Table des matières

Liste des Figures

Liste des Tables

Liste des Algorithmes

Chapitre 1

Introduction

« Unie à l'océan, la goutte d'eau demeure »

Proverbe indien

Sommaire

1.1 L'évolution des besoins en sécurité de l'information

Le besoin de protéger une information a toujours existé. Depuis l'Antiquité jusqu'à la fin du XIX^{e} siècle, l'objectif premier est de protéger l'information contenue d'une communication entre deux entités.

Au fil des siècles, la sécurité de l'information, au sens large, a évolué. Il ne s'agit plus forcément de sécuriser une communication entre deux entités, mais d'échanges entre des systèmes qui communiquent. La cryptographie peut être décrite en trois âges : l'âge artisanal, l'âge mécanique et l'âge mathématique.

1.1.1 Âge 1 : le message secret

L'exemple du premier secret connu remonte à l'Antiquité. Un potier voulant conserver une recette secrète avait gravé une recette en supprimant des consonnes et en modifiant l'orthographe des mots [Kah96]. Au I^{e} siècle av. J.-C., Jules César souhaitait communiquer de façon secrète avec ses armées. Le premier code est né, il s'agit d'une substitution monoalphabétique, la faible alphabétisation de la population de l'époque la rendant suffisamment efficace. Les besoins évoluent, la transmission à distance est une problématique que le carré de Polybe peut résoudre [pol04]. L'utilisation d'un carré permettait la substitution homophonique et les nombres obtenus étaient transmis au moyen de torches. Par exemple, la lettre e correspond à la position $(1, 5)$ sur le carré, il suffit donc d'allumer une torche à droite et cinq à gauche pour la transmettre.

Au XVI^{e} siècle, Vigenère publie une méthode introduisant la notion de clef [dV86]. Une clef est généralement un mot ou une phrase. Le chiffrement d'un texte, se fait en prenant chacun des caractères, et en le substituant à l'aide d'une lettre de la clef. Plus la clef est longue et variée et « plus » le message chiffré est illisible. Les plus grands secrets pouvaient même être chiffrés avec une œuvre littéraire complète comme clef. Il suffisait pour les deux correspondants d'avoir en possession un exemplaire du même livre pour s'assurer de la bonne compréhension des messages. L'utilisation de ce code est devenue obsolète en 1863, due à l'analyse fréquentielle faite par Kasiski [Kas63].

1.1.2 Âge 2 : protéger le système de communications

Lors de la Première Guerre mondiale, des messages étaient chiffrés avant leurs transmissions, par télégrammes notamment. Cependant, les méthodes de chiffrement et de déchiffrement étaient encore manuelles et donc très longues (plusieurs heures). La première automatisation pour chiffrer et déchiffrer des messages provient d'Enigma [Ber73]. Cette machine électromécanique crée des substitutions d'une lettre vers une autre. Elle a été utilisée pendant la Seconde Guerre mondiale.

La première évolution majeure intervient avec les réseaux de communication du XIX^{e} siècle. Ces réseaux sont par exemple : le réseau ferroviaire, les canaux fluviaux, le télégraphe optique et électrique. Cela s'accompagne d'une professionnalisation de la cryptologie, et du développement de la mécanisation.

La théorie de l'information introduite par Claude Shannon permet de démontrer que le *one time pad*, ou *masque jetable* est un système inconditionnellement sûr [Sha49]. Cette preuve est valide sous les conditions suivantes :

- La clef doit être de même longueur que le message.

- La clef doit être aléatoire.
- La clef est à usage unique.

Le contenu d'une communication reste l'information à protéger, mais l'importance est aussi donnée à la capacité à échanger des informations de manière fiable et non disséminée sur un réseau de communications.

La cryptologie passe maintenant par des moyens de calcul automatiques (ordinateurs, microprocesseurs). Par ailleurs, elle est de moins en moins une activité confidentielle et devient académique.

1.1.3 Âge 3 : le réseau mondial

La création et le développement de l'internet permettent la mise en réseau des ordinateurs à grande échelle. Elle est liée à la mondialisation des échanges, à un changement d'échelle et à la généralisation des réseaux de communications. Par exemple, des acteurs économiques doivent traiter d'affaires, de transactions, sans se connaître ni avoir eu de liaisons préalables.

Il est alors question d'échanger des clefs à grande échelle, les méthodes permettant cela reposent sur des problèmes mathématiques difficiles, comme la factorisation des entiers, qui est en pratique impossible lorsque le nombre à factoriser devient trop grand [KAF+10]. On ne parle plus que de sécurité calculatoire. Une théorie calculatoire de la sécurité se développe, certaines branches des mathématiques comme la théorie des nombres ou la géométrie algébrique sont utilisées. Il s'agit d'enrayer la course illimitée entre le codeur et le casseur de code pour, dès la conception d'un nouveau procédé, l'assortir d'arguments attestant de sa sécurité qui assurera la confiance de ses utilisateurs.

Aujourd'hui, nous entrons dans une nouvelle ère. Il faut en effet prendre en considération que le nombre d'objets communicants est en train d'exploser. La gestion de la communication de ces objets impose aux cryptologues de revoir ce qui était vu pour acquis. Par exemple : les systèmes nomades à très faibles ressources doivent embarquer des systèmes de chiffrement très économes.

1.2 Cryptologie

Le mot cryptologie provient du grec kryptos (caché) et logos (science), il signifie littéralement science du secret et désigne l'ensemble des techniques permettant de rendre indéchiffrable un message à l'aide d'un code secret pour une autre personne que son émetteur ou son destinataire. Cette définition se décline en deux notions :

- La cryptographie : qui est la science des écritures secrètes et des documents chiffrés.
- La cryptanalyse : qui désigne les techniques mises en œuvre pour tenter de déchiffrer un message chiffré dont on ne connaît pas la clef de chiffrement.

Sécuriser un système d'information nécessite la prise en compte de ces deux concepts fondamentaux afin de répondre aux besoins de :

- Confidentialité.
- Intégrité.
- Authenticité.

1.2.1 Cryptographie

Peu importe l'application pour laquelle il est développé, l'algorithme cryptographique doit respecter un des principes de Kerckhoffs (XIX^{e} siècle, [Ker83]) :

« La sécurité d'un chiffré doit dépendre seulement de la clef de chiffrement et non pas de la méthode de chiffrement. »

Dans la suite, nous allons aborder les méthodes de cryptographie en deux segments : la cryptographie symétrique et asymétrique.

Cryptographie symétrique

La cryptographie symétrique se nomme aussi cryptographie à clef secrète. Deux entités qui souhaitent communiquer de manière confidentielle chiffrent leurs messages avec un secret commun. Seule la connaissance de ce secret permet de déchiffrer les chiffrés.

Chiffrements par blocs Le besoin de protéger les communications et les données personnelles mémorisées sur les ordinateurs pousse en 1973 le NBS (National Bureau of Standards) à publier un appel à contributions pour définir un algorithme de chiffrement standard, qui serait utilisable par les entreprises qui ont besoin de protéger les fichiers et les communications sensibles. Cet appel d'offres a conduit à la standardisation en 1976 d'un algorithme de chiffrement pour l'usage civil, le Data Encryption Standard (DES), et d'une publication en janvier 1977 [S+77]. Cette première publication à grande échelle d'un algorithme de chiffrement marque l'entrée de la cryptologie dans le domaine public et finalement la première réalisation effective du principe de Kerckhoffs.

Le DES est un chiffrement par blocs, les données à chiffrer sont des suites de bits regroupés en blocs de 64 bits. Cet algorithme est devenu obsolète en raison du nombre de possibilités que peut prendre la clef utilisée. La puissance de calcul des ordinateurs modernes permet d'énumérer toutes les possibilités, cette méthode s'appelle attaque par force brute. L'Advanced Encryption Standard (AES) [RD01] a été standardisé afin de le remplacer [NBB+01]. D'autres algorithmes de chiffrements par blocs ont été proposés : Blowfish, Serpent ou encore Twofish. Pour répondre à des besoins de ressources très restreintes, la cryptographie dite « légère » est aussi en cours d'études[1].

Chiffrements par flots Contrairement aux chiffrements par blocs, les algorithmes de chiffrement par flots arrivent à traiter les données de longueur quelconque et n'ont pas besoin de les découper. Ils répondent souvent à des besoins de traitement des données à la volée. Parmi les exemples les plus courants, on note l'algorithme A5/1, il est utilisé dans les téléphones mobiles de type GSM pour chiffrer la communication par radio entre le mobile et l'antenne-relais la plus proche [BGW99]. RC4 est lui utilisé

[1] Le site https://www.cryptolux.org/index.php/Lightweight_Cryptography référence les algorithmes de cryptographie légère (chiffrements par blocs, par flots, fonctions de hachage).

notamment par le protocole WEP du Wi-Fi [HHHS00]. Py est une alternative plus sûre est plus performante que RC4. Enfin, notons que le protocole Bluetooth utilise le chiffrement par flots E0 [Vai00].

Les algorithmes de cryptographie symétriques sont utilisés pour des applications nécessitant des performances. L'inconvénient de l'utilisation de la cryptographie à clef secrète est que chaque paire d'entités qui veut communiquer de manière sécurisée se partage une clef. Ainsi, pour un réseau à n utilisateurs, on compte n^2 clefs au total. Le nombre et la transmission des clefs sont des enjeux capitaux.

Distribution des clefs Un moyen pour distribuer des clefs symétriques est le protocole Kerberos [NT94]. Il crée des clefs de session qu'il transmet à deux participants, qui vont pouvoir communiquer de manière sécurisée. Cependant un des inconvénients de Kerberos est que toutes les machines du réseau doivent être synchronisées, à cause de l'utilisation de tickets. Ces tickets, qui ont une validité de 8 heures environ, laissent la possibilité d'une attaque par répétition [BM90]. D'autre part, la machine serveur Kerberos doit être parfaitement sûre, et ne doit jamais tomber, faute de quoi il n'y aura plus aucun accès aux services et donc aucune distribution.

Cryptographie asymétrique

La notion de cryptographie asymétrique – aussi désignée : cryptographie à clefs publiques (Public Key Cryptography (PKC)) – est apparue au milieu des années 70 lorsque Diffie et Hellman [DH76] ont publié le premier article sur la cryptographie à clef publique.

Définition de la cryptographie asymétrique L'idée de la cryptographie asymétrique est l'utilisation d'une *fonction à sens unique avec trappe*. Une telle fonction est facilement calculable, mais le calcul de son inverse n'est pas réalisable en temps raisonnable sauf pour celui qui possède la clef (privée). Dans la cryptographie à clefs publiques, une entité $\mathcal{A}$ a deux clefs :

- sa clef privée, qu'elle garde secrète et qui sert à déchiffrer les chiffrés qui lui sont destinés,
- sa clef publique, celle-ci est connue de tous et permet de chiffrer des messages, que seul $\mathcal{A}$ pourra déchiffrer.

Les algorithmes de cryptographie asymétrique sont beaucoup plus lents que ceux de la cryptographie symétrique (facteur 100, 1000 ou plus encore). À cause de cette lenteur, la cryptographie à clef publique est utilisée pour d'autres applications que le chiffrement de données brutes, à savoir, l'échange de clefs secrètes [DH76], la signature [RSA78] et l'authentification. Lorsque les données impliquées ne sont pas trop grandes, elles peuvent être traitées par des algorithmes de cryptographie à clef publique.

La propriété des fonctions à être : *à sens unique avec trappe* se base sur des problèmes mathématiques difficiles à résoudre. Pour le cryptosystème de Rivest–Shamir–Adleman (RSA), il s'agit de la factorisation de grands entiers (entiers dont la représentation binaire dépasse 2048 bits). L'autre problème couramment utilisé est celui du logarithme discret – ou *Discrete logarithm* – (DLOG) dont l'énoncé est le suivant : soient $\mathbb{G}$ un groupe cyclique, g un générateur de $\mathbb{G}$ et h un élément de $\mathbb{G}$, comment

trouver $x \in \mathbb{N}$ tel que $g^x = h$. Ce problème est notamment à la base de l'échange de clefs Diffie–Hellman, du chiffrement ElGamal [ElG85] et donc de la cryptographie sur les courbes elliptiques – ou *Elliptic Curve Cryptography* – (ECC) [Mil85, Kob87].

Les couplages mathématiques permettent de construire une *fonction à sens unique avec trappe*. Cependant, les couplages ont d'abord été utilisés en cryptanalyse pour ramener le problème du logarithme discret depuis les courbes elliptiques à celui des corps finis, pour lequel on connaît des algorithmes sous-exponentiels [MOV93, FR94]. L'utilisation des couplages en cryptologie est apparue au début des années 2000. La première application proposée est un échange de clefs entre trois utilisateurs en une seule étape [Jou00]. Les couplages ont aussi permis de mettre en œuvre un protocole de chiffrement basé sur l'identité [BF01]. Ce dernier donne une alternative aux architectures de gestion de clefs couramment utilisées, qui souffrent de difficultés face au déploiement pour le grand public.

Distribution des clefs Le principe d'une Public Key Infrastructure (PKI) est de mettre en œuvre la cryptographie asymétrique. Il y a cependant un problème majeur : comment vérifier que l'identité de la personne avec qui l'on communique est bien celle qu'elle prétend être ? C'est-à-dire, qu'il faut vérifier que la personne est associée à la clef publique qu'elle détient. Dans une PKI, les clefs publiques des utilisateurs sont certifiées par un tiers de confiance. Elles peuvent ensuite être mises à disposition des utilisateurs à l'aide d'annuaires. Des utilisateurs non conformes à la politique de sécurité du tiers de confiance peuvent être mis sur liste noire, cette liste est une Liste de révocation des certificats – ou *Certificate Revocation List* – (CRL).

Les notions de confiance et de certification peuvent être gérées de deux manières distinctes : les architectures hiérarchiques où la confiance peut être transmise d'une autorité vers des autorités filles (PKIX [HFPS99], *PKI for X.509 certificates*) ; ou bien non hiérarchiques telles qu'avec Pretty Good Privacy (PGP). La confiance n'est plus transmise par des autorités. Il s'agit d'un modèle pair-à-pair où une entité peut signer le certificat d'une autre personne si elle le sait valide. Un réseau de confiance est ainsi créé.

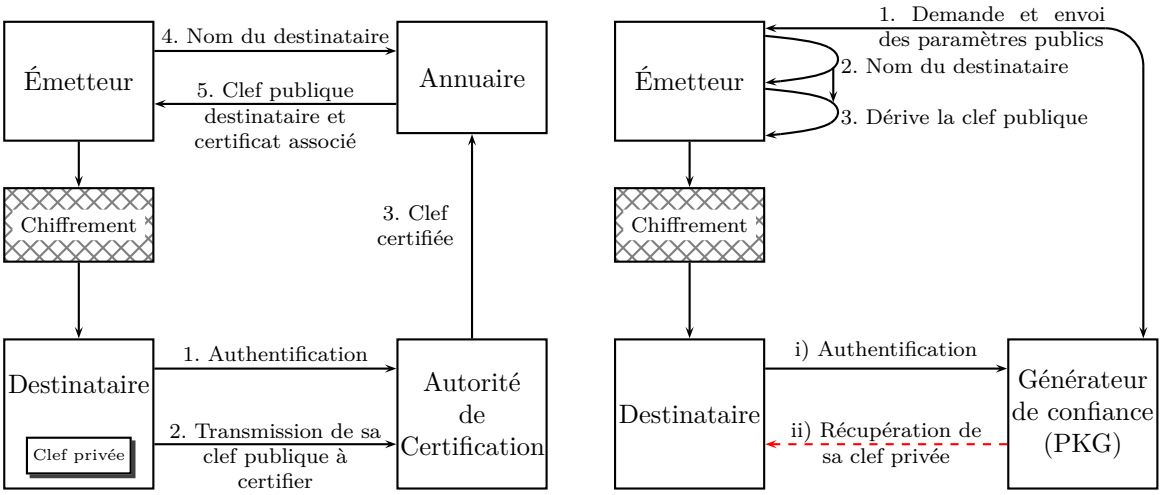
L'étude des PKI est un domaine à part entière et n'est pas l'objectif premier de cette thèse. Pour une esquisse complète, nous renvoyons au livre de Dumas *et al.* [DLR15], qui contient toutes les informations nécessaires à l'appréhension des PKI et des communications sécurisées.

Le quatrième âge de la cryptographie peut être identifié comme l'âge de l'ubiquité. Le développement de l'internet va au-delà des ordinateurs, il s'étend aux objets. La cryptographie doit être adaptée à ces nouveaux besoins qui en découlent, la quantité d'objets est une problématique phare.

Les protocoles de chiffrement basés sur l'identité sont intéressants par le fait que l'émetteur d'un message chiffré n'a besoin d'aucune information sur le destinataire mis à part son identité. Dans ce modèle, le message chiffré pourra être lu par son destinataire seulement lorsqu'il aura fait le nécessaire auprès de l'autorité de confiance (le Private Key Generator (PKG)). En d'autres termes, le destinataire doit se procurer les droits (sa clef privée) pour déchiffrer ses messages reçus. Ainsi, un nouveau venu dans un ensemble d'entités qui communiquent ensemble s'enrôle auprès de l'autorité et est immédiatement apte à déchiffrer ses messages, et bien sûr à en envoyer aux autres, car il lui suffit de leurs identités. Cette fluidité est une propriété qui est de plus en plus utile. En effet, on retrouve ce schéma dans un réseau d'objets connectés, l'ajout d'un

nouvel objet ne nécessitera aucune manipulation de la part des objets déjà en place, qui sont des émetteurs et destinataires potentiels.

La figure 1.1 montre un exemple d'échange dans une PKI et dans une architecture Identity-Based Encription (IBE). Pour la PKI, figure 1.1a, l'ordre des échanges est immuable. L'émetteur peut envoyer un message chiffré au destinataire souhaité dès qu'il reçoit et vérifie sa clef publique. Dans le schéma IBE, figure 1.1b, nous distinguons deux styles de numérotations. L'émetteur utilise directement l'identité du destinataire pour dériver la clef publique, et il peut alors lui envoyer un message chiffré. Le destinataire peut faire sa demande de clef privée à tout moment du processus, avant ou après avoir reçu des messages chiffrés.



(a) Exemple d'échange dans une PKI (b) Exemple d'échange dans l'IBE

Figure 1.1 – Exemples d'échanges PKI et IBE

Nous notons les principales différences entre une PKI hiérarchique et l'IBE dans la table 1.1.

Table 1.1 – Différences notables PKI vs IBE

	PKI	**IBE**
Clefs publiques	Aléatoire	Identité
Génération des clefs	**Peut** être fait par l'utilisateur	**Doit** être fait par l'autorité
Révocation	Listes (CRL) Validité (donnée du certificat)	Renouvellement
Propriété des clefs privées	L'utilisateur	L'autorité

1.2.2 Cryptanalyse

Le cryptanalyste passe d'abord par la définition d'un attaquant qui essaye de retrouver le message et/ou le code secret. Ce secret, appelé clef, est un entier naturel[2] $\boldsymbol{s}$ appartenant à un ensemble $\mathcal{K}$. L'idée naïve d'un attaquant est d'essayer tous les éléments de

[2]Le secret peut aussi être un point sur une courbe elliptique, dans ce cas $\boldsymbol{s}$ est un couple d'entier.

$\mathcal{K}$ pour retrouver $\boldsymbol{s}$, cette méthode s'appelle attaque par force brute. Cette méthode naïve sur les algorithmes de chiffrement est de nos jours impossible, en raison de la taille de $\mathcal{K}$ qui est trop grande pour qu'un ordinateur soit capable d'énumérer tous les cas dans un temps humainement raisonnable. Par exemple, le DES a été considéré comme inutilisable lorsque le DES cracker a mis moins d'une journée à retrouver la clef[3].

La cryptanalyse permet de trouver une méthode plus efficace pour retrouver la clef $\boldsymbol{s}$. Dans cette thèse, nous distinguerons deux types de cryptanalyses :

- La cryptanalyse mathématique, où l'attaquant utilise des outils purement mathématiques pour trouver des failles dans les algorithmes de chiffrement. L'exploitation de ces failles rendra plus facile une attaque.
- La cryptanalyse physique, où l'attaquant exploite les phénomènes physiques de l'implémentation d'un algorithme sur un dispositif embarqué.

Conséquence de la cryptanalyse mathématique

La cryptanalyse mathématique permet de rendre plus efficace l'attaque par force brute, soit en rendant les algorithmes d'attaques plus efficaces, soit en cherchant des failles dans les cryptosystèmes. Pour se prémunir de ces attaques, le cryptographe doit s'assurer que la taille de $\mathcal{K}$ est suffisamment grande. La taille de $\mathcal{K}$ peut varier selon le niveau de sécurité souhaité. Les recommandations sont en constante évolution. Par exemple, souhaiter un niveau de sécurité 128 bits signifie qu'il faut une énumération de 2^{128} hypothèses de clef pour faire une recherche exhaustive. L'institut national des normes et de la technologie – ou *National Institute of Standards and Technology* – (NIST), recommande en 2017 les tailles précisées dans la table 1.2 (source : https://www.keylength.com/fr/4/).

Table 1.2 – Recommandation sur les tailles des paramètres et niveaux de sécurité équivalents

Résistance	**Cryptosystème**	
	RSA	**ECC**
112 bits	2048	224
128 bits	3072	256
256 bits	15360	512

Il est encore difficile de faire une colonne concernant les algorithmes de couplages. Leur sécurité dépend de l'efficacité des attaques mathématiques contre le DLOG. Les recherches sur ces attaques sont très actives récemment, et les tailles de paramètres associées sont recalculées en conséquence [MSS16, BD17].

Cryptanalyse physique

Dans cette thèse, nous allons nous pencher sur les attaques physiques. Comme les implémentations des algorithmes de cryptographie sont de plus en plus efficaces (temps, mémoire), il est devenu facile de les implémenter sur des dispositifs embarqués.

[3]Plus d'informations sur le DES Cracker : https://w2.eff.org/Privacy/Crypto/Crypto_misc/DESCracker/HTML/19980716_eff_des_faq.html

Le système embarqué de référence est la carte à puce, ce dispositif à faibles ressources embarque des données, des codes, des clefs secrètes et aussi des cryptosystèmes. Mais, le fait qu'ils soient justement embarqués les rend vulnérables face à un attaquant qui prend possession du dispositif, il se retrouve en environnement hostile. L'attaquant a donc accès au dispositif et est en mesure de mener des attaques physiques. Il va pouvoir observer son comportement pendant qu'il lui demande d'exécuter des commandes choisies.

L'observation se fait via un canal auxiliaire, il peut être de différentes natures, le temps de calcul [KJJ99], la consommation de courant [KJJ99], les émanations électromagnétiques [Qui00] ou les ondes acoustiques [ST04].

Ces mesures acquises sont liées aux données manipulées et donc aux clefs secrètes impliquées dans les algorithmes de chiffrement. L'attaquant procède donc à une analyse statistique qui permet d'exploiter ce lien pour retrouver le secret. Les attaques sans profilages courantes sont : la Simple Power Analysis (SPA) [KJJ99], la Differential Power Analysis (DPA) [KJJ99] et la Correlation Power Analysis (CPA) [BCO04]. Les attaques par profilage sont introduites par Chari *et al.* [CRR02]. Un modèle stochastique est appréhendé par Schindler *et al.* [SLP05]. Enfin, on note que la frontière entre les attaques avec et sans phase de profilage est mince, par exemple les articles [DPRS11] et [LPR13] ont repris l'attaque stochastique et améliorent l'efficacité en supprimant les profilages.

Pour protéger les implémentations des algorithmes cryptographiques, la communauté étudie activement les contre-mesures. Les protections logicielles sont souvent propres aux algorithmes, en cryptographie asymétrique, l'AES peut être protégé par des masques [AG01], du brassage [CCD00, Man04] (*shuffling* en anglais). Le RSA peut être protégé en utilisant une exponentiation dite « régulière » (échelle de Montgomery [Mon87], double et ajoute toujours [Cor99]) en plus d'une randomisation de l'exposant [Koc96]. La cryptographie sur courbe elliptique utilise au moins les trois contre-mesures de Coron [Cor99].

1.3 Positionnement de la thèse

Nous avons introduit les enjeux autour de la sécurité des systèmes d'information. Les problématiques de déploiements sécurisés font apparaître l'intérêt d'étudier les protocoles cryptographiques à base de couplage. La sécurité de leur implémentation face aux attaques par canaux auxiliaires est l'objet de cette thèse.

Les études récentes ont montré l'existence de failles dans les algorithmes de couplages envers des attaques par canaux auxiliaires. Nous allons revenir en détail sur ces failles, car elles représentent un danger que les développeurs doivent prendre en compte dans la conception de protocoles à base de couplages. La littérature a introduit quelques attaques sur les algorithmes de couplages. Au début de cette thèse, il n'y avait cependant aucune réalisation pratique. La transition vers une attaque en environnement réel n'est pas toujours évidente, nous proposons donc la réalisation pratique de telles attaques. La faisabilité de ces attaques en pratique est réalisée en deux phases. La première est une phase de caractérisation, nous cherchons à créer des modèles théoriques adaptés à la cible. Les études autour de la sécurité de l'implémentation des couplages n'étant pas aussi étendues que pour les algorithmes de chiffrement par blocs, RSA et ECC, il est utile de poser les premières briques de la caractérisation de la cible. Pour

la deuxième phase, les schémas théoriques devront être mis en pratique. Pour cela, il faudra construire un environnement d'étude semblable à un environnement réel.

De plus, les attaques récentes de la littérature exploitent toutes le même schéma de CPA verticale. Un des enjeux de cette thèse est d'étudier si de nouvelles failles sont identifiables. On pense ici aux liens possibles à faire avec les attaques récentes sur la cryptographie sur courbes elliptiques. Des attaques optimisées, dites « horizontales », ont vu le jour [Wal01, CFG+10]. Nous étudierons ces attaques dans le contexte des couplages afin de signaler si elles représentent un danger.

Dans cette thèse, nous nous concentrons sur les attaques physiques par observations. Les attaques physiques par injection de fautes ont largement été étudiées et mises en pratique [LFG13, LPEM+14, EMFGL15].

Par ailleurs, il est intéressant de se pencher sur les contre-mesures de la littérature pour protéger les implémentations face aux attaques par canaux auxiliaires. Il n'y a aucune preuve de sécurité quant à leurs résistances. L'une d'entre elles, introduite originellement pour la cryptographie sur courbes elliptiques, représente un faible surcoût. De plus, la randomisation obtenue est admise comme efficace. Cette contre-mesure prometteuse est à l'étude dans cette thèse. Le but est d'analyser sa résistance face aux attaques connues puis face à de nouvelles menaces.

Dans un dernier temps, nous chercherons à identifier les contre-mesures qui peuvent être combinées afin de protéger des attaques que nous répertorions, en maintenant un faible surcoût.

1.4 Plan de la thèse

Cette thèse se concentre sur la sécurité des implémentations des algorithmes de couplages face aux attaques par canaux auxiliaires. Dans un premier temps, nous introduirons des outils mathématiques nécessaires à la définition des couplages. Deux axes seront développés, les rappels mathématiques et l'implémentation des algorithmes. Le chapitre 2 décrit ces éléments, en partant du plus bas niveau applicatif, avec l'introduction des corps finis et de leurs arithmétiques, au plus haut niveau. Il contient des éléments sur les couplages et leurs implémentations, en passant par les courbes elliptiques. Ce chapitre est aussi destiné à rappeler les principaux protocoles que permettent les couplages.

Le chapitre 3 retrace l'état de l'art des attaques par canaux auxiliaires. Le principe général de ces attaques est introduit, puis nous décrivons plus particulièrement les attaques visant les algorithmes de couplages. Par la suite, nous détaillons les contre-mesures existantes.

Nous proposons une caractérisation de la cible dans le chapitre 4. Ce chapitre contient aussi les résultats théoriques et pratiques des attaques sur une implémentation non protégée d'un couplage.

Dans le chapitre 5 nous nous intéressons aux attaques contre une implémentation des couplages protégée par la randomisation des coordonnées jacobiennes.

Enfin, dans le chapitre 6 nous proposons une synthèse des travaux de cette thèse ainsi que les perspectives qui en découlent.

1.5 Communications

Les travaux de cette thèse ont fait l'objet des publications et communications suivantes :

- [JFG17] Damien Jauvart, Jacques J.A. Fournier et Louis Goubin. – First Practical Side-Channel Attack to Defeat Point Randomization in Secure Implementations of Pairing-Based Cryptography. – **International Conference on Security and Cryptography**, Secrypt 2017.
- [EMGG+17] Nadia El Mrabet, Louis Goubin, Sylvain Guilley, Jacques J.A. Fournier, Damien Jauvart, Martin Moreau, Pablo Rauzy et Franck Rondepierre. – Physical attacks. Dans **Guide to Pairing-Based Cryptography**, Nadia El Mrabet et Marc Joye éd. – Chapman and Hall/CRC, 2016.
- [JFEMG16] Damien Jauvart, Jacques J.A. Fournier, Nadia El Mrabet et Louis Goubin. – Improving Side-Channel Attacks against Pairing-Based Cryptography. – **Risks and Security of Internet and Systems**, Frédéric Cuppens, Nora Cuppens, Jean-Louis Lanet et Axel Legay éd. – Springer International Publishing, 2016.

Chapitre 2

Cryptographie à base de couplage

« Un mur ne se fait pas avec une seule pierre »

Proverbe Russe

Sommaire

Les couplages sont des objets mathématiques élaborés, ils sont basés sur des théories de géométrie algébrique. Nous allons aborder les couplages avec une approche pratique. Ce chapitre permet de poser les bases nécessaires à la compréhension des protocoles cryptographiques qui vont être ciblés.

2.1 Prérequis mathématiques

Il est d'abord essentiel de faire quelques rappels mathématiques. Les couplages sont des objets mathématiques basés sur des variétés abéliennes. Cependant, nous allons pouvoir faire en partie abstraction des rudiments de géométrie algébrique, car la cryptographie basée sur les couplages sur laquelle nous nous concentrons utilise des variétés abéliennes bien précises, celles de dimension 1 : les courbes elliptiques.

Nous commencerons par introduire de façon informelle la théorie des corps finis, puis nous mentionnerons des corps qui s'accordent avec l'implémentation efficace des courbes elliptiques. Nous allons d'abord aborder l'aspect algorithmique de l'arithmétique, il sera question de l'implémentation des opérations élémentaires. Cette section est inspirée de [LN97].

2.1.1 Introduction aux corps finis

De façon informelle, un anneau (commutatif unitaire et intègre) est un ensemble où l'on peut additionner, soustraire et multiplier comme on a l'habitude de le faire pour les entiers. Cela se traduit par le fait qu'il est possible de définir une addition $a+b$ et une multiplication ab à l'intérieur de l'anneau avec les propriétés suivantes :

- Les opérations sont commutatives ($a+b=b+a$ et $ab=ba$) et associatives ($(a+b)+c=a+(b+c)$ et $(ab)c=a(bc)$), de sorte que l'ont peut manipuler des sommes $\sum_i a_i$ et des produits $\prod_i a_i$ finis sans tenir compte de l'ordre des termes.
- Le fait d'ajouter 0 ou multiplier par 1 ne change rien. Multiplier par 0 donne 0. C'est la notion d'éléments neutres, respectivement pour l'addition et la multiplication.
- L'addition et la multiplication sont reliées par la propriété de distributivité $a(b+c)=ab+ac$, ce qui permet de développer des produits de sommes (par exemple, la formule du binôme de Newton : $(a+b)^n=\sum_i \binom{i}{n} a^i b^{n-i}$).
- La possibilité de soustraire ($a+x=b$ donne $x=b-a$), mais pas diviser (par exemple, il n'y a pas d'entier x tel que $2x=1$). Par contre, on peut toujours simplifier ($a+c=b+c$ ou $ac=bc$ donnent $a=b$). Pour la multiplication, cette propriété provient de l'intégrité de l'anneau.

Deux modèles d'anneaux sont constamment utilisés : les *nombres entiers* $\mathbb{Z}$ et les *polynômes* à coefficients eux-mêmes dans un anneau. Pour passer d'un anneau (commutatif unitaire et intègre) à un corps (commutatif [1]), il faut de plus être capable de diviser (par tout élément non nul). La division de a par $b\neq 0$ donne l'élément x tel que $bx=a$. Il est noté $x=\frac{a}{b}$. En particulier, l'inverse de a est $a^{-1}=\frac{1}{a}$. Notez que l'écriture $x=\frac{a}{b}$ est *a priori* ambiguë, car elle peut signifier ab^{-1} ou $b^{-1}a$, mais comme la multiplication est commutative cette ambiguïté disparaît.

[1] Le cas des corps non commutatifs ne sera pas traité dans cette thèse.

Opérations de corps

Un corps $\mathbb{K}$ est muni de deux opérations, une addition et une multiplication. La soustraction est définie comme une addition : pour $a, b \in \mathbb{K}, a - b = a + (-b)$ où $-b$ est l'unique élément de $\mathbb{K}$ tel que $b + (-b) = 0$. De manière similaire, la division dans un corps est définie à l'aide de la multiplication : $a, b \in \mathbb{K}$ avec $b \neq 0$, $\frac{a}{b} = ab^{-1}$ où b^{-1} est l'unique élément de $\mathbb{K}$ tel que $bb^{-1} = 1$ (b^{-1} est appelé inverse de b).

Existence et unicité

Si l'ensemble $\mathbb{K}$ est fini, alors le corps $\mathbb{K}$ est appelé corps fini. L'ordre d'un corps est défini par :

Définition 1. *Le cardinal d'un corps $\mathbb{K}$ est le nombre d'éléments de l'ensemble $\mathbb{K}$.*

Le théorème suivant est fondamental dans la théorie des corps :

Théorème 1 ([PCD96], chapitre III.9.)**.** *Soit un nombre premier p et d un entier positif, il existe un corps à $q = p^d$ éléments. Ce corps est unique, à isomorphisme près, et est noté $\mathbb{F}_q$.*

Le nombre p est appelé caractéristique de $\mathbb{F}_q$, on utilise la notation $\mathrm{Char}(\mathbb{F}_q) = p$. Lorsque $d = 1$, $\mathbb{F}_q$ est dit corps premier. Et si $d \geq 2$, alors $\mathbb{F}_q$ est appelé extension de corps. Le théorème 1 montre l'existence et l'unicité d'un corps $\mathbb{F}_q$ avec $q = p^d$; informellement, cela induit que deux corps finis qui ont le même cardinal q ont la même structure, mais n'ont pas la même représentation de leurs éléments.

Corps premier

Soit p un nombre premier, l'ensemble des entiers modulo p (c'est-à-dire $\{0, 1, \ldots, p-1\}$) muni de l'addition et de la multiplication modulo p est **le** corps fini d'ordre p, il est donc noté $\mathbb{F}_p$.

Corps binaires

Avant de décrire les corps qui vont être utilisés plus loin pour les courbes elliptiques, nous nous arrêtons quelques instants sur les corps binaires, dont la construction et la structure sont finalement très proches des extensions de corps plus générales. Les courbes elliptiques peuvent aussi être définies sur les corps binaires, mais dans le cas des couplages, leur utilisation n'est plus sûre [JOP14].

Un corps binaire est un corps fini à 2^d éléments, il est donc de caractéristique 2. Une façon de construire un tel corps $\mathbb{F}_{2^d}$ est d'utiliser la représentation polynomiale. Ainsi, les éléments de $\mathbb{F}_{2^d}$ sont les polynômes binaires de degré au plus $d - 1$:

$$\mathbb{F}_{2^d} = \left\{ a_0 + a_1 z + a_2 z^2 + \cdots + a_{d-2} z^{d-2} + a_{d-1} z^{d-1} | a_i \in \{0, 1\} \right\}. \tag{2.1}$$

Un polynôme binaire irréductible $f(z)$ de degré d est choisi. L'irréductibilité de $f(z)$ signifie que f ne peut pas être factorisé comme un produit de polynômes de degrés chacun strictement inférieur à d. Ainsi, l'addition dans $\mathbb{F}_{2^m}$ est l'addition usuelle sur les polynômes, les opérations sur les coefficients s'effectuant modulo 2. La multiplication est, quant à elle, effectuée modulo la réduction polynomiale $f(z)$. Pour un polynôme

binaire $a(z)$, la réduction modulo $f(z)$, notée $a(z) \mod f(z)$ dénote l'unique polynôme $r(z)$ de degré inférieur à d correspondant au reste de la division de $a(z)$ par $f(z)$; cette opération est la réduction modulo $f(z)$.

Extension de corps

Les corps binaires sont un cas particulier des extensions de corps avec p premier égal à 2. Soit maintenant p un nombre premier et $d \geq 2$. La notation $\mathbb{F}_p[z]$ dénote l'ensemble des polynômes en z à coefficients dans $\mathbb{F}_p$. Soit $f(z) \in \mathbb{F}_p[z]$ un polynôme irréductible[2] de degré d, $\mathbb{F}_{p^d}$ est isomorphe à $\mathbb{F}_p[z]/(f(z))$. Une représentation des éléments de $\mathbb{F}_{p^d}$ sont les polynômes de $\mathbb{F}_p[z]$ de degré au plus $d-1$:

$$\mathbb{F}_{p^d} = \left\{ a_0 + a_1 z + a_2 z^2 + \cdots + a_{d-2} z^{d-2} + a_{d-1} z^{d-1} | a_i \in \mathbb{F}_p \right\}. \tag{2.2}$$

De même qu'avec les corps binaires, les additions s'effectuent usuellement, l'addition sur les coefficients est faite dans $\mathbb{F}_p$. La multiplication est effectuée modulo $f(z)$. De façon générale, on peut considérer des extensions algébriques pour n'importe quel corps $\mathbb{K}$, on notera par $\mathfrak{L}$ une extension algébrique d'un corps $\mathbb{K}$.

Bases d'un corps fini

Le corps fini $\mathbb{F}_{p^d}$ peut être vu comme un espace vectoriel de dimension d sur $\mathbb{F}_p$. Les vecteurs sont les éléments de $\mathbb{F}_{p^d}$, tandis que les scalaires sont les éléments de $\mathbb{F}_p$. L'addition de vecteurs est l'addition de $\mathbb{F}_{p^d}$. La multiplication par un scalaire correspond à la multiplication d'un $\mathbb{F}_p-$élément (scalaire) avec un $\mathbb{F}_{p^d}-$élément (vecteur) dans $\mathbb{F}_{p^d}$.

L'un des aspects pratiques des espaces vectoriels est l'utilisation des bases. Si $\mathcal{B} = \{b_1, b_2, \ldots, b_d\}$ est une base, alors $a \in \mathbb{F}_{p^d}$ est représenté de manière unique par un $d-$uplet $(a_1, a_2, \ldots, a_d)$ d'éléments de $\mathbb{F}_p$ où $a = a_1 b_1 + a_2 b_2 + \cdots + a_d b_d$. Par exemple, dans la représentation décrite au paragraphe *Sous-corps d'un corps fini*, $\mathbb{F}_{p^d}$ est un $\mathbb{F}_p-$espace vectoriel de dimension d et $\left\{ z^{d-1}, z^{d-2}, \ldots, z^2, z, 1 \right\}$ est une base pour $\mathbb{F}_{p^d}$ sur $\mathbb{F}_p$.

Sous-corps d'un corps fini

Un sous-ensemble $\mathfrak{K}$ du corps $\mathbb{K}$ est un sous-corps de $\mathbb{K}$ si $\mathfrak{K}$ est lui-même un corps relativement aux opérations de $\mathbb{K}$. De cette façon, $\mathbb{K}$ est dit extension de corps de $\mathfrak{K}$.

Les sous-corps d'un corps fini sont faciles à décrire. Un corps fini $\mathbb{F}_{p^d}$ a précisément un sous-corps d'ordre p^l pour chaque diviseur (positif) l de d. Les éléments d'un tel sous-corps $\mathbb{F}_{p^l}$ sont les éléments a de $\mathbb{F}_{p^d}$ tels que $a^{p^l} = a$.

Groupe multiplicatif d'un corps fini

Les éléments non nuls d'un corps fini $\mathbb{F}_q$ forment un ensemble noté $\mathbb{F}_q^\star$.

Un groupe est un ensemble $\mathbb{G}$ muni d'une opération $\star$ qui satisfait les quatre axiomes :

- La loi $\star$ est interne : pour tous a et b dans $\mathbb{G}$, le résultat $a \star b$ est aussi dans $\mathbb{G}$.

[2] La preuve de l'existence d'un tel polynôme utilise la fonction de Möbius et ne sera pas détaillée ici (voir par exemple [FG94]).

- L'associativité : pour tous a, b et c de $\mathbb{G}$, l'égalité $(a \star b) \star c = a \star (b \star c)$ est vraie.
- L'élément neutre : il existe un élément e de $\mathbb{G}$ tel que, pour tout a dans $\mathbb{G}$, $e \star a = a \star e = a$.
- La symétrie : pour tout élément a de $\mathbb{G}$, il existe b dans $\mathbb{G}$ tel que $a \star b = b \star a = e$, où e est l'élément neutre. b est appelé symétrique de a.

Le théorème 2 s'applique pour tous les corps finis $\mathbb{F}_q$.

Théorème 2 ([PCD96] page 74)**.** *Muni de la multiplication, l'ensemble $\mathbb{F}_q^\star$ est un groupe cyclique. Par conséquent, il existe des éléments $b \in \mathbb{F}_q^\star$ appelés générateurs tels que :*

$$\mathbb{F}_q^\star = \left\{ b^i | 0 \leq i \leq q-2 \right\}. \tag{2.3}$$

Les générateurs d'un groupe $\mathbb{G}$ sont aussi appelés racines primitives de $\mathbb{G}$. Le nombre de racines primitives de $\mathbb{F}_q^\star$ s'exprime facilement avec la fonction φ d'Euler : $\varphi(q-1)$. Pour rappel, la fonction φ d'Euler est ainsi définie :

$$\varphi(N) = \#\left\{ x | 1 \leq x \leq N, \text{pgcd}(x, N) = 1 \right\}. \tag{2.4}$$

$\varphi(N)$ peut se calculer facilement si on dispose de la décomposition en facteurs premiers de N, la formule est la suivante :

$$\text{si} \quad N = \prod_{i=1}^{r} n_i^{\alpha_i} \quad \text{alors} \quad \varphi(N) = N \prod_{i=1}^{r} \left(1 - \frac{1}{n_i}\right). \tag{2.5}$$

L'ordre d'un élément $a \in \mathbb{F}_q^\star$ est le plus petit entier positif $\mathfrak{O}$ tel que $a^{\mathfrak{O}} = 1$. Puisque $\mathbb{F}_q^\star$ est un groupe cyclique, il vient que $\mathfrak{O}$ est un diviseur de $q-1$.

Remarque 1. *Si e divise $q-1$ alors il y a exactement $\varphi(e)$ éléments d'ordre e dans $\mathbb{F}_q^\star$. D'ailleurs, la fonction $f(\alpha) = \alpha^e$ est une bijection* ***si et seulement si*** *$pgcd(e, q-1) = 1$.*

Morphisme de Frobenius

La notion de morphisme de Frobenius sera plus loin utilisée dans la théorie des courbes elliptiques. Soit p un nombre premier et $d \in \mathbb{N}^\star$ et $q = p^d$.

Définition 2 (Morphisme de Frobenius)**.** *Soit A un anneau commutatif unitaire de caractéristique p. L'application*

$$\Psi_A : \left\{ \begin{array}{ccc} A & \to & A \\ x & \mapsto & x^p \end{array} \right. \tag{2.6}$$

est un morphisme d'anneaux unitaires appelé morphisme de Frobenius de A.

Cette définition nécessite de montrer que l'application est bien un morphisme d'anneaux. C'est à dire, que l'on a bien $\Psi_A(a+b) = \Psi_A(a) + \Psi_A(b)$ et $\Psi_A(ab) = \Psi_A(a)\Psi_A(b)$ pour a et b éléments de A, ainsi que $\Psi_A(1) = 1$.

Démonstration. On a $1^p = 1$, et puisque A est commutatif on a $(xy)^p = x^p y^p$ pour tout $x, y \in A$. Ensuite, il faut montrer que $(x+y)^p = x^p + y^p$ pour tous $x, y \in A$, cela se montre grâce à la formule du binôme (cf. page 14) et au fait que p divise $\binom{p}{i} = \frac{p!}{i!(p-i)!}$ pour tout $i \in \{1, \ldots, p-1\}$, la notation $p!$ désigne le factoriel de p ($p! = 1 \times 2 \times 3 \times \cdots \times p$). Puisque $i\binom{p}{i} = p(p-1)\cdots(p-i+1)$, p divise $i\binom{p}{i}$. Comme p est premier et $1 \leq i \leq p-1$ on sait que p est premier avec tous les facteurs de $i!$, le lemme de Gauss, qui dit que si un nombre entier a divise le produit de deux autres nombres entiers b et c, et si a est premier avec b, alors a divise c, permet de conclure. □

2.1.2 Arithmétique des corps finis

L'arithmétique des corps finis est très importante vis-à-vis de l'implémentation efficace des opérations sur courbes elliptiques que nous verrons à la section 2.1.4. En effet, les opérations sur courbes elliptiques sont effectuées en utilisant des opérations arithmétiques dans le corps sous-jacent. Cette section est la conjonction de [CFA+05] et [HMV06].

Addition et soustraction multiprécision

Nous allons maintenant voir l'aspect informatique, il s'agit de voir comment sont réalisées les opérations de corps sur une machine (ordinateur, microcontrôleur, processeur en général). Nous considérons que la machine est dotée de l'arithmétique simple précision, nous allons voir l'arithmétique multiprécision. C'est-à-dire, qu'à partir des opérations élémentaires entre des mots machine nous montrons comment construire les opérations de tailles arbitraires. Nous montrerons ensuite comment ces opérations sont adaptées pour l'arithmétique des corps finis. L'architecture de la machine est sur $\mathfrak{W}$ bits. Couramment, on trouve des architectures sur $\mathfrak{W} = 8, 16, 32$ ou $\mathfrak{W} = 64$ bits. Un entier a (et *a fortiori*, un élément de $\mathbb{F}_p$) de $\mathfrak{n}$ bits sera représenté en base $\overline{\mathfrak{W}} = 2^{\mathfrak{W}}$ de la manière suivante :

$$\begin{aligned} a &= a_0 + 2^{\mathfrak{W}} a_1 + \cdots + a_{\mathfrak{N}-1} 2^{(\mathfrak{N}-1)\mathfrak{W}} \\ &= (a_{\mathfrak{N}-1} a_{\mathfrak{N}-2} \ldots a_1 a_0)_{\overline{\mathfrak{W}}} . \end{aligned} \tag{2.7}$$

où $\mathfrak{N} = \left\lceil \frac{\mathfrak{n}}{\mathfrak{W}} \right\rceil$ est le nombre de mots de a.

Dans la suite, l'affectation « $(\varepsilon, z) \leftarrow w$ » pour un entier w fait référence aux opérations :

$$\begin{aligned} z &\leftarrow w \mod 2^{\mathfrak{W}}, \text{ et} \\ \varepsilon &\leftarrow \begin{cases} 0 \text{ si } w \in \left[0, 2^{\mathfrak{W}}\right[\\ 1 \text{ sinon} \end{cases} . \end{aligned} \tag{2.8}$$

ε est le bit de retenue de l'addition (*carry* en anglais). L'algorithme 2.1 décrit la méthode d'addition en multiprécision.

La soustraction est semblable à l'addition. La retenue est toutefois différente, pour la démarquer, elle est nommée retenue d'emprunt (*borrow* en anglais). L'algorithme 2.2 décrit la méthode de soustraction des entiers en multiprécision.

Opérations élémentaires sur les corps premiers

Cette fois-ci, les opérations seront effectuées sur des entiers a et b de $\mathbb{F}_p$, avec p premier. C'est-à-dire que l'addition et la soustraction sont réduites modulo p, ces opérations

Entrées : Deux entiers de $\mathfrak{N}$ mots $a = (a_{\mathfrak{N}-1} \dots a_0)_{\overline{\mathfrak{W}}}$ et $b = (b_{\mathfrak{N}-1} \dots b_0)_{\overline{\mathfrak{W}}}$.
Sorties : (ε, c) tels que $c = a + b \mod 2^{\mathfrak{W}\times\mathfrak{N}}$ et ε est le bit de retenue.

$(\varepsilon, c_0) \leftarrow a_0 + b_0$;
pour $i = 1$ **à** $\mathfrak{N} - 1$ **faire**
  $(\varepsilon, c_i) \leftarrow a_i + b_i + \varepsilon$;
fin
retourner (ε, c) ;

Algorithme 2.1 : Addition d'entiers multiprécisions positifs [CFA+05]

Entrées : Deux entiers de $\mathfrak{N}$ mots $a = (a_{\mathfrak{N}-1} \dots a_0)_{\overline{\mathfrak{W}}}$ et $b = (b_{\mathfrak{N}-1} \dots b_0)_{\overline{\mathfrak{W}}}$.
Sorties : (ε, c) tels que $c = a - b \mod 2^{\mathfrak{W}\times\mathfrak{N}}$ et ε est le bit de retenue d'emprunt.

$(\varepsilon, c_0) \leftarrow a_0 - b_0$;
pour $i = 1$ **à** $\mathfrak{N} - 1$ **faire**
  $(\varepsilon, c_i) \leftarrow a_i - b_i - \varepsilon$;
fin
retourner (ε, c) ;

Algorithme 2.2 : Soustraction d'entiers multiprécisions positifs [CFA+05]

peuvent être directement adaptées des précédents algorithmes (cf. algorithme 2.1 et 2.2) en ajoutant l'étape de réduction modulo p.

Multiplication sur les corps premiers

La multiplication est une opération très importante, elle est utilisée par la plupart des algorithmes de cryptographie asymétrique. C'est évidemment le cas des algorithmes de couplages. De plus, le nombre de multiplications effectuées dans de tels calculs est considérable, ce qui invite à avoir une multiplication efficace.

La multiplication de a et b dans $\mathbb{F}_p$ (p premier) est faisable en multipliant d'abord a et b dans $\mathbb{Z}$ puis en effectuant la réduction modulo p avec l'algorithme d'Euclide. Montgomery [Mon85] propose de représenter les éléments $\mathbb{Z}/N\mathbb{Z}$ de façon à ce que les réductions modulaires soient « faciles ». D'un point de vue pratique N est premier.

Définition 3. *Soit $\mathfrak{R}$ un entier plus grand que N tel que $\mathfrak{R}$ et N soient premiers entre eux. La représentation de Montgomery de $a \in [\![0, N-1]\!]$ est $[a] = (a\mathfrak{R}) \mod N$. $\mathfrak{R}$ est appelé résidu de Montgomery. La réduction de Montgomery de $a \in [\![0, \mathfrak{R}N-1]\!]$ est* $\text{REDC}(a) = (a\mathfrak{R}^{-1}) \mod N$.

Remarque 2. $[a] = \text{REDC}\left((a\mathfrak{R}^2) \mod N\right)$, *de plus , si* $a \in [\![0, N-1]\!]$ *alors* $\text{REDC}([a]) = a$.

Pour des raisons d'efficacité, $\mathfrak{R}$ est une puissance de $\overline{\mathfrak{W}}$, car une étape de la réduction de Montgomery de a consiste à diviser par $\mathfrak{R}$. Détaillons cela, notons $N' = -N^{-1} \mod \mathfrak{R}$ et $\gamma \in [\![0, N-1]\!]$ l'unique entier tel que $\gamma = aN' \mod \mathfrak{R}$. Ainsi $(a + \gamma N)$ est un multiple de $\mathfrak{R}$. Et, soit maintenant $\lambda = \frac{a+\gamma N}{\mathfrak{R}}$, le fait que N et $\mathfrak{R}$ soient premiers entre eux implique naturellement que $\lambda = (a\mathfrak{R}^{-1}) \mod N$. Et, puisque $0 \leq a < \mathfrak{R}N$ par hypothèse et que $0 \leq \lambda < 2N$ est facilement montrable il vient que λ ou $\lambda - N$ est bien la réduction de Montgomery de a : $\text{REDC}(a)$.

Ce principe de réduction est transposable vers une méthode calculatoire, l'algorithme 2.3 représente cette méthode.

Entrées : Un entier $N = (N_{\mathfrak{N}-1} \dots N_0)_{\overline{\mathfrak{W}}}$ tel que pgcd $\left(N, \overline{\mathfrak{W}}\right) = 1$, $\mathfrak{R} = \overline{\mathfrak{W}}^{\mathfrak{N}}$, $N' = (-N^{-1}) \mod \overline{\mathfrak{W}}$ et un entier $a = (a_{2\mathfrak{N}-1} \dots a_0)_{\overline{\mathfrak{W}}} < \mathfrak{R}N$.
Sorties : L'unique entier $\lambda = (\lambda_{\mathfrak{N}-1} \dots \lambda_0)_{\overline{\mathfrak{W}}}$ tel que $\lambda = \text{REDC}(a) = (a\mathfrak{R}^{-1}) \mod N$.

1 $(\lambda_{2\mathfrak{N}-1} \dots \lambda_0)_{\overline{\mathfrak{W}}} \leftarrow (a_{2\mathfrak{N}-1} \dots a_0)_{\overline{\mathfrak{W}}}$;
2 **pour** $i = 0$ **à** $\mathfrak{N} - 1$ **faire**
3 $\quad \gamma_i \leftarrow (\lambda_i N') \mod \overline{\mathfrak{W}}$; `// Faire d'abord` $\lambda_i N'$`, puis le modulo`
4 $\quad \lambda \leftarrow \lambda + \gamma_i N \overline{\mathfrak{W}}^i$;
5 **fin**
6 **si** $\lambda \geq N$ **alors** $\lambda \leftarrow \lambda - N$;
7 **retourner** λ ;

Algorithme 2.3 : Réduction de Montgomery [CFA+05] page 181

Multiplication de Montgomery La réduction de Montgomery que l'on vient de voir a pour but de calculer efficacement la multiplication modulaire. L'équation 2.9 montre la relation entre la multiplication modulaire et la représentation de Montgomery avec $\mathfrak{R}$.

$$(a\mathfrak{R} \mod p)\,(b\mathfrak{R} \mod p)\,\mathfrak{R}^{-1} \mod p = ab\mathfrak{R} \mod p. \tag{2.9}$$

Grâce à l'équation 2.9 on déduit que $\text{REDC}([a]\,[b]) = [ab]$.

L'exemple 1 montre comment effectuer la multiplication de deux entiers en passant par la représentation de Montgomery. C'est un exemple minime, mais qui permet de bien comprendre le *domaine de Montgomery*.

Exemple 1. *Soient* $p = 32771$, $\mathfrak{W} = 8$, $\mathfrak{R} = \overline{\mathfrak{W}}^2 = (2^8)^2 = 65536$, $a = 9964$ *et* $b = 27565$. *D'où* $\mathfrak{R}' = \mathfrak{R}^2 \mod p = 36$ *et* $\mathfrak{R}^{-1} \mod p = 27309$.

$$\begin{aligned}
[a] &= \text{REDC}(a\mathfrak{R}') = \text{REDC}(9964 \times 36) = 9964 \times 36 \times \underbrace{27309}_{\mathfrak{R}^{-1}} \mod p = 5758 \\
[b] &= \text{REDC}(b\mathfrak{R}') = 31236 \\
[ab] &= \text{REDC}([a]\,[b]) = \text{REDC}(5758 \times 31236) = 9317 \\
ab &= \text{REDC}(9317) = 9317 \times 27309 \mod p = 3909.
\end{aligned}$$

Vérifions le résultat par un calcul direct sans le passage dans le domaine de Montgomery : $9964 \times 27565 \mod 32771 = \mathbf{3909}$ ✓.

Pour calculer directement $\text{REDC}(ab)$ avec a et b des entiers multiprécisions on va utiliser l'algorithme 2.4, qui combine l'algorithme 2.3 et la bien connue multiplication scolaire.

Exemple 2. *Soient* $p, \mathfrak{W}, \mathfrak{R}, a$ *et* b *les mêmes que pour l'exemple 1. Au vu de* a *et* b *et puisque* $\mathfrak{W} = 8$ *bits, on a* $\mathfrak{N} = 2$ *mots. Ainsi* $a = 9964 = [5758] = (22 \quad 126)_{\overline{\mathfrak{W}}}$ *et* $b = 27565 = [31236] = (112 \quad 4)_{\overline{\mathfrak{W}}}$

La table 2.1 illustre le déroulement de l'algorithme 2.4. Le résultat est $c = \text{REDC}(ab) = 9317$, *et correspond donc bien au résultat de l'exemple 1.*

Entrées : Le modulo $p = (p_{\mathfrak{N}-1} \dots p_0)_{\overline{\mathfrak{W}}}$ premier avec $\overline{\mathfrak{W}}$, $\mathfrak{R} = \overline{\mathfrak{W}}^{\mathfrak{N}}$, $p' = (-p^{-1}) \mod \overline{\mathfrak{W}}$ et deux entiers $[a] = (a_{\mathfrak{N}-1} \dots a_0)_{\overline{\mathfrak{W}}}$ et $[b] = (b_{\mathfrak{N}-1} \dots b_0)_{\overline{\mathfrak{W}}}$ en représentation de Montgomery tels que $0 \leq a, b < p$.
Sorties : L'unique entier $c = (c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}}$ tel que $c = (ab\mathfrak{R}^{-1}) \mod p$.

1 $c \leftarrow 0$;
2 **pour** $i = 0$ **à** $\mathfrak{N} - 1$ **faire**
3 $\quad u_i \leftarrow ((c_0 + a_i b_0)p') \mod \overline{\mathfrak{W}}$;
4 $\quad c \leftarrow \frac{c + a_i b + u_i p}{\overline{\mathfrak{W}}}$;
5 **fin**
6 **si** $c \geq p$ **alors** $c \leftarrow c - p$;
7 **retourner** c ;

Algorithme 2.4 : Multiplication multiprécision avec la représentation de Montgomery [CFA+05] page 204

Table 2.1 – Illustration de l'algorithme 2.4

i	a_i	u_i	c	
–	–	–	0	
0	126	88	26639	
1	22	51	**9317**	✓

Implémentation de la multiplication de Montgomery En raison de la valeur du quotient $\overline{\mathfrak{W}}$, cette méthode de multiplication d'entiers multiprécisions a une implémentation logicielle très efficace. Il y a plusieurs façons pour l'implanter, l'article [KAK96] est une référence classique pour renvoyer à la comparaison des différentes implémentations. Cette référence précise justement que la méthode de balayage de l'opérande grossièrement intégré – ou *Coarsely Integrated Operand Scanning* – (CIOS) est intéressante par rapport aux Finely Integrated Operand Scanning (FIOS), Coarsely Integrated Hybrid Scanning (CIHS), Finely Integrated Product Scanning (FIPS) ou encore Separated Operand Scanning (SOS). Nous allons ici présenter la méthode CIOS, elle est donnée par l'algorithme 2.5. La multiplication de Montgomery implémentée en CIOS est un choix généralement justifié pour son nombre d'additions, de multiplications et d'accès lecture et écriture intéressant par rapport aux autres [KAK96]. Cet algorithme est très important car nous allons le cibler avec des attaques canaux auxiliaires.

Autres opérations sur les corps premiers

L'importance est donnée à l'algorithme CIOS de multiplication modulaire car il est ciblé par les attaques par canaux auxiliaires. Les autres opérations arithmétiques sont brièvement décrites ci-dessous.

Racine carrée Un entier a modulo p est ou n'est pas un carré dans $\mathbb{F}_p$, dans le cas où a est un carré, on dit que c'est un résidu quadratique, donc il existe x tel que $x^2 = a \mod p$. Le symbole de Legendre[3] $\left(\frac{a}{p}\right)$ vaut 1 **si et seulement si** a est un résidu

[3] Voir par exemple [CFA+05] page 210 pour un algorithme permettant de calculer ce symbole.

Entrées : Le modulo $p = (p_{\mathfrak{N}-1} \dots p_0)_{\overline{\mathfrak{W}}}$ premier avec $\overline{\mathfrak{W}}$, $\mathfrak{R} = \overline{\mathfrak{W}}^{\mathfrak{N}}$, p' tel que $\mathfrak{R}\mathfrak{R}^{-1} - pp' = 1$ et deux entiers $a = (a_{\mathfrak{N}-1} \dots a_0)_{\overline{\mathfrak{W}}}$ et $b = (b_{\mathfrak{N}-1} \dots b_0)_{\overline{\mathfrak{W}}}$.
Sorties : L'unique entier $c = (c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}}$ tel que $c = \text{Redc}(ab) = (ab\mathfrak{R}^{-1}) \mod p$.

$c \leftarrow 0$;
pour $i = 0$ **à** $\mathfrak{N} - 1$ **faire**
  $u \leftarrow 0$;
  pour $j = 0$ **à** $\mathfrak{N} - 1$ **faire**
    $(uv) \leftarrow c_j + a_j b_i + u$;
    $c_j \leftarrow v$;
  fin
  $(uv) \leftarrow c_{\mathfrak{N}} + u$;
  $c_{\mathfrak{N}} \leftarrow v$;
  $c_{\mathfrak{N}+1} \leftarrow u$;
  $m \leftarrow c_0 p'_0 \mod \overline{\mathfrak{W}}$;
  $(uv) \leftarrow c_0 + m p_0$;
  pour $j = 1$ **à** $\mathfrak{N} - 1$ **faire**
    $(uv) \leftarrow c_j + m p_j + u$;
    $c_{j-1} \leftarrow v$;
  fin
  $(uv) \leftarrow c_{\mathfrak{N}} + u$;
  $c_{\mathfrak{N}-1} \leftarrow v$;
  $c_{\mathfrak{N}} \leftarrow c_{\mathfrak{N}+1} + u$;
fin
si $(c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}} < p$ **alors**
  $c \leftarrow (c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}}$;
sinon
  $c \leftarrow (c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}} - p$;
fin
retourner c ;

Algorithme 2.5 : Multiplication modulaire de Montgomery CIOS [KAK96]

quadratique modulo p. L'algorithme de Tonelli-Shanks permet de calculer les racines carrées, il fonctionne quel que soit p premier ([CFA+05] page 212).

Inversion modulaire L'inverse d'un élément $a \in \mathbb{F}_p^\star$, noté $a^{-1} \mod p$ est l'unique élément $x \in \mathbb{F}_p$ tel que $ax = 1 \mod p$. Une implémentation de l'algorithme d'Euclide étendu pour calculer l'inversion dans $\mathbb{F}_p$ est proposée dans [TKL86].

Montgomery [Mon85] a donné une méthode pour calculer l'inversion en utilisant la représentation de Montgomery. Pour un entier a, l'inverse de Montgomery de a est $\text{Inv}(a) = (a^{-1}\mathfrak{R}^2) \mod p$. Ainsi, $\text{Inv}([a]) = (a^{-1}\mathfrak{R}) \mod p = [a^{-1}]$. D'où la relation 2.10.

$$\text{Redc}([a]\,\text{Inv}([a])) = \mathfrak{R} \mod p = [1] \tag{2.10}$$

La traduction algorithmique de cette méthode a été déployée par Kaliski [Kal95].

2.1.3 Arithmétique sur les extensions de corps

Cette section apporte les informations théoriques sur les extensions de corps finis qui seront utilisées pour les couplages. Une partie est aussi destinée à leur implémentation.

Nous avons vu dans le paragraphe *Extension de corps* comment créer des corps finis d'ordre $q = p^d$, p premier et $d \geq 2$. La construction d'une extension finie d'un corps fini est donc donnée par le lemme 1.

Lemme 1. *Soit p premier et $d \geq 2$, alors*

$$\mathbb{F}_{p^d} \simeq \mathbb{F}_p[X]/(P), \tag{2.11}$$

avec P un polynôme de degré exactement d.

Une preuve du lemme 1 est donnée dans [Dem09] page 84.

Nous allons voir comment procéder efficacement aux calculs dans les extensions de degrés 2 et 3. Les travaux de Beuchat *et al.* [BGDM+10] détaillent les calculs dans les extensions de corps, ces extensions sont impliquées dans les calculs de couplages. Nous proposons ici de retranscrire les résultats pour les extensions de degrés 2 et 3.

Calcul dans les extensions quadratiques

Une extension quadratique d'un corps fini d'ordre p premier donne un corps fini, d'ordre p^2, isomorphe à $\mathbb{F}_p[X]/(X^2 - \beta)$. Koblitz *et al.* [KM05] montrent le lemme 2.

Lemme 2. *Soit p premier avec $p = 1 \mod 12$. Soit $d \geq 2$ un entier sous la forme $d = 2^i 3^j$ et soit $\beta \in \mathbb{F}_p$ qui est ni un carré ni un cube, alors le polynôme $X^d - \beta$ est irréductible sur $\mathbb{F}_p$.*

En notant u une racine du polynôme $X^2 - \beta$, les éléments de $\mathbb{F}_{p^2}$ sont les polynômes de degré 1 à coefficient dans $\mathbb{F}_p$ en la variable u. C'est-à-dire, $x \in \mathbb{F}_{p^2} \simeq \mathbb{F}_p[X]/(X^2 - \beta)$ s'écrit $x = x_0 + x_1 u$ avec $x_0, x_1 \in \mathbb{F}_p$.

Plus généralement, à partir des lemmes 1 et 2 on déduit la proposition 1.

Proposition 1 (Koblitz *et al.* [KM05] théorème 2)**.** *Soit un entier p premier avec $p = 1 \mod 12$. Soit $d \geq 2$ un entier sous la forme $d = 2^i 3^j$. Si $\beta \in \mathbb{F}_p$ est ni un carré ni un cube alors :*

$$\mathbb{F}_{p^d} \simeq \mathbb{F}_p[X]/(X^d - \beta). \tag{2.12}$$

À partir de cette représentation, les opérations sur $\mathbb{F}_{p^2}$ sont réalisées comme sur des polynômes. L'addition de x et y, éléments de $\mathbb{F}_{p^2}$ se fait simplement de la manière suivante :

$$x + y = (x_0 + x_1 u) + (y_0 + y_1 u) = (x_0 + y_0) + (x_1 + y_1)u.$$

La soustraction est construite sur le même modèle. La multiplication par un élément a du corps de base est très simple :

$$xa = (x_0 + x_1 u)a = (x_0 a) + (x_1 a)u.$$

Pour multiplier deux éléments de $\mathbb{F}_{p^2}$ il suffit de développer l'équation $(x_0+x_1u)(y_0+y_1u)$ ainsi :

$$\begin{aligned}(x_0+x_1u)(y_0+y_1u) &= x_0y_0+x_0y_1u+x_1y_0u+x_1y_1u^2 \quad \text{avec } u^2=\beta \\ &= (x_0y_0+x_1y_1\beta)+(x_0y_1+x_1y_0)u \\ &= (x_0y_0+x_1y_1\beta)+((x_0+x_1)(y_0+y_1)-x_0y_0-x_1y_1)\,u.\end{aligned} \tag{2.13}$$

La dernière égalité permet un gain d'une multiplication.
Pour la mise au carré, il suffit d'écrire :

$$\begin{aligned}(x_0+x_1u)^2 &= x_0^2+x_1^2u^2+2x_0x_1u \quad \text{avec } u^2=\beta \\ &= (x_0^2+x_1^2\beta)+((x_0+x_1)^2-x_0^2-x_1^2)\,u.\end{aligned} \tag{2.14}$$

Le calcul d'inverse s'obtient d'après l'équation :

$$(x_0+x_1u)(x_0-x_1u)=x_0^2-x_1^2u^2=x_0^2-x_1^2\beta,$$

qui permet d'obtenir l'équation :

$$(x_0+x_1u)^{-1} = \tfrac{x_0-x_1u}{x_0^2-x_1^2\beta} \tag{2.15}$$

Grâce à l'équation 2.15, l'inverse à calculer est dans le corps de base $\mathbb{F}_p$. L'annexe A.1 comporte les algorithmes de multiplication, mise au carré et inverse dans $\mathbb{F}_{p^2}$, ils sont issus de [BGDM+10].

