Université de la Méditerranée, Aix-Marseille 2
École Doctorale en Mathématiques et Informatique E.D. 184

THÈSE

présentée pour obtenir le grade de
Docteur de l'Université Aix-Marseille
Spécialité : Informatique

par

Alexandre VENELLI

Contribution à la sécurité physique des cryptosystèmes embarqués

Soutenue le 31 Janvier 2011

Composition du jury

Rapporteurs :	Marc JOYE	HDR, Cryptographe, Technicolor
	David NACCACHE	Professeur Univ. Paris II & ENS
Président du jury :	Traian MUNTEAN	Professeur Univ. de la Méditerranée
Directeur de thèse :	Alexis BONNECAZE	Professeur Univ. de la Méditerranée
Examinateurs :	Vincent DUPAQUIS	Cryptographe, Inside Secure
	Pierre LIARDET	Professeur Univ. de Provence
	François-Xavier STANDAERT	Professeur UCL

Remerciements

Je remercie en premier lieu mon directeur de thèse Alexis Bonnecaze pour ses conseils et tout son temps passé à me guider dans la bonne direction au cours de ces trois années. Je dois énormément à Alexis qui a su m'insuffler l'envie et le plaisir d'effectuer de la recherche. J'adresse un remerciement spécial à Vincent Dupaquis, mon responsable scientifique chez Inside Secure, pour tout son soutien. Vincent m'a toujours laissé une grande liberté dans l'organisation de mon travail et je l'en remercie grandement. J'exprime aussi ma reconnaissance à Traian Muntean qui est à l'origine de cette thèse. Traian est un formidable chef d'équipe qui m'a beaucoup aidé et conseillé au cours de ces trois années. Ce fut un réel plaisir de travailler au quotidien aux côtés de personnes aussi compétentes, attentionnées et disponibles.

Je remercie de même les membres de mon jury. Merci à Marc Joye et David Naccache d'avoir accepté de rapporter mon travail. C'est pour moi un réel honneur d'avoir des références incontestables dans leurs domaines juger mon travail de thèse. Je suis très reconnaissant à François-Xavier Standaert pour avoir accepté de faire partie de mon jury et ainsi évaluer mes travaux. Je souhaite remercier vivement Pierre Liardet pour son aide dans mes travaux de recherches ainsi que pour sa gentillesse et sa disponibilité. J'ai effectué cette thèse en étant rattaché à deux équipes de recherches le groupe ATI de l'IML ainsi que le groupe ERISCS. Je tiens à remercier chacun des membres de ces équipes. En particulier François Rodier, responsable de l'équipe ATI et Gilles Lachaud, directeur de l'IML. J'aimerais remercier Robert Rolland et Yves Aubry qui ont aussi contribué à la réussite de cette thèse.

J'ai passé de très bons moments au sein de l'équipe cryptographie d'Atmel, puis récemment d'Inside Secure. J'ai une sympathie particulière à l'égard de Michel Douguet, Azeddine Hossayni, Miguel Garcia et Shivam Bhasin qui y ont grandement contribués. Je remercie spécialement François Dassance qui a beaucoup participé à la réussite de cette thèse grâce à nos nombreuses discussions scientifiques. En plus d'être un formidable coauteur, François est aussi devenu un véritable ami.

Je me dois aussi de remercier Nadia, Victor, Tony, Laurent, Fabien, Alban, Johann, l'équipe appli d'Atmel, tous les bus de la France entière, ma marraine électronique, Apple mais uniquement pour son iPod, les Princes et l'Orangina, les boards de développement STK600 morts pour cette thèse, les oscilloscopes du passé et du futur, les Google Fights pour la correction orthographique et les portables DELL.

Pour terminer, je tiens à remercier spécialement les personnes qui m'ont soutenu et aidé tout au long de ces années : mes parents, mes oncles et tantes, mon cousin, ma grand-mère et le reste de ma famille. Merci énormément pour tout.

Résumé

La cryptographie à base de courbes elliptiques (ECC) est de plus en plus utilisée dans les cryptosystèmes à clé publique. Son principal avantage est, qu'à niveau de sécurité égal, la taille des clés est beaucoup plus petite par rapport aux cryptosystèmes asymétriques classiques tel que RSA. Des clés plus petites signifient une consommation de courant réduite, moins de calculs à effectuer et moins d'espace de stockage nécessaire. L'ECC est ainsi devenue indispensable sur des composants cryptographiques embarqués tels que les cartes à puces. Étant donné la faible puissance de calculs de ces systèmes, de nombreuses recherches sont effectuées afin d'améliorer l'efficacité de l'ECC. Outre les performances, la sécurisation de ces composants embarqués est un autre problème majeur. Les cryptosystèmes, notamment l'ECC, qui sont considérés comme mathématiquement sûrs, ne le sont plus forcément une fois les algorithmes implémentés en pratique. En effet, les composants électroniques tels que les cartes à puces interagissent et sont influencés par l'environnement dans lequel ils se trouvent. Ces interactions peuvent être contrôlées par un attaquant pour réaliser des attaques dites par canaux cachés ou canaux auxiliaires.

Ces travaux de thèse se concentrent sur l'étude des attaques par canaux cachés et les implications sur les mesures à prendre pour un concepteur de circuits sécurisés. Nous nous intéressons d'abord aux différentes attaques par canaux cachés en proposant une amélioration pour un type d'attaque générique particulièrement intéressante : l'attaque par analyse d'information mutuelle. Nous étudions l'effet des différentes techniques d'estimation d'entropie sur les résultats de l'attaque. Nous proposons l'utilisation de fonctions B-splines comme estimateurs étant donné qu'elles sont bien adaptées à notre scénario d'attaques par canaux cachés. Nous étudions aussi l'impact que peut avoir ce type d'attaques sur un cryptosystème symétrique connu, l'Advanced Encryption Standard (AES), en proposant une contre-mesure basée sur la structure algébrique de l'AES. L'opération principale de la majorité des systèmes ECC est la multiplication scalaire qui consiste à additionner un certain nombre de fois un point de courbe elliptique avec lui-même. Dans une deuxième partie, nous nous intéressons à la sécurisation de cette opération. Nous proposons un algorithme de multiplication scalaire à la fois efficace et résistant face aux principales attaques par canaux cachés. Nous étudions enfin les couplages, une construction mathématique basée sur les courbes elliptiques, qui possède des propriétés intéressantes pour la création de nouveaux protocoles cryptographiques. Nous évaluons finalement la résistance aux attaques par canaux cachés de ces constructions.

Abstract

Elliptic Curve Cryptography (ECC) is used increasingly in public key cryptosystems. Its main advantage is that, at a given security level, key sizes are much smaller compared to classical asymetric cryptosystems like RSA. Smaller keys imply less power consumption, less cryptographic computation and also requiere less memory. Implementing ECC is therefore becoming essential in constrained devices like smart cards. As these systems have small computational power, numerous activities are performed in order to improve the efficiency of ECC algorithms. Besides performance, security is another major problem in embedded devices. Cryptosystems, like ECC, that are considered mathematically secure, are not necessarily considered safe when implemented in practice. Indeed, electronic components like smart cards interact and are perturbed by the environment in which computations are performed. These interactions can be monitored by an attacker in order to mount attacks called side-channel attacks.

This thesis focuses on the study of side-channel attacks as well as their consequences on the secure implementation of cryptographic algorithms. We first analyze different side-channel attacks and we propose an improvement of a particularly interesting generic attack : the mutual information analysis. We study the effect of state of the art entropy estimation techniques on the results of the attack. We propose the use of B-spline funtions as estimators as they are well suited to the side-channel attack scenario. We also investigate the consequences of this kind of attack on a well known symmetric cryptosystem, the Advanced Encryption Standard (AES), and we propose a countermeasure based on the algebraic structure of AES. The main operation of ECC is the scalar multiplication that consists of adding an elliptic curve point to itself a certain number of times. In the second part, we investigate how to secure this operation. We propose a scalar multiplication algorithm that is both efficient and secure against main side-channel attacks. We then study pairings, a mathematical construction based on elliptic curves. Pairings have many interesting properties that allow the creation of new cryptographic protocols. We finally evaluate the side-channel resistance of pairings.

Table des matières

Table des figures

Liste des tableaux

Liste des Algorithmes

Introduction

De plus en plus d'objets de notre vie quotidienne possèdent une certaine puissance de calcul. Des produits tels que téléphones portables, voitures, cartes de transport et bien d'autres effectuent des calculs que seuls des ordinateurs pouvaient réaliser il a quelques années. Des composants parfois encore plus simples ou même jetables, comme des cartouches d'imprimantes, ont un microprocesseur embarqué. Tous ces objets ont souvent la capacité de communiquer les uns avec les autres à travers différents réseaux. Une contrepartie de cette interconnectivité est la présence grandissante de vulnérabilités dans ces objets. Des attaques qui n'affectaient que des ordinateurs personnels s'appliquent maintenant à des tickets de transports ou des véhicules. De plus, les utilisateurs de ces composants ne comprennent pas forcément le besoin de sécurité dans ces objets de tous les jours. Ils ont une tolérance bien plus forte vis à vis de la sécurité quand il s'agit de leur ordinateur personnel alors que les risques sont tous aussi importants.

La cryptographie propose un certain nombre de mécanismes de sécurité, notamment pour les composants embarqués. Néanmoins, l'ajout de cryptographie augmente la complexité, les coûts et la consommation de ces microcontrôleurs. Une implémentation efficace de ces méthodes cryptographiques, pour un niveau de sécurité important et un surcoût acceptable, est un des objectifs majeurs de nombreux industriels de ce secteur. On distingue dans la cryptographie deux grands types de constructions avec la cryptographie symétrique et la cryptographie asymétrique. La cryptographie moderne est marquée par la proposition d'une méthode de chiffrement de données basée sur un schéma de Feistel : le *Data Encryption Standard* (DES) [Nat93]. Il a été remplacé quelques années plus tard par un nouveau standard plus robuste basé sur une structure algébrique : l'*Advanced Encryption Standard* (AES) [Nat01]. Ces deux méthodes de chiffrement font partie des cryptosystèmes symétriques car deux parties désirant communiquer doivent partager un même secret, appelé clé secrète, pour chiffrer et déchiffrer des messages. Ce type de cryptosystème est très efficace en terme de performance mais effectue néanmoins une hypothèse forte : les deux parties doivent se mettre d'accord au préalable sur une même valeur secrète. Une deuxième contrainte des systèmes symétriques est que chaque utilisateur doit disposer des clés secrètes de toutes les personnes avec qui il souhaite communiquer. En pratique, cela peut poser des problèmes pour stocker de manière sécurisée des clés secrètes lorsque de très grands groupes d'utilisateurs veulent communiquer les uns avec les autres.

Une solution à ce problème est proposée en 1976 par Diffie et Hellman [DH76] grâce à l'introduction du concept de cryptosystème asymétrique. Dans ce système, un utilisateur A possède une clé publique qu'il transmet à tous ses interlocuteurs pour leur permettre de lui envoyer des messages. Il a aussi une clé privée, connue uniquement de lui, qui permet de déchiffrer les messages qui lui sont destinés. Les deux contraintes majeures des cryptosystèmes symétriques sont résolues par la cryptographie asymétrique. Néanmoins, alors que la cryptographie asymétrique offre de nouvelles possibilités, elle a des performances nettement moins intéressantes que la cryptographie symétrique. Ainsi, en pratique on se sert de systèmes asymétriques pour échanger des clés entre utilisateurs, effectuer des signatures électroniques de documents ou s'authentifier. Le chiffrement et déchiffrement de conversations reste effectué par un système symétrique. De nombreux cryptosystèmes asymétriques basent leur sécurité sur la difficulté mathématique de résoudre le problème du logarithme

discret (*Discrete Logarithm Problem*, DLP). Des recherches sur la difficulté de ce problème ont montré que le DLP sur un groupe d'entiers modulo un nombre premier peut être résolu en un temps sous-exponentiel [LLMP90]. Le problème est donc difficile à résoudre mais moins difficile qu'un problème exponentiel. Pour conserver un même niveau de sécurité, un cryptosystème asymétrique basé sur un problème sous-exponentiel doit utiliser des clés de longueur plus grande qu'un système basé sur un problème exponentiel.

En 1985, Koblitz [Kob87] et Miller [Mil86b] proposent indépendamment le principe de cryptographie asymétrique basée sur des courbes elliptiques, aussi appelé cryptosystème à base de courbes elliptiques (*Elliptic Curve Cryptosystem*, ECC). La sécurité de ces systèmes repose sur le même problème du DLP mais sur des points d'une courbe elliptique au lieu de grands entiers. Le problème est alors appelé problème du logarithme discret basé sur les courbes elliptiques (*Elliptic Curve Discrete Logarithm Problem*, ECDLP). Après plus de 20 années de recherches, l'algorithme le plus rapide pour résoudre l'ECDLP reste de complexité exponentielle. Pour un même niveau de sécurité, on peut utiliser des clés plus petites par rapport aux cryptosystèmes asymétriques classiques utilisant des groupes de nombres. L'ECC est donc particulièrement intéressante pour une implémentation sur des composants embarqués disposant de peu de mémoire. De plus, des clés plus petites signifient moins de calculs et donc plus de performances. L'utilisation de courbes elliptiques dans les cryptosystèmes asymétriques devient de plus en plus importante notamment dans les systèmes embarqués.

Parmi les nombreuses recherches effectuées sur les courbes elliptiques, certains se sont intéressés à la sécurité d'ECC utilisant des courbes elliptiques avec une structure particulière. Un couplage est justement une fonction qui a été utilisée pour attaquer des ECC utilisant des courbes cryptographiquement faibles. Le couplage prend en argument deux points de courbe elliptique et sort un grand entier. De plus, il possède des propriétés très spéciales, la plus intéressante étant la bilinéarité. De part ses propriétés particulières, il est difficile de construire une fonction de couplage. On recensait jusqu'à peu seulement deux familles de couplages : le couplage de Weil et le couplage de Tate. Menezes et al. [MOV93] ont été les premiers à utiliser les couplages en cryptographie. Ils montrent que le problème de l'ECDLP peut être résolu en temps sous-exponentiel lorsque des courbes elliptiques particulières sont utilisées pour un système ECC.

Joux [Jou00] est le premier en 2000 à proposer un système cryptographique utilisant des couplages avec un protocole d'échange de clés entre trois personnes en un seul tour. Un protocole de ce type était jusqu'alors impossible à réaliser sans les propriétés des couplages. De très nombreux articles ont suivi afin d'exploiter ces propriétés pour des applications nouvelles. En 2001, Boneh et Franklin [BF01] proposent une solution à un problème posé par Shamir en 1984 : créer un cryptosystème où la clé publique est directement liée à l'identité des utilisateurs. L'intérêt premier d'un tel cryptosystème est d'éviter la gestion coûteuse des clés publiques. Avec le chiffrement basé sur l'identité (*Identity-Based Encryption*, IBE), Boneh et Franklin ont initié une multitude de recherches dans le domaine de la cryptographie basée sur les couplages. On peut trouver un état de l'art de certains de ces cryptosystèmes dans [BSS05, Chapitre X]. Depuis peu, des recherches sont effectuées concernant l'efficacité de la construction des couplages avec des améliorations aux

couplages de Weil et Tate.

Faire de la cryptographie de manière efficace n'est qu'un aspect de la sécurité d'un composant embarqué. De part leur nature, un attaquant peut facilement avoir un accès physique à ces objets. Une nouvelle classe d'attaques, qui n'avaient pas de sens sur des ordinateurs personnels, a vu le jour. L'accès physique offre beaucoup plus de possibilités à l'attaquant. Il peut enregistrer le comportement du composant ou même modifier physiquement le composant pendant que celui-ci effectue des opérations cryptographiques. En mesurant, par exemple, la consommation de courant du système, l'attaquant peut déduire des informations sur les données manipulées par les fonctions cryptographiques et donc déduire des informations sur le secret. La consommation de courant ou encore les émissions électromagnétiques d'un composant sont appelées canaux cachés ou canaux auxiliaires. Cette classe d'attaques est donc souvent appelée attaques par analyse des canaux cachés. De nombreuses contre-mesures ont depuis été proposées dans la littérature afin de protéger les différentes fonctions cryptographiques. Néanmoins, elles coûtent souvent beaucoup en termes de performance, de consommation de courant et de taille de code si la fonction est implémentée en *software*. Il est donc important de trouver un compromis entre performance et sécurité.

Cette thèse est composée de trois parties. La première partie introduit le concept central d'attaques par canaux cachés. Elle se divise en six chapitres. Le premier décrit une catégorisation classique de ces attaques en attaques actives et attaques passives. Nous commençons par les attaques par injection de faute et décrivons ensuite les principales techniques d'attaques par canaux cachés. Les différentes méthodes d'injections ainsi que les modèles de fautes les plus utilisés en pratique sont introduits. Dans le chapitre suivant, on s'intéresse à un canal caché intéressant et facilement utilisable : la consommation de courant d'un composant embarqué. Nous décrivons brièvement le principe de la technologie *Complementary Metal Oxyde Semiconductor* (CMOS) dont est composée la plupart des systèmes embarqués. Cette technologie possède des propriétés de consommation de courant qui sont utilisées pour des attaques sur les composants CMOS. En effet, celle-ci varie en grande partie en fonction de la valeur des données traitées par le microprocesseur à un instant donné. Cette modélisation très simple du courant en fonction des données permet d'effectuer des attaques. Ce canal caché particulier est détaillé car nous l'utilisons au travers des attaques par canaux cachés que nous avons réalisées. Nous présentons ensuite les principales attaques passives dont l'attaque par analyse simple. Nous nous intéressons plus en détail à l'attaque par analyse différentielle, plus dangereuse en pratique dans la plupart des cas. Au centre de cette attaque se trouve un test statistique qui joue un rôle majeur dans son efficacité. Nous analysons donc les principaux tests proposés dans la littérature. Nous nous intéressons aux attaques différentielles dans le cas spécifique du cryptosystème symétrique AES. Nous proposons une contre-mesure basée sur la structure algébrique de cet algorithme qui permet de se protéger efficacement contre les attaques du premier ordre. Dans le dernier chapitre, nous nous concentrons sur l'attaque différentielle utilisant l'information mutuelle comme test statistique. Elle est particulièrement intéressante car elle offre

une approche plus générique par rapport aux attaques différentielles. Nous rappelons son principe ainsi qu'un problème majeur dans son efficacité : le choix de la méthode d'estimation d'information mutuelle. Plus l'estimation est juste plus l'attaque produit des résultats meilleurs. Néanmoins, il faut différencier deux types d'estimations : les estimateurs paramétriques et non-paramétriques. L'estimation paramétrique donne de très bons résultats mais oblige l'attaquant à faire plus de suppositions sur le composant attaqué. D'un autre côté, l'estimation non-paramétrique semble légèrement moins efficace mais permet de conserver le principe originel d'attaque générique de l'attaque par analyse d'information mutuelle. Nous étudions donc en particulier ces estimateurs non-paramétriques. Nous proposons notamment l'utilisation de l'estimateur non-paramétrique à base de B-splines qui offre de très bons résultats dans le cadre des attaques par canaux cachés. Nous comparons finalement les performances des principales attaques différentielles sur différentes plateformes.

Dans la partie II, nous nous intéressons aux ECC et particulièrement à leur opération centrale, la multiplication scalaire. Un premier chapitre introduit les préliminaires nécessaires pour comprendre les courbes elliptiques et leur utilisation en cryptographie. Dans le chapitre suivant, nous étudions les algorithmes classiques de multiplication scalaire. Pour chacun des types d'attaques par canaux cachés introduits dans le chapitre précédent, nous détaillons les attaques proposées dans le cadre de la sécurisation de la multiplication scalaire. Nous catégorisons d'abord différents types d'attaques par injection de fautes qui impliquent des stratégies de protection différentes. Nous montrons ensuite la dangerosité de l'attaque par analyse simple sur un algorithme de multiplication scalaire classique. Nous détaillons les principaux types d'attaques différentielles sur ECC. Certaines sont particulièrement adaptées aux courbes elliptiques en utilisant leur structure mathématique pour trouver des failles. D'autres, plus génériques, s'appliquent aussi aux cryptosystèmes asymétriques classiques. Le chapitre suivant présente les principales protections proposées dans la littérature. Ces contre-mesures sont souvent adaptées pour se protéger contre un type particulier d'attaque par canaux cachés. Un concepteur de circuits sécurisés doit donc choisir un ensemble de contre-mesures afin d'avoir une protection efficace. Dans le dernier chapitre, nous proposons un algorithme de multiplication scalaire à la fois efficace et résistant par construction à certaines attaques par canaux cachés. Notre proposition est basée sur un algorithme de multiplication classique, appelé l'échelle de Montgomery, largement étudié dans la littérature. En modifiant légèrement sa structure, nous pouvons l'utiliser conjointement avec la formule d'addition simplifiée de Méloni [Mel07] qui est très performante. Nous proposons plusieurs versions de notre algorithme en fonction de la taille de code désirée par le concepteur de circuits. Nous obtenons l'algorithme de multiplication scalaire le plus performant de la littérature pour le niveau de sécurité considéré.

La partie III s'intéresse à des fonctions mathématiques introduites relativement récemment en cryptographie, les couplages. Bien que les couplages offrent des possibilités de protocoles inédits, leur complexité de calcul reste élevée pour être implémentés en pratique sur des composants embarqués. Cette partie est composée de quatre chapitres. Le premier introduit les préliminaires mathématiques nécessaires à la construction de couplages. Le deuxième chapitre présente les deux principales constructions de couplages, le couplage de Weil et de Tate. Nous détaillons les propriétés de ces constructions et l'algorithme de

Miller permettant de les calculer. L'algorithme de Miller a une structure similaire à l'algorithme de multiplication scalaire de doublement-et-addition. Nous mentionnons ensuite les paramètres à prendre en compte afin d'avoir un couplage d'un certain niveau de sécurité. De nombreuses recherches ont été effectuées afin d'améliorer l'efficacité de l'algorithme de Miller. Nous présentons dans le chapitre suivant les dernières constructions de couplages efficaces proposées dans la littérature. Certaines sont adaptées à des cas spécifiques comme le couplage eta très efficace sur des courbes elliptiques définies sur des corps de caractéristique 2 ou 3. D'autres auteurs ont proposé des constructions plus génériques notamment avec la notion de couplage optimal. La plupart de ces méthodes tentent de réduire le nombre de tours de boucle de l'algorithme de Miller qui représente une grande partie de la complexité d'un couplage. Dans le dernier chapitre, nous nous intéressons aux attaques par canaux cachés dans le cadre des couplages. Le sujet est récent et relativement peu de littérature est disponible. Le scénario d'attaque est très différent de celui utilisé pour attaquer un ECC. Alors que dans une multiplication scalaire le secret correspond à la valeur du scalaire, dans un couplage le secret est un point de courbe elliptique. Les premières attaques par canaux cachés proposées sur couplages sont des attaques par injection de fautes. Nous détaillons ces attaques qui s'appliquent à la plupart des constructions de couplages. Nous nous intéressons ensuite plus particulièrement aux attaques par analyse différentielle et notamment aux travaux de El Mrabet [EMDNF09]. La section suivante présente une attaque différentielle, réalisée sur circuit, d'une implémentation standard de couplage. Ces travaux sont le résultat d'une collaboration avec Nadia El Mrabet et Victor Lomné. Nous détaillons la procédure d'attaque et confirmons la possibilité d'une attaque quelle que soit la position du point secret dans les arguments du couplage. Il avait été précédemment proposé dans la littérature de placer le secret comme premier argument du couplage de certains couplages afin d'éviter les attaques différentielles. Nous listons finalement les méthodes de protections contre les attaques par injection de fautes et par analyse différentielle proposées dans la littérature qui permettent d'obtenir une implémentation sûre.

Première partie

Attaques par canaux cachés

Chapitre 1

Classification des attaques par canaux cachés

Les algorithmes cryptographiques qui sont considérés comme sûrs grâce à des preuves mathématiques, ne le sont plus forcément une fois l'algorithme implémenté en pratique. Ces algorithmes sont codés dans des composants qui interagissent et sont influencés par l'environnement dans lequel ils se trouvent. Les composants électroniques, comme les cartes à puces, consomment du courant et émettent des radiations lors de leur fonctionnement. Ces composants réagissent aussi aux changements de température et aux différents champs magnétiques dans lesquels ils se trouvent. Ces interactions dues à l'environnement peuvent être contrôlées et enregistrées par un attaquant pour réaliser des attaques par canaux cachés (*Side-Channel Analysis*, SCA).

Depuis quelques années, il existe de nombreuses propositions d'attaques par canaux cachés. Suivant le scénario considéré et la puissance supposée d'un attaquant, ces attaques peuvent être classifiées suivant différents critères.

1.1 Attaques actives

Les attaques actives peuvent être sous catégorisées en trois parties, les attaques : invasives, semi-invasives et non-invasives. Nous détaillons ensuite les attaques par injection de fautes qui constituent une des principales attaques actives.

1.1.1 Invasives

Une attaque est invasive lorsque l'attaquant a la possibilité d'altérer de manière physique, souvent irrémédiable, le composant. Il peut ainsi avoir un accès plus direct au système cryptographique. Une attaque invasive commence souvent par la décapsulation du boîtier du circuit intégré, technique appelée *depackaging* en anglais (Figure 1.1). Le boîtier sert de protection pour le circuit imprimé et sert aussi pour la dissipation thermique du composant. On utilise souvent de l'acide nitrique fumant pour dissoudre le boîtier sans endommager le

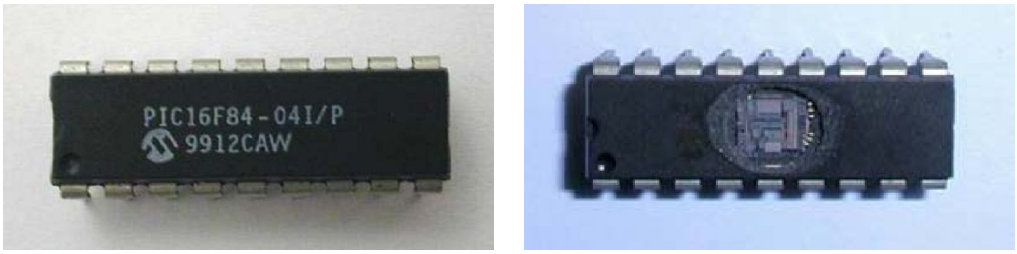


FIGURE 1.1 – Microcontrôleur PIC16F84 avec et sans boîtier [SA03].

circuit situé dessous. Grâce à une *probing station*, l'attaquant peut alors avoir accès à différents composants du circuit et observer directement les signaux transportant les données. Les attaques invasives sont très efficaces mais requièrent souvent une compétence élevée de la part de l'attaquant ainsi que du matériel onéreux [KK99, Sko05].

1.1.2 Semi-invasives

Pour ce type d'attaque, le composant est aussi décapsulé de son boîtier. Néanmoins, l'attaquant n'effectue aucun contact direct avec la surface du circuit intégré. On classe dans les attaques semi-invasives les techniques permettant simplement de lire le contenu d'une cellule mémoire sans utiliser de *probing* [SSAQ02]. Certains scénarios précis d'attaques par injection de faute sont aussi considérés comme semi-invasifs [SA03, Sko05]. Ce type d'attaques requiert moins de compétences et matériels que les invasives. Mais il n'est pas évident de repérer, sur des circuits modernes, à quel endroit il est préférable de se positionner pour réaliser une attaque.

1.1.3 Non-invasives

On considère pour ce type d'attaque que le composant ne subit aucunes modifications physiques définitives. Ces attaques demandent peu de matériel et posent donc un problème important pour la sécurité des composants. Le but de ces attaques est d'injecter une faute et perturber le fonctionnement du système par des moyens extérieurs : des pics de tension sur l'alimentation ou changements de température par exemple [BECN$^+$06].

1.1.4 Attaques par injection de fautes

L'attaque par injection de faute est une attaque active qui modifie le comportement d'un composant, soit de manière définitive, soit de manière transitoire. Si un attaquant est capable de modifier le contenu d'un espace mémoire ou l'exécution d'une opération au cours d'un algorithme cryptographique, des erreurs dans le calcul se produiront sûrement. Si un résultat final erroné est fourni à l'attaquant et que celui-ci dépend de la valeur du secret, il se peut qu'il obtienne des informations sur celui-ci en comparant avec un résultat correct de la même opération. La première attaque par injection de faute a été présentée

en 1997 par Boneh et al. [BDL97] sur l'algorithme de Rivest Shamir Adelman utilisant le thèorème des restes chinois (*RSA with Chinese Remainder Theorem*, RSA-CRT).

1.1.4.1 Méthodes d'injection

Il existe de nombreuses manières de produire des fautes sur un circuit électrique [BECN+06]. Nous détaillons brièvement quelques méthodes les plus utilisées.

Radiations lumineuses. En 2002, Skorobogatov et Anderson [SA03] proposent d'utiliser un rayon de lumière concentré pour induire des fautes. Un simple flash d'appareil photo est utilisé comme source de lumière blanche. Les auteurs se servent ensuite d'un microscope et d'une feuille d'aluminium pour concentrer le rayon. L'attaque leur permet de modifier le contenu de bits choisis dans la mémoire SRAM. La puce doit être au préalable décapsulée afin que le rayon lumineux atteigne la zone d'intérêt. Néanmoins l'attaque est très puissante puisque l'attaquant a le contrôle sur les bits de son choix en mémoire.

Pics de courant ou de tension (*power spikes*). La source d'alimentation d'une carte à puce provient d'un lecteur qui peut être aisément remplacé par un attaquant. Il peut alors contrôler l'alimentation de la carte à puce. De grandes variations de courant dans de brefs intervalles de temps peuvent être envoyées à la carte. Ces pics de courant induisent soit des erreurs mémoires, soit des modifications dans l'exécution du code. Ces dernières peuvent permettre de modifier un compteur de boucle ou un saut conditionnel dans le code. On trouve dans la littérature de nombreuses attaques par fautes utilisant cette technique [BDL97, BS97, BDL01, ABF+02, BS03, YMH03].

Modifier la fréquence de l'horloge du système (*clock glitches*). Comme pour l'alimentation en courant, les cartes à puces n'ont pas leur propre signal d'horloge. Le lecteur de carte fournit le signal, ainsi l'attaquant a la possibilité de l'altérer. Les cartes à puces fonctionnent souvent à la fréquence de 3,57 Mhz. De grandes variations en fréquence dans de brefs délais peuvent être envoyées à la puce pour perturber son fonctionnement. Ces perturbations sont similaires à celles occasionnées par des pics de courant. Cette méthode, comme la précédente, est l'une des plus simple et est donc très utilisée en pratique [AK98, BMM00, PV04].

1.1.4.2 Caractérisation des fautes

Il existe de nombreuses propositions d'attaques par injections de fautes dans la littérature. Elles se classent en différentes catégories suivant les critères détaillés ci-dessous. Pour chacun des paramètres considérés, nous décrivons les scénarios du plus permissif au plus strict pour l'attaquant.

Contrôle spatial. L'attaquant doit avoir un certain contrôle sur l'endroit où doit avoir lieu la faute. On considère généralement les scénarios suivants :

- l'attaquant peut seulement modifier une variable donnée,
- l'attaquant ne peut modifier que des bits précis de cette variable,
- l'attaquant n'a aucun contrôle sur la variable qui sera affectée.

Contrôle temporel. Le contrôle temporel sur l'injection de l'erreur est un paramètre crucial. On considère les scénarios suivants :

- l'attaquant peut choisir exactement à quel moment injecter la faute,
- la faute peut affecter un bloc de quelques opérations,
- l'attaquant n'a aucun contrôle temporel.

Nombre de bits. L'attaquant ne peut pas forcément contrôler le nombre de bits qui sont affecté par l'injection de faute. On considère les scénarios suivants :

- l'attaquant est capable de modifier un seul bit,
- l'attaquant est capable de modifier un seul octet,
- l'attaquant est capable de modifier un nombre inconnu de bits dans une variable.

Probabilité de la faute. L'injection d'une faute n'a pas forcément d'effet sur le circuit, ou bien n'a pas l'effet escompté par l'attaquant. Pour prendre en compte la dangerosité d'une attaque par faute, il faut donc considérer la probabilité de réussite de la faute. Par exemple si l'attaque requiert une injection à un endroit précis de la mémoire mais que l'attaquant ne peut qu'injecter des fautes dans un endroit aléatoire, cela affecte la probabilité de réussite de l'attaque.

Durée de la faute. Une faute peut affecter un circuit plus ou moins longtemps. On considère trois types de fautes suivant leur durée : transitoire, permanente ou destructrice.

Une faute transitoire n'a qu'un court effet dans le temps avant que le circuit ne reprenne son comportement normal. Généralement, ce laps de temps est considéré très court. Ainsi, si une faute transitoire est injectée dans une variable lors d'un calcul, lors de sa prochaine manipulation la variable aura retrouvée sa valeur originale.

Une faute permanente affecte une variable jusqu'à ce que l'algorithme se termine, ou qu'elle soit explicitement écrasée par une autre valeur au cours de l'exécution.

Une faute destructrice détruit définitivement une partie du circuit pour fixer certains bits ou une variable à une valeur donnée.

Effet sur les bits. Une faute peut affecter un ensemble de bits en mémoire de manière différente :

- des bits peuvent être fixés à une valeur arbitraire grâce à une faute destructive ou permanente,
- des bits peuvent prendre une valeur opposée, i.e. si un bit vaut 1 la faute le transforme en 0 et vice-versa,

– des bits peuvent prendre une valeur aléatoire, l'attaquant n'a alors aucun contrôle sur la nouvelle valeur de la variable après la faute.

1.2 Attaques passives

Les attaques par canaux cachés ont été introduites comme un danger majeur à la communauté en 1996 par Kocher [Koc96]. Le principe est d'observer les propriétés physiques du composant afin de retrouver de l'information sur le secret qui est en train d'être utilisé. Les SCA, ont depuis été l'objet de nombreuses recherches. On peut différencier ces attaques suivant le type de propriété physique du composant dont elles tirent parti. Nous présentons ici quelques unes des propriétés physiques qui peuvent être utilisées pour une attaque. La consommation de courant est la plus facile à mesurer et est présentée plus en détail.

1.2.1 Temps de calcul

L'attaque par analyse du temps de calcul est la première attaque SCA proposée. Kocher [Koc96] utilise les différences en temps, entre des exécutions de RSA notamment, pour retrouver des bits du secret. Certaines variations de temps dépendent de la valeur des données traitées. Ainsi, dans le cas de la multiplication modulaire de Montgomery originale [Mon85], une opération de soustraction par le modulo est susceptible d'être calculée. Le temps supplémentaire au calcul possible de cette soustraction peut ensuite être utilisé par un attaquant pour retrouver des bits de l'exposant secret utilisé dans une signature RSA par exemple. Les mesures de temps demandent toutefois une très grande précision pour être utilisable.

1.2.2 Émissions électromagnétiques

Les attaques utilisant les émissions électromagnétiques d'un composant sont basées sur le fait que de faibles charges de courant qui sont en mouvement produisent un champ magnétique qui lui-même produit un champ électrique. Les cartes à puces actuelles sont constituées de millions de transistors et d'interconnexions qui génèrent ces émissions électromagnétiques. Cette propriété a été utilisée pour réaliser des attaques par canaux cachés sur des composants cryptographiques. Quisquater et Samyde [QS01] ainsi que Gandolfi et al. [GMO01] sont les premiers à donner des résultats expérimentaux d'attaques sur carte à puce utilisant les émissions électromagnétiques. Agrawal et al. [AARR02] proposent ensuite une étude plus complète de ce type d'attaque par canaux cachés. Ils montrent que ce type de canal caché peut être utilisé lorsqu'un attaquant ne peut pas avoir accès à une mesure de consommation de courant. De plus, ils utilisent les émissions électromagnétiques pour casser des contre-mesures devant résister aux attaques par analyse de courant. Les émissions électromagnétiques sont souvent considérées comme donnant des informations très précises sur les données traitées par le composant. Néanmoins, il est très compliqué en pratique d'obtenir ces informations de manière optimale. De nombreux critères comme

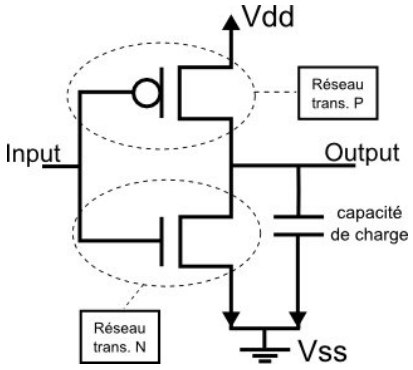


FIGURE 1.2 – Structure CMOS de base : inverseur.

l'endroit où placer la sonde permettant d'enregistrer ces émissions, ou bien les dimensions et le type de sonde à utiliser restent encore à caractériser de manière précise.

1.2.3 Consommation de courant comme canal caché

L'analyse de la consommation de courant d'une carte à puce est très facile à réaliser et nécessite un équipement peu onéreux. Kocher [KJJ99] l'a utilisé en premier dans le cadre d'attaques par canaux cachés.

1.2.3.1 Logique CMOS

La grande majorité des cartes à puces utilisent la technologie CMOS car elle est peu chère et efficace. L'une des propriétés les plus intéressantes de cette technologie est que la consommation statique d'un circuit CMOS est faible. Celle-ci correspond à la consommation du circuit lorsqu'il est dans un état d'équilibre. Il y a une consommation de courant plus significative lorsque des transistors CMOS changent d'état, on l'appelle consommation dynamique. Plus généralement, un circuit CMOS ne consomme de manière significative que lorsqu'il y a une activité des transistors et donc de la logique. Il existe une relation forte entre la consommation du composant est le nombre de bits qui changent d'états à un instant donné.

La Figure 1.2 montre un inverseur CMOS, une structure de base d'un circuit CMOS. On remarque que l'inverseur est constitué de réseaux de transistors *P-channel Metal Oxyde Semiconductor* (PMOS) et *N-channel Metal Oxyde Semiconductor* (NMOS). Lorsqu'aucune opération n'est effectuée, la tension est soit de V_{dd} soit de V_{ss} au niveau de l'*Input* et de l'*Output*. Néanmoins, lors d'une transition de l'*Input* de V_{dd} vers V_{ss}, ou vice-versa, il y a durant un court instant un courant de court circuit. De plus à cet instant, les capacités de charges, comme les bus ou les portes logiques, sont chargées ou déchargées.

La consommation de courant d'un composant s'observe en mesurant la différence de tension divisée par la résistance aux bornes d'une résistance montée en série entre le V_{ss} du composant et une alimentation externe. On utilise ensuite un oscilloscope pour numériser le courant et l'enregistrer sur un ordinateur.

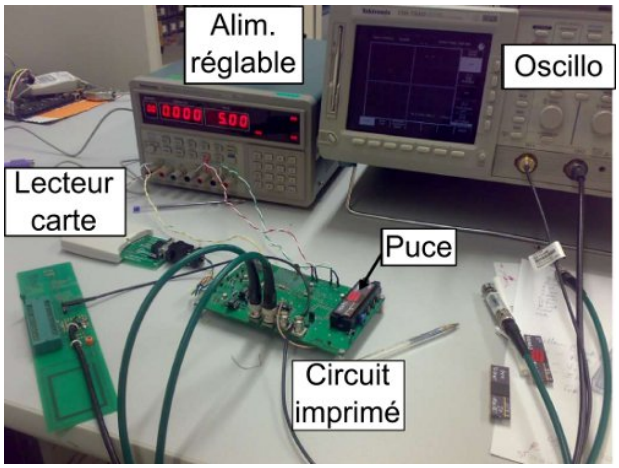


FIGURE 1.3 – Matériel classique nécessaire pour réaliser des mesures de consommation sur une carte à puce.

1.2.3.2 Matériel de mesure

La Figure 1.3 représente une partie du matériel nécessaire à l'acquisition de courbes de consommation. Un ordinateur sert tout d'abord à envoyer des commandes à la carte à puce afin de lancer des opérations sur celle-ci avec des paramètres donnés. Une première sonde de courant permet de mesurer la consommation alors qu'une seconde sert à déclencher l'acquisition des courbes de consommation par l'oscilloscope. Cette deuxième sonde est ainsi liée au signal d'entrée envoyé à la carte. L'alimentation stable permet de fixer avec précision et régularité l'intensité de courant fourni au circuit. Les attaques par analyse de courant (*Power Analysis*, PA), posent une menace très sérieuse de part leur efficacité et leur facilité de mise en œuvre. On s'intéresse donc plus particulièrement aux attaques par canaux cachés utilisant la consommation de courant par la suite.

1.2.3.3 Modèles de consommation

Nous avons vu que l'attaquant peut disposer assez facilement de courbes de consommation d'un circuit. À partir de ces courbes, il veut savoir quelle donnée est traitée par le circuit. Pour cela, plusieurs méthodes de modélisation ont été proposées mais deux modèles sont les plus utilisés en pratique : le modèle en distance de Hamming et en poids de Hamming.

Messerges et al. [MDS99a] observent une relation linéaire entre le nombre de bits à 1 présents dans un bus ou un registre interne à un instant t et la consommation de courant du composant à ce même instant. Ce modèle a d'ailleurs été étendu en considérant la distance de Hamming entre deux états consécutifs du composant [BDJ04], au lieu du poids de Hamming. On modélise alors la consommation de courant $\mathcal{C}$ à un instant t par :

$$\mathcal{C}(t) = a \cdot HW(M \oplus R) + b, \tag{1.1}$$

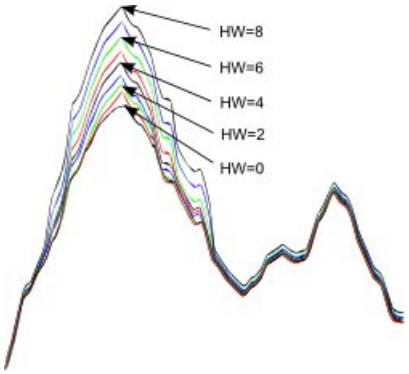


FIGURE 1.4 – La consommation de courant est liée au poids de Hamming des données traitées.

où a est une constante liée au composant utilisé, $HW(M \oplus R)$ est la distance de Hamming entre l'état actuel M d'un registre interne ou d'un bus et l'état précédent R. Enfin, b correspond à un bruit gaussien, plus particulièrement une somme de bruits provenant de différentes sources : bruit intrinsèque au composant, bruit dû à la mesure de l'oscilloscope, etc. D'après le théorème central limite, cette somme de bruits tend vers une distribution gaussienne. La Figure 1.4 illustre une relation approximativement linéaire entre la consommation de courant et le poids de Hamming d'une donnée de 8 bits transférée dans un registre interne du composant.

Une modélisation beaucoup plus simple et générique est proposée par Gierlichs et al. [GBTP08]. Les auteurs considèrent directement la valeur de la donnée. Ce modèle semble moins bien fonctionner que les précédents, qui utilisent le poids de Hamming, sur des composants CMOS. Néanmoins, ce modèle qui utilise simplement la valeur peut être plus efficace lorsque d'autres types de logiques sont utilisées.

Chapitre 2

Attaques par analyse de courant

Nous nous concentrons dans le reste de notre travail sur les attaques par canaux cachés utilisant la consommation de courant. Il est important de noter que les principes décrits s'appliquent, pour la plupart, à un autre choix de canal caché. Nous étudions deux types d'attaques par analyse de courant : les attaques par analyse simple (*Simple Side-Channel Analysis*, SSCA) et les attaques par analyse différentielle (*Differential Side-Channel Analysis*, DSCA).

2.1 Attaques par analyse simple de courant

En 1998, Kocher et al. [KJJ99] présentent le principe d'attaques par analyse simple, une sous-catégorie d'attaques par canaux cachés. Ils ont démontré l'efficacité de leur attaque en utilisant la consommation de courant. Ils la nomment attaque par analyse simple de courant (*Simple Power Analysis*, SPA). Gandolfi et al. [GMO01] utilisent les radiations électromagnétiques comme canal caché et appellent l'attaque par analyse simple d'émissions électromagnétiques (*Simple ElectroMagnetic Analysis*, SEMA). Plus généralement, peu importe le canal caché utilisé, ce type d'attaque par analyse simple est appelé SSCA. Le principe de ces attaques est d'interpréter directement à partir du canal caché des informations sur le secret utilisé au cours de l'opération observée. Ceci est possible si :

- l'implémentation de l'algorithme cryptographique utilise des branches conditionnelles dépendant des bits du secret,
- on observe des motifs, sur l'observation du canal caché, permettant de distinguer des informations sur le secret.

Un autre type d'attaques appelées *template attacks* [CRR02] peut être considéré comme une attaque par analyse simple même si l'attaquant doit, au préalable, construire une bibliothèque de mesures de consommation. Nous ne considérons pas dans notre étude ce type d'attaque qui demande à l'attaquant un contrôle très important sur le circuit attaqué. De nombreux articles dans la littérature sont disponibles sur ce sujet [ARRS05, RO05, APSQ06, OM07].

De nombreux algorithmes cryptographiques symétriques et asymétriques ont été attaqués par des attaques SSCA. On peut citer l'attaque sur le DES de Messerges et al. [MDS99a] ou celle de Mangard [Man02] sur l'AES. Une description plus détaillée de l'attaque sur le DES se trouve dans [Koç09, Section 13.3.3]. Une implémentation soignée des algorithmes de DES et AES permet toutefois d'éviter les attaques SSCA. Ces cryptosystèmes sont beaucoup plus sensibles aux attaques par analyse différentielle que nous étudions ci-dessous. Dans le cadre des attaques SSCA, les cryptosystèmes asymétriques sont les moins évidents à protéger comme nous le voyons à la section 7.2.

2.2 Attaques par analyse différentielle de courant

Les SSCA s'intéressent à des relations directes entre les opérations effectuées et leur effet sur le canal caché étudié. Ces relations sont souvent visibles à l'œil nu et ne requièrent que peu de courbes de consommation pour être distinguables. Pour réaliser une SSCA efficace, il faut connaître en détail l'implémentation du composant attaqué ce qui n'est pas forcément trivial pour un attaquant.

Les DSCA se révèlent plus puissantes même si l'attaquant n'a pas une grande connaissance des détails du système. Ce type d'attaque enregistre les relations entre données et canal caché grâce à une analyse statistique. Des relations bien moins évidentes peuvent donc être détectées par des DSCA. En contrepartie, elles nécessitent beaucoup plus de courbes de consommation et donc un accès au composant prolongé de la part de l'attaquant. Nous détaillons dans cette section le principe des attaques DSCA, une classification des attaques de ce type et des tests statistiques parmi les plus utilisés dans la littérature pour effectuer une DSCA.

2.2.1 Principe

Les attaques par analyse différentielle de courant (*Differential Power Analysis*, DPA) correspondent à une DSCA utilisant la consommation de courant comme canal caché. Ce type d'attaque suit ces grands principes :

- l'attaquant utilise un grand nombre de courbes de consommation de courant du fonctionnement d'un algorithme cryptographique ayant une clé secrète fixée sur un composant,
- il cherche à attaquer une variable intermédiaire de cet algorithme cryptographique, variable qui doit dépendre de la valeur de la clé secrète,
- il émet des hypothèses sur la partie de la clé secrète utilisée dans le calcul de la variable intermédiaire visée,
- il applique un test statistique afin de connaître la validité de son hypothèse de clé.

Ce processus générique aux attaques DSCA est résumé par la Figure 2.1 dans le cadre de l'analyse de courant.

Nous donnons à présent une description plus formelle. Soit K une variable aléatoire représentant une partie du secret. Soit X une variable aléatoire représentant une partie de

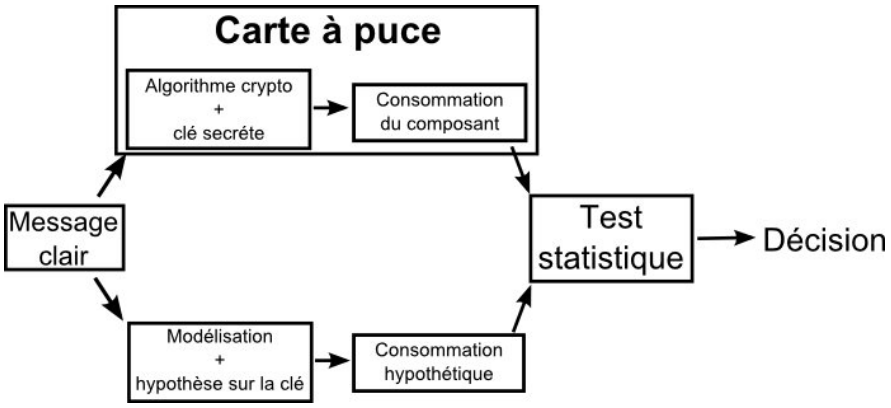


FIGURE 2.1 – Principe d'une analyse différentielle

la donnée d'entrée, ou de sortie, de l'algorithme cryptographique. On suppose que l'attaquant s'intéresse à une valeur intermédiaire calculée par la fonction F, appelée fonction de sélection, qui prend X et K en paramètres. Soit L la variable aléatoire représentant la fuite d'information du canal caché produite par le calcul de $F(X, K)$. En pratique, l'attaquant est seulement capable d'obtenir N réalisations de la variable aléatoire L, notées $V_L = (l_1, \ldots, l_N)$, à partir de N valeurs d'entrées X différentes, notées $V_X = (x_1, \ldots, x_N)$. Grâce à un distingueur D, il combine ces deux vecteurs avec une hypothèse sur le secret k'. Si le distingueur D est pertinent et si le vecteur V_L donne assez d'informations sur $F(X, K)$, alors la valeur correcte k prise par K peut être retrouvée. Des recherches ont été effectuées sur la création d'un modèle pour $F(X, K)$. Par exemple, il a été proposé de choisir le poids de Hamming [MDS99b], la distance de Hamming [BDJ04] ou simplement la valeur [GBTP08] (voir section 1.2.3.3). D'autres recherches ont été conduites sur le distingueur D qui joue un rôle fondamental dans l'attaque. Selon le choix effectué, le distingueur peut extraire plus ou moins d'information à partir du canal caché (voir section 2.2.3).

2.2.2 Caractérisation des attaques

On peut classifier les attaques par canaux cachés d'une autre manière que celle présentée dans le chapitre 1 et qui est plus pertinente dans le cadre des attaques par analyse différentielle.

2.2.2.1 Clair connu ou choisi

Pour réaliser une attaque par canaux cachés, l'attaquant connaît souvent le clair et le chiffré de l'algorithme attaqué. Dans le cadre des attaques à clair connu, le message clair est supposé être uniformément distribué dans l'espace des messages alors que pour une attaque à clair choisi, l'attaquant fixe la valeur qu'il souhaite pour le message. Dans ce deuxième cas, l'attaquant a donc supposément plus de pouvoir car, suivant les protocoles attaqués, le choix d'un message d'entrée n'est pas toujours possible. De plus, en choisissant

la valeur du clair, il peut faire en sorte que son attaque soit plus efficace pour des valeurs précises que pour d'autres tirées aléatoirement.

2.2.2.2 Avec ou sans phase de profilage

Une phase de profilage permet à l'attaquant de connaître assez précisément le comportement du circuit attaqué pour un canal caché donné. Par exemple, pour une valeur de consommation de courant du circuit donnée, il peut être capable d'évaluer avec une certaine probabilité le poids de Hamming de la donnée traitée. Ce profilage suppose que l'attaquant a accès à une copie du circuit qu'il souhaite attaquer et qu'il ait un contrôle total sur le fonctionnement de cette copie. Par exemple, il peut fixer des valeurs pour une clé secrète et des messages clairs et enregistrer des courbes de consommation de courant pour en déduire le comportement du circuit. Cette supposition très puissante est notamment faite pour les *template attacks* rapidement présentées précédemment. Dans une attaque sans phase de profilage, l'attaquant n'a accès qu'au circuit final qu'il souhaite attaquer.

2.2.2.3 Univarié ou multivarié

La consommation de courant ou les radiations électromagnétiques d'un circuit varient suivant le temps. L'acquisition d'une courbe enregistre ces mesures dans le temps. Dans une attaque univariée, l'attaquant utilise un seul point de mesure dans le temps pour son attaque. S'il ne sait pas à quel instant la valeur intermédiaire attaquée est calculée, il applique son attaque à chaque point dans le temps de manière indépendante. Dans le meilleur cas, l'attaque réussira au point dans le temps où le circuit a calculé la valeur intermédiaire attaquée et elle ne donnera aucun résultat significatif aux autres points. Dans une attaque multivariée, plusieurs points dans le temps sont utilisés pour une attaque. Le choix de ces points peut provenir d'une phase de profilage au préalable, ou d'une stratégie spécifique de l'attaquant. Dans la littérature, les attaques univariées, respectivement multivariées, sont également connues sous le nom d'attaques de premier ordre, respectivement d'attaques d'ordre supérieur.

Pour le reste de notre étude, nous considérons principalement les attaques à clair aléatoire, sans phase de profilage, univariées car elles correspondent au scénario d'attaque le plus classique.

2.2.3 Quelques méthodes statistiques

De nombreux travaux ont été réalisés dans l'étude de différents tests statistiques dans le contexte d'attaques par canaux cachés. On présente ici une liste non exhaustive des principales propositions les plus efficaces.

2.2.3.1 Distance de moyennes

L'attaque DPA historique de Kocher et al. [KJJ99] utilise comme test statistique la distance de moyennes. Une fois l'hypothèse de clé réalisée par l'attaquant, celui-ci partitionne

en deux ensembles la sortie de la fonction de sélection suivant sa valeur. Soit N courbes de consommation de courant $\mathcal{C}_i(t)$ avec $i = 1, \ldots, N$ acquises sur composant par l'attaquant qui correspondent à N réalisations de la variable aléatoire L. Pour chaque acquisition, il utilise un message clair aléatoire x_i qui correspond à une réalisation de la variable aléatoire X. On suppose, pour plus de simplicité, que la fonction de sélection F retourne le bit le plus significatif de la valeur intermédiaire visée. On appelle alors l'attaque, DPA mono-bit. Soit k' l'hypothèse sur le secret. L'attaquant forme deux ensembles :

$$G_0 = \{l_i \mid F(x_i, k') = 0\} \quad \text{et} \quad G_1 = \{l_i \mid F(x_i, k') = 1\}.$$

On calcule ensuite la courbe de distance des moyennes $\Delta_{k'}(t)$ entre les deux partitions G_0 et G_1. En principe, la courbe $\Delta_{k'}(t)$ est différente de zéro lorsque k' correspond à la vrai sous-clé du composant car on trouve une relation entre la consommation de courant et l'état du bit le plus significatif de la valeur intermédiaire visée dans notre exemple. De plus, cette déviation de zéro apparaît à l'instant t_0 où la vraie sous-clé est manipulée par le composant. On définit la courbe de distance de moyennes $\Delta_{k'}(t)$ comme :

$$\Delta_{k'}(t) = \left| \frac{\sum_{l \in G_0} l}{|G_0|} - \frac{\sum_{l \in G_1} l}{|G_1|} \right|.$$

L'attaquant calcule ensuite $|K|$ courbes de différences $\Delta_{k'}(t)$ pour chaque valeur possible de la sous-clé k'. Il décide que l'hypothèse k' est la plus probable d'être la vraie en considérant le plus grand pic sur les courbes $|\Delta_{k'}(t)|$. La qualité d'une attaque DPA, qui est liée à la qualité des courbes de différences $\Delta_{k'}(t)$, dépend principalement du nombre N de mesures de courbes de consommations réalisée par l'attaquant. La Figure 2.2 montre un exemple de courbes de différences pour différentes hypothèses de clés lors d'une attaque DPA sur l'algorithme DES.

2.2.3.2 Coefficient de corrélation de Pearson

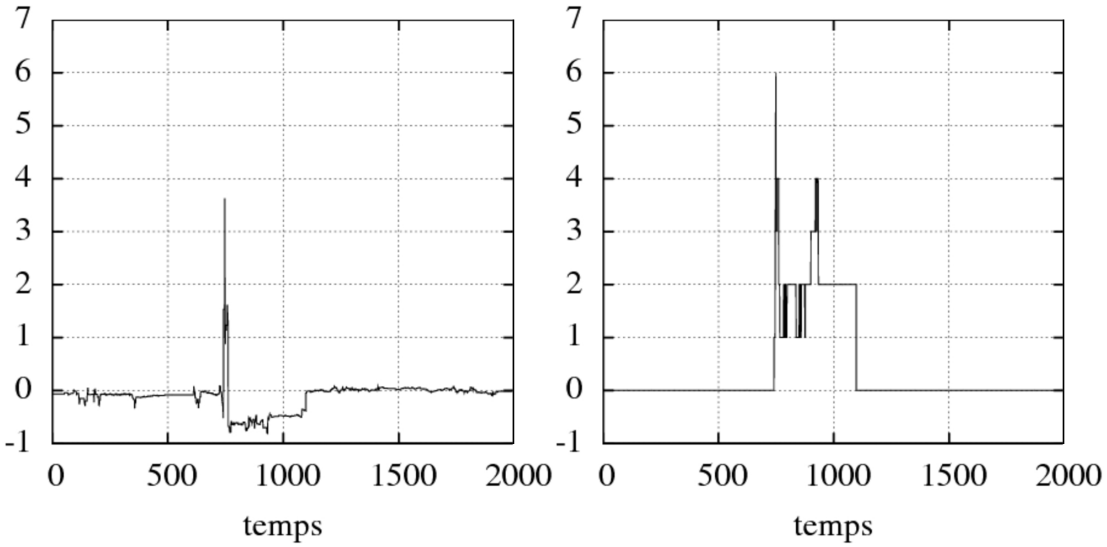
Le coefficient de Pearson est un test connu dans le domaine des statistiques. Il permet de mesurer les relations linéaires entre deux variables aléatoires X et Y. Ce coefficient est défini comme :

$$\rho(X, Y) = \frac{\operatorname{cov}(X, Y)}{\sqrt{\operatorname{var}(X)\operatorname{var}(Y)}} = \frac{E(XY) - E(X)E(Y)}{\sqrt{\operatorname{var}(X)\operatorname{var}(Y)}}.$$

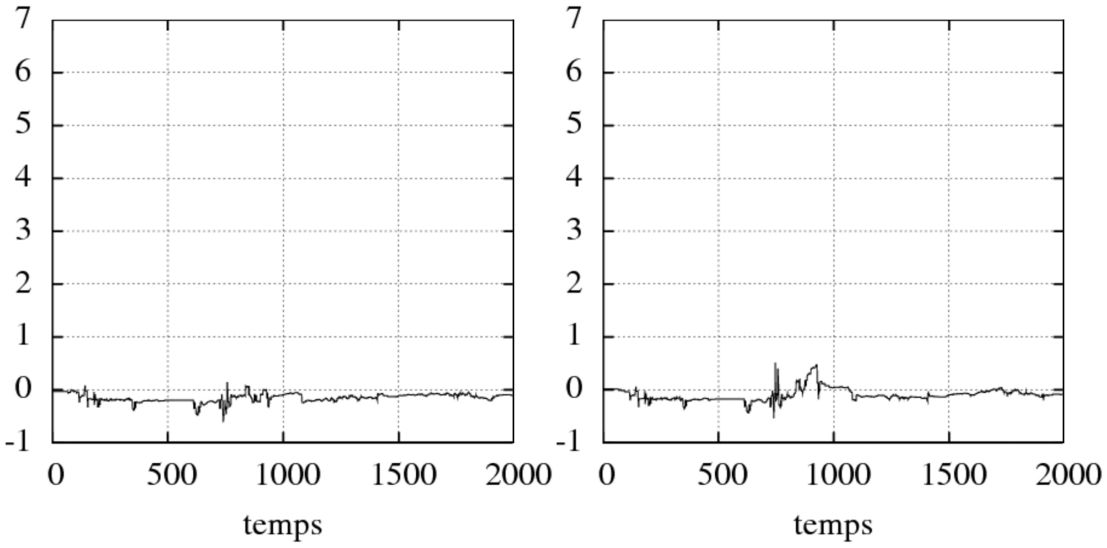
La formule donne un coefficient qui a une valeur comprise entre -1 et $+1$. Soit k' une hypothèse sur le secret. Si on considère N observations des variables X et L, on obtient :

$$\rho_{k'}(X, L) = \frac{N \sum_i l_i F(x_i, k') - (\sum_i l_i \sum_i F(x_i, k'))}{\sqrt{N \sum_i l_i^2 - (\sum_i l_i)^2} \sqrt{N \sum_i F(x_i, k')^2 - (\sum_i F(x_i, k'))^2}}. \tag{2.1}$$

Si les valeurs de L augmentent en même temps que celles de X, les points (x_i, l_i) forment une ligne droite de pente positive, alors $\rho_{k'}(X, L) = +1$. Si les points forment une ligne



(a) $k' = 0$ (b) $k' = k = 1$, vraie clé



(c) $k' = 24$ (d) $k' = 60$

FIGURE 2.2 – Attaque DPA en sortie de SBox au premier tour sur les premiers 6 bits d'une clé DES. Courbes de différences pour les hypothèses de clés 0, 1, 24 et 60. La vraie clé vaut 1.

droite de pente négative, alors $\rho_{k'}(X, L) = -1$. S'il n'y a aucune relations entre X et L alors $\rho_{k'}(X, L) = 0$.

Dans beaucoup de composants, la consommation de courant est, en grande partie, linéaire au poids de Hamming de la donnée traitée à un instant t (voir section 1.2.3). L'utilisation de ce test statistique dans le contexte des attaques par canaux cachés est appelée attaque par analyse différentielle de courant utilisant le coefficient de Pearson (*Correlation Power Analysis*, CPA) [BDJ04]. On obtient souvent des résultats d'attaques meilleurs que la DPA avec distance des moyennes. On entend par résultats meilleurs, le fait qu'un attaquant a besoin de moins de mesures de consommation pour que la vraie valeur de la clé apparaisse plus clairement.

Pour utiliser ce coefficient, on suppose que les variables X et L suivent une loi de distribution normale. Ainsi, le facteur de Pearson fait partie des tests paramétriques. À l'inverse, on parle de tests non-paramétriques lorsqu'aucune supposition n'est faite sur la distribution de probabilité des variables.

2.2.3.3 Partitionnement de données

Brièvement proposé dans [SGV08] puis plus en détail dans [BGLR09], la méthode statistique de partitionnement de données, *clustering* en anglais, a été proposée pour les attaques par canaux cachés. Les auteurs l'appellent attaque par analyse différentielle utilisant le partitionnement de données (*Differential Cluster Analysis*, DCA). Le principe de partitions est déjà présent dans les attaques utilisant la distance de moyennes ou l'information mutuelle que nous décrivons dans le chapitre 4. Dans le cas de la distance de moyennes, l'attaquant place les courbes dans deux ensembles. Soit F_k une fonction surjective dépendant de la clé k telle que $F_k : \{0,1\}^u \to \{0,1\}^w$ où u et w dépendent de l'algorithme considéré. Par exemple, pour une attaque sur l'algorithme AES, on peut choisir F_k comme la fonction SBox, d'où $u = 8$ et $w = 8$ car l'entrée et la sortie d'une SBox sont sur 8 bits. Pour l'attaque DCA, on considère un partitionnement en 2^w ensembles. Une fois les courbes de consommations placées dans l'ensemble correspondant à la valeur de sortie de F_k, l'attaque DCA utilise un critère de partitionnement pour quantifier la qualité de la séparation. Batina et al. proposent différents critères [BGLR09, Section 2.1] dont un calcul de variance, très efficace, entre les ensembles.

2.2.3.4 Corrélation de Spearman

La corrélation de Spearman est très similaire au facteur de Pearson. Alors que Pearson utilise les valeurs des variables X et Y, Spearman s'applique sur leur rang, i.e. leur représentation ordinale. Le coefficient de Spearman est un test de corrélation non-paramétrique, i.e. il mesure des relations entre les variables sans faire de supposition sur leurs distributions de probabilité. Si plusieurs valeurs d'une variable sont identiques, la formule pour le calcul du coefficient de Spearman est identique à Pearson (2.1). Le rang des valeurs identiques est alors la moyenne de leurs rangs. S'il n'y a pas de valeurs identiques, une formule simplifiée

peut alors être utilisée :

$$\rho(X,Y) = 1 - \frac{6\sum (X_i - Y_i)^2}{N(N^2-1)}.$$

Dans le cadre des attaques par canaux cachés, plusieurs valeurs d'une variable peuvent être identiques, on utilise donc la formule (2.1). L'utilisation de ce test pour des attaques a été proposée dans [BGLR08].

