Implémentation légère et sécurisée pour la cryptographie sur Courbes Elliptiques pour l'Internet des Objets

Antoine Loiseau

N°d'ordre NNT : 2019LYSEM028

THESE de DOCTORAT DE L'UNIVERSITE DE LYON
opérée au sein de
l'Ecole des Mines de Saint-Etienne

Ecole Doctorale N° 488
Sciences, Ingénierie, Santé

Spécialité de doctorat : Microélectronique
Discipline : Cryptographie

Soutenue publiquement le 22/11/2019, par :
Antoine Loiseau

Implémentation légère et sécurisée pour la cryptographie sur Courbes Elliptiques pour l'Internet des Objets

---

Devant le jury composé de :

Clavier, Christophe Professeur de l'Université de Limoges	Président du jury
Quisquater, Jean-Jacques Professeur émérite de l'Université Catholique de Louvain	Rapporteur
Goubin, Louis Professeur de l'Université de Versailles Saint-Quentin	Rapporteur
Vitse, Vanessa Maître de Conférences de l'Université Grenoble Aples	Examinatrice
Fournier, Jacques J.A. Chef de laboratoire, CEA Leti	Directeur de thèse
Tria, Assia Chargé de recherche, CEA Leti	Co-encadrante

Spécialités doctorales	Responsables :	Spécialités doctorales	Responsables
SCIENCES ET GENIE DES MATERIAUX	K. Wolski Directeur de recherche	MATHEMATIQUES APPLIQUEES	O. Roustant, Maître-assistant
MECANIQUE ET INGENIERIE	S. Drapier, professeur	INFORMATIQUE	O. Boissier, Professeur
GENIE DES PROCEDES	F. Gruy, Maître de recherche	SCIENCES DES IMAGES ET DES FORMES	JC. Pinoli, Professeur
SCIENCES DE LA TERRE	B. Guy, Directeur de recherche	GENIE INDUSTRIEL	N. Absi, Maitre de recherche
SCIENCES ET GENIE DE L'ENVIRONNEMENT	D. Graillot, Directeur de recherche	MICROELECTRONIQUE	Ph. Lalevée, Professeur

EMSE : Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

ABSI	Nabil	MR	Génie industriel	CMP
AUGUSTO	Vincent	CR	Image, Vision, Signal	CIS
AVRIL	Stéphane	PR2	Mécanique et ingénierie	CIS
BADEL	Pierre	MA(MDC)	Mécanique et ingénierie	CIS
BALBO	Flavien	PR2	Informatique	FAYOL
BASSEREAU	Jean-François	PR	Sciences et génie des matériaux	SMS
BATTON-HUBERT	Mireille	PR2	Sciences et génie de l'environnement	FAYOL
BEIGBEDER	Michel	MA(MDC)	Informatique	FAYOL
BLAYAC	Sylvain	MA(MDC)	Microélectronique	CMP
BOISSIER	Olivier	PR1	Informatique	FAYOL
BONNEFOY	Olivier	PR	Génie des Procédés	SPIN
BORBELY	Andras	MR(DR2)	Sciences et génie des matériaux	SMS
BOUCHER	Xavier	PR2	Génie Industriel	FAYOL
BRODHAG	Christian	DR	Sciences et génie de l'environnement	FAYOL
BRUCHON	Julien	MA(MDC)	Mécanique et ingénierie	SMS
CAMEIRAO	Ana	MA(MDC)	Génie des Procédés	SPIN
CHRISTIEN	Frédéric	PR	Science et génie des matériaux	SMS
DAUZERE-PERES	Stéphane	PR1	Génie Industriel	CMP
DEBAYLE	Johan	MR	Sciences des Images et des Formes	SPIN
DEGEORGE	Jean-Michel	MA(MDC)	Génie industriel	Fayol
DELAFOSSE	David	PR0	Sciences et génie des matériaux	SMS
DELORME	Xavier	MA(MDC)	Génie industriel	FAYOL
DESRAYAUD	Christophe	PR1	Mécanique et ingénierie	SMS
DJENIZIAN	Thierry	PR	Science et génie des matériaux	CMP
BERGER-DOUCE	Sandrine	PR1	Sciences de gestion	FAYOL
DRAPIER	Sylvain	PR1	Mécanique et ingénierie	SMS
DUTERTRE	Jean-Max	MA(MDC)		CMP
EL MRABET	Nadia	MA(MDC)		CMP
FAUCHEU	Jenny	MA(MDC)	Sciences et génie des matériaux	SMS
FAVERGEON	Loïc	CR	Génie des Procédés	SPIN
FEILLET	Dominique	PR1	Génie Industriel	CMP
FOREST	Valérie	MA(MDC)	Génie des Procédés	CIS
FRACZKIEWICZ	Anna	DR	Sciences et génie des matériaux	SMS
GARCIA	Daniel	MR(DR2)	Sciences de la Terre	SPIN
GAVET	Yann	MA(MDC)	Sciences des Images et des Formes	SPIN
GERINGER	Jean	MA(MDC)	Sciences et génie des matériaux	CIS
GOEURIOT	Dominique	DR	Sciences et génie des matériaux	SMS
GONDRAN	Natacha	MA(MDC)	Sciences et génie de l'environnement	FAYOL
GONZALEZ FELIU	Jesus	MA(MDC)	Sciences économiques	FAYOL
GRAILLOT	Didier	DR	Sciences et génie de l'environnement	SPIN
GROSSEAU	Philippe	DR	Génie des Procédés	SPIN
GRUY	Frédéric	PR1	Génie des Procédés	SPIN
HAN	Woo-Suck	MR	Mécanique et ingénierie	SMS
HERRI	Jean Michel	PR1	Génie des Procédés	SPIN
KERMOUCHE	Guillaume	PR2	Mécanique et Ingénierie	SMS
KLOCKER	Helmut	DR	Sciences et génie des matériaux	SMS
LAFOREST	Valérie	MR(DR2)	Sciences et génie de l'environnement	FAYOL
LERICHE	Rodolphe	CR	Mécanique et ingénierie	FAYOL
MALLIARAS	Georges	PR1	Microélectronique	CMP
MOLIMARD	Jérôme	PR2	Mécanique et ingénierie	CIS
MOUTTE	Jacques	CR	Génie des Procédés	SPIN
NAVARRO	Laurent	CR		CIS
NEUBERT	Gilles			FAYOL
NIKOLOVSKI	Jean-Pierre	Ingénieur de recherche	Mécanique et ingénierie	CMP
NORTIER	Patrice	PR1	Génie des Procédés	SPIN
O CONNOR	Rodney Philip	MA(MDC)	Microélectronique	CMP
PICARD	Gauthier	MA(MDC)	Informatique	FAYOL
PINOLI	Jean Charles	PR0	Sciences des Images et des Formes	SPIN
POURCHEZ	Jérémy	MR	Génie des Procédés	CIS
ROUSSY	Agnès	MA(MDC)	Microélectronique	CMP
ROUSTANT	Olivier	MA(MDC)	Mathématiques appliquées	FAYOL
SANAUR	Sébastien	MA(MDC)	Microélectronique	CMP
SERRIS	Eric	IRD		FAYOL
STOLARZ	Jacques	CR	Sciences et génie des matériaux	SMS
TRIA	Assia	Ingénieur de recherche	Microélectronique	CMP
VALDIVIESO	François	PR2	Sciences et génie des matériaux	SMS
VIRICELLE	Jean Paul	DR	Génie des Procédés	SPIN
WOLSKI	Krzystof	DR	Sciences et génie des matériaux	SMS
XIE	Xiaolan	PR0	Génie industriel	CIS
YUGMA	Gallian	CR	Génie industriel	CMP

Mise à jour : 30/08/2018

À Bernard

Remerciements

Je tiens d'abord à remercier Jacques Fournier pour m'avoir donné l'opportunité de mener ces recherches. Je le remercie pour avoir dirigée cette thèse tout en me permettant de développer mes propres axes de recherches. Il aura fallu deux ans avant de pouvoir se lancer dans cette aventure, ces deux années ont été ponctuées de faux départs et je remercie Jacques Fournier pour avoir persévéré à lancer ce projet.

Je remercie aussi Assia Tria pour le co-encadrement de cette thèse et pour son aide précieuse lors de la rédaction du manuscrit.

Je tiens à remercier les différents membres du jury, Louis Goubin et Jean-Jacques Quisquater pour le temps qu'ils ont consacré à la relecture du manuscrit, Christophe Clavier et Vanessa Vitse pour avoir accepté de faire parti de ce jury.

Je souhaite remercier tous les membres des laboratoires LSOSP et LSAS du CEA pour leur accueil. Ces deux équipent forment un environnement idéal pour mener à bien une thèse. L'effervescence en leur sein permet l'émergence d'idées et les personnes formant ces laboratoires permettent de donner vie à ces mêmes idées.

Cette thèse n'aurait pas été la même sans les personnes qui m'ont formés. C'est pour cela que je suis reconnaissant à Sylvain Duquesne pour l'intérêt qu'il aura suscité en moi pour la cryptographie, Michel Douguet pour avoir été le tuteur de fin stage et Jean-Pierre Enguent pour m'avoir donner l'opportunité d'une première expérience professionnelle.

Je remercie mes parents et ma sœur pour leur soutient indéfectible. Merci à eux pour ces longs repas où j'essayais de leur expliquer du mieux que je pouvais ce qu'était une courbe elliptique. Merci aussi pour m'avoir soutenu tout au long de mes études et m'avoir fait confiance dans la poursuite de mes études en mathématiques.

Je tiens à remercier feu mon grand-père, Bernard, pour le goût de la science qu'il m'aura transmit. Ces nuits d'observations des étoiles ont été précieuses et m'auront montré qu'un simple point lumineux dans la nuit peut renfermer une complicité insoupçonnée. Les courbes elliptiques sont en ce sens les étoiles des mathématiques, simples et complexes à la fois.

Merci à Anne-Charlotte pour son soutien et son aide incroyable qu'elle m'a apportée pour cette thèse et qui malgré le stress grandissant de la fin thèse aura su m'apaiser et m'encourager. Son soutien sans faille aura été une pierre angulaire pour mener à bien cette thèse.

Table des matières

Liste des Figures

Liste des Tables

Liste des Algorithmes

Chapitre 1

Introduction

Toute connaissance dégénère en probabilité.

David Hume, *Traité de la nature humaine*

Sommaire

1.1 L'Internet des Objets

L'habitat connecté, les villes intelligentes ou les véhicules connectés sont des exemples de réseaux de capteurs et d'appareils interconnectés. Le nombre croissant de ces appareils a fait émerger le concept d'« Internet des Objets » (IoT). Définir avec précision cette expression est une tâche difficile au regard de l'hétérogénéité des appareils connectés considérés. L'IoT peut être défini comme un système de systèmes où chacun d'eux est composé de nœuds connectés dans le but d'enregistrer, opérer et transférer de l'information à un autre nœud ou à un ordinateur, offrant ainsi une intelligence et des services. Les services offerts par de tels réseaux couvrent une grande partie des besoins d'un utilisateur classique. Il est impossible de faire une liste exhaustive des applications de l'IoT, tant ces objets sont omniprésents dans nos vies.

Si nous considérons individuellement chacun de ces systèmes, leur complexité intrinsèque ne fait qu'amplifier la difficulté de les intégrer dans un écosystème complet d'IoT [Sta14]. Ce n'est que récemment que les chercheurs se sont penchés sur ces problèmes liés à l'IoT [Sta14]. L'un des principaux enjeux de ces nouveaux réseaux est leur sécurité en termes de respect de la vie privée [FTPN14], de la gestion des données, de la disponibilité des services et des enjeux industriels. Ce problème de sécurité a déjà été identifié comme étant la clef de l'adoption de ces réseaux et de leur déploiement [SM14, Rub14]. La majeure partie des risques a été identifiée dans la littérature, on peut citer parmi eux [Sta14, FTPN14, iot15] :

- L'accès non autorisé aux données propres à l'utilisateur.
- L'ajout de points d'entrée qui peuvent servir à attaquer le réseau.
- La mise en péril de la sûreté publique.
- La mise à disposition de données personnelles telles que les données de localisation ou les données médicales.
- La perte de confiance des utilisateurs.

Les principaux freins au déploiement de l'IoT viennent de l'hétérogénéité de ces systèmes de systèmes et la sécurité associée. Les problèmes de sécurité en termes d'intégrité, de confidentialité et d'authenticité nécessitent des outils particuliers. L'un de ces outils est la cryptographie et les systèmes IoT embarquent généralement une ou plusieurs briques de cryptographiques permettant de protéger une information.

1.2 Introduction à la cryptographie

Historiquement la cryptographie était réservée pour des applications militaires. L'avènement de la carte à puce et des applications bancaires ainsi que le développement de l'informatique domestique ont permis une démocratisation de son usage.

L'idée de vouloir protéger une information en la rendant inintelligible existe depuis l'Antiquité. Les méthodes pour cacher une information n'ont pas cessé d'évoluer au fur et à mesure des âges. Dans cette introduction, nous proposons un portrait historique de ces méthodes de protection de l'Antiquité à nos jours. Nous ferons référence à ces méthodes comme étant des techniques de chiffrement. De ce portrait, nous pourrons alors décrire les différents enjeux relatifs à cette science et, enfin, explicitement définir le cadre dans lequel se place cette thèse et quels en seront les principaux enjeux.

1.2.1 Cryptologie

Étymologiquement, le mot cryptologie vient du grec *crypto* (caché) et de *logie* (science). Le domaine de la cryptologie se décompose en deux grandes branches d'études :

- **La Cryptographie** est la science relative à l'écriture des messages cachés ou méthodes de chiffrement.
- **La Cryptanalyse** est la science permettant de retrouver le message chiffré sans connaissances préalable, on parlera dans ce cas de « décryptage »

Les premières traces de méthodes de chiffrement remontent à l'Antiquité. Le premier témoignage historique [Kah96] de techniques de chiffrement date du XVI^e siècle avant J-C. La méthode fut employée par un potier qui voulut protéger sa recette. Il la cacha en omettant certaines consonnes et en modifiant l'orthographe des mots.