L'estimation du coût de calculs de ces opérations se mesure en énumérant le nombre d'opérations sur $\mathbb{F}_p$. La table 2.2 indique ces coûts. Par exemple, une multiplication dans $\mathbb{F}_{p^2}$ nécessite 4 multiplications dans $\mathbb{F}_p$.

Table 2.2 – Coût des opérations dans $\mathbb{F}_{p^2}$

	$\boldsymbol{M}_{\mathbb{F}_p}$	$\boldsymbol{C}_{\mathbb{F}_p}$	$\boldsymbol{I}_{\mathbb{F}_p}$
$\boldsymbol{M}_{\mathbb{F}_{p^2}}$	4	0	0
$\boldsymbol{C}_{\mathbb{F}_{p^2}}$	1	3	0
$\boldsymbol{I}_{\mathbb{F}_{p^2}}$	3	1	1

Calcul dans les extensions cubiques

La marche à suivre est relativement identique pour écrire les opérations dans les extensions de degré 3 [BGDM+10]. On écrit $\mathbb{F}_{p^3}=\mathbb{F}_p[X]/(X^3-\xi)$, alors en notant v une racine du polynôme irréductible $X^3-\xi$, les éléments de $\mathbb{F}_{p^3}$ s'écrivent comme des polynômes de degré 2 à coefficients dans $\mathbb{F}_p$ en la variable v. C'est-à-dire, $x\in\mathbb{F}_{p^3}\simeq\mathbb{F}_p[X]/(X^3-\xi)$ se décompose en $x=x_0+x_1v+x_2v^2$ avec $x_0,x_1,x_2\in\mathbb{F}_p$.

L'addition est triviale. Pour la multiplication, le développement de $(x_0+x_1v+x_2v^2)(y_0+y_1v+y_2v^2)$ donne :

$$\begin{aligned}xy &= x_0y_0+x_1y_2\xi+x_2y_1\xi+(x_0y_1+x_1y_0+x_2y_2\xi)v+(x_0y_2+x_1y_1+x_2y_0)v^2 \\ &= ((x_1+x_2)(y_1+y_2)-x_1y_1-x_2y_2)\xi+x_0y_0 \\ &\quad +((x_0+x_1)(y_0+y_1)-x_0y_0-x_1y_1+x_2y_2\xi)v \\ &\quad +((x_0+x_2)(y_0+y_2)-x_0y_0-x_2y_2+x_1y_1)v^2.\end{aligned} \tag{2.16}$$

La mise au carré s'écrit de la manière suivante :

$$x^2 = x_0^2 + 2x_1x_2\xi + (x_2^2 + 2x_0x_1)v + (x_1^2 + 2x_0x_2)v^2. \tag{2.17}$$

L'inverse de $x = x_0 + x_1v + x_2v^2$ se trouve en résolvant l'équation 2.18.

$$(x_0 + x_1v + x_2v^2)(z_0 + z_1v + z_2v^2) = 1. \tag{2.18}$$

En se servant de l'équation 2.16, on trouve le système suivant à résoudre.

$$\begin{cases} x_0y_0 + x_1y_2\xi + x_2y_1\xi = 1 \\ x_0y_1 + x_1y_0 + x_2y_2\xi = 0 \\ x_0y_2 + x_1y_1 + x_2y_0 = 0 \end{cases} \tag{2.19}$$

La recherche de la solution peut se faire via un logiciel de calcul formel. Un exemple est donné dans le code 2.1 en utilisant Xcas[4].

```
1 f(x0,x1,x2,z0,z1,z2,ksi):=x0*z0+x1*z2*ksi+x2*z1*ksi
2 g(x0,x1,x2,z0,z1,z2,ksi):=x0*z1+x1*z0+x2*z2*ksi
3 h(x0,x1,x2,z0,z1,z2,ksi):=x0*z2+x1*z1+x2*z0
4 [z0,z1,z2]:=linsolve([f(x0,x1,x2,z0,z1,z2,ksi)=1,g(x0,x1,x2,
    z0,z1,z2,ksi)=0,h(x0,x1,x2,z0,z1,z2,ksi)=0],[z0,z1,z2])
```

Code 2.1 – Code Xcas pour la résolution du système d'équations 2.19

Il en résulte que $z = z_0 + z_1v + z_2v^2$ est donné par

$$\begin{aligned} z_0 &= \frac{x_0^2 - \xi x_1x_2}{t} \\ z_1 &= \frac{\xi^2x_2^2 - x_0x_1}{t} \\ z_2 &= \frac{x_1^2 - x_0x_2}{t} \\ t &= \xi^2x_2^3 - 3x_0x_1x_2 + \xi x_1^3 + x_0^3 \end{aligned} \tag{2.20}$$

Les algorithmes correspondants sont donnés dans l'annexe A.2. Les coûts des opérations sont explicités dans la table 2.3.

Table 2.3 – Coût des opérations dans $\mathbb{F}_{p^3}$

	$\boldsymbol{M}_{\mathbb{F}_p}$	$\boldsymbol{C}_{\mathbb{F}_p}$	$\boldsymbol{I}_{\mathbb{F}_p}$
$\boldsymbol{M}_{\mathbb{F}_{p^3}}$	8	0	0
$\boldsymbol{C}_{\mathbb{F}_{p^3}}$	5	3	0
$\boldsymbol{I}_{\mathbb{F}_{p^3}}$	13	3	1

[4] [PDG08] *le couteau suisse des mathématiques* https://www-fourier.ujf-grenoble.fr/~parisse/giac_fr.html.

Calcul dans les tours d'extensions de corps finis

Dans le cadre des couplages que nous avons choisi d'utiliser, nous allons devoir faire des calculs sur $\mathbb{F}_{p^{12}}$. Puisque 12 se décompose en $12 = 2^2 \cdot 3$, il y a trois tours d'extensions possibles : $2 \cdot 2 \cdot 3$ ou bien $2 \cdot 3 \cdot 2$ ou bien $3 \cdot 2 \cdot 2$. Beuchat *et al.* [BGDM+10] justifient le choix $2 \cdot 3 \cdot 2$ en raison du coût de calcul (nombre d'opérations de $\mathbb{F}_p$ pour calculer les opérations de $\mathbb{F}_{p^{12}}$). En pratique, cela consiste donc à construire $\mathbb{F}_{p^2}$, puis, se servir de ce corps comme un corps de base pour faire une extension cubique. On obtiendra $\mathbb{F}_{p^6}$, et recommencer avec une extension quadratique. La tour d'extension s'écrit :

$$\begin{aligned} \mathbb{F}_{p^2} &= \mathbb{F}_p[u]/(u^2 - \beta), \quad \text{avec } \beta = -5 \\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[v]/(v^3 - \xi), \quad \text{avec } \xi = u \\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[w]/(w^2 - \gamma), \quad \text{avec } \gamma = v. \end{aligned} \tag{2.21}$$

Avec une telle tour d'extension, le coût des opérations dans $\mathbb{F}_{p^{12}}$ est donné dans la table 2.4.

Table 2.4 – Coût des opérations dans $\mathbb{F}_{p^{12}}$ avec la tour $2 \cdot 3 \cdot 2$

	$\boldsymbol{M}_{\mathbb{F}_p}$	$\boldsymbol{C}_{\mathbb{F}_p}$	$\boldsymbol{I}_{\mathbb{F}_p}$
$\boldsymbol{M}_{\mathbb{F}_{p^{12}}}$	128	0	0
$\boldsymbol{C}_{\mathbb{F}_{p^{12}}}$	101	27	0
$\boldsymbol{I}_{\mathbb{F}_{p^{12}}}$	200	29	1

Les opérations dans $\mathbb{F}_{p^6}$, puis dans $\mathbb{F}_{p^{12}}$ sont détaillées, respectivement, dans les annexes A.3 et A.4.

2.1.4 Courbes elliptiques

Les courbes elliptiques sont utilisées en cryptographie pour différentes fins. En 1985, deux mathématiciens proposent indépendamment l'utilisation des courbes elliptiques pour l'échange de clefs et la cryptographie asymétrique. Il s'agit de Miller [Mil85] et de Koblitz [Kob87], ils sont les pionniers de ce que l'on appelle aujourd'hui, la cryptographie sur courbes elliptiques (ECC). L'aspect calcul est réalisé grâce à l'arithmétique des points d'une courbe elliptique. L'aspect sécurité est basé sur le problème du logarithme discret (DLOG). Les courbes elliptiques sont aussi utilisées pour calculer efficacement des couplages. En effet, un opérateur bilinéaire peut être défini entre les groupes des points des courbes elliptiques. Nous décrivons les couplages dans la section 2.1.6. En 1992, Vanstone [Van92] répond à un appel du NIST sur les signatures numériques. Cette méthode de Elliptic Curve Digital Signature Algorithm (ECDSA) est plus avantageuse que celles utilisées à l'époque (RSA et Digital Signature Algorithm (DSA)) en termes de longueur de clefs et de rapidité des calculs.

Par ailleurs, Lenstra [Len85] montre l'utilisation des courbes elliptiques pour la factorisation des entiers. Cette méthode est l'une des plus efficaces lorsqu'il s'agit de factoriser des entiers de 64 bits maximum.

Propriétés élémentaires

Nous allons donner les éléments pratiques pour définir la notion de courbe elliptique. Les livres [Hin11] et [Sil09] sont d'excellentes références pour approfondir le sujet.

Définition 4. *Une courbe elliptique E sur un corps $\mathbb{K}$ est définie par l'équation de Weierstrass :*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.22}$$

où les coefficients $a_1, a_2, a_3, a_4, a_6 \in \mathbb{K}$ sont tels que pour chaque point (x, y) à coordonnées dans $\overline{\mathbb{K}}$ satisfaisant 2.22 alors les dérivées partielles $2y + a_1x + a_3$ et $3x^2 + 2a_2x + a_4 - a_1y$ ne s'annulent pas simultanément.

La condition de non-annulation simultanée des dérivées partielles indique si une courbe est lisse, on dit aussi que la courbe est *non singulière*. On vérifie généralement cette condition à l'aide du discriminant de E. En effet, $\Delta(E) \neq 0$ est une condition suffisante pour que les dérivées partielles ne s'annulent pas simultanément.

Définition 5 (Discriminant et $j-$invariant, [Sil09] page 42)**.** *Soit E une courbe elliptique sur $\mathbb{K}$ définie par l'équation 2.22. Le discriminant de E, $\Delta(E)$, est $-b_2^2b_8 - 2b_4^3 - 27b_6^2 + 9b_2b_4b_6$ et le $j-$invariant de E, $\mathfrak{j}(E)$, est $\dfrac{c_4^3}{\Delta(E)}$. Avec*

$$\begin{cases} b_2 &= a_1^2 + 4a_2 \\ b_4 &= a_1a_3 + a_4 \\ b_6 &= a_3^2 + 4a_6 \\ b_8 &= a_1^2 + a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \\ c_4 &= b_2^2 - 24b_4. \end{cases}$$

Loi de groupe

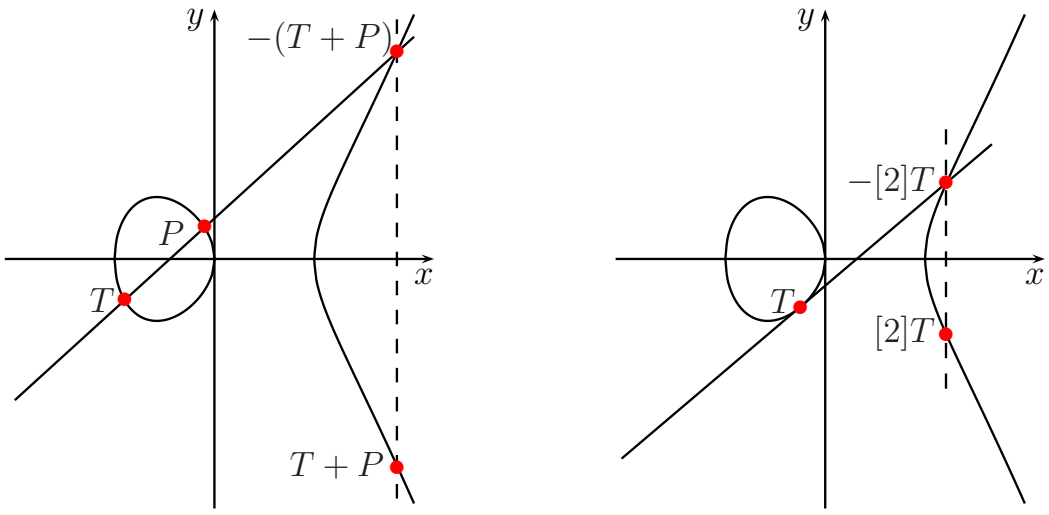
Soit E une courbe elliptique sur un corps $\mathbb{K}$, l'addition de deux points $T = (x_T, y_T)$ et $P = (x_P, y_P)$ sur la courbe est facile à visualiser (sur le corps $\mathbb{K} = \mathbb{R}$). Il suffit de tracer la ligne passant par T et P, elle intersecte la courbe E en un troisième point (d'après le théorème de Bézout pour les courbes elliptiques[5], il faut alors prendre l'opposé sur l'axe des x de ce point pour obtenir $T + P$. Dans le cas $T = P$, la ligne passant par T et P est remplacée par la tangente en T. La figure 2.1 permet de visualiser l'addition de deux points (figure 2.1a) et le doublement (figure 2.1b) dans le cas où $\mathbb{K} = \mathbb{R}$.

Voyons maintenant ce qu'il se passe lorsque les deux points T et P ont la même coordonnée en x. La droite passant par ces deux points intersecte la courbe en un troisième point, on ne peut pas le représenter, c'est le point à l'infini. Il est noté $\mathcal{P}_\infty$ et il est l'élément neutre pour la loi de groupe. On peut virtuellement imaginer ce point comme celui qui intersecte toutes les lignes verticales $x = constante$. Deux points ayant la même coordonnée sur x sont de la forme $T = (x, y)$ et $P = (x, -y)$. Comme la ligne passant par T et P intersecte $\mathcal{P}_\infty$, l'addition $T + P$ donne ce point $\mathcal{P}_\infty$. Le point $\mathcal{P}_\infty$ est bien l'élément neutre pour l'addition ($T + P = \mathcal{P}_\infty$), et donc $P = -T$.

Proposition 2 ([Sil09] chapitre III.2)**.** *$(E, +)$ est un groupe abélien, $\mathcal{P}_\infty$ est l'élément neutre.*

Démonstration. Il y a différents points à démontrer :

[5] Une démonstration de ce théorème est donnée dans [Per95].



(a) $T + P$ (b) $[2]T$

Figure 2.1 – Ajout et doublement de points sur courbe elliptique ($\mathbb{K} = \mathbb{R}$)

- Loi de composition interne : le théorème de Bézout pour les courbes elliptiques (voir par exemple [Per95]) assure que la droite passant par T et P coupe la courbe en un troisième point, l'opposé de ce point est $T + P$, il est sur la courbe grâce à la symétrie de la courbe elliptique selon l'axe x.

- Associativité : montrer directement que pour tout $P, Q, R \in E$ alors $(P+Q)+R = P + (Q + R)$ est très fastidieux, et nécessite un logiciel de calcul formel, Friedl propose une preuve sur sa page http://math.rice.edu/~friedl/papers/AAELLIPTIC.PDF avec l'utilisation de Cocoa. Une autre méthode[6] utilise les diviseurs des courbes elliptiques que nous verrons à la section 2.1.6.

- Élément neutre : pour tout $P \in E$, $P + \mathcal{P}_\infty = P + \mathcal{P}_\infty = P$.

- Symétrique : pour tout $P \in E$, $(x_P, y_P) + (x_P, -y_P) = \mathcal{P}_\infty$. Le point $(x_P, -y_P)$ est noté $-P$ et se nomme l'opposé de P. Et $-P \in E$ car $y_{-P}^2 = (-y_P)^2 = y_P^2 = x_P^3 + ax_P + b$.

- Commutativité : pour tout $P, Q \in E$ il faut montrer que $P + Q = Q + P$. Par construction, la ligne (PQ) passant par P et Q est la même que celle passant par Q et P.

□

Définition 6. *Soient E_1 et E_2 deux courbes elliptiques sur $\mathbb{K}$ données par leurs équations de Weierstrass :*

$$\begin{array}{rrcl} E_1 : & y^2 + a_1xy + a_3y & = & x^3 + a_2x^2 + a_4x + a_6 \\ E_2 : & y'^2 + a'_1x'y' + a'_3y' & = & x'^3 + a'_2x'^2 + a'_4x' + a'_6. \end{array}$$

[6] Disponible sur http://www.math.uwaterloo.ca/~djao/co690.2007/grouplaw.pdf.

E_1 et E_2 sont dites $\mathfrak{L}-$isomorphes s'il existe $u, r, s, t \in \mathfrak{L}$ avec $u \neq 0$ tels que le changement de variable

$$(x, y) \to (u^2x' + r, u^3y' + u^2sx' + t) \tag{2.23}$$

transforme l'équation de E_1 en celle de E_2. La transformation 2.23 est appelée changement de variables admissible.

L'équation de Weierstrass 2.22 de E peut être simplifiée en appliquant un changement de variables admissible (voir par exemple [BSS99] chapitre III.2). Nous utiliserons l'équation simplifiée pour la suite de cette thèse. Il y a trois cas à différencier selon la caractéristique du corps $\mathbb{K}$. Nous présentons uniquement le cas des grandes caractéristiques, c'est-à-dire $\text{Char}(\mathbb{K}) \neq 2, 3$, car les couplages utilisent seulement ces courbes pour des raisons de sécurité [JOP14]. Pour les autres cas, nous renvoyons à [HMV06] page 78.

Le changement de variables admissible

$$(x, y) \to \left(\frac{x - 3a_1^2 - 12a_2}{36}, \frac{y - 3a_1x}{216} - \frac{a_1^3 + 4a_1a_2 - 12a_3}{24}\right), \tag{2.24}$$

transforme l'équation de E en

$$E : y^2 = x^3 + ax + b, \tag{2.25}$$

avec $a, b \in \mathbb{K}$. On a $\Delta(E) = -16(4a^3 + 27b^2)$ et $\text{j}(E) = \frac{1728 \cdot 4a^3}{4a^3+27b^2}$.

Les constructions géométriques d'addition et de doublement de points se traduisent par les formules explicites données ci-dessous. De même que pour les changements de variables admissibles, nous présentons les formules pour $\text{Char}(\mathbb{K}) \neq 2, 3$, voir par exemple [HMV06] page 81 pour les autres caractéristiques.

Formules d'addition et doublement pour $\text{Char}(\mathbb{K}) \neq 2, 3$ Nous considérons l'équation courte de Weierstrass 2.25 pour définir E, c'est-à-dire $E : y^2 = x^3 + ax + b$. Soient $T = (x_T, y_T)$ et $P = (x_P, y_P)$ deux points de E avec $T \neq \pm P$, alors

$$x_{T+P} = \left(\frac{y_P - y_T}{x_P - x_T}\right)^2 - x_T - x_P \quad \text{et} \quad y_{T+P} = \left(\frac{y_P - y_T}{x_P - x_T}\right)(x_T - x_{T+P}) - y_T. \tag{2.26}$$

Le doublement de $P \neq -P$ est donné par

$$x_{[2]T} = \left(\frac{3x_T^2 + a}{2y_T}\right)^2 - 2x_T \quad \text{et} \quad y_{[2]T} = \left(\frac{3x_T^2 + a}{2y_T}\right)\left(x_T - x_{[2]T}\right) - y_T. \tag{2.27}$$

Points rationnels et de torsion

Structure de groupe des points rationnels On considère la courbe elliptique E sur $\mathbb{K}$ définie par l'équation 2.22. L'ensemble des points $\mathbb{K}-$rationnels de E, noté $E(\mathbb{K})$ est

$$E(\mathbb{K}) = \left\{(x, y) \in \mathbb{K} | y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6\right\} \cup \{\mathcal{P}_\infty\}. \tag{2.28}$$

Définition 7 ([Sil09] page 70)**.** *Soient E une courbe elliptique sur $\mathbb{K}$ d'équation E : $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ et*

$$\begin{array}{rccc} \sigma : & \mathbb{K} & \to & \mathbb{K} \\ & a & \mapsto & a^p, \end{array}$$

le morphisme de corps de Frobenius. Soit $E^\sigma : y^2 + \sigma(a_1)xy + \sigma(a_3)y = x^3 + \sigma(a_2)x^2 + \sigma(a_4)x + \sigma(a_6)$, alors E^σ est une courbe elliptique sur $\mathbb{K}$ avec $\Delta(E^\sigma) = \sigma(\Delta(E))$ et $\mathrm{j}(E^\sigma) = \sigma(\mathrm{j}(E))$.

Le morphisme de Frobenius Ψ_p de E vers E^σ est

$$\begin{array}{rccc} \Psi_p : & E & \to & E^\sigma \\ & (x, y) & \mapsto & (x^p, y^p) \end{array}. \tag{2.29}$$

Le point à l'infini est envoyé sur lui-même, $\Psi_p(\mathcal{P}_\infty) = \mathcal{P}_\infty$.

Étudions le cas $\mathbb{K} = \mathbb{F}_q$ avec $q = p^d$. Puisque $\Psi_q = \underbrace{\Psi_p \circ \cdots \circ \Psi_p}_{d \text{ fois}} = \mathcal{I}d_{E(\mathbb{F}_q)}$, on a $E^{\sigma^d} = E$. Plus précisément

$$P \in E(\mathbb{F}_q) \Leftrightarrow \Psi_q(P) = P. \tag{2.30}$$

Le théorème 3 donne une information sur la structure de groupe des points $\mathbb{F}_q-$rationnels d'une courbe E.

Théorème 3 (Mordell-Weil)**.** *Le groupe $E(\mathbb{F}_q)$ est soit cyclique soit le produit de deux groupes cycliques. Dans le premier cas, on a :*

$$E(\mathbb{F}_q) \simeq \mathbb{Z}/d_2\mathbb{Z}, \tag{2.31}$$

où $d_2 = \#E(\mathbb{F}_q)$. Dans le second cas, on a :

$$E(\mathbb{F}_q) \simeq \mathbb{Z}/d_1\mathbb{Z} \times \mathbb{Z}/d_2\mathbb{Z}, \tag{2.32}$$

avec d_1 qui divise d_2 et $q - 1$.

Structure de groupe des points de $m-$torsion L'ensemble des points de $m-$torsion est utile pour définir les couplages, ces derniers sont définis à la section 2.1.6.

Nous définissons l'application :

$$\begin{array}{rccc} [m] : & E & \to & E \\ & P & \mapsto & \underbrace{P + \cdots + P}_{m \text{ fois}}, \end{array} \tag{2.33}$$

pour $m \in \mathbb{N}^\star$. Si $m \leq 0$, nous posons $[m]P = -[-m]P$ et $[0]P = \mathcal{P}_\infty$. Ainsi $[m]$ est un morphisme pour tout $m \in \mathbb{Z}$.

Définition 8 (Points de $m-$torsion)**.** *Soient $m \in \mathbb{N}^\star$ et E une courbe elliptique sur un corps $\mathbb{K}$. L'ensemble des points de $m-$torsion de E, noté $E[m]$, est*

$$E[m] = \ker([m]) = \left\{ P \in E\left(\overline{\mathbb{K}}\right) | [m]P = \mathcal{P}_\infty \right\}. \tag{2.34}$$

$E[m]$ est un sous-groupe de E. La notation $\overline{\mathbb{K}}$ désigne la clôture algébrique de $\mathbb{K}$ ([DD79, Jac80]). L'ensemble des points de torsion $\mathbb{K}-$rationnels est $E(\mathbb{K})[m] = E(\mathbb{K}) \cap E[m]$.

Si l'on prend deux entiers $m, n \in \mathbb{N}^\star$ premiers entre eux alors le théorème de Bézout donne l'existence de $u, v \in \mathbb{Z}$ tels que $um + vn = 1$. On peut ainsi définir le morphisme suivant :

$$\begin{array}{ccc} E[mn] & \to & E[m] \times E[n] \\ P & \mapsto & ([vn]P, [um]P) \\ P+Q & \hookleftarrow & (P, Q). \end{array}$$

Pour $m \in \mathbb{N}^\star$, la structure des points de $m-$torsion de E est alors donnée par le théorème 4.

Théorème 4 ([Sil09] page 89)**.** *On fait la distinction des cas selon la caractéristique de $\mathbb{K}$:*

- *Si $Char(\mathbb{K}) = 0$ ou si $pgcd(m, Char(\mathbb{K})) = 1$ alors*

$$E[m] \simeq \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}. \tag{2.35}$$

- *Si $Char(\mathbb{K}) = p \neq 0$ et $m = p^r$ alors,*
  - *soit :*
  $$E[p^r] = \{\mathcal{P}_\infty\} \quad \text{pour tout} \quad r \geq 1, \tag{2.36}$$
  et alors E est dite supersingulière,
  - *soit :*
  $$E[p^r] \simeq \mathbb{Z}/p^r\mathbb{Z} \quad \text{pour tout} \quad r \geq 1, \tag{2.37}$$
  et dans ce cas E est dite ordinaire.

Ainsi, une courbe E sur $\mathbb{F}_p, p > 3$ est supersingulière **si et seulement si** $\#E(\mathbb{F}_p) = p + 1$. Dans le cas où $Char(\mathbb{F}_q) = 2$ ou 3 alors E est supersingulière **si et seulement si** son $j-$invariant est nul.

Exemple 3. *Soit $p = 503$ et E la courbe elliptique sur $\mathbb{F}_p$ définie par $E : y^2 = x^3 + 14x + 9$. On a $\Delta(E) = 149$ et $j(E) = 64$. Puisque $p > 3$ et $\#E(\mathbb{F}_p) = 528 \neq p + 1$, E est une courbe ordinaire.*

Le point $P = (240, 145)$ est d'ordre 528 donc le groupe $E(\mathbb{F}_p)$ est cyclique, un générateur de ce groupe est P (nous sommes dans le cas 1 du théorème 3).

Il y des points de $m-$torsion pour chaque m qui divise 528. Comme $528 = 2^4 \cdot 3 \cdot 11$ on note par exemple que le point $Q = (121, 503)$ est d'ordre 3 (c'est-à-dire $Q \in E[3]$) et $(157, 208) \in E[11]$.

En revanche, si m n'est pas un diviseur de $\#E(\mathbb{F}_p)$ alors les points de $m-$torsion ont leurs coordonnées dans une extension de $\mathbb{F}_p$. Par exemple, il y a un point de $60-$torsion dont ces coordonnées sont dans $\mathbb{F}_{p^2} \simeq \mathbb{F}_p[u]$ avec u racine du polynôme irréductible $X^2 + X + 1$. Ce point est $S = (284u + 242, 192u + 431)$.

Il est intéressant de noter le résultat suivant concernant les racines de l'unité d'un corps fini.

Proposition 3 ([Vit08] lemme 1.12)**.** *Soit E une courbe elliptique sur $\mathbb{F}_q$, avec $q = p^d$, p premier, et soit $k \geq 1$ un entier. Alors pour un entier m premier avec p :*

$$E[m] \subset E\left(\mathbb{F}_{q^k}\right) \implies \mu_m \subset \mathbb{F}_{q^k}^{\star}, \tag{2.38}$$

où μ_m désigne l'ensemble des racines $m-$ièmes de l'unité de $\overline{\mathbb{F}_q}$ et $\mathbb{F}_{q^k}^{\star} = \mathbb{F}_{q^k} \setminus \{0\}$.

La preuve de cette proposition utilise le couplage de Weil que nous introduisons dans la définition 16.

Ordre du groupe

Nous venons de le voir dans l'exemple précédent, il est parfois nécessaire de connaître l'ordre de $E(\mathbb{F}_q)$. Mais le plus important est que pour la construction de primitive cryptographique basée sur les courbes elliptiques, il est indispensable de se placer dans un environnement où le DLOG est difficile. Et justement, le niveau de difficulté à résoudre le DLOG dépend de $\#E(\mathbb{F}_q)$.

Le théorème d'Euler, qui donne $\left(\frac{a}{p}\right) = a^{(p-1)/2}$, permet d'identifier le nombre de résidus quadratiques non nuls modulo p, il y en a $(p-1)/2$. Ainsi, on considère une courbe E sur $\mathbb{F}_p$, p premier, définie par $E : y^2 = x^3 + ax + b$ alors pour $x \in \mathbb{F}_p$,

$$\begin{cases} \text{si } x^3 + ax + b \text{ est un carré alors on compte deux points sur } E(\mathbb{F}_p), \\ \text{si } x^3 + ax + b \text{ n'est pas un carré alors cela ne crée pas de points sur } E(\mathbb{F}_p). \end{cases}$$

Et comme il y a exactement la moitié des éléments de $\mathbb{F}_p$ qui sont des carrés on compte qu'il y a approximativement $p + 1$ points (ne pas oublier $\mathcal{P}_\infty$) sur $E(\mathbb{F}_p)$.

Le théorème de Hasse donne une approximation plus précise de $E(\mathbb{F}_p)$.

Théorème 5 (Borne de Hasse)**.** *Soit E une courbe elliptique sur $\mathbb{F}_q$, alors :*

$$q + 1 - 2\sqrt{q} \leq \#E(\mathbb{F}_q) \leq q + 1 + 2\sqrt{q}. \tag{2.39}$$

Autrement dit $\#E(\mathbb{F}_q) = q + 1 - t$ avec $|t| \leq 2\sqrt{q}$. L'entier t est appelé trace de Frobenius (noté $\mathbf{tr}\,(\Psi_q)$), cela vient du fait que $E(\mathbb{F}_q) = ker\,([1] - \Psi_q)$ (cf. équation 2.30). De plus, pour tout entier $t \in [\![-2\sqrt{q}, 2\sqrt{q}]\!]$ il existe au moins une courbe elliptique E sur $\mathbb{F}_q$ d'ordre $q + 1 - t$ [Deu41].

Si E est une courbe elliptique sur $\mathbb{F}_q$ alors E est définie sur n'importe quelle extension $\mathbb{F}_{q^k}$ de $\mathbb{F}_q$. On note $t_k = \mathbf{tr}\left(\Psi_{q^k}\right) = \mathbf{tr}\left((\Psi_q)^k\right)$. La séquence (t_k) satisfait la relation :

$$t_{k+2} = t \cdot t_{k+1} - q \cdot t_k,$$

où $t_1 = t = \mathbf{tr}\,(\Psi_q)$ et $t_0 = 2$. On a alors

$$\#E(F_{q^k}) = q^k + 1 - t_k. \tag{2.40}$$

Cette relation est notamment utile pour prouver la proposition 4.

Proposition 4. *Soit E une courbe sur $\mathbb{F}_q$ avec $q = p^d$ et p premier. Alors*

$$E \text{ et supersingulière } \Leftrightarrow p \text{ divise } \boldsymbol{tr}\,(\Psi_q)\,.$$

Isomorphisme de courbes

Dans cette partie, nous nous intéressons aux isomorphismes de courbes elliptiques. De telles applications sont des morphismes bijectifs entre deux courbes elliptiques. Ils ont une forme particulière et permettent d'introduire la notion de courbe tordue (*twist* en anglais). Les courbes tordues ont un rôle important pour les couplages, car des optimisations de calcul consistent à utiliser ces courbes. Les deux propositions suivantes sont tirées de [Sil09] chapitre III.1.

Proposition 5. *Soient E_1 et E_2 deux courbes elliptiques sur $\mathbb{K}$ définies par E_1 : $y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$ et $E_2 : y'^2 + a'_1x'y' + a'_3y' = x'^3 + a'_2x'^2 + a'_4x' + a'_6$. L'isomorphisme entre les courbes E_1 et E_2 décrit par l'équation 2.23 (définition 6 page 28) donne l'unique forme que peut prendre un isomorphisme de courbes.*

Proposition 6. *Deux courbes elliptiques E_1 et E_2 sur $\mathbb{K}$ sont isomorphes* ***si et seulement si*** $\mathrm{j}(E_1) = \mathrm{j}(E_2)$.

Cet isomorphisme n'est pas nécessairement défini sur $\mathbb{K}$, dans ce cas on dit que E_2 est la tordue de E_1. Plus de précisions avec $\mathrm{Char}(\mathbb{K}) \neq 2, 3$ ([Sil09] page 47), on note les deux courbes elliptiques E_1 et E_2 sur $\mathbb{K}$ définies par $E_1 : y^2 = x^2 + ax + b$ et $E_2 : y'^2 = x'^2 + a'x' + b'$ de même $j-$invariant différent de 0 et 1728. Alors

$$\mathrm{j}(E_1) = \mathrm{j}(E_2) \implies a^3b'^2 = a'^3b^2.$$

On pose $v = \frac{a'b}{ab'} \in \mathbb{K} \setminus \{0\}$, alors

$$E_2 : y'^2 = x'^3 + av^{-2}x' + bv^{-3}. \tag{2.41}$$

On distingue deux cas selon si v est un carré dans $\mathbb{K}$:

- Soit il existe $u \in \mathbb{K}$ tel que $u^2 = v$ et donc

$$\phi : (x, y) \mapsto (u^2x', u^3y'), \tag{2.42}$$

est un $\mathbb{K}-$isomorphisme entre les courbes E_1 et E_2.

- Soit v n'est pas un carré dans $\mathbb{K}$, alors les courbes E_1 et E_2 ne sont pas $\mathbb{K}-$isomorphes. Mais on peut prendre une racine u de v dans une extension quadratique $\mathfrak{L}$ de $\mathbb{K}$. Ainsi ϕ est un $\mathfrak{L}-$isomorphisme entre E_1 et E_2. Dans ce cas on dit que E_2 est une tordue quadratique de E_1.

Corollaire 1 ([Vit08] page 5)**.** *Soient deux courbes elliptiques E_1 et E_2 sur $\mathbb{F}_q$ avec $q = p^d$ (p premier) qui ont le même $j-$invariant. Alors si $\mathrm{j} \neq 0, 1728$ les deux courbes sont $\mathbb{F}_{q^2}-$isomorphes.*

Prenons maintenant le cas où $\mathrm{j}(E) = 0$. Cela implique que $a = 0$, la seule condition sur u est $u^6 = \frac{b}{b'}$, donc E admet une tordue de degré 3 ou 6 (appelée respectivement tordue cubique et tordue sextique).

Avec $\mathrm{j}(E) = 1728$, on a $b = 0$ et donc la seule condition sur u est $u^4 = \frac{a}{a'}$ et alors E admet une tordue de degré 2 ou 4 (les tordues de degré 4 s'appellent tordues quartiques).

Systèmes de coordonnées et arithmétique des courbes elliptiques

Dans cette section, nous allons nous attarder sur la représentation des courbes elliptiques sur $\mathbb{F}_q$ avec $\mathrm{Char}(\mathbb{F}_q) > 3$. Le terme *arithmétique des courbes elliptiques* fait référence aux formules d'addition et de doublement de points. La complexité algorithmique d'un tel calcul s'exprime en énumérant les opérations coûteuses sur $\mathbb{F}_p$. Il sera donc noté par M (resp. C, I) le nombre de multiplications (resp. carrés, inversions) dans $\mathbb{F}_p$.

Coordonnées affines Soit E une courbe elliptique sur $\mathbb{F}_p$ définie par $E : y^2 = x^3 + ax + b$, la représentation la plus directe d'un point $P \in E$ est de noter $P = (x_P, y_P)$ avec x_P et y_P dans $\mathbb{F}_p$ qui satisfont l'équation de E. On dit que ce système de représentation utilisé est en coordonnées affines. L'arithmétique de ce système de coordonnées a été présentée au paragraphe *Formules d'addition et doublement pour Char($\mathbb{K}$) $\neq 2, 3$* page 29 à l'aide des équations 2.26 et 2.27.

Les formules d'addition et de doublement en coordonnées affines ont une complexité de $M + C + I$ et $M + 2C + I$ respectivement.

Coordonnées projectives Une autre manière de représenter les points d'une courbe elliptique est l'utilisation des coordonnées projectives. Même si elle rajoute de la redondance au sens où un point en coordonnées affines a plusieurs représentations possibles en coordonnées projectives, ce système permet d'éviter le très coûteux calcul d'inverse. Pour donner un ordre de grandeur du coût de l'inversion, Beuchat *et al.* [BGDM+10] ont une implémentation qui leur donne $I = 48, 3M$.

Soit $E : y^2 = x^3 + ax + b$ une courbe elliptique sur $\mathbb{F}_p$, on opère le changement de variable :

$$\begin{cases} x &= X/Z \\ y &= Y/Z. \end{cases} \tag{2.43}$$

Ainsi l'équation de E devient :

$$E : Y^2Z = X^3 + aXZ^2 + bZ^3. \tag{2.44}$$

Donc pour un $Z \neq 0$, le point $(X : Y : Z)$ correspond au point $({}^{X}/_{Z}, {}^{Y}/_{Z})$. Dès qu'un point voit sa coordonnée en Z devenir 0 alors il correspond au point à l'infini. Dans ce système $\mathcal{P}_\infty$ sera une ligne et donc noté $(0 : 1 : 0)$. Enfin, l'opposé d'un point $P = (X_P : Y_P : Z_P)$ est $-P = (X_P : -Y_P : Z_P)$.

Les formules d'addition et de doublement en coordonnées projectives s'obtiennent avec le changement de variable 2.43 appliqué à l'équation 2.26.

$$x_{T+P} = \frac{A^2 Z_T Z_P - B^3 - 2B^2 X_P Z_T}{B^2 Z_T Z_P},$$

où $A = Y_T Z_P - Y_T Z_P$ et $B = X_T Z_P - X_P Z_T$.

$$y_{T+P} = \frac{A X_T B^2 Z_P - AC - Y_T B^3 Z_P}{B^3 Z_T Z_P},$$

avec $C = A^2 Z_T Z_P - B^3 - 2B^2 X_P Z_T$. Il convient donc de poser $Z_{T+P} = B^3 Z_T Z_P$ pour obtenir les formules d'additions en coordonnées projectives, données par l'équation 2.45.

$$X_{T+P} = BC, \quad Y_{T+P} = A X_T B^2 Z_P - AC - Y_T B^3 Z_P \quad \text{et} \quad Z_{T+P} = B^3 Z_T Z_P, \quad (2.45)$$

où $T \neq \pm P$.

Les formules de doublement viennent de la même façon :

$$x_{[2]T} = \frac{D^2 - 8F Y_T X_T}{4F^2},$$

avec $D = 3X_T^2 + aZ_T^2$ et $F = Y_T Z_T$.

$$y_{[2]T} = \frac{D(4G - H) - 8Y_T^2 F^2}{8F^3},$$

avec $G = F Y_T X_T$ et $H = D^2 - 8G$. Il est alors pratique de poser $Z_{[2]T} = 8F^3$, et ainsi l'équation 2.46 donne les formules de doublement en coordonnées projectives.

$$X_{[2]T} = 2FH, \quad Y_{[2]T} = D(4G - H) - 8Y_T^2 F^2 \quad \text{et} \quad Z_{[2]T} = 8F^3. \quad (2.46)$$

À l'aide de ce système de coordonnées, il devient efficace d'additionner et de doubler les points, car il n'y a plus d'opération d'inversion. Il faut en effet $12M + 2C$ pour l'addition et $8M + 5C$ pour le doublement. Une remarque intéressante est que si l'un des deux points P ou Q est donné en coordonnées affines, c'est-à-dire $(X : Y : 1)$, alors l'addition ne demande plus que $9M$ et $2C$.

Coordonnées jacobiennes Le système de coordonnées jacobiennes représente aussi un grand intérêt pour l'efficacité des calculs impliqués dans les couplages. Les coordonnées ont trois composantes, comme pour la représentation projective, et le changement de variable est le suivant :

$$\begin{cases} x &= {}^{X}/_{Z^2} \\ y &= {}^{Y}/_{Z^3}. \end{cases} \quad (2.47)$$

L'équation de E devient :

$$E : Y^2 = X^3 + aXZ^4 + bZ^6. \quad (2.48)$$

C'est-à-dire qu'un point de coordonnées $(X : Y : Z)$ avec $Z \neq 0$ correspond au point $({}^{X}/_{Z^2}, {}^{Y}/_{Z^3})$. Un point qui voit sa coordonnée en Z devenir 0 correspond au point à l'infini. De même que pour la représentation projective, $\mathcal{P}_\infty$ est une ligne et est notée $(1 : 1 : 0)$. Enfin, l'opposé d'un point $P = (X_P : Y_P : Z_P)$ est $-P = (X_P : -Y_P : Z_P)$.

Les formules d'addition et de doublement se retrouvent avec la même méthodologie que précédemment. Elles sont données ci-dessous. Soient $T = (X_T : Y_T : Z_T)$ et $P = (X_P : Y_P : Z_P)$ avec $T \neq \pm P$. Nous posons :

$$\begin{cases} A &= X_T Z_P^2 \\ B &= X_P Z_T^2 \\ C &= Y_T Z_P^3 \\ D &= Y_P Z_T^3 \\ F &= B - A \\ G &= D - C \end{cases}$$

Les formules de X_{T+P}, Y_{T+P} et Z_{T+P} sont donc :

$$X_{T+P} = -F^3 - 2AF^2 + G^2, \quad Y_{T+P} = -CF^3 + G\left(AF^2 - X_{T+P}\right) \quad \text{et} \quad Z_{T+P} = Z_T Z_P F. \tag{2.49}$$

Pour le doublement de $T = (X_T : Y_T : Z_T)$ nous posons :

$$\begin{cases} H &= 4X_T Y_T^2 \\ I &= 3X_T^2 + aZ_T^4 \end{cases}$$

Cela donne les formules suivantes pour doubler le point T :

$$X_{[2]T} = -2H + I^2, \quad Y_{[2]T} = -8Y_T^4 + I\left(H - X_{[2]T}\right) \quad \text{et} \quad Z_{[2]T} = 2Y_T Z_T. \tag{2.50}$$

Dans le cas de l'addition il faut $12M + 4C$ opérations et le doublement est effectué en $4M + 5C$. Tout comme avec les coordonnées projectives, si l'un des deux points est directement donné en coordonnées affines, à savoir $(X : Y : 1)$, alors il ne faut plus que $8M$ et $3C$ pour l'addition.

Degré de plongement

La notion de degré de plongement est utile pour spécifier les courbes elliptiques qui seront utilisées dans les couplages.

Remarque 3. *Pour un entier premier m, les assertions suivantes sont équivalentes :*

i. $\left(\mathbb{F}_q^\star\right)^m \neq \mathbb{F}_q^\star$, *avec la notation* $\left(\mathbb{F}_q^\star\right)^m = \left\{x^m | x \in \mathbb{F}_q^\star\right\}$.

ii. $x \mapsto x^m$ *n'est pas surjective.*

iii. $x \mapsto x^m$ *n'est pas injective (puisque nous sommes dans un corps fini).*

iv. m *divise* $q - 1$.

Si l'une de ces conditions n'est pas atteinte alors on va chercher à se positionner dans une extension $\mathbb{F}_{q^k}$ telle que l'on ait m qui divise $q^k - 1$.

Définition 9. *Soient E une courbe elliptique sur $\mathbb{F}_q$ et m un entier premier avec q. Le degré de plongement (associé à m et q) est le plus petit entier positif k tel que m divise q^{k-1}.*

Les calculs sur une courbe elliptique sur $\mathbb{F}_{q^k}$ ont un coût qui croît avec k. Pour les applications cryptographiques, nous sommes donc intéressés par les courbes $\mathbb{F}_{q^k}$ avec k petit. Les courbes elliptiques supersingulières se comportent très bien dans ce cadre-là, le théorème 6, démontré dans [MOV93] montre que $k \leq 6$.

Théorème 6 ([CFA+05] page 124)**.** *Soit E une courbe elliptique **supersingulière** sur $\mathbb{F}_q$, avec $q = p^d$, p premier. Lorsque la courbe E a un point d'ordre m, avec $m \neq p$ et premier, alors le degré de plongement k associé à m et q vérifie :*

- *si $p = 2$ alors $k \leq 4$,*
- *si $p = 3$ alors $k \leq 6$,*
- *si $p \geq 5$ alors $k \leq 3$; si de plus $d = 1$ alors $k \leq 2$.*

Ces bornes sont toujours atteintes.

Théorème 7 (Balasubramanian-Koblitz [BK98])**.** *Soit E une courbe elliptique sur $\mathbb{F}_q$, avec $q = p^d$, p premier. Soient m un entier premier avec p et $k > 1$ le degré de plongement associé à m et q, alors*

$$E\left(\mathbb{F}_{q^k}\right)[m] = E[m] \simeq \mathbb{Z}/m\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}. \tag{2.51}$$

Courbes adaptées aux couplages

Beaucoup de recherches sont effectuées pour trouver les courbes elliptiques qui fournissent des paramètres permettant d'effectuer les opérations efficacement. C'est-à-dire que l'on cherche à maîtriser le degré de plongement pour qu'il reste petit. Par exemple si on prend une courbe elliptique générée aléatoirement, possédant un sous-groupe d'ordre m, alors il y a de fortes chances pour que $k \approx m$. L'article de journal [FST10] offre une taxinomie complète en 2010.

Les méthodes de construction de courbes elliptiques bien adaptées aux couplages (dites *pairing-friendly elliptic curves* en anglais) partent d'un degré de plongement k donné et fournissent des formules pour q, t et m tels que E sur $\mathbb{F}_q$ soit d'ordre $n = q + 1 - t$ et m divise $q^k - 1$.

Un critère est utilisé pour qualifier les courbes ainsi obtenues. Le paramètre $\rho = \frac{\log q}{\log m}$ mesure la taille du corps de base relativement à la taille du sous-groupe d'ordre premier de la courbe. Obtenir des courbes avec un $\rho \approx 1$ est très recherché.

Courbes de Barreto-Naehrig (BN) Parmi les méthodes de construction, la méthode de Barreto-Naehrig [BN05] fournit des courbes ordinaires dont le degré de plongement est fixé à 12. Nous allons introduire de façon succincte la construction de ces courbes elliptiques. Pour cela nous nous inspirons de [BLS02] et [BN05].

Polynômes cyclotomiques La construction de ces courbes adaptées aux couplages fait intervenir les polynômes cyclotomiques. La table 2.5 rappelle les premiers polynômes cyclotomiques. Pour plus de détails sur les propriétés et la construction de ces polynômes, nous renvoyons à [PCD96] page 80.

Table 2.5 – Les premiers polynômes cyclotomiques

k	Φ_k
1	$x-1$
2	$x+1$
3	x^2+x+1
4	x^2+1
5	$x^4+x^3+x^2+x+1$
6	x^2-x+1
7	$x^6+x^5+x^4+x^3+x^2+x+1$
8	x^4+1
9	x^6+x^3+1
10	$x^4-x^3+x^2-x+1$
11	$x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1$
12	x^4-x^2+1

Construction des courbes BN L'objectif est de construire une courbe elliptique E sur $\mathbb{F}_q$ d'ordre $\#E(\mathbb{F}_q) = n$, la trace de Frobenius est notée t (c'est-à-dire $n = q+1-t$). Alors, pour un $k > 0$ donné on cherche :

$$\begin{cases} q \\ t \text{ tel que } |t| \leq 2\sqrt{q} \text{ (borne de Hasse)} \\ m \text{ tel que } m \text{ divise } q^k - 1 \text{ mais } m \text{ ne divise pas } q^i - 1 \text{ pour } 0 < i < k \\ \text{ et } m \text{ divise } n. \end{cases}$$

Puisque $n = q+1-t$ et que m divise n (c'est-à-dire qu'il existe $h \in \mathbb{Z}$ tel que $n = mh$) on a $q = (t-1)+hm$, que l'on peut écrire $q = t-1 \mod m$. Par conséquent, pour $u > 0$, $q^u = (t-1)^u \mod m$, ainsi

$$q^u - 1 = (t-1)^u - 1 \mod m, \quad u > 0. \tag{2.52}$$

Comme m divise $q^k - 1$, il existe $s \in \mathbb{Z}$ tel que $q^k - 1 = ms$ et donc d'après 2.52, il vient que $(t-1)^k - 1 = ms = 0 \mod m$, mais ce n'est pas le cas lorsque $0 < u < k$. D'où

$$\begin{cases} m \text{ divise } (t-1)^k - 1 \\ \text{et } m \text{ ne divise pas } (t-1)^i - 1 \text{ pour } 0 < i < k. \end{cases} \tag{2.53}$$

Par ailleurs, on connaît la propriété suivante sur les polynômes cyclotomiques[7] :

$$x^u - 1 = \prod_{d \text{ divise } u} \Phi_d(x), \quad u > 0. \tag{2.54}$$

[7] Plus de détails dans [LN97].

Ainsi, les équations 2.53 et 2.54 donnent

$$m \text{ divise } \Phi_k(t-1).$$

Les travaux de Barreto-Naehrig [BN05] consistent à examiner le cas $k = 12$. On sait que le polynôme Φ_{12} est de degré 4 et d'après Hasse (théorème 5) $m \sim q \sim t^2$. Donc $t(x)$ peut être choisi de degré 2, de façon que $m(x)$ soit un facteur de $\Phi_{12}(t-1)$ de degré 4.

Maintenant, on cherche à scinder $\Phi_{12}(u(x))$ pour un polynôme $u(x)$ de degré 2. Les travaux de Galbraith *et al.* [GMV04] montrent qu'il n'y a que deux polynômes qui conviennent :

$$u(x) = 2x^2 \quad \text{ou} \quad \underline{u(x) = 6x^2}.$$

Pour la suite de la construction des courbes BN, le choix est $u(x) = 6x^2$. Ce qui donne $t(x) = u(x) + 1 = 6x^2 + 1$. Par suite

$$\begin{aligned} \Phi_{12}(t(x)-1) &= (6x^2)^4 - (6x^2)^2 + 1 = 1296x^8 - 36x^4 + 1 \\ &= (\underbrace{36x^4 + 36x^3 + 18x^2 + 6x + 1}_{m(x)})(36x^4 - 36x^3 + 18x^2 - 6x + 1). \end{aligned}$$

Il ne reste plus qu'à trouver $q(x)$. Pour rappel, $q = (t-1) + mh$ pour $h \in \mathbb{Z}$. D'après Miyaji *et al.* [MNT01] la valeur du cofacteur h peut être prise à 1. On obtient donc $q(x) = 6x^2 + m(x) = 36x^4 + 36x^3 + 24x^2 + 6x + 1$.

L'autre partie du travail est de trouver l'équation de E reliée aux paramètres qui viennent d'être construits. Atkin *et al.* [AM93] adaptent la méthode de multiplication complexe (CM) utilisée pour les courbes elliptiques définies sur $\mathbb{C}$ au cas des corps finis.

L'idée est d'utiliser le polynôme caractéristique de l'endomorphisme de Frobenius sur $E(\mathbb{F}_q)$ pour trouver des informations sur la courbe. Ce polynôme est $X^2 - tX + q$ et son discriminant est

$$4q - t^2 = DV^2. \tag{2.55}$$

L'entier $-D$ est appelé <u>discriminant fondamental</u>, il ne doit pas être un carré et

- soit $D = 3 \mod 4$,
- soit $D = 4s$ avec $s = 1 \mod 4$ ou $s = 2 \mod 4$.

Dans le cas des courbes BN nous avons :

$$DV^2 = 3(6x^2 + 4x + 1)^2.$$

Avec $D = 3$, il est montré dans [LZ94] et [Mor91] que les courbes produites sont de la forme

$$E : y^2 = x^3 + b, \quad \text{avec } b \neq 0. \tag{2.56}$$

Pour résumer, les paramètres q et m sont calculés à partir de l'équation 2.57 pour un certain x tel que q et m soient premiers.

$$\begin{cases} q(x) &= 36x^4 + 36x^3 + 24x^2 + 6x + 1 \\ m(x) &= 36x^4 + 36x^3 + 18x^2 + 6x + 1 \\ t(x) &= 6x^2 + 1 \\ DV^2 &= 3(6x^2 + 4x + 1)^2 = 108x^4 + 144x^3 + 84x^2 + 24x + 3 \end{cases} \tag{2.57}$$

2.1.5 Fonctions de hachage sur courbe elliptique

Les protocoles cryptographiques basés sur les couplages nécessitent de hacher un message en un point d'une courbe elliptique. Le premier algorithme qui transforme un élément de $\mathbb{F}_{q^d}$ en un point d'une courbe elliptique de façon déterministe et en temps constant a été proposé par Shallue *et al.* [SvdW06]. Il permet de hacher en un temps $\mathcal{O}(\log_2^4 q)$ pour n'importe quel $q = p^d$ avec p premier, et en temps $\mathcal{O}(\log_2^3 q)$ lorsque $q = 3 \mod 4$. Plus tard, Icart [Ica09] dévoile un algorithme déterministe, qui en temps constant $\mathcal{O}(\log_2^3 q)$, permet de hacher sur une courbe elliptique définie sur $\mathbb{F}_q$ avec $q = 2 \mod 3$. Des travaux [FT12, Tib12] plus spécifiques aux courbes BN ont été réalisés pour répondre à cette problématique.

Soit la fonction χ_q définie par :

$$\begin{array}{rccl} \chi_q : & \mathbb{F}_q & \to & \{-1, 0, 1\} \\ & x & \mapsto & \begin{cases} 1 \text{ si } x \text{ est un carré,} \\ -1 \text{ sinon.} \end{cases} \\ & 0 & \mapsto & 0. \end{array} \tag{2.58}$$

Et soient les x_i définis de la sorte :

$$\begin{array}{rcl} x_1 & = & \frac{-1+\sqrt{-3}}{2} - \frac{u^2\sqrt{-3}}{1+b+u^2}, \\ x_2 & = & \frac{-1-\sqrt{-3}}{2} + \frac{u^2\sqrt{-3}}{1+b+u^2}, \\ x_3 & = & \frac{(1+b+u^2)^2}{3u^2}. \end{array} \tag{2.59}$$

Alors le papier [FT12] donne la construction suivante pour une courbe de Barreto-Naehrig $E : y^2 = x^3 + b$ sur $\mathbb{F}_q$ avec $q = 7 \mod 12$:

$$\begin{array}{rccl} f : & \mathbb{F}_q & \to & E(\mathbb{F}_q) \\ & u & \mapsto & \left(x_i, \chi_q(u)\sqrt{x_i^3 + b}\right) \\ & 0 & \mapsto & \left(\frac{-1+\sqrt{-3}}{2}, \sqrt{1+b}\right), \end{array} \tag{2.60}$$

avec $i \in \{1, 2, 3\}$ le plus petit indice tel que $x_i^3 + b$ soit un carré dans $\mathbb{F}_q$.

2.1.6 Couplages

Le concept de diviseurs d'une courbe elliptique est nécessaire à la compréhension de la construction des couplages.

Nous nous plaçons dans le cadre où E est une courbe elliptique sur $\mathbb{F}_q$ avec $q = p^d$, p premier, et m un entier premier avec p.

Diviseurs d'une courbe elliptique et loi de réciprocité de Weil

Cette section sur les diviseurs provient de [Sil09] chapitre II.3. Pour plus de détails sur la clôture algébrique $\overline{\mathbb{F}_q}$ du corps $\mathbb{F}_q$ nous renvoyons à [PCD96] ou autre cours d'algèbre[8].

[8] Par exemple https://webusers.imj-prg.fr/~patrick.polo/M1Galois/ATGch8.pdf.

Définition 10. *Un diviseur D de E est une somme formelle*

$$D = \sum_{P \in E(\overline{\mathbb{F}_q})} n_P(P),$$

avec les $n_P \in \mathbb{Z}$ presque tous nuls.

La notation $n_P(P)$ indique que l'on ne fait pas le calcul de la multiplication scalaire $[n_P]P$, qui donne un point de E. Ici, D est une somme formelle, pas un point de E.

L'ensemble formé par les diviseurs de E est un groupe (additif) abélien libre engendré par les points de la courbe, il est noté $Div_{\overline{\mathbb{F}_q}}(E)$ ([Sil09] page 28). L'élément neutre de ce groupe est le diviseur dont tous les $n_p = 0$, il est noté 0.

Définition 11. *Soit $D = \sum_{P \in E(\overline{\mathbb{F}_q})} n_P(P)$ un diviseur de E. Le degré de D est*

$$\deg(D) = \sum_{P \in E(\overline{\mathbb{F}_q})} n_P.$$

Le support de D est l'ensemble

$$supp(D) = \{P \in E\left(\overline{\mathbb{F}_q}\right) | n_P \neq 0\}.$$

Il est aussi possible de définir le diviseur d'une fonction f définie sur E. On note $\mathrm{ord}_P(f)$ la valuation de la fonction f au point P (c'est-à-dire, la multiplicité de f en P)[9].

Définition 12. *Le diviseur de la fonction f est*

$$div(f) = \sum_{P \in E(\overline{\mathbb{F}_q})} ord_P\,(f)\,(P).$$

Exemple 4. *Soient P et Q deux points de E et $(l = 0)$ l'équation de la droite affine passant par P et Q. Cette droite intersecte E en trois points P, Q et $-(P+Q)$ avec multiplicité 1 et elle intersecte E en $\mathcal{P}_\infty$ avec multiplicité -3, c'est le théorème de Bézout [Per95]. Ainsi le diviseur de l est $div(l) = (P) + (Q) + (-(P+Q)) - 3(\mathcal{P}_\infty)$. La tangente en P intersecte, quant à elle, le point P avec multiplicité (nombre de fois) 2, $-[2]P$ avec multiplicité 1 et $\mathcal{P}_\infty$ avec multiplicité -3. Donc $div(l) = 2(P) + (-[2]P) - 3(\mathcal{P}_\infty)$.*

Il y a quelques propriétés remarquables sur les diviseurs.

Proposition 7. *Soient f et g deux fonctions définies sur E, alors :*

- $div(f) = 0$ ***si et seulement si*** *f est constante.*
- $div(fg) = div(f) + div(g)$.
- $div(f/g) = div(f) - div(g)$.
- $\deg(div(f)) = 0$.

[9] Voir par exemple [Sil09] page 52 pour plus de détails.

Définition 13. *Un diviseur D de E est dit principal s'il est égal au diviseur d'une fonction, c'est-à-dire, si $D = div(f)$.*

Définition 14 ([Sil09] page 39)**.** *Soient f une fonction définie sur E et $D = \sum_{P\in E(\overline{\mathbb{F}_q})} n_p(P)$ un diviseur de E tels que $supp(div(f)) \cap supp(D) = \emptyset$ (c'est-à-dire, $ord_P(f) = 0$ lorsque $n_P \neq 0$), alors il est possible de définir $f(D)$ de la façon suivante :*

$$f(D) = \prod_{P\in E(\overline{\mathbb{F}_q})} f(P)^{n_P} \in \mathbb{F}_q{}^{\star}. \tag{2.61}$$

Proposition 8 (Loi de réciprocité de Weil [Gal05])**.** *Soient f et g deux fonctions définies sur E avec $supp(div(f)) \cap supp(div(g)) = \emptyset$, alors*

$$f(div(g)) = g(div(f)). \tag{2.62}$$

La proposition suivante donne une caractérisation des diviseurs principaux et sera utilisée pour la construction des couplages sur courbes elliptiques.

Proposition 9. *Soit $D = \sum_P n_p(P)$ un diviseur de E, alors*

$$D \text{ est principal } \Leftrightarrow \sum_P n_P = 0 \text{ et } \sum_P [n_P]P = \mathcal{P}_\infty. \tag{2.63}$$

Il est possible de définir une relation d'équivalence sur $Div_{\overline{\mathbb{F}_q}}(E)$:

Définition 15. *Soient D_1 et D_2 deux diviseurs de E, ils sont linéairement équivalents si le diviseur $D_1 - D_2$ est principal, on note alors $D_1 \sim D_2$.*

Autrement dit, $D_1 \sim D_2$ **si et seulement si** il existe une fonction f définie sur E tel que $D_1 - D_2 = \mathrm{div}(f)$.

Couplage de Weil

Soient deux points P et Q de $m-$torsion, deux diviseurs D_P et D_Q tels que $D_P \sim (P) - (\mathcal{P}_\infty)$ et $D_Q \sim (Q) - (\mathcal{P}_\infty)$ avec $\mathrm{supp}(D_P) \cap \mathrm{supp}(D_Q) = \emptyset$. Les points P et Q sont d'ordre m, ainsi la proposition 9 nous assure qu'il existe deux fonctions f_P et f_Q définies sur E telles que $\mathrm{div}(f_{m,P}) = m(P) - m(\mathcal{P}_\infty)$ et $\mathrm{div}(f_{m,Q}) = m(Q) - m(\mathcal{P}_\infty)$. Ces fonctions sont définies à constante près. Enfin, la loi de réciprocité de Weil (cf. proposition 8) assure que $e_m(P,Q) = \frac{f_{m,P}(D_Q)}{f_{m,Q}(D_P)}$ est bien définie. C'est-à-dire que e_m est indépendant du choix des représentants de $D_Q \sim (Q) - (\mathcal{P}_\infty)$ et des constantes choisies pour $f_{m,P}$ et $f_{m,Q}$. La loi de réciprocité de Weil permet aussi d'écrire :

$$\left(\frac{f_{m,P}(D_Q)}{f_{m,Q}(D_P)}\right)^m = \frac{f_{m,P}(mD_Q)}{f_{m,Q}(mD_P)} = \frac{f_{m,P}(\mathrm{div}(f_{m,Q})}{f_{m,Q}(\mathrm{div}(f_{m,P})} = 1. \tag{2.64}$$

Donc $e_m(P,Q) \in \mu_m$.

Définition 16 (Couplage de Weil)**.** *Le couplage de Weil e_m de P et Q est :*

$$\begin{array}{rccl} e_m : & E[m] \times E[m] & \to & \mu_m \\ & (P,Q) & \mapsto & (-1)^m \frac{f_{m,P}(D_Q)}{f_{m,Q}(D_P)}. \end{array} \tag{2.65}$$

Proposition 10 ([Sil09], chapitre III.8.)**.** *Le couplage de Weil $e_m : E[m] \times E[m] \to \mu_m$ satisfait les propriétés suivantes :*

*i. Il est **antisymétrique** : pour tout $P, Q \in E[m]$*

$$e_m(P,Q) = e_m(Q,P)^{-1}.$$

*ii. Il est **bilinéaire** : pour tout $P, P_1, P_2, Q, Q_1, Q_2 \in E[m]$*

$$\begin{aligned} e_m(P, Q_1 + Q_2) &= e_m(P,Q_1)e_m(P,Q_2) \\ e_m(P_1 + P_2, Q) &= e_m(P_1,Q)e_m(P_2,Q). \end{aligned}$$

Ainsi, $\forall a, b \in \mathbb{Z} \quad e_m([a]P, [b]Q) = e_m(P,Q)^{ab}$.

*iii. Il est **non dégénéré** : pour tout $P \in E[m] \setminus \{\mathcal{P}_\infty\}$ il existe $Q \in E[m]$ tel que $e_m(P,Q) \neq 1$.*

Couplage de Tate

On considère deux points $P \in E\left(\mathbb{F}_{q^k}\right)[m]$ et $Q \in E\left(\mathbb{F}_{q^k}\right)$, la proposition 9 assure qu'il existe une fonction $f_{m,P}$ définie sur $E\left(\mathbb{F}_{q^k}\right)$ ainsi qu'un diviseur $D_Q \sim (Q) - (\mathcal{P}_\infty), \quad D_Q \in Div^0_{F_{q^k}}(E)$ tels que $\mathrm{supp}(D_Q) \cap \mathrm{supp}(\mathrm{div}(f_{m,P})) = \emptyset$. La loi de réciprocité de Weil nous permet de bien définir le couplage

$$\begin{aligned} \langle \cdot, \cdot \rangle_m : \quad E\left(\mathbb{F}_{q^k}\right)[m] \times E\left(\mathbb{F}_{q^k}\right) &\to \mathbb{F}^\star_{q^k} / \left(\mathbb{F}^\star_{q^k}\right)^m \\ (P,Q) &\mapsto f_{m,P}(D_Q). \end{aligned} \tag{2.66}$$

La bilinéarité de $\langle \cdot, \cdot \rangle_m$ se montre de la manière suivante : soient $D_{P_1} \sim (P_1) - (\mathcal{P}_\infty)$ et $D_{P_2} \sim (P_2) - (\mathcal{P}_\infty)$ tels que leurs supports soient disjoints de celui de $D_Q \sim (Q) - (\mathcal{P}_\infty)$, et soient f_1, f_2 et f_Q trois fonctions telles que $\mathrm{div}(f_i) = mD_{P_i}$ et $\mathrm{div}(f_Q) = mD_Q$. Par construction, le diviseur $D' = aD_{P_1} + bD_{P_2}$ est linéairement équivalent à $([a]P_1 + [b]P_2) - (\mathcal{P}_\infty)$, le support de D' est disjoint du support de D_Q et $mD' = \mathrm{div}(f')$ avec $f' = f_1^a f_2^b$. D'où

$$\langle [a]P_1 + [b]P_2, Q \rangle_m = f'(D_Q) = \left(f_1^a f_2^b\right)(D_Q) = f_1(D_Q)^a f_2(D_Q)^b = \langle P_1, Q \rangle_m^a \langle P_2, Q \rangle_m^b.$$

Pour montrer la non-dégénérescence, nous devons montrer que $\forall P \in E\left(\mathbb{F}_{q^k}\right)[m]$, il existe $Q \in E\left(\mathbb{F}_{q^k}\right)$ tel que $e(P,Q) \neq 1$. Or, tout Q peut s'écrire comme $[m]Q'$ pour $Q' \in E\left(\mathbb{F}_{q^k}\right)$. Alors, $\langle P, Q \rangle_m = \langle P, [m]Q' \rangle_m = \langle P, Q' \rangle_m^m = 1$, ce couplage est dégénéré.

