoie tout élément de $\mathbb{G}_2$ sur un mot binaire de taille n bits.

La clé publique de Bob est obtenue par le calcul de $Q_B = H_1(Bob) \in \mathbb{G}_1$, où Bob est l'identité de Bob. Les clés privées sont fabriquées par l'autorité de confiance T_A. T_A choisit s sa clé privée, et calcule la clé publique du protocole $K_{pub} = [s]P_{pub} \in \mathbb{G}_2$, ainsi que la clé privée de Bob : $P_B = [s]Q_B \in \mathbb{G}_1$.

Alice choisit $m \in \mathbb{F}_p$. Avec les données publiques, elle peut calculer $Q_B = H_1(Bob)$ la clé publique de Bob, et le couplage $g_B = e(Q_B, K_{pub})$.
Elle envoie à Bob : $C = \langle [m]K_{pub}, M \oplus H_2(g_B^m) \rangle$, que nous notons $C = \langle U, V \rangle$.

Pour déchiffrer le message $C = \langle U, V \rangle$, Bob récupère sa clé privée P_B auprès de l'autorité de confiance et calcule $V \oplus H_2(e(P_B, U))$. La propriété de bilinéarité des couplages a pour conséquence que :

$$\underline{e(P_B, U)} = e([s]Q_B, [ms]P_{pub}) = e(Q_B, [s]P_{pub})^m = e(Q_B, K_{pub})^m = \underline{g_B^m}.$$

Ainsi, Bob peut déchiffrer le message en calculant $V \oplus H_2((g_B^m))$.

Le point important à souligner est que le déchiffrement d'un message transmis par un protocole basé sur l'identité nécessite un calcul de couplage $(e(P_B, U))$ entre la clé privée de Bob (P_B) et une partie du message chiffré (U). Le message reçu U est par définition public puisqu'il transite par un canal non sécurisé.

Un attaquant potentiel aura la connaissance de l'algorithme utilisé, en l'occurrence il s'agit de l'algorithme de Miller, et de ces paramètres, en particulier l'ordre r des points qui définit le nombre d'itérations faites durant l'algorithme de Miller. La clé privée n'influence ni le temps d'exécution, ni le nombre d'itérations de l'algorithme, ce qui est différent d'une attaque à canaux cachés contre un protocole RSA ou AES.

Un scénario d'attaque possible serait qu'Alice soit la cliente d'une banque, et Bob le terminal de cette banque. Si Eve parvenait à récupérer la clé privée du terminal elle pourrait déchiffrer toutes les transactions transitant par ce terminal.

V.3 Fuites par mesure de consommation

Kocher, Jaffe et Jun [KJJ99] sont les premiers à avoir présenté l'idée de mesurer la consommation de puissance d'un circuit électronique afin de retrouver des informations sur la clé secrète à l'intérieur d'un circuit non modifiable. Il existe deux méthodes visant à extraire les clés, il s'agit de l'analyse simple et l'analyse différentielle de puissance.
Un circuit électronique comme la carte à puce est composé de portes logiques dont les changements d'états se font en fonction des opérations effectuées. Ces changements génèrent des variations de la consommation électrique, dépendant des informations circulant dans les différents composants.
La consommation d'une opération donnée dépend quant à elle du poids de Hamming des données manipulées. Ainsi une même opération peut avoir une courbe de consommation totalement différente suivant les valeurs manipulées. Par exemple, pour la multiplication entre deux nombres, la consommation de puissance est différente suivant le nombre de bits à zéro de chacun des nombres considérés. En particulier, lors des calculs cryptographiques la consommation varie en fonction des données traitées. En enregistrant les courbes de consommations, nous pouvons déduire des informations sur les données sensibles permettant de les reconstruire, par exemple à l'aide d'un traitement statistique ou de corrélations. Les attaques simples par analyse de courant nécessitent un oscilloscope numérique, un ordinateur et un lecteur de carte modifié pour communiquer avec la carte. L'attaque différentielle, plus complexe, utilise d'autres outils dans le but d'obtenir et traiter les courbes de consommations. Avec les moyens actuels, le matériel nécessaire pour attaquer les circuits récents est abordable par de petites organisations.

V.3.1 Attaques simples

Les attaques simples par consommation de courant utilisent la structure de l'algorithme cible, elles sont notées SPA, de l'acronyme anglais Simple Power Analysis. Considérons le calcul de l'exponentiation rapide sur la courbe elliptique qui est effectué en appliquant l'Algorithme 7.

Algorithme 7 : Algorithme d'exponentiation rapide

Données : P un point d'une courbe elliptique, $r = \overline{r_n r_{n-1} \dots r_1 r_0}^2$ la décomposition binaire de l'exposant ;
Résultat : $[r]P$;
$T \leftarrow P$;
pour $i = n-1$ **to** 0 **faire**
  $T \leftarrow T + T$;
  si $r_i = 1$ **alors**
    $T \leftarrow T + P$
  fin
fin
retourner T

En analysant la succession d'opérations de l'Algorithme 7, nous pouvons nous apercevoir qu'il n'y a jamais deux additions de points consécutives. En effet, à chaque itération, un doublement est calculé, mais l'addition n'est pas faite pour toutes les itérations. Il y a clairement deux comportements différents de l'algorithme suivant la valeur du bit r_i de r. La courbe de consommation a un tracé régulier, qui traduit les opérations effectuées pendant l'exponentiation, soit un doublement, soit une addition. Comme ces deux opérations ne nécessitent pas le même nombre d'opérations élémentaires, la consommation pour effectuer un doublement et une addition n'est pas la même, ce que nous pouvons deviner dans la Figure V.1. Utilisant le fait qu'il ne peut y avoir deux additions consécutives, une simple lecture de la courbe de consommation nous permet de déterminer les parties de la courbe correspondant à un doublement, et les parties de la courbe correspondant à une addition. Ce qui nous permet directement de savoir si le bit r_i de l'exposant r est à 0 ou à 1.

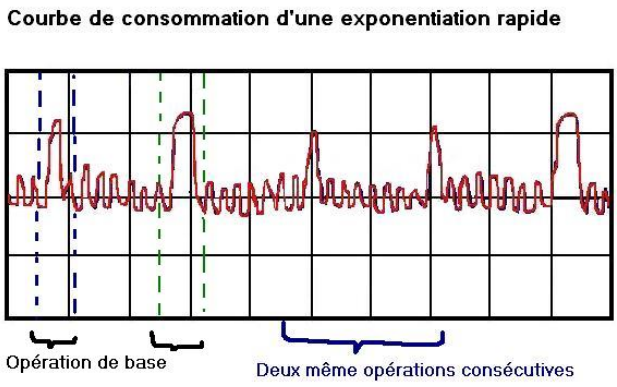


FIGURE V.1 – Consommation de l'exponentiation rapide

L'attaque que nous venons de présenter utilise une faille de l'algorithme que nous pouvons déceler dans la courbe de consommation de courant. Celle-ci a un comportement différent suivant la valeur du bit de l'exposant. Il existe des modifications simples à apporter à l'algorithme pour rendre cette attaque inefficace. Nous pouvons citer en exemple l'algorithme de Montgomery's Ladder [CF06, Chap. 29].

Une attaque simple contre un protocole cryptographique basé sur l'identité serait plus complexe. En effet, durant un protocole basé sur l'identité, le secret n'est pas l'exposant, mais une des données utilisées pendant les calculs. Nous allons introduire un autre type d'attaque par consommation de courant, les attaques différentielles par analyse de courant.

V.3.2 Principe général d'une attaque DPA

Une attaque DPA est une attaque différentielle par consommation de courant. Ces attaques à canaux cachés sont appliquées en particulier lorsque les attaques simples ont échoué, par exemple si les informations qui filtrent ne peuvent être directement liées au secret que l'attaquant veut déterminer.

Pour mener une attaque différentielle, il faut un accès aux canaux cachés, où pour des entrées différentes, le même secret est utilisé à chaque exécution du programme visé. Ces conditions sont difficiles à satisfaire. L'attaquant doit être en mesure de simuler plusieurs fois l'exécution d'un algorithme, et si possible connaître la structure interne du processeur attaqué.

Nous appliquons une attaque DPA à l'algorithme de Miller. DPA est un acronyme anglais signifiant Differential Power Analysis, ce que nous pouvons traduire par attaque par analyse différentielle de la consommation électrique. Le principe de ces attaques est le même que pour une attaque simple, nous utilisons les courbes de consommation de courant pour retrouver des informations sur le secret utilisé pendant les calculs. Nous commençons par décrire le principe général d'une attaque DPA.

L'idée de base d'une attaque DPA est de récupérer les courbes de consommation électrique d'un protocole cryptographique, puis à l'aide d'un traitement statistique de retrouver le secret utilisé pendant les calculs. Le principe est de faire une hypothèse sur le secret utilisé puis de vérifier si cette hypothèse est vraie ou pas. Elle porte souvent sur la valeur d'un ou plusieurs bits du secret.

Les calculs exécutés par l'algorithme de Miller se traduisent matériellement par des changements d'états des portes logiques. Par exemple en technologie CMOS, le changement d'état d'une porte entraîne la charge ou la décharge électrique des transistors pouvant être considérés comme des capacités, notées C dans les schémas. En particulier, lorsque le changement d'état d'un bit est un changement de 0 à 1, une charge électrique est stockée dans le condensateur C comme nous l'illustrons dans la flèche (a) de

la Figure V.2. Cela revient à connecter la capacité au VDD ce qui entraîne de la consommation d'énergie. De manière réciproque, lorsque un changement d'état du bit est un transfert de 1 à 0, alors l'énergie accumulée dans le condensateur est déchargée dans la terre (ou neutre électrique) comme l'illustre la flèche (b) de la Figure V.2, ce changement d'état est plus rapide que le précédent. Les transitions de bit de 0 à 0 ou de 1 à 1 ne participent pas à la variable globale en électricité du circuit.

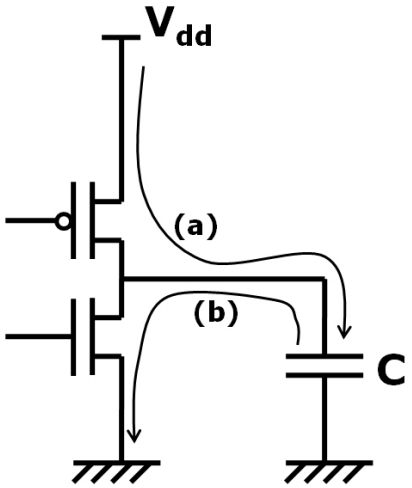


FIGURE V.2 – Consommation d'électricité dans un circuit CMOS

La variation de la consommation du circuit est donc la conséquence des changements d'états de l'ensemble des portes logiques pendant l'exécution de l'algorithme. Nous donnons maintenant quelques notions théoriques dont nous aurons besoin pour comprendre le principe d'une attaque DPA. Nous définissons le point critique comme étant l'instant où nous prenons en compte les variations de la courbe de consommation électrique. Il s'agit du moment où l'opération cible entre la clé secrète, ou un morceau de la clé secrète, est utilisée en interaction avec le message connu. Le point critique est appelé target node en anglais.

Pour des entrées P et Q connues, nous savons quel sera le comportement du point critique. Nous émettons une hypothèse sur la valeur d'un bit de P, et nous nous donnons plusieurs entrées différentes connues Q. En exécutant plusieurs fois l'algorithme avec les entrées choisies, nous obtenons plusieurs séquences $S_0, S_1, \ldots, S_n$ générant dans le circuit les transitions
$Tr_1(S_0 \to S_1), Tr_2(S_1 \to S_2), \ldots, Tr_n(S_{n-1} \to S_n)$.

Nous recueillons les courbes de consommation de chaque exécution de l'algorithme en supposant que le même secret P est utilisé à chaque fois. La consommation de courant dépend de la valeur de P et des entrées Q.

Connaissant le protocole utilisé ainsi que les algorithmes de calcul, il est possible de prévoir un résultat intermédiaire à l'aide de l'entrée connue et de l'hypothèse faite sur le secret.

L'ensemble des courbes de consommation du circuit est divisé en deux sous ensembles SA et SB. D'un côté, nous mettons les courbes pour lesquelles les transitions simulées du point critique entraînent une variation de la consommation d'électricité, il s'agit des passages de 0 à 1. Et de l'autre, nous plaçons les courbes pour lesquelles la transition simulée du point critique ne participe pas à la variation de consommation, il s'agit des transitions de bits de 0 à 0, de 1 à 1 et de 1 à 0.

- L'ensemble SA regroupe l'ensemble des courbes de consommation pour lesquelles le point critique présente une transition de 0 à 1, ce qui implique une consommation de courant de la porte visée.
- L'ensemble SB regroupe les courbes de consommation pour lesquelles le point critique présente les transitions de 1 à 0, de 0 à 0 ou de 1 à 1 qui n'influencent pas la consommation électrique du circuit.

Une partie de la consommation électrique du paquet SA est la conséquence de la consommation électrique de la porte logique visée par l'attaque. Bien entendu, d'autres portes logiques consomment de l'électricité durant l'exécution de l'algorithme, mais nous considérons que suffisamment de mesures sont faites pour pouvoir émettre l'hypothèse réaliste selon laquelle les consommations des portes qui ne sont pas le point critique sont aléatoirement réparties entre les ensembles SA et SB. Ainsi, pour un nombre de mesures assez grand, la différence de consommation électrique entre SA et SB n'est due qu'à la contribution de la porte cible.

Comme nous avons choisi de placer dans SA les courbes pour lesquelles la transition critique consomme du courant, et dans SB les courbes pour lesquelles la transition critique ne consomme pas de courant ; lorsque nous faisons la moyenne de consommation dans les deux paquets puis la différence entre ces deux moyennes deux cas de figure peuvent se produire. Soit l'hypothèse de départ est la bonne, alors le paquet de courbes SA contient réellement les courbes qui ont consommé de l'électricité au point critique et SB celles qui n'auront pas consommé de courant au point critique. Dans ces cas là, la différence des moyennes comportera un pic de consommation pour le point critique. Soit l'hypothèse de départ n'est pas la bonne, alors la séparation en deux paquets distincts aura été totalement aléatoire, et la différence des moyennes sera une courbe nulle partout puisque toutes les consommations électriques auront été aléatoirement réparties et par conséquent lissées par la moyenne. Donc le résultat final permet de savoir si l'hypothèse faite au départ sur un bit du secret est correcte ou pas, et par conséquent de découvrir un bit du secret. Nous déterminons ainsi le secret bit à bit.

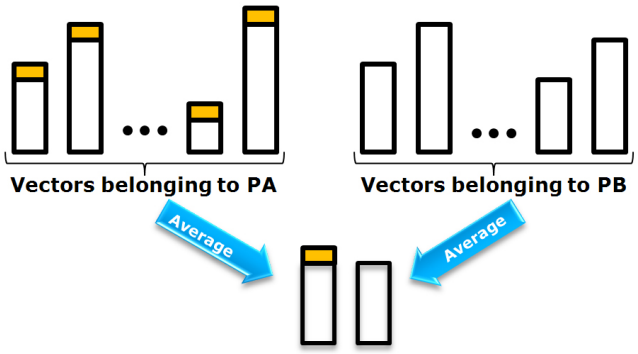


FIGURE V.3 – Classification des ensembles SA et SB

Dans la Figure V.3, nous représentons la consommation électrique du circuit soumis à différentes entrées connues. Nous supposons que l'hypothèse faite sur la clé secrète est la bonne. Autrement dit, les simulations sont menées en utilisant le secret. Chaque rectangle représente l'électricité totale consommée par le circuit pour chaque entrée différente connue. Pour cette Figure V.3, afin de rendre les explications plus claires, la consommation de courant est représentée par un rectangle correspondant à la consommation électrique moyenne de la porte visée. La partie grisée correspond à la consommation du point critique. Nous voyons qu'en calculant la moyenne des paquets de courbes SA et SB, seule la consommation grisée les différencie. Ainsi, si l'hypothèse faite sur la clé est la bonne, la différence des moyennes est non nulle.

Durant une attaque DPA, le point critique est choisi de telle sorte qu'il dépend d'une petite partie de la clé secrète seulement. Ainsi, les hypothèses sur le bout de clé peuvent être faites de façon exhaustive. Par exemple, pour conduire une attaque DPA sur l'algorithme DES (pour l'acronyme anglais Data Encryption Standard), le point critique choisi est la sortie d'une Sbox qui dépend de 6 (parmi 56) bits de la clé. Comme nous cherchons la clé secrète par morceaux de 6 bits, nous devons faire $2^6 = 64$ hypothèses, ce qui est négligeable comparé au 2^{54} hypothèses nécessaires pour une attaque brute force.

Il est important de remarquer que l'attaque que nous décrivons se base sur la mesure et l'analyse des consommations de courant instantanées uniquement sur la période de transition qui nous intéresse, et sans prendre en compte les moyennes en temps comme indiqué dans la Figure V.3. Lors d'une attaque réelle, qui se ferait sans connaître l'implémentation matérielle du circuit, l'attaquant ne pourrait savoir à quel moment les transitions cibles se font et il doit alors considérer la totalité de la courbe de consommation du circuit. Il doit tenir compte de l'évolution de la consommation électrique dans le temps. La Figure V.4 représente les courbes obtenues pour une at-

taque d'une durée de 1 nanoseconde.

Chacune des 8 courbes représente en fonction du temps la différence entre les moyennes de consommation électrique pour la transition cible du paquet SA et du paquet SB, pour une hypothèse de clé. Il existe plusieurs façons de sélectionner la courbe correspondant à la bonne hypothèse de clé. Nous pouvons choisir la courbe présentant le plus grand pic de consommation, ou encore celle dont l'aire sous la courbe est la plus grande. Quel que soit le moyen utilisé, il permet de choisir la courbe admettant la plus grande différence de consommation entre les ensembles SA et SB. Dans la Figure V.4, la bonne hypothèse correspond à la courbe en gras ; elle correspond à la courbe présentant le plus grand pic de consommation.