Par la suite, les enjeux de conquêtes militaires ont amené l'emploi de techniques de chiffrement afin de pouvoir communiquer en toute sécurité entre les différentes armées. Ce fut le cas dans l'armée de César au I^{er} siècle avant J-C avec le code éponyme qui fut le premier système de chiffrement à substitution mono-alphabétique. De cet exemple simple, nous pouvons introduire les notions fondamentales relatives à la cryptologie.

Par méthode de chiffrement nous entendons une suite de règles fixes permettant de cacher une information. Par exemple, dans le cas du code de César, le décalage des lettres de l'alphabet est la méthode de chiffrement utilisée, *e.g.* un décalage de 4 lettres transformant toutes les lettres A du message en E, le B en F et ainsi de suite. Le résultat obtenu par l'application d'une méthode de chiffrement est appelé chiffré et correspond à l'information sous forme cachée. Nous pouvons par exemple appliquer cette règle ou méthode de chiffrement au message suivant :

Message :	ATTAQUEZ	PAR	LE	FLANC	DROIT
Chiffré :	EXXEUYIV	TEV	PI	JPERG	HVSMX

Le déchiffrement, opération inverse du chiffrement, consiste à retrouver l'information d'origine à partir du chiffré et de la clef associée. En suivant notre exemple, l'opération de déchiffrement consiste à effectuer un décalage dans le sens inverse. Ainsi, dans le cas du code César nous voyons que le nombre de lettres du décalage, ici 4, est essentiel dans le bon déroulement de nos opérations de chiffrement et déchiffrement. Le paramètre permettant d'effectuer correctement un chiffrement ou déchiffrement est appelé clef. La connaissance de cette clef permet de retrouver le message caché à partir de son chiffré moyennant la connaissance la méthode de chiffrement utilisée. Dans le cas du code de César, le faible espace de clefs possibles, *i.e.* le nombre de clefs possibles, rend la méthode peu résistante à une recherche exhaustive parmi toutes les possibilités de clef. En soi, la sécurité d'un tel chiffrement ne tient pas tant dans sa robustesse intrinsèque que dans l'ignorance de la méthode de chiffrement qu'en a un attaquant.

Nous pouvons formaliser mathématiquement le code de César. En représentant chaque lettre de l'alphabet par un élément de $\mathbb{Z}/26\mathbb{Z}$, A correspond à 0, le B à 1 et ainsi de suite, et en choisissant un nombre k de $\mathbb{Z}/26\mathbb{Z}$ qui correspond au nombre de décalages à effectuer dans l'alphabet. Nous pouvons appliquer la transformation linéaire $E : m \to m + k$ dans $\mathbb{Z}/26\mathbb{Z}$ sur la correspondance numérique de chaque

lettre du message. De même, la transformation linéaire $E^{-1} : c \to c - k$ dans $\mathbb{Z}/26\mathbb{Z}$ correspond à l'opération de déchiffrement et s'applique au chiffré c de la même manière que la fonction E s'applique au message. Cette méthode de chiffrement est très peu sécurisée, car il existe seulement 25 décalages possibles. Ainsi, un attaquant peut tester chaque clef une à une jusqu'à retrouver la bonne clef.

En 1586, Blaise de Vigenère publie son *Traité des chiffres ou secrètes manières d'escrire*, [dV18], qui présente une technique de chiffrement de substitution polyalphabétique. La force de la technique réside dans sa simplicité de chiffrement et déchiffrement. Il a fallu attendre trois siècles avant que la cryptanalyse ne rende le chiffrement de Vigenère désuet. La stratégie de Vigenère est d'utiliser comme clef non pas un simple décalage dans l'alphabet, mais une suite de décalages. Ainsi, la clef ne va pas juste être un nombre de décalages à effectuer, mais un mot entier dont la correspondance numérique dans $\mathbb{Z}/26\mathbb{Z}$ donnera les décalages successifs à effectuer lettre après lettre. La clef est alors répétée tout on long du message. Ainsi, la méthode de Vigenère appliqué à notre exemple donne :

Message :	ATTAQUEZ	PAR	LE	FLANC	DROIT
Clef :	CESARCES	ARC	ES	ARCES	ARCES
Chiffré :	CXLAHWIJ	PRT	PW	FCCRU	DIQML

Avec le même formalisme mathématique que pour le code de César, nous pouvons construire deux transformations de $\mathbb{Z}/26\mathbb{Z}$ correspondant à la méthode de chiffrement et déchiffrement :

$$\begin{array}{rccccc} E : & m_i & \to & m_i & + & k_i \\ E^{-1} : & c_i & \to & c_i & - & k_i \end{array}$$

Où m_i correspond à la i-ème lettre du message m ou du chiffré c, et k_i à la i-ème lettre de la clef répétée sur tout le message.

La cryptographie se formalise au fil de l'Histoire, notamment en 1883 avec *La cryptographie militaire* d'Auguste Kerckhoffs [Ker83], qui pose six principes fondateurs de la cryptographie :

1. *Le système doit être matériellement, sinon mathématiquement indéchiffrable.*
2. *Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi.*
3. *La clef doit pouvoir en être communiquée et retenue sans le secours de notes écrites, et être changée ou modifiée au gré des correspondants.*
4. *Il faut qu'il soit applicable à la correspondance télégraphique.*
5. *Il faut qu'il soit portatif, et que son maniement ou son fonctionnement n'exige pas le concours de plusieurs personnes.*
6. *Enfin, il est nécessaire, vu les circonstances qui en commandent l'application, que le système soit d'un usage facile, ne demandant ni tension d'esprit, ni la connaissance d'une longue série de règles à observer.*

Ces grands principes restent pour la plupart valables encore aujourd'hui. Cependant, la littérature dans le domaine de la cryptographie fait souvent référence au principe numéro 2 que nous pourrions reformuler comme suit :

La sécurité d'un système cryptographique ne doit pas reposer sur le secret de la méthode de chiffrement.

La cryptanalyse des méthodes de chiffrement devient un enjeu crucial au cours du XX^e siècle. Les deux guerres mondiales ont montré l'importance stratégique des systèmes de cryptographie. L'exemple le plus parlant fut la résolution de la cryptanalyse de la machine de chiffrement allemande, Enigma, au travers des travaux de Marian Rejewski et par suite d'Alan Turing, avec l'aide des renseignements polonais, français et anglais. L'accès à une copie de la machine Engima, associé à des erreurs d'utilisation afférentes aux allemands ont permis la cryptanalyse de ce système de chiffrement. Ceci a permis, entre autres, de renverser la suprématie des sous-marins allemands lors de la bataille de l'Atlantique.

1.2.2 Cryptographie moderne

L'ère moderne de la cryptographie a été initiée par Claude E. Shannon, après la 2^{nd} Guerre Mondiale, via la publication de deux articles fondateurs, [Sha49b, Sha49a]. Dans ces articles, Shannon développe différents grands principes conduisant à la formalisation des fondements de la cryptographie. Il définit la notion de secret (*secrecy*) [Sha49b].

Par secret, Shannon entend un ensemble de transformations d'un espace (celui des messages possibles) dans un second espace (celui des chiffrés possibles). Ainsi, chaque transformation de l'espace correspond à un chiffrement pour une clef donnée.

Shannon suppose aussi que chaque transformation est non singulière, *i.e.* inversible. Il définit aussi la probabilité *a priori* d'une clef et donc d'une transformation. De même, chaque message possible a, *a priori*, une probabilité. Alors, un attaquant qui intercepte un chiffré peut calculer *a posteriori* les probabilités associées correspondant aux messages et clefs possibles. Le calcul de ces probabilités *a posteriori* est le problème relatif à la cryptanalyse d'un chiffrement. Il démontrera ainsi qu'une méthode de chiffrement invulnérable ne peut être qu'un système de chiffrement dont l'espace de clefs est de même cardinal que l'espace des messages, dans ce cas Shannon parle de secret parfait. Le chiffrement de Vernam [Kah96] ou masque jetable est un secret parfait selon la définition de Shannon si la clef est aussi longue que le message, la clef est totalement aléatoire et la clef n'est utilisée qu'une seule fois.

Simmons [Sim81, Sim85] étend la notion de secret de Shannon en définissant l'authenticité (*authenticity*). La théorie de l'authenticité de Simmons est très proche de celle de la théorie du secret de Shannon. Ces deux théories diffèrent sur les possibilités d'attaques d'un attaquant. L'attaquant dans la théorie de Shannon est complètement passif alors que selon Simmons, il a la possibilité de construire de faux chiffrés (attaque par imitation) ou en intercepter et les modifier (attaque par substitution). Ainsi, par analogie à la théorie du secret, Simmons définit les probabilités d'un attaquant de former un faux chiffré et de modifier un chiffré. L'étude de ces probabilités et leurs calculs est le centre de la théorie de l'authenticité de Simmons.

Nous pouvons développer les notions de secret de Shannon et d'authenticité de Simmons en trois grandes propriétés :

Définition 1.1 (Confidentialité). *La confidentialité est l'assurance de la protection contre des accès non autorisés à l'information.*

Définition 1.2 (Intégrité). *L'intégrité est l'assurance qu'une information n'a pas été modifiée.*

Définition 1.3 (Authenticité). *L'authenticité est l'assurance de l'origine de l'information.*

Ainsi, la notion de confidentialité découle naturellement des propriétés de l'attaquant de Shannon qui cherche seulement à accéder à l'information protégée par le secret. Tandis que les notions d'intégrité et d'authenticité sont tirées des deux propriétés d'interférence qu'a l'attaquant dans la théorie de l'authenticité de Simmons. Plus précisément, le concept d'intégrité suit la possibilité de l'attaquant de Simmons d'effectuer des attaques par substitution. Tandis que le concept d'authenticité est plus précis que celui de Simmons et découle de la capacité qu'a l'attaquant de construire de faux chiffrés. Nous pouvons donc associer une probabilité de succès pour chacune des attaques :

- $\mathbb{P}_C$: la probabilité qu'un attaquant de Shannon a de réussir à résoudre la cryptanalyse d'un chiffrement. Cette probabilité est relative à la confidentialité.
- $\mathbb{P}_I$: la probabilité qu'un attaquant de Simmons a de pouvoir forger un faux chiffré. Cette probabilité est relative à l'intégrité.
- $\mathbb{P}_A$: la probabilité qu'un attaquant de Simmons a de modifier un chiffré. Cette probabilité est relative à l'authenticité.

Ainsi, si une méthode de chiffrement permet d'avoir l'une de ces probabilités inférieures à un certain ϵ, lorsque ϵ sera proche de zéro, nous dirons que cette méthode de chiffrement garantit la confidentialité, l'intégrité et/ou l'authenticité. En pratique, nous prendrons $\epsilon = 1/2^{128}$ comme expliqué dans la section 1.2.3. Le choix de cet ϵ est grandement influencé par la puissance de calcul offerte par les ordinateurs actuels. Ainsi, plus la puissance de calcul des ordinateurs augmente plus cet ϵ doit diminuer.

Pour garantir ces grandes propriétés, il existe deux familles de chiffrement : la cryptographie symétrique et la cryptographie asymétrique.

Cryptographie symétrique

La cryptographie dite symétrique ou à clef privée est ce qui se rapproche le plus du développement historique de la cryptographie. Deux entités, Alice et Bob, voulant échanger de façon sécurisée de l'information vont devoir connaître un secret commun qui fera office de clef. Ce secret est alors utilisé afin de chiffrer et déchiffrer l'information qui sera échangée entre Alice et Bob. Nous pouvons synthétiser l'idée de cryptographie symétrique de la manière suivante :

- Alice et Bob possèdent, tous deux, un exemplaire de la clef permettant d'ouvrir un coffre-fort. C'est le secret commun qu'ils partagent.
- Si Alice souhaite envoyer un message à Bob, elle dépose le message qu'elle veut protéger dans le coffre-fort et elle le verrouille grâce à la clef du coffre-fort qu'elle possède. L'algorithme cryptographique joue le rôle de coffre-fort et le chiffrement du message celui d'action de verrouillage du coffre.
- Si Bob souhaite retrouver le message protégé qu'il a reçu d'Alice, il utilise son exemplaire de la clef du coffre-fort afin de déverrouiller celui-ci et, ainsi, récupérer le message. Toujours par analogie, l'action de déverrouiller le coffre-fort représente l'action de déchiffrement que l'on effectue en cryptographie.

De tout temps l'humain a essayé de construire des coffres-forts virtuels offrant la même sécurité qu'un coffre-fort réel. Par exemple, dans le cas du code de César,

la méthode de substitution mono-alphabétique fait office de coffre-fort tandis que le nombre de décalages effectué dans l'alphabet représente la clef du coffre et doit rester secrète, tout comme la clef d'un coffre-fort ne doit pas se trouver en libre accès.