Le couplage de Tate est modifié pour être non-dégénéré :

Proposition 11 ([Gal05])**.** *Le couplage de Tate est l'application bilinéaire suivante et est bien défini :*

$$\begin{aligned} \tau_m : \quad E\left(\mathbb{F}_{q^k}\right)[m] \times E\left(\mathbb{F}_{q^k}\right)/[m]E\left(\mathbb{F}_{q^k}\right) &\to \mathbb{F}^\star_{q^k} / \left(\mathbb{F}^\star_{q^k}\right)^m \xrightarrow{\sim} \mu_m \subset \mathbb{F}^\star_{q^k} \\ (P,Q) &\mapsto f_{m,P}(D_Q) \mapsto f_{m,P}(D_Q)^{\frac{q^k-1}{m}}. \end{aligned} \tag{2.67}$$

2.2 Implémentation

Étant donné qu'une seule fonction $f_{m,P}$ est évaluée pour le couplage de Tate contre deux pour Weil, qui comprend en plus une inversion, il est aisé de voir que le couplage de Tate est plus rapide que celui de Weil. Cependant, le calcul de $f_{m,P}$ avec $\mathrm{div}(f_{m,P}) = m(P) - m(\mathcal{P}_\infty)$ est longtemps resté l'élément bloquant du développement des couplages pour la cryptographie. Victor Miller propose une solution en 1986 dans [Mil86].

2.2.1 Algorithme de Miller

L'objectif est de construire une fonction $f_{m,P}$ avec $\mathrm{div}(f_{m,P}) = m(P) - m(\mathcal{P}_\infty)$ et qu'elle soit calculable efficacement. L'idée est de construire récursivement une fonction f_i avec $\mathrm{div}(f_i) = i(P) - ([i]P) - (i-1)(\mathcal{P}_\infty)$. Ainsi, dès que $i = m$ on obtient $\mathrm{div}(f_m) = m(P) - \underbrace{([m]P)}_{=\mathcal{P}_\infty} - (m-1)(\mathcal{P}_\infty) = m(P) - m(\mathcal{P}_\infty) = \mathrm{div}(f_{m,P})$, ce qui est recherché.

Ensuite, pour incrémenter i jusqu'à m il faut une relation de récurrence. C'est-à-dire que l'on cherche à calculer f_{i+j} à partir de f_i et f_j. On écrit les diviseurs de f_{i+j} et de $f_i f_j$:

$$\begin{array}{rcl} \mathrm{div}(f_{i+j}) & = & (i+j)(P) - ([i+j]P) - (i+j-1)(\mathcal{P}_\infty) \\ \mathrm{div}(f_i f_j) & = & (i+j)(P) - ([i]P) - ([j]P) - (i+j-2)(\mathcal{P}_\infty). \end{array} \tag{2.68}$$

L'équation 2.68 montre que les expressions de $\mathrm{div}(f_{i+1})$ et de $\mathrm{div}(f_i f_j)$ sont très proches. Pour que ces expressions soient égales, il faut « ajouter » et « enlever » certains termes.

Soit $(l_{i,j} = 0)$ l'équation de la droite passant par $[i]P$ et $[j]P$, comme vu dans l'exemple 4 on a

$$\mathrm{div}(l_{i,j}) = [i]P + [j]P + (-[i+j]P) - 3(\mathcal{P}_\infty). \tag{2.69}$$

Soit $(v_{i+j} = 0)$ l'équation de la ligne verticale passant par $[i+j]P$, on a alors

$$\mathrm{div}(v_{i+j}) = ([i+j]P) + (-[i+j]P) - 2(\mathcal{P}_\infty). \tag{2.70}$$

En prenant les équations 2.69 et 2.70, le diviseur $\mathrm{div}\left(\frac{f_i f_j l_{i,j}}{v_{i+j}}\right)$ s'écrit :

$$\begin{array}{rcl} \mathrm{div}\left(\frac{f_i f_j l_{i,j}}{v_{i+j}}\right) & = & (i+j)P - ([i]P) - ([j]P) - (i+j-2)(\mathcal{P}_\infty) \\ & & +([i]P) + ([j]P) + (-[i+j]P) - 3(\mathcal{P}_\infty) \\ & & -(([i+j]P) + (-[i+j]P) - 2(\mathcal{P}_\infty)) \\ & = & (i+j)(P) - ([i+j]P) - (i+j-1)(\mathcal{P}_\infty) \\ & = & \mathrm{div}(f_i f_j). \end{array} \tag{2.71}$$

En particulier, $f_{i+j} = \frac{f_i f_j l_{i,j}}{v_{i+j}}$ convient. Il suffit d'initialiser la suite avec $f_0 = 1$ et de l'itérer jusqu'à f_m, pour finalement obtenir $f_m = f_{m,P}$. L'algorithme de Miller 2.6 est construit en utilisant le schéma de *double et ajoute* en remarquant :

$$f_{i+1} = f_i \frac{l_{i,1}}{v_{i+1}} \quad \text{et} \quad f_{2i} = f_i^2 \frac{l_{i,i}}{v_{2i}}. \tag{2.72}$$

Entrées : Une courbe elliptique E sur $\mathbb{F}_q$, $m = (m_{\mathfrak{n}-1} \dots m_0)_2$, $P \in E\left(\mathbb{F}_{q^k}\right)[m]$ et $Q \in E\left(\mathbb{F}_{q^k}\right)$.
Sorties : $f_{m,P}(Q)$ avec $\mathrm{div}(f_{m,P}) = m(P) - m(\mathcal{P}_\infty)$.

$f \leftarrow 1$;
$T \leftarrow P$;
pour $i = \mathfrak{n} - 2$ **à 0 faire**
  $l_{T,T} \leftarrow$ tangente en T ;
  $v_{[2]T} \leftarrow$ ligne verticale en $[2]T$;
  $f \leftarrow f^2 \frac{l_{T,T}(Q)}{v_{[2]T}(Q)}$;
  $T \leftarrow [2]T$;
  si $m_i = 1$ **alors**
    $l_{T,P} \leftarrow$ ligne passant par T et P ;
    $v_{T+P} \leftarrow$ ligne verticale en $T + P$;
    $f \leftarrow f \frac{l_{T,P}(Q)}{v_{T+P}(Q)}$;
    $T \leftarrow T + P$;
  fin
fin
retourner f ;

Algorithme 2.6 : Boucle de Miller [Mil86]

L'algorithme qui en ressort est l'algorithme 2.6.

Les fonctions retournées par l'algorithme de Miller 2.6 permettent de calculer les couplages de Weil et de Tate. De plus, la fonction $f_{m,P}$ peut directement être évaluée aux points Q, plutôt que D_Q, avec l'algorithme de Miller. Ces vérifications sont données par la proposition 12.

Proposition 12. *Soient $P \in E\left(\mathbb{F}_{q^k}\right)[m] \setminus \{\mathcal{P}_\infty\}$ et $Q \in E\left(\mathbb{F}_{q^k}\right) \setminus \{\mathcal{P}_\infty, P\}$. Si $f_{m,P}$ satisfait $div(f_{m,P}) = m(P) - m(\mathcal{P}_\infty)$ et $f_{m,P}(\mathcal{P}_\infty) = 1$, par exemple $f_{m,P}$ issu de l'algorithme de Miller 2.6, alors*

$$\tau_m(P,Q) = f_{m,P}(Q)^{\frac{q^k-1}{m}}.$$

Si de plus Q est un point de $m-$torsion et $f_{m,Q}$ satisfait $div(f_{m,Q}) = m(Q) - m(\mathcal{P}_\infty)$ et $f_{m,Q}(\mathcal{P}_\infty) = 1$, alors

$$e_m(P,Q) = (-1)^m \frac{f_{m,P}(Q)}{f_{m,Q}(P)}.$$

Démonstration. (Cas du couplage de Tate) On définit $\mathfrak{t}_Q : R \mapsto R + Q$ la translation par le point. Soit $S \in E(\mathbb{F}_{q^k}^\star) \setminus \{\mathcal{P}_\infty, P, -Q, P-Q\}$, alors $\tau_m(P,Q) = \left(\frac{f_{m,P}(Q+S)}{f_{m,P}(S)}\right)^{\frac{q^k-1}{m}} = \left(\frac{f_{m,P} \circ \mathfrak{t}_Q(S)}{f_{m,P}(S)}\right)^{\frac{q^k-1}{m}}$. Et donc

$$\begin{aligned} \mathrm{div}\left(\frac{f_{m,P}\circ\mathfrak{t}_Q}{f_{m,P}}\right) &= \mathrm{div}(f_{m,P} \circ \mathfrak{t}_Q) - \mathrm{div}(f_{m,P}) \\ &= m(P-Q) - m(-Q) - m(P) + m(\mathcal{P}_\infty). \end{aligned}$$

Par ailleurs, on note $(l = 0)$ l'équation de la droite passant par P et $-Q$, puis $(v = 0)$ l'équation de la ligne verticale en $P - Q$. Ainsi :

$$\begin{cases} \mathrm{div}(l) &= (P) + (-Q) + (-(P-Q)) - 3(\mathcal{P}_\infty) \\ \mathrm{div}(v) &= (P-Q) + (-(P-Q)) - 2(\mathcal{P}_\infty). \end{cases}$$

Ce qui donne $\mathrm{div}\left(\frac{l}{v}\right) = (P)+(-Q)-(P-Q)-(\mathcal{P}_\infty)$. Donc $\mathrm{div}\left(\frac{f_{m,P}\circ \mathfrak{t}_Q}{f_{m,P}}\right) = \left(\mathrm{div}\left(\frac{v}{l}\right)\right)^m$. Par conséquent, il existe une constante $c \in \mathbb{F}_{q^k}$ tel que $\frac{f_{m,P}\circ \mathfrak{t}_Q}{f_{m,P}}(S) = c\left(\frac{v(S)}{l(S)}\right)^m$. D'où

$$\tau_m(P,Q) = \left(c\left(\frac{v(S)}{l(S)}\right)^m\right)^{\frac{q^k-1}{m}} = c^{\frac{q^k-1}{m}}.$$

Il reste à déterminer c. On a :

$$c = \frac{f_{m,P}\circ \mathfrak{t}_Q}{f_{m,P}}(S) \cdot \left(\frac{l(S)}{v(S)}\right)^m,$$

qui est constant quel que soit S. Donc, au point à l'infini on obtient $c = f_{m,P} \circ \mathfrak{t}_Q(\mathcal{P}_\infty) = f_{m,P}(Q)$, car $f_{m,P}(\mathcal{P}_\infty) = \left(\frac{l(\mathcal{P}_\infty)}{v(\mathcal{P}_\infty)}\right)^m = 1$ (on dit que ces fonctions ont des pôles normalisés d'ordre m en $\mathcal{P}_\infty$). □

Remarque 4. *Quelques remarques générales sur l'algorithme de Miller (algorithme 2.6) :*

- *L'algorithme 2.6 peut échouer. C'est le cas lorsque Q annule l ou v (c'est-à-dire $Q = \pm T$) à une certaine itération. Il y a $\mathcal{O}(\log_2(m))$ entrées pour lesquelles l'algorithme échoue. On évite ces cas en prenant $Q \in \mathbb{G}_2 \setminus E(\mathbb{F}_p)$.*
- *Le calcul de $[2]T$ et $T + P$ est fait en même temps que le calcul de l et v.*
- *Les inversions calculées à chaque itération peuvent être repoussées à la fin de l'algorithme. En effet, il est possible de calculer séparément les numérateurs et dénominateurs. C'est-à-dire, remplacer $f \leftarrow f^2 \frac{l_{T,T}(Q)}{v_{[2]T}(Q)}$ par $f_1 \leftarrow l_{T,T}(Q); f_2 \leftarrow v_{[2]T}(Q)$ et renvoyer $\frac{f_1}{f_2}$.*
- *L'ordre du point P peut se vérifier à la fin de l'algorithme de Miller. En effet, l'ordre de P est bien m si à la fin de l'algorithme $T = \mathcal{P}_\infty$.*
- *L'utilisation des coordonnées projectives ou jacobiennes pour le point T permet de réduire le nombre d'inversions (détails dans la section 2.2.2).*
- *La publication de Barreto et al. [BLS04] montre que les lignes verticales $v_{[2]T}(Q)$ et $v_{T+P}(Q)$ prennent leurs valeurs dans un sous-corps strict $\mathbb{F}_{q^{k/\delta}}$ de $\mathbb{F}_{q^k}$ pour $x_Q \in \mathbb{F}_{q^{k/\delta}}$. Ces valeurs seront donc envoyées sur 1 par l'exponentiation finale, il n'y a donc pas besoin de les calculer. De manière générale, des astuces d'optimisations consistent à identifier les calculs qui prennent leurs valeurs dans un sous-corps strict de $\mathbb{F}_{q^k}$ pour ne pas avoir à les exécuter.*

2.2.2 Équations de tangentes et de lignes

Les équations des tangentes et des lignes sur une courbe elliptique sont assez proches des calculs de doublement et addition de points (cf. section 2.1.4). L'utilisation de coordonnées jacobiennes offre une efficacité certaine [NNS10, BGDM+10, CLN10, ALNR11] du fait qu'il n'y a pas d'inversion.

Équations

Pour chacun des systèmes de coordonnées suivants, nous donnons les équations de tangentes et de lignes dans l'annexe B.

- Coordonnées affines.
- Coordonnées projectives.
- Coordonnées mixtes affines-projectives.
- Coordonnées jacobiennes.
- Coordonnées mixtes affines-jacobiennes.

Comparaison La table 2.6 résume les complexités des opérations nécessaires au calcul de l'algorithme de Miller. Cette table est construite en comptant les opérations des formules précédemment énoncées.

Table 2.6 – Complexité des additions, doublements de point, tangentes et lignes selon le système de coordonnées

		M	C	I
Affines	Addition	1	1	1
	Doublement	1	2	1
	Ligne	1	0	1
	Tangente	1	1	1
Projectives	Addition	12	2	0
	Doublement	8	5	0
	Ligne	12	0	1
	Tangente	9	2	1
Affines-Projectives	Addition	9	2	0
	Ligne	6	0	1
	Tangente	4	2	1
Jacobiennes	Addition	12	2	0
	Doublement	8	5	0
	Ligne	18	3	1
	Tangente	11	4	1
Affines-jacobiennes	Addition	8	3	0
	Ligne	8	1	1
	Tangente	6	4	1

De la même façon que l'astuce de Barreto *et al.* [BLS04] à propos de la ligne verticale dans l'algorithme de Miller, les inversions demandées pour calculer les tangentes et lignes peuvent être ignorées. En effet, en pratique on cherchera à prendre des points P et Q satisfaisant certaines propriétés, comme Z_Q dans un sous-corps strict de $\mathbb{F}_{q^k}$.

Calcul combiné addition/ligne et doublement/tangente

Comme mentionné dans la remarque 4, à propos de l'algorithme de Miller, le calcul de la ligne passant par deux points P et T évaluée en Q a la même base que l'addition $T + P$. Il est possible de faire ces calculs en même temps dans le but d'économiser quelques opérations élémentaires. L'annexe C contient les formules concernées.

Comparaison La complexité de chacune des méthodes est résumée dans la table 2.7.

Table 2.7 – Complexité des calculs des additions/lignes et doublements/tangentes simultanés selon le système de coordonnées

		Avec dénominateur			Sans dénominateur		
		M	***C***	***I***	***M***	***C***	***I***
Projectives	Addition/ligne	20	2	1	18	2	0
	Doublement/tangente	14	5	1	13	5	0
Affines-projectives	Addition/ligne	15	2	1	14	2	0
	Doublement/tangente	12	5	1	11	5	0
Jacobiennes	Addition/ligne	21	5	1	19	5	0
	Doublement/tangente	12	6	1	11	6	0
Affines-jacobiennes	Addition/ligne	14	3	1	13	3	0
	Doublement/tangente	8	5	1	7	5	0

Nous pouvons conclure que le système de représentation en coordonnées affines-jacobiennes est le meilleur pour l'implémentation de l'algorithme de Miller.

2.2.3 Exponentiation finale

La sortie de l'algorithme de Miller est $f_{m,P}(Q)$ avec $\mathrm{div}(f_{m,P}) = m(P) - m(\mathcal{P}_\infty)$. Cependant, le couplage de Tate nécessite d'élever cette valeur à la puissance $\frac{q^k-1}{m}$, afin d'obtenir un résultat unique dans $\mu_m \subset \mathbb{F}^\star_{q^k}$.

Une méthode efficace est proposée par Scott *et al.* [SBC+09] lorsque k est pair. En écrivant $k = 2\delta$, les auteurs expriment l'exposant sous la forme :

$$\frac{q^k - 1}{m} = \left(q^\delta - 1\right) \frac{q^\delta + 1}{\Phi_k(q)} \frac{\Phi_k(q)}{m}. \tag{2.73}$$

Le but est de décomposer l'exposant en utilisant le morphisme de Frobenius Ψ_q.

Les trois membres de la décomposition de l'exposant représentent deux exponentiations faciles $\left(q^\delta - 1\right)$ et $\frac{q^\delta+1}{\Phi_k(q)}$, ainsi qu'une difficile $\frac{\Phi_k(q)}{m}$.

Remarque 5. *Soit $a \in \mathbb{F}_{q^k}$, alors $a = \sum_{i=0}^{k-1} a_i \beta^i$, où $a_i \in \mathbb{F}_q$ et $(1, \beta, \beta^2, \ldots, \beta^{k-1})$ est une base de $\mathbb{F}_{q^k}$ vu comme un $k-$espace vectoriel sur $\mathbb{F}_q$. On a alors*

$$a^{q^n} = \sum_{i=0}^{k-1} a_i^{q^n} \beta^{iq^n} = \sum_{i=0}^{k-1} a_i \beta^{iq^n}, \tag{2.74}$$

Table 2.8 – Les premiers $\frac{q^\delta+1}{\Phi_k(q)}$

k	δ	$\frac{q^\delta+1}{\Phi_k(q)}$
2	1	1
4	2	1
6	3	$q+1$
8	4	1
10	5	$q+1$
12	6	q^2+1
14	7	$q+1$
16	8	1
18	9	q^3+1
20	10	q^2+1
22	11	$q+1$
24	12	q^4+1

Remarque 6. *Le polynôme* $\frac{q^\delta+1}{\Phi_k(q)}$ *est un monôme en* q*. La table 2.8 énumère ces polynômes pour* $k = 2, \ldots, 24$*.*

Grâce aux remarques 5 et 6 les exponentiations faciles se résument à l'algorithme 2.7.

Entrées : $f \in \mathbb{F}_{q^k}$.
Sorties : $f^{\left(q^\delta-1\right)\frac{q^\delta+1}{\Phi_k(q)}}$.

1 $f_1 \leftarrow \Psi_\delta(f) \cdot f^{-1}$;
2 $f_2 \leftarrow \Psi_\alpha(f_1) \cdot f_1$; // α tel que $q^\alpha + 1 = \frac{q^\delta+1}{\Phi_k(q)}$ (cf. table 2.8)
3 **retourner** f_2 ;

Algorithme 2.7 : Exponentiations faciles (Scott *et al.* [SBC+09])

La partie difficile de l'exponentiation est l'élévation à la puissance $\frac{\Phi_k(q)}{m}$. Il est usuel de décomposer cet exposant en base q, c'est-à-dire $\frac{\Phi_k(q)}{m} = \lambda_{\varphi(k)-1}q^{\varphi(k)-1} + \cdots + \lambda_1 q + \lambda_0$. L'exponentiation de f_2 devient

$$f_2^{\lambda_{\varphi(k)-1}q^{\varphi(k)-1}} \cdots f_2^{\lambda_1 q} f_2^{\lambda_0} = \left(f_2^{q^{\varphi(k)-1}}\right)^{\lambda_{\varphi(k)-1}} \cdots (f_2^q)^{\lambda_1} f_2^{\lambda_0}.$$

Les $f_2^{q^i}$ sont à nouveaux réalisées par l'application de Frobenius. Le reste de la partie difficile utilise la multi-exponentiation rapide [MVOV96, HDP05, GPS06].

2.2.4 Les couplages optimaux

De nombreuses recherches ont été produites ces dix dernières années sur l'optimisation des algorithmes de couplages. Cependant, trois méthodes notables consistent à l'amélioration du couplage de Tate : Ate, Twisted Ate et Optimal Ate.

Couplage de Ate

Le couplage de Ate [Ver06, HSV06] a pour but de réduire le nombre d'itérations dans l'algorithme de Miller. Soit E une courbe elliptique sur $\mathbb{F}_q$ avec $q = p^d$, où p premier.

Un entier m premier avec q et k le degré de plongement associé à q et m. De plus $t = q + 1 - \#E(\mathbb{F}_q)$ désigne la trace de Frobenius. Le morphisme de Frobenius sur E est défini par

$$\begin{array}{rccc} \Psi_q : & E\left(\mathbb{F}_{q^k}\right) & \to & E\left(\mathbb{F}_{q^k}\right) \\ & P = (x_P, y_P) & \mapsto & (x_P^q, y_P^q)\,. \end{array}$$

Le morphisme de Frobenius Ψ_q a deux valeurs propres sur $E\left(\mathbb{F}_{q^k}\right)[m]$ qui sont 1 et q. Les vecteurs propres associés donnent des générateurs pour les groupes $\mathbb{Z}/m\mathbb{Z}$ du théorème de Balasubramanian-Koblitz 7. Plus précisément :

$$E\left(\mathbb{F}_{q^k}\right)[m] =< P > \times < Q >,$$

pour P et Q tels que $\Psi_q(P) = P$ et $\Psi_q(Q) = [q]Q$.

Le couplage de Ate est construit en utilisant les groupes $\mathbb{G}_1$ et $\mathbb{G}_2$ définis par l'intermédiaire des espaces propres du Frobenius :

$$\left\{ \begin{array}{rcl} \mathbb{G}_1 & = & E[m] \cap \ker(\Psi_q - [1]) = E\left(\mathbb{F}_q\right)[m] \\ \mathbb{G}_2 & = & E[m] \cap \ker(\Psi_q - [q]) \subset E\left(\mathbb{F}_{q^k}\right)[m] \\ \mathbb{G}_T & = & \mu_m \subset \mathbb{F}_{q^k}^\star. \end{array} \right.$$

Le couplage de Ate est alors l'application

$$\begin{array}{rccc} e_{m,A} : & \mathbb{G}_2 \times \mathbb{G}_1 & \to & \mathbb{G}_T \\ & (Q, P) & \mapsto & (f_{t-1,Q}(P))^{\frac{q^k-1}{m}}\,, \end{array} \tag{2.75}$$

avec la fonction $f_{t-1,Q}$ telle que $\mathrm{div}(f_{t-1,Q}) = (t-1)(Q) - ([t-1]Q) - (t-2)(\mathcal{P}_\infty)$. D'après le théorème de Hasse (théorème 5) $m \sim t^2$, ce qui se traduit par le fait que la longueur de t est environ deux fois plus courte de celle de m. Ainsi, l'algorithme de Miller pour le couplage de Ate nécessite deux fois moins d'itérations que pour le couplage de Tate.

Enfin, le couplage de Tate τ_m et de Ate $e_{m,A}$ sont liés par la relation

$$\tau_m(Q, P)^L = e_{m,A}(Q, P)^{c/N}, \tag{2.76}$$

avec $N = \mathrm{pgcd}\left((t-1)^k - 1, q^k - 1\right)$, puis L tel que $(t-1)^k - 1 = LN$ et $c = \sum_{i=0}^{k-1}(t-1)^{k-1-i}q^i = kq^{k-1} \mod m$ si m ne divise pas L (cf. [HSV06] théorème 1).

Couplage Twisted Ate

Le coût de calcul de l'algorithme de Miller dépend fortement de k. En effet, certains calculs sont effectués dans $\mathbb{F}_{q^k}$. Il est possible d'amortir cet effet en utilisant une tordue de E sur un sous-corps de $\mathbb{F}_{q^k}$. C'est le but du couplage Twisted Ate [Ver06, HSV06, MKHO07]

Soit E_2 une courbe tordue de degré δ de E sur $\mathbb{F}_{q^h}$ avec $h = k/\delta$. On note ϕ l'isomorphisme tel que $\phi^{-1} : E_2\left(\mathbb{F}_{q^\delta}\right)[m] \xrightarrow{\sim} E\left(\mathbb{F}_{q^k}\right)[m]$.

Les points de $\mathbb{G}_2$ vont maintenant être pris sur la tordue E_2. C'est-à-dire

$$\mathbb{G}_2 = E[m] \cap \ker([\zeta]\Psi_q^h - [q]),$$

avec $\zeta \in \mathbb{F}_{q^h}$ tel que le polynôme $X^\delta - \zeta$ soit irréductible dans $\mathbb{F}_{q^h}$. L'application $[\zeta]$ est définie par

$$\begin{array}{rccc} [\zeta]: & E_2 & \to & E \\ & (x,y) & \mapsto & (\zeta^2 x, \zeta^3 y)\,. \end{array} \tag{2.77}$$

Le couplage Twisted Ate est l'application

$$\begin{array}{rccc} e_{m,\widetilde{A}}: & \mathbb{G}_1 \times \mathbb{G}_2 & \to & \mathbb{G}_T \\ & (P,Q) & \mapsto & \left(f_{(t-1)^h,P}(Q)\right)^{\frac{q^k-1}{m}}\,. \end{array} \tag{2.78}$$

Contrairement au couplage de Ate, certains calculs sont maintenant effectués dans $\mathbb{F}_{q^{k/\delta}}$. Ce couplage comporte moins d'itérations que le couplage de Tate **si et seulement si** $h\log_2(t-1) \leq \log_2(m)$.

Couplage Optimal Ate

D'après [Ver08, Ver10] un couplage est dit optimal s'il est calculable en $\frac{\log_2(m)}{\varphi(k)} + \epsilon(k)$ itérations de l'algorithme de Miller, où $\epsilon(k) \leq \log_2(k)$. L'astuce vient du fait qu'un couplage n'est pas forcément calculé en une seule évaluation de $f_{\lambda,Q}$, mais peut être composé de produits et de fractions de $f_{\lambda_i,Q}$. L'idée est de décomposer un multiple de m dans une base de puissance de morphisme de Frobenius Ψ_q^i pour $i = 0,\ldots,k-1$. Cette méthode de décomposition est détaillée dans [Ver10].

Dans le cas de l'utilisation des courbes de Barreto-Naehrig [BN05] (vue au paragraphe 2.1.4, page 37) la décomposition donne le couplage Optimal Ate :

$$\begin{array}{rccc} e_{m,OA}: & \mathbb{G}_2 \times \mathbb{G}_1 & \to & \mathbb{G}_T \\ & (Q,P) & \mapsto & (f_{\lambda,Q}(P)M)^{\frac{q^{12}-1}{m}}\,, \end{array}$$

avec $\lambda = 6x+2$, $M = l_{Q_3,-Q_2}(P) l_{-Q_2+Q_3,Q_1}(P) l_{Q_1-Q_2+Q_3,[\lambda]Q}(P)$ où $Q_i = \Psi_q^i(Q) = Q^{q^i}$ pour $i = 1,2,3$ et

$$\left\{\begin{array}{rcl} \mathbb{G}_1 & = & E[m] \cap \ker(\Psi_q - [1]) = E\left(\mathbb{F}_q\right)[m] \\ \mathbb{G}_2 & = & E[m] \cap \ker(\Psi_q - [q]) \subset E\left(\mathbb{F}_{q^{12}}\right)[m] \\ \mathbb{G}_T & = & \mu_m \subset \mathbb{F}_{q^{12}}^\star. \end{array}\right.$$

Pour rappel, $(l_{T_1,T_2} = 0)$ est l'équation de la ligne passant par T_1 et T_2. L'algorithme de Miller, adapté avec cette optimisation (algorithme 2.8), qui en résulte est très efficace [BGDM+10, NNS10].

Entrées : Une courbe elliptique E sur $\mathbb{F}_q$, sa trace de Frobenius t ainsi que deux points $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$.
Sorties : $e_{m,OA}(Q, P)$.

Écrire $s = 6t + 2$ comme $s = \sum_{i=0}^{\widehat{\mathfrak{n}}-1} s_i 2^i$ avec $s_i \in \{-1, 0, 1\}$;
$f \leftarrow 1$; $T \leftarrow Q$;
pour $i = \widehat{\mathfrak{n}} - 2$ **à 0 faire**
  $l_{T,T} \leftarrow$ tangente en T ;
  $f \leftarrow f^2 l_{T,T}(P)$;
  $T \leftarrow [2]T$;
  si $s_i = -1$ **alors**
    $l_{T,-Q} \leftarrow$ ligne passant par T et $-Q$;
    $f \leftarrow f l_{T,-Q}(P)$;
    $T \leftarrow T - Q$;
  sinon si $s_i = 1$ **alors**
    $l_{T,Q} \leftarrow$ ligne passant par T et Q ;
    $f \leftarrow f l_{T,Q}(P)$;
    $T \leftarrow T + Q$;
  fin
fin
$Q_1 \leftarrow \Psi_q(Q)$; $Q_2 \leftarrow \Psi_q^2(Q)$;
$f \leftarrow f l_{T,Q_1}(P)$; $T \leftarrow T + Q_1$;
$f \leftarrow f l_{T,-Q_2}(P)$; $T \leftarrow T - Q_2$;
$f \leftarrow f^{p^6-1}$; $f \leftarrow f^{p^2+1}$;
$f \leftarrow f^{(p^4-p^2+1)/m}$;
retourner f ;

Algorithme 2.8 : Couplage Optimal Ate sur courbes BN [BGDM+10]

2.3 Protocoles à base de couplage

Les applications des couplages en cryptologie sont apparues au début des années 2000. D'une part, Boneh *et al.* [BF01] ont proposé un schéma à bases de couplages pour répondre au défi de Shamir [Sha84] d'établir un protocole de chiffrement basé sur l'identité. D'autre part, Joux [Jou00] a établi un schéma d'échange de clefs à trois entités.

Dans cette section, les groupes manipulés sont des groupes de points d'une courbe elliptique, ainsi la notation $[a]P$ est la multiplication scalaire usuelle de $P \in \mathbb{G}_1$, les groupes $\mathbb{G}_1, \mathbb{G}_2$ et $\mathbb{G}_T$ sont d'ordre premier m. On conserve les notations multiplicatives dans $\mathbb{G}_T$.

2.3.1 Problèmes du logarithme discret, Diffie–Hellman et bilinéarité

Dans cette section, nous allons voir sur quels problèmes mathématiques se basent les protocoles de chiffrement reposant sur les couplages.

Définition 17 (Problème Computational Diffie–Hellman (CDH) dans $\mathbb{G}_1$)**.** *Le problème CDH est :*

- *Données :* $(P, [a]P, [b]P)$ *avec* $P \in \mathbb{G}_1$ *et des entiers non nuls* a *et* b.
- *Problématique : calculer* $[ab]P$ *à partir de* $(P, [a]P, [b]P)$.

Définition 18 (Problème Decisional Diffie–Hellman (DDH) dans $\mathbb{G}_1$)**.** *Le problème DDH est :*

- *Données :* $(P, [a]P, [b]P, [c]P)$ *avec* $P \in \mathbb{G}_1$ *et des entiers non nuls* a, b *et* c.
- *Problématique : répondre oui si* $c = ab \mod q$ *et non sinon.*

Le problème DDH dans $\mathbb{G}_1$ est facile, il peut être résolu en temps polynomial en vérifiant $e([a]P, [b]P) = e(P, [c]P)$. Il s'agit de la réduction MOV [MOV93].

Définition 19. *Soit* $\mathbb{G}_1$ *un groupe d'ordre premier ; il est dit groupe Gap Diffie–Hellman (GDH) s'il existe un algorithme qui, en temps polynomial, résout le problème DDH dans* $\mathbb{G}_1$ *et s'il n'y a pas d'algorithme probabiliste qui, en temps polynomial, résout le problème CDH.*

L'existence d'un tel groupe est importante pour la sécurité des signatures, ils ont été introduits par [OP01].

Définition 20 (Problème Computational Bilinear Diffie–Hellman (CBDH))**.** *Soit* $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, *avec* $\mathbb{G}_1$ *et* $\mathbb{G}_2$ *d'ordre* m, *un couplage. Le problème CBDH est :*

- *Données :* $(P, [a]P, [b]P, [c]P)$ *avec* $P \in \mathbb{G}_1$ *et des entiers non nuls* a, b *et* c.
- *Problématique : calculer* $e(P, P)^{abc}$.

On en vient à la description du problème d'inversion des couplages. En effet, la sécurité des protocoles que nous allons décrire dans les lignes qui suivent repose sur ce problème[10].

Définition 21 (Problème Fixed Argument Pairing Inversion 1 (FAPI1))**.** *Soit* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, *avec* $\mathbb{G}_1, \mathbb{G}_2$ *et* $\mathbb{G}_T$ *d'ordre* m, *un couplage. Le problème FAPI1 est :*

- *Données :* (P, z) *avec* $P \in \mathbb{G}_1$ *et* $z \in \mathbb{G}_T$.
- *Problématique : calculer* $Q \in \mathbb{G}_2$ *tel que* $e(P, Q) = z$.

Le problème FAPI2 est évidemment le symétrique :

- Données : (Q, z) avec $Q \in \mathbb{G}_2$ et $z \in \mathbb{G}_T$.
- Problématique : calculer $P \in \mathbb{G}_1$ tel que $e(P, Q) = z$.

Définition 22 (Problème Decisional Bilinear Diffie–Hellman (DBDH))**.** *Le problème DBDH est :*

- *Données :* $(P, [a]P, [b]P, [c]P, e(P, P)^{abc}, e(P, P)^z)$ *avec* $P \in \mathbb{G}_1$ *et des entiers non nuls* a, b, c *et* z.
- *Problématique : répondre oui si* $z = abc \mod q$ *et non sinon.*

Définition 23 (Problème Modified Decisional Bilinear Diffie–Hellman (MDBDH))**.** *Le problème MDBDH est :*

- *Données :* $(P, [a]P, [b]P, [c]P, e(P, P)^{abc}, e(P, P)^z)$ *avec* $P \in \mathbb{G}_1$ *et des entiers non nuls* a, b, c *et* z.
- *Problématique : répondre oui si* $z = \frac{ab}{c} \mod q$ *et non sinon.*

[10] Plus de détails sur les problèmes d'inversion des couplages dans [GHV08] par exemple.

2.3.2 Échange triparti

En 2001, Joux [Jou00] propose un schéma de mise en accord de clef à trois entités, un protocole Diffie–Hellman triparti en un tour. Le protocole est très simple, il suffit d'un couplage symétrique $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. $\mathbb{G}_1$ est un groupe cyclique de points d'une courbe elliptique E, et P est un point de E. Les trois entités Alice, Bob et Charlie tirent au hasard, respectivement trois entiers a, b et c modulo la caractéristique de $\mathbb{G}_1$. Chaque individu calcule sa clef publique $[a]P, [b]P$ et $[c]P$ et la diffuse. Par exemple, Alice qui a maintenant $a, [b]P$ et $[c]P$ peut calculer $K = e([b]P, [c]P)^a$. K est la clef partagée, en effet

$$K = e([b]P, [c]P)^a = e([a]P, [c]P)^b = e([b]P, [a]P)^c = e(P, P)^{abc}.$$

La figure 2.2 reprend le fonctionnement du protocole.

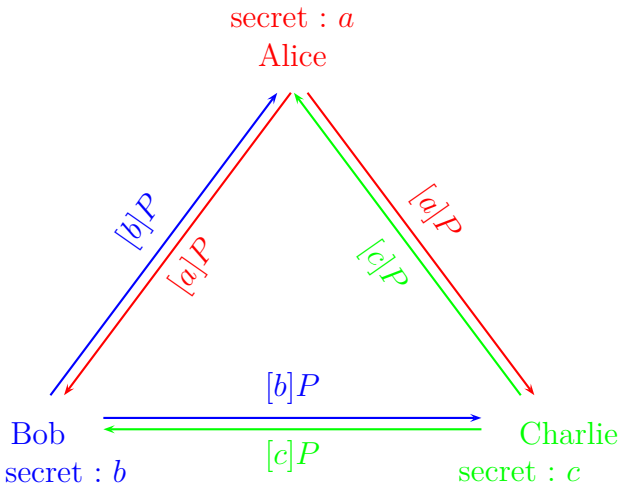


Figure 2.2 – Échange triparti en un tour

2.3.3 Chiffrement basé sur l'identité

Il a fallu attendre le début des années 2000 pour qu'un schéma de chiffrement basé sur l'identité réunissant sécurité et applicabilité soit proposé. Les propositions viennent de Cocks [Coc01] ainsi que Boneh *et al.* [BF01].

Desmedt *et al.* [DB04] font état des avantages et désavantages de l'utilisation de protocoles basés sur l'identité. Nous présentons le protocole de Boneh et Franklin, car il repose sur les calculs de couplage.

Dans un schéma IBE la clef publique **est** l'identité de l'entité. Ainsi, la clef privée associée ne peut pas être calculée par cette entité, mais doit être générée par le PKG. Le déchiffrement doit bien sûr être possible seulement si la clef privée est connue de l'utilisateur. Une version du schéma IBE de Boneh-Franklin [BF01] consiste en 4 étapes :

1. Configuration.
2. Génération de clefs.
3. Chiffrement.
4. Déchiffrement.

Configuration

Le PKG génère les paramètres communs pour les calculs de couplages. Il choisit e et $\mathbb{G}_1, \mathbb{G}_2$ deux groupes d'ordre m tel que $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ soit un couplage. Il choisit $P \in \mathbb{G}_1$ un générateur de $\mathbb{G}_1$. Il choisit deux fonctions de hachages, $H_1 : \{0,1\}^\star \to \mathbb{G}_1^\star$ et $H_2 : \mathbb{G}_2 \to \{0,1\}^n$. Le PKG tire au hasard un $\boldsymbol{s} \in \mathbb{Z}_m$, sa clef privée (appelée clef maître), et calcule $P_{PUB} = [\boldsymbol{s}]P$ la clef publique globale.

L'ensemble des paramètres publics est donc $\{m, n, \mathbb{G}_1, \mathbb{G}_2, e, P, P_{PUB}, H_1, H_2\}$.

Génération de clefs

L'algorithme de génération de clef donne à l'utilisateur sa clef privée. Soit Bob un utilisateur avec l'identité $ID =$ « Bob » $\in \{0,1\}^\star$, le PKG calcule le point associé $Q_B = H_1(ID)$ et la clef privée associée $d_B = [\boldsymbol{s}]Q_B$.

Chiffrement

Alice veut écrire un message $M \in \{0,1\}^n$ à Bob, elle utilise l'algorithme de chiffrement suivant :

1. Elle calcule $Q_B = H_1($« Bob »$)$.
2. Elle tire aléatoirement un nonce[11] r.
3. Elle calcule $g_B = e(Q_B, P_{PUB}) \in \mathbb{G}_2^\star$.
4. Enfin, elle calcule le chiffré $C = \{[r]P, M \oplus H_2(g_B^r)\}$ et l'envoie à Bob.

Déchiffrement

Bob veut déchiffrer le chiffré $C = \{U, V\}$ avec $U \in \mathbb{G}_1, V \in \{0,1\}^n$, il utilise l'algorithme de déchiffrement :

1. Il calcule $K = e(d_B, U)$.
2. Il obtient le message $M = V \oplus H_2(K)$.

Hypothèses de sécurité et exactitude

La sécurité du schéma repose sur le problème CBDH. Le schéma fonctionne, car Bob calcule

$$K = e(d_B, U) = e([s]Q_B, [r]P) = e(Q_B, P)^{sr} = e(Q_B, [s]P)^r = e(Q_B, P_{PUB})^r = g_B^r.$$

2.3.4 Signature courte

Les signatures numériques permettent d'assurer l'intégrité d'un document, ainsi que d'authentifier l'auteur puisqu'il signe avec sa clef privée dont lui seul a la connaissance. La signature numérique DSA est basée sur l'algorithme de chiffrement RSA. Pour des paramètres RSA de 1024 bits, la signature DSA sera une empreinte de 1024 bits. La dérivation sur courbe elliptique de ce système est l'ECDSA [JMV01], qui, pour un

[11] Un nonce est un nombre destiné à être utilisé une seule fois.

même niveau de sécurité, fournit une empreinte de 320 bits. La signature proposée par Boneh *et al.* [BLS01] donnera une empreinte de 160 bits[12].

Configuration

Le système de paramètres publics contient e et $\mathbb{G}_1, \mathbb{G}_2$ deux groupes d'ordre m tel que $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ soit un couplage. $P \in \mathbb{G}_1$ un générateur de $\mathbb{G}_1$ et une fonction de hachage $H : \{0,1\}^\star \to \mathbb{G}_1^\star$.

Signature

Alice signe un message $M \in \{0,1\}^n$ avec sa clef secrète a par

- $S \leftarrow [a]H(M)$.

Vérification de la signature

Pour que Bob puisse vérifier que la signature S est bien la signature du message M par Alice, il se sert de la clef publique d'Alice $P_A = [a]P$.

- $u \leftarrow e(P, S)$.
- $v \leftarrow e(P_A, H(M))$.
- Si $u = v$ alors la signature est acceptée, sinon elle est rejetée.

Hypothèses de sécurité et exactitude

Le schéma de signature fonctionne sous l'hypothèse de l'existence d'un groupe GDH. La sécurité repose sur le problème CDH dans $\mathbb{G}_1$.

Le schéma fonctionne, car

$$v = e(P_A, H(M)) = e([a]P, H(M)) = e(P, H(M))^a = e(P, [a]H(M)) = e(P, S) = u.$$

2.3.5 Chiffrement par attributs

Le chiffrement par attribut est initialement proposé par Sahai *et al.* [SW05]. Dans l'IBE, l'identité sert de clef publique et la clef privée en est dérivée. Pour l'Atribute-Based Encription (ABE), une entité sera capable de déchiffrer des messages si elle a les attributs requis. Ci-dessous, nous décrivons le schéma de base proposé dans [SW05].

On note $\mathcal{U}$ l'univers des attributs. Une entité est un ensemble $\omega \subseteq \mathcal{U}$. On note ϵ la tolérance d'erreur, c'est-à-dire, pour qu'une entité ω' souhaite déchiffrer un message chiffré par ω elle doit satisfaire $\#(\omega \cap \omega') \geq \epsilon$. Nous allons aussi avoir besoin des coefficients de Lagrange[13], pour $i \in \mathbb{Z}_m$ et $S \subset \mathbb{Z}_m$ on note $l_{i,S}(x)$ le polynôme :

$$l_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x - j}{i - j}. \tag{2.79}$$

[12] Ces longueurs sont données à titre indicatif à partir des résultats datant de 2004 issus de [DBS04].

[13] Dans l'article d'origine [SW05], les coefficients de Lagrange sont utilisés pour une interpolation lagrangienne, ces coefficients peuvent être de simples symboles de Kronecker.

Configuration

Le PKG choisit e et $\mathbb{G}_1, \mathbb{G}_2$ deux groupes d'ordre m tel que $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ soit un couplage. Il choisit $P \in \mathbb{G}_1$ un générateur de $\mathbb{G}_1$. Il choisit une fonction de hachage, $H : \{0,1\}^\star \to \mathbb{G}_2^\star$. Le PKG tire au hasard $t_1, \ldots, t_{\#\mathcal{U}}, \alpha \in \mathbb{Z}_r$, sa clef privée (appelée clef maître), et calcule $T_1 = [t_1]P, \ldots, T_{\#U} = [t_{\#\mathcal{U}}]P$ et $Y = e(P,P)^\alpha$.

L'ensemble des paramètres publics est donc $\{m, \mathbb{G}_1, \mathbb{G}_2, e, P, T_1, \ldots, T_{\#\mathcal{U}}, H, Y\}$.

Génération de clefs

L'algorithme de génération de clef donne à l'utilisateur sa clef privée. Soit $\omega \subseteq \mathcal{U}$ les attributs de l'utilisateur, le PKG tire aléatoirement un polynôme Υ de degré $\epsilon - 1$ tel que $\Upsilon(0) = \alpha$. La clef privée associée à ω est $(D_i)_{i\in\omega}$ où $D_i = \left[\frac{\Upsilon(i)}{t_i}\right] P$ pour chaque $i \in \omega$.

Chiffrement

Le chiffrement de $M \in \{0,1\}^\star$ par une entité ω' se fait de la manière suivante.

- Le calcul du haché $H(M) \in \mathbb{G}_2^\star$.
- Le tirage d'un nonce $r \in \mathbb{Z}_m$.
- Le chiffré est $C = \{\omega', MY^r, \{[r]T_i\}_{i\in\omega'}\}$.

Notons que la taille du chiffré croît avec le nombre d'attributs et que l'identité de l'expéditeur ω' est incluse dans le chiffré.

Déchiffrement

L'entité ω, qui satisfait $\#(\omega \cap \omega') \geq \epsilon$, a reçu le chiffré $C = \{\omega', E', \{E_i\}_{i\in\omega'}\}$ de la part de ω'.

1. Il choisit arbitrairement ϵ éléments de $\omega \cap \omega'$ et forme un ensemble S.
2. Il obtient le message
$$M = \frac{E'}{\prod_{i\in S} e(D_i, E_i)^{l_{i,S}(0)}}. \tag{2.80}$$

Hypothèses de sécurité et exactitude

La sécurité repose sur le problème MDBDH.

L'entité qui déchiffre le message calcule $\frac{E'}{\prod_{i\in S} e(D_i,E_i)^{l_{i,S}(0)}}$. Or

$$\begin{aligned}\prod_{i\in S} e(D_i, E_i)^{l_{i,S}(0)} &= \prod_{i\in S} e\left(\left[\tfrac{\Upsilon(i)}{t_i}\right] P, [r]T_i\right)^{l_{i,S}(0)} = \prod_{i\in S} e\left(\left[\tfrac{\Upsilon(i)}{t_i}\right] P, [rt_i]P\right)^{l_{i,S}(0)} \\ &= \prod_{i\in S} e(P,P)^{r\Upsilon(i)l_{i,S}(0)} = \left(\prod_{i\in S} e(P,P)^{\Upsilon(i)l_{i,S}(0)}\right)^r \\ &= e\left(P, \left[\textstyle\sum_{i\in S} \Upsilon(i)l_{i,S}(0)\right] P\right)^r.\end{aligned}$$

Il reste à montrer que $\sum_{i\in S} \Upsilon(i)l_{i,S}(0) = \alpha$, car $\frac{E'}{e(P,[\alpha]P)^r} = \frac{Me(P,P)^{r\alpha}}{e(P,P)^{r\alpha}} = M$.

On note $L(x) = \sum_{i \in S} \Upsilon(i) l_{i,S}(x)$. Les coefficients de Lagrange $l_{i,S}(x)$ ont une forme particulière pour des $x = k \in S$, en effet, ils sont de simples Kronecker :

$$l_{i,S}(k) = \delta_{i,k} = \begin{cases} 1 \text{ si } i = k \\ 0 \text{ sinon} \end{cases}.$$

Donc, pour un $k \in S$ on a $L(k) = \sum_{i \in S} \Upsilon(i)\delta_{i,k}$. Pour conclure,

$$L(k) = \sum_{i \in S} \Upsilon(i)\delta_{i,k} = \overbrace{\Upsilon(i)\underbrace{\delta_{i,k}}_{=1}}^{i=k} + \overbrace{\sum_{i \in S, i \neq k} \Upsilon(i)\underbrace{\delta_{i,k}}_{=0}}^{i \neq k} = \Upsilon(k).$$

D'où $L(0) = \Upsilon(0) = \alpha$.

2.4 Sécurité des protocoles à base de couplages

La sécurité des protocoles que l'on vient d'introduire repose sur la difficulté de résoudre le problème CBDH. Ce problème se résout facilement si l'on est capable de résoudre le DLOG.

Définition 24 (Problème DLOG dans $\mathbb{G}_T$)**.** *Le problème DLOG est :*

- *Données : (g, g^a) avec $g \in \mathbb{G}_T$ et entier non nuls a.*
- *Problématique : renvoyer a.*

De nombreuses recherches sont faites sur la résolution du problème DLOG. D'une part, les couplages basés sur des courbes elliptiques définies sur des corps finis en petite caractéristique sont devenus obsolètes. En effet, les travaux de Barbulescu *et al.* [BGJT14] montrent que le DLOG se résout en temps quasi-polynomial. Dans le cas de l'utilisation des corps premiers, c'est l'algorithme du crible de corps de nombres (Number Field Sieve (NFS) en anglais) qui est le plus efficace. De plus, c'est un algorithme qui a été amélioré ces dernières années [BGK14, KB16]

Par ailleurs, une partie difficile des travaux des cryptanalystes consiste à relier la complexité des algorithmes NFS et de ses dérivations à la taille des clefs utilisées pour la cryptographie. La taille des clefs doit se traduire par la taille des paramètres des courbes elliptiques. Les travaux de Menezes *et al.* [MSS16] apportent des éléments de réponse. Pour le cas des courbes BN sur le corps $\mathbb{F}_q$, ils relatent qu'un niveau de sécurité de 128 bits est obtenu pour q un nombre premier de 383 bits. Plus récemment encore, Barbulescu *et al.* [BD17] concluent que le niveau de sécurité 128 bits pour des courbes BN est obtenu avec une taille pour q de 456 bits.

La sécurité des protocoles à bases de couplages sur les courbes BN avec une taille de q de 456 bits est aujourd'hui assurée. Il faut cependant suivre avec attention s'il y a de prochaines évolutions.

Chapitre 3

Attaques par canaux auxiliaires

« Les grandes choses peuvent se manifester par de petits signes. »

Sigmund Freud

Sommaire

L'objectif de ce chapitre est d'introduire les attaques par canaux auxiliaires contre les implémentations des algorithmes de couplages. La dangerosité de ces attaques est en grande partie due à leurs faibles coûts de mise en œuvre.

Ce chapitre est organisé de la manière suivante. Nous commençons par évoquer les attaques par observations de façon générale. Après être revenus sur l'origine des attaques par canaux auxiliaires en section 3.1, nous présentons en section 3.2 les menaces qu'elles représentent contre l'implémentation des algorithmes cryptographique. La section 3.3.1 nous permet d'introduire les résultats obtenus depuis une dizaine d'années sur les attaques par canaux auxiliaires contre les couplages. Les contre-mesures de la littérature pour protéger les implémentations de couplages sont présentées en section 3.3.2. Enfin, en section 3.4 nous positionnons la thèse et définissons les objectifs par rapport à l'état de l'art que nous aurons alors présenté.

3.1 Généralités sur les attaques par observations

Les circuits intégrés sont à la base du monde numérique. Ils sont donc présents dans les appareils électroniques communicants. Un circuit intégré est constitué d'un ensemble de composants électroniques. Le composant de base est le transistor. Le transistor utilisé aujourd'hui est le Complementary Metal-Oxide-Semiconductor (CMOS) (voir par exemple [Sko01]). Cette technologie est à la base des microprocesseurs, des circuits intégrés dédiés à des applications spécifiques (Application-Specific Integrated Circuit (ASIC)) et des circuits logiques programmables (par exemple les *réseaux de portes programmables in situ* Field-Programmable Gate Array (FPGA)).

3.1.1 Fuites par canaux auxiliaires

Les transistors permettent de réaliser des opérations logiques élémentaires. La réunion de ces blocs élémentaires permet la conception de fonctionnalités complexes. Ces fonctionnalités sont par exemple, l'exécution d'un calcul numérique, la mémorisation d'une donnée et l'exécution d'un algorithme numérique.

Les calculs effectués par le circuit intégré se traduisent par des changements d'état des portes logiques. En technologie CMOS, le changement d'état d'une porte consiste en la charge et la décharge électrique des transistors. Les transistors peuvent alors être considérés comme des capacités. Lorsque le changement d'état d'un bit est un changement de 0 vers 1, une charge électrique vient se stocker dans le condensateur C_L. Ce phénomène est illustré par la flèche (a) de la figure 3.1. Cela revient à connecter la capacité au V_{cc} ce qui entraîne une consommation d'énergie. De manière réciproque, lorsqu'un changement d'état du bit est un transfert de 1 vers 0, l'énergie accumulée dans le condensateur est déchargée dans la terre (ou neutre électrique) comme l'illustre la flèche (b) de la figure 3.1, il n'y a pas de consommation d'énergie. Les transitions de bit de 0 vers 0 ou de 1 vers 1 ne participent pas à la variabilité globale en électricité du circuit. La variation de la consommation du circuit est donc la conséquence des changements d'état de l'ensemble des portes logiques pendant l'exécution d'instructions.

On dit alors que le composant à des pertes d'informations. Dans cette thèse, nous parlerons principalement d'analyse d'émission électromagnétique et non d'analyse de courant. Ce canal auxiliaire permet lui aussi de récupérer les fuites du composant [QS01, GMO01, AARR03]. En effet, la matière qui est électriquement chargée produit les champs électromagnétiques. Le lien entre le courant et le champ électromagnétique est

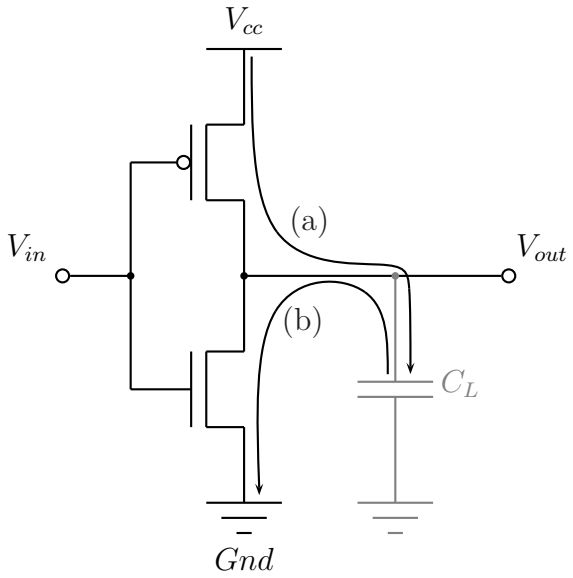


Figure 3.1 – Inverseur CMOS

décrit par les équations de Maxwell. Les détails théoriques concernant ces équations ne sont pas nécessaires pour la suite de ce manuscrit.

3.1.2 Exploitation des fuites

Les circuits intégrés sont embraqués dans de nombreuses applications numériques. Les premières applications sont le missile balistique Minuteman II (1962) et les ordinateurs de contrôles pour les missions Apollo (voir par exemple [Isa15], *Le décollage foudroyant des circuits intégrés*). Aujourd'hui, les circuits intégrés se retrouvent dans diverses applications numériques (passeport biologique, carte bancaire, carte vitale, véhicule autonome, téléphone, etc.). Le besoin en fiabilité est toujours présent, mais l'accent est aussi porté sur la sécurité. Effectivement, des données personnelles et/ou secrètes sont embarqués dans les dispositifs.

Ces données peuvent être directement menacées par les attaques dites « invasives » où le composant est modifié ou analysé [AK98]. Il est possible d'extraire des secrets, ou le design à des fins de clonage. C'est en particulier le but des techniques de rétro-ingénierie matérielle, elles consistent à retrouver les éléments de conception du circuit pour en extraire tout ou partie de celui-ci. Ces méthodes nécessitent généralement un retrait successif de chaque niveau du composant. Des microscopes optiques ou électroniques à balayage, suivi de technique de traitement d'images, sont utilisés pour reconstruire le schéma électrique du composant (voir par exemple [Cou15]).

Des techniques de rétro-ingénierie « non-invasives », à base de canaux auxiliaires (*Side Channel Analysis for Reverse Engineering (SCARE)* – en anglais), permettent aussi de reconnaître des fonctions exécutées par du code embarqué. Les méthodes de SCARE ont été introduites pour retrouver des spécifications secrètes d'un algorithme cryptographique ciblé. Novak [Nov03] décrit comment retrouver les valeurs d'une des deux tables de substitution de l'algorithme A3/A8, utilisé pour l'authentification et la génération de clefs de session de la télécommunication Groupe Spécial Mobile (GSM). L'attaque est améliorée par Clavier [Cla04] pour retrouver les deux tables secrètes et la clef secrète. Daudigny *et al.* [DLMV05] ont appliqué le concept des SCARE à l'algo-

rithme Data Encryption Standard (DES). Ils retrouvent des constantes impliquées dans l'algorithme, telles que les tables de permutations. Ces données sont, bien entendues, publiques, mais la présence de ces attaques illustre le principe de Kerckhoffs [Ker83]. La sécurité doit reposer sur le secret de la clef, et non sur l'algorithme. Avec ces attaques, l'algorithme secret serait retrouvé.

Les fuites des circuits intégrés causées par la manipulation des données sont aussi exploitées par les attaques par canaux auxiliaires en prenant pour cible l'implémentation des algorithmes cryptographiques.

3.2 Attaques par canaux auxiliaires : outils de cryptanalyses

Ces attaques permettent de retrouver les clefs secrètes utilisées par des cryptosystèmes pourtant prouvés mathématiquement sûrs. Les implémentations sur circuits intégrés induisent des fuites liées au traitement des données. Chaque opération est liée à un état interne dans le dispositif qui prend alors une condition différente selon les données manipulées. Afin d'exploiter ce phénomène, les attaques par canaux auxiliaires nécessitent un minimum de matériel. La figure 3.2 montre les différents éléments qui constituent un banc d'observation de la consommation.

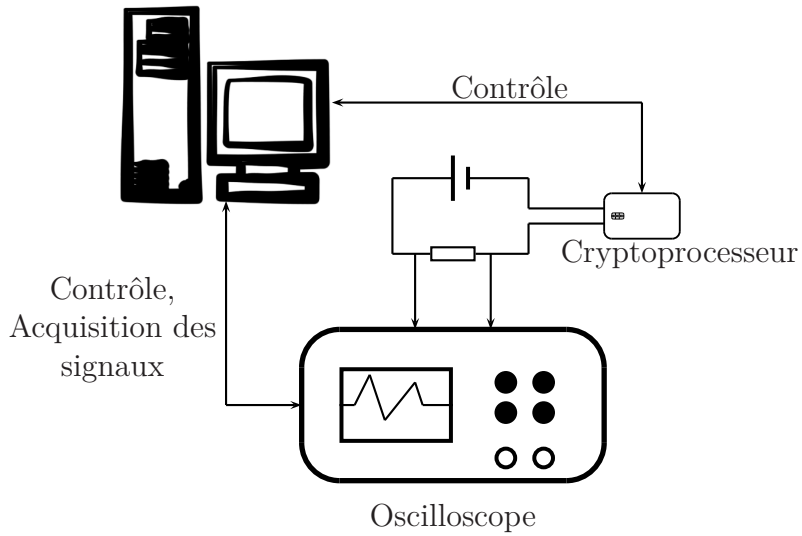


Figure 3.2 – Illustration d'un banc d'attaque par analyse de la consommation

3.2.1 Attaques de bases

Nous introduisons les attaques par canaux auxiliaires contre l'implémentation des algorithmes cryptographiques avec un exemple très parlant : l'attaque Simple Power Analysis (SPA) [KJJ99] appliquée à ECC.

Rappel de ElGamal pour ECC La fonction *à sens unique avec trappe* de l'ECC est construite sur la difficulté à résoudre le logarithme discret.

Paramètres publics Le cryptosystème nécessite que des paramètres soient publics pour l'ensemble de ses utilisateurs. Ces paramètres sont :

- Un entier q, qui définit le corps $\mathbb{F}_q$, et la courbe elliptique $E(\mathbb{F}_q)$.
- Un point de base $P \in E(\mathbb{F}_q)$, et n l'ordre de P.

Clef privée La clef privée d'un utilisateur $\mathcal{A}$ est un entier d.

Clef publique La clef publique de $\mathcal{A}$ est le point $Q \in E(F_q)$ défini par $Q = [d]P$.

Chiffrement ElGamal pour ECC Lorsqu'un utilisateur veut chiffrer un message $M \in E(F_q)$ pour $\mathcal{A}$ il effectue :

1. Tirer aléatoirement r dans $[\![1, n-1]\!]$.
2. Calculer $K_r = [r]P$.
3. Calculer $S = M + [r]Q$.
4. Envoyer le couple (K_r, S) à $\mathcal{A}$.

Déchiffrement Pour lire un message chiffré (K_r, S), l'utilisateur $\mathcal{A}$ fait le calcul

$$M = S - [d]K_r. \tag{3.1}$$

Attaque SPA sur ECC La clef secrète d est impliquée dans le calcul $[d]K_r$, c'est une multiplication scalaire. La difficulté à résoudre le logarithme discret, c'est-à-dire trouver d à partir de K_r et $[d]K_r$, assure la sécurité de la clef secrète. Dans le cadre des attaques par canaux auxiliaires, la menace est la suivante : un attaquant possède le dispositif où est embarquée la clef secrète d et possède un banc d'observation. Il exécute un déchiffrement pour un chiffré S, donc il calcule $[d]K_r$, pendant qu'il récupère la consommation du circuit. L'algorithme 3.1 permet de réaliser un tel calcul.

Entrées : Un point $P \in \mathbb{G}$ et $d = (d_{\mathfrak{n}-1} \dots d_0)_2$.
Sorties : $Q = [d]P$.

$R_0 \leftarrow P$;
pour $j = \mathfrak{n} - 2$ **à** 0 **faire**
  $R_0 \leftarrow [2]R_0$;
  si $d_j == 1$ **alors**
    $R_0 \leftarrow R_0 + P$;
  fin
fin
retourner R_0 ;

Algorithme 3.1 : Double-et-ajoute gauche-à-droite

La figure 3.3 illustre une trace de consommation de la multiplication scalaire avec un schéma de double-et-ajoute. Suivant le bit du scalaire d, la consommation du circuit

n'est pas la même. Lorsque le bit est 1, il y a une opération supplémentaire : l'« addition ». Cette irrégularité est observable dans la trace de consommation. En effet, une opération supplémentaire est exécutée, ce qui fait augmenter le temps requis pour faire l'itération en cours et, visuellement, ajoute un motif dans la trace. Il suffit à l'attaquant d'analyser la trace pour voir la durée de l'itération et retrouver les bits du secret d.

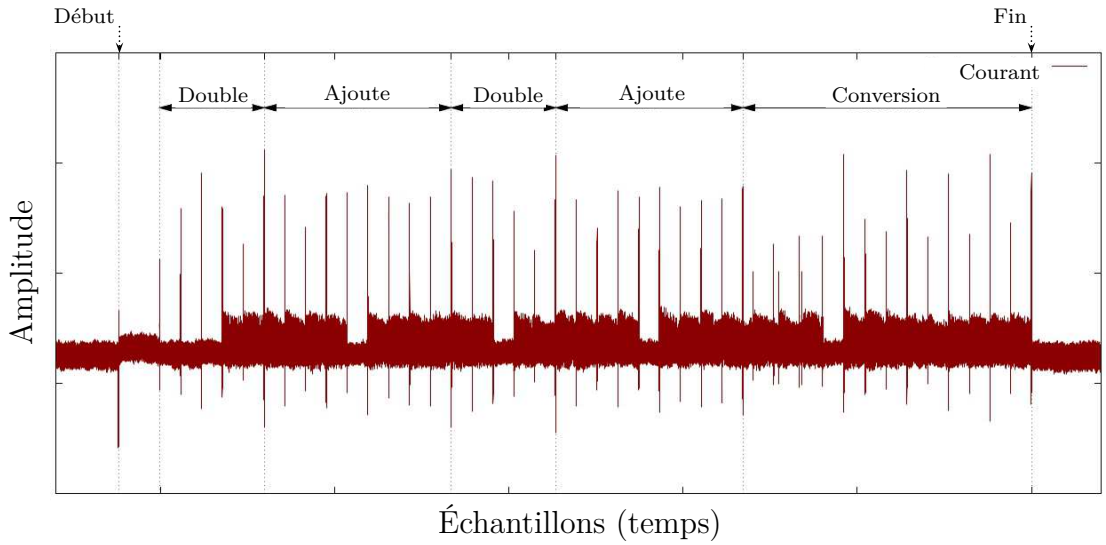


Figure 3.3 – Illustration d'une multiplication scalaire

3.2.2 Attaques avancées

Kocher *et al.* [KJJ99] proposent une nouvelle attaque plus avancée que la SPA. En effet, la Differential Power Analysis (DPA) prend en compte le fait que les fuites des canaux auxiliaires dépendent des données manipulées. Une cible est déterminée, c'est une opération qui dépend d'une entrée connue (ou calculable) et d'une petite partie de la clef (sous-clef). Ainsi, la dépendance entre les fuites des canaux auxiliaires provoquées par le traitement de cette opération et les entrées de cette opération peut être utilisée pour récupérer la sous-clef. Le principe général est de construire un modèle de fuite du dispositif afin de pouvoir estimer les fuites émises en fonction des données manipulées. Une partie de ces données sont connues, l'autre partie, sous-clef, est une hypothèse, cela permet de faire une prédiction sur la valeur d'une variable intermédiaire choisie. Cette variable intermédiaire prédite est envoyée dans le modèle de fuite pour estimer quelle serait la fuite réelle dans le cas où ces données sont effectivement manipulées. Ensuite, les mesures des canaux auxiliaires sont comparées avec les estimations. Intuitivement, pour la bonne hypothèse de sous-clef, la comparaison aura une meilleure « correspondance ». Cette « correspondance » est en pratique détectée par un test statistique, appelé un distingueur.

Le premier exemple que nous fournissons est une attaque DPA sur un algorithme de chiffrement basé sur les courbes elliptiques [Cor99].

Attaques DPA contre ECC

L'attaque SPA que nous avons vue précédemment est toujours possible. Des contre-mesures doivent être mises en place. Les contre-mesures à base de multipli-

cations scalaires régulières protègent le secret contre les attaques SPA. Les exemples les plus courants sont les multiplications scalaire double-et-ajoute toujours [Cor99] et l'échelle de Montgomery [Mon87]. Le test du bit du scalaire est source de fuites dans l'algorithme naïf 3.1. L'algorithme 3.2 représente la multiplication scalaire avec la méthode double-et-ajoute toujours. Le test critique est aussi évité avec l'échelle de Montgomery (cf. algorithm 3.3).

Entrées : Un point $P \in \mathbb{G}$ et $d = (d_{\mathfrak{n}-1} \dots d_0)_2$.
Sorties : $Q = [d]P$.

$R_0 \leftarrow P$;
pour $j = \mathfrak{n} - 2$ **à** 0 **faire**
  $R_0 \leftarrow [2]R_0$;
  $R_1 \leftarrow R_0 + P$;
  $b \leftarrow d_j$;
  $R_0 \leftarrow R_b$;
fin
retourner R_0 ;

Algorithme 3.2 : Double-et-ajoute toujours

Entrées : Un point $P \in \mathbb{G}$ et $d = (d_{\mathfrak{n}-1} \dots d_0)_2$.
Sorties : $Q = [d]P$.