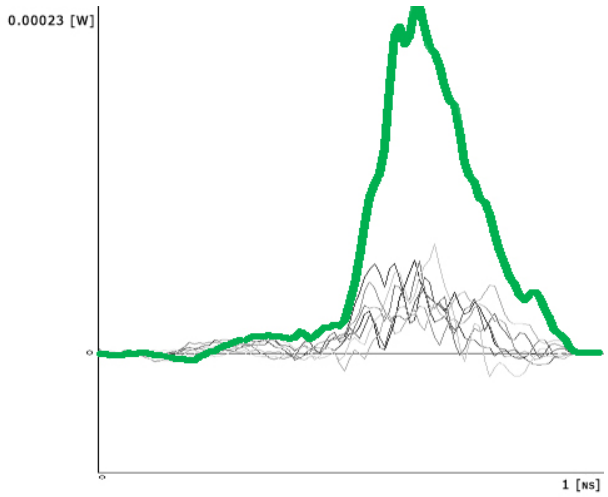


FIGURE V.4 – Résultat d'une attaque DPA

Nous mettons en place le scénario à appliquer pour des attaques par canaux cachés à l'algorithme de Miller.

V.4 L'algorithme de Miller face aux attaques par canaux cachés

V.4.1 Historique des attaques par canaux cachés à la cryptographie basée sur les couplages

L'utilisation d'un couplage dans un protocole basé sur l'identité implique l'utilisation d'un secret comme argument du couplage. De nombreuses implémentations efficaces des couplages existent, nous pouvons citer par exemple [BCF+05]. Les premières études des faiblesses d'un couplage face aux attaques à canaux cachés ont été publiées à partir de 2006 [PV06], [WS06],

[TTD+06] et [WS07].

Page et Vercauteren suggèrent dans [PV06] la possibilité de mener une attaque DPA fondée sur le principe de l'attaque de Messerges contre l'algorithme de Duursma et Lee, qui est utilisé pour calculer le couplage Eta en coordonnées affines. Malheureusement, ils n'ont pas développé l'attaque dans sa totalité. Dans [WS06], Whelan et Scott étudient de manière générale l'application d'attaques à canaux cachés contre des protocoles basés sur les couplages et parviennent à la conclusion selon laquelle placer le point secret utilisé comme premier argument du couplage est une contre mesure naturelle contre toute attaque à canaux cachés. Nous montrons que si les arguments du couplage sont les points P et Q, tels que le point P est secret et le point Q public ; alors il est possible pendant le calcul du couplage $e(P,Q)$ de retrouver le point P, malgré le fait qu'il soit le premier argument dans le calcul du couplage. Nous proposons une réalisation pratique de l'attaque DPA contre l'algorithme de Miller dans la partie V.5, ainsi qu'une validation expérimentale de cette attaque, ce qui faisait défaut aux attaques théoriques introduites en 2006. Nous terminons ce chapitre par une attaque par injection de fautes contre l'algorithme de Miller dans la partie V.6.

V.4.2 Contexte des attaques contre l'algorithme de Miller

La majorité des calculs de couplages repose sur l'utilisation de l'algorithme de Miller. C'est en tout cas le cas pour les couplages de Weil, de Tate et Ate comme nous l'avons vu dans la partie I.8. Nous menons une attaque DPA et une attaque par fautes contre l'algorithme de Miller dans les parties V.5 et V.6. Le contexte d'attaque est le même pour les deux.

Nous supposons que l'algorithme de Miller est implémenté en soft dans du matériel électronique, par exemple une carte à puce. Les attaques sont effectuées pendant l'exécution d'un protocole de cryptographie basée sur l'identité. Nous notons Q le message public. La clé privée sera représentée par le point P dans le calcul du couplage de $e(P,Q)$. Nous restreignons notre étude au cas où le secret est le premier argument du couplage. Placer le secret en premier paramètre du couplage est une contre-mesure proposée dans [WS06]. Si le secret est le second argument du couplage, les mêmes schémas d'attaques peuvent s'appliquer et permettent de retrouver le secret utilisé. L'attaquant peut calculer autant de fois que nécessaire le couplage entre le secret P qui ne changera pas et les entrées Q qu'il modifiera à sa guise. Il peut enregistrer et stocker les courbes de consommation électrique de chaque exécution, ainsi que le résultat final de l'algorithme de Miller.

Nous notons r l'ordre des sous groupes $\mathbb{G}_1$, $\mathbb{G}_2$ et $\mathbb{G}_3$. Nous considérerons que le couplage de Miller se calcule en r itérations. Les attaques que

nous menons s'adaptent très facilement aux couplages se calculant en moins d'itérations. Nous considérons que l'algorithme de Miller est implémenté en coordonnées Jacobiennes. Les mêmes schémas d'attaques s'appliquent aux autres système de coordonnées. Nous ne considérons pas les autres coordonnées pour l'attaque DPA car l'attaque s'effectuerait exactement de la même façon. Par contre, pour l'attaque par fautes la valeur de k, le degré de plongement de la courbe, influence l'attaque. En effet, nous construisons pour mener à bien cette attaque des systèmes dont la résolution nous permet de retrouver le secret utilisé. Le nombre d'équations nécessaires dépend de la valeur de k et du système de coordonnées utilisé. Nous considérons donc les coordonnées affines, projectives et d'Edwards dans la partie V.6.4.

V.5 Attaque DPA contre l'algorithme de Miller

Nous nous intéressons aux faiblesses de l'algorithme de Miller face à une attaque DPA. Nous expliquons comment l'attaque peut être menée, et illustrons notre démarche par une simulation d'attaque. Nous menons l'attaque DPA contre l'algorithme de Miller développé en coordonnées Jacobiennes, introduites dans la partie I.2.5, ce qui est a priori une première dans la littérature. L'attaque que nous décrivons s'adapte aisément à tous autres types de coordonnées utilisées en cryptographie à base de courbes elliptiques, par exemple affines, projectives et les récentes coordonnées d'Edwards.

Nous décrivons le schéma de l'attaque DPA contre l'algorithme de Miller dans la partie V.5.1. Puis nous présentons dans la partie V.5.2 le circuit que nous attaquons. Nos résultats sont consignés dans la partie V.5.3. Enfin, nos conclusions et perspectives de travaux sont donnés dans la partie V.5.4. Ces travaux ont été menés en collaboration avec Giorgio Di Natale et Marie Lise Flottes, tous deux Chargés de Recherche au CNRS et membres du département de Micro-électronique du LIRMM. Ils ont été publiés dans les proceedings IEEE de la conférence Prime 2009 [EDF09].

V.5.1 Convention

Nous présentons les notations de cette partie, et nous plaçons le cadre de l'attaque DPA.

Schéma d'attaque

Nous voulons retrouver à l'aide d'une attaque DPA les coordonnées (X, Y, Z) du point T, qui est un multiple du secret P, construit pendant l'exécution de l'algorithme de Miller décrit dans la partie I.8.2. Nous focalisons notre

attention sur les opérations effectuées durant le calcul de $l_1(Q)$ et impliquant les coordonnées du point T courant. L'équation de $l_1(Q)$ est

$$l_1(x_Q, y_Q) = Z_3 Z^2 y_Q - 2Y^2 - (3X^2 - aZ^4)(Z^2 x_Q - X).$$

1. Tout d'abord nous voulons retrouver la coordonnée Z à l'aide de la multiplication $(Z^2) \times x_Q$ intervenant dans le calcul de $l_1(Q)$.

2. Ensuite, deux possibilités s'offrent à nous, déterminer soit Y, soit X.

   Pour déterminer Y nous pouvons utiliser la multiplication $Z_3 Z^2 y_Q$, sachant que $Z_3 = 2YZ$ et que nous connaissons les différents y_Q et la coordonnée Z. L'attaque se ferait sur le schéma d'attaque utilisé pour retrouver la coordonnée Z.

   Pour déterminer X, nous pouvons utiliser l'opération $(Z^2 x_Q - X)$; sachant que nous connaissons les valeurs x_Q et Z.

3. Une fois que nous aurons obtenus deux coordonnées sur trois du point T, l'équation de la courbe elliptique nous permettra de retrouver la troisième.

Multiplication dans $\mathbb{F}_p$

Nous décrivons l'attaque en considérant que $k = 1$; à ce moment là les coordonnées du point Q sont des éléments de $\mathbb{F}_p$. La multiplication $(Z^2) \times x_Q$ est alors une multiplication dans $\mathbb{F}_p$, ce qui simplifie les explications. Pour les valeurs de $k > 1$, le même raisonnement s'applique sans aucune modification.

En effet, si $k > 1$, la multiplication $(Z^2) \times x_Q$ est alors la multiplication d'un élément de $\mathbb{F}_{p^k}$ (x_Q) par un élément de $\mathbb{F}_p$ (Z^2), or une multiplication de ce genre se fait dans la base de $\mathbb{F}_{p^k}$ en k multiplications dans $\mathbb{F}_p$. Soit $\mathcal{B} = \{1, \gamma^1, \ldots, \gamma^{k-1}\}$ une base de $\mathbb{F}_{p^k}$, il existe un polynôme irréductible sur $\mathbb{F}_p$ de degré k et à coefficient dans $\mathbb{F}_p$ admettant γ comme racine. Ce polynôme est connu de tous. L'élément x_Q se décompose dans cette base en $x_Q = \sum_{i=0}^{k-1} x_{Q_i} \gamma^i$, avec pour tout i compris entre 0 et $(k-1)$, $x_{Q_i} \in \mathbb{F}_p$. La multiplication $(Z^2) \times x_Q$ peut alors s'écrire $Z^2 x_Q = \sum_{i=0}^{k-1} \left(Z^2 \times x_{Q_i}\right) \gamma^i$, elle s'effectue en k multiplications dans $\mathbb{F}_p$.

Donc pour $k > 1$, il suffit d'appliquer le même schéma d'attaque que pour $k = 1$ en se focalisant sur une ou plusieurs des multiplications intermédiaires.

La différence $(Z^2 x_Q - X)$ est une différence dans $\mathbb{F}_p$, pour toutes les valeurs de k. En effet, $Z^2 x_Q = \sum_{i=0}^{k-1} (Z^2 x_{Q_i}) \gamma^i$ et par la suite :

$$Z^2x_Q - X = \left((Z^2x_{Q_0}) - X\right) + \sum_{i=1}^{k-1}(Z^2x_{Q_i})\gamma^i.$$

Première itération

Nous détaillons le principe de l'attaque en nous basant sur le fait qu'à la première itération le point T n'est autre que le point P. Ainsi, retrouver le point T revient à révéler directement la clé privée. L'attaque peut être menée pour n'importe quelle autre itération. Supposons que pour des raisons pratiques il soit plus facile de récupérer des courbes minimisant le bruit pour la j-ième itération. A la fin de l'attaque, nous aurons trouvé les coordonnées du point $T = [j]P$.

P et Q sont des points du même ordre r, en comptant le nombre de cycles d'horloge nécessaires pour exécuter l'algorithme de Miller, nous commençons par déterminer la valeur de j quasiment en lisant la décomposition binaire de r. Pour retrouver l'entier j et pouvoir en déduire le point P, nous pouvons déterminer Γ le nombre de cycles d'horloge effectués par l'algorithme avant le début de l'attaque DPA. Nous connaissons le nombre de cycles d'horloge des étapes de doublement et d'addition de l'algorithme de Miller. La donnée de Γ nous permet de déterminer τ le nombre d'itérations de l'algorithme de Miller qui ont été effectuées. Le nombre τ nous permettra de trouver la valeur de j en lisant les bits de r. Nous posons $j = 1$, puis nous lisons la décomposition binaire de r du bit de rang n au $(n-1-\tau)$-ième bit de r. Si $r_i = 0$ nous calculons $j \leftarrow 2j$, sinon nous calculons $j \leftarrow 2j+1$, et nous continuons jusqu'à arriver au $(n-1-\tau)$-ième bit de r, où n est le nombre de bit total de r. Par exemple soit $r = 1000010000101$ en base 2 ($n = 13$) et supposons que nous avons calculé 5 itérations de l'algorithme de Miller $\tau = 5$.

- $j = 1$
- Première itérations $r_{12} = 0$ donc $j = 2$.
- Deuxième itération $r_{11} = 0$ donc $j = 4$.
- Troisième itération $r_{10} = 0$ donc $j = 8$.
- Quatrième itération $r_9 = 0$ donc $j = 16$.
- Cinquième itération $r_8 = 1$ donc $j = 33$.

Une fois l'entier j déterminé, il suffit de calculer j' l'inverse de j modulo r puis de calculer $[j'][j]P = P$, et ainsi nous aurons retrouvé P.

V.5.2 Description du circuit

Afin de valider de manière pratique l'attaque DPA contre l'algorithme de Miller, nous avons implémenté un circuit pour évaluer la fonction l_1 de l'algorithme de Miller présenté dans la partie I.8.2. Nous nous sommes concentrés sur une architecture de 8 bits, et nous avons considéré un petit nombre premier (p = 251). Il est évident qu'un nombre premier de 8 bits est loin des tailles cryptographiques utilisées, mais nous avons voulu valider l'attaque par une réalisation pratique. En effet, s'il est possible de mener l'attaque sur un modulo de 8 bits, alors pour des nombres premiers de taille 160 bits, il suffit de découvrir la clé secrète par morceaux de 8 bits. Le principe de l'attaque DPA sur un grand nombre est de faire des hypothèses sur un petit morceau de la clé et de cibler une porte logique dont la consommation dépendra directement de ce morceau de clé considéré. Donc en divisant la clé secrète de taille 160 bits, en des morceaux de taille 8 bits, nous mènerons l'attaque de la même façon que nous l'avons réalisée pour un octet. Nous avons utilisé une simulation pour effectuer les mesures. Même si la simulation d'un circuit présente l'avantage qu'il n'y ait pas de bruits parasites pendant les mesures, il s'agit d'une étape importante pour valider une attaque théorique. Effectivement, si une attaque DPA peut être menée à bien sur un circuit simulé, alors cette même attaque pourra être reproduite sur un circuit réel. Réciproquement, si une attaque DPA ne peut être menée à bien à travers des simulations, alors il est certain que l'attaque DPA ne sera pas efficace sur un circuit réel.

Le circuit représenté par la Figure V.5 comporte trois registres pour les trois coordonnées du secret (X, Y, Z), trois registres $R1, R2$ et $R3$ servant à stocker les résultats intermédiaires et une unité arithmétique et logique, notée ALU pour l'acronyme anglais Arithmetic Logic Unit, l'ALU permet d'effectuer la multiplication et l'addition/soustraction de deux variables. L'unité de contrôle notée CU dirige tous les signaux pour effectuer le calcul de la fonction l_1. En particulier, le CU collecte les données d'entrées qui sont les coordonnées affines (x_Q, y_Q) du point Q. L'évaluation de la fonction l_1 au point Q se fait en 15 cycles d'horloge. L'algorithme exécuté par le CU pour calculer $l_1(Q)$ est décrit Tableau V.1.

Le circuit a été écrit en VHDL, puis simulé en utilisant une technologie de 130 nanomètres proposée par STMicroelectronics [1]. La surface totale du circuit est équivalente à 79 059 portes logiques et les calculs sont effectués à une fréquence de 120 Mhz.

[1] `http://www.st.com`

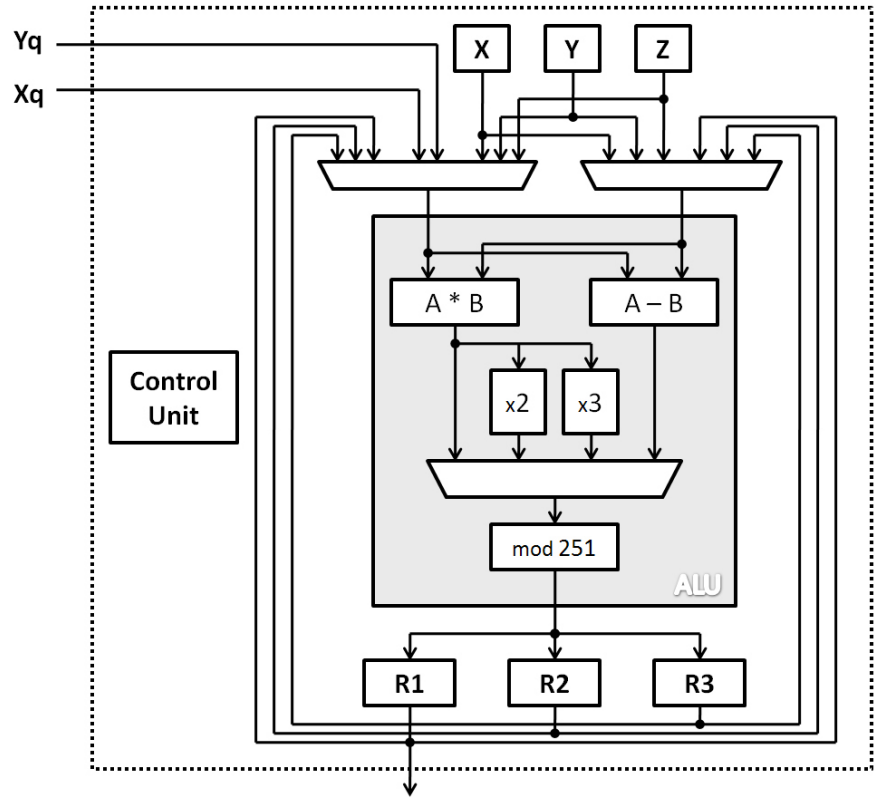


FIGURE V.5 – Circuit pour le calcul de $l_1(x_Q, y_Q)$

V.5.3 Description de l'attaque et résultat

Nous voulons retrouver la clé privée $P = (X_P, Y_P, Z_P)$, en enregistrant les courbes de consommation durant l'exécution de l'algorithme de Miller. Nous concentrons les mesures sur un laps de temps limité durant lequel des calculs impliquant la clé privée P et le message publique Q sont effectués. Nous nous sommes concentrés sur la consommation électrique de la multiplication entre Z_P^2, une des coordonnées secrètes, et x_Q, une des coordonnées du message connu Q. Nous avons concentré nos mesures au niveau de la première itération pour laquelle $T = P$. Cette multiplication est effectuée durant la seconde étape du calcul de $l_1(Q)$ comme indiqué dans le Tableau V.1.