Les différents algorithmes de chiffrement symétrique sont généralement classés en deux catégories :

- **Chiffrement par bloc** : ce type d'algorithme va chiffrer un message m préalablement découpé en blocs de taille w définie. Ainsi, pour une clef k donnée la méthode de chiffrement sera appliquée à chaque bloc. Nous avons alors un ensemble de blocs chiffrés qui représente le chiffré c de m. L'algorithme AES [RD01] est actuellement la méthode de chiffrement standardisée la plus utilisée. À un algorithme de chiffrement symétrique on lui associe généralement un mode de chiffrement qui va désigner la manière dont le chiffrement de chaque bloc du message est effectué. Le mode classique consiste à chiffrer de façon séquentielle chaque bloc les uns après les autres, ce mode est généralement désigné sous l'acronyme ECB [BR19]. Cependant, ce type de mode de chiffrement a des faiblesses. En effet, pour deux blocs identiques le chiffré sera lui aussi identique, de l'information peut ainsi fuir d'un tel mode de chiffrement. Une méthode pour éviter ce genre de fuite d'information est l'utilisation d'un mode de chiffrement qui va exploiter le bloc précédemment chiffré pour chiffrer le bloc courant, ainsi même si deux blocs du message sont identiques leur chiffré respectif sera différent. Le mode de chiffrement CBC [BR19] respectant les propriétés énoncées juste avant permet de pallier à ce genre de fuites.
- **Chiffrement par flot** : l'idée de ce type de chiffrement est de pouvoir chiffrer l'information au fur et à mesure, lors d'une transmission, sans attendre d'avoir un bloc d'information. Ce type de chiffrement se rapproche de celui de Vernam combinant chaque bit du message avec celui de la clef en utilisant l'opérateur logique Xor (ou exclusif), dans le cas du chiffrement de Vernam la clef est aussi longue que le message à chiffrer. Dans le cas du chiffrement par flot, les bits de la clef sont quand eux xorés à un bit aléatoire issue d'un générateur pseudo-aléatoire. Ainsi, les différents chiffrements par flot spécifient plusieurs méthodes pour générer ce bit de manière pseudo-aléatoire. Généralement, ce générateur est construit par une association de LFSR, FCSR, NLFSR, T-fonction... L'algorithme Trivium [DCP05, DC06] est un exemple de chiffrement par flot.

En 1977, est publié le premier algorithme de chiffrement symétrique [DES77], résultant d'un appel d'offres lancé par le NBS (*National Bureau of Standards*). Une telle publication permet d'ancrer le principe de Kerchoffs dans la cryptographie moderne. De plus, cette démarche permet au grand public d'utiliser facilement cette technique de chiffrement. Le *Data Encryption Standard* (DES) devient alors omniprésent dans tous les systèmes publics utilisant la cryptographie. Cet algorithme opère par blocs de 64 bits. La clef de chiffrement d'un longueure de 56 bits rend obselète le DES dès la fin des années 90. En effet, la montée en puissance des ordinateurs permet une recherche exaustive de la clef de l'algorithme de DES ce qui amène à une réévalutaion de la sécurité du DES par la construction du Triple-DES (TDES), en 2001 le *National Institue of Standards and Technology* (NIST) standardise un nouvel algorithme de chiffrement symétrique l'*Advanced Encryption Standard* [RD01], AES. L'AES est un chiffrement par blocs de taille fixe 128 bits et utilisant des clefs

de 128, 192 ou 256 bits. La construction de l'AES et les différents choix de tailles de clefs augmentent grandement la sécurité en offrant plusieurs niveaux de protection.

Cependant, ce type de cryptographie possède une importante contrainte liée à la distribution des clefs. En effet, pour qu'Alice ou Bob puissent s'échanger des messages chiffrés, ils doivent au préalable partager un secret qui fera office de clef. Cette étape d'échange de clef est un problème nécessitant de considérer différemment la cryptographie. Une telle problématique peut être résolue par l'utilisation de la cryptographie asymétrique.

Cryptographie asymétrique

La cryptographie asymétrique ou cryptographie à clef publique fut introduite par Diffie et Hellman en 1976 au travers de deux publications [DH76a, DH76b], qui définissent les principaux concepts de la cryptographie asymétrique. Cette nouvelle forme de cryptographie fut créée afin de répondre à certains problèmes que la cryptographie symétrique n'arrivait pas à résoudre. Les principaux enjeux étant de trouver une solution au problème d'échange de clefs et à l'authentification entre deux entités, Alice et Bob. Pour ce faire, chacun possède une paire de clef dite publique et privée liées entre elles, la clef publique se construit à partir de la clef privée. La clef publique peut être diffusée à toute personne voulant communiquer tandis que la clef privée doit rester secrète. Si Bob souhaite envoyer un message de manière sécurisé à Alice, il lui suffit de le chiffrer avec la clef publique d'Alice. Alice pourra le déchiffrer en utilisant sa clef privée.

Mathématiquement, la cryptographie asymétrique utilise des fonctions à sens unique avec trappe dont leur construction repose sur des problèmes mathématiques difficiles à résoudre. Ces fonctions E sont simples à calculer, mais leur inverse E^{-1} ne peut pas être calculé dans un temps raisonnable. Ainsi, la sécurité de cette cryptographie repose sur la difficulté de calculer E^{-1}. Diffie et Hellman proposent alors, [DH76a, DH76b], l'utilisation d'un problème de mathématique connu : le problème du logarithme discret (DLP) dans un corps fini de grande caractéristique. Le problème vient du fait qu'étant donnés $X, Y \in \mathbb{F}_p$ avec p un nombre premier, il est alors difficile de trouver a tel que $Y = X^a[p]$. Diffie et Hellman proposent une fonction à sens unique qui va calculer Y en fonction de X et de a, a étant la clef privée et Y la clef publique associée.

Il existe de nombreux problèmes difficiles pouvant être exploités à des fins cryptographiques. Le plus célèbre est sans doute la primitive cryptographique inventée par Rivest, Shamir et Adleman (RSA) [RSA78]. Cette primitive repose sur la difficulté de factoriser les grands nombres. Il existe d'autres alternatives au DLP classique. En effet, sur le groupe d'une courbe elliptique que nous détaillerons au chapitre 2, le problème du logarithme discret devient plus complexe pour une taille de clef équivalente. Ce problème du logarithme discret sur courbes elliptiques ou ECDLP a été introduit indépendamment par Miller [Mil86] et Koblitz [Kob87].

Ainsi, la cryptanalyse en cryptographie asymétrique va reposer sur la résolution d'un problème mathématique par exemple factoriser un grand nombre représentant le produit de deux nombres premiers inconnus dans le cas du chiffrement RSA. La résolution de ces problèmes mathématiques est difficile lorsque l'on considère seulement les algorithmes classiques et la puissance de calcul offerte par les ordinateurs classiques. Dès les années 70 émerge l'idée d'un calculateur quantique, en 1994 Peter Shor va démontrer la possibilité de factoriser efficacement un nombre à l'aide d'un

ordinateur quantique [Sho94]. L'algorithme de Shor permet également de résoudre le problème du logarithme discret.

Ainsi, sous l'hypothèse de l'existence d'un calculateur quantique assez puissant, un système de chiffrement tel que le RSA ne serait plus sécurisé, il en va de même pour la cryptographie basée sur les courbes elliptiques et l'ECDLP.

En 2019, IBM annonce le premier calculateur quantique de 20 qubits physiques [noa19]. Cependant, les capacités de cet ordinateur sont encore loin de pouvoir résoudre facilement les problèmes difficiles utilisés en cryptographie asymétrique que l'on évalue à 50 qubits logiques. D'autant que, les experts estiment qu'il faudrait entre 1000 et 100000 qubits physiques pour avoir un qubits logique [CTV17].

Avec la menace d'un calculateur quantique des problèmes mathématiques alternatifs sont utilisés afin de pouvoir leur résister. Une nouvelle cryptographie asymétrique appelée cryptographie post-quantique a ainsi vu le jour. Les principaux problèmes considérés pour la construction de primitives cryptographiques post-quantique, sont :

- le problème de vecteurs courts dans un réseau euclidien [BDK+18, DKL+18]
- le décodage de codes linéaires [ABB+17]
- le problème de parcours dans les isogénies des courbes elliptiques supersingulières [DFJP14]
- l'inversion de polynômes multivariés [CFMR+17]

En 2016 que le NIST lance un nouvel appel d'offres [CJL+16] pour l'établissement d'un nouveau standard de cryptographie asymétrique résistant aux calculateurs quantiques.

Pour résoudre le problème de l'échange de clefs, on s'appuie sur la cryptographie asymétrique permettant à Alice et Bob d'échanger une clef secrète symétrique. Dans l'absolue on pourrait traiter tous les échanges de cette manière. Cependant, ces primitives cryptographiques étant très coûteuses en ressources et en temps de calcul leur utilisation sera limité et restreinte à l'échange de la clef secrète qui pourra par suite être utilisée par la cryptographie symétrique [Res18].

Cette cryptographie permet également de répondre au problème de l'authentification. De la même manière que dans le problème de l'échange de clef, il est difficile d'authentifier une entité de façon sécurisée sans avoir échangé au préalable une clef avec celle-ci. Cependant, pour échanger une clef secrète, il faut au préalable s'assurer de l'identité de l'entité avec laquelle l'échange est effectué. Ainsi, la cryptographie à clef publique offre la possibilité d'authentifier une entité sans avoir à échanger en amont une clef secrète via un mécanisme de signature. Si Alice veut alors authentifier Bob, elle va transmettre un nombre aléatoire que Bob signera avec sa clef privée. Alice pourra ensuite vérifier la signature en utilisant la clef publique de Bob.

Malgré tout ce système reste sensible à différentes attaques telles que les attaques de type *Man-in-the-Middle* (MitM) qui consistent à s'interposer entre Alice et Bob et d'usurper l'identité de chacun afin d'espionner les échanges effectuer entre ces deux entités. Cette faiblesse vient de la difficulté pour Alice et Bob de s'authentifier. L'utilisation d'une infrastructure à clef publique (PKI) est généralement utilisée pour pallier à ce genre d'attaque. Cette infrastructure repose sur une autorité de certification (CA), aussi appelée tiers de confiance. Cette autorité va certifier et gérer un annuaire de clefs publiques qui chacune sera associée à une entité grâce à l'utilisation d'un certificat. Ce dernier est constitué de l'identité de l'entité et de la

clef publique associée qui servira par suite à l'authentifier. Le certificat est alors signé par le CA. Ainsi, il est possible de s'assurer de l'identité d'une entité en vérifiant les informations contenues dans le certificat de cette dernière et la signature du CA. Si l'on reprend la méthode d'authentification vue précédemment avec l'intégration d'une autorité de certification, les grands principes de l'authentification de Bob par Alice peuvent être découpés de la manière suivante :

1. Bob a préalablement fait certifier sa clef publique auprès du CA.
2. Alice demande à BOB le certificat de sa clef publique.
3. Alice vérifie le certificat de Bob.
4. Alice envoie un challenge à Bob qu'il doit signer.
5. Bob signe le challenge reçu et transmet la signature à Alice.
6. Alice vérifie la signature reçue en utilisant la clef publique de Bob.

L'étude des PKI est un domaine de recherche à part entière et ne fait pas l'objet de cette thèse. Pour une étude complète des PKIs, nous renvoyons au livre de Dumas *et al.* [DLR15].

1.2.3 Cryptanalyse

La cryptanalyse consiste à se mettre à la place d'un attaquant afin de caractériser la sécurité d'un système cryptographique en retrouvant le message ou la clef qui a permis de le déchiffrer. Le but d'un attaquant peut être de différentes sortes : il peut souhaiter retrouver la clef secrète ou privée d'une entité, « décrypter » un message.

Quel que soit le système cryptographique considéré, il existe un nombre fini n de clefs possibles, nous noterons cet ensemble $\mathcal{K}$. Ainsi, un attaquant pourra effectuer une attaque exhaustive sur l'ensemble $\mathcal{K}$ en essayant toutes les possibilités de clefs. Intuitivement si le cardinal de $\mathcal{K}$, noté $|\mathcal{K}|$, est assez grand alors une telle attaque n'est pas envisageable, car le temps de calcul nécessaire pour retrouver la bonne clef serait supérieur à la durée de vie de l'information protégée. Dans l'exemple du code de César, $\mathcal{K}$ ne possède que 25 clefs qui correspondent à tous les décalages possibles dans l'alphabet latin. Dans le cas de l'AES-128 il y a 2^{128} clefs possibles, ce nombre est bien au-delà des capacités informatiques disponibles actuellement pour permettre une recherche exhaustive de la clef.

Il existe de nombreuses attaques plus efficaces permettant de mettre à mal la sécurité d'un algorithme de cryptographie considéré. Celles-ci sont classifiées en deux grandes familles :

- **Attaques mathématiques** : le but de ces attaques est de passer outre la sécurité mathématique intrinsèque d'un algorithme de cryptographie. Une telle attaque, si elle existe, est redoutable car elle peut s'appliquer à n'importe quelle implémentation du dit système ciblé. Ainsi, une telle attaque peut rendre une méthode de chiffrement complètement inutilisable. Ces attaques sont rares et les algorithmes cryptographiques sont construits de telles manières à résister à ce type d'attaque.
- **Attaques physiques** : cet ensemble d'attaques ne cherche pas des faiblesses directes dans la conception des algorithmes de cryptographie, mais exploite

les faiblesses d'implémentations de ces algorithmes qui laissent fuir de l'information sensible via des canaux auxiliaires (consommation du courant, rayonnement électromagnétique) ou en injectant une faute lors du calcul (photo-émission, impulsion électromagnétique).

Les attaques mathématiques sont redoutables, mais leurs influences sont quantifiables. La robustesse des algorithmes de cryptographie est mesurée par leur niveau de sécurité qui est exprimé en bits. Ce dernier donne le nombre d'opérations nécessaires pour mener l'attaque la plus performante relative à un algorithme, il est exprimé en puissance de deux dont l'exposant correspond à ce niveau de sécurité. Ainsi, toute attaque mathématique va diminuer le niveau de sécurité. De fait, tant que ce niveau de sécurité ne décroit pas en dessous d'un seuil critique, l'algorithme est toujours considéré comme sécurisé. À ce niveau de sécurité nous pouvons associer une taille de clefs nécessaires pour assurer la sécurité de ce dernier. A ce propos, il existe de nombreuses recommandations dictées par différents organismes travaillant sur la cryptographie. La table 1.1 présente les différentes recommandations de trois organismes : l'ANSSI, le NIST et l'ECRYPT-CSA.

Niveau de sécurité	Algorithme symétrique	RSA	ECC
80	2TDEA	1024	160
112	3TDEA	2048	224
128	AES-128	3072	256
192	AES-192	7680	384
256	AES-256	15360	512

TABLE 1.1 – Recommandations sur la taille des clefs en fonction du niveau de sécurité. La ligne bleue correspond à la recommandation minimale de l'ANSSI [ans14], en orange celle du NIST [Bar16] et en vert celle de ECRYPT-CSA [ecr18].