$R_0 \leftarrow \mathcal{P}_\infty$; $R_1 \leftarrow P$;
pour $j = \mathfrak{n} - 1$ **à** 0 **faire**
  si $d_j == 0$ **alors**
    $R_1 \leftarrow R_0 + R_1$; $R_0 \leftarrow [2]R_0$;
  sinon
    $R_0 \leftarrow R_0 + R_1$; $R_1 \leftarrow [2]R_1$;
  fin
fin
retourner R_0 ;

Algorithme 3.3 : Echelle de Montgomery

Cependant, lorsque l'attaquant est en mesure d'acquérir **plusieurs traces** de multiplications scalaires $[d]K^{(i)}$ avec le même secret d et des bases $K^{(i)}$ différentes, des attaques de type DPA sont possibles [Cor99, Joy03]. La DPA cherche toujours à identifier si le bit du scalaire est 0 ou 1, mais ce sont les données manipulées $(K^{(i)})$ qui provoquent des fuites exploitables par des outils statistiques. L'idée est la suivante :

On note $d = (d_{\mathfrak{n}-1} \dots d_0)_2$ le scalaire secret. On suppose que l'attaquant connaît les bits de poids fort $d_{\mathfrak{n}-1}, \dots, d_{j+1}$, il fait l'hypothèse que le bit d_j vaut 1. Il récupère $C^{(i)}, i = 1, \dots, N$, les mesures d'un canal auxiliaire du dispositif qui exécute :

$$Q^{(i)} = \left[\sum_{h=j}^{\mathfrak{n}-1} d_h 2^{h-j} \right] P^{(i)}. \tag{3.2}$$

L'attaquant utilise une fonction de sélection booléenne Ξ pour trier les signaux en deux paquets selon $\Xi\left(Q^{(i)}\right)$. Par exemple $\Xi\left(Q^{(i)}\right)$ renvoie le bit de poids faible de $x_{Q^{(i)}}$.

L'attaquant moyenne ces paquets et fait la différence. Ainsi, si l'hypothèse $d_j = 1$ est fausse alors les deux paquets seront construits aléatoirement et donc leur différence sera proche de 0, il mettra $d_j = 0$ pour la suite, et va recommencer sur les bits restants. Dans le cas où l'hypothèse $d_j = 1$ est vraie, la différence des moyennes montrera un pic, car les paquets des traces pour les bits à 1 et ceux à 0 regroupent chacun des traces de fuites provenant des mêmes données manipulées. L'attaquant conservera $d_j = 1$ avant de passer aux bits suivants.

Attaques horizontales

Les attaques contre la multiplication scalaire ont été largement étudiées ces dernières années. Des attaques de type Big Mac [Wal01] ou d'autres, dites « horizontales » ont vu le jour [CFG+10, CFG+12, BJPW13a, PIMT15], elles sont capables de retrouver le scalaire secret en une seule trace.

Une attaque classique (DPA dite verticale par exemple) exploite plusieurs traces au même instant (cf. figure 3.4). Cet instant est la fenêtre où l'opération ciblée est traitée par le circuit intégré. L'approche horizontale est que cette opération critique est réalisée plusieurs fois dans la même exécution de l'algorithme. La fenêtre ciblée se répète, les traces sont sélectionnées à ces instants pour faire une attaque de type DPA par exemple. La figure 3.5 illustre le principe d'une attaque horizontale.

Les attaques horizontales seront étudiées dans le cadre des couplages en section *Attaque horizontale contre les couplages*, page 111.

Application de la DPA aux algorithmes de chiffrement par bloc

Dans cette thèse, nous nous intéressons aux attaques contre l'implémentation des algorithmes de couplage, nous allons voir que ces attaques sont analogues aux attaques contre les algorithmes de chiffrement par bloc. Nous allons nous servir du Advanced Encryption Standard (AES) comme exemple pour introduire les attaques avancées qui seront utilisées dans le chapitre 4 contre les algorithmes de couplages.

Il est pratique de présenter les attaques contre l'AES (voir par exemple [GT02]), pour cela commençons par décrire le cryptosystème.

Description de l'AES [FIP01] L'AES est un algorithme de chiffrement par blocs qui manipule les données octet par octet. Les blocs de données sont représentés par une matrice d'octet ayant 4 lignes et un nombre de colonnes N_b de 4, 6 ou 8. De façon similaire, la clef est sous forme d'une matrice à 4 lignes et $N_k = 4, 6$ ou 8 colonnes. L'AES est constitué d'opérations qui sont répétées lors de r tours. Le nombre de tours est défini selon les tailles N_b et N_k, la table 3.1 montre ce nombre de tours.

Table 3.1 – Nombre de tours selon les valeurs de N_b et N_k

	N_b		
N_k	**4**	**6**	**8**
4	10	12	14
6	12	12	14
8	14	14	14

AES

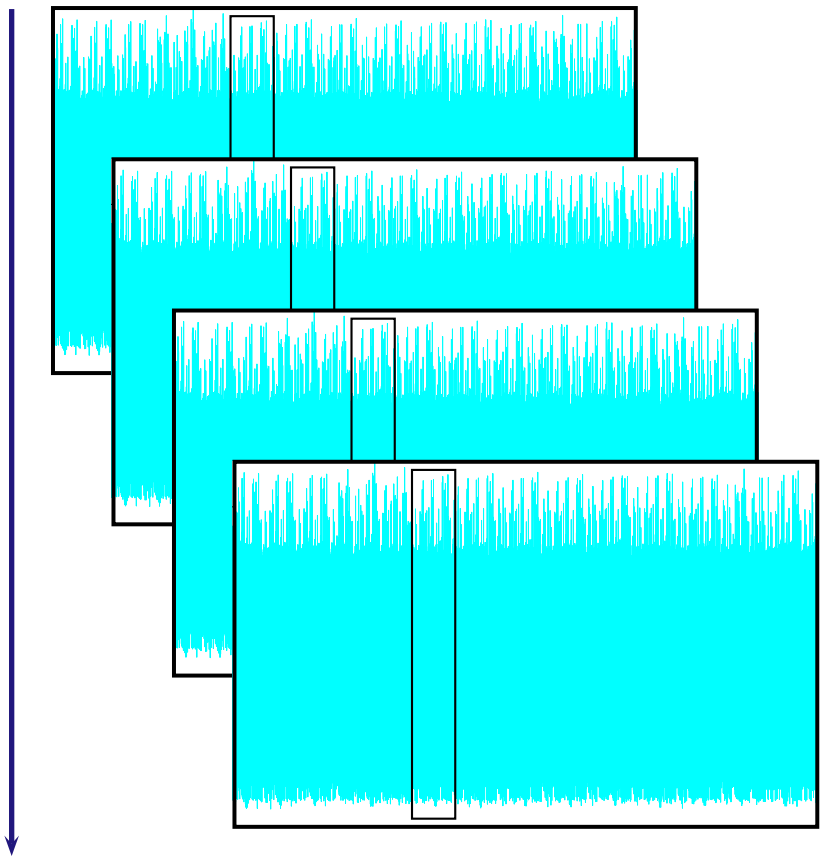


Figure 3.4 – Illustration d'une attaque verticale

Les opérations d'un tour sont les suivantes :

- Substitution d'octets (*SubBytes* – en anglais), qui est non-linéaire.
- Décalage de lignes (*ShiftRows* – en anglais), transformation linéaire.
- Mixage de colonnes (*MixColumns* – en anglais), qui est aussi une transformation linéaire.
- Ajout de la clef de tour (*AddRoundKey* – en anglais).

Les $r-1$ premiers tours comportent ces quatre opérations, tandis que le tour final n'exécute pas le *MixColumns*.

SubBytes Les octets sont vus comme des éléments de $\mathbb{F}_{256}$. Le corps $\mathbb{F}_{256}$ est vu comme un espace vectoriel isomorphe à $(\mathbb{F}_2)^8$. Par exemple $[57] = 01010111$ correspond à $X^6 + X^4 + X^2 + X + 1$ dans $\mathbb{F}_{2^8} \simeq {}^{\mathbb{F}_2[X]}/_{(P_8)}$. Pour construire $\mathbb{F}_{2^8}$, le polynôme irréductible de degré 8 choisi est $P_8 = X^8 + X^4 + X^3 + X + 1$. L'opération $SubBytes(x)$,

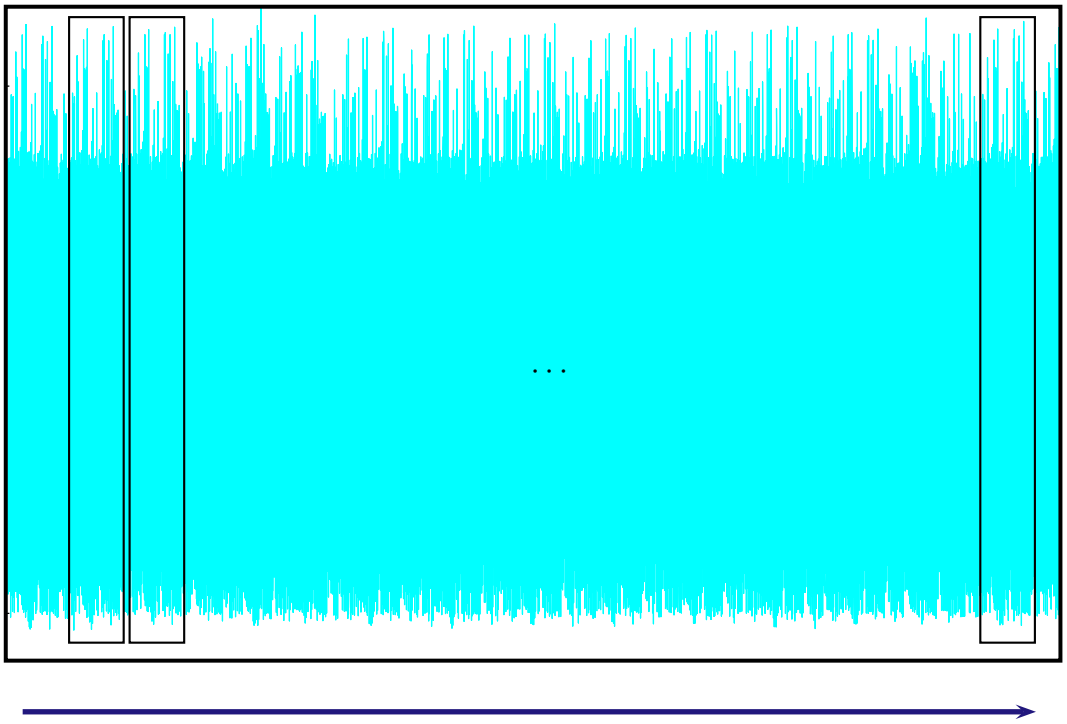


Figure 3.5 – Illustration d'une attaque horizontale

procède d'abord à l'inversion de x :

$$\begin{array}{c} y = x^{-1} \mod P_8 \text{ dans } \mathbb{F}_{256}, \\ \text{si } x = 0 \text{ alors } y = 0. \end{array} \tag{3.3}$$

Suivi de la transformation affine :

$$SubBytes\,(x) = \begin{pmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}. \tag{3.4}$$

En pratique, une table de substitution (de taille 16×16) peut être utilisée pour réaliser le *SubBytes*. Un octet en entrée x est déterminé par 4 bits de poids forts et 4 de poids faibles, ils définissent respectivement la ligne et la colonne de l'entrée de la table. Par exemple, $SubBytes\,([57])$ renvoie la ligne 5 et la colonne 7 de la table, à savoir, $[5b]$.

ShiftRows La fonction *ShiftRows* est une transposition d'octet (un décalage circulaire). La ligne i est décalée de i positions. La figure 3.6 illustre ces décalages dans le cas $N_b = 4$.

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

$\longrightarrow$

a_{00}	a_{01}	a_{02}	a_{03}
a_{11}	a_{12}	a_{13}	a_{10}
a_{22}	a_{23}	a_{20}	a_{21}
a_{33}	a_{30}	a_{31}	a_{32}

Figure 3.6 – Illustration du *ShiftRows*

MixColumns Les 4 composantes d'une colonne sont vues comme un polynôme de degré 3 à coefficients dans $\mathbb{F}_{256}$. Le *MixColumns* consiste à faire ces deux manipulations :

1. Multiplier par $C(X) = [03]X^3 + [01]X^2 + [01]X + [02]$.
2. Prendre le reste de la division par $X^4 + 1 \in \mathbb{F}_{256}[X]$.

D'un point de vue matriciel, le *MixColumns* est donné par la relation :

$$\begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} [02] & [03] & [01] & [01] \\ [01] & [02] & [03] & [01] \\ [01] & [01] & [03] & [02] \\ [03] & [01] & [01] & [02] \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{pmatrix}. \tag{3.5}$$

AddRoundKey La clef de chiffrement est intégrée à cette étape. Il s'agit d'un eXclusive OR (XOR) bloc à bloc. La clef en entrée de l'AES est dérivée en clef de tour via la fonction de dérivation de clef (*KeySchedule* – en anglais). Une clef de 128 bits (resp. 192, resp. 256) sera dérivée en 10 clefs de tour (resp. 12, resp. 14) de 128 bits (resp. 192, resp. 256).

Données observées L'opération ciblée est la fonction *SubBytes*, la figure 3.7 illustre ses entrées/sorties. L'attaquant possède le dispositif embarqué qui manipule la clef secrète $k^\star$, et il est en mesure d'exécuter **des** chiffrements pour **des** messages $x^{(i)}$, pendant qu'il récupère la consommation du circuit.

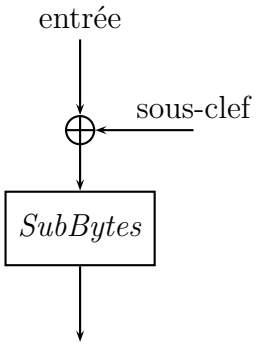


Figure 3.7 – Illustration de la cible *SubBytes*

L'entrée, qui est connue, subit un XOR avec la clef de tour avant d'être envoyée dans la fonction *SubBytes*. L'attaque porte sur la valeur intermédiaire en sortie de *SubBytes*.

Cette valeur intermédiaire est facilement calculable au premier tour à partir de l'entrée x et d'une hypothèse k sur la sous-clef : $SubBytes\,(x \oplus k)$. Le déroulement de l'attaque consiste d'abord à récupérer des mesures $C^{(i)}, i = 1, \ldots, N$ d'un canal auxiliaire pour des entrées $x^{(i)}$ fixées et une sous-clef $k^\star \in \mathcal{K} = \{0, \ldots, 255\}$ secrète. Chaque mesure $C^{(i)}$ est une trace échantillonnée sur m points.

La même attaque est possible si l'attaquant est capable d'observer le chiffrement de messages (qu'il ne connaît pas) avec une clef secrète $k^\star$ et dont il récupère les chiffrés correspondants. Il va cibler le dernier tour de l'AES. Puisqu'il connaît le chiffré, il peut remonter à la sortie du *SubBytes*, puis à l'entrée du *SubBytes*. Ensuite, en faisant une hypothèse sur la sous-clef, il peut remonter à l'« entrée » (figure 3.7) pour faire la même attaque que précédemment.

La différence des moyennes comme distingueur

Une fois les traces acquises, l'objectif de la DPA de Kocher *et al.* [KJJ99] est de ranger les signaux dans deux ensembles selon un critère. Ce critère est la fonction de sélection, elle prend en entrée les données connues et l'hypothèse sur la sous-clef. Pour chaque hypothèse de sous-clef, on va obtenir deux paquets de traces, qui seront ensuite moyennés. Leur différence sera représentative de l'hypothèse de sous-clef traitée. L'algorithme 3.4 décrit le principe de l'attaque.

Entrées : Des mesures $C^{(i)}, i = 1, \ldots, N$ pour les entrées $x^{(i)}$.
Sorties : $\widehat{k} \in \mathcal{K}$, une sous-clef candidate .

pour $k = 0$ **à** 255 **faire**
  $S_{\text{haut}} \leftarrow \emptyset$; `// Initialisation du paquet « haut »`
  $S_{\text{bas}} \leftarrow \emptyset$; `// Initialisation du paquet « bas »`
  pour $l = 1$ **à** N **faire**
    si $\Xi(x^{(l)}, k) == 1$ **alors**
      $C^{(l)}$ est ajouté à S_{haut} ;
    sinon
      $C^{(l)}$ est ajouté à S_{bas} ;
    fin
  fin
  $M_k \leftarrow \left|\overline{S_{\text{haut}}} - \overline{S_{\text{bas}}}\right|$; `// Différence des moyennes`
fin
retourner $\widehat{k}$ *pour lequel* M_k *a le plus grand pic* ;

Algorithme 3.4 : DPA de Kocher *et al.* [KJJ99]

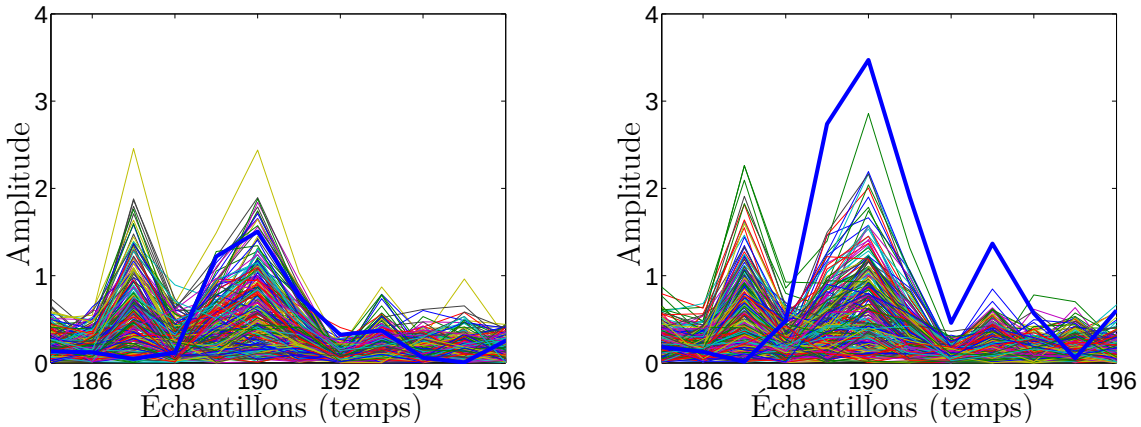
Dans l'article de Kocher *et al.* [KJJR11], la fonction de sélection $\Xi(x^{(l)}, k)$ se réduit à prendre le bit de poids faible de $SubBytes\left(x^{(l)} \oplus k\right)$. Cette attaque permet de retrouver l'octet $k^\star$.

La figure 3.8 montre deux exemples de DPA pour les fonctions de sélections Ξ_1 et Ξ_2 définies respectivement par les équations 3.6 et 3.7.

$$\Xi_1(x^{(l)}, k) = lsb\left(SubBytes\left(x^{(l)} \oplus k\right)\right), \tag{3.6}$$

$$\Xi_2(x^{(l)}, k) = \begin{cases} 1 \text{ si } HW_8\left(SubBytes\left(x^{(l)} \oplus k\right)\right) > 4, \\ 0 \text{ si } HW_8\left(SubBytes\left(x^{(l)} \oplus k\right)\right) < 4. \end{cases} \tag{3.7}$$

On y voit clairement que la fonction Ξ_2 donne une meilleure distinction de la clef. La seconde fonction de sélection donne un pic avec une amplitude plus élevée qu'avec la première fonction. De plus, la valeur la plus élevée est atteinte pour la bonne hypothèse de clef avec la seconde fonction de sélection.



(a) Différences des moyennes pour Ξ_1 (b) Différences des moyennes pour Ξ_2

Figure 3.8 – Différences des moyennes pour deux fonctions de sélections

La corrélation comme distingueur

Un autre distingueur, qui ne nécessite pas de fonction de sélection, est proposé par Brier *et al.* [BCO04]. Il s'agit d'évaluer le coefficient de corrélation entre les prédictions et les données mesurées. La méthode est décrite via l'algorithme 3.5

Entrées : Des mesures $C^{(i)}, i = 1, \ldots, N$ pour les entrées $x^{(i)}$.
Sorties : $\widehat{k} \in \mathcal{K}$, une sous-clef candidate .

$H_0 \leftarrow \mathcal{M}_{N,256}(0)$; `// Initialisation de la matrice des sorties hypothétiques`
pour $l = 1$ **à** N **faire**
  pour $k = 0$ **à** 255 **faire**
    $H_0(l,k) \leftarrow HW\left(SubBytes\left(x^{(l)} \oplus k\right)\right)$; `// Modèle en poids de Hamming`
  fin
fin
pour $k = 0$ **à** 255 **faire**
  $\mathfrak{C}_k(i) \leftarrow \left|\rho\left(H_0(\cdot,k), C^{(\cdot)}(i)\right)\right|, \forall i = 1, \ldots, m$; `// Corrélations`
fin
retourner $\widehat{k}$ *pour lequel* $\mathfrak{C}_k$ *a le plus grand pic* ;

Algorithme 3.5 : CPA de Brier *et al.* [BCO04]

La boucle finale, ligne 7 de l'algorithme 3.5, correspond aux calculs de corrélations entre les valeurs intermédiaires prédites et les mesures. Plus généralement, ce calcul s'écrit $\mathfrak{C} = H_0 \mathfrak{R} C$. La procédure $\mathfrak{R}$ qui fait le calcul de corrélations entre les prédictions et les mesures est illustrée par la figure 3.9.

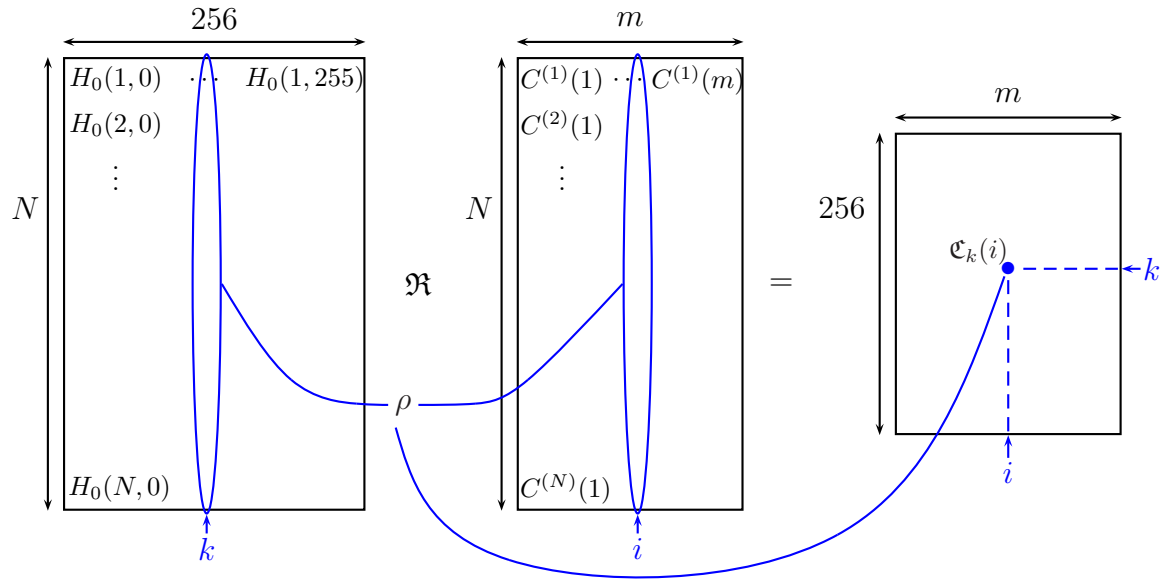
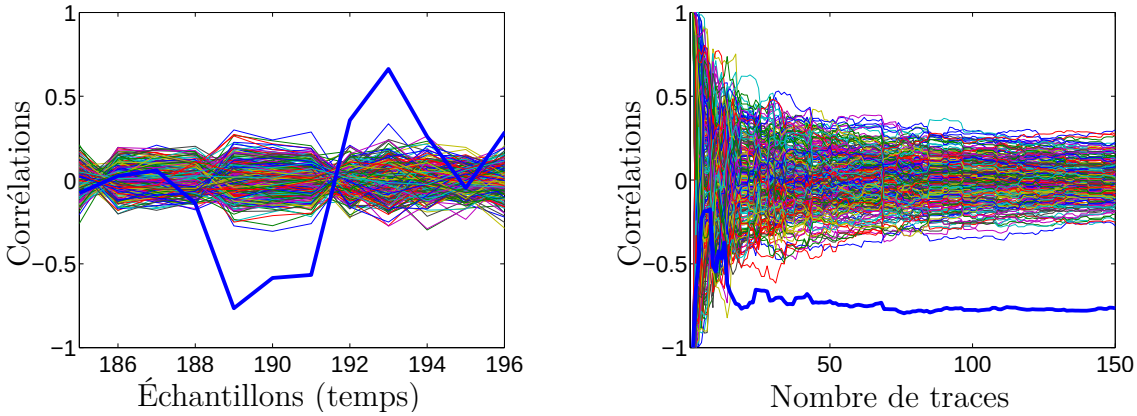


Figure 3.9 – Principe de la CPA sur 8 bits

Un exemple de résultat de Correlation Power Analysis (CPA) est fourni par la figure 3.10a. Avec les mêmes données que la DPA, la courbe en gras, qui correspond à la bonne clef, est clairement identifiable. De plus, la figure 3.10b montre l'évolution des corrélations en fonction du nombre de traces. Ici encore, nous voyons que la bonne hypothèse de clef se distingue largement des autres.



(a) Corrélations issues d'une CPA

(b) Corrélations en fonction du nombre de traces

Figure 3.10 – Résultats d'une CPA en sortie de *SubBytes* de l'AES

Attaques par profilage

Les attaques par profilage ont été introduites par Chari *et al.* [CRR02]. Ici, nous introduisons leur fonctionnement contre l'AES, et nous verrons dans la section 4.3.4 nos

résultats de telles attaques contre les couplages.

Ces attaques sont très dangereuses [CRR02, SLP05, SKS09, CK14]. L'adversaire considéré est le plus puissant dans le contexte des attaques par canaux auxiliaires. Il doit avoir une copie du dispositif ciblé. L'algorithme cryptographique ciblé sur ce dispositif test est caractérisé, cette phase est possible uniquement car, cet attaquant contrôle lui-même les entrées et les données secrètes sur ce dispositif. Il se sert de cette copie pour créer des profils, il caractérise le composant pour avoir un modèle précis du comportement avec les variables intermédiaires. La dangerosité de ces attaques est aussi due à sa structure en deux phases, ce sont deux étapes de l'attaque qui peuvent être incarnées par des acteurs différents.

Phase de profilages L'attaquant a le contrôle des données du dispositif de test. Il doit pouvoir demander l'exécution du chiffrement avec des entrées et des clefs différentes qu'il connaît. De cette manière, il récupère des traces $C^{(i)}, i = 1, \ldots, N_p$, échantillonnées sur m points, pour des entrées $x^{(i)}$ et des clefs $k^{(i)}$ qu'il connaît. L'attaquant définit ensuite une cible dans l'algorithme de chiffrement.

Supposons toujours que l'objectif soit l'AES, les entrées sont les messages à chiffrer et la cible est la fonction *SubBytes* au premier tour. L'attaquant crée ainsi une fonction g qui pour un message et une clef renvoie une classe, dans notre exemple cette classe sera la sortie du *SubBytes* au premier tour, c'est-à-dire $g\left(x^{(i)}, k^{(i)}\right) = SubBytes\left(x^{(i)} \oplus k^{(i)}\right)$.

Un profil est un couple (moyenne, matrice de covariance). Il y a autant de profils que de nombre de classes, toujours dans l'exemple, il y a 256 profils $\left(\overline{C_{cl}}, S_{cl}\right)_{0 \leq cl \leq 255}$ construits de la manière suivante :

$$\overline{C_{cl}} = \frac{1}{N_{cl}} \sum_{i=1}^{N_p} C^{(i)} \cdot \mathbf{1}_{\left\{\iota | g\left(x^{(\iota)}, k^{(\iota)}\right) = cl\right\}}(i), \tag{3.8}$$

pour toutes les $cl \in \{0, \ldots, 255\}$. Le nombre N_{cl} est le nombre de traces qui sont associées à la classe cl et $\mathbf{1}_A$ désigne la fonction indicatrice de l'ensemble A :

$$\mathbf{1}_A(x) = \begin{cases} 1 \text{ si } x \in A, \\ 0 \text{ sinon.} \end{cases}$$

Les matrices de covariances pour chaque classe sont définies par :

$$S_{cl} = \frac{1}{N_{cl} - 1} \sum_{i=1}^{N_p} \left(C^{(i)} - \overline{C_{cl}}\right) \left(C^{(i)} - \overline{C_{cl}}\right)^T \cdot \mathbf{1}_{\left\{\iota | g\left(x^{(\iota)}, k^{(\iota)}\right) = cl\right\}}(i). \tag{3.9}$$

La transposée d'un vecteur étant notée $\cdot^T$.

Phase d'attaque Cette phase exploite des signaux acquis pour une sous-clef $k^\star$ secrète fixée. On note $C'^{(i)}$ les N_A traces correspondant au chiffrement de $x'^{(i)}$. Le bruit contenu dans un signal est supposé suivre une distribution gaussienne, c'est-à-dire que la probabilité qu'une trace C soit observée sachant qu'elle est associée à une classe cl est :

$$\mathbb{P}(C|cl) = \frac{1}{\sqrt{(2\pi)^m |S_{cl}|}} \exp\left(-\frac{1}{2}\left(C - \overline{C_{cl}}\right)^T S_{cl}^{-1} \left(C - \overline{C_{cl}}\right)\right). \tag{3.10}$$

Grâce à la distribution gaussienne, il suffit de mesurer la probabilité du bruit de chacun des signaux sachant de quelle classe il provient. Une hypothèse sur la sous-clef donnera la classe hypothétique associée. Le théorème de Bayes permet d'exprimer la probabilité $\mathbb{P}\left(k_j|C'^{(i)}\right)$ en fonction des probabilités *a priori* $\mathbb{P}(k_l)$ et $\mathbb{P}(C^{(i)}|k_l)$. Cette forme est donnée par l'équation :

$$\mathbb{P}\left(k_j|C'^{(i)}\right) = \frac{\mathbb{P}\left(C'^{(i)}|k_j\right)\mathbb{P}(k_j)}{\sum_{k\in\mathcal{K}}\left(\mathbb{P}\left(C'^{(i)}|k\right)\mathbb{P}(k)\right)}. \tag{3.11}$$

Cette probabilité est facilement transposable pour plusieurs traces. On note $\mathcal{C}'$ cet ensemble de traces.

$$\mathbb{P}\left(k_j|\mathcal{C}'\right) = \frac{\left(\prod_{C'\in\mathcal{C}'}\mathbb{P}\left(C'|k_j\right)\right)\mathbb{P}(k_j)}{\sum_{k\in\mathcal{K}}\left(\left(\prod_{C'\in\mathcal{C}'}\mathbb{P}\left(C'|k\right)\right)\mathbb{P}(k)\right)}. \tag{3.12}$$

La probabilité $\mathbb{P}\left(C'^{(i)}|k\right)$ se calcule grâce à l'équation 3.10 de la fonction de densité de probabilité. Il faut calculer la classe de k et du message correspondant $x^{(i)}$, sortie du *SubBytes* par exemple pour l'AES, puis calculer la probabilité de l'équation 3.12. Cette phase d'attaque est reprise par l'algorithme 3.6.

Entrées : Des mesures $C'^{(i)}, i = 1, \ldots, N_A$ pour les entrées $x'^{(i)}$ et les profils $\left(\overline{C_{cl}}, S_{cl}\right)$.
Sorties : $\widehat{k} \in \mathcal{K}$, une sous-clef candidate.

$L \leftarrow \mathcal{M}_{1,256}(0)$; `// Initialisation du vecteur de vraisemblance`
pour $k_j = 0$ **à** 255 **faire**
  $L(k_j) = \frac{\left(\prod_{l=1}^{N_A}\mathbb{P}\left(C'^{(l)}|k_j\right)\right)\mathbb{P}(k_j)}{\sum_{k\in\mathcal{K}}\left(\left(\prod_{l=1}^{N_A}\mathbb{P}\left(C'^{(l)}|k\right)\right)\mathbb{P}(k)\right)}$;
fin
$\widehat{k} \leftarrow \arg\max_k L$;
retourner $\widehat{k}$;

Algorithme 3.6 : Phase d'extraction de la clef d'une attaque par profilage [CRR02]

Il est souvent préférable de prendre le logarithme de la probabilité (cf. équation 3.10). En effet, les probabilités sont multipliées entre elles, numériquement, les valeurs de L vont rapidement tomber à zéro. Le logarithme de vraisemblance évite ce problème, car la multiplication devient une addition, dans ce cas il faut prendre l'argument minimum plutôt que le maximum pour extraire $\widehat{k}$.

Attaques en collisions

Les attaques par canaux auxiliaires par collisions ont été introduites par Schramm *et al.* [SLFP04] contre l'AES. Le principe de ces attaques est d'exploiter l'apparition de collisions entre des valeurs intermédiaires de l'algorithme. Les canaux auxiliaires sont utilisés par l'attaquant pour détecter ces collisions.

Attaques par canaux auxiliaires en collisions sur AES Ces attaques sont spécifiques aux algorithmes ciblés. Pour l'AES par exemple, il s'agit d'observer les collisions à la sortie du *MixColumns*.

Avec les notations précédentes, la sortie b_0 du *MixColumns* s'écrit

$$b_0 = [02]a_0 + [03]a_1 + [01]a_2 + [01]a_3. \tag{3.13}$$

En se positionnant à la sortie du premier tour de l'AES, a_0, a_1, a_2 et a_3 sont les sorties du *SubBytes*, ils s'écrivent donc $SubBytes\,(x_0 \oplus k_0)$, $SubBytes\,(x_1 \oplus k_1)$, $SubBytes\,(x_2 \oplus k_2)$ et $SubBytes\,(x_3 \oplus k_3)$. L'idée de l'attaque de Schramm *et al.* [SLFP04] est de chercher deux entrées différentes qui produisent le même octet b_0. Les messages x considérés sont de la forme $x_0 = x_1 = 0$ et $x_2 = x_3$. Si deux messages x et x' sont tels que $x_2 = x_3 = \delta$, $x'_2 = x'_3 = \epsilon \neq \delta$ et qu'ils produisent le même b_0 alors l'équation suivante est satisfaite :

$$SubBytes\,(\delta \oplus k_2) \oplus SubBytes\,(\delta \oplus k_3) = SubBytes\,(\epsilon \oplus k_2) \oplus SubBytes\,(\epsilon \oplus k_3) \tag{3.14}$$

Schramm *et al.* [SLFP04] montrent également comment résoudre un tel système de manière efficace afin de retrouver le secret k.

Par ailleurs, il existe aussi des attaques en collisions contre la cryptographie à clefs publiques (RSA et ECC) [FV03, YLMH05, HNI+06, HMA+08, AF08, ASYZ13, BJPW13b, DLLM15]. Nous illustrons ici l'attaque par doublement de Fouque *et al.* [FV03] contre la multiplication scalaire $Q = [d]P$ (algorithme 3.1). Par exemple, pour $d = 19 = (10011)_2$, le détail des calculs de doublement et d'addition est donné dans la table 3.2.

Table 3.2 – Exemple de l'attaque en collisions par doublement

j	d_j	$[d]P$	$[d]([2]P)$
3	0	$[2]P$	$[2]([2]P)$
2	0	$[2]([2]P)$	$[2]([4]P)$
1	1	$[2]([4]P)$ $[8]P + P$	$[2]([8]P)$ $[16]P + [2]P$
0	1	$[2]([9]P)$ $[18]P + P$	$[2]([18]P)$ $[36]P + [2]P$
Total		$[19]P$	$[38]P$

L'attaquant récupère deux traces $C^{(1)}$ et $C^{(2)}$ d'un canal auxiliaire (consommation de courant, émanation électromagnétique, ...) des deux multiplications scalaire de P et $[2]P$. Il cherche ensuite à détecter s'il y a des collisions entre un doublement à la $(i+1)$-ième itération dans $C^{(1)}$ et un doublement à la i-ième itération dans $C^{(2)}$. Une telle collision apparait **si et seulement si** $d_i = 0$.

Cette classe d'attaque sera plus largement détaillée au chapitre 5 lorsque nous présenterons l'effet de cette attaque contre l'implémentation des algorithmes de couplage.

Il est à noter que toutes ces attaques contre ECC visent le scalaire (secret) d'une multiplication scalaire. Le cas des couplages est bien différent. Les algorithmes de

couplages ont effectivement une structure de multiplication scalaire (double–et–ajoute) mais le scalaire est connu, le secret que l'attaquant vise est un point en entrée du couplage.

3.3 État de l'art des attaques contre les couplages

Dans les protocoles cryptographiques à base de couplage, le secret qu'un attaquant va cibler est un des deux points en entrée du couplage. Les attaques contre les implémentations des algorithmes de couplages ont été mises en lumière par Page *et al.* [PV04].

3.3.1 Synthèse des attaques contre les couplages

Les attaques par canaux auxiliaires sur les algorithmes cryptographiques ont été largement étudiées depuis plus de deux décennies. Les cryptosystèmes à clef publique tels que RSA ou ECC ont principalement été montrés vulnérables face aux attaques dont l'objectif est de retrouver l'exposant secret (en RSA) ou le scalaire secret (en ECC) utilisé dans un système de déchiffrement ou de signature. Le cas des couplages est différent.

Plusieurs publications ont abordé les attaques par canaux auxiliaires sur les couplages sur des corps de caractéristiques 2 ou 3. Ces études sont simplement mentionnées à titre de référence, étant donné que notre implémentation repose sur des corps premiers en grandes caractéristiques. Page *et al.* [PV04] ont proposé l'un des premiers articles décrivant théoriquement les attaques physiques (attaques passives par canaux auxiliaires et attaques actives par injections de fautes) sur les algorithmes de couplage. Ils ont ciblé l'algorithme Duursma-Lee [DL03], qui est utilisé pour calculer le couplage de Tate sur les courbes elliptiques sur des corps finis de caractéristique 3. La manipulation des données pendant l'algorithme Duursma-Lee implique le produit d'un élément secret et d'une valeur dérivée du point connu en entrée de couplage. Les auteurs proposent une attaque de type SPA sur des algorithmes de multiplication modulaire qui sont implémentés en utilisant la méthode décale-et-additionne (*shift-and-add*). Ils décrivent de plus une attaque DPA qui vise à récupérer le secret bit par bit. Kim *et al.* [KTH+06] ont proposé des attaques temporelles, SPA et DPA pour cibler les opérations arithmétiques qui sont impliquées dans les couplages en caractéristique 2. Dans le contexte du couplage Eta sur les corps binaires, l'opération ciblée est $a(b + r)$, où a et b sont dérivés du secret, et r est dérivé de l'entrée connue. Les auteurs concluent que, théoriquement, la DPA bit à bit proposée par Page *et al.* [PV04] pourrait encore récupérer le point secret utilisé dans le calcul du couplage. Pan *et al.* [PM11] ont proposé une attaque CPA en pratique basée sur un modèle de distance Hamming sur le couplage Eta sur un corps en caractéristique 2 sur des courbes supersingulières.

L'un des premiers articles décrivant les attaques par canaux auxiliaires sur les couplages sur des corps de grande caractéristique a été proposé par Whelan *et al.* [WS06], qui ont utilisé une CPA pour cibler les opérations arithmétiques afin de récupérer le secret. Pour le calcul de la CPA, ils proposent le schéma suivant : calculs des corrélations entre les sorties hypothétiques de l'opération arithmétique $x \times k$ pour toutes les clefs possibles k et les traces de fuite. Pour chaque hypothèse de clef, il y aura une courbe de corrélation correspondante. La clef candidate sera celle avec le pic le plus élevé (la même CPA que contre l'AES vue au paragraphe *La corrélation comme distingueur*,

page 71 mais en remplaçant la fonction *SubBytes* par la multiplication modulaire). Dans le même article, les auteurs ont discuté de l'utilisation de la longueur des mots (8, 16, 32 ou 64) pour représenter des nombres multiprécision. En effet, ils détaillent les calculs de corrélation partielle qui peuvent être utilisés par une CPA pour cibler une partie du mot. Plus précisément, si la taille de l'architecture de la cible est 32 bits, l'énumération des sous-clefs devrait se faire sur 2^{32} éléments, ce qui est considéré irraisonnable aujourd'hui. Ainsi, il faudra cibler une partie de ces 32 bits, par exemple l'octet de poids faible, et donc travailler avec des corrélations partielles. El Mrabet *et al.* [EDFL09] ont ensuite proposé une simulation de l'attaque de l'algorithme de Miller pour un couplage sur le corps $\mathbb{F}_{251}$. L'équation de la tangente a été ciblée, car elle contient une multiplication modulaire d'une coordonnée dérivée d'un point d'entrée public par une valeur déterministe dérivée du point secret.

Ghosh *et al.* [GR11] ont détaillé une attaque DPA sur la soustraction modulaire dans un couplage Tate sur une courbe elliptique de Barreto–Naehrig [BN05]. Blömer *et al.* [BGL13] ont ensuite décrit les attaques par canaux auxiliaires sur les additions modulaires et les multiplications d'éléments dans un corps fini en grande caractéristique. Tout comme El Mrabet *et al.* [EDFL09], ils montrent que ces attaques sont possibles même si le point secret est utilisé en tant que premier argument du calcul du couplage. Les auteurs accompagnent leurs résultats par des simulations. En effet, ils utilisent des traces simulées avec un modèle de poids de Hamming additionnées d'un bruit gaussien. Unterluggauer *et al.* [UW14] ont publié la première attaque pratique contre un couplage, de plus, en grande caractéristique. Ils utilisent une approche de type CPA, comme précédemment décrite par Whelan *et al.* [WS06]. Les auteurs ont ciblé les opérations modulaires lors d'un couplage Ate pour trouver les 16 bits de poids faible du secret, puis les 16 bits de poids fort. Ils travaillent sur une architecture 32 bits et profitent du fait que le processeur fonctionne avec un multiplicateur 16 bits. Leur configuration nécessitait plus de 1500 traces pour retrouver le point secret.

3.3.2 Synthèse des contre-mesures

Les attaques par canaux auxiliaires nécessitent toutes la connaissance des coordonnées d'un point en entrée du couplage. Ces dernières sont utilisées pour calculer des variables intermédiaires. L'objectif des contre-mesures est de supprimer le lien entre l'entrée connue du couplage et les variables intermédiaires.

Contre-mesures externes au couplage

La première contre-mesure est proposée par les auteurs de la première attaque. Page *et al.* [PV04] proposent d'exploiter la propriété de bilinéarité des couplages. Si a et b sont des entiers quelconques, alors $e([a]P, [b]Q)^{1/ab} = e(P, Q)$. La contre-mesure utilise cette randomisation pour que l'attaquant perde la connaissance des coordonnées du point public. Le calcul de $e(P, Q)$ est remplacé par l'algorithme 3.7. On nommera cette contre-mesure masquage multiplicatif.

La seconde contre-mesure de Page *et al.* [PV04] consiste en l'utilisation d'un masque additif. Par exemple, si Q est le secret, alors le point P sera masqué, c'est-à-dire que l'on utilise la relation $e(P, Q) = e(P + R, Q)e(R, Q)^{-1}$ pour un point $R \in \mathbb{G}_1 \setminus \mathcal{P}_\infty$ aléatoire. L'algorithme 3.8 décrit la démarche de la contre-mesure par masquage additif.

Entrées : $P \in \mathbb{G}_1$ et $Q \in \mathbb{G}_2$.
Sorties : $e(P, Q)$.

1 Tirer aléatoirement deux entiers a et b ;
2 $P' \leftarrow [a]P$; $Q' \leftarrow [b]Q$;
3 $h \leftarrow e(P', Q')$;
4 $h \leftarrow h^{1/ab}$;
5 **retourner** h ;

Algorithme 3.7 : Premiere contre-mesure de Page *et al.* [PV04]

Entrées : Le secret $Q \in \mathbb{G}_2$ et le point $P \in \mathbb{G}_1$.
Sorties : $e(P, Q)$.

1 Tirer aléatoirement $R \in \mathbb{G}_1 \setminus \{\mathcal{P}_\infty\}$;
2 $P' \leftarrow P + R$;
3 $h_1 \leftarrow e(P', Q)$;
4 $h_2 \leftarrow e(R, Q)$;
5 $h \leftarrow h_1 h_2^{-1}$;
6 **retourner** h ;

Algorithme 3.8 : Seconde contre-mesure de Page *et al.* [PV04]

Le principal inconvénient de ces contre-mesures est le surcoût. En effet, la première contre mesure ajoute le calcul de deux multiplications scalaire et d'une inversion modulo m dans $\mathbb{G}_3$ (m est l'ordre des groupes $\mathbb{G}_1$ et $\mathbb{G}_2$). Le calcul de $1/ab \mod m$ peut être évité en prenant a et b tels que $ab = 1 \mod m$. La seconde contre mesure nécessite un calcul supplémentaire du couplage e.

Blömer *et al.* [BGL13] proposent une amélioration de la seconde contre-mesure de Page *et al.* [PV04]. La configuration requise est $P \in \mathbb{G}_1$ secret. Cette amélioration exploite la relation d'équivalence sur lequel $\mathbb{G}_2$ est construit pour le couplage de Tate. On a $\mathbb{G}_2 = E(\mathbb{F}_{q^k})/[m]E(\mathbb{F}_{q^k})$, et quel que soit $R \in [r]E(\mathbb{F}_{q^k})$ où r est premier avec m alors $R + Q \sim Q$ et donc $e(P, R + Q) = e(P, Q)$. C'est une contre-mesure par masquage additif choisi.

Contre-mesures internes au couplage

Kim *et al.* [KTH+06] ont proposé l'utilisation de la troisième contre-mesure de Coron [Cor99]. En effet, il s'agit de bénéficier de la représentation des points en coordonnées projectives. Supposons que l'algorithme de Miller est implémenté en coordonnées projectives avec les calculs combinés addition/ligne et doublement/tangente (cf. équations 7.15 et 7.16). L'instruction 2 de l'algorithme 2.6, page 45, initialise le point T aux coordonnées de P avec

$$X_T \leftarrow x_P; \quad Y_T \leftarrow y_P \text{ et } Z_T \leftarrow 1. \tag{3.15}$$

Cette affectation sera remplacée par

$$\begin{array}{l} \lambda \text{ est tiré aléatoirement dans } \mathbb{F}_q^\star \\ X_T \leftarrow \lambda x_P; \quad Y_T \leftarrow \lambda y_P \text{ et } Z_T \leftarrow \lambda. \end{array} \tag{3.16}$$

La sortie de couplage avec randomisation est bien identique à celle sans la randomisation, car l'exponentiation finale envoie tous les éléments de $\mathbb{F}_q$ sur 1, ce qui est le cas du masque λ. Cette contre-mesure de randomisation des coordonnées est très peu coûteuse, elle n'ajoute que deux multiplications dans $\mathbb{F}_q$.

La méthode de protection proposée par Scott [Sco05] est assez similaire puisqu'elle se base sur le principe qu'il est possible de multiplier par un élément de $\mathbb{F}_q$ sans affecter le résultat. Il propose de masquer les calculs sensibles, c'est-à-dire les multiplications entre un opérande connu et un opérande secret, en appliquant la randomisation multiplicative par $\lambda \in \mathbb{F}_q^\star$.

La table 3.3 est un récapitulatif des contre-mesures de la littérature. Le paramètre **symétrie** permet de qualifier les contre-mesures selon le critère suivant : une contre-mesure est dite **symétrique** si elle s'applique aussi bien lorsque le point secret est P ou Q.

Table 3.3 – Résumé des contre-mesures de la littérature pour protéger l'implémentation de couplage en grande caractéristique

Contre-mesures	**Surcoût**	**Symétrie**	**Remarques**
Masque multiplicatif [PV04]	2 multiplications scalaires et 1 exponentiation dans $\mathbb{F}_{q^k}^\star$	Oui	L'exponentiation peut être évitée
Masque additif [PV04]	1 addition sur E, un couplage e et 1 inversion dans $\mathbb{F}_{q^k}^\star$	Pas totalement	La symétrie est facile à obtenir
Masque additif choisi [BGL13]	1 addition sur E	Non, P secret	Tirage du masque plus compliqué
Randomisation des coordonnées [KTH+06]	Dépend du choix des coordonnées (voir table 3.4)	Dépend du choix des coord.	Voir détails table 3.4
Randomisation multiplicative [Sco05]	N multiplications modulaires	Oui	N dépend du choix des coordonnées

La contre-mesure de randomisation des coordonnées de Kim *et al.* [KTH+06] a un coût qui dépend du choix du système de coordonnées et du point secret qu'il soit P ou Q.

Prenons l'exemple du choix de coordonnées affines-jacobiennes. L'algorithme de Miller s'écrit avec P et Q en coordonnées affines, tandis que T est en coordonnées jacobiennes. La contre-mesure initialise le point T avec P de façon randomisée, son coût est de $3M + C$. Le point T est donc différent à chaque exécution du couplage, l'objectif est bien rempli lorsque Q est secret : la contre-mesure vient masquer les données publiques (dépendant de P).

En revanche lorsque P est secret, c'est le secret (le point T) qui est randomisé alors que le but est de randomiser le point public Q. Il faut le passer en coordonnées jacobiennes, avec randomisation. Ainsi, le système de coordonnées est P et T en affine et Q en jacobiennes. Cette configuration n'est pas intéressante, les opérations sur T

seront effectuées en coordonnées affines, ce qui est coûteux. Ainsi, le point T est lui aussi représenté en coordonnées jacobiennes. Donc pour appliquer la contre-mesure de randomisation des coordonnées projectives sur le point Q public il est recommandé de passer en coordonnées complètement jacobiennes.

Table 3.4 – Détails du surcoût de la randomisation des coordonnées [KTH+06] par itérations de Miller

Système de coordonnées	P **secret**	Q **secret**
Projectives	$2M + C$	
Affines-Projectives		$2M + C$
Jacobiennes	$3M + C$	
Affine-jacobiennes		$3M + C$

3.4 Objectifs de la thèse

L'état de l'art sur les attaques par canaux auxiliaires contre les implémentations des algorithmes de couplage est principalement constitué de schémas théoriques. Les chemins d'attaques proposés sont souvent les mêmes, il s'agit de cibler une opération arithmétique (addition ou multiplication modulaire) pour retrouver une coordonnée du point secret.

La structure de l'algorithme de Miller fait que le choix de placer le secret dans P, le premier argument, ou bien dans Q, le second, est souvent discuté [WS06, EDFL09, BGL13]. Un des objectifs sera de clarifier la situation à ce niveau selon deux axes :

- Choix du secret (P ou Q).
- Choix du système de coordonnées.

Les attaques proposées dans la littérature n'ont pas de validation pratique, sauf celle d'Unterluggauer *et al.* [UW14], qui a été publiée après le début de cette thèse. Ce domaine de recherche récent manque donc d'études de cas réels. Nous complétons cette partie, avec notamment la caractérisation de la multiplication. Cela nous a permis de construire un outil efficace pour attaquer la multiplication modulaire. Cette caractérisation contient les aspects : recherche d'un modèle de fuite adapté, étude théorique des schémas d'attaques.

Comme nous l'avons évoqué au paragraphe *Contre-mesures internes au couplage* (page 78), la contre-mesure de randomisation des coordonnées est intéressante en termes de temps de calcul additionnel. Concernant sa sécurité, elle semble robuste, mais aucune preuve ne vient justifier ces propos. Comme la contre-mesure est intéressante du point de vue de son efficacité nous nous penchons plus en détail sur celle-ci afin de voir s'il n'y a pas des failles à exploiter. Dans un premier temps, nous cherchons à exploiter une attaque par profilage pour contrecarrer la randomisation multiplicative. Dans un second temps, nous l'utiliserons pour déjouer la randomisation des coordonnées. Nos recherches nous ont menées à trouver un point faible dans cette dernière. Nous construisons un schéma d'attaque basé sur les collisions afin de déjouer la contre-mesure. Puis dans un second temps, nous faisons une réalisation pratique de cette attaque.

Nous représentons l'état de l'art des attaques par le tableau 3.5. Nous identifions ces attaques selon les cas P ou Q secret et avec ou sans la contre-mesure de randomisation des coordonnées.

Table 3.5 – Résumé des attaques par canaux cachés

	Secret	P	
	Contre-mesure	Aucune	Randomisation des coordonnées
Attaques	CPA verticale	Une seule attaque pratique	
	Secret	Q	
	Contre-mesure	Aucune	Randomisation des coordonnées
Attaques	CPA verticale	Pas de validation pratique	

Code couleur : Attaque relevée dans la littérature Pas d'attaque identifiée

Chapitre 4

Amélioration des attaques par canaux auxiliaires contre les couplages

« Ce qui est mouillé ne craint pas la pluie »

Proverbe Grec

Sommaire

Nous avons vu dans le chapitre 3 l'état de l'art des attaques par canaux auxiliaires. La cible est une multiplication modulaire $ab \mod p$ avec un opérande connu et l'autre secret. Le paragraphe *Attaque de la multiplication modulaire* page 85 montre qu'il n'y a pas de véritable difficulté à faire une attaque lorsque a est secret et b connu ou bien l'inverse. L'attaque pratique la plus récente de l'état de l'art est celle de Unterluggauer *et al.*. Elle cible l'implémentation d'un couplage de Twisted Ate $e_{m,A}(Q,P)$ avec Q secret sur un cœur ARM Cortex-M0. Leur attaque fonctionne à partir de 1500 traces.

Dans ce chapitre, nous proposons et validons des travaux d'optimisation de l'attaque de la multiplication de mots machine de 32 bits. Le chapitre est décomposé en trois grandes sections. Nous commençons par rappeler le contexte des attaques contre les couplages et nous présentons l'attaque de la multiplication de mots machines en section 4.1. Nous étudions cette attaque d'un point de vue théorique en section 4.2. Puis, dans la section 4.3 nous détaillerons l'approche pratique avec nos résultats en environnement réel.

4.1 Contexte et présentation théorique des attaques

Le calcul sensible dans l'utilisation d'un couplage e en cryptographie est un calcul de $e(P,Q)$ avec :

- soit P est connu et Q est secret (**cas 1**),
- soit Q est connu et P est secret (**cas 2**).

Une telle situation est atteinte dans le déchiffrement de l'IBE (cf. *Déchiffrement*, page 55), le déchiffrement de l'ABE, et peut aussi être exhibée dans l'échange tripartie [Jou00] :

- L'attaquant a observé les échanges, il connaît donc $[a]P, [b]P, [c]P$ et P, car il est public.
- Il calcule $h = e([a]P, [c]P)$.
- Il retrouve Q tel que $e(P,Q) = h$ avec une attaque par canaux auxiliaires. C'est cette étape qui est étudiée ici.
- Il calcule la clef commune $K = e([b]P, Q)$.

4.1.1 Attaque de l'équation de la tangente

Nous avons vu, au travers de la section précédente, les principales contributions de la communauté autour des attaques par canaux auxiliaires contre les couplages. Techniquement, il s'agit de faire une CPA contre la multiplication modulaire entre deux opérandes, dont l'un dépend du point public en entrée du couplage et l'autre du point secret. L'attaquant cherche une telle opération dans l'algorithme de Miller (cf. algorithme 2.6). On se place dans le cas d'une implémentation combinée addition/ligne et doublement/tangente en **coordonnées mixtes affine-jacobienne** (cf. annexe C, page 171).

Cas 1 : P est connu et Q est secret

L'élimination du dénominateur simplifie l'équation de la tangente à son numérateur, qui est :

$$2Y_T Z_T^3 y_Q - 2Y_T^2 - I(x_Q Z_T^2 - X_T).$$

Les coordonnées attaquées sont x_Q et y_Q, elles sont respectivement impliquées dans les multiplications $x_Q(Z_T^2)$ et $(2Y_T Z_T^3)y_Q$. Ces coordonnées sont attaquées par une CPA contre la multiplication modulaire. Les coordonnées du point $T = (X_T : Y_T : Z_T)$ sont connues puisqu'elles sont initialisées à celles du point connu P.

Remarque 7. *Plusieurs remarques à propos de cette attaque :*

- *Si $Q \in \mathbb{G}_2 \subset \mathbb{F}_{q^k}$ il faudra retrouver chacune des composantes de x_Q, et donc faire k attaques.*
- *Lorsque x_Q est retrouvé, l'attaquant peut utiliser l'équation de la courbe elliptique E pour retrouver y_Q. Il retrouve deux candidats pour y_Q notés y_1 et y_2, et il les teste jusqu'à retrouver le même résultat du couplage qu'avec le secret y_Q.*

Cas 2 : Q est connu et P est secret

Considérons l'équation de la tangente lors de la première itération de la boucle de Miller, les coordonnées de T sont $T = (X_T : Y_T : Z_T) = (x_P : y_P : 1)$. Le numérateur de l'équation de la tangente s'écrit donc :

$$2y_P y_Q - 2y_P^2 - I(x_Q - x_P).$$

L'attaquant est alors en mesure de faire une attaque contre la multiplication modulaire $y_P y_Q$ pour retrouver y_P, il utilise ensuite l'équation de la courbe elliptique pour retrouver x_P. L'équation à résoudre étant de degré trois, il retrouve plusieurs candidats x_1, x_2, x_3, il peut les tester jusqu'à retrouver le même résultat du couplage qu'avec le secret x_P.

4.1.2 Attaque de la multiplication modulaire

Les attaques par canaux auxiliaires contre les couplages passent par l'attaque de la tangente, qui elle-même doit passer par une attaque de la multiplication modulaire. Cette dernière doit être attaquée en fonction de la taille de l'architecture du dispositif.

La stratégie pour cibler un algorithme de couplage consiste à chercher une opération qui implique des données connues et secrètes. Pour trouver une telle opération, il suffit de descendre dans l'architecture de l'implémentation du plus haut au plus bas niveau. Ce raisonnement est justifié par la traditionnelle pyramide qui illustre l'architecture de l'implémentation, la figure 4.1 présente cette pyramide dans le cadre des couplages.

Nous allons donc décrire les détails de l'attaque contre la multiplication modulaire. La multiplication modulaire $a \times b \mod p$ ciblée est la multiplication multiprécision CIOS de Montgomery (algorithme 4.1). Elle est souvent choisie pour les applications cryptographiques pour son coût, et, finalement, ce choix n'affecte pas la stratégie d'attaque.

Il faut identifier l'opération qui implique à la fois a et b. Dans la méthode CIOS, il s'agit du calcul $(uv) \leftarrow c_j + \mathbf{a_j b_i} + u$ (algorithme 4.1, instruction 5). Il y a deux cas à étudier :

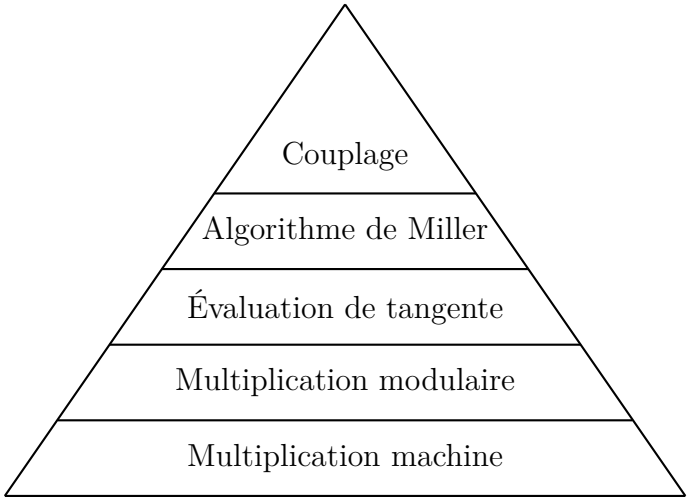


Figure 4.1 – Architecture des couplages

- soit b est connu et a est secret (**cas 1**),
- soit a est connu et b est secret (**cas 2**).

Le choix de l'algorithme de la multiplication modulaire (SOS, CIOS, FIOS, FIPS, CIHS [KAK96]) ne change pas la cible ; elle est toujours la multiplication machine $a_j b_i$. Le calcul des variables intermédiaires, par exemple c_j et u doit être adapté. Nous détaillons le cas de l'algorithme CIOS.

Cas 1 : b est connu et a est secret

La première itération de l'algorithme CIOS (cf. algorithme 4.1) correspond aux indices $i = j = 0$.

L'instruction 5 s'écrit $(uv) \leftarrow a_0 b_0$, la multiplication de deux mots machine s'attaque comme dans [WS06, EDFL09, BGL13, UW14]. Il s'agit de faire une CPA comme sur un AES en ciblant le résultat de la multiplication des mots machine plutôt que la sortie du *SubBytes*. Nous reviendrons plus largement sur cette attaque, car nous avons effectué une phase de caractérisation pour finalement proposer une attaque efficace contre la multiplication de mots machine (section 4.2.2). L'attaque permet de retrouver a_0. Le mot ciblé est maintenant a_1, il intervient pour $i = 0$ et $j = 1$, l'instruction 5 s'écrit $(uv) \leftarrow a_1 b_0 + u$, où le u est la retenue de la précédente opération, c'est-à-dire $(uv) \leftarrow a_0 b_0$, donc u est connu. Il est alors tout à fait possible de faire une CPA pour retrouver a_1. Les autres mots de $a = (a_{\mathfrak{n}-1} \dots a_0)_{\overline{\mathfrak{w}}}$ sont ciblés de la même manière.

Cas 2 : a est connu et b est secret

Le cas 2 cible les mots b_i, cela nécessite une étude plus approfondie de l'algorithme, car, l'indice i est celui de la boucle principale. La première itération ($i = j = 0$) produit l'instruction $(uv) \leftarrow a_0 b_0$, qui permet de retrouver b_0 comme précédemment. Afin d'identifier la retenue u à travers les itérations, elle sera notée u_{00}.

La cible est le mot b_1. Il est impliqué dans l'instruction $(uv) \leftarrow c_0 + a_0 b_1$. le mot a_0 est connu, il reste à vérifier que l'on connaît bien c_0.

Ce mot c_0 provient de l'instruction 15 de l'algorithme 4.1 pour $i = 0$ et $j = 1$. D'abord, il y a le calcul $(uv) \leftarrow c_1 + mp_1 + u$, puis l'instruction 15 $c_0 \leftarrow v$. Nous

Entrées : Le modulo $p = (p_{\mathfrak{N}-1} \dots p_0)_{\overline{\mathfrak{W}}}$ premier avec $\overline{\mathfrak{W}}$, $\mathfrak{R} = \overline{\mathfrak{W}}^{\mathfrak{N}}$, p' tel que $\mathfrak{R}\mathfrak{R}^{-1} - pp' = 1$ et deux entiers $a = (a_{\mathfrak{N}-1} \dots a_0)_{\overline{\mathfrak{W}}}$ et $b = (b_{\mathfrak{N}-1} \dots b_0)_{\overline{\mathfrak{W}}}$.
Sorties : L'unique entier $c = (c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}}$ tel que $c = \text{Redc}(ab) = (ab\mathfrak{R}^{-1}) \mod p$.

1 $c \leftarrow 0$;
2 **pour** $i = 0$ **à** $\mathfrak{N} - 1$ **faire**
3 $\quad u \leftarrow 0$;
4 $\quad$ **pour** $j = 0$ **à** $\mathfrak{N} - 1$ **faire**
5 $\quad\quad (uv) \leftarrow c_j + a_j b_i + u$;
6 $\quad\quad c_j \leftarrow v$;
7 $\quad$ **fin**
8 $\quad (uv) \leftarrow c_{\mathfrak{N}} + u$;
9 $\quad c_{\mathfrak{N}} \leftarrow v$;
10 $\quad c_{\mathfrak{N}+1} \leftarrow u$;
11 $\quad m \leftarrow c_0 p'_0 \mod \overline{\mathfrak{W}}$;
12 $\quad (uv) \leftarrow c_0 + m p_0$;
13 $\quad$ **pour** $j = 1$ **à** $\mathfrak{N} - 1$ **faire**
14 $\quad\quad (uv) \leftarrow c_j + m p_j + u$;
15 $\quad\quad c_{j-1} \leftarrow v$;
16 $\quad$ **fin**
17 $\quad (uv) \leftarrow c_{\mathfrak{N}} + u$;
18 $\quad c_{\mathfrak{N}-1} \leftarrow v$;
19 $\quad c_{\mathfrak{N}} \leftarrow c_{\mathfrak{N}+1} + u$;
20 **fin**
21 **si** $(c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}} < p$ **alors**
22 $\quad c \leftarrow (c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}}$;
23 **sinon**
24 $\quad c \leftarrow (c_{\mathfrak{N}-1} \dots c_0)_{\overline{\mathfrak{W}}} - p$;
25 **fin**
26 **retourner** c ;

Algorithme 4.1 : Multiplication modulaire de Montgomery CIOS (rappel)

devons alors vérifier que les mots c_1, m et u sont connus (p_1 est évidemment connu). La table 4.1 montre la dépendance des variables intermédiaires, et comment on récupère à c_1, m et u.

Table 4.1 – Origines des variables intermédiaires pour calculer c_0

Variables	Origines			
	Lignes de l'algorithme 4.1	Opérations	Variables intermédiaires	
			Connues	Inconnues
c_1	6	$a_1 b_0 + u_{00}$	a_1, b_0, u_{00}	–
u	12	$c_0 + m p_0$	–	c_0, m
m	11	$c_0 p'_0 \mod \overline{\mathfrak{W}}$	$p'_0, \overline{\mathfrak{W}}$	c_0
c_0	6	$a_0 b_0$	a_0, b_0	–

La table 4.1 montre qu'il est possible de remonter à la valeur du mot c_1. Ce mot est essentiel pour remonter à b_1. En effet, il est maintenant possible de cibler b_1 dans l'instruction $(uv) \leftarrow c_0 + a_0 b_1$ pour retrouver b_1.

4.2 Études théoriques de la multiplication de mots machine

Le but de ces études est d'aborder l'aspect théorique de l'attaque de la multiplication de deux mots machine. Le taux de succès d'une attaque est simplement la probabilité que celle-ci retrouve la clef secrète.

Le contexte est le suivant : nous ciblons une multiplication, sur 16 bits dans un premier temps, en nous focalisant d'abord sur l'octet de poids faible. Cette première sous-attaque donnera des candidats pour la suivante, nous noterons α le nombre de candidats retenus. La méthode sera étendue sur les 32 bits du mot machine à la section *Enchaînement des sous-attaques*, page 105.