Pour effectuer deux types d'attaques DPA différentes, nous avons choisi de retrouver la coordonnée X_P du secret en nous focalisant sur la soustraction entre le résultat de la multiplication $Z_P^2 \times x_Q$ et X_P. Cette opération est effectuée durant la troisième étape du Tableau V.1.

Une fois retrouvées les coordonnées X_P et Z_P, la troisième coordonnée

1	$\mathbf{R1 = R3 = Z^2}$	
2	$\mathbf{R1 = R1 \times x_Q}$	$R1 = Z^2 \times x_Q$
3	$\mathbf{R1 = R1 - X}$	$R1 = Z^2 x_Q - X$
4	$\mathbf{R2 = X - R3}$	$R2 = X - Z^2$
5	$\mathbf{R3 = X + R3}$	$R3 = X + Z^2$
6	$\mathbf{R2 = 3 \times R2 \times R3}$	$R2 = 3(X - Z^2)(X + Z^2)$
7	$\mathbf{R3 = R2 \times R1}$	$R3 = 3(X^2 - Z^4)(Z^2 x_Q - X)$
8	$\mathbf{R1 = Z^2}$	
9	$\mathbf{R1 = R1 \times y_Q}$	$R1 = Z^2 y_Q$
10	$\mathbf{R1 = R1 \times Y}$	$R1 = YZ^2 y_Q$
11	$\mathbf{R1 = 2 \times R1 \times Z}$	$R1 = 2YZZ^2$
12	$\mathbf{R2 = 2 \times Y^2}$	
13	$\mathbf{R1 = R1 - R2}$	$R1 = 2YZZ^2 y_Q$
14	$\mathbf{R1 = R1 - R3}$	$R1 = l_1(x_Q, y_Q)$
15	Output ready	

TABLE V.1 – Algorithme dans le CU

Y_P peut être retrouvée à l'aide de l'équation de la courbe
$E : Y^2 = X^3 + aXZ^4 + bZ^6$.

Afin d'éviter les coûts engendrés par une véritable architecture, et les inconvénients des expérimentations réelles, nous avons utilisé un circuit simulé pour effectuer l'attaque DPA.

Pour notre simulation, nous avons utilisé l'environnement Synopsys NanoSim [2] qui est un simulateur avancé de circuits analogiques avec de hautes performances. Nous avons mené l'attaque sur des courbes de consommation électriques obtenues via l'environnement de simulation [NFR08]. Cet environnement récupère les courbes de consommation et effectue le traitement statistique décrit dans la partie V.3.2.

L'attaque sur la coordonnée Z_P du point secret P a été menée en utilisant 256 vecteurs en entrée (256 transitions). Le point critique choisi est le résultat de la multiplication avant la réduction modulaire modulo 251. Nous avons construit la multiplication de la manière qui nous semblait judicieuse pour des tailles cryptographiques. Nous effectuons d'abord le produit, puis la réduction modulaire. La cible Z_P est codée sur 8 bits, nous avons testé les $2^8 = 256$ possibilités pour la clé secrète. Dans la Figure V.6, nous donnons les résultats de l'attaque. La courbe des différences de moyennes des ensembles SA et SB correspondante à la bonne hypothèse est identifiable, car c'est la courbe présentant le pic de consommation le plus important. Cette courbe, et de manière générale toutes les courbes présentant un pic important de consommation, correspond à la différence de consommation la plus impor-

[2] `http://www.synopsys.com`

tante entre les moyennes des paquets SA et SB. Ce qui signifie que pour cette courbe la répartition entre les deux paquets s'est faite en accord avec les calculs réels. Pour la Figure V.6, l'axe des abscisses représente le temps, et le temps 0 correspond au début du calcul de la deuxième étape de l'unité de contrôle.

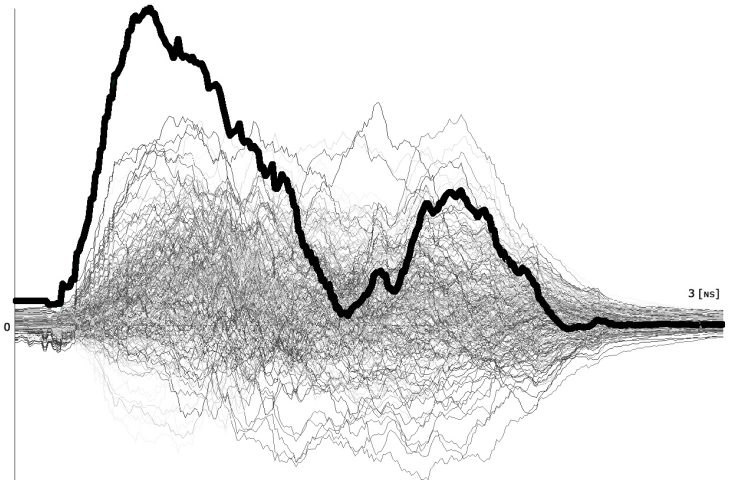


FIGURE V.6 – Attaque DPA contre la multiplication

La seconde partie de la clé privée, la coordonnée X_P, est obtenue en se focalisant sur la consommation de la troisième étape du CU. L'attaque DPA sur la différence nécessite un nombre plus important de courbes pour être concluante. En effet, la différence est une opération durant laquelle moins de transition de bits sont effectuées, et la portion de circuit dédiée à la soustraction est beaucoup plus petite que la portion de circuit dédiée à la multiplication. La variation de consommation électrique due à une différence est plus réduite que celle d'une multiplication. De ce fait, nous avons besoin de plus de traces de consommation pour obtenir un résultat. Nous avons été contraints d'utiliser toutes les possibilités pour la coordonnées x_Q, à savoir 65 280 valeurs. Dans la Figure V.7, nous montrons le résultat de l'attaque. Pour cette attaque, la courbe correspondant à la bonne hypothèse et au bon partage des courbes dans SA et SB est identifiable à son pic de consommation. Il est donc facile de retrouver la coordonnée X_P du secret en repérant la courbe dont le pic de consommation est le plus important. D'autres méthodes pour repérer la courbe correspondant à la bonne hypothèse existent. Il peut s'agir d'étudier l'aire sous la courbe par exemple dans les cas où aucun pic significatif ne serait révélé de prime abord. Ces méthodes sont détaillées dans [NFR08].

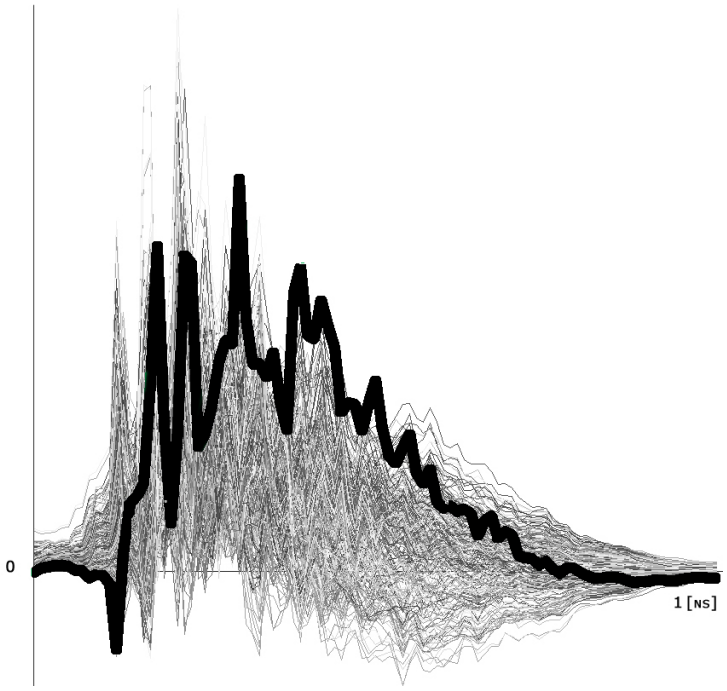


FIGURE V.7 – Attaque DPA contre la différence

V.5.4 Conclusion et perspectives

Nous avons présenté une attaque DPA effective contre l'algorithme de Miller et montré à l'aide de simulations que l'algorithme de Miller est sensible à ce genre d'attaque. Nous avons mené l'attaque en coordonnées Jacobiennes, elle peut aisément se traduire en coordonnées affines, projectives ou encore en coordonnées d'Edwards.

Les résultats expérimentaux obtenus à l'aide d'une simulation électrique montrent que sans contre-mesure l'algorithme de Miller est sensible à une attaque DPA ; et que la contre-mesure consistant à placer le secret comme premier argument du couplage n'est pas efficace. A fortiori, les couplages de Weil, de Tate et Ate sont eux-même sensibles à une attaque DPA. Les perspectives liées à ces travaux sont d'étudier les contre-mesures proposées pour protéger les couplages, la plus efficace étant la troisième contre-mesure de Coron.

Cette contre-mesure utilise l'homogénéité des coordonnées Jacobiennes pour modifier les valeurs des coordonnées du secret sans changer le secret en lui même :

$$\left(\forall\ \lambda \in \mathbb{F}_p^\star, (X,Y,Z) = (\lambda^2 X, \lambda^3 Y, \lambda Z)\right).$$

Une contre-mesure dérivée de celle ci fut proposée par Page et Vercauteren [PV06]. Cette contre-mesure utilise la propriété de bilinéarité des cou-

plages. A savoir que $e_*([s]P,[r]Q) = e_*(P,Q)^{sr}$ (où $e_*(P,Q)$ représente un couplage quelconque). Ainsi pour un couple (r,s) tel que $rs \equiv 1\ mod(l)$, cette contre-mesure nécessite seulement deux multiplications dans $\mathbb{F}_p$ avant de lancer les calculs de couplage. Cette contre-mesure serait a priori efficace contre une attaque DPA, mais il pourrait être possible de la rendre obsolète en augmentant le nombre de courbes ou en utilisant une attaque DPA de grand ordre notée HODPA [CF06].

Toutes ces pistes sont encore à explorer.

V.6 Attaque par fautes contre l'algorithme de Miller

Nous présentons une attaque par injection de fautes contre l'algorithme de Miller, étape centrale du calcul des couplages de Weil, de Tate et Ate. Notre motivation est d'étudier les faiblesses de cet algorithme soumis à une attaque par injection de fautes. Nous complétons ainsi les études de [PV06] et [WS07].

Nous montrons que cet algorithme est sensible aux attaques par fautes, et ce quel que soit le système de coordonnées. Nous prouvons que placer le secret en premier argument du couplage n'est pas une contre-mesure aux attaques par injections de fautes.

Le principe des attaques par injection de fautes est de perturber l'exécution d'un protocole cryptographique, ceci afin de retrouver des informations sur le secret utilisé. Les injections de fautes peuvent être provoquées en utilisant un laser, des radiations électromagnétiques ou encore des changements de température. Nous réalisons une attaque par fautes consistant à modifier le nombre d'itérations pendant l'exécution de l'algorithme de Miller.

Dans [PV06], Page et Vercauteren décrivent une attaque par fautes contre l'algorithme de Duursma et Lee. L'attaque par fautes consiste à modifier le nombre d'itérations de cet algorithme. Nous avons adapté leur idée à l'algorithme de Miller, et nous donnons une description détaillée de la méthode d'attaque, ainsi que les probabilités de vérifier les conditions nécessaires à l'aboutissement de l'attaque.

Dans [WS07], Whelan et Scott présentent une attaque par fautes contre le couplage de Weil et le couplage Eta. Ils considèrent le cas pour lequel leur injection de fautes modifie exactement la dernière itération de l'algorithme cible. Ils concluent dans leur article que l'algorithme de Miller n'est pas vulnérable à une attaque par faute, car le système qu'ils obtiennent en appliquant l'attaque est non linéaire et donc impossible à résoudre. De plus dans [WS06], ils déclarent que passer le paramètre secret du couplage en premier argument durant le calcul est une contre-mesure naturelle contre toute attaque par canaux auxiliaires. Contrairement à leurs affirmations, dans la

partie V.5 nous avons montré que ce n'était pas le cas pour une attaque DPA, nous montrons que ce n'est pas plus valable pour une attaque par injection de fautes. Nous décrivons un moyen d'attaquer n'importe quelle itération de l'algorithme de Miller.

Chacun des articles cités décrit l'attaque en coordonnées affines. Nous montrons dans la partie V.6.4 que, quel que soit le système de coordonnées utilisé : affine, projectives, Jacobiennes et même Edwards, notre attaque est efficace.

La démonstration de la vulnérabilité de l'algorithme de Miller à une attaque par fautes est décrite dans la partie V.6.1, puis nous faisons une analyse de la vulnérabilité à ce type d'attaque des couplages de Weil, de Tate et Ate dans la partie V.6.5. Nous montrons que le couplage de Weil est très sensible à une attaque par faute. Dans le cas des couplages de Tate, Ate et Twisted Ate, nous donnons des pistes d'études possibles pour contourner la contre-mesure naturelle qu'est l'exponentiation finale des couplages de Tate et Ate.

Enfin, nous concluons nos travaux dans la partie V.6.6. Ces travaux sont publiés dans les proceedings LNCS de la conférence ISA 2009 [El 09].

V.6.1 Description de l'attaque par fautes contre l'algorithme de Miller

Le but d'une attaque par injection de fautes est de provoquer des erreurs durant l'exécution d'un algorithme afin de modifier son déroulement et ainsi de révéler des informations sur le secret utilisé. Les injections de fautes peuvent consister en la modification de la mémoire interne par des émissions électromagnétique ou laser. Ce type d'attaques nécessite un appareillage très précis et assez onéreux. Néanmoins, les technologies actuelles rendent ces attaques réalisables [Hab92].

Nous complétons le schéma de l'attaque présentée dans [PV06] pour l'algorithme de Duursma et Lee, et nous l'appliquons à l'algorithme de Miller.

La première étape de l'attaque par fautes consiste à retrouver les registres mémoires de l'appareil électronique dédiées au nombre d'itérations de l'algorithme de Miller ; il s'agit du registre mémoire contenant l'entier $N = \lfloor \log_2(r) \rfloor + 1$, il s'agit de la longueur binaire de r. Cette étape peut être menée en utilisant une procédure de *reverse engeenering* (en anglais). Le principe est « d'effeuiller » le processeur couche après couche pour pouvoir reconstruire le schéma du circuit.

Dans les architectures classiques le compteur du nombre d'itérations est codé dans un petit bloc de 8 bits. Nous exécutons plusieurs fois l'algorithme et à chaque fois nous modifions aléatoirement ce registre à l'aide d'une attaque électromagnétique ou laser [AK96] pour provoquer des erreurs durant

l'exécution de l'algorithme. Les lasers actuels ont des diamètres suffisamment fins [Hab92] pour pouvoir viser des composants très précis d'un processeur.

En comptant le nombre d'itérations faites après perturbation nous pouvons déterminer le nouveau nombre d'itérations effectuées, nous le notons τ.

Pour chaque exécution, nous enregistrons le résultat de l'algorithme de Miller et nous lui associons l'entier τ qui correspond au nombre d'itérations faites par cet algorithme. Le but de l'attaque est de retrouver un couple $(\tau, \tau+1)$ de deux nombres d'itérations consécutives, ainsi que les résultats correspondants. La probabilité d'obtenir deux exécutions avec deux nombres d'itérations consécutifs est donnée dans la partie V.6.2.

V.6.2 La τ-ième itération

Nous exécutons l'algorithme de Miller plusieurs fois. A chaque exécution, nous provoquons une erreur par injection de fautes pour modifier le nombre d'itérations, et ce tant que nous n'aurons pas trouvé le résultat de l'algorithme de Miller pour τ et $\tau+1$ itérations de l'algorithme.

Nous notons $f_{\tau,P}(Q)$ et $f_{\tau+1,P}(Q)$ les deux résultats de Miller pour la τ-ième et la $(\tau+1)$-ième itérations. Afin de mener l'attaque à bien, nous devons étudier le rapport

$$\frac{f_{\tau+1,P}(Q)}{f_{\tau,P}(Q)^2}.$$

Ce rapport est la conséquence de la mise à jour de la fonction f_1 pendant l'étape de doublement de l'algorithme de Miller. La donnée de ce rapport nous permet de retrouver les coordonnées du point secret P. En effet, en fonction de la base de $\mathbb{F}_{p^k}$ et du système de coordonnées choisi pour représenter P, nous construisons un système d'équations dont le point P est solution. Nous détaillerons la construction du système et sa résolution dans la partie V.6.3. Nous étudions auparavant la probabilité d'obtenir avec succès deux nombres d'itérations consécutifs.

Probabilité de trouver un couple d'itérations consécutives $(\tau, \tau+1)$

L'attaque par fautes que nous décrivons repose sur l'hypothèse que nous puissions obtenir un couple de deux exécutions de l'algorithme de Miller dont les nombres d'itérations seraient consécutifs. Pour que l'attaque soit réalisable, nous devons pouvoir obtenir cette condition au bout d'un nombre fini d'essais.