2.2.3.5 Corrélation de Kendall

Le coefficient de Kendall [Ken38] noté τ est un test de corrélation non-paramétrique, comme celui de Spearman, qui mesure les relations entre deux variables représentées sous forme ordinale. Alors que les tests de Pearson et Spearman servent à tester l'hypothèse nulle qu'il n'existe pas de relation entre les variables, le coefficient de Kendall mesure le degré de liaison entre les deux variables. Nous expliquons le test par un exemple. Soit X et Y deux variables ordinales à $N = 4$ valeurs telles que :

$$\begin{cases} X = \{X_1, X_2, X_3, X_4\} & = \{3, 4, 2, 1\}, \\ Y = \{Y_1, Y_2, Y_3, Y_4\} & = \{3, 1, 4, 2\}. \end{cases}$$

On ordonne maintenant les variables de sorte que les valeurs de X soient en ordre croissant :

$$\begin{cases} X & = \{1, 2, 3, 4\}, \\ Y & = \{2, 4, 3, 1\}. \end{cases}$$

Puisque les rangs sont croissants pour X, il suffit de compter combien de rangs sont aussi en rang croissant pour Y pour déterminer le degré de correspondance entre les deux variables. Si les rangs de Y était eux aussi dans l'ordre croissant les deux variables seraient en relation parfaite par rapport à leurs rangs. Si les rangs de Y était en ordre décroissant, les variables auraient une relation inverse parfaite. Pour les situations intermédiaires, on compte le nombre de paires ordonnées de la même manière ainsi que les paires ordonnées de manière opposée. En reprenant notre exemple :

- la paire $(X_2, Y_2) = (2, 4)$ est en ordre croissant,
- la paire $(X_2, Y_3) = (2, 3)$ est en ordre croissant,
- la paire $(X_2, Y_4) = (2, 1)$ est en ordre décroissant,
- la paire $(Y_2, X_2) = (4, 2)$ est en ordre décroissant,
- la paire $(Y_2, X_3) = (4, 3)$ est en ordre décroissant,
- la paire $(Y_2, X_4) = (4, 4)$ est considéré comme en ordre décroissant.

On note n_c le nombre de paires en ordre croissant, n_d le nombre de paires en ordre décroissant et soit $S = n_c - n_d$. Il y a en tout, $N(N-1)/2$ paires, le coefficient de Kendall τ_a est alors :

$$\tau_a = \frac{2S}{N(N-1)},$$

la différence entre le nombre de paires en ordre croissant et celles en ordre décroissant, divisée par le nombre total de paires.

Dans le cas où plusieurs rangs ont la même valeur, la formule du coefficient de Kendall est différente. Soit α le nombre de rangs identiques dans la première variable X. Soit β le nombre de rangs identiques dans la deuxième variable Y. Soit τ_b le coefficient de Kendall :

$$\tau_b = \frac{S}{\sqrt{(N(N-1)/2 - \alpha)(N(N-1)/2 - \beta)}}.$$

Le coefficient γ de Goodman et Kruskal [GK54] est assez similaire à celui de Kendall. Dans le cadre des attaques par canaux cachés, on utilise le coefficient τ_b [BGLR08].

2.2.3.6 Autres tests non-paramétriques

Dans [VCS09], les auteurs proposent l'attaque par analyse différentielle utilisant le test Kolmogorov-Smirnov (K-S) et l'attaque par analyse différentielle utilisant le test Cramér-von Mises (CVM). Ces tests sont assez similaires à la méthode DCA (section 2.2.3.3) ou à la méthode utilisant l'information mutuelle (chapitre 4). En effet, les données sont placées dans différents ensembles, chacun couvrant typiquement un intervalle de valeurs pris par la variable intermédiaire attaquée. Nous rappelons brièvement la définition d'une fonction de répartition. Soit X une variable aléatoire discrète ayant pour valeurs $\{x_1, \ldots, x_N\}$ avec probabilités $P(X = x_i)$ pour $i = 1, \ldots, N$. La fonction de répartition de X, notée $F_X(x)$, est définie telle que :

$$F_X(x) = P(X \leq x) = \sum_{x_i \leq x} P(X = x_i).$$

Nous notons les fonctions de répartition pour les variables X et Y par $F_X(i)$ et $F_Y(i)$ respectivement. Le test Kolmogorov-Smirnov est défini par :

$$D_{\mathrm{KS}}(X \parallel Y) = \sup_i \left| F_X(i) - F_Y(i) \right|.$$

De la même manière, le test Cramér-von Mises est défini par :

$$D_{\mathrm{CVM}}(X \parallel Y) = \sum_i \left(F_X(i) - F_Y(i) \right)^2.$$

Nous considérons par la suite le test Cramér-von Mises qui donne des résultats légèrement meilleurs au test Kolmogorov-Smirnov.

Chapitre 3

Contre-mesure DSCA pour AES

Le cryptosystème symétrique AES [Nat01] a remplacé l'ancien standard DES [Nat93] il y a quelques années. L'AES est désormais utilisé dans de nombreux composants embarqués et donc sa résistance aux attaques par canaux cachés a été étudiée en profondeur dans la littérature. Différents types de contre-mesures pour cet algorithme ont été proposées. La méthode la plus classique de protection consiste à ajouter de l'aléa dans les valeurs intermédiaires du calcul. Comme la fuite d'information par le canal caché dépend des valeurs traitées par le microcontrôleur à un instant donné, les données ne sont plus corrélées avec les observations. La plupart des contre-mesures proposées pour l'AES se concentrent sur l'opération de *SubBytes* qui est la seule partie non-linéaire de l'algorithme.

Les implémentations *hardware* les plus efficaces du *SubBytes* utilisent l'algèbre sur une tour d'extensions de corps finis telle que $GF(2^8) \supset GF(2^4) \supset GF(2^2)$. Ainsi les techniques proposées dans les articles [WOL02, OMPR05, OS06] calculent le *SubBytes* dans un sous-corps de $GF(256)$. Dans ces articles, les auteurs fixent une construction de tour d'extensions de corps finis donnée. Rudra et al. [RDJ+01] s'intéressent plutôt à une construction minimisant les opérations dans le corps fini. Dans ce travail, nous nous intéressons à l'effet de rendre aléatoire cette construction afin d'obtenir une protection face aux attaques par canaux cachés. En calculant l'inverse dans $GF(256)$, nous devons, dans notre cas, calculer la norme dans l'extension de corps $GF(16) \subset GF(256)$. Ainsi, la distribution des valeurs de norme, pour un élément de $GF(256)$ donné, doit être le plus distribué possible sur $GF(16)$. Nous étudions par la suite une méthode efficace pour réaliser cela. Ce travail a été réalisé en collaboration avec Alexis Bonnecaze et Pierre Liardet [BLV10].

3.1 Rappels sur AES

Nous donnons une rapide description de la fonction de ronde d'AES. Nous omettons l'opération d'expansion de la clé. Le lecteur intéressé peut se référer à [Nat01]. L'AES est défini en bloc de 128 bits et pour des clés de tailles de $128, 192$ et 256 bits. Le message clair de 128 bits est vu comme une matrice 4×4 d'octets. Les octets représentent des éléments de $GF(256)$.

L'AES opère sur une matrice d'octets en itérant des tours composés de diverses transformations sur cette matrice. Le tour initial consiste en une opération d'*AddRoundKey*. Les tours suivants correspondent à l'application successive des opérations de *SubBytes*, *ShiftRows*, *MixColumns* et *AddRoundKey*. Le dernier tour n'effectue pas la transformation *MixColumns*. L'*AddRoundKey* correspond à une opération XOR bit à bit entre la matrice est la clé de tour. Les clés de tour sont dérivées de la clé originale grâce à l'opération *Key Expansion*. Le *ShiftRows* est un décalage cyclique de chacune des lignes de la matrice. La première ligne est inchangée, la seconde est décalée de manière cyclique d'un octet vers la gauche, la troisième de deux octets et la quatrième de trois octets. Le *MixColumns* considère chaque colonne de la matrice comme des coefficients d'un polynôme de degré trois et le multiplie modulo $x^4 + 1$ par le polynôme fixé $\{03\}.x^3 + \{01\}.x^2 + \{01\}.x + \{02\}$. Le *SubBytes* est la partie principale de l'algorithme. Chaque octet de la matrice est remplacé par son substitué dans une SBox. La SBox est une composition de deux transformations : un calcul d'inverse dans $GF(256)$ et une transformation affine.

3.2 État de l'art des méthodes de masquage

Le but d'une contre-mesure aux attaques par canaux cachés est de rendre la consommation de courant du composant aussi indépendante que possible des données traitées par l'algorithme cryptographique. Les techniques de masquage, qui ont été largement étudiées dans la littérature, ont cet objectif. Le principe d'une implémentation masquée est de remplacer la valeur intermédiaire v par une combinaison $C(v, r)$ de v et d'une valeur aléatoire r. Dans le cas de l'AES, v et r sont des valeurs binaires et $C(v, r) = v \oplus r = v'$ correspond à un XOR bit à bit.

L'attaque par analyse différentielle de courant d'ordre supérieur (*Higher-Order Differential Power Analysis*, HODPA) est un type d'attaque par canaux cachés proposé pour contrer les méthodes de masquage. Alors qu'une attaque DPA classique analyse l'information contenue à un instant dans le temps d'une courbe de consommation, l'attaque HODPA combine différents instants. Par exemple, si l'attaquant est capable de trouver l'instant où le masque r est généré par le composant et l'instant où v' est calculé, il peut utiliser ces informations pour retrouver la valeur correcte v. Pour contrer cela, des techniques de masquage d'ordre supérieur sont proposées. Néanmoins, empêcher des attaques d'ordre n reste un problème difficile. Dans cette étude, nous nous concentrons sur les attaques de premier ordre, i.e. les attaques différentielles classiques, étant donné qu'elles sont les plus faciles à réaliser pour un attaquant et qu'elles représentent donc un risque majeur.

La seule partie non-linéaire de l'AES est l'inversion sur $GF(2^8)$ dans l'opération *SubBytes*. En utilisant une méthode de masquage, nous devons calculer l'inverse de l'entrée $v + r_1$ afin d'obtenir $v^{-1} + r_2$ avec r_1, r_2 des valeurs aléatoires. Nous étudions par la suite les principales méthodes de masquage proposées dans la littérature. Nous présentons en premier les méthodes génériques s'appliquant à l'AES. Nous considérons ensuite les méthodes utilisant la construction en tour d'extensions de corps finis. Ces méthodes sont particulièrement intéressantes pour une implémentation en *hardware*.

The Transform Masking Method [AG01]. La méthode consiste à transformer un masque booléen $v + r_1$ en un masque multiplicatif $v.r'$, à calculer l'inverse, puis à retrouver le masque booléen tel que $v^{-1} + r_2$. Trichina et al. [TDSG03] simplifient la méthode en considérant les masques $r_1 = r_2$. Ces techniques sont sensibles à l'attaque DPA utilisant la valeur zéro. Si $v = 0 \in GF(2^8)$, alors aucun masque multiplicatif ne peut cacher cette valeur spéciale.

Embedded Multiplicative Masking [GT02]. Les auteurs proposent une solution au problème du zéro. L'idée consiste à plonger le corps $GF(2^8)$ dans l'anneau $R_k = GF(2)[x]/(pq) \cong GF(2^8) \times GF(2^k)$ où p est le polynôme de degré 8 de l'AES et q est un polynôme irréductible de degré k et premier à p. Nous considérons la fonction :

$$\begin{aligned} \rho : GF(2^8) &\to R_k \\ v &\mapsto v + rp \bmod pq, \end{aligned}$$

où r est un polynôme de degré inférieur à k choisi aléatoirement. Alors la valeur $v = 0 \in GF(2^8)$ a 2^k valeurs de sortie possibles dans R_k et devrait donc être moins visible lors d'une attaque par canaux cachés. Néanmoins, pour avoir un effet notable face à l'attaque utilisant la valeur zéro, il faut choisir k assez grand et donc travailler dans un anneau R_k grand ce qui pose des problèmes d'efficacité.

Masquage de la SBox [Mes01]. Considérons une table précalculée, appelée SBox, qui est utilisée pour calculer le *SubBytes*. La méthode de Messerges consiste à masquer cette table précalculée par le masque généré pour cacher la valeur d'entrée du *SubBytes*. Étant donné que le masque doit changer à chaque entrée dans le *SubBytes*, la table doit être recalculée à chaque fois. Cette contre-mesure est donc très coûteuse en temps. Itoh et al. [ITT02] simplifient l'idée de Messerges et proposent d'utiliser un petit ensemble de valeurs de masques précalculés. Ainsi, un petit nombre de tables SBox sont stockées et choisies aléatoirement au début de l'AES. La contre-mesure reste néanmoins assez coûteuse en temps et en espace mémoire.

Contre-mesure Split-Mask [Geb06]. L'auteur considère une SBox masquée et une table de masques M définies telles que :

$$\begin{aligned} \mathrm{SBox}'(v + r_1) &= \mathrm{SBox}(v) + r_v, \\ M(v + r_1) &= r_v + r_2. \end{aligned}$$

Les masques r_1, r_2 sont fixés, générés aléatoirement uniquement lorsque les tables SBox$'$ et M sont précalculées. Nous avons la relation $\mathrm{SBox}'(v + r_1) \oplus M(v + r_1) = \mathrm{SBox}(v) + r_2$ qui signifie que le masque r_2 est en fait découpé en r_v et $M(v + r_1)$. Le masque r_v peut être renouvelé au début de chaque AES. Le recalcul de la table M est alors facile à réaliser. La contre-mesure n'est pas très coûteuse en temps ou en mémoire mais Coron et Kizhvatov [CK09] ont récemment exposé une faille face aux attaques de premier ordre.

Exponentiation modulaire masquée [BGK05]. L'idée des auteurs consiste à calculer l'inverse de v dans $GF(2^8)$ par v^{254} en utilisant un algorithme d'exponentiation modulaire spécial. Les auteurs proposent les algorithmes *Perfectly Masked Squaring* et *Perfectly Masked Multiplication* pour effectuer respectivement l'élévation au carré et la multiplication tout en conservant le masquage des valeurs intermédiaires. Cette méthode est néanmoins très coûteuse en temps.

Masquage utilisant des tables log [TK05]. Soit γ un générateur de $GF(2^8)$. Nous précalculons toutes les paires (α, i) tel que $\alpha = \gamma^i$ pour $0 \leq i \leq 255$. Ces paires sont stockées dans deux tables telles que :

$$\log(\alpha) = i \quad \text{et} \quad \text{alog}(i) = \alpha.$$

Les opérations dans $GF(2^8)$ sont alors calculées en utilisant les tables log et alog. En particulier, la propagation du masque dans le calcul de l'inverse est désormais plus facile. Soit $v' = v + r$ la valeur v, masquée par une valeur aléatoire r, qui doit être inversée. Alors, avec la relation $v = \gamma^i$ et $r = \gamma^j$, nous avons que $v'^{-1} = (\gamma^i)^{-1}(\gamma^{j-i} + 1)^{-1}$. Le masque après l'inverse devient $(\gamma^{j-i} + 1)^{-1}$. Cette méthode requiert le stockage en mémoire des tables log et alog, ce qui peut poser un problème dans des environnements embarqués.

SBox résistante basée sur la transformation de Fourier [PGA06]. Nous identifions en premier l'espace vectoriel $GF(2^n)$ à $GF(2)^n$, pour une base quelconque, et ensuite à $\{0, \ldots, 2^n - 1\}$. Tout élément X de $GF(2^n)$ peut être écrit comme $X = (X_{n-1}, \ldots, X_0)$ et peut aussi être identifié à l'entier $\text{val}(X) = \sum_{0 \leq k < n} 2^k X_k$. Finalement, toute fonction $F : GF(2^n) \to GF(2^n)$ peut s'identifier à la fonction sur les entiers $X \mapsto \text{val}(F(X))$ toujours notée F malgré la confusion possible. Le produit scalaire classique sur les entiers $A \cdot X = \sum_{0 \leq k < n} A_k X_k$ dans $GF(2^n)$ permet d'identifier le groupe additif $GF(2^n)$ à son dual et ainsi la transformation de Fourier discrète (*Discrete Fourier Transform*, DFT), notée $\widehat{F}$, de la fonction F est aussi définie sur $GF(2^n)$ par :

$$\widehat{F}(X) = \sum_{A \in GF(2^n)} F(A)(-1)^{A \cdot X}.$$

Notons que $\widehat{F}$ est à valeur dans $\mathbb{Z}$ et que sa DFT donne la formule d'inverse suivante :

$$F(X) = \frac{1}{2^n} \sum_{A \in GF(2^n)} \hat{F}(A)(-1)^{A \cdot X}.$$

Prouff et al. [PGA06] observent que si la fonction F correspond à une SBox, alors la relation ci-dessus peut être utilisée pour calculer un *SubBytes* masqué. Néanmoins, Coron et al. dans [CGPR08] montrent une faiblesse face aux attaques par canaux cachés et proposent la transformation suivante qui utilise quatre masques R_1, R_2, R_3, R_4 de $GF(2^n)$ (identifié à $GF(2)^n$). Soit X le vecteur correspondant à la valeur d'entrée du *SubBytes*. Soit $\widetilde{X} = X \oplus R_1$

le vecteur masqué par R_1. La DFT masquée prend alors $\widetilde{X}$ en entrée et donne en sortie $F' := (-1)^{(\widetilde{X} \oplus R_2).R_1} F(X) + R_3 \bmod 2^n$ calculé par la formule :

$$F' = \left\lfloor \frac{1}{2^n} \Big(R' + \sum_{A \in GF(2^n)} \hat{F}(A)(-1)^{(A \cdot \widetilde{X}) + (R_1 \cdot (\widetilde{X} \oplus A \oplus R_2))} \bmod 2^{2n} \Big) \right\rfloor ,$$

où $R' = 2^n \mathrm{val}(R_3) + \mathrm{val}(R_4)$. Dans [LSK+09], Li et al. exposent une nouvelle faiblesse dans la proposition de Coron et al. due à un biais dans le masque utilisé.

Masquage affine [FMPR10]. Fumaroli et al. proposent une méthode de masquage résistante face aux attaques HODPA basée sur une transformation affine aléatoire sur les données. Soit $r_0 \in GF(256)$ et $r_1 \in GF(256)^*$ deux valeurs aléatoires. Soit G_{r_0,r_1} la transformation affine définie telle que :

$$\begin{aligned} G_{r_0,r_1} : GF(256) &\to GF(256) \\ x &\mapsto r_1.x + r_0. \end{aligned}$$

On peut donc considérer r_1 comme une masque multiplicatif et r_0 un masque additif appliqués sur la donnée x. Les auteurs combinent ainsi les deux techniques de masquage afin d'obtenir une meilleure résistance aux attaques. L'implémentation de cette contre-mesure requiert un certain nombre de tables précalculées stockées en mémoire afin que le surcoût en temps soit acceptable. Il faut précalculer la transformation G_{r_0,r_1} au début de l'algorithme étant donné qu'on considère que le couple (r_0, r_1) change à chaque exécution. Il est aussi nécessaire de précalculer la table SBox modifiée par G_{r_0,r_1}. La transformation inverse $G^{-1}_{r_0,r_1}$ peut se calculer à la volée assez efficacement. Toutes ces opérations requièrent des multiplications et inversions dans $GF(256)$ qui sont effectuées par des accès à des tables log et alog comme définies ci-dessus. Le surcoût de la contre-mesure est donc au total assez important ce qui s'explique par le fait que son but principal est de proposer une résistance aux attaques HODPA.

Isomorphismes aléatoires dans le corps de l'AES [RS05]. Les auteurs suggèrent l'utilisation de représentations aléatoires des éléments de $GF(2^8)$ comme protection face aux attaques par canaux cachés. Il existe 30 polynômes irréductibles de degré huit sur $GF(2)$ et chaque corps engendré par un polynôme a huit générateurs. Ainsi, il y a 240 représentations possibles du corps $GF(2^8)$ utilisé dans l'AES. Le principe consiste à choisir aléatoirement, au début du chiffrement, une de ces représentations, à transformer le message clair et à adapter les fonctions de ronde pour cette nouvelle construction de $GF(2^8)$. La sortie du chiffrement doit finalement être transformée dans le corps original de l'AES. Cette méthode requiert le changement de fonctions de ronde de l'AES pour chaque isomorphisme choisi et est donc très coûteuse.

Dans la suite, nous considérons les techniques utilisant l'arithmétique sur une tour d'extensions de corps finis. Cette construction est très efficace pour une implémentation *hardware*

étant donné que l'arithmétique sur de petits corps peut facilement s'implémenter en *hardware*.

Masque booléen dans une tour d'extensions [OMP04, OMPR05]. L'utilisation d'une tour d'extensions $GF(2^8) \supset GF(2^4) \supset GF(2^2)$ a d'abord été étudiée dans le cadre d'une implémentation rapide de l'AES [WOL02]. Le calcul de l'inverse est transféré dans un sous-corps de $GF(2^8)$. La protection de cette opération est alors effectuée à ce même niveau. Dans [OMPR05], Oswald et al. proposent une inversion masquée dans $GF(2^4)$ pour une implémentation *hardware*. Une version *software* de cette méthode est présentée dans [OS06]. Considérons une valeur d'entrée masquée dans $GF(2^8)$, vu comme une extension quadratique de $GF(2^4)$ et notée $GF(2^4)\square GF(2^4)$. Une fois la valeur mappée, on peut la représenter sous la forme : $(a_h + m_h)x + (a_l + m_l)$ avec $a_h, a_l, m_h, m_l \in GF(2^4)$. Soit $f_{a_h}, f_{a_l}, f_d, f_{d'}$ des fonctions dans $GF(2^4)$. Le calcul de l'inverse se fait par les formules suivantes :

$$\begin{aligned}
((a_h + m_h)x + (a_l + m_l))^{-1} &= (a'_h + m'_h)x + (a'_l + m'_l) \\
a'_h + m'_h &= f_{a_h}((a_h + m_h), (d' + m'_d), m_h, m'_h, m'_d) \\
&= a_h \times d' + m'_h \\
a'_l + m'_l &= f_{a_l}((a'_h + m'_h), (a_l + m_l), (d' + m'_d), m_l, m'_h, m'_l, m'_d) \\
&= (a_h + a_l) \times d' + m'_l \\
d + m_d &= f_d((a_h + m_h), (a_l + m_l), \lambda, m_h, m_l, m_d) \\
&= a_h^2 \times \lambda + a_h \times a_l + a_l^2 + m_d \\
d' + m'_d &= f_{d'}(d + m_d, m_d, m'_d) \\
&= d^{-1} + m'_d,
\end{aligned}$$

où $\lambda \in GF(2^4)$ provient du polynôme $x^2 + x + \lambda$ utilisé pour construire $GF(2^4)\square GF(2^4)$. Les fonctions f sont détaillées dans [OS06] grâce à des tables précalculées pour une implémentation *software* et dans [OMPR05] pour une implémentation *hardware*.

Bien que Oswald et al. prouvent que, grâce à leur méthode, toutes les valeurs intermédiaires sont masquées, leur étude n'est faite qu'au niveau algorithmique. Dans une implémentation *hardware* de la contre-mesure, Mangard et al. [MPO05] montrent une vulnérabilité aux attaques de premier ordre. La faiblesse serait due à des glitches qui apparaissent dans les composants CMOS.

3.3 Masquage avec une construction aléatoire de tour d'extensions de corps finis

Nous nous intéressons à un AES utilisant une construction de tour d'extensions de corps finis. Nous proposons une contre-mesure face aux attaques de premier ordre basée sur cette structure algébrique.

3.3.1 Calcul de l'inverse dans $GF(2^8)$

Le polynôme irréductible $R(z) = z^8 + z^4 + z^3 + z + 1$, spécifié dans l'AES, est utilisé pour définir le corps de Galois $GF(2^8)$. Sauf mention contraire, nous considérons par la suite que $GF(2^8)$ est construit par le polynôme fixé $R(z)$. L'opération *SubBytes* est la seule partie non-linéaire de l'AES. Elle prend en entrée un élément $X \in GF(2^8)$. L'opération est composé de :

1. un calcul multiplicatif dans $GF(2^8)$, $Y = X^{-1}$ en fixant l'inverse de $X = 0$ à $Y = 0$,
2. une transformation affine définie par $f(Y) = AY + B^T$ avec :

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{pmatrix} \quad \text{et} \quad B = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

L'algorithme AES considère des vecteurs de 8 bits $a = (a_7, \ldots, a_0)$ comme des éléments de $GF(2^8)$ représentés en base polynômiale : $a_7 z^7 + \cdots + a_1 z + a_0$. Un élément a est ainsi représenté dans l'AES par un vecteur colonne ${}^t[a_7, \ldots, a_0]$. L'inverse d'un polynôme de degré sept modulo un polynôme de degré huit est une opération très coûteuse. Afin d'améliorer son efficacité, nous calculons l'inverse dans $GF(2^4)\square GF(2^4)$, une extension quadratique de $GF(2^4)$. En effet, l'inverse d'un polynôme de degré un modulo un polynôme de degré deux est beaucoup plus simple et, dans notre cas, ne requiert que des opérations dans $GF(2^4)$. Pour construire $GF(2^4)\square GF(2^4)$, nous considérons un polynôme quadratique primitif de la forme $P(x) = x^2 + \psi x + \lambda$ avec $\psi, \lambda \in GF(2^4)$. Chaque élément $a \in GF(2^8)$ est alors représenté comme un élément de $GF(2^4)\square GF(2^4)$: $a_h x + a_l$ avec $a_h, a_l \in GF(2^4)$. Finalement, le corps $GF(2^4)$ est construit par un polynôme primitif de degré quatre noté $Q(y)$.

La formule permettant de calculer l'inverse dans $GF(2^4)\square GF(2^4)$ est :

$$(a_h x + a_l)^{-1} = (a_h \otimes d^{-1})x + (\psi a_h \oplus a_l) \otimes d^{-1} \tag{3.1}$$

$$\text{avec} \quad d = (a_h^2 \otimes \lambda) \oplus (\psi \otimes a_h \otimes a_l) \oplus a_l^2, \tag{3.2}$$

où $\otimes$ et $\oplus$ sont respectivement la multiplication et l'addition dans $GF(2^4)$. L'élément $d \in GF(2^4)$ est en fait la norme de $a \cong a_h x + a_l$. Soit N la fonction norme telle que :

$$\begin{aligned} N : (GF(2^4)\square GF(2^4))^* &\to GF(2^4)^* \\ (a_h, a_l) &\mapsto (a_h^2 \otimes \lambda) \oplus (\psi \otimes a_h \otimes a_l) \oplus a_l^2. \end{aligned}$$

La performance d'une tour d'extensions de la forme $\mathbb{F}_{((2^2)^2)^2}$, dans le cas de l'AES, a été étudiée en détail. Dans [MS02], les auteurs choisissent une base polynômiale alors que Canright [Can05] étudie le cas d'une base normale. Récemment, Nogami et al. [NNT+10] proposent même l'utilisation de bases mixtes pour une meilleure efficacité.

3.3.2 Trouver un isomorphisme entre $GF(2^8)$ et $GF(2^4)\square GF(2^4)$

Soit $X \in GF(2^8)$. En considérant le *SubBytes* dans une tour d'extensions, nous obtenons :

$$A\left(M^{-1}\left((MX)^{-1}\right)\right) + B^T, \tag{3.3}$$

avec A, B les paramètres de la transformation affine définis précédemment, M la matrice définie par le passage de $GF(2^8)$ vers $GF(2^4)\square GF(2^4)$ via les bases de références induites par les constructions du corps à 2^8 éléments et M^{-1} la matrice inverse. Nous notons que l'élément MX est dans $GF(2^4)\square GF(2^4)$ où l'inverse est calculé. Nous étudions l'effet d'une modification de la matrice M sur la résistance aux attaques par canaux cachés.

Soit α et θ des racines des polynômes $P(x)$ et $Q(y)$ respectivement. Les éléments de $GF(2^4)\square GF(2^4)$ et $GF(2^4)$ sont alors représentés comme une combinaison linéaire d'éléments dans les bases respectives $\{1, \alpha\}$ et $\{1, \theta, \theta^2, \theta^3\}$. Comme $P(x)$ et $Q(y)$ sont primitifs, les éléments peuvent aussi être représentés respectivement par les α^i ou les θ^j avec $i \in \{0, \ldots, 254\}$ et $j \in \{0, \ldots, 15\}$.

Afin de générer un isomorphisme entre $GF(2^4)\square GF(2^4)$ et $GF(2^8)$, il faut transformer α l'élément primitif de $GF(2^4)\square GF(2^4)$, en γ un élément primitif de $GF(2^8)$ en respectant l'homomorphisme de corps. L'algorithme suivant, proposé dans [RDJ+01], permet de trouver un tel isomorphisme :

1. Choisir un des $\phi(255) = 128$ éléments primitifs $\gamma \in GF(2^8)$. Calculer α^i et γ^i pour $i \in [0, \ldots, 254]$.
2. Vérifier que, pour $i = 1$, il existe un r tel que $\alpha^r = \alpha^i + 1$ et $\gamma^r = \gamma^i + 1$. Si c'est le cas, nous avons un isomorphisme entre $GF(2^8)$ et $GF(2^4)\square GF(2^4)$. Sinon, il faut répéter les deux premières étapes pour l'élément primitif γ suivant.
3. Construire la matrice de passage M de $GF(2^8)$ vers $GF(2^4)\square GF(2^4)$ et la matrice inverse M^{-1}.

3.3.3 Choix des paramètres

La construction d'un isomorphisme de $GF(2^8)$ vers $GF(2^4)\square GF(2^4)$ dépend des paramètres suivant :

- le polynôme $Q(y)$ utilisé pour construire $GF(2^4)$,
- le polynôme $P(x)$ utilisé pour construire $GF(2^4)\square GF(2^4)$,
- l'élément primitif $\gamma \in GF(2^8)$ utilisé dans l'algorithme ci-dessus.

Choix de $Q(y)$. Le polynôme $Q(y)$ est primitif, de degré 4, et à coefficients dans $GF(2)$. Il existe deux polynômes ayant ces propriétés : $y^4 + y + 1$ et $y^4 + y^3 + 1$. Le calcul des multiplications, élévations au carré et inversions dans $GF(2^4)$ dépend de ce polynôme.

Choix de $P(x)$. Le polynôme $P(x)$ est primitif, de degré 2, à coefficients dans $GF(2^4)$. Pour un $Q(y)$ donné, il existe 64 tels polynômes de la forme $P(x) = x^2 + \psi x + \lambda$ avec $\psi, \lambda \in GF(2^4)$. Les coefficients ψ et λ sont utilisés dans le calcul d'inverse dans $GF(2^4)\square GF(2^4)$ (3.1) et (3.2).

Exemple 3.3.1. *Soit $Q(y) = y^4 + y + 1$. Il existe 4 polynômes primitifs $P(x)$ de la forme $x^2 + x + \lambda$:*

- $x^2 + x + \omega^7$,
- $x^2 + x + \omega^{11}$,
- $x^2 + x + \omega^{13}$,
- $x^2 + x + \omega^{14}$.

Choix de γ. Un élément primitif γ de $GF(2^8)$ est utilisé pour trouver un isomorphisme avec $GF(2^4)\square GF(2^4)$. Pour un $Q(y)$ et un $P(x)$ fixés, il existe 8 tels éléments.

Dans [WOL02], les auteurs choisissent $Q(y) = y^4 + y + 1$ et $P(x) = x^2 + x + \omega^{11}$ alors que dans [RDJ+01], les auteurs préfèrent le même $Q(y)$ avec $P(x) = x^2 + x + \omega^{14}$.

3.3.4 Construction aléatoire de tour d'extensions

Soit $R(z) = z^8 + z^4 + z^3 + z + 1$ le polynôme irréductible spécifié dans l'AES. Le polynôme $Q(y)$ est aussi fixé car les opérations dans $GF(2^4)$ sont généralement soit précalculées dans des tables en mémoire, soit implémentées en *hardware*. Soit $X \in GF(2^8)$, nous pouvons représenter cet élément différemment en modifiant la construction de $GF(2^4)\square GF(2^4)$. Comme nous l'avons vu précédemment, différents polynômes $P(x)$ et éléments primitifs γ peuvent être choisis dans ce but. En fait, 64 polynômes $P(x)$ peuvent être combinés avec 8 éléments γ pour un total de 512 constructions possibles, et donc 512 matrices de passages notées M_i (3.3).

Nous laissons de côté le cas de l'élément $0 \in GF(2^8)$ car notre méthode ne permet pas de le masquer. Ainsi, notre proposition doit être utilisée conjointement avec une méthode de masquage booléen comme celles proposées dans [OMP04, OMPR05].

La partie la plus sensible du *SubBytes* est l'inverse dans $GF(2^4)$ (3.2) qui correspond à l'inverse de la norme d'un élément de $GF(2^8)$. Nous nous intéressons d'abord à la distribution des valeurs de $X \in GF(2^8)$ en considérant les 512 représentations possibles (Figure 3.1). Nous remarquons que les éléments de $GF(2^8)$ sont représentés par, au plus, 234 éléments distincts de $GF(2^4)\square GF(2^4)$. Ainsi, même en considérant tous les isomorphismes possibles, il est impossible d'obtenir les 256 valeurs possibles de $GF(2^4)\square GF(2^4)$.

Soit θ un élément primitif de $GF(2^4)$ tel que chaque élément du corps est représenté par θ^j pour un certain j. Nous étudions plus précisément les valeurs possibles de $N(X)$ (3.2) qui sont sensibles aux attaques. La distribution de $N(X)$ pour tout $X \in GF(2^8)$ et pour tous les isomorphismes possibles est présenté dans la Figure 3.2. Soit S_i un ensemble de valeurs de $GF(2^4)$ tel que $N(X) \in S_i$ pour un certain i. Nous remarquons qu'il existe six ensembles S_i définis tels que :

- $S_1 = \{1\}$ pour 17 valeurs $X \in GF(2^8)$,

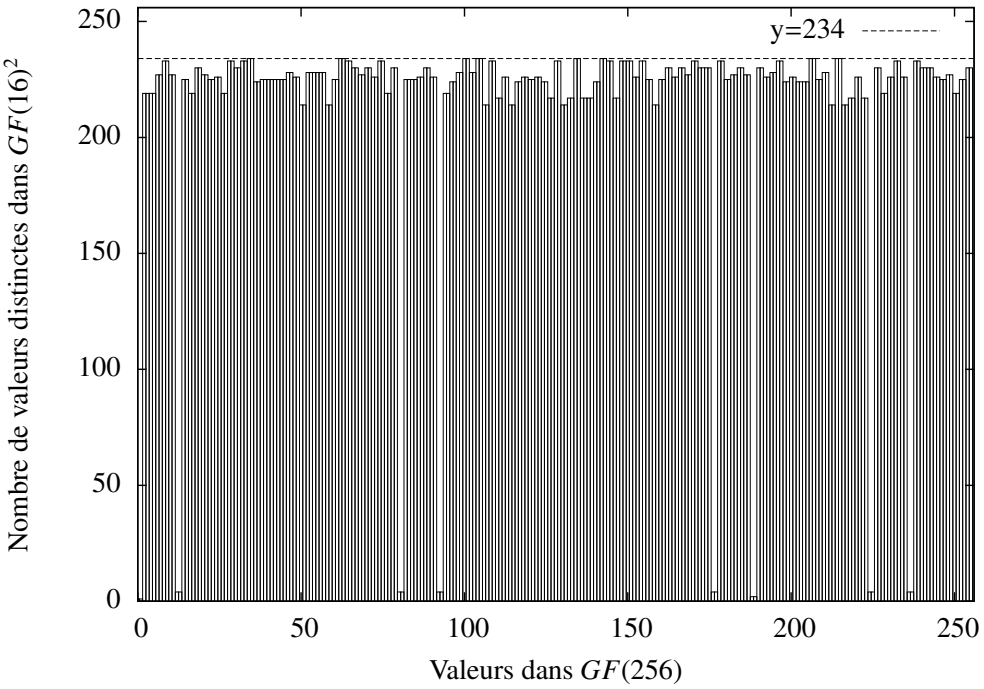


FIGURE 3.1 – Distribution des valeurs $x \in GF(256)$ en considérant les 512 représentations possibles.

- $S_2 = \{\theta^5, \theta^{10}\}$ pour 34 valeurs $X \in GF(2^8)$,
- $S_3 = \{\theta, \theta^2, \theta^4, \theta^8\}$ pour 68 valeurs $X \in GF(2^8)$,
- $S_4 = \{\theta^3, \theta^6, \theta^9, \theta^{12}\}$ pour 68 valeurs $X \in GF(2^8)$,
- $S_5 = \{\theta^7, \theta^{11}, \theta^{13}, \theta^{14}\}$ pour 68 valeurs $X \in GF(2^8)$.

Ces résultats impliquent que pour un $X \in GF(2^8)$ donné, sa norme atteint seulement quatre valeurs distinctes au maximum. En augmentant le nombre d'isomorphismes considérés, nous avons remarqué que le nombre de représentations distinctes pour X augmente aussi. Néanmoins, lorsque nous étudions la norme de X, nous constatons que les 512 isomorphismes possibles n'entraînent que quatre normes différentes, au plus, pour un X donné. De plus, nous pouvons atteindre cette taille maximale pour les ensembles S_i grâce à n'importe quel polynôme $P(x)$ et en ne considérant que quatre des huit éléments primitifs γ pour ce $P(x)$ fixé. Ces observations sont confirmées dans la section 3.4 grâce à des résultats expérimentaux.

3.3.5 Proposition de contre-mesure en améliorant la distribution de la norme

Notre objectif est de maximiser le nombre de normes distinctes pour un $X \in GF(2^8)$ donné. Nous venons de voir qu'une construction aléatoire de tour d'extensions de corps finis ne permet que d'obtenir, au maximum, quatre normes distinctes pour un X donné. Nous aimerions que la norme d'un élément puisse appartenir à plus d'un seul ensemble S_i

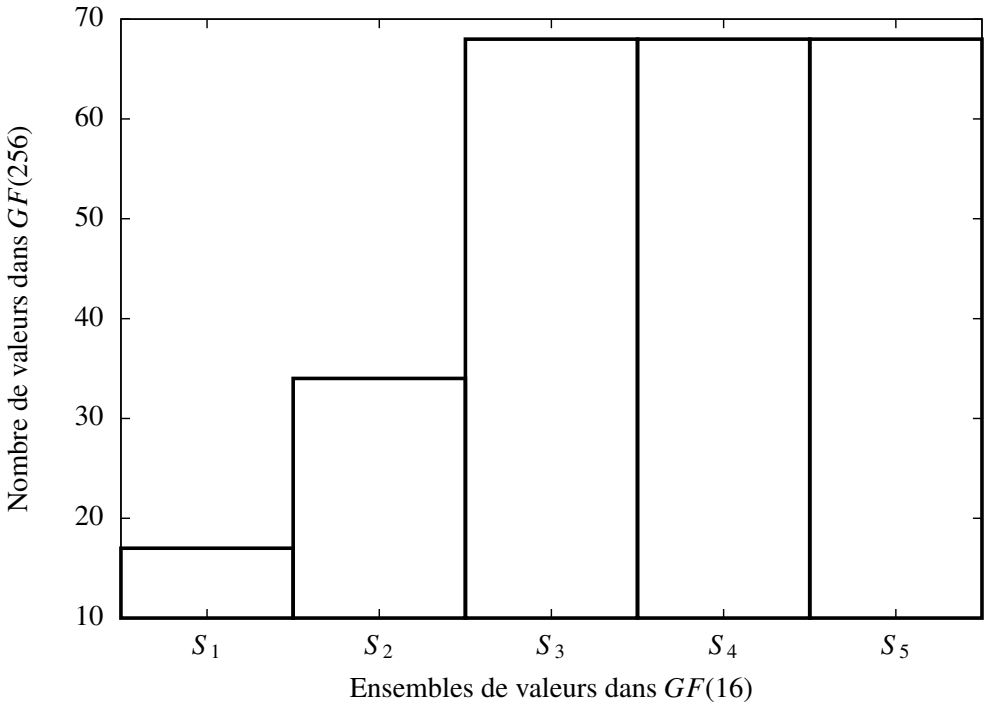


FIGURE 3.2 – Cette Figure représente le nombre d'éléments de $GF(256)$ dans chacun des ensembles de valeurs de normes si on considère tous les isomorphismes possibles.

afin d'augmenter le nombre de normes distinctes d'un élément X.

Nous observons une relation entre l'ordre d'un élément $X \in GF(2^8)$ et les ensembles S_i :

- $N(X) \in S_1 \Leftrightarrow \text{ord}(X) \in \{1, 17\}$,
- $N(X) \in S_2 \Leftrightarrow \text{ord}(X) \in \{3, 51\}$,
- $N(X) \in S_3 \Leftrightarrow \text{ord}(X) \in \{15, 255\}$,
- $N(X) \in S_4 \Leftrightarrow \text{ord}(X) \in \{5, 85\}$,
- $N(X) \in S_5 \Leftrightarrow \text{ord}(X) \in \{15, 255\}$.

Afin que la norme d'un élément puisse appartenir à plusieurs S_i, l'ordre de X doit être modifié. Si nous connaissions son ordre, juste avant de calculer sa norme, nous pourrions multiplier X par un autre élément U d'ordre adéquat afin que l'ordre de XU permette d'obtenir des normes d'un ensemble S_i différent. Néanmoins, en pratique, il n'est pas possible de connaître aisément la norme de X. De plus, cette méthode pourrait ne pas être sûre, d'un point de vue des attaques par canaux cachés, étant donné que le choix de U dépendrait de la valeur de X.

L'ensemble des éléments d'ordre i dans $GF(2^8)$ est noté O_i. Soit O' l'ensemble des éléments de O_3 plus deux éléments de O_{17} tirés au hasard, tel que $|O'| = 4$. Nous notons que $|O_5| = 4$. Notre proposition pour l'étape d'inversion d'un élément $X \in GF(2^8)$ consiste à :

1. choisir aléatoirement entre quatre matrices de passage M_i,

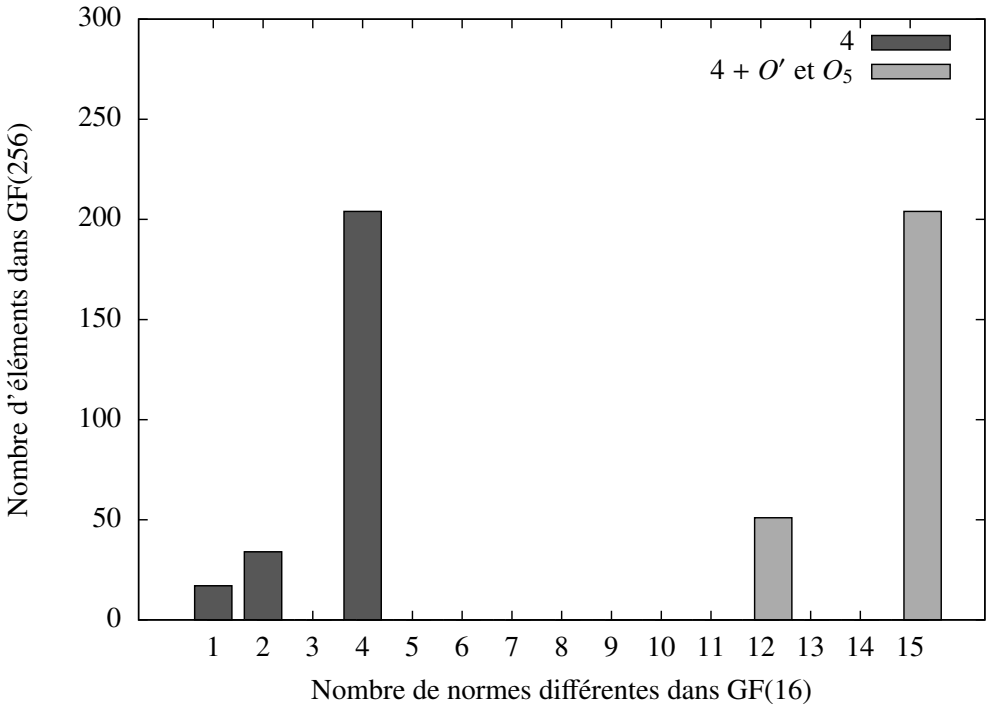


FIGURE 3.3 – Répartition des valeurs de $N(X)$ en considérant tous les X possibles pour quatre isomorphismes et répartition des valeurs de $N(XUV)$ en considérant tous les X possibles pour quatre isomorphismes et pour U et V pris dans les ensembles O' et O_5.

2. choisir aléatoirement $U \in O'$ et $V \in O_5$,
3. calculer $X' := M_i(XUV)$,
4. calculer la norme $N(X')$, puis son inverse $N(X')^{-1}$,
5. obtenir la norme inverse correcte par $N(X)^{-1} = N(X')^{-1} \otimes N(U) \otimes N(V)$ avec $N(U)$ et $N(V)$ précalculés pour tout $U \in O'$, $V \in O_5$ et pour chaque isomorphisme M_i,
6. une fois la norme inverse correcte calculée, les formules de calcul d'inverse dans $GF(2^8)$ sont similaires à (3.1).

Le choix des ensembles O' et O_5 semble être le mieux adapté. Nous calculons pour chaque $X \in GF(2^8)$ toutes ses normes possibles en utilisant cette proposition. Les résultats sont présentés dans la Figure 3.3. Nous observons une nette amélioration grâce à l'utilisation de ces ensembles. En effet, en utilisant seulement une randomisation parmi les quatre représentations des éléments de $GF(2^8)$ nous obtenons, comme noté précédemment, uniquement quatre normes distinctes. En ajoutant notre proposition de multiplication par des éléments de O' et O_5, nous obtenons que, pour la plupart des X, leur norme peut avoir le maximum possible de valeurs dans $GF(2^4)^*$, soit quinze valeurs différentes.

Si nous considérons une implémentation du *SubBytes* avec une construction de tour d'extensions de corps, le stockage mémoire supplémentaire nécessaire pour notre proposition est limité. Les quatre matrices de passage de $GF(2^8)$ vers $GF(2^4)\square GF(2^4)$ et les

quatre matrices inverses sont stockées sur 64 octets. Les éléments des ensembles O' et O_5 font 8 octets. Enfin, leurs normes inverses pour chacune des représentations sont stockées en 32 octets. L'augmentation de la complexité en temps du *SubBytes* est aussi limitée puisque nous ajoutons seulement deux multiplications dans $GF(2^8)$ et deux multiplications dans $GF(2^4)$.

Comme nous l'avons auparavant noté, notre méthode ne résout pas le problème du masquage de la valeur zéro. Néanmoins, elle s'utilise naturellement en combinaison avec la proposition de masquage booléen de Oswald et al. [OMP04, OMPR05] afin d'obtenir une contre-mesure plus robuste face aux attaques par canaux cachés de premier ordre.

3.4 Résultats expérimentaux de cette contre-mesure

Afin d'évaluer notre proposition de contre-mesure, nous effectuons des attaques par canaux cachés sur un AES *software* implémenté pour une architecture 8 bits. La consommation de courant lors du calcul sensible $N(x)$ est enregistrée. Ces courbes sont ensuite utilisées pour des attaques par analyse différentielle. Nous utilisons le coefficient de Pearson comme test statistique (voir section 2.2.3.2) étant donné ses très bonnes performances sur des composants CMOS. Pour valider la contre-mesure, nous plaçons l'attaquant dans le meilleur scénario possible. Nous utilisons un simulateur de consommation de courant pour obtenir les courbes. Ainsi, le simulateur renvoie, à chaque cycle d'horloge, le poids de Hamming de la donnée traitée à cet instant. La consommation est donc parfaitement linéaire en le poids de Hamming des données. Le bruit gaussien qui apparaît lors des mesures de consommation sur composant est donc éliminé.

L'efficacité des attaques par canaux cachés est évaluée grâce à deux métriques proposées dans la littérature [SGV08] :

- la *guessed entropy*, qui est la position, en moyenne, de l'hypothèse correcte dans le vecteur d'hypothèses triées à la sortie d'une attaque,
- le *first-order success rate* qui est, pour un nombre de traces données, la probabilité que l'hypothèse correcte soit en première position dans le vecteur d'hypothèses triées.

Nous évaluons dans nos expérimentations l'effet d'une construction aléatoire de tour d'extensions de corps en considérant un différent nombre d'isomorphismes. Nous considérons en premier un polynôme fixé $P(x)$ de la forme $x^2+x+\lambda$ et nous fixons arbitrairement un isomorphisme entre $GF(2^8)$ et $GF(2^4)\square GF(2^4)$. Il s'agit d'une implémentation classique d'AES utilisant un tour d'extension de corps. Une autre attaque est effectuée en choisissant quatre des huit éléments primitifs de $GF(2^8)$ tel que nous prenons uniquement les éléments sans leur conjugué. L'attaque suivante est réalisée en considérant les huit isomorphismes obtenus avec les huit éléments primitifs. Pour ces trois attaques, nous considérons que l'attaquant connaît le polynôme $P(x)$ qui a été fixé. S'il lui était inconnu, l'attaquant n'aurait qu'à essayer tous les 64 polynômes quadratiques $P(x)$ et réaliser l'attaque autant de fois. Ces calculs supplémentaires ne sont pas pris en compte dans les résultats d'attaques. Nous réalisons ensuite une attaque en choisissant aléatoirement un polynôme primitif de la forme $x^2+x+\lambda$ et un des huit éléments primitifs de $GF(2^8)$ pour un total de 32 possibilités.

Cette méthode aurait pu être intéressante car, étant donné que $\psi = 1$, les formules de calcul d'inverse sont plus simples. L'attaque suivante est réalisée en considérant tous les 512 isomorphismes possibles utilisant les 64 polynômes $P(x)$. Finalement, nous examinons la résistance de notre dernière proposition en utilisant seulement quatre isomorphismes et en multipliant par des éléments de O' et O_5.

Les attaques sont effectuées sur 20 ensembles de 500 courbes pour que les métriques soient moins biaisées. Les résultats sont présentés dans la Figure 3.4. L'implémentation classique, sans masquage, d'AES utilisant une tour d'extensions de corps est cassée avec, à peu près, 50 courbes. En utilisant quatre isomorphismes, nous notons une légère amélioration. Les deux métriques indiquent qu'un attaquant a besoin de six fois plus de courbes pour casser l'implémentation par rapport à la classique. Avec 8, 32 ou 512 isomorphismes les résultats ne s'améliorent pas. Cette observation confirme les propriétés de la norme observées dans la section 3.3.4 : quatre isomorphismes sont suffisants pour obtenir le maximum de valeurs de normes distinctes pour un élément donné. Finalement, l'amélioration apportée par notre dernière proposition est assez nette. Combiner les quatre isomorphismes et la multiplication par des éléments de O' et O_5 donne une bonne résistance face aux attaques par canaux cachés pour un faible surcoût en termes de performances et stockage mémoire.

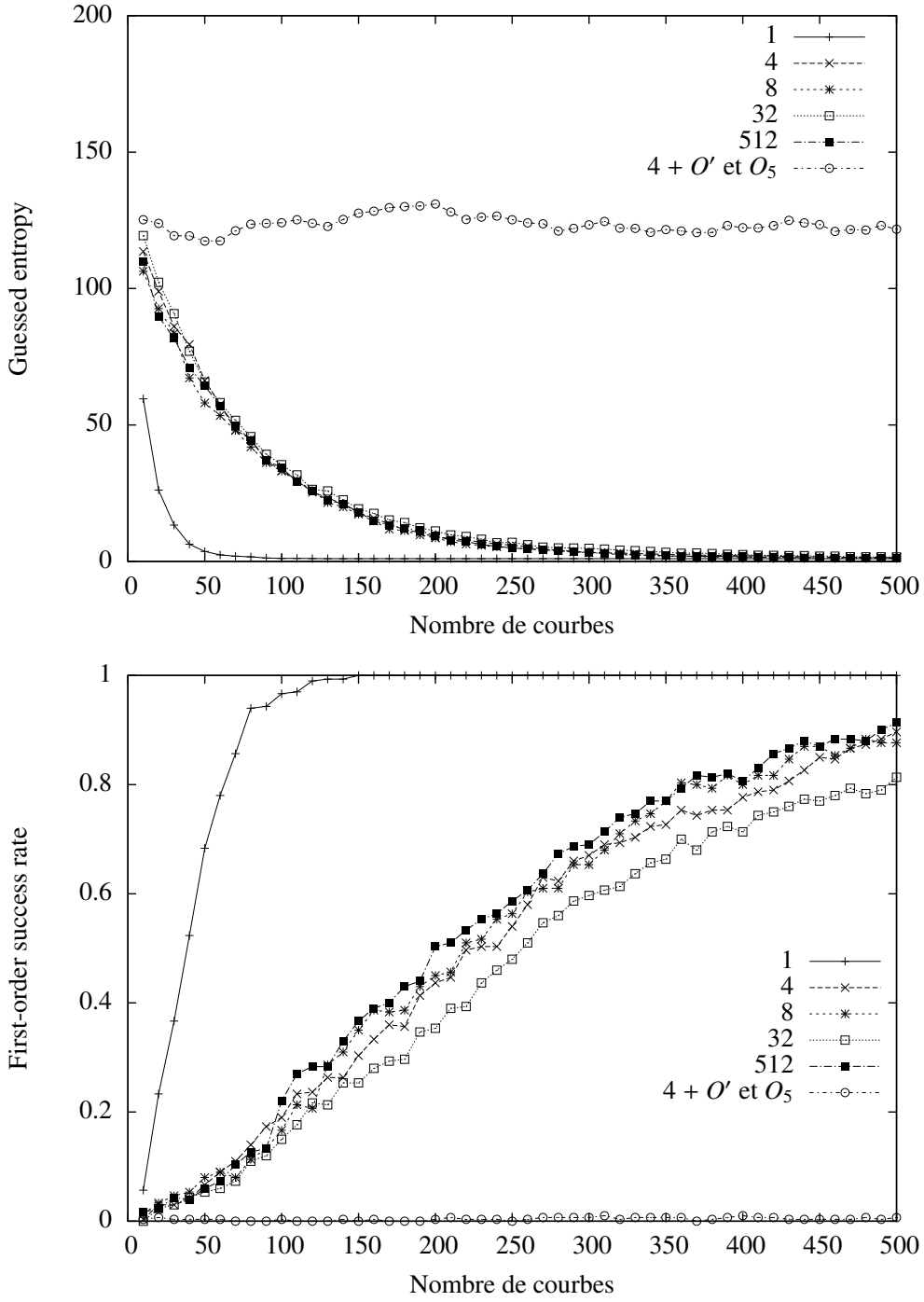


FIGURE 3.4 – Résultats d'attaques par analyse de courant utilisant les métriques *guessed entropy* et *first-order success rate*. Nous nous intéressons à des implémentation d'AES basées sur une construction de tour d'extensions de corps. Nous comparons une implémentation : classique, utilisant 4 matrices de passages aléatoires, 8 matrices de passages aléatoires, 32 matrices de passages aléatoires, 512 matrices de passages aléatoires et notre dernière proposition avec 4 matrices de passages aléatoires plus des multiplications par des éléments de O' et O_5.

Chapitre 4

Attaque différentielle par analyse d'information mutuelle

L'information mutuelle est une méthode statistique, alternative de celles présentées section 2.2.3, proposée dans le cadre des attaques par canaux cachés par Gierlichs et al. [GBTP08]. Nous développons en détail dans cette section l'utilisation de cette mesure. Ces travaux ont fait l'objet de publications [Ven10b, Ven10a].

4.1 Rappels de théorie de l'information

4.1.1 Information mutuelle classique

En théorie de l'information, l'information mutuelle (IM) est définie comme une mesure de la dépendance mutuelle entre deux variables. Contrairement au coefficient de corrélation de Pearson, elle est aussi sensible aux relations entre variables qui n'apparaissent pas dans la covariance.

Soit X une variable aléatoire ayant un ensemble fini de M_X états possibles X_i avec $i = 1, \ldots, M_X$. Soit $\mathbb{P}_X$ la distribution de probabilités de X. L'entropie de Shannon de X, notée $H(X)$ ou $H(\mathbb{P}_X)$, est définie comme :

$$H(X) = -\sum_{i=1}^{M_X} p(X_i) \log(p(X_i)), \tag{4.1}$$

avec $p(X_i)$ la probabilité de l'état X_i.

L'entropie conjointe $H(X, Y)$ de deux variables aléatoires X et Y est définie de la même manière :

$$H(X, Y) = -\sum_{i=1,j=1}^{M_X, M_Y} p(X_i, Y_j) \log(p(X_i, Y_j)). \tag{4.2}$$

L'entropie conditionnelle $H(X \mid Y)$ indique l'incertitude liée à X connaissant Y. Elle est définie comme :

$$H(X \mid Y_j) = -\sum_{i=1}^{M_X} p(X_i \mid Y_j) \log(p(X_i \mid Y_j)), \tag{4.3}$$

$$H(X \mid Y) = \sum_{j=1}^{M_Y} p(Y_j) H(X \mid Y_j), \tag{4.4}$$

avec $p(X_i \mid Y_j)$ la probabilité de l'état X_i conditionnelle à celle de Y_j.

L'information mutuelle $I(X;Y)$ est définie par :

$$I(X;Y) = H(X) - H(X \mid Y), \tag{4.5}$$

$$\text{ou} \quad I(X;Y) = H(X) + H(Y) - H(X,Y). \tag{4.6}$$

4.1.2 Information mutuelle généralisée

Soit X une variable aléatoire discrète définie comme ci-dessus. L'entropie de Rényi d'ordre α est :

$$\mathrm{H}_\alpha(X) = \begin{cases} \frac{1}{1-\alpha} \log \sum_{i=1}^{M_X} p(X_i)^\alpha & \text{for } \alpha \geq 0, \alpha \neq 1 \\ -\sum_{i=1}^{M_X} p(X_i) \log p(X_i) & \text{for } \alpha = 1. \end{cases} \tag{4.7}$$

On peut noter l'entropie de Shannon $\mathrm{H}_1(X)$. Grâce à ces définitions d'entropie de Rényi, on peut introduire la quantité :

$$\mathrm{I}_\alpha(X;Y) = \mathrm{H}_\alpha(X) + \mathrm{H}_\alpha(Y) - \mathrm{H}_\alpha(X,Y). \tag{4.8}$$

L'information mutuelle I_α a la propriété suivante :

$$\mathrm{I}_\alpha \geq 0 \quad \text{si et seulement si } \alpha = 0 \text{ ou } 1.$$

La valeur I_α ne correspond à la définition usuelle d'information mutuelle que dans ces deux cas.

L'entropie de Rényi H_2 est aussi appelée entropie de collision. Elle correspond à l'opposé du logarithme de la probabilité que deux variables aléatoires indépendantes, ayant la même distribution de probabilité, aient la même valeur. Les valeurs les plus probables ont plus tendance à entrer en collision et ont donc plus d'effets sur l'entropie de collision que sur l'entropie de Shannon.

Grâce au théorème de [PP93, Chapitre 3, Basic Theorem], les auteurs utilisent l'entropie de collision H_2 et appellent $\mathrm{I}_2(X;Y)$ information mutuelle généralisée (*Generalized Mutual Information*, GMI) où une variable, X ou Y, suit la distribution uniforme. La GMI et l'IM ont plusieurs propriétés en commun [PBHE98]. Toutes les deux :

– sont positives,

– détectent l'indépendance entre deux variables.

La GMI a donc les propriétés nécessaires pour notre étude. Nous discutons dans la section 4.5 de l'application de la GMI à notre étude et de l'algorithme pour la calculer.

4.2 Principe de l'attaque

Nous présentons brièvement l'attaque par analyse différentielle de courant utilisant l'information mutuelle (*Mutual Information Analysis*, MIA) proposée par Gierlichs et al. [GBTP08].

Pour comprendre la définition d'une fonction de sélection F, nous prenons comme exemple l'étude de l'algorithme DES. Soit x une partie du message clair et k une partie du secret. Des exemples, très utilisés en pratique, de valeurs intermédiaires attaquées sont :

- $F_1(x,k) = HW(Reg_L \oplus Reg_R)$ tel que Reg_L et Reg_R sont les valeurs sur 4 bits des registres L et R du DES, d'où $F_1(x,k) = i$, pour $i \in \{0 \dots 4\}$,
- $F_2(x,k) = HW(\text{SBox}(x \oplus k))$ tel que la sortie de la SBox du DES soit sur 4 bits, d'où $F_2(x,k) = i$, pour $i \in \{0 \dots 4\}$,
- $F_3(x,k) = \text{SBox}(x \oplus k)$ tel que $F_3(x,k) = i$, pour $i \in \{0 \dots 15\}$.

Soit $L(t)$ une variable aléatoire ayant pour valeurs des observations physiques du circuit électronique étudié à un instant t.

Supposons qu'un attaquant acquière N courbes de consommation de courant du circuit $\{l_{x_1}, \dots, l_{x_N}\}$ liées aux messages clairs $\{x_1, \dots, x_N\}$ et avec une clé secrète k fixée. Soit m la valeur maximale que peut prendre la fonction choisie par l'attaquant pour une valeur intermédiaire de l'algorithme. Par exemple, pour $F_1(x,k)$ nous avons $m = 4$. Le prototype d'une attaque par analyse d'information mutuelle est le suivant :

1. Nous estimons l'entropie $H(L(t))$ à chaque temps t.
2. Pour chaque hypothèse k' sur la clé secrète, nous appliquons la fonction F. Celle-ci a comme sortie une valeur comprise dans $\{0, \dots, m\}$. Comme vu précédemment, F prend en paramètre une partie de clé secrète k' et une partie du message clair. Nous appliquons donc N fois la fonction F pour chaque message clair x_i, $i = 1, \dots, N$.
3. Nous partitionnons les messages clairs x_i suivant la valeur de sortie de F dans les ensembles :

$$\mathcal{H}_j^{k'} = \{x_i \mid F(x_i, k') = j\} \text{ avec } j \in \{0, \dots, m\}\,.$$

   Nous avons de manière analogue une partition sur les observations :

$$\mathcal{G}_j^{k'} = \left\{ l_{x_i} \mid x_i \in \mathcal{H}_j^{k'} \right\}.$$

4. Pour $j \in \{0, \dots, m\}$, et à chaque instant t, nous estimons l'entropie $H(L(t) \mid F)$. Nous calculons en premier les entropies $H(L(t) \mid F = j)$ pour chaque $j \in \{0, \dots, m\}$ en utilisant (4.3) et les partitions $\mathcal{G}_j^{k'}$. Nous utilisons ensuite (4.4) pour trouver l'entropie conditionnelle $H(L(t) \mid F)$.
5. À chaque instant t, nous calculons pour une hypothèse de clé k' l'information mutuelle grâce à la définition (4.5) :

$$I_{k'}(L; F)(t) = H(L(t)) - H(L(t) \mid F).$$

Gierlichs et al. [GBTP08] démontrent que l'information mutuelle $I_{k'=k}(L;F)$ doit être maximum pour une hypothèse de clé k' correcte :

$$I_{k'=k}(L;F)(t) = \max_{k',t}(I_{k'}(L;F)(t)).$$

4.3 Techniques d'estimations d'entropie

Gierlichs et al. [GBTP08] présentent la MIA comme une alternative intéressante à la CPA étant donné que l'attaquant n'a pas à supposer un modèle de consommation pour le composant attaqué (voir section 1.2.3.3). En effet, l'information mutuelle enregistre à la fois les relations linéaires et non-linéaires entre les variables alors que la CPA ne mesure que les linéaires. En théorie, la MIA peut être considérée comme une attaque plus générique car l'attaquant a besoin de moins d'informations sur le composant. Néanmoins en pratique, les résultats de la MIA ne sont pas bons comparés à la CPA [VCS09, PR09, MMPS09]. L'efficacité de la MIA est en fait liée au choix de l'estimateur d'information mutuelle. Quelques auteurs ont étudié des estimateurs paramétriques et leur efficacité lorsqu'on les utilise avec la MIA [PR09, FGD$^+$10, LB10]. La performance des estimateurs non-paramétriques dans le cas de la MIA a été moins étudiée [Ven10b] alors que ceux-ci sont mieux adaptés au principe originel de la MIA.

4.3.1 Estimation paramétrique ou non-paramétrique

Il existe deux approches principales à l'estimation d'entropie : l'estimation paramétrique et non-paramétrique. Nous nous contentons d'étudier des méthodes non-paramétriques. Les méthodes paramétriques supposent plusieurs propriétés à propos des fonctions de régression qui décrivent la relation entre des variables. En effet, les densités de probabilité considèrent que les données proviennent d'une loi de probabilité connue, comme la loi normale. Les paramètres des fonctions de densité sont ainsi optimisés en faisant correspondre le modèle provenant de la loi de probabilité aux données. L'estimation non-paramétrique, au contraire, est une méthode où le choix des paramètres se fait sans aucune supposition sur la loi de probabilité des données. Nous présentons par la suite différentes méthodes d'estimation non-paramétrique efficaces dans le contexte d'attaques par canaux auxiliaires.

4.3.2 Utilisation d'histogrammes

Toutes les définitions de la section 4.1 supposent la connaissance de la loi de probabilité des données. Néanmoins, en pratique, les probabilités sont inconnues et doivent être estimées à partir de mesures. La méthode la plus simple et la plus utilisée est l'utilisation d'histogrammes.