En ce qui concerne la sécurité des implémentations cryptographique, il est difficile de quantifier les vulnérabilités. En effet, le succès d'une attaque dépend grandement de l'implémentation de l'algorithme et de l'environnement matériel sur lequel est exécuté le chiffrement. D'une implémentation à l'autre ou d'un environnement matériel à un autre, nous pouvons avoir de grandes disparités quant au succès d'une attaque physique.

Dans cette thèse, nous nous exposerons plus en détails sur ces attaques physiques que nous étudierons dans le cadre de la cryptographie basée sur les courbes elliptiques. Nous détaillerons ces attaques lors du chapitre 3.

L'attaque consiste à perturber l'exécution du calcul en injectant une faute, puis en étudiant les conséquences de celle-ci. La faute peut être injectée de différentes manières : via l'utilisation d'un laser ou par impulsion électromagnétique. Ces méthodes d'injection ne sont que deux exemples classiques, pour plus de détail, le livre de Marc Joye et Micheal Tunstall, *Fault Analysis in Cryptography* [JT17], est une bonne introduction aux différentes méthodes d'exploitation des fautes.

Il existe une autre famille d'attaques physiques : celles dites par observation. Le but de ces attaques est de mesurer une grandeur physique émanant des composants sur lequel le calcul cryptographique est réalisé. Ainsi, selon les opérations effectuées lors du calcul et les données manipulées au cours du temps, un attaquant peut inférer de l'information secrète telle qu'une clef privée. D'autres grandeurs physiques

peuvent être exploitées afin d'inférer de l'information telles que le temps de calcul [KJJ99], la consommation de courant [KJJ99] ou le rayonnement électromagnétique [Qui00].

Afin d'exploiter ces mesures un attaquant peut opérer une analyse statistique, selon l'attaque choisie, dans le but de lier les mesures obtenues à une information secrète telle que la clef privée utilisée. SPA [KJJ99], DPA [KJJ99], CPA [BCO04] proposent différentes méthodes statistiques pour retrouver le secret, [CRR03] propose une méthode analogue basée sur du profilage.

La majeure partie des algorithmes cryptographiques est vulnérable à ce type d'attaque. En effet, leur construction n'est pas pensée à l'origine pour y résister. Ainsi, pour se prémunir contre celles-ci de nombreuses recherches ont été menées dans ce sens, et il en résulte l'obligation d'ajouter des protections ou contremesures au sein de leur implémentation. Le choix des contremesures va dépendre de l'algorithme considéré, de l'attaque contre laquelle il est nécessaire de se protéger et des motivations de l'attaquant. Les contremesures que nous considérerons dans le cas de la cryptographie sur les courbes elliptiques, seront détaillées dans cette thèse.

1.3 Positionnement de la thèse

Les contraintes induites par le matériel sur lequel doit être déployée une implémentation cryptographique influent grandement sur les performances et la sécurité. Cependant, lors de la construction des standards cryptographiques basés sur les courbes elliptiques, les contraintes de sécurité ou matérielles ne sont que partiellement prises en compte. Ce constat s'explique à plusieurs niveaux.

En premier lieu, le standard le plus utilisé pour les ECC [Gal13] n'est plus aussi bien adapté contenu des avancées mathématiques, en terme d'optimisation et de sécurité, depuis sa création. En effet, les critères de sécurité ont évolué, au regard des résultats obtenus sur la résolution de l'ECDLP.

En second lieu, les contraintes en termes de sécurité physique qui sont apparues *a posteriori*. Le concept d'IoT et d'objets tout connectés émergeant pour lesquels les critères de sécurité n'ont pas été pris en compte.

En troisième lieu, les standards qui ne prennent en compte que l'aspect mathématiques de la cryptographie sur courbes elliptiques sans tenir compte des contraintes matérielles, or celles-ci affectent les performances et la sécurité, notamment physique en raison de l'adaptation de la cryptographie au matériel.

Ainsi, cette thèse a pour objectif de proposer une approche de construction de la cryptographie sur les courbes elliptiques en prenant en compte les contraintes matérielles dès leur construction. Pour autant, certaines approches décrites dans les standards seront reprises voire améliorées.

En 2011, Bernstein *et al.* [BDL+11a] initie cette démarche en proposant une nouvelle cryptographie sur les courbes elliptiques prenant en compte certaines contraintes de sécurité physique et d'optimisation. Cependant, cette approche n'envisage pas toutes les contraintes de sécurité physique tel que le problème induit par la propagation de la retenue mais aussi celles liées à l'IoT. Cette approche novatrice fait figure d'exemple.

Ainsi, nous proposons donc d'approfondir cette approche en considérant le maximum de contraintes dans l'environnement IoT tant au niveau sécurité physique que matérielles. Dans notre cas d'étude l'IoT de référence et notre microcontrôleur RISC

V choisi qui n'a pas été conçu spécifiquement pour les implémentations sur courbes elliptiques. Ceci nous permet de construire une primitive cryptographique extrêmement contrainte pour répondre aux critères de performances et de sécurité, et ainsi la rendre plus facilement adaptable à d'autres environnements IoT.

De plus, cette nouvelle cryptographie sur les courbes elliptiques résultant de cette étude devra être intégrable dans un environnement IoT déjà largement déployé. Ceci nous impose donc d'être totalement compatible avec les environnements déjà préexistants à ce jour. Ainsi, toute proposition de nouvelle cryptographie devra parfaitement s'intégrer dans les protocoles standards d'échange de clefs ou de signatures.

Nous devons définir le système IoT que nous considérerons dans la suite de ce manuscrit. Les problématiques de sécurité matériel sont différentes si l'on considère une carte à puce, un microcontrôleur ou une implémentation FPGA d'une primitive cryptographique. Nous devons alors définir les différentes catégories d'IoT existantes sur lesquelles une primitive cryptographique peut être exécutée.

- **Implémentations matérielles** : La carte à puce se distingue du microcontrôleur par sa taille et sa puissance de calcul du processeur intégré dans ces derniers. Dans la suite de ce manuscrit, seul les microcontrôleurs seront considérés.

  Une primitive cryptographique peut faire l'objet d'une implémentation matérielle dédiée afin d'augmenter l'efficacité de cette dernière en termes de temps de calcul et consommation de courant, on parlera dans ce cas d'accélérateur cryptographique ou coprocesseur. Une telle implémentation peut être intégrée directement au sein d'un microcontrôleur ou sur FPGA, dans les deux cas, les fuites d'informations seront plus difficiles à exploiter. Le temps d'exécution d'une implémentation matérielle est nécessairement moindre qu'une implémentation logicielle, un plus grand nombre de mesures physiques sont nécessaires ce qui implique une plus grande difficulté pour extraire l'information pour mener à bien une attaque par canaux auxiliaires. De même, la rapidité d'exécution des implémentations matérielles complexifie les attaques par fautes, aussi bien dans le domaine spatial que temporel. Par exemple l'injection d'une faute laser nécessite un positionnement du faisceau sur les parties sensibles du circuit, mais aussi une synchronisation précise entre le moment d'injection de la faute et le calcul.

- **Implémentations logicielles** : L'implémentation logicielle de la primitive cryptographique est exécutée sur le processeur générique intégré du microcontrôleur choisi.

  Les systèmes IoT multicœurs ne sont pas considérés dans cette étude, car leur complexité se rapproche plus de l'ordinateur que de l'IoT monocœur. D'une architecture de microcontrôleur à une autre, les mêmes difficultés relatives au temps d'exécution ou à la taille du chemin de données sont observées aussi bien sur une implémentation matérielle que logiciel. La complexité de telles architectures rend difficilement distinguable l'information sensible de l'information non-sensible (bruit).

  Dans le cas d'architectures complexes, une instruction Assembleur peut passer par plusieurs étages (*pipeline*) du processeur. Ainsi quand une instruction est disponible sur un étage, d'autres instructions peuvent arriver au même moment

sur les autres étages ce qui conduit à un ensemble d'informations offusqué intrinsèquement. Il s'en suit qu'un processeur disposant d'une architecture à plusieurs étages laissera fuir une information plus bruitée qu'un processeur sur un seul étage.

Dans le cas de notre étude, notre choix de plateforme se porte sur un microcontrôleur cadencé à 100 MHz embarquant un RISC V 32 bits, les primitives cryptographiques seront implémentées en logiciel et exécutées par le RISC V. Ce choix d'architecture est beaucoup étudié dans la recherche et est en cours d'adoption dans l'industrie (Google, Microchip, Nvidia, Qualcomm...) [1].

Une architecture RISC V dispose d'un jeu d'instructions réduit et offre plusieurs avantages :

- Il est libre de droit.
- Il est évolutif et son architecture personnalisable offre des jeux d'instructions additionnels qui permettent de concevoir une implémentation dédiée au cas d'usage considéré.
- Il permet de créer son propre jeu d'instructions via l'ajout d'instructions spécifiques qui ne seraient pas disponibles dans les différents jeux d'instructions proposés.

Le principal défaut de l'architecture RISC V dans le cas d'implémentations cryptographiques est l'absence de bit de contrôle de la retenue (*carry flag*). Ce choix est justifié et motivé par la fondation RISC V [WAD17] contenu du surplus de consommation induit par ce bit de contrôle. L'absence de gestion de la retenue peut conduire à une implémentation plus complexe et peu performante de ces primitives cryptographiques à clefs publiques. Ainsi, ce choix influencera considérablement les primitives cryptographiques que nous sélectionnerons.

1.4 Plan de la thèse

Ce manuscrit s'articule autour de huit grands chapitres qui auront pour but de décrire la construction de cette nouvelle cryptographie sur les courbes elliptiques. Ce premier chapitre nous a permis de contextualiser la thèse et de décrire ses enjeux.

Les deux chapitres suivants détaillerons le cadre mathématique de la cryptographie sur les courbes elliptiques (chapitre 2) ainsi que le concept d'attaques physiques (chapitre 3).

Le chapitre 4 exposera les différentes étapes de la génération d'un nouvel ensemble, de courbes elliptiques adaptées aux contraintes de l'IoT et l'intégration de celui-ci au sein des protocoles standards de signatures et d'échange de clefs. Nous conclurons le chapitre 4 par une comparaison des performances obtenues sur RISC V par rapport à l'état de l'art.

Dans le chapitre 5, nous évaluerons les forces et faiblesses de cette nouvelle cryptographie face aux attaques par canaux auxiliaires.

L'influence des techniques de profilage sur une implémentation de courbes elliptiques sera détaillée au chapitre 6, une application de ces techniques sera présentée sur la multiplication scalaire confirmant la possibilité d'extraire d'une clef secrète.

1. `https://riscv.org/`

1.5 Publications et communications

Ces travaux ont fait l'objet de plusieurs publications, communications et brevets :

- Brevet FR3071081, Antoine Loiseau et Jacques J. A. Fournier, *Méthode cryptographique sur courbes elliptiques binaires d'Edwards*, Septembre 2017.
- Brevet FR1904667, Antoine Loiseau, *Procédé et système de masquage pour la cryptographie*, Mai 2019.
- [LF18] Antoine Loiseau et Jacques J. A. Fournier, *Binary Edwards Curves for Intrinsically Secure ECC Implementations for the IoT*, Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 2 : SECRYPT, Porto, Portugal, Juillet 26-28, 2018.
- Antoine Loiseau, Maxime Lecomte et Jacques J. A. Fournier, *Template Attacks against ECC : practical implementation against Curve25519*, Soumission à HOST 2020, Août 2019.
- Séminaire SoSySec, Antoine Loiseau, *Binary Edwards Curves for Intrinsically Secure ECC Implementations for the IoT*, Septembre 2018.

Chapitre 2

Cryptographie sur Courbes Elliptiques

Aucune ligne de conduite ne saurait être déterminée par une règle, puisque toute ligne de conduite peut être considérée comme conforme à la règle

Ludwig Wittgenstein, *Tractatus logico-philosophicus*

Sommaire

Les courbes elliptiques, objets mathématiques complexes nécessitent une bonne compréhension de la géométrie algébrique : dans ce chapitre nous allons définir les différentes notions mathématiques sous-jacentes. Tous les concepts mathématiques propres aux courbes elliptiques sont décrits de façon exhaustive dans [CFA+06, Sil09]. Nous nous concentrerons sur les notions de base dont nous aurons besoin dans la suite de ce document.

Après quelques rappels sur les corps et leur arithmétique nous détaillerons, dans un second temps, la géométrie et l'arithmétique propre aux courbes elliptiques. Nous présenterons les principaux modèles alternatifs de courbes elliptiques et nous décrirons en détails sur modèle alternatif nommé Courbes Binaires d'Edwards (BEC). Nous conclurons ce chapitre par une présentation des différents protocoles cryptographiques de signatures et d'échange de clefs.

2.1 Prérequis mathématiques

Nous concentrerons nos rappels sur le concept de corps. Cette structure fondamentale d'algèbre constitue le socle sur lequel repose la construction des courbes elliptiques à des fins d'application cryptographique.