Pour trouver la probabilité de cet évènement, nous appliquons le principe du paradoxe des anniversaires. Obtenir un couple d'exécutions ayant deux

nombres d'itérations consécutifs est le complémentaire de l'évènement obtenir des exécutions dont pas une des exécutions n'a un nombre d'itérations consécutif d'une autre. Notons $B(n,N)$ le nombre de tirages possibles de n entiers parmi N, tels que parmi les n entiers tirés au sort aucun ne soit le consécutif d'un autre. La valeur de $B(n,N)$ peut être déterminée par la formule de récurrence suivante :

$$\begin{cases} N \leq 0, n > 0, B(n,N) &= 0, \\ \forall N, n = 0, B(n,N) &= 1 \\ \forall N, n > 0, B(n,N) &= \sum_{j=1}^{N} \sum_{i=1}^{n} B(n-i, j-2). \end{cases}$$

Nous donnons une idée de la construction de cette formule dans l'Annexe A.2. Cette formule nous permet de trouver la probabilité $P(n,N)$ d'obtenir deux entiers consécutifs en tirant aléatoirement n entiers parmi N. Cette probabilité est

$$P(n,N) = 1 - \frac{B(n,N)}{C_{n+N}^{n}}$$

Pour une architecture 8 bits au plus 15 tirages sont nécessaires pour obtenir deux nombres d'itérations consécutifs avec une probabilité supérieur à 50% : $P(15, 2^8) = 0,56$. Ainsi, les hypothèses que nous faisons sont réalisables en pratique.

Trouver j

Après τ itérations, l'algorithme de Miller a calculé le point $[j]P$, donc durant la $(\tau+1)$-ième itération, il va calculer le point $[2j]P$, ou $[2j+1]P$, suivant la valeur du $(\tau+1)$-ième bit de r.

Pour déterminer la valeur de j, nous comptons le nombre de cycles d'horloge effectué durant les τ itérations de l'algorithme. Nous connaissons le nombre de cycles d'horloge pour les étapes de doublement et d'addition. Avec le nombre total de cycles d'horloge, nous pouvons déterminer τ le nombre d'itérations que nous avons effectuées, et en déduire l'entier j, exactement de la même façon que dans la partie V.5.1 pour l'attaque DPA. P et Q sont des points du même ordre r, avec la connaissance de τ, nous pouvons déterminer la valeur de j quasiment en lisant la décomposition binaire de r.

Par exemple soit $r = 1000010000101$ en base 2 et admettons que nous avons calculé 5 itérations de l'algorithme de Miller $\tau = 5$.

- $j = 1$
- Première itérations $r_{12} = 0$ donc $j = 2$.
- Deuxième itération $r_{11} = 0$ donc $j = 4$.

- Troisième itération $r_{10} = 0$ donc $j = 8$.
- Quatrième itération $r_9 = 0$ donc $j = 16$.
- Cinquième itération $r_8 = 1$ donc $j = 33$.

V.6.3 Courbe et équation

Les articles [PV06] et [WS07] décrivent leurs attaques en coordonnées affines. Pour les coordonnées affines, le résultat se trouve facilement par une identification des éléments dans la base de $\mathbb{F}_{p^k}$. Nous démontrons que pour tout système de coordonnées, l'attaque par fautes contre l'algorithme de Miller présentée ici donne des résultats. Nous décrivons l'attaque en coordonnées Jacobiennes. La différence avec les coordonnées affines est que l'attaque demande de résoudre un système non-linéaire.

Le degré de la courbe

Notre attaque repose sur la résolution d'un système obtenu par identification des éléments dans la base de $\mathbb{F}_{p^k}$. Lorsque $k = 3$, les équations obtenues par identification suffisent à trouver le résultat. $k = 3$ est d'ailleurs la valeur minimale pour que le système obtenu soit résoluble « à la main », sans utiliser l'équation de la courbe et la méthode des résultants, ce que nous décrivons dans la partie V.6.3. Pour que l'écriture du système soit simple nous décrirons l'attaque en considérant que $k = 4$; bien que l'attaque fonctionne de la même façon pour les valeurs de k supérieures ou égales à trois. Nous notons $B = \{1, \xi, \sqrt{\nu}, \xi\sqrt{\nu}\}$ une base de $\mathbb{F}_{p^4}$ construit par une tour de deux extensions de degré deux. $P \in E(\mathbb{F}_p)$ est donné en coordonnées Jacobiennes : $P = (X_P, Y_P, Z_P)$; et le point $Q = (x_Q, y_Q) \in E(\mathbb{F}_{p^k})$ en coordonnées affines.

Comme nous l'avons vu dans la partie I.3.5, nous pouvons utiliser une courbe tordue de $E(\mathbb{F}_{p^4})$ pour écrire les coordonnées de Q : $Q = (x_Q, y_Q\sqrt{\nu})$, avec x_Q, y_Q et $\nu \in \mathbb{F}_{p^2}$ et $\sqrt{\nu} \in \mathbb{F}_{p^4}$.

Les équations des fonctions l_1 et l_2, pour $P = (X_P, Y_P, Z_P)$, $Q = (x_Q, y_Q\sqrt{\nu})$ et $T = (X, Y, Z)$ sont :

$$\begin{cases} l_1(x_Q, y_Q\sqrt{\nu}) &= Z_3 Z^2 y_Q\sqrt{\nu} - 2Y^2 - (3X^2 - aZ^4)(x_Q Z^2 - X), \\ \quad \text{avec } Z_3 &= 2YZ \text{ dans l'étape 5 de l'Algorithme 3,} \\ \\ l_2(x_Q, y_Q\sqrt{\nu}) &= Z_3 y_Q\sqrt{\nu} - (Y_P Z^3 - Y Z_P^3)x_Q - (X_p Y Z_p - X Y_P Z), \\ \quad \text{avec } Z_3 &= Z Z_P (X_P Z^2 - X Z_P^2) \text{ dans l'étape 8 de l'Algorithme 3.} \end{cases}$$

En provoquant des erreurs, nous modifions aléatoirement la valeur de r durant l'attaque, nous supposons que l'algorithme de Miller exécute τ itérations. Les points étant d'ordre r, il suffit de s'intéresser aux valeurs de τ telles que $1 < \tau < N$, car $[j + qr]P = [j]P$ pour $q \in \mathbb{N}$.

Première possibilité : $r_{\tau+1} = 0$

Nous connaissons les résultats de l'algorithme de Miller après τ et $(\tau+1)$ itérations, nous les notons $f_{\tau,P}(Q)$ et $f_{\tau+1,P}(Q)$. Nous allons écrire les équations des calculs effectués à la $(\tau+1)$-ième itération.

Durant l'étape 4 de l'algorithme de Miller décrit dans la partie I.8.2, nous calculons $[2j]P = (X_{2j}, Y_{2j}, Z_{2j})$ et stockons le résultat dans la variable intermédiaire T. Les coordonnées de $[2j]P$ sont données par la formule :

$$\begin{cases} X_{2j} &= -8X_jY_j^2 + 9(X_j - Z_j^2)^2(X_j + Z_j^2)^2, \\ Y_{2j} &= 3(X_j - Z_j^2)(X_j + Z_j^2)(4X_jY_j^2 - X_2) - 8Y_j^4, \\ Z_{2j} &= 2Y_jZ_j, \end{cases}$$

où nous notons $[j]P = (X_j, Y_j, Z_j)$.

L'étape 5 de l'algorithme de Miller nous permet d'écrire que

$$f_{\tau+1,P}(Q) = (f_{\tau,P}(Q))^2 \times \left(Z_{2j}Z_j^2 y_Q\sqrt{\nu} - 2Y_j^2 - (3X_j - aZ_j^4)(x_QZ_j^2 - X_j)\right).$$

Comme nous considérons le cas $r_{d+1} = 0$, l'étape d'addition des points P et T n'est pas effectuée. Nous avons à disposition $f_{\tau,P}(Q)$, $f_{\tau+1,P}(Q)$ et le point $Q = (x_Q, y_Q\sqrt{\nu})$, avec x_Q et $y_Q \in \mathbb{F}_{p^2}$. Rappelons que les coordonnées de Q peuvent être librement choisies.

Nous pouvons calculer la valeur du rapport $R = \frac{f_{\tau+1,P}(Q)}{(f_{\tau,P}(Q))^2} \in \mathbb{F}_{p^4}^\star$

$$R = R_3\xi\sqrt{\nu} + R_2\sqrt{\nu} + R_1\xi + R_0,$$

avec $R_3,\ R_2,\ R_1,\ R_0 \in \mathbb{F}_p$.

Le point crucial est que nous connaissons la décomposition théorique de R dans la base $B = \{1, \xi, \sqrt{\nu}, \xi\sqrt{\nu}\}$, cette décomposition dépend des coordonnées de $[j]P$ et Q :

$$R = 2Y_jZ_j^3 y_Q\sqrt{\nu} - Z_j^2(3X_j^2 - aZ_j^4)x_Q - 3X_j(X_j^2 - Z_j^4) - 2Y_j^2.$$

Comme nous connaissons le point $Q = (x_Q, y_Q\sqrt{\nu})$, nous connaissons la décomposition de $x_Q,\ y_Q \in \mathbb{F}_{p^2}$, $x_Q = x_0 + x_1\xi,\ y_Q = y_0 + y_1\xi$, où $(1, \xi)$ définie une base de $\mathbb{F}_{p^2}$. Les valeurs $x_0,\ x_1,\ y_0,\ y_1$ sont connues, de plus $X_j,\ Y_j$, et Z_j sont des éléments de $\mathbb{F}_p$.

Par conséquent, par identification des termes dans la base de $\mathbb{F}_{p^k}$ nous pouvons construire le système V.6.1 liant la valeur trouvée de R et son expression théorique en fonction des coordonnées des points Q et $[j]P$.

$$\begin{cases} 2Y_jZ_j^3y_1 = R_3, \\ 2Y_jZ_j^3y_0 = R_2, \\ (-Z_j^2(3X_j^2 - aZ_j^4))x_1 = R_1, \\ (-Z_j^2(3X_j^2 - aZ_j^4))x_0 - X_j(3X_j^2 - aZ_j^4) - 2Y_j^2 = R_0. \end{cases} \quad \text{(V.6.1)}$$

Le système V.6.1 peut être simplifié pour donner le système V.6.2, où les valeurs des paramètres λ_0, λ_1 et λ_2 sont déterminées connaissant les valeurs x_Q, y_Q et les R_i pour $i = 1, 2, 3$.

$$\begin{cases} Y_j Z_j^3 &= \lambda_2 \\ Z_j^2(3X_j^2 - aZ_j^4) &= \lambda_1 \\ X_j(3X_j^2 - aZ_j^4) + 2Y_j^2 &= \lambda_0 \end{cases} \tag{V.6.2}$$

Nous indiquons les étapes à suivre pour résoudre ce système non linéaire.

La première équation permet d'écrire Y_j comme une fonction de Z_j.

$$Y_j = \frac{\lambda_2}{Z_j^3}.$$

La deuxième équation permet d'exprimer le terme $(3X_j^2 - aZ_j^4)$ en fonction de Z_j.

$$(3X_j^2 - aZ_j^4) = \frac{\lambda_1}{Z_j^2}.$$

Nous substituons l'expression obtenue dans la troisième équation pour exprimer X_j comme une fonction de Z_j.

$$X_j = \frac{\lambda_0 Z_j^6 - 2\lambda_2^2}{\lambda_1 Z_j^4}.$$

En ré-injectant l'expression de X_j trouvée dans la deuxième équation nous obtenons une équation de degré 12 en une seule inconnue, la coordonnée Z_j :

$$(3\lambda_0^2 - a\lambda_1^2)Z^{12} - (12\lambda_0\lambda_2^2 + 3\lambda_1^3)Z^6 + 12\lambda_1^4 = 0$$

Nous savons que ce système admet le point P comme solution. Par conséquent, l'équation obtenue en Z_j admet par construction la coordonnée Z_P comme solution.

Comme l'équation est de degré douze, elle admet dans le pire des cas douze solutions. Afin de retrouver la ou les solutions potentielles, nous pouvons utiliser la fonction `factorff` de PariGP [PAR05], un logiciel de calcul formel. Cette fonction nous renvoie la factorisation dans $\mathbb{F}_p$ du polynôme en Z_j. Les racines de ce polynôme sont les valeurs possibles trouvées pour Z.

A partir de ces solutions potentielles, nous pouvons retrouver les valeurs de X_j et Y_j à l'aide de la première et la deuxième équation du système V.6.2.

La résolution de ce système peut générer au plus $24 = 12 \times 2 \times 1$ triplets de coordonnées (X_j, Y_j, Z_j) pour les coordonnées du point $[j]P$. En pratique, la factorisation du polynôme en Z_j donne au plus huit solutions possibles, un exemple est détaillé dans l'Annexe B.3. Une fois que nous avons obtenu les

triplets potentiels, nous pouvons en éliminer certains en utilisant l'équation de la courbe. Pour trouver les coordonnées possibles de P, nous devons trouver j' l'inverse de j modulo r, afin de multiplier les triplets potentiels par j'. Si un seul triplet passe toutes les étapes, nous aurons retrouvé les coordonnées de P. Si plus d'un triplet est possible pour les coordonnées de P il suffit alors de les tester à travers une exécution de l'algorithme de Miller pour déterminer le bon triplet.

La résolution de ce système nous permet de retrouver les coordonnées du point P dans le cas où $r_{\tau+1} = 0$.

Seconde possibilité : $r_{\tau+1} = 1$

Pour le cas où $r_{\tau+1} = 1$, la $(\tau+1)$-ième itération nécessite une addition pendant l'algorithme de Miller. L'étape de doublement vient d'être décrite. L'étape d'addition fait intervenir les coordonnées du point $[2j+1]P = (X_{2j+1}, Y_{2j+1}, Z_{2j+1})$, avec les notations suivantes $[j]P = (X_j, Y_j, Z_j)$, $[2j]P = (X_{2j}, Y_{2j}, Z_{2j})$ et $P = (X_P, Y_P, Z_P)$.

Nous savons que

$$l_2(x_Q, y_Q\sqrt{\nu}) = Z_{2j+1}y_Q\sqrt{\nu} - (Y_P Z_{2j}^3 - Y_{2j}Z_P^3)x_Q - (X_P Y_{2j} Z_P - X_{2j} Y_P Z_{2j}).$$

Remarquons que seule la coordonnée Z_{2j+1} intervient lors de l'étape 8 de l'Algorithme 3, décrit dans la partie II.2. Or, $Z_{2j+1} = Z_P Z_{2j}(X_P Z_{2j}^2 - X_{2j} Z_P^2)$.

Lors de la $(\tau+1)$-ième itération, nous devons calculer :

$$f_{\tau+1,P}(Q) = (f_{\tau,P}(Q))^2 \times l_1(Q) l_2(Q).$$

Pour le cas $r_{\tau+1} = 1$, le système que nous obtenons par identification dans la base de $\mathbb{F}_{p^k}$ fait intervenir les coordonnées de $[j]P = (X_j, Y_j, Z_j)$ et $P = (X_P, Y_P, Z_P)$. Par identification nous obtenons un système de quatre équations (nous rappelons que nous avons supposé $k = 4$) et six inconnues. En utilisant le fait que P et $[j]P$ sont des éléments de la courbe elliptique, l'équation de la courbe nous donne deux équations de plus pour retrouver les valeurs inconnues. Nous obtenons un système de la forme :

$$\begin{cases} W_1(X_P, Y_P, Z_P, X_j, Y_j, Z_j) = \lambda_1, \\ W_2(X_P, Y_P, Z_P, X_j, Y_j, Z_j) = \lambda_2, \\ W_3(X_P, Y_P, Z_P, X_j, Y_j, Z_j) = \lambda_3, \\ W_4(X_P, Y_P, Z_P, X_j, Y_j, Z_j) = \lambda_4, \\ Y_P^2 - X_P^3 + 3X_P Z_P^4 - bZ_P^6 = 0 \\ Y_j^2 - X_j^3 + 3X_j Z_j^4 - bZ_j^6 = 0 \end{cases} \tag{V.6.3}$$

Nous connaissons l'expression des $W_i()$ pour $i = 1, 2, 3, 4$ qui sont des polynômes en les inconnues. Les valeurs λ_i pour $i = 1, 2, 3, 4$ sont connues et

sont des éléments de $\mathbb{F}_p$. Le Système V.6.3 est un peu plus compliqué que le précédent, mais permet de retrouver directement les coordonnées de P qui en sont solutions. Nous proposons de résoudre ce système en utilisant la méthode des résultants dont le principe est décrit ci après.

Soient $S_1(X,Y)$ et $S_2(X,Y)$ deux polynômes en les inconnues X et Y. Nous pouvons les considérer comme des polynômes en X à coefficients dans $\mathbb{F}_p[Y]$. Le résultant de S_1 et S_2 et alors un polynôme en Y dont les racines sont solutions du système composé par $S_1(X,Y)$ et $S_2(X,Y)$. En appliquant plusieurs fois le résultant entre les équations du système V.6.3, nous obtenons une équation en une seule des coordonnées inconnues. Expérimentalement, l'équation obtenue est de degré 48, et elle possède au plus huit racines dans $\mathbb{F}_p$. Afin d'appliquer le résultant au système V.6.3 nous avons utilisé la fonction `polresultant` de PariGP.

V.6.4 Autres systèmes de coordonnées

Pour ne pas rendre les explications trop indigestes, nous décrivons uniquement le cas où $r_{\tau+1} = 0$. Le cas $r_{\tau+1} = 1$ se traite de la même façon.

Coordonnées affines

Les coordonnées affines pour $k = 2$ ont été étudiées dans les articles [PV06, WS07]. Bien que les courbes telles que $k = 1$ ne soient pas utilisées en cryptographie à base de couplage, notre méthode permettrait de retrouver le secret utilisé lorsque $k = 1$. Comme nous résolvons un système dont le nombre d'équations dépend de k, nous donnons pour chaque résolution la valeur de k minimale pour laquelle nous savons résoudre ce système. Les valeurs de k supérieures permettent de construire un système comportant plus d'équations, et que nous pourrons donc résoudre. Ainsi, puisqu'en coordonnées affines notre attaque fonctionne pour $k = 1$, elle fonctionne pour toutes les valeurs de k.