Soit un ensemble de N mesures d'une variable aléatoire X. Les données sont découpées en B partitions. Ces partitions sont définies grâce à B intervalles $a_i = [o + ih, o + (i+1)h]$ tel que o est la valeur d'origine, h est la longueur des partitions et $i = 0, \ldots, B-1$. On

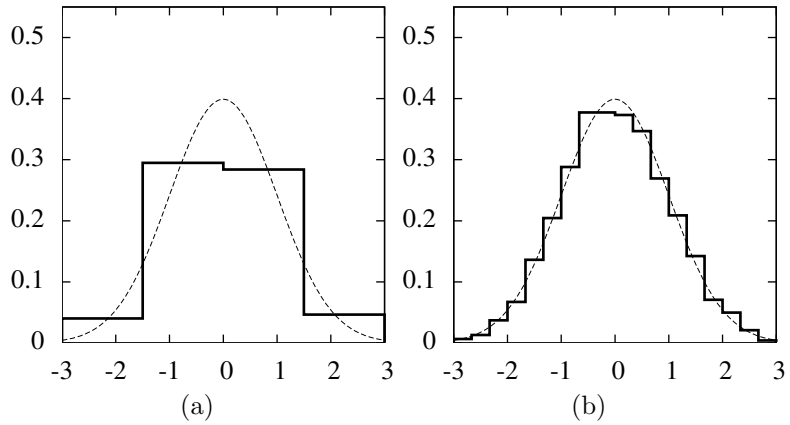


FIGURE 4.1 – Effet du nombre de partitions sur la correspondance entre l'estimation et la vraie loi de probabilité. Dans les deux figures, la ligne pointillée est une distribution gaussienne, la ligne pleine correspond à l'estimation. Figure 4.1a est une estimation avec 4 partitions. Figure 4.1b est une estimation avec 18 partitions.

note k_i le nombre de mesures qui appartiennent à l'intervalle a_i. Les probabilités $p(a_i)$ sont ensuite approximées grâce aux fréquences :

$$p(a_i) = \frac{k_i}{N}.$$

À partir de ces approximations, on peut calculer les entropies et l'information mutuelle. Le choix du nombre de partitions B est crucial. En effet, il s'agit d'un paramètre de lissage de l'estimation (Figure 4.1). Le nombre de partitions détermine à quel point l'approximation reflète la distribution idéale continue et à quel point le partitionnement correspond au traitement des données par le composant en pratique. L'estimation de densités grâce aux histogrammes se calcule très rapidement mais donne souvent des résultats très approximatifs.

4.3.3 Estimation par noyau

Il existe de nombreuses méthodes concurrentes à l'estimation par histogrammes. On s'intéresse à l'estimation par noyau [MRL95], aussi appelée méthode de Parzen [Par62], une technique bien connue qui donne généralement de bons résultats. Cette méthode suppose que la densité de probabilité soit assez lisse pour que les structures présentes en dessous d'une certaine fenêtre puissent être ignorées. Les noyaux se contentent essentiellement de mettre des poids aux distances de chaque point de l'échantillon à un point référence. Ces poids dépendent de la forme du noyau et de la fenêtre h utilisée. La méthode la plus simple est d'estimer une densité à un point x en comptant le nombre de points contenus dans une boîte centrée en x de taille h et en divisant par son volume. Au lieu de simplement compter les points, les noyaux sont utilisés pour donner des poids dépendants de la distance entre les

points. Un estimateur naïf $f(x)$, qui améliore tout de même l'estimation de la probabilité $p(x)$, peut s'écrire :

$$f(x) = \frac{1}{2Nh} \sum_{i=1}^{N} \Theta(h - |x - x_i|),$$

avec Θ la fonction de Heaviside définie par :

$$\Theta(z) = \begin{cases} 1 & \text{si} \quad z > 0 \\ 0 & \text{si} \quad z \leq 0. \end{cases}$$

Pour une définition plus générale, on note $K(x)$ le noyau. On définit alors un estimateur par noyau $f(x)$ comme :

$$f(x) = \frac{1}{Nh} \sum_{i=1}^{N} K\left(\frac{x - x_i}{h}\right). \tag{4.9}$$

Un exemple de noyau K souvent utilisé est le noyau gaussien, la fonction d'estimation est ainsi :

$$f(x) = \frac{1}{Nh\sqrt{2\pi}} \sum_{i=1}^{N} \exp\left(-\frac{(x - x_i)^2}{2h^2}\right).$$

Le noyau gaussien peut être vu comme le fait de placer de petites 'bosses' gaussiennes à chaque point x_i. L'estimation correspond alors à la somme des ces 'bosses'.

On se rappelle qu'un paramètre critique de l'estimation par histogrammes est le choix du nombre de partitions. Dans l'estimation par noyau, le choix de la valeur de la fenêtre h devient crucial. Si h est trop grand, l'estimation souffre de trop peu de précision alors que si h est trop petit, l'estimation a une trop grande variabilité statistique (Figure 4.2).

Même si l'estimation par noyau donne des résultats plus précis que les histogrammes, elle requiert une puissance de calcul importante.

4.3.4 k-plus-proches voisins

Kraskov et al. présentent dans [KSG04] un estimateur d'entropie basé sur la distance entre les K-plus-proches voisins (*K-Nearest Neighbors*, KNN). Soit X et Y des variables aléatoires. On considère l'espace à deux dimensions (X, Y) et, pour chaque point, on calcule une longueur telle que les k plus proches voisins de ce point soient dans cette longueur notée $\epsilon(i)$ pour le point i. Le nombre de points à distance $\epsilon(i)/2$ donne une estimation de la densité d'entropie jointe au point i. La distance est ensuite projetée dans le sous-espace de chacune des variables pour estimer l'entropie marginale de chaque variable. L'estimation KNN implique le choix du paramètre k qui n'est pas trivial comme pour la plupart des estimateurs non-paramétriques. Cette méthode donne de bons résultats avec moins d'erreurs statistiques que les précédentes mais avec un surcoût en temps calculatoire important.

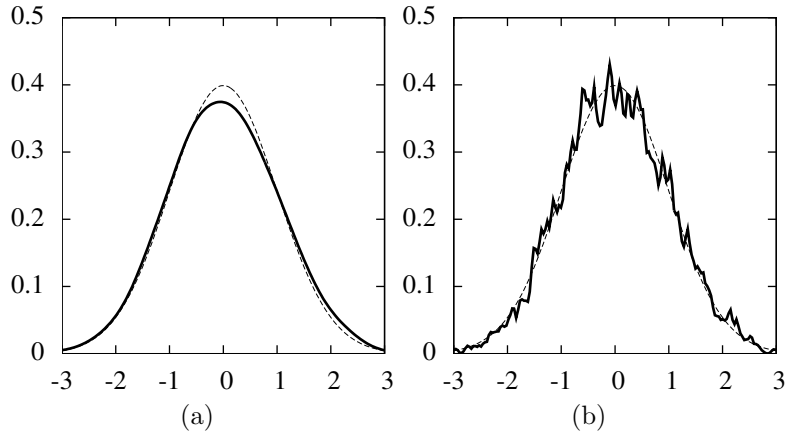


FIGURE 4.2 – Estimation par noyau utilisant le noyau de Heaviside avec différentes valeurs de fenêtres. Dans les deux figures, la ligne pointillée est une distribution gaussienne, la ligne pleine correspond à l'estimation. Figure 4.2a est une estimation avec pour fenêtre $h = 0.3$. Figure 4.2b est une estimation avec pour fenêtre $h = 0.03$.

On étudie dans la partie suivante un compromis entre efficacité et temps de calcul avec un estimateur de densités de probabilités à l'aide de B-splines. Grâce aux B-splines, on peut nettement améliorer les résultats donnés par l'estimation par histogrammes tout en conservant un faible temps de calcul.

4.4 Améliorer l'attaque par analyse d'information mutuelle grâce aux B-splines

Nous étudions dans ce chapitre l'application de la méthode d'estimation d'entropie par B-splines originellement proposée dans [DSSK04] dans le cadre des attaques par canaux cachés. Ce travail a fait l'objet de la publication suivante [Ven10b].

4.4.1 Introduction aux polynômes par morceaux et splines

Soit X une variable à une dimension. Une fonction polynômiale par morceaux $f(X)$ s'obtient en divisant le domaine de X en intervalles contigus, et en représentant f par un polynôme différent dans chaque intervalle. Figures 4.3a et 4.3b montrent des polynômes par morceaux simples. On préfère souvent des fonctions polynomiales plus lisses qui s'obtiennent en augmentant le degré des polynômes. Figure 4.3d correspond à une fonction polynômiale cubique, on l'appelle spline cubique. De manière générale, une spline de degré k avec des nœuds t_i, $i = 0, \ldots, m$ est un polynôme par morceaux de degré k et est deux fois continûment dérivable. Une spline cubique a pour degré $k = 4$. Ainsi, Figure 4.3a est une spline d'ordre 1 et Figure 4.3b est une spline d'ordre 2.

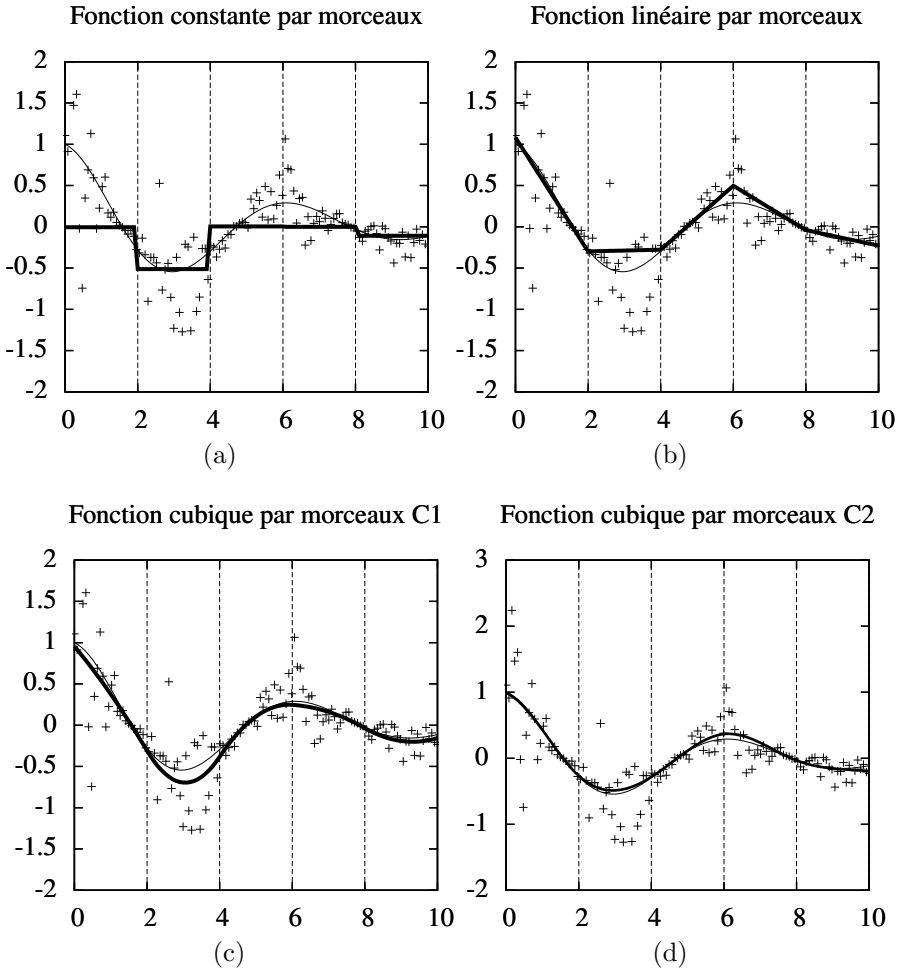


FIGURE 4.3 – Dans chacune des figures, les lignes pointillées correspondent à la position des nœuds. La ligne fine est la fonction $y(x) = \cos(x)\exp(-x/5)$. Les croix sont des données générées depuis la fonction $y(x)$ auxquelles on ajoute du bruit gaussien. La ligne épaisse représente l'estimation.

4.4.2 Calcul des B-splines

Nous introduisons les B-splines qui sont une généralisation des courbes de Bézier. Pour plus de détails sur les splines, le lecteur intéressé peut consulter [Deb78].

Une courbe B-spline définie sur l'intervalle $[a, b]$ est définie par :

- son degré d (ou ordre $k = d+1$), tel que chaque morceau du polynôme par morceaux est de degré d ou moins,
- une séquence de $m + 1$ entiers, $t_0, \ldots, t_m$, appelé vecteur de nœuds, tel que $t_i \leq t_{i+1}, \forall i \in \{1, \ldots, m-1\}$,
- des points de contrôles, $b_0, \ldots, b_n$.

Une courbe B-spline est définie en tant que fonctions B-spline de base. La i-ème fonction de base de degré d, notée $B_{i,d}$, déterminée par le vecteur de nœuds $t_0, \ldots, t_m$ est définie par récurrence par la formule de Cox-de Boor :

$$B_{i,0}(z) = \begin{cases} 1 & \text{si} \quad t_i \leq z < t_{i+1} \\ 0 & \text{sinon.} \end{cases} \tag{4.10}$$

$$B_{i,d}(z) = \frac{z - t_i}{t_{i+d} - t_i} B_{i,d-1}(z) + \frac{t_{i+d+1} - z}{t_{i+d+1} - t_{i+1}} B_{i+1,d-1}(z), \tag{4.11}$$

pour $i = 0, \ldots, n$ et $d \geq 1$.

La courbe B-spline de degré d avec points de contrôle $b_0, \ldots, b_n$, nœuds $t_0, \ldots, t_m$ est définie par :

$$B(z) = \sum_{i=0}^{n} b_i B_{i,d}(z),$$

avec $B_{i,d}(z)$ la fonction B-spline de base définie précédemment.

Grâce à (4.11), on peut noter que $B_{i,d}(z)$ est non-nul sur l'intervalle $[t_i, t_{i+d+1}]$. Par exemple, une fonction B-spline de base cubique $B_{i,3}(z)$ est non-nulle sur l'intervalle $[t_i, t_{i+4}]$. On peut aussi remarquer que, si les nœuds ne sont pas répétés, la B-spline est nulle aux nœuds extrêmes t_i et t_{i+d+1}. Mais les nœuds peuvent être répétés dans la définition d'une B-spline. Si le vecteur de nœuds contient un nombre suffisant de nœuds répétés alors une division par zéro pourra apparaître. On suppose donc que $0/0 := 0$. Finalement, la propriété la plus importante pour notre étude est la partition de l'unité d'une courbe B-spline :

$$\sum_{i=0}^{n} B_{i,d}(z) = 1, \quad \forall z.$$

On peut ainsi facilement adapter les fonctions B-spline pour être des estimateurs de densités de probabilité. Figure 4.4 montre des exemples de fonctions B-spline de base de différents degrés.

4.4.3 Estimation de la densité de probabilité avec des B-splines

Dans [DSSK04], les auteurs comparent la méthode d'estimation d'entropie à base de B-splines avec d'autres techniques d'estimation de probabilités. Ils montrent sur des données

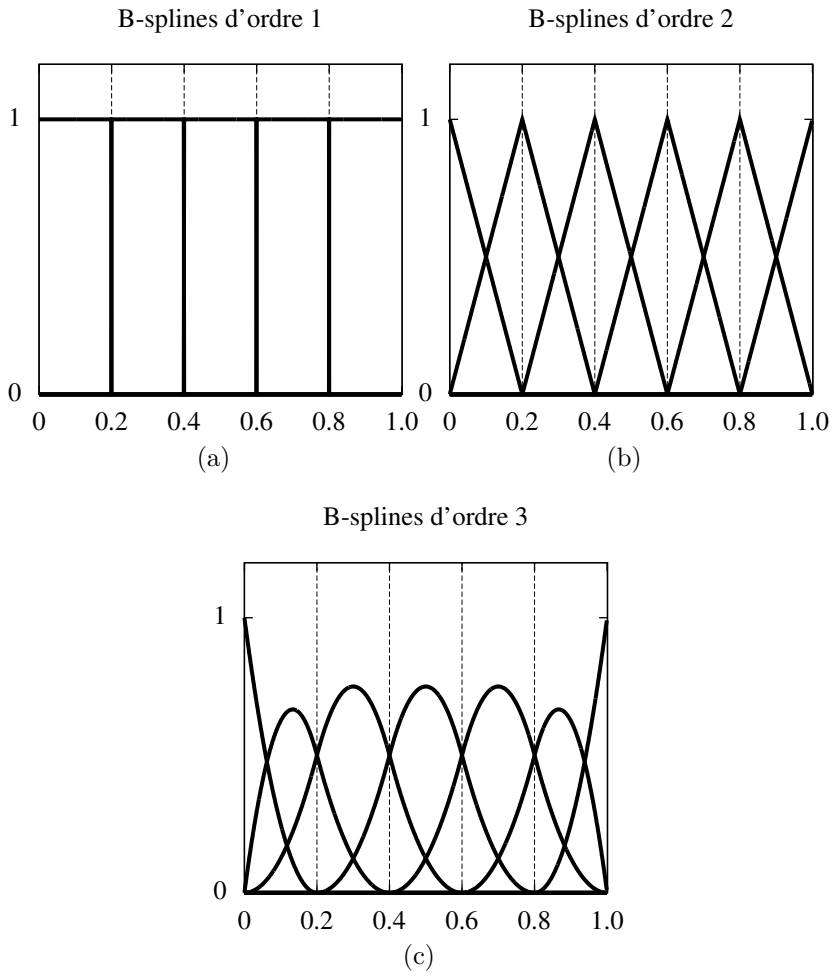


FIGURE 4.4 – Exemples de formes de fonctions B-splines de base $B_{i,k}(z)$ de degrés $k = 0$ (a), 1 (b), 2 (c) en utilisant le vecteur de nœuds $T = [0, 0.25, 0.5, 0.75, 1]$ avec $z \in [0, 1[$.

artificiellement générées que l'estimation par B-splines offre des résultats, en moyenne, deux fois meilleurs comparés à la technique d'histogrammes. Ils concluent qu'utiliser des degrés de splines trop élevés ne donnent pas forcément de meilleurs résultats qu'un ordre $k = 2$ ou $k = 3$. Nous utilisons pour la fin de l'article l'ordre $k = 3$ étant arrivé à la même conclusion que [DSSK04].

L'inconvénient majeur de la technique d'estimation par histogrammes est que chaque donnée n'est affectée qu'à une seule partition. On perd ainsi de l'information sur les données situées à la limite de deux partitions. En effet, suivant le bruit lié à cette donnée lors de la mesure par exemple, elle peut se retrouver arbitrairement dans l'une ou l'autre des partitions. L'idée principale de [DSSK04] est de permettre à une donnée d'être dans plusieurs partitions à la fois en utilisant des fonctions B-splines.

On veut reproduire une approche par histogrammes en remplaçant le partitionnement naïf de l'intervalle de valeurs par un partitionnement plus évolué grâce aux fonctions B-splines. Dans ces deux types de partitionnement, l'axe des abscisses est découpé en un certain nombre d'intervalles où chaque point limitant l'intervalle s'appelle point d'arrêt. Pour changer la forme d'une courbe B-spline, on a précédemment vu qu'on peut modifier : l'ordre de la courbe, les points de contrôles ou le vecteur de nœuds. Le nombre de points d'arrêt est liée à ces valeurs grâce à la formule : $narret = n - k + 2$ avec n le nombre de points de contrôles, ou fonctions de base, et k l'ordre de la courbe. L'ordre de la courbe B-spline est généralement fixé au préalable. On peut donc modifier le vecteur de nœuds et le nombre de points d'arrêt pour que les fonctions B-splines se comportent comme des partitions d'un histogramme. En général, les courbes B-splines ne sont pas tangentes aux nœuds extrêmes. Pour notre étude, nous voulons que les B-splines soient non-nulles à ces extrémités. On veut que les fonctions B-splines de base couvrent l'intervalle entier. Pour cela, on répète le premier et dernier nœud $d + 1$ fois dans le vecteur de nœuds.

Les courbes B-splines correspondantes à ces propriétés du vecteur de nœuds sont appelées courbes B-splines ouvertes. On les construit avec un vecteur de nœuds appelé vecteur de nœuds uniforme non-périodique. On utilise ce type de construction pour notre application à la MIA. Premièrement, nous définissons ce type de vecteur de nœuds.

4.4.4 Vecteur de nœuds uniforme non-périodique

Soit $B_{i,d}(z)$ une fonction B-spline de degré d (ordre $k = d + 1$) avec $i = 0, \ldots, n$ et $z \in [0, n - k + 2]$. On définit le vecteur de nœuds $t_0, \ldots, t_{n+k}$ tel que :

$$t_i = \begin{cases} 0 & \text{si} \quad 0 \leq \ i < k \\ i - k + 1 & \text{si} \quad k \leq \ i \leq n \\ n - k + 2 & \text{si} \quad n < \ i \leq n + k. \end{cases}$$

Par exemple, le vecteur de nœuds uniforme non-périodique pour $n = 5$ et $k = 3$ est $[0, 0, 0, 1, 2, 3, 4, 4, 4]$. En général, ce type de vecteur à la structure suivante :

$$\underbrace{0, \ldots, 0}_{k \text{ noeuds}}, 1, 2, \ldots, n - k - 1, \underbrace{n - k + 2, \ldots, n - k + 2}_{k \text{ nœuds}}.$$

4.4.5 Apport des B-splines dans le contexte des attaques par canaux cachés

Il existe une similitude importante entre l'estimation de densité par B-splines et la méthode par histogrammes puisque les B-splines d'ordre 1 sont en fait des fonctions escalier (Figure 4.4a). En effet, au lieu d'affecter une donnée à une seule partition, i.e. un intervalle, on peut la placer dans un intervalle plus grand et en lui affectant des poids grâce aux fonctions B-splines. Plus le degré d de la spline est élevé, plus l'intervalle considéré est grand. Cette particularité est notamment intéressante dans le contexte des attaques par canaux cachés. Chaque point d'une courbe de consommation est généralement composé d'une partie de bruit gaussien provenant de la méthode de mesure. Ce bruit peut faire passer un point d'une partition à une partition voisine alors erronée. L'estimation par B-splines affecte un poids à chaque donnée afin que celle-ci soit placée dans plusieurs intervalles voisins prenant ainsi en compte le possible bruit.

De plus, chaque point a un poids affecté par une courbe sur l'intervalle contrairement à la méthode par histogramme qui affecte un poids par une simple fonction escalier. Grâce à cette propriété, l'estimation par B-splines semble similaire à l'estimation par noyau tout en étant plus simple et donc demandant moins de puissance de calcul. Cela est démontré dans la section 4.6. L'estimation de densités par B-splines est donc un bon compromis entre la méthode classique avec histogrammes, rapide à calculer, et l'estimation par noyau, trop complexe.

4.4.6 Exemple de paramétrage des B-splines pour une attaque sur l'algorithme DES

Soit F une fonction de sélection dépendant d'une hypothèse de clé et d'une valeur intermédiaire dépendant du message d'entrée. Soit L la variable aléatoire représentant la fuite d'information. On considère le vecteur de fuites d'information V_L et sa partition en B ensembles. Par exemple, nous supposons que l'attaquant vise les trois bits les plus significatifs de $\mathrm{SBox}(x \oplus k)$ dans un DES. Il semble naturel d'utiliser $B = 8$ partitions dans une méthode d'estimation par histogrammes afin de ranger les valeurs de sortie qui appartiennent à $[0, 2^3 - 1]$. Dans le cas de l'estimation par B-splines, nous voulons aussi couvrir l'ensemble des valeurs visées possibles. On se rappelle que les fonctions B-splines sont définies sur $[0, n - k + 2]$ et sont non-nulles aux nœuds extrêmes car on utilise un vecteur de nœuds uniforme non-périodique. Le paramètre k est généralement fixé à $k = 2$ ou $k = 3$ pour que le calcul de fonctions B-splines ne soit pas trop complexe tout en garantissant des courbes assez lisses [DSSK04]. Le nombre de points d'arrêt $narret = n-k+2$ correspond au paramètre B de l'estimation par histogrammes. Dans notre exemple, avec $k = 3$ et $narret = B = 8$, on obtient $n = narret + k - 2 = 9$ fonctions de bases. Il suffit ainsi de ne modifier que les paramètres k et $narret$, le nombre de fonctions de bases n est déduit.

4.4.7 Algorithme calculant l'IM en utilisant l'estimation par B-splines

L'algorithme permettant d'estimer l'IM à partir de l'estimation B-spline entre deux variables X et Y est de la forme suivante [DSSK04] :

- **Entrées :** variables aléatoires $X = \{x_1, \ldots, x_N\}$ et $Y = \{y_1, \ldots, y_N\}$, ordre de la spline : k, n_X le nombre de fonctions B-spline de base pour X et n_Y pour Y.
- **Sortie :** $I(X;Y)$.

1. Estimer l'entropie de X.
   (a) Déterminer les n_X Coefficients B-spline (CB) pour chaque $x_u, u = 1, \ldots, N$ tel que $B_{i,d}(x), i = 1, \ldots, n_X$. Sauvegarder la matrice $MatrixCB_X$ de taille $(n_X \times N)$ contenant tous les Coefficients B-spline.

$$MatrixCB_X[i][u] = B_{i,d}(x_u).$$

   (b) Calculer les n_X probabilités $p(a_i)$, $i = 1, \ldots, n_X$:

$$p(a_i) = \frac{1}{N} \sum_{u=1}^{N} B_{i,d}(x_u).$$

   (c) Calculer l'entropie (4.1) :

$$H(X) = -\sum_{i=1}^{n_X} p(a_i) \log(p(a_i)).$$

2. Répéter l'étape 1 pour la variable Y afin d'obtenir la matrice $MatrixCB_Y$ ainsi que l'entropie $H(Y)$.
3. Déterminer les probabilités jointes $p(a_i, b_j)$ pour toutes les $(n_X \times n_Y)$ partitions :

$$\begin{aligned} p(a_i, b_j) &= \tfrac{1}{N} \textstyle\sum_{u=1}^{N} (B_{i,k}(x_u).B_{j,k}(y_u)) \\ &= \tfrac{1}{N} \textstyle\sum_{u=1}^{N} (MatrixCB_X[i][u].MatrixCB_Y[j][u]). \end{aligned}$$

4. Calculer l'entropie jointe $H(X,Y)$ (4.2) :

$$H(X,Y) = -\sum_{i=1,j=1}^{n_X,n_Y} p(a_i, b_j) \log(p(a_i, b_j)).$$

5. Calculer l'information mutuelle (4.6) :

$$I(X;Y) = H(X) + H(Y) - H(X,Y).$$

4.4.8 Utilisation du test de Cramér-von-Mises avec les B-splines

Le test CVM est similaire au test K-S. Ce dernier est largement utilisé dans le domaine des statistiques non-paramétriques. Le test K-S à deux échantillons évalue la différence maximale entre deux fonctions de répartition. Ce test K-S à deux échantillons peut d'ailleurs être rapproché du test non-paramétrique Mann-Whitney qui est l'équivalent non-paramétrique du T-test utilisé dans la DPA. Ces deux tests ont été introduits à la section 2.2.3.6. Dans [VCS09], les auteurs proposent une attaque basée sur le test de CVM. Ils montrent son efficacité face à différentes attaques différentielles.

En pratique, les fonctions de répartition se calculent à partir d'une structure d'histogramme. La méthode des B-splines présentée précédemment peut s'appliquer de manière similaire dans ce contexte d'estimation de fonctions de répartition. Les différentes valeurs des échantillons sont affectées à plus d'une partition grâce aux fonctions B-splines. Une fois cet histogramme lissé créé, les fonctions de répartition et le test de CVM peuvent être calculés de manière classique. L'amélioration apportée par les B-splines est néanmoins moins importante que dans le cas d'estimation de densités de probabilités. Même faible, cette amélioration peut toutefois être intéressante dans certains cas.

4.5 Calcul efficace de l'information mutuelle généralisée

Nous étudions dans cette section l'évaluation de l'entropie de Rényi par la méthode de Parzen.

On peut obtenir une estimation simple et précise de l'entropie de Rényi en introduisant la méthode de Parzen vue précédemment (4.9).

L'entropie de Rényi d'ordre α d'une variable aléatoire X suivant la loi de probabilité p_X est définie par :

$$H_\alpha(X) = \frac{1}{1-\alpha} \log E_X \left[p_X^{\alpha-1}(X) \right],$$

avec E_X la fonction de calcul d'espérance. Avec les mêmes notations, l'entropie de Shannon de X, notée H_S, est :

$$H_S(X) = -E_X \left[\log p_X(X) \right].$$

Lors de l'estimation de la mesure d'entropie à partir d'un nombre fini d'échantillons, on substitue l'espérance E_X par la moyenne et la probabilité p_X par la méthode de Parzen. L'entropie de Shannon devient alors :

$$\widehat{H}_S(Y) = -\frac{1}{N} \sum_{j=1}^{N} \log \left[\frac{1}{N} \sum_{i=1}^{N} K_h(y_j - y_i) \right],$$

avec h la fenêtre du noyau.

De la même manière, l'entropie de Rényi est :

$$\widehat{H}_\alpha(X) = \frac{1}{1-\alpha} \log \left[\frac{1}{N^\alpha} \sum_{j=1}^{N} \left(\sum_{i=1}^{N} K_h(x_j - x_i) \right)^{\alpha-1} \right].$$

D'après la section 4.1, $\widehat{H}_2(X)$ s'utilise comme l'entropie de Shannon :

$$\widehat{H}_2(X) = -\log \left[\frac{1}{N^2} \sum_{j=1}^{N} \left(\sum_{i=1}^{N} K_h(x_j - x_i) \right) \right].$$

L'entropie de Rényi est très intéressante du point de vue de l'estimation. En effet, on somme des puissances d'un noyau, ce qui est plus simple que l'utilisation de la méthode de Parzen avec Shannon. En combinant l'entropie de Rényi et la méthode de Parzen, on gagne en temps de calcul. Néanmoins cette technique requiert toujours beaucoup de calculs comparée à l'estimation par histogrammes.

Algorithme calculant la GMI. Nous adaptons à notre étude un des algorithmes proposé dans [PH95] permettant de calculer la GMI. Comme expliqué dans la section 4.1, l'algorithme requiert que l'une des deux variables aléatoires suive la distribution uniforme ce qui est très rarement le cas en pratique. Pour l'efficacité de l'algorithme, on applique la transformation en distribution uniforme aux deux variables. La première étape consiste donc à transformer les données de telle manière qu'elles suivent cette distribution uniforme. Soit X et Y deux variables aléatoires discrètes ayant le même nombre d'états N.

1. On utilise le rang des données triées pour obtenir une distribution uniforme à partir des états de X. On note X_i avec $i = 1, \ldots, N$ les différents états possibles de X. Soit les deux tableaux de taille N :
$$\begin{cases} \mathrm{TData}_X[i] = X_i & \text{pour } i \in \{1, \ldots, N\} \\ \mathrm{TIndex}_X[i] = i & \text{pour } i \in \{1, \ldots, N\}. \end{cases}$$
2. On modifie un algorithme de tri classique pour qu'il réarrange le tableau d'index TIndex_X en fonction des valeurs du tableau TData_X.
3. Soit TRang_X un tableau de taille N tel que :
$$\mathrm{TRang}_X[\mathrm{TIndex}_X[i]] = i \quad \text{pour } i \in \{1, \ldots, N\}.$$
On obtient alors une transformation en distribution uniforme.
4. Les trois premières étapes sont répétées pour la variable Y afin d'obtenir TRang_Y et TIndex_Y.
5. Un certain niveau de granularité ϵ est fixé [PH95]. Dans notre cas, ϵ équivaut en quelque sorte au nombre de partitionnement d'une estimation par histogrammes.
6. Le principe de l'algorithme est de trouver les plus proches voisins d'un point tel que la distance n'excède pas ϵ.

Nous étudions dans le chapitre suivant l'efficacité des différents tests statistiques présentés précédemment dans le cadre d'attaques par canaux cachés.

Algorithme 1: Calcul de l'information mutuelle généralisée

Données : Nombre d'états N, tableau de rangs TRang_Y, tableau d'index TIndex_X, paramètre ϵ

Résultat : $I_2(X;Y)$

```
N_total = (N − 1)(N − 2)/2
C_{1,ε} = (2N − ε)(ε − 1)/(N(N − 1))
Somme = 0
i = 1
tant que i ≤ n − 1 faire
    idx_1 = TIndex_X[i]
    j = i + 1
    tant que j < (i + ε) et j < N faire
        idx_2 = TIndex_X[j]
        Somme = Somme + 1
        si |TRang_Y[idx_1] − TRang_Y[idx_2]| < ε alors
            Somme = Somme + 1
        j = j + 1
    i = i + 1
Somme = Somme/N_total
retourner log(Somme/C_{1,ε})
```

4.6 Comparaison expérimentale d'attaques par analyse différentielle

Comme précisé précédemment, nous nous contentons de comparer l'efficacité d'estimateurs de densités de probabilités non-paramétriques dans un contexte d'attaques par canaux cachés. Nous avons mené des attaques sur les trois algorithmes : DES, AES et un algorithme de multiplication multi-précision. Nous comparons les attaques suivantes classées par catégories :

- tests paramétriques classiques, CPA (section 2.2.3.2), multi-bits DPA (section 2.2.3.1), Void Hypothesis DPA (VDPA) [ARR03],
- tests non-paramétriques, Spearman noté SPE (section 2.2.3.4), CVMB (section 4.4.8), CVM (section 2.2.3.6),
- partitionnement de données, DCA utilisant la variance (section 2.2.3.3),
- l'information mutuelle avec estimation paramétrique, *Cumulant-based Estimator* (CE) [LB10] qui est, pour l'instant, le meilleur estimateur paramétrique de la littérature,
- l'information mutuelle avec estimation non-paramétrique, estimation d'information mutuelle généralisée appelée *Generalized Mutual Information Analysis* (GMIA) (section 4.1.2), estimation avec histogrammes appelée *Histogram Estimation* (HE) (section 4.3.2), estimation par noyaux appelée *Kernel Density Estimation* (KDE) (section 4.3.3), estimation par KNN (section 4.3.4), estimation grâce aux B-splines appelée *B-Spline Estimation* (BSE) (section 4.4).

Tout d'abord, nous introduisons quelques métriques importantes permettant de mesurer l'efficacité d'attaques par canaux cachés.

4.6.1 Évaluation de l'efficacité d'attaques à canaux cachés

Nous utilisons deux métriques bien connues de la littérature qui permettent d'identifier différentes propriétés d'une attaque. Nous rappelons les définitions présentées dans la section 3.4 :

- la *guessed entropy* [SGV08], qui est la position, en moyenne, de l'hypothèse correcte dans le vecteur d'hypothèses triées à la sortie d'une attaque,
- le *first-order success rate* [SGV08] qui est, pour un nombre de traces données, la probabilité que l'hypothèse correcte soit en première position dans le vecteur d'hypothèses triées.

Nous analysons brièvement la complexité de calcul nécessaire pour chacune des attaques dans la Figure 4.5. Les mesures de temps ont été enregistrées sur un ordinateur de bureau classique équipé d'un processeur Pentium 4. Les ordres de grandeurs entres attaques sont particulièrement intéressants. On remarque sur la Figure 4.5a que les attaques KDE et KNN requièrent, comme on pouvait le deviner, beaucoup de puissance de calcul. La Figure 4.5b représente la même courbe en omettant ces deux attaques. Les attaques GMIA et BSE sont les attaques suivantes demandant le plus de temps de calcul. La Table 4.1 permet d'avoir une vision plus claire de la complexité des attaques non-paramétriques et notamment de

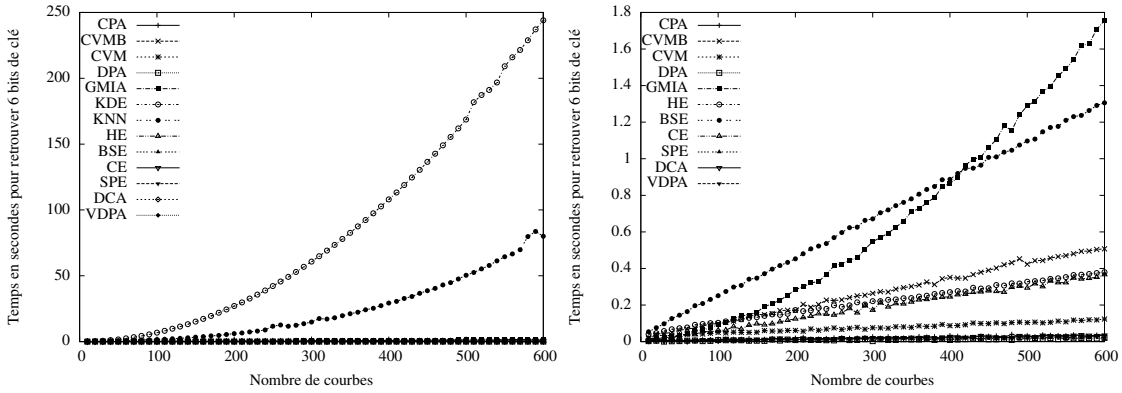


FIGURE 4.5 – Comparaison du temps de calcul moyen nécessaire pour attaquer 6 bits de clés dans le cas du DES. La figure de droite est un agrandissement de celle de gauche.

Attaques	KDE	KNN	GMIA	BSE	HE
Temps (en sec)	244	80	1.75	1.3	0.37
Pourcentage	100	32.79	0.72	0.53	0.15

TABLE 4.1 – Récapitulatif du temps d'attaque moyen en considérant 600 courbes de consommation pour les attaques non-paramétriques.

l'ordre de grandeur entre celles-ci.

4.6.2 Comparaison sur les traces du DPA Contest 2008/2009 d'un DES

Notre premier jeu d'attaque (Figure 4.6) est réalisé en utilisant les traces du DPA Contest 2008/2009 [VLSa]. On peut classer les attaques en trois groupes selon leur performance. En tête on retrouve les attaques CE, CPA, SPE et DPA qui utilisent toutes des tests paramétriques. Un deuxième groupe est constitué des attaques KDE, DCA, VDPA, CVM et CVMB parmi lesquelles on a KDE, la première méthode d'estimation non-paramétrique d'information mutuelle en terme de performance. On trouve ensuite l'estimation non-paramétrique à base de B-splines BSE suivie de KNN, GMIA et HE. On constate donc bien que les performances de la MIA avec l'estimateur naïf HE sont très mauvaises et que d'autres méthodes plus précises d'estimation non-paramétriques permettent d'atteindre des résultats beaucoup plus satisfaisants. D'après la Figure 4.5, on peut considérer l'écart de performances entre KDE et BSE négligeable comparé à leur complexité de calcul respectives, BSE ayant des résultats corrects pour un temps de calcul bien moins élevé.

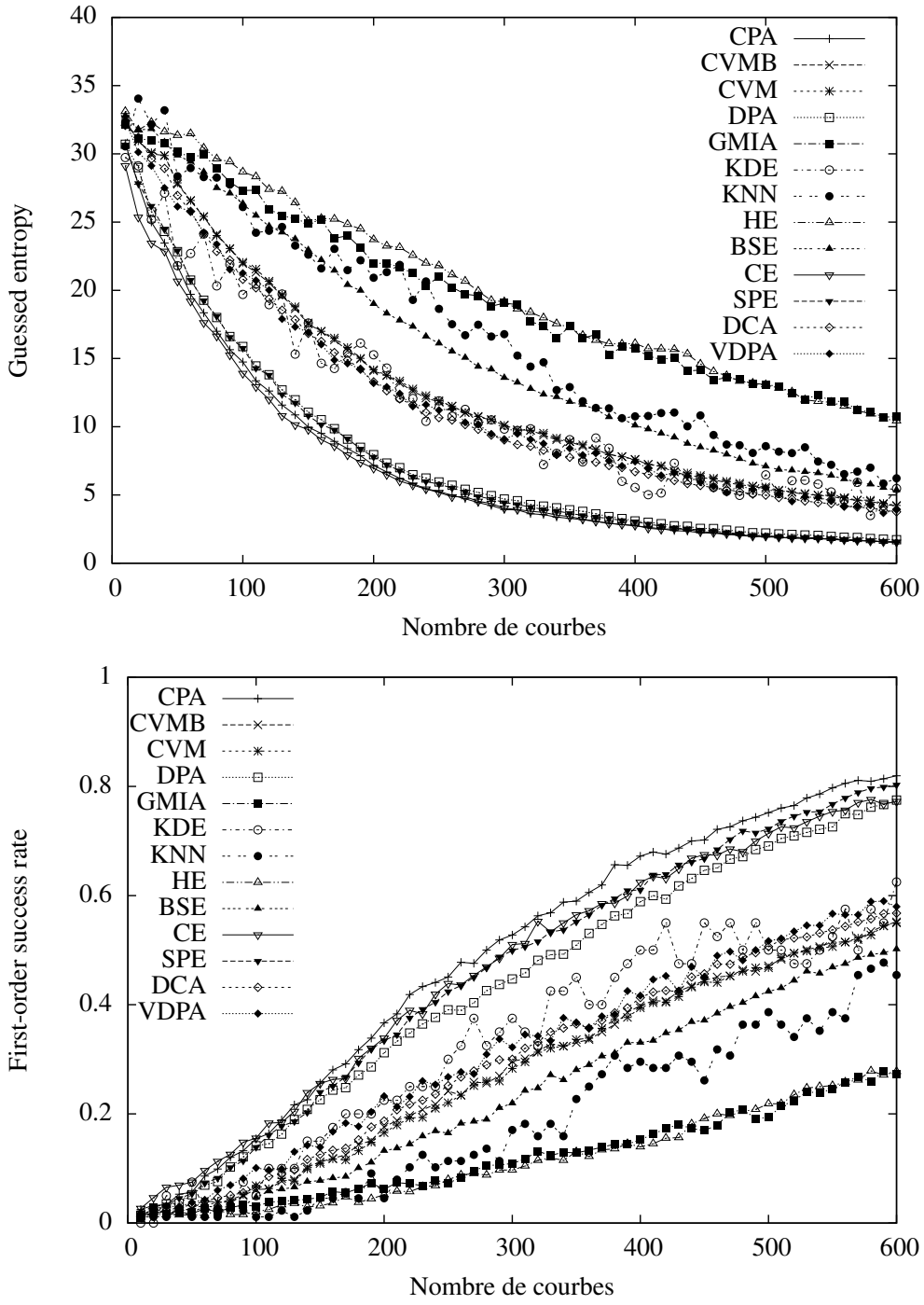


FIGURE 4.6 – Comparatif de résultats d'attaques sur des traces du DPA Contest 2008/2009 implémentant un DES. L'axe des abscisses correspond au nombre de courbes de consommation utilisées. L'axe des ordonnées correspond au rang pour les attaques *guessed entropy* ou à une probabilité pour les attaques *first-order success rate*.

4.6.3 Comparaison sur les traces d'une multiplication sur un Atmel STK600

Nous testons ensuite l'efficacité des attaques sur une plateforme différente et en utilisant un autre algorithme. Nous implémentons sur une carte Atmel STK600 [ATMb] utilisant un processeur 8-bit AVR ATmega2561 [ATMa] un algorithme de multiplication multi-précision. L'algorithme utilise la méthode bien connue de Comba [Com90] (section 13.3.3.2). Le but de l'attaquant est de retrouver les octets d'un multiplicande secret fixé alors que de nombreux multiplieurs aléatoires, connus de l'attaquant, sont donnés à l'algorithme. Le contexte d'acquisition de ces traces est bien différent de celui du DPA Contest. En effet, la carte Atmel STK600 n'est pas adaptée pour des mesures de courant. Ainsi les traces de consommation contiennent beaucoup plus de bruit que celles du DPA Contest. Les résultats dans ce contexte sont particulièrement intéressants (Figure 4.7). On peut encore séparer les attaques en trois groupes par leur performance. On retrouve toujours en première position les tests paramétriques CE, CPA, SPE et DPA. On remarque ensuite que BSE obtient des résultats légèrement meilleurs plaçant l'attaque au même niveau que CVM, CVMB, VDPA, DCA et KDE.

4.6.4 Comparaison sur les traces du DPA Contest 2009/2010 d'un AES

Un dernier jeu d'attaques (Figure 4.8) est effectué sur les courbes du DPA Contest 2009/2010 implémentant un AES [VLSb]. Ces résultats ont seulement pour but de confirmer certaines des observations précédentes. On élimine les attaques KDE, KNN, GMIA et CVMB qui n'apportent pas beaucoup en considérant leur rapport entre temps de calcul et performances. On note encore une fois la bonne efficacité de l'estimateur BSE qui offre un gain considérable par rapport à l'estimateur HE classique. L'attaque MIA avec estimation non-paramétrique devient alors compétitive grâce à cette méthode.

4.7 Remarques finales

Avec cette comparaison de la plupart des distingueurs utilisés dans des attaques par canaux cachés, on note des différences entre les performances des tests statistiques classiques par rapport à leur efficacité lorsqu'ils sont utilisés dans le contexte des canaux cachés. Ainsi l'estimateur par k-plus-proches voisins est censé avoir moins d'erreurs statistiques que les estimateurs KDE ou BSE. Néanmoins, dans notre cas, ses performances sont décevantes. Les tests statistiques paramétriques classiques sont toujours parmi les plus intéressants. La méthode d'estimation paramétrique à base de cumulant proposée par Lee et Berthier [LB10] donne notamment de très bons résultats. Ces expériences montrent aussi le gain obtenu lorsqu'on utilise des estimateurs non-paramétrique d'information mutuelle efficaces. Même si l'attaque MIA n'est toujours pas la plus puissante, on améliore grandement ses performances comparé à l'utilisation d'histogrammes pour l'estimation qui est utilisé dans

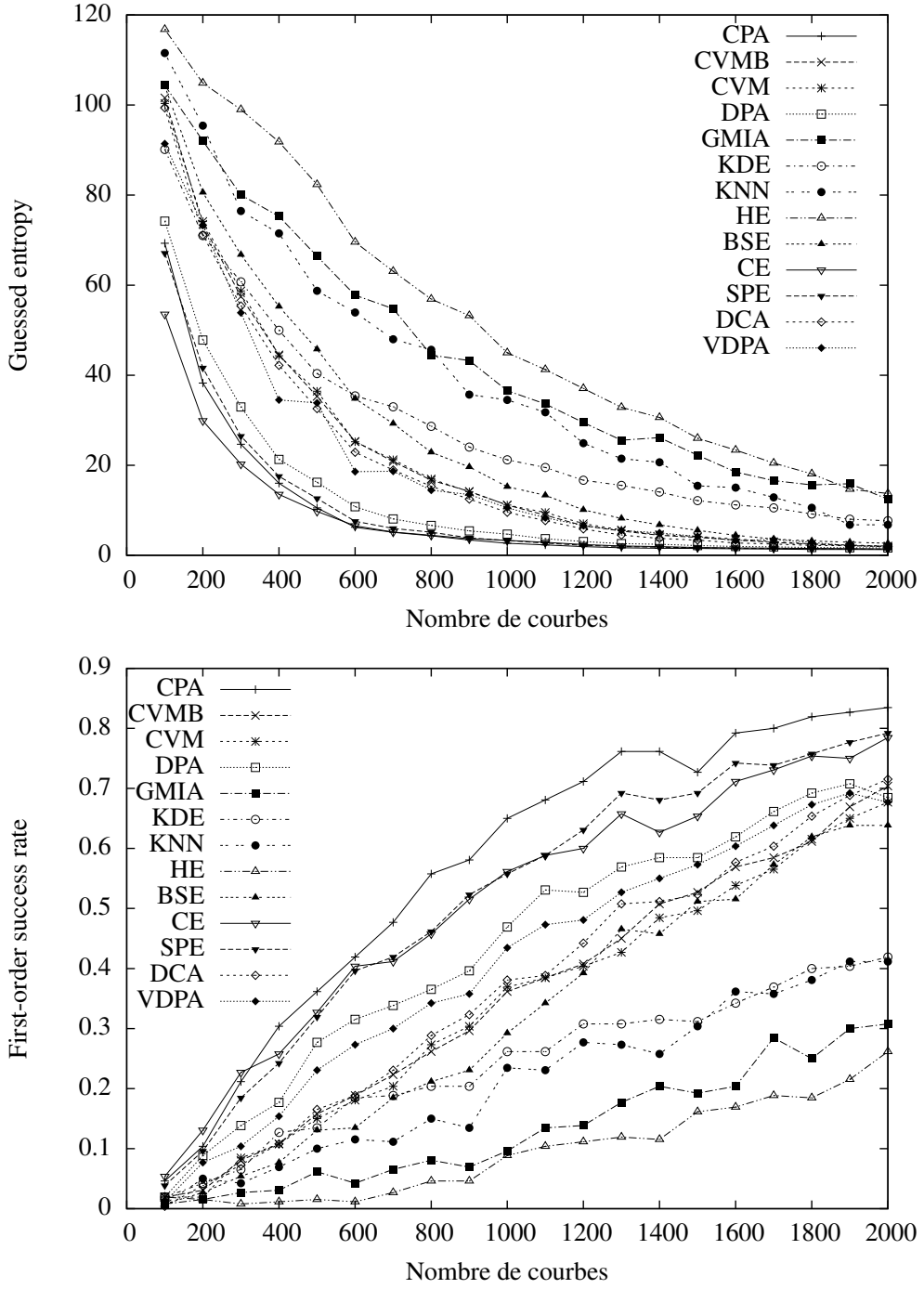


FIGURE 4.7 – Comparatif de résultats d'attaques sur des traces enregistrées sur une carte Atmel STK600 utilisant un AVR ATmega2561 implémentant une multiplication multi-précision. L'axe des abscisses correspond au nombre de courbes de consommation utilisées. L'axe des ordonnées correspond au rang pour la métrique *guessed entropy* ou à une probabilité pour la métrique *first-order success rate*.

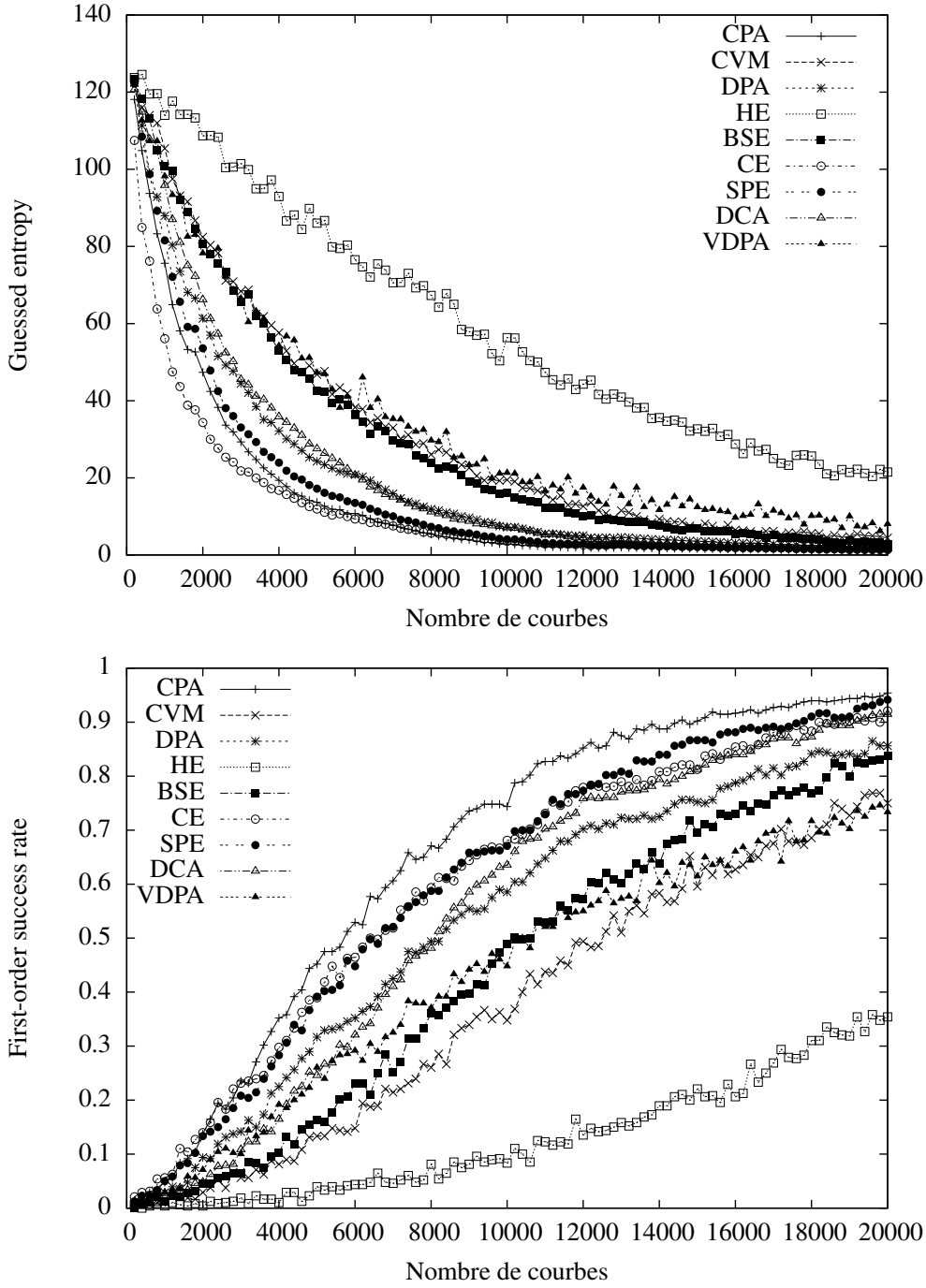


FIGURE 4.8 – Comparatif de résultats d'attaques sur des traces du DPA Contest 2009/2010 implémentant un AES. L'axe des abscisses correspond au nombre de courbes de consommation utilisées. L'axe des ordonnées correspond au rang pour la métrique *guessed entropy* ou à une probabilité pour la métrique *first-order success rate*.

de nombreux articles comme référence. On rappelle que toutes ces expériences sont effectuées sur des composants CMOS. Les attaques paramétriques qui utilisent le modèle du poids de Hamming sont donc naturellement avantagées. Améliorer l'estimation non-paramétrique de la MIA a un intérêt pratique plus évident si on considère des scénarios d'attaques qui lui sont plus adaptés, comme des attaques sur des composants utilisant un autre technologie.

Deuxième partie

Algorithmes de multiplication scalaire résistants aux attaques par canaux cachés

Chapitre 5

Préliminaires mathématiques

5.1 Équation de Weierstrass

Soit K un corps parfait et $\overline{K}$ sa clôture algébrique, on note K^* pour $K \setminus \{0\}$. Soit F un polynôme homogène non constant de $\overline{K}[X, Y, Z]$ défini tel que :

$$F(X, Y, Z) = Y^2Z + a_1XYZ + a_3YZ^2 - X^3 - a_2X^2Z - a_4XZ^2 - a_6Z^3,$$

avec $a_1, a_2, a_3, a_4, a_6 \in \overline{K}$. Une courbe de Weierstrass E est définie comme l'ensemble des solutions $F(X, Y, Z) = 0$ dans le plan projectif $\mathbb{P}^2(\overline{K})$. Nous considérons par la suite les courbes de Weierstrass non-singulières, dites courbes elliptiques de Weierstrass, c'est-à-dire tel que pour tout point $P \in E$, les trois dérivées partielles $\partial F/\partial X$, $\partial F/\partial Y$ et $\partial F/\partial Z$ sont non-nulles simultanément en P.

Une courbe elliptique de Weierstrass peut être vue comme la réunion d'un point à l'infini $\mathcal{O}$ et d'une courbe plane affine d'équation :

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \tag{5.1}$$

La courbe E est dite définie sur le corps K si $a_1, a_2, a_3, a_4, a_6 \in K$. On la note E/K. L'équation (5.1) est souvent appelée équation généralisée de Weierstrass.

Le critère sur la valeur des dérivées partielles implique que la courbe elliptique est non-singulière. On teste facilement ce critère grâce aux valeurs définies ci-après. Soit les quantités suivantes associées à la courbe E :

$$\begin{aligned}
d_2 &= a_1^2 + 4a_2, \\
d_4 &= 2a_4 + a_1a_3, \\
d_6 &= a_3^2 + 4a_6, \\
d_8 &= a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2, \\
c_4 &= d_2^2 - 24d_4, \\
\Delta &= -d_2^2d_8 - 8d_4^3 - 27d_6^2 + 9d_2d_4d_6, \\
j(E) &= c_4^3/\Delta.
\end{aligned}$$

La quantité $\Delta(E)$, est appelée discriminant de la courbe E. Elle permet de caractériser les courbes E elliptiques par le théorème suivant.

Théorème 5.1.1 ([Sil86]). *La courbe elliptique E est non singulière si et seulement si $\Delta \neq 0$.*

Lorsque E est elliptique, la quantité $j(E)$ est bien définie. Elle est appelée le j-invariant de la courbe à cause du résultat classique suivant.

Théorème 5.1.2 ([Sil86]). *Si deux courbes elliptiques E_1/K et E_2/K sont isomorphes sur K, alors $j(E_1) = j(E_2)$. Réciproquement, si $j(E_1) = j(E_2)$ les courbes elliptiques sont isomorphes sur $\overline{K}$.*

Les points d'une courbe elliptique possèdent une propriété particulièrement intéressante lorsqu'on veut faire de la cryptographie. On considère l'ensemble des points qui satisfont à l'équation de Weierstrass (5.1) définissant la courbe E/K. On s'intéresse plus précisément aux solutions de l'équation à valeurs dans K plus un point appelé point à l'infini et noté $\mathcal{O}$. Ce point peut être vu comme étant l'intersection de toutes droites parallèles à l'axe des ordonnées y. Ce point n'existe pas dans l'espace affine mais est nécessaire pour la création d'une loi de groupe. Des clarifications sur ce point $\mathcal{O}$ sont données dans la section 5.8.

Définition 5.1.1. *L'ensemble des points $(x, y) \in K \times K$ satisfaisant à l'équation* (5.1) *de la courbe E/K accompagné du point à l'infini $\mathcal{O}$ est noté $E(K)$.*

Cet ensemble de points $E(K)$ possède une loi de groupe comme détaillé dans la section 5.3.

5.2 Équation simplifiée de Weierstrass

Il existe différentes manière d'écrire une équation de courbe elliptiques suivant si elle est définie sur un corps fini de grande caractéristique $\mathbb{F}_q$, ou sur un corps de caractéristique 2, aussi appelé corps binaire, $\mathbb{F}_{2^m}$. Nous n'étudions pas dans cette thèse les corps de caractéristique 3. Dans la pratique, on peut utiliser des corps finis de ces deux grandes familles ayant des propriétés et des contraintes qui leur sont spécifiques. La Figure 5.1 représente quelques exemples de types de corps finis utilisés.

L'équation de Weierstrass d'une courbe peut s'écrire différemment, et plus simplement, grâce à un changement de variables. L'équation obtenue, dite équation simplifiée, est très utile en pratique.

Définition 5.2.1 ([Eng99]). *Soit E_1 et E_2 deux courbes elliptiques définie sur K. Alors E_1 et E_2 sont isomorphes sur K s'il existe $u, r, s, t \in K$, $u \neq 0$ avec le changement de variables suivant :*

$$(x, y) \mapsto (u^2x + r, u^3y + u^2sx + t),$$

transformant l'équation de E_1 en celle de E_2.

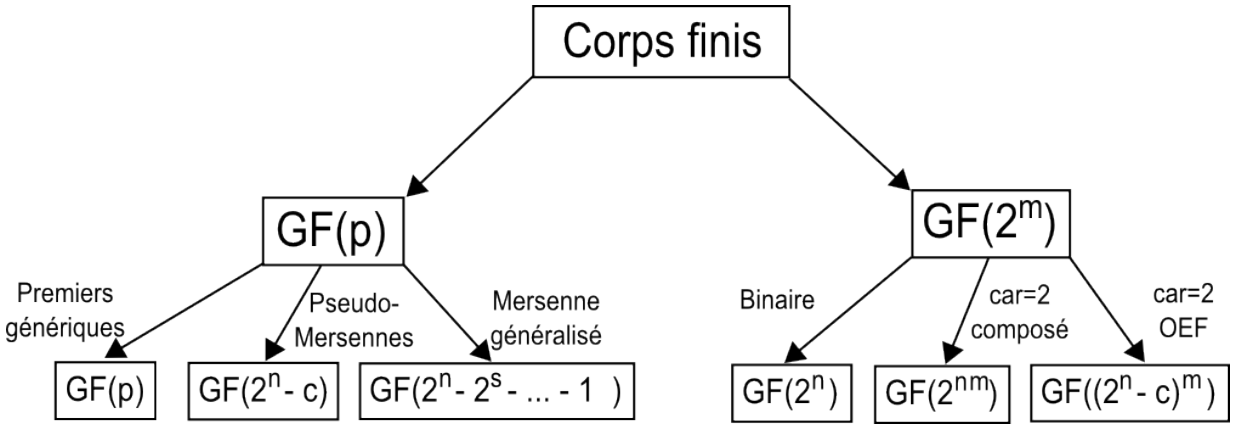


FIGURE 5.1 – Classification de quelques types de corps finis utilisés dans le cadre des courbes elliptiques.

En choisissant de manière appropriée les valeurs $u, r, s, t \in K$, on peut trouver une forme de l'équation de Weierstrass avec moins de paramètres [Sil86]. Pour les corps K de caractéristique strictement supérieure à 3, le changement de variables

$$(x, y) \mapsto \left(x - \frac{a_1^2 + 4a_2}{12}, y - \frac{a_1}{2}\left(x - \frac{a_1^2 + 4a_2}{12} \right) - \frac{a_3}{2} \right),$$

transforme l'équation (5.1) en équation simplifiée de Weierstrass :

$$E : y^2 = x^3 + ax + b, \tag{5.2}$$

avec $a, b \in K$ et avec $\Delta = -16(4a^3 + 27b^2) \neq 0$.

Pour les corps de caractéristique 2, on dispose de changements de variables différents suivant que l'on souhaite définir une courbe supersingulière ou non. Les courbes supersingulières ne sont pas utilisées dans la grande majorité des cryptosystèmes à base de courbes elliptiques car elles sont vulnérables à l'attaque MOV [MOV93].

Soit K un corps de caractéristique 2 et le paramètre $a_1 = 0$ de (5.1), la courbe elliptique est alors supersingulière et le changement de variables :

$$(x, y) \mapsto \left(a_1^2 x + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3} \right),$$

donne l'équation simplifiée :

$$E : y^2 + cy = x^3 + ax + b, \tag{5.3}$$

avec $a, b, c \in K$ et avec $\Delta = c^4 \neq 0$.

Soit K un corps de caractéristique 2 et $a_1 \neq 0$, la courbe elliptique est alors ordinaire et le changement de variables :

$$(x, y) \mapsto \left(x a_1^2 + \frac{a_3}{a_1}, a_1^3 y + \frac{a_1^2 a_4 + a_3^2}{a_1^3} \right),$$

donne l'équation simplifiée :

$$E : y^2 + xy = x^3 + ax^2 + b, \tag{5.4}$$

avec $a, b \in K$ et avec $\Delta = b \neq 0$.

5.3 Loi de groupe

Soit E une courbe elliptique définie sur un corps fini K et notée E/K. Il existe une règle d'addition appelée « sécante-tangente » qui, pour deux points de E/K, renvoie un troisième point de E/K. L'ensemble des points $E(K)$ auquel on associe le point à l'infini $\mathcal{O}$ muni de cette règle forment un groupe abélien. La règle d'addition est alors simplement notée $+$. Les cryptosystèmes à base de courbes elliptiques sont tous construits grâce à cette structure de groupe.

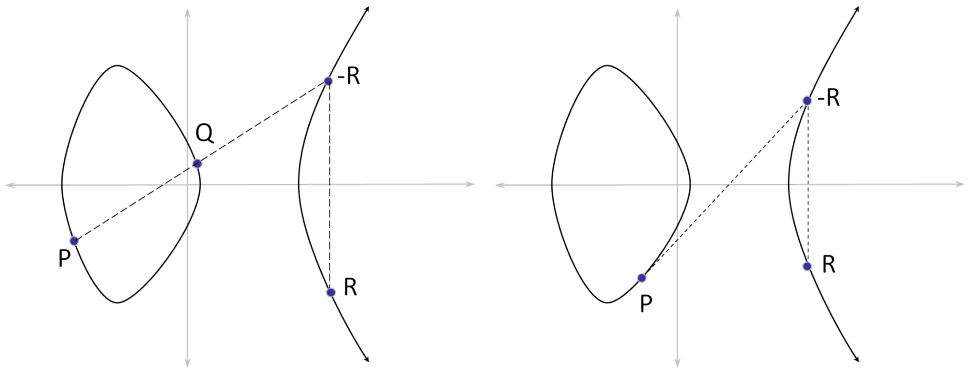
Théorème 5.3.1 (Bézout simplifié)**.** *Soit C_1 et C_2 deux courbes définies sur un corps algébriquement clos d'équations respectives $F(x, y) = 0$ et $G(x, y) = 0$. Alors, le nombre de points, en comptant les multiplicités, qui se trouvent dans l'ensemble $C_1 \cap C_2$ est $\#(C_1 \cap C_2) = \deg(C_1)\deg(C_2)$. En particulier, si $C_1 = E$ est une courbe elliptique et $C_2 = l$ est une droite, alors $\#(C_1 \cap C_2) = 3$.*

À partir du théorème de Bezout, on peut en déduire que le nombre de points d'intersections entre une courbe elliptique et une droite est au plus de 3. Grâce à cette relation, on peut définir une opération de groupe. On décrit d'abord cette opération de manière géométrique car l'étude est plus intuitive. On trouve tout d'abord l'opposé d'un point $R \in E(K)$ tel que $R \neq \mathcal{O}$ de la manière suivante :

1. On trace la droite passant par R et $\mathcal{O}$. Dans le plan affine, une droite passant par $\mathcal{O}$ est une droite parallèle à l'axe des ordonnées donc de la forme $x = a$ pour une constante $a \in K$.
2. Soit l'intersection de cette droite et de E/K correspond seulement aux points R et $\mathcal{O}$, alors $R + R = \mathcal{O}$, d'où $-R = R$.
3. Soit l'intersection de cette droite et de E/K contient un autre point S, alors $R + S + \mathcal{O} = \mathcal{O}$, d'où $-R = S$.