Parmi les objets fondamentaux d'algèbre, il existe deux structures sur lesquelles se reposent la construction des corps : le groupe et l'anneau. La première structure algébrique fondamentale est le couple formé par un ensemble G et une loi de composition interne $+$, aussi appelée groupe. Par souci de clarté, nous noterons la loi de composition $+$, mais celle-ci ne fait pas obligatoirement référence à l'addition sur les entiers comme loi de composition. Cette notation est purement symbolique et est destinée à ne pas multiplier les notations tout au long de ce manuscrit. Pour les mêmes raisons, nous noterons l'élément neutre d'un groupe, 0 ou 1. Ainsi, le couple $(G, +)$ est un groupe s'il vérifie un certain nombre de propriétés :

1. **Associativité** : $\forall a, b, c, \in G : (a + b) + c = a + (b + c)$.
2. **Élément neutre** : $\exists! 0 \in G : \forall a \in G, a + 0 = 0 + a = a$.
3. **Inverse** : $\forall a \in G, \exists! b \in G : a + b = b + a = 0$

Une dernière propriété importante peut être vérifiée par certains groupes aussi appelés groupe abéliens :

4. **Commutativité** : $\forall a, b \in G : a + b = b + a$

Nous pouvons citer comme exemple classique de groupe abélien : l'ensemble des entiers munis de l'addition usuelle. Un groupe n'est pas nécessairement abélien : l'ensemble de matrices carrées muni de la multiplication comme loi de composition forme un groupe non-abélien. Dans la suite de la thèse, nous travaillerons essentiellement avec des groupes abéliens.

La structure d'anneau se construit à partir d'un groupe abélien. À ce groupe nous ajoutons une seconde loi de composition interne qui sera notée $\times$. Le triplet $(G, +, \times)$ doit vérifier toutes les propriétés précédentes ainsi que :

5. **Associativité** : $\forall a, b, c \in G : (a \times b) \times c = a \times (b \times c)$
6. **Élément neutre** : $\exists! 1 \in G, 1 \neq 0 : \forall a \in G, a \times 1 = 1 \times a = a$
7. **Distributivité** : $\forall a, b, c \in G : a \times (b+c) = a \times b + a \times c, (b+c) \times a = b \times a + c \times a$

Comme précédemment certains anneaux sont dits commutatifs s'ils vérifient :

8. **Commutatif** : $\forall a, b \in G : a \times b = b \times a$

De la même manière, l'ensemble des entiers muni de l'addition et de la multiplication $(\mathbb{Z}, +, \times)$ forme un anneau commutatif et l'ensemble des matrices carrées muni de l'addition et de la multiplication de matrices $(\mathbb{M}_{(n,n)}, +, \times)$ un anneau non-commutatif.

Le corps est une structure algébrique qui est construite à partir d'un anneau commutatif et qui vérifie une dernière propriété :

9. **Inverse** : $\forall a \in G, \exists! b \in G : a \times b = b \times a = 1$

Pour compléter notre exemple : l'anneau $(\mathbb{Z}, +, \times)$ ne forme pas un corps car seul 1 admet un inverse multiplicatif dans $\mathbb{Z}$. Tandis que $(\mathbb{R}, +, \times)$ forme un corps.

Afin de mieux appréhender les courbes elliptiques nous détaillerons ci-après plusieurs notions importantes sur les corps.

Caractéristique d'un corps

La caractéristique d'un corps est celle de son anneau commutatif. Pour définir formellement la caractéristique d'un anneau, il nous faut alors définir la notion d'homomorphisme d'anneau.

Définition 2.1. *Soient $(R, +, \times)$ et $(R', \oplus, \otimes)$ deux anneaux. Un homomorphisme d'anneau est une application ϕ de R dans R' tel que pour tout $a, b, \in R$:*

- $\phi(a + b) = \phi(a) \oplus \phi(b)$
- $\phi(a \times b) = \phi(a) \otimes \phi(b)$
- $\phi(1) = 1$

Cette définition nous permet de construire une application homomorphique ψ entre $\mathbb{Z}$ et un anneau R. L'invariant ψ est essentiel en théorie des anneaux et se définit par :

$$\psi(n) = \begin{cases} 1 + ... + 1 & n \text{ fois si } n \geq 0 \\ -(1 + .. + 1) & n \text{ fois si } n < 0 \end{cases}$$

Le noyau de ψ en un idéal de $\mathbb{Z}$. Deux cas de figures sont alors possibles :

1. Si tous les multiples de 1 sont différents, alors le noyau $\ker(\psi) = \{\emptyset\}$.
2. Si certains multiples de 1 sont nuls, alors le noyau $\ker(\psi)$ est généré par un entier positif m.

Nous pouvons alors définir la caractéristique de R comme suit :

Définition 2.2 (Caractéristique). *Soient R un anneau et ψ l'homomorphisme défini ci-dessus. Le noyau de ψ est un idéal de $\mathbb{Z}$ engendré par un entier positif m. m est alors appelé caractéristique de R.*

De cette définition, on peut en déduire la caractéristique du corps des réels $\mathbb{R}$ comme étant nulle, ou celle des entiers modulo n, $\mathbb{Z}/n\mathbb{Z}$, comme étant égale à n.

De la définition d'un corps donné en préambule de ce chapitre, on peut en déduire qu'un corps est un domaine intégral. De plus, en considérant le quotient d'un corps $\mathbb{F}$ par le noyau de ψ, on observe que $\mathbb{F}$ contient un sous-corps isomorphique à $\mathbb{Z}/m\mathbb{Z}$, où m est la caractéristique de $\mathbb{F}$. Cela conduit au résultat suivant :

Proposition 2.3. *La caractéristique d'un corps est soit nulle, soit égale à un nombre premier p.*

Corps Finis

Dans la suite de ces rappels mathématiques sur les corps, nous concentrerons sur le cas particulier des corps finis. Dans la suite des notations nous désignerons par $\mathbb{F}$ un corps fini, sauf mention contraire.

Définition 2.4 (Corps Finis). *Un corps $\mathbb{F}$ est dit fini si son cardinal est fini.*

La proposition 2.3 permet d'affirmer qu'un corps fini $\mathbb{F}$ admet comme caractéristique un nombre premier, de ce résultat, on en peut déterminer le cardinal. En effet, pour tout corps fini $\mathbb{F}$ de caractéristique p, $\mathbb{Z}/p\mathbb{Z}$ est un sous-corps de $\mathbb{F}$. Ceci implique que $\mathbb{F}$ est un espace vectoriel sur $\mathbb{Z}/p\mathbb{Z}$ de dimension d, en particulier, son cardinal est égal à p^d.

Le théorème suivant est essentiel car il classifie tous les corps finis.

Théorème 2.5. *Pour tout nombre premier p et entier positif d, il existe un unique corps fini $\mathbb{F}_{p^d}$ défini à isomorphisme près.*

Il est essentiel de définir la notion d'ordre d'un corps $\mathbb{F}$ qui est défini par son cardinal, il s'en suit un résultat fondamental que nous exprimerons sous sa forme générale par :

Théorème 2.6. *Pour tout élément $a \in \mathbb{F}$:*

$$a^{|\mathbb{F}|-1} = 1$$

Ce résultat peut être vu comme une généralisation du Théorème d'Euler sur les anneaux $\mathbb{Z}/n\mathbb{Z}$ qui lui-même est une première généralisation du Petit Théorème de Fermat.

La notion que nous voudrions rappeler à présent est celle d'ordre multiplicatif d'un élément d'un corps fini qui découle du théorème 2.6.

Définition 2.7 (Ordre multiplicatif). *L'ordre multiplicatif d'un élément $a \neq 0, 1$ du corps fini $\mathbb{F}$ est le plus petit entier n tel que $a^n = 1$.*

Une manière similaire de définir cette notion est de considérer n comme étant l'ordre de a dans le groupe $(\mathbb{F}^*, \times)$, le théorème 2.6 implique alors l'existence de ce nombre n. De même, nous pouvons en déduire que l'ordre multiplicatif maximal d'un élément a est $|\mathbb{F}| - 1$. Les éléments de $\mathbb{F}$ dont l'ordre multiplicatif est maximal sont appelés générateurs du corps. Le théorème suivant implique donc l'existence de ces éléments générateurs.

Théorème 2.8. *Soit $\mathbb{F}$ un corps fini. Le groupe $\mathbb{F}^*$ est cyclique.*

Les générateurs sont aussi appelés racines primitives de $\mathbb{F}^*$. Le nombre de générateurs est défini par la fonction indicatrice ϕ d'Euler et est égale à $\phi(|\mathbb{F}| - 1)$. Pour rappel :

$$\phi(N) = |\{x | 1 \leq x \leq N, \text{pgcd}(x, N) = 1\}|$$

La fonction indicatrice d'Euler peut facilement se calculer à partir de la décomposition en facteur premier de N. Soit $N = \prod_{i=1}^{r} p_i^{\alpha_i}$ alors :

$$\phi(N) = \prod_{i=1}^{r} (p_i - 1) p_i^{\alpha_i - 1} = N \prod_{i=1}^{r} \left(1 - \frac{1}{p_i}\right)$$

Il s'en suit que si d divise $|\mathbb{F}| - 1$ alors il existe exactement $\phi(d)$ éléments d'ordre multiplicatif d.

Morphisme de Frobenius

Une application importante en théorie des corps est le morphisme de Frobenius Ψ.

Définition 2.9 (Morphisme de Frobenius). *Soit $\mathbb{F}_{p^d}$ un corps fini, alors on définit l'application Ψ comme suit :*

$$\Psi = \begin{cases} \mathbb{F}_{p^d} & \longrightarrow \mathbb{F}_{p^d} \\ a & \longmapsto a^p \end{cases}$$

On peut démontrer que Ψ est un morphisme en utilisant la formule du binôme de Newton.

Démonstration. Pour montrer que Ψ est un morphisme de corps, il faut montrer que pour tous $a, b \in \mathbb{F}_{p^d}$ les trois assertions suivantes sont vraies :

1. $\Psi(1) = 1$
2. $\Psi(ab) = \Psi(a)\Psi(b)$
3. $\Psi(a+b) = \Psi(a) + \Psi(b)$

Les assertions 1 et 2 sont triviales.

L'assertion 3 demande de prouver que $(a+b)^p = a^p + b^p$. En développant $(a+b)^p$ avec la formule du binôme nous obtenons que tous les termes autres que a^p et b^p ont un coefficient de la forme $\binom{p}{i}$ qui est divisible par p. Du fait que $\mathbb{F}_{p^d}$ a une caractéristique égale à p et que tous les coefficients binomiaux sont des multiples de p, ceux-ci s'annulent. Ainsi, nous avons bien $(a+b)^p = a^p + b^p$.

□

Cette démonstration conduit à un résultat important des corps finis : $(a+b)^p = a^p + b^p$. Cette formule permet d'avoir une arithmétique plus simple que celle des entiers, notamment lorsque $p = 2$ comme nous le verrons dans la section 2.2.

Extension de corps

L'utilisation de corps finis à grand cardinal est une condition *sine qua none* pour la cryptographie sur les courbes elliptiques. En effet, la sécurité de cette cryptographie va en partie dépendre de ce cardinal. Il existe alors deux manières immédiates d'augmenter la taille des corps finis. La première est d'augmenter la caractéristique du corps et travailler, ainsi, modulo un nombre premier p large. La seconde est d'augmenter la dimension d de $\mathbb{F}$ en tant que $\mathbb{Z}/p\mathbb{Z}$-espace vectoriel. Pour expliquer la manière de construire une extension de corps fini, nous pouvons partir de la définition générique d'extension de corps :

Définition 2.10 (Extensions de Corps). *Soient K et L deux corps. On dit que L est une extension de corps de K noté L/K, s'il existe un homomorphisme de K dans L.*

Nous pouvons construire une extension d'un corps fini en considérant le quotient de l'anneau des polynômes $\mathbb{F}[X]$ par l'idéal généré par un polynôme irréductible de $\mathbb{F}$. Ainsi, la construction de l'extension de corps $\mathbb{F}_{p^d}/\mathbb{F}_p$ revient à considérer un

polynôme irréductible $m_d(X)$ de degrés d et de travailler à isomorphisme près dans le quotient $\mathbb{F}_p[X]/(m_d(X))$. Ceci est équivalent à ajouter au corps $\mathbb{F}_p$ une racine α de $m_d(X)$. Il s'en suit un résultat fondamental :

Théorème 2.11. *Toutes extensions finies $\mathbb{F}_{p^d}/\mathbb{F}_p$ est une extension galoisienne et le groupe des automorphismes $Gal(\mathbb{F}_{p^d}/\mathbb{F}_p)$ est un groupe cyclique d'ordre d et généré par le morphisme de Frobenius Ψ_p.*

De ce résultat et du fait que $m_d(X)$ est irréductible nous pouvons en déduire que les conjugués de α sont les éléments distincts $\alpha, \alpha^p, \alpha^{p^2}, ..., \alpha^{p^{d-1}}$. Nous pouvons alors définir la trace d'un élément de $\mathbb{F}_{p^d}$ par :

Définition 2.12 (Trace). *La trace d'un élément a du corps fini $\mathbb{F}_{p^d}$ est définie par :*

$$Tr(a) = \sum_{i=1}^{d} a^{p^i}$$

Représentation des éléments

Il existe plusieurs façons de représenter un élément d'un corps fini. Nous en citerons deux :

- **Base polynomiale** : La représentation des éléments dans une base polynomiale est la plus naturelle mais aussi celle qui se rapproche le plus de la construction d'un corps fini. Si α est une racine de $m(X)$ alors $\{1, \alpha, ..., \alpha^{d-1}\}$ est une base de $\mathbb{F}_{p^d}$. L'arithmétique qui en découle est la même que l'arithmétique polynomiale. Les éléments de $\mathbb{F}_{p^d}$ peuvent être vus comme des polynômes modulo le polynôme irréductible $m_d(X)$.
- **Base normale** : La notion de base normale est moins immédiate mais dans certains cas elle permet d'accélérer certaines opérations arithmétiques comme la multiplication. On dit que $a \in \mathbb{F}_{p^d}$ est normal si tous les éléments $a, a^p, a^{p^2}, ..., a^{p^{d-1}}$ sont linéairement indépendants. De cet élément a on peut alors déduire une base dite normal $\{a, \Psi_p(a), ..., \Psi_p^{d-1}(a)\}$.

Dans la suite de nos travaux, nous considérerons uniquement la base polynomiale. En effet, on optimise généralement l'arithmétique de cette base en considérant des polynômes irréductibles creux lors de la construction de l'extension de corps. Comme nous le verrons par la suite, cf. section 4.3.1, nous pouvons améliorer cette représentation en choisissant judicieusement les polynômes irréductibles considérés.