En effet, le rapport entre deux itérations consécutives de l'algorithme de Miller s'écrit dans ce cas $R = \alpha x_j + \beta y_j + \gamma$, avec α, β et γ connus. En ajoutant l'équation de la courbe elliptique nous obtenons le système :

$$\begin{cases} \alpha x_j + \beta y_j + \gamma &= R \\ y_j^2 &= x_j^3 + a x_P + b \end{cases}$$

Pour ce système, les valeurs de R,α, β, γ, a et b sont connues (a et b correspondent aux paramètres de la courbe). La première équation donne y_j en fonction de x_j et en l'injectant dans la deuxième équation nous obtenons un polynôme de degré trois en x_j. A chaque racine de ce polynôme, nous pouvons associer une valeur de y_j. Nous obtenons au plus trois couples de

solutions possibles. Ainsi, l'attaque par fautes en coordonnées affines permet de récupérer le secret P, et ce pour toutes les valeurs de k possibles.

Coordonnées projectives

L'attaque par injection de fautes en coordonnées projectives fonctionne de la même façon que celle que nous avons décrite pour les coordonnées Jacobiennes. Nous construisons le système V.6.4 en les coordonnées de $[j]P$ par identification des éléments dans la base de $\mathbb{F}_{p^k}$.

$$\begin{cases} Z_j^2 = \lambda_0 \\ Z_j(3X_j + aZ^2) = \lambda_1 \\ X_j(3X_j + aZ_j^2) + Y_jZ_j = \lambda_3 \end{cases}$$

Ce système se résout facilement pour les valeurs de $k \geq 3$. Nous trouvons deux valeurs possibles pour Z_j à l'aide de la première équation, puis deux valeurs possibles pour X_j à l'aide de la deuxième équation ; et enfin quatre valeurs pour Y_j à l'aide de la troisième équation. Au plus, nous obtenons seize triplets possibles. L'équation de la courbe permet d'en filtrer quelques un. S'il en reste plus d'un, des exécutions de l'algorithme de Miller permettront de déterminer le bon.

Si $k = 2$ il est aussi possible de retrouver le secret P en appliquant une méthode de résolution de système utilisant les résultants.

Coordonnées d'Edwards

Les coordonnées d'Edwards sont un nouveau système de coordonnées introduit par H. Edwards en 2007 [Edw07], où l'équation de la courbe elliptique est $x^2 + y^2 = c^2(1 + dx^2y^2)$, avec c et d deux entiers tels que $cd(1 - dc^4) \neq 0$. L'avantage de ces coordonnées est que les formules de la loi de groupe sur la courbe elliptique exprimées en coordonnées d'Edwards sont homogènes et protégées naturellement contre les attaques par consommation de courant. Néanmoins, lorsque les courbes d'Edwards sont utilisées dans un protocole cryptographique, les équations pendant l'exécution du protocole ne sont pas homogènes, et nous pouvons appliquer des attaques par canaux cachés pour retrouver le secret utilisé. L'attaque DPA présentée dans la partie V.5 est efficace pour un algorithme de couplage en coordonnée d'Edwards. Nous montrons que l'attaque par fautes aussi. En utilisant les équations fournies par l'article [IJ08], nous obtenons un système non linaire de k équations grâce à la relation $\frac{l(x,y)}{v(x,y)} = R$, où $l(x, y)$ représente l'équation de la droite intervenant lors du doublement des points, et $v(x, y)$ représente la droite verticale au point courant de l'algorithme de Miller. Nous obtenons un système qui se résout à l'aide de la méthode des résultants, pour les valeurs de k supérieures strictement à deux. Le rapport $\frac{l(x,y)}{v(x,y)}$ est donné par l'équation

V.6.4. Dans cette équation, le point T courant est en coordonnées homogènes, $T = (X_1, Y_1, Z_1)$, l'entier d est l'élément déterminant l'équation de la courbe d'Edwards.

$$A = X_1^2 + Y_1^2 - Z_1^2 \qquad B = X_1^2 - Y_1^2$$

$$C = X_1 Y_1 \quad D = X_1^2 + Y_1^2 \quad E = Z_1^2 - A$$

$$\begin{array}{rcl} [2AB(C(\frac{x}{y} - \frac{y}{x}) - B) & + & DE(dZ_1^2(xy)^2 - A)] \\ \times[4C^2E^2A & + & B^2D^2 - D^2E^2] = \\ R \times 2ABC & \times & [dD^2E^2(xy)^2 - 4C^2E^2 - B^2D^2 + D^2E^2] \end{array}$$

Les éléments $(\frac{x}{y} - \frac{y}{x})$, $(xy)^2$ et R sont des éléments de $\mathbb{F}_{p^k}$, le système à résoudre s'obtient par identification des termes dans la base de $\mathbb{F}_{p^k}$.

Lorsque le secret est le second paramètre du couplage

Si le secret est le second argument du couplage, le point Q dans nos notations, alors quel que soit le système de coordonnées utilisé, tous les systèmes écrits ci dessus deviennent des systèmes linéaires en les coordonnées de Q et se résolvent très facilement par identification dans la base $\mathbb{F}_{p^k}$.

V.6.5 Vulnérabilité des couplages basés sur l'algorithme de Miller

Couplage de Weil

Le couplage de Weil est composé de deux exécutions de l'algorithme de Miller, ce qui le rend directement vulnérable à l'attaque par fautes.

En effet, il est défini par $e_W(P, Q) = \frac{f_{r,P}(Q)}{f_{r,Q}(P)}$.

Nous pouvons raisonnablement supposer que l'implémentation du couplage de Miller est faite de telle sorte que les calculs de Miller Lite et de Miller Full sont effectués en parallèle, ceci permettrait de diviser par deux le temps d'exécution du couplage de Weil. Le même compteur r sera utilisé pour déterminer les itérations à effectuer ; ainsi, si nous modifions la valeur de r alors les exécutions de Miller Lite et Miller Full utiliseront cette même valeur modifiée.

Nous pouvons appliquer l'attaque décrite ci dessus. Soient L_1 et L_2 les équations utilisées pour les étapes 5 et 7 dans le calcul de l'opération Miller Full que nous avons introduite dans la partie I.8.2. Par exemple, $L_1(P)$ est l'équation de la tangente au point T dans Miller Full, où T représente le multiple $[2j]Q$ du point Q pendant l'exécution de l'algorithme de Miller. Le rapport R du résultat de deux itérations consécutives est alors $\frac{l_1(Q)}{L_1(P)} = R$,

nous obtenons par identification des éléments dans la base de $\mathbb{F}_{p^k}$ un système de quatre équations et six inconnues. L'équation de la courbe elliptique nous donne deux équations de plus, ce qui nous permet de résoudre ce système en utilisant la méthode des résultants comme nous l'avons décrit dans la partie V.6.3.

Les couplages de la famille Tate

Nous étudions l'application de l'attaque par fautes pour les couplages de Tate, Ate et Twisted Ate. Ces couplages sont construits sur le même modèle. Leur exécution est constituée d'une exécution de l'algorithme de Miller suivie par une exponentiation à la puissance $\frac{p^k-1}{r}$. La première difficulté que nous devons lever est de trouver une racine $(\frac{p^k-1}{r})$-ième du résultat. Or, il y a au plus $(\frac{p^k-1}{r})$ résultats possibles, ce qui rend impossible toute recherche exhaustive. En effet, même pour les plus bas niveaux de sécurité, cet exposant est de taille 1000 bits au moins. Les auteurs de [WS07] en ont donc déduit que l'exponentiation finale est une contre-mesure naturelle pour les couplages construits sur le modèle du couplage de Tate. Pourtant, il existe de nombreuses méthodes dans la littérature micro-électronique permettant d'arrêter les calculs avant que l'exponentiation finale ne soit exécutée, ou bien de lire le résultat intermédiaire entre l'exécution de l'algorithme de Miller et l'exponentiation, ou tout simplement d'annuler l'étape d'exponentiation.

Nous décrivons l'une d'entre d'elle, l'attaque par scan présentée dans l'article [YWK04]. Cette attaque consiste à lire les états intermédiaires des transistors dans la carte à puce durant un calcul. Ceci est possible en plaçant la carte à puce en mode test. Le principe est de stopper les calculs au beau milieu de leurs exécutions. Une fois la carte stoppée, nous pouvons lancer le scan de ses transistors. Cette opération donne tous leurs états intermédiaires. Donc, en stoppant les calculs juste avant l'exponentiation, une attaque par scan permettrait de lire le résultat intermédiaire à l'issue de l'exécution de l'algorithme de Miller.

D'autres attaques sur le calcul de l'exponentiation existent, elles sont difficiles à mettre en place mais pas irréalisables. Nous pouvons citer par exemple, la technique de réduction de courant de [AK96], ou encore l'attaque par recherche de chaîne d'instructions réalisée par M. Khun et décrite dans [AK96]. Cette attaque consiste à reconnaître les chaînes de codes associées à des instructions précises en observant les conséquences des appels à ces lignes de code.

Une classification des niveaux d'attaque possibles est faite dans [ADDS91]. Chaque niveau correspond à des moyens matériels précis. Le niveau 1 correspond à des attaques menées par de petits attaquants, de manière grossière un hacker dans son garage. Le niveau 2 correspond à des experts dans le domaine, par exemple des universitaires et le niveau 3 (le plus élevé) cor-

respond à de puissantes organisations comme des entreprises ou les armées. L'attaque par fautes que nous décrivons est une attaque d'un niveau 2. Les attaques pour surpasser la protection de l'exponentiation finale pour les couplages sur le modèle du couplage de Tate sont des attaques de niveau 3.

V.6.6 Conclusion et perspectives

Nous venons de présenter une attaque par fautes contre l'algorithme de Miller utilisé pendant un protocole de cryptographie basée sur l'identité.

L'attaque consiste à modifier le compteur interne de l'appareil électrique utilisé, afin de modifier le nombre d'itérations de l'algorithme. Nous décrivons précisément comment réaliser cette attaque par faute. Nous donnons les probabilités d'obtenir deux itérations consécutives et nous prouvons qu'un nombre d'essais raisonnables permet d'obtenir deux nombres d'itérations consécutives.

Nous menons l'attaque dans le cas où le point P, premier argument du calcul, est secret. L'attaque par fautes permet d'obtenir un système non linaire en les coordonnées de P. La résolution de ce système nous permet d'obtenir le point P secret. Si le secret est le second argument du couplage, le point Q, alors l'attaque est facilitée, et permet aussi de retrouver le secret. Quel que soit la position du point à découvrir, notre attaque par fautes permet de le retrouver. Nous l'avons décrite en coordonnées Jacobiennes, mais elle peut être menée dans tous les systèmes de coordonnées existants : affine, projectif ou Edwards.

Le couplage de Weil est directement sensible à cette attaque. Les couplages de Tate, Ate et Twisted Ate sont partiellement protégés par leur exponentiation finale. Nous avons présenté quelques méthodes de micro électronique pour outrepasser l'exponentiation afin d'appliquer notre attaque par faute. L'attaque par scan, l'attaque par sous alimentation électrique ou encore l'attaque par recherche de chaîne d'instruction sont des attaques possibles pour récupérer le résultat de l'algorithme de Miller avant l'exponentiation.

Des contre-mesures matérielles peuvent être mises en place pour protéger la carte à puce ou le matériel électronique des attaques par fautes par émission de lumière ou d'onde électromagnétique, par exemple des boucliers. Il est aussi possible d'ajouter un code de Hamming à la fin du registre pour détecter si une attaque par fautes est en train d'être exécutée [MS98], ou encore d'utiliser une horloge asynchrone pour brouiller les calculs et donc faire en sorte que les fautes ne soient pas analysables. Toutes ces pistes de recherche restent à explorer.

Conclusion

Les couplages sont des outils mathématiques définis à partir de courbes elliptiques et dont l'utilisation en cryptographie ne cesse de se développer. Dans cette thèse nous avons étudié deux aspects de la cryptographie à base de couplages : l'arithmétique des corps finis et les faiblesses de la cryptographie basée sur l'identité face aux attaques par canaux cachés. Nous avons proposé deux optimisations possibles des calculs en améliorant l'arithmétique des corps finis. Puis nous avons étudié les faiblesses des algorithmes de couplages soumis à des attaques à canaux cachés.

Nous avons commencé par comparer les complexités du calcul des couplages de Weil et de Tate dans le Chapitre 2. Ce travail [BE07] nous a permis de décrire les différents algorithmes de multiplication pour deux éléments d'une extension de corps $\mathbb{F}_{p^k}$, et d'introduire les optimisations mathématiques simplifiant les calculs en tirant partie de la structure de groupe des courbes elliptiques. L'utilisation d'une courbe tordue nous permet d'effectuer la majorité des opérations dans un sous corps propre de $\mathbb{F}_{p^k}$. Nos conclusions ont été que pour les degrés de plongement supérieurs à six, le couplage de Tate est toujours plus efficace à calculer que le couplage de Weil. Néanmoins, de nombreuses pistes de recherche ont été soulevées par ce chapitre et restent encore à explorer. Afin de comparer encore plus finement les complexités de calcul, nous devons descendre au niveau des mots machines et jouer sur l'ordre des opérations pour les effectuer de la manière la plus efficace. Par exemple, nous pourrions exécuter la réduction modulaire dans $\mathbb{F}_p$ après une série d'opérations n'augmentant pas beaucoup la taille des entiers. Un exemple typique serait la situation où nous devons additionner les résultats de deux produits. Les étapes de calculs classiques seraient d'effectuer trois réductions, une après chaque produit puis une après l'addition. Nous pourrions nous contenter de n'effectuer qu'une réduction après l'addition des résultats non réduits des produits. Passer de trois à une réduction permettrait de réduire les coûts de calcul et donc les complexités des évaluations. Pour un algorithme aussi complexe que l'algorithme de Miller, alléger la complexité d'une itération est un gain en temps de calcul non négligeable.

Il est intéressant de noter que pour faire leurs comparaisons dans [KM05] et [GPS06] comptent un carré dans $\mathbb{F}_p$ comme un produit dans $\mathbb{F}_p$. Or, il est reconnu que $S = 0,75M$ dans un corps fini, où S représente le coût d'un carré et M celui d'un produit dans le corps fini en question. Ils surestiment donc le coût d'un carré. Inversement, ils estiment qu'une inversion se fait en l'équivalent de dix multiplications, mais il s'agit d'une borne inférieure sachant qu'une inversion est de 9 à 40 fois plus coûteuse en termes d'opérations machines qu'une multiplication. Il serait intéressant de compléter leur travail en descendant au niveau des mots machines pour obtenir des approximations plus réalistes des complexités des opérations dans un corps fini.

De nombreux articles ont pour sujet le calcul des couplages dans des corps finis en caractéristiques 2, 3 ou $p > 3$ [AHM07, GPS05, BBS+07, BDE+09]. Nous pourrions essayer de comparer l'efficacité de ces couplages en différentes caractéristiques. Enfin, la méthode de Lenstra et Stam pour effectuer les carrés dans les corps cyclotomiques n'est valable que pour des extensions de corps finis dont le degré est un multiple de six, réussir à adapter leur méthode pour d'autres degrés d'extension reste un problème ouvert.

Pour les travaux précédents, les multiplications ont été effectuées par des combinaisons de multiplication de Karatsuba et Toom Cook. Ceci était possible car nous travaillions avec des degrés de plongement k de la forme $k = 2^i 3^j$. L'inconvénient de cette écriture est que les valeurs de k ne sont pas très nombreuses. Comment l'arithmétique pour des extensions F_{p^k} pour un degré k quelconque peut être améliorée ? Que se passe-t-il pour les valeurs de k moins adaptées aux multiplications de Karatsuba et Toom Cook ?

Nous avons apporté quelques éléments de réponses dans nos travaux [EN09] et [EGI09].

L'article [EN09] propose une méthode pour effectuer efficacement la multiplication dans une extension $\mathbb{F}_{p^k}$, pour un degré de plongement k quelconque. Notre méthode repose sur l'utilisation de la représentation AMNS. Cette représentation permet de construire une base de $\mathbb{F}_p$ de sorte que le coût d'une multiplication par une racine de l'unité soit négligeable. Cette propriété nous permet d'améliorer l'arithmétique de $\mathbb{F}_{p^k}$ en effectuant la multiplication de deux éléments par des méthodes de DFT et FFT.

Avec notre approche, une multiplication dans $\mathbb{F}_{p^k}$ nécessite moins de multiplications dans $\mathbb{F}_p$ comparativement aux algorithmes classiques de Karatsuba et de Toom Cook. En contre partie, nous augmentons le nombre d'additions. L'étude du ratio entre une addition et une multiplication dans $\mathbb{F}_p$, nous permet de montrer que théoriquement que le surcoût engendré par les additions ajoutées est négligeable comparativement au gain en multiplications.

L'approche que nous avons proposée est donc une amélioration de l'ef-

ficacité de la multiplication d'éléments d'une extension de corps $\mathbb{F}_{p^k}$. La comparaison de notre étude avec les complexités des multiplications de Karatsuba et Toom Cook montre que nos résultats sont intéressants. Nous montrons en particulier que pour des degrés d'extension k supérieurs à 16, la combinaison de la multiplication AMNS et l'approche DFT permet un gain de 50% du nombre de multiplication dans $\mathbb{F}_p$.

Afin de vérifier dans la pratique les gains théoriques que nous obtenons, nous sommes en train d'implémenter en langage C la multiplication que nous proposons. Nous projetons d'utiliser ce programme pour construire un couplage en base AMNS et comparer l'efficacité de notre approche avec les méthodes plus classiques.