Soit $P = (x_1, y_1)$ et $Q = (x_2, y_2)$ deux points de E/K tel que $P \neq Q$ et $x_1 \neq x_2$. On définit l'addition $P + Q$ de la manière suivante :

1. On trace la droite unique passant par P et Q et on considère son intersection avec E/K.
2. Soit l'intersection ne contient que les points P et Q, alors la droite est une tangente à la courbe elliptique au point P ou Q. Cela signifie que soit $P + P + Q = \mathcal{O}$ et donc $P + Q = -P$, soit $Q + Q + P = \mathcal{O}$ et donc $P + Q = -Q$.
3. Soit l'intersection contient un point R, alors $P + Q + R = \mathcal{O}$, d'où $P + Q = -R$.



(a) Addition de points (b) Doublement de point

FIGURE 5.2 – Opérations de groupe sur une courbe elliptique E définie sur un corps réel $\mathbb{R}$.

Dans le cas où $x_1 = x_2$, la droite passant par P et Q est parallèle à l'axe y et donc l'intersection est le point $\mathcal{O}$, d'où $P + Q = \mathcal{O}$ et donc $P = -Q$.

On définit de manière particulière l'opération $P + P$, aussi notée $[2]P$, qui correspond au doublement du point P :

1. On trace la droite unique tangente à E au point P et on considère son intersection avec E/K.
2. Soit l'intersection contient seulement P, alors P est un point d'inflexion donc $[2]P + P = \mathcal{O}$ d'où $[2]P = -P$.
3. Soit l'intersection contient un point R, alors $[2]P + R = \mathcal{O}$ d'où $[2]P = -R$.

Lorsque $K = \mathbb{R}$, les opérations d'additions et doublements de points se comprennent aisément de manière visuelle (Figure 5.2).

On peut transposer cette description géométrique de manière algébrique. Soit K un corps fini et E/K une courbe elliptique définie par l'équation de Weierstrass généralisée $E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ avec $a_1, a_2, a_3, a_4, a_6 \in K$. Soit deux points $P, Q \in E(K)$ tel que $P, Q \neq \mathcal{O}$. Soit $R \in E(K)$ le point qui correspond à la somme de P et Q. On note $P = (x_1, y_1)$, $Q = (x_2, y_2)$ et $R = (x_3, y_3)$.

Si $Q = -P = (x_1, -y_1 - a_1x - a_3)$ alors la droite passant par P et Q est parallèle à l'axe des ordonnées, ainsi leur somme donne le point à l'infini $R = P + Q = \mathcal{O}$.

Sinon, il existe deux cas :

– si $P \neq Q$, la droite L passant par ces deux points a pour pente :

$$\lambda_L = \frac{y_1 - y_2}{x_1 - x_2}.$$

– si $P = Q$, la tangente T passant par P a pour pente :

$$\lambda_T = \frac{3x_1^2 + 2a_2x_1 + a_4 - a_1y_1}{2y_1 + a_1x_1 + a_3}.$$

Soit μ un terme constant, l'équation de L et T est donnée par :

$$L, T : y = \lambda_{L,T} x + \mu.$$

Comme P satisfait à l'équation, on a $\mu = y_1 - \lambda_{L,T} x_1$.

Pour trouver le nouveau point d'intersection de L ou T avec la courbe elliptique, on substitue dans l'équation de E afin de retrouver la coordonnée x :

$$(\lambda_{L,T} x + \mu)^2 + (a_1 x + a_3)(\lambda_{L,T} x + \mu) = x^3 + a_2 x^2 + a_4 x + a_6.$$

En développant et en réarrangeant, on obtient le polynôme $R(X)$:

$$R(X) = X^3 + (a_2 - \lambda_{L,T}^2 - a_1 \lambda_{L,T}) X^2 + (a_4 - 2\lambda_{L,T}\mu - a_3 \lambda_{L,T} - a_1 \mu) X + a_6 - \mu^2 - a_3 \mu.$$

Ce polynôme est de la forme $x^3 - ax^2 + bx + c$. Les trois racines de cette équation cubique correspondent aux trois points d'intersection de L ou T avec la courbe. Or si on a une équation cubique $x^3 + ax^2 + bx + c$ avec pour racines r, s, t, on sait que :

$$x^3 + ax^2 + bx + c = (x - r)(x - s)(x - t) = x^3 - (r + s + t)x^2 + \dots.$$

Donc,

$$r + s + t = -a.$$

Dans notre cas, pour la ligne L, on connaît déjà deux des trois racines, x_1 et x_2, car les points P_1 et P_2 appartiennent à L et à E. On retrouve alors la coordonnée x_3 du troisième point d'intersection par :

$$x_3 = \lambda_{L,T}^2 + a_1 \lambda_{L,T} - a_2 - x_1 - x_2.$$

Enfin, en prenant le symétrique par rapport à l'axe des abscisses, on obtient la coordonnée y_3 du point $P_3 = (x_3, y_3)$:

$$y_3 = \lambda_{L,T}(x_1 - x_3) - y_1 - a_1 x_3 - a_3.$$

Proposition 5.3.1. *Soit K un corps fini et E/K une courbe elliptique définie sur K. On note $+$ la règle d'addition entre points de la courbe définie comme ci-dessus. Alors $(E(K), +)$ forme un groupe abélien.*

La preuve de cette proposition se trouve notamment dans [Was08].

5.4 Formules d'addition dans les cas spécifiques

En pratique, afin d'obtenir les meilleures performances possibles, on utilise la forme simplifiée de l'équation de Weierstrass et on spécifie l'addition de points suivant la caractéristique du corps de base.

Soit $K = \mathbb{F}_q$ le corps fini à q éléments, avec $q = p^m$ une puissance $m \geq 1$ d'un nombre premier p. La clôture algébrique de K correspond alors à $\overline{K} = \bigcup_{i \geq 1} \mathbb{F}_{q^i}$. Si la caractéristique p de K est strictement supérieure à 3, on peut noter l'équation d'une courbe elliptique dans sa forme de Weierstrass simple (5.2).

On donne quelques propriétés de la loi d'addition sur $E/\mathbb{F}_q$.

– Identité : $P + \mathcal{O} = \mathcal{O} + P = P$ pour tout $P \in E/\mathbb{F}_q$.
– Opposé : si $P = (x, y) \in E/\mathbb{F}_q$, alors $P + (-P) = \mathcal{O}$ avec $-P = (x, -y)$ le point opposé de P. Le point $-P$ appartient aussi à $E/\mathbb{F}_q$.
– Addition de points : Soit $P_1 = (x_1, y_1)$ et $P_2 = (x_2, y_2)$ deux points de $E/\mathbb{F}_q$ tel que $P_2 \neq \pm P_1$. Soit $R = P + Q = (x_3, y_3)$ calculé ainsi :

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

– Doublement de point : Soit $P_1 = (x_1, y_1) \in E/\mathbb{F}_q$ tel que $P \neq \pm P$. Soit $R = P + P = (x_3, y_3)$ calculé ainsi :

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

Soit $\mathbb{F}_{2^m}$ un corps fini de caractéristique 2. On note $E/\mathbb{F}_{2^m}$ une courbe elliptique non supersingulière définie sur ce corps. Cette courbe peut s'écrire sous la forme de Weierstrass simple suivante (5.4).

On donne quelques propriétés de la loi d'addition sur $E/\mathbb{F}_{2^m}$.
– Identité : $P + \mathcal{O} = \mathcal{O} + P = P$ pour tout $P \in E/\mathbb{F}_{2^m}$.
– Opposé : si $P = (x, y) \in E/\mathbb{F}_{2^m}$, alors $P + (-P) = \mathcal{O}$ avec $-P = (x, x + y)$ le point opposé de P. Le point $-P$ appartient aussi à $E/\mathbb{F}_{2^m}$.
– Addition de points : Soit $P_1 = (x_1, y_1)$ et $P_2 = (x_2, y_2)$ deux points de $E/\mathbb{F}_q$ tel que $P_2 \neq \pm P_1$. Soit $R = P + Q = (x_3, y_3)$ calculé ainsi :

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

– Doublement de point : Soit $P_1 = (x_1, y_1) \in E/\mathbb{F}_q$ tel que $P \neq \pm P$. Soit $R = P + P = (x_3, y_3)$ calculé ainsi :

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

5.5 Cardinal du groupe de points

On note $\#E(\mathbb{F}_q)$ ou $|E(\mathbb{F}_q)|$ le cardinal du groupe de points $E(\mathbb{F}_q)$ pour la courbe elliptique $E/\mathbb{F}_q$. On appelle trace de Frobenius, ou trace de la courbe, la valeur t telle que $\#E(\mathbb{F}_q) = q + 1 - t$. Le théorème de Hasse permet de connaître des bornes sur $\#E(\mathbb{F}_q)$.

Théorème 5.5.1 (Hasse, voir [Sil86, Théorème 5.1.1]). *Soit $t = q + 1 - \#E(\mathbb{F}_q)$. Alors $|t| \leq 2\sqrt{q}$.*

Il existe différentes méthodes pour retrouver le nombre de points exact pour les courbes elliptiques $E/\mathbb{F}_q$ [BSS05, Chapitre VI]. On peut alors, grâce à ce cardinal, retrouver le nombre de points de la courbe E définie sur un corps $\mathbb{F}_{q^n}$ pour un $n \geq 1$ quelconque.

Théorème 5.5.2 (voir [Was08, Théorème 4.12]). *Soit $t = q+1-\#E(\mathbb{F}_q)$. Soit le polynôme factorisé $X^2 - tX + q = (X-\alpha)(X-\beta)$. Alors, pour tout $n \geq 1$,*

$$\#E(\mathbb{F}_{q^n}) = q^n + 1 - (\alpha^n + \beta^n).$$

L'utilité de ce théorème est plus évidente grâce au Lemme suivant.

Lemme 5.5.1 (voir [Was08] Lemme 4.13). *Soit $t_n = \alpha^n + \beta^n$. Alors $t_0 = 2$, $t_1 = q + 1 - \#E(\mathbb{F}_q)$ et $t_{n+1} = t_1 t_n - q t_{n-1}$ pour tout $n \geq 1$.*

On a donc $\#E(\mathbb{F}_{q^n}) = q^n + 1 - t_n$. Une méthode efficace de calcul du cardinal du groupe de points sur une extension de corps a été proposée par Joye et Quisquater [JQ96].

Exemple 5.5.1. *Soit $E : y^2 = x^3 + 5x + 2$ une courbe elliptique définie sur $\mathbb{F}_{11}$. On peut calculer $\#E(\mathbb{F}_{11}) = 10$ et donc la trace de la courbe est $t = 2$. On cherche à calculer $\#E(\mathbb{F}_{11^2})$:*

$$\#E(\mathbb{F}_{11^2}) = 11^2 + 1 - t_2,$$

avec $t_2 = t_1 t_1 - q t_0 = 2.2 - 11.2 = -18$. On obtient $\#E(\mathbb{F}_{11^2}) = 140$.

5.6 Cryptographie à base de courbes elliptiques

Soit $\mathbb{G}$ un sous-groupe cyclique de $E(K)$ d'ordre premier n généré par le point P :

$$\mathbb{G} = \langle P \rangle = \{\mathcal{O}, P, [2]P, \ldots, [n-1]P\} \subseteq E(K),$$

avec $[n]P = \mathcal{O}$.

On appelle l'opération $[k]P$, pour k un entier et P un point de la courbe, une multiplication scalaire (ou multiplication de point). On omet souvent les crochets pour noter l'opération kP. Elle consiste à calculer un multiple entier d'un élément dans un groupe additif d'une courbe elliptique. Soit $k \in \mathbb{Z}$, et $P, Q \in \mathbb{G}$ alors :

$$Q = [k]P = \underbrace{P + P + \cdots + P}_{k \text{ fois}}.$$

L'opération de multiplication scalaire est dite à « sens unique ». En effet, connaissant le point P et le point $Q = [k]P$, il est difficile de retrouver l'entier k, alors que connaissant k et P il est facile de calculer $Q = [k]P$. Ce problème difficile s'appelle l'ECDLP. La sécurité d'une grande majorité de cryptosystèmes à base de courbes elliptiques repose sur l'ECDLP. Une revue en détail des différentes attaques se trouve dans [Was08, Section 5]. L'opération de multiplication scalaire est donc un point majeur des ECC à la fois du point de vue de la sécurité mais aussi de l'efficacité de ces cryptosystèmes.

5.7 Représentation projective

Depuis le début de l'étude, nous considérons un point de courbe elliptique comme le couple $(x, y) \in K \times K$. On dit que le point est représenté en coordonnées affines. Une alternative est le système de coordonnées projectives. Grâce au système projectif, on évite le calcul d'inverses dans les opérations d'additions et doublements de points (voir section 5.4). Cela est très intéressant lorsqu'on désire implémenter ces opérations car sur de nombreuses plateformes le coût d'une inversion dans un corps fini est prohibitif.

Définition 5.7.1. *Pour un corps K, on définit la relation d'équivalence $\cong$ sur l'espace $K^3\backslash\{(0,0,0)\}$ telle que :*

$$(X_1, Y_1, Z_1) \cong (X_2, Y_2, Z_2) \quad si\ X_1 = \lambda X_2, Y_1 = \lambda Y_2, Z_1 = \lambda Z_2 \quad pour\ un\ \lambda \in K^*.$$

On appelle la classe d'équivalence contenant le point (X, Y, Z) un point projectif noté $(X : Y : Z)$. On confond souvent les notations (X, Y, Z) et $(X : Y : Z)$.

Un polynôme $f \in K[x_1, \ldots, x_n]$ pour $n \in \mathbb{N}$ est appelé homogène si tous ses termes ont le même degré. Une courbe elliptique est définie par une équation qui est l'« homogénéisation » de son équation dans l'espace affine. On l'appelle alors équation de Weierstrass projective. Soit K un corps fini de caractéristique strictement supérieure à 3. Soit E/K une courbe elliptique ordinaire définie sur K d'équation simplifiée (5.2). Son équation de Weierstrass projective est :

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

Soit K un corps fini de caractéristique 2. Soit E/K une courbe elliptique ordinaire définie sur K d'équation simplifiée (5.4). Son équation de Weierstrass projective est :

$$E : Y^2Z + XYZ = X^3 + aX^2Z + bZ^3.$$

On remarque que toute solution $(X, Y, Z) = (X/Z, Y/Z, 1)$ avec $Z \neq 0$ de l'équation projective correspond à la solution $(X/Z, Y/Z)$ de l'équation affine et inversement. Quand $Z = 0$ la résolution de l'équation projective donne le point $(0, 1, 0)$. Afin de respecter la bijection entre les points d'une courbe en représentation projective et en affine, on associe le point $(0, 1, 0)$ au point à l'infini noté $\mathcal{O}$. Le Théorème 5.3.1 de Bézout présenté dans la section précédente n'a d'ailleurs de sens que si l'on se place dans un espace projectif.

Le système de coordonnées projectives peut être généralisé avec la définition suivante.

Définition 5.7.2. *Pour un corps K, on définit la relation d'équivalence $\cong_{c,d}$ sur l'espace $K^3\backslash\{(0,0,0)\}$ telle que :*

$$(X_1, Y_1, Z_1) \cong_{c,d} (X_2, Y_2, Z_2) \quad si\ X_1 = \lambda^c X_2, Y_1 = \lambda^d Y_2, Z_1 = \lambda Z_2 \quad pour\ un\ \lambda \in K^*, c, d \in \mathbb{N}^*.$$

On appelle la classe d'équivalence contenant le point (X, Y, Z) un (c, d)-point projectif noté $(X : Y : Z)$. On confond souvent les notations (X, Y, Z) et $(X : Y : Z)$.

De nombreux systèmes de coordonnées projectives ont été proposés :

- Les coordonnées projectives homogènes : $c = 1, d = 1$, notées $\mathcal{P}$. Le point $\mathcal{O}$ est représenté par $(0, 1, 0)$.
- Les coordonnées projectives jacobiennes : $c = 2, d = 3$, notées $\mathcal{J}$. On les appelle souvent simplement coordonnées jacobiennes. Ce système est très performant pour les courbes définies sur des corps de grande caractéristique. Le point $\mathcal{O}$ est représenté par $(1, 1, 0)$.
- Les coordonnées jacobiennes Chudnovski : $((X, Y, Z), Z^2, Z^3)$, notées $\mathcal{J}^c$. Les trois premières coordonnées sont des coordonnées jacobiennes auxquelles on ajoute deux puissances de Z. Ces informations supplémentaires dans la représentation permettent d'accélérer l'addition de points au détriment d'un espace mémoire plus grand pour stocker les points. Le point $\mathcal{O}$ est représenté par $((1, 1, 0), 0, 0)$.
- Les coordonnées jacobiennes modifiées : $((X, Y, Z), aZ^4)$, notées $\mathcal{J}^m$. Les trois premières coordonnées correspondent à des coordonnées jacobiennes auxquelles on ajoute une information redondante. Le coefficient a provient de l'équation de Weierstrass simplifiée (5.2). Cela permet de calculer un doublement de point plus rapide au détriment d'un espace mémoire supplémentaire requis. Cet avantage par rapport aux coordonnées jacobiennes est perdu lorsque le coefficient a a pour valeur -3. Le point $\mathcal{O}$ est représenté par $((1, 1, 0), 0)$.
- Les coordonnées López-Dahab [LD99b] : $c = 1, d = 2$, notées $\mathcal{L}$. Ce système est particulièrement performant sur les courbes définies sur des corps de caractéristique 2. Le point $\mathcal{O}$ est représenté par $(1, 0, 0)$.

Plus de précisions sur ces différents systèmes sont disponibles dans [CF06]. On étudie dans la section suivante leurs performances suivant la caractéristique du corps de base choisi.

5.8 Efficacité des différents systèmes de coordonnées

Comme nous l'avons vu précédemment, l'ensemble des points d'une courbe elliptique et un point à l'infini forment un groupe abélien. L'efficacité du calcul des opérations de groupe, l'addition et le doublement de points, est primordiale. La complexité de ces calculs varie grandement suivant la représentation choisie des points de la courbe. Dans cette section, nous traitons à la fois le cas de courbes elliptiques définies sur des corps finis de caractéristique strictement supérieure à 3 mais aussi les corps de caractéristique 2.

5.8.1 Dans des corps de grande caractéristique

5.8.1.1 Coordonnées affines

Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_q$ d'équation (affine) $y^2 = x^3 + ax + b$. Soit $P_1 = (x_1, y_1), P_2 = (x_2, y_2) \in E(\mathbb{F}_q)$ deux points de la courbe. Les formules d'addition et de doublement sont données dans la section 5.3. Une addition de points a une complexité de 1 inversion, 2 multiplications et 2 élévations au carré, que l'on note

$1I+2M+2S$. Un doublement de point coûte $1I+2M+S$. On remarque que ces opérations nécessitent le calcul d'une inversion dans $\mathbb{F}_q$ qui est une opération complexe et très coûteuse. Ces coordonnées sont donc très peu utilisées pour une implémentation. On note ce système de coordonnées $\mathcal{A}$.

5.8.1.2 Coordonnées projectives jacobiennes

Il est avantageux de représenter les points de la courbe en utilisant un système de coordonnées projectives. En représentation projective homogène, le point projectif $(X, Y, Z), Z \neq 0$ correspond au point affine $(X/Z, Y/Z)$. Le système de coordonnées projectives jacobiennes est aussi très utilisé car il offre des performances meilleures. Ainsi, le point en coordonnées jacobiennes $(X, Y, Z), Z \neq 0$ correspond au point affine $(X/Z^2, Y/Z^3)$. On s'intéresse plus particulièrement à ce système par la suite. Les formules permettant de calculer un doublement d'un point $P = (X, Y, Z)$ tel que $P_3 = [2]P = (X_3, Y_3, Z_3)$ sont :

$$A = X^2, \quad B = Y^2, \quad C = B^2, \quad D = Z^2, \quad E = 2((X + B)^2 - A - C),$$

$$F = 3A + aD^2, \quad G = F^2 - 2E$$

$$\begin{cases} X_3 &= G, \\ Y_3 &= F(E - G) - 8C, \\ Z_3 &= (Y + Z)^2 - B - D. \end{cases}$$

Le calcul d'un doublement de point nécessite 1 multiplication et 8 élévations au carré dans $\mathbb{F}_q$ que l'on note $1M + 8S$.

Soit $P_1 = (X_1, Y_1, Z_1)$, $P_2 = (X_2, Y_2, Z_2)$ tous deux différents de $\mathcal{O}$ et $P_2 \neq \pm P_1$. Soit $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$. Les formules d'additions en coordonnées projectives jacobiennes sont :

$$A = Z_1^2, \quad B = Z_2^2, \quad C = X_1B, \quad D = X_2A, \quad E = Y_1Z_2B, \quad F = Y_2Z_1A,$$

$$G = D - C, \quad H = (2G)^2, \quad I = GH, \quad J = 2(F - E), \quad K = CH$$

$$\begin{cases} X_3 &= J^2 - I - 2K, \\ Y_3 &= J(K - X_3) - 2EI, \\ Z_3 &= ((Z_1 + Z_2)^2 - A - B)G. \end{cases}$$

Le calcul d'une addition de points coûte $11M + 5S$.

Nous résumons les propriétés de ces systèmes dans la Table 5.1. Dans la Table 5.2, nous récapitulons les complexités des opérations d'additions et doublements dans chacune des coordonnées.

Coordonnées	P projectif	P affine	Équation de la courbe
$\mathcal{P}$	(X, Y, Z)	$(X/Z, Y/Z)$	$Y^2Z = X^3 + aXZ^2 + bZ^3$
$\mathcal{J}$	(X, Y, Z)	$(X/Z^2, Y/Z^3)$	$Y^2 = X^3 + aXZ^4 + bZ^6$
$\mathcal{J}^m$	(X, Y, Z, aZ^4)	$(X/Z^2, Y/Z^3)$	$Y^2 = X^3 + aXZ^4 + bZ^6$
$\mathcal{J}^c$	(X, Y, Z, Z^2, Z^3)	$(X/Z^2, Y/Z^3)$	$Y^2 = X^3 + aXZ^4 + bZ^6$

TABLE 5.1 – Tableau résumant les propriétés des systèmes de coordonnées sur $\mathbb{F}_q$.

Coordonnées	Addition	Doublement
$\mathcal{A}$	$1I + 2M + 1S$	$1I + 2M + 2S$
$\mathcal{P}$	$12M + 2S$	$5M + 6S$
$\mathcal{J}$	$11M + 5S$	$1M + 8S$
$\mathcal{J}^m$	$11M + 7S$	$3M + 5S$
$\mathcal{J}^c$	$12M + 2S$	$6M + 4S$

TABLE 5.2 – Récapitulatif des complexités des opérations d'addition et doublement dans différentes coordonnées sur $\mathbb{F}_q$.

5.8.1.3 Formule d'addition jacobienne simplifiée sur $\mathbb{F}_q$

Une formule d'addition plus spécifique a été proposée par Méloni [Mel07]. On considère deux points en coordonnées jacobiennes $P_1 = (X_1, Y_1, Z)$ et $P_2 = (X_2, Y_2, Z)$ ayant la même coordonnée Z. La formule d'addition de points peut alors se simplifier :

$$A = (X_2 - X_1)^2, \quad B = X_1A, \quad C = X_2A, \quad D = (Y_2 - Y_1)^2, E = Y_1(C - B),$$

$$\begin{cases} X_3 &= D - B - C, \\ Y_3 &= (Y_2 - Y_1)(B - X_3) - E, \\ Z_3 &= Z(X_2 - X_1). \end{cases}$$

L'addition ne coûte alors que $5M + 2S$, encore moins qu'un doublement de point. De plus Méloni remarque, qu'au cours du calcul d'addition, on peut très facilement modifier le point en entrée P_1 afin que P_1 et $P_3 = P_1 + P_2$ aient la même coordonnée Z en sortie. Ils peuvent ainsi être additionnés à nouveau grâce à cette même formule. Le point P_1 devient :

$$\begin{cases} X_1 &= B, \\ Y_1 &= E, \\ Z &= Z_3. \end{cases}$$

Méloni [Mel07] appelle cet algorithme

$$\text{NewAdd}(P_1, P_2) \to (\tilde{P_1}, P_1 + P_2)$$

alors que Goundar et al. [GJM10] le renomme, de manière plus évidente, *co-Z addition with update* (ZADDU). Dans le chapitre 9, nous proposons une utilisation de cette formule dans un algorithme de multiplication de points.

Coordonnées	P projectif	P affine	Équation de la courbe
$\mathcal{P}$	(X, Y, Z)	$(X/Z, Y/Z)$	$Y^2Z + XYZ = X^3 + aX^2Z + bZ^3$
$\mathcal{J}$	(X, Y, Z)	$(X/Z^2, Y/Z^3)$	$Y^2 + XYZ = X^3 + aX^2Z^2 + bZ^6$
$\mathcal{L}$	(X, Y, Z)	$(X/Z, Y/Z^3)$	$Y^2 + XYZ = X^3Z + aX^2Z^2 + bZ^4$

TABLE 5.3 – Tableau résumant les propriétés des systèmes de coordonnées sur $\mathbb{F}_{2^m}$.

5.8.2 Dans des corps de caractéristique 2

5.8.2.1 Coordonnées affines

Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_{2^m}, m \in \mathbb{N}^*$ d'équation (affine) $y^2 + xy = x^3 + ax^2 + b$ avec $a, b \in \mathbb{F}_{2^m}$. Une addition de points a une complexité de $1I + 2M + S$. Un doublement de point coûte $1I + 2M + S$. Comme dans le cas de corps à grande caractéristique, l'addition et le doublement de points requièrent une inversion dans le corps fini. Là encore le coût d'une inversion dans $\mathbb{F}_{2^m}$ peut souvent être tellement prohibitif qu'on considère des coordonnées projectives.

5.8.2.2 Coordonnées López-Dahab projectives

Les coordonnées de López-Dahab [LD99b] proposent qu'à un point $(X, Y, Z), Z \neq 0$ corresponde un point affine $(X/Z, Y/Z^2)$. La complexité de l'addition de points est $13M + 4S$, celle du doublement est $3M+5S$. Ces coordonnées ont de très bonnes performances dans les corps de caractéristique 2. Nous détaillons aussi le système de coordonnées jacobiennes qui nous servira par la suite.

5.8.2.3 Coordonnées projectives jacobiennes

Un point en coordonnées jacobiennes $(X, Y, Z), Z \neq 0$ correspond au point affine $(X/Z^2, Y/Z^3)$. Une addition de points coûte $14M + 5S$ alors qu'un doublement coûte $4M + 5S$. Ces coordonnées sont donc moins performantes que les López-Dahab néanmoins elles sont plus intéressantes lorsqu'on souhaite appliquer l'idée d'addition simplifiée de Méloni dans $\mathbb{F}_{2^m}$ [VD10c].

Nous résumons les propriétés de ces systèmes dans la Table 5.3. Dans la Table 5.4, nous récapitulons les complexités des opérations d'addition et doublement dans chacune des coordonnées.

5.8.2.4 Formule d'addition jacobienne simplifiée sur $\mathbb{F}_{2^m}$

Soient $P_1 = (X_1, Y_1, Z)$ et $P_2 = (X_2, Y_2, Z)$ deux points en coordonnées jacobiennes avec la même coordonnées Z tels qu'ils soient différents de $\mathcal{O}$ et $P_2 \neq \pm P_1$. Soit $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$. La formule d'addition de points simplifiée est :

$$A = X_1 + X_2, \quad B = A^2, \quad C = AB, \quad D = Y_1 + Y_2, \quad E = CY_2, \quad F = BX_2,$$

Coordonnées	Addition	Doublement
$\mathcal{A}$	$1I + 2M + 1S$	$I + 2M + 1S$
$\mathcal{P}$	$14M + 1S$	$7M + 3S$
$\mathcal{J}$	$14M + 5S$	$4M + 5S$
$\mathcal{L}$	$13M + 4S$	$3M + 5S$

TABLE 5.4 – Récapitulatif des complexités des opérations d'additions et doublements dans différentes coordonnées sur $\mathbb{F}_{2^m}$.

$$G = FD, \quad H = Z_3 + D$$

$$\begin{cases} X_3 & = aZ_3^2 + DH + C, \\ Y_3 & = X_3H + E + G, \\ Z_3 & = Z_1A. \end{cases}$$

Comme dans $\mathbb{F}_q$, on peut modifier le point d'entrée P_1 tel qu'il ait la même coordonnée Z en sortie que P_3 :

$$\begin{cases} X_1 & = F, \\ Y_1 & = E, \\ Z & = Z_3. \end{cases}$$

L'addition simplifiée requiert $7M + 2S$. Nous utilisons cette formule pour construire un algorithme de multiplication scalaire présenté dans le chapitre 9.

Chapitre 6

Algorithmes efficaces de multiplication scalaire

Comme nous l'avons présentée dans la section 5.6, la multiplication scalaire est une opération majeure dans la plupart des cryptosystèmes à base de courbes elliptiques. L'étude de sa performance et de sa résistance face aux attaques est donc primordiale. L'opération consiste à calculer $[k]P$ où k est un entier et P un point d'une courbe elliptique. Dans certaines situations le point P ou le scalaire k peuvent être fixés et connus à l'avance permettant ainsi différentes optimisations. On considère pour notre étude le cas où P et k sont inconnus à l'avance, tirés au hasard au début de la multiplication.

Pour analyser la complexité d'un algorithme de multiplication scalaire, il faut compter le nombre d'opérations calculées sur la courbe elliptique. Mais il faut aussi tenir compte du nombre d'opérations calculées dans le corps de base qui dépend du système de coordonnées choisi et du type de la courbe elliptique. On ne considère pas dans notre étude les différentes familles de courbes elliptiques mais seulement les courbes génériques sous forme de Weierstrass. Pour les opérations sur la courbe elliptique, on note A le temps nécessaire au calcul d'une addition de points et D le temps nécessaire pour un doublement. Pour les opérations sur le corps de base, on note M le coût d'une multiplication, S le coût d'une élévation au carré et I celui d'une inversion. On néglige le temps nécessaire aux calculs d'additions et soustractions dans le corps de base.

6.1 Méthode de doublement-et-addition

Une méthode naïve pour calculer $[k]P$, noté aussi simplement kP, est d'ajouter P à lui-même k fois. Cette méthode très inefficace requiert k opérations sur la courbe elliptique. Mais la représentation binaire du scalaire k permet d'obtenir un algorithme ne demandant plus que de l'ordre de $\log_2(k)$ opérations. Soit $k \in \mathbb{N}$ de représentation binaire $(k_{n-1}, \ldots, k_1, k_0)_2$ avec $k_i \in \{0, 1\}$ et tel que $k_{n-1} \neq 0$, alors $k = \sum_{i=0}^{n-1} k_i 2^i$. En utilisant la

méthode de Horner [Knu98, Chapitre 4], on peut écrire :

$$kP = \sum_{i=0}^{n-1} k_i 2^i P \tag{6.1}$$

$$= k_0 P + k_1 2^1 P + k_2 2^2 P + \cdots + k_{n-1} 2^{n-1} P \tag{6.2}$$

$$= k_0 P + 2(k_1 P + 2(k_2 P + \cdots + (2(k_{n-2} P + 2(k_{n-1} P)) \ldots). \tag{6.3}$$

La formule (6.2) consiste, à partir de k_0, et à additionner des termes $k_i 2^i P$ pour chaque $k_i \neq 0$ jusqu'à k_{n-1} afin d'obtenir au final le résultat kP. Si le point $2^{i-1}P$ est connu, on peut calculer facilement $2^i P$ tel que $2^i P = 2 \times 2^{i-1} P$. Pour accélérer le calcul, les valeurs $2^i P$ sont calculées pour chaque valeur de $i \in \{0, \ldots, n-1\}$ en doublant la valeur calculée au tour précédent. Il suffit ensuite d'additionner dans une variable d'accumulation cette valeur si $k_i = 1$. Cette méthode est appelée *right-to-left* doublement-et-addition (Algorithme 6) car on effectue des doublements et additions de points en partant de k_0 jusqu'à k_{n-1}.

Grâce à la formule (6.3), on peut interpréter la multiplication comme partant de k_{n-1} jusqu'à k_0. On additionne P si $k_i \neq 0$ et on calcule un doublement pour chaque i. Cette méthode est appelée *left-to-right* doublement-et-addition (Algorithme 2).

Algorithme 2: *Left-to-right* doublement-et-addition

Entrées : $P \in E$ et $k = (k_{n-1} \ldots k_1 k_0)_2$
Sorties : $[k]P \in E$

1 $P_0 \leftarrow \mathcal{O}$
2 $P_1 \leftarrow P$
3 **pour** $i \leftarrow n-1$ **a** 0 **faire**
4 $\quad P_0 \leftarrow [2]P_0$
5 $\quad$ **si** $k_i = 1$ **alors**
6 $\quad\quad P_0 \leftarrow P_0 + P_1$
7 **retourner** P_0

La complexité de l'Algorithme 2 dépend du nombre d'additions et doublements de points effectués. Seules les opérations calculées au sein de la boucle sur les bits du scalaire ne sont considérées dans la complexité finale. En effet, le temps de calcul des premières lignes de l'algorithme est négligeable dans le temps d'exécution total. À la ligne 4, on calcule un doublement de point indépendamment de la valeur du bit k_i. Le scalaire k étant considéré comme tiré au hasard, le nombre de 1 dans sa représentation binaire, i.e. son poids de Hamming, est en moyenne de $n/2$. L'addition de points à la Ligne 6 est donc effectuée en moyenne $n/2$ fois. La complexité moyenne de l'Algorithme 2 est :

$$nD + \frac{n}{2}A.$$

L'Algorithme 6 effectue les mêmes opérations dans l'ordre inverse. Sa complexité est donc identique.

Lorsqu'on considère la complexité en opérations dans le corps de base d'un algorithme de multiplication, on donne souvent la complexité d'un tour de boucle, soit la complexité $D + 0,5A$ pour les algorithmes précédents. Dans les corps finis à grande caractéristique, on utilise souvent les coordonnées jacobiennes car elles offrent un bon compromis entre la performance et l'espace mémoire requis. On se rappelle qu'une addition de points coûte $11M + 5S$ et un doublement $1M + 8S$ dans ce système. La complexité, en opérations de base, de l'algorithme *left-to-right* doublement-et-addition, et de son symétrique *right-to-left* doublement-et-addition, est alors en moyenne de :

$$\underbrace{(1M + 8S)}_{\text{DBL}} + 0,5\,\underbrace{(11M + 5S)}_{\text{ADD}} = 6,5M + 10,5S.$$

6.2 Forme non-adjacente

Nous avons vu dans les sections précédentes que calculer l'opposé d'un point de courbe elliptique se fait rapidement. En effet, si $P = (x, y) \in E(K)$ avec la caractéristique de K strictement supérieure à 3, alors $-P = (x, -y)$ et si la caractéristique de K est égale à 2, alors $-P = (x, x + y)$. L'idée consiste à utiliser une représentation du nombre scalaire k telle que :

$$k = \sum_{i=0}^{n-1} k_i 2^i \quad \text{avec } k_i \in \{-1, 0, 1\}$$

pour accélérer le calcul de kP. On l'appelle représentation binaire signée.

La complexité d'un algorithme de multiplication scalaire de type doublement-et-addition augmente proportionnellement avec le poids de Hamming du scalaire dans sa représentation binaire. Le but est donc de trouver une représentation possédant le plus de 0 possibles tout en étant relativement courte. Une représentation binaire signée particulière satisfait à ces propriétés.

Définition 6.2.1. *Une forme non-adjacente (*Non-Adjacent Form*, NAF) d'un entier positif k est de la forme $k = \sum_{i=0}^{n-1} k_i 2^i$ avec $k_i \in \{-1, 0, 1\}$, $k_{n-1} \neq 0$ et telle qu'il ne peut pas y avoir deux k_i consécutifs qui soient différents de zéro.*

Pour un entier k positif, la représentation NAF a les propriétés suivantes :

Théorème 6.2.1. *Soit k un entier positif.*

- *k a une unique forme non-adjacente, notée* NAF(k) *[Rei60].*
- NAF(k) *est la représentation binaire signée qui possède le moins de chiffres différents de zéro [Rei60].*
- *Le nombre moyen de chiffres différents de zéro dans* NAF(k) *de longueur n est $n/3$ [MO90].*
- *La longueur de* NAF(k) *est, au plus, d'un chiffre plus grande que la représentation binaire classique de k [MO90].*

Algorithme 3: NAF d'un entier positif

Entrées : $k = (k_{n-1} \dots k_1 k_0)_2$ positif
Sorties : NAF(k)

```
1  i ← 0
2  tant que k > 0 faire
3      si k est impair alors
4          k'_i ← 2 − (k mod 4)
5          k ← k − k'_i
6      sinon
7          k'_i ← 0
8      k ← k/2
9      i ← i + 1
10 retourner (k'_n, k'_{n−1}, ..., k'_0)
```

Différents algorithmes ont été proposés pour calculer NAF(k) pour un k donné [Rei60, MO90, HVM04]. Une version classique est présentée avec l'Algorithme 3.

On peut désormais présenter un algorithme de multiplication scalaire utilisant cette représentation NAF (Algorithme 4).

Algorithme 4: *Left-to-right* NAF

Entrées : $P \in E$ et $NAF(k) = (k_{n-1} \dots k_1 k_0)$
Sorties : $[k]P \in E$

```
1  P_0 ← O
2  P_1 ← P
3  pour i ← n − 1 a 0 faire
4      P_0 ← [2]P_0
5      si k_i = 1 alors
6          P_0 ← P_0 + P_1
7      si k_i = −1 alors
8          P_0 ← P_0 − P_1
9  retourner P_0
```

On effectue un doublement à chaque tour de boucle Ligne 4. Grâce au Théorème 6.2.1, on sait que, en moyenne, les Lignes 6 et 8 sont calculées $n/3$ fois. La complexité moyenne de l'Algorithme 4 est donc :

$$nD + \frac{n}{3}A.$$

Si on considère, comme précédemment, l'utilisation de coordonnées jacobiennes on ob-

tient, pour un tour de boucle, une complexité moyenne de :

$$\underbrace{(1M + 8S)}_{\text{DBL}} + \frac{1}{3}\underbrace{(11M + 5S)}_{\text{ADD}} \approx 4,6M + 9,6S.$$

On peut aussi effectuer les techniques classiques de fenêtrage et fenêtrage glissant bien connues dans le cadre de l'exponentiation modulaire sur des corps finis. Ces méthodes permettent de gagner en performances au prix de précalculs et de stockage mémoire supplémentaire. Notre étude étant placée dans le cadre des environnements embarqués possédant un espace mémoire limité, nous n'abordons pas ces techniques. Le lecteur intéressé peut se reporter au récent état de l'art fait par Sullivan [Sul08].

Chapitre 7

Attaques physiques sur cryptosystèmes à base de courbes elliptiques

7.1 Attaques par injection de fautes

Nous présentons dans cette section les principales attaques par injection de fautes dans le cas des cryptosystèmes à base de courbes elliptiques. Ce type d'attaque a d'abord été appliqué contre l'algorithme RSA [BDL01], elles ont été rapidement adaptées pour les courbes elliptiques. On peut les classer en trois catégories. Les attaques sans conséquence sur les calculs permettent à l'attaquant d'obtenir des informations sur le secret lorsqu'une faute injectée n'implique aucun résultat faux. L'attaquant peut aussi injecter une faute afin d'obliger le programme à effectuer des calculs sur une courbe elliptique cryptographiquement faible. Enfin les attaques par fautes différentielles analysent les différences entre un calcul correct et des calculs erronés afin de retrouver le secret. Alors que dans la majorité des cryptosystèmes asymétriques classiques on cherche à retrouver l'exposant secret utilisé dans une exponentiation modulaire. Dans les cryptosystèmes à base de courbes elliptiques, on cherche le scalaire secret dans l'opération de multiplication scalaire. L'exponentiation modulaire est l'équivalent dans un groupe multiplicatif de la multiplication scalaire qui est utilisée dans un groupe additif. On confond donc souvent l'analyse de ces deux algorithmes qui, très souvent, souffrent de vulnérabilités similaires.

7.1.1 Attaques sans conséquences

Le concept d'attaques par fautes sans conséquences a été présenté par Joye et al. et Yen et al. dans [YJ00, JY02, YKLM02] sous le nom de *safe-error* dans le cadre de la signature RSA. Cette signature utilise un algorithme d'exponentiation modulaire. Pour présenter ces attaques, nous reprenons le cas de l'algorithme carré-et-toujours-multiplication (Algorithme 5) d'exponentiation modulaire étudié dans les articles originaux. Cet algorithme utilise

une opération factice afin de toujours calculer le même nombre d'opérations quelque soit la valeur du bit secret d_i. L'adaptation au cas des courbes elliptiques est immédiate étant donné qu'il existe un algorithme similaire à l'Algorithme 5, appelé doublement-et-toujours-addition (Algorithme 7).

Algorithme 5: Carré-et-toujours-multiplication

Entrées : Un message $M \in \mathbb{Z}_N$, un exposant secret RSA d, n la longueur en bits de d, N le modulo RSA

Sorties : $M^d \bmod N$

1 $y_0 \leftarrow 1$
2 $y_1 \leftarrow M$
3 **pour** $i \leftarrow n-1$ **a** 0 **faire**
4 $\quad y_0 \leftarrow y_0^2 \bmod N$
5 $\quad y_1 \leftarrow y_0 M \bmod N$
6 $\quad y_0 \leftarrow y_{d_i}$
7 **retourner** y_0

7.1.1.1 Faute sans conséquence sur la mémoire

On considère que l'attaquant est capable d'injecter une faute à la Ligne 5 de l'Algorithme 5 lors du calcul de $y_1 \leftarrow y_0 M \bmod N$. La faute injectée modifie en mémoire des bits de y_0 une fois que ceux-ci ont été utilisés. Ainsi la valeur y_1 calculée est correcte alors que y_0 est faux. Si le bit du secret $d_i = 1$, alors à la Ligne 6 la valeur de y_0 est écrasée par y_1 qui est juste. La faute n'a donc aucun effet sur le résultat final. Mais si $d_i = 0$, alors la valeur de y_0 reste sa valeur erronée. Ainsi, aux prochains tours, la faute se propage et entraîne un résultat final faux. L'attaquant est alors capable de retrouver la valeur du bit d_i suivant si le résultat final est correct ou non. Ces fautes requièrent un attaquant relativement puissant puisqu'il doit pouvoir injecter une faute lors d'une opération précise, à l'endroit précis où se trouve y_0 en mémoire, une fois qu'il a été calculé. Dans la littérature, ce type d'attaque est souvent appelé *memory-safe error* ou *M-safe error*.

7.1.1.2 Faute sans conséquence sur les calculs

L'attaque précédente utilise le fait d'injecter une faute dans des bits en mémoire qui ne sont, peut-être, plus utilisés. L'idée peut être étendue afin d'injecter des fautes sur une opération dont la sortie n'est, peut-être, pas utilisée. Ces opérations sont généralement appelées opérations factices ou *dummy operations*. On se place encore dans le cas de l'Algorithme 5. On considère désormais qu'une faute est injectée dans le calcul de y_1 Ligne 5. Si le bit $d_i = 1$ au tour où cette faute est injectée, alors la valeur erronée y_1 se propage aux tours suivants pour un résultat final faux. Mais si $d_i = 0$, l'erreur n'a aucun effet sur le résultat. Comme précédemment, l'attaquant peut facilement retrouver le bit d_i en

analysant le résultat de l'exponentiation. Ce type d'attaque requiert moins d'hypothèses sur l'attaquant que l'attaque sans conséquence sur la mémoire. En effet, il n'a plus qu'à injecter une faute lors du calcul de y_1, Ligne 5. Il dispose d'une fenêtre plus grande pour injecter la faute, l'attaque est donc plus facilement réalisable. Dans la littérature, ce type d'attaque est souvent appelé *computational-safe error* ou *C-safe error*. Elles s'appliquent plus généralement sur tous les algorithmes effectuant des opérations factices en fonction de la valeur du secret.

7.1.2 Courbes cryptographiquement plus faibles

Soit E une courbe elliptique définie par l'équation de Weierstrass (5.1). Biehl et al. [BMM00] remarquent que lors du calcul d'une multiplication scalaire, le coefficient a_6 n'est pas utilisé. En changeant la courbe E en une courbe E' telle que :

$$E' : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a'_6,$$

le résultat de la multiplication scalaire est identique. Si un cryptosystème reçoit un point $P' = (x', y')$ tel que $x', y' \in K$ mais $P' \notin E$, alors la multiplication scalaire $[k]P'$ s'effectue sur la courbe E' définie ci-dessus avec $a'_6 = y'^2 + a_1x'y' + a_3y' - x'^3 - a_2x'^2 - a_4x'$ au lieu de la courbe originale E. Le point P' est choisi tel que le cardinal du groupe de points de E' ait un petit facteur r et tel que P' soit d'ordre r. Si r est relativement petit, l'attaquant peut résoudre le problème du logarithme discret dans le sous-groupe d'ordre r et retrouver $k_r = k \bmod r$. En répétant l'injection de faute, l'attaquant obtient plusieurs k_r pour différents r et, grâce au théorème des restes chinois, retrouve k. Cette attaque n'est réalisable que si l'attaquant peut choisir P', ce qui n'est pas possible dans le cas de l'*Elliptic Curve Digital Signature Algorithm* (ECDSA) par exemple [HVM04, Section 4.4.1].

Ciet et Joye [CJ05] généralisent les travaux de [BMM00] en proposant des attaques avec des contraintes moins fortes pour l'attaquant. En particulier, une attaque est possible en injectant une faute quelconque sur la coordonnée x ou y du point P. Avec des suppositions plus fortes, l'attaquant peut même retrouver le secret k en ayant injecté une faute quelconque sur les deux coordonnées. Ciet et Joye proposent aussi d'injecter une erreur dans le corps de base de la courbe elliptique, soit dans le premier p de $\mathbb{F}_p$ si un corps de grande caractéristique est utilisé, soit dans le polynôme irréductible de $\mathbb{F}_{2^m}$ pour un corps de caractéristique 2. Les auteurs considèrent aussi le cas d'une injection de faute dans un des paramètres de la courbe a_1, a_2, a_3 ou a_4. Plus de détails sur ces attaques sont disponibles dans l'article des auteurs [CJ05].

En 2008, Fouque et al. [FLRV08] présentent une attaque par faute sur l'algorithme de multiplication scalaire appelé échelle de Montgomery (Algorithme 9). Une faute peut permettre de passer d'une courbe elliptique E à une courbe $\tilde{E}$ qui est la courbe tordue quadratique de E (voir section 10.7) qui peut être cryptographiquement faible. Un algorithme de multiplication scalaire qui effectue le calcul sans la coordonnée y des points, comme certaines versions de l'échelle de Montgomery, retourne le bon résultat peu importe si les calculs sont faits dans E ou $\tilde{E}$. Pour les courbes elliptiques définies sur $\mathbb{F}_q$, une coordonnée

$x \in \mathbb{F}_q$ choisie aléatoirement correspond à soit un point de E, soit un point de $\tilde{E}$. L'attaque est donc très facilement réalisable.

7.1.3 Attaque par fautes différentielle

La première attaque par fautes différentielle sur cryptosystèmes à base de courbes elliptiques a été proposée par Biehl et al. [BMM00]. On utilise l'algorithme de multiplication scalaire classique *right-to-left* doublement-et-addition (Algorithme 6) pour expliciter l'attaque.

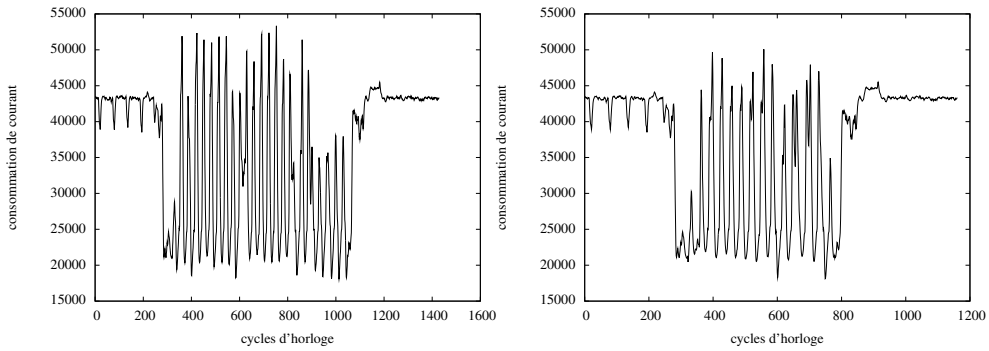
Algorithme 6: *Right-to-left* doublement-et-addition

Entrées : $P \in E$ et $k = (k_{n-1} \dots k_1 k_0)_2$
Sorties : $[k]P \in E$

1 $P_0 \leftarrow \mathcal{O}$
2 $P_1 \leftarrow P$
3 **pour** $i \leftarrow 0$ **a** $n-1$ **faire**
4 **si** $k_i = 1$ **alors**
5 $P_0 \leftarrow P_0 + P_1$
6 $P_1 \leftarrow [2]P_1$
7 **retourner** P_0

Soit P_0^i et P_1^i les valeurs de P_0 et P_1 à la fin de la i-ème itération de boucle. On note $\tilde{P}^i$ lorsqu'une faute a été injectée dans P_0^i. L'attaquant a tout d'abord besoin d'un résultat de multiplication scalaire correct $Q = [k]P$. Il va retrouver les bits de k en partant du bit le plus significatif. L'attaquant injecte une faute sur un seul bit de P_0^i avec $n - m \leq i < n$ où m est supposé petit. Il obtient le résultat erroné $Q' = \tilde{P}^i + (\lfloor k/2^i \rfloor 2^i)P$ alors qu'il possède $Q = P_0^i + (\lfloor k/2^i \rfloor 2^i)P$. Il n'a plus qu'à essayer tous les $\lfloor k/2^i \rfloor \in \{0, 1, \dots, 2^m - 1\}$ possibles et calculer des P_0^i et $\tilde{P}^i$ correspondants. La vraie valeur de $\lfloor k/2^i \rfloor$ donne un couple $(P_0^i, \tilde{P}^i)$ qui diffère d'un seul bit.

En 2006, Blömer et al. [BOS06] proposent l'attaque par changement de signe, ou *sign-change fault attack*. Pour se prémunir de la grande majorité des attaques précédentes, on peut vérifier à la fin de la multiplication scalaire que le point appartient bien à la courbe avant de le renvoyer. L'attaque de Blömer et al. est basée sur l'idée de changer le signe du point lors du calcul. On reste alors avec des points appartenant à la courbe d'origine. Il faut noter que cette attaque n'est réalisable que pour des courbes elliptiques définies sur des corps $\mathbb{F}_q$ où changer le signe d'un point revient seulement à changer le signe de sa coordonnée y. Dans l'article [BOS06], les auteurs considèrent l'attaque sur un algorithme de type NAF (Algorithme 4). Dans ce cas, l'attaquant doit injecter une faute par changement de signe à la Ligne 6 pour calculer $P_0 \leftarrow -P_0 + P_1$. La faute peut être injectée à un tour de boucle i, inconnu de l'attaquant, et il peut tout de même retrouver le secret. Otto [Ott04] étend ce type d'attaque à d'autres algorithmes de multiplication scalaire comme le



(a) Addition de points (b) Doublement de point

FIGURE 7.1 – Consommation de courant des opérations de base sur les points d'une courbe elliptique.

doublement-et-addition et l'échelle de Montgomery utilisant les coordonnées y.

7.2 Attaques par analyse simple

Comme précédemment, on étudie l'opération de multiplication scalaire, essentielle dans de nombreux cryptosystèmes à base de courbes elliptiques. Cette étude peut en grande partie s'étendre à l'opération similaire d'exponentiation modulaire. L'attaquant observe une courbe de consommation de courant d'une multiplication scalaire $Q = [k]P$ avec Q et P deux points d'une courbe elliptique et $k \in \mathbb{Z}$ un entier. Il cherche à retrouver ce scalaire k grâce aux informations obtenues par le canal caché : la consommation du circuit. L'algorithme de multiplication scalaire effectue une suite d'additions de doublements de points (par exemple, Algorithme 6). À leur tour, ces opérations d'additions et doublements consistent en différentes opérations dans le corps de base (voir section 5.8). Ces différences s'observent facilement sur une courbe de consommation (Figure 7.1).

Dans [Cor99], Coron remarque que l'algorithme classique de doublement-et-addition (Algorithme 6) est vulnérable à une attaque par analyse simple. La branche conditionnelle à la Ligne 4 de l'Algorithme 6 implique qu'une addition de points n'est calculée que lorsque le bit k_i du scalaire k vaut 1. Ainsi, en inspectant une courbe de consommation de cet algorithme, l'attaquant déduit très facilement le secret. Par exemple, dans la Figure 7.2 on observe la suite d'opérations sur les points : DADDDAD, en notant D un doublement de point et A une addition. On déduit donc les bits du scalaire $k = (10010)_2 = 18$, grâce à une seule courbe de consommation acquise sur le circuit.

Un autre type d'attaque, proposé par Fouque et Valette [FV03], est appelée *doubling attack* ou attaque du doublement. Plus généralement, elle appartient aux attaques par collision de données. Les auteurs supposent qu'un attaquant est capable de discerner, sur une courbe de consommation d'une multiplication scalaire, si le composant calcule $[2]A$ ou $[2]B$ sans pour autant connaître les valeurs de A ou B. Ainsi, si il y a une collision et que A

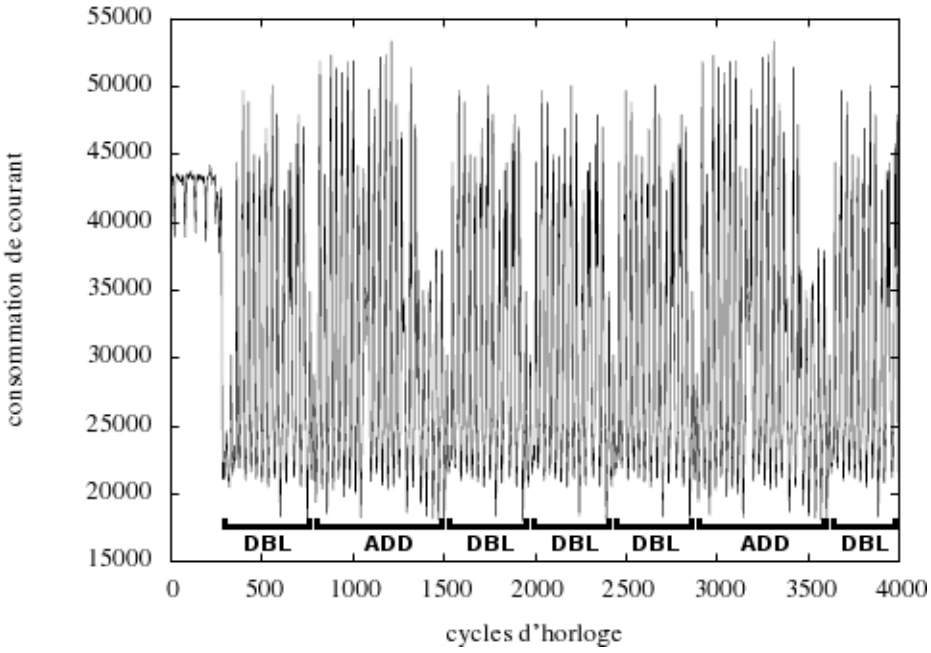


FIGURE 7.2 – Consommation de courant d'un algorithme classique de multiplication scalaire doublement-et-addition. Les bits du scalaire apparaissent clairement sur la courbe en repérant les opérations d'additions et doublements de points.

est égal à B, il est capable de le détecter. Cette supposition a été validée par Schramm et al. [SWP03]. Fouque et Valette étudient l'algorithme de multiplication scalaire doublement-et-toujours-addition, ou *double-and-always-add* (Algorithme 7), parcourant les bits du scalaire du plus significatif au moins significatif, appelé *left-to-right*. Cet algorithme est l'équivalent pour les courbes elliptiques de l'Algorithme 5. Fouque et Valette préconisent l'utilisation d'algorithmes parcourant les bits du moins au plus significatif, *right-to-left*, car ceux-ci résistent à cette attaque. On étudie pour plus de simplicité cette attaque sur l'algorithme classique de doublement-et-addition (Algorithme 2).

La variable P_0, à la Ligne 6 Algorithme 2, contient la somme partielle des P à la fin de chaque tour de boucle. On note $P_0^i(P)$ cette valeur à la fin du tour i où P est le point de départ. On peut noter :

$$\begin{aligned} P_0^i(P) &= \left[\sum_{j=i}^{n-1} k_j 2^{j-i}\right] P \\ &= \left[\sum_{j=i+1}^{n-1} k_j 2^{j-i-1}\right] [2]P + [k_j]P \\ &= P_0^{i+1}([2]P) + [k_j]P. \end{aligned}$$

On remarque donc que $P_0^i(P) = P_0^{i+1}([2]P)$ si et seulement si le bit $k_j = 0$. Cela signifie que la valeur intermédiaire au tour i avec P comme point de départ est égale à la

i	4	3	2	1	0
k_i	1	**0**	**0**	1	1
$P_0^i(P)$	P	**[2]P**	**[4]P**	$[9]P$	$[19]P$
$P_0^i([2]P)$	**[2]P**	**[4]P**	$[8]P$	$[18]P$	$[38]P$

TABLE 7.1 – Calculs de $[k]P$ et $[k]([2]P)$ où $P_0^i(P)$ est la valeur intermédiaire de la multiplication scalaire de $[k]P$ pour le $i$ème bit de k (de même pour $P_0^i([2]P)$).

valeur intermédiaire au tour $i+1$ avec $[2]P$ comme point de départ. L'attaquant enregistre donc deux courbes de consommation, une pour la multiplication $[k]P$ et une autre pour $[k]([2]P)$. Si, au tour i pour P et $i-1$ pour $[2]P$, l'attaquant remarque une collision, alors les valeurs intermédiaires des deux multiplications sont égales. Il sait alors que le bit k_i vaut 0. Les autres bits du secret, sauf le moins significatif, se retrouve de la même manière. Un exemple de fonctionnement de l'attaque est présenté dans la Table 7.1. Le scalaire secret $k = (10011)_2 = 19$ est utilisé lors de deux multiplications scalaires avec P et $[2]P$ en entrée. On remarque des collisions pour $i = 3, 2$ et donc k_3 et k_2 valent 0. On déduit donc que les bits k_4 et k_1 valent 1. Pour retrouver complètement le scalaire, on teste les deux valeurs possibles pour le dernier bit k_0 car l'attaque n'est pas capable de la retrouver.

Une modification de l'attaque a ensuite été proposée par Yen et al. [YKMH06] sur l'algorithme de multiplication scalaire l'échelle de Montgomery. Au lieu de trouver la valeur des bits, cette attaque donne des relations entre les bits pour deux calculs avec P et $[2]P$ en entrée. Une généralisation de ces attaques, dites par collision, est étudiée par Homma et al. [HMA+08]. Alors que Fouque et Valette [FV03] proposent d'utiliser les variantes *right-to-left* des algorithmes de multiplication scalaire, dans [YLT04] Yen et al. affirment que l'algorithme *left-to-right* proposé dans [YLT04] est résistant aux attaques du doublement. Kim et al. [KQ08] attaquent cet algorithme. L'attaque est ensuite améliorée dans [Wu10].

7.3 Attaques par analyse différentielle

7.3.1 Attaque un exposant multiple données

L'attaque un exposant multiple données (*Single Exponent Multiple Data*, SEMD) [MDS99b] est une attaque différentielle proposée pour l'algorithme RSA mais elle s'applique directement aux cryptosystèmes à base de courbes elliptiques. L'opération que l'on cherche à attaquer est encore une fois la multiplication scalaire $Q = [k]P$ où k est un entier, secret, et P et Q sont deux points d'une courbes elliptique. L'attaque SEMD suppose que l'attaquant a accès à N courbes de consommation de l'opération $[d]P_i$ où d est public et P_i avec $i = 1, \ldots, N$ sont des points aléatoires de la courbe. Ces courbes de consommation sont notées $\mathcal{C}_i^d$. On suppose qu'il cherche à retrouver le secret k, fixé dans le composant. Il peut enregistrer N courbes de consommation de courant pour les mêmes N points que précédemment avec l'exposant secret k. Ces courbes sont notées $\mathcal{C}_i^k$. L'attaquant calcule la

courbe moyenne de chacun des deux ensembles et les soustrait :

$$\Delta = \frac{1}{N}\sum_{i=1}^{N} \mathcal{C}_i^d - \frac{1}{N}\sum_{i=1}^{N} \mathcal{C}_i^k = \bar{\mathcal{C}}_i^d - \bar{\mathcal{C}}_i^k.$$

Si, à l'instant t, la même opération est effectuée avec chacun des deux exposants, alors la différence des courbes moyennes est nulle. Dans le cas contraire, la différence est non nulle. Comme généralement les courbes de consommation contiennent du bruit, la différence n'est jamais exactement égale à zéro même pour des opérations identiques. Il faut donc, en pratique, supposer une valeur seuil.

7.3.2 Attaque multiple exposants une donnée

L'attaque multiple exposants une donnée (*Multiple Exponent Single Data*, MESD) [MDS99b] est plus puissante que l'attaque SEMD mais requiert des suppositions supplémentaires sur le composant attaqué. En effet, on suppose que l'attaquant est capable de réaliser des multiplications scalaires $[d]P$ pour N valeurs de d, connues de l'attaquant, et pour P un point fixé de la courbe. Pour chaque multiplication, il enregistre la courbe de consommation correspondante $\mathcal{C}^d$. Soit k le scalaire secret cherché par l'attaquant et $\mathcal{C}^k$ la courbe correspondant à la multiplication $[k]P$. On suppose que l'adversaire connait déjà les $(j-1)$ bits de poids forts de k. Il cherche donc le bit j du secret. Il va pour cela choisir un scalaire d tel que les $(j-1)$ bits de poids forts correspondent à ceux déjà connus de k et il va fixer le bit j à 0 et à 1. Il réalise ces deux multiplications scalaires et enregistre les courbes de consommations respectives $\mathcal{C}^d_{j=0}$ et $\mathcal{C}^d_{j=1}$. L'attaquant calcule ensuite les deux traces :

$$\Delta_0 = \mathcal{C}^k - \mathcal{C}^d_{j=0},$$
$$\Delta_1 = \mathcal{C}^k - \mathcal{C}^d_{j=1}.$$

Pour la valeur du bit j correcte, les calculs intermédiaires à cette étape correspondent à ceux effectués lors du calcul avec le secret k. La différence entre les deux courbes doit donc être proche de zéro à cet instant. Au contraire, pour la fausse valeur du bit j, on observe une différence plus importante. L'attaquant retrouve ainsi les bits de k au fur et à mesure.

7.3.3 Attaque zero exposant multiple données

L'attaque zero exposant multiple données (*Zero Exponent Multiple Data*, ZEMD) [MDS99b] ressemble à MESD, car l'attaque cherche le secret bit à bit, mais diffère sur les hypothèses de départ. Nous supposons désormais l'attaquant capable de calculer des multiplications scalaires avec différents points P_i aléatoires connus et avec le scalaire secret k. Ce scénario est plus plausible que celui de MESD dans la majorité des cas. L'adversaire simule l'exécution d'une multiplication scalaire avec les points P_i en supposant un scalaire d. On considère que l'attaquant connait les $(j-1)$ bits de k et cherche le bit j qu'il fixe à 1. Il

peut recalculer une multiplication scalaire jusqu'à l'étape où le bit j du scalaire est traité. Il dispose ainsi d'une valeur intermédiaire de calcul dépendante du bit recherché. En appliquant un modèle de consommation (voir section 1.2.3.3), par exemple le poids de Hamming de la valeur intermédiaire, l'attaquant classe les courbes de consommation des multiplications $[k]P_i$ en deux ensembles suivant l'importance du poids de Hamming. Si l'adversaire suppose une valeur correcte pour le bit j, la différence entre les deux ensembles va produire des pics significatifs. Si la valeur est fausse, la différence va être proche de zéro. Comme dans le cas de la MESD, l'attaquant retrouve les bits de k au fur et à mesure.