4.2.1 Formalisation du taux de succès

Le succès d'une telle attaque s'obtient sous deux conditions :

1. La première sous-attaque classe la bonne sous-clef $k^\star$ parmi les α premières. La sous-clef fait référence aux 8 bits de poids faibles de la clef.

2. La seconde sous-attaque renvoie la bonne clef secrète.

Dans la suite $\widehat{\mathfrak{D}}$ désignera un distingueur, c'est-à-dire une fonction qui prend une clef candidate et qui renvoie son classement (le rang 1 étant celui de la bonne clef secrète).

Le taux de succès de la première sous-attaque est aussi connu comme le taux de succès d'ordre α. Il s'écrit :

$$\mathfrak{T}_1(\widehat{\widehat{\mathfrak{D}}}) = \mathbb{P}\left(\widehat{\mathfrak{D}}\left(k^\star\right) \geq \widehat{\mathfrak{D}}\left(\arg\max_k {}^{(\alpha)}\widehat{\mathfrak{D}}\left(k\right)\right)\right). \tag{4.1}$$

Traditionnellement, la notation $\arg\max_k \vec{v}$ désigne l'indice k pour lequel le vecteur $\vec{v}(k)$ est maximal. L'exposant (α) permet de signifier que $\arg\max_k {}^{(\alpha)}\vec{v}$ désigne l'indice k pour lequel $\vec{v}(k)$ est la $\alpha-$ième plus grande valeur.

Dans la suite, nous allons établir une borne inférieure au taux de succès de la première sous-attaque en fonction de α.

Lomne *et al.* [LPR+14] s'attardent sur les attaques d'ordre supérieur. Leur estimation du taux de succès est intéressante. Il s'agit, pour un distingueur additif (la CPA par exemple) de considérer le vecteur de score $\overrightarrow{d} = (d_k)_{k\in\mathcal{K}}$. Plus précisément, une valeur est attribuée pour chaque hypothèse de clef. Le vecteur de comparaison $\overrightarrow{c} = (c_k)_{k\in\mathcal{K}\setminus\{k^\star\}}$ (de taille $\#\mathcal{K}-1$) est défini par :

$$c_k = d_k^\star - d_k. \tag{4.2}$$

L'attaque est donc un succès si les composantes du vecteur de comparaison $\overrightarrow{c}$ sont toutes positives. On note $\overrightarrow{x}$ les messages utilisés, $\overrightarrow{0}$ le vecteur nul, $\overrightarrow{\infty}$ le vecteur infini $\overrightarrow{\infty} = (\infty, \infty, \ldots, \infty)$ et

$$\overrightarrow{\Phi}_{\overrightarrow{\mu},\overrightarrow{\Sigma}}(\overrightarrow{u}, \overrightarrow{v}) = \int_{u_1}^{v_1} \int_{u_2}^{v_2} \ldots \int_{u_T}^{v_T} \phi_{\overrightarrow{\mu},\overrightarrow{\Sigma}}(\overrightarrow{x}) d\overrightarrow{x}. \tag{4.3}$$

La fonction de distribution cumulée (*Cumulative Distribution Function* en anglais) de la distribution gaussienne multivariée de dimension T est :

$$\phi_{\overrightarrow{\mu},\overrightarrow{\Sigma}}(\overrightarrow{x}) = \frac{1}{\sqrt{(2\pi)^T |\overrightarrow{\Sigma}|}} \exp\left(-\frac{1}{2}(\overrightarrow{x} - \overrightarrow{\mu})' \overrightarrow{\Sigma}^{-1} (\overrightarrow{x} - \overrightarrow{\mu})\right). \tag{4.4}$$

Les auteurs de [LPR+14] donnent l'approximation :

$$\mathfrak{T}^d_{\overrightarrow{x},k^\star} = \mathbb{P}\left(\overrightarrow{c} > \overrightarrow{0}\right) \approx \overrightarrow{\Phi}_{\overrightarrow{\mu},\overrightarrow{\Sigma}}(\overrightarrow{0}, \overrightarrow{\infty}), \tag{4.5}$$

Pour appliquer cette étude à notre cas des sous-attaques successives, en premier lieu nous allons essayer de calculer le taux de succès de la première sous-attaque. Si nous reprenons le principe du vecteur de comparaison, nous cherchons à ce que la bonne clef soit, par exemple, parmi les 2 meilleures (pour l'exemple $\alpha = 2$). Autrement dit, une valeur de c_k peut être négative. Cela se traduit par l'équation 4.6, avec les notations $\overrightarrow{m_i} = (0, \ldots, 0, \underbrace{-\infty}_{\text{rang } i}, 0, \ldots, 0)$ et $\overrightarrow{M_i} = (\infty, \ldots, \infty, \underbrace{0}_{\text{rang } i}, \infty, \ldots, \infty)$.

$$\mathfrak{T}^{d,2}_{\overrightarrow{x},k^\star} = \mathbb{P}\left(\cup_i \left\{\overrightarrow{m_i} \leq \overrightarrow{c} < \overrightarrow{M_i}\right\}\right). \tag{4.6}$$

Comme les événements de probabilités de l'équation 4.6 sont indépendants (répartition uniforme des clefs) il suffit alors de calculer la somme :

$$\mathfrak{T}^{d,2}_{\overrightarrow{x},k^\star} \approx \sum_i \overrightarrow{\Phi}_{\overrightarrow{\mu},\overrightarrow{\Sigma}}(\overrightarrow{m_i}, \overrightarrow{M_i}). \tag{4.7}$$

Par la suite, les calculs doivent être étendus pour un α donné. En effet, nous cherchons à ce que la bonne clef $k^\star$ soit dans les α meilleures. Pour $\alpha = 8$ par exemple, cela signifie que nous devons énumérer tous les sous-ensembles de $\mathcal{K} = \{0, \ldots, 256\}$ de cardinal $1, 2, \ldots, 7$, ce qui coûte trop cher. En effet, le nombre de sous-ensembles est de :

$$\binom{256}{1} + \binom{256}{2} + \cdots + \binom{256}{8} = 423203101008288 > 2^{48}.$$

Nous utilisons coefficient de confusion [FLD12] pour minorer l'équation 4.7. Le coefficient de confusion est une métrique qui relate la confusion entre clefs hypothétiques différentes. Autrement dit, pour une opération ciblée (par exemple le *SubBytes* de l'AES) le coefficient de confusion permet de mesurer l'écart entre deux entrées, et donc deux clefs :

$$\begin{array}{rccl} \kappa : & \mathcal{K} \times \mathcal{K} & \to & \mathbb{R} \\ & (k_1, k_2) & \mapsto & \kappa(k_1, k_2) \text{ pour } k_1 \neq k_2. \end{array} \tag{4.8}$$

Le calcul du coefficient de confusion entre la clef $k^\star$ et les autres clefs permet d'identifier les clefs « similaires » à $k^\star$, et ainsi de prédire les clefs qui seront les mieux classées par un distingueur. La notion de clefs similaires n'est pas formalisée, il s'agit d'un abus de langage pour désigner des clefs qui ont par exemple une distance de Hamming petite, un poids de Hamming proche, ou encore si l'une des clefs est un shift (décalage binaire à gauche ou à droite) de la seconde.

Concernant l'évolution de l'équation 4.7, nous nous servons du coefficient de confusion pour limiter le nombre d'intégrales à calculer dans la formule 4.3. En effet, si les coefficients de confusion indiquent les clefs les plus probables, alors il n'y a plus qu'à énumérer les sous-ensembles nécessaires. Au lieu de tester tous les sous-ensembles de taille 8 parmi 256, on ne teste que ceux avec les 8 meilleures clefs associées aux meilleurs coefficients de confusion.

En reprenant les notations précédentes, on écrit :

Lemme 3. *Si les plus grandes valeurs du distingueur sont obtenues pour les meilleures clefs données par le coefficient de confusion alors :*

$$\mathbb{P}\left(\overrightarrow{m_{\underset{k\neq k^\star}{argmin}\ \kappa}} \leq \overrightarrow{c} \leq \overrightarrow{M_{\underset{k\neq k^\star}{argmin}\ \kappa}}\right) \leq \mathbb{P}\left(\overrightarrow{m_i} \leq \overrightarrow{c} < \overrightarrow{M_i}\right), \forall i. \tag{4.9}$$

Démonstration. On va réorganiser les valeurs du distingueur pour les différentes clefs, c'est-à-dire : $d_1 \geq d_2 \geq \ldots \geq d_{256}$. Sous l'hypothèse du lemme on a $d_1 = d_{\underset{k\neq k^\star}{\text{argmin}}\ \kappa}$. Or $\overrightarrow{m_{\underset{k\neq k^\star}{\text{argmin}}\ \kappa}} \leq \overrightarrow{c} \leq \overrightarrow{M_{\underset{k\neq k^\star}{\text{argmin}}\ \kappa}}$ signifie :

$$\begin{cases} c_1 & < & 0 \\ c_2 & > & 0 \\ & \vdots & \\ c_{256} & > & 0 \end{cases}. \tag{4.10}$$

D'après la définition de $\overrightarrow{c}$, l'équation 4.10 est équivalente à :

$$\begin{cases} d_k^\star & < & d_1 \\ d_k^\star & > & d_2 \\ & \vdots & \\ d_k^\star & > & d_{256} \end{cases}. \tag{4.11}$$

Ainsi, l'événement $\overrightarrow{m_1} \leq \overrightarrow{c} \leq \overrightarrow{M_1}$ est moins probable que $\overrightarrow{m_2} \leq \overrightarrow{c} \leq \overrightarrow{M_2}$ etc. D'où

$$P\left[\overrightarrow{m_1} \leq \overrightarrow{c} \leq \overrightarrow{M_1}\right] \leq P\left[\overrightarrow{m_2} \leq \overrightarrow{c} \leq \overrightarrow{M_2}\right] \leq \ldots \leq P\left[\overrightarrow{m_{256}} \leq \overrightarrow{c} \leq \overrightarrow{M_{256}}\right] \tag{4.12}$$

□

On déduit une minoration du taux de succès $\mathfrak{T}^{d,2}_{\overrightarrow{x},k^\star}$ avec la proposition 13.

Proposition 13. *Sous l'hypothèse du Lemme 3 on a :*

$$\mathfrak{T}^{d,2}_{\overrightarrow{x},k^\star} = \sum_i \mathbb{P}\left(\overrightarrow{m_i} \leq \overrightarrow{c} < \overrightarrow{M_i}\right) \geq \#\mathcal{K} \cdot \mathbb{P}\left(\overrightarrow{m_{\underset{k\neq k^\star}{argmin}\ \kappa}} \leq \overrightarrow{c} \leq \overrightarrow{M_{\underset{k\neq k^\star}{argmin}\ \kappa}}\right). \tag{4.13}$$

Démonstration. D'après l'équation 4.6 nous avons

$$\mathfrak{T}^{d,2}_{\overrightarrow{x},k^\star} = \mathbb{P}\left(\cup_i \left\{\overrightarrow{m_i} \leq \overrightarrow{c} < \overrightarrow{M_i}\right\}\right).$$

L'indépendance totale des événements permet d'écrire

$$\mathfrak{T}^{d,2}_{\overrightarrow{x},k^\star} = \sum_i \mathbb{P}\left(\overrightarrow{m_i} \leq \overrightarrow{c} < \overrightarrow{M_i}\right).$$

Nous appliquons le lemme 3 :

$$\mathfrak{T}^{d,2}_{\overrightarrow{x},k^\star} \geq \sum_{i\in\mathcal{K}} \mathbb{P}\left(\overrightarrow{m_{\underset{k\neq k^\star}{\operatorname{argmin}}\,\kappa}} \leq \overrightarrow{c} \leq \overrightarrow{M_{\underset{k\neq k^\star}{\operatorname{argmin}}\,\kappa}}\right).$$

□

On généralise le résultat de la proposition 13 pour un α quelconque :

Théorème 8. *Si les plus grandes valeurs du distingueur sont obtenues pour les meilleures clefs données par le coefficient de confusion alors :*

$$\mathfrak{T}^{d,\alpha}_{\overrightarrow{x},k^\star} \geq \#\mathcal{K}\cdot\mathbb{P}\left(\overrightarrow{m_{\underset{k\neq k^\star}{argmins^{(\alpha-1)}}\kappa}} \leq \overrightarrow{c} \leq \overrightarrow{M_{\underset{k\neq k^\star}{argmins^{(\alpha-1)}}\kappa}}\right), \tag{4.14}$$

avec

- $\underset{k\neq k^\star}{argmins^{(\alpha-1)}}\kappa$ *les* $\alpha-1$*-ème arguments minimums de* κ*,*
- $\overrightarrow{m_{\underset{k\neq k^\star}{argmins^{(\alpha-1)}}\kappa}}$ *est le vecteur qui contient* $-\infty$ *pour les indices* $\underset{k\neq k^\star}{argmins^{(\alpha-1)}}\kappa$ *et* 0 *ailleurs,*
- $\overrightarrow{M_{\underset{k\neq k^\star}{argmins^{(\alpha-1)}}\kappa}}$ *est le vecteur qui contient* 0 *pour les indices* $\underset{k\neq k^\star}{argmins^{(\alpha-1)}}\kappa$ *et* $+\infty$ *pour les autres.*

Démonstration. La preuve se fait par récurrence sur α. L'initialisation de la récurrence pour $\alpha = 2$ est vérifiée par la proposition 13.

On suppose l'hypothèse vraie pour α, c'est-à-dire :

$$\mathfrak{T}^{d,\alpha}_{\overrightarrow{x},k^\star} \geq \#\mathcal{K}\cdot\mathbb{P}\left(\overrightarrow{m_{\underset{k\neq k^\star}{\operatorname{argmins}^{(\alpha-1)}}\kappa}} \leq \overrightarrow{c} \leq \overrightarrow{M_{\underset{k\neq k^\star}{\operatorname{argmins}^{(\alpha-1)}}\kappa}}\right).$$

Le taux de succès à l'ordre $\alpha+1$ s'écrit :

$$\mathfrak{T}^{d,\alpha+1}_{\overrightarrow{x},k^\star} = \mathfrak{T}^{d,\alpha}_{\overrightarrow{x},k^\star} + \mathbb{P}\left(\underset{k}{\operatorname{rang}}(\overrightarrow{d}) = \alpha+1\right),$$

avec la notation $\underset{k}{\operatorname{rang}}(\overrightarrow{d})$ qui indique le rang de la clef k dans $\overrightarrow{d}$. L'hypothèse de récurrence permet d'écrire :

$$\mathfrak{T}^{d,\alpha+1}_{\overrightarrow{x},k^\star} \geq \mathbb{P}\left(\underbrace{\overrightarrow{m_{\underset{k\neq k^\star}{\operatorname{argmins}^{(\alpha-1)}}\kappa}} \leq \overrightarrow{c} \leq \overrightarrow{M_{\underset{k\neq k^\star}{\operatorname{argmins}^{(\alpha-1)}}\kappa}}}_{p_1} \text{ OU } \underbrace{\underset{k^\star}{\operatorname{rang}}(\overrightarrow{d}) = \alpha+1}_{p_2}\right) = \mathbb{P}_{\alpha+1}$$

En supposant, après éventuelle réorganisation, que les valeurs d_k soient triées, p_1 est l'union des événements :

$$\left\{\begin{array}{l} d_k^\star < d_0 \\ d_k^\star > d_1 \\ d_k^\star > d_2 \\ \vdots \\ d_k^\star > d_{255} \end{array}\right. \text{ ou } \left\{\begin{array}{l} d_k^\star < d_0 \\ d_k^\star < d_1 \\ d_k^\star > d_2 \\ \vdots \\ d_k^\star > d_{255} \end{array}\right. \text{ ou} \cdots \text{ou } \left\{\begin{array}{l} d_k^\star < d_0 \\ d_k^\star < d_1 \\ \vdots \\ d_k^\star < d_{\alpha-1} \\ d_k^\star > d_\alpha \\ \vdots \\ d_k^\star > d_{255} \end{array}\right. .$$

Tandis que p_2 s'écrit :

$$\left\{\begin{array}{l} d_k^\star < d_0 \\ d_k^\star < d_1 \\ \vdots \\ d_k^\star < d_\alpha \\ d_k^\star > d_{\alpha+1} \\ \vdots \\ d_k^\star > d_{255} \end{array}\right. .$$

Et donc

$$\mathbb{P}_{\alpha+1} = \mathbb{P}\left(\overrightarrow{m_{\underset{k\neq k^\star}{\text{argmins}^{(\alpha)}\kappa}}} \leq \overrightarrow{c} \leq \overrightarrow{M_{\underset{k\neq k^\star}{\text{argmins}^{(\alpha)}\kappa}}}\right).$$

□

4.2.2 Méthodes diviser pour mieux régner

Nous venons de voir une minoration du taux de succès pour la première sous-attaque. Maintenant, nous étudions deux schémas pour enchaîner les sous-attaques intermédiaires. En effet, les α candidats renvoyés par la première sous-attaque permettent de créer les hypothèses pour l'octet ciblé suivant. Nous analysons l'enchaînement de l'attaque et donc l'enchaînement des taux de succès.

Méthode basique

La figure 4.2 schématise le premier schéma d'attaque. C'est le schéma basique, où, à partir des candidats renvoyés par la première sous-attaque (ensemble $\mathcal{K}_1$) on crée les hypothèses pour cibler l'octet de poids fort (ensemble $\mathcal{L}$). Ensuite, nous faisons une attaque sur toutes ces hypothèses pour retourner la clef candidate $\widehat{k}$.

Définition des ensembles de la Figure 4.2 :

- $\mathcal{K}_0$ est l'ensemble des $2^8 = 256$ hypothèses de clefs pour les 8 bits de poids faibles.
- $\mathcal{K}_1$ est l'ensemble des α candidats retenus après la première sous-attaque.
- $\mathcal{L}$: pour chacun des éléments de $\mathcal{K}_1$ on créé 256 hypothèses de clefs (énumération des 8 bits de poids fort), on note ces ensembles $\mathcal{L}_i$, donc $\mathcal{L}$ est l'ensemble $\mathcal{L} = \cup_{i=1}^{\alpha}\mathcal{L}_i$, de cardinal $\alpha \times 256$.

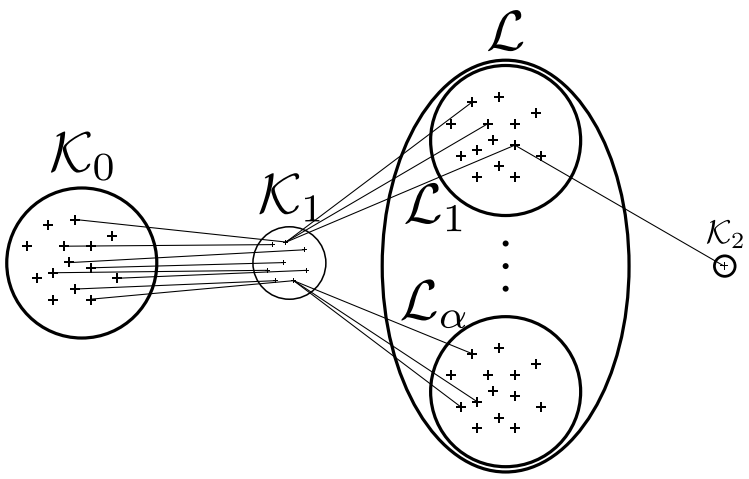


Figure 4.2 – Schéma d'attaque basique

- $\mathcal{K}_2$ est l'ensemble des clefs renvoyées par l'attaque avec les hypothèses de $\mathcal{L}$. Puisque l'attaque renvoie seulement la clef hypothétique de $\mathcal{L}$ avec la meilleure corrélation, $\mathcal{K}_2$ est un singleton.

Nous cherchons simplement le taux de succès après la seconde sous-attaque :

$$\mathfrak{T}_{\overline{2},\underline{1}} = \mathbb{P}\left(\forall k \in \mathcal{L} \setminus \{k^\star_{\mathcal{L}}\}, \quad \widehat{\mathfrak{D}}\left(k^\star_{\mathcal{L}}\right) > \widehat{\mathfrak{D}}\left(k\right) \text{ ET } k^\star_1 \in \mathcal{L}\right). \tag{4.15}$$

$\mathcal{K}_1$ est l'ensemble (de cardinal α) des clefs créées à partir de la première sous-attaque. L'appartenance de la bonne clef $k^\star_1$ à $\mathcal{L}$ correspond au taux de succès d'ordre α de la première sous-attaque. Le taux de succès s'écrit alors :

$$\mathfrak{T}_{\overline{2},\underline{1}} = \underbrace{\mathbb{P}\left(\forall k \in \mathcal{L} \setminus \{k^\star_{\mathcal{L}}\}, \quad \widehat{\mathfrak{D}}\left(k^\star_{\mathcal{L}}\right) > \widehat{\mathfrak{D}}\left(k\right)\right)}_{\mathfrak{T} \text{ classique}} \cdot \mathfrak{T}^{\alpha}_{\overline{1}} = \overline{\mathfrak{T}} \cdot \mathfrak{T}^{\alpha}_{\overline{1}}. \tag{4.16}$$

Méthode avancée

Nous proposons de rajouter une sous-attaque intermédiaire afin de faire une présélection des clefs de $\mathcal{L}$. L'attaque est schématisée par la figure 4.3.

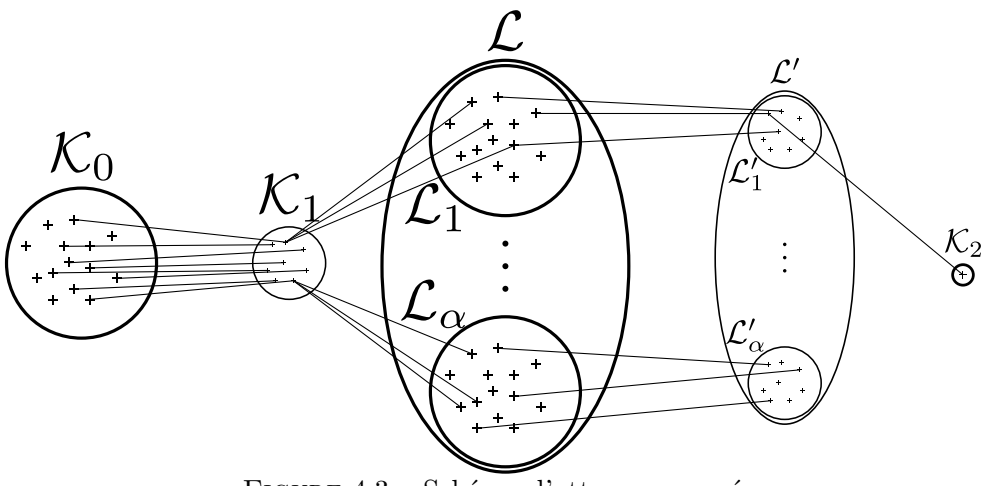


Figure 4.3 – Schéma d'attaque avancé

Cette méthode ajoute α sous-attaques intermédiaires. Pour chaque $\mathcal{L}_i$ (pour rappel, $\#\mathcal{L}_i = 2^8 = 256$ hypothèses de clefs) une sous-attaque est effectuée pour finalement

présélectionner α candidats, qui sont mis dans $\mathcal{L}'_i$. L'ensemble $\mathcal{L}'$ contient ainsi $\alpha \times \alpha$ clefs, la dernière attaque utilise ces clefs hypothétiques pour en renvoyer une seule (par construction de l'attaque $\mathcal{K}_2$ est un singleton).

La notation $k^\star_{\mathcal{L}'}$ représente la bonne clef sur $8+8$ bits (c'est la même que $k^\star_{\mathcal{L}}$). L'expression du taux de succès pour cette méthode est donnée par :

$$\begin{array}{rcl} \mathfrak{T}_{\overline{2},\underline{2}} & = & \mathbb{P}\left(\forall k \in \mathcal{L}' \setminus \{k^\star_{\mathcal{L}'}\}, \quad \widehat{\mathfrak{D}}\left(k^\star_{\mathcal{L}'}\right) > \widehat{\mathfrak{D}}\left(k\right) \text{ ET } k^\star_{\mathcal{L}'} \in \mathcal{L}'\right) \\ & = & \underbrace{\mathbb{P}\left(\forall k \in \mathcal{L}' \setminus \{k^\star_{\mathcal{L}'}\}, \quad \widehat{\mathfrak{D}}\left(k^\star_{\mathcal{L}'}\right) > \widehat{\mathfrak{D}}\left(k\right)\right)}_{\text{Taux de succès classique noté } \overline{\mathfrak{T}}} \cdot \mathbb{P}\left(k^\star_{\mathcal{L}'} \in \mathcal{L}'\right) \\ & \underset{\text{indép.}}{=} & \overline{\mathfrak{T}} \cdot \sum_{i=1}^{\alpha} \left(\mathfrak{T}^{\alpha}_{\mathcal{L}_i} \cdot \mathbb{P}\left(k^\star \in \mathcal{K}_1\right)\right) \\ & = & \overline{\mathfrak{T}} \cdot \alpha \cdot \mathfrak{T}^{\alpha}_{\overline{1}} \cdot \sum_{i=1}^{\alpha} \mathfrak{T}^{\alpha}_{\overline{\mathcal{L}_i}}, \end{array} \tag{4.17}$$

où $\mathfrak{T}^{\alpha}_{\overline{\mathcal{L}_i}}$ est la probabilité que l'attaque intermédiaire i classe la bonne clef dans les α meilleures. C'est-à-dire que si la bonne clef est dans $\mathcal{L}_i$, $\mathfrak{T}^{\alpha}_{\overline{\mathcal{L}_i}}$ est la probabilité que la bonne clef soit identifiée par l'attaque dans les α meilleures, à savoir, classée dans $\mathcal{L}'_i$.

Comparaison des méthodes d'enchaînement d'attaques

Nous pouvons comparer les méthodes d'enchaînement des sous-attaques à l'aide des taux de succès que l'on vient d'introduire. Nous écrivons d'abord le lemme suivant :

Lemme 4. *Soit $\alpha \geq 2$, si*

$$\min_{i=1,\ldots,\alpha} \mathbb{E}\left[\underset{k^\star_{\mathcal{L}'_i}}{rang}(\widehat{\mathfrak{D}})\right] < \alpha \tag{4.18}$$

alors

$$\max_{i=1,\ldots,\alpha} \left(1 - \frac{\mathbb{E}\left[\underset{k^\star_{\mathcal{L}'_i}}{rang}(\widehat{\mathfrak{D}})\right]}{\alpha+1}\right) > \frac{1}{\alpha^2} \tag{4.19}$$

Démonstration. Pour $\alpha \geq 2$ on a

$$\min E < \alpha < \frac{(\alpha+1)^2(\alpha-1)}{\alpha^2}$$

donc

$$1 - \frac{\min E}{\alpha+1} > \frac{1}{\alpha^2}$$

□

Le théorème permet de conclure quant à la comparaison des méthodes d'enchaînement des sous-attaques, toutefois, nous devons poser une hypothèse supplémentaire.

Théorème 9. *Sous l'hypothèse :*

$$\min_{i=1,\ldots,\alpha} \mathbb{E}\left[\underset{k^\star_{\mathcal{L}'_i}}{rang}(\widehat{\mathfrak{D}})\right] < \alpha \tag{4.20}$$

la méthode avancée est plus performante que la méthode basique, c'est-à-dire :

$$\mathfrak{T}_{\overline{2},\underline{2}} > \mathfrak{T}_{\overline{2},\underline{1}} \tag{4.21}$$

Démonstration. Réécrivons l'inégalité à démontrer :

$$\begin{array}{rcl} & \mathfrak{T}_{\overline{2},\underline{2}} & \geq \mathfrak{T}_{\overline{2},\underline{1}} \\ \iff & \overline{\mathfrak{T}} \cdot \alpha \cdot \mathfrak{T}_{\overline{1}}^{\alpha} \cdot \sum_{i=1}^{\alpha} \mathfrak{T}_{\mathcal{L}_i}^{\alpha} & \geq \overline{\mathfrak{T}} \cdot \mathfrak{T}_{\overline{1}}^{\alpha} \\ \iff & \alpha \sum_{i=1}^{\alpha} \underbrace{\mathbb{P}\left(\underset{k^{\star}_{\mathcal{L}'_i}}{\mathrm{rank}}(\widehat{\mathfrak{D}}) < \alpha + 1\right)}_{p_i} & \geq 1 \end{array}$$

Par ailleurs, on a :

$$\alpha \sum_{i=1}^{\alpha} p_i \geq \alpha^2 \max_i p_i \underset{\text{Markov}}{\geq} \alpha^2 \max_{i=1,\ldots,\alpha} \left(1 - \frac{\mathbb{E}\left[\underset{k^{\star}_{\mathcal{L}'_i}}{\mathrm{rank}}(\widehat{\mathfrak{D}})\right]}{\alpha + 1}\right).$$

Grâce au lemme 4, on obtient :

$$\alpha \sum_{i=1}^{\alpha} p_i \geq \alpha^2 \frac{1}{\alpha^2} = 1.$$

D'où le théorème 9. □

L'inconvénient de la méthode avancée est son coût en matière d'hypothèses de clefs énumérées. Dans le raisonnement précédent, le nombre de bits ciblés est $z = 8$, de manière générale, le nombre d'hypothèses de clefs énumérées pour les deux méthodes est donné en table 4.2.

Table 4.2 – Détails du coût des méthodes d'enchaînement des sous-attaques

Méthode	**Coût**
Basique	$2^z + \alpha 2^z$
Avancée	$2^z + \alpha 2^z + \boldsymbol{\alpha^2}$

4.2.3 Effet du paramètre α

Dans l'étude que nous venons d'effectuer, il ressort un paramètre prépondérant, le nombre α de candidats sélectionnés après une sous-attaque. Analysons plus en détail son effet pour les deux méthodes vues précédemment.

Effet de α avec la méthode basique

Nous avons établi que $\mathfrak{T}_{\overline{2},\underline{1},\alpha} = \overline{\mathfrak{T}} \cdot \mathfrak{T}_{\overline{1}}^{\alpha}$. Comme le taux de succès $\mathfrak{T}_{\overline{1}}^{\alpha}$ d'ordre α croît avec α, il vient alors que :

$$\mathfrak{T}_{\overline{2},\underline{1},\alpha} \leq \mathfrak{T}_{\overline{2},\underline{1},\alpha+1}. \tag{4.22}$$

Autrement dit, le taux de succès de l'attaque basique est amélioré avec l'augmentation des valeurs de α.

Effet de α avec la méthode avancée

Pour la seconde méthode, le taux de succès s'écrit $\mathfrak{T}_{\overline{2},\underline{2}} = \overline{\mathfrak{T}} \cdot \alpha \cdot \mathfrak{T}_{\overline{1}}^{\alpha} \cdot \sum_{i=1}^{\alpha} \mathfrak{T}_{\overline{\mathcal{L}_i}}^{\alpha}$. La somme $\sum_{i=1}^{\alpha} \mathfrak{T}_{\overline{\mathcal{L}_i}}^{\alpha}$ est une fonction $f(\alpha)$ croissante en la variable α. De même, on a

$$\begin{array}{rcl} \mathfrak{T}_{\overline{1}}^{\alpha} & \leq & \mathfrak{T}_{\overline{1}}^{\alpha+1} \\ \alpha & \leq & \alpha + 1 \\ f(\alpha) & \leq & f(\alpha+1), \end{array} \tag{4.23}$$

D'où le résultat 4.24, le taux de succès de méthode avancée croît avec α.

$$\mathfrak{T}_{\overline{2},\underline{2},\alpha} \leq \mathfrak{T}_{\overline{2},\underline{2},\alpha+1}. \tag{4.24}$$

La valeur minimum du paramètre α est 2, cette condition est nécessaire pour le lemme 4. L'une des perspectives données à la fin de cette thèse consiste à étudier si cette hypothèse est facilement atteignable. La valeur maximale de α est 2^z, mais les ressources requises pour faire l'attaque avec une telle valeur sont importantes. En pratique, la valeur maximale de α est définie par la puissance de l'attaquant.

4.3 Réalisation pratique des attaques

Les attaques qui ont été menées dans le cadre de cette thèse ont été validées en pratique. En effet, nous avons eu la volonté de montrer que les chemins d'attaques théoriques que nous proposons se réalisent en environnement réel. Nous décrivons les éléments qui ont été utilisés afin de mettre en place un banc d'attaque par canaux auxiliaires. Nous détaillons la cible algorithmique visée et son implémentation logicielle. Ensuite, nous décrivons les attaques réalisées.

4.3.1 Description de la cible matérielle et du banc d'attaque

Nous décrivons d'abord le dispositif qui sert de cible tout au long de la thèse ainsi que le banc d'acquisition qui sert pour les attaques par canaux auxiliaires.

Dispositif sous tests

La cible est un microcontrôleur 32 bits, implémenté en technologie CMOS 130 nm, qui peut être utilisée dans de nombreuses applications. Ce circuit intégré de base ne possède pas de protections matérielles. L'utilisation de ce dispositif doit se faire avec précaution pour les applications cryptographiques. Il s'agit d'un ARM Cortex-M3 [Arm] sur une carte de développement STM32F100 Value Line [STM], le processeur est cadencé à 24 MHz. La carte est présentée sur la figure 4.4.

L'opération ciblée par canaux auxiliaires dans les algorithmes de couplage est une multiplication modulaire. La méthode pour faire la multiplication modulaire va toujours impliquer des multiplications entre les mots machine des opérandes. C'est l'instruction assembleur UMULL qui a été utilisée à cet effet. Plus précisément, UMULL permet de faire une multiplication non signée avec des opérandes de 32 bits, et le résultat est renvoyé sur 64 bits.

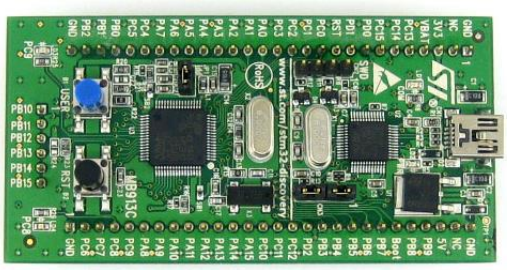
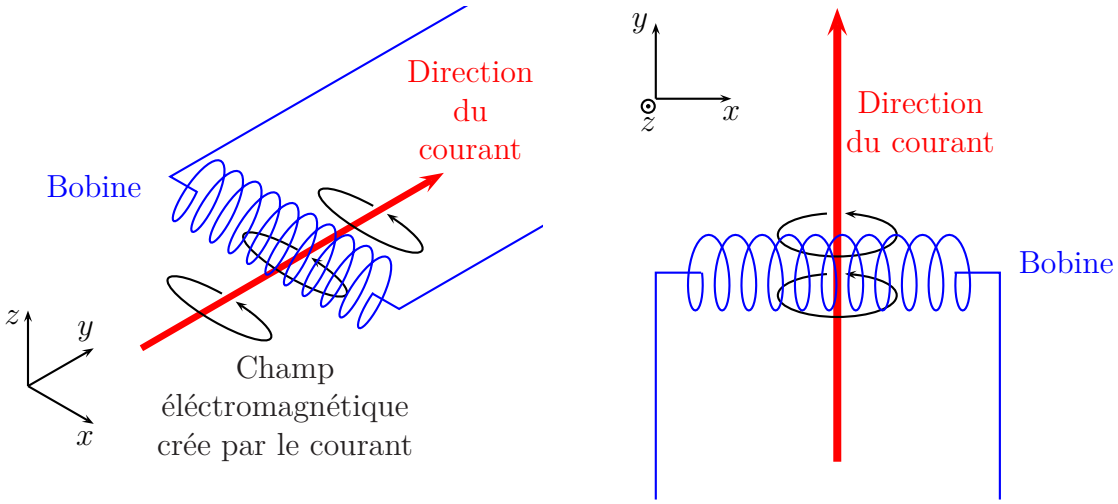


Figure 4.4 – Carte de développement STM32F100 utilisée pour nos tests

Banc d'attaque

L'exécution de l'implémentation de l'algorithme crée les fuites physiques, elles ont ensuite exploitées par les attaques par canaux auxiliaires. Plus exactement, nos expériences et celles d'autres chercheurs [NFM+17, FMN+14, ITK+16, BDGN14], permettent d'assurer que les fuites d'informations sur les données manipulées sont liées aux accès dans les registres du dispositif. Un accès à un registre, qu'il soit en lecture ou en écriture, est le résultat d'une donnée qui transite à travers les Bus horizontaux du circuit. La donnée qui transite est un courant électrique : il crée un champ électromagnétique autour de sa direction. Une des possibilités pour capter un tel champ est de placer une bobine au-dessus de celui-ci. Le champ électromagnétique causé par le courant doit traverser les spires de la bobine (celui-ci est illustré en figure 4.5a). Ceci explique la position horizontale de la bobine, la vue de dessus en figure 4.5b montre la situation, la bobine est orthogonale à la direction du courant.



(a) Champ électromagnétique généré par le courant et capté par la bobine

(b) La bobine est orthogonale à la direction du courant (vue de dessus)

Figure 4.5 – Illustration du champ électromagnétique généré par le courant et capté par la bobine

Une sonde électromagnétique est un outil qui embarque une bobine et donne une valeur analogique au champ mesuré. Une sonde qui permet de répondre aux besoins explicités ci-dessus est la Langer EMV-Technik LF-U 5. Nous plaçons la sonde au-dessus du circuit intégré sans décapsuler la puce. La position locale de la sonde permet d'être précis et de se positionner directement sur le capot de protection du circuit intégré.

La position de la sonde a son importance. Au vu de la description ci-dessus, des champs électromagnétiques mesurés, pour capter les transitements sur les Bus de données, la bobine doit être placée au-dessus des Bus concernés. Nous essayons quelques positions au-dessus du circuit, puis, visuellement on cherche à observer des motifs qui se répètent. Par exemple, pour une multiplication modulaire de deux entiers sur 8 mots de 32 bits, il y a une boucle qui parcourt 8 mots, on cherche donc à identifier 8 motifs similaires. La cartographie est un travail minutieux de placement de la sonde et de test afin de déterminer quelle position de la sonde nous donnera les meilleurs résultats.

La sonde est reliée à un amplificateur Langer Amplifier PA303 BNC (30dB). Les signaux reçus par l'amplificateur sont acquis à l'aide de l'oscilloscope Lecroy WaveRunner 640Zi.

Le banc d'attaque, avec le matériel précisé ci-dessus, est représenté en figure 4.6.

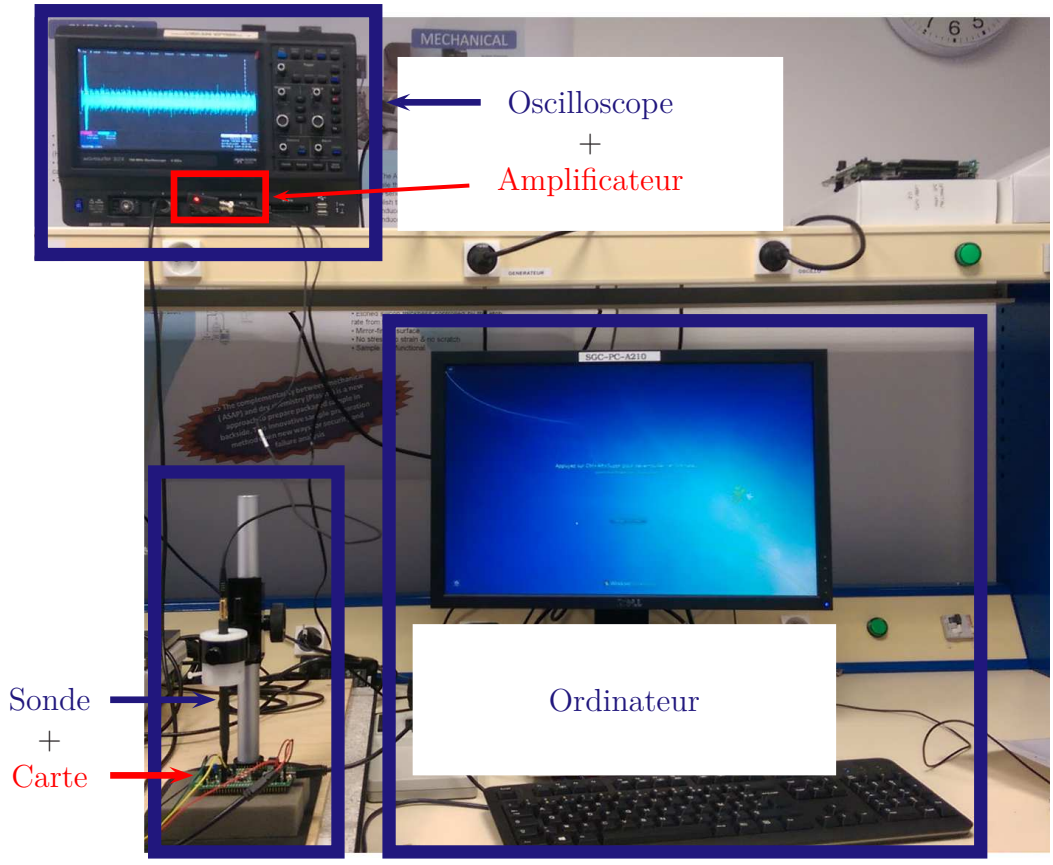


Figure 4.6 – Banc d'attaque par analyse des émissions électromagnétiques

Les connexions entre les différents éléments sont les suivantes :

- Ordinateur–Carte : pour l'alimentation de la carte (USB), et pour que l'ordinateur envoie les données que la carte va traiter (RS-232).
- Carte–Oscilloscope : signal de déclenchement.
- Sonde–Oscilloscope : les données mesurées transitent vers l'oscilloscope.
- Oscilloscope–Ordinateur : les traces de fuites sont formatées pour les traiter avec l'ordinateur (RJ45).

Le signal de déclenchement (communément appelé *trigger*), est un signal carré, qui prend sa valeur haute à un moment souhaité par le programmeur. Dans le cadre des attaques par canaux auxiliaires, le signal de déclenchement permet « d'encadrer » les opérations ciblées. Ici, nous souhaitons caractériser l'implémentation des couplages dont nous possédons le code source, donc nous pouvons facilement placer des instructions dans le code C pour déclencher le signal haut et bas.

Procédure d'acquisition

Lors de la programmation de la carte, il faut réserver deux broches pour la communication avec l'ordinateur. En effet, la communication par Universal Asynchronous Receiver Transmitter (UART) se fait généralement avec deux canaux : la transmission et la réception de données, souvent identifiées par Tx/Rx. La carte est connectée à l'ordinateur via RS-232. En pratique, un câble RS-232/USB est branché en USB à l'ordinateur, et le RS-232 à un convertisseur Pmod. Quatre broches du Pmod sont utilisées, deux pour le Tx/Rx, et deux pour alimenter le convertisseur Pmod.

La figure 4.7 montre les différentes connexions avec la carte.

- L'ellipse jaune discontinue montre les deux broches de la carte qui sont connectées au Pmod, ce sont les Tx/Rx.
- Les ellipses en orange continu désignent l'alimentation du convertisseur Pmod.
- Les pointillés verts désignent le signal de déclenchement, ces broches sont connectées à l'oscilloscope.
- Le bleu discontinu désigne l'alimentation de la carte via USB.

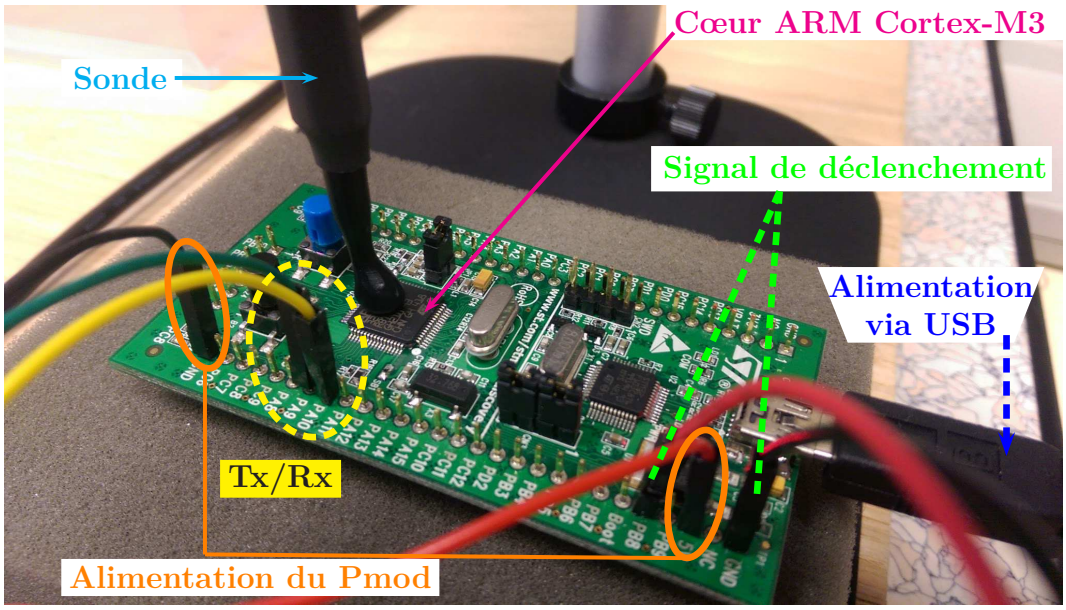


Figure 4.7 – Connexions Carte–Ordinateur et Carte–Oscilloscope

Contrôle de la carte Le code C embarqué sur la carte contient des données en dur, c'est le cas par exemple pour les paramètres de la courbe elliptique. D'autres données sont transmises par l'ordinateur. Nous avons utilisé un script de contrôle MatLab pour remplir cette tâche. C'est aussi ce script qui récupère les courbes de l'oscilloscope.

Prenons l'exemple où l'on cible simplement une multiplication modulaire $a \times b \mod p$. Le modulo p est stocké en dur dans la carte. Le secret, disons b, est lui aussi directement mis dans une variable dans le code C. On va donc envoyer des $a^{(i)}$ et récupérer les mesures d'émanations électromagnétiques.

La première étape est de lancer le programme sur la carte. Le programme va s'exécuter jusqu'à atteindre l'instruction d'attente de réception d'un entier, il est dans une boucle qui terminera lorsque les bits de l'entier $a^{(i)}$ seront reçus. Les premières instructions du script de contrôle MatLab concernent l'ouverture de la communication avec l'oscilloscope. Ensuite, dans une boucle sur i on fait les instructions suivantes :

1. Mettre l'oscilloscope dans son état « prêt à acquérir », il démarrera l'acquisition d'une courbe dès que le signal de déclenchement sera haut.

2. Envoyer $a^{(i)}$ à la carte, elle renvoie ensuite cet entier et le script de contrôle fait un test d'égalité pour vérifier qu'il n'y a pas eu d'erreur lors des transmissions.

3. Attendre un drapeau (*flag*) venant de la carte qui arrivera dès que l'opération $a \times b \mod p$ aura été exécutée par le dispositif.

4. Attendre un drapeau venant de l'oscilloscope pour prévenir qu'il a fini d'acquérir la mesure.

5. Sauvegarde de la trace dans l'ordinateur. La trace est transférée jusqu'à l'ordinateur, puis sauvegardée.

Exemple de courbe acquise La procédure précédente permet d'acquérir les traces d'émissions électromagnétiques d'une multiplication modulaire $a \times b \mod p$ avec b secret et a qui est maîtrisé. La figure 4.8 permet de visualiser une telle mesure. Une attaque de type CPA verticale contre cette opération a été détaillée dans le chapitre 3, elle sera utilisée et améliorée tout au long du chapitre 4. La fréquence d'échantillonnage de l'oscilloscope est de 4 GS/s (*Giga Sample per second*), c'est-à-dire qu'une mesure acquise sur une seconde sera composée de 4 milliards de points. Dans l'exemple ci-dessous, la multiplication prend 5 ms, elle est donc échantillonnée sur 20 millions de points.

4.3.2 Algorithmique ciblée

Les failles des implémentations des algorithmes de couplages sont les mêmes d'un couplage à l'autre. Concrètement, dès que le couplage fait appel à l'algorithme de Miller[1], alors le calcul de la tangente à la première itération va manipuler les données dépendantes de la clef secrète et de l'entrée connue.

Par ailleurs, les récentes avancées sur le DLOG [MSS16, BD17] ont montré que la taille des clefs doit être revue à la hausse. Cela n'affectera pas la cible visée par nos attaques par canaux auxiliaires. Effectivement, les coordonnées du point secret seront

[1] C'est le cas pour Tate, Ate, Twisted Ate, Optimal Ate.

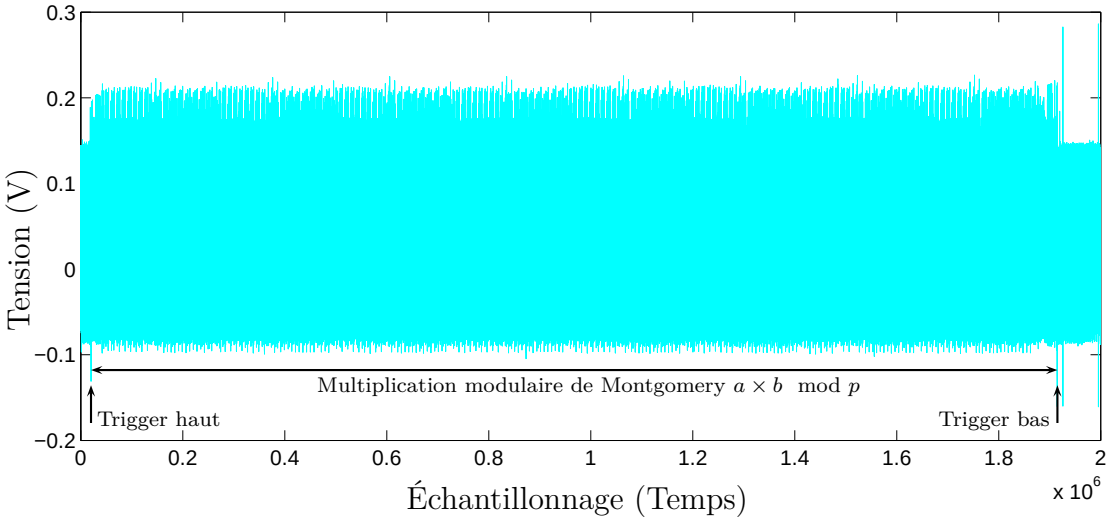


Figure 4.8 – Une mesure de l'émanation électromagnétique d'une multiplication modulaire de Montgomery

plus grandes, dans le cas des courbes BN à 128 bits de sécurité, elles vont passer de 256 à 456 bits (voir par exemple [BD17]). Les coordonnées seront simplement des entiers stockés dans 15 mots de 32 bits plutôt que 8 mots, dans le cas d'une architecture 32 bits. La cible de l'attaque est la multiplication entre deux mots machine, cette opération est toujours réalisée par les algorithmes de multiplications modulaires.

Couplage visé

Les couplages ciblés pendant la thèse sont le couplage de Ate et Optimal Ate sur courbes BN (cf. algorithme 2.8) avec les paramètres suivants : $q(x), m(x)$ et $t(x)$ sont pris pour $x = \texttt{0x3FC0100000000000}$. La courbe elliptique E est $y^2 = x^3 + 5$. La tour d'extension pour construire $\mathbb{F}_{q^{12}}$ est :

$$\begin{array}{rcll} \mathbb{F}_{q^2} & \simeq & \mathbb{F}_q[u]/(u^2-\beta) & \text{avec } \beta = -5 \text{ un non-résidu quadratique dans } \mathbb{F}_q \\ \mathbb{F}_{q^6} & \simeq & \mathbb{F}_{q^2}[v]/(v^3-u) & \text{avec } u \text{ un non-résidue cubique dans } \mathbb{F}_{q^2} \\ \mathbb{F}_{q^{12}} & \simeq & \mathbb{F}_{q^6}[w]/(w^2-v) & \text{avec } v \text{ un non-résidu quadratique dans } \mathbb{F}_{q^6} \end{array} \tag{4.25}$$

E admet une tordue de degré 6 donnée par $E_2 : y^2 = x^3 - 5/u$. Un point $Q' = (x_{Q'}, y_{Q'}) \in E_2(\mathbb{F}_{q^2})$ est en bijection avec le point $Q = (x_{Q'}w^2, y_{Q'}w^3)$ sur $E(\mathbb{F}_{q^{12}})$. Le couplage $e_{m,OA}(Q', P)$ manipule (double-et-ajoute) le point $T \leftarrow Q'$ qui a ses coordonnées dans $\mathbb{F}_{q^2}$ plutôt que $Q \in E(\mathbb{F}_{q^{12}})$ qui a ses coordonnées dans $\mathbb{F}_{q^{12}}$.

Arithmétique

La représentation des points se fait via le système de coordonnées suivant :

- $P \in E(\mathbb{F}_q)$ est en coordonnées affines.
- $Q' \in E_2(\mathbb{F}_{q^2})$ est en coordonnées affines.
- $T \in E_2(\mathbb{F}_{q^2})$ est en coordonnées jacobiennes.

Les calculs de doublements et d'additions sur les courbes elliptiques et les évaluations de tangentes et de lignes sont faits de manière combinés, section *Calcul combiné addition/ligne et doublement/tangente*, équations de l'annexe C page 171.

L'arithmétique sur $\mathbb{F}_{q^{12}}$ est celle que nous avons introduite à la section *Calcul dans les tours d'extensions de corps finis*, page 26. L'implémentation de l'arithmétique sur les extensions quadratiques est détaillée dans les algorithmes de l'annexe A.1. Les calculs dans les extensions cubiques sont donnés en annexes A.2. Les arithmétiques sur $\mathbb{F}_{p^6}$ et $\mathbb{F}_{p^{12}}$ avec les choix d'implémentations efficaces sont présentées respectivement dans les annexes A.3 et A.4.

Le code C complet de l'implémentation du couplage Optimal Ate est livré dans la thèse de Ronan Lashermes [Las14]. Nous avons adapté le code source en insérant les instructions pour que le signal de déclenchement « encadre » l'opération ciblée par la CPA.

4.3.3 Attaque CPA verticale

Nous ciblons l'entrée secrète $k = (k_{\mathfrak{n}-1} \dots k_0)_2$, avec $\mathfrak{n} = 32$, impliquée dans $k \times x$. Dans un premier temps, nous faisons l'acquisition des émanations électromagnétiques de cette opération pour plusieurs valeurs de x.

Traitement des traces avant l'attaque

Les traces que nous allons utiliser pour faire les attaques sont synchronisées par l'oscilloscope grâce au signal de déclenchement. L'oscilloscope sert de convertisseur qui transforme le signal analogique (renvoyé par la sonde, puis filtré par l'amplificateur) en un signal numérique. Cette étape est une source de gigue. D'autres sources de gigue sont principalement liées au circuit intégré. La conséquence de ces gigues sur les signaux acquis est une désynchronisation. D'un signal à l'autre, les fenêtres où les opérations critiques sont exécutées sont décalées « aléatoirement » de quelques points d'échantillonnage.

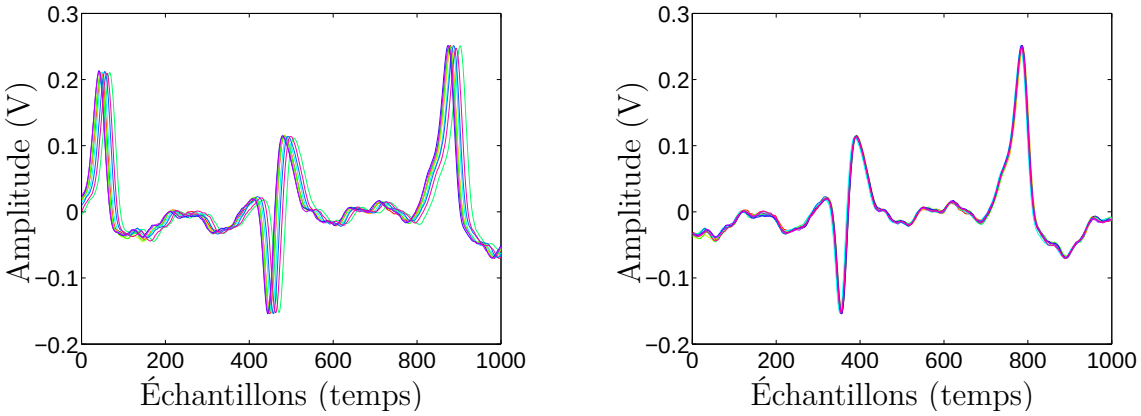
Dans notre cas, nous analysons des mesures électromagnétiques d'une implémentation logicielle sur un microcontrôleur synchrone dont l'horloge est cadencée à 24MHz. Les gigues observées sont liées à l'horloge interne qui n'est pas parfaite. Ces décalages observés sont très faibles[2], c'est pour cela qu'une méthode simple de synchronisation des traces suffit.

Du fait de cette faible désynchronisation, nous utilisons une méthode d'alignement basique. Il s'agit, pour chaque trace, de trouver l'indice du premier pic, et de prendre les traces à partir de cet indice. Le résultat de la méthode est illustré sur la figure 4.9.

CPA de vérification sur 32 bits

Afin de vérifier les fuites du dispositif et de l'opération ciblée, nous faisons une CPA de caractérisation sur 32 bits. Quelques hypothèses de clefs sont tirées au hasard parmi

[2]Comparés à un dispositif cadencé à plus de 1GHz par exemple. Par ailleurs, des sources importantes de gigues peuvent être provoquées par des interruptions de processeur [OP12], les systèmes d'exploitation embarqués sur le dispositif [HWH13]. La présence d'une architecture à plusieurs cœurs est aussi une problématique [WL07], la synchronisation des tâches (*thread* en anglais) qui ont été parallélisées est aussi cause de gigues.



(a) Traces brutes (b) Traces synchronisées superposées

Figure 4.9 – Synchronisation des traces

les 2^{32} possibles, puis la bonne clef sert aussi d'hypothèse. Le modèle est en poids de Hamming [CKN00, MS00, Osw03]. Plus précisément, la prédiction de l'état intermédiaire de la CPA est $HW_{32}(k \times x)$ pour la clef k et le message x tous deux sur 32 bits, le résultat de la multiplication est aussi pris sur 32 bits. Seuls les 32 bits de poids faibles sont considérés.

La figure 4.10 rapporte les corrélations obtenues avec 1000 traces.

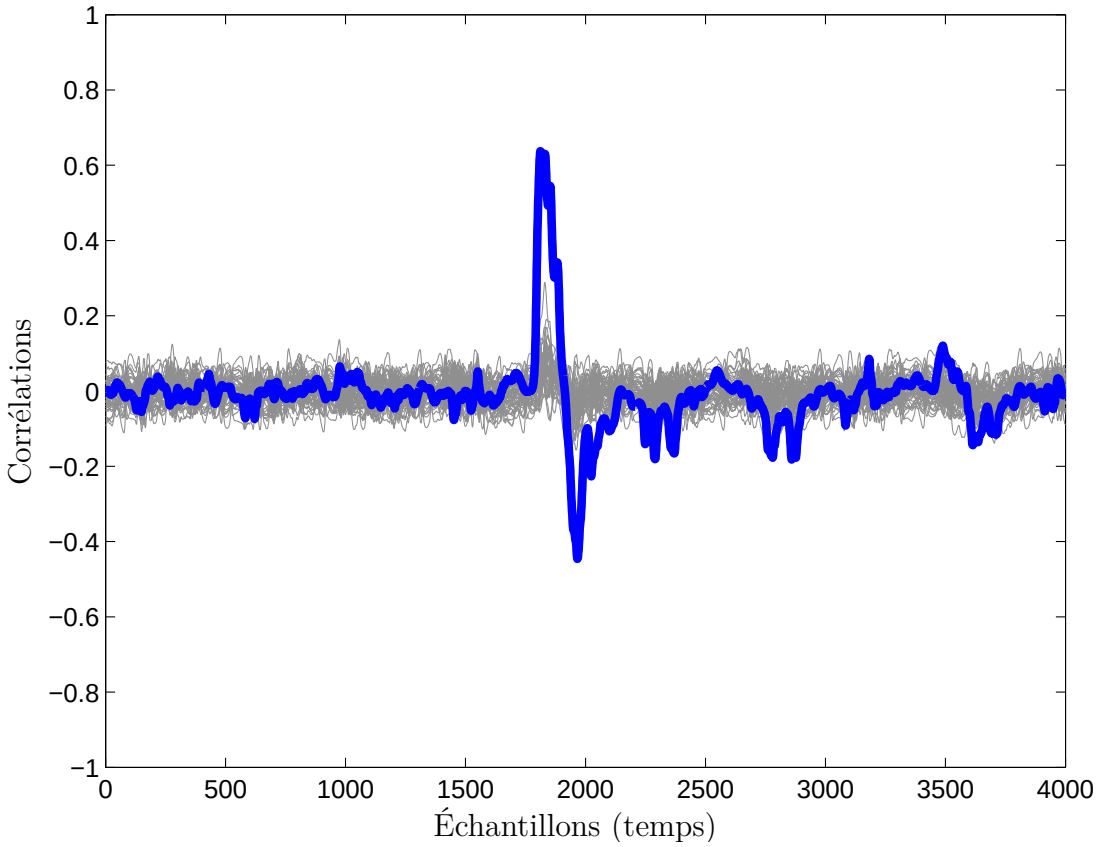


Figure 4.10 – CPA de caractérisation sur 32 bits

On observe un pic de corrélation important sur la figure 4.10. La fuite est claire. Cependant, afin de mener entièrement cette attaque, il serait nécessaire d'énumérer toutes les hypothèses de clef possibles sur un mot machine, c'est-à-dire 2^{32} possibilités,

ce qui est encore considéré infaisable avec un ordinateur classique. D'où la nécessité d'identifier un modèle de fuite sur seulement une partie des 32 bits du secret.

Modèles de fuites

L'objectif du modèle de fuite est de pouvoir prédire la fuite du dispositif en fonction des données traitées. Le secret k va devoir être ciblé octet par octet en partant de l'octet de poids faible. L'attaquant peut alors faire une hypothèse sur cet octet et sur le mot machine du message x. À partir de l'hypothèse sur les bits de k et de x, l'attaquant veut reproduire la fuite créée par le produit $k \times x$. Le produit d'un entier de z bits et de 32 bits donne un entier jusqu'à $32+z$ bits. Les modèles de fuites étudiés correspondent à un attaquant qui prend les z ou les $2z$ bits de poids faibles. La figure 4.11 illustre la situation.

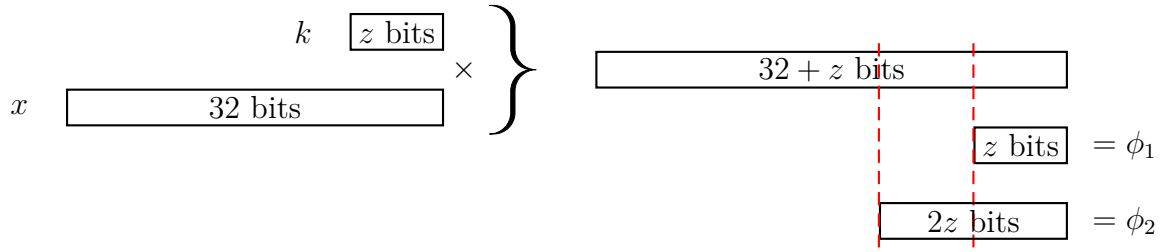


Figure 4.11 – Illustration des modèles de fuites

Nous écrivons formellement les modèles ϕ_1 et ϕ_2.

$$\begin{aligned} c = k \times x = (c_{2z-1} \dots c_0)_2 \text{ alors } \phi_1(k,x) = \textstyle\sum_{i=0}^{z-1} c_i, \\ c = k \times x = (c_{2z-1} \dots c_0)_2 \text{ alors } \phi_2(k,x) = \textstyle\sum_{i=0}^{2z-1} c_i. \end{aligned} \tag{4.26}$$

Remarque 8. *Prendre les z bits de poids faible dans le modèle ϕ_1 revient à prendre $k \times x \mod 2^z$.*

Tests statistiques pour évaluer les modèles de fuites

Nous évaluons les deux modèles de fuites précédemment introduits à l'aide du t-test. Ce test statistique est introduit pour la détection de points d'intérêt par Gierlichs *et al.* [GLRP06] sous le nom de SOST (*Sum of Squared pairwise T-differences*).

Pour cela, nous utilisons les acquisitions précédentes. La clef k est fixée et les messages varient. Nous fixons $z = 8$ bits. Pour chaque trace, le calcul de $\phi_i(k,x)$ renvoie à un ensemble. À la fin, la taille de chaque ensemble est notée $\eta_{\phi,i}, i = 1, \dots, N_\phi$. Dans le cas du premier modèle, il y a $N_{\phi_1} = 9$ ensembles (9 poids de Hamming possible pour 8 bits). Et pour le second modèle $N_{\phi_2} = 17$. Pour chacun des ensembles, la moyenne $m_{\phi,i}$ et la variance $\sigma^2_{\phi,i}$ sont calculées. Le SOST est alors calculé par la formule 4.27.

$$SOST_\phi = \sum_{i,j=1}^{N_\phi} \left(\frac{m_{\phi,i} - m_{\phi,j}}{\sqrt{\frac{\sigma^2_{\phi,i}}{\eta_{\phi,i}} + \frac{\sigma^2_{\phi,j}}{\eta_{\phi,j}}}} \right)^2 \text{ pour } i \geq j. \tag{4.27}$$

En pratique, nous obtenons le résultat présenté en figure 4.12. Pour cette figure, un zoom est fait sur la partie contenant la fuite. La fuite est bien plus marquée avec le modèle ϕ_2. Le pic est plus franc et il est aussi plus élevé avec le second modèle.