Dans notre étude [EGI09], nous nous sommes intéressés à une valeur de degré de plongement plus spécifique. Nous avons étudié le calcul des couplages pour les courbes de degré de plongement 15. Ce degré de plongement nous a permis d'utiliser un twist de degré trois pour procéder à l'élimination des dénominateurs dans l'algorithme de Miller. En appliquant le principe de couplage optimal, consistant à atteindre le nombre minimal d'itérations pour l'algorithme de Miller, nous atteignons bien la borne minimale pour les couplages Ate et Twisted Ate. Nous proposons une arithmétique originale et efficace pour les extensions de degré cinq, où nous construisons sur mesure une multiplication par interpolation sur $\mathbb{F}_{p^5}$. Notre approche consiste en une multiplication par interpolation de Newton, où les divisions exactes nécessaires sont effectuées par des opérations de complexité équivalente à celle d'une soustraction. Cette approche des divisions exactes effectuées en la complexité d'une soustraction a été validé par des expérimentations sur FPGA. Nous avons comparé notre multiplication aux multiplications de Karatsuba et Toom Cook, et nos résultats améliorent les calculs dans $\mathbb{F}_{p^5}$. Nous avons utilisé cette arithmétique pour calculer les couplages Ate et Twisted Ate sur des courbes de degré de plongement 15. Le résultat de nos comparaisons est qu'avec notre approche la complexité du couplage Ate pour les courbes de degré de plongement $k = 15$ nécessite 46% d'opérations en moins pour le niveau de sécurité 192 bits, et 62% pour le niveau de sécurité de 256 bits par rapport au couplage Ate sur les courbes de Barreto-Naehrig. Pour les bas niveaux de sécurité, le couplage sur les courbes $k = 15$ reste plus efficace que pour les courbes de Barreto-Naehrig dont le degré de plongement est 12.

Les perspectives de ce travail sont d'implémenter sur FPGA le calcul des couplages pour les courbes de degré de plongement $k = 15$. L'avantage des FPGA est que nous pouvons optimiser notre division exacte et construire une architecture adaptée à notre arithmétique sur $\mathbb{F}_{p^5}$. Nous cherchons aussi à construire des courbes de degré de plongement $k = 15$ possédant une valeur de ρ plus proche de 1 que les paramétrisations actuelles. Si cette construction

est possible, elle rendrait l'utilisation des courbes avec $k = 15$ compétitive avec les courbes de Barreto-Naehrig pour les bas niveaux de sécurité. De plus, nous devons considérer les courbes dont le degré de plongement est 18 et 24. En effet, ces courbes sont plus adaptées pour les hauts niveaux de sécurité que les courbes de Barreto Naehrig. L'étude de la complexité du calcul des couplages pour les degrés de degré de plongement 15 serait alors complète.

Ces travaux sur l'arithmétique des couplages nous ont permis de nous familiariser avec l'implémentation des couplages, et d'en comprendre les rouages les plus fins. Nous nous sommes alors intéressés à la cryptanalyse des couplages en étudiant les faiblesses de l'algorithme de Miller confronté à des attaques par canaux cachés. Notre intérêt s'est porté sur deux attaques différentes, une attaque différentielle par consommation de courant, et une attaque par injection de fautes. La vulnérabilité de l'algorithme de Miller implique celle des couplages de Weil, de Tate et Ate. Nous avons validé expérimentalement l'attaque différentielle par consommation de courant par des simulations de circuit. Nos résultats montrent que les couplages sont sensibles aux attaques par consommation de courant. De plus, nous avons prouvé que la contre-mesure consistant à mettre le secret utilisé en premier paramètre du couplage n'est pas efficace, puisque nous parvenons à récupérer les coordonnées du point P secret.

De même, cette contre-mesure est inefficace dans le cadre de l'attaque par injection de fautes. Nous avons montré que l'attaque par fautes consistant à modifier le nombre d'itérations de l'algorithme de Miller, pour obtenir le résultat de deux exécutions avec des nombres d'itérations consécutifs, permet de retrouver le secret quel que soit sa position. La vulnérabilité de l'algorithme de Miller à notre attaque par fautes implique directement celle du couplage de Weil. Les couplages de Tate, Ate et Twisted Ate sont plus difficiles à attaquer, mais ce n'est pas irréalisable. Nous avons proposé quelques méthodes pour pouvoir appliquer l'attaque par fautes à ces couplages en contournant l'exponentiation finale. Nous pouvons citer en exemple l'attaque par scan. Les perspectives de ces travaux sont de réaliser pratiquement l'attaque par fautes, et d'étudier sur des FPGA les surcoûts induits par les protections des algorithmes de couplages aux attaques à canaux cachés.

En conclusion, cette thèse nous a permis de découvrir et comprendre les tenants et aboutissants de l'utilisation des couplages en cryptographie. Nous avons proposé des améliorations de l'arithmétique des corps finis afin d'augmenter l'efficacité de l'algorithme de Miller. Nous nous sommes ensuite penchés sur la vulnérabilité de l'algorithme de Miller soumis à des attaques différentielles par consommation de courant et par injection de fautes. Nos travaux ont été concluants et ouvrent plusieurs pistes de recherche. Nous

pouvons citer la recherche de courbes elliptiques minimisant la valeur de ρ, l'implémentation en software et hardware de nos arithmétiques ou encore la réalisation pratique de notre attaque par fautes.

Bibliographie

[ADDS91] D.G. Abraham, G.M. Dolan, G.P. Double, and J.V. Stevens. Transaction security system. In *IBM Systems Journal*, volume 30, pages 206 – 229, Riverton, NJ, USA, 1991. IBM Corp.

[AHM07] O. Ahmadi, D. Hankerson, and A. Menezes. Software Implementation of Arithmetic in $\mathbb{F}_{3^m}$. In *WAIFI '07 : Proceedings of the 1st international workshop on Arithmetic of Finite Fields*, pages 85–102, Berlin, Heidelberg, 2007. Springer-Verlag.

[AK96] R. Anderson and M. Kuhn. Tamper resistance : a cautionary note. In *WOEC'96 : Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 1–11, Berkeley, CA, USA, 1996. USENIX Association.

[ALNR09] C. Arène, T. Lange, M. Naehrig, and C. Ritzenhaler. Faster Pairing Computation. Cryptology ePrint Archive, Report 2009/155, `http://eprint.iacr.org/2009/155`, 2009.

[BBS+07] J.-L. Beuchat, N. Brisebarre, M. Shirase, T. Takagi, and E. Okamoto. A coprocessor for the final exponentiation of the η_T pairing in characteristic three. In C. Carlet and B. Sunar, editors, *Proceedings of Waifi 2007*, number 4547 in Lecture Notes in Computer Science, pages 25–39. Springer, 2007.

[BCF+05] G. M. Bertoni, L. Chen, P. Fragneto, K.A. Harrison, and G. Pelosi. Computing Tate pairing on smartcards. White Paper, ST-Microelectronics. Available : `http://www.st.com/stonline/products/families/smartcard/ches2005_v4.pdf`, 2005.

[BCP97] W. Bosma, J. Cannon, and C. Playoust. The magma algebra system. In *I. The user language. Jounal of Symbolic Computation*, volume 24, 1997.

[BDE+09] J-L. Beuchat, J. Detrey, N. Estibals, E. Okamoto, and F. Rodríguez-Henríquez. Hardware accelerator for the Tate

pairing in characteristic three based on Karatsuba-Ofman multipliers. In C. Clavier and K. Gaj, editors, *Cryptographic Hardware and Embedded Systems – CHES 2009*, number 5747 in Lecture Notes in Computer Science, pages 225–239. Springer, 2009.

[BDEM06] J.-C. Bajard, S. Duquesne, M. Ercegovac, and N. Meloni. Residue systems efficiency for modular products summation : Application to Elliptic Curves Cryptography. In *Advanced Signal Processing Algorithms, Architectures, and Implementations XVI, part of the SPIE Optics Photonics 2006 Symposium (Proceedings of SPIE))*, volume 6313, pages 631304.1–631304.11, August 2006.

[BE07] J.C. Bajard and N. El Mrabet. Pairing in cryptography : an arithmetic point of view. In *Advanced Signal Processing Algorithms, Architectures, and Implementations XVII, part of the SPIE Optics & Photonics 2007 Symposium (Proceedings of SPIE)*, volume 6697, pages 66970O.1–66970O.11, August 2007.

[BF01] D. Boneh and M.K. Franklin. Identity-Based Encryption from the Weil Pairing. In *CRYPTO '01 : Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 213–229, London, UK, 2001. Springer-Verlag.

[BJ03] E. Brier and M. Joye. Fast point multiplication on elliptic curves through isogenies. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, AAECC 2003*, volume 2643, pages 43–50, London, UK, 2003. Springer-Verlag.

[BK98] R. Balasubramanian and N. Koblitz. The improbability that an elliptic curve has sub-exponential discrete log problem under the Menezes Okamoto Vanstone algorithm. In *Journal of Cryptology*, volume 11, pages 141–145. SpringerLink, 1998.

[BKLS02] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient Algorithms for Pairing-Based Cryptosystems. In *CRYPTO '02 : Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology*, pages 354–368, London, UK, 2002. Springer-Verlag.

[BL07] D. J. Bernstein and T. Lange. Performance evaluation of a new side channel resistant coordinate system for elliptic curves. `cr.yp.to/antiforgery/newelliptic-20070410.pdf`, 2007.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *Advances in Cryptology – ASIACRYPT 2001*, volume 2248 of *Lecture Notes in Computer Science*, pages 514–532. Springer Verlag, 2001.

[BLS04a] P. Barreto, B. Lynn, and M. Scott. Efficient Implementation of Pairing-Based Cryptosystems. In *Journal of Cryptology*, volume 17, pages 321–334, Secaucus, NJ, USA, 2004. Springer-Verlag New York, Inc.

[BLS04b] P. Barreto, B. Lynn, and M. Scott. On the Selection of Pairing-Friendly Groups. In Mitsuru Matsui and Robert J. Zuccherato, editors, *SAC 2003 : 10th Annual International Workshop on Selected Areas in Cryptography*, volume 3006 of *Lecture Notes in Computer Science*, pages 17–25. Springer Verlag, 2004.

[BN06] P. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography - SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319 –331. Springer-Verlag, 2006.

[BS04] D. Boneh and H. Shacham. Group Signatures with Verifier-Local Revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 04 : 11th Conference on Computer and Communications Security*, pages 168–177. ACM Press, 2004.

[BSS05] F. Blake, G. Seroussi, and N. Smart. *Advances in Elliptic Curve Cryptography*. London Mathematical Society Lecture Note Series (No. 317). Cambridge University Press, 2005.

[BW05] F. Brezing and A. Weng. Elliptic curves suitable for pairing based cryptography. In *Designs Codes and Cryptography*, volume 37, pages 133–141, Norwell, MA, USA, 2005. Kluwer Academic Publishers.

[CF06] H. Cohen and G. Frey. *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete Math. Appl., Chapman Hall/CRC, 2006.

[CGCDF50] Marchese de'Toschi e di'Sant'Onorio Conte Giulio Carlo Di Fagnano. Benedetto xiv, pontefice massimo, tomo secondo. Produzioni mathematiche, 1750.

[CHNW09] C. Costello, H. Hisil, J.M.G. Nieto, and K.K.H. Wong. Faster Pairings on Special Weierstrass Curves. Pairing 2009, 2009. to appear.

[CP01] C. Cocks and R.G.E. Pinch. Indentity-based cryptosystems based on the Weil pairing. unplublished manuscript, 2001.

[Cra92] R. Crandall. Method and apparatus for public key exchange in a cryptographic system. `http://www.patentstorm.us/patents/5463690.html`, 1992.

[CSB04] S. Chatterjee, P. Sarkar, and R. Barua. Efficient computation of Tate pairing in projective coordinate over general characteristic fields. In *Information Security and Cryptology - ICISC 2004*, volume 3506 of *Lecture Notes in Computer Science*, pages 168–181, Berlin / Heidelberg, 2004. Springer.

[DCC06] P. Duan, S. Cui, and C.W. Chan. Special polynomial families for generating more suitable pairing-friendly elliptic curves. In *EHAC'06 : Proceedings of the 5th WSEAS International Conference on Electronics, Hardware, Wireless and Optical Communications*, pages 187–192, Stevens Point, Wisconsin, USA, 2006. World Scientific and Engineering Academy and Society (WSEAS).

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, volume IT-22, pages 644– 654, IEEE Xplore, Nov 1976. IEEE Transactions on Information Theory.

[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.

[Edw07] H. Edwards. A normal form for elliptic curve. In *Bulletin of the American Mathematical Society*, volume 44, pages 393–422, July 2007.

[EGI09] N. El Mrabet, N. Guillermin, and S. Ionica. Pairing computation for elliptic curves with embedding degree 15. Cryptology ePrint Archive, Report 2009/370, 2009.

[EGT09] A. Enge, P. Gaudry, and E. Thomé. An $l(1/3)$ discrete logarithm algorithm for low degree curves. Hal INRIA, 2009.

[El 85] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 10–18, New York, NY, USA, 1985. Springer-Verlag New York, Inc.

[El 09] N. El Mrabet. What about vulnerability to a fault attack of the Miller's algorithm during an identity based protocol ? In *Advances in Information Security and Assurance*, volume 5576, pages 122–134, London, UK, 2009. Springer-Verlag.

[EN09] N. El Mrabet and C. Nègre. Finite field multiplication combining AMNS and DFT approach for pairing cryptography. In *ACISP '09 : Proceedings of the 14th Australasian conference on Information Security and Privacy*, volume 5594, pages 422–436, London, UK, 2009. Springer-Verlag.

[FR94] G. Frey and H.G. Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of groups. In *Mathematics of Computation*, volume 62, pages 865–874, Boston, MA, USA, 1994. American Mathematical Society.

[FST06] D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. Cryptology ePrint Archive, Report 2006/372, 2006. `http://eprint.iacr.org/`.

[Gal05] S. Galbraith. *Pairings in Advances in Elliptic Curve Cryptography, F. Blake and G. Seroussi and N. Smart editors.* London Mathematical Society Lecture Note Series (No. 317),Cambridge University Press, 2005.

[GPS02] S. D. Galbraith, S. M. Paulus, and N. P. Smart. Arithmetic on superelliptic curves. In *Mathematics of Computation*, volume 71, pages 393–405, Boston, MA, USA, 2002. American Mathematical Society.

[GPS05] R. Granger, D. Page, and M. Stam. Hardware and software normal basis arithmetic for pairing-based cryptography in characteristic three. In *IEEE Trans. Comput.*, volume 54, pages 852–860, Washington, DC, USA, 2005. IEEE Computer Society.

[GPS06] R. Granger, D. Page, and N. P. Smart. High security pairing-based cryptography revisited. In Florian Hess, Sebastian Pauli, and Michael E. Pohst, editors, *ANTS VI*, volume 4076 of *Lectures Notes in Computer Science*, pages 480–494, 2006.

[GPS07] E. Gorla, C. Puttmann, and J. Shokrollahi. Explicit formulas for efficient multiplication in $\mathbb{F}_{3^{6m}}$. In *Selected Areas in Cryptography (SAC'2007)*, volume 4876, pages 173–183, London, UK, 2007. Springer-Verlag.

[Hab92] D.H Habing. The use of lasers to simulate radiation-induced transients in semiconductor devices and circuits. In *IEEE Transactions On Nuclear Science*, volume 39, 1992.

[Har60] B. Harris. Probability distributions related to random mappings. In *Annals of Math. Statistics, 31*, volume 31, pages 1045–1062, 1960.

[Hes08] F. Hess. Pairing Lattices. In Steven Galbraith and Kenny Peterson, editors, *Pairing 2008*, volume 5209 of *Lectures Notes in Computer Science*, pages 18–38, Berlin, Heidelberg, 2008. Springer-Verlag.

[HSV06] F. Hess, N. P. Smart, and F. Vercauteren. The Eta Pairing Revisited. In *IEEE Transactions on Information Theory*, volume 52, pages 4595–4602, 2006.

[IJ08] S. Ionica and A. Joux. Another approach to pairing computation in Edwards coordinates. In *INDOCRYPT '08 : Proceedings of the 9th International Conference on Cryptology in India*, pages 400–413, Berlin, Heidelberg, 2008. Springer-Verlag.

[JL07] A. Joux and R. Lercier. Algorithmes pour résoudre le problème du logarithme discret dans les corps finis. In *Nouvelles Méthodes Mathématiques en Cryptographie*, volume Fascicule Journées Annuelles, pages 23–53. Société Mathématique de France, June 2007.

[JN09] M. Joye and G. Neven. *Identity-Based Cryptography.* vol. 2 of Cryptology and Information Security Series, IOS Press, 2009.

[Jou00] A. Joux. A one round protocol for tripartite Diffie-Hellman. In *ANTS-IV : Proceedings of the 4th International Symposium on Algorithmic Number Theory*, pages 385–394, London, UK, 2000. Springer-Verlag.

[KJJ99] P. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In *Advances in Cryptology Ű Crypto 1999*, volume 1666 of *Lectures Notes in Computer Science*, pages 388–397, Berlin, Heidelberg, 1999. Springer-Verlag.

[KM05] N. Koblitz and A. Menezes. Pairing-based cryptography at high security levels. In Nigel P. Smart, editor, *Cryptography and Coding*, volume 3796 of *Lectures Notes in Computer Science*, pages 13–36, Berlin, Heidelberg, 2005. Springer-Verlag.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. volume 48, pages 203–209. Mathematics of Computation, 1987.

[KSS08] E.J. Kachisa, E. F. Schaefer, and M. Scott. Constructing Brezing-Weng pairing-friendly elliptic curves using elements in the cyclotomic field. In *Pairing '08 : Proceedings of the 2nd international conference on Pairing-Based Cryptography*, pages 126–135, Berlin, Heidelberg, 2008. Springer-Verlag.

[Lan09] T. Lange. Elliptic curves formulas date base `http://www.hyperelliptic.org/EFD/`, 2009.