7.3.4 Attaque sur les bits d'adresse en mémoire

L'attaque sur les bits d'adresse en mémoire (*Address-bit DPA*, ADPA), contrairement aux précédentes, ne tient pas compte de la valeur des données. Ainsi les contre-mesures basées sur l'ajout d'aléa dans les données n'ont pas d'effet. Le principe a été proposé par Messerges et al. [MDS99a] puis il a été développé et réalisé en pratique par Itoh et al. [IIT03b]. Si une même donnée est chargée dans deux registres mémoires différents, l'attaquant doit pouvoir le distinguer dans la consommation de courant du composant grâce au poids de Hamming des adresses. L'influence des données sur les courbes est réduite en réalisant des moyennes. L'attaque fonctionne donc lorsqu'il y a une relation nette entre la clé et les adresses des registres accédés. Une solution à ce type d'attaque est d'ajouter de l'aléa dans le scalaire k de la multiplication $[k]P$ (voir section 8.2.1).

Il existe aussi des techniques afin de randomiser l'utilisation des registres utilisés comme celles présentées dans [MMS01] et [IIT03a].

7.3.5 Attaque sur des points « spéciaux »

L'attaque sur des points spéciaux (*Refined Power Analysis*, RPA) est une attaque différentielle contre les cryptosystèmes à base de courbes elliptiques. Cette attaque, proposée par Goubin [Gou03], se base sur les points « spéciaux » d'une courbe elliptique définie sur un corps fini K. Un point « spécial » P_0 est tel que $P_0 \neq \mathcal{O}$ en ayant une de ses coordonnées égale à 0 dans K. L'auteur suppose que la consommation de courant de 0 se remarque par rapport à une valeur quelconque. On considère que l'attaquant connait les $(j-1)$ bits plus significatifs du scalaire secret k. Il fait une hypothèse sur le bit k_j de telle sorte qu'il puisse trouver un multiple P_1 d'un point « spécial » P_0. Ainsi, si au tour j de l'algorithme de multiplication scalaire de $[k]P_0$ la valeur intermédiaire $[c]P_0$ est calculée alors l'attaquant choisit le point $P_1 = [c^{-1} \bmod |E(K)|]P_0$. Si l'hypothèse sur k_j est correcte, alors la multiplication $[k]P_1$ va calculer le point $[cc^{-1}]P_0 = P_0$ au tour j. Or la consommation de courant du point « spécial » P_0 est remarquable par rapport à celle d'un point quelconque. L'attaquant peut donc retrouver la valeur de k_j. Il procède de manière similaire pour les bits suivants.

Dans [AT03], Akishita et Takagi proposent une généralisation de l'attaque de Goubin qu'ils appellent *Zero-value Point Attack* (ZPA). Leur attaque utilise le fait que des valeurs égales à 0 dans le corps K apparaissent dans des registres lors d'une multiplication scalaire.

Il faut pour cela étudier quel algorithme de multiplication scalaire est utilisé ainsi que les algorithmes de doublement et d'addition de points utilisés. On peut alors trouver le point qui, donné en entrée de la multiplication scalaire, implique un calcul intermédiaire égal à zéro à un tour donné.

Chapitre 8

Protections contre les attaques par canaux cachés

Nous avons étudié dans le chapitre précédent les différentes attaques par canaux cachés qui peuvent s'adapter aux cryptosystèmes à base de courbes elliptiques. Nous présentons dans ce chapitre des contre-mesures afin de se protéger face à chacun des types d'attaques présentés.

8.1 Face à une attaque par analyse simple

Nous avons vu dans la section 2.1 que le principe d'une attaque par analyse simple est de déduire le secret à partir d'une observation de canal caché. Il existe deux catégories de protections face à ce type d'attaque pour les cryptosystèmes à base de courbes elliptiques. On peut faire en sorte que :

- le chemin d'exécution de l'algorithme de multiplication scalaire soit indépendant de la valeur du secret,
- l'addition et le doublement de points exécutent les mêmes opérations afin de les rendre indistinguables.

Les formules pour l'addition ou le doublement de points sur courbes elliptiques standards sont très différentes (voir section 5.8). Par une simple observation des courbes de consommation de courant, un attaquant peut facilement distinguer ces algorithmes et donc peut retrouver des bits du secret. En exécutant une séquence fixe d'opérations, indépendante de la valeur du secret, on peut se protéger de ces attaques.

8.1.1 Rendre les opérations indistinguables

On distingue deux méthodes permettant d'avoir des formules d'addition et doublement de points indistinguables. Les standards internationaux traitant des courbes elliptiques [X9.98, SEC00, Nat00] imposent toujours l'utilisation de courbes elliptiques sous forme de Weierstrass. Dans une première partie, nous analysons donc les propositions de formules

unifiées pour ces courbes standards. Malheureusement, elles n'ont pas, par nature, des formules d'addition et doublement proches l'une de l'autre. Ainsi les formules unifiées ne sont pas souvent efficaces. Dans une deuxième partie, nous étudions le cas de familles de courbes elliptiques qui ont, par construction, une formule unifiée. Leur efficacité est donc souvent bien meilleure mais ces familles ne couvrent généralement pas le cas des courbes sous forme de Weierstrass standardisées.

8.1.1.1 Formules unifiées

Une solution aux attaques par analyse simple consiste à avoir un même algorithme permettant de calculer l'addition et le doublement de points. Sur les courbes de Weierstrass, ces deux opérations ont des complexités différentes. Brier et Joye [BJ02] proposent une formulation valable à la fois pour l'addition et le doublement. Le principe est de modifier algébriquement le calcul de la pente λ afin qu'il soit identique pour les deux opérations. Brier et Joye donnent les formules suivantes en coordonnées projectives homogènes dans le cas où la courbe elliptique est définie sur un corps de grande caractéristique. La somme de deux points $P_1 = (X_1, Y_1, Z_1)$ et $P_2 = (X_2, Y_2, Z_2)$ tels que $P_1, P_2 \neq \mathcal{O}$ et $P_1 \neq -P_2$ est le point $P_3 = (X_3, Y_3, Z_3)$ défini ainsi :

$$U_1 = X_1Z_2, U_2 = X_2Z_1, S_1 = Y_1Z_2, S_2 = Y_2Z_1, Z = Z_1Z_2, T = U_1 + U_2,$$
$$M = S_1 + S_2, R = T^2 - U_1U_2 + aZ^2, F = ZM, L = MF, G = TL, W = R^2 - G,$$
$$\begin{cases} X_3 &= 2FW, \\ Y_3 &= R(G - 2W) - L^2, \\ Z_3 &= 2F^3. \end{cases}$$

Cette formule unifiée coûte $13M + 5S$. On rappelle que a est un paramètre de la courbe elliptique sous forme de Weierstrass simplifiée (5.2). Les formules de Brier et Joye ne sont valides que lorsque $y_1 + y_2 \neq 0$ ce qui est toujours le cas lors d'un doublement mais pas forcément vrai pour toutes les additions entre deux points. Izu et Takagi [IT03b] proposent une attaque sur les formules unifiées de Brier et Joye basée sur ce cas particulier, lorsqu'on additionne deux points tels que $y_1 + y_2 = 0$. Walter [Wal04] propose aussi une attaque contre ces formules dans le cas où la multiplication de Montgomery est mal utilisée pour les calculs sur le corps de base. Il utilise le fait que certaines implémentations de cette multiplication ont une soustraction conditionnelle finale. Une implémentation correcte de la multiplication de Montgomery peut être trouvée dans [HQ00]. Une dernière attaque est possible sur ces formules en utilisant le fait que lorsque $P_1 = P_2$ certaines multiplications deviennent des élévations au carrés. Amiel et al. [AFT$^+$09] montrent qu'il est possible de distinguer, à partir de mesures de canaux cachés, une addition unifiée avec $P_1 \neq P_2$ en entrée d'une addition avec $P_1 = P_2$.

Afin d'éviter l'attaque de Izu et Takagi [IT03b], Brier et al. [BDJ04] proposent une autre formule d'addition unifiée sans cas particuliers. La somme de deux points $P_3 = P_1 + P_2$ en coordonnées projectives homogènes est définie ainsi :

$$U_1 = X_1Z_2, U_2 = X_2Z_1, S_1 = Y_1Z_2, S_2 = Y_2Z_1, Z = Z_1Z_2, T = U_1 + U_2,$$

$$M = S_1 + S_2, V = (-1)^\delta(U_1 - U_2), N = (-1)^\delta(S_1 - S_2), E = M + V, F = ZE,$$
$$G = LT, R = T^2 - U_1U_2 + Z(aZ + N), W = R^2 - G,$$
$$\begin{cases} X_3 &= 2FW, \\ Y_3 &= R(G - 2W) - LFM, \\ Z_3 &= 2F^3. \end{cases}$$

L'addition unifiée coûte alors $16M + 3S$. On définit δ tel que $\delta = 0$ si $S_1 + S_2 + U_1 - U_2 \neq 0$ et $\delta = 1$ dans le cas contraire. L'attaque de Walter sur les précédentes formules unifiées a été étendue à celles-ci par Stebila et Thériault [ST06]. Alors que dans son attaque Walter détecte la présence d'une soustraction conditionnelle finale dans la multiplication de Montgomery, Stebila et Thériault détectent une addition conditionnelle finale dans une soustraction dans le corps fini. En effet, soient $a, b, c \in \mathbb{F}_q$, si $b > a$ alors la soustraction $c = a - b$ produit un résultat négatif et l'addition $c = c + q$ est effectuée. Cette addition n'est pas calculée lorsque $b \leq a$. Dans ces formules unifiées, Stebila et Thériault remarquent que lorsque $P_1 = P_2$, des soustractions modulaires sont calculées telles que $a = b$, donc sans addition conditionnelle. Si on observe cette addition conditionnelle, cela signifie que l'opération est une addition de points avec $P_1 \neq P_2$. Cette attaque peut être évitée en utilisant des algorithmes d'addition et soustraction modulaire sans condition finale. En plus de l'attaque de Stebila et Thériault, ces formules unifiées sont aussi sensibles à l'attaque de Amiel et al. [AFT+09]. Une contre-mesure consiste à modifier aléatoirement le scalaire avant chaque multiplication scalaire (voir section 8.2).

8.1.1.2 Familles de courbes spécifiques

Nous avons vu précédemment que dans le cas des courbes elliptiques de Weierstrass avoir une formule d'addition et doublement unifiée n'est pas trivial. Il existe néanmoins des familles de courbes elliptiques qui possèdent intrinsèquement une formule unifiée. Les courbes standardisées [X9.98, SEC00, Nat00] sont des courbes de Weierstrass standard n'appartenant à aucunes de ces familles spécifiques présentées ci-après. Ainsi leur application pratique dans le milieu industriel est très limitée malgré leurs avantages. Nous étudions les paramétrisations de courbes définies sur des corps de grande caractéristique $\mathbb{F}_q$. Pour le cas $\mathbb{F}_{2^m}$, le lecteur intéressé peut se reporter aux articles respectifs des auteurs.

Courbe Hessiennes. Une courbe elliptique possédant un point d'ordre 3 peut être exprimée sous forme Hessienne de manière projective [JQ01] :

$$H_d : X^3 + Y^3 + Z^3 = dXYZ,$$

avec $d \in \mathbb{F}_q$ et $d^3 \neq 27$. Soit deux points $P_1 = (X_1, Y_1, Z_1)$ et $P_2 = (X_2, Y_2, Z_2)$ d'une courbe Hessienne H_d. L'opposé de P_1 est $-P_1 = (Y_1, X_1, Z_1)$. La somme $P_3 = (X_3, Y_3, Z_3)$

de P_1 et P_2 est donnée par les formules suivantes :

$$\begin{cases} X_3 &= X_2Y_1Z_2 - X_1Y_2^2Z_1, \\ Y_3 &= X_1^2Y_2Z_2 - X_2^2Y_1Z_1, \\ Z_3 &= X_2Y_2Z_1^2 - X_1Y_1Z_2^2. \end{cases}$$

La formule pour le doublement de points peut être réécrite en utilisant une addition de points de la manière suivante :

$$[2]P_1 = [2](X_1, Y_1, Z_1) = (Z_1, X_1, Y_1) + (Y_1, Z_1, X_1).$$

On obtient donc une formule unifiée pour les courbes Hessiennes de complexité $12M$. Farashahi et Joye [FJ10] présentent une généralisation des courbes Hessiennes en considérant les courbes définies ainsi de manière projective :

$$H_{c,d} : X^3 + Y^3 + cZ^3 = dXYZ,$$

avec $c, d \in \mathbb{F}_q$ tels que $d^3 \neq 27c$ et c ne soit pas un cube dans $\mathbb{F}_q$. Cette dernière condition sur c permet d'avoir une formule unifiée complète, i.e. une formule d'addition qui fonctionne quelque soient les points en entrée. L'addition des points P_1 et P_2 est donnée par les formules suivantes :

$$A = X_1X_2, B = Y_1Y_2, C = cZ_1Z_2, D = X_1Z_2, E = Y_1X_2, F = Z_1Y_2,$$

$$\begin{cases} X_3 &= CF - AE, \\ Y_3 &= BE - CD, \\ Z_3 &= AD - BF. \end{cases}$$

La complexité est de $12M$ plus une multiplication par la constante c.

Courbes Jacobiennes. La famille de courbes Jacobiennes a été introduite par Liardet et Smart [LS01]. Ils montrent qu'une courbe elliptique sous forme de Weierstrass possédant trois points d'ordre 2 peut être isomorphe à une courbe Jacobienne. Cela implique que la courbe elliptique doit avoir un cardinal du groupe de points multiple de 4. Les auteurs donnent une formule d'addition et doublement de points unifiée de complexité $13M + 2S$ plus une multiplication par une constante. Cette approche a été généralisée et améliorée par Billet et Joye [BJ03]. Ils montrent qu'une courbe elliptique sous forme de Weierstrass possédant un seul point d'ordre 2 peut être isomorphe à une courbe Jacobienne d'équation projective :

$$J : Y^2 = \epsilon X^4 - 2\delta X^2Z^2 + Z^4.$$

L'addition unifiée de Billet et Joye pour les points P_1 et P_2 en coordonnées projectives homogènes est donnée par :

$$\begin{cases} X_3 &= X_1Z_1Y_2 + Y_1X_2Z_2, \\ Y_3 &= ((Z_1Z_2)^2 + \epsilon(X_1X_2)^2)(Y_1Y_2 - 2\delta X_1X_2Z_1Z_2) + 2\epsilon X_1X_2Z_1Z_2(X_1^2Z_2^2 + Z_1^2X_2^2), \\ Z_3 &= (Z_1Z_2)^2 - \epsilon(X_1X_2)^2. \end{cases}$$

Cette formule unifiée a une complexité de $10M + 3S$ plus trois multiplications par des constantes. Les auteurs montrent que leur modèle permet de retrouver les courbes proposées par Liardet et Smart tout en obtenant une complexité meilleure : $10M + 3S$ plus une multiplication par une constante. Un autre avantage de la proposition de Billet et Joye est que leurs complexités sont obtenues en considérant un système de coordonnées projectives (X, Y, Z) alors que Liardet et Smart considèrent le quadruplet (X, Y, Z, T) pour obtenir leur formulation. Duquesne [Duq07] propose une amélioration de la formule d'addition de Billet et Joye en considérant un nouveau système de coordonnées (X^2, XZ, Z^2, Y). L'auteur obtient une complexité de $9M + 2S$ plus trois multiplications par des constantes. Néanmoins l'addition n'est plus complète avec ces coordonnées. D'autres travaux ont été réalisés sur les courbes Jacobiennes afin d'améliorer la complexité de l'addition de points et du doublement indépendamment [HCD07, HWCD09b, HWCD09a].

Courbes d'Edwards. Edwards a récemment proposé [Edw07] des courbes elliptiques définies sur $\mathbb{F}_q$. Bernstein et Lange [BL08] étendent la notion de courbes d'Edwards aux courbes de la forme :

$$(X^2 + Y^2)Z^2 = c^2(Z^4 + dX^2Y^2),$$

avec $cd(1-dc^4) \neq 0$. Bernstein et Lange montrent qu'une courbe de Weierstrass qui possède un point d'ordre 4 est isomorphe à une courbe d'Edwards. Cette famille de courbes a une formule d'addition unifiée et complète si d n'est pas un carré dans $\mathbb{F}_q$. Outre cette addition unifiée, le grand intérêt des courbes d'Edwards par rapport aux autre familles est que la complexité de l'addition est très faible. L'addition unifiée de deux points P_1 et P_2 en coordonnées projectives homogènes est donnée par les formules suivantes :

$$A = Z_1Z_2, B = A^2, C = X_1X_2, D = Y_1Y_2, E = dCD, F = B - E, G = B + E,$$

$$\begin{cases} X_3 &= AF((X_1 + Y_1)(X_2 + Y_2) - C - D), \\ Y_3 &= AG(D - C), \\ Z_3 &= cFG. \end{cases}$$

La complexité de ces formules est de $10M+1S$ plus deux multiplications par des constantes. Dans [BL07b], Bernstein et Lange proposent un système de coordonnées « inversées » pour les courbes d'Edwards. Soit un point représenté par (X, Y, Z) en coordonnées projectives homogènes, sa représentation en coordonnées inversées est (YZ, XZ, XY). Les auteurs obtiennent une addition unifiée de complexité $9M + 1S$ plus une multiplication par une constante. Néanmoins la formule d'addition n'est plus complète en utilisant ces coordonnées. Les courbes d'Edwards tordues sont introduites comme une généralisation par Bernstein et al. dans [BBJ+08]. Les auteurs obtiennent une addition unifiée en $10M + 1S$ en coordonnées projectives homogènes et en $9M+1S$ avec les coordonnées inversées plus deux multiplications par des constantes dans ces deux systèmes. Hisil et al. [HWCD08] proposent un système de coordonnées étendues sur les courbes d'Edwards tordues. Soit (X, Y, Z) des coordonnées projectives homogènes, les coordonnées étendues sont (XZ, YZ, XY, Z^2). Hisil et al. obtiennent une addition unifiée en $9M$ plus deux multiplications par des constantes.

Une base de données des meilleures complexités pour les algorithmes d'additions et doublements de points dans différents systèmes de coordonnées et pour différentes familles de courbes est disponible [BL07a].

Courbes de Huff. Plus récemment, Joye et al. [JTV10] revisitent la famille de courbes elliptiques de Huff. Soit $\mathbb{F}_q$ un corps fini de grande caractéristique. Une courbe elliptique de Huff a pour équation projective :

$$aX(Y^2 - Z^2) = bY(X^2 - Z^2),$$

avec $a, b \in \mathbb{F}_q$ et $a^2 \neq b^2$. Ces courbes ont un sous-groupe de points isomorphe à $\mathbb{Z}/4\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. Joye et al. proposent une formule d'addition unifiée et complète sur ces courbes. Soit P_1 et P_2 deux points en coordonnées projectives homogènes tel que leur somme donne le point P_3 grâce aux formules suivantes :

$$A = X_1X_2, B = Y_1Y_2, C = Z_1Z_2, D = (X_1 + Z_1)(X_2 + Z_2) - A - C,$$

$$E = (Y_1 + Z_1)(Y_2 + Z_2) - B - C, F = (B + C)(C - A), G = (A + C)(C - B),$$

$$H = D(B + C), I = E(A + C),$$

$$\begin{cases} X_3 &= HF, \\ Y_3 &= IG, \\ Z_3 &= FG. \end{cases}$$

Cette formule a une complexité de $12M$. Les mêmes auteurs présentent une généralisation du modèle de Huff [JTV10, Section 3.3].

8.1.2 Algorithmes de multiplication scalaire réguliers

Une autre méthode pour se protéger contre les attaques par analyse simple consiste à rendre l'algorithme de multiplication scalaire régulier, i.e. les opérations exécutées sont les mêmes quelque soit le bit de scalaire traité. Dans ce cas, les opérations d'additions et doublements ne sont pas modifiées, on s'intéresse plutôt à ordonnancer l'algorithme de multiplication afin qu'aucune information sur le scalaire ne soit visible.

8.1.2.1 Doublement-et-toujours-addition

Avec les différentes formules d'addition et doublement vues précédemment, un attaquant peut détecter la valeur des bits du scalaire facilement grâce à une attaque par analyse simple [Cor99]. Les courbes de consommation d'une addition et d'un doublement sont assez différentes pour être distinguées. Coron [Cor99] propose donc l'ajout d'une addition factice dans l'algorithme de multiplication. Ainsi, peu importe la valeur du bit du scalaire, les mêmes opérations sont effectuées. Cet algorithme s'appelle doublement-et-toujours-addition, ou *double-and-always-add* (Algorithme 7).

Algorithme 7: Doublement-et-toujours-addition

Entrées : $P \in E$ et $k = (k_{n-1} \dots k_1 k_0)_2$
Sorties : $[k]P \in E$

1 $P_0 \leftarrow \mathcal{O}$
2 $P_1 \leftarrow P$
3 **pour** $i \leftarrow n-1$ **a** 0 **faire**
4 $\quad P_0 \leftarrow [2]P_0$
5 $\quad P_1 \leftarrow P_0 + P$
6 $\quad P_0 \leftarrow P_{k_i}$ `/* ` P_{k_i} `est égal à soit` P_0`, soit` P_1 `*/`
7 **retourner** P_0

Cette contre-mesure est la première à avoir été proposée contre les attaques par analyse simple. Elle possède cependant quelques inconvénients majeurs. Soit k un scalaire de n bits. La complexité de cet algorithme est de $nD + nA$ où D est le coût d'un doublement de point et A celui d'une addition de points. On rappelle que la complexité d'un algorithme de multiplication classique (Algorithme 2) est en moyenne de $nD + \frac{n}{2}A$. De plus l'algorithme doublement-et-toujours-addition est sensible à l'attaque du doublement présentée dans la section 7.2. Une contre-mesure possible à cette attaque consiste à randomiser la valeur du scalaire k ou du point de départ P. Une attaque par injection de fautes sans conséquence sur les calculs (voir section 7.1.1) est aussi possible. Étant donné que cet algorithme ajoute une opération d'addition de points inutile dans le cas où le bit du scalaire vaut 0, un attaquant peut injecter une faute dans cette addition et obtenir à la fin de la multiplication le résultat correct. Il en déduit donc la valeur du bit du scalaire. Une contre-mesure possible consiste à randomiser le scalaire (voir section 8.2.1).

L'algorithme de doublement-et-toujours-addition possède donc beaucoup d'inconvénients en termes de performance mais aussi de sécurité.

8.1.2.2 Atomicité

Le principe de protection de l'algorithme doublement-et-toujours-addition est d'ajouter une opération factice afin d'enlever le branchement conditionnel. Ce principe a été étendu par Chevallier-Mames et al. [CMCJ04] avec le principe d'atomicité face aux canaux cachés.

Cet algorithme est plus efficace que le doublement-et-toujours-addition dans le cas où il existe une formule d'addition et doublement de points indistinguable (voir section 8.1.1). En effet, en moyenne cet algorithme a une complexité de $nD + \frac{n}{2}A$, donc équivalente à celle du doublement-et-addition (Algorithme 2). L'attaquant peut néanmoins identifier le poids de Hamming du scalaire en observant le nombre d'additions calculées par l'algorithme doublement-et-addition atomique. Si ce poids de Hamming est très grand ou très petit, un attaquant peut retrouver le scalaire [CKQ03]. De plus, l'attaque du doublement (voir section 7.2) est toujours possible sur cet algorithme mais, comme précisé précédemment, la randomisation du scalaire suffit pour s'en prémunir.

Algorithme 8: Doublement-et-addition atomique

Entrées : $P \in E$ et $k = (k_{n-1} \dots k_1 k_0)_2$
Sorties : $[k]P \in E$

$P_0 \leftarrow \mathcal{O}$
$P_1 \leftarrow P$
$i \leftarrow n - 1$
$b \leftarrow 0$
tant que $i \geq 0$ **faire**
    $P_0 \leftarrow P_0 + P_b$
    $b \leftarrow b \oplus k_i$
    $i \leftarrow i + k_i - 1$
retourner P_0

L'atomicité présentée ci-dessus, au niveau de l'algorithme de multiplication n'est applicable que si l'on dispose d'une formule d'addition unifiée efficace, ce qui n'est pas le cas pour les courbes sous forme de Weierstrass standards. Dans [CMCJ04], Chevallier-Mames et al. proposent aussi l'atomicité au niveau des opérations d'additions et doublements de points afin de les rendre indistinguables. Le principe est de définir un motif d'opérations que l'on répète pour réaliser une addition et un doublement de telle sorte que ces opérations consistent en une suite de motifs dits atomiques. Ainsi dans [CMCJ04], les auteurs proposent dans le cas de courbes elliptiques définies sur $\mathbb{F}_q$ le motif suivant :

$$\begin{cases} R_1 \leftarrow R_2.R_3 \\ R_4 \leftarrow R_5 + R_6 \\ R_7 \leftarrow -R_8 \\ R_9 \leftarrow R_{10} + R_{11} \end{cases}$$

soit une multiplication, deux additions et une négation dans le corps de base. Chevallier-Mames et al. donnent ensuite une réécriture de l'addition et du doublement de points uniquement à l'aide de ce motif. L'utilisation de ce motif entraîne l'ajout d'additions et négations factices dans le corps de base. Une injection de faute sans conséquence sur les calculs (voir section 7.1.1) est donc applicable même si elle demande beaucoup plus de précision que dans le cas de l'algorithme doublement-et-toujours-addition. En effet, l'attaquant essaye ici d'injecter une faute dans une opération sur le corps de base et non dans une opération sur les points. Le motif d'atomicité suppose aussi que les élévations au carré sont indistinguables de multiplications ce qui n'est pas toujours vrai comme le montre l'attaque détaillée dans [AFT+09]. On peut noter que des améliorations sur le choix de ce motif d'opérations ont été proposées par Longa [Lon07] et Giraud et Verneuil [GV10].

8.1.2.3 Échelle de Montgomery

Montgomery [Mon87] a originellement proposé l'algorithme échelle de Montgomery, ou *Montgomery ladder* (Algorithme 9), pour les courbes elliptiques de la forme de Montgomery. Brier et Joye [BJ02] ont ensuite généralisé l'idée aux courbes sous forme de Weierstrass dans les corps de grande caractéristique $\mathbb{F}_q$. L'idée de Montgomery est d'utiliser le fait que la somme de deux points, dont la différence est un point connu, peut être calculée sans la coordonnée y des deux points. Pour les courbes de Montgomery, il obtient une complexité de $4M + 2S$ pour l'addition et $2M + 2S$ pour le doublement. L'adaptation de Brier et Joye requiert $9M + 2S$ pour une addition et $6M + 3S$ pour un doublement. Pour les courbes standards, la complexité totale de l'algorithme de multiplication scalaire est $n(15M + 5S) + 3M + S + I$ pour un scalaire de n bits, où I est le coût d'une inversion dans $\mathbb{F}_q$ et $3M + S + I$ est le coût pour retrouver la coordonnée y à la fin de l'algorithme. Au même moment, Izu et Takagi [IT02] ont la même idée et obtiennent une complexité légèrement meilleure : $n(13M + 4S) + 11M + 2S$. Si on néglige le coût des multiplications par les paramètres de la courbe, Goundar et al. [GJM10] en réécrivant les formules obtiennent une complexité de $9M + 7S$ par bit de scalaire. On considère donc cette dernière complexité comme celle de l'échelle de Montgomery sans calcul de coordonnée y pour les courbes de Weierstrass. Dans le cas de $\mathbb{F}_{2^m}$, López et Dahab [LD99a] ont aussi généralisé l'idée de Montgomery avec un algorithme de complexité $n(6M + 5S) + 1I + 10M + 1S$.

Algorithme 9: Échelle de Montgomery

Entrées : $P \in E$ et $k = (k_{n-1} \dots k_1 k_0)_2$
Sorties : $[k]P \in E$

1 $P_0 \leftarrow \mathcal{O}$
2 $P_1 \leftarrow P$
3 **pour** $i \leftarrow n-1$ **a** 0 **faire**
4 $\quad P_{\bar{k_i}} \leftarrow P_{\bar{k_i}} + P_{k_i}$
5 $\quad P_{k_i} \leftarrow [2]P_{k_i}$
6 **retourner** P_0

Contrairement aux solutions précédentes, dans le calcul de l'échelle de Montgomery tous les résultats intermédiaires sont utilisés. Ainsi, peu importe la valeur du bit et l'opération à l'intérieur de la boucle qui est visée, l'injection d'une faute au cours du calcul entraîne à chaque fois un résultat faux, inutilisable par un attaquant. L'algorithme est néanmoins sensible à l'attaque du doublement (voir section 7.2) mais qui peut être contrecarrée en randomisant le scalaire ou les points intermédiaires (voir section 8.2).

8.1.2.4 Doublement-et-addition de Joye

Il existe une variante de l'Algorithme 2 où on parcourt les bits du scalaire dans l'autre sens. Cet algorithme est appelé *right-to-left* doublement-et-addition (Algorithme 6). Il

souffre des mêmes vulnérabilités que sa version *left-to-right*.

On peut d'ailleurs facilement créer l'équivalent *right-to-left* doublement-et-toujours-addition qui est, comme son symétrique, vulnérable aux injections de fautes sans conséquence sur les calculs. Joye [Joy07] applique le principe de l'échelle de Montgomery dans le sens *right-to-left* et propose un algorithme appelé doublement-et-addition de Joye, ou *Joye's double-add* (Algorithme 10). Cet algorithme bénéficie des mêmes propriétés de résistance face aux attaques par canaux cachés que l'échelle de Montgomery.

Algorithme 10: Doublement-et-addition de Joye

Entrées : $P \in E$ et $k = (k_{n-1} \dots k_1 k_0)_2$
Sorties : $[k]P \in E$

1 $P_0 \leftarrow \mathcal{O}$
2 $P_1 \leftarrow P$
3 **pour** $i \leftarrow 0$ **a** $n-1$ **faire**
4 $\quad P_{\bar{k_i}} \leftarrow [2]P_{\bar{k_i}}$
5 $\quad P_{\bar{k_i}} \leftarrow P_{\bar{k_i}} + P_{k_i}$
6 **retourner** P_0

Comme les algorithmes de Montgomery et Joye sont, par construction, intéressants du point du vue résistance aux attaques par canaux cachés, on utilise la même structure d'algorithme hautement régulier pour notre proposition dans le chapitre 9. Sauf mention contraire par la suite, les observations qui sont faites sur l'échelle de Montgomery s'appliquent de manière symétrique au doublement-et-addition de Joye.

8.2 Face à une attaque par analyse différentielle

L'utilisation d'un algorithme de multiplication scalaire résistant face aux attaques simples ne suffit pas à se protéger des attaques différentielles. Il faut donc prendre des mesures spécifiques pour contrecarrer ce type d'attaques. Les contre-mesures face aux attaques différentielles peuvent se classer en trois catégories. On peut ajouter de l'aléa au niveau du scalaire k, ou utiliser les propriétés mathématiques des courbes elliptiques pour randomiser la représentation du point P, ou encore modifier les paramètres de la courbe afin que les calculs se fassent sur une courbe différente. Le but est bien entendu d'ajouter de l'aléa dans les calculs intermédiaires de $[k]P$ tout en obtenant le résultat correct de la multiplication.

8.2.1 Modifier le scalaire

Soit E une courbe elliptique définie sur un corps fini K. Soit $P \in E(K)$ tel que $[n]P = \mathcal{O}$ où n est l'ordre de P sur la courbe. On sait que le scalaire $k \in \mathbb{Z}_n$.

8.2.1.1 Randomisation du scalaire

Le principe est d'ajouter une valeur r aléatoire telle que $k' = k + rn$. La multiplication scalaire de P s'écrit alors :

$$[k']P = [k + rn]P = [k]P + [rn]P = [k]P.$$

Au lieu de calculer $[k]P$, on peut désormais calculer de manière équivalente $[k']P$. Nous avons vu précédemment que les algorithmes de multiplication scalaire effectuent des opérations suivant la valeur des bits du scalaire. On obtient donc grâce à cette technique des valeurs de scalaires différentes pour un même résultat de multiplication. Cette contre-mesure a été proposée par Coron [Cor99] dans le cadre des courbes elliptiques.

Elle possède toutefois quelques inconvénients. En ajoutant la valeur rn au scalaire k de base, on augmente sa taille et donc on augmente aussi la complexité de l'algorithme de multiplication. Cette contre-mesure peut aussi entraîner une faille de sécurité lors de son utilisation avec des courbes elliptiques standardisées [X9.98, SEC00, Nat00]. En effet, pour des besoins d'efficacité ces courbes sont définies sur des corps finis $\mathbb{F}_q$ avec q un nombre de Mersenne généralisé de la forme $2^m \pm 2^n \pm 1$. D'après le théorème de Hasse (Théorème 5.5.1), le cardinal du groupe de points de la courbe $E(\mathbb{F}_q)$ a une valeur assez proche de q. Ainsi, ce cardinal contient souvent des plages de zéros importantes dans sa représentation binaire. Le calcul $k' = k + r\#E(\mathbb{F}_q)$ risque alors de ne pas masquer une partie des bits de k. Ciet a observé ce problème dans [Cie03]. Une autre attaque proposée par Fouque et al. [FRVD08] contrecarre cette technique dans le cas où l'attaquant est capable d'observer une fuite d'information sur la présence éventuelle de retenues lors de l'addition $k + rn$.

8.2.1.2 Découpage du scalaire

Pour masquer l'opération $[k]P$, on considère désormais le découpage du scalaire k de manière aléatoire. Une première proposition de découpage est proposée par Clavier et Joye [CJ01] :

$$[k]P = [k - r]P + [r]P.$$

Cette méthode est aussi appelée découpage additif du scalaire. Le principe est donc de calculer deux multiplications avec chacune des scalaires contenant de l'aléa. Pour cacher l'intégralité de k, le nombre aléatoire r doit être de même taille. Ceci double au total la complexité de la multiplication puisque $[k - r]P$ et $[r]P$ doivent être calculés.

On peut aussi découper le scalaire de manière multiplicative comme proposé par Trichina et Belleza [TB02]. La multiplication est alors calculée ainsi :

$$[k]P = [kr^{-1}]([r]P).$$

Cette méthode nécessite le calcul de l'inverse modulaire dans $\mathbb{F}_q$ du nombre aléatoire r ce qui est relativement coûteux.

Une dernière technique est proposée par Ciet et Joye [CJ03] et est appelée découpage Euclidien. Elle consiste à calculer :

$$[k]P = [k \bmod r]P + [\lfloor k/r \rfloor]([r]P)$$

Les auteurs suggèrent l'utilisation de la multiplication de points simultanée, appelée méthode de Shamir-Strauss [ElG85], pour calculer les multiplications scalaires intermédiaires.

8.2.2 Modifier le point de base

Ces contre-mesures qui agissent sur le point P utilisé dans la multiplication $[k]P$ utilisent principalement les propriétés mathématiques des courbes elliptiques.

8.2.2.1 Aveuglement du point de base

Cette première contre-mesure, initialement proposée pour RSA, a été adaptée aux courbes elliptiques par Coron dans [Cor99]. Soit R un point de la courbe elliptique tiré au hasard et gardé secret. La multiplication scalaire devient :

$$[k]P = [k](P + R) - [k]R.$$

On doit donc calculer successivement une addition de points $P + R$, une multiplication scalaire $S = [k]R$ et une dernière addition de points $[k](P + R) - S$. Les points R et S peuvent toutefois être stockés en mémoire et renouvelés assez efficacement en calculant $R \leftarrow [r]R$ et $S \leftarrow [r]S$ pour une valeur aléatoire r petite.

8.2.2.2 Randomisation des coordonnées projectives

Nous avons vu à la section 5.8 que les coordonnées projectives permettent un gain de performance intéressant en évitant de calculer des inversions modulaires. Ce système de coordonnées est aussi particulièrement intéressant pour randomiser facilement la représentation d'un point. En effet, dans ce système, chaque coordonnée est en fait un élément d'une classe d'équivalence (voir section 5.7). On peut donc utiliser, sans conséquence sur les calculs, un autre élément de cette classe d'équivalence. Coron propose l'utilisation de cette technique dans [Cor99]. Soit $P = (X, Y, Z)$ un point en coordonnées projectives homogènes. Soit $\lambda \in \mathbb{F}_q \setminus \{0\}$ choisi aléatoirement. Toutes les représentations $(\lambda X, \lambda Y, \lambda Z)$ correspondent au même point P. Si le point $P = (X, Y, Z)$ est représenté en coordonnées projectives jacobiennes, alors ses représentations équivalentes sont $(\lambda^2 X, \lambda^3 Y, \lambda Z)$. Cette méthode coûte peu en complexité, $3M$ dans le cas projectif homogène et $4M + 1S$ dans le cas projectif jacobien. Ainsi avant chaque calcul de $[k]P$, on peut modifier aléatoirement la représentation du point P.

8.2.2.3 Isomorphismes de corps aléatoires

Afin de modifier le point P, on modifie le corps de base $K = \mathbb{F}_q$ avec cette méthode proposée par Joye et Tymen [JT01]. Le principe consiste à construire aléatoirement un isomorphisme de corps $\phi : K \to K'$. Le calcul de la multiplication scalaire devient :

$$[k]P = \phi^{-1}([k](\phi(P))).$$

Le point est donc passé par un isomorphisme aléatoire dans le corps K' où est calculée la multiplication et, finalement, on retourne pas l'isomorphisme inverse dans le corps K. Cette contre-mesure est néanmoins assez complexe à mettre en place car l'isomorphisme ϕ et son inverse doivent être stockés en mémoire. De plus, on doit les randomiser entre chaque multiplication ce qui est relativement coûteux malgré la technique proposée dans [JT01].

8.2.2.4 Isomorphismes de courbes aléatoires

Dans le même article, Joye et Tymen [JT01] proposent cette fois de changer de courbe elliptique à chaque multiplication scalaire grâce à des isomorphismes de courbes aléatoires. La représentation des points sur cette nouvelle courbe est donc différente par rapport à la courbe originale. Les calculs effectués avec ce nouveau point ne donnent ainsi aucune information à l'attaquant. Soit deux courbes elliptiques sous forme de Weierstrass simplifiée $E_1 : y^2 = x^3 + a_1x + b_1$ et $E_2 : y^2 = x^3 + a_2x + b_2$ définies sur le corps fini $K = \mathbb{F}_q$. Ces courbes sont isomorphes si et seulement si il existe $u \in K \setminus \{0\}$ tel que $u^4a_2 = a_1$ et $u^6b_2 = b_1$. L'isomorphisme est donné par :

$$\begin{aligned}\varphi : E_1(K) &\to E_2(K)\\ \begin{cases}\mathcal{O}_{E_1} & \mapsto \mathcal{O}_{E_2}\\ (x,y) & \mapsto (u^{-2}x, u^{-3}y)\end{cases}\end{aligned}$$

l'isomorphisme inverse est donné par :

$$\begin{aligned}\varphi^{-1} : E_2(K) &\to E_1(K)\\ \begin{cases}\mathcal{O}_{E_2} & \mapsto \mathcal{O}_{E_1}\\ (x,y) & \mapsto (u^2x, u^3y).\end{cases}\end{aligned}$$

La multiplication du point $P \in E_1(K)$ par un scalaire k est alors calculée ainsi :

$$[k]P = \varphi^{-1}([k](\varphi(P))).$$

On doit donc d'abord calculer l'image du point P par l'isomorphisme φ sur la courbe E_2. On réalise ensuite la multiplication scalaire du point $\varphi(P)$ par k. On revient finalement sur la courbe E_1 une fois le résultat de la multiplication calculé. On génère donc un isomorphisme de courbe de manière aléatoire en $1I+8M+2S$. Tunstall et Joye proposent une extension de ce principe dans [TJ10]. Si on se place dans le cas de coordonnées projectives jacobiennes, les auteurs définissent une fonction Φ sur les points de la courbe telle que $\Phi(X, Y, Z) = (f^\mu X, f^\upsilon Y, Z)$ avec $f \in K\setminus\{0\}$, μ et υ de petits entiers. Le point $\Phi(X, Y, Z)$ peut ne pas être sur la même courbe que (X, Y, Z). L'inverse de la fonction est calculé par $\Phi^{-1}(X, Y, Z) = (f^{\mu+2\upsilon}X, f^{3\mu+2\upsilon}Y, f^{\mu+\upsilon}Z)$. En fixant $\mu = 2$ et $\upsilon = 3$, on retrouve notamment la contre-mesure proposée par Joye et Tymen [JT01].

8.3 Face à une attaque par injection de fautes

Il est souvent difficile de se prémunir efficacement et sans trop de surcoût contre les attaques par perturbation. Nous étudions dans cette partie des contre-mesures face aux attaques sans conséquences et face aux attaques utilisant des courbes cryptographiquement faibles.

Comme nous l'avons défini dans la section 7.1.1, les attaques sans conséquences peuvent se diviser en deux sous-catégories : les fautes sans conséquence sur la mémoire (*M-safe*) et les fautes sans conséquence sur les calculs (*C-safe*). Les attaques *C-safe* se contre-carrent assez facilement en choisissant un algorithme de multiplication scalaire n'utilisant aucunes opérations factices. Par exemple, au lieu d'utiliser l'algorithme de doublement-et-toujours-addition (Algorithme 7), on préfère l'échelle de Montgomery (Algorithme 9) ou le doublement-et-addition de Joye (Algorithme 10). Une fois un algorithme résistant aux attaques *C-safe* sélectionné, il est facile de le modifier pour le rendre résistant aux attaques *M-safe*. Il suffit de faire en sorte que tous les registres soient utilisés quelque soit la valeur du bit de clé traité. Ainsi, une faute effectuée sur une valeur d'un registre affecte toujours le résultat final. L'Algorithme 9 et l'Algorithme 10 sont écrits pour être résistants aux attaques *M-safe*.

Dans la section 7.1.2, nous avons présenté des attaques par fautes permettant de faire passer le calcul sur une courbe elliptique cryptographiquement faible pour en déduire plus facilement le secret. Une faute peut être injectée sur le point de base P pour l'envoyer sur une courbe elliptique E' plus faible. Dans ce cas, une simple contre-mesure consiste à vérifier que le point est valide grâce à la définition suivante.

Définition 8.3.1. *Un point $P = (x, y)$, associé à un ensemble de paramètres de définition de la courbe elliptique, est dit valide si il vérifie les conditions suivantes :*

1. $P \neq \mathcal{O}$,
2. *les coordonnées de P sont correctement représentées comme des éléments du corps de base,*
3. *P satisfait à l'équation de la courbe,*
4. $[\#E(K)]P = \mathcal{O}$.

Dans le cas où l'attaquant injecte une faute dans les paramètres de la courbe afin de la rendre plus faible, Ciet et Joye [CJ05] proposent d'utiliser une somme de contrôle sur ces paramètres. Comme les sommes de contrôles ne sont applicables qu'à des données déjà stockées en mémoire, elles ne protègent pas des fautes sur des variables intermédiaires.

Fouque et al. [FLRV08] (voir section 7.1.2) présentent une injection de faute qui permet de faire passer le point de base sur la courbe tordue. Or beaucoup de courbes cryptographiquement sûres ont des courbes tordues faibles. Cette attaque est très puissante lorsque qu'on l'applique sur des algorithmes de multiplication scalaire qui calculent sans la coordonnée y des points, comme certaines versions de l'échelle de Montgomery. Les autres algorithmes peuvent facilement prévenir cette attaque en vérifiant que le point appartient à la courbe originale.

L'attaque différentielle de Biehl et al. [BMM00] (voir section 7.1.3) se contrecarre en utilisant une technique de randomisation du scalaire (voir section 8.2.1). Il est aussi utile de tester si le point est valide au sortie de la multiplication. L'attaque par changement de signe de Blömer et al. [BOS06] s'applique sur la version de l'échelle de Montgomery utilisant les coordonnées y, contrairement à l'attaque de Fouque et al. [FLRV08]. Blömer et al. proposent une contre-mesure basée sur une idée de Shamir [Sha99] dans le cas du RSA-CRT. Si on considère une multiplication scalaire sur une courbe définie sur le corps $\mathbb{F}_q$, le principe de la contre-mesure est d'effectuer une deuxième multiplication scalaire mais sur un corps fini $\mathbb{F}_{q_0}$ avec $q_0 < q$. Si q_0 est choisi très petit alors le nombre d'opérations supplémentaires est raisonnable comparé à une simple duplication des calculs très coûteuse. L'utilisation de techniques de randomisation du scalaire (voir section 8.2.1) permet aussi de contrecarrer l'attaque de Blömer et al.

Chapitre 9

Proposition de multiplication scalaire efficace et résistante

Notre objectif est de construire un algorithme de multiplication scalaire pour courbes standards sous forme de Weierstrass qui soit à la fois efficace et résistant aux attaques par canaux cachés. Pour cela, nous combinons la structure hautement régulière de l'échelle de Montgomery (Algorithme 9) avec l'idée d'addition simplifiée de Méloni (voir section 5.8).

9.1 Modification de l'échelle de Montgomery

Afin de tirer partie de l'efficacité de l'addition simplifiée, notée ZADDU [GJM10], on réécrit l'algorithme de Montgomery pour n'utiliser que des additions de points (Algorithme 11).

Algorithme 11: Échelle de Montgomery avec additions

Entrées : $P \in E$ et $k = (k_{n-1} \dots k_1 k_0)_2$
Sorties : $[k]P \in E$

1 $P_0 \leftarrow \mathcal{O}$
2 $P_1 \leftarrow P$
3 **pour** $i \leftarrow n-1$ **a** 0 **faire**
4 $\quad P_0 \leftarrow P_0 + P_1$
5 $\quad P_1 \leftarrow P_0 + (-1)^{\bar{k}_i} P$
6 **retourner** P_1

Toutefois, l'addition simplifiée requiert que les deux points en entrée aient la même coordonnée Z ce qui n'est pas le cas dans l'algorithme actuel. Il faut que $Z_{P_0} = Z_{P_1}$ à la fin de chaque tour de boucle pour pouvoir les additionner au tour suivant. Heureusement, il s'agit d'une propriété de l'algorithme ZADDU. Nous avons aussi besoin que le point $\pm P$

(Algorithme 11 Ligne 5) ait le même Z que P_0. Une solution naïve est de recalculer, à chaque tour, un point P tel que $Z_P = Z_{P_0}$:

1. On stocke le point $P = (X_P, Y_P, Z_P)$ durant la multiplication.
2. On calcule et on stocke l'inverse Z^{-1} au début de l'algorithme.
3. Soit $P = (X_P, Y_P, Z_P)$ et $P_0 = (X_0, Y_0, Z_0)$. On calcule, à chaque tour, un entier $\lambda = Z_0 Z_P^{-1}$. Finalement, on obtient $P' = \pm(\lambda X_P, \lambda Y_P, \lambda Z_P)$ pour une complexité totale de $4M$ par tour. On appelle ce recalcul naïf de P, Z_P *naive update* (ZPNU).

Par la suite, sauf mention contraire, on considère la complexité des algorithmes de multiplication scalaire en ne notant que la complexité d'un tour de boucle. Dans le cas $\mathbb{F}_q$, la complexité de l'Algorithme 11 n'utilisant que des additions simplifiées et le recalcul naïf de P est : $2(5M + 2S) + 4S = 14M + 4S$. Dans le cas $\mathbb{F}_{2^m}$, on obtient une complexité de : $2(7M + 2S) + 4M = 18M + 4S$. A ces coûts, il faut ajouter une inversion modulaire calculée en début d'algorithme à cause de ZPNU.

On peut recalculer un point P de manière plus efficace en modifiant l'algorithme d'addition simplifiée pour y ajouter une soustraction.

9.2 Formules d'addition-soustraction simplifiées

Basé sur l'addition de Méloni, nous proposons des algorithmes d'addition-soustraction simplifiés. Dans le cas $\mathbb{F}_q$, nous avons publié cette formule [VD10b, VD10a] de manière parallèle et indépendante à Goundar et al. [GJM10]. Alors que nous appelons cet algorithme NewAddSub, Goundar et al. le nomme *conjugate co-Z addition* (ZADDC). Nous étendons aussi les formules au cas $\mathbb{F}_{2^m}$ [VD10c]. Nous utilisons par la suite la notation plus concise ZADDC. Le principe de ZADDC est de calculer :

$$\text{ZADDC}(P_1, P_2) \rightarrow (\tilde{P_1}, P_1 + P_2, P_1 - P_2) \text{ avec } Z_{\tilde{P_1}} = Z_{P_1+P_2} = Z_{P_1-P_2}.$$

Soit $P_1 = (X_1, Y_1, Z)$, $P_2 = (X_2, Y_2, Z)$ tous deux différents de $\mathcal{O}$ et $P_2 \neq \pm P_1$. Soit $P_3 = P_1 + P_2 = (X_3, Y_3, Z_3)$, $P_4 = P1 - P2 = (X_4, Y_4, Z_4)$ et $\tilde{P_1} = (\tilde{X_1}, \tilde{Y_1}, \tilde{Z_1})$.

ZADDC dans $\mathbb{F}_q$.

$$\begin{array}{lll} A = (X_2 - X_1)^2, & B = X_1 A, & C = X_2 A, \\ D = (Y_2 - Y_1)^2, & E = (-Y_1 - Y_2)^2, & F = Y_1(C - B). \end{array}$$

$$\begin{array}{lll} \tilde{X_1} = B, & \tilde{Y_1} = F, & \tilde{Z_1} = Z(X_2 - X_1), \\ X_3 = D - B - C, & Y_3 = (Y_2 - Y_1)(B - X_3) - F, & Z_3 = Z(X_2 - X_1), \\ X_4 = E - B - C, & Y_4 = (B - X_4)(-Y_1 - Y_2) - F, & Z_4 = Z(X_2 - X_1). \end{array}$$

	$\mathbb{F}_q$	$\mathbb{F}_{2^m}$
ZADDU	$5M + 2S$	$7M + 2S$
ZADDC	$6M + 3S$	$11M + 2S$

TABLE 9.1 – Complexités en opérations dans le corps de base des formules d'additions simplifiées.

ZADDC dans $\mathbb{F}_{2^m}$.

$$\begin{aligned} A &= X_1 + X_2, & B &= A^2, & C &= AB, & D &= Y_1 + Y_2, \\ D' &= ZX_1 + D, & E &= CY_2, & F &= BX_2, & G &= D + Z_3, \\ G' &= D' + Z_3, & H &= FD, & H' &= FD', & I &= aZ_3^2. \end{aligned}$$

$$\begin{aligned} \tilde{X}_1 &= F, & \tilde{Y}_1 &= E, & \tilde{Z}_1 &= ZA, \\ X_3 &= I + DG + C, & Y_3 &= X_3G + E + H, & Z_3 &= ZA, \\ X_4 &= I + D'G' + C, & Y_4 &= X_4G' + E + H', & Z_4 &= ZA. \end{aligned}$$

On rappelle les différentes complexités des algorithmes d'additions simplifiées dans la Table 9.1.

9.3 Nouveaux algorithmes de multiplication scalaire

Nous pouvons désormais proposer un algorithme de multiplication basé sur l'échelle de Montgomery en utilisant les formules ZADDU et ZADDC. L'Algorithme 12 est une version n'utilisant que ZADDC. En l'étudiant plus en détail, on remarque que le point P ayant une coordonnée Z appropriée est calculé à chaque tour Ligne 6. Après le deuxième ZADDC, Ligne 7, on a toujours, si $P_0 = [r]P$, alors $P_1 = [r-1]P$. Ainsi, le tour suivant, on a bien $(P_0 - P_1) = P$. La supposition Ligne 2 s'explique par le fait que le doublement de point initial peut être facilement modifié afin que les points aient la même coordonnée Z [Mel07, GJM10].

La complexité de l'échelle de Montgomery modifiée utilisant uniquement ZADDC, notée ML+2ZADDC, est :

- sur $\mathbb{F}_{2^m}$: $22M + 4S$,
- sur $\mathbb{F}_q$: $12M + 6S$.

En réécrivant l'Algorithme 12 [VD10b, GJM10], on peut noter que le deuxième ZADDC Ligne 7 peut être remplacé par un simple ZADDU. On le note ML+ZADDC+ZADDU. On a les complexités :

- sur $\mathbb{F}_{2^m}$: $18M + 4S$,
- sur $\mathbb{F}_q$: $11M + 5S$.

Algorithme 12: Échelle de Montgomery avec ZADDC

Entrées : $P \in E$ et $k = (k_{n-1} \dots k_1 k_0)_2$
Sorties : $[k]P \in E$

$P_0 \leftarrow P$
$P_1 \leftarrow [2]P$
// On suppose $Z_{P_0} = Z_{P_1}$
pour $i \leftarrow n-2$ **a** 0 **faire**
  $(Q_0, Q_1, Q_2) \leftarrow \texttt{ZADDC}(P_{k_i}, P_{\bar{k}_i})$
  $P_{\bar{k}_i} \leftarrow Q_1$ /* $P_{\bar{k}_i} \leftarrow (P_0 + P_1)$ */
  $P_{k_i} \leftarrow Q_2$ /* $P_{k_i} \leftarrow (P_{k_i} - P_{\bar{k}_i}) = \pm P$ */
  $(Q_0, Q_1, Q_2) \leftarrow \texttt{ZADDC}(P_{\bar{k}_i}, P_{k_i})$
  $P_{\bar{k}_i} \leftarrow Q_0$ /* $P_{\bar{k}_i} \leftarrow \tilde{P}_{\bar{k}_i}$ */
  $P_{k_i} \leftarrow Q_1$ /* $P_{k_i} \leftarrow P_0 + P_1$ */
retourner P_0

Sur $\mathbb{F}_q$, Goundar et al. [GJM10] proposent une formule appelée *co-Z double-add with update* (ZDAU). A partir de deux points P et Q, on a $\text{ZDAU}(P, Q) \rightarrow (2P + Q, \tilde{Q})$ tels que les coordonnées Z des points de sortie soient égales. L'opération ZDAU, qui permet de calculer un tour de boucle de multiplication scalaire, coûte $9M + 7S$ soit un échange entre $2S$ et $2M$ par rapport à précédemment. On note l'algorithme obtenu ML+ZDAU.

Dans [VD10b], nous proposons d'améliorer l'algorithme sur $\mathbb{F}_q$ en remarquant que la coordonnée Z des points dans la boucle n'est pas utilisée pour les calculs des autres coordonnées X et Y. Nous pouvons donc omettre totalement les calculs sur Z, pour ne recalculer le Z du point final, qu'à la fin de l'algorithme. Nous gagnons ainsi $1M$ dans les algorithmes ZADDU et ZADDC que l'on appelle respectivement *co-Z addition with update without Z coordinate* (ZADDUwoZ) et *conjugate co-Z addition without Z coordinate* (ZADDCwoZ). Leurs complexités sur $\mathbb{F}_q$ sont :

- ZADDUwoZ : $4M + 2S$,
- ZADDCwoZ : $5M + 3S$.

Nous devons légèrement modifier l'algorithme de Montgomery afin d'utiliser ces nouvelles formules. Le dernier tour de boucle doit être calculé à part afin de retrouver la coordonnée Z du point final (Algorithme 13). Nous obtenons deux nouveaux algorithmes de multiplication scalaire pour $\mathbb{F}_q$ que l'on note ML+2ZADDCwoZ de complexité $10M+6S$ et ML+ZADDCwoZ+ZADDUwoZ de complexité $9M + 5S$. Nous pouvons justifier le fait d'utiliser les algorithmes ML+2ZADDC ou ML+2ZADDCwoZ, légèrement plus lent, par le fait qu'un seul algorithme d'addition a besoin d'être codé. Sur des environnements embarqués, où les contraintes d'espace mémoire sont strictes, ces algorithmes peuvent être préférés. Cette astuce ne s'applique pas sur $\mathbb{F}_{2^m}$ car la coordonnée Z est nécessaire dans le calcul des autres coordonnées.

Algorithme 13: Échelle de Montgomery avec ZADDCwoZ

Entrées : $P \in E(\mathbb{F}_q)$ et $k = (k_{n-1} \dots k_1 k_0)_2$
Sorties : $[k]P \in E(\mathbb{F}_q)$

$P_0 \leftarrow P$
$P_1 \leftarrow [2]P$
// On suppose $Z_{P_0} = Z_{P_1}$
$P_{save} \leftarrow P$
pour $i \leftarrow n-2$ **a** 1 **faire**
    $(Q_0, Q_1, Q_2) \leftarrow$ ZADDCwoZ$(P_{k_i}, P_{\bar{k}_i})$
    $P_{\bar{k}_i} \leftarrow Q_1$ /* $P_{\bar{k}_i} \leftarrow (P_0 + P_1)$ */
    $P_{k_i} \leftarrow Q_2$ /* $P_{k_i} \leftarrow (P_{k_i} - P_{\bar{k}_i}) = \pm P$ */
    $(Q_0, Q_1, Q_2) \leftarrow$ ZADDCwoZ$(P_{\bar{k}_i}, P_{k_i})$
    $P_{\bar{k}_i} \leftarrow Q_0$ /* $P_{\bar{k}_i} \leftarrow \tilde{P}_{\bar{k}_i}$ */
    $P_{k_i} \leftarrow Q_1$ /* $P_{k_i} \leftarrow P_0 + P_1$ */
// Dernier tour
$(Q_0, Q_1, Q_2) \leftarrow$ ZADDCwoZ$(P_{k_i}, P_{\bar{k}_i})$
$P_{\bar{k}_i} \leftarrow Q_1$ /* $P_{\bar{k}_i} \leftarrow (P_0 + P_1)$ */
$P_{k_i} \leftarrow Q_2$ /* $P_{k_i} \leftarrow (P_{k_i} - P_{\bar{k}_i}) = \pm P$ */
// Calcul de Z_{final}
$Z_{final} \leftarrow X_{P_{k_i}} \cdot Y_{P_{save}}$
$Z_{final} \leftarrow (Z_{final})^{-1}$
$Z_{final} \leftarrow Z_{final} \cdot Y_{P_{k_i}}$
$Z_{final} \leftarrow Z_{final} \cdot X_{P_{save}}$
$Z_{final} \leftarrow Z_{final} \cdot Z_{P_{save}}$
$Z_{final} \leftarrow (Z_{final} \cdot (X_{P_{k_i}} - X_{P_{\bar{k}_i}}))$
$(Q_0, Q_1, Q_2) \leftarrow$ ZADDCwoZ$(P_{\bar{k}_i}, P_{k_i})$
$P_{\bar{k}_i} \leftarrow Q_0$ /* $P_{\bar{k}_i} \leftarrow \tilde{P}_{\bar{k}_i}$ */
$P_{k_i} \leftarrow Q_1$ /* $P_{k_i} \leftarrow P_0 + P_1$ */
$P_0 \leftarrow [X_{P_0}, Y_{P_0}, Z_{final}]$
retourner P_0

9.4 Comparaison de performances

En combinant la structure hautement régulière de l'échelle de Montgomery avec la formule efficace d'addition simplifiée de Méloni, nous avons obtenu différentes propositions d'algorithmes rapides et résistants face aux attaques par canaux cachés. Des propositions d'algorithmes similaires peuvent être construites en se basant sur le doublement-et-addition de Joye [GJM10]. De nombreuses variantes de l'échelle de Montgomery sans calcul de la coordonnée y sont brevetées [VMAG99], notamment la version très efficace sur $\mathbb{F}_{2^m}$. Une alternative consiste à utiliser la structure régulière de l'algorithme de Montgomery mais en utilisant des formules d'additions de doublements de points classiques. On appelle ML Basique cette version générique dans la Table 9.2. Sa complexité est donc la somme de la complexité d'une addition de points en coordonnées jacobiennes avec celle d'un doublement, soit sur $\mathbb{F}_q$:

$$\underbrace{11M + 5S}_{\text{ADD}} + \underbrace{1M + 8S}_{\text{DBL}} = 12M + 13S,$$

et sur $\mathbb{F}_{2^m}$:

$$\underbrace{14M + 5S}_{\text{ADD}} + \underbrace{4M + 5S}_{\text{DBL}} = 18M + 10S.$$

Nous incluons dans la comparaison nos propositions qui utilisent un ou deux algorithmes d'additions. Comme dit précédemment, il n'est pas évident que sur des environnements embarqués un programmeur ait assez d'espace pour coder deux additions. On laisse ainsi le choix d'échanger un gain en performance par un gain en taille de code. Dans $\mathbb{F}_q$, notre proposition ML+ZADDCwoZ+ZADDUwoZ est, à notre connaissance, l'algorithme de multiplication scalaire résistant aux attaques par canaux cachés pour courbes de Weierstrass génériques le plus efficace de la littérature. Dans $\mathbb{F}_{2^m}$, nos deux propositions apportent un léger gain par rapport à la version basique de l'échelle de Montgomery mais elles sont loin en terme de performances de la version ML X-only [LD99a]. Elles ont tout de même l'avantage de proposer une alternative à cet algorithme breveté.

9.5 Évaluation de la résistance aux attaques

Comme nous l'avons vu à la section 8.1, l'échelle de Montgomery est par construction résistante face aux attaques par analyse simple. Nos propositions d'algorithmes ont des structures très similaires à celle de Montgomery. L'algorithme effectue à chaque tour de boucle, quelque soit la valeur du bit de scalaire, une addition et un doublement. L'échelle de Montgomery est aussi efficace contre les attaques par injection de fautes sans conséquences (voir section 7.1.1). En effet, toutes les valeurs intermédiaires sont réutilisées dans le calcul de la multiplication. Une écriture attentive de l'algorithme permet aussi de se défaire des attaques sans conséquence sur la mémoire. D'autres types d'attaques par injection de fautes sont annulés par une procédure de test si le point de la courbe elliptique est valide (voir section 8.3). La résistance aux attaques différentielles se traite indépendamment du choix

$\mathbb{F}_q$	Complexité (pour un bit de scalaire)
ML Basique	$12M + 13S$
ML *X-only* [BJ02, IT02, GJM10]	$9M + 7S$
ML+2ZADDC	$12M + 6S$
ML+ZADDC+ZADDU	$11M + 5S$
ML+2ZADDCwoZ	$10M + 6S$
ML+ZADDCwoZ+ZADDUwoZ	$9M + 5S$
$\mathbb{F}_{2^m}$	Complexité (pour un bit de scalaire)
ML Basique	$18M + 10S$
ML *X-only* [LD99a]	$6M + 5S$
ML+2ZADDC	$22M + 4S$
ML+ZADDC+ZADDU	$18M + 4S$

TABLE 9.2 – Complexités de différents algorithmes de multiplication scalaire résistants aux attaques par canaux cachés.

de l'algorithme de multiplication. Ainsi, des contre-mesures spécifiques (voir section 8.2) doivent être appliquées à notre méthode pour la rendre résistante.

Par construction, nos propositions sont sures face à plusieurs attaques par canaux cachés. L'application de contre-mesures face aux attaques différentielles ainsi que de test si le point est valide permettent d'obtenir un algorithme résistant face aux principales attaques de la littérature.

Troisième partie

Couplages

Chapitre 10

Préliminaires mathématiques

10.1 Points de torsion

Soit E une courbe elliptique définie sur un corps fini K. On appelle ensemble de points de n-torsion les points de $E(K)$ qui, multipliés par l'entier positif n, deviennent le point à l'infini $\mathcal{O}$. On note cet ensemble $E(K)[n]$. Par convention, lorsqu'on considère les points de n-torsion de la courbe E sur la clôture algébrique $\overline{K}$, on écrit $E(\overline{K})[n] = E[n]$. De manière plus formelle :

$$E[n] = \left\{ P \in E(\overline{K}) \mid nP = \mathcal{O} \right\}.$$

On peut définir une courbe elliptique sur un corps de caractéristique p comme supersingulière si $E[p] \cong 0$ et ordinaire, ou non-supersingulière, si $E[p] \cong \mathbb{Z}_p$ avec la notation $\mathbb{Z}_p$ pour $\mathbb{Z}/p\mathbb{Z}$.

Il est évident que $E[n]$ est muni d'une structure de groupe que l'on définit en distinguant deux cas :

- si $n = p^m$ avec p la caractéristique du corps sur lequel E est défini, alors $E[n] = \{\mathcal{O}\}$ si E est supersingulière ou $E[n] \cong \mathbb{Z}_{p^m}$ si la courbe est ordinaire.
- si $n = r$ est premier à p alors $E[r] \cong \mathbb{Z}_r \times \mathbb{Z}_r$ et donc $E[r]$ à r^2 éléments. De plus, si r est premier, le groupe de r-torsion est alors généré par deux points de r-torsion linéairement indépendants.

On utilise principalement par la suite un entier r premier à p.