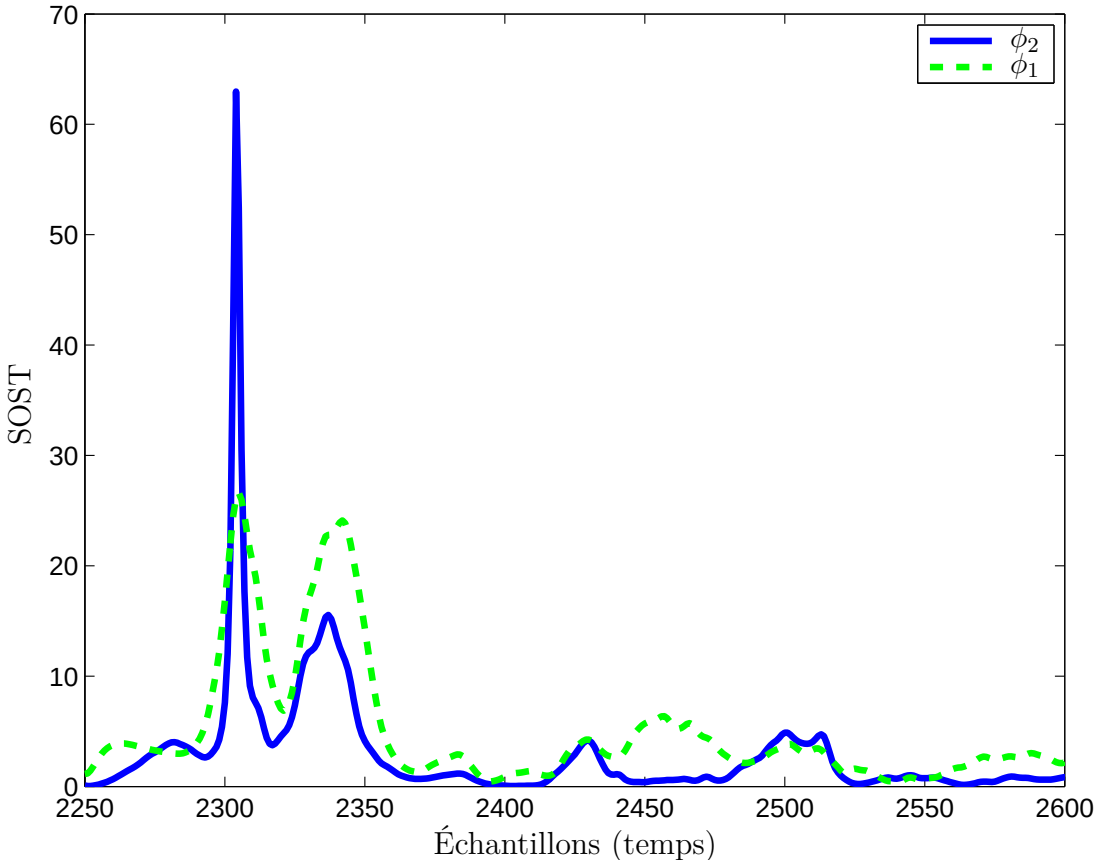


Figure 4.12 – Sum of Squared pairwise T-differences sur deux modèles pour la multiplication

Enchaînement des sous-attaques

L'étude théorique des méthodes basique et avancée d'enchaînement des sous-attaques a permis de conclure que la méthode avancée que nous proposons donnera de meilleurs résultats pour l'attaque. Il s'agit maintenant de comparer en pratique les deux méthodes. Les études ont été menées sur l'enchaînement de deux sous-attaques, mais il est aisé de transposer les résultats sur quatre sous-attaques avec un mot de 32 bits comme cible. La figure 4.13 illustre cet enchaînement des sous-attaques pour arriver à une clef candidate sur 32 bits.

La réalisation pratique de l'attaque sur le mot de 32 bits avec les deux méthodes d'enchaînement donne les résultats de la figure 4.14. Une nette amélioration du taux de succès est obtenue avec la seconde méthode.

Nous appliquons le schéma d'enchaînement des sous-attaques sur les 32 bits. Ce schéma est illustré par la figure 4.13, nous identifions les sept étapes où une sous-attaque est faite. L'attaque i est la première sous-attaque, sur l'octet de poids faible. À partir des α meilleurs candidats pour cet octet, les hypothèses de clefs pour l'octet suivant sont faites. Pour chacun de ces ensembles, l'attaque ii permet de sélectionner les

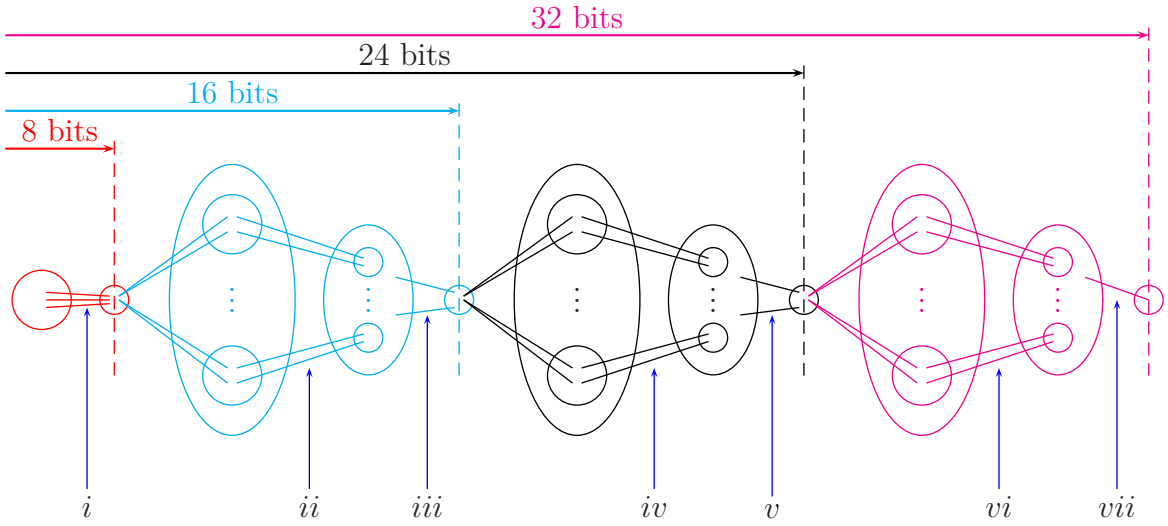


Figure 4.13 – Schéma d'attaque avancée de la multiplication sur 32 bits

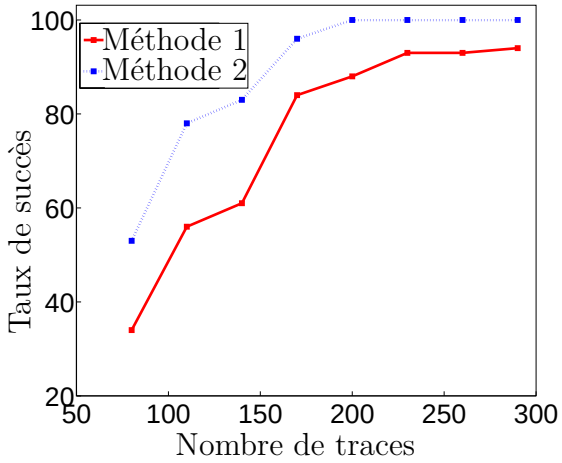


Figure 4.14 – Comparaison pratique des deux méthodes d'enchaînement des sous-attaques

α meilleurs candidats (sur 1 octet). Cet ensemble de $\alpha \times \alpha$ clefs contient les hypothèses pour l'attaque iii, qui permet de sélectionner α candidats sur 2 octets. Ce principe est répété deux fois de plus. La seconde fois, une seule clef est retenue, c'est la clef candidate $\widehat{k}$ pour $k^\star$ sur 32 bits.

Nos résultats expérimentaux montrent que le succès de l'attaque ($\widehat{k} = k^\star$ sur les 32 bits) peut passer par des étapes où les attaques intermédiaires $i, ii, \ldots, vi$ ne parviennent pas à classer la bonne clef en première position. L'effet de α se manifeste ici, plus il est grand et plus il est probable que la bonne clef soit sélectionnée pour les sous-attaques suivantes. L'objectif est d'arriver à mener la bonne clef jusqu'à la sous-attaque vii. Cette attaque CPA va parcourir les quelques hypothèses retenues jusqu'à présent, les valeurs intermédiaires sont sur 32 bits, la taille de l'architecture. Donc les données prédites sont aux plus proches des données manipulées par UMULL.

L'effet du paramètre α sur le taux de succès de l'attaque est démontré théoriquement à la section 4.2.3. La section suivante est consacrée à la validation pratique de ce résultat.

Effet du paramètre α

Le nombre α de candidats qui sont présélectionnés après une sous-attaque est un paramètre important pour la réussite des attaques CPA de la multiplication de deux mots machine. Nous avons montré théoriquement que le taux de succès de l'attaque croît avec α. Nous validons maintenant ce résultat en pratique. En faisant varier α de 8 à 64, les résultats des attaques avec des nombres de traces différents, il sera possible d'évaluer expérimentalement l'effet de ce paramètre. La figure 4.15 illustre ces résultats.

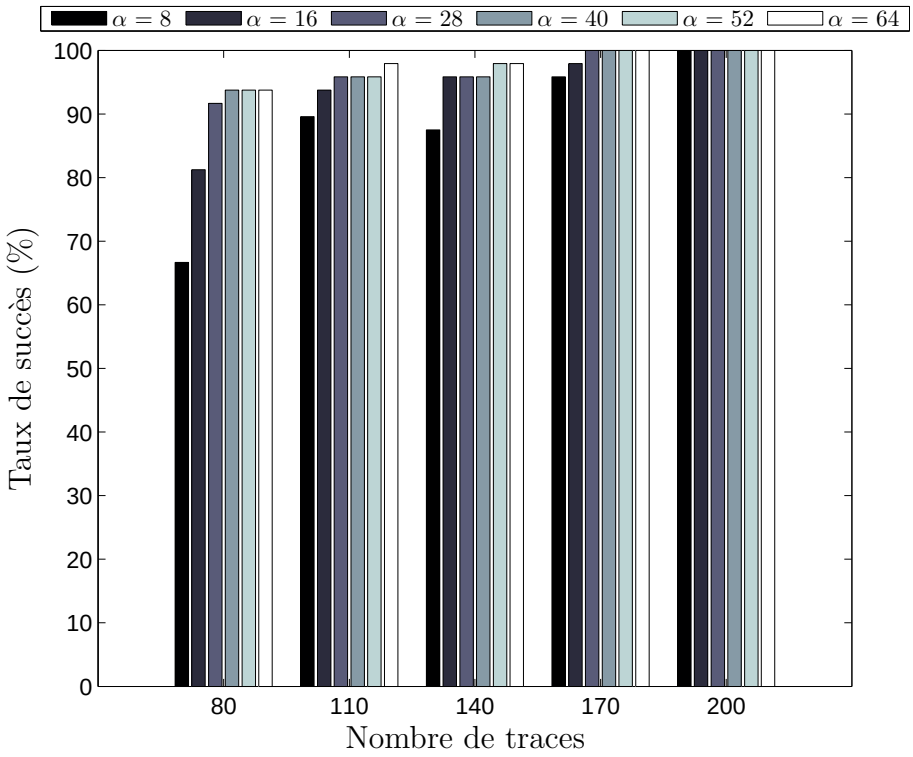


Figure 4.15 – Comparaison des taux de succès pour différentes valeurs de α

Unterluggauer *et al.* [UW14] ont testé seulement les valeurs $\alpha = 5$ et $\alpha = 10$ sans noter de différences. Grâce à notre expérimentation (figure 4.15) nous confirmons notre résultat théorique, à savoir : pour un nombre fixe de traces utilisées pour l'attaque, le taux de succès augmente avec α. Par exemple, pour une base de données de 80 traces, avec $\alpha \geq 40$ le taux de succès de l'attaque grimpe à plus de 90%. Pour 170 traces, l'attaque à un taux de succès de 100% dès que $\alpha \geq 28$.

En contrepartie, les ressources nécessaires à la CPA augmentent aussi avec α. Il y a deux ressources qui ont besoin d'être évaluées :

- Le « temps », c'est le nombre de clefs énumérées.
- La « mémoire », c'est le nombre de candidats à stocker.

Le temps dépend de α de la façon suivante : $2^8 + 2^8\alpha + 2^8\alpha + 2^8\alpha$ énumérations de clefs. L'espace requis doit contenir $4(\alpha + 2^8\alpha + 8\alpha)$ candidats. La figure 4.16 montre l'évolution de ces ressources en fonction de α.

Notre stratégie de diviser pour mieux régner consiste à diviser les 32 bits du secret en 4 fois 8 bits. Elle diffère de celle de Unterluggauer *et al.* [UW14] qui proposent un découpage en 2 fois 16 bits. L'étude de ce découpage nous permet d'optimiser l'attaque.

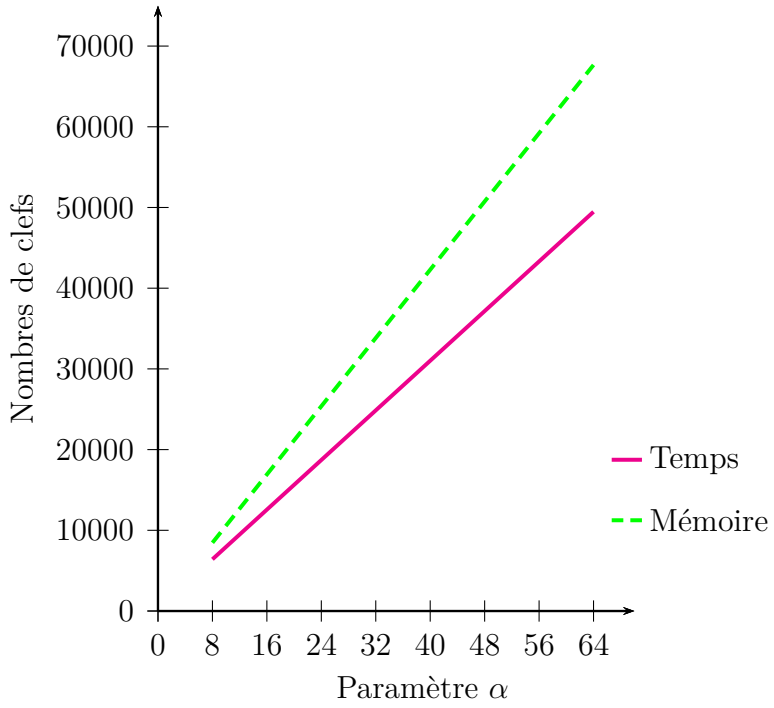


Figure 4.16 – Ressources en fonction de α

D'une part, notre attaque permet de retrouver le mot secret avec 200 traces, ce qui est inférieur aux 1500 qu'ils utilisent. D'autre part, les ressources de notre attaque sont moindres. La table 4.3 compare ces ressources requises pour leur meilleure configuration ($\alpha = 5$) et notre configuration la plus coûteuse ($\alpha = 64$).

Table 4.3 – Comparaisons des ressources

	Unterluggauer *et al.*	**Notre méthode (avec $\alpha = 64$)**
Temps	$2^{18} < 2^{16} + 5 \times 2^{16} < \mathbf{2^{19}}$	$2^{15} < 2^8 + 3(2^8\alpha) < \mathbf{2^{16}}$
Mémoire	$2^{18} < 5 \times 2^{16} < \mathbf{2^{19}}$	$4\alpha(9 + 2^8) \approx \mathbf{2^{16}}$

4.3.4 Attaques par profilage

Les attaques par profilage sont considérées comme un outil puissant pour retrouver les clefs secrètes en exploitant les fuites de l'implémentation du cryptosystème. Elles se différencient des autres attaques grâce à leur phase de profilage. En effet, la caractérisation de la cible (dispositif et opération) permet de créer une base de données. Celle-ci sera utile afin de faciliter la phase d'attaque.

Cas d'étude

Nous allons expérimenter cette attaque sur la multiplication $k \times x$ entre deux mots machine sur une architecture 32 bits. Nous ciblons l'octet de poids faible de k. Les classes sont les couples créés de la manière suivante :

$$\langle HW_8(x), HW_{16}(k \times x) \rangle \,. \tag{4.28}$$

Puisque l'octet de poids faible de k est ciblé, les hypothèses de clefs sont sur 8 bits. Le résultat $k \times x$ est sur 16 bits, ainsi il y a $9 \times 17 = 153$ profils $\left(\overline{C_{cl}}, S_{cl}\right)_{0 \leq cl \leq 152}$ à créer. Nous ne conservons que les moyennes, car les matrices de covariances ont des valeurs propres proches de 0. Les signaux correspondent aux mesures de l'opération $k \times x$ avec k et x qui varient pour la phase de profilage. Les traces sont échantillonnées sur $m = 34$ points. Les traces acquises pour construire les profils et pour faire l'attaque sont extraites du même dispositif. Nous nous épargnons ici les problématiques de variations des procédés de fabrication d'un dispositif à l'autre [RO04, EG12, CK14].

Résultats

Nous faisons cette attaque pour comparer les résultats avec ceux d'une CPA. Les nombres de traces pour la phase de profilage sont $N_p \in \{1000, 1200, 1400, \ldots, 4800, 5000, 10000, 15000, \ldots, 80000, 85000\}$. Pour la phase d'attaque, nous faisons varier N_a dans $\{20, 40, 60, \ldots, 980, 1000\}$. Pour chaque couple (N_p, N_a), nous faisons 25 attaques afin de moyenner les résultats. Pour chacune de ces 25 attaques, le rang de la bonne clef $k^\star$ dans le vecteur de vraisemblance est retenu. Enfin, ce rang est moyenné pour les 25 attaques. Les résultats sont représentés par la figure 4.17. La hauteur de chaque intersection est le rang de la bonne clef. Un rang égal à 1 signifie que les 8 bits de la clef sont retrouvés. Une précision concernant la figure 4.17, l'axe *Nombre de traces : phase de profilage* est composé de deux échelles différentes.

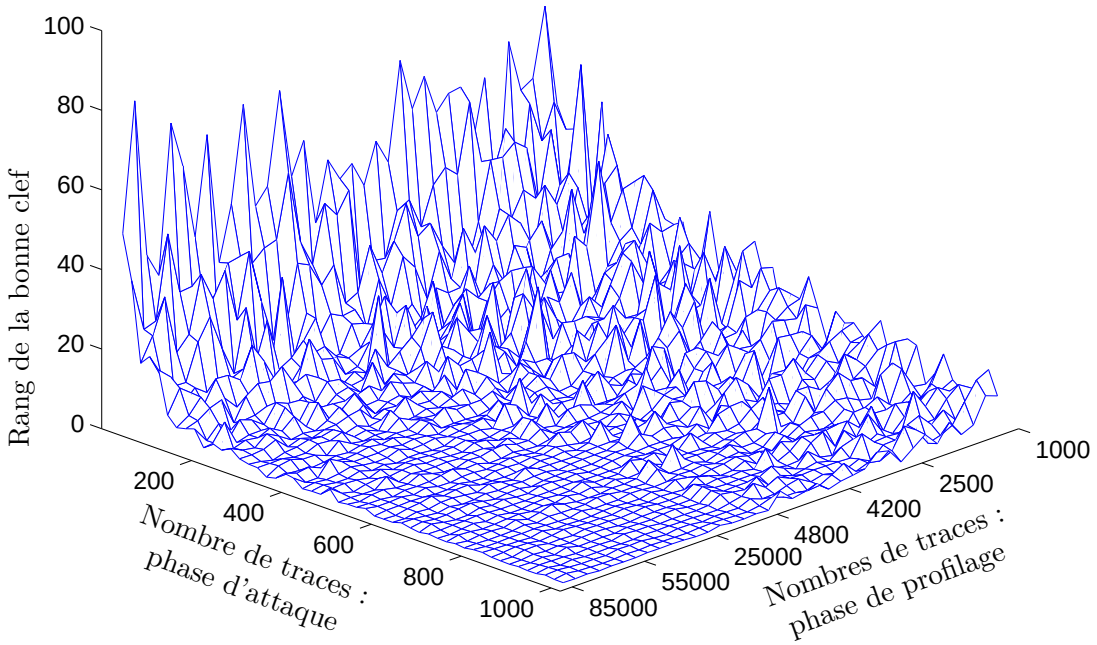


Figure 4.17 – Résultats d'attaques par profilage sur la multiplication

Notre attaque CPA permet de retrouver les 32 bits de $k^\star$ avec 200 traces, les résultats de cette attaque par profilage sont cohérents, mais ne montrent pas une amélioration drastique par rapport à la CPA.

Nous avons mené cette attaque pour deux raisons. La première était de vérifier la faisabilité de l'attaque contre la multiplication, le but final est d'attaquer une version

protégée d'une implémentation d'un couplage avec ce type d'attaque (chapitre 5, section 5.2). La seconde raison était de comparer les résultats avec la CPA, ils n'apportent qu'une amélioration modérée.

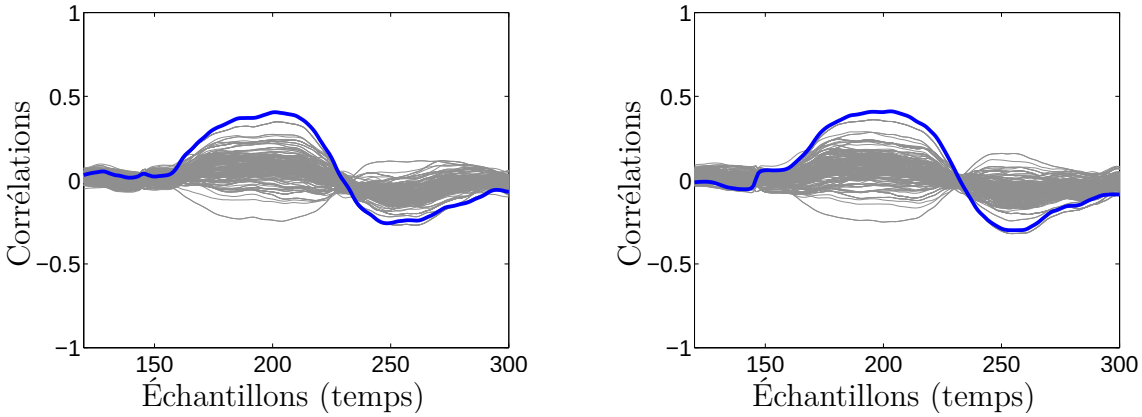
4.3.5 Conséquences du choix P ou Q secret

Le choix de placer le secret dans le point P ou Q en entrée de couplage a fait l'objet de discussions dans [WS06, EDFL09] et [BGL13]. Afin de bien fixer les notations, nous nous plaçons dans le cadre d'un couplage de Tate $\tau_m(P, Q)$ calculé avec l'algorithme de Miller (algorithme 2.6). Cette précision est nécessaire, car pour le couplage Ate ou Optimal Ate, l'agencement de P et Q est inversé.

Whelan *et al.* [WS06] ont affirmé que la solution de placer le secret dans P implique une contre-mesure naturelle. Ce n'est malheureusement pas le cas. En effet, comme nous le reprenons dans cette thèse, la cible se ramène à une multiplication modulaire entre des coordonnées qui dépendent, d'une part de P, et d'autre part de Q. Et, bien que l'algorithme de la multiplication modulaire ne soit pas symétrique, dans le cas de la multiplication de Montgomery, nous avons montré en section 4.1, qu'il importe peu que d'attaquer $a \times b \mod p$ avec a ou b secret.

Attaque de la multiplication de mots machine $a_0 \times b_0$ et $b_0 \times a_0$

La multiplication $a \times b \mod p$ est, dans les deux cas a ou b secret, une cible pour les attaques CPA. La section 4.1 comporte les schémas d'attaques contre la multiplication modulaire de Montgomery dans les deux cas. Il reste un point à vérifier en pratique : les multiplications des mots machine $a_0 \times b_0$ et $b_0 \times a_0$ se comportent-elles de la même manière face à une CPA. Pour cela, nous faisons les deux attaques dans les deux cas, en ciblant les 8 bits de poids faibles. Il s'agit de l'attaque i de la figure 4.13. Les résultats des corrélations sont donnés en figure 4.18.



(a) CPA sur $a_0 \times b_0$ avec a_0 secret (b) CPA sur $a_0 \times b_0$ avec b_0 secret

Figure 4.18 – CPAs sur $a_0 \times b_0$ avec a_0, puis b_0 secret

D'une part, la figure 4.18 montre que le choix de a_0 ou b_0 secret n'influe pas sur la forme globale des corrélations. D'autre part, la bonne hypothèse de sous-clef (8 bits de poids faibles) est distinguée dans les deux cas.

Nous venons de valider en pratique que d'opter pour a_0 ou b_0 secret n'importe pas sur l'attaque.

Attaque horizontale contre les couplages

Toutefois, la structure de l'algorithme de Miller est asymétrique en P et Q. Plus précisément, le point temporaire T, qui est affecté à la valeur de Q au début de Miller (couplage de Ate), va évoluer au fil des itérations. L'étude de cette structure nous permet d'identifier une faille.

Faille dans l'algorithme de Miller Le calcul de l'équation de la tangente, par exemple en coordonnées mixtes affine-jacobienne (cf. équation 7.20 de l'annexe C, page 171) implique une multiplication entre x_P et Z_T^2. Une exécution de l'algorithme de Miller comporte $\mathfrak{n} - 1$ itérations, donc $\mathfrak{n} - 1$ calculs de tangente, et donc $\mathfrak{n} - 1$ multiplications $x_P(Z_T^2)$ avec x_P fixe et Z_T qui varie.

Dans le cas où le point P est le secret, chaque exécution d'un couplage permet d'observer $\mathfrak{n}-1$ multiplications qui impliquent x_P et un entier connu. L'attaque devient une CPA classique avec $\mathfrak{n} - 1$ traces.

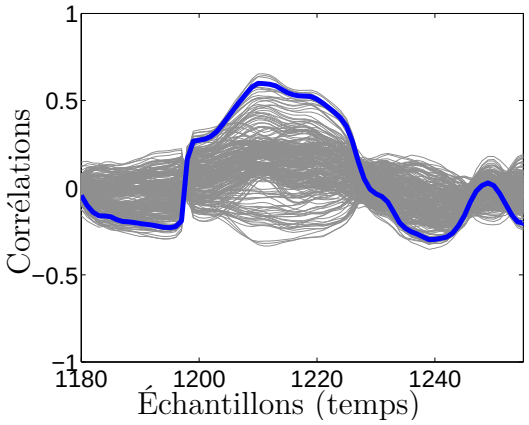
Cette faille est présente uniquement dans le cas où le point P est secret (notation pour un couplage de Ate $e_{m,A}(Q, P)$).

Réalisation pratique de l'attaque horizontale Expérimentalement, nous ciblons un couplage de Ate avec les paramètres précisés en section 4.3.2. Le nombre d'itérations s'élève à 126 pour les courbes BN utilisées, nous allons alors relever 126 traces de multiplications modulaires de Montgomery $a \times b \mod q$ avec b secret. Pour l'acquisition des traces, nous récupérons les 126 traces des multiplications modulaires ciblées, plutôt que de récupérer une trace complète d'un couplage de Ate. Pour cela, nous paramétrons le signal de déclenchement pour qu'il soit synchronisé avec la multiplication $a \times b \mod p$ ciblée. De cette manière, l'oscilloscope va recevoir l'ordre d'acquérir 126 traces, dès qu'il a terminé les acquisitions, il les envoie à l'ordinateur. Cinq minutes ont suffi à faire l'acquisition des 126 traces issue d'une exécution du couplage de Ate.

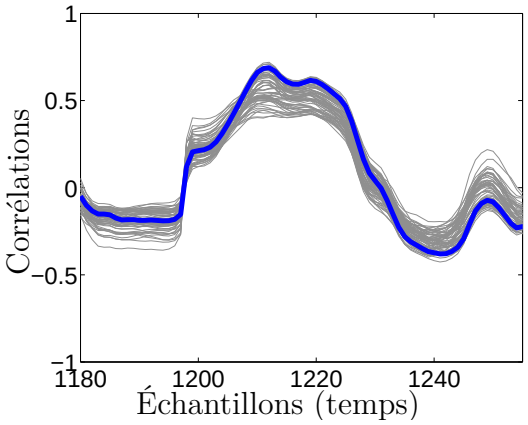
La situation de l'attaque est celle du cas 2 de la section *Attaque de la multiplication modulaire*, page 86. Pour le paramètre α, nous choisissons une petite valeur pour commencer : $\alpha = 8$. Cette petite valeur est celle qui nécessite le moins de ressources pour faire l'attaque à défaut d'être la meilleure. L'attaque avec $\alpha = 8$ a permis de retrouver le secret b, il est donc inutile de faire d'autres attaques avec $\alpha > 8$.

Les figures 4.19 et 4.20 montrent les résultats des attaques pour les deux mots b_0 et b_1 de poids faibles de $b = (b_{\mathfrak{n}-1} \dots b_0)_{\overline{\mathfrak{w}}}$. Plus exactement, les figures 4.19a et 4.20a sont les résultats des corrélations après les premières sous-attaques (attaques i de la figure 4.13) respectivement pour les mots b_0 et b_1. Les figures 4.19b et 4.20b représentent les corrélations après les attaques iii. Ensuite, les attaques v sont exposées en figures 4.19c et 4.20c. Enfin, les dernières sous-attaques (vii) sont représentées par les figures 4.19d et 4.20d.

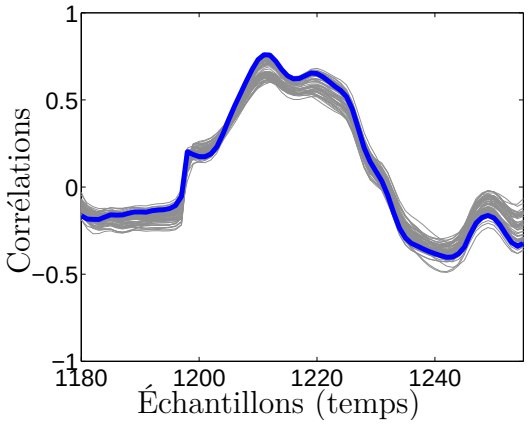
Sur les figures 4.19 et 4.20, les sous figures 4.19a et 4.20a sont le résultat des premières sous-attaques, la clef secrète (en gras) est parmi les meilleures hypothèses de clefs. Mais elle n'est pas forcément identifiée en première position. Les résultats des attaques finales (permettant de retrouver les 32 bits du secret) sont représentés par les figures 4.19d et 4.20d. La clef secrète est à chaque fois identifiée. Les attaques sur les mots 3, 4, ..., 8 du secret k donnent des résultats similaires. Finalement, les 256 bits du secret sont retrouvés à l'aide d'une seule exécution du couplage de Ate. Cela a pris 5 minutes pour les acquisitions et une heure pour les calculs de l'attaque.



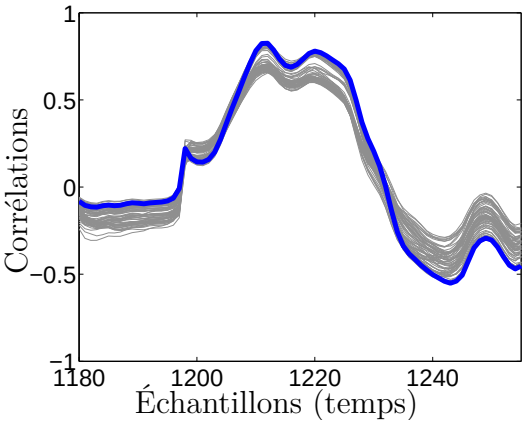
(a) Corrélations : 8 bits de poids faibles



(b) Corrélations : 16 bits de poids faibles

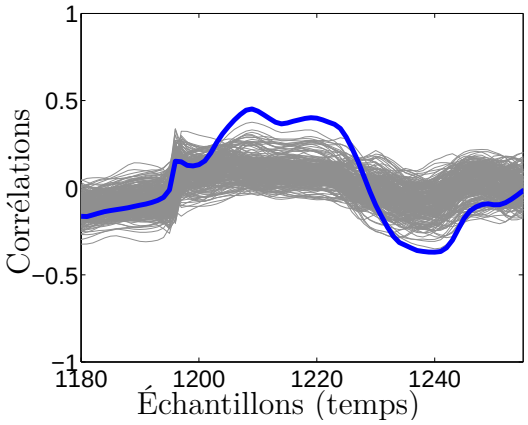


(c) Corrélations : 24 bits de poids faibles

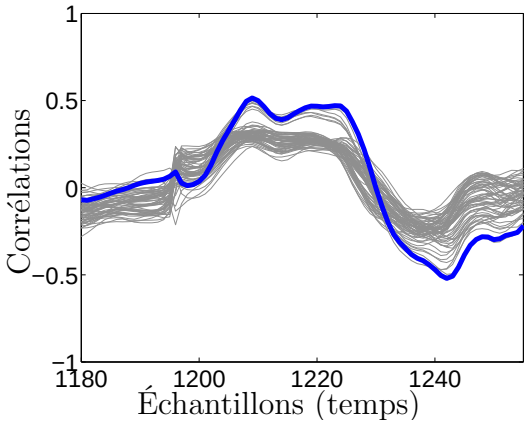


(d) Corrélations : 32 bits du secret

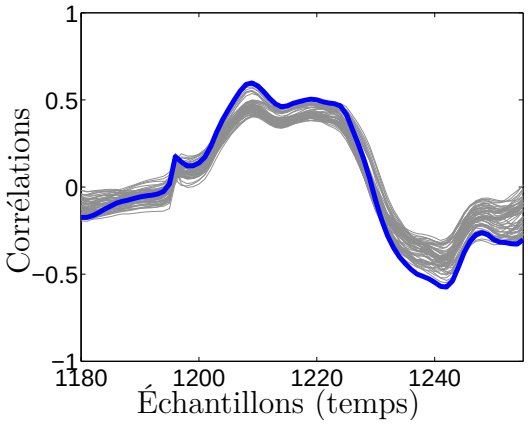
Figure 4.19 – Attaques du mot 1 en CPA verticale



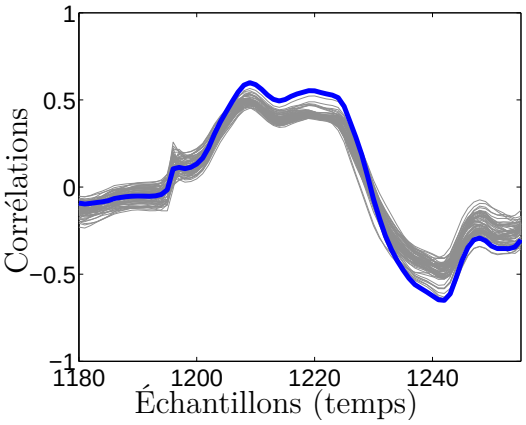
(a) Corrélations : 8 bits de poids faibles



(b) Corrélations : 16 bits de poids faibles



(c) Corrélations : 24 bits de poids faibles



(d) Corrélations : 32 bits du secret

Figure 4.20 – Attaques du mot 2 en CPA verticale

4.4 Conclusion

Nous avons proposé une étude théorique des attaques par canaux auxiliaires contre la multiplication de mots machine. Nous avons formalisé l'attaque afin de donner deux résultats théoriques, et confirmés en pratique, qui permettent l'optimisation des attaques. La méthode de diviser pour mieux régner qui s'applique au mot secret de 32 bits à besoin d'enchaîner des attaques intermédiaires. Nous avons proposé une méthode améliorée et montré théoriquement qu'elle fournit une attaque au meilleur taux de succès. Ce résultat est justifié en pratique. Un paramètre qui influence cette attaque est le nombre d'hypothèses de clefs retenues après les attaques intermédiaires. Nous montrons, là aussi en théorie et en pratique, que lorsque ce nombre augmente, le taux de succès de l'attaque augmente aussi.

Deux autres schémas d'attaques ont été étudiés. Les attaques par profilage, nous montrons qu'elles fonctionnent lorsqu'une multiplication de mots machine est ciblée. Enfin, nous implémentons une attaque horizontale, elle permet, lorsque P est secret, de retrouver ce secret dans une seule exécution du couplage de Ate $e_{m,A}(Q, P)$.

L'impact des travaux présentés dans ce chapitre est illustré par la table 4.4.

Table 4.4 – Résumé des attaques par canaux cachés

	Secret	P	
	Contre-mesure	Aucune	Randomisation des coordonnées
Attaques	CPA verticale	Une seule attaque pratique Amélioration d'un fact. 10	
	CPA horizontale		
	Secret	Q	
	Contre-mesure	Aucune	Randomisation des coordonnées
Attaques	CPA verticale	Pas de validation pratique Amélioration d'un fact. 10	
	CPA horizontale	1 seule trace	

Code couleur : Attaque relevée dans la littérature | Attaques en pratiques | Pas d'attaque identifiée

Chapitre 5

Attaques par canaux auxiliaires contre la randomisation des coordonnées

« Le requin qui ne nage pas se noie »

Proverbe Russe

Sommaire

Les attaques sur les implémentations non protégées de couplage sont efficaces, comme nous venons de le voir dans le chapitre 4. Les contre-mesures proposées dans la littérature ont été présentées dans la section 3.3.2. Nous nous sommes intéressés plus spécifiquement à la randomisation des coordonnées pour deux raisons. D'abord, il s'agit d'une contre-mesure connue dans la cryptographie asymétrique, elle a été proposée par Coron [Cor99] pour protéger le secret dans un cryptosystème à base d'ECC. Par ailleurs, la contre-mesure en question ajoute seulement quelques opérations dans le calcul du couplage. Son surcoût est très faible.

La section 5.1 de ce chapitre décrit la contre-mesure, et expose pourquoi elle permet de sécuriser l'implémentation des couplages contre les attaques vues dans le précédent chapitre. Nous étudions par la suite deux schémas d'attaques pour déjouer cette contre-mesure. La description et les résultats du premier schéma sont rapportés en section 5.2. Nous voyons des techniques d'attaques par profilage qui sont capables de déjouer certaines protections. Ce type de protection ciblée est un masquage assimilable à la randomisation sur les couplages. Nous verrons le second schéma d'attaque, en section 5.3. Nous nous attarderons sur une faille découverte dans la contre-mesure de randomisation des coordonnées. Nous montrons comment l'exploiter en explicitant un scénario et un schéma d'attaque. Ensuite, nous présenterons la mise en pratique de cette attaque. Enfin, dans la section 5.4 nous tirons une conclusion pour cette contre-mesure. La conséquence de notre nouvelle attaque est de mettre l'accent sur l'implémentation de la randomisation, pour aboutir à la sécurisation des couplages. Nous finissons par conclure avec une proposition pour l'implémentation du couplage Optimal Ate avec la contre-mesure modifiée.

5.1 Contre-mesure ciblée

L'implémentation du couplage nécessite de faire appel à l'algorithme de Miller. L'algorithme 2.6 permet de réaliser ce calcul. La contre-mesure de randomisation des coordonnées est une opération qui vient s'appliquer au début de l'algorithme de Miller pour ajouter de l'aléa sur le point connu en entrée de couplage.

5.1.1 Masquage avec Q secret

À l'origine, cette contre-mesure est présentée pour randomiser le point public P. Cette opération de randomisation intervient lors de l'affectation du point temporaire T à P. On profite de cette affectation pour passer le point T en coordonnées jacobiennes (ou projectives, mais les raisons d'efficacité discutées à la section 2.2.2, concluent que le choix jacobien est meilleur). En coordonnées jacobiennes, les coordonnées d'un point donné ne sont pas uniques, en effet elles satisfont l'équation :

$$(x, y) = \left(XZ^2 : YZ^3 : Z\right), \quad \forall Z \neq 0. \tag{5.1}$$

Le principe de la contre-mesure est de faire l'affectation de T avec un $\lambda \in \mathbb{F}_q^\star$ différent entre chaque exécution. L'algorithme de Miller, muni de cette contre-mesure, est décrit par l'algorithme 5.1. Il résulte de l'application des équations 7.19 et 7.20 de l'annexe C, page 171.

Entrées : Une courbe elliptique $E : y^2 = x^3 + ax + b$ sur $\mathbb{F}_q$,
$m = (m_{\mathfrak{n}-1} \dots m_0)_2$, $P \in E\left(\mathbb{F}_{q^k}\right)[m]$ et $Q \in E\left(\mathbb{F}_{q^k}\right)$.
Sorties : $f_{m,P}(Q)$ avec $\mathrm{div}(f_{m,P}) = m(P) - m(\mathcal{P}_\infty)$.

$f \leftarrow 1$;
Tirer $\lambda \in \mathbb{F}_q^\star$ aléatoire ;
$T = (X_T : Y_T : Z_T)$ avec $X_T \leftarrow x_P\lambda^2$; $Y_T \leftarrow y_P\lambda^3$; $Z_T \leftarrow \lambda$;
pour $i = \mathfrak{n} - 2$ **à 0 faire**
  $H \leftarrow 4X_TY_T^2$; $I \leftarrow 3X_T^2 + aZ_T^4$;
  $X_R \leftarrow -2H + I^2$; $Y_R \leftarrow -8Y_T^2 + I(H - X_R)$; $Z_R \leftarrow 2Y_TZ_T$;
  $l_{T,T}(Q) \leftarrow 2Y_TZ_T^3y_Q - 2Y_T^2 - I(x_QZ_T^2 - X_T)$; `// Tangente`
  $T \leftarrow R$; `// T est mis à jour (Doublement)`
  $f \leftarrow f^2 l_{T,T}(Q)$;
  si $m_i = 1$ **alors**
    $A \leftarrow y_PZ_T^3 - Y_T$; $B \leftarrow x_PZ_T^2 - X_T$;
    $X_R \leftarrow A^2 - B^2(X_T + x_PZ_T^2)$;
    $Y_R \leftarrow B^2(A(X_T - X_RZ_T^2) - BY_T)$;
    $Z_R \leftarrow Z_TB$;
    $l_{T,P}(Q) \leftarrow B(Z_T^3y_Q - Y_T) - A(Z_T^2x_Q - X_T)$; `// Ligne`
    $f \leftarrow f l_{T,P}(Q)$;
    $T \leftarrow R$; `// T est mis à jour (Addition)`
  fin
fin
retourner f ;

Algorithme 5.1 : Algorithme de Miller [Mil86] en coordonnées mixtes affines-jacobiennes avec randomisation (Q secret)

Une attaque CPA verticale vise les multiplications modulaires entre des coordonnées dépendantes du secret Q avec des entiers connus. Dans l'algorithme 5.1, les opérations qui impliquent x_Q et y_Q sont les suivantes :

- $2Y_TZ_T^3y_Q = H(T)y_Q$, avec la fonction $H(T) = 2Y_TZ_T^3$ qui dépend uniquement du point T.
- $x_QZ_T^2$.
- $Z_T^3y_Q$.
- $Z_T^2x_Q$.

Toutes ces multiplications sont opérées entre des entiers dépendants de Q et des entiers aléatoires. En effet, les coordonnées de T ne sont plus connues. La CPA ne peut plus être appliquée pour retrouver le secret Q car les prédictions sur les variables intermédiaires ne sont plus possibles. En effet, $\lambda \in \mathbb{F}_q^\star$ prend une valeur aléatoire dans $\{1, \dots, q-1\}$ donc λ^2 a $\frac{q-1}{2}$ valeurs possibles, ce qui implique que $X_T \leftarrow x_T\lambda^2$ est une valeur aléatoire parmi un ensemble de cardinal $\frac{q-1}{2}$. Les valeurs de X_T, Y_T et Z_T étant imprévisibles, il n'est plus possible de prédire les états internes de l'algorithme pour faire la CPA, cette attaque devient alors inefficace.

5.1.2 Masquage avec P secret

Dans le cas où P est le secret, il s'agit de rendre imprévisibles les valeurs intermédiaires liées à Q. La contre-mesure ne peut pas être appliquée de manière identique. Le stratagème d'utiliser l'affectation du point T n'est plus valide, ici, c'est le point Q qui doit être randomisé. Pour cela, une étape supplémentaire doit permettre la randomisation de Q par le biais des coordonnées jacobiennes. Les systèmes de coordonnées qui en découlent sont les suivants :

- P en coordonnées affines.
- Q en coordonnées jacobiennes pour pouvoir être randomisé.
- T en coordonnées jacobiennes pour l'efficacité des calculs $T \leftarrow [2]T$ et $T \leftarrow T+P$.

Le calcul combiné de l'addition $T+P$ et de la ligne $l_{T,P}(Q)$ se fait via les formules :

$$\begin{aligned} A &= y_P Z_T^3 - Y_T \\ B &= x_P Z_T^2 - X_T \\ X_{T+P} &= A^2 - B^2(X_T + x_P Z_T^2) \\ Y_{T+P} &= B^2(A(X_T - X_{T+P} Z_T^2) - BY_T) \\ Z_{T+P} &= Z_T B \\ l_{T,P}(Q) &= \frac{B(Y_Q Z_T^3 - Y_T Z_Q^3) - Z_Q A(X_Q Z_T^2 - X_T Z_Q^2)}{Z_Q^3 Z_T^3 B}. \end{aligned} \tag{5.2}$$

Pour le calcul combiné du doublement $[2]T$ et de la tangente $l_{T,T}(Q)$, il s'agit de la même formule que l'équation 7.18 de l'annexe C.

En termes d'efficacité, en reprenant le modèle de la table 2.7, nous donnons les nombres d'opérations pour ce système de coordonnées dans la table 5.1.

Table 5.1 – Complexité des calculs des additions/lignes et doublements/tangentes combinés selon le système de coordonnées résultant du masquage avec P secret

		Avec dénominateur			**Sans dénominateur**		
		M	***C***	***I***	***M***	***C***	***I***
Jacobiennes-affines	Addition/ligne	19	4	1	17	4	0
	Doublement/tangente	12	6	1	11	6	0

Le système de coordonnées dérivé de la représentation pour masquer le point Q en coordonnées jacobiennes n'est pas la représentation la plus efficace.

Les équations des calculs combinés 7.18 et 5.2 permettent d'écrire l'algorithme de Miller dans ce système de coordonnées (cf. algorithme 5.2).

L'objectif d'un attaquant qui cherche les opérations critiques pour faire une CPA verticale est d'identifier les opérations impliquant le secret P. Les opérations concernées interviennent dans le calcul de $T+P$ et de la ligne $l_{T,P}(Q)$ à travers les relations suivantes :

- $y_P Z_T^3$.
- $x_P Z_T^2$.

Entrées : Une courbe elliptique $E : y^2 = x^3 + ax + b$ sur $\mathbb{F}_q$,
$m = (m_{\mathfrak{n}-1} \dots m_0)_2$, $P \in E\left(\mathbb{F}_{q^k}\right)[m]$ et $Q \in E\left(\mathbb{F}_{q^k}\right)$.
Sorties : $f_{m,P}(Q)$ avec $\mathrm{div}(f_{m,P}) = m(P) - m(\mathcal{P}_\infty)$.

$f \leftarrow 1$;
$T = (X_T : Y_T : Z_T)$ avec $X_T \leftarrow x_P$; $Y_T \leftarrow y_P$; $Z_T \leftarrow 1$;
Tirer $\lambda \in \mathbb{F}_q^\star$ aléatoire ;
$Q = (X_Q : Y_Q : Z_Q)$ avec $X_Q \leftarrow x_Q\lambda^2$; $Y_Q \leftarrow y_Q\lambda^3$; $Z_Q \leftarrow \lambda$;
pour $i = \mathfrak{n} - 2$ **à 0 faire**
  $H \leftarrow 4X_TY_T^2$; $I \leftarrow 3X_T^2 + aZ_T^4$;
  $X_R \leftarrow -2H + I^2$; $Y_R \leftarrow -8Y_T^2 + I(H - X_R)$; $Z_R \leftarrow Y_TZ_T$;
  $l_{T,T}(Q) \leftarrow 2Y_TZ_T^2Y_Q - 2Y_T^2Z_Q^3 - Z_QI(X_QZ_T^2 - X_TZ_Q^2)$; `// Tangente`
  $T \leftarrow R$; `// T est mis à jour (Doublement)`
  $f \leftarrow f^2 l_{T,T}(Q)$;
  si $m_i = 1$ **alors**
    $A \leftarrow y_PZ_T^3 - Y_T$; $B \leftarrow x_PZ_T^2 - X_T$;
    $X_R \leftarrow A^2 - B^2(X_T + x_PZ_T^2)$;
    $Y_R \leftarrow B^2(A(X_T - X_RZ_T^2) - BY_T)$;
    $Z_R \leftarrow Z_TB$;
    $l_{T,P}(Q) \leftarrow B(Y_QZ_T^3 - Y_TZ_Q^3) - Z_QA(X_QZ_T^2 - X_TZ_Q^2)$; `// Ligne`
    $f \leftarrow f l_{T,P}(Q)$;
    $T \leftarrow R$; `// T est mis à jour (Addition)`
  fin
fin
retourner f ;

Algorithme 5.2 : Algorithme de Miller [Mil86] en coordonnées mixtes affines-jacobiennes avec randomisation (P secret)

Comme le point T dépend du secret P, il n'y a aucune interaction avec les coordonnées du point connu. Le principe de base des attaques visant une telle interaction n'est donc pas atteint, ces attaques ne sont pas réalisables.

Il faut considérer les opérations qui manipulent les coordonnées de Q, on retrouve :

- $Y_QZ_T^3$.
- $Y_TZ_Q^3$.
- $X_TZ_Q^2$.

L'objectif de l'attaque est d'utiliser la connaissance de Q pour retrouver les coordonnées de T, puis de remonter à P. Mais, ce n'est plus possible à cause de la randomisation des coordonnées de Q. Les valeurs de X_Q, Y_Q et Z_Q étant imprévisibles, il n'est plus possible de prédire les états internes de l'algorithme pour faire la CPA, cette attaque devient alors inefficace.

Cette analyse est représentative de la littérature sur les contre-mesures. Elle permet de justifier que les attaques des versions protégées des algorithmes de couplage ne sont plus faisables. Sans pour autant que leurs efficacités soient prouvées faces à d'autres schémas d'attaques.

La suite du chapitre considère des classes d'attaques par canaux auxiliaires différentes. Nous allons d'abord voir les attaques par profilage sans connaissance des messages, puis nous verrons le cas des attaques en collisions.

5.2 Attaques par profilage sur les masquages

D'après la section précédente, la situation dans laquelle un attaquant se trouve à cause de la contre-mesure est la suivante : il est capable d'observer des multiplications modulaires $a \times b \mod q$ avec b secret et a aléatoire et inconnu (c'est le cas dans l'algorithme 5.1 avec l'opération $Z_T^3 y_Q$). La littérature propose des attaques contre des implémentations d'opérations masquées. De telles attaques sont, par exemple, décrites contre les chiffrements par blocs par Oswald *et al.* [OM06] et Hanley *et al.* [HTM09].

De plus, nous nous intéressons à cette implémentation masquée, car la présence d'opérations manipulant le secret est source de fuites. Les données manipulées dépendent du secret donc les mesures des canaux auxiliaires aussi. De l'information sur le secret est contenue dans les signaux. Nous allons essayer d'extraire cette information grâce à ces attaques par profilage sans connaissance des messages.

5.2.1 Description de l'attaque d'Oswald *et al.* sur l'AES

La phase de caractérisation ne pose pas de problème particulier, l'attaquant a un contrôle total sur les entrées et les variables de l'algorithme. Il peut donc caractériser des exécutions avec des clefs, des messages et des masques différents et connus. La phase d'attaque ne peut plus être faite de la même manière que sans la contre-mesure de masquage. Le masquage consiste en un XOR avec les variables intermédiaires[1]. Le masque (inconnu) rend aléatoires les états intermédiaires de l'algorithme de chiffrement. Il est maintenant impossible de faire les prédictions sur les valeurs intermédiaires. Cela implique qu'il faut faire la phase d'attaque en prenant en compte toutes les valeurs possibles du masque $\lambda \in \Lambda$. Ainsi, lors de la phase d'attaque, la probabilité d'une trace $C'^{(i)}$ sachant une clef k_j doit être remplacée par la probabilité d'une trace $C'^{(i)}$ sachant une clef k_j et un masque λ :

$$\mathbb{P}\left(C'^{(i)} | k_j \text{ ET } \lambda\right). \tag{5.3}$$

Phase de profilage Le but est de parcourir tous les masques, la formule des probabilités totale permet d'écrire $\mathbb{P}\left(C'^{(i)} | k_j\right)$ en fonction des probabilités conditionnelles données dans l'équation 5.3. On obtient donc :

$$\mathbb{P}\left(C'^{(i)} | k_j\right) = \sum_{\lambda \in \Lambda} \mathbb{P}\left(C'^{(i)} | k_j \text{ ET } \lambda\right) \mathbb{P}(\lambda). \tag{5.4}$$

Avec $\mathbb{P}\left(C'^{(i)} | k_j\right)$ on peut calculer $\mathbb{P}\left(k_j | \mathcal{C}'\right)$ comme dans la phase d'attaque classique grâce à l'équation 3.12, page 74.

Cette attaque a été testée par ses auteurs [OM06] contre une implémentation masquée d'un AES. Les données sont sur 8 bits, cela laisse donc 9 valeurs de poids de

[1] De tels masquages sont plus détaillés dans [AG03, BGK04].

Hamming possibles. Les classes sont les couples :

$$\left\langle HW_8(\lambda), HW_8\left(SubBytes\left(x^{(i)} \oplus k_j\right) \oplus \lambda\right)\right\rangle. \tag{5.5}$$

Il y a donc 81 profils qui sont créés.

Phase d'attaque Pour un message connu $x^{(i)}$, un masque donné λ et une hypothèse de clef k_j, la probabilité $\mathbb{P}\left(C'^{(i)}|k_j \text{ ET } \lambda\right)$ est calculée via :

$$\mathbb{P}\left(C'^{(i)}|k_j \text{ ET } \lambda\right) = \mathbb{P}\left(C'^{(i)}|\left\langle HW_8(\lambda), HW_8\left(SubBytes\left(x^{(i)} \oplus k_j\right) \oplus \lambda\right)\right\rangle\right). \tag{5.6}$$

Dans l'ordre des choses, l'attaquant calcule d'abord $\mathbb{P}\left(C'^{(i)}|k_j \text{ ET } \lambda\right)$. L'équation 5.4 lui permet ensuite de calculer $\mathbb{P}\left(C'^{(i)}|k_j\right)$, il se retrouve alors dans une situation d'attaque par profilage classique. L'équation 3.12 permet de calculer la probabilité $\mathbb{P}\left(k_j|\mathcal{C}'\right)$ d'une hypothèse de clef k_j sachant l'ensemble de traces $\mathcal{C}'$.

Pour faire leur attaque, Oswald *et al.* [OM06] prennent des signaux contenant les instants temporels où λ, $x^{(i)} \oplus k_j \oplus \lambda$ et $SubBytes\left(x^{(i)} \oplus k_j\right) \oplus \lambda$ sont calculés. Il s'agit d'une implémentation logicielle de l'AES masquée sur un microcontrôleur 8 bits. À partir d'une quinzaine de traces, la bonne clef se distingue en ayant une probabilité de 1. La figure 5.1 montre les résultats de leur attaque, la ligne en gras représente la probabilité de la bonne clef en fonction du nombre de traces N_A pour la phase d'attaque.

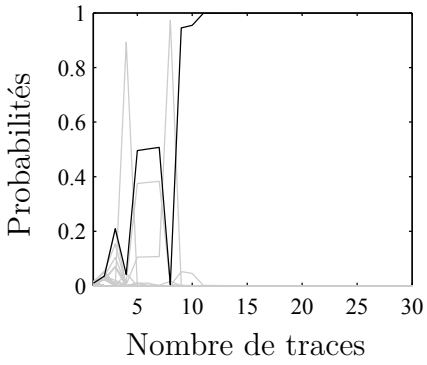


Figure 5.1 – Résultats d'une attaque par profilage sur une implémentation masquée de l'AES [OM06]

C'est une attaque prometteuse qui cible les masquages. Nous allons essayer cette approche en l'appliquant au cas d'une implémentation masquée d'un couplage.

5.2.2 Application de l'attaque sur les couplages

Nous cherchons à identifier les similitudes entre les couplages avec randomisation des coordonnées et la situation de l'AES masqué. Dans cette section, nous présentons un jeu de test.

Schéma d'attaque

La situation n'est pas exactement la même, nous nous plaçons dans le contexte suivant : dans l'algorithme 5.2, on recherche les opérations $Y_Q \leftarrow y_Q\lambda^3$ dans le calcul de la tangente $2Y_TZ_T^2Y_Q$. Lors de la première itération de la boucle, T n'a pas évolué, donc $Z_T = 1$. L'opération $2 \cdot s$ (avec $s = Y_TZ_T^2Y_Q$) peut être réalisée en additionnant $s + s$. Pour résumer, cette opération est réalisée de la manière suivante :

1. $t_0 \leftarrow Y_TY_Q$ (t_0 étant un premier registre).
2. $t_1 \leftarrow t_0 + t_0$ (t_1 est un second registre).

Puisque $Y_Q \leftarrow y_Q\lambda^3$ masque les données connues, $t_0 \leftarrow Y_TY_Q$ manipule ces données avec des données secrètes (Y_T dépends du point secret P).

Le but d'une première attaque est de retrouver les 8 bits de poids faibles de Y_T, tout comme nous l'avons fait avec une CPA optimisée à la section 4.3.

Sur le modèle d'Oswald *et al.* [OM06], nous cherchons comment construire des classes pour créer les profils. L'idée générale est de prendre en considération le poids de Hamming du masque et la sortie d'une opération ciblée. Cette opération est une fois de plus le *SubBytes* pour l'AES. Dans la situation des couplages, il s'agit d'une multiplication modulaire[2]. Les classes choisies sont construites par un couple. Le premier élément est le poids de Hamming du masque. Le second est la sortie de l'opération ciblée, c'est-à-dire la multiplication $\left(\lambda x^{(i)}\right) k^{(j)}$, prise sur 16 bits. L'équation 5.7 formalise la construction choisie :

$$\left\langle HW_8(\lambda), HW_{16}\left(\left(\lambda x^{(i)}\right) k^{(j)}\right)\right\rangle. \tag{5.7}$$

L'indice en bas de HW indique sur combien de bits est calculé le poids de Hamming. Le poids de Hamming, du produit avec la clef $k^{(j)}$, est mesuré sur 16 bits, effectivement, nous nous servons de notre expérience sur les modèles de fuites (cf. section 4.3) pour reproduire la meilleure fuite possible.

Enfin, les instants temporels sélectionnés pour faire l'attaque correspondent aux opérations $\lambda x^{(i)}$ et $\left(\lambda x^{(i)}\right) k^{(j)}$.

Mise en pratique de l'attaque

Avec la configuration donnée ci-dessus, nous utilisons un ensemble de 100000 mesures d'émanations électromagnétiques des opérations $\lambda x^{(i)}$ et $\left(\lambda x^{(i)}\right) k^{(j)}$. Pour cela, le code C embarqué correspond au pseudo-langage donné dans l'algorithme 5.3.

L'allure générale des courbes obtenues est présentée en figure 5.2.

Pour chaque trace, la connaissance du message $x \in \{0, \ldots, 2^8 - 1\}$, du masque $\lambda \in \{0, \ldots, 2^8 - 1\}$ et de la clef $k \in \{0, \ldots, 2^8 - 1\}$ permet de calculer la classe correspondante. La trace en question est dirigée vers l'ensemble des traces de cette classe. Une fois toutes les traces traitées, nous obtenons les $9 \times 17 = 153$ profils. Comme nous avions réussi l'attaque par profilage classique contre la multiplication à la section 4.3.4, sans prendre les covariances, nous ne les prenons pas non plus ici. Les profils sont simplement les moyennes $\left(\overline{C_{cl}}\right)_{0 \leq cl \leq 152}$.

[2] C'est l'une des opérations qui manipulent les données secrètes et connues.

```
1 pour i = 1 à 100000 faire
2 |   USART_receive_integer(message) ;  // Lecture du message sur l'interface UART
3 |   USART_receive_integer(lambda) ;   // Lecture du masque sur l'interface UART
4 |   USART_receive_integer(secret) ;   // Lecture du secret sur l'interface UART
5 |   GPIOB− >BSRR = GPIO_Pin_8 ;              // Signal de déclenchement HAUT
6 |   (acc, prod) = lambda_0 × message_0 ;  // Multiplication entre les mots de poids
  |    faibles du masque et du message
7 |   (acc, prod) = prod × secret_0 ;       // Multiplication entre les mots de poids
  |    faibles du résultat avec le mot de poids faible du secret
8 |   GPIOB− >BRR = GPIO_Pin_8 ;               // Signal de déclenchement BAS
9 fin
```

Algorithme 5.3 : Code C embarqué pour tester les attaques par profilage sur la multiplication masquée (pseudo-langage)

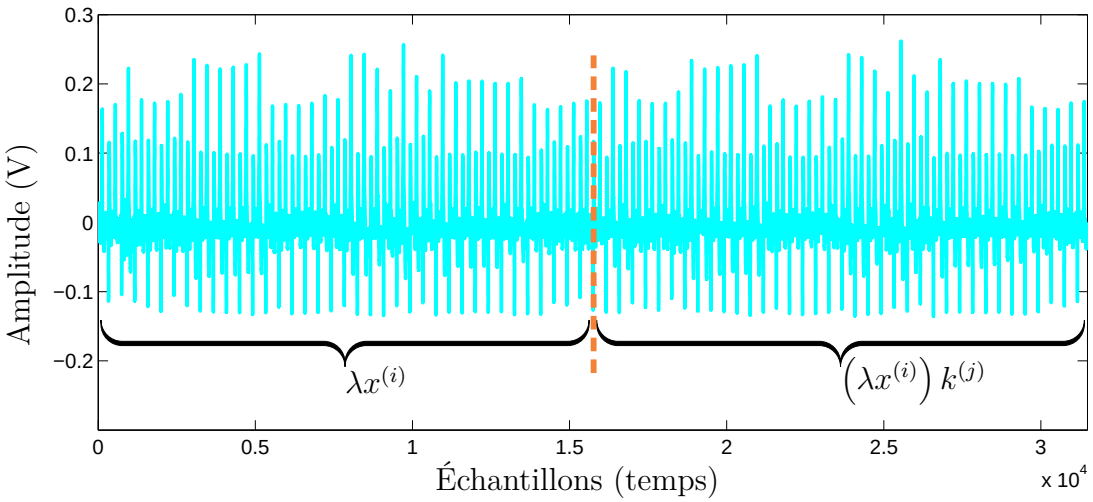


Figure 5.2 – Une trace acquise pour l'attaque par profilage contre la multiplication masquée (phase de profilage)

La phase d'attaque décrite ci-dessus est donc effectuée en insérant une somme sur toutes les valeurs possibles du masque λ. Le résultat de l'attaque est sous forme d'un vecteur de vraisemblance L qui attribue une probabilité pour chacune des clefs. La figure 5.3 montre les résultats avec 100000 traces pour les profilages et 10000 pour la phase d'attaque. La bonne clef (le carré rouge) est complètement perdue avec les autres, l'attaque ne permet pas de la distinguer.

Conclusion

L'attaque n'ayant pas donné un résultat convaincant, nous faisons varier les paramètres :

- Le nombre de bits utilisés dans les modèles en poids de Hamming.
- Le fait de prendre les logarithmes de vraisemblances : pour éviter de faire des produits entre des probabilités proches de 0, ce qui évite d'engendrer des erreurs numériques.
- La synchronisation des traces pour les différents points d'intérêts.

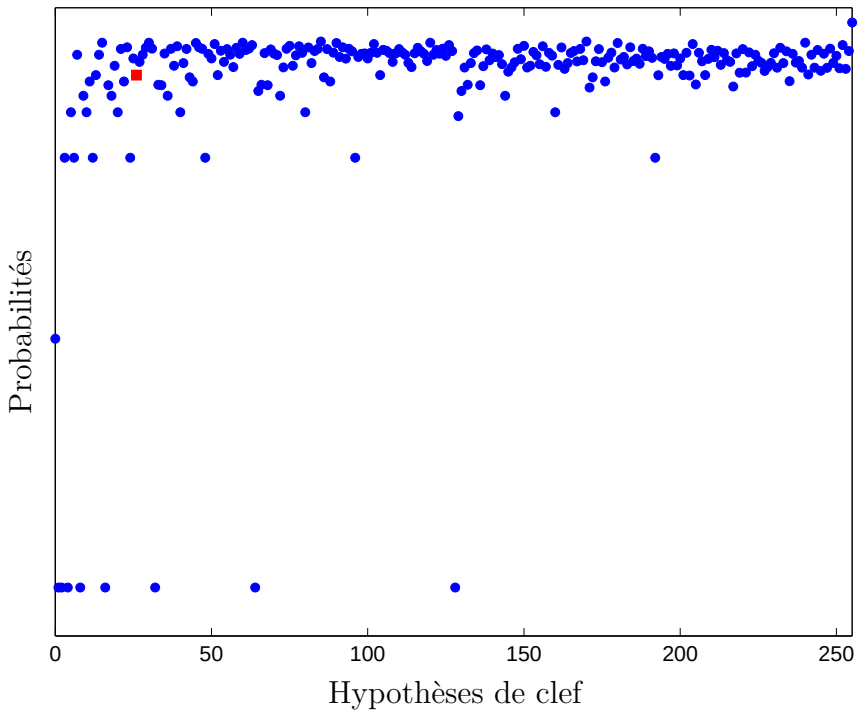


Figure 5.3 – Résultats d'une l'attaque par profilage contre la multiplication masquée

- La manipulation avec les matrices de variances-covariances (voir par exemple Choudary *et al.* [CK13] : une matrice par classe, une matrice commune, la matrice identité).

Ces différents paramétrages n'ont pas permis d'avoir une attaque capable de distinguer la bonne clef.

Le manque de résultat probant de cette attaque s'explique par le fait que la situation de Oswald *et al.* n'est pas évidente à obtenir dans le cadre d'une attaque contre la randomisation des coordonnées dans les couplages. Parallèlement à ces recherches, nous étudions comment l'introduction de la randomisation est réalisée. L'opération insérée est une multiplication entre le masque et la valeur connue en entrée de couplage. Puisque l'attaque par profilage contre le masquage n'a pas donnée les résultats escomptés, nous avons étudié l'intégration même de la contre-mesure. La faille que nous allons chercher à exploiter et une répétition d'opérations. Si elle est détectable en cherchant des collisions dans les mesures des canaux auxiliaires alors la clef secrète pourra être distinguée.

5.3 Attaque en collisions contre la randomisation des coordonnées

Dans le scénario de l'IBE, le déchiffrement d'un chiffré $\{U, V\}$, par Bob, passe par l'étape $K = e(d_B, U)$ où d_B est la clef secrète de Bob. Si Bob perd le dispositif où est exécuté $e(d_B, U)$, alors un attaquant peut maîtriser les chiffrés U qui sont manipulés. L'attaque que nous proposons est une attaque à message choisi.

5.3.1 Cas d'étude de base

Le calcul $K = e(d_B, U)$ est réalisé avec l'algorithme de Miller, où la contre-mesure de randomisation des coordonnées est utilisée, il s'agit donc de l'algorithme 5.2 avec « P est secret », c'est le cas dans $K = e(d_B, U)$. Si, en revanche c'est « Q le secret », par exemple $K = e(U, d_B)$, alors c'est l'algorithme 5.1 qui sera exécuté (plus efficace que l'algorithme 5.2).

Prenons le cas « Q secret », attardons-nous sur les opérations de randomisation et le calcul de la tangente à la première itération. La randomisation du point connu P est effectuée à la ligne 3, nous nous concentrons sur l'opération $X_T \leftarrow x_P\lambda^2$. Dans le calcul de la tangente à la première itération, le calcul $x_Q Z_T^2$ est réalisé avec $Z_T = \lambda$. Les deux opérations qui sont réalisées sont des multiplications entre une coordonnée connue (x_P) avec un masque $\lambda' = \lambda^2$, puis une coordonnée secrète (x_Q) avec le même masque. Dans la suite, nous noterons simplement λ au lieu de λ'.