2.2 Arithmétiques sur les corps finis de caractéristique 2

Dans cette thèse nous orienterons nos travaux sur les corps finis de caractéristiques 2. Les résultats précédents restent toujours valides dans ce cas particulier.

Nous décrirons la représentation polynômiale de ces corps d'un point de vue informatique. Nous détaillerons l'arithmétique des corps finis, nécessaire pour effectuer des calculs sur les courbes elliptiques. Dans le cas de la caractéristique 2, l'arithmétique est très différente de l'arithmétique en grande caractéristique. Cette différence provient de la représentation des éléments dans $\mathbb{F}_{2^d}$ et de l'arithmétique

simple de $\mathbb{F}_2$. Il en découle une arithmétique beaucoup plus simple à implémenter, car en caractéristique 2, les éléments sont non signés et nous n'avons pas à propager la retenue. Ce dernier point est essentiel dans d'une implémentation sur RISC V car comme nous l'avons vu, cf. section 1.1, l'architecture du RISC V ne possède pas de bit de contrôle de la retenue. Ainsi, la nécessité de programmer la gestion de la retenue peut rendre l'implémentation complexe et plus difficilement optimisable. Cela conduira à deux conséquences, le ralentissement significatif des performances et une sécurité plus faible contre les attaques physiques basées sur la propagation de la retenu, cf. section 3.2.

2.2.1 Représentation polynomiale des éléments de $\mathbb{F}_{2^d}$

Le corps de base de l'extension $\mathbb{F}_{2^d}$ est le corps $\mathbb{F}_2$ qui admet pour seuls éléments 0 et 1. De ce fait, il est plus aisé de représenter un élément de $\mathbb{F}_{2^d}$ sous forme polynomiale. Ainsi, les coefficients des monômes d'un élément $a \in \mathbb{F}_{2^d}$ sont soit 0, soit 1.

$$a = \sum_{i=0}^{d-1} a_i \alpha^i, \quad a_i \in \mathbb{F}_2$$

Nous pouvons alors synthétiser cette représentation en une suite de bits représentant les coefficients des monômes de a. Cette liste de bits comporte $d-1$ bits et est facilement intégrable dans une implémentation logicielle de l'arithmétique en caractéristique 2. Dans la suite de ce travail, nous considérerons un élément de $\mathbb{F}_{2^d}$ essentiellement sous la forme d'un vecteur de bits de longueur $d-1$. De plus, dans le cadre de l'IoT et comme vu à la section 1.1, nous optimiserons notre implémentation pour une architecture 32 bits. Il est alors possible de généraliser les algorithmes décrient pour une architecture de taille arbitraire w. Par conséquent, le vecteur de $d-1$ bits sera séparé en $n = \lceil \frac{d-1}{w} \rceil$ mots de w bits et tous les algorithmes de cette section traiterons ce vecteur comme une suite de mots de w bits.

2.2.2 Addition

Du fait de l'absence de la retenue, l'addition sur $\mathbb{F}_{2^d}$ est simplifiée. Pour additionner deux éléments a et b de $\mathbb{F}_{2^d}$, il suffit d'effectuer mot à mot un ou exclusif (Xor). L'algorithme 2.1 présente en pseudo-code l'addition dans $\mathbb{F}_{2^d}$.

Algorithme 2.1 Addition de deux éléments de $\mathbb{F}_{2^d}$

Entrées $a = a_{n-1}...a_1a_0, b = b_{n-1}...b_1b_0$
Sorties: $c = a + b$ dans le corps fini $\mathbb{F}_{2^d}$
1: **Pour** $i = 0$ **à** $n-1$ **Faire**
2: $\quad c_i = a_i \oplus b_i$
3: **Fin Pour**
4: **Retourner** $c = c_{n-1}...c_1c_0$

2.2.3 Multiplication

Multiplier deux éléments de $\mathbb{F}_{2^d}$ peut s'avérer plus complexe. En effet, la majeure partie des processeurs disposent seulement d'une instruction de multiplication dans

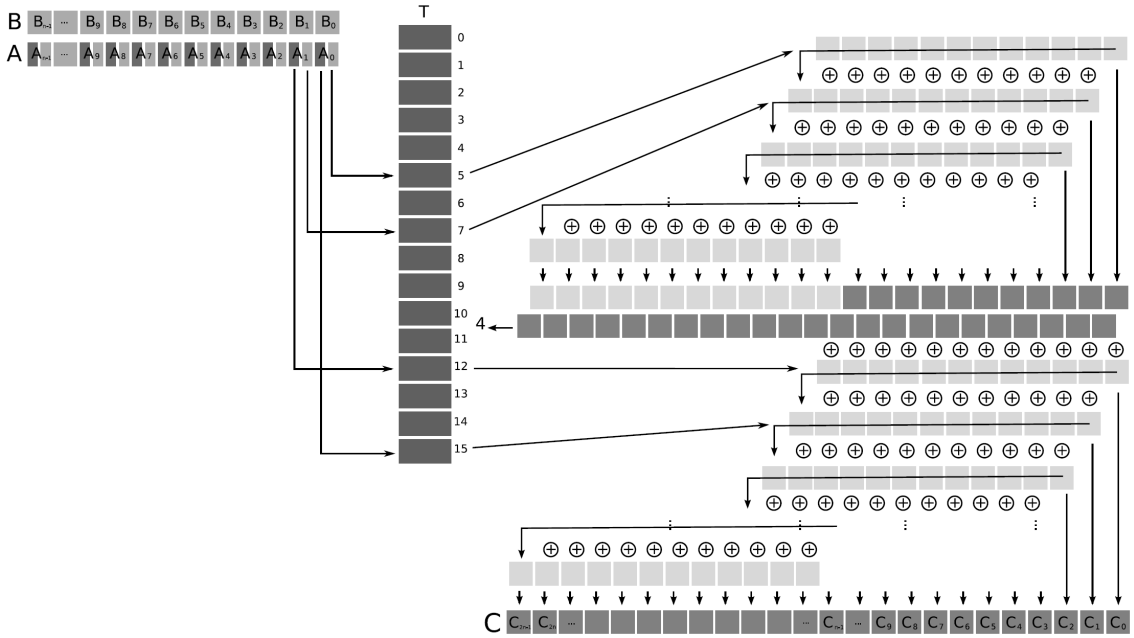


FIGURE 2.1 – Multiplication de López et Dahab pour les éléments de $\mathbb{F}_{2^n}$ pour $t = 4$ [ALH10].

les entiers et non dans $\mathbb{F}_{2^w}$. Ainsi, il faut réussir à multiplier deux éléments de $\mathbb{F}_{2^d}$ sans avoir recours à une instruction de multiplication. Dans [LD00], López et Dahab proposent une méthode de multiplication dans $\mathbb{F}_{2^d}$ efficace, l'algorithme se divise en deux parties.

La première partie consiste en un pré-calcul à partir d'un des opérandes : a. Pour ce faire, tous les résultats de la multiplication de cette dernière par tous les polynômes de $\mathbb{F}_{2^t}$ où t est un paramètre arbitraire, sont sauvegardés en mémoire. En général t est choisi petit afin de limiter la taille des pré-calculs et la taille des données à sauvegarder.

La seconde partie opère sur le second opérande b en appliquant une fenêtre glissante de longueur t bits. Pour chaque fenêtre, l'algorithme va accumuler la valeur dans les pré-calculs correspondant à la multiplication de a par le polynôme de longueur t de la fenêtre courante. Avant d'accumuler la fenêtre suivante, un décalage est à opérer en amont sur la valeur à accumuler. La longueur de ce décalage correspond au décalage de la fenêtre sur l'opérande b.

Aranha *et al.* [ALH10] proposent une amélioration de cet algorithme. Cette optimisation vise à réduire le nombre d'instructions de chargement et de sauvegarde qui sont coûteuses en termes de temps de calcul pour un microprocesseur. L'idée vient du fait que l'algorithme calcule en premier les bits de poids faibles et que les longueurs des données manipulées sont de $d - 1 + t$ bits. Ainsi, un vecteur de longueur $d - 1 + t$ bits est sauvegardé au sein du processeur afin d'accumuler le résultat. Lorsque le décalage de la fenêtre est égal à un multiple de la longueur w des registres, le registre de poids faible du vecteur de travail peut alors être retourné dans le vecteur de sortie. Une rotation du vecteur de travail au sein du processeur est alors opérée avant de continuer le calcul. La Figure 2.1 résume les différentes étapes de l'algorithme de multiplication de López et Dahab.

L'algorithme 2.2 présente en détails l'algorithme de multiplication de López-

Dahab dans $\mathbb{F}_{2^d}$ dans le cas où la taille de la fenêtre t est égale à 4 et où la taille des registres est de 32 bits. 8 fenêtres de 4 bits sont traitées avant d'effectuer une sauvegarde du mot de poids faible et la rotation du vecteur d'accumulation. Le vecteur $r = r_{n+1}r_n...r_0$ représente le vecteur de travail au sein du processeur ce qui implique que le processeur possède assez de registres pour contenir un vecteur de longueur $d - 1 + t$ bits.

L'algorithme 2.2 retourne un vecteur c de $2n$ registres. Pour obtenir un élément de $\mathbb{F}_{2^d}$ de telle sorte que le degré de c soit strictement inférieur à d, il faut alors réduire c par le module du corps fini.

L'architecture du RISC V étant modulaire et offrant la possibilité d'ajouter sa propre instruction, il pourrait être alors envisager d'ajouter une multiplication dans $\mathbb{F}_{2^{32}}$ donnant le résultat sur 64 bits. Cette instruction supplémentaire pourrait grandement améliorer les performances de l'opération de multiplication. En effet, nous pourrions alors envisager de changer l'algorithme 2.2 de López-Dahab par une approche de type Karatsuba-Offman [KO63] ce qui permettrait d'avoir une implémentation proche de l'arithmétique en large caractéristique. Au final, l'ajout d'une telle instruction pourrait nettement améliorer les performances d'une implémentation de la cryptographie sur les courbes elliptiques en caractéristique 2.

2.2.4 Élévation au carré

Nous pouvons directement utiliser l'algorithme de multiplication pour effectuer l'opération $a \times a$. Cependant, cet algorithme bien que très lent, il existe une solution alternative. Nous pouvons effectuer le calcul du carré en utilisant le fait que $(a+b)^2 = a^2 + b^2$ comme nous l'avons démontré dans la preuve relative à la définition 2.9 du Frobenius. Ainsi, pour tout élément a de $\mathbb{F}_{2^d}$:

$$a^2 = \left(\sum_{i=0}^{d-1} a_i \alpha^i\right)^2 = \sum_{i=0}^{d-1} a_i \alpha^{2i}$$

Il en découle que pour mettre au carré un élément a de $\mathbb{F}_{2^d}$ en représentation binaire, il suffit d'insérer un bit nul entre chaque bit de sa représentation binaire. Par exemple, pour $a = \alpha^3 + \alpha + 1$ qui a comme représentation binaire 1011, nous obtenons alors $a^2 = \alpha^6 + \alpha^2 + 1$ qui a pour représentation binaire 1000101.

L'implémentation d'un algorithme d'élévation au carré qui insère un bit nul entre chaque bit d'un vecteur offrira de meilleures performances que de l'algorithme de multiplication 2.2 de López-Dahab. Cependant, les processeurs de type RISC V ne possèdent pas d'instruction permettant de faire une telle opération. Nous pouvons pallier à cela en utilisant une table de pré-calculs comportant les carrés de tous les polynômes d'un degré t ou plus petit. De cette table, nous pourrons élever au carré chacun des éléments a de $\mathbb{F}_{2^d}$ en parcours a avec une fenêtre glissante de taille t et en recherchant dans la table de pré-calculs le résultat correspondant. L'algorithme 2.3 détaille les différentes étapes de l'élévation au carré. La table T est la table de pré-calcul des carrés. La longueur t est choisie en fonction de l'architecture sur laquelle est effectuée l'implémentation. Dans le cas de notre architecture 32 bits, nous choisissons de prendre une taille t égale à 8 afin d'opérer par octet ce qui conduit à une table de pré-calculs T de 256 valeurs de 16 bits.