10.2 Fonctions rationnelles

Définition 10.2.1. *Soit $I = \langle f_1, \ldots, f_s \rangle$ un idéal dans $\overline{K}[X_1, \ldots, X_n]$. La variété affine correspondante $X = V(I) \subset \overline{K}^n = A_K^n$ est l'ensemble des solutions du système d'équations $\{f_1 = 0, \ldots, f_s = 0\}$. On note $\overline{K}[X] = \overline{K}[X_1, \ldots, X_n]/I$ l'anneau de coordonnées de X.*

Soit E la courbe elliptique définie sur $\overline{K}$ d'équation $y^2 = x^3 + ax + b$. On note $f(x, y) \in \overline{K}[x, y],\ f(x, y) = y^2 - x^3 - ax - b$ le polynôme correspondant à E. On définit alors l'anneau

$\overline{K}[E]$ grâce à la Définition 10.2.1 :

$$\overline{K}[E] = \overline{K}[x, y]/\langle f \rangle,$$

où $\langle f \rangle$ est l'idéal engendré par f. Cela signifie que l'on considère tous les polynômes comme étant équivalents dans $\overline{K}[E]$ s'ils prennent les mêmes valeurs en tous les points de la courbe E. Chaque fonction $g(x, y) \in \overline{K}[E]$ peut s'écrire sous forme canonique :

$$g(x, y) = v(x) + yw(x), \quad v, w \in \overline{K}[x],$$

en remplaçant toutes les puissances de y supérieure ou égale à 2 grâce à l'équation de la courbe E.

Soit $\overline{K}(E)$ le corps de fractions associé à $\overline{K}[E]$. Les éléments de $\overline{K}(E)$ s'écrivent $g(x, y)/h(x, y)$ pour $g, h \in \overline{K}[E]$. On appelle un élément de $\overline{K}(E)$ une fonction rationnelle.

10.2.1 Zéros et pôles

Une fonction rationnelle $f = g/h \in \overline{K}(E)$ non nulle est définie en un point $P \in E \setminus \{\mathcal{O}\}$ si $h(P) \neq 0$. On dit que f a un zéro au point P si $f(P) = 0$ et qu'elle a un pôle au point P si f n'est pas définie en P, on note alors $f(P) = \mathcal{O}$. Une fonction rationnelle non nulle a un nombre fini de zéros et de pôles.

L'étude de la fonction $f = g/h$ au point à l'infini $\mathcal{O}$ nécessite de comparer les degrés du numérateur g et du dénominateur h. On définit le degré d'une fonction non nulle $g = v(x) + yw(x) \in \overline{K}[E]$ comme :

$$\deg(g) = \max\left\{2\deg_x(v),\ 3 + 2\deg_x(w)\right\},$$

où $\deg_x$ renvoie le degré en la variable x de la fonction. On applique les coefficients 2 et 3 aux variables x et y respectivement à cause de relation $E : y^2 = x^3 + ax + b$ qui les lie. Les zéros et pôles de la fonction $f = g/h$ au point à l'infini $\mathcal{O}$ sont alors :

- si $\deg(g) < \deg(h)$, alors $f(\mathcal{O}) = 0$, f à un zéro en $\mathcal{O}$,
- si $\deg(g) > \deg(h)$, alors f a un pôle en $\mathcal{O}$ ($f(\mathcal{O}) = \mathcal{O}$),
- si $\deg(g) = \deg(h)$, alors $f = a/b$ où a et b sont les coefficients des termes principaux de g et h respectivement.

Exemple 10.2.1. *Prenons comme exemple une fonction rationnelle plus simple, à une seule variable, et étudions son comportement à l'infini comme ci-dessus.*

Soit $R(z) \in \mathbb{C}[z]$ une fonction rationnelle :

$$R(z) = \frac{a_m z^m + a_{m-1} z^{m-1} + \cdots + a_0}{b_n z^n + b_{n-1} z^{n-1} + \cdots + b_0},$$

avec un numérateur et un dénominateur irréductibles aux coefficients $a_i, b_j \in \mathbb{C}$ tel que $a_m, b_n \neq 0$. Soit $|z| \to \infty$. On écrit alors :

$$R(z) = z^{m-n} . \frac{a_m + \frac{a_{m-1}}{z} + \cdots + \frac{a_0}{z^m}}{b_n + \frac{b_{n-1}}{z} + \cdots + \frac{b_0}{z^n}}.$$

Trois cas se distinguent :

- *si $m < n$, alors $\lim_{|z|\to\infty} R(z) = 0$, R a un zéro à l'infini,*
- *si $m > n$, alors $\lim_{|z|\to\infty} R(z) = \infty$, R a un pôle à l'infini,*
- *si $m = n$, alors $\lim_{|z|\to\infty} R(z) = \frac{a_m}{b_n}$ et donc $R(z)$ a une valeur finie non nulle à l'infini.*

10.2.2 Multiplicité des zéros et pôles

Soit $P = (a, b) \in E(K)$ un point qui n'est pas d'ordre 2. En considérant la fonction rationnelle $g(x, y) = (x - a)^k$ pour un $k > 0$, on remarque que $g(P) = 0$. On dit alors que le point P à un zéro de multiplicité k. De même si $g(x, y) = \frac{1}{(x-a)^k}$ avec $k > 0$, on dit que P a un pôle de multiplicité k.

Cette technique peut être généralisée à toutes les fonctions rationnelles afin de retrouver facilement la multiplicité de leurs zéros et pôles.

Définition 10.2.2. *Pour tout point $P \in E(K)$, il existe une fonction rationnelle u telle que $u(P) = 0$ et telle que toute fonction rationnelle $f(x, y) \in \overline{K}(E)^*$ puisse s'écrire :*

$$f = u^d g, \quad \text{avec } d \in \mathbb{Z},\ g \in \overline{K}(E) \text{ et } g(P) \neq 0, \mathcal{O}.$$

Cette fonction u, unique à une constante multiplicative non nulle près, est appelée uniformisante en P. Elle est notée u_P.

Définition 10.2.3. *L'ordre de $f = u_P^d g$ au point P, noté $\mathrm{ord}_P(f)$, est d.*

Si P est un zéro de f, alors $\mathrm{ord}_P(f) > 0$ et on dit que f a un zéro de multiplicité $\mathrm{ord}_P(f)$ en P. De même, si P est un pôle de f, alors $\mathrm{ord}_P(f) < 0$ et on dit que f a un pôle de multiplicité $-\mathrm{ord}_P(f)$ en P. On note que si P n'est ni un pôle, ni un zéro pour f alors $\mathrm{ord}_P(f) = 0$.

On utilise le théorème suivant pour trouver l'uniformisante en un point P.

Théorème 10.2.1 (voir [Men93, Théorème 2.22])**.** *Soit $P \in E(K)$. Si $l : ax + by + c = 0$ est une ligne passant par P et qui n'est pas tangente à E au point P, alors l est un uniformisante au point P.*

En pratique, on distingue trois cas :

- si $P \notin E[2]$, c.-à-d. $P = (a, b)$ est tel que $b \neq 0$, l'uniformisante évidente en P est la ligne verticale $u = x - a$ qui passe par P et n'est pas tangente à E,
- si $P \in E[2] \setminus \{\mathcal{O}\}$, c.-à-d. $P = (a, 0)$, une ligne qui passe par P et qui n'est pas tangente à E est $u = y$.
- si $P = \mathcal{O}$, alors on peut montrer que $u = x/y$ (voir [Men93]).

Exemple 10.2.2. *Reprenons l'exemple 10.2.1. La factorisation sur $\mathbb{C}[z]$ de $R(z)$ donne :*

$$R(z) = \frac{a_m z^m + a_{m-1} z^{m-1} + \cdots + a_0}{b_n z^n + b_{n-1} z^{n-1} + \cdots + b_0}$$

$$R(z) = \frac{a_m (z - \alpha_1)^{\mu_1} (z - \alpha_2)^{\mu_2} \ldots (z - \alpha_r)^{\mu_r}}{b_n (z - \beta_1)^{\nu_1} (z - \beta_2)^{\nu_2} \ldots (z - \beta_s)^{\nu_s}}$$

avec $\alpha_i \neq \beta_j \ \forall i, j$. *On peut réécrire* $R(z)$ *sous la forme :*

$$R(z) = (z - \alpha_i)^{\mu_i} S_i(z), \quad \forall\ i \in \{1, \ldots, r\},$$

où $S_i(z)$ *est une fonction rationnelle non nulle et non infinie en* $z = \alpha_i$. *On dit ainsi que* α_i *est un zéro de* $R(z)$ *de multiplicité* μ_i. *De même, on peut réécrire* $R(z)$ *comme :*

$$R(z) = \frac{1}{(z - \beta_j)^{\nu_j}} T_j(z), \quad \forall\ j \in \{1, \ldots, s\},$$

où $T_j(z)$ *est une fonction rationnelle non nulle et non infinie en* $z = \beta_j$. *On dit alors que* β_j *est un pôle de* $R(z)$ *de multiplicité* ν_j.

L'étude des zéros et des pôles d'une fonction, ainsi que de leurs multiplicités, suffit à déterminer l'expression de cette dernière à une constante près. On a ensuite besoin d'un outil, les diviseurs, pour les manipuler facilement.

10.3 Diviseurs

10.3.1 Définitions

On définit un symbole formel (P) associé à un point $P \in E(\overline{K})$.

Définition 10.3.1. *Un diviseur* D *sur la courbe* E *est une combinaison linéaire finie à coefficients entiers des symboles définis précédemment :*

$$D = \sum_{P \in E} a_P (P),$$

avec $a_P \neq 0$ *pour un nombre fini de* $P \in E$.

Il faut bien noter qu'un diviseur D est une somme de symboles formels (P) associés à des points P et non une somme de ces points par la loi de groupe sur E. Les diviseurs de E forment un groupe noté $\mathrm{Div}(E)$.

On définit deux fonctions :

– $\deg : \mathrm{Div}(E) \to \mathbb{Z}$ telle que :

$$\deg\left(\sum_{P \in E} a_P (P)\right) = \sum_{P \in E} a_P,$$

– $\mathrm{sum} : \mathrm{Div}(E) \to E(\overline{K})$ telle que :

$$\mathrm{sum}\left(\sum_{P \in E} a_P (P)\right) = \sum_{P \in E} a_P P.$$

La fonction sum utilise, elle, la loi d'addition de points sur E pour son calcul.

On note $\mathrm{Div}^0(E)$ les diviseurs de degré 0, ils forment un sous-groupe de $\mathrm{Div}(E)$.

Comme le nombre de zéros et pôles d'une fonction rationnelle est fini, on peut définir le diviseur d'une fonction $f \in \overline{K}(E)^*$, noté $\mathrm{div}(f)$, comme :

$$\mathrm{div}(f) = \sum_{P \in E} \mathrm{ord}_P(f)(P).$$

10.3.2 Diviseurs principaux

Un diviseur $D \in \mathrm{Div}(E)$ est dit principal si $D = \mathrm{div}(f)$ pour une fonction rationnelle f. L'ensemble des diviseurs principaux est noté $\mathrm{Prin}(E)$.

Proposition 10.3.1 (voir [Ful89, Chapitre 8, Proposition 1]). *Soit une fonction rationnelle $f \in \overline{K}(E)^*$, alors :*

1. *f a autant de pôles que de zéros, de plus elle en a un nombre fini,*
2. $\deg(\mathrm{div}(f)) = 0$,
3. *si* $\mathrm{div}(f) = 0$, *alors f est la fonction constante.*

Il est évident que $\mathrm{Prin}(E)$ forme un sous-groupe de $\mathrm{Div}^0(E)$ grâce à la loi :

$$\mathrm{div}(f_1 f_2) = \mathrm{div}(f_1) + \mathrm{div}(f_2), \quad \forall f_1, f_2 \in \overline{K}(E).$$

Théorème 10.3.1 (voir [Was08, Théorème 11.2]). *Soit $D \in \mathrm{Div}^0(E)$. Alors il existe une fonction $f \in \overline{K}(E)$ telle que :*

$$\mathrm{div}(f) = D,$$

si et seulement si,

$$\mathrm{sum}(D) = \mathcal{O}.$$

Grâce au théorème 10.3.1, on sait que si $D \in \mathrm{Div}^0(E)$ alors $D \in \mathrm{Prin}(E)$ si et seulement si $\mathrm{sum}(D) = \mathcal{O}$. En remarquant que,

$$\mathrm{sum}((P) - (\mathcal{O})) = P,$$

la fonction sum définit un homomorphisme surjectif tel que :

$$\mathrm{sum} : \mathrm{Div}^0(E) \to E(\overline{K}).$$

Le théorème 10.3.1 prouve que le noyau de cet homomorphisme est $\mathrm{Prin}(E)$ et donc on a un isomorphisme de groupes défini par la fonction :

$$\mathrm{sum} : \mathrm{Div}^0(E)/\mathrm{Prin}(E) \to E(\overline{K}).$$

La loi de groupe sur $E(\overline{K})$ correspond à la loi de groupe de $\mathrm{Div}^0(E)/\mathrm{Prin}(E)$ qui est l'ensemble des diviseurs de degré zéro qui ne sont pas principaux.

Deux diviseurs $D_1, D_2 \in \mathrm{Div}(E)$ sont équivalents, noté $D_1 \sim D_2$, si il existe une fonction rationnelle f telle que $D_1 = D_2 + \mathrm{div}(f)$, c.-à-d. si $D_1 - D_2 \in \mathrm{Prin}(E)$.

Théorème 10.3.2. *Soit $D \in \mathrm{Div}(E)$. Il existe alors un unique $P \in E(\overline{K})$ tel que :*

$$D \sim (P) + (\deg(D) - 1)(\mathcal{O}).$$

Plus particulièrement, si $D \in \mathrm{Div}^0(E)$, alors $D \sim (P) - (\mathcal{O})$ pour un certain point $P \in \mathrm{Div}(E)$. On dit que D est sous forme canonique.

10.3.3 Action d'une fonction sur un diviseur

On définit en premier le support d'un diviseur $D = \sum_{P \in E} a_P(P)$ comme étant :

$$\mathrm{supp}(D) = \{P \in E \mid a_P \neq 0\}.$$

On définit l'action d'une fonction rationnelle $f \in \overline{K}(E)$ sur un diviseur $D = \sum_{P \in E} a_P(P)$ tel que $\mathrm{supp}(D) \cap \mathrm{supp}(\mathrm{div}(f)) = \emptyset$, tel que D et $\mathrm{div}(f)$ n'ont aucuns points en commun, par la formule :

$$f(D) = f\left(\sum_{P \in \mathrm{supp}(D)} a_P(P)\right) = \prod_{P \in \mathrm{supp}(D)} f(P)^{a_P}.$$

Lemme 10.3.1. *Soit $D \in \mathrm{Div}^0(E)$ et $f_1, f_2 \in \overline{K}(E)$. On suppose que* $\mathrm{supp}(D) \cap \mathrm{supp}(\mathrm{div}(f_1)) = \emptyset$. *Soit $c \in \overline{K}^*$, la fonction $f_2 = cf_1$ satisfait alors :*

$$f_2(D) = f_1(D).$$

Démonstration. Soit $\mathcal{D} = \mathrm{supp}(D)$. On remarque que $\mathrm{div}(f_2)$ a le même support que $\mathrm{div}(f_1)$ et est donc aussi disjoint de celui de D. On peut donc calculer :

$$f_2(D) = \prod_{P \in \mathcal{D}} f_2(P)^{a_P} = \prod_{P \in \mathcal{D}} (cf_1(P))^{a_P} = c^{\sum_{P \in \mathcal{D}} a_P} \prod_{P \in \mathcal{D}} f_1(P)^{a_P}.$$

Or on a $\sum_{P \in \mathcal{D}} a_P = 0$ car D est de degré zéro. D'où :

$$f_2(D) = \prod_{P \in \mathcal{D}} f_1(P)^{a_P} = f_1(D).$$

□

Lemme 10.3.2 (Loi de réciprocité de Weil)**.** *Soit $f, g \in \overline{K}(E)$ et* $\mathrm{supp}(\mathrm{div}(f)) \cap \mathrm{supp}(\mathrm{div}(g)) = \emptyset$, *alors :*

$$f(\mathrm{div}(g)) = g(\mathrm{div}(f)).$$

Pour la preuve de ce lemme, voir [Sil86].

10.3.4 Construire une fonction associée à un diviseur

Soit $D \in \mathrm{Div}^0(E)$ écrit sous la forme canonique :

$$D = (P) - (\mathcal{O}) + \mathrm{div}(f),$$

pour une certaine fonction rationnelle $f \in \overline{K}(E)$ et pour un certain point $P \in E(\overline{K})$.

Soient $P_1, P_2, P_3 \in E(\overline{K})$ tels que $P_1 + P_2 = P_3$. Soit $L : ax + by + c = 0$ la ligne passant par P_1 et P_2 on a :

$$\mathrm{div}(L) = (P_1) + (P_2) + (-P_3) - 3(\mathcal{O}).$$

Soit $V : x + x_3 = 0$, la droite verticale passant par $P_3 = (x_3, y_3)$, alors :

$$\mathrm{div}(V) = (P_3) + (-P_3) - 2(\mathcal{O}).$$

Soient $D_1, D_2 \in \mathrm{Div}^0(E)$ tels que :

$$\begin{aligned} D_1 &= (P_1) - (\mathcal{O}) + \mathrm{div}(f_1) \\ D_2 &= (P_2) - (\mathcal{O}) + \mathrm{div}(f_2). \end{aligned}$$

La somme $D_1 + D_2$ se calcule grâce aux fonctions $L(x, y)$ et $V(x)$:

$$\begin{aligned} D_1 + D_2 &= (P_1) + (P_2) - 2(\mathcal{O}) + \mathrm{div}(f_1 f_2) \\ &= (P_3) - (\mathcal{O}) + \mathrm{div}(L) - \mathrm{div}(V) + \mathrm{div}(f_1 f_2) \\ &= (P_3) - (\mathcal{O}) + \mathrm{div}(f_1 f_2 f_3), \end{aligned}$$

avec $f_3 = L/V$ une fonction rationnelle sur $\overline{K}(E)$ définie partout sauf en P_3 et $-P_3$. On peut noter que dans l'équation ci-dessus : $(P_1) + (P_2) = (P_3) + (\mathcal{O}) = (P_1 + P_2) + (\mathcal{O})$.

Utiliser les fonctions de lignes et verticales en des points de la courbe elliptique est très avantageux pour construire une fonction rationnelle associée à un diviseur. En effet, ces constructions géométriques sont identiques à celles utilisées dans la loi d'addition de points sur courbe elliptique ce qui fait gagner énormément de temps comme nous le verrons plus tard.

10.3.5 Exemple de construction

Soit la courbe elliptique $E : y^2 = x^3 + 5x$ définie sur le corps $\mathbb{F}_{11}$ telle que $\#E(\mathbb{F}_{11}) = 12$, d'où $t = q + 1 - \#E(\mathbb{F}_{11}) = 0$ donc E est supersingulière. Le tableau 10.1 donne la liste des points et leur ordre.

Soit $D = 6(P_3) - 6(\mathcal{O})$ un diviseur qui est clairement principal car l'ordre de P_3 est égal à 6. On cherche la fonction rationnelle $f = \mathrm{div}(D)$ associée au diviseur D. On calcule

Point	Ordre	Point	Ordre
$P_0 = \mathcal{O}$	1	$P_6 = (7, 2)$	4
$P_1 = (0, 0)$	2	$P_7 = (7, 9)$	4
$P_2 = (3, 3)$	6	$P_8 = (9, 2)$	3
$P_3 = (3, 8)$	6	$P_9 = (9, 9)$	3
$P_4 = (6, 2)$	12	$P_{10} = (10, 4)$	12
$P_5 = (6, 9)$	12	$P_{11} = (10, 7)$	12

TABLE 10.1 – Liste des points de $E : y^2 = x^3 + 5x$ définie sur $\mathbb{F}_{11}$.

une chaîne de diviseurs principaux afin de calculer D. Tout d'abord :

$$\begin{aligned} 2(P_3) - 2(\mathcal{O}) &= ((P_3) - (\mathcal{O})) + ((P_3) - (\mathcal{O})) \\ &= (P_3 + P_3) + (\mathcal{O}) - 2(\mathcal{O}) + \operatorname{div}(L_{P_3,P_3}/V_{P_8}) \\ &= (P_8) - (\mathcal{O}) + \operatorname{div}\left(\underbrace{\frac{y + 9x + 9}{x + 2}}_{g_1}\right), \end{aligned}$$

avec $(P_3) + (P_3) = (P_3 + P_3) + (\mathcal{O}) = (P_8) + (\mathcal{O})$, $L_{P_3,P_3}(x, y) = y + 9x + 9$ la tangente en P_3 à E, $V_{P_8}(x) = x + 2$ la verticale en P_8 et g_1 la fonction rationnelle telle que $g_1(x, y) = L_{P_3,P_3}(x, y)/V_{P_8}(x)$.

De même on calcule :

$$\begin{aligned} 4(P_3) - 4(\mathcal{O}) &= (2(P_3) - 2(\mathcal{O})) + (2(P_3) - 2(\mathcal{O})) \\ &= (P_8 + P_8) + (\mathcal{O}) - 2(\mathcal{O}) + \operatorname{div}(g_1^2) + \operatorname{div}(L_{P_8,P_8}/V_{P_9}) \\ &= (P_9) - (\mathcal{O}) + \operatorname{div}\left(\underbrace{\frac{(y + 9x + 9)^2}{(x + 2)^2} . \frac{y + 4x + 6}{x + 2}}_{g_2}\right), \end{aligned}$$

avec $L_{P_8,P_8}(x, y) = y + 4x + 6$ la tangente en P_8 à E, $V_{P_9}(x) = x + 2$ la verticale en P_9 et g_2 la fonction rationnelle telle que $g_2(x, y) = g_1^2 . L_{P_8,P_8}(x, y)/V_{P_9}(x)$.

Pour finir,

$$\begin{aligned} 6(P_3) - 6(\mathcal{O}) &= (4(P_3) - 4(\mathcal{O})) + (2(P_3) - 2(\mathcal{O})) \\ &= (P_8 + P_9) + (\mathcal{O}) - 2(\mathcal{O}) + \operatorname{div}(g_1) + \operatorname{div}(g_2) + \operatorname{div}(L_{P_8,P_9}/V_{\mathcal{O}}) \\ &= \operatorname{div}\left(\frac{(y + 9x + 9))}{(x + 2)} . \frac{(y + 9x + 9)^2 (y + 4x + 6)}{(x + 2)^3} . \frac{(x + 2)}{1}\right) \\ &= \operatorname{div}\left(\frac{(y + 9x + 9)^3 (y + 4x + 6)}{(x + 2)^3}\right), \end{aligned}$$

avec $L_{P_8,P_9}(x,y) = x + 2$ la ligne entre P_8 et P_9 qui est une verticale car $P_9 = -P_8$ et $V_{\mathcal{O}}(x) = 1$ par convention.

La fonction rationnelle f associée au diviseur $D = 6(P_3) - 6(\mathcal{O})$ est donc :

$$f(x,y) = \frac{(y + 9x + 9)^3(y + 4x + 6)}{(x + 2)^3}.$$

En considérant simplement $f \in \overline{K}(x,y)$, la fonction n'est pas définie en P_8 et P_9. Mais en tant que fonction rationnelle, $f \in \overline{K}(E)$, on peut réarranger les termes tels que :

$$\begin{aligned}
f(x,y) &= \frac{(y + 9x + 9)^3(y + 4x + 6)}{(x + 2)^3} . \frac{(y - 9x - 9)^3}{(y - 9x - 9)^3} \\
&= \frac{(y^2 + 7x^2 + 3x + 7)^3}{(x + 2)^3} . \frac{(y + 4x + 6)}{(y - 9x - 9)^3} \\
&= \frac{(x^3 + 7x^2 + 8x + 7)^3}{(x + 2)^3} . \frac{(y + 4x + 6)}{(y - 9x - 9)^3} \\
&= \frac{(x + 2)^3(x + 8)^6}{(x + 2)^3} . \frac{(y + 4x + 6)}{(y - 9x - 9)^3} \\
&= \frac{(y + 4x + 6)(x + 8)^6}{(y - 9x - 9)^3},
\end{aligned}$$

la fonction f est alors définie en $P_9 = (9, 9)$. De même, on peut écrire :

$$\begin{aligned}
f(x,y) &= \frac{(y + 9x + 9)^3(y + 4x + 6)}{(x + 2)^3} . \frac{(y - 4x - 6)}{(y - 4x - 6)} \\
&= \frac{(y^2 + 6x^2 + 7x + 8)}{(x + 2)^3} . \frac{(y + 9x + 9)^3}{(y - 4x - 6)} \\
&= \frac{(x^3 + 6x^2 + x + 8)}{(x + 2)^3} . \frac{(y + 9x + 9)^3}{(y - 4x - 6)} \\
&= \frac{(x + 2)^3}{(x + 2)^3} . \frac{(y + 9x + 9)^3}{(y - 4x - 6)} \\
&= \frac{(y + 9x + 9)^3}{(y - 4x - 6)}
\end{aligned}$$

avec la fonction f désormais définie en $P_8 = (9, 2)$.

10.4 Distortion map

Nous définissons les *distortion maps* qui ont été introduites par Verheul [Ver04] pour les courbes supersingulières. Ces fonctions permettent de trouver facilement des points de courbe elliptique indépendants.

Définition 10.4.1. *Soit $E/\mathbb{F}_q$ une courbe elliptique et m un grand premier divisant $\#E(\mathbb{F}_q)$. Soit $P \in E(\mathbb{F}_q)[m]$. Une distortion map, relativement à P, est un endomorphisme de courbe ϕ qui transforme le point P en $\phi(P)$ linéairement indépendant à P.*

On note qu'un endomorphisme envoie toujours $\mathcal{O}$ sur $\mathcal{O}$. Il n'existe donc pas de *distortion map* relative à $\mathcal{O}$. De plus, l'image d'un point de m-torsion est encore un point de m-torsion par l'endomorphisme. Soit $P \in E(\mathbb{F}_q)[m]$, alors $\phi(P) \in E[m]$. Si on suppose le groupe $E(\mathbb{F}_q)[m]$ cyclique, comme souvent en pratique, alors $\phi(P)$ a ses coordonnées sur une extension de corps de $\mathbb{F}_q$. Une *distortion map* sur une courbe E de degré de plongement $k > 1$ existe si et seulement si E est supersingulière [Ver04, GR10]. Le cas $k = 1$ est traité par Charles [Cha06].

10.5 Endomorphisme de Frobenius

Dans la section 5.5, nous avons défini le cardinal du groupe de points d'une courbe elliptique $E/\mathbb{F}_q$ comme : $\#E(\mathbb{F}_q) = q + 1 - t$ avec $|t| \leq 2\sqrt{q}$. La valeur t est appelée trace de E ou trace de l'endomorphisme de Frobenius de $E/\mathbb{F}_q$. L'endomorphisme de Frobenius est un morphisme de courbe très utilisé.

Définition 10.5.1. *L'endomorphisme de Frobenius est une application de la courbe $E/\mathbb{F}_q$ définie par :*

$$\begin{aligned} \pi_q : E &\to E \\ (x, y) &\mapsto (x^q, y^q). \end{aligned}$$

Étant donné que $x \in \mathbb{F}_q$ si et seulement si $\pi_q(x) = x$, l'action de π_q sur l'ensemble des points de la courbe donne le groupe de points $E(\mathbb{F}_q)$. Le polynôme caractéristique de l'endomorphisme de Frobenius est : $X^2 - tX + q$ où $t = q + 1 - \#E(\mathbb{F}_q)$ et $q \equiv \det(\pi_q) \bmod m$.

10.6 Degré de plongement

Soit E une courbe elliptique définie sur le corps $\mathbb{F}_q$. Dans de nombreuses applications cryptographiques, on travaille dans un sous-groupe d'ordre m premier, un grand facteur de $n = \#E(\mathbb{F}_q)$.

Définition 10.6.1. *Le degré de plongement de la courbe E, relativement à m, est le plus petit entier k tel que $m \mid (q^k - 1)$.*

Si $m \nmid (q^k - 1)$, le degré de plongement détermine le degré de la plus petite extension de $\mathbb{F}_q$ sur laquelle tous les points de m-torsion de E sont définis.

Théorème 10.6.1 ([BK98, Théorème 1])**.** *Soit E une courbe elliptique définie sur $\mathbb{F}_q$. Soit G un sous-groupe de $E(\mathbb{F}_q)$ d'ordre m tel que $m \mid \#E(\mathbb{F}_q)$ et $m \nmid q - 1$. Alors, pour tout entier positif k, $E(\mathbb{F}_{q^k})$ contient tous les m^2 points de m-torsion si et seulement si $m \mid (q^k - 1)$.*

En pratique, on utilise des courbes telles que m et $\#E(\mathbb{F}_q)$ sont très proches. On omet donc souvent de préciser que le degré de plongement est défini relativement à m. Lorsqu'on parle de degré de plongement de la courbe $E(\mathbb{F}_q)$, on considère le degré de plongement du sous-groupe de points de $E(\mathbb{F}_q)$ de cardinal m avec m le plus grand premier divisant $\#E(\mathbb{F}_q)$.

10.7 Courbe elliptique tordue

Définition 10.7.1. *Soit deux courbes elliptiques E et E' définies sur $\mathbb{F}_q$, on appelle E' une courbe tordue, ou twist, de degré d de E s'il existe un isomorphisme $\sigma : E' \to E$ défini sur $\mathbb{F}_{q^d}$ et que d est minimal.*

Il n'existe qu'un nombre limité de degrés de *twists* possibles. Pour chacun de ces degrés, les *twists* peuvent être décrits [Sil86, HSV06].

Proposition 10.7.1. *Soit E une courbe elliptique définie sur K avec* $\mathrm{car}(K) \notin \{2,3\}$ *d'équation $E : y^2 = x^3 + ax + b$. Soit $\delta = 2$ si $j(E) \notin \{0, 1728\}$, $\delta = 4$ si $j(E) = 1728$ et $\delta = 6$ si $j(E) = 0$. Pour $\xi \in K^*$, la courbe tordue E_ξ correspondant à $\xi \bmod (K^*)^\delta$ a pour équation :*

$$\begin{array}{lll} E_\xi : y^2 = x^3 + \xi^{-2}ax + \xi^{-3}b & \text{si } \ j(E) \notin \{0, 1728\} & \text{et donc } \ \delta = 2, \\ E_\xi : y^2 = x^3 + \xi^{-1}ax & \text{si } \ j(E) = 1728 & \text{et donc } \ \delta = 4, \\ E_\xi : y^2 = x^3 + \xi^{-1}b & \text{si } \ j(E) = 0 & \text{et donc } \ \delta = 6. \end{array}$$

Proposition 10.7.2. *Les homomorphismes correspondants $\sigma_\xi : E_\xi \to E$ sont définis par :*

$$\begin{array}{lll} (x, y) \mapsto (\xi x, \xi^{3/2} y) & \text{si } \ j(E) \notin \{0, 1728\} & \text{et donc } \ \delta = 2, \\ (x, y) \mapsto (\xi^{1/2} x, \xi^{3/4} y) & \text{si } \ j(E) = 1728 & \text{et donc } \ \delta = 4, \\ (x, y) \mapsto (\xi^{1/3} x, \xi^{1/2} y) & \text{si } \ j(E) = 0 & \text{et donc } \ \delta = 6. \end{array}$$

Il existe une relation entre δ, le j-invariant $j(E)$ et le degré d du *twist*. Elle est détaillée dans la table suivante :

$j(E)$	δ	ξ	d
$\notin \{0, 1728\}$	2	$\in (K^*)^2$	1
		$\notin (K^*)^2$	2
1728	4	$\in (K^*)^4$	1
		$\in (K^*)^2, \notin (K^*)^4$	2
		$\notin (K^*)^2$	4
0	6	$\in (K^*)^6$	1
		$\in (K^*)^3, \notin (K^*)^2$	2
		$\in (K^*)^2, \notin (K^*)^3$	3
		$\notin (K^*)^2, \notin (K^*)^3$	6

Hess et al. [HSV06] déterminent les cardinaux des groupes de points des courbes tordues. On note tout de même que $\#E(\mathbb{F}_{q^d}) = \#E'(\mathbb{F}_{q^d})$ pour un d fixé.

Proposition 10.7.3 ([HSV06, Proposition 8])**.** *Soit E une courbe elliptique définie sur $\mathbb{F}_q$ et $\#E(\mathbb{F}_q) = q+1-t$. Soit E' la courbe tordue de E de degré d. Le groupe de points $E'(\mathbb{F}_q)$ peut avoir différents cardinaux possibles définis tels que :*

- *si $d = 2$, alors*

$$\#E'(\mathbb{F}_q) = q + 1 - t,$$

- *si $d = 3$, alors*

$$\#E'(\mathbb{F}_q) = \begin{cases} q + 1 - (3v - t)/2 & \text{avec } \ t^2 - 4q = -3v^2, \\ q + 1 - (-3v - t)/2 & \text{avec } \ t^2 - 4q = -3v^2, \end{cases}$$

- *si $d = 4$, alors*

$$\#E'(\mathbb{F}_q) = \begin{cases} q + 1 - v & \text{avec } \ t^2 - 4q = -v^2, \\ q + 1 + v & \text{avec } \ t^2 - 4q = -v^2, \end{cases}$$

- *si $d = 6$, alors*

$$\#E'(\mathbb{F}_q) = \begin{cases} q + 1 - (3v + t) & \text{avec } \ t^2 - 4q = -3v^2, \\ q + 1 - (-3v + t) & \text{avec } \ t^2 - 4q = -3v^2. \end{cases}$$

Chapitre 11

Couplages de Weil et Tate

Nous nous intéressons dans ce chapitre aux constructions des couplages de Weil et Tate. Ces couplages sont les premiers à être apparus en cryptographie. Ils ont d'abord été utilisés pour des attaques sur les cryptosystèmes à base de courbes elliptiques. Dans un deuxième temps, la propriété de bilinéarité de ces fonctions mathématiques a permis de construire des protocoles cryptographiques originaux.

11.1 Couplage de Weil

11.1.1 Définition

Soit E/K une courbe elliptique définie sur le corps K de caractéristique p. Soit un entier m premier à p. On sait, d'après la section 10.1, que $E[m] \cong \mathbb{Z}_m \times \mathbb{Z}_m$.

11.1.1.1 Racines m-ième de l'unité

Soit μ_m le groupe des racines m-ième de l'unité dans $\overline{K}$ défini par :

$$\mu_m = \left\{ x \in \overline{K} \mid x^m = 1 \right\}.$$

Comme m est premier à p, le groupe μ_m est un groupe cyclique à m éléments généré par ζ, appelé racine m-ième primitive de l'unité, c.-à-d. $\zeta^k = 1$ si et seulement si m divise k.

11.1.1.2 Construction du couplage de Weil

Soit $T \in E[m]$. On sait grâce au Théorème 10.3.1 qu'il existe une fonction $f \in \overline{K}(E)$ telle que :

$$\mathrm{div}(f) = m(T) - m(\mathcal{O}).$$

Soit $T' \in E[m^2]$ tel que $mT' = T$. Un tel T' existe car on travaille dans la clôture algébrique $\overline{K}$. Soit D un diviseur tel que :

$$D = \sum_{R \in E[m]} ((T' + R) - (R)).$$

Nous allons montrer que D est un diviseur principal, c.-à-d. qu'il existe une fonction g telle que $D = \mathrm{div}(g)$.

Le diviseur D est clairement de degré zéro. De plus, on sait d'après la section 10.1 que $\#E[m] = m^2$ étant donné que m est premier à p. On a donc en appliquant la fonction sum :

$$\mathrm{sum}(D) = \sum_{R \in E[m]} (T' + R - R) = m^2 T' = \mathcal{O}.$$

Il existe donc une fonction $g \in \overline{K}(E)$ telle que $D = \mathrm{div}(g)$. On remarque que g ne dépend pas de la valeur de T' puisqu'un autre choix revient à avoir une différence d'un élément $R \in E[m]$ par rapport au premier. Étant donné que l'on parcours l'ensemble des $R \in E[m]$ cette différence n'a pas d'importance.

Soit $f \circ f_m$ avec $f_m : E(\overline{K}) \to E(\overline{K})$ la fonction qui multiplie un point de la courbe par m, c.-à-d. $f_m(P) = mP$ pour $P \in E$, et f la fonction définie par son diviseur $\mathrm{div}(f)$ ci-dessus. On calcule :

$$\mathrm{div}(f \circ f_m) = m \left(\sum_{mP=T} (P) - \sum_{mQ=\mathcal{O}} (Q) \right).$$

En remarquant que les points $P = T' + R$ avec $R \in E[m]$ sont tels que $mP = T$, et que, évidemment, $mR = \mathcal{O}$, $\forall R \in E[m]$, on peut noter :

$$\begin{aligned} \mathrm{div}(f \circ f_m) &= m \left(\sum_R (T' + R) - \sum_R (R) \right) \\ &= m(\mathrm{div}(g)) = \mathrm{div}(g^m). \end{aligned}$$

Donc d'après le Lemme 10.3.1, les fonctions $f \circ f_m$ et g^m sont égales à une constante près.

Soient $S \in E[m]$ et $P \in E(\overline{K})$, alors :

$$g(P+S)^m = cf(f_m(P+S)) = cf(mP + mS) = cf(f_m(P)) = g(P)^m,$$

en notant $g^m = c(f \circ f_m)$ pour un $c \in \overline{K}^*$. On a donc que $g(P+S)/g(P) \in \mu_m$, c.-à-d. est une racine m-ième de l'unité dans $\overline{K}$.

On peut désormais définir le couplage de Weil.

Définition 11.1.1. *Le couplage de Weil est la fonction e_m telle que :*

$$e_m : E[m] \times E[m] \to \mu_m \tag{11.1}$$

$$e_m(S,T) = \frac{g(P+S)}{g(P)}, \tag{11.2}$$

avec $P \in E$ un point tel que $g(P+S), g(P) \neq 0, \mathcal{O}$.

La fonction g étant déterminée à une constante près, le couplage de Weil ne dépend pas de sa valeur. De plus, le couplage ne dépend pas non plus du point $P \in E(\overline{K})$ choisi. En effet, soit la fonction $\mathcal{T}_P : E \to E$ telle que $\mathcal{T}_P(Q) = Q + P$ pour un point $Q \in E$. On peut réécrire la relation (11.2) comme :

$$e_m(S,T) = \frac{g \circ \mathcal{T}_S}{g}.$$

D'après [CR88, Lemme 13.3], on sait que $\mathrm{div}(g \circ \mathcal{T}_S) = \mathrm{div}(g)$, donc $(g \circ \mathcal{T}_S)/g$ est une constante. Ainsi, le couplage e_m ne dépend pas du choix du point P.

11.1.2 Propriétés

Théorème 11.1.1 (voir [Was08] Théorème 11.7)**.** *Soit e_m le couplage de Weil défini par* (11.2)*. Soient $S, S_1, S_2, T, T_1, T_2 \in E[m]$. Le couplage satisfait alors aux propriétés suivantes :*

1. $e_m(S_1 + S_2, T) = e_m(S_1, T)e_m(S_2, T)$ *(linéaire en son premier argument),*
2. $e_m(S, T_1 + T_2) = e_m(S, T_1)e_m(S, T_2)$ *(linéaire en son second argument),*
3. $e_m(T, T) = 1, \ \forall\ T \in E[m]$ *(identité),*
4. $e_m(S, T) = e_m(T, S)^{-1}, \ \forall\ S, T \in E[m]$ *(alternée),*
5. *si* $e_m(S, T) = 1\ \forall\ S \in E[m]$*, alors* $T = \mathcal{O}$ *(non-dégénérée),*
6. $e_m(\alpha(S), \alpha(T)) = e_m(S, T)^{\deg(\alpha)}$ *pour tout endomorphisme α. En particulier, la propriété s'applique pour l'endomorphisme de Frobenius ϕ_q lorsque tous les coefficients de la courbe E sont dans $\mathbb{F}_q$ (voir section 10.5). On a :* $e_m(\phi_q(S), \phi_q(T)) = e_m(S, T)^q$.

Proposition 11.1.1. *Soient m un entier premier, S et T deux points de m-torsion linéairement indépendants. Le couplage de Weil $e_m(S, T)$ est alors non-trivial, $e_m(S, T) \neq 1$.*

Démonstration. Le couplage de Weil étant non-dégénéré, il existe un point $S' \in E[m]$, linéairement indépendant de S, tel que $e_m(S, S') \neq 1$. Le groupe de m-torsion peut donc être engendré par S et S', m étant premier. On peut écrire $T \in E[m]$ comme $T = aS + bS'$ avec $0 \leq a \leq m-1$ et $0 \leq b \leq m-1$. D'où :

$$e_m(S, T) = e_m(S, aS + bS') = e_m(S, S)^a e_m(S, S')^b = e_m(S, S')^b \neq 1.$$

La dernière égalité est due au fait que m est premier et donc que $e_m(S, S')$ est une racine m-ième primitive de l'unité. □

11.1.3 Définition alternative

Cette deuxième définition du couplage de Weil permet un calcul beaucoup plus efficace en pratique. En effet, avec la précédente définition, il fallait parcourir les m^2 points de m-torsion dans $E[m]$ sachant qu'en pratique m est choisi comme un entier très grand.

Définition 11.1.2. *Soit m un entier premier à p la caractéristique du corps K. Soit $S, T \in E[m]$ des points de m-torsion sur la courbe E. Soit $D_S, D_T \in \mathrm{Div}^0(E)$ tels que :*

$$\mathrm{sum}(D_S) = S \quad et \quad \mathrm{sum}(D_T) = T,$$

et $\mathrm{supp}(D_S) \cap \mathrm{supp}(D_T)) = \emptyset$. *Soit $f_S, f_T \in \overline{K}(E)$ deux fonctions rationnelles telles que :*

$$\mathrm{div}(f_S) = mD_S \quad et \quad \mathrm{div}(f_T) = mD_T,$$

qui sont bien définies car $S, T \in E[m]$. On définit alors le couplage de Weil e'_m comme :

$$e'_m(S,T) = \frac{f_T(D_S)}{f_S(D_T)}.$$

Théorème 11.1.2. *Le couplage de Weil tel que défini dans la Définition 11.1.2 retourne bien une racine m-ième de l'unité et est indépendant du choix des diviseurs D_T, D_S et des fonctions f_T, f_S.*

Démonstration. Par la loi de réciprocité de Weil (Lemme 10.3.2), on a :

$$\left(\frac{f_T(D_S)}{f_S(D_T)}\right)^m = \frac{f_T(D_S)^m}{f_S(D_T)^m} = \frac{f_T(mD_S)}{f_S(mD_T)} = \frac{f_T(\mathrm{div}(f_S))}{f_S(mD_T)} = \frac{f_S(\mathrm{div}(f_T))}{f_S(mD_T)} = \frac{f_S(mD_T)}{f_S(mD_T)} = 1.$$

Donc $e'_m(S,T)$ est bien une racine m-ième de l'unité.

On prouve maintenant que le couplage ne dépend pas du choix du diviseur D_T (le raisonnement est similaire pour D_S). Soit $D'_T \sim (T) - (\mathcal{O})$ un diviseur de support disjoint de D_S. Soit f' la fonction telle que $\mathrm{div}(f') = mD'_T$. Comme $D_T \sim D'_T$, on écrit $D'_T = D_T + \mathrm{div}(g)$ pour une fonction $g \in \overline{K}(E)$. Donc $f' = f_T.g^m$. D'où :

$$\frac{f'(D_S)}{f_S(D'_T)} = \frac{f_T(D_S).g^m(D_S)}{f_S(D_T).f_S(\mathrm{div}(g))} = \frac{f_T(D_S).g(mD_S)}{f_S(D_T).f_S(\mathrm{div}(g))} = \frac{f_T(D_S)}{f_S(D_T)}.\frac{g(\mathrm{div}(f_S))}{f_S(\mathrm{div}(g))} = \frac{f_T(D_S)}{f_S(D_T)},$$

où la dernière égalité vient de la loi de réciprocité de Weil. Le couplage ne dépend donc pas du choix du diviseur D_T (de même pour D_S).

Les fonctions f_T et f_S sont uniques à une constante près. Comme elles sont toutes les deux évaluées en un diviseur de degré zéro, d'après le Lemme 10.3.1, le résultat des fonctions est indépendant du choix de la constante. □

Les deux définitions du couplage de Weil ne sont pas tout à fait égales, comme souvent écrit dans la littérature.

Théorème 11.1.3 (voir [Hes04])**.** *Soit $S, T \in E[m]$. Alors $e_m(S,T) = 1/e'_m(S,T)$.*

En pratique, on utilise une version légèrement modifiée de cette définition. Soit $S, T \in E[m]$. On choisit $S', T' \in E(\overline{K})$ tels que S', $S + S'$, T' et $T + T'$ soient différents. Soient les diviseurs D_S et D_T tels que :

$$D_S = (S + S') - (S') \quad \text{et} \quad D_T = (T + T') - (T').$$

On a alors $\mathrm{div}(f_S) = mD_S$ et $\mathrm{div}(f_T) = mD_T$ qui sont clairement principaux. On peut alors calculer e'_m comme :

$$e'_m(S,T) = \frac{f_T(D_S)}{f_S(D_T)} = \frac{f_T((S+S') - (S'))}{f_S((T+T') - (T'))} = \frac{f_T(S+S')f_S(T')}{f_T(S')f_S(T+T')}. \tag{11.3}$$

Les conditions sur S' et T' assurent que f_S et f_T sont respectivement bien définies en T', $T+T'$ et S', $S+S'$. On procède ainsi car on veut traiter les fonctions f_S et f_T comme fonctions dans $\overline{K}(x,y)$ plutôt que dans $\overline{K}(E)$ et donc qu'elles soient bien définies en les points ci-dessus. On peut montrer que la probabilité de choisir une paire (S',T') au hasard qui satisfait à ces conditions est proche de 1 pour un $m >> 1024$, comme utilisé en pratique.

On considère, par la suite, cette définition alternative du couplage de Weil sauf mention contraire. Pour plus de facilité et lorsque m est implicite, on note simplement e'_m par e.

Remarque 11.1.1. *On a défini le couplage de Weil comme une fonction à valeurs dans $\overline{K}$ avec $K = \mathbb{F}_q$. En pratique, on n'a pas besoin de toute la clôture algébrique mais juste d'une extension de corps $\mathbb{F}_{q^k}$ où k est le plus petit entier positif tel que $E[m] \subseteq E(\mathbb{F}_{q^k})$. On appelle ce paramètre k le degré de plongement de Weil de E relatif à m (voir section 10.6).*

11.1.4 Algorithme de Miller appliqué à Weil

Le principe de l'algorithme de Miller (voir [Mil86a, ELM03]) est d'évaluer les fonctions rationnelles définies ci-dessus uniquement en un diviseur donné.

Il faut ainsi trouver la valeur d'une fonction f_T qui a comme diviseur $\mathrm{div}(f_T) = mD_T = m(T+T') - m(T')$ (de même pour la fonction f_S). On rappelle que $T \in E[m]$ et $T' \in E(\overline{K})$.

On définit les notations suivantes pour les points $P = (x_1, y_1), Q = (x_2, y_2)$:

– soit $L_{P,Q}(x,y)$ l'équation de la ligne qui passe par les points P et Q :

$$L_{P,Q}(x,y) = -\lambda x + y + (\lambda x_1 - y_1) \quad \text{avec} \quad \lambda = \frac{y_2 - y_1}{x_2 - x_1},$$

$$\mathrm{div}(L_{P,Q}) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{O}),$$

– soit $T_P(x,y)$ l'équation de la tangente en P à la courbe :

$$T_P(x,y) = -\lambda x + y + (\lambda x_1 - y_1) \quad \text{avec} \quad \lambda = \frac{3x^2 + a}{2y},$$

– soit $V_P(x,y)$ l'équation de la verticale en P :

$$V_P(x,y) = x - x_1,$$

$$\mathrm{div}(V_P) = (P) + (-P) - 2(\mathcal{O}).$$

On remarque que $L_{P,P} = T_P$, $V_P = L_{P,-P}$ et si $P = -P$ alors $T_P = V_P$.

Pour $i \geq 1$, soit f_i la fonction rationnelle de diviseur :

$$\mathrm{div}(f_i) = i(T + T') - i(T') - (iT) + (\mathcal{O}).$$

On remarque que $f_m = f_T$ définie ci-dessus car $mT = \mathcal{O}$. On cherche à calculer la valeur $f_T = f_m$, on utilise pour cela les f_i intermédiaires et une formule de récursion donnée par le théorème suivant.

Théorème 11.1.4 (Formule de Miller pour Weil)**.** *Soit $S \in E[m]$, i et j deux entiers positifs. Soit $l = L_{iT,jT}$ la ligne passant par les points iT et jT. Soit $v = V_{iT+jT}$ la verticale passant par $iT + jT$. Alors :*

$$f_{i+j} = f_i.f_j.\frac{l}{v}.$$

Démonstration. La formule se vérifie simplement en calculant avec les diviseurs des fonctions :

$$\begin{aligned}
\mathrm{div}(f_i) + \mathrm{div}(f_j) + \mathrm{div}(l) - \mathrm{div}(v) = & \\
& (i(T + T') - i(T') - (iT) + (\mathcal{O})) \\
& + (j(T + T') - j(T') - (jT) + (\mathcal{O}]) \\
& + ((iT) + (jT) + (-(i + j)T) - 3(\mathcal{O})) \\
& - (((i + j)T) + (-(i + j)T) - 2(\mathcal{O})) \\
& = (i + j)(T + T') - (i + j)(T') - ((i + j)T) + (\mathcal{O}) \\
& = f'_{i+j}.
\end{aligned}$$

□

Ce théorème donne une formule récursive avec comme conditions initiales : $f_0 = 1$ et $f_1 = \frac{V_{T+T'}}{L_{T,T'}}$. La valeur de f_1 vient du fait que $\mathrm{div}(f_1) = (T + T') - (T') - (T) + (\mathcal{O})$.

On rappelle que le couplage à calculer est (11.3) :

$$e(S, T) = \frac{f_T(S + S')f_S(T')}{f_T(S')f_S(T + T')}.$$

L'algorithme suivant calcule la valeur $f_T(S')$.

La Ligne 7 de l'Algorithme 14 peut être remplacée grâce à la formule :

$$f_{i+1} = f_i.L_{T+T',iT}/L_{T',(i+1)T},$$

par la ligne $f \leftarrow f.L_{T+T',Z}(S')/L_{T',Z+T}(S')$. On évite ainsi le calcul et le stockage de f_1 tout au long de l'algorithme. Cette technique n'est avantageuse que lorsque le poids de Hamming de m est très faible car le calcul de lignes est bien plus coûteux que le calcul de verticales.

Algorithme 14: Algorithme de Miller pour le couplage de Weil

Données : Un entier $m = (m_n \dots m_1 m_0)_2$. Des points $T \in E[m]$ et $S' \in E(\overline{K})$
Résultat : $f = f_T(S')$

```
1 Z ← T
2 f ← f_1
3 pour i ← n − 1 a 0 faire
4 |   f ← f^2.T_Z(S')/V_{2Z}(S')
5 |   Z ← 2Z
6 |   si m_i = 1 alors
7 |   |   f ← f.f_1.L_{Z,T}(S')/V_{Z+T}(S')
8 |   |   Z ← Z + T
9 retourner f
```

On obtient bien à la fin de l'algorithme $f = f_T(S')$ et $Z = mT = \mathcal{O}$. On peut très facilement modifier l'algorithme pour calculer à la fois $f_T(S')$ et $f_T(S + S')$. Mais on est obligé de le relancer une seconde fois pour calculer $f_S(T')$ et $f_S(T + T')$.

Des travaux récents de Boxall et al. [BEML10] proposent une variante à l'algorithme de Miller qui n'est plus basée sur le théorème 11.1.4. Les auteurs proposent le lemme suivant :

Lemme 11.1.1 ([BEML10, Lemme 2])**.** *Soit $l_{i,j}$ la ligne passant par les points iT et jT. Pour deux entiers i et j, à une multiplicité près, on a :*

$$f_{i+j} = \frac{1}{f_{-s} f_{-j} l_{-i,-j}}.$$

Ce lemme est valable pour tous types de couplages, notamment Weil et Tate. L'algorithme de Miller modifié qui en résulte apporte des performances intéressantes.

11.1.5 Définition simplifiée

On rappelle la définition du couplage de Weil décrite ci-dessus (11.3) :

$$e(S,T) = \frac{f_T(S + S') f_S(T')}{f_T(S') f_S(T + T')}.$$

La simplification consiste à considérer le point $T' = \mathcal{O}$. On obtient alors la formule :

$$e(S,T) = \frac{f_T(S + S')}{f_T(S') f_S(T)},$$

où la fonction f_T a désormais pour diviseur $\mathrm{div}(f_T) = m(T) - m(\mathcal{O})$ et la fonction f_S a $\mathrm{div}(f_S) = m(S + S') - m(S')$.

Dans la définition originale, le choix de la fonction f_S peut être variable à une constante près car la différence s'annule lors de la division dans le couplage. La définition simplifiée

oblige à définir avec plus de précision cette fonction f_S. À chaque tour de la boucle de l'algorithme de Miller, la fonction est directement évaluée en un point. Lors du calcul de $f_S(T)$, on construit les lignes et tangentes telles que le coefficient en y soit un. De même pour les verticales, on fait en sorte que le coefficient en x soit un. On peut vérifier que, de cette manière, $f_S(\mathcal{O})$ est alors égal à 1. La simplification proposée est alors bien valide.

La même méthode peut être utilisée avec $S' = \mathcal{O}$, on obtient :

$$e(S,T) = \frac{f_T(S)f_S(T')}{f_S(T+T')},$$

avec des fonctions f_T et f_S définies correctement, comme précédemment.

On ne peut pas, à la fois, fixer $T' = \mathcal{O}$ et $S' = \mathcal{O}$ car les fonctions f_T et f_S auraient alors toutes les deux des pôles en $\mathcal{O}$, le couplage ne pourrait donc pas être évalué en $\mathcal{O}$.

11.2 Couplage de Tate

Il existe un deuxième type de couplage, appelé couplage de Tate (ou Tate-Lichtenbaum) [FR94, FMR99]. Il est le plus utilisé en cryptographie à ce jour, premièrement car, pour le calculer, il suffit de lancer une fois l'algorithme de Miller et deuxièmement car il possède le plus d'optimisations qui le rendent encore plus rapide comparé à Weil. De plus, le couplage de Tate peut être utilisé dans certains cas où celui de Weil ne s'applique pas. Finalement, le deuxième paramètre de Tate doit seulement être un représentant d'un sous-ensemble de points et non un point de m-torsion comme pour Weil.

11.2.1 Définition

On rappelle quelques définitions nécessaires pour le couplage de Tate.

11.2.1.1 Généralités

Soit E une courbe elliptique définie sur un corps $K = \mathbb{F}_q$. Soit m un entier premier à $\mathrm{car}(K)$ tel que E contient au moins un point d'ordre m. On note $\mu_m = \left\{x \in \overline{K} \mid x^m = 1\right\}$ l'ensemble des racines m-ième de l'unité dans $\overline{K}$. Soit $E(K)[m] = \{P \in E(K) \mid mP = \mathcal{O}\}$ les éléments de m-torsion appartenant à $E(K)$.

Soit $mE(K) = \{mP \mid P \in E(K)\}$ l'ensemble des points de $E(K)$ multipliés par m. On peut ainsi partitionner les points de $E(K)$ en m sous-ensembles de cette forme. Si on suppose m premier (on rappelle qu'il y a alors m^2 points de m-torsion) :

- soit $E[m] \subseteq E(K)$, alors il y a m points de m-torsion dans chaque sous-ensemble,
- soit $E[m] \not\subseteq E(K)$, alors chaque sous-ensemble a un seul point de m-torsion (Exemple 11.2.1, ci-dessous).

On note $E(K)/mE(K)$ le groupe quotient formé de tous ces sous-ensembles.

Exemple 11.2.1. *On reprend la courbe $E/\mathbb{F}_{11}$ utilisée dans la section 10.3.5 avec son tableau de points (voir Table 10.1). Soit $m = 3$, on a $E[3] \not\subseteq E(\mathbb{F}_{11})$. On partitionne $E(\mathbb{F}_{11})$ en sous-ensembles :*

$$3\mathcal{O} = 3P_8 = 3P_9 = \mathcal{O}, \qquad 3P_1 = 3P_2 = 3P_3 = P_1$$
$$3P_4 = 3P_7 = 3P_{11} = P_6, \qquad 3P_5 = 3P_6 = 3P_{10} = P_7.$$

On note $C_{\mathcal{O}} = \{\mathcal{O}, P_1, P_6, P_7\}$ le sous-ensemble qui contient le point $\mathcal{O}$. En ajoutant aux éléments de $C_{\mathcal{O}}$ un autre point, par exemple P_2, on obtient $C_{P_2} = \{P_2, P_8, P_{11}, P_5\}$. De même en ajoutant P_3, on obtient $C_{P_3} = \{P_3, P_9, P_4, P_{10}\}$. Chaque sous-ensemble $C_{\mathcal{O}}, C_{P_2}$ et C_{P_3} n'a qu'un seul point de 3-torsion, soit respectivement $\mathcal{O}$, P_8 et P_9.

Le groupe quotient est donc $E(\mathbb{F}_{11})/3E(\mathbb{F}_{11}) = \{C_{\mathcal{O}}, C_{P_2}, C_{P_3}\}$.

11.2.1.2 Définition du couplage de Tate

On peut maintenant définir le couplage de Tate. Soit K_0 la plus petite extension de corps de K contenant toutes les racines m-ième de l'unité. On note :

$$(K_0^*)^m = \{u^m \mid u \in K_0^*\}.$$

On a alors que $(K_0^*)^m$ est un sous-groupe de K_0^* et que les groupes $K_0^*/(K_0^*)^m$ et μ_m sont isomorphes. Soit $P \in E(K_0)[m]$ et $Q \in E(K_0)/mE(K_0)$, Q est donc un représentant d'une classe d'équivalence de $E(K_0)/mE(K_0)$. Comme $mP = \mathcal{O}$, on peut définir une fonction f_P telle que $\mathrm{div}(f_P) = m(P) - m(\mathcal{O})$. Soit $D_Q \in \mathrm{Div}^0(E)$ tel que $D_Q \sim (Q) - (\mathcal{O})$ et tel que le support de D_Q et $\mathrm{div}(f_P)$ soient distincts, ainsi $f_P(D_Q) \neq 0$ et $f_P(D_Q) \in K_0^*$.

On définit alors le couplage de Tate comme :

$$\langle ., . \rangle_m : E(K_0)[m] \times E(K_0)/mE(K_0) \to K_0^*/(K_0^*)^m$$

$$\langle P, Q \rangle_m = f_P(D_Q).$$

On remarque que la sortie de ce couplage n'est pas un élément unique mais une classe d'équivalence de $K_0^*/(K_0^*)^m$. Par exemple si $a, b \in K_0$, on dit que $a \equiv b$ si et seulement si il existe $c \in K_0$ tel que $a = b.c^m$. Cela peut poser un problème dans des applications cryptographiques, on cherche donc à ce que le couplage donne un représentant unique pour une paire d'entrées données. En notant $K_0 = \mathbb{F}_{q^k}$, on peut mettre le résultat à la puissance $(q^k - 1)/m$ pour enlever toutes les puissances m-ième et ne donner qu'une racine m-ième de l'unité dans K_0. En effet, pour tout $a \in \mathbb{F}_{q^k}$, on a $a^{(q^k-1)} = 1$. Donc en élevant à la puissance $(q^k - 1)/m$, on obtient bien des racines m-ième de l'unité.

11.2.1.3 Définition alternative du couplage de Tate

On a la définition alternative du couplage de Tate suivante :

$$\langle ., . \rangle'_m : E(K_0)[m] \times E(K_0)/mE(K_0) \to \mu_m$$

$$\langle P, Q\rangle'_m = f_P(D_Q)^{(q^k-1)/m}.$$

On utilisera, sauf mention contraire, cette définition alternative du couplage de Tate comme définition principale par la suite. De plus, lorsque le couplage est de façon évidente lié à m, on le notera simplement $\langle ., . \rangle$.

Théorème 11.2.1. *Pour tout $P \in E(K_0)[m]$ et tout $Q \in mE(K_0)$,*

$$\langle P, Q\rangle \in (K_0^*)^m,$$

c.-à-d., le couplage de Tate est bien défini.

Démonstration. Soit $P \in E(K_0)[m]$, $Q \in mE(K_0)$ et une fonction $f \in K_0(E)$ telle que $\mathrm{div}(f) = m(P) - m(\mathcal{O})$. Il existe un point $Q' \in E(K_0)$ tel que $mQ' = Q$. Soit $D \sim (Q) - (\mathcal{O})$ et $D' \sim (Q') - (\mathcal{O})$ deux diviseurs de supports disjoints de celui de $\mathrm{div}(f)$. Alors :

$$D - mD' \sim (Q) - (\mathcal{O}) - m(Q') + m(\mathcal{O}) = (mQ') - m(Q') + (m-1)(\mathcal{O}),$$

est clairement principal, donc $D \sim mD'$. On écrit alors :

$$\langle P, Q\rangle = f(D) \equiv f(mD') = f(D')^m \in (K_0^*)^m.$$

□

Grâce à ce théorème, on remarque que la sortie du couplage de Tate ne dépend pas du choix du second argument, c.-à-d. du choix de l'élément appartenant à un certain sous-ensemble. Chaque élément de $mE(K_0)$ va être envoyé sur $(K_0^*)^m$ et ainsi disparaître dans $K_0/(K_0^*)^m$. On peut donc prendre comme second argument n'importe quel point de K_0, de n'importe quel ordre, alors qu'avec le couplage de Weil il faut obligatoirement que le second argument Q satisfasse $mQ = \mathcal{O}$.

Théorème 11.2.2. *La valeur de sortie du couplage de Tate ne dépend pas du choix de la fonction rationnelle f_P, ni du diviseur D_Q.*

Démonstration. La fonction f_P est définie à une constante près et d'après le Lemme 10.3.1, comme D_Q est de degré 0, celle-ci n'influe pas sur l'évaluation de f en D_Q.

Soit $\mathrm{div}(f_P) = m(P) - m(\mathcal{O})$ et $D_1 \sim D_2 \sim (Q) - (\mathcal{O})$ tel que le support de D_1 et D_2 soit différent de celui de $\mathrm{div}(f_P)$. On peut écrire $D_1 = D_2 + \mathrm{div}(g)$ avec $\mathrm{supp}(\mathrm{div}(g)) \cap \mathrm{supp}(\mathrm{div}(f_P)) = \emptyset$. D'où :

$$\begin{aligned} f_P(D_1) &= f_P(D_2 + \mathrm{div}(g)) = f_P(D_2) f_P(\mathrm{div}(g)) = f_P(D_2) g(\mathrm{div}(f_P)) \\ &= f_P(D_2) g(m(P) - m(\mathcal{O})) = f_P(D_2) g((P) - (\mathcal{O}))^m \equiv f_P(D_2). \end{aligned}$$

□

11.2.2 Propriétés

Théorème 11.2.3. *Le couplage de Tate satisfait aux propriétés suivantes :*

1. *Pour tout $P, P_1, P_2 \in E(K_0)[m]$ et $Q, Q_1, Q_2 \in E(K_0)$,*

$$\langle P_1 + P_2, Q\rangle = \langle P_1, Q\rangle \langle P_2, Q\rangle ,$$

et

$$\langle P, Q_1 + Q_2\rangle = \langle P, Q_1\rangle \langle P, Q_2\rangle ,$$

(Bilinéarité).

2. *Pour chaque $P \in E(K_0)[m] \setminus \{\mathcal{O}\}$, il existe un point $Q \in E(K_0)[m]$ tel que $\langle P, Q\rangle \notin (K_0^*)^m$ (Non-dégénérée).*

Démonstration. 1. Soit $P, P_1, P_2 \in E(K_0)[m]$ et $Q, Q_1, Q_2 \in E(K_0)$. Soit $D \sim (Q) - (\mathcal{O})$ et g une fonction rationnelle telle que $\mathrm{div}(g) = m(P_1 + P_2) - m(\mathcal{O})$. On écrit :

$$\langle P_1 + P_2, Q\rangle = g(D),$$

avec $\mathrm{supp}(\mathrm{div}(g)) \cap \mathrm{supp}(D) = \emptyset$. Soit g_1 et g_2 deux fonctions telles que :

$$\mathrm{div}(g_1) = m(P_1) - m(\mathcal{O}) \quad \text{et} \quad \mathrm{div}(g_2) = m(P_2) - m(\mathcal{O}).$$

Soit A le diviseur $A = (P_1 + P_2) - (P_1) - (P_2) + (\mathcal{O})$ qui est clairement principal. Soit la fonction h telle que $\mathrm{div}(h) = A$. Alors on peut écrire $g = g_1 g_2 h^m$. D'où :

$$\langle P_1 + P_2, Q\rangle = g(D) = g_1(D) g_2(D) h^m(D) \equiv \langle P_1, Q\rangle \langle P_2, Q\rangle .$$

Il reste à prouver la linéarité en son second argument. Soit $D \sim (Q_1 + Q_2) - (\mathcal{O})$ et g une fonction rationnelle telle que $\mathrm{div}(g) = m(P) - m(\mathcal{O})$. On écrit :

$$\langle P, Q_1 + Q_2\rangle = g(D),$$

avec $\mathrm{supp}(\mathrm{div}(g)) \cap \mathrm{supp}(D) = \emptyset$. Soit D_1, D_2 deux diviseurs tels que :

$$D_1 \sim (Q_1) - (\mathcal{O}) \quad \text{et} \quad D_2 \sim (Q_2) - (\mathcal{O}),$$

avec le support de D_1 et D_2 disjoint de celui de $\mathrm{div}(g)$. Alors :

$$D - D_1 - D_2 \sim (Q_1 + Q_2) - (Q_1) - (Q_2) + (\mathcal{O}),$$

est principal, donc $D \sim D_1 + D_2$. On écrit donc :

$$\langle P, Q_1 + Q_2\rangle = g(D) \equiv g(D_1 + D_2) = g(D_1) g(D_2) = \langle P, Q_1\rangle \langle P, Q_2\rangle .$$

2. La non-dégénérescence est prouvée dans [FR94, Section 2] ou [Hes04, Théorème 3].

□

Propriétés du couplage de Tate défini sur des corps finis. Soit $K = \mathbb{F}_q$ et $E/\mathbb{F}_q$ la courbe elliptique définie sur $\mathbb{F}_q$. Soit m un entier positif, premier à q, tel que $E(\mathbb{F}_q)$ contienne un point d'ordre m. En cryptographie, on prend m un très grand nombre premier tel que $m \mid E(\mathbb{F}_q)$. Soit k le plus petit entier positif tel que $m \mid (q^k - 1)$. On appelle k le degré de plongement de E relatif à m (voir section 10.6). En reprenant nos notations ci-dessus, on a alors $K_0 = \mathbb{F}_{q^k}$, la plus petite extension de corps contenant toutes les racines m-ième de l'unité.