Ainsi, si $x_P = x_Q$ alors le même calcul est exécuté deux fois, si les canaux auxiliaires permettent d'identifier une telle situation grâce à une technique de détection de collisions alors l'attaquant est en mesure de retrouver le secret x_Q. Cette valeur est la même que la coordonnée x_P qu'il connaît.

Si les valeurs de x_P et x_Q sont égales (ou partiellement égales) alors les traces mesurées sont similaires. Les données x_P et x_Q sont des entiers multiprécisions, par exemple 8 mots de 32 bits. Il est donc impossible de tester les 2^{256} valeurs pour x_P. Cependant, le dispositif réalise la multiplication modulaire sur les entiers multiprécisions en traitant les mots individuellement et de façon déterministe. Par exemple, la multiplication modulaire de Montgomery va à un instant précis, faire la multiplication $x_0\lambda_0$ en notant $x = x_P$ pour faciliter la lecture. Même avec cette remarque, l'attaque consisterait à faire une recherche exhaustive sur 32 bits. Comme pour faire la CPA, l'attaque des 32 bits est faite octet par octet. C'est pour cela que les mots x_0 et k_0 sont dits partiellement égaux s'ils ont leurs octets de poids faibles identiques (k correspond à la coordonnée secrète, c'est-à-dire x_Q).

La suite de cette section concerne la détection de similitudes entre deux mots x_0 et k_0 lors de leur manipulation dans une multiplication avec le même entier λ inconnu.

5.3.2 Problématique de la détection de collisions

Pour maximiser l'effet des bits de poids faibles du mot connu x_0 il suffit de fixer les bits de poids forts à 0. De cette manière, les mots $x_0^{(i)}$ utilisés sont tout simplement $x_0^{(i)} = i$, d'un point de vue binaire :

$$\begin{aligned} x_0^{(0)} &= (00\ldots0\mathbf{00000000})_2 \\ x_0^{(1)} &= (00\ldots0\mathbf{00000001})_2 \\ &\vdots \\ x_0^{(255)} &= (00\ldots0\mathbf{11111111})_2. \end{aligned} \tag{5.8}$$

La problématique est donc la suivante : à partir des traces correspondant aux opérations $x_0^{(i)}\lambda_0^{(j)}$ et $k_0\lambda_0^{(j)}$, il faut arriver à dire si $LSB8\left(x_0^{(i)}\right) = LSB8\left(k_0\right)$. Lorsqu'il y a une telle égalité, on dit qu'il y a une collision.

Le premier outil qui vient à l'esprit pour comparer deux traces est la corrélation croisée entre ces deux vecteurs de données. On note $c^{(i,j)}$ le coefficient de corrélation croisé entre les traces $C_x^{(i,j)}$ et $C_k^{(i,j)}$, correspondant respectivement aux opérations $x_0^{(i)}\lambda_0^{(j)}$ et $k_0\lambda_0^{(j)}$. Avec cet outil, nous donnons une esquisse de l'attaque dans l'algorithme 5.4.

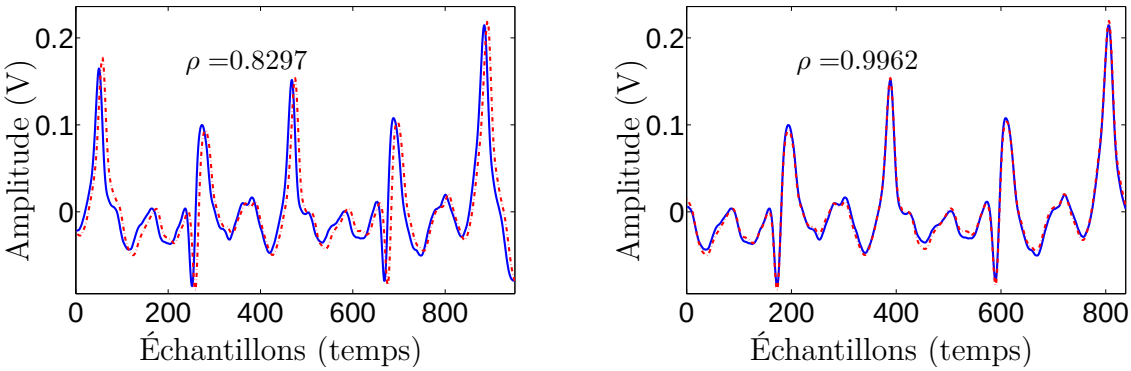
Entrées : Les messages $x_0^{(i)}$ de l'équation 5.8, un entier positif N.
Sorties : Un candidat $\widehat{k}$ pour les 8 bits de poids faibles de $k^\star$.

$c \leftarrow \mathcal{M}_{0,255}(0)$;
pour $i = 0$ **à 255 faire**
  pour $j = 0$ **à** N **faire**
    $C_x^{(i,j)} \leftarrow$ trace de l'opération $x_0^{(i)}\lambda_0^{(j)}$;
    $C_k^{(i,j)} \leftarrow$ trace de l'opération $k_0\lambda_0^{(j)}$;
    $c^{(i,j)} \leftarrow \rho\left(C_x^{(i,j)}, C_k^{(i,j)}\right)$;
  fin
  $c(i) \leftarrow \frac{1}{N}\sum_j c^{(i,j)}$;
fin
$\widehat{k} \leftarrow \arg\max |c|$;
retourner $\widehat{k}$;

Algorithme 5.4 : Schéma de l'attaque en collision par corrélations horizontales

Dans le schéma d'attaque proposé dans l'algorithme 5.4, l'entier positif N désigne une certaine force de l'attaquant. N est le nombre de répétitions que l'attaquant peut faire avec un même message $x_0^{(i)}$. Après avoir fait les N mesures, les coefficients de corrélations croisées sont calculés puis moyennés. Concrètement, le fait de moyenner les corrélations plutôt que les traces est utile pour donner une valeur de la similitude réelle entre les opérations $x_0^{(i)}\lambda_0^{(j)}$ et $k_0\lambda_0^{(j)}$ sans prendre en compte l'effet du masque.

L'un des problèmes majeurs de la détection de collision, de surcroît avec l'outil de corrélation croisé, vient de la synchronisation des traces. En effet, le coefficient de corrélation croisée entre deux traces est très sensible à l'alignement de ces dernières. La figure 5.4 montre un exemple de coefficient de corrélation croisée entre deux traces avec une paire brute et une paire réalignée.



(a) Traces brutes

(b) Traces synchronisées sur le premier pic (décalage de 7 points)

Figure 5.4 – Effet de l'alignement des traces sur le coefficient de corrélation

L'attaque donne un coefficient de corrélation pour chacune des hypothèses de clef. On fait donc un classement de ces hypothèses, l'hypothèse rangée en première position est la clef candidate. Une fois le classement effectué, le rang de la bonne hypothèse de clef est retenu. Un rang égal à 1 indique que la bonne clef est distinguée en première position, donc l'attaque fonctionne. Le résultat de l'attaque avec la moyenne des coefficients de corrélations croisées est donné en figure 5.5.

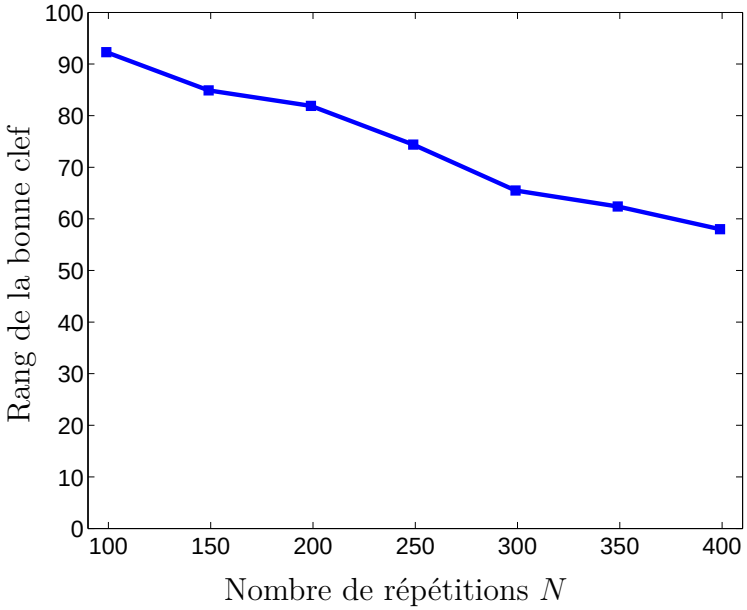


Figure 5.5 – Résultats de l'attaque contre la multiplication masquée par collision, détection par corrélations horizontales

Comme le montre la figure 5.5, l'attaque est un échec, la bonne sous-clef n'est pas retrouvée.

L'utilisation de la corrélation croisée pour détecter la collision entre deux traces est un outil d'analyse qui compare les traces de façon horizontale. Dans la suite, nous allons voir un outil qui apporte une vision verticale.

Description de la détection verticale

Dans l'approche horizontale, les traces sont comparées deux à deux avec un coefficient de similitude, puis moyennées par la suite. L'approche verticale consiste à comparer directement la base de données, sans passer par une moyenne des coefficients de corrélations. Pour cela, on désigne un point temporel pris dans les traces $\left\{C_x^{(i,j)}\right\}_j$, ces données sont mises sous forme d'un vecteur de dimension N. Puis, on désigne aussi un point dans les traces $\left\{C_k^{(i,j)}\right\}_j$, ces données sont aussi mises sous forme d'un vecteur. Enfin, le coefficient de corrélation de ces deux vecteurs est calculé. La figure 5.6 illustre ce principe.

Plus précisément, cette attaque proposée initialement contre ECDSA par Varchola *et al.* [VDRZ15] fonctionne de la manière suivante : Pour une entrée $x_0^{(i)}$, les N traces du canal auxiliaire peuvent être vues comme la matrice R_i de l'équation 5.9, m_x et m_k indiquent le nombre de points qui composent les traces correspondant respectivement à la première puis la seconde multiplication.

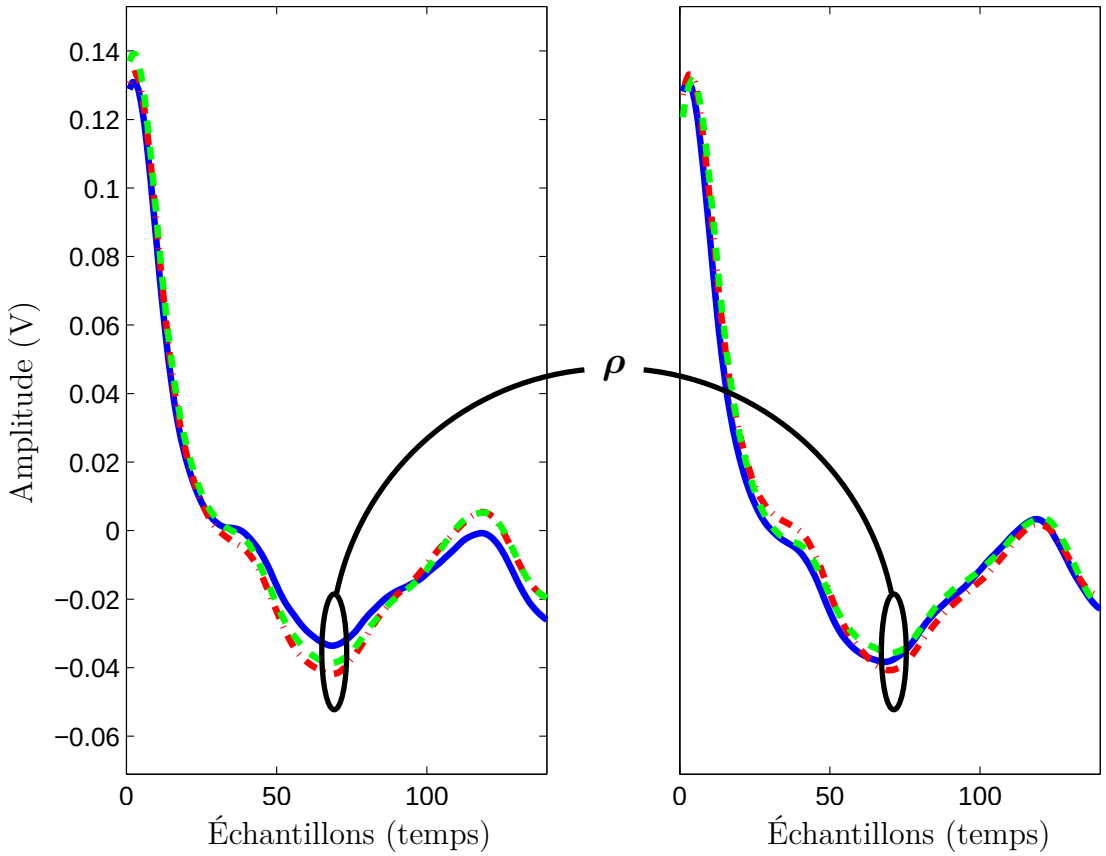


Figure 5.6 – Illustration de la détection de collision verticale

$$R_i = \begin{pmatrix} C_{x,1}^{(i,1)} & C_{x,2}^{(i,1)} & \dots & C_{x,m_x}^{(i,1)} & C_{k,1}^{(i,1)} & \dots & C_{k,m_k}^{(i,1)} \\ C_{x,1}^{(i,2)} & \dots & & C_{x,m_x}^{(i,2)} & C_{k,1}^{(i,2)} & \dots & C_{k,m_k}^{(i,2)} \\ \vdots & & & \vdots & \vdots & & \vdots \\ C_{x,1}^{(i,N)} & \dots & & C_{x,m_x}^{(i,N)} & C_{k,1}^{(i,N)} & \dots & C_{k,m_k}^{(i,N)} \end{pmatrix}. \tag{5.9}$$

Avec $C_{(\cdot,\cdot)}^{(\cdot,\cdot)}$ les traces obtenues par le procédé décrit par l'algorithme 5.4.

À partir de la matrice R_i pour une hypothèse de clef, l'attaquant crée une « trace de corrélation ». Il choisit un instant $1 \leq m_{\text{interset}} \leq m_x$ dans la première multiplication, et un intervalle $I \subset \{1, \dots, m_k\}$ dans la seconde multiplication. La trace de corrélation est obtenue en calculant la corrélation entre le vecteur colonne $\left(C_{x,m_{\text{interset}}}^{(i,1)} C_{x,m_{\text{interset}}}^{(i,2)} \cdots C_{x,m_{\text{interset}}}^{(i,N)}\right)^T$ et les vecteurs colonnes $\left(C_{k,m}^{(i,1)} C_{k,m}^{(i,2)} \cdots C_{k,m}^{(i,N)}\right)^T$ pour tout $m \in I$. C'est-à-dire, le calcul de l'équation 5.10 :

$$c_i = \rho\left(\begin{pmatrix} C_{x,m_{\text{interest}}}^{(i,1)} \\ \vdots \\ C_{x,m_{\text{interest}}}^{(i,N)} \end{pmatrix}, \begin{pmatrix} C_{k,m}^{(i,1)} \\ \vdots \\ C_{k,m}^{(i,N)} \end{pmatrix}\right), \forall m \in I. \tag{5.10}$$

L'attaque se termine en renvoyant la clef candidate qui maximise les corrélations, c'est-à-dire $\widehat{k} \leftarrow \arg\max_i |c_i|$.

Application au cas des couplages

Afin de vérifier la faisabilité de l'attaque contre la multiplication de mots machine de 32 bits nous ciblons l'implémentation logicielle des multiplications $\lambda_0 x_0$ et $\lambda_0 k_0$.

Approche théorique Ce cas d'étude permet de reproduire la situation d'une implémentation de l'algorithme de Miller. La cible de notre attaque est une contre-mesure qui introduit une étape de randomisation sur la coordonnée publique et qui se répète sur la coordonnée secrète.

Dans le cas de la randomisation du point public P, comme sur l'algorithme 5.1, cette répétition est atteinte aux lignes 3 :

$$T = (X_T : Y_T : Z_T) \text{ avec } X_T \leftarrow \boldsymbol{x_P \lambda^2}; \quad Y_T \leftarrow y_P \lambda^3; \quad Z_T \leftarrow \lambda,$$

et 7 :

$$l_{T,T}(Q) \leftarrow 2Y_T Z_T^3 y_Q - 2Y_T^2 - I(\boldsymbol{x_Q Z_T^2} - X_T).$$

Mise en pratique Nous utilisons le code C embarqué en adéquation avec le pseudo-langage présenté dans l'algorithme 5.5. Il reproduit la répétition des multiplications critiques.

```
1  USART_receive_integer(secret) ;                          // Lecture du secret
2  pour i = 1 à 255 faire
3  |  USART_receive_integer(message) ;                      // Lecture du message
4  |  pour j = 1 à N faire
5  |  |  USART_receive_integer(lambda) ;                    // Lecture du masque
6  |  |  GPIOB- >BSRR = GPIO_Pin_8 ;          // Signal de déclenchement HAUT
7  |  |  (acc, prod) = lambda_0 × message_0 ;  // Multiplication entre les mots de
   |  |   poids faibles du masque et du message
8  |  |  (acc, prod) = lambda_0 × secret_0 ; // Multiplication entre les mots de poids
   |  |   faibles du masque et du secret
9  |  |  GPIOB- >BRR = GPIO_Pin_8 ;           // Signal de déclenchement BAS
10 |  fin
11 fin
```

Algorithme 5.5 : Forme du code C embarqué pour tester les attaques en collisions contre la randomisation des coordonnées

La forme d'une telle mesure est donnée dans la figure 5.7.

La figure 5.8 montre les corrélations c_i pour une attaque avec détection de collisions par corrélation verticales avec $N = 400$ répétitions. La courbe plus épaisse correspond à l'hypothèse de sous-clef correcte.

Nous voyons sur la figure 5.8 que l'attaque a permis de distinguer la bonne hypothèse de sous-clef. La courbe épaisse bleue, correspondant à la bonne hypothèse de sous-clef (sur 8 bits) a une corrélation plus élevée que les autres.

Varchola *et al.* [VDRZ15] proposent une attaque avec $N = 256$ répétitions, mais leur environnement se distingue du nôtre pour plusieurs raisons. La table 5.2 résume ces différences.

La différence de l'implémentation, matérielle dans le cas de Varchola *et al.* [VDRZ15] et logicielle dans notre cas, est bien sûr essentielle. Mais dans le cadre de notre attaque

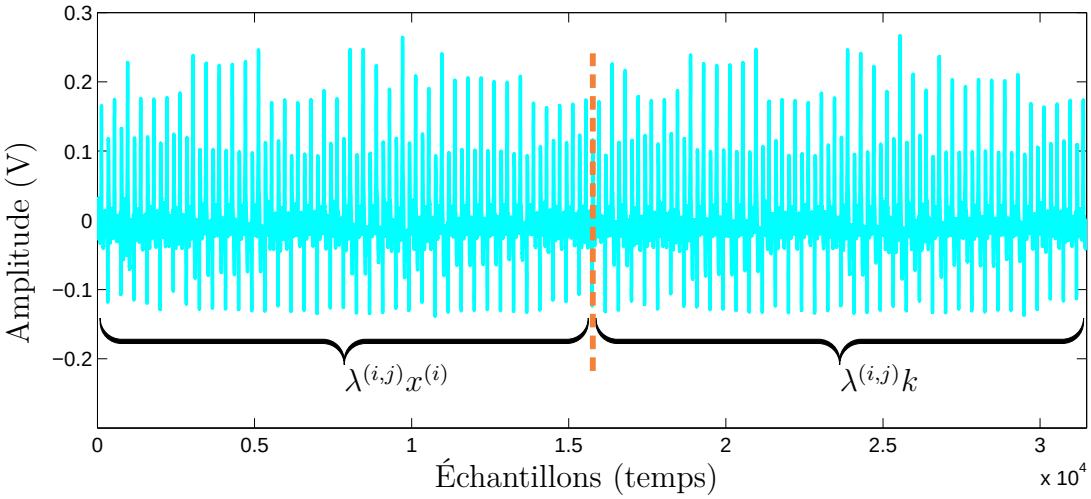


Figure 5.7 – Une trace acquise pour l'attaque en collisions contre la randomisation des coordonnées

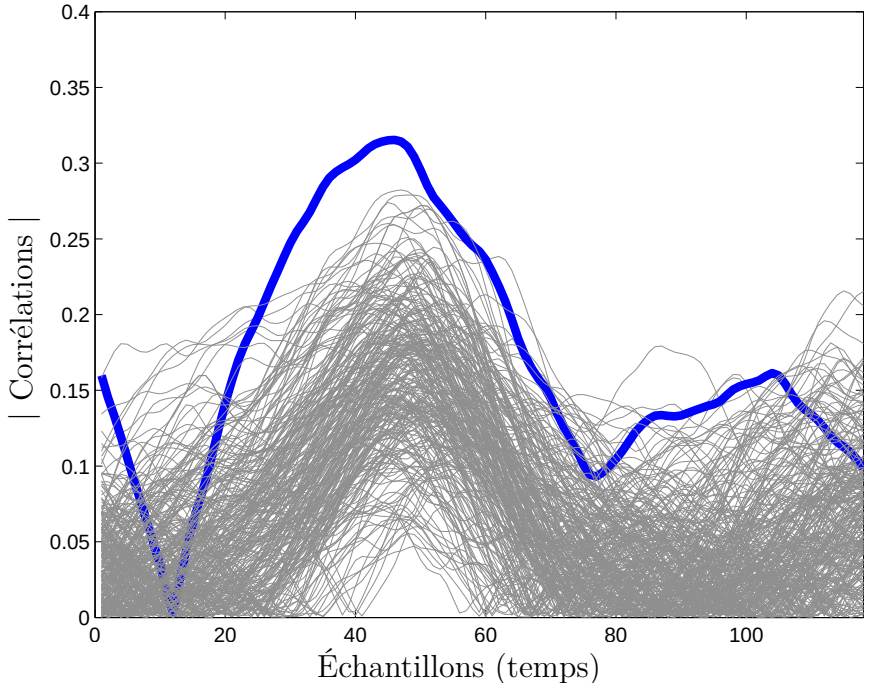


Figure 5.8 – Corrélations issues d'une attaque en collisions verticales

en collisions, c'est la taille de l'architecture qui joue un rôle fondamental. Nous subissons l'effet de la taille des mots de l'architecture sur la notion d'entiers « partiellement égaux ». En effet, sur une architecture 8 bits, l'implémentation de la multiplication modulaire traite les mots (de 8 bits) $x_0\lambda_0$ et $k_0\lambda_0$, donc l'égalité $x_0 = k_0$ est avérée pour un mot machine complet, ce qui se traduit par le fait que les deux multiplications $x_0\lambda_0$ et $k_0\lambda_0$ sont réalisées exactement de la même manière par le processeur. Ce n'est pas le cas lorsqu'on cible les 8 bits de poids faibles des mots machine de 32 bits. Comme nous le montrons en pratique, sur une implémentation logicielle en architecture 32 bits, les fuites sont suffisantes pour détecter des collisions sur 8 bits.

Puisque l'attaque a fonctionné avec $N = 400$ répétitions (c'est-à-dire $400 \times 256 = 102400$ traces), nous faisons varier ce paramètre afin de voir l'évolution du rang de la bonne sous-clef. Pour $N \in \{100, 110, \ldots, 400\}$, la figure 5.9a montre cette évolution.

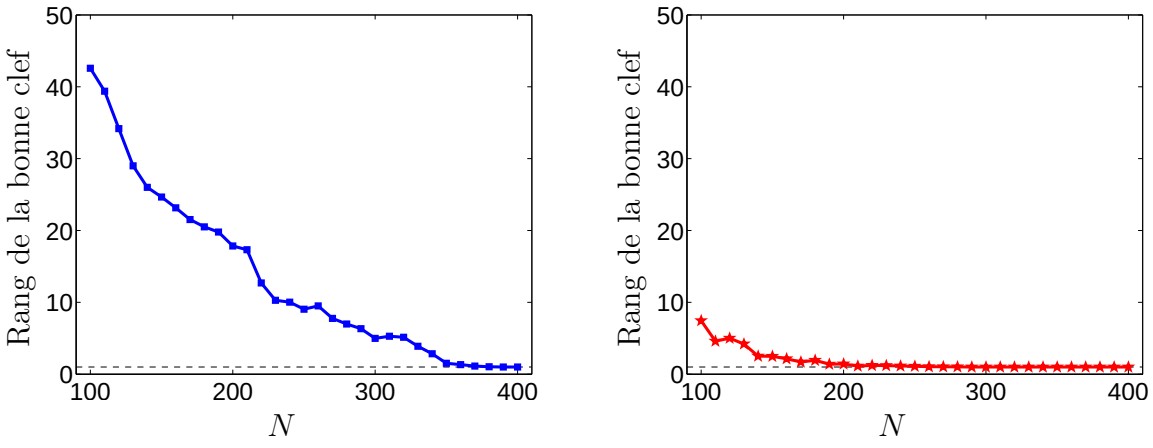
Entrées : Les messages $x_0^{(i)}$ de l'équation 5.8, un entier positif N, des indications temporelles $m_{\text{interset}} \in [\![1, m_x]\!]$ et $I \subset [\![1, m_k]\!]$.
Sorties : Un candidat $\widehat{k}$ pour les 8 bits de poids faibles de $k^\star$.

$c \leftarrow \mathcal{M}_{0,255}(0)$;
pour $i = 0$ **à** 255 **faire**
  pour $j = 0$ **à** N **faire**
    $C_x^{(i,j)} \leftarrow$ trace de l'opération $x_0^{(i)}\lambda_0^{(j)}$;
    $C_k^{(i,j)} \leftarrow$ trace de l'opération $k_0\lambda_0^{(j)}$;
  fin
  $c_i \leftarrow \rho\left(\begin{pmatrix} C_{x,m_{\text{interest}}}^{(i,1)} \\ \vdots \\ C_{x,m_{\text{interest}}}^{(i,N)} \end{pmatrix}, \begin{pmatrix} C_{k,m}^{(i,1)} \\ \vdots \\ C_{k,m}^{(i,N)} \end{pmatrix}\right), \forall m \in I$; `// Trace de corrélation`
  $c(i) \leftarrow \max|c_i|$;
fin
$\widehat{k} \leftarrow \arg\max c$;
retourner $\widehat{k}$;

Algorithme 5.6 : Schéma de l'attaque en collision par corrélations verticales

Table 5.2 – Différence entre la cible et la configuration entre Varchola *et al.* [VDRZ15] et notre cas.

Configuration	[VDRZ15]	**Nôtre cas**
Cible	FPGA	Microcontrôleur
Taille des mots de l'architecture	8 bits	32 bits
Fréquence de l'horloge	16.384 MHz	24 MHz
Fréquence d'échantillonnage	20 GS/s	10 GS/s
Canal auxiliaire	Consommation	EM



(a) Octet 1 (b) Octet 2

Figure 5.9 – Évolution du rang de la bonne clef pour l'attaque avec corrélation verticale octet 1 et octet 2

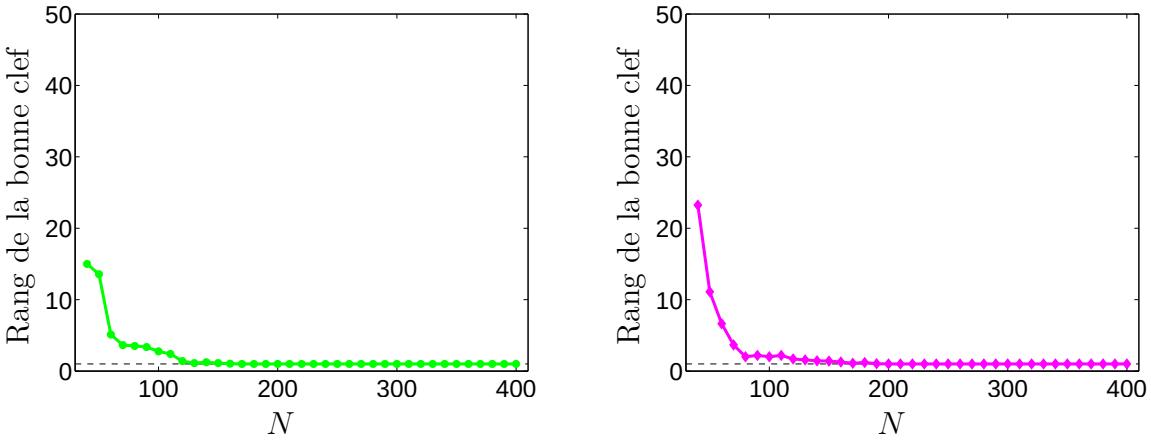
À partir de $N = 350$, l'attaque permet de distinguer la bonne sous-clef (l'octet de poids faible de k_0). La comparaison avec la méthode horizontale de détection par corrélations horizontales en figure 5.5 est claire, la détection de collisions verticale permet le succès de l'attaque.

L'octet de poids faible étant retrouvé, la suite de l'attaque est de cibler l'octet suivant, à savoir, les bits $k_{0,15}k_{0,14}\dots k_{0,8}$ (avec la notation $k_0 = (k_{0,31}k_{0,30}\dots k_{0,16}\boldsymbol{k_{0,15}\dots k_{0,8}}k_{0,7}\dots k_{0,0})_2$). L'octet de poids faible des messages choisis est fixé à la valeur $\widehat{k} = (\widehat{k_{0,7}\dots k_{0,0}})_2$, qui a été trouvée avec la première attaque. Puis, les bits 8 à 15 sont fixés de telle manière qu'ils parcourent les 256 possibilités :

$$\begin{aligned} x_0^{(0)} &= (00\dots0\mathbf{00000000}\widehat{k_{0,7}\dots k_{0,0}})_2 \\ x_0^{(1)} &= (00\dots0\mathbf{00000001}\widehat{k_{0,7}\dots k_{0,0}})_2 \\ &\vdots \\ x_0^{(255)} &= (00\dots0\mathbf{11111111}\widehat{k_{0,7}\dots k_{0,0}})_2 \end{aligned} \tag{5.11}$$

Ensuite, l'attaque est la même que pour l'octet de poids faible. Cette fois, si collision il y a, elle sera sur 16 bits. Donc, les données manipulées seront encore plus identiques et les phénomènes physiques du circuit intégré seront eux aussi encore plus identiques, cela se traduira sur les mesures du canal auxiliaire. La détection de collision est donc plus facile, la figure 5.9b montre l'évolution du rang de la bonne clef en fonction du nombre de répétitions N. Il suffit de $N = 250$ environ pour retrouver les 8 bits ciblés.

De même pour les deux octets suivants. Cette procédure donne des résultats encore meilleurs pour l'octet 3. Plus précisément, la figure 5.10a montre qu'avec $N = 200$ traces, l'attaque permet de retrouver le bon octet de clef. L'échelle de la figure 5.10 est adaptée par rapport à celle de la figure 5.9 car l'attaque est plus performante, c'est-à-dire que le rang de la bonne clef est très bas avec un nombre de répétitions N petit. Enfin, lorsque la cible est le dernier octet, la figure 5.10b montre que $N = 200$ est satisfaisant pour identifier les 8 bits du secret.



(a) Octet 3 (b) Octet 4

Figure 5.10 – Évolution du rang de la bonne clef pour l'attaque avec corrélation verticale octet 3 et octet 4

5.3.3 Coût de l'attaque

L'attaque que nous venons de présenter a été mise en pratique en environnement réel en ciblant l'implémentation décrite dans l'algorithme 5.5. Une borne maximale du nombre nécessaire d'acquisitions pour faire l'attaque sur le mot k_0 est :

$$E_{k_0} \simeq 2^8 \left(\underbrace{400}_{\text{Octet 1}} + \underbrace{250}_{\text{Octet 2}} + \underbrace{200}_{\text{Octet 3}} + \underbrace{200}_{\text{Octet 4}} \right) \simeq 2,7 \times 10^5. \tag{5.12}$$

Pour le cas expérimental que nous avons mis en pratique, il faut $0,4$ seconde pour acquérir une trace de 30000 échantillons avec une fréquence d'échantillonnage de 10 GS/s. L'acquisition des $2,7 \times 10^5$ traces prend donc environ 30 heures.

Puisqu'il y a les 7 autres mots de k qu'il reste à retrouver, il faut réaliser sept fois supplémentaires cette attaque. Le nombre total de traces est :

$$E_k \simeq 8 \times 2^8(400 + 250 + 200 + 200) \simeq 2,2 \times 10^6. \tag{5.13}$$

5.4 Conclusion sur l'efficacité de la contre-mesure

L'implémentation directe de la contre-mesure laisse apparaître de nouvelles failles que nous avons montrées exploitables. Nos travaux théoriques et pratiques sont menés afin de révéler un schéma de détection de collisions qui fonctionne dans le cas de la répétition de l'opération avec le secret et le message. Le comportement de la détection de collisions est testé dans un environnement idéal. Les deux multiplications cibles sont implémentées de façon à ce qu'elles soient sous le signal de déclenchement. Pour cibler le couplage tout entier, une étape supplémentaire est nécessaire : retrouver ces opérations dans une trace complète du couplage.

5.4.1 Randomisation des coordonnées avec P ou Q secret : quel impact ?

Nous avons montré aux sections 5.1.1 et 5.1.2 l'implémentation de la randomisation, respectivement, avec Q secret et P secret. De plus, dans chacun des cas, nous avons identifié les opérations critiques utilisées par l'attaque en collision, c'est-à-dire les multiplications avec le masque et le message, puis le produit entre le masque est le secret.

Les schémas d'attaques sont identiques, quelle que soit la configuration. Le seul aspect qui entre en jeu, vient du choix du couplage, et plus précisément du choix des groupes $\mathbb{G}_1$ et $\mathbb{G}_2$. Plus rigoureusement, avec un couplage Optimal Ate $e_{m,OA} : \mathbb{G}_2 \times \mathbb{G}_1$ par exemple, le point $P \in \mathbb{G}_1 = E(\mathbb{F}_q)[m]$ a trois coordonnées (en représentation jacobienne) qui sont simplement des entiers sur $|q|_2$ bits. Alors que le point $Q \in \mathbb{G}_2 \subset E(\mathbb{F}_{q^{12}})[m]$ a trois coordonnées qui sont des éléments de $\mathbb{F}_{q^{12}}$ (ou $\mathbb{F}_{q^2}$ dans le cas de l'utilisation d'une tordue). Donc, si $Q \in \mathbb{G}_2$ est secret, et que l'on cible d'abord $x_Q \in \mathbb{F}_{q^{12}}$ alors il faut faire 12 fois l'attaque pour retrouver les 12 entiers de $\mathbb{F}_q$ qui composent x_Q.

Par ailleurs, le point Q est un élément composé de $12 \times |q|_2$ bits alors que P est constitué de $2 \times |q|_2$ bits, il y a 6 fois plus de bits à retrouver si Q est le secret plutôt que P.

En outre, nous avons vu à la section 5.1 que le système de coordonnées affines-jacobiennes pour le cas Q secret est plus intéressant en termes d'efficacité. Le choix Q secret semble préférable, cependant, pour pallier à l'attaque en collision que nous avons introduite dans cette thèse, il faut trouver d'autres stratagèmes.

La structure de l'algorithme de Miller fait que les points P et Q ont un rôle asymétrique. Plutôt que de randomiser seulement le point public, comme proposé dans la littérature pour des raisons de surcoût, il devient intéressant d'étudier le cas de figure où la randomisation porte sur les deux points. Pour contrer l'effet de l'attaque en collision, il faut que les deux masques soient différents. Effectivement, si le masque λ est le même on ajoute de l'huile sur le feu (cf. algorithme 5.7).

Entrées : Une courbe elliptique $E : y^2 = x^3 + ax + b$ sur $\mathbb{F}_q$, $P \in E\left(\mathbb{F}_{q^k}\right)[m]$ et $Q \in E\left(\mathbb{F}_{q^k}\right)$.
Sorties : P et Q avec randomisation des coordonnées jacobiennes.

1 Tirer $\lambda \in \mathbb{F}_q^\star$ aléatoire ;
2 $P = (X_P : Y_P : Z_P)$ avec $X_P \leftarrow x_P\lambda^2$; $Y_P \leftarrow y_P\lambda^3$; $Z_P \leftarrow \lambda$;
3 $Q = (X_Q : Y_Q : Z_Q)$ avec $X_Q \leftarrow x_Q\lambda^2$; $Y_Q \leftarrow y_Q\lambda^3$; $Z_Q \leftarrow \lambda$;
4 **retourner** P *et* Q ;

Algorithme 5.7 : Randomisation de P et Q avec le même masque (exemple avec P secret)

Entrées : Une courbe elliptique $E : y^2 = x^3 + ax + b$ sur $\mathbb{F}_q$, $P \in E\left(\mathbb{F}_{q^k}\right)[m]$ et $Q \in E\left(\mathbb{F}_{q^k}\right)$.
Sorties : P et Q avec randomisation des coordonnées jacobiennes.

1 Tirer $\lambda_1 \in \mathbb{F}_q^\star$ aléatoire ;
2 $P = (X_P : Y_P : Z_P)$ avec $X_P \leftarrow x_P\lambda_1^2$; $Y_P \leftarrow y_P\lambda_1^3$; $Z_P \leftarrow \lambda_1$;
3 Tirer $\lambda_2 \in \mathbb{F}_q^\star$ aléatoire ;
4 $Q = (X_Q : Y_Q : Z_Q)$ avec $X_Q \leftarrow x_Q\lambda_2^2$; $Y_Q \leftarrow y_Q\lambda_2^3$; $Z_Q \leftarrow \lambda_2$;
5 **retourner** P *et* Q ;

Algorithme 5.8 : Randomisation de P et Q avec deux masques (exemple avec P secret)

Maintenant que les deux points sont pourvus d'aléa (cf. algorithme 5.8), le système de coordonnées n'est plus mixte. Dans ce cas, comme le montre la table 2.7, la représentation projective permet de réaliser les opérations en un minimum de coût.

La menace est maintenant transférée à l'opération de randomisation en elle-même, elle doit être implémentée de façon sécurisée. Si cette étape est accomplie de manière entièrement sécurisée, l'algorithme de couplage va manipuler le secret masqué. La distribution du masque est supposée uniforme, c'est-à-dire que $\lambda \in \mathbb{F}_q^\star$ est tiré aléatoirement, ce qui s'écrit $\mathbb{P}(\Lambda = \lambda) = \frac{1}{q-1}$, où Λ dénote les valeurs que prennent la variable aléatoire discrète. Nous avons déjà vu à la fin de la section 5.1.2 que $x_Q\lambda$ prend une valeur aléatoire dans un ensemble à $q-1$ éléments. En d'autres mots, le secret n'est plus manipulé. La seule cible restante est effectivement l'étape de randomisation. La sécurisation de cette opération reste une perspective à ce jour.

Pour conclure, nos études nous ont permis de faire des choix sur la contre-mesure, le système de représentation des coordonnées et la position du secret (P ou Q). La réunion de ces choix et de l'algorithme Optimal Ate sur courbe BN est proposée dans l'algorithme 5.9.

Nous faisons le même constat que pour le chapitre précédent, l'impact des attaques est rapporté dans la table 5.3.

Table 5.3 – Résumé des attaques par canaux cachés

	Secret	P	
	Contre-mesure	Aucune	Randomisation des coordonnées
Attaques	CPA verticale	Une seule attaque pratique Amélioration d'un fact. 10	
	CPA horizontale		
	Profilage		
	Collisions		
	Secret	Q	
	Contre-mesure	Aucune	Randomisation des coordonnées
Attaques	CPA verticale	Pas de validation pratique	
	CPA horizontale	Amélioration d'un fact. 10 1 seule trace	
	Profilage		
	Collisions		

Code couleur : Attaque relevée dans la littérature | Attaques en pratiques | Pas étudié | Pas d'attaque identifiée

Entrées : Une courbe elliptique E sur $\mathbb{F}_q$, sa trace de Frobenius t, le point $P \in E(\mathbb{F}_q)[m]$ (l'entrée connue) et $Q \in E[m] \cap \ker(\Psi_q - [q])$ (l'entrée secrète).
Sorties : $e_{m,OA}(Q,P) \in \mu_m \subset \mathbb{F}^{\star}_{q^{12}}$.

1 Écrire $s = 6t+2$ comme $s = \sum_{i=0}^{\widehat{\mathfrak{n}}-1} s_i 2^i$ avec $s_i \in \{-1,0,1\}$;
2 Tirer $\lambda_1 \in \mathbb{F}_q^{\star}$ aléatoire ;
3 $P = (X_T : Y_T : Z_T)$ avec $X_P \leftarrow x_P\lambda_1$; $Y_P \leftarrow y_P\lambda_1$; $Z_P \leftarrow \lambda_1$;
4 Tirer $\lambda_2 \in \mathbb{F}_q^{\star}$ aléatoire ;
5 $Q = (X_Q : Y_Q : Z_Q)$ avec $X_Q \leftarrow x_Q\lambda_2$; $Y_Q \leftarrow y_Q\lambda_2$; $Z_Q \leftarrow \lambda_2$;
6 // P et Q sont en coordonnées projectives
7 $f \leftarrow 1$;
8 $T \leftarrow Q$;
9 **pour** $i = \widehat{\mathfrak{n}} - 2$ **à 0 faire**
10 $D \leftarrow 3X_T^2 + aZ_T^2$; $F \leftarrow Y_T Z_T$;
11 $G \leftarrow FY_TX_T$; $H \leftarrow D^2 - 8G$;
12 $X_R \leftarrow 2FH$;
13 $Y_R \leftarrow D(4G - H) - 8(Y_TF)^2$;
14 $Z_R \leftarrow 8F^3$;
15 $l_{T,T}(P) \leftarrow 2F(Z_TY_P - Y_TZ_P) - D(X_PZ_T - X_TZ_P)$;
16 $f \leftarrow f^2 l_{T,T}(P)$;
17 $T \leftarrow R$;
18 **si** $s_i = -1$ **alors**
19 $Q_{neg} \leftarrow (X_Q : -Y_Q : Z_Q)$; // $-Q$
20 $A \leftarrow Y_TZ_{Q_{neg}} - Y_{Q_{neg}}Z_T$; $B \leftarrow X_TZ_{Q_{neg}} - X_{Q_{neg}}Z_T$;
21 $C \leftarrow A^2Z_TZ_{Q_{neg}} - B^3 - 2B^2X_PZ_T$;
22 $X_R \leftarrow BC$;
23 $Y_R \leftarrow A(B^2X_TZ_{Q_{neg}} - C) - Y_TZ_{Q_{neg}}B^3$;
24 $Z_R \leftarrow B^3Z_TZ_{Q_{neg}}$;
25 $l_{T,P}(Q_{neg}) \leftarrow B(Z_TY_P - Z_PY_T) - A(Z_TX_P - Z_PX_T)$;
26 $f \leftarrow f l_{T,Q_{neg}}(P)$;
27 $T \leftarrow R$;
28 **sinon si** $s_i = 1$ **alors**
29 $A \leftarrow Y_TZ_Q - Y_QZ_T$; $B \leftarrow X_TZ_Q - X_QZ_T$;
30 $C \leftarrow A^2Z_TZ_Q - B^3 - 2B^2X_PZ_T$;
31 $X_R \leftarrow BC$;
32 $Y_R \leftarrow A(B^2X_TZ_Q - C) - Y_TZ_QB^3$;
33 $Z_R \leftarrow B^3Z_TZ_Q$;
34 $l_{T,P}(Q) \leftarrow B(Z_TY_P - Z_PY_T) - A(Z_TX_P - Z_PX_T)$;
35 $f \leftarrow f l_{T,Q}(P)$;
36 $T \leftarrow R$;
37 **fin**
38 **fin**
39 $Q_1 \leftarrow \Psi_q(Q)$; $Q_2 \leftarrow \Psi_q^2(Q)$;
40 $f \leftarrow f l_{T,Q_1}(P)$; $T \leftarrow T + Q_1$;
41 $f \leftarrow f l_{T,-Q_2}(P)$; $T \leftarrow T - Q_2$;
42 $f \leftarrow f^{p^6-1}$;
43 $f \leftarrow f^{p^2+1}$;
44 $f \leftarrow f^{(p^4-p^2+1)/m}$;
45 **retourner** f ;

Algorithme 5.9 : Couplage Optimal Ate sur courbes BN

Chapitre 6

Conclusion

La problématique concerne la sécurité des implémentations des algorithmes de couplages face aux attaques par canaux auxiliaires. En effet, les couplages pour la cryptographie sont étudiés depuis une quinzaine d'années maintenant, les protocoles qu'ils font apparaître permettent de répondre à de nouveaux besoins. De plus, les implémentations des algorithmes sont de plus en plus efficaces et offrent la possibilité d'intégrer des solutions cryptographiques à base de couplages dans les dispositifs embarqués [BGDM+10, DEMHR15, MHZ15, GSZM16].

La question de la sécurité physique de ces implémentations est identifiée en 2004 par Page *et al.* [PV04]. Depuis, d'autres travaux théoriques ont montré, soit de nouvelles vulnérabilités, soit de nouvelles contre-mesures. Dans cette thèse, nous nous sommes intéressés aux attaques par canaux auxiliaires contre les couplages. Nous avons donné l'état de l'art des attaques contre les algorithmes munis d'aucune protection. Concernant les implémentations des couplages munis de contre-mesures, nous montrons qu'il reste des failles, et nous avons exploité ces failles pour exposer la cryptanalyse physique.

Nous avons analysé l'état de l'art des attaques par canaux auxiliaires contre les couplages. Nous commençons par décrire les principaux résultats, essentiellement théoriques, depuis une douzaine d'années. Ensuite, nous détaillons techniquement les attaques pertinentes. Ce qui nous permet d'étudier la cible commune aux attaques de l'état de l'art. Et ce, en traitant les deux cas : P ou Q est secret. L'analyse de l'état de l'art nous mène à l'opération critique à cibler : la multiplication entre deux mots machine. L'étude de l'état de l'art nous a permis de contribuer à un chapitre de livre [EMGG+17].

L'une des premières validations pratiques des attaques contre l'implémentation des couplages est faite dans cette thèse. Nous utilisons un banc qui permet la capture des émissions électromagnétiques du dispositif ciblé. Le dispositif sous test comporte un circuit intégré qui est susceptible d'intéresser les concepteurs de solutions embarquées. Bien qu'il ne soit pas matériellement sécurisé, il est représentatif des solutions qui peuvent être intégrées dans les objets connectés, par exemple, qui ne comportent pas forcément de la cryptographie. Le couplage visé est un Optimal Ate sur courbe de Barreto-Naehrig qui garantissaient un niveau de sécurité 128 bits avant la parution des dernières attaques contre le logarithme discret [MSS16, BD17]. Afin de garantir à nouveau un niveau de sécurité raisonnable, la taille des entrées des couplages doit être augmentée. Par exemple, l'utilisation des courbes BN permettrait d'obtenir la sécurité susmentionnée en travaillant sur le corps fini $\mathbb{F}_q$ avec q sur 256 bits. Les nouvelles estimations (voir [BD17] par exemple) montrent que q doit être sur 456 bits. Cependant,

la cible présentée dans le chapitre 3 pour les attaques par canaux auxiliaires reste valide. En effet, il est toujours question de manipuler les éléments de $\mathbb{F}_q$. En particulier, la multiplication modulaire de deux d'entre eux, qui peut être réalisée par la multiplication modulaire de Montgomery, nécessitera toujours des multiplications entre deux mots machine. Les fuites exploitées seront encore exploitables même avec cette augmentation de la taille des données.

Nous développons une attaque optimisée contre la multiplication de mots machine. Nous élaborons et validons théoriquement un schéma d'attaque performant. La cible est la multiplication de deux mots machine de 32 bits, nous appliquons la stratégie de « diviser pour mieux régner » pour scinder l'attaque des 32 bits en quatre « sous-attaques » sur 8 bits. La méthode permettant d'enchaîner les sous-attaques est justement un point que nous améliorons. Nous comparons une méthode basique, et une seconde, plus évoluée, qui intègre des attaques internes afin d'atténuer l'effet de l'attaque basique qui est trop globale. Plus formellement, les attaques internes permettent de faire une présélection des hypothèses de clefs qui servent à l'attaque de base. Nous étudions aussi l'effet d'un paramètre important qui apparaît dans la stratégie de diviser pour mieux régner. En effet, pour passer d'une sous-attaque à une autre, il n'est pas garanti que la première sous-attaque permette de distinguer l'hypothèse de clef correspondant à la clef secrète. Elle peut très bien ne pas être distinguée en première position, mais être classée parmi les hypothèses prédominantes. Il est alors nécessaire de considérer plusieurs de ces hypothèses de clefs prééminentes pour la sous-attaque suivante. Nous étudions l'effet de ce nombre, α, de clefs retenues. Avec un $1 \leq \alpha \leq 2^z$ qui croît, l'attaque se voit avoir un succès qui croît lui aussi.

Concernant la validation pratique des résultats, la première étape est la caractérisation de la fuite. C'est-à-dire, vérifier que la multiplication de mots machine sur ce dispositif fuit pour l'attaque CPA. Nous validons le modèle de fuite en poids de Hamming ainsi que la méthode avancée d'enchaînement des sous-attaques. Par surcroît, nous validons l'effet du paramètre α, effectivement, plus il est grand et plus le succès de l'attaque est bon. Ces résultats ont fait l'objet de la publication [JFEMG16] à CRISIS 2016.

Nous faisons aussi la validation pratique d'une attaque par profilages sur la multiplication. Le but est de valider si les fuites conviennent à ce type d'attaque. Dans ce contexte, les phases de profilages et d'attaques sont faites sur le même dispositif. L'attaque est validée, elle nécessite un nombre de traces équivalent à l'attaque CPA, qui ne comporte pas de phase de profilage.

Ensuite nous étudions l'effet du choix de P ou Q secret face aux attaques par canaux auxiliaires. Ce choix n'est pas considérable à cause de l'opération ciblée, puisqu'une multiplication machine fuit quelle que soit l'ordre des ces opérandes. La structure des couplages, et plus précisément celle de l'algorithme de Miller, nous permet d'identifier une faille. Cette faille est exploitable par une approche d'attaque horizontale. Nous montons ainsi une attaque qui tire avantage de la manipulation des données dans la boucle principale de Miller. Les coordonnées du point secret interagissent avec des données connues à chaque itération. L'algorithme non protégé est la proie d'une attaque horizontale que nous réalisons avec une seule exécution/trace du couplage.

Parmi les contre-mesures de la littérature, la contre-mesure de randomisation des coordonnées projectives/jacobiennes est une solution qui a un faible surcoût de calcul. C'est une protection initialement présentée par Coron [Cor99] pour protéger les échelles

de multiplication scalaire spécifiquement dans le cadre de ECC. Ce principe de randomisation a ensuite été proposé pour protéger les couplages par Kim *et al.* [KTH+06] sur des corps binaires. Cette contre-mesure est extensible aux couplages basés sur les corps de grandes caractéristiques. Le coût de cette protection en fait l'une des plus intéressantes. Nous montrons dans un premier temps que la contre-mesure n'est pas applicable directement de la même façon lorsque P ou Q est secret. La structure de la boucle de Miller induit une meilleure intégration de la contre-mesure lorsque Q est secret. Il en découle une meilleure efficacité dans ce cas là.

Dans un second temps, nous proposons une application de l'attaque par profilage sans connaissance des messages de Oswald *et al.* [OM06]. Les attaques par profilage classiques nous ont permis d'attaquer avec succès la multiplication $x \times k$, cette fois nous ciblons une multiplication $(\lambda \times x) \times k$ avec un masque λ. L'attaque que propose Oswald *et al.* cible une implémentation masquée de l'AES. Le masque est appliqué par un XOR sur les blocs de données, il est donc sur 8 bits. La phase de profilage est classique, la connaissance de toutes les valeurs intermédiaires est utilisée pour la création des classes, qui permettent de classer les traces. La phase d'attaque, qui doit créer les classes pour chaque hypothèse de clef, prend en compte toutes les valeurs ($2^8 = 256$) potentiellement prises par le masque. Nous adaptons cette attaque sur la multiplication de mots machine. Beaucoup de paramètres sont à prendre en considération pour recréer une situation similaire à celle d'Oswald *et al.*. Nous en essayons certains, sans succès. Parallèlement, nous travaillons sur un schéma d'attaque en collisions, qui, lui, s'avère être un succès.

La contre-mesure de randomisation des coordonnées n'est pas gratuite. L'étape de randomisation permet de rendre aléatoires les données connues en entrée de couplage et donc de perdre toute connaissance des variables intermédiaires. Cependant, cette nouvelle étape introduit une multiplication qui est répétée quelques instructions plus loin avec les données secrètes. Le prédicat de base de l'attaque est que si les données connues sont égales (ou partiellement égales) aux données secrètes alors les données manipulées seront les mêmes, et donc les canaux auxiliaires acquis seront similaires. La notion de similarité signifie que les traces comportent des collisions, mais les détecter est une étape exigeante. Nous appliquons un premier schéma d'attaque avec une détection de collisions par simple corrélation croisée entre les signaux aux instants où les opérations critiques sont effectuées. L'attaque est validée en pratique contre la répétition de la multiplication de mots machine. Le coût de l'attaque contre un couplage pourvu de la randomisation des coordonnées est alors estimé suffisamment bas pour affaiblir la contre-mesure. Ces travaux ont été publiés à la conférence SECRYPT 2017 [JFG17].

Dans un troisième temps, nous concluons avec les combinaisons possibles de plusieurs contre-mesures. En effet, la protection que nous venons de déjouer ouvre sur une nouvelle situation. La randomisation des coordonnées ne permet plus à l'attaquant de faire les attaques précédentes. Cependant, l'implémentation de l'étape de randomisation provoque une répétition d'opération avec des données connues et secrètes. Cette étape de randomisation doit donc être réalisée de manière sécurisée. La littérature propose déjà des solutions de randomisation de l'arithmétique à bas coût lorsqu'il s'agit de protéger seulement quelques opérations. Nous proposons une solution avec le choix Q secret et la randomisation des coordonnées projectives comme un compromis face aux attaques que nous avons pu valider dans cette thèse.

Perspectives

Nous avons donné des arguments sur de nombreux paramétrages intrinsèques aux couplages : le choix P ou Q secret, le système de coordonnées. Puis concernant le paramétrage des attaques : le modèle en poids de Hamming, découpage et enchaînement, pour l'attaque de la multiplication de mots machine. Nous proposons un exemple d'implémentation des couplages qui résiste aux attaques développées au cours de cette thèse. Quelques points restent à valider pour approuver ce niveau de résistance. Dans la solution proposée, nous conjecturons que le seul point faible est l'étape de randomisation, il est intéressant de prouver formellement cette conjecture. Nous avons exposé les idées intuitives, mais une preuve de sécurité appuierait nos propos. Pour réaliser de façon sécurisée cette étape de randomisation, les pistes que nous souhaitons envisager sont :

- La randomisation de l'arithmétique. Par exemple en représentation Residue number system – ou *Système modulaire de représentation* – (RNS) avec une nouvelle base qui est calculée à chaque exécution du couplage.

- L'étape de randomisation que l'on souhaite protéger est constituée des trois multiplications modulaires $x_Q\lambda, y_Q\lambda$ et $z_Q\lambda$. La protection de la multiplication modulaire a été proposée dans [MH10, CFG+10, BJPW13a]. Ces protections peuvent donc être utilisées pour sécuriser les trois multiplications critiques, $x_Q\lambda, y_Q\lambda$ et $z_Q\lambda$, qui manipulent les coordonnées secrètes.

S'il n'est pas possible d'exhiber une preuve de sécurité pour répondre à la conjecture proposée, alors il faudra envisager une contre-mesure supplémentaire/complémentaire. La contre-mesure de randomisation de la variable de Miller proposée par Scott [Sco05] est une solution intéressante. Chaque multiplication modulaire dans la boucle de Miller est masquée par une multiplication avec un masque. Par exemple, la $i-$ème multiplication $a^{(i)} \times b^{(i)} \mod p$ est remplacée par $\left(\lambda^{(i)} \times a^{(i)}\right) \times b^{(i)} \mod p$ pour un $\lambda^{(i)} \in \mathbb{F}_q^\star$ aléatoire. Nous voyons l'évaluation de cette contre-mesure selon trois axes :

- La résistance face aux attaques. C'est la suite des travaux que nous avons commencés avec les attaques par profilage sur les masquages (cf. section 5.2).

- Le coût de la contre-mesure. La contre-mesure double le nombre de multiplications modulaires, le coût est donc très implémentation-dépendant.

- L'automatisation de la vérification de la (des) contre-mesure(s). La protection de Scott peut s'avérer coûteuse, il n'est surement pas nécessaire d'appliquer le masquage à toutes les multiplications. On peut envisager un outil, qui permettrait de tester si la randomisation de la variable de Miller est convenable, pour un choix arbitraire des multiplications à masquer.

Des travaux sur l'attaque en collisions sont aussi pertinents à faire. L'attaque que nous avons proposée a été validée en pratique sur des traces dans un environnement de tests favorable. L'attaque sur le couplage complet nécessite une étape supplémentaire : l'analyse de la trace pour trouver les points d'intérêts. C'est un classique des attaques par canaux auxiliaires, mais cette analyse est souvent coûteuse en temps. De plus, pour évaluer le coût total de l'attaque, il faut prendre en compte le temps d'acquisition d'un couplage entier, au lieu des deux seules multiplications que nous faisons. Pour améliorer la méthode de détection de collisions, on peut envisager des méthodes d'apprentissage.

On peut entraîner l'algorithme à détecter des collisions, en prenant même en compte un environnement bruité, sans resynchronisation (voir [PHG17] par exemple). La détection de collisions n'en sera que meilleure. De plus, la phase d'attaque est aidée par la connaissance des messages.

Nos études théoriques sont fondées sur l'hypothèse que les « meilleures clefs » ciblées à la sortie de la multiplication sont identifiées par le coefficient de confusion. C'est une hypothèse qui est vérifiée en sortie de *SubBytes* [GHR15], mais la multiplication est une opération linéaire, ce qui s'oppose au *SubBytes*. Les coefficients de confusion donnent à la bonne clef, la valeur 0. Les clefs « similaires » qui sont identifiées par le coefficient de confusion sont les shifts (décalages à gauche et droite) de celle-ci. C'est un phénomène que nous avons aussi relevé dans toutes les attaques menées en pratique. Des schémas d'attaques exploitant ce résultat sont certainement envisageables. Enfin, nos travaux sur l'étude du paramètre α nous ont permis de conclure quant à l'effet de ce paramètre sur le taux de succès des attaques. Cependant, ce critère du nombre d'hypothèses de clef est fixe, il existe des méthodes adaptatives, souvent appelées KEA (key enumeration algorithm) dans la littérature [VCGRS12]. Il serait intéressant de positionner notre méthode par rapport à ces travaux. Cette perspective rejoint la précédente, on peut imaginer une méthode adaptative et qui prenne en compte les shifts des hypothèses de clefs élevées.