De même que pour l'algorithme de multiplication 2.2, l'algorithme 2.3 d'élévation

Algorithme 2.2 Multplication de López-Dahab de deux éléments a et b dans $\mathbb{F}_{2^d}$

Entrées $a = a_n a_{n-1}...a_1 a_0, b = b_n b_{n-1}...b_1 b_0$
Sorties: $c = a \times b$ dans le corps fini $\mathbb{F}_{2^d}$

1: /* Première partie : pré-calculs de $a \times u_i$ où u_i correspond à tous les polynômes de 4 bits, les résultats sont sauvegardés dans T */
2: $T[u_0] \leftarrow 0$
3: $T[u_1] \leftarrow a$
4: $T[u_2] \leftarrow a << 1$
5: $T[u_3] \leftarrow a \oplus (a << 1)$
6: $T[u_4] \leftarrow a << 2$
7: $T[u_5] \leftarrow (a << 2) \oplus a$
8: $T[u_6] \leftarrow (a << 2) \oplus (a << 1)$
9: $T[u_7] \leftarrow (a << 2) \oplus (a << 1) \oplus a$
10: $T[u_8] \leftarrow (a << 3)$
11: $T[u_9] \leftarrow (a << 3) \oplus a$
12: $T[u_{10}] \leftarrow (a << 3) \oplus (a << 1)$
13: $T[u_{11}] \leftarrow (a << 3) \oplus (a << 1) \oplus a$
14: $T[u_{12}] \leftarrow (a << 3) \oplus (a << 2)$
15: $T[u_{13}] \leftarrow (a << 3) \oplus (a << 2) \oplus a$
16: $T[u_{14}] \leftarrow (a << 3) \oplus (a << 2) \oplus (a << 1)$
17: $T[u_{15}] \leftarrow (a << 3) \oplus (a << 2) \oplus (a << 1) \oplus a$
18: /* Seconde partie : accumulation du résultat par lecture de b avec une fenêtre glissante de taille 4 */
19: $r \leftarrow 0$
20: **Pour** $i = 0$ **à** $d - 1$ **par** 32 **Faire**
21: $\quad r \leftarrow r \oplus T[u_{i,i+3}]$
22: $\quad r \leftarrow r \oplus (T[u_{i+4,i+7}] << 4)$
23: $\quad r \leftarrow r \oplus (T[u_{i+8,i+11}] << 8)$
24: $\quad r \leftarrow r \oplus (T[u_{i+12,i+15}] << 12)$
25: $\quad r \leftarrow r \oplus (T[u_{i+16,i+19}] << 16)$
26: $\quad r \leftarrow r \oplus (T[u_{i+20,i+23}] << 20)$
27: $\quad r \leftarrow r \oplus (T[u_{i+24,i+27}] << 24)$
28: $\quad r \leftarrow r \oplus (T[u_{i+28,i+31}] << 28)$
29: $\quad c_{i>>5} \leftarrow r_0$
30: $\quad r_0 \leftarrow r_1, r_1 \leftarrow r_2, r_2 \leftarrow r_3, r_3 \leftarrow r_4, r_5 \leftarrow r_6, r_6 \leftarrow r_7, r_7 \leftarrow r_8$
31: $\quad r_8 \leftarrow r_9, r_9 \leftarrow r_{10}, r_{10} \leftarrow r_{11}, r_{11} \leftarrow r_{12}, r_{12} \leftarrow r_{13}, r_{13} \leftarrow r_{14}, r_{14} \leftarrow r_{15}$
32: $\quad r_{15} \leftarrow 0$
33: **Fin Pour**
34: **Pour** $i = n + 1$ **à** $2n$ **Faire**
35: $\quad c_i = r_{i-n}$
36: **Fin Pour**
37: **Retourner** $c = c_{2n} c_{2n-1}...c_0$

Algorithme 2.3 Calcul du carré d'un élément a de $\mathbb{F}_{2^d}$.

Entrées $a = a_n...a_0$
Sorties: $c = a^2$ dans le corps fini $\mathbb{F}_{2^d}$
  Pour $i = 0$ **à** n **Faire**
    $c_{2i} \leftarrow (T[a_{i,8-15}] << 16)|T[a_{i,0-7}]$
    $c_{2i+1} \leftarrow (T[a_{i,24-31}] << 16)|T[a_{i,16-23}]$
  Fin Pour
  Retourner $c = c_{2n}...c_0$

au carré retourne un vecteur de $2n$ registres. Il faut alors le réduire afin d'obtenir un élément de $\mathbb{F}_{2^d}$ dont le degré est strictement inférieur à d.

2.2.5 Inversion

En 1985, Wang *et al.* [WTS+85] introduisent une méthode pour calculer l'inverse d'un élément a dans $\mathbb{F}_{2^d}$. Leur stratégie est d'utiliser le petit théorème de Fermat qui est un cas particulier du théorème 2.6. Cette méthode permet de remplacer le calcul d'inverse par une exponentiation comportant majoritairement des élévations au carré. Ce calcul découle de l'équation dans $\mathbb{F}_{2^d}$ suivante :

$$a^{-1} = a^{2^d-2}$$

Il en découle $a^{2^d-1} = 1$ par application du théorème 2.6. Cette méthode d'inversion a été en premier lieu créée pour calculer des inverses sur une base normale de $\mathbb{F}_{2^d}$ puis a été améliorée par Itoh et Tsujii [IT88]. La chaîne d'addition nécessaire au calcul de a^{2^d-2} dépend du degré d, nous essayerons de ce fait de limiter le nombre de multiplications dans $\mathbb{F}_{2^d}$ à son minimum. Le détail de ces algorithmes est donné dans l'annexe B pour les différents degrés d'extensions de corps définis à la section 4.3.1.

2.2.6 Réduction

Le résultat d'une multiplication dans $\mathbb{F}_{2^d}$ utilisant l'algorithme 2.2, ou le résultat d'une élévation au carré, fait le double de la taille des opérandes d'entrées. Nous devons alors le réduire afin de limiter la taille des calculs des opérations à venir. Cette réduction est alors possible à l'aide du polynôme irréductible définissant l'extension de corps dans laquelle sont effectués les calculs.

Pour ce faire, nous optons pour une implémentation dédiée à chaque polynôme irréductible considéré afin de réduire au maximum le temps de calcul de cette opération. Les polynômes utilisés sont définis dans la section 4.3.1 tandis que les détails des algorithmes de réduction associés sont donnés par l'annexe A.2.

2.2.7 Calcul de la Trace

En considérant la définition 2.12 de la trace d'un élément a de $\mathbb{F}_{2^d}$, son calcul peut s'avérer coûteux en termes de performance. Cependant, l'application des formules de Newton-Girard à partir de la somme des conjugués α^i associée au fait que

l'application de trace est linéaire conduit à :

$$\text{Si } a = \sum_{i=0}^{d-1} a_i \alpha^k \in \mathbb{F}_{2^d}, \quad \text{Tr}(a) = a_0 + \sum_{i=1}^{d-1} i m_i a_{d-i}$$

Où les m_i sont les coefficients des monômes du polynôme irréductible dont α est une racine. Cette formule simplifie grandement le calcul de la trace notamment en caractéristique 2. En effet, dans ce cas précis, le calcul de la trace revient à effectuer une suite d'additions dans $\mathbb{F}_2$, le polynôme irréductible $m(X)$ est généralement choisi creux ce qui limite le nombre d'additions nécessaires pour le calcul de la trace.

Par exemple, si $m(X) = X^{233} + X^{74} + 1$ (qui est l'un des polynômes irréductible utilisé dans la norme du NIST [Gal13]), on obtient alors $\text{Tr}(a) = a_0 + a_{159}$ pour tout a dans $\mathbb{F}_{2^{233}}$.

2.2.8 Calcul de la demi-trace

Dans le cadre d'une implémentation de l'arithmétique des courbes elliptiques, nous serons amenés à résoudre des équations quadratiques dans $\mathbb{F}_{2^d}$ du type :

$$x^2 + x = A \tag{2.1}$$

Où A est un élément de $\mathbb{F}_{2^d}$. Si la trace de A est nulle, alors une telle équation admet pour solution x et $x+1$. Le calcul de la demi-trace (HT) de A, défini en fonction de la parité de d, conduit à une solution de l'équation 2.1.

Si d est impair, alors :

$$HT(A) = \sum_{i=0}^{(d-3)/2} A^{2^{2i+1}} \tag{2.2}$$

Si d est pair, alors :

$$HT(A) = \sum_{i=0}^{d-1} \left(\sum_{j=0}^{i} A^{2^j} \right) \tag{2.3}$$

2.3 Courbes Elliptiques

L'utilisation des courbes elliptiques en cryptographie (ECC) a été introduit en premier par Miller [Mil86] et Koblitz [Kob87]. Cette cryptographie repose sur la difficulté de résoudre le problème du logarithme discret sur le groupe des points d'une courbe elliptique. Les ECC ont fait l'objet d'une norme du NIST [Gal13] à partir d'une proposition de Vanstone [Van92]. Dans cette dernière, l'auteur présente l'ECDSA (*Elliptic Curve Digital Signature Algorithm*) qui est un algorithme de signature dont la sécurité repose sur l'ECDLP *Elliptic Curve Diffie-Hellman*. Ce schéma de signature est plus avantageux que le schéma RSA de par la petite taille des clefs utilisées qui permettent de réaliser des calculs de signatures performant, notamment dans un environnement contraint tel que l'IoT.

Dans cette section, nous détaillerons la géométrie et l'arithmétique des courbes elliptiques.

2.3.1 Généralité sur les courbes elliptiques

Il existe de nombreuses façons d'aborder la théorie des courbes elliptiques. Nous pouvons les considérer comme un pan de la théorie des variétés algébriques ou comme l'étude des équations de Weierstrass. Nous l'aborderons sous cet angle qui se rapproche des applications cryptographiques. De manière générale, nous pouvons définir une courbe elliptique comme suit :

Définition 2.13 (Courbes elliptiques). *Une courbe elliptique E sur un corps K est une courbe algébrique projective lisse de genre 1 possédant un point rationnel.*

Cette première définition peut se traduire à l'aide des équations de Weierstrass. Ainsi, une courbe elliptique E sur un corps K est définie par l'ensemble des points projectifs vérifiant l'équation de Weierstrass :

$$E : Y^2Z + a_1XYZ + a_3YZ^2 = X^3 + a_2X^2Z + a_4XZ^2 + a_6Z^3 \tag{2.4}$$

Par soucis de clarté, il est commun d'effectuer le changement de variable $x = X/Z$ et $y = Y/Z$ afin d'obtenir une équation non-homogène. Dans ce cas une courbe elliptique est définie par :

Définition 2.14 (Courbes elliptiques (Weierstrass)). *Une courbe elliptique E sur un corps K est définie par l'ensemble des points affines vérifiant l'équation de Weierstrass :*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \tag{2.5}$$

Où $a_1, .., a_6 \in K$. À cet ensemble de point nous ajoutons le point projectif $\mathcal{O} = [0 : 1 : 0]$, dit point à l'infini. De plus, nous avons comme condition que les dérivées partielles de la courbe E ne s'annulent pas simultanément.

Si la caractéristique de K est différente de 2, alors nous pouvons réduire l'équation 2.5 en effectuant une substitution $y \to \frac{1}{2}(y - a_1x - a_3)$ ce qui donne l'équation suivante.

$$E : y^2 = 4x^3 + b_2x^2 + 2b_4x + b_6 \tag{2.6}$$

Où $b_2 = a_1^2 + 4a_4$, $b_4 = 2a_4 + a_1a_3$ et $b_6 = a3^2 + 4a_6$. De même, si la caractéristique de K est différente de 3, alors nous pouvons opérer une seconde substitution $(x, y) \to \left(\frac{x-3b_2}{36}, \frac{y}{108}\right)$ ce qui nous permet d'éliminer le terme en x^2 afin d'avoir une version courte de l'équation 2.5 de Weierstrass :

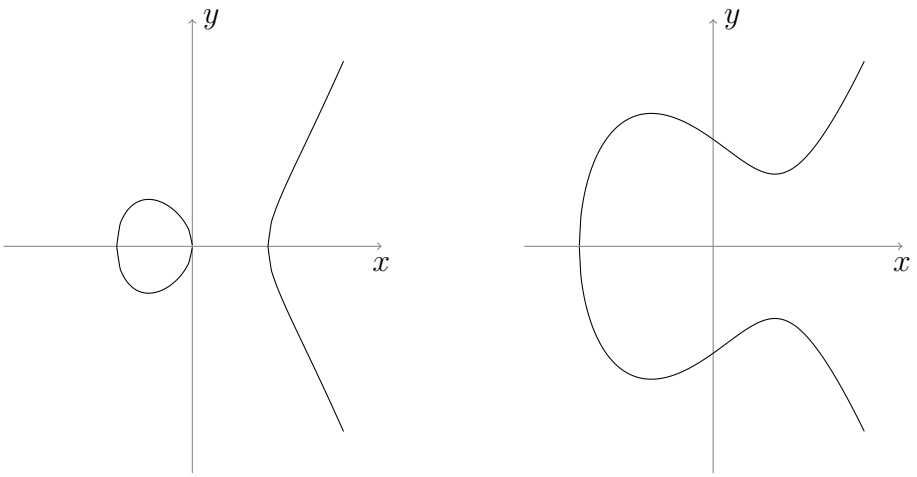
$$E := y^2 = -27c_4x - 54c_6 \tag{2.7}$$

En définissant les valeurs :

$$\begin{cases} b_8 = a_1^2a_6 + 4a_2a_6 - a_1a_3a_4 + a_2a_3^2 - a_4^2 \\ c_4 = b_2^2 - 24b_4 \\ c_6 = -b_2^3 + 36b_2b_4 - 216b_6 \end{cases}$$

Dans le cas où la caractéristique de K est égale à 2, nous pouvons tout de même réduire l'équation 2.5 en effectuant la transformation :

$$x \to a_1^2x + \frac{a_3}{a1}, \quad y \to a1^3y + \frac{a3^2 + a1^2a_4}{a1^3}$$



(a) $y^2 = x^3 - x$ (b) $y^2 = x^3 - 2x + 2$

FIGURE 2.2 – Courbes elliptiques sur $\mathbb{R}$

Nous obtenons alors la version courte de l'équation 2.5 de Weierstrass après une division par a_1^6 :

$$E : y^2 + xy = x^3 + b_2x^2 + b_6 \tag{2.8}$$

Le cas de la caractéristique 3 ne sera pas détaillé dans cette thèse, mais le lecteur pourra se référer aux livres [CFA+06, Sil09].

Nous pouvons alors définir les équations de Weierstrass courtes pour une courbe elliptique e sur un corps K.

Définition 2.15 (Courbes elliptiques (Weierstrass courte)). *L'équation courte de Weierstrass d'une courbe elliptique E sur K est donnée par :*

$$\begin{cases} E : y^2 = x^3 + ax + b & si\ car(K) \neq 2, 3 \\ E : y^2 + xy = x^3 + ax^2 + b & si\ car(K) = 2 \end{cases}$$

Où a, b sont des éléments de K

La Figure 2.2 représente deux exemples de courbes elliptiques définies sur $\mathbb{R}$ tandis que la Figure 2.3 représente une courbe elliptique sur $\mathbb{F}_{31}$. À ces deux représentations nous devons ajouter le point à l'infini qui ne peut pas être représenté sur ces figures. Notons que la grande différence entre une courbe elliptique sur $\mathbb{R}$ et sur $\mathbb{F}_p$ est son nombre de points. En effet, dans le premier cas, celui-ci est infini alors que dans le deuxième cas, il est fini et borné par le théorème 2.20 de Hasse comme nous le verrons. Le cardinal d'une courbe elliptique quant à lui joue un rôle primordial dans le cadre d'une application cryptographique.