[LN94] R. Lidl and H. Niederreiter. *Finite Fields*. Cambridge University Press, Cambridge, second edition, 1994.

[LS03] A. Lenstra and M. Stam. Efficient subgroup exponentiation in quadratic and sixth degree extensions. In *CHES '02 : Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*, pages 318–332, London, UK, 2003. Springer-Verlag.

[LZZW08] X. Lin, C. Zhao, F. Zhang, and Y. Wang. Computing the Ate Pairing on elliptic curves with embedding degree k = 9. In *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, volume E91-A, pages 2387–2393, Oxford, UK, 2008. Oxford University Press.

[Mil86a] V. Miller. Short programs for functions on curves. unpublished, 1986. `http://crypto.stanford.edu/miller/miller.pdf`.

[Mil86b] V. Miller. Use of elliptic curves in cryptography. In *Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85*, pages 417–426, New York, NY, USA, 1986. Springer-Verlag New York, Inc.

[Mil04] V. S. Miller. The Weil pairing, and its efficient calculation. In *Jounal of Cryptology*, volume 17, pages 235–261, Secaucus, NJ, USA, 2004. Springer-Verlag New York, Inc.

[Min10] H. Minkowski. *Geometrie der Zahlen*. Leipzig und Berlin, Druck und Verlag von B.G. Teubner, 1910.

[MKHO07] S. Matsuda, N. Kanayama, F. Hess, and E. Okamoto. Optimised versions of the Ate and Twisted Ate pairings. In *Cryptography and Coding*, Lecture Notes in Computer Science, pages 302–312, London, UK, 2007. Springer-Verlag.

[MNT01] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curve traces for FR-reduction. In *IEICE Trans., Fundamentails*, volume E84-A, pages 1234–1243, 2001.

[Mon85] P. L. Montgomery. Modular multiplication without trial division. In *Mathematics of Computation*, volume 44, pages 519–521, Apr 1985.

[MOV91] A. Menezes, T. Okamoto, and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. In *STOC '91 : Proceedings of the twenty-third annual ACM symposium on Theory of computing*, pages 80–89, New York, NY, USA, 1991. ACM.

[MS98] F.J. Macwilliams and N.J.A. Sloane. The theory of error-correcting codes ii. In *North-Holland Mathematical Library*, volume 16, pages i–xv + 1–369, 1998.

[NFR08] G. Di Natale, M.-L. Flottes, and B. Rouzeyre. An integrated validation environment for differential power analysis. In *IEEE International Symposium on Electronic Design, Test & Applications (DELTA 2008), Hong Kong*, pages 527–532, IEEE Computer Science, January 2008.

[NIS07] Recommendations for Key Management, 2007. Special Publication 800-57 Part 1.

[NP08] C. Nègre and T. Plantard. Efficient modular arithmetic in adapted modular number system using Lagrange representation. In *ACISP '08 : Proceedings of the 13th Australasian conference on Information Security and Privacy*, pages 463–477, Berlin, Heidelberg, 2008. Springer-Verlag.

[PAR05] version `2.1.7` PARI/GP. `http://pari.math.u-bordeaux.fr/`, 2005.

[Pau98] S. Paulus. Lattice basis reduction in function fields. In Joe Buhler, editor, *ANTS-III : Proceedings of the Third International Symposium on Algorithmic Number Theory*, pages 567–575, London, UK, 1998. Springer-Verlag.

[PFSR99] C. Paar, P. Fleischmann, and P. Soria-Rodriguez. Fast arithmetic for public-key algorithms in Galois fields with composite exponents. In *IEEE Trans. Comput.*, volume 48, pages 1025–1034, Washington, DC, USA, 1999. IEEE Computer Society.

[Pla05] T. Plantard. *Modular arithmetic for cryptography*. PhD thesis, LIRMM, Université Montpellier 2, 2005.

[Pol78] J. M. Pollard. Monte carlo methods for index computations (mod p). In *Mathematics of Computation*, 1978.

[PV06] D. Page and F. Vercauteren. Fault and side channel attacks on pairing based cryptography. In *IEEE Transactions on Computers*, 2006.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. In *Communications of the ACM*, volume 21, pages 120–126, 1978.

[SB04] M. Scott and P. S. L. M. Barreto. Compressed Pairings. In *In Advances in cryptology - Crypto'2004*, volume 3152, pages 140–156. Springer-Verlag, 2004.

[SBC+09] M. Scott, N. Benger, M. Charlemagne, L. J. Dominguez Perez, and E. J. Kachisa. On the final exponentiation for calculating pairings on ordinary elliptic curves, 2009.

[Sch95] R. Schoof. Counting points on elliptic curves over finite fields. In *Jounal de Theorie des Nombres Bordeaux*, 1995.

[Sco05] M. Scott. Computing the Tate Pairing. In *Topics in Cryptology - CT-RSA 2005*, volume 3376 of *LNCS*, pages 293–304, 2005.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *Proceedings of CRYPTO 84 on Advances in cryptology*, pages 47–53, New York, NY, USA, 1984. Springer-Verlag New York, Inc.

[Sho96] V. Shoup. Number Theory Library, `http://www.shoup.net/ntl`, 1996.

[Sil92] J. H. Silverman. *The arithmetic of elliptic curves*, volume 106 of *Graduate Texts in Mathematics*. Springer-Verlag, New York, 1992. Corrected reprint of the 1986 original.

[TTD+06] K. Tae Hyun, T. Tsuyoshi, H. Dong-Guk, K. Ho Won, and L. Jongin. Side channel attacks and countermeasures on pairing based cryptosystems over binary fields. In *The 5th International Conference on Cryptology and Network Security (CANS 2006)*, volume 4301 of *Lectures Notes in Computer Science*, pages 168–181, Berlin, Heidelberg, 2006. Springer-Verlag.

[Ver08] F. Vercauteren. Optimal Pairings. Cryptology ePrint Archive, Report 2008/096, http://eprint.iacr.org/2008/096, 2008.

[Was03] L. Washington. *Elliptic Curves, Numbers Theory and Cryptography*. Chapmann and all, 2003.

[Wei48] A. Weil. *Courbes algébriques et variétés abéliennes (in french)*. Hermann, 1948.

[WS06] C. Whelan and M. Scott. Side channel analysis of practical pairing implementation : Which path is more secure ? In *VietCrypt 2006*, Lecture Notes in Computer Science, 2006.

[WS07] C. Whelan and M. Scott. The importance of the final exponentiation in pairings when considering fault attacks. In *Pairing-Based Cryptography Ű Pairing 2007*, volume 4575 of *Lecture Notes in Computer Science*, pages 225–246, 2007.

[X9.99] ANSI X9.62-1999. Public key cryptography for the financial services industry : The elliptic curve digital signature algorithm (ecdsa). http://www.techstreet.com/cgi-bin/detail?product_id=1327225, 1999.

[YWK04] B. Yang, K. Wu, and R. Karri. Scan based side channel attack on dedicated hardware implementation of data encryption standard. In *Test Conference 2004, proceedings ITC 2004*, pages 339 – 344, 2004.

[ZG03] J. Von ZurGathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, New York, NY, USA, 2003.

Annexe A

Preuves

A.1 Démonstration de la construction du couplage Ate

A.1.1 Schéma de la preuve

La preuve du Théorème I.9.2 repose sur trois lemmes, que nous démontrons. Le premier établit une relation entre le couplage de Tate et la fonction $f_{T^k,Q}(P)$, en utilisant des propriétés mathématiques de la fonction $f_{r,P}(Q)$ et des diviseurs. Le deuxième repose sur les propriétés des diviseurs et décompose la fonction $f_{T^k,Q}(P)$ en un produit des puissances des $f_{T,[\lambda]Q}(P)$, pour λ un entier en relation avec les puissances intervenant. Enfin, le troisième permet d'exprimer les puissances des $f_{T,[\lambda]Q}(P)$ en fonction de la fonction $f_{T,Q}(P)$ en utilisant l'application Frobenius sur la courbe elliptique.

Lemme A.1.1. *Avec les notations de la section I.9.1*

$$e_T(Q,P)^L = f_{T^k,Q}(P)^{(p^k-1)/N}$$

Lemme A.1.2. *L'égalité des diviseurs suivante est satisfatite :*

$$f_{T^k,Q} = f_{T,Q}^{T^{k-1}} \times f_{T,[T]Q}^{T^{k-2}} \times \ldots \times f_{T,[T^{k-1}]Q}$$

Lemme A.1.3. $\forall Q \in G_2,\ i \in \mathbb{N}\ f_{T,\pi_p^i(Q)} = f_{T,Q}^{\pi_p^i}$, *où* π_p *est le Frobenius à la puissance* p *sur* $\overline{\mathbb{F}_p}$.

Preuve du théorème

Supposons que nous avons démontré les Lemmes A.1.1, A.1.2 et A.1.3. Nous décrivons comment l'emboîtement de ces lemmes permet de démontrer le Théorème I.9.2. L'égalité du Lemme A.1.3 peut aussi s'écrire

$f_{T,\pi_p^i(Q)} = f_{T,[T^i](Q)} = f_{T,Q}^{\pi_p^i}$. En effet, par définition de G_2, $\pi_p(Q) = [p]Q$, or modulo le cardinal de la courbe elliptique $p = t-1 = T$ modulo r. D'après la définition de $\pi_p(Q) = [T]Q$; nous obtenons par récurrence immédiate que $\pi_p^i(Q) = [T^i]Q$.

En injectant le résultat du Lemme A.1.3 dans l'égalité du Lemme A.1.2 nous obtenons les équations suivantes :

$$\begin{cases} f_{T^k,Q} &= f_{T,Q}^{T^{k-1}} \times f_{T,[T]Q}^{T^{k-2}} \times \ldots \times f_{T,[T^{k-1}]Q} \\ &= f_{T,Q}^{T^{k-1}} \times f_{T,Q}^{T^{k-2}p} \times \ldots \times f_{T,Q}^{p^{k-1}} \\ &= f_{T,Q}^{c} \\ \text{avec } c &= \sum_{i=0}^{k-1} T^{k-1-i}p^i. \end{cases}$$

De plus, nous savons que $T \equiv p \bmod(r)$ ce qui permet d'écrire que $c \equiv \sum_{i=0}^{k-1} p^{k-1-i}p^i \equiv kp^{k-1} \bmod(r)$.

Cette égalité permet de démontrer le Théorème I.9.2. en utilisant le Lemme A.1.1.

$$\begin{cases} e_T(Q,P)^L &= f_{T^k,Q}(P)^{(p^k-1)/N} \\ &= f_{T,Q}(P)^{c(p^k-1)/N} \\ \text{avec } c &= \sum_{i=0}^{k-1} T^{k-1-i}p^i. \end{cases}$$

Nous allons maintenant donner des démonstrations pour les Lemmes A.1.1, A.1.2 et A.1.3.

A.1.2 Démonstration des lemmes

Démonstration du Lemme A.1.1

Nous rappelons la définition du couplage de Tate :

$$e_T(Q,P) = f_{r,Q}(P)^{(p^k-1)/r}.$$

Cette définition nous servira de point de départ pour construire l'égalité du Lemme A.1.1.

Nous rappellons une propriété intéressante démontrée par Galbraith Paulus et Smart dans [GPS02].

Propriété A.1.4. *Pour tout entier N tel que r divise N qui divise (p^k-1), l'égalité ci dessous est vraie :*

$$f_{r,Q}(P)^{(p^k-1)/r} = f_{N,Q}(P)^{(p^k-1)/N}$$

Soit N le pgcd de (T^k-1) et (p^k-1), alors nous savons que N vérifie la condition $r|N|(p^k-1)$. En effet, si
$N = \text{pgcd}(T^k-1, p^k-1)$, alors par définition du plus grand commun diviseur de (T^k-1) et (p^k-1), n'importe quel autre diviseur commun de (T^k-1) et (p^k-1) est un diviseur de N. Or, r est un diviseur du cardinal de la courbe elliptique, par conséquent $p \equiv t-1 \bmod r$, et donc $p \equiv T \bmod r$. Nous savons que r divise (p^k-1), nous en déduisons que r divise (T^k-1). r est donc un diviseur commun à (T^k-1) et (p^k-1), N étant le plus grand commun diviseur de ces valeurs cela impose que r divise N. Ce qui permet d'écrire l'égalité $e_T(Q,P) = f_{N,Q}(P)^{(p^k-1)/N}$. En élevant cette égalité à la puissance L nous obtenons $e_T(Q,P)^L = f_{N,Q}(P)^{L(p^k-1)/N}$, qui nous donne le membre de gauche de l'égalité que nous souhaitons prouver.

Sachant que $T^k-1 = LN$, nous aurons démontrer le Lemme A.1.1 si nous parvenons à montrer l'égalité de fonction suivante $f_{N,Q}^L = f_{LN,Q}$.

Propriété A.1.5. *Soit N et L deux entiers, alors l'égalité suivante est vraie : $f_{N,Q}^L = f_{LN,Q}$.*

Pour montrer cette égalité de fonction, nous allons montrer l'égalité des diviseurs associés à chacune des fonctions. L'égalité des diviseurs nous donne l'égalité des fonctions à multiplication par des constantes de $\mathbb{F}_p$ près, comme nous l'avons souligné dans la remarque I.6.17. Les résultats étant élevés à la puissance $\frac{p^k-1}{r}$, les constantes sont neutralisées par l'exponentiation.

Démonstration. Soit $f_{N,Q}$ une fonction dont le diviseur est
$div(f_{N,Q}) = N(Q) - ([N]Q) - (N-1)(P_\infty)$, le diviseur $div(f_{N,Q})$ peut se simplifier comme suit :

$$\begin{cases} div(f_{N,Q}) &= N(Q) - ([N]Q) - (N-1)(P_\infty) \\ &= N(Q) - (P_\infty) - (N-1)(P_\infty) \text{ car } [N]Q = P_\infty \\ &= N(Q) - N(P_\infty) \end{cases}$$

Nous pouvons alors en déduire le diviseur de la fonction $f_{N,Q}^L$.

$$\begin{cases} div(f_{N,Q}^L) &= L \times div(f_{N,Q}) \\ &= LN(Q) - LN(P_\infty) \\ &= LN(Q) - ([LN]Q) - (N-1)(P_\infty) \text{ car } [LN]Q = [L][N]Q \\ &= div(f_{LN,Q}) \end{cases}$$

A partir de l'égalité des diviseurs, nous pouvons déduire l'égalité des fonctions $f_{N,Q}^L = f_{LN,Q}$.

□

La proprité A.1.5 nous donne l'égalité $e_T(Q,P)^L = f_{T^k-1,Q}(P)^{(p^k-1)/N}$. En effet,

$$\begin{aligned} e_T(Q,P)^L &= f_{N,Q}(P)^{(p^k-1)/N} \\ &= f_{LN,Q}(P)^{(p^k-1)/N} \\ &= f_{T^k-1,Q}(P)^{(p^k-1)/N}. \end{aligned}$$

Pour terminer la démonstration du Lemme A.1.1, nous n'avons plus qu'à démontrer l'égalité $f_{T^k-1,Q} = f_{T^k,Q}$, à l'aide de diviseurs.

$$\begin{aligned} div(f_{T^k-1,Q}) &= (T^k-1)(Q) - (T^k-1)(P_\infty) \text{ car } r|(T^k-1) \\ div(f_{T^k,Q}) &= T^k(Q) - ([T^k]Q) - (T^k-1)(P_\infty) \\ \text{or } [T^k]Q &= (Q) \text{ sur } E \text{ car } T^k \equiv 1 \bmod (r) \\ div(f_{T^k,Q}) &= T^k(Q) - (Q) - (T^k-1)(P_\infty) \\ div(f_{T^k,Q}) &= div(f_{T^k-1,Q}) \end{aligned}$$

Ceci conclut la démonstration du Lemme A.1.1.

Démonstration du Lemme A.1.2

Cette démonstration repose elle aussi uniquement sur les propriétés des diviseurs.

L'égalité de fonctions : $f_{T^k,Q} = f_{T,Q}^{T^{k-1}} \times f_{T,[T]Q}^{T^{k-2}} \times \ldots \times f_{T,[T^{k-1}]Q}$ se démontre en prouvant que l'égalité est vraie pour les diviseurs. Le point important de cette démonstration est que le diviseur associé à un produit de fonction se décompose en la somme des diviseurs des fonctions. Ainsi :

$$div(f_{T,Q}^{T^{k-1}} \times f_{T,[T]Q}^{T^{k-2}} \times \ldots \times f_{T,[T^{k-1}]Q}) =$$

$$div(f_{T,Q}^{T^{k-1}}) + div(f_{T,[T]Q}^{T^{k-2}}) + \ldots + div(f_{T,[T^{k-1}]Q})$$

Nous pouvons décomposer chacun des diviseurs intervenants en utilisant la définition des fonctions $f_{T,[T^i]Q}$:

$$div(f_{T,[T^i]Q}^{T^{k-i}}) = T^{k-i}\left(T([T^i]Q) - ([T][T^i]Q) - (T-1)(P_\infty)\right).$$

En substituant ces égalités de diviseurs dans l'égalité du lemme A.1.2, des simplifications en diagonales de produisent et le diviseur $div(f_{T,Q}^{T^{k-1}} \times f_{T,[T]Q}^{T^{k-2}} \times \ldots \times f_{T,[T^{k-1}]Q})$ se réduit tout simplement à $div(f_{T^k,Q})$. L'égalité de fonction suit comme nous l'avons expliqué lors de la démonstration du Lemme A.1.1.

Démonstration lemme A.1.3

Nous voulons démontrer l'égalité de fonctions :

$$\forall Q \in G_2,\ i \in \mathbb{N}\ f_{T,\pi_p^i(Q)} = f_{T,Q}^{\pi_p^i},$$

où π_p est le Frobenius sur $\overline{\mathbb{F}_p}$, la cloture algébrique de $\mathbb{F}_p$.