Références

[AARR03] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. *The EM Side—Channel(s)*, pages 29–45. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ↑60

[AF08] Frédéric Amiel and Benoit Feix. On the BRIP Algorithms Security for RSA. In *IFIP International Workshop on Information Security Theory and Practices*, pages 136–149. Springer, 2008. ↑75

[AG01] Mehdi-Laurent Akkar and Christophe Giraud. An Implementation of DES and AES, Secure against Some Attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 309–318. Springer, 2001. ↑9

[AG03] Mehdi-Laurent Akkar and Louis Goubin. A Generic Protection against High-Order Differential Power Analysis. In *International Workshop on Fast Software Encryption*, pages 192–205. Springer, 2003. ↑120

[AK98] Ross Anderson and Markus Kuhn. Low Cost Attacks on Tamper Resistant Devices. In *Security protocols*, pages 125–136. Springer, 1998. ↑61

[ALNR11] Christophe Arene, Tanja Lange, Michael Naehrig, and Christophe Ritzenthaler. Faster Computation of the Tate Pairing. *Journal of number theory*, 131(5) :842–857, 2011. ↑46

[AM93] A Oliver L Atkin and François Morain. Elliptic curves and primality proving. *Mathematics of computation*, 61(203) :29–68, 1993. ↑39

[Arm] Developer Arm. Cortex-m3. https://developer.arm.com/products/processors/cortex-m/cortex-m3. ↑96

[ASYZ13] Chen Aidong, Xu Sen, Chen Yun, and Qin Zhiguang. Collision-Based Chosen-Message Simple Power Clustering Attack Algorithm. *China Communications*, 10(5) :114–119, 2013. ↑75

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation Power Analysis with a Leakage Model. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 16–29. Springer, 2004. ↑9, ↑71

[BD17] Razvan Barbulescu and Sylvain Duquesne. Updating key size estimations for pairings. Cryptology ePrint Archive, Report 2017/334, 2017. ↑8, ↑58, ↑100, ↑101, ↑137

[BDGN14] Shivam Bhasin, Jean-Luc Danger, Sylvain Guilley, and Zakaria Najm. Side-Channel Leakage and Trace Compression using Normalized Inter-Class Variance. In *Proceedings of the Third Workshop on Hardware and Architectural Support for Security and Privacy*, page 7. ACM, 2014. ↑97

[Ber73] Gustave Bertrand. *Enigma : ou, La plus grande énigme de la guerre 1939-1945*. Plon, 1973. ↑2

[BF01] Dan Boneh and Matt Franklin. Identity-Based Encryption from the Weil Pairing. In *Annual International Cryptology Conference*, pages 213–229. Springer, 2001. ↑6, ↑52, ↑54

[BGDM+10] Jean-Luc Beuchat, Jorge E González-Díaz, Shigeo Mitsunari, Eiji Okamoto, Francisco Rodríguez-Henríquez, and Tadanori Teruya. High-Speed Software Implementation of the Optimal Ate Pairing over Barreto–Naehrig Curves. In *International Conference on Pairing-Based Cryptography*, pages 21–39. Springer, 2010. ↑23, ↑24, ↑26, ↑34, ↑46, ↑51, ↑52, ↑137

[BGJT14] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, and Emmanuel Thomé. A quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 1–16. Springer, 2014. ↑58

[BGK04] Johannes Blömer, Jorge Guajardo, and Volker Krummel. Provably Secure Masking of AES. In *International Workshop on Selected Areas in Cryptography*, pages 69–83. Springer, 2004. ↑120

[BGK14] Razvan Barbulescu, Pierrick Gaudry, and Thorsten Kleinjung. The Tower Number Field Sieve. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 31–55. Springer, 2014. ↑58

[BGL13] Johannes Blömer, Peter Günther, and Gennadij Liske. Improved Side Channel Attacks on Pairing Based Cryptography. *COSADE*, pages 154–168, 2013. ↑77, ↑78, ↑79, ↑80, ↑86, ↑110

[BGW99] Marc Briceno, Ian Goldberg, and David Wagner. A pedagogical implementation of the gsm a5/1 and a5/2 "voice privacy" encryption algorithms. *Originally published at http ://www. scard. org, mirror at http ://-cryptome. org/gsm-a512. htm*, 1999. ↑4

[BJPW13a] Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal and Vertical Side-Channel Attacks against Secure RSA Implementations. In *Cryptographers' Track at the RSA Conference*, pages 1–17. Springer, 2013. ↑66, ↑140

[BJPW13b] Aurélie Bauer, Eliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal Collision Correlation Attack on Elliptic Curves. In *International Conference on Selected Areas in Cryptography*, pages 553–570. Springer, 2013. ↑75

[BK98] Ramachandran Balasubramanian and Neal Koblitz. The improbability that an elliptic curve has subexponential discrete log problem under the menezes—okamoto—vanstone algorithm. *Journal of cryptology*, 11(2) :141–145, 1998. ↑37

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 514–532. Springer, 2001. ↑56

[BLS02] Paulo SLM Barreto, Ben Lynn, and Michael Scott. Constructing Elliptic Curves with Prescribed Embedding Degrees. In *International Conference on Security in Communication Networks*, pages 257–267. Springer, 2002. ↑37

[BLS04] Paulo Barreto, Ben Lynn, and Michael Scott. On the Selection of Pairing-Friendly Groups. In *Selected areas in cryptography*, pages 17–25. Springer, 2004. ↑46, ↑47

[BM90] Steven M Bellovin and Michael Merritt. Limitations of the kerberos authentication system. *ACM SIGCOMM Computer Communication Review*, 20(5) :119–132, 1990. ↑5

[BN05] Paulo SLM Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *International Workshop on Selected Areas in Cryptography*, pages 319–331. Springer, 2005. ↑37, ↑39, ↑51, ↑77

[BSS99] Ian Blake, Gadiel Seroussi, and Nigel Smart. *Elliptic curves in cryptography*, volume 265. Cambridge university press, 1999. ↑29

[CCD00] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential Power Analysis in the Presence of Hardware Countermeasures. In *Cryptographic Hardware and Embedded Systems—CHES 2000*, pages 13–48. Springer, 2000. ↑9

[CFA+05] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography*. CRC press, 2005. ↑18, ↑19, ↑20, ↑21, ↑22, ↑37

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal Correlation Analysis on Exponentiation. In *International Conference on Information and Communications Security*, pages 46–61. Springer, 2010. ↑10, ↑66, ↑140

[CFG+12] Christophe Clavier, Benoit Feix, Georges Gagnerot, Christophe Giraud, Mylene Roussellet, and Vincent Verneuil. ROSETTA for Single Trace Analysis. In *International Conference on Cryptology in India*, pages 140–155. Springer, 2012. ↑66

[CK13] Omar Choudary and Markus G Kuhn. Efficient Template Attacks. In *International Conference on Smart Card Research and Advanced Applications*, pages 253–270. Springer, 2013. ↑124

[CK14] Omar Choudary and Markus G Kuhn. Template Attacks on Different Devices. In *International Workshop on Constructive Side-Channel Analysis and Secure Design*, pages 179–198. Springer, 2014. ↑73, ↑109

[CKN00] Jean-Sébastien Coron, Paul Kocher, and David Naccache. Statistics and Secret Leakage. In *Financial Cryptography*, pages 157–173. Springer, 2000. ↑103

[Cla04] Christophe Clavier. Side Channel Analysis for Reverse Engineering (SCARE)-An Improved Attack Against a Secret A3/A8 GSM Algorithm. 2004. ↑61

[CLN10] Craig Costello, Tanja Lange, and Michael Naehrig. Faster Pairing Computations on Curves with High-Degree Twists. In *International Workshop on Public Key Cryptography*, pages 224–242. Springer, 2010. ↑46

[Coc01] Clifford Cocks. An Identity Based Encryption Scheme Based on Quadratic Residues. In *IMA International Conference on Cryptography and Coding*, pages 360–363. Springer, 2001. ↑54

[Cor99] Jean-Sébastien Coron. Resistance against Differential Power Analysis for Elliptic Curve Cryptosystems. In *Cryptographic Hardware and Embedded Systems*, pages 725–725. Springer, 1999. ↑9, ↑64, ↑65, ↑78, ↑116, ↑138

[Cou15] Franck Courbon. *Rétro-conception matérielle partielle appliquée à l'injection ciblée de fautes laser et à la détection efficace de Chevaux de Troie Matériels*. PhD thesis, Saint-Etienne, EMSE, 2015. ↑61

[CRR02] Suresh Chari, Josyula R Rao, and Pankaj Rohatgi. Template Attacks. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 13–28. Springer, 2002. ↑9, ↑72, ↑73, ↑74

[DB04] Yvo Desmedt and Mike Burmester. Identity-Based Key Infrastructures (IKI). In *IFIP International Information Security Conference*, pages 167–176. Springer, 2004. ↑54

[DBS04] Ratna Dutta, Rana Barua, and Palash Sarkar. Pairing-Based Cryptographic Protocols : A Survey. *IACR Cryptology ePrint Archive*, 2004 :64, 2004. ↑56

[DD79] Régine Douady and Adrien Douady. Algebre et théories galoisiennes. 1979. ↑31

[Dem09] Michel Demazure. *Cours d'algèbre.* Cassini, 8 juin 2009. ↑23

[DEMHR15] Sylvain Duquesne, Nadia El Mrabet, Safia Haloui, and Franck Rondepierre. Choosing and generating parameters for low level pairing implementation on bn curves. *IACR Cryptology ePrint Archive*, 2015 :1212, 2015. ↑137

[Deu41] Max Deuring. Die typen der multiplikatorenringe elliptischer funktionenkörper. In *Abhandlungen aus dem mathematischen Seminar der Universität Hamburg*, volume 14, pages 197–272. Springer, 1941. ↑32

[DH76] Whitfield Diffie and Martin Hellman. New Directions in Cryptography. *IEEE transactions on Information Theory*, 22(6) :644–654, 1976. ↑5

[DL03] Iwan Duursma and HS Lee. Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$. *Advances in cryptology - AsiaCrypt 2003*, 4 :111–123, 2003. ↑76

[DLLM15] Ibrahima Diop, Pierre-Yvan Liardet, Yanis Linge, and Philippe Maurine. Collision Based Attacks in Practice. In *Digital System Design (DSD), 2015 Euromicro Conference on*, pages 367–374. IEEE, 2015. ↑75

[DLMV05] Rémy Daudigny, Hervé Ledig, Frédéric Muller, and Frédéric Valette. SCARE of the DES (Side Channel Analysis for Reverse Engineering of the Data Encryption Standard). In *International Conference on Applied Cryptography and Network Security*, pages 393–406. Springer, 2005. ↑61

[DLR15] Jean-Guillaume Dumas, Pascal Lafourcade, and Patrick Redon. Architectures PKI et communications sécurisées, 2015. ↑6

[DPRS11] Julien Doget, Emmanuel Prouff, Matthieu Rivain, and François-Xavier Standaert. Univariate Side Channel Attacks and Leakage Modeling. *Journal of Cryptographic Engineering*, 1(2) :123–144, 2011. ↑9

[dV86] Blaise de Vigenère. *Traicté des chiffres, ou Secrètes manières d'escrire.* Chez Abel L'Angelier, 1586. ↑2

[EDFL09] Nadia El Mrabet, Giorgio Di Natale, Flottes, and Marie Lise. A Practical Differential Power Analysis Attack Against the Miller Algorithm. *PRIME*, pages 308–311, 2009. ↑77, ↑80, ↑86, ↑110

[EG12] M Abdelaziz Elaabid and Sylvain Guilley. Portability of Templates. *Journal of Cryptographic Engineering*, 2(1) :63–74, 2012. ↑109

[ElG85] Taher ElGamal. A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE transactions on information theory*, 31(4) :469–472, 1985. ↑6

[EMFGL15] Nadia El Mrabet, Jacques Fournier, Louis Goubin, and Ronan Lashermes. A survey of fault attacks in pairing based cryptography. 2015. ↑10

[EMGG+17] Nadia El Mrabet, Louis Goubin, Sylvain Guilley, Jacques J.A. Fournier, Damien Jauvart, Martin Moreau, Pablo Rauzy, and Franck Rondepierre. Physical attacks. *Guide to Pairing-Based Cryptography*, 2017. ↑11, ↑137

[FG94] Serge Francinou and Hervé Gianella. Exercices de mathématiques pour l'agregation. 1994. ↑16

[FIP01] NIST FIPS. 197 : Advanced encryption standard (aes). *Federal information processing standards publication*, 197(441) :0311, 2001. ↑66

[FLD12] Yunsi Fei, Qiasi Luo, and A Adam Ding. A Statistical Model for DPA with Novel Algorithmic Confusion Analysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 233–250. Springer, 2012. ↑89

[FMN+14] Daisuke Fujimoto, Noriyuki Miura, Makoto Nagata, Yuichi Hayashi, Naofumi Homma, Hori Yohei, Toshihiro Katashita, Kazuo Sakiyama, Le Thanh-Ha, Julien Bringer, et al. Power noise measurements of cryptographic vlsi circuits regarding side-channel information leakage. *IEICE Transactions on Electronics*, 97(4) :272–279, 2014. ↑97

[FR94] Gerhard Frey and Hans-Georg Rück. A remark concerning m-divisibility and the discrete logarithm in the divisor class group of curves. *Mathematics of computation*, 62(206) :865–874, 1994. ↑6

[FST10] David Freeman, Michael Scott, and Edlyn Teske. A Taxonomy of Pairing-Friendly Elliptic Curves. *Journal of cryptology*, 23(2) :224–280, 2010. ↑37

[FT12] Pierre-Alain Fouque and Mehdi Tibouchi. Indifferentiable Hashing to Barreto–Naehrig Curves. *Progress in Cryptology–LATINCRYPT 2012*, pages 1–17, 2012. ↑40

[FV03] Pierre-Alain Fouque and Frédéric Valette. The Doubling Attack–Why Upwards Is Better than Downwards. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 269–280. Springer, 2003. ↑75

[Gal05] Steven Galbraith. Pairings. *London Mathematical Society Lecture Note Series*, (317), 2005. ↑42, ↑43

[GHR15] Sylvain Guilley, Annelie Heuser, and Olivier Rioul. A Key to Success – Success Exponents for Side-Channel Distinguishers. *Progress in Cryptology-INDOCRYPT*, pages 6–9, 2015. ↑141

[GHV08] Steven Galbraith, Florian Hess, and Frederik Vercauteren. Aspects of Pairing Inversion. *IEEE Transactions on Information Theory*, 54(12) :5719–5728, 2008. ↑53

[GLRP06] Benedikt Gierlichs, Kerstin Lemke-Rust, and Christof Paar. Templates vs. Stochastic Methods. In *CHES*, pages 15–29. Springer, 2006. ↑104

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. *Electromagnetic Analysis : Concrete Results*, pages 251–261. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ↑60

[GMV04] Steven D. Galbraith, J. McKee, and P. Valenca. Ordinary abelian varieties having small embedding degree. Cryptology ePrint Archive, Report 2004/365, 2004. ↑39

[GPS06] Robert Granger, Dan Page, and Nigel P Smart. High Security Pairing-Based Cryptography Revisited. In *International Algorithmic Number Theory Symposium*, pages 480–494. Springer, 2006. ↑49

[GR11] Santosh Ghosh and Dipanwita Roychowdhury. Security of prime field pairing cryptoprocessor against differential power attack. pages 16–29. Springer, 2011. ↑77

[GSZM16] Souhir Gabsi, Anissa Sghaier, Medien Zeghid, and Mohsen Machhout. Efficient software implementation of the final exponentiation for pairing. In *Image Processing, Applications and Systems (IPAS), 2016 International*, pages 1–4. IEEE, 2016. ↑137

[GT02] Jovan Dj Golic and Christophe Tymen. Multiplicative masking and power analysis of aes. In *CHES*, volume 2523, pages 198–212. Springer, 2002. ↑66

[HDP05] Lei Hu, Jun-Wu Dong, and Ding-Yi Pei. Implementation of Cryptosystems Based on Tate Pairing. *Journal of Computer Science and Technology*, 20(2) :264–269, 2005. ↑49

[HFPS99] R Housley, W Ford, W Polk, and D Solo. Rfc 2459 : Internet x. 509 public key infrastructure certificate and crl profile. *Status : PROPOSED STANDARD*, 9, 1999. ↑6

[HHHS00] Panu Hämäläinen, Marko Hännikäinen, Timo Hämäläinen, and Jukka Saarinen. Hardware implementation of the improved wep and rc4 encryption algorithms for wireless terminals. In *EUPSICO 2000 : European signal processing conference*, pages 2289–2292, 2000. ↑5

[Hin11] Marc Hindry. *Arithmetics*. Springer Science & Business Media, 2011. ↑26

[HMA+08] Naofumi Homma, Atsushi Miyamoto, Takafumi Aoki, Akashi Satoh, and Adi Shamir. Collision-based Power Analysis of Modular Exponentiation Using Chosen-message Pairs. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 15–29. Springer, 2008. ↑75

[HMV06] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Science & Business Media, 2006. ↑18, ↑29

[HNI+06] Naofumi Homma, Sei Nagashima, Yuichi Imai, Takafumi Aoki, and Akashi Satoh. High-Resolution Side-Channel Attack Using Phase-Based Waveform Matching. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 187–200. Springer, 2006. ↑75

[HSV06] Florian Hess, Nigel P Smart, and Frederik Vercauteren. The Eta Pairing Revisited. *IEEE Transactions on Information Theory*, 52(10) :4595–4602, 2006. ↑49, ↑50

[HTM09] Neil Hanley, Michael Tunstall, and William P. Marnane. *Unknown Plaintext Template Attacks*, pages 148–162. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. ↑120

[HWH13] Ralf Hund, Carsten Willems, and Thorsten Holz. Practical Timing Side Channel Attacks Against Kernel Space ASLR. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 191–205. IEEE, 2013. ↑102

[Ica09] Thomas Icart. How to Hash into Elliptic Curves. In *Advances in Cryptology-CRYPTO 2009*, pages 303–316. Springer, 2009. ↑40

[Isa15] Walter Isaacson. *Les innovateurs*. JC Lattès, 2015. ↑61

[ITK+16] Kengo Iokibe, Nobuhiro Tai, Hiroto Kagotani, Hiroyuki Onishi, Yoshitaka Toyota, and Tetsushi Watanabe. Analysis of side-channel information leaking behavior in cryptographic circuit using internal current source. *IEEJ Transactions on Fundamentals and Materials*, 136(6) :365–371, 2016. ↑97

[Jac80] Nathan Jacobson. Basic algebra ii, ch. 9, 1980. ↑31

[JFEMG16] Damien Jauvart, Jacques J.A. Fournier, Nadia El Mrabet, and Louis Goubin. Improving Side-Channel Attacks against Pairing-Based Cryptography. *CRiSIS*, 2016. ↑11, ↑138

[JFG17] Damien Jauvart, Jacques J.A. Fournier, and Louis Goubin. First Practical Side-Channel Attack to Defeat Point Randomization in Secure Implementations of Pairing-Based Cryptography. In *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications - Volume 4 : SECRYPT, (ICETE 2017)*. INSTICC, ScitePress, 2017. ↑11, ↑139

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *International journal of information security*, 1(1) :36–63, 2001. ↑55

[JOP14] Antoine Joux, Andrew Odlyzko, and Cécile Pierrot. The past, evolving present, and future of the discrete logarithm. In *Open Problems in Mathematics and Computational Science*, pages 5–36. Springer, 2014. ↑15, ↑29

[Jou00] Antoine Joux. A One Round Protocol for Tripartite Diffie–Hellman. In *International Algorithmic Number Theory Symposium*, pages 385–393. Springer, 2000. ↑6, ↑52, ↑54, ↑84

[Joy03] Marc Joye. Elliptic curves and side-channel attacks. *ST Journal of System Research 4(1) :283-–306*, 2003. ↑65

[KAF+10] Thorsten Kleinjung, Kazumaro Aoki, Jens Franke, Arjen Lenstra, Emmanuel Thomé, Joppe Bos, Pierrick Gaudry, Alexander Kruppa, Peter Montgomery, Dag Arne Osvik, et al. Factorization of a 768-bit rsa modulus. In *CRYPTO 2010*, volume 6223, pages 333–350. Springer, 2010. ↑3

[Kah96] David Kahn. *The Codebreakers : The comprehensive history of secret communication from ancient times to the internet*. Simon and Schuster, 1996. ↑2

[KAK96] C Kaya Koc, Tolga Acar, and Burton S Kaliski. Analyzing and Comparing Montgomery Multiplication Algorithms. *IEEE micro*, 16(3) :26–33, 1996. ↑21, ↑22, ↑86

[Kal95] Burton S Kaliski. The Montgomery Inverse and Its Applications. *IEEE transactions on computers*, 44(8) :1064–1065, 1995. ↑23

[Kas63] Friedrich Kasiski. *Die Geheimschriften und die Dechiffrierkunst*. 1863. ↑2

[KB16] Taechan Kim and Razvan Barbulescu. Extended Tower Number Field Sieve : A New Complexity for the Medium Prime Case. In *Annual Cryptology Conference*, pages 543–571. Springer, 2016. ↑58

[Ker83] Auguste Kerckhoffs. La cryptographie militaire, 1883. ↑4, ↑62

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Advances in cryptology—CRYPTO'99*, pages 789–789. Springer, 1999. ↑9, ↑62, ↑64, ↑70

[KJJR11] Paul Kocher, Joshua Jaffe, Benjamin Jun, and Pankaj Rohatgi. Introduction to differential power analysis. *Journal of Cryptographic Engineering*, 1(1) :5–27, 2011. ↑70

[KM05] Neal Koblitz and Alfred Menezes. Pairing-Based Cryptography at High Security Levels. In *IMA International Conference on Cryptography and Coding*, pages 13–36. Springer, 2005. ↑23

[Kob87] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177) :203–209, 1987. ↑6, ↑26

[Koc96] Paul Kocher. Timing Attacks on Implementations of Diffie–Hellman, RSA, DSS, and Other Systems. In *Advances in Cryptology—CRYPTO'96*, pages 104–113. Springer, 1996. ↑9

[KTH+06] Tae Hyun Kim, Tsuyoshi Takagi, Dong-Guk Han, Ho Won Kim, and Jongin Lim. Side Channel Attacks and Countermeasures on Pairing Based Cryptosystems over Binary Fields. *Cryptology and Network Security*, pages 168–181, 2006. ↑76, ↑78, ↑79, ↑80, ↑139

[Las14] Ronan Lashermes. *Etude de la sécurité des implémentations de couplage*. PhD thesis, Versailles-St Quentin en Yvelines, 2014. ↑102

[Len85] Hendrik Lenstra. Elliptic curve factorization. *announcement of February*, 14 :16, 1985. ↑26

[LFG13] Ronan Lashermes, Jacques Fournier, and Louis Goubin. Inverting the final exponentiation of tate pairings on ordinary elliptic curves using faults. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 365–382. Springer, 2013. ↑10

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite Fields*, volume 20. Cambridge university press, 1997. ↑14, ↑38

[LPEM+14] Ronan Lashermes, Marie Paindavoine, Nadia El Mrabet, Jacques JA Fournier, and Louis Goubin. Practical validation of several fault attacks against the miller algorithm. In *Fault Diagnosis and Tolerance in Cryptography (FDTC), 2014 Workshop on*, pages 115–122. IEEE, 2014. ↑10

[LPR13] Victor Lomné, Emmanuel Prouff, and Thomas Roche. Behind the Scene of Side Channel Attacks. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 506–525. Springer, 2013. ↑9

[LPR+14] Victor Lomné, Emmanuel Prouff, Matthieu Rivain, Thomas Roche, and Adrian Thillard. How to estimate the success rate of higher-order side-channel attacks. In *Cryptographic Hardware and Embedded Systems–CHES 2014*, pages 35–54. Springer, 2014. ↑88, ↑89

[LZ94] Georg-Johann Lay and Horst G. Zimmer. *Constructing elliptic curves with given group order over large finite fields*, pages 250–263. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994. ↑39

[Man04] Stefan Mangard. Hardware Countermeasures against DPA – A Statistical Analysis of Their Effectiveness. In *Cryptographers' Track at the RSA Conference*, pages 222–235. Springer, 2004. ↑9

[MH10] Marcel Medwed and Christoph Herbst. Randomizing the Montgomery Multiplication to Repel Template Attacks on Multiplicative Masking. *COSADE, Lecture Notes in Computer Science*, 9, 2010. ↑140

[MHZ15] Lukas Malina, Jan Hajny, and Vaclav Zeman. Usability of pairing-based cryptography on smartphones. In *Telecommunications and Signal Processing (TSP), 2015 38th International Conference on*, pages 617–621. IEEE, 2015. ↑137

[Mil85] Victor S Miller. Elliptic Curves and their use in Cryptography. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 417–426. Springer, 1985. ↑6, ↑26

[Mil86] Victor S. Miller. Short Programs for functions on Curves. *Unpublished manuscript*, 97 :101–102, 1986. ↑44, ↑45, ↑117, ↑119

[MKHO07] Seiichi Matsuda, Naoki Kanayama, Florian Hess, and Eiji Okamoto. Optimised versions of the Ate and Twisted Ate Pairings. In *IMA International Conference on Cryptography and Coding*, pages 302–312. Springer, 2007. ↑50

[MNT01] Atsuko Miyaji, Masaki Nakabayashi, and Shunzou Takano. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE transactions on fundamentals of electronics, communications and computer sciences*, 84(5) :1234–1243, 2001. ↑39

[Mon85] Peter L Montgomery. Modular Multiplication Without Trial Division. *Mathematics of computation*, 44(170) :519–521, 1985. ↑19, ↑22

[Mon87] Peter L Montgomery. Speeding the Pollard and Elliptic Curve Methods of Factorization. *Mathematics of computation*, 48(177) :243–264, 1987. ↑9, ↑65

[Mor91] François Morain. Building cyclic elliptic curves modulo large primes. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 328–336. Springer, 1991. ↑39

[MOV93] Alfred J Menezes, Tatsuaki Okamoto, and Scott A Vanstone. Reducing Elliptic Curve Logarithms to Logarithms in a Finite Field. *iEEE Transactions on information Theory*, 39(5) :1639–1646, 1993. ↑6, ↑37, ↑53

[MS00] Rita Mayer-Sommer. Smartly Analyzing the Simplicity and the Power of Qimple Power Analysis on Smartcards. In *CHES*, pages 78–92. Springer, 2000. ↑103

[MSS16] Alfred Menezes, Palash Sarkar, and Shashank Singh. Challenges with Assessing the Impact of NFS Advances on the Security of Pairing-based Cryptography. In *Proceedings of Mycrypt*, 2016. ↑8, ↑58, ↑100, ↑137

[MVOV96] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 1996. ↑49

[NBB+01] James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the development of the advanced encryption standard (aes). *Journal of Research of the National Institute of Standards and Technology*, 106(3) :511, 2001. ↑4

[NFM+17] Makoto Nagata, Daisuke Fujimoto, Noriyuki Miura, Naofumi Homma, Yu-ichi Hayashi, and Kazuo Sakiyama. Protecting cryptographic integrated circuits with side-channel information. *IEICE Electronics Express*, 14(2) :20162005–20162005, 2017. ↑97

[NNS10] Michael Naehrig, Ruben Niederhagen, and Peter Schwabe. New software speed records for cryptographic pairings. In *International Conference on Cryptology and Information Security in Latin America*, pages 109–123. Springer, 2010. ↑46, ↑51

[Nov03] Roman Novak. *Side-Channel Attack on Substitution Blocks*, pages 307–318. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ↑61

[NT94] B Clifford Neuman and Theodore Ts'o. Kerberos : An Authentication Service for Computer Networks. *IEEE Communications magazine*, 32(9) :33–38, 1994. ↑5

[OM06] Elisabeth Oswald and Stefan Mangard. *Template Attacks on Masking—Resistance Is Futile*, pages 243–256. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ↑120, ↑121, ↑122, ↑139

[OP01] Tatsuaki Okamoto and David Pointcheval. The Gap-Problems : A New Class of Problems for the Security of Cryptographic Schemes. In *International Workshop on Public Key Cryptography*, pages 104–118. Springer, 2001. ↑53

[OP12] David Oswald and Christof Paar. Improving Side-Channel Analysis with Optimal Pre-Processing. In *Eleventh Smart Card Research and Advanced Application Conference*, pages 219–233. Springer, 2012. ↑102

[Osw03] Elisabeth Oswald. *On Side-Channel Attacks and the Application of Algorithmic Countermeasures*. Ph.D, 2003. ↑103

[PCD96] Daniel Perrin, Marc Cabanes, and Martine Duchene. *Cours d'algèbre*, volume 30. Ellipses Paris, 1996. ↑15, ↑17, ↑38, ↑40

[PDG08] Bernard Parisse and Renée De Graeve. Giac/xcas, a free computer algebra system. Technical report, Technical report, University of Grenoble, 2008. ↑25

[Per95] Daniel Perrin. Géométrie algébrique, une introduction savoirs actuels. *InterEditions, Paris*, 1995. ↑27, ↑28, ↑41

[PHG17] Stjepan Picek, Annelie Heuser, and Sylvain Guilley. Template attack vs bayes classifier. Cryptology ePrint Archive, Report 2017/531, 2017. ↑141

[PIMT15] Guilherme Perin, Laurent Imbert, Philippe Maurine, and Lionel Torres. Vertical and Horizontal Correlation Attacks on RNS-Based Exponentiations. *Journal of Cryptographic Engineering*, 5(3) :171–185, 2015. ↑66

[PM11] Weibo Pan and WP Marnane. A correlation power analysis attack against Tate pairing on FPGA. *Reconfigurable Computing : Architectures, Tools and Applications*, pages 340–349, 2011. ↑76

[pol04] *Histoire*. Quarto Gallimard, 2004. ↑2

[PV04] Dan Page and Frederik Vercauteren. Fault and Side-Channel Attacks on Pairing Based Cryptography. 2004. ↑76, ↑77, ↑78, ↑79, ↑137

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA) : Measures and Countermeasures for Smart Cards. *Smart Card Programming and Security*, pages 200–210, 2001. ↑60

[Qui00] Jean-Jacques Quisquater. A new tool for non-intrusive analysis of smart cards based on electromagnetic emissions. The SEMA and DEMA methods. *Eurocrypt2000 Rump Session, May*, 2000. ↑9

[RD01] Vincent Rijmen and Joan Daemen. Advanced encryption standard. *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, pages 19–22, 2001. ↑4

[RO04] Christian Rechberger and Elisabeth Oswald. Practical Template Attacks. In *International Workshop on Information Security Applications*, pages 440–456. Springer, 2004. ↑109

[RSA78] Ron Rivest, Adi Shamir, and Leonard Adleman. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978. ↑5

[S+77] Data Encryption Standard et al. Federal information processing standards publication 46. *National Bureau of Standards, US Department of Commerce*, 1977. ↑4

[SBC+09] Michael Scott, Naomi Benger, Manuel Charlemagne, Luis J Dominguez Perez, and Ezekiel J Kachisa. On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves. In *International Conference on Pairing-Based Cryptography*, pages 78–88. Springer, 2009. ↑48, ↑49

[Sco05] Michael Scott. Computing the Tate pairing. *CT-RSA*, pages 293–304, 2005. ↑79, ↑140

[Sha49] Claude E Shannon. Communication theory of secrecy systems. *Bell Labs Technical Journal*, 28(4) :656–715, 1949. ↑2

[Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Workshop on the Theory and Application of Cryptographic Techniques*, pages 47–53. Springer, 1984. ↑52

[Sil09] Joseph H Silverman. *The Arithmetic of Elliptic Curves*, volume 106. Springer Science & Business Media, 2009. ↑26, ↑27, ↑30, ↑31, ↑33, ↑40, ↑41, ↑42, ↑43

[Sko01] Thomas Skotnicki. *Circuits intégrés CMOS sur silicium*. Ed. Techniques Ingénieur, 2001. ↑60

[SKS09] François-Xavier Standaert, François Koeune, and Werner Schindler. How to Compare Profiled Side-Channel Attacks ? In *International Conference on Applied Cryptography and Network Security*, pages 485–498. Springer, 2009. ↑73

[SLFP04] Kai Schramm, Gregor Leander, Patrick Felke, and Christof Paar. A Collision-Attack on AES. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 163–175. Springer, 2004. ↑74, ↑75

[SLP05] Werner Schindler, Kerstin Lemke, and Christof Paar. A Stochastic Model for Differential Side Channel Cryptanalysis. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 30–46. Springer, 2005. ↑9, ↑73

[ST04] Adi Shamir and Eran Tromer. Acoustic cryptanalysis. 2004. ↑9

[STM] STMicroelectronics. Stm32f1 series. http://www.st.com/en/microcontrollers/stm32f1-series.html?querycriteria=productId=SS1031. ↑96

[SvdW06] Andrew Shallue and Christiaan E van de Woestijne. Construction of Rational Points on Elliptic Curves over Finite Fields. In *International Algorithmic Number Theory Symposium*, pages 510–524. Springer, 2006. ↑40

[SW05] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 457–473. Springer, 2005. ↑56

[Tib12] Mehdi Tibouchi. A Note on Hasing to BN Curves. *SCIS. IEICE*, 2012. ↑40

[TKL86] JJ Thomas, JM Keller, and GN Larsen. The Calcualtion of Multiplicative Inverses Over GF(p) Efficiently Where p is a Mersenne Prime. *IEEE transactions on computers*, 35(5) :478–482, 1986. ↑22

[UW14] Thomas Unterluggauer and Erich Wenger. Practical Attack on Bilinear Pairings to Disclose the Secrets of Embedded Devices. *ARES*, pages 69–77, 2014. ↑77, ↑80, ↑86, ↑107

[Vai00] Juha T Vainio. Bluetooth security. In *Proceedings of Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, Seminar on Internetworking : Ad Hoc Networking, Spring*, 2000. ↑5

[Van92] Scott Vanstone. Responses to NIST's Proposal. *Communications of the ACM*, 35 :50–52, 1992. ↑26

[VCGRS12] Nicolas Veyrat-Charvillon, Benoît Gérard, Mathieu Renauld, and François-Xavier Standaert. An Optimal Key Enumeration Algorithm and its Application to Side-Channel Attacks. In *International Conference on Selected Areas in Cryptography*, pages 390–406. Springer, 2012. ↑141

[VDRZ15] M. Varchola, M. Drutarovsky, M. Repka, and P. Zajac. Side channel attack on multiprecision multiplier used in protected ECDSA implementation. In *ReConFig*, pages 1–6, Dec 2015. ↑127, ↑129, ↑131

[Ver06] Frederik Vercauteren. The Ate Pairing. 2006. ↑49, ↑50

[Ver08] Frederik Vercauteren. Optimal Ate Pairings. 2008. ↑51

[Ver10] Frederik Vercauteren. Optimal Pairings. *IEEE Transactions on Information Theory*, 56(1) :455–461, 2010. ↑51

[Vit08] Vanessa Vitse. Couplages sur courbes elliptiques définies sur des corps finis. 2008. ↑32, ↑34

[Wal01] Colin D Walter. Sliding Windows Succumbs to Big Mac Attack. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 286–299. Springer, 2001. ↑10, ↑66

[WL07] Zhenghong Wang and Ruby B Lee. New Cache Designs for Thwarting Software Cache-Based Side Channel Attacks. In *ACM SIGARCH Computer Architecture News*, volume 35, pages 494–505. ACM, 2007. ↑102

[WS06] Claire Whelan and Mike Scott. Side Channel Analysis of Practical Pairing Implementations : Which Path Is More Secure ? *VIETCRYPT 2006*, pages 99–114, 2006. ↑76, ↑77, ↑80, ↑86, ↑110

[YLMH05] Sung-Ming Yen, Wei-Chih Lien, SangJae Moon, and JaeCheol Ha. Power Analysis by Exploiting Chosen Message and Internal Collisions–vulnerability of checking mechanism for RSA-Decryption. In *International Conference on Cryptology in Malaysia*, pages 183–195. Springer, 2005. ↑75

Chapitre 7

Annexes

A Appendice 1

Cette Annexe vient supporter la section *Arithmétique sur les extensions de corps* en page 23 de ce manuscrit.

A.1 Arithmétiques sur les extensions quadratiques

La multiplication d'éléments de $\mathbb{F}_{p^2}$ est fourni par l'équation 2.13 page 24. Le procédé de calcul est donc le suivant :

Entrées : Deux éléments x et y de $\mathbb{F}_{p^2}$ tels que $x = x_0 + x_1 u$ et $y = y_0 + y_1 u$.
Sorties : $z = xy \in \mathbb{F}_{p^2}$.

1 $t_0 \leftarrow x_0 y_0$;
2 $t_1 \leftarrow x_1 y_1$;
3 $t_2 \leftarrow x_0 + x_1$;
4 $t_3 \leftarrow y_0 + y_1$;
5 $t_4 \leftarrow t_2 t_3$;
6 $t_5 \leftarrow t_4 - t_0$;
7 $z_1 \leftarrow t_5 - t_1$;
8 $t_6 \leftarrow t_1 \beta$;
9 $z_0 \leftarrow t_0 + t_6$;
10 **retourner** $z = z_0 + z_1 u$;

Algorithme 7.1 : Multiplication dans $\mathbb{F}_{p^2}$

Entrées : Un élément x de $\mathbb{F}_{p^2}$ tel que $x = x_0 + x_1 u$.
Sorties : $z = x^2 \in \mathbb{F}_{p^2}$.

1 $t_0 \leftarrow x_0^2$;
2 $t_1 \leftarrow x_1^2$;
3 $t_2 \leftarrow x_0 + x_1$;
4 $t_3 \leftarrow t_2^3$;
5 $t_4 \leftarrow t_3 - t_0$;
6 $z_1 \leftarrow t_4 - t_1$;
7 $t_5 \leftarrow t_1 \beta$;
8 $z_0 \leftarrow t_0 + t_5$;
9 **retourner** $z = z_0 + z_1 u$;

Algorithme 7.2 : Mise au carré dans $\mathbb{F}_{p^2}$

Entrées : Un élément x de $\mathbb{F}_{p^2}$ tel que $x = x_0 + x_1 u$.
Sorties : $z = x^{-1} \in \mathbb{F}_{p^2}$.

1 $t_0 \leftarrow x_0^2$;
2 $t_1 \leftarrow x_1^2$;
3 $t_2 \leftarrow \beta t_1$;
4 $t_3 \leftarrow t_0 - t_2$;
5 $t_4 \leftarrow t_3^{-1}$;
6 $y_0 \leftarrow x_0 t_3$;
7 $y_1 \leftarrow -x_1 t_3$;
8 **retourner** $z = z_0 + z_1 u$;

Algorithme 7.3 : Inversion dans $\mathbb{F}_{p^2}$

A.2 Arithmétiques sur les extensions cubiques

Entrées : Deux éléments x et y de $\mathbb{F}_{p^3}$ tels que $x = x_0 + x_1 v + x_2 v^2$ et $y = y_0 + y_1 v + y_2 v^2$.
Sorties : $z = xy \in \mathbb{F}_{p^3}$.

1 $t_0 \leftarrow x_0 y_0$;
2 $t_1 \leftarrow x_1 y_1$;
3 $t_2 \leftarrow x_2 + y_2$;
4 $t_3 \leftarrow x_0 + x_1$;
5 $t_4 \leftarrow x_0 + x_2$;
6 $t_5 \leftarrow x_1 + x_2$;
7 $t_6 \leftarrow y_0 + y_1$;
8 $t_7 \leftarrow y_0 + y_2$;
9 $t_8 \leftarrow y_1 + y_2$;
10 $t_9 \leftarrow t_5 t_8$;
11 $t_{10} \leftarrow t_3 t_6$;
12 $t_{11} \leftarrow t_4 t_7$;
13 $t_{12} \leftarrow t_9 - t_1$;
14 $t_{13} \leftarrow t_{12} - t_2$;
15 $t_{14} \leftarrow t_{10} - t_0$;
16 $t_{15} \leftarrow t_{14} - t_1$;
17 $t_{16} \leftarrow t_{11} - t_0$;
18 $t_{17} \leftarrow t_{16} - t_2$;
19 $z_2 \leftarrow t_{17} + t_1$;
20 $t_{18} \leftarrow t_{13} \xi$;
21 $z_0 \leftarrow t_{18} + t_0$;
22 $t_{19} \leftarrow t_2 \xi$;
23 $z_1 \leftarrow t_{15} + t_{19}$;
24 **retourner** $z = z_0 + z_1 v + z_2 v^2$;

Algorithme 7.4 : Multiplication dans $\mathbb{F}_{p^3}$

Entrées : Un élément x de $\mathbb{F}_{p^3}$ tel que $x = x_0 + x_1 v + x_2 v^2$.
Sorties : $z = x^2 \in \mathbb{F}_{p^3}$.

1 $t_0 \leftarrow x_0^2$;
2 $t_1 \leftarrow x_1^2$;
3 $t_2 \leftarrow x_2^2$;
4 $t_3 \leftarrow x_0 x_1$;
5 $t_4 \leftarrow x_0 x_2$;
6 $t_5 \leftarrow x_1 x_2$;
7 $t_6 \leftarrow t_2 \xi$;
8 $t_7 \leftarrow 2t_3$;
9 $t_8 \leftarrow 2t_4$;
10 $t_9 \leftarrow t_5 \xi$;
11 $t_{10} \leftarrow 2t_9$;
12 $z_0 \leftarrow t_0 + t_{10}$;
13 $z_1 \leftarrow t_6 + t_7$;
14 $z_2 \leftarrow t_1 + t_8$;
15 **retourner** $z = z_0 + z_1 v + z_2 v^2$;

Algorithme 7.5 : Mise au carré dans $\mathbb{F}_{p^3}$

Entrées : Un élément x de $\mathbb{F}_{p^3}$ tel que $x = x_0 + x_1 v + x_2 v^2$.
Sorties : $z = x^{-1} \in \mathbb{F}_{p^3}$.

1 $t_0 \leftarrow x_0^2$;
2 $t_1 \leftarrow x_1^2$;
3 $t_2 \leftarrow x_2^2$;
4 $t_3 \leftarrow x_0 x_1$;
5 $t_4 \leftarrow x_0 x_2$;
6 $t_5 \leftarrow x_1 x_2$;
7 $t_6 \leftarrow t_5 \xi$;
8 $t_7 \leftarrow t_0 - t_6$;
9 $t_8 \leftarrow t_2 \xi$;
10 $t_9 \leftarrow t_8 - t_3$;
11 $t_{10} \leftarrow t_1 - t_4$;
12 $t_{11} \leftarrow x_0 t_7$;
13 $t_{12} \leftarrow x_2 t_9$;
14 $t_{13} \leftarrow t_{12} \xi$;
15 $t_{14} \leftarrow x_1 t_{10}$;
16 $t_{15} \leftarrow t_{14} \xi$;
17 $t_{16} \leftarrow t_{11} + t_{13}$;
18 $t_{17} \leftarrow t_{16} + t_{15}$;
19 $t_{18} \leftarrow t_{17}^{-1}$;
20 $y_0 \leftarrow t_7 t_{18}$;
21 $y_1 \leftarrow t_9 t_{18}$;
22 $y_2 \leftarrow t_{10} t_{18}$;
23 **retourner** $z = z_0 + z_1 v + z_2 v^2$;

Algorithme 7.6 : Inversion dans $\mathbb{F}_{p^3}$

A.3 Arithmétiques sur $\mathbb{F}_{p^6}$

Entrées : Deux éléments x et y de $\mathbb{F}_{p^6}$ tels que $x = x_0 + x_1 v + x_2 v^2$ et $y = y_0 + y_1 v + y_2 v^2$ avec $x_0, x_1, x_2, y_0, y_1, y_2 \in \mathbb{F}_{p^2}$.
Sorties : $z = x + y \in \mathbb{F}_{p^6}$.

1 $z_0 \leftarrow x_0 + y_0$;
2 $z_1 \leftarrow x_1 + y_1$;
3 $z_2 \leftarrow x_2 + y_2$;
4 **retourner** $z = z_0 + z_1 v + z_2 v^2$;

Algorithme 7.7 : Addition dans $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$

Entrées : Un éléments x de $\mathbb{F}_{p^6}$ tel que $x = x_0 + x_1 v + x_2 v^2$ avec $x_0, x_1, x_2 \in \mathbb{F}_{p^2}$.
Sorties : $z = x\xi \in \mathbb{F}_{p^6}$.

1 $z_0 \leftarrow x_2 \xi$;
2 **retourner** $z = z_0 + x_0 v + x_1 v^2$;

Algorithme 7.8 : Multiplication par ξ dans $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$

Entrées : Deux éléments x et y de $\mathbb{F}_{p^6}$ tels que $x = x_0 + x_1 v + x_2 v^2$ et $y = y_0 + y_1 v + y_2 v^2$ avec $x_0, x_1, x_2, y_0, y_1, y_2 \in \mathbb{F}_{p^2}$.
Sorties : $z = xy \in \mathbb{F}_{p^6}$.

1 $t_0 \leftarrow x_0 y_0$;
2 $t_1 \leftarrow x_1 y_1$;
3 $t_2 \leftarrow x_2 y_2$;
4 $t_3 \leftarrow x_1 + x_2$;
5 $t_4 \leftarrow y_1 + y_2$;
6 $t_5 \leftarrow t_3 t_4$;
7 $t_6 \leftarrow t_5 - t_1$;
8 $t_7 \leftarrow t_6 - t_2$;
9 $t_8 \leftarrow t_7 \xi$;
10 $z_0 \leftarrow t_8 + t_0$;
11 $t_9 \leftarrow x_0 + x_1$;
12 $t_{10} \leftarrow y_0 + y_1$;
13 $t_{11} \leftarrow t_9 t_{10}$;
14 $t_{12} \leftarrow t_{11} - t_0$;
15 $t_{13} \leftarrow t_{12} - t_1$;
16 $t_{14} \leftarrow t_2 \xi$;
17 $z_1 \leftarrow t_{13} + t_{14}$;
18 $t_{15} \leftarrow x_0 + x_2$;
19 $t_{16} \leftarrow y_0 + y_2$;
20 $t_{17} \leftarrow t_{16} - t_0$;
21 $t_{18} \leftarrow t_{17} - t_2$;
22 $z_2 \leftarrow t_{18} + t_1$;
23 **retourner** $z = z_0 + z_1 v + z_2 v^2$;

Algorithme 7.9 : Multiplication dans $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$

Entrées : Un éléments x de $\mathbb{F}_{p^6}$ tel que $x = x_0 + x_1 v + x_2 v^2$ avec $x_0, x_1, x_2 \in \mathbb{F}_{p^2}$ et $a \in \mathbb{F}_{p^2}$.
Sorties : $z = xa \in \mathbb{F}_{p^6}$.

1 $z_0 \leftarrow x_0 a$;
2 $z_1 \leftarrow x_1 a$;
3 $z_2 \leftarrow x_2 a$;
4 **retourner** $z = z_0 + x_0 v + x_1 v^2$;

Algorithme 7.10 : Multiplication par $a \in \mathbb{F}_{p^2}$ dans $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$

Entrées : Un éléments x de $\mathbb{F}_{p^6}$ tel que $x = x_0 + x_1 v + x_2 v^2$ avec $x_0, x_1, x_2 \in \mathbb{F}_{p^2}$ et $a_0, a_1 \in \mathbb{F}_{p^2}$.
Sorties : $z = x(a_0 + a_1 v) \in \mathbb{F}_{p^6}$.

1 $t_0 \leftarrow x_0 a_0$;
2 $t_1 \leftarrow x_1 a_1$;
3 $t_2 \leftarrow x_1 + x_2$;
4 $t_3 \leftarrow t_2 a_1$;
5 $t_4 \leftarrow t_3 - t_1$;
6 $t_5 \leftarrow t_4 \xi$;
7 $z_0 \leftarrow t_5 + t_0$;
8 $t_6 \leftarrow x_0 + x_1$;
9 $t_7 \leftarrow a_0 + a_1$;
10 $t_8 \leftarrow t_6 t_7$;
11 $t_9 \leftarrow t_8 - t_0$;
12 $z_1 \leftarrow t_9 - t_1$;
13 $t_{10} \leftarrow x_2 a_1$;
14 $z_2 \leftarrow t_{10} + t_1$;
15 **retourner** $z = z_0 + x_0 v + x_1 v^2$;

Algorithme 7.11 : Multiplication par $a_0 + a_1 v$ dans $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$

Entrées : Un élément x de $\mathbb{F}_{p^6}$ tels que $x = x_0 + x_1 v + x_2 v^2$ avec $x_0, x_1, x_2 \in \mathbb{F}_{p^2}$.
Sorties : $z = x^2 \in \mathbb{F}_{p^6}$.

1 $t_0 \leftarrow x_0 x_1$;
2 $t_1 \leftarrow 2t_0$;
3 $t_2 \leftarrow x_2^2$;
4 $t_3 \leftarrow t_2 \xi$;
5 $z_1 \leftarrow t_3 + t_1$;
6 $t_4 \leftarrow t_1 - t_2$;
7 $t_5 \leftarrow x_0^2$;
8 $t_6 \leftarrow x_0 - x_1$;
9 $t_7 \leftarrow t_6 + x_2$;
10 $t_8 \leftarrow x_1 x2$;
11 $t_9 \leftarrow 2t_8$;
12 $t_{10} \leftarrow t_7^2$;
13 $t_{11} \leftarrow t_9 \xi$;
14 $z_0 \leftarrow t_{10} + t_5$;
15 $t_{12} \leftarrow t_4 + t_{10}$;
16 $t_{13} \leftarrow t_{12} + t_9$;
17 $z_2 \leftarrow t_{13} - t_5$;
18 **retourner** $z = z_0 + z_1 v + z_2 v^2$;

Algorithme 7.12 : Mise au carré dans $\mathbb{F}_{p^6} = \mathbb{F}_{p^2}[v]/(v^3 - \xi)$

Entrées : Un élément x de $\mathbb{F}_{p^6}$ tels que $x = x_0 + x_1 v + x_2 v^2$ avec $x_0, x_1, x_2 \in \mathbb{F}_{p^2}$.
Sorties : $z = x^{-1} \in \mathbb{F}_{p^6}$.

1 $t_0 \leftarrow x_0^2$;
2 $t_1 \leftarrow x_1^2$;
3 $t_2 \leftarrow x_2^2$;
4 $t_3 \leftarrow x_0 x_1$;
5 $t_4 \leftarrow x_0 x_2$;
6 $t_5 \leftarrow x_1 x_2$;
7 $t_6 \leftarrow t_5 \xi$;
8 $t_7 \leftarrow t_0 - t_6$;
9 $t_8 \leftarrow t_2 \xi$;
10 $t_9 \leftarrow t_8 - t_3$;
11 $t_{10} \leftarrow t_1 t_4$;
12 $t_{11} \leftarrow x_0 t_7$;
13 $t_{12} \leftarrow x_2 t_9$;
14 $t_{13} \leftarrow t_{12} \xi$;
15 $t_{14} \leftarrow t_{11} + t_{13}$;
16 $t_{15} \leftarrow x_1 t_{10}$;
17 $t_{16} \leftarrow t_{15} \xi$;
18 $t_{17} \leftarrow t_{14} + t_{16}$;
19 $t_{18} \leftarrow t_{17}^{-1}$;
20 $z_0 \leftarrow t_7 t_{18}$;
21 $z_1 \leftarrow t_9 t_{18}$;
22 $z_2 \leftarrow t_{10} t_{18}$;
23 **retourner** $z = z_0 + z_1 v + z_2 v^2$;

Algorithme 7.13 : Inversion dans $\mathbb{F}_{p^6} = {}^{\mathbb{F}_{p^2}[v]}\!/_{(v^3 - \xi)}$

A.4 Arithmétiques sur $\mathbb{F}_{p^{12}}$

Entrées : Deux éléments x et y de $\mathbb{F}_{p^{12}}$ tels que $x = x_0 + x_1 w$ et $y = y_0 + y_1 w$ avec $x_0, x_1, y_0, y_1 \in \mathbb{F}_{p^6}$.
Sorties : $z = x + y \in \mathbb{F}_{p^{12}}$.

1 $z_0 \leftarrow x_0 + y_0$;
2 $z_1 \leftarrow x_1 + y_1$;
3 **retourner** $z = z_0 + z_1 w$;

Algorithme 7.14 : Addition dans $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2 - \gamma)$

Entrées : Deux éléments x et y de $\mathbb{F}_{p^{12}}$ tels que $x = x_0 + x_1 w$ et $y = y_0 + y_1 w$ avec $x_0, x_1, y_0, y_1 \in \mathbb{F}_{p^6}$.
Sorties : $z = xy \in \mathbb{F}_{p^{12}}$.

1 $t_0 \leftarrow x_0 y_0$;
2 $t_1 \leftarrow x_1 y_1$;
3 $t_2 \leftarrow t_1 \gamma$;
4 $z_0 \leftarrow t_0 + t_2$;
5 $t_3 \leftarrow x_0 + x_1$;
6 $t_4 \leftarrow y_0 + y_1$;
7 $t_5 \leftarrow t_3 t_4$;
8 $t_6 \leftarrow t_5 - t_0$;
9 $z_1 \leftarrow t_6 - t_1$;
10 **retourner** $z = z_0 + z_1 w$;

Algorithme 7.15 : Multiplication dans $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2 - \gamma)$

Entrées : Deux éléments x et y de $\mathbb{F}_{p^{12}}$ tels que $x = x_0 + x_1 w$ et $y = y_0 + y_1 w$ avec $x_0, x_1 \in \mathbb{F}_{p^6}$, $y_0 = y_{00} + 0v + 0v^2 \in \mathbb{F}_{p^2}$ et $y_1 = y_{10} + y_{11} v + 0v^2 \in \mathbb{F}_{p^6}$.
Sorties : $z = xy \in \mathbb{F}_{p^{12}}$.

1 $t_0 \leftarrow x_0 y_0$; // Algorithme 7.10
2 $t_1 \leftarrow x_1 y_1$; // Algorithme 7.11
3 $t_2 \leftarrow t_1 \gamma$;
4 $z_0 \leftarrow t_0 + t_2$;
5 $t_3 \leftarrow y_0 + y_{10}$;
6 $t_4 \leftarrow t_3 v + y_{11} v + 0v^2 \in \mathbb{F}_{p^6}$;
7 $t_5 \leftarrow x_0 + x_1$;
8 $t_6 \leftarrow t_5 t_2$; // Algorithme 7.11
9 $t_7 \leftarrow t_6 - t_0$;
10 $z_1 \leftarrow t_7 - t_1$;
11 **retourner** $z = z_0 + z_1 w$;

Algorithme 7.16 : Multiplication dans $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2 - \gamma)$ sans multiplications par 0.

Entrées : Un élément x de $\mathbb{F}_{p^{12}}$ tels que $x = x_0 + x_1 w$ avec $x_0, x_1 \in \mathbb{F}_{p^6}$.
Sorties : $z = x^2 \in \mathbb{F}_{p^{12}}$.

1 $t_0 \leftarrow x_0^2$;
2 $t_1 \leftarrow x_1^2$;
3 $t_2 \leftarrow x_0 + x_1$;
4 $t_3 \leftarrow t_2^3$;
5 $t_4 \leftarrow t_3 - t_0$;
6 $z_1 \leftarrow t_4 - t_1$;
7 $t_5 \leftarrow t_1 \beta$;
8 $z_0 \leftarrow t_0 + t_5$;
9 **retourner** $z = z_0 + z_1 w$;

Algorithme 7.17 : Mise au carré dans $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2 - \gamma)$

Entrées : Un élément x de $\mathbb{F}_{p^{12}}$ tels que $x = x_0 + x_1 w$ avec $x_0, x_1 \in \mathbb{F}_{p^6}$.
Sorties : $z = x^{-1} \in \mathbb{F}_{p^{12}}$.

1 $t_0 \leftarrow x_0^2$;
2 $t_1 \leftarrow x_1^2$;
3 $t_2 \leftarrow \beta t_1$;
4 $t_3 \leftarrow t_0 - t_2$;
5 $t_4 \leftarrow t_3^{-1}$;
6 $y_0 \leftarrow x_0 t_3$;
7 $y_1 \leftarrow -x_1 t_3$;
8 **retourner** $z = z_0 + z_1 w$;

Algorithme 7.18 : Inversion dans $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^6}[w]/(w^2 - \gamma)$

B Appendice 2

Cette annexe décrit les différentes équations d'addition et de doublement de points, ainsi que les lignes et tangentes selon le système de coordonnées.

Soit E une courbe elliptique sur F_q avec $q = p^d$ et $p \geq 5$ premier.

Coordonnées affines Nous avons déjà vu les équations d'addition et de doublement de points en coordonnées affines, ce sont les équations 2.26 et 2.27.

L'équation de la ligne passant par T et P évaluée au point Q est :

$$l_{T,P}(Q) = y_Q - y_T - \frac{y_P - y_T}{x_P - x_T}(x_Q - x_T). \tag{7.1}$$

L'équation de la tangente en T évaluée au point Q est :

$$l_{T,T}(Q) = y_Q - y_T - \frac{3x_T^2 + 1}{2y_T}(x_Q - x_T). \tag{7.2}$$

Coordonnées projectives Pour rappel, le point $T = (X_T : Y_T : Z_T)$ avec $Z_T \neq 0$ en coordonnées projectives correspond au point $(X_T/Z_T, Y_T/Z_T)$ en coordonnées affines. Les équations d'addition et de doublement de point en coordonnées projectives ont déjà été établies (cf. équations 2.45 et 2.46 page 35).

L'équation de la ligne passant par T et P évaluée au point Q est :

$$l_{T,P}(Q) = \frac{(Z_T Y_Q - Z_Q Y_T)(Z_T X_P - Z_P X_T) - (Z_T Y_P - Z_P Y_T)(Z_T X_Q - Z_Q X_T)}{Z_Q Z_T (Z_T X_P - Z_P X_T)}. \tag{7.3}$$

L'équation de la tangente en T évaluée au point Q est :

$$l_{T,T}(Q) = \frac{2Y_T(Z_T Y_Q - Y_T Z_Q Z_T) - (3X_T^2 + aZ_T^2)(X_Q Z_T - X_T Z_Q)}{2Y_T Z_Q Z_T^2}. \tag{7.4}$$

Coordonnées mixtes affines-projectives Les coordonnées projectives ont l'avantage de supprimer l'inversion lors de l'addition et du doublement de points. Il va être intéressant d'écrire en coordonnées projectives seulement le point qui va subir ces opérations d'ajout et de doublement.

On va donc avoir une addition entre un point P représenté en coordonnées affines et l'autre (le point T) en coordonnées projectives, ce nouveau point sera en coordonnées projectives. Il est calculé via les formules suivantes :

$$\begin{aligned} A &= y_P Z_T - Y_T \\ B &= x_P Z_T - X_T \\ &\text{Ainsi} \\ X_{T+P} &= Z_T A^2 - B^2(X_T + x_P Z_T) \\ Y_{T+P} &= B\left(A(X_T - X_{T+P} Z_T) - Y_T B\right) \\ Z_{T+P} &= Z_T B^2. \end{aligned} \tag{7.5}$$

Le doublement de T revient à un doublement classique en coordonnées projectives (cf. équation 2.46).

L'équation de la ligne passant par T et P évaluée au point Q est :

$$l_{T,P}(Q) = \frac{(y_Q Z_T - Y_T)(x_P Z_T - X_T) - (y_P Z_T - Y_T)(x_Q Z_T - X_T)}{Z_T(x_P Z_T - X_T)}. \tag{7.6}$$

L'équation de la tangente en T évaluée au point Q est :

$$l_{T,T}(Q) = \frac{2Y_T Z_T^2 y_Q - 2Z_T Y_T^2 - (3X_T^2 + aZ_T^2)(x_Q Z_T - X_T)}{2Y_T Z_T^2}. \tag{7.7}$$

Coordonnées jacobiennes Dans ce système de coordonnées, les points $T = (X_T : Y_T : Z_T)$ avec $Z_T \neq 0$ en coordonnées jacobiennes correspondent au point $({}^{X_T}/_{Z_T^2}, {}^{Y_T}/_{Z_T^3})$ en coordonnées affines. De même, les équations d'addition et de doublement de points en coordonnées jacobiennes ont déjà été établies (cf. équations 2.49 et 2.50).

L'équation de la ligne passant par T et P évaluée au point Q est :

$$l_{T,P}(Q) = \frac{Z_{T+Q}(Z_T^3 Y_Q - Z_Q^3 Y_T) - Z_T Z_Q (Z_T^3 Y_P - Z_P^3 Y_T)(Z_T^2 X_Q - Z_Q^2 X_T)}{Z_Q^3 Z_T^3 Z_{T+Q}}, \tag{7.8}$$

avec

$$Z_{T+Q} = Z_P Z_T (Z_T^2 X_P - Z_P^2 X_T).$$

L'équation de la tangente en T évaluée au point Q est :

$$l_{T,T}(Q) = \frac{2Y_T(Z_T^2 Y_Q - Z_Q^3 Y_T) - Z_Q(3X_T^2 + aZ_T^4)(X_Q Z_T^2 - X_T Z_Q^2)}{2Y_T Z_Q^3 Z_T^3}. \tag{7.9}$$

Coordonnées mixtes affines-jacobiennes De même que pour les coordonnées projectives, il est possible de représenter les points avec un système mixte.

L'addition entre un point P représenté en coordonnées affines et l'autre (le point T) en coordonnées jacobiennes, donne un point en coordonnées jacobiennes. Il est calculé via les formules suivantes :

$$\begin{aligned} A &= y_P Z_T^3 - Y_T \\ B &= x_P Z_T^2 - X_T \\ &\text{Ainsi} \\ X_{T+P} &= A^2 - B^2(X_T + x_P Z_T^2) \\ Y_{T+P} &= B^2\left(A(X_T - X_{T+P} Z_T^2) - BY_T\right) \\ Z_{T+P} &= Z_T B. \end{aligned} \tag{7.10}$$

Le doublement de T revient à un doublement classique en coordonnées jacobiennes (cf. équation 2.50).

L'équation de la ligne passant par T et P évaluée au point Q est :

$$l_{T,P}(Q) = \frac{(y_Q Z_T^3 - Y_T)(x_P Z_T^2 - X_T) - (y_P Z_T^3 - Y_T)(x_Q Z_T^2 - X_T)}{Z_T^2(x_P Z_T^2 - X_T)}. \tag{7.11}$$

L'équation de la tangente en T évaluée au point Q est :

$$l_{T,T}(Q) = \frac{2Y_T Z_T^3 y_Q - 2Y_T^2 - (3X_T^2 + aZ_T^4)(x_Q Z_T^2 - X_T)}{2Y_T Z_T^3}. \tag{7.12}$$

C Appendice 3

L'annexe C décrit les équations combinées d'addition/ligne et de doublement/tangente en fonction du système de coordonnées.

Coordonnées projectives La combinaison du calcul de $T+P$ avec celui de la ligne $l_{T,P}(Q)$ s'écrit :

$$\begin{aligned} A &= Y_T Z_P - Y_P Z_T \\ B &= X_T Z_P - X_P Z_T \\ C &= A^2 Z_T Z_P - B^3 - 2B^2 X_P Z_T \\ X_{T+P} &= BC \\ Y_{T+P} &= A(B^2 X_T Z_P - C) - Y_T Z_P B^3 \\ Z_{T+P} &= B^3 Z_T Z_P \\ l_{T,P}(Q) &= \frac{B(Z_T Y_Q - Z_Q Y_T) - A(Z_T X_Q - Z_Q X_T)}{Z_Q Z_T B}. \end{aligned} \tag{7.13}$$

Le doublement de T se fait simultanément avec $l_{T,T}(Q)$ comme suit :

$$\begin{aligned} D &= 3X_T^2 + aZ_T^2 \\ F &= Y_T Z_T \\ G &= F Y_T X_T \\ H &= D^2 - 8G \\ X_{[2]T} &= 2FH \\ Y_{[2]T} &= D(4G - H) - 8(Y_T F)^2 \\ Z_{[2]T} &= 8F^3 \\ l_{T,T}(Q) &= \frac{2F(Z_T Y_Q - Y_T Z_Q) - D(X_Q Z_T - X_T Z_Q)}{2F Z_Q Z_T}. \end{aligned} \tag{7.14}$$

Coordonnées affines-projectives On fait le même travail avec la représentation mixte affines-projectives. Le calcul de $T+P$ et de la ligne $l_{T,P}(Q)$ est :

$$\begin{aligned} A &= y_P Z_T - Y_T \\ B &= x_P Z_T - X_T \\ X_{T+P} &= Z_T A^2 - B(X_T + x_P Z_T) \\ Y_{T+P} &= B(A(X_T - X_{T+P} Z_T) - Y_T B) \\ Z_{T+P} &= Z_T B^2 \\ l_{T,P}(Q) &= \frac{B(Z_T y_Q - Y_T) - A(Z_T x_Q - X_T)}{Z_T B}. \end{aligned} \tag{7.15}$$

Doublement/tangente

$$\begin{aligned} D &= 3X_T^2 + aZ_T^2 \\ F &= Y_T Z_T \\ G &= F Y_T X_T \\ H &= D^2 - 8G \\ X_{[2]T} &= 2FH \\ Y_{[2]T} &= D(4G - H) - 8(Y_T F)^2 \\ Z_{[2]T} &= 8F^3 \\ l_{T,T}(Q) &= \frac{2F(Z_T y_Q - Y_T) - D(x_Q Z_T - X_T)}{2F Z_T}. \end{aligned} \tag{7.16}$$

Coordonnées jacobiennes En coordonnées jacobiennes cela donne :
Addition/ligne

$$\begin{aligned} A &= Y_T Z_P^2 \\ B &= X_P Z_T^2 \\ C &= Y_T Z_P^3 \\ D &= Y_P Z_T^3 \\ F &= B - A \\ G &= D - C \\ X_{T+P} &= -F^2 - 2AF^2 + G^2 \\ Y_{T+P} &= -CF^3 + G(AF^2 - X_{T+P}) \\ Z_{T+P} &= Z_T Z_P F \\ l_{T,P}(Q) &= \frac{Z_P F(Z_T^3 Y_Q - Z_Q^3 Y_T) - Z_Q G(X_Q Z_T^2 - X_T Z_Q^2)}{Z_Q^3 Z_T^2 Z_{T+P}}. \end{aligned} \tag{7.17}$$

Doublement/tangente

$$\begin{aligned} H &= 4X_T Y_T^2 \\ I &= 3X_T^2 + aZ_T^4 \\ X_{[2]T} &= -2H + I^2 \\ Y_{[2]T} &= -8Y_T^2 + I(H - X_{[2]T}) \\ Z_{[2]T} &= 2Y_T Z_T \\ l_{T,T}(Q) &= \frac{2Y_T Z_T^2 Y_Q - 2Y_T^2 Z_Q^3 - Z_Q I(X_Q Z_T^2 - X_T Z_Q^2)}{2Y_T Z_Q^3 Z_T^2}. \end{aligned} \tag{7.18}$$

Coordonnées affines-jacobiennes Enfin, nous faisons de même pour le système de coordonnées mixtes affines-jacobiennes.
Addition/ligne

$$\begin{aligned} A &= y_P Z_T^3 - Y_T \\ B &= x_P Z_T^2 - X_T \\ X_{T+P} &= A^2 - B^2(X_T + x_P Z_T^2) \\ Y_{T+P} &= B^2(A(X_T - X_{T+P} Z_T^2) - BY_T) \\ Z_{T+P} &= Z_T B \\ l_{T,P}(Q) &= \frac{B(Z_T^3 y_Q - Y_T) - A(Z_T^2 x_Q - X_T)}{Z_T^2 B}. \end{aligned} \tag{7.19}$$

Doublement/tangente

$$\begin{aligned} H &= 4X_T Y_T^2 \\ I &= 3X_T^2 + aZ_T^4 \\ X_{[2]T} &= -2H + I^2 \\ Y_{[2]T} &= -8Y_T^2 + I(H - X_{[2]T}) \\ Z_{[2]T} &= 2Y_T Z_T \\ l_{T,T}(Q) &= \frac{2Y_T Z_T^3 y_Q - 2Y_T^2 - I(x_Q Z_T^2 - X_T)}{Z_{[2]T} Z_T^2}. \end{aligned} \tag{7.20}$$

Liste des acronymes

ABE Atribute-Based Encription. 56, 84

AES Advanced Encryption Standard. 4, 9, 66, 69, 70, 72–76, 86, 89, 120–122, 139

ASIC Application-Specific Integrated Circuit. 60

CBDH Computational Bilinear Diffie–Hellman. 53, 55, 58

CDH Computational Diffie–Hellman. 52, 53, 56

CIHS Coarsely Integrated Hybrid Scanning. 21, 86

CIOS balayage de l'opérande grossièrement intégré – ou *Coarsely Integrated Operand Scanning* –. 21, 85, 86

CMOS Complementary Metal-Oxide-Semiconductor. 60, 96

CPA Correlation Power Analysis. 9, 10, 72, 76, 77, 84–86, 88, 100, 102, 103, 106, 107, 109–111, 117–119, 122, 125, 138

CRL Liste de révocation des certificats – ou *Certificate Revocation List* –. 6, 7

DBDH Decisional Bilinear Diffie–Hellman. 53

DDH Decisional Diffie–Hellman. 53

DES Data Encryption Standard. 4, 8, 62

DLOG logarithme discret – ou *Discrete logarithm* –. 5, 8, 26, 32, 58, 100

DPA Differential Power Analysis. 9, 64–66, 70, 72, 76, 77

DSA Digital Signature Algorithm. 26, 55

ECC cryptographie sur les courbes elliptiques – ou *Elliptic Curve Cryptography* –. 6, 9, 26, 62, 75, 76, 116, 139

ECDSA Elliptic Curve Digital Signature Algorithm. 26, 55, 127

FAPI1 Fixed Argument Pairing Inversion 1. 53

FIOS Finely Integrated Operand Scanning. 21, 86

FIPS Finely Integrated Product Scanning. 21, 86

FPGA Field-Programmable Gate Array. 60, 131

GDH Gap Diffie–Hellman. 53, 56

GSM Groupe Spécial Mobile. 61

IBE Identity-Based Encription. 7, 54, 56, 84, 124

MDBDH Modified Decisional Bilinear Diffie–Hellman. 53, 57

NFS Number Field Sieve. 58

NIST institut national des normes et de la technologie – ou *National Institute of Standards and Technology* –. 8, 26

PGP Pretty Good Privacy. 6

PKC Public Key Cryptography. 5

PKG Private Key Generator. 6, 54, 55, 57

PKI Public Key Infrastructure. 6, 7

RNS Residue number system – ou *Système modulaire de représentation* –. 140

RSA Rivest–Shamir–Adleman. 5, 9, 26, 55, 75, 76

SCARE Side Channel Analysis for Reverse Engineering. 61

SOS Separated Operand Scanning. 21, 86

SPA Simple Power Analysis. 9, 62, 64, 65, 76

UART Universal Asynchronous Receiver Transmitter. 99

XOR eXclusive OR. 69, 120, 139

Liste des symboles

Char Caractéristique. 15, 29, 31, 33, 34

Δ Discriminant d'une courbe elliptique. 27, 29–31

$\widehat{\mathfrak{D}}$ Distingueur. 88

div Diviseur. 41–46, 48, 50, 117, 119

$Div_{\overline{\mathbb{F}_q}}(E)$ Ensemble des diviseurs de E. 41

$\mathbb{E}$ Espérance mathématique. 94

$\mathbb{F}_q$ Le corps fini à q éléments. 15–17, 30–34, 37–42, 45, 58, 117, 119, 134, 137, 138

$\mathbb{F}_q^\star$ L'ensemble des éléments non nuls de $\mathbb{F}_q$. 16

Ψ Morphisme de Frobenius. 17, 30, 32, 33, 48, 51

1 Fonction indicatrice d'un ensemble. 73

Inv Inverse de Montgomery. 22

$\mathfrak{j}$ Le $j-$invariant. 27, 29–31, 33, 34

κ Coefficient de confusion. 89

$\overline{\mathbb{K}}$ Clôture algébrique de $\mathbb{K}$. 27, 31

$\mathbb{K}$ Corps commutatif. 15, 16, 27, 29–31

$\mathcal{K}$ Ensemble de toutes les clefs possibles. 7, 8, 70, 71, 74, 88, 89

$\mathfrak{L}$ Extension algébrique du corps $\mathbb{K}$. 16, 29, 31, 34, 50, 51

$\mathfrak{K}$ Sous-corps de $\mathbb{K}$. 16

k Degré de plongement. 37–39, 48, 50

$\left(\frac{a}{p}\right)$ Symbole de Legendre de a modulo p. 21, 32

μ Racines de l'unité. 32, 43, 48, 50, 51, 136

$\mathfrak{n}$ Nombre de bits d'un entier. 18, 45, 111, 117, 119

$\mathfrak{N}$ Nombre de mots d'un entier. 18

$\mathcal{O}$ Dénote le caractère dominé d'une fonction par rapport à une autre. 40, 46

ω Entité dans une structure ABE. 56

ord Ordre d'une fonction en un point. 41

$\mathfrak{O}$ Ordre d'un élément. 17

pgcd Plus grand diviseur commun. 17

φ Fonction indicatrice d'Euler. 17, 49, 51

Φ_k Le $k-$ième polynôme cyclotomique. 38, 48, 49, XII

$\mathcal{P}_\infty$ Point à l'infini. 27, 29

$\mathbb{P}$ Probabilité de $\cdot$. 73, 74, 88, 120, 121, 134

Redc Réduction de Montgomery. 19, 20, 22, 87

$\mathfrak{R}$ Résidu de Montgomery. 19–22, 87

ε Retenue (*carry* en anglais). 18

ρ La $\rho-$valeur d'une courbe elliptique. 37

$\boldsymbol{s}$ Clef secrète. 7, 8, 55

Ξ Fonction de sélection (DPA). 65, 70, 71

supp Support. 41–43

$\mathfrak{T}$ Taux de succès. 88

$\mathfrak{t}$ Translation. 45

$\mathcal{U}$ Univers des attributs. 56, 57

$\mathfrak{W}$ Taille de l'architecture de la machine (en bits). 18, 20

$\overline{\mathfrak{W}}$ Égal à $2^{\mathfrak{W}}$. 18–22, 86, 87

$\mathbb{Z}$ Ensemble des entiers relatifs. 14, 19

$\mathbb{Z}/N\mathbb{Z}$ Ensemble des entiers modulo N. 19