Contrairement au couplage de Weil, Tate n'a pas la propriété d'identité : $e_m(P, P) = 1$, $\forall\, P \in E[m]$. En effet, le résultat du couplage de Tate étant dans $\mathbb{F}_{q^k}/(\mathbb{F}_{q^k}^*)^m$, $\langle P, P\rangle$ n'est pas nécessairement l'identité (mais peut être une puissance m-ième). En cryptographie, on utilise très souvent $P \in E(\mathbb{F}_q)[m]$, c.-à-d. P un point de m-torsion appartenant au corps de base, et m premier à q. Le lemme suivant s'applique alors.

Lemme 11.2.1 (voir [Gal01])**.** *Soit $P \in E(\mathbb{F}_q)[m]$, $P \neq \mathcal{O}$ et m premier à q. Alors pour que le couplage de Tate $\langle P, P\rangle$ ne soit pas trivial, il est nécessaire que $k = 1$.*

Lemme 11.2.2. *Soit $P \in E(\mathbb{F}_q)[m]$, $P \neq \mathcal{O}$ et m premier. Soit $k > 1$ et $Q \in E(\mathbb{F}_{q^k})[m]$ un point de m-torsion indépendant de P. Alors $\langle P, Q\rangle \not\equiv 1$.*

Démonstration. On suppose que $\langle P, Q\rangle \equiv 1$. On rappelle que chaque sous-ensemble de $E(\mathbb{F}_{q^k})/mE(\mathbb{F}_{q^k})$ contient un point de m-torsion. Soit $R \in E(\mathbb{F}_{q^k})$ et R' un point de m-torsion dans le même sous-ensemble que R tel que $R = R' + mR''$ pour un $R'' \in E(\mathbb{F}_{q^k})$. Alors $\langle P, R\rangle \equiv \langle P, R'\rangle$. Comme $E[m]$ est engendré par P et Q, car ils sont linéairement indépendants l'un de l'autre, on peut écrire le point de m-torsion $R' = aP + bQ$ pour certains $1 \leq a, b \leq m - 1$. D'où :

$$\langle P, R\rangle \equiv \langle P, R'\rangle = \langle P, aP + bQ\rangle \equiv \langle P, P\rangle^a \equiv \langle P, Q\rangle^b \equiv 1,$$

grâce au Lemme 11.2.1 pour la dernière équivalence. Mais ce résultat est en contradiction avec la propriété de non-dégénérescence du couplage de Tate. Donc $\langle P, Q\rangle \not\equiv 1$. □

11.2.3 Algorithme de Miller appliqué à Tate

On rappelle que l'on cherche à calculer $\langle P, Q\rangle = f_P(D_Q)^{(q^k-1)/m}$ et, en particulier, la valeur $f_P(D_Q)$ grâce à l'algorithme de Miller. On a $\mathrm{div}(f_P) = m(P) - m(\mathcal{O})$. On se retrouve dans une situation similaire à celle de la section 11.1.4 avec $T' = \mathcal{O}$.

Théorème 11.2.4 (Formule de Miller pour Tate)**.** *Soit $P \in E(K)$ et f_i une fonction rationnelle telle que* $\mathrm{div}(f_i) = i(P) - (iP) - (i-1)(\mathcal{O})$ *pour $i \in \mathbb{Z}$. Alors :*

$$f_{i+j} = f_i . f_j . \frac{L_{iP,jP}}{V_{(i+j)P}}.$$

Démonstration. On rappelle les diviseurs d'une ligne $L_{iP,jP}$ et d'une verticale $V_{(i+j)P}$:

$$\mathrm{div}(L_{iP,jP}) = (iP) + (jP) + (-(i+j)P) - 3(\mathcal{O}),$$

$$\mathrm{div}(V_{(i+j)P}) = ((i+j)P) + (-(i+j)P) - 2(\mathcal{O}).$$

D'où :

$$\mathrm{div}\left(\frac{L_{iP,jP}}{V_{(i+j)P}}\right) = \mathrm{div}(L_{iP,jP}) - \mathrm{div}(V_{(i+j)P}) = (iP) + (jP) - ((i+j)P) - (\mathcal{O}).$$

D'après la définition de f_i, on a :

$$\begin{aligned}\mathrm{div}(f_{i+j}) &= (i+j)(P) - ((i+j)P) - (i+j-1)(\mathcal{O}) \\ &= i(P) - (iP) - (i-1)(\mathcal{O}) \\ &+ j(P) - (jP) - (j-1)(\mathcal{O}) \\ &+ (iP) + (jP) - ((i+j)P) - (\mathcal{O}) \\ &= \mathrm{div}(f_i) + \mathrm{div}(f_j) + \mathrm{div}(L_{iP,jP}) - \mathrm{div}(V_{(i+j)P}).\end{aligned}$$

Donc, $f_{i+j} = f_i.f_j.\frac{L_{iP,jP}}{V_{(i+j)P}}$. □

On a $f_m = f_P$, la fonction que l'on cherche à calculer. On remarque que $\mathrm{div}(f_0) = \mathrm{div}(f_1) = 1$. Au lieu d'utiliser directement la formule de Miller, on utilise plutôt en pratique les relations suivantes :

$$f_{2i} = f_i^2.\frac{T_{iP}}{V_{2iP}}, \quad \text{et} \quad f_{i+1} = f_i.\frac{L_{iP,P}}{V_{(i+1)P}}.$$

On doit évaluer la fonction f_P au diviseur $D_Q \sim (Q) - (\mathcal{O})$. Soit $D' = (Q+Q') - (Q')$ pour un $Q' \in E(K_0)$ tel que $Q' \notin \{P, P-Q, -Q, \mathcal{O}\}$. Le couplage de Tate peut alors se réécrire comme :

$$\langle P, Q\rangle = \frac{f_P(Q+Q')}{f_P(Q')}.$$

Comme dans le cas de Weil, le but est de calculer f_P tout en l'évaluant au fur et à mesure. L'algorithme de Miller associé à Tate est proche de celui pour Weil. L'Algorithme 15 calcule $f_P(Q')$. On obtient bien à la fin de l'algorithme $f = f_P(Q')$ et $Z = mP = \mathcal{O}$. On peut très facilement modifier l'algorithme pour calculer à la fois $f_P(Q')$ et $f_P(Q+Q')$.

11.2.4 Définition simplifiée

Une simplification du couplage de Tate est proposée par Barreto et al. dans [BKLS02] puis généralisée dans [BLS04a].

Soit $P \in E(\mathbb{F}_q)[m]$ et $Q \in E(\mathbb{F}_{q^k})$ deux points linéairement indépendants. On rappelle que le couplage de Tate est $\langle P, Q\rangle = f_P(D_Q)^{(q^k-1)/m}$ avec $\mathrm{div}(f_P) = m(P) - m(\mathcal{O})$ et $D_Q \sim (Q) - (\mathcal{O})$.

Soit d un entier tel que $d \mid k$ et $d < k$.

Lemme 11.2.3 ([BLS04a, Lemme 5]). *$q^d - 1$ est un facteur de $q^k - 1$.*

Algorithme 15: Algorithme de Miller pour le couplage de Tate

Données : Un entier $m = (m_n \dots m_1 m_0)_2$. Des points $P \in E[m]$ et $Q' \in E(\overline{K})$
Résultat : $f = f_P(Q')$

```
Z ← P
f ← 1
pour i ← n − 1 a 0 faire
    f ← f².T_Z(Q')/V_2Z(Q')
    Z ← 2Z
    si m_i = 1 alors
        f ← f.L_Z,P(Q')/V_Z+P(Q')
        Z ← Z + P
retourner f
```

Lemme 11.2.4 ([BLS04a, Corollaire 2])**.** *On peut multiplier $f_P(D_Q)$ par n'importe quel $x \in \mathbb{F}_{q^d}$ non nul sans affecter la valeur de sortie du couplage.*

Théorème 11.2.5 ([BLS04a, Théorème 1])**.** $\langle P, Q\rangle = f_P(Q)^{(q^k-1)/m}$.

Démonstration. Soit $R \notin \{\mathcal{O}, -P, Q, Q-P\}$ un point de la courbe. Soit f'_P une fonction de diviseur $\mathrm{div}(f'_P) = m(P+R) - m(R) \sim \mathrm{div}(f_P)$ telle que $\langle P, Q\rangle = f'_P((Q)-(\mathcal{O}))^{(q^k-1)/m}$. Comme $P \in E(\mathbb{F}_q)[m]$, il a ses coordonnées dans $\mathbb{F}_q$ et comme f'_P n'a pas de zéro ou pôle en $\mathcal{O}$, on a que $f'_P(\mathcal{O}) \in \mathbb{F}_q^*$. Donc $f'_P((Q)-(\mathcal{O})) = f'_P(Q)/f'_P(\mathcal{O})$. D'après les Lemmes 11.2.3 et 11.2.4, on en déduit que $f'_P(\mathcal{O})^{(q^k-1)/m}$ n'a aucun effet sur le résultat du couplage, donc $\langle P, Q\rangle = f'_P(Q)^{(q^k-1)/m}$.

On a $\mathrm{div}(f'_P) = m((P+R)-(R)) = m((P)-(\mathcal{O})+\mathrm{div}(g)) = \mathrm{div}(f_P) + m\,\mathrm{div}(g)$, pour une certaine fonction rationnelle g, car $(P+R)-(R) \sim (P)-(\mathcal{O})$. Donc $f'_P = f_P.g^m$. Comme Q n'est ni un zéro, ni un pôle de f'_P ou f_P par le choix de R, $g(Q) \in \mathbb{F}_{q^k}^*$ est bien définie. D'où, $f'_P(Q)^{(q^k-1)/m} = f_P(Q)^{(q^k-1)/m}.g(Q)^{(q^k-1)} = f_P(Q)^{(q^k-1)/m}$. □

L'utilisation de cette définition simplifiée combinée à l'algorithme de calcul du couplage de Tate (Algorithme 15) est souvent appelée BKLS dans la littérature. Cette méthode, qui s'applique pour tout types de courbes, a l'avantage de diviser par deux le temps de calcul d'un couplage.

11.3 Sécurité des couplages

Les couplages ont d'abord été utilisés en cryptographie comme outil d'attaque. En 1993, Menezes, Okamoto et Vanstone (MOV) [MOV93] ont découvert une méthode qui permet de transformer le groupe de points d'une courbe elliptique sur $\mathbb{F}_q$ en groupe multiplicatif sur une extension de corps $\mathbb{F}_{q^k}$, où k est le degré de plongement défini dans la section 10.6. Le meilleur algorithme permettant de résoudre le problème du logarithme discret sur un

groupe de points d'une courbe elliptique est Pollard-ρ qui s'exécute en temps exponentiel. Le même problème sur un groupe multiplicatif dans un corps fini s'attaque avec des algorithmes beaucoup plus spécifiques et donc plus performants, ils s'exécutent en temps sous-exponentiel. L'algorithme MOV (et de façon similaire celui de Frey-Rück [FR94]) permet donc de transformer un problème se résolvant en temps exponentiel en un problème sous-exponentiel.

Pour éviter ce type d'attaques, il faut donc que le corps $\mathbb{F}_{q^k}$ soit assez grand pour résister à une attaque sous-exponentielle. Les auteurs de [BK98] ont montré que, pour une courbe choisie au hasard, il existait une très forte probabilité pour que k soit trop grand pour réaliser ce type d'attaque. En effet, en général, k est de l'ordre de q pour une courbe tirée au hasard.

La condition demandée dans cette section est clairement en contradiction avec le fait d'avoir une extension de corps la plus petite possible afin que les calculs soit plus rapides. Il faut donc trouver un compromis entre sécurité et rapidité de calculs.

On reprend [KM05, Table 1] qui illustre les relations entre taille du corps de base, de l'extension de corps nécessaire et du niveau de sécurité demandé. On note b_m la taille en bits minimum (pour un niveau de sécurité donné) du sous-groupe de points d'ordre premier m. On note b_{q^k} la taille en bits minimum de l'extension de corps nécessaire.

Niveau de sécurité (bits)	b_m (bits)	b_{q^k} (bits)	b_{q^k}/b_m
80	160	1024	$6,4$
128	256	3072	12
192	384	8192	$21\frac{1}{3}$
256	512	15360	30

TABLE 11.1 – Relation entre la taille du sous-groupe de points et la taille de l'extension de corps nécessaire.

Pour atteindre le nombre de bits minimum écrits dans la Table 11.1, il faut noter $b_{q^k}/b_m = \rho.k$ où $\rho = \log(q)/\log(m)$, c.-à-d. le rapport, en nombre de bits, entre la taille du corps de base et la taille du sous-groupe de points utilisé. En général, il vaut mieux que ρ se rapproche le plus possible de 1. On gagne, en effet, en bande passante dans les protocoles (les points de la courbe sont codés sur moins de bits) et en vitesse pour les calculs sur la courbe elliptique.

Exemple 11.3.1. *On se place dans le cas d'un niveau de sécurité de* 80 *bits. On considère deux courbes :*

- *soit la courbe A définie sur un corps à* 340 *bits telle que* $\rho = 2, b_m = 170, k = 3$,
- *soit la courbe B définie sur un corps à* 170 *bits telle que* $\rho = 1, b_m = 170, k = 6$.

Ces deux courbes sont équivalentes d'un point de vue niveau de sécurité. Néanmoins, les calculs sur la courbe elliptique B sont bien plus rapides que ceux sur A. La courbe B doit être préférée si on utilise un protocole de signatures courtes [BLS04b] car les points sont codés sur moins de bits qu'avec la courbe A. D'un autre côté, utiliser la courbe B demande

de travailler dans une extension de degré six ce qui est, a priori, bien plus lent que les calculs dans l'extension de degré trois de la courbe A. De plus, les courbes avec de grands degrés de plongement sont difficiles à construire, voire impossible pour certains degrés, pour le moment.

Chapitre 12

Constructions efficaces de couplages

Nous présentons dans cette section, quelques unes des constructions de couplages les plus efficaces proposées dans la littérature. Certaines constructions sont intéressantes dans des cas très spécifiques alors que d'autres peuvent remplacer le couplage Tate dans la majorité des cas. La boucle de l'algorithme de Miller représente une grande partie du temps de calcul d'un couplage. Ainsi, de nombreuses propositions tentent d'optimiser ce nombre de tour de boucle afin d'améliorer les performances.

12.1 Couplage eta

Le couplage eta [BGhS07] est une généralisation du travail de Duursma et Lee [DL03]. On ne considère, dans cette partie, que des courbes elliptiques supersingulières avec un degré de plongement pair et un endomorphisme de courbes spécifique : une *distortion map* ψ (voir section 10.4). Seules les courbes supersingulières définies sur des corps de caractéristique 2 ou 3 sont intéressantes étant donné qu'elles sont les seules à avoir un degré de plongement supérieur à 2 ainsi que des *distortion maps*.

Soit E une courbe elliptique définie sur K. Soit $n = \#E(K)$ le cardinal du groupe de points. Soit r le plus grand premier divisant n. Soit P et Q deux diviseurs d'ordre divisant r. D'après la section 11.2.1, on peut écrire le couplage de Tate, sans l'exponentiation finale, comme :

$$\langle P, Q \rangle_r = f_P(Q) = f_{r,P}(Q).$$

La technique de Barreto et al. [BGhS07] consiste à réduire le nombre de tours de l'algorithme de Miller afin d'accélérer le calcul. Soit $T \in \mathbb{Z}$, le couplage eta est défini comme :

$$\eta_T = f_{T,P}(\psi(Q)).$$

Cette définition ne produit un couplage bilinéaire non dégénéré que pour certaines valeurs de T, notamment pour $T = q - r$ comme proposé dans [BGhS07]. Les auteurs retrouvent η, le couplage de Duursma et Lee [DL03], avec la valeur $T = q$. Comme $T = q - r = \pm t - 1$ où t est la trace de Frobenius de la courbe, le couplage η_T peut être calculé avec une boucle de Miller à peu près deux fois plus courte que le couplage η.

12.2 Couplage ate et ate tordu

12.2.1 Couplage ate

Dans [HSV06], Hess et al. ont étendu l'idée de Barreto et al. [BGhS07] aux courbes ordinaires.

Un couplage est traditionnellement une fonction bilinéaire de la forme :

$$\hat{e} : G_1 \times G_2 \to G_T,$$

où G_1 et G_2 sont deux groupes notés additivement et G_T est un groupe multiplicatif.

Soit π_q l'endomorphisme de Frobenius défini tel que (voir section 10.5) :

$$\begin{aligned} \pi_q : E &\to E \\ (x, y) &\mapsto (x^q, y^q). \end{aligned}$$

Soit les groupes G_1 et G_2 tels que :

- $G_1 = E[r] \cap \ker(\pi_q - [1])$,
- $G_2 = E[r] \cap \ker(\pi_q - [q])$.

Les auteurs de [HSV06] proposent alors un couplage appelé ate (eta écrit à l'envers) qui est défini sur $G_2 \times G_1$. Soit $T = t - 1$, où t est la trace de Frobenius de la courbe. Soit $N = \gcd(T^k - 1, q^k - 1)$, on peut écrire $T^k - 1 = LN$. Pour $Q \in G_2$ et $P \in G_1$, le couplage ate est défini tel que :

$$\begin{aligned} a_T : G_2 \times G_1 &\to G_T \\ (Q, P) &\mapsto f_{T,Q}(P)^{c_T(q^k-1)/N}, \end{aligned}$$

avec $c_T = \sum_{i=0}^{k-1} T^{k-1-i} q^i \equiv kq^{k-1} \bmod r$. Le couplage a_T est non dégénéré si $m \nmid L$. On calcule $f_{T,Q}(P)$ grâce à l'algorithme de Miller avec une boucle de longueur $\lfloor \log_2 |T| \rfloor$. Si $q \approx r$ et que la taille de la trace est en moyenne $\sqrt{q}$, le couplage ate peut se calculer avec une boucle deux fois plus courte que pour le couplage de Tate.

12.2.2 Couplage ate tordu

Soit E une courbe elliptique de degré de plongement k admettant un twist E' de degré d. On note $m = \gcd(k, d)$ et $e = k/m$. Soit $G_1 = E[r] \cap \ker(\pi_q - [1])$ et $G_2 = E[r] \cap \ker([\xi_m]\pi_q^e - [1])$ où ξ_m est la racine m-ième de l'unité telle que $[\xi_m] : (x, y) \mapsto (\xi_m^2 x, \xi_m^3 y)$. Le couplage ate tordu est défini sur $G_1 \times G_2$. Pour $P \in G_1$ et $Q \in G_2$, on définit le couplage ate tordu tel que :

$$\begin{aligned} a_T^{\text{twist}} : G_1 \times G_2 &\to G_T \\ (P, Q) &\mapsto f_{T^e,P}(Q)^{c_{T^e}(q^k-1)/N}, \end{aligned}$$

avec $c_{T^e} = \sum_{i=0}^{m-1} T^{e(m-1-i)} q^{ei} \equiv kq^{k-1} \bmod r$. Le couplage a_T^{twist} est non dégénéré si $r \nmid L$. Le couplage ate tordu n'est plus rapide que Tate que lorsque $|T^e| \leq r$. Néanmoins, on

remarque que le premier argument de ce couplage appartient à G_1, comme le couplage de Tate classique. On peut noter que l'algorithme de Miller effectue beaucoup d'opérations à partir de multiples du points en premier paramètre. Ainsi, il est préférable que celui-ci soit définit sur un corps où l'arithmétique est efficace ce qui est généralement le cas pour G_1.

12.2.3 Amélioration de ate et ate tordu

Définition 12.2.1 ([MKHO07, Théorème 1]). *Soit $r \geq 5$ l'ordre du sous-groupe premier de la courbe. Soit S un entier tel que $S \equiv q \bmod r$. Soit $N = \gcd(S^k - 1, q^k - 1)$ et $S^k - 1 = LN$. Pour $Q \in G_2$ et $P \in G_1$, le couplage ate amélioré est tel que :*

$$\begin{aligned} a_S : G_2 \times G_1 &\to G_T \\ (Q, P) &\mapsto f_{S,Q}(P)^{c_S(q^k-1)/N}, \end{aligned}$$

alors que le couplage de ate tordu amélioré est défini tel que :

$$\begin{aligned} a_S^{twist} : G_1 \times G_2 &\to G_T \\ (P, Q) &\mapsto f_{S,P}(Q)^{c_S(q^k-1)/N}, \end{aligned}$$

avec, dans les deux cas, $c_S = \sum_{i=0}^{k-1} S^{k-1-i} q^i \bmod N$. Les couplages sont non dégénérés si $r \nmid L$.

Dans [MKHO07], les auteurs proposent une amélioration des couplages ate et ate tordu en réduisant le nombre de tours de l'algorithme de Miller. Au lieu du paramètre $T = t - 1$ utilisé dans les propositions précédentes, ils utilisent un entier $S \equiv q \bmod r$. On peut trouver un entier S tel que les couplages a_S et a_S^{twist} soient plus performants. Soit le paramètre $\rho = \log q / \log r$ qui mesure la taille du corps de base relativement à la taille du sous-groupe d'ordre premier de la courbe. Lorsque $\rho \approx 2$, les versions améliorées de ate et ate tordu sont deux fois plus rapides que leurs versions originales. Dans tous les cas, ces versions améliorées sont au moins aussi rapides que le couplage de Tate.

Une dernière amélioration peut être apportée au couplage ate [Hes08, Corollaire 14] :

$$\begin{aligned} a_S : G_2 \times G_1 &\to G_T \\ (Q, P) &\mapsto f_{S,Q}(P)^{(q^k-1)/r}, \end{aligned}$$

avec $S^k \neq 1 \bmod r^2$. Cette amélioration s'applique de manière similaire au couplage ate tordu.

12.3 Couplage ate$_i$

Zhao et al. [ZZH08] proposent de réduire la taille de la boucle de l'algorithme de Miller dans le couplage ate en généralisant la construction.

Définition 12.3.1 ([ZZH08, Théorème 1]). *Soit E une courbe elliptique définie sur $\mathbb{F}_q$, t sa trace de Frobenius, r l'ordre du sous-groupe de points premier, k le degré de plongement et $T = t - 1$. Pour $T^i = (t-1)^i \equiv q^i \bmod r$ où $0 < i < k$, on note $T_i = T^i \bmod r$. Soit a_i le plus petit entier tel que $T_i^{a_i} \equiv 1 \bmod r$. Soit $N_i = \gcd(T_i^{a_i} - 1, q^k - 1)$ et $T_i^{a_i} - 1 = L_i N_i$. Pour $Q \in G_2$ et $P \in G_1$, le couplage ate$_i$ est défini comme :*

$$\begin{aligned} ate_i : G_2 \times G_1 &\to G_T \\ (Q, P) &\mapsto f_{T_i,Q}(P)^{(q^k-1)/N_i}, \end{aligned}$$

Le couplage est non dégénéré si $r \nmid L_i$.

Le couplage ate tordu peut être généralisé de la même manière.

La rapidité de calcul du couplage ate$_i$ dépend de la taille de T_i. Plus T_i est petit plus la boucle de l'algorithme de Miller est courte. Plus généralement, la famille de couplages ate se calcule plus rapidement si la trace de Frobenius t a une longueur en bits la plus courte possible. On introduit la valeur $\omega(E) = \frac{\log r}{\log |t|}$ [FST10, Définition 2.8]. Ainsi, plus la valeur $\omega(E)$ est grande plus le couplage sera rapide. Justement, [FST10, Proposition 2.9] donne une borne supérieure : $\omega(E) \leq \varphi(k)$, où $\varphi(k)$ est l'indicatrice d'Euler. En écrivant différemment la proposition, on obtient que la longueur de la boucle de l'algorithme de Miller d'un couplage de la famille ate peut être aussi petite que $\Lambda_{r,k} = \log(r^{1/\varphi(k)})$. Les couplages qui atteignent cette limite sont appelés couplages optimaux [Ver10]. Le couplage ate$_i$ n'atteint $\Lambda_{r,k}$ que pour quelques courbes.

12.4 Couplage R-ate

Le couplage R-ate [LLP09] est une généralisation du couplage ate$_i$ et donc de ate. Les auteurs réduisent encore la boucle de l'algorithme de Miller afin qu'elle soit jusqu'à deux à trois fois plus courte que ate$_i$. Le nom R-ate de ce couplage vient du fait qu'on peut le voir comme un ratio entre deux couplages ate$_i$. La définition du couplage R-ate est la suivante :

Définition 12.4.1. *Soit E une courbe elliptique définie sur $\mathbb{F}_q$, t sa trace de Frobenius, r l'ordre du sous-groupe de points premier, k le degré de plongement. Soit $T_i \equiv q^i \bmod r$ où $0 < i < k$. Soit a_i le plus petit entier tel que $T_i^{a_i} \equiv 1 \bmod r$. Soit $N_i = \gcd(T_i^{a_i} - 1, q^k - 1)$ et $T_i^{a_i} - 1 = L_i N_i$. Soit $c_{T_i} = \sum_{j=0}^{a_i-1} T_i^{a_i-1-j}(q^i)^j \bmod N_i$ et $M_i = (q^k - 1)/N_i$. Soit $A, B \in \mathbb{Z}$ tel que $A = aB + b$ pour $a, b, \in \mathbb{Z}$. On rappelle que le couplage ate$_i$ est défini par $f_{T_i,Q}(P)$ (voir Déf. 12.3.1). On note les couplages :*

$$e(Q, P)^{L_1} = f_{A,Q}(P)^{M_1}, \quad e(Q, P)^{L_2} = f_{B,Q}(P)^{M_2},$$

pour les entiers L_1, L_2, M_1 et M_2. Soit $M = lcm(M_1, M_2), d_1 = M/M_1, d_2 = M/M_2$ et $L = d_1 L_1 - a d_2 L_2$. Si $r \nmid L_i$, on peut définir le couplage R-ate tel que :

$$e(Q, P)^L = R_{A,B}(Q, P)^M,$$

avec

$$R_{A,B}(Q,P) = f_{a,BQ}(P).f_{b,Q}(P).G_{aBQ,bQ}(P),$$

où $G_{aBQ,bQ}$ est la ligne qui passe par les points aBQ et bQ divisée par la verticale par le point $(aB+b)Q$. Pour chaque valeur du couple (A,B), les valeurs des paramètres L et M sont définis par [LLP09, Corollaire 3.3].

12.5 Couplage optimal

Les travaux de Vercauteren [Ver10] et Hess [Hes08] généralisent l'idée des propositions précédentes : réduire la longueur de la boucle de l'algorithme de Miller à sa plus petite valeur. Vercauteren [Ver10] conjecture que la limite minimale de la longueur de boucle pour avoir un couplage non dégénéré sur courbes elliptiques est $\log_2 r/\varphi(k)$. Hess prouve cette conjecture [Hes08] et justifie ainsi le concept de couplage optimal, un couplage dont la longueur de boucle atteint cette limite.

Définition 12.5.1 ([Ver10, Définition 1])**.** *Soit $e : G_1 \times G_2 \to G_T$ un couplage bilinéaire non dégénéré avec $|G_1| = |G_2| = |G_T| = r$ et G_T de corps de définition $\mathbb{F}_{q^k}$. Le couplage e est appelé couplage optimal s'il peut être évalué avec au plus $(\log_2 r)/\varphi(k) + \epsilon(k)$ itérations de l'algorithme de Miller, où $\epsilon(k)$ est plus petit que $\log_2 k$.*

Le couplage ate optimal se construit grâce au théorème suivant.

Théorème 12.5.1 ([Ver10, Théorème 1])**.** *Soit $\lambda = mr$ avec $r \nmid m$. On écrit $\lambda = \sum_{i=0}^{l} c_i q^i$ alors,*

$$a_{[c_0,\ldots,c_l]} : (Q,P) \mapsto \left(\prod_{i=0}^{l} f_{c_i,Q}^{q^i}(P). \prod_{i=0}^{l} \frac{l_{s_{i+1}Q,c_i q^i Q}(P)}{v_{s_i Q}(P)} \right)^{(q^k-1)/r},$$

avec $s_i = \sum_{j=i}^{l} c_j q^j$, est un couplage bilinéaire. De plus, il est non dégénéré si

$$mkq^{k-1} \neq \frac{q^k-1}{r}. \sum_{i=0}^{l} i c_i q^{i-1} \bmod r.$$

12.6 Couplage Xate

Pour le couplage R-ate, Lee et al. [LLP09] cherchent $T_x = \sum c_i p^i$ avec des petits coefficients c_i où $T_x = (t-1)^x$. Vercauteren [Ver10] cherche un multiple $\lambda = mr$ avec $r \nmid m$ et de décomposition canonique en nombre q-adic $\lambda = \sum c_i q^i$ avec des petits coefficients c_i. Nogami et al. [NAS+08, SKNM08] utilisent une approche différente pour leur proposition : le couplage Xate (ou χ-ate).

Beaucoup de courbes elliptiques *pairing-friendly* intéressantes proviennent de familles spécifiques comme les courbes Barreto-Naehrig (BN) [BN06] ou les courbes de Freeman [Fre06]. Les paramètres de ces courbes sont donnés par des polynômes en une variable

entière notée χ. Ainsi les paramètres des courbes BN de degré de plongement 12 sont donnés par :

$$p(\chi) = 36\chi^4 - 36\chi^3 + 6\chi^2 - 6\chi + 1,$$
$$t(\chi) = 6\chi^2 + 1.$$

Alors que le nombre d'itérations de l'algorithme de Miller dans le couplage ate est $\log_2(t-1)$, Nogami et al. avec leur couplage Xate effectuent $\log_2(\chi)$ itérations. Ce couplage semble particulièrement bien adapté aux courbes BN. La première étape de la construction du couplage Xate consiste à trouver des petits coefficients c_i et d_j tels que $\sum_j d_j\chi^i = \sum_i c_i p^i$. Le couplage Xate est alors :

$$\zeta : (Q, P) \mapsto \hat{f}_{\chi,Q}(P)^{(p^k-1)/r},$$

où le calcul de $\hat{f}_{\chi,Q}$ dépend majoritairement du calcul de $f_{\chi,Q}$ qui atteint le minimum $\log_2(r/\varphi(k))$. Le poids de Hamming de χ affecte directement le calcul de $\hat{f}_{\chi,Q}$. Dans le cas précis des courbes BN de degré de plongement 12, le couplage Xate est jusqu'à deux fois plus rapide que ate.

12.7 Couplage omega

Inspiré par le travail de Gallant et al. [GLV01] et leur algorithme de multiplication scalaire utilisant des morphismes intéressants, Scott [Sco05b] se base sur une idée similaire pour proposer un couplage rapide. Pour cela, Scott propose une famille de courbes elliptiques possédant des morphismes intéressants : les courbes NSS (*Not-SuperSingular*). Ces courbes sont ordinaires, i.e. non-supersingulières, et définies sur $\mathbb{F}_q$ par :

$$E_1 : y^2 = x^3 + B, \quad \text{avec} \quad p \equiv 1 \bmod 3,$$
$$E_2 : y^2 = x^3 + Ax, \quad \text{avec} \quad p \equiv 1 \bmod 4,$$

avec p un grand nombre premier. Scott construit ces courbes avec un degré de plongement $k = 2$ afin d'avoir un couplage le plus rapide possible pour des niveaux de sécurité faible. Zhao et Zhang [ZZ08] proposent sur ces mêmes courbes l'utilisation d'un couplage qu'ils appellent couplage omega. Les auteurs reprennent les idées de Scott en se servant des morphismes rapidement calculables sur ces courbes pour construire leur proposition de couplage qui peut être vu comme une variante du couplage de Weil. La définition du couplage peut être généralisée à des courbes de type E_1 et E_2 ayant un degré de plongement pair. Celle qui est donnée ci-après est simplifiée au cas E_1 avec un degré $k = 2$.

Définition 12.7.1. *Soit p un premier tel que $p \equiv 1 \bmod 3$ et E_1 une courbe elliptique définie sur $\mathbb{F}_p$ d'équation $E_1 : y^2 = x^3 + B$. Soit r un grand premier tel que $r|\#E_1(\mathbb{F}_p)$. On suppose que E_1 a un degré de plongement $k = 2$. Soit E_1' le twist quadratique d'équation*

$E'_1 : y^2 = x^3 + D^{-3}B$, où D est un non-résidu quadratique modulo p. On suppose que $r^2 \nmid \#E_1(\mathbb{F}_p)$ et $r^2 \nmid \#E'_1(\mathbb{F}_p)$. Soit $P \in E_1(\mathbb{F}_p)[r]$ et $Q' \in E'_1(\mathbb{F}_p)[r]$. Soit l'isomorphisme :

$$\begin{aligned} \psi : E'_1 &\to E_1 \\ (x, y) &\mapsto (Dx, D^{3/2}y). \end{aligned}$$

On écrit $Q = \psi(Q') \in E_1(\mathbb{F}_p)[r]$. Soit $\beta \in \mathbb{F}_p$ un élément d'ordre 3. On peut définir deux endomorphismes :

$$\begin{aligned} \phi : E_1 &\to E_1 \\ (x, y) &\mapsto (\beta x, y) \end{aligned} \qquad \begin{aligned} \hat{\phi} : E_1 &\to E_1 \\ (x, y) &\mapsto (\beta^2 x, y). \end{aligned}$$

Soit λ une racine de l'équation $X^2 + X + 1 = 0 \bmod r$ telle que $\lambda P = \phi(P)$ et $\lambda Q = \hat{\phi}(Q)$. Soit a un entier tel que $ar = \lambda^2 + \lambda + 1$. Le couplage omega est alors défini par :

$$\omega(P, Q) = \left(\frac{f_{\lambda,P}(Q)}{f_{\lambda,Q}(P)} \right)^{(p-1)}.$$

Le couplage est non dégénéré si $r \nmid a$.

La preuve de cette définition se trouve dans [ZZ08]. Les auteurs atteignent un gain en temps d'exécution d'environ 20% par rapport à la proposition de couplage sur les courbes NSS de Scott [Sco05b].

Chapitre 13

Attaques physiques contre l'algorithme de Miller

La cryptographie à base de couplages (*Pairing Based Cryptography*, PBC) se développe de plus en plus ces dernières années avec de nombreuses propositions de protocoles. Des recherches sont aussi effectuées sur l'applicabilité des couplages sur différents supports et pour différents niveaux de sécurité. Parallèlement la résistance des couplages face aux attaques par canaux cachés est considérée dans plusieurs articles. On distingue les attaques par injection de fautes [PV04, PV06, WS07, EM09, EM10] et les attaques par analyse différentielle [PV04, WS06, KTH+06, KTH+08, EMDNF09]. Nous détaillons dans les sections suivantes les différentes propositions d'attaques et nous présentons des résultats pratiques d'attaques différentielles sur les couplages.

13.1 Attaques par injection de fautes

Deux stratégies d'attaques par injections de fautes ont été étudiées dans la littérature. Page et Vercauteren [PV04, PV06] ont d'abord étudié l'effet d'une faute sur le compteur de boucle de l'algorithme utilisé pour le calcul du couplage. Whelan et Scott [WS07] se sont plutôt intéressés à l'injection d'une faute dans une valeur intermédiaire de la boucle.

13.1.1 Faute dans le compteur de boucle

Page et Vercauteren sont les premiers à avoir proposé une attaque par canaux cachés sur un couplage dans [PV04, PV06]. Les auteurs attaquent l'algorithme de Duursma-Lee [DL03] amélioré par Barreto et al. [BGhS07] avec leur proposition du couplage eta (voir section 12.1). L'étude est donc réalisée sur des courbes supersingulières définies sur des corps de caractéristique 2 ou 3. Le principe de l'attaque de Page et Vercauteren consiste à injecter une faute sur le compteur de boucle de l'algorithme. Soit $\eta_{m,P}(Q)$ le couplage eta où m tours de boucle sont effectués. On considère maintenant les valeurs $\eta_{m\pm r,P}(Q)$ et $\eta_{m\pm(r+1),P}(Q)$ obtenues par un attaquant qui injecte une faute sur le compteur de boucle

afin d'ajouter ou soustraire respectivement r et $r+1$ tours. En pratique, un attaquant n'est souvent pas capable de contrôler à l'avance la valeur r qu'il va ajouter au compteur en injectant une faute. Il peut toutefois trouver cette valeur en comptant sur la courbe de consommation de courant le nombre de tours effectués sur le composant. Il doit donc réaliser de nombreuses injections jusqu'à obtenir le couple de résultats avec r et $r+1$ tours. Ce nombre d'attaques est réaliste et peut être évalué grâce au paradoxe des anniversaires. À partir du couple de valeurs désirées, Page et Vercauteren montrent alors que les coordonnées du point P, considéré secret, peuvent se retrouver facilement en résolvant un système d'équations linéaires. Leur proposition d'attaque est applicable car on peut facilement inverser l'exponentiation finale du couplage eta. En effet, l'exposant vaut soit q^3-1, soit q^2-1 suivant la caractéristique du corps fini considéré.

El Mrabet [EM09] adapte l'attaque de Page et Vercauteren [PV06] à l'algorithme de Miller. De plus, El Mrabet généralise l'attaque pour l'utilisation de tout types de coordonnées et notamment les coordonnées projectives jacobiennes, les plus utilisées en pratique pour leur efficacité. L'auteur montre que l'utilisation de ce système implique la résolution d'un système d'équations non-linéaires tout à fait réalisable. Néanmoins, l'exponentiation finale qui est complexe lors de l'utilisation de l'algorithme de Miller dans un couplage sur un corps fini $\mathbb{F}_q$ empêche une mise en pratique immédiate de l'attaque. Pour cela, l'attaquant doit pouvoir accéder à la valeur de sortie de l'algorithme de Miller avant son exponentiation, ce qui requiert un attaquant beaucoup plus puissant. Dans [EM10], El Mrabet propose aussi une attaque par injection de faute sur un couplage utilisant une courbe sous la forme d'Edwards.

13.1.2 Faute dans une variable intermédiaire

Contrairement à Page et Vercauteren, Whelan et Scott [WS07] proposent une attaque par injection de fautes sur une valeur intermédiaire de la boucle de Miller. Comme précédemment, les attaques proposées par Whelan et Scott ne sont possibles que lorsque l'exponentiation finale est très simple. La complexité de cette exponentiation dépend de l'algorithme de couplage choisi. Ainsi pour un algorithme tel que celui de Tate sur un corps de grande caractéristique, l'exponentiation finale est de la forme $(q^k-1)/m$. Dans ce cas, inverser l'exponentiation revient à résoudre le problème difficile de trouver une racine n-ième. Whelan et Scott appliquent ainsi une attaque par injection de faute sur le couplage η_G, proposé par Galbraith et al. [GHS07], qui correspond à une variante du couplage eta sans l'exponentiation finale. Si une faute est injectée lors du calcul d'une ligne ou d'une verticale d'un tour de boucle, l'effet de la faute sera local à ce tour et les coordonnées du secret peuvent être retrouvées facilement. Si on injecte une faute dans une des coordonnées des points, le cas est plus problématique mais l'attaque est toujours réalisable dans la plupart des cas.

Whelan et Scott [WS07] proposent aussi une attaque sur une implémentation du couplage de Weil avec une exponentiation finale par $p-1$ pour des courbes elliptiques ordinaires définies sur $\mathbb{F}_p$. Une faute injectée sur une variable intermédiaire quelconque ne permet plus de récupérer le secret à cause de l'exponentiation finale. Les auteurs proposent donc une

attaque basée sur la technique de Blömer et al. [BOS06] de changement de signe (voir section 7.1.3). On considère l'attaquant capable d'injecter une faute sur une variable de valeur x afin d'obtenir la valeur $-x$, ce qui est une contrainte beaucoup plus forte que précédemment. Néanmoins, en considérant ce modèle de faute, une attaque est possible lors du dernier tour de boucle sur l'implémentation du couplage de Weil telle que définie ci-dessus.

13.2 Attaques par analyse différentielle

Page et Vercauteren [PV04] ont proposé en premier la possibilité d'une attaque par analyse différentielle sur l'algorithme de Duursma-Lee ainsi que sur BKLS utilisé pour le couplage de Tate. Les auteurs remarquent que, dans les deux cas, un calcul intermédiaire fait intervenir une opération entre une coordonnée du point public et une coordonnée du point secret. Une attaque différentielle peut donc être réalisable.

Dans [WS06], Whelan et Scott étudient plus en détails comment effectuer de telles attaques sur les couplages eta, ate et sur l'algorithme BKLS du couplage de Tate. Les auteurs distinguent le cas où P est secret du cas où Q est secret pour le couplage $e(P, Q)$ utilisant l'algorithme BKLS. Si Q est secret, l'opération sensible semble être une multiplication dans le corps $\mathbb{F}_q$. Dans l'autre cas, les auteurs n'arrivent pas à proposer d'attaque et suggèrent d'ailleurs comme contre-mesure de placer, si possible, le secret dans le premier point P. L'attaque sur le couplage ate est très similaire, une multiplication sur une extension de $\mathbb{F}_q$ est cette fois visée. Comme précédemment, les auteurs proposent de placer le secret dans le premier paramètre du couplage afin d'éviter l'attaque. Whelan et Scott proposent des attaques ciblant les opérations de multiplication et racine carrée dans le couplage eta. Cette fois, des stratégies d'attaques lorsque le secret est le premier ou second paramètre sont proposées par les auteurs.

Kim et al. [KTH+06] étudient plus précisément le cas d'une attaque sur le couplage eta sur un corps de caractéristique 2. Ils s'intéressent notamment à l'attaque d'une multiplication dans $\mathbb{F}_{2^n}$ alors que Whelan et Scott [WS06] détaillent plus précisément le cas d'une attaque sur une multiplication dans $\mathbb{F}_q$.

Whelan et Scott dans [WS06] indiquent qu'une attaque sur l'algorithme BKLS du couplage Tate ou sur le couplage ate n'est plus possible lorsque P, le premier paramètre du couplage, est secret. Dans [EMDNF09], El Mrabet et al. décrivent en détail une attaque sur l'algorithme BKLS utilisé pour l'évaluation du couplage $f_P(Q)$ (Algorithme 15) où P est secret et Q est public. L'attaque se situe lors d'un calcul intermédiaire à la Ligne 4 Algorithme 15 que l'on rappelle ici :

$$f \leftarrow f^2.T_R(Q)/V_{2R}(Q).$$

Plus précisément, l'attaque cible le calcul de $T_R(Q)$, la tangente au point R évaluée au point Q. En pratique, on utilise souvent des coordonnées jacobiennes pour le point $P = (X_P, Y_P, Z_P)$ et des coordonnées affines pour $Q = (x_Q, y_Q)$ à des fins de performances.

Dans ce cas, la formule de la tangente $T_R(Q)$ s'écrit pour $R = (X, Y, Z)$:

$$T_R(Q) = Z_3 Z^2 y_Q - 2Y^2 - (3X^2 - aZ^4)(Z^2 x_Q - X),$$

avec $Z_3 = 2YZ$ la coordonnée Z du point $[2]R = (X_3, Y_3, Z_3)$. Le principe de l'attaque est de retrouver les coordonnées du point R qui est un multiple de P. On peut alors trouver le point secret P facilement puisqu'on peut délimiter à quel tour de boucle on se situe sur une courbe de consommation de courant. La coordonnée Z peut être trouvée lors de la multiplication $Z^2 y_Q$ entre Z^2 secret et y_Q public. Ensuite, on peut utiliser la multiplication $Z_3(Z^2 y_Q)$ afin de trouver la coordonnée Y en connaissant y_Q et Z et en rappelant que $Z_3 = 2YZ$. Grâce à l'équation de la courbe elliptique, on retrouve la coordonnée X du point R et on en déduit P. En pratique, on choisit une courbe elliptique telle que $P \in E(\mathbb{F}_q)$ et $Q \in E(\mathbb{F}_{q^k})$ où k est son degré de plongement. Ainsi les multiplications par des coordonnées du point Q sont en fait des multiplications sur $\mathbb{F}_{q^k}$. Le même schéma d'attaque s'applique toujours en supposant que l'attaquant connaisse l'algorithme de multiplication et la représentation des éléments de $\mathbb{F}_{q^k}$.

13.3 Réalisation pratique d'une attaque différentielle sur un couplage

Le travail décrit dans cette section a été réalisé en collaboration avec Nadia El Mrabet et Victor Lomné. L'objectif est d'appliquer en pratique l'attaque proposée dans [EMDNF09] et décrite ci-dessus. En effet, elle correspond à une implémentation classique et efficace de couplage avec l'utilisation de l'algorithme BKLS pour le couplage de Tate sur une courbe elliptique définie sur un corps de grande caractéristique.

13.3.1 Détails sur l'implémentation

Nous avons implémenté le début d'un calcul de couplage jusqu'à la fonction à attaquer : la mise à jour de la variable f par $T_R(Q)$ (Ligne 4 Algorithme 15). On considère donc une courbe elliptique E définie sur un corps $\mathbb{F}_q$ de grande caractéristique. Soit m le cardinal du sous-groupe premier de points de la courbe et k le degré de plongement associé. Soit $P \in E(\mathbb{F}_q)[m]$ et $Q \in E(\mathbb{F}_{q^k})$. Le couplage de Tate entre ces deux points est donc :

$$\langle P, Q \rangle = f_P(Q)^{(q^k-1)/m}.$$

On décompose les premières opérations effectuées par l'Algorithme 15 jusqu'à la fonction attaquée au premier tour de boucle :

1. $R \leftarrow P$, on sauvegarde le point d'entrée P dans un point temporaire R,
2. $A \leftarrow [2]R$, on calcule le doublement du point R,
3. on calcule $T_R(Q)$, la tangente au point R évaluée en Q, qui contient des données sensibles pour notre attaque.

Soit les points $R = (X, Y, Z) \in E(\mathbb{F}_q), [2]R = A = (X_3, Y_3, Z_3) \in E(\mathbb{F}_q), Q = (x_Q, y_Q) \in E(\mathbb{F}_{q^k})$ et a le paramètre de la courbe $E : y^2 = x^3 + ax + b$. La fonction $T_R(Q)$ se calcule alors par la formule :

$$T_R(Q) = Z_3 Z^2 y_Q - 2Y^2 - 3(X - Z^2)(X + Z^2)(Z^2 x_Q - X). \tag{13.1}$$

Le calcul de la fonction $T_R(Q)$ est détaillé dans l'Algorithme 16. Comme précisé précédemment, nous nous plaçons dans le cadre de l'attaque proposée dans [EMDNF09] avec une implémentation utilisant des coordonnées jacobiennes pour le point dans $E(\mathbb{F}_q)$ et des coordonnées affines pour le point dans $E(\mathbb{F}_{q^k})$. Cette combinaison de système de coordonnées est efficace pour une implémentation de couplage. On peut toutefois noter que si d'autres systèmes de coordonnées sont considérés, même si les opérations de l'Algorithme 16 sont modifiées, le principe de l'attaque est toujours applicable.

Algorithme 16: Évaluation de $T_R(Q)$

Entrées : $R = (X, Y, Z) \in E(\mathbb{F}_q), [2]R = A = (X_3, Y_3, Z_3) \in E(\mathbb{F}_q), Q = (x_Q, y_Q) \in E(\mathbb{F}_{q^k})$, a le paramètre de la courbe $E : y^2 = x^3 + ax + b$.

Sorties : $T_R(Q) \in E$

1 $Z' \leftarrow Z^2$
2 $R_1 \leftarrow Z' y_Q$
3 $R_1 \leftarrow R_1 Z_3$
4 $R_2 \leftarrow 2Y^2$
5 $R_1 \leftarrow R_1 - R_2$
6 $R_2 \leftarrow 3X^2$
7 $R_3 \leftarrow Z'^2$
8 $R_3 \leftarrow aR_3$
9 $R_2 \leftarrow R_2 + R_3$
10 $R_3 \leftarrow Z' x_Q$
11 $R_3 \leftarrow R_3 - X$
12 $R_2 \leftarrow R_2 R_3$
13 $R_1 \leftarrow R_1 - R_2$
14 **retourner** R_1

Notre évaluation s'effectue sur la carte de développement Atmel STK600 [ATMb] utilisant un processeur 8-bit AVR ATmega2561 [ATMa].

13.3.2 Principe de l'attaque

Lors du calcul du couplage de Tate $\langle P, Q \rangle$, Whelan et Scott [WS06] distinguent deux cas : soit P est secret, soit Q est secret. Les auteurs proposent une attaque lorsque Q est le secret et arrivent à la conclusion que l'implémentation serait sûre si on choisit P secret. Whelan et Scott considèrent des systèmes de coordonnées affines pour les deux points.

Nous nous plaçons dans le cas plus efficace en pratique où le point P est représenté en coordonnées jacobiennes et le point Q en coordonnées affine. Nous précisons les données sensibles à attaquer dans l'Algorithme 16.

13.3.2.1 Si le point Q est secret

Nous avons simplement besoin de retrouver la coordonnée x_Q, la coordonnée y_Q se calculant à partir de l'équation de la courbe. Dans ce scénario, le point P est public. Ainsi, les points intermédiaires R et A, des multiples de P, sont aussi connus de l'attaquant. En effet, connaissant l'ordre m du sous-groupe premier, le point A correspond à la valeur, à un tour donné, d'un algorithme de doublement-et-addition avec P comme point de base. On note que la coordonnée x_Q apparaît à la Ligne 10 de l'Algorithme 16 lors de la multiplication $Z'x_Q$ avec $Z' = Z^2$. Il s'agit donc d'une multiplication entre un élément connu de l'attaquant, Z', et le secret x_Q. Nous voyons ci-après, dans la section 13.3.3, que la multiplication dans un corps correspond à une fonction de sélection possible pour une attaque différentielle.

Un premier scénario d'attaque consiste à considérer N réalisations du couplage $\langle P, Q\rangle$ pour N points P publics. Alors, au premier tour de boucle de l'algorithme de Miller, on a $R = P$ et donc $A = [2]R = [2]P$. L'attaquant a donc connaissance de la coordonnée $Z' = Z^2$ lors du calcul de $Z'x_Q$.

Lors du calcul d'un couplage $\langle P, Q\rangle$ avec P connu, nous avons noté que l'attaquant peut en déduire les valeurs des points R et A à un tour donné. Soit n la longueur en bits de m, l'ordre du sous-groupe premier de points, et aussi le compteur de boucle de l'algorithme de Miller. L'évaluation de $T_R(Q)$ par l'Algorithme 16 s'effectue donc n fois pour n valeurs de R et A connues de l'attaquant et pour une valeur de Q fixée. Ainsi, au lieu de réaliser N calculs de couplages et ne considérer l'opération $Z'x_Q$ qu'au premier tour de l'algorithme de Miller, un attaquant peut considérer cette opération sur les n tours de l'algorithme et donc certainement diminuer le nombre N de couplages nécessaires pour réaliser cette attaque.

13.3.2.2 Si le point P est secret

Une attaque lorsque le point P du couplage $\langle P, Q\rangle$ est secret a été proposée dans [EMDNF09]. Le dernier scénario d'attaque évoqué ci-dessous est impossible dans ce cas, le point Q public étant invariant au cours des tours de l'algorithme de Miller. Le point secret P et les points intermédiaires R et A évoluant à chaque tour, il est plus simple de considérer une attaque lors du premier tour de boucle afin de retrouver directement P et non un de ses multiples. L'attaque doit se dérouler en deux temps. En effet, nous devons retrouver deux des trois coordonnées de P pour déterminer la troisième à partir de l'équation de la courbe. En considérant l'Algorithme 16, on note à la Ligne 2 l'opération $Z'y_Q$ avec $Z' = Z^2$. La coordonnée y_Q étant publique, il est possible de retrouver par attaque différentielle Z' puis, par une racine carrée modulaire, une des deux solutions correspondant au Z recherché. Une fois la coordonnée Z déterminée par la première attaque, l'attaquant est désormais capable de calculer la valeur de $R_1 \leftarrow Z'y_Q$ à la Ligne 2. En considérant l'opération de la

ligne suivante $R_1 Z_3$, l'attaquant connaît R_1 et peut donc retrouver Z_3. On rappelle que Z_3 est la coordonnée Z du point $A = [2]P$ au premier tour de boucle. Or, on sait que $Z_3 = 2YZ$ avec la formule classique du doublement de points en coordonnées jacobiennes. En trouvant Z_3, l'attaquant peut facilement en déduire la coordonnée Y du point P secret.

13.3.3 Multiplication multi-précision

La multiplication multi-précision sur corps finis est au centre de l'attaque par analyse différentielle étudiée ici. Nous développons dans cette partie quelques méthodes classiques. On introduit d'abord quelques notations. Les grands entiers sont représentés en mémoire comme des tableaux de valeurs sur w bits. En général, w correspond à la taille maximale des mots gérés par le processeur. Ainsi, pour notre implémentation sur un processeur 8-bit ATmega2561, on a $w = 8$. La longueur en bits d'un grand entier est notée s et n représente le nombre de valeurs de w bits nécessaires pour le stocker, donc $n = \lceil s/w \rceil$. Un grand entier A s'écrit donc avec cette représentation $A = (a_{n-1}, \ldots, a_1, a_0)$ avec $0 \leq a_i \leq 2^w$. On considère dans la suite la multiplication de deux grands entiers A et B de même longueur n.

13.3.3.1 Méthode naïve par lignes

Cette méthode correspond à celle utilisée lorsqu'on calcule à la main une multiplication (Algorithme 17). La stratégie consiste à fixer le multiplicande b_i et à le multiplier par chaque a_j du multiplicateur $(a_{n-1}, \ldots, a_1, a_0)$ avant de passer au multiplicande suivant b_{i+1}. Les produits partiels avec un multiplicande sont additionnés dans s registres accumulateurs. Une fois b_i traité, seul le registre accumulateur de poids faible va correspondre à une valeur du produit final. La méthode a donc besoin de $n + 2$ registres puisqu'il il faut un registre pour charger b_i et un autre pour a_j. Si l'architecture du processeur, comme celle du ATmega2561, ne permet pas de stocker dans ses registres à la fois les n registres d'accumulation et un multiplicande entier, il faut jusqu'à $n^2 + 3n$ opérations de stockage et chargement de données dans les registres. On évite donc la méthode de multiplication naïve sur des processeurs possédant peu de registres, ce qui est le cas de la plupart des processeurs pour l'embarqué.

13.3.3.2 Méthode de Comba

La méthode de multiplication de Comba [Com90] correspond à une multiplication par colonnes. La stratégie consiste à additionner les produits partiels $a_j.b_i$ pour $i + j = l$, ce qui correspond à la colonne l. À la fin de chaque colonne, on obtient un mot de w bits correspondant à une partie du produit final. Cette méthode requiert seulement cinq registres, ce qui est très intéressant sur des plateformes embarquées avec très peu de registres. En contrepartie, il faut effectuer plus d'opérations de stockage et chargement de données dans des registres : $2n^2 + 2n$.

Algorithme 17: Multiplication multi-précision par lignes (méthode naïve).

Entrées : Deux grands entiers $A = (a_{n-1}, \ldots, a_1, a_0)$ et $B = (b_{n-1}, \ldots, b_1, b_0)$ de même longueur n.

Sorties : Le produit $P = AB = (p_{2n-1}, \ldots, p_1, p_0)$ de longueur $2n$.

```
1 P ← 0
2 pour i ← 0 a n − 1 faire
3     u ← 0
4     pour j ← 0 a n − 1 faire
          // La notation (u, v) correspond à l'entier u.2^w + v.
5         (u, v) ← a_j.b_i + p_{i+j} + u
6         p_{i+j} ← v
7     p_{n+i} ← u
8 retourner P
```

Algorithme 18: Multiplication multi-précision par colonnes (méthode de Comba).

Entrées : Deux grands entiers $A = (a_{n-1}, \ldots, a_1, a_0)$ et $B = (b_{n-1}, \ldots, b_1, b_0)$ de même longueur n.

Sorties : Le produit $P = AB = (p_{2n-1}, \ldots, p_1, p_0)$ de longueur $2n$.

```
// La notation (t, u, v) correspond à l'entier t.2^{2w} + u.2^w + v.
1 (t, u, v) ← 0
2 pour i ← 0 a n − 1 faire
3     pour j ← 0 a n − 1 faire
4         (t, u, v) ← (t, u, v) + a_j.b_{i−j}
5     p_i ← v
6     v ← u, u ← t, t ← 0
7 pour i ← n a 2n − 2 faire
8     pour j ← i − n + 1 a n − 1 faire
9         (t, u, v) ← (t, u, v) + a_j.b_{i−j}
10    p_i ← v
11    v ← u, u ← t, t ← 0
12 p_{2n−1} ← v
13 retourner P
```

On peut aussi noter la multiplication hybride [GPW+04, SS07] qui combine certains avantages des deux méthodes présentées ci-dessus. Karatsuba et Ofman [KO63] ont présenté la première méthode de multiplication de complexité sous-exponentielle en temps dont le principe est de découper les entiers en deux parties. Toom et Cook [Too63, Coo66] ont ensuite étendu l'idée de Karatsuba en divisant les entiers en trois parties pour obtenir une complexité encore inférieure. Ces complexités sous-exponentielles ne sont toutefois atteintes que lorsque les entiers à multiplier sont très grands. De plus, de part la nature récursive de ces algorithmes, ils requièrent souvent plus d'espace mémoire et plus d'accès mémoire ce qui est problématique dans des environnements embarqués. Ils peuvent toutefois être intéressants dans le cadre de multiplications sur des extensions en corps comme nous l'étudions ci-dessous.

13.3.3.3 Cas des extensions de corps

En pratique, un des deux points du couplage est généralement défini dans une extension de corps $\mathbb{F}_{q^k}$. Cela implique que, pour les différents scénarios d'attaques proposés, nous devons considérer le cas où la multiplication dans un corps fini peut être une multiplication dans une extension de corps fini. Nous explicitons brièvement le fait que le principe des attaques reste inchangé.

Étant donné que les opérations dans une extension $\mathbb{F}_{q^k}$ de grand degré sont très coûteuses, on fait en sorte de travailler le plus possible dans des extensions de petit degré. On considère ici le cas d'une extension de quadratique pour plus de simplicité.

Soit l'extension quadratique $\mathbb{F}_{q^2} = \mathbb{F}_q[X]/(X^2 - \alpha)$ avec $\alpha \in \mathbb{F}_q$ un non-résidu quadratique. Un élément $A \in \mathbb{F}_{q^2}$ est représenté par $a_1X + a_0$ avec $a_0, a_1 \in \mathbb{F}_q$. La multiplication $C = AB$ avec $A, B, C \in \mathbb{F}_{q^2}$ se calcule alors par différentes méthodes :

– la méthode naïve,

$$\begin{aligned} c_0 &= a_0b_0 + \alpha a_1b_1 \\ c_1 &= a_0b_1 + a_1b_0, \end{aligned}$$

pour un coût de 4 multiplications dans le corps de base $\mathbb{F}_q$, 2 additions et une multiplication par α,

– par la méthode de Karatsuba, soit $v_0 = a_0b_0$ et $v_1 = a_1b_1$,

$$\begin{aligned} c_0 &= v_0 + \alpha v_1 \\ c_1 &= (a_0 + a_1)(b_0 + b_1) - v_0 - v_1, \end{aligned}$$

pour un coût de 3 multiplications, 5 additions et une multiplication par α.

Si on suppose l'entier A comme secret et B comme public, il va falloir que l'attaquant retrouve les deux éléments de $\mathbb{F}_q$, a_1 et a_0, de sa représentation afin d'obtenir la valeur de A. L'attaquant doit donc réaliser deux attaques pour retrouver séparément ces deux valeurs. Il s'agit alors de simples attaques sur des multiplications multi-précisions. Dans la méthode naïve, il peut retrouver l'élément a_0 soit par la multiplication a_0b_0, soit par a_0b_1.

De manière similaire, il retrouve a_1 soit par la multiplication $a_1 b_1$, soit par $a_1 b_0$. Dans la méthode de Karatsuba, il lui suffit de d'attaquer les multiplications $v_0 = a_0 b_0$ et $v_1 = a_1 b_1$.

Ces observations se généralisent pour des multiplications dans des extensions de degré supérieur. Évidemment, plus le degré est grand, plus l'attaquant doit effectuer d'attaques pour retrouver les éléments de la représentation de l'entier dans l'extension. Plus de détails sur les opérations de multiplication et d'élévation au carré dans des extensions de corps sont disponibles dans [DDhS06]. On réalise l'attaque, pour plus de simplicité, en considérant des multiplications dans un corps $\mathbb{F}_q$.

13.3.4 Résultats expérimentaux

Sur notre implémentation à base de ATmega2561, on rappelle qu'un grand entier A est représenté par $(a_{n-1}, \ldots, a_1, a_0)$ avec $0 \leq a_i \leq 2^8$. Les a_i sont donc des octets. Si on considère l'Algorithme 17 de multiplication multi-précision par lignes, on peut retrouver A, octet par octet, en attaquant lors de la multiplication Ligne 5 entre un octet de A et un octet de B. On peut réaliser une attaque en cherchant les octets de A du moins significatif au plus significatif ou inversement sans grande différence sur les résultats de l'attaque.

Pour l'attaque, nous considérons le cas où les deux points du couplage P et Q appartiennent à un corps fini $\mathbb{F}_q$. Nous avons vu que le cas où un point est sur une extension de corps complique légèrement l'attaque mais son principe reste identique. L'implémentation utilise la multiplication multi-précision de Comba et la réduction de Barrett de la librairie YACL [Ven10c]. Nous nous plaçons dans le scénario le plus complexe où le point P est considéré comme secret (voir section 13.3.2). L'attaque a été réalisée sur un total de 9000 courbes de consommation enregistrées à partir du circuit imprimé Atmel STK600. La Figure 13.1 représente une courbe de consommation centrée sur les deux opérations sensibles pour réaliser l'attaque.

On rappelle que l'on réalise en premier une attaque permettant de retrouver la coordonnée Z de P grâce à la multiplication $Z' y_Q$ (Algorithme 16 Ligne 2). Dans un deuxième temps, on effectue une autre attaque sur les mêmes courbes afin de trouver la coordonnée Y à partir de l'opération $R_1 Z_3$ (Algorithme 16 Ligne 3) puisque R_1 est désormais connu grâce à la première attaque. Pour évaluer l'efficacité des attaques, nous utilisons les métriques de *guessed entropy* et de *first-order success rate* détaillées dans la section 4.6. Les attaques sont répétées sur 45 ensembles de 200 courbes de consommation afin d'obtenir des résultats moyennés (Figure 13.2).