Cette égalité repose essentiellement sur le petit théorème de Fermat. La fonction $f_{T,Q}$ est une fonction rationnelle en les coordonnées du point P auquel on évalute la fonction, et les coefficients de la fonction rationnelle dépendent des coordonnées de Q et des coefficients de la courbe. Les coefficients de la courbe et les coordonnées du point P sont des éléments de $\mathbb{F}_p$. Par conséquent, d'après le petit théorème de Fermat, la fonction Frobenius agit comme l'identité sur ces éléments. Donc $f_{T,Q}^{\pi_p^i}$ correspond juste à la fonction à laquelle on applique le Frobenius aux coordonnées de Q, soit $f_{T,\pi_p^i(Q)}$.

La démonstration du Lemme A.1.3 achève la démonstration du Théorème I.9.2.

A.2 Construction de la formule de probabilité pour l'attaque par faute

Nous allons montrer comment construire la formule pour donner les probabilités de tirer n entiers parmi N sans qu'aucun ne soit consécutifs.

A.2.1 Construction de la formule

Nous tirons donc n entiers i_j parmis N, dans l'ensemble $[1, N]$. En rangeant ces entiers par ordre croissant, nous pouvons construire une liste $(i_0, k_0, i_1, k_1, \ldots, i_m, k_m)$ avec pour tout j, $i_j \in [1, N]$, et k_j représente le nombre de fois où l'entier i_j est tiré.

Une fois que nous avons tiré le plus grand entier i_m il nous reste à tirer $n - k_m$ entiers dans l'ensemble $[1, i_m - 2]$. Nous devons considérer les entiers compris entre $[1, i_m - 2]$ car nous cherchons tout d'abord le cardinal de l'évènement aucun entier tiré n'est le consécutif d'un autre.

Ainsi, nous obtenons la formule :

$$B(n, N) = \sum_{j=1}^{N} \sum_{k=1}^{n} B(n-k, j-2)$$

Normalisation :

1. $N \leq 0, n > 0$: nombre de possibilités nul.

2. N quelconque, $n = 0$: 1 possibilité (il y a une unique application de l'ensemble vide vers n'importe quel ensemble).

A.2.2 Procédure Maple

```
> B := proc(n, N)
  option remember;
  local S, j, k;
  if n = 0 then return 1;
  elif N < 1 then return 0;
  else
    S := 0;
    for j from 1 to N do
      for k from 1 to n do
        S := S + B(n-k, j-2);
      od;
    od;
    return S;
  fi;
  end proc;
```

Nombre d'appels :

- $B(20, 5)$: 2302 sans `remember`, 101 avec `remember`
- $B(20, 10)$: 295830 sans `remember`, 201 avec `remember`

A.2.3 Tests

Cette procédure calcule toutes les probabilités de ne pas avoir deux entiers successifs pour des tirages de 1 à n entiers :

```
> with(plots);
  with(stats);
  with(stats[statplots]);

  N := 256;
  n := 20;

  s := [seq(Weight(k..k+1, evalf(B(k, N)/binomial(k+N, k))), k=1..n)];
  histogram(s);
```

A.2.4 Calcul pour des petites valeurs de n

- $n = 1$: N possibilités.
- $n = 2$: N possibilités de tirer 2 fois le même entier. Si les deux sont différents, on place le plus grand j et il y a $k-2$ possibilités de choisir le second. Au total :

$$\sum_{j=3}^{N}(j-2) = \sum_{j=1}^{N-2} j = \frac{(N-2)(N-1)}{2}\text{possibilités.}$$

  Finalement, il y a :

$$N + \frac{(N-2)(N-1)}{2} = \frac{N^2-N+2}{2}\text{possibilités.}$$

- $n = 3$:
  - Trois fois le même : N possibilités.
  - Deux fois le plus grand (j) : $\sum_{j=3}^{N}(j-2) = \frac{(N-2)(N-1)}{2}$
  - Une fois le plus grand : $\sum_{j=3}^{N} \frac{(j-2)^2-(j-2)+2}{2} = \frac{N^3-6N^2-17N-18}{6}$.

  En additionnant, on a donc :

$$\frac{(N-1)(N^2-2N+12)}{6}\text{possibilités.}$$

Annexe B

Exemples

B.1 Exemple de base AMNS

Nous donnons dans cette partie des exemples de courbe de degré $6 < k \leq 32$ pour lesquelles nous appliquons la multiplication DFT. Nous rappelons brièvement la méthode décrite dans [NP08] pour construire une base AMNS pour un nombre premier p fixé. Nous choisissons d'abord un polynôme $P_{\mathcal{B}}(\delta) = \delta^{\ell} - \lambda$ de degré ℓ et nous calculons γ une racine de $P_{\mathcal{B}}$ dans $\mathbb{F}_p$. Puis nous construisons la matrice M suivante.

$$M = \begin{bmatrix} p & 0 & 0 & \cdots & 0 \\ -\gamma & 1 & 0 & \cdots & 0 \\ -\gamma^2 & 0 & 1 & \cdots & 0 \\ \vdots & & & \ddots & \vdots \\ -\gamma^{(\ell-1)} & 0 & \cdots & 0 & 1 \end{bmatrix}$$

L'application de l'algorithme LLL à cette matrice nous permet d'obtenir un court vecteur $\mathfrak{m}$ satisfaisant l'égalité $\mathfrak{m}(\gamma) = 0 \mod p$. Nous obtenons au final $\rho = 2\ell|\lambda|\|\mathfrak{m}\|_{\infty}$.

B.1.1 Courbes de degré $k = 12$

Nous utilisons la paramétrisation de Barreto et Naehrig [BN06] qui permet de construire des courbes elliptiques de degré $k = 12$.

$$\begin{aligned} k &= 12, \\ p &= 36x^4 + 36x^3 + 24x^2 + 6x + 1, \\ r &= 36x^4 + 36x^3 + 18x^2 + 6x + 1, \\ t &= 6x^2 + 1. \end{aligned}$$

Nous utilisons le polynôme $P_{\mathcal{B}}(\delta) = \sum_{i=0}^{10} (-t)^i$ pour construire la base AMNS associée au premier p.

Pour un niveau de sécurité de 80 bits, c'est à dire qu'un attaquant doit mener au moins 2^{80} opérations sur le corps fini pour réussir une attaque brute force, nous avons trouvé les exemples suivants :

$$\begin{aligned}
x &= 1099511637026,\\
p &= 526140607144920699926592600935421554404299113222 53,\\
r &= 526140607144920699926592528399871157068636666574197,\\
t &= 725355503973356624474 8057,\\
\gamma &= 1434862295316848707004673170099045197398534 8345534,\\
\mathfrak{m}(\delta) &= 12376 - 49167\delta + 48460\delta^2 + 18281\delta^3 + 15213\delta^4\\
&\quad -10299\delta^5 + 11263\delta^6 - 70120\delta^7 - 13636\delta^8 - 18106\delta^9.
\end{aligned}$$

Pour un niveau de sécurité de 160 bits :

$$\begin{aligned}
x &= 18446744073709692895,\\
p &= 41685152125435107379370057363152675115521120051443074 05\\
&\quad 27638681004597639694997 1,\\
r &= 41685152125435107379370057363152675115500703109427817 43\\
&\quad 22195589052592511606382 1,\\
t &= 20416942015256620544309195200512808861 51,\\
\gamma &= 37771108084317046107302986198165197419883853864047699 31\\
&\quad 7641286502190684739139,\\
\mathfrak{m}(\delta) &= 8053715 - 20923230\delta + 23417521\delta^2 - 26826999\delta^3\\
&\quad +19243643\delta^4 + 1059907\delta^5 - 41954237\delta^6 - 42180723\delta^7\\
&\quad +5371359\delta^8 - 19196965\delta^9.
\end{aligned}$$

B.1.2 Courbes de degré $k = 16$

Nous utilisons la paramétrisation de [KSS08] :

$$
\begin{aligned}
k &= 16,\\
p &= (x^{10}+2x^9+5x^8+48x^6+152x^5+240x^4+625x^2+2398x\\
&\quad +3125)/980,\\
r &= (x^8+48x^4+625)/61250,\\
t &= (2x^5+41x+35)/35.
\end{aligned}
$$

Nous utilisons le polynôme $P_{\mathcal{B}}(\delta) = \delta^{16}+1$.

Pour un niveau de sécurité de 80 bits nous trouvons les exemples suivants :

$$
\begin{aligned}
x &= 74156485,\\
p &= 513174771603192518069857791127277415092088396567880595361684047893395993456 1,\\
r &= 149309347072601793039402841900662885259628529084818905369993,\\
t &= 128146760584932038247348983414439772062,\\
\gamma &= 386968286582177389475518658240604863510095482299776733841319721386977404674,\\
\mathfrak{m}(\delta) &= 7400\delta+49262\delta^2-3010\delta^3-14335\delta^4+34360\delta^5\\
&\quad +43021\delta^6+6813\delta^7+5184\delta^8+13206\delta^9+10037\delta^{10}\\
&\quad +2540\delta^{11}-7384\delta^{12}-66117\delta^{13}-57557\delta^{14}+32450\delta^{15}.
\end{aligned}
$$

Pour un niveau de sécurité de 160 bits :

$$
\begin{aligned}
x &= 300650886015,\\
p &= 615742037941290064431987586434433942899976129045006271675961520936707961430145111275245127149377594529712107468 9,\\
r &= 108991796562856949188268637826460013069843005574422145615453925993037858204960705875499 3,\\
t &= 140370029614552009401267496693144064107592433030506438440,\\
\gamma &= 551737151471267013665906013312108810638488413639246814149706947418249015319821682977158544996914977939922628908,\\
\mathfrak{m}(\delta) &= 11792 + 15441\delta - 25387\delta^2 + 11348\delta^3 + 20103\delta^4 + 25605\delta^5\\
&\quad -8716\delta^6 + 9091\delta^7 + 19039\delta^8 + 13855\delta^9 - 22021\delta^{10}\\
&\quad -15182\delta^{11} - 4543\delta^{12} + 1417\delta^{13} - 26776\delta^{14} + 11502\delta^{15}.
\end{aligned}
$$

B.2 Complexité des couplages pour les courbes $k = 18$ et $k = 24$

B.2.1 Courbes $k = 18$

Pour le niveau de sécurité de 192 bits les courbes les plus adaptées semblent être les courbes telles que $k = 18$ et $\rho = 1,333$. Elles sont paramétrées par

$$
\begin{aligned}
p &= \frac{1}{21}(x^8 + 5x^7 + 7x^6 + 37x^5 + 188x^4 + 259x^3 + 343x^2 + 1763x + 2401),\\
r &= x^6 + 37x^3 + 343,\\
t &= \frac{1}{7}(x^4 + 16x + 7).
\end{aligned}
$$

Les valeurs de x congru à 14 modulo 42 permettent d'obtenir des valeurs de $p(x)$ premier.Ces courbes avec $k = 18$ sont obtenues à l'exemple 6.12 de [FST06]. Elles permettent d'utiliser un twist de degré 6, et donc de ramener la plupart des calculs dans les corps $\mathbb{F}_p$ et $\mathbb{F}_{p^3}$.

Le calcul des couplages Ate et Twisted Ate nécessite de faire des opérations dans $\mathbb{F}_p$ et $\mathbb{F}_{p^3}$. Les complexités sont rappelées dans le tableau suivant.

Le nombre d'itérations pour le couplage Ate est donné par t la trace du Frobenius. Pour un niveau de sécurité de 192 bits, r doit avoir une taille de

Ate	$4M_p + 5S_{p^3} + 3M_{p^3} + S_{p^{18}} + M_{p^{18}} \approx 174M_p$
Twisted Ate	$5S_p + 15M_p + S_{p^{18}} + M_{p^{18}} \approx 154M_p$

FIGURE B.1 – Complexité Ate et Twisted Ate $k = 18$

384 bits. La donnée de $\rho = 1,333$ permet d'en déduire la taille de p qui est 427 bits, et celle de p^k : 9216 bits. La trace t est de taille 256 bits. Sans application de la notion de couplage optimal, le couplage Ate nécessite donc 256 itérations, et le Twisted Ate en nécessite $3 \times 256 = 768$.

Au total le calcul du couplage pour un niveau de sécurité de 192 bits a les complexités suivantes pour les courbes $k = 18$.

Ate	24 320 M_p
Twisted Ate	118 212 M_p
Tate	59 136 M_p

FIGURE B.2 – Coût en multiplication Ate et Twisted Ate $k = 18$

Théoriquement, si l'on applique la notion de couplage optimale, alors le couplage Ate se calcul en $O(r^{1/6})$ itérations, et le couplage Twisted Ate en $O(r^{1/2})$ itérations. Le couplage Ate pour $k = 18$ demanderait alors 11 $136M_p$, et le couplage Twisted Ate 29 $568M_p$.

B.2.2 Courbes $k = 24$

Pour le niveau 256, les courbes les plus adaptées sont les courbes telles que $k = 24$ et $\rho = 1,25$. Elles peuvent etre construites par la méthodes 6.6 de l'article Taxanomy de Freeman et sont paramétrées par

$$\begin{aligned} p &= \frac{1}{3}((x-1)^2(x^8 - x^4 + 1) + x, \\ r &= \Phi_{24}(x) = x^8 - x^4 + 1, \\ t &= x + 1. \end{aligned}$$

Le polynome $r(x)$ est de degré 8 car $\varphi(24) = 8$. Nous pouvons utiliser un twist de degré 6 pour cette famille de courbe et ainsi ramener la plupart des calculs intervenant dans les couplages à des calculs dans les corps $\mathbb{F}_p$ et $\mathbb{F}_{p^4}$.

Le calcul des couplages Ate et Twisted Ate nécessite de faire des opérations dans $\mathbb{F}_p$ et $\mathbb{F}_{p^4}$. Les complexités sont rappelées dans le tableau suivant.

Le nombre d'itérations pour le couplage Ate est donné par t la trace du Frobenius. Pour un niveau de sécurité de 256 bits, r doit avoir une taille de 512 bits. La donnée de $\rho = 1,25$ permet d'en déduire la taille de p qui est 427 bits, et celle de p^k 15 360 bits. La trace t est de taille 64. Le couplage Ate nécessite donc 64 itérations, et le Twisted Ate en nécessite $4 \times 64 = 256$.

Ate	$4M_p + 5S_{p^4} + 3M_{p^4} + S_{p^{24}} + M_{p^{24}} \approx 148M_p$
Twisted Ate	$5S_p + 15M_p + S_{p^{24}} + M_{p^{24}} \approx 100M_p$

FIGURE B.3 – Complexité Ate et Twisted Ate $k = 124$

Au total le calcul du couplage pour un niveau de sécurité de 256 bits a les complexités suivantes pour les courbes $k = 24$.

Ate	9 472 M_p
Twisted Ate	25 600 M_p

FIGURE B.4 – Coût en multiplication Ate et Twisted Ate $k = 24$

B.3 Exemple d'attaque par faute

Afin d'illustrer les calculs à faire pour l'attaque par faute, nous détaillons un exemple. Les calculs sont faits avec PariGP.

$k = 4$
$p = 680241220348515477477949259894191902369939655391504568151207016994661689050587617052536187229749$ (319 bit)

$E : Y^2 = X^3 + 3XZ^4$

$card(E(\mathbb{F}_p)) = 680241220348515477477949259894191902369939655390338170945836123217606411022317222264735061564936$ (319 bit)
$r = 1166397205370893777055276948271688598347500051217$ (160 bit)

$P = [12, 48, 2]$

Nous construisons $\mathbb{F}_{p^k}$ en considérant un entier $a \in \mathbb{F}_{p^k}$ tel que $a^4 = 2$.

$Q = [a^2, 100512916629999457534083547932541900367294743582692206264363320753064855041994266311971573488636 \times a]$
Nous arrêtons l'algorithme de Miller au bout de 46 itérations.

La rapport R est
$337259586468080683488399539046229874795973242322339094577672485334431934756557508827480079490557 \times a^2 +$
$624752062739857009467545836695395127071983321507188174321543153770228940196002139337802972603156 \times a +$
$290466295014915698560156774394046481806928474873516631676810692056670$

491562068356785654141784610 3

Nous obtenons le système suivant :

$$\begin{cases} Y_j Z_j^3 = \lambda_2 = \\ 526421537150286598896703298483149985967580207398544590133171776285079049014186714839235255813297 \\ 3Z_j^2(X_j^2 - Z_j^4) = \lambda_1 = \\ 475148309417543639369621311346136013833460391400891264127160029381884835668719747434801612007813 \\ 3X_j(X_j^2 - Z_j^4) - 2Y_j^2 = \lambda_0 = \\ 389774925333599778917792485500145420563011180517987936474396324937986773429904049195994769383646 \end{cases}$$

$$(\lambda_0^2 - 9\lambda_1^2)Z^{12} - (4\lambda_0\lambda_2^2 + 9\lambda_1^3)Z^6 + 4\lambda_1^4 = 0$$

La fonction `factorff(f(Z),p)` de PariGP nous donne six solutions différentes en Z.

$[Mod(166072817587205561850915284007509914230746546515582328522504517934359493296306626253218820537301, p), Mod(186129439623952388290499049901750065782778388777506228654648579224687036546331407445908022796608, p), Mod(328038963137357527336534925984931922356414720098416010974053919835615159207949583353409343895840, p), Mod(352202257211157950141414333909259980013524935293088557177153097159046529842638033699126843333909, p), Mod(494111780724563089187450209992441836587161266613998339496558437769974652504256209606628164433141, p), Mod(514168402761309915627033975886681988139193108875922239628702499060302195754280990799317366692448, p)]$

Les six points que l'on en déduit appartiennent à la courbe elliptique.

Un simple test nous permet de retrouver le bon triplet de coordonnées pour le point P.