13.4 Protections face aux attaques par injection de fautes

Comme nous l'avons vu précédemment, les attaques par fautes s'appliquent principalement sur les algorithmes d'évaluation de couplage qui ne disposent pas d'exponentiation finale assez complexe. En pratique, cela concerne surtout l'algorithme pour le couplage eta et pour le couplage de Weil. Même si le couplage de Weil est peu utilisé, le couplage eta est

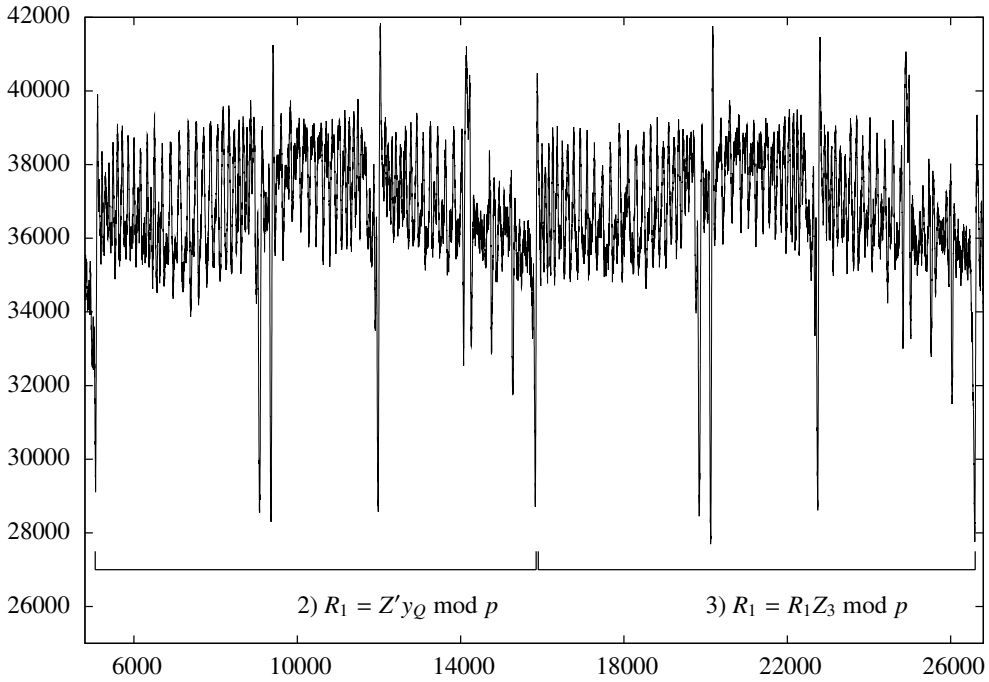


FIGURE 13.1 – Courbe de consommation du début d'un couplage où se situent les deux opérations qui nous intéressent pour l'attaque.

très efficace sur des corps de caractéristique 2 ou 3. Page et Vercauteren [PV06] proposent donc quelques contre-mesures pour se protéger des attaques par fautes. Une première est basée sur la bilinéarité du couplage et sur la relation :

$$e([a]P, [b]Q) = e(P, Q)^{ab},$$

pour a et b deux valeurs quelconques du corps de base. Les points de base P et Q sont alors randomisés par des facteurs choisis au hasard. Néanmoins, cela ajoute un facteur ab à l'exponentiation finale qui est déjà très coûteuse. Une solution est de choisir les entiers tels que $ab \cong 1 \bmod m$ où m est l'ordre du sous-groupe de points premier. Une fois une telle relation trouvée entre deux entiers, il est facile de les mettre à jour tout en gardant la relation valide.

Une méthode d'aveuglement du point de base, similaire à celle présentée dans la section 8.2.2, est proposée par Page et Vercauteren [PV06]. Soit d un scalaire secret, un point aléatoire R sur la courbe elliptique. On considère le couplage $e(P, Q)$ où Q est le paramètre secret et P public. Soit $S = e(P, R)^{-1}$, alors on a :

$$e(P, Q) = e(P, Q + R)S.$$

La mise à jour des valeurs de R et S peut s'effectuer à moindre coût comme expliqué dans [PV06].

Une autre contre-mesure plus efficace peut consister à vérifier, au cours de l'algorithme, si les points sont toujours sur la courbe. Néanmoins, l'attaque par changement de signe

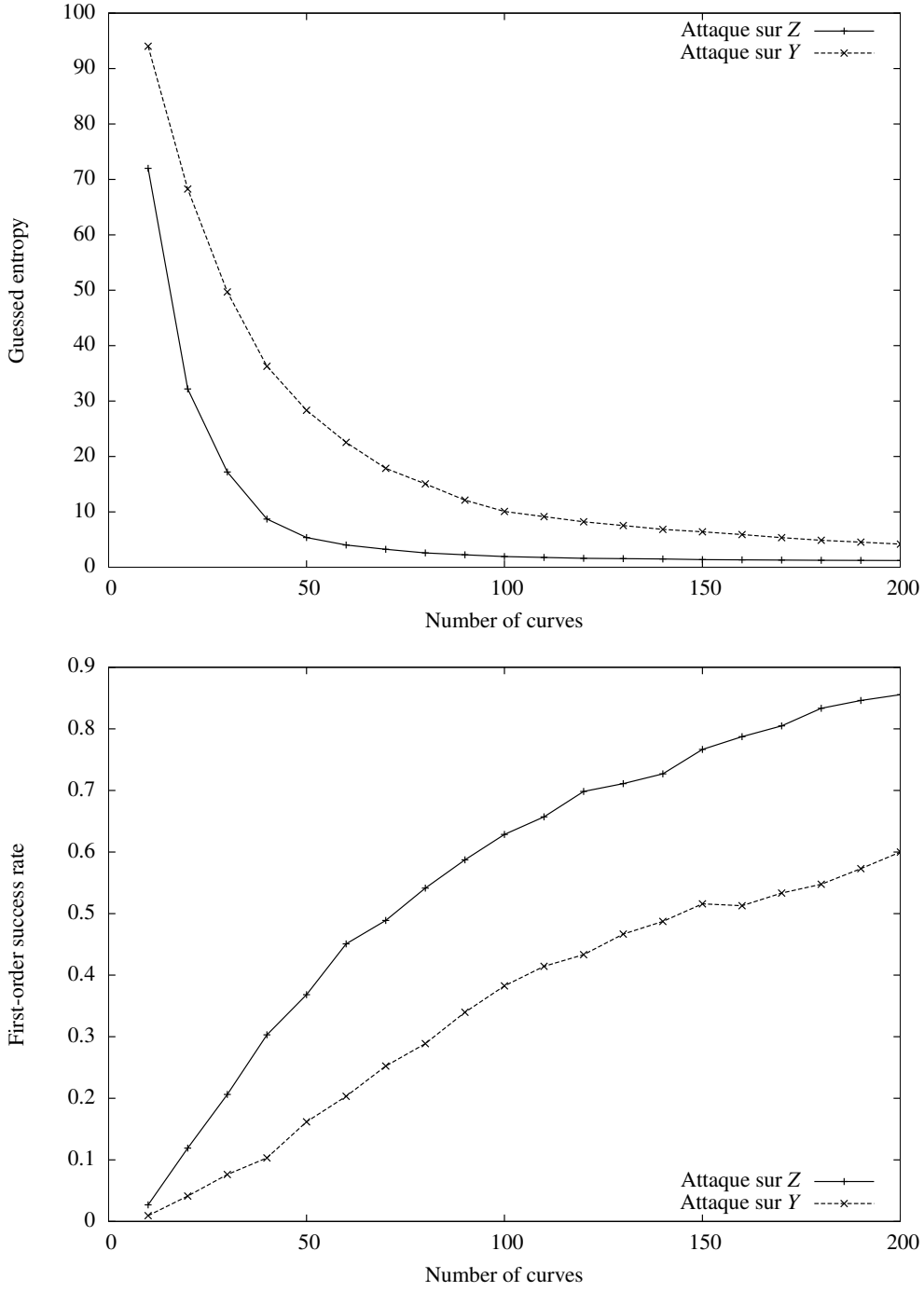


FIGURE 13.2 – Résultats d'attaques sur un couplage en considérant que le premier paramètre P est secret. L'axe des abscisses correspond au nombre de courbes de consommation utilisées. L'axe des ordonnées correspond au rang pour les attaques *guessed entropy* ou à une probabilité pour les attaques *first-order success rate*.

proposée sur le couplage de Weil [WS07] peut ne pas être détectée. Finalement, une solution radicale et très coûteuse pour se prémunir des injections de faute consiste à effectuer deux fois le calcul de couplage pour vérifier l'exactitude des résultats.

13.5 Protections face aux attaques par analyse différentielle

Les contre-mesures décrites ci-dessus et proposées par Page et Vercauteren [PV06] dans le cadre des attaques par injection de fautes s'appliquent aussi pour se protéger des attaques différentielles. Il existe aussi des protections spécifiques face à ce type d'attaques que nous détaillons dans cette partie.

Si le couplage est implémenté en utilisant des coordonnées projectives, alors on peut utiliser la contre-mesure de Coron [Cor99] qui consiste à randomiser les coordonnées. Cette méthode qui est déjà très utile dans le cadre de la cryptographie à base de courbes elliptiques (voir section 8.2.2) et très efficace lorsqu'on choisit d'utiliser des coordonnées projectives dans un couplage. Kim et al. [KTH+06] proposent une méthode de calcul pour le couplage eta dans un corps de caractéristique 2 et utilisant des coordonnées projectives randomisées. Shirase et al. [STO08] proposent une adaptation au cas du couplage eta sur des corps de caractéristique 3. Même si le choix de coordonnées projectives est souvent plus performants [IT03a], des recherches récentes par Lauter et al. [LMN10] montrent qu'un couplage utilisant les coordonnées affines peut être très efficace dans certains cas.

Une contre-mesure plus générique est proposée par Scott dans [Sco05a]. Le principe est de randomiser la variable de Miller, f dans l'Algorithme 15, en la multipliant par un élément de $\mathbb{F}_q$, si on considère un couplage de Tate. En effet, les éléments de $\mathbb{F}_q$ sont éliminés du résultat du couplage par l'exponentiation finale. On remarque qu'il ne suffit pas de multiplier à chaque tour de boucle la variable f par une valeur aléatoire. Les opérations sensibles sont des calculs intermédiaires qui ont lieu pendant le tour de boucle. Il convient donc de multiplier par une valeur aléatoire à tous les endroits où des données sensibles sont manipulées. Le calcul de la fonction $T_R(Q)$ (13.1) devient alors :

$$T_R(Q) = rZ_3Z^2y_Q - 2rY^2 - 3(rX - rZ^2)(rX + rZ^2)(rZ^2x_Q - rX),$$

pour une valeur aléatoire $r \in \mathbb{F}_q$. Le surcoût ajouté par cette contre-mesure dépend de la formule de couplage choisie. Il reste toutefois assez important étant donné que plusieurs multiplications par r sont effectuées à chaque tour de boucle de l'algorithme.

Conclusion

Dans cette thèse, nous nous sommes intéressés à l'effet des attaques par canaux cachés (SCA) sur la sécurité d'implémentations d'algorithmes cryptographiques. Nous avons considéré dans notre étude à la fois des cryptosystèmes symétriques, comme l'AES, et asymétriques, comme les cryptosystèmes à base de courbes elliptiques et leur opération centrale, la multiplication scalaire. Enfin, nous nous sommes intéressés à la résistance des couplages, constructions mathématiques relativement récentes proposant des propriétés très intéressantes pour la construction de protocoles innovants.

Nous avons proposé, dans une première partie, une protection face aux attaques différentielles de premier ordre pour l'AES [BLV10]. La contre-mesure est basée sur une implémentation de l'algorithme utilisant une tour d'extensions de corps fini qui est très performante, particulièrement dans des implémentations *hardware*. Notre proposition, basée sur cette construction algébrique, permet d'ajouter un niveau de protection par rapport à la technique de masquage booléen proposée par Oswald et al. [OMPR05, OS06] pour un surcoût assez faible en temps et mémoire. L'un des principes consiste à randomiser la construction de la tour d'extensions de corps. Lors du calcul de l'inverse par cette méthode, l'évaluation de la norme des éléments est le point central à protéger. Néanmoins, avec cette construction aléatoire, les valeurs de normes se dispersent en ensembles de petit cardinal, pour un gain de protection assez faible. Nous nous sommes aperçus qu'il existait une relation entre la norme et l'ordre des éléments et nous avons proposé de modifier aléatoirement l'ordre afin que les normes se dispersent mieux. Nous obtenons une contre-mesure particulièrement bien adaptée aux implémentations d'AES utilisant une tour d'extensions qui, appliquée avec le masquage booléen d'Oswald et al., permet d'obtenir une protection efficace face aux attaques de premier ordre.

Nous nous sommes aussi particulièrement intéressés aux attaques différentielles par analyse d'information mutuelle (MIA). Ce type d'attaque, proposé par Gierlichs et al. [GBTP08], a suscité beaucoup d'intérêt récemment traduit par un certain nombre de publications sur le sujet. Certaines tentent d'améliorer les performances de l'attaque pour la rendre compétitive avec le test statistique de Pearson (CPA). De nombreuses améliorations sont basées sur la supposition par l'attaquant d'une certaine distribution de probabilités afin d'utiliser des estimateurs paramétriques d'entropie très efficaces. On peut citer la proposition récente de Lee et Berthier [LB10] d'un estimateur paramétrique qui offre des résultats comparables à la CPA. Néanmoins, l'un des principes originaux de la MIA est de proposer une attaque générique, applicable sur tout type de composants, sans que l'attaquant n'ait à supposer un certain modèle de consommation ou la distribution de probabilités de sa variable aléatoire représentant sa fuite d'information. Nous nous sommes donc concentrés sur l'étude des estimateurs non-paramétriques d'entropie, estimateurs qui ne supposent aucune distribution de probabilités particulière des variables. Nous avons appliqué les meilleurs estimateurs dans le cadre des attaques par canaux cachés pour étudier leur efficacité dans ce scénario [Ven10a]. Nous avons étudié en détail l'estimateur à base de B-splines qui semble particulièrement bien adapté aux attaques comme nous l'avions remarqué dans l'article [Ven10b]. Nous nous sommes aussi intéressés au principe d'infor-

mation mutuelle généralisée basée sur l'entropie de Rényi dans le cadre des SCA. Grâce à différentes expérimentations sur plusieurs plateformes, nous avons obtenu des résultats qui confirment que l'estimateur par B-splines offre l'estimation non-paramétrique la plus intéressante bien qu'étant derrière les tests paramétriques.

Dans la partie II, nous avons étudié l'application des SCA sur les cryptosystèmes à base de courbes elliptiques (ECC). L'opération principale de nombreux ECC est la multiplication scalaire qui peut être vue comme une opération d'exponentiation modulaire dans un groupe additif. Il existe donc de nombreux algorithmes équivalents notamment le doublement-et-addition pour les ECC qui correspond au carré-et-multiplication du RSA par exemple. Après un état de l'art des différentes attaques et techniques de protections SCA, nous nous sommes intéressés à l'algorithme de multiplication scalaire très connu nommé échelle de Montgomery. Cet algorithme a l'avantage d'avoir été étudié en profondeur dans la littérature ainsi que d'être résistant, par construction, à différentes attaques SCA. En modifiant simplement l'algorithme de Montgomery, nous pouvons faire en sorte qu'il n'utilise que des additions. Néanmoins, en général, une addition de points est beaucoup plus coûteuse qu'un doublement de point. Nous avons alors étudié la formule d'additions de points simplifiée de Méloni [Mel07] qui offre, sur des corps de grande caractéristique, des performances meilleures qu'un doublement de point. Nous avons adapté la formule de Méloni afin d'ajouter, pour un faible surcoût, une soustraction de points. Nous avons alors proposé de combiner l'échelle de Montgomery utilisant uniquement des additions avec la formule modifiée de Méloni. En la combinant avec l'échelle de Montogomery modifiée, nous avons obtenu l'algorithme de multiplication scalaire le plus performant de la littérature pour le niveau de sécurité considéré. Ce travail a fait l'objet d'une publication [VD10b] et d'un brevet [VD10a]. Nous avons appliqué la méthode sur des corps de caractéristique 2 dans l'article [VD10c]. Néanmoins, dans ce cas, l'échelle de Montgomery proposée par López et Dahab [LD99a] en 1999 reste l'algorithme le plus performant. Notre proposition offre tout de même une alternative à cet algorithme breveté.

Nous avons étudié les couplages dans la partie III, plus précisément, leur efficacité et leur résistance face aux SCA. Les couplages sont des constructions mathématiques, assez récentes dans le domaine de la cryptographie, qui permettent la création de nouveaux protocoles jusqu'alors impossibles avec la cryptographie classique. Néanmoins leur complexité était, il y a quelques temps encore, assez prohibitive pour une implémentation sur un composant embarqué. Les couplages de Weil et Tate étaient les seules constructions. De plus, elles étaient toutes deux évaluées grâce à l'algorithme de Miller. Récemment, quelques auteurs ont proposé des couplages mieux adaptés aux différents corps de base sur lesquels ils sont définis. Par exemple, le coulage eta est particulièrement efficace sur les corps de caractéristiques 2 et 3. Des améliorations générales, notamment au couplage de Tate, ont aussi été proposées avec le concept de couplage optimal. Tous ces résultats permettent d'envisager, plus concrètement, l'implémentation de couplages sur cartes à puces. Dans ce

contexte, nous nous sommes intéressés aussi à l'étude de la résistance des couplages face aux SCA. Nous avons étudié une implémentation classique utilisant le couplage de Tate sur un corps de grande caractéristique. Basé sur le travail de El Mrabet et al. [EMDNF09], nous avons proposé une réalisation pratique d'une attaque par analyse différentielle de courant dans différents scénarios d'utilisation du couplage. Nous avons considéré les cas où le secret est soit le premier, soit le second argument du couplage, qui entraînent des procédures d'attaques différentes. Nous nous sommes concentrés sur le cas où le secret est le premier argument, technique qui était considérée comme sûre face aux SCA dans [WS06]. Malgré une complexité plus élevée, une attaque est toujours possible. Il est donc nécessaire d'utiliser dans tous les cas une des contre-mesures SCA proposées dans la littérature pour obtenir une protection efficace.

En conclusion, ces travaux de recherches nous ont permis de mieux cerner la problématique d'attaques par canaux cachés qui est essentielle pour la sécurisation des composants embarqués. Nous avons pu aborder à la fois des cryptosystèmes symétriques classiques, comme l'AES, des systèmes asymétriques qui deviennent standards, comme les ECC, et de nouvelles constructions avec les couplages. Ce travail de thèse ouvre toutefois plusieurs pistes de recherche. Nous pouvons évaluer notre proposition de contre-mesure AES en l'implémentant en *hardware* pour avoir une meilleure vision pratique de son effet, comme cela est souvent nécessaire lorsqu'on travaille sur les SCA. L'AES, de part sa structure algébrique, est particulièrement adapté à l'utilisation d'autres contre-mesures SCA tirant parti de ses propriétés mathématiques. De récentes publications sur l'attaque par analyse d'information mutuelle indiquent qu'elle est particulièrement bien adaptée au cas des attaques différentielles d'ordre supérieur. Nous n'avons pas pris en compte ce type d'attaque dans notre travail, et notamment, l'efficacité des attaques MIA dans ce cas. Il serait aussi intéressant d'étudier la pertinence de la MIA lorsque d'autres types de logiques que la CMOS sont utilisées afin de vérifier si les propriétés de l'information mutuelle permettent d'obtenir de meilleurs résultats que les tests paramétriques. Dans le cas des systèmes ECC, bien que notre proposition soit intéressante dans le cas des corps de grande caractéristique, le cas de la caractéristique 2 reste à améliorer. En effet, l'algorithme de López et Dahab offre, depuis quelques années maintenant, les meilleures performances pour un très bon niveau de sécurité. Finalement, dans le cadre des couplages, il serait aussi intéressant d'étudier la possibilité d'attaques différentielles et par injection de fautes sur les récentes constructions efficaces. Il pourrait aussi être pratique d'évaluer les performances de ces constructions sur composant pour vérifier si leur applicabilité est réaliste en pratique.

Bibliographie

[AARR02] D. Agrawal, B. Archambeault, J. Rao et P. Rohatgi : The EM Side-Channel(s). *CHES 2002, LNCS*, 2523:29–45, 2002.

[ABF+02] C. Aumüller, P. Bier, W. Fischer, P. Hofreiter et J.P. Seifert : Fault Attacks on RSA with CRT : Concrete Results and Practical Countermeasures. *CHES 2002, LNCS*, 2523:81–95, 2002.

[AFT+09] F. Amiel, B. Feix, M. Tunstall, C. Whelan et W. Marnane : Distinguishing Multiplications from Squaring Operations. *Selected Areas in Cryptography, LNCS*, 5381:346–360, 2009.

[AG01] M.L. Akkar et C. Giraud : An Implementation of DES and AES, Secure against some Attacks. *CHES 2001, LNCS*, 2162:309–318, 2001.

[AK98] R. Anderson et M. Kuhn : Low Cost Attacks on Tamper Resistant Devices. *Security Protocols, LNCS*, 1361:125–136, 1998.

[APSQ06] C. Archambeau, E. Peeters, F.X. Standaert et J.J. Quisquater : Template Attacks in Principal Subspaces. *CHES 2006, LNCS*, 4249:1–14, 2006.

[ARR03] D. Agrawal, J.R. Rao et P. Rohatgi : Multi-Channel Attacks. *CHES 2003, LNCS*, 2779:2–16, 2003.

[ARRS05] D. Agrawal, J.R. Rao, P. Rohatgi et K. Schramm : Templates as Master Keys. *CHES 2005, LNCS*, 3659:15–29, 2005.

[AT03] T. Akishita et T. Takagi : Zero-Value Point Attacks on Elliptic Curve Cryptosystem. *Information Security, LNCS*, 2851:218–233, 2003.

[ATMa] ATMEL : ATmega 2561 Data Sheet. `http://www.atmel.com/dyn/resources/prod_documents/doc2549.pdf`.

[ATMb] ATMEL : Atmel AVR STK600. `http://www.atmel.com/dyn/products/tools_card.asp?tool_id=4254`.

[BBJ+08] D. Bernstein, P. Birkner, M. Joye, T. Lange et C. Peters : Twisted Edwards Curves. *AFRICACRYPT 2008, LNCS*, 5023:389–405, 2008.

[BDJ04] E. Brier, I. Déchène et M. Joye : Unified Point Addition Formulæfor Elliptic Curve Cryptosystems. *Embedded Cryptographic Hardware : Methodologies and Architectures. Nova Science Publishers*, pages 247–256, 2004.

[BDL97] D. Boneh, R.A. DeMillo et R.J. Lipton : On the Importance of Checking Cryptographic Protocols for Faults. *EUROCRYPT 1997, LNCS*, 1233:37–51, 1997.

[BDL01] D. Boneh, R.A. DeMillo et R.J. Lipton : On the Importance of Eliminating Errors in Cryptographic Computations. *Journal of Cryptology*, 14:101–119, 2001.

[BECN+06] H. Bar-El, H. Choukri, D. Naccache, M. Tunstall et C. Whelan : The Sorcerer's Apprentice Guide to Fault Attacks. *Proceedings of the IEEE*, 94:370–382, 2006. http ://eprint.iacr.org/2004/100.

[BEML10] J. Boxall, N. El Mrabet et D.-P. Laguillaumie, F. Le : A Variant of Miller's Formula and Algorithm. *PAIRING 2010, LNCS*, 6487:417–434, 2010.

[BF01] D. Boneh et M. Franklin : Identity-Based Encryption from the Weil Pairing. *CRYPTO 2001, LNCS*, 2139:213–229, 2001.

[BGhS07] P.S.L.M. Barreto, S.D. Galbraith, C.Ó. hÉigeartaigh et M. Scott : Efficient Pairing Computation on Supersingular Abelian Varieties. *Designs, Codes and Cryptography*, 42(3):239–271, 2007.

[BGK05] J. Blömer, J. Guajardo et V. Krummel : Provably Secure Masking of AES. *Selected Areas in Cryptography, LNCS*, 3357:69–83, 2005.

[BGLR08] L. Batina, B. Gierlichs et K. Lemke-Rust : Comparative Evaluation of Rank Correlation Based DPA on an AES Prototype Chip. *ISC 2008, LNCS*, 5222:341–354, 2008.

[BGLR09] L. Batina, B. Gierlichs et K. Lemke-Rust : Differential Cluster Analysis. *CHES 2009, LNCS*, 5747:112–127, 2009.

[BJ02] E. Brier et M. Joye : Weierstraß Elliptic Curves and Side-Channel Attacks. *PKC 2002, LNCS*, 2274:335–345, 2002.

[BJ03] O. Billet et M. Joye : The Jacobi Model of an Elliptic Curve and Side-Channel Analysis. *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes - AAECC, LNCS*, 2643:34–42, 2003.

[BK98] R. Balasubramanian et N. Koblitz : The Improbability that Elliptic Curve has Subexponential Discrete Log Problem under the Menezes-Okamoto-Vanstone Algorithm. *Journal of Cryptology*, 11:141–145, 1998.

[BKLS02] P.S.L.M. Barreto, H.Y. Kim, B. Lynn et M. Scott : Efficient Algorithms for Pairing-Based Cryptosystems. *CRYPTO 2002, LNCS*, 2442:354–368, 2002.

[BL07a] D. Bernstein et T. Lange : Explicit-Formulas Database, 2007. `http://www.hyperelliptic.org/EFD`.

[BL07b] D. Bernstein et T. Lange : Inverted Edwards Coordinates. *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, LNCS*, 4851:20–27, 2007.

[BL08] D. Bernstein et T. Lange : Faster Addition and Doubling on Elliptic Curves. *ASIACRYPT 2007, LNCS*, 4833:29–50, 2008.

[BLS04a] P.S.L.M. Barreto, B. Lynn et M. Scott : On the Selection of Pairing-Friendly Groups. *Selected Areas In Cryptography, LNCS*, 3006:17–25, 2004.

[BLS04b] D. Boneh, B. Lynn et H. Shacham : Short Signatures from the Weil Pairing. *Journal of Cryptology*, 17:297–319, 2004.

[BLV10] A. Bonnecaze, P. Liardet et A. Venelli : AES Side-Channel Countermeasure using Random Tower Fields Constructions. Preprint, 2010.

[BMM00] I. Biehl, B. Meyer et V. Müller : Differential Fault Attacks on Elliptic Curve Cryptosystems. *CRYPTO 2000, LNCS*, 1880:131–146, 2000.

[BN06] P.S.L.M. Barreto et M. Naehrig : Pairing-Friendly Elliptic Curves of Prime Order. *Selected areas in cryptography, LNCS*, 3897:319–331, 2006.

[BOS06] J. Blömer, M. Otto et J.P. Seifert : Sign Change Fault Attacks on Elliptic Curve Cryptosystems. *FDTC 2005, LNCS*, 4236:36–52, 2006.

[BS97] E. Biham et A. Shamir : Differential Fault Analysis of Secret Key Cryptosystems. *CRYPTO 1997, LNCS*, 1294:513–525, 1997.

[BS03] J. Blömer et J.P. Seifert : Fault Based Cryptanalysis of the Advanced Encryption Standard (AES). *Financial Cryptography, LNCS*, 2742:162–181, 2003.

[BSS05] I.F. Blake, G. Seroussi et N.P. Smart : *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005.

[Can05] D. Canright : A Very Compact S-box for AES. *CHES 2005, LNCS*, 3659: 441–455, 2005.

[CF06] H. Cohen et G. Frey : *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC Press, 2006.

[CGPR08] J.S. Coron, C. Giraud, E. Prouff et M. Rivain : Attack and Improvement of a Secure S-box Calculation Based on the Fourier Transform. *CHES 2008, LNCS*, 5154:1–14, 2008.

[Cha06] D. Charles : On the Existence of Distortion Maps on Ordinary Elliptic Curves. *Arxiv preprint math.NT/0603724*, 2006.

[Cie03] M. Ciet : *Aspects of Fast and Secure Arithmetics for Elliptic Curve Cryptography*. Thèse de doctorat, Université Catholique de Louvain, 2003.

[CJ01] C. Clavier et M. Joye : Universal Exponentiation Algorithm A First Step towards Provable SPA-Resistance. *CHES 2001, LNCS*, 2162:300–308, 2001.

[CJ03] M. Ciet et M. Joye : (Virtually) Free Randomization Techniques for Elliptic Curve Cryptography. *Information and Communications Security, LNCS*, 2836:348–359, 2003.

[CJ05] M. Ciet et M. Joye : Elliptic Curve Cryptosystems in the Presence of Permanent and Transient Faults. *Designs, Codes and Cryptography*, 36:33–43, 2005.

[CK09] J.S. Coron et I. Kizhvatov : Analysis of the Split Mask Countermeasure for Embedded Systems. *Proceedings of the 4th Workshop on Embedded Systems Security*, pages 1–10, 2009.

[CKQ03] J. Cathalo, F. Koeune et J.J. Quisquater : A New Type of Timing Attack : Application to GPS. *CHES 2003, LNCS*, 2779:291–303, 2003.

[CMCJ04] B. Chevallier-Mames, M. Ciet et M. Joye : Low-Cost Solutions for Preventing Simple Side-Channel Analysis : Side-Channel Atomicity. *IEEE Transactions on Computers*, 53:760–768, 2004.

[Com90] P.G. Comba : Exponentiation Cryptosystems on the IBM PC. *IBM Syst. J.*, 29:526–538, 1990.

[Coo66] S.A. Cook : *On the Minimum Computation Time of Functions*. Thèse de doctorat, Harvard University Department of Mathematics, 1966.

[Cor99] J.-S. Coron : Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. *CHES 1999, LNCS*, 1717:292–302, 1999.

[CR88] L.S. Charlap et D.P. Robbins : *An Elementary Introduction to Elliptic Curves*. CRD Expository Report No. 31, 1988.

[CRR02] S. Chari, J. Rao et P. Rohatgi : Template Attacks. *CHES 2002, LNCS*, 2523:51–62, 2002.

[DDhS06] R. Dahab, A. J. Devegili, C.Ó. hÉigeartaigh et M. Scott : Multiplication and Squaring on Pairing-Friendly Fields. Cryptology ePrint Archive, Report 2006/471, 2006.

[Deb78] C. Deboor : *A Practical Guide to Splines*. Springer-Verlag Berlin and Heidelberg GmbH & Co. K, December 1978.

[DH76] W. Diffie et M. Hellman : New Directions in Cryptography. *IEEE Transactions on Information Theory*, 22:644–654, 1976.

[DL03] I. Duursma et H.S. Lee : Tate-Pairing Implementations for Tripartite Key Agreement. *ASIACRYPT 2003, LNCS*, 2894:111–123, 2003.

[DSSK04] C. Daub, R. Steuer, J. Selbig et S. Kloska : Estimating Mutual Information Using B-spline Functions - an Improved Similarity Measure for Analysing Gene Expression Data. *BMC Bioinformatics*, 5:118, 2004.

[Duq07] S. Duquesne : Improving the Arithmetic of Elliptic Curves in the Jacobi Model. *Information Processing Letters*, 104:101–105, 2007.

[Edw07] H.M. Edwards : A Normal Form for Elliptic Curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007.

[ElG85] T. ElGamal : A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[ELM03] K. Eisenträger, K. Lauter et P.L. Montgomery : Fast Elliptic Curve Arithmetic and Improved Weil Pairing Evaluation. *CT-RSA 2003, LNCS*, 2612:343–354, 2003.

[EM09] N. El Mrabet : What about Vulnerability to a Fault Attack of the Miller's Algorithm During an Identity Based Protocol ? *Advances in Information Security and Assurance, LNCS*, 5576:122–134, 2009.

[EM10] N. El Mrabet : Fault Attacks against the Miller's Algorithm in Edwards Coordinates. *Information Security and Assurance*, 76:72–85, 2010.

[EMDNF09] N. El Mrabet, G. Di Natale et M.L. Flottes : A Practical Differential Power Analysis Attack against the Miller Algorithm. *Research in Microelectronics and Electronics, 2009. PRIME 2009. Ph.D.*, pages 308–311, 2009.

[Eng99] A. Enge : *Elliptic Curves and their Applications to Cryptography : an Introduction*. Kluwer Academic Publishers, 1999.

[FGD+10] F. Flament, S. Guilley, J.L. Danger, M.A. Elaabid, H. Maghrebi et L. Sauvage : About Probability Density Function Estimation for Side Channel Analysis. *In COSADE 2010*, 2010.

[FJ10] R. Farashahi et M. Joye : Efficient Arithmetic on Hessian Curves. *PKC 2010, LNCS*, 6056:243–260, 2010.

[FLRV08] P.A. Fouque, R. Lercier, D. Réal et F. Valette : Fault Attack on Elliptic Curve Montgomery Ladder Implementation. *FDTC 2008, LNCS*, pages 92–98, 2008.

[FMPR10] G. Fumaroli, A. Martinelli, E. Prouff et M. Rivain : Affine Masking against Higher-Order Side Channel Analysis. Cryptology ePrint Archive, Report 2010/523, 2010.

[FMR99] G. Frey, M. Müller et H.G. Rück : The Tate Pairing and the Discrete Logarithm Applied to Elliptic Curve Cryptosystems. *IEEE Transactions on Information Theory*, 45:1717–1719, 1999.

[FR94] G. Frey et H.G. Rück : A Remark Concerning the m-Divisibility and the Discrete Logarithm in the Divisor Class Group of Curves. *Mathematics of Computation*, 62(206):865–874, 1994.

[Fre06] D. Freeman : Constructing Pairing-Friendly Elliptic Curves with Embedding Degree 10. *Algorithmic Number Theory, LNCS*, 4076:452–465, 2006.

[FRVD08] P.A. Fouque, D. Réal, F. Valette et M. Drissi : The Carry Leakage on the Randomized Exponent Countermeasure. *CHES 2008, LNCS*, 5154:198–213, 2008.

[FST10] D. Freeman, M. Scott et E. Teske : A Taxonomy of Pairing-Friendly Elliptic Curves. *Journal of Cryptology*, 23:224–280, 2010.

[Ful89] W. Fulton : *Algebraic Curves*. Addison-Wesley Publishing Company Advanced Book Program, 1989.

[FV03] P.A. Fouque et F. Valette : The Doubling Attack–Why Upwards is Better than Downwards. *CHES 2003, LNCS*, 2779:269–280, 2003.

[Gal01] S.D. Galbraith : Supersingular Curves in Cryptography. *ASIACRYPT 2001, LNCS*, 2248:495–513, 2001.

[GBTP08] B. Gierlichs, L. Batina, P. Tuyls et B. Preneel : Mutual Information Analysis - A Generic Side-Channel Distinguisher. *CHES 2008, LNCS*, 5154: 426–442, 2008.

[Geb06] C.H. Gebotys : A Split-Mask Countermeasure for Low-Energy Secure Embedded Systems. *ACM Transactions on Embedded Computing Systems (TECS)*, 5:577–612, 2006.

[GHS07] S.D. Galbraith, C.O. HEigeartaigh et C. Sheedy : Simplified Pairing Computation and Security Implications. *Journal of Mathematical Cryptology*, 1:267–281, 2007.

[GJM10] R. Goundar, M. Joye et A. Miyaji : Co-Z Addition Formulæ and Binary Ladders. *CHES 2010, LNCS*, 6225:65–79, 2010.

[GK54] L.A. Goodman et W.H. Kruskal : Measures of Association for Cross Classifications. II : Further Discussion and References. *Journal of the American Statistical Association*, 49:732–764, 1954.

[GLV01] R. Gallant, R. Lambert et S. Vanstone : Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms. *CRYPTO 2001, LNCS*, 2139:190–200, 2001.

[GMO01] K. Gandolfi, C. Mourtel et F. Olivier : Electromagnetic Analysis : Concrete Results. *CHES 2001, LNCS*, 2162:251–261, 2001.

[Gou03] L. Goubin : A Refined Power-Analysis Attack on Elliptic Curve Cryptosystems. *PKC 2003, LNCS*, 2567:199–211, 2003.

[GPW+04] N. Gura, A. Patel, A. Wander, H. Eberle et S.C. Shantz : Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs. *CHES 2004, LNCS*, 3156:925–943, 2004.

[GR10] S. Galbraith et V. Rotger : Easy decision-Diffie-Hellman groups. *LMS Journal of Computation and Mathematics*, 7:201–218, 2010.

[GT02] J. Golić et C. Tymen : Multiplicative Masking and Power Analysis of AES. *CHES 2002, LNCS*, 2535:31–47, 2002.

[GV10] C. Giraud et V. Verneuil : Atomicity Improvement for Elliptic Curve Scalar Multiplication. *CARDIS 2010, LNCS*, 6035:80–101, 2010.

[HCD07] H. Hisil, G. Carter et E. Dawson : New Formulae for Efficient Elliptic Curve Arithmetic. *INDOCRYPT 2007, LNCS*, 4859:138–151, 2007.

[Hes04] F. Hess : A Note on the Tate Pairing of Curves over Finite Fields. *Archiv der Mathematik*, 82:28–32, 2004.

[Hes08] F. Hess : Pairing Lattices. *PAIRING 2008, LNCS*, 5209:18–38, 2008.

[HMA+08] N. Homma, A. Miyamoto, T. Aoki, A. Satoh et A. Shamir : Collision-Based Power Analysis of Modular Exponentiation Using Chosen-Message Pairs. *CHES 2008, LNCS*, 5154:15–29, 2008.

[HQ00] G. Hachez et J.J. Quisquater : Montgomery Exponentiation with no Final Subtractions : Improved Results. *CHES 2000, LNCS*, 1965:91–100, 2000.

[HSV06] F. Hess, N.P. Smart et F. Vercauteren : The Eta Pairing Revisited. *IEEE transactions on information theory*, 52(10):4595–4602, 2006.

[HVM04] D.R. Hankerson, S.A. Vanstone et A.J. Menezes : *Guide to Elliptic Curve Cryptography*. Springer-Verlag New York Inc, 2004.

[HWCD08] H. Hisil, K. Wong, G. Carter et E. Dawson : Twisted Edwards Curves Revisited. *ASIACRYPT 2008, LNCS*, 5350:326–343, 2008.

[HWCD09a] H. Hisil, K. Wong, G. Carter et E. Dawson : Jacobi Quartic Curves Revisited. *Information Security and Privacy, LNCS*, 5594:452–468, 2009.

[HWCD09b] H. Hisil, K.K.H. Wong, G. Carter et E. Dawson : Faster Group Operations on Elliptic Curves. *AISC 2009*, 98:7–19, 2009.

[IIT03a] K. Itoh, T. Izu et M. Takenaka : A Practical Countermeasure against Address-Bit Differential Power Analysis. *CHES 2003, LNCS*, 2779:382–396, 2003.

[IIT03b] K. Itoh, T. Izu et M. Takenaka : Address-Bit Differential Power Analysis of Cryptographic Schemes OK-ECDH and OK-ECDSA. *CHES 2002, LNCS*, 2523:399–412, 2003.

[IT02] T. Izu et T. Takagi : A Fast Parallel Elliptic Curve Multiplication Resistant against Side Channel Attacks. *PKC 2002, LNCS*, 2274:371–374, 2002.

[IT03a] T. Izu et T. Takagi : Efficient Computations of the Tate Pairing for the Large MOV Degrees. *IEIC Technical Report (Institute of Electronics, Information and Communication Engineers)*, 102(746):47–52, 2003.

[IT03b] T. Izu et T. Takagi : Exceptional Procedure Attack on Elliptic Curve Cryptosystems. *PKC 2003, LNCS*, 2567:224–239, 2003.

[ITT02] K. Itoh, M. Takenaka et N. Torii : DPA Countermeasure Based on the "Masking Method". *ICISC 2001, LNCS*, 2288:440–456, 2002.

[Jou00] A. Joux : A One Round Protocol for Tripartite Diffie-Hellman. *ANTS-IV, LNCS*, 1883:385–394, 2000.

[Joy07] M. Joye : Highly Regular Right-to-Left Algorithms for Scalar Multiplication. *CHES 2007, LNCS*, 4727:135–147, 2007.

[JQ96] M. Joye et J.J. Quisquater : Efficient Computation of Full Lucas Sequences. *Electronics Letters*, 32:537–538, 1996.

[JQ01] M. Joye et J.J. Quisquater : Hessian Elliptic Curves and Side-Channel Attacks. *CHES 2001, LNCS*, 2162:402–410, 2001.

[JT01] M. Joye et C. Tymen : Protections against Differential Analysis for Elliptic Curve Cryptography – An Algebraic Approach. *CHES 2001, LNCS*, 2162: 377–390, 2001.

[JTV10] M. Joye, M. Tibouchi et D. Vergnaud : Huff's Model for Elliptic Curves. *Algorithmic Number Theory, LNCS*, 6197:234–250, 2010.

[JY02] M. Joye et S.M. Yen : The Montgomery Powering Ladder. *CHES 2002, LNCS*, 2523:1–11, 2002.

[Ken38] M.G. Kendall : A New Measure of Rank Correlation. *Biometrika*, 30:1–2, 1938.

[KJJ99] P. Kocher, J. Jaffe et B. Jun : Differential Power Analysis. *CRYPTO 1999, LNCS*, 1666:388–397, 1999.

[KK99] O. Kömmerling et M.G. Kuhn : Design Principles for Tamper-Resistant Smartcard Processors. *In USENIX Workshop on Smartcard Technology*, pages 9–20, 1999.

[KM05] N. Koblitz et A. Menezes : Pairing-Based Cryptography at High Security Levels. *Cryptography and Coding, LNCS*, 3796:13–36, 2005.

[Knu98] D. Knuth : *The Art of Computer Programming*. Addison-Wesley, 1998.

[KO63] A. Karatsuba et Y. Ofman : Multiplication of Multidigit Numbers on Automata. *Soviet Physics Doklady*, 7:595–596, 1963.

[Kob87] N. Koblitz : Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48:203–209, 1987.

[Koc96] P. Kocher : Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems. *CRYPTO 1996, LNCS*, 1109:104–113, 1996.

[Koç09] C.K. Koç, éditeur. *Cryptographic Engineering*. Springer-Verlag, 2009.

[KQ08] C. Kim et J.J. Quisquater : Method for Detecting Vulnerability to Doubling Attacks. *Information and Communications Security, LNCS*, 5308:97–110, 2008.

[KSG04] A. Kraskov, H. Stogbauer et P. Grassberger : Estimating Mutual Information. *Physical Review E*, 69:66138, 2004.

[KTH+06] T. Kim, T. Takagi, D.G. Han, H. Kim et J. Lim : Side Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields. *Cryptology and Network Security, LNCS*, 4301:168–181, 2006.

[KTH+08] T.H. Kim, T. Takagi, D.G. Han, H. Kim et J. Lim : Power Analysis Attacks and Countermeasures on ηT Pairing over Binary Fields. *ETRI Journal*, 30:68–80, 2008.

[LB10] T.-H. Lee et M. Berthier : Mutual Information Analysis under the View of Higher-Order Statistics. *Advances in Information and Computer Security, LNCS*, 6434:285–300, 2010.

[LD99a] J. López et R. Dahab : Fast Multiplication on Elliptic Curves over GF(2^m) without Precomputation. *CHES 1999, LNCS*, 1717:316–328, 1999.

[LD99b] J. López et R. Dahab : Improved Algorithms for Elliptic Curve Arithmetic in GF(2n). *SAC 1998, LNCS*, 1556:201–212, 1999.

[LLMP90] A.K. Lenstra, H. W. Lenstra, M. S. Manasse et J. M. Pollard : The Number Field Sieve. *In STOC 1990, ACM Symposium on Theory of Computing*, pages 564–572, 1990.

[LLP09] E. Lee, H.-S. Lee et C.-M. Park : Efficient and Generalized Pairing Computation on Abelian Varieties. *IEEE Transactions on Information Theory*, 55:1793–1803, 2009.

[LMN10] K. Lauter, P. L. Montgomery et M. Naehrig : An Analysis of Affine Coordinates for Pairing Computation. *PAIRING 2010, LNCS*, 6487:1–20, 2010.

[Lon07] P. Longa : *Accelerating the Scalar Multiplication on Elliptic Curve Cryptosystems over Prime Fields*. Thèse de doctorat, School of Information Technology and Engineering, University of Ottawa, 2007.

[LS01] P. Liardet et N. Smart : Preventing SPA/DPA in ECC Systems using the Jacobi Form. *CHES 2001, LNCS*, 2162:391–401, 2001.

[LSK+09] Y. Li, K. Sakiyama, S. Kawamura, Y. Komano et K. Ohta : Security Evaluation of a DPA-Resistant S-Box Based on the Fourier Transform. *Information and Communications Security, LNCS*, 5927:3–16, 2009.

[Man02] S. Mangard : A Simple Power-Analysis (SPA) Attack on Implementations of the AES Key Expansion. *ICISC 2002, LNCS*, 2587:343–358, 2002.

[MDS99a] T. S. Messerges, E. A. Dabbish et R. H. Sloan : Investigations of Power Analysis Attacks on Smartcards. *In USENIX Workshop on Smartcard Technology*, pages 151–162, 1999.

[MDS99b] T. S. Messerges, E. A. Dabbish et R. H. Sloan : Power Analysis Attacks of Modular Exponentiation in Smartcard. *CHES 1999, LNCS*, 1717:144–157, 1999.

[Mel07] N. Meloni : New Point Addition Formulae for ECC Applications. *Arithmetic of Finite Fields, LNCS*, 4547:189–201, 2007.

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

[Mes01] T. Messerges : Securing the AES Finalists against Power Analysis Attacks. *Fast Software Encryption, LNCS*, 1978:293–301, 2001.

[Mil86a] V. Miller : Short Programs for Functions on Curves. 1986.

[Mil86b] V. Miller : Use of Elliptic Curve in Cryptology. *CRYPTO 1986, LNCS*, 218:417–426, 1986.

[MKHO07] S. Matsuda, N. Kanayama, F. Hess et E. Okamoto : Optimised Versions of the Ate and Twisted Ate Pairings. *Cryptography and Coding, LNCS*, 4887: 302–312, 2007.

[MMPS09] A. Moradi, N. Mousavi, C. Paar et M. Salmasizadeh : A Comparative Study of Mutual Information Analysis under a Gaussian Assumption. *Information Security Applications, LNCS*, 5932:193–205, 2009.

[MMS01] D. May, H. Muller et N. Smart : Random Register Renaming to Foil DPA. *CHES 2001, LNCS*, 2162:28–38, 2001.

[MO90] F. Morain et J. Olivos : Speeding up the Computations on an Elliptic Curve using Addition-Subtraction Chains. *Theoretical Informatics and Applications*, 24:531–543, 1990.

[Mon85] P.L. Montgomery : Modular Multiplication without Trial Division. *Mathematics of computation*, 44:519–521, 1985.

[Mon87] P.L. Montgomery : Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of Computation*, 48:243–264, 1987.

[MOV93] A.J. Menezes, T. Okamoto et S.A. Vanstone : Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. *IEEE Transactions on Information Theory*, 39:1639–1646, 1993.

[MPO05] S. Mangard, N. Pramstaller et E. Oswald : Successfully Attacking Masked AES Hardware Implementations. *CHES 2005, LNCS*, 3659:157–171, 2005.

[MRL95] Y.-I. Moon, B. Rajagopalan et U. Lall : Estimation of Mutual Information using Kernel Density Estimators. *Physical Review E*, 52(3):2318–2321, 1995.

[MS02] S. Morioka et A. Satoh : An Optimized S-Box Circuit Architecture for Low Power AES Design. *CHES 2002, LNCS*, 2523:271–295, 2002.

[NAS+08] Y. Nogami, M. Akane, Y. Sakemi, H. Kato et Y. Morikawa : Integer Variable X-based Ate Pairing. *PAIRING 2008, LNCS*, 5209:178–191, 2008.

[Nat93] National Institute Standards and Technology : Data Encryption Standard (DES). Publication 46–2, 1993.

[Nat00] National Institute Standards and Technology : Difital Signature Standard (DSS). Publication 186–2, 2000.

[Nat01] National Institute Standards and Technology : Advanced Encryption Standard (AES). Publication 197, 2001.

[NNT+10] Y. Nogami, K. Nekado, T. Toyota, N. Hongo et Morikawa Y. : Mixed Bases for Efficient Inversion in $\mathbb{F}_{((2^2)^2)^2}$ and Conversion Matrices of SubBytes of AES. *CHES 2010, LNCS*, 6225:234–247, 2010.

[OM07] E. Oswald et S. Mangard : Template Attacks on Masking – Resistance is Futile. *CT-RSA 2007, LNCS*, 4377:243–256, 2007.

[OMP04] E. Oswald, S. Mangard et N. Pramstaller : Secure and Efficient Masking of AES-A Mission Impossible. Cryptology ePrint Archive, Report 2004/134, 2004.

[OMPR05] E. Oswald, S. Mangard, N. Pramstaller et V. Rijmen : A Side-Channel Analysis Resistant Description of the AES S-Box. *Fast Software Encryption, LNCS*, 3557:413–423, 2005.

[OS06] E. Oswald et K. Schramm : An Efficient Masking Scheme for AES Software Implementations. *Information Security Applications, LNCS*, 3786:292–305, 2006.

[Ott04] M. Otto : *Fault Attacks and Countermeasures*. Thèse de doctorat, University of Paderborn, 2004.

[Par62] E. Parzen : On the Estimation of a Probability Density Function and Mode. *Annals of Mathematical Statistics*, 33:1065–1076, 1962.

[PBHE98] B. Pompe, P. Blidh, D. Hoyer et M. Eiselt : Using Mutual Information to Measure Coupling in the Cardio Respiratory System. *IEEE Engineering in Medicine and Biology Magazine*, 17(6):32–39, 1998.

[PGA06] E. Prouff, C. Giraud et S. Aumônier : Provably Secure S-box Implementation Based on Fourier Transform. *CHES 2006, LNCS*, 4249:216–230, 2006.

[PH95] B. Pompe et M. Heilfort : On the Concept of the Generalized Mutual Information Function and Efficient Algorithms for Calculing it. 1995.

[PP93] B. Pompe et F. Physik : Measuring Statistical Dependences in a Time Series. *Journal of Statistical Physics*, 73:587–610, 1993.

[PR09] E. Prouff et M. Rivain : Theorical and Practical Aspects of Mutual Information Based Side Channel Analysis. *ACNS 2009, LNCS*, 5536:499–518, 2009.

[PV04] D. Page et F. Vercauteren : Fault and Side-Channel Attacks on Pairing Based Cryptography. Cryptology ePrint Archive, Report 2004/283, 2004.

[PV06] D. Page et F. Vercauteren : A Fault Attack on Pairing-Based Cryptography. *IEEE Transactions on Computers*, 55:1075–1080, 2006.

[QS01] J.J. Quisquater et D. Samyde : Electromagnetic Analysis (EMA) : Measures and Counter-Measures for Smart Cards. *Smart Card Programming and Security, LNCS*, 2140:200–210, 2001.

[RDJ+01] A. Rudra, P. Dubey, C. Jutla, V. Kumar, J. Rao et P. Rohatgi : Efficient Rijndael Encryption Implementation with Composite Field Arithmetic. *CHES 2001, LNCS*, 2162:171–184, 2001.

[Rei60] G.W. Reitwiesner : Binary Arithmetic. *Advances in computers*, 1:231–308, 1960.

[RO05] C. Rechberger et E. Oswald : Practical Template Attacks. *Information Security Applications, LNCS*, 3325:440–456, 2005.

[RS05] A.G. Rostovtsev et O.V. Shemyakina : AES Side Channel Attack Protection Using Random Isomorphisms. Cryptology ePrint Archive, Report 2005/087, 2005.

[SA03] S. Skorobogatov et R. Anderson : Optical Fault Induction Attacks. *CHES 2002, LNCS*, 2523:31–48, 2003.

[Sco05a] M. Scott : Computing the Tate Pairing. *CT-RSA 2005, LNCS*, 3376:293–304, 2005.

[Sco05b] M. Scott : Faster Pairings Using an Elliptic Curve with an Efficient Endomorphism. *INDOCRYPT 2005, LNCS*, 3797:258–269, 2005.

[SEC00] SEC2 : Standards for Efficient Cryptography Group/Certicom Research. Recommanded Elliptic Curve Cryptography Domain Parameters, 2000.

[SGV08] F.-X. Standaert, B. Gierlichs et I. Verbauwhede : Partition vs . Comparison Side-Channel Distinguishers : An Empirical Evaluation of Statistical Tests for Univariate Side-Channel Attacks against Two Unprotected CMOS Devices. *ICISC 2008, LNCS*, 5461:253–267, 2008.

[Sha99] A. Shamir : Method and Apparatus for Protecting Public Key Schemes from Timing and Fault Attack. Patent (5991415), 1999.

[Sil86] J.H. Silverman : *The Arithmetic of Elliptic Curves*. Springer-Verlag, 1986.

[SKNM08] Y. Sakemi, H. Kato, Y. Nogami et Y. Morikawa : An Improvement of Twisted Ate Pairing Using Integer Variable with Small Hamming Weight. *In Proc. of the 23rd International Technical Conference on Circuits/Systems, Computers and Communications*, pages 269–272, 2008.

[Sko05] S. P. Skorobogatov : Semi-Invasive Attacks – A New Approach to Hardware Security Analysis. Rapport technique, University of Cambridge, Computer Laboratory, 2005. `http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.pdf`.

[SS07] M. Scott et P. Szczechowiak : Optimizing Multiprecision Multiplication for Public Key Cryptography. Cryptology ePrint Archive, Report 2007/299, 2007.

[SSAQ02] D. Samyde, S. Skorobogatov, R. Anderson et J.J. Quisquater : On a New Way to Read Data from Memory. *In Proceedings of the First International IEEE Security in Storage Workshop*, pages 65–69, 2002.

[ST06] D. Stebila et N. Thériault : Unified point addition formulæ and side-channel attacks. *CHES 2006, LNCS*, 4249:354–368, 2006.

[STO08] M. Shirase, T. Takagi et E. Okamoto : An Efficient Countermeasure against Side Channel Attacks for Pairing Computation. *Information Security Practice and Experience*, 4991:290–303, 2008.

[Sul08] N.T. Sullivan : Fast Algorithms for Arithmetic on Elliptic Curves Over Prime Fields. Cryptology ePrint Archive, Report 2008/060, 2008.

[SWP03] K. Schramm, T. Wollinger et C. Paar : A New Class of Collision Attacks and its Application to DES. *Fast Software Encryption, LNCS*, 2887:206–222, 2003.

[TB02] E. Trichina et A. Bellezza : Implementation of Elliptic Curve Cryptography with Built-In Counter Measures against Side Channel Attacks. *CHES 2002, LNCS*, 2523:297–312, 2002.

[TDSG03] E. Trichina, D. De Seta et L. Germani : Simplified Adaptive Multiplicative Masking for AES. *CHES 2002, LNCS*, 2523:71–85, 2003.

[TJ10] M. Tunstall et M. Joye : Coordinate Blinding over Large Prime Fields. *CHES 2010, LNCS*, 6225:443–455, 2010.

[TK05] E. Trichina et L. Korkishko : Secure and Efficient AES Software Implementation for Smart Cards. *Information Security Applications, LNCS*, 3325:425–439, 2005.

[Too63] A.L. Toom : The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers. *Soviet Mathematics Doklady*, 3:714–716, 1963.

[VCS09] N. Veyrat-Charvillon et F.X. Standaert : Mutual Information Analysis : How, When and Why ? *CHES 2009, LNCS*, 5747:429–443, 2009.

[VD10a] A. Venelli et F. Dassance : Fast Scalar Multiplication for Elliptic Curve Cryptosystems over Prime Fields. Patent (20100040225), 2010.

[VD10b] A. Venelli et F. Dassance : Faster Side-Channel Resistant Elliptic Curve Scalar Multiplication. *Arithmetic, Geometry, Cryptography and Coding Theory 2009, Contemporary Mathematics*, 521:29–40, 2010.

[VD10c] A. Venelli et F. Dassance : Side-Channel Resistant Scalar Multiplication Algorithms over Finite Fields. *In 5th Conf. on Network Architectures and Information Systems Security*, 2010.

[Ven10a] A. Venelli : Analysis of Nonparametric Estimation Methods for Mutual Information Analysis. *In To appear in ICISC 2010*, 2010.

[Ven10b] A. Venelli : Efficient Entropy Estimation for Mutual Information Analysis using B-splines. *WISTP 2010, LNCS*, 6033:17–30, 2010.

[Ven10c] A. Venelli : Yet Another Crypto Library (YACL), 2010. `http://code.google.com/p/yacl/`.

[Ver04] E.R. Verheul : Evidence that XTR Is More Secure than Supersingular Elliptic Curve Cryptosystems. *Journal of Cryptology*, 17(4):277–296, 2004.

[Ver10] F. Vercauteren : Optimal Pairings. *IEEE Transactions on Information Theory*, 56:455–461, 2010.

[VLSa] VLSI research group and TELECOM ParisTech : The DPA Contest 2008/2009. `http://www.dpacontest.org`.

[VLSb] VLSI research group and TELECOM ParisTech : The DPA Contest 2009/2010. http://www.dpacontest.org/v2.

[VMAG99] S. Vanstone, R. Mullin, A. Antipa et R. Gallant : Accelerated Finite Field Operations on an Elliptic Curve. Patent (049386), 1999.

[Wal04] C.D. Walter : Simple Power Analysis of Unified Code for ECC Double and Add. *CHES 2004, LNCS*, 3156:86–115, 2004.

[Was08] L.C. Washington : *Elliptic Curves Number Theory and Cryptography - Second Edition*. CRC Press, 2008.

[WOL02] J. Wolkerstorfer, E. Oswald et M. Lamberger : An ASIC Implementation of the AES SBoxes. *CT-RSA 2002, LNCS*, 2271:29–52, 2002.

[WS06] C. Whelan et M. Scott : Side Channel Analysis of Practical Pairing Implementations : Which Path is More Secure ? *VIETCRYPT 2006, LNCS*, 4341:99–114, 2006.

[WS07] C. Whelan et M. Scott : The Importance of the Final Exponentiation in Pairings When Considering Fault Attacks. *PAIRING 2007, LNCS*, 4575:225–246, 2007.

[Wu10] B.C. Wu : Enhancement on Scalar Multiplication in Elliptic Curve Cryptosystems. Mémoire de D.E.A., National Central University, Taiwan, 2010.

[X9.98] ANSI X9.62 : Public Key Cryptography for the Financial Services Industry : The Elliptic Curve Digital Signature Algorithm (ECDSA). Cornell University, Research Report, 1998.

[YJ00] S.M. Yen et M. Joye : Checking Before Output May Not Be Enough Against Fault-Based Cryptanalysis. *IEEE Transactions on Computers*, 49:967–970, 2000.

[YKLM02] S.M. Yen, S. Kim, S. Lim et S. Moon : A Countermeasure against One Physical Cryptanalysis May Benefit Another Attack. *ICISC 2001, LNCS*, 2288:414–427, 2002.

[YKMH06] S.M. Yen, L.C. Ko, S.J. Moon et J.C. Ha : Relative Doubling Attack Against Montgomery Ladder. *ICISC 2005, LNCS*, 3935:117–128, 2006.

[YLT04] S.M. Yen, C.C. Lu et S.Y. Tseng : Method for Protecting Public Key Schemes from Timing, Power and Fault attacks. US Patent Number US2004/0125950, 2004.

[YMH03] S.M. Yen, S. Moon et J.C. Ha : Hardware Fault Attack on RSA with CRT Revisited. *ICISC 2002, LNCS*, 2587:374–388, 2003.

[ZZ08] C.-A. Zhao et F. Zhang : Computing the Bilinear Pairings on Elliptic Curves with Automorphisms. *Designs, Codes and Cryptography*, pages 1–10, 2008.

[ZZH08] C.-A. Zhao, F. Zhang et J. Huang : A Note on the Ate Pairing. *International Journal of Information Security*, 7:379–382, 2008.

Acronymes

ADPA Attaque sur les bits d'adresse en mémoire (*Address-bit DPA*). 85

AES *Advanced Encryption Standard*. xv, xvii, 11, 16, 19–21, 23–25, 27, 31, 32, 50, 53, 159, 161

BKLS Algorithme de calcul du couplage de Tate combiné avec la définition simplifiée de celui-ci. L'algorithme a été proposé par les auteurs Barreto, Kim, Lynn et Scott, on le nomme donc BKLS. 135, 147, 148

BSE Technique d'estimation non-paramétrique d'information mutuelle utilisant les B-splines (*B-Spline Estimation*). 50, 51, 53

CE Technique d'estimation paramétrique d'information mutuelle utilisant les cumulants (*Cumulant-based Estimation*). 50, 51, 53

CMOS *Complementary Metal Oxyde Semiconductor*. xvii, 7, 9, 24, 31, 56, 161

CPA Attaque par analyse différentielle de courant utilisant le coefficient de Pearson (*Correlation Power Analysis*). 16, 37, 50, 51, 53, 159

CVM Attaque par analyse différentielle utilisant le test Cramér-von Mises. 18, 47, 50, 51, 53

CVMB Attaque par analyse différentielle de courant utilisant le test Cramér-von Mises et le lissage avec des B-splines. 50, 51, 53

DCA Attaque par analyse différentielle utilisant le partitionnement de données (*Differential Cluster Analysis*). 16, 18, 50, 51, 53

DES *Data Encryption Standard*. xv, 11, 14, 19, 36, 45, 50

DFT Transformation de Fourier discrète (*Discrete Fourier Transform*). 22

DLP Problème du logarithme discret (*Discrete Logarithm Problem*). xv, xvi

DPA Attaque par analyse différentielle de courant (*Differential Power Analysis*). 11, 13, 14, 16, 20, 21, 47, 50, 51, 53

DSCA Attaque par analyse différentielle (*Differential Side-Channel Analysis*). 10, 11

ECC Cryptosystème à base de courbes elliptiques (*Elliptic Curve Cryptosystem*). xvi, xviii, 65, 160, 161

ECDLP Problème du logarithme discret basé sur les courbes elliptiques (*Elliptic Curve Discrete Logarithm Problem*). xvi, 65

ECDSA *Elliptic Curve Digital Signature Algorithm*. 79

GMI Information mutuelle généralisée (*Generalized Mutual Information*). 35

GMIA Attaque par analyse différentielle de courant utilisant l'information mutuelle généralisée (*Generalized Mutual Information Analysis*). 50, 51, 53

HE Technique d'estimation non-paramétrique d'information mutuelle utilisant les histogrammes (*Histogram Estimation*). 50, 51, 53

HODPA Attaque par analyse différentielle de courant d'ordre supérieure (*Higher-Order Differential Power Analysis*). 20, 23

IBE Chiffrement basé sur l'identité (*Identity-Based Encryption*). xvi

IM Information mutuelle. 34, 35, 46

K-S Attaque par analyse différentielle utilisant le test Kolmogorov-Smirnov. 18, 47

KDE Technique d'estimation non-paramétrique d'information mutuelle utilisant les noyaux (*Kernel Density Estimation*). 50, 51, 53

KNN K-plus-proches voisins (*K-Nearest Neighbors*). 39, 50, 51, 53

MESD Attaque multiple exposants une donnée (*Multiple Exponent Single Data*). 84, 85

MIA Attaque par analyse différentielle de courant utilisant l'information mutuelle (*Mutual Information Analysis*). 36, 37, 51, 53, 56, 159, 161

ML X-only Échelle de Montgomery calculé sans la coordonnée y des points. 107

ML+2ZADDC Échelle de Montgomery où l'algorithme ZADDC est utilisé deux fois par tour de boucle. 104, 105

ML+2ZADDCwoZ Échelle de Montgomery où l'algorithme ZADDCwoZ est utilisé deux fois par tour de boucle. 105

ML+ZADDC+ZADDU Échelle de Montgomery où les algorithmes ZADDC et ZADDU sont utilisés une fois chacun par tour de boucle. 104

ML+ZADDCwoZ+ZADDUwoZ Échelle de Montgomery où les algorithmes ZADDCwoZ et ZADDUwoZ sont utilisés une fois chacun par tour de boucle. 105, 107

ML+ZDAU Échelle de Montgomery où l'algorithme ZDAU est utilisé une fois par tour de boucle. 105

NAF Forme non-adjacente (*Non-Adjacent Form*). 74, 75, 80

NMOS *N-channel Metal Oxyde Semiconductor*. 7

PA Attaque par analyse de courant (*Power Analysis*). 8

PBC Cryptographie à base de couplages (*Pairing Based Cryptography*). 145

PMOS *P-channel Metal Oxyde Semiconductor*. 7

RPA Attaque sur des points « spéciaux » (*Refined Power Analysis*). 85

RSA Algorithme de Rivest Shamir Adelman. 6, 77, 83, 98, 160

RSA-CRT Algorithme de Rivest Shamir Adelman utilisant le thèorème des restes chinois (*RSA with Chinese Remainder Theorem*). 4

SCA Attaque par canaux cachés (*Side-Channel Analysis*). 2, 6, 159–161

SEMA Attaque par analyse simple d'émissions électromagnétiques (*Simple ElectroMagnetic Analysis*). 10

SEMD Attaque un exposant multiple données (*Single Exponent Multiple Data*). 83, 84

SPA Attaque par analyse simple de courant (*Simple Power Analysis*). 10

SPE Attaque par analyse différentielle de courant utilisant le coefficient de Spearman. 50, 51, 53

SSCA Attaque par analyse simple (*Simple Side-Channel Analysis*). 10, 11

ZADDC Algorithme d'addition et soustraction de points où les deux points en entrée ont la même coordonnée Z (*conjugate co-Z addition*). 103–105

ZADDCwoZ Algorithme d'addition et soustraction de points où les deux points en entrée ont la même coordonnée Z mais cette coordonnée n'est pas calculée (*conjugate co-Z addition without Z coordinate*). 105

ZADDU Algorithme d'addition de points où les deux points en entrée ont la même coordonnée Z (*co-Z addition with update*). 69, 102, 104, 105

ZADDUwoZ Algorithme d'addition de points où les deux points en entrée ont la même coordonnée Z mais cette coordonnée n'est pas calculée (*co-Z addition with update without Z coordinate*). 105

ZDAU Algorithme de doublement et addition de points où les deux points en entrée ont la même coordonnée Z (*co-Z double-add with update*). 105

ZEMD Attaque zero exposant multiple données (*Zero Exponent Multiple Data*). 84

ZPA *Zero-value Point Attack*. 85

ZPNU Algorithme naïf de mise à jour de la coordonnée Z du point P (Z_P *naive update*). 103