Pour une courbe elliptique E sur K, il est possible de lui lier une autre courbe E^{tw} de telle sorte que E et E^{tw} sont isomorphes sur la clôture algébrique de K. Cette courbe E^{tw} est appelée tordue de E. La tordue d'une courbe elliptique E va contribuer à la sécurité du protocole cryptographique utilisé. Ce point sera abordé dans la section 4.2.

Nous décrivons ci-après deux invariants nécessaires pour l'étude des courbes elliptiques.

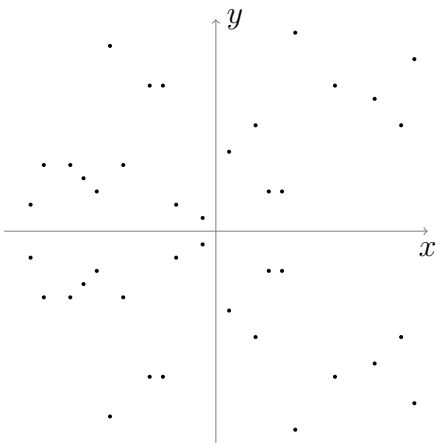


FIGURE 2.3 – Courbe elliptique définie sur $\mathbb{F}_{31}$ avec $E : y^2 = x^3 + x + 3$ et $|E| = 41$

Définition 2.16 (Discriminant). *Le discriminant d'une courbe elliptique E sur un corps K est défini par :*

$$\begin{cases} \Delta = -16(4a_4^3 + 27a_6^2) & si\ car(K) \neq 2, 3 \\ \Delta = a_6 & si\ car(K) = 2 \end{cases}$$

Définition 2.17 (j-invariant). *Le j-invariant d'une courbe elliptique E définie sur un corps K est :*

$$\begin{cases} j = \frac{1728a_4^3}{\Delta} & si\ car(K) \neq 2, 3 \\ j = \frac{1}{a_6} & si\ car(K) = 2 \end{cases}$$

De plus, la condition de la définition 2.14 sur les dérivées partielles d'une courbe elliptique est équivalent à la condition que $\Delta \neq 0$.

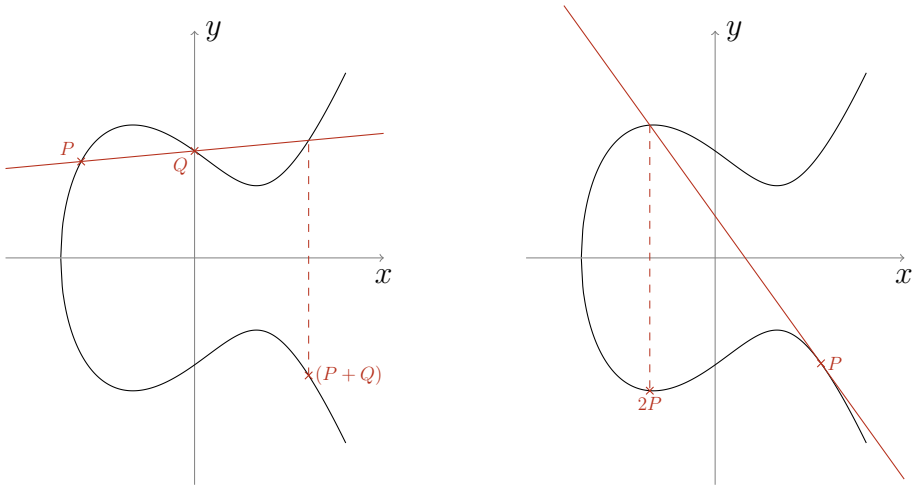
Loi de composition

Dans le but d'utiliser les courbes elliptiques dans des applications cryptographiques, on définit une loi de composition interne entre les points de la courbe elliptique de telle sorte à construire un groupe. Cette structure de groupe sur l'ensemble des points d'une courbe elliptique permettra de construire des protocoles de signatures ou d'échange de clefs.

La loi de composition interne sur une courbe elliptique sera notée additivement car, comme nous le verrons, celle-ci est abélienne. Ainsi, pour obtenir un groupe $(E, +)$, la loi de composition $+$ doit vérifier les conditions rappelées dans la section 2.1, à savoir dans ce cas :

1. **Associativité** : $\forall P, Q, R \in E : (P + Q) + R = P + (Q + R)$
2. **Élément neutre** : $\exists! \mathcal{O} \in E : \forall P \in E, P + \mathcal{O} = \mathcal{O} + P$
3. **Inverse** : $\forall P\ in E, \exists! Q \in E : P + Q = Q + P = \mathcal{O}$
4. **Commutativité** : $\forall P, Q \in E : P + Q = Q + P$

La construction d'une telle loi de compositions permettant de définir un groupe abélien sur la courbe elliptique E qui peut être abordée de différentes manières. La première approche est géométrique.



(a) Calcul de $P + Q$ (b) Calcul de $2P$

FIGURE 2.4 – Loi de groupe d'une courbe elliptique sur $\mathbb{R}$

La Figure 2.4 présente la loi de composition considérée pour construire un groupe sur l'ensemble des points de la courbe E, on définit comme élément neutre le point à l'infini $\mathcal{O}$.

La seconde approche quant à elle est algébrique, elle nous permettra de vérifier que la loi de groupe définie géométriquement par la figure 2.4 possède bien les propriétés relatives aux lois de composition de groupes abéliens. Pour cela, il nous faut décrire algébriquement cette loi de composition. Pour $P = (x_1, y_1)$, $Q(x_2, y_2)$ deux points distants de la courbe E. Nous définissons $P+Q = (x_3, y_3)$ de la manière suivante :

$$x_3 = \lambda^2 - a - x_1 - x_2, \quad y_3 = \lambda(x_1 - x_3) - y_1 - x_3$$

λ est définie différemment si $P = Q$ ou si $P \neq Q$.

$$\begin{cases} \lambda = \frac{y_1 - y_2}{x_1 - x2} & \text{si } P \neq \pm Q \\ \lambda = \frac{3x_1^2 + 2ax_1 - y_1}{2y_1 + x_1} & \text{si } P = Q \end{cases}$$

Dans le cas particulier, où $P = -Q$, la droite passant par P et Q est parallèle à l'axe des ordonnées, nous avons dans ce cas $y_1 = -y_2$ et donc $P + Q = \mathcal{O}$.

Nous avons vu précédemment que pour additionner deux points, il existe plusieurs cas selon que $P = Q$, $P = -Q$ ou $P = \mathcal{O}$. Ainsi, cette loi de groupe admet de ce fait différentes formules pour additionner et doubler des points sur E. Le point à l'infini $\mathcal{O}$ est un cas particulier à traiter à part. Deux notions essentielles permettent alors de qualifier une loi de composition sur E.

Définition 2.18 (Loi de groupe uniforme). *Une loi de groupe sur une courbe elliptique E est dite uniforme si les formules d'additions et de doublement sont identiques.*

Définition 2.19 (Loi de groupe complète). *Une loi de groupe sur une courbe elliptique E est dite complète si les formules d'additions et de doublement sont valides pour tout point de E.*

Ces deux définitions nous serons utiles dans la suite de cette étude afin de qualifier les lois de groupes de modèles alternatifs de courbes elliptiques.

Cardinal d'une courbe elliptique

Dans le cas où une courbe elliptique E est définie sur un corps fini, nous avons vu que le cardinal de E est fini. Un moyen d'obtenir ce cardinal est de chercher, de façon exhaustive, tous les points de E. Pour les courbes définies sur un corps fini de grand ordre cette méthode n'est pas envisageable car elle nécessiterait un temps de calcul considérable.

L'endomorphisme de Frobenius joue un rôle important pour compter le nombre de points d'une courbe elliptique sur un corps fini. Il est défini pour tout point $P = (x, y)$ de E sur le corps $\mathbb{F}_{p^d}$ par :

$$\Psi(P) = \begin{cases} (x^{p^d}, y^{p^d}) & \text{si } P \neq \mathcal{O} \\ \mathcal{O} & \text{si } P = \mathcal{O} \end{cases}$$

De cette définition du Frobenius en découle le théorème suivant :

Théorème 2.20 (Hasse-Weil). *Soit une courbe elliptique E définie sur $\mathbb{F}_{p^d}$, alors :*

$$|E| = p^d + 1 - t \text{ et } |t| \leq 2\sqrt{p^d}$$

Où t est la trace de l'endomorphisme de Frobenius.

Par conséquent, la difficulté de compter les points d'une courbe elliptique sur un corps fini revient à calculer la trace de l'endomorphisme de Frobenius. Il existe de nombreuses méthodes pour calculer celle-ci qui dépendent de la caractéristique du corps sur lequel est défini la courbe E. Des exemples sont donnés dans le chapitre 17 de [CFA+06] qui présente plusieurs techniques dont l'algorithme de Schoof-Elkies-Atkin (SEA) [Sch85, Atk88, elk91]. Dans le cas de la caractéristique 2, la méthode de la moyenne arithmético-géométrique (AGM) [Mes00b, Mes00a] sur les nombres p-adic est une approche habituellement utilisée pour compter le nombre de points d'une courbe elliptique. Pour les détails de cet l'algorithme et de l'arithmétique des nombres p-adic nous renvoyons le lecteur à la section 17.3.2 du livre [CFA+06].

L'algorithme 2.4 résume les différentes étapes de la méthode AGM pour compter le nombre de points d'une courbe elliptique E sur un corps fini de caractéristique 2.

Il existe un résultat intéressant qui lie le cardinal d'une courbe elliptique et celui de sa tordue E^{tw}.

Proposition 2.21. *Soit E une courbe elliptique sur $\mathbb{F}_{p^d}$ et E^{tw} une tordue de E, alors :*

$$|E| + |E^{tw}| = 2p^d + 2$$

Cette proposition aura un impact important dans la génération d'un nouvel ensemble de courbes elliptiques comme nous le verrons dans le chapitre 4. Afin de choisir une courbe elliptique destinée à des applications cryptographiques, nous devons vérifier la friabilité de son cardinal et celui de sa tordue. Ainsi, nous avons un seul calcul de cardinal à effectuer grâce à la propriété 2.21 ce qui nous permettra d'accélérer grandement la génération de nouvelles courbes elliptiques sécurisées pour la cryptographie.

Algorithme 2.4 Méthode AGM pour compter le nombre de point d'une courbe elliptique sur $\mathbb{F}_{2^d}$.

Entrées Un courbe elliptique $E : y^2 + xy = x^3 + b$ sur $\mathbb{F}_{2^d}$
Sorties: Le nombre de points de E sur $\mathbb{F}_{2^d}$
$N \leftarrow \lceil \frac{d}{2} \rceil + 3$
$a \leftarrow 1$
$b \leftarrow (1 + 8b) \mod 2^4$
Pour $i = 5$ **à** N **Faire**
$a \leftarrow \frac{a+b}{2} \mod 2^i$
$b \leftarrow \sqrt{ab} \mod 2^i$
Fin Pour
$a_0 \leftarrow \frac{a+b}{2} \mod 2^N$
$t \leftarrow \frac{a}{a_0} \mod 2^{N-1}$
Si $t^2 > 2^{d+2}$ **alors**
$t \leftarrow t - 2^{N-1}$
Fin Si
Retourner $2^d + 1 - t$

2.3.2 Arithmétique des courbes elliptiques

L'implémentation cryptographique basée sur les courbes elliptiques dépend de l'arithmétique de ces dernières et impacte sur l'efficacité du protocole cryptographique. De façon plus détaillée, les calculs effectués sur une courbe elliptique sont basés sur l'arithmétique des corps finis. Nous avons vu dans la section 2.2 les algorithmes composant une arithmétique en caractéristique 2. Dans cette section, nous présenterons les différents opérations clefs du calcul sur une courbe elliptique que nous devons considérer pour des applications cryptographiques.

Dans un protocole d'échange de clefs tel que l'ECDH [BCR+03] ou dans un protocole de signature tel que l'ECDSA [Gal13], l'opération principale effectuée sur une courbe elliptique est la multiplication scalaire. Cette opération consiste à calculer :

$$kP = \underbrace{P + ... + P}_{k \text{ fois}}$$

Où k est un entier et P un point de E.

L'arithmétique des courbes elliptiques se décompose alors en une suite d'opérations élémentaires dans $\mathbb{F}_{p^d}$ et dans la suite nous traiterons seulement le cas où $p = 2$.

Nous avons vu dans la section 2.2 qu'il existe une hiérarchie en termes de performance dans les opérations sur $\mathbb{F}_{2^d}$. En effet, l'inversion est l'opération la plus coûteuse, car elle nécessite plusieurs multiplications pour être effectuée. La multiplication est aussi très onéreuse en temps de calcul. En comparaison, l'addition et l'élévation au carré sont très peu coûteuses même si pour le cas de l'élévation au carré, nous devons effectuer une réduction sur le résultat obtenu. Afin de mesurer les performances des différentes implémentations de courbes elliptiques nous les comparerons entre elles en termes d'opérations élémentaires sur $\mathbb{F}_{2^d}$. Nous utiliserons comme métrique [1] qu'une inversion (I) équivaut à 10 multiplications et que l'élévation au carré (S) et l'addition (A) sont négligeables. Ainsi, toutes les mesures seront

1. https ://hyperelliptic.org/EFD/g12o/index.html

données en fonction de nombre de multiplications nécessaires. Dans certains cas, nous serons malgré tout amené à comparer le nombre d'élévations au carré afin de départager deux systèmes de coordonnées ayant le même coût en multiplications.

Systèmes de coordonnées

L'opération fondamentale sur une courbe elliptique E est l'addition de deux points P et Q. Cette opération va grandement dépendre du système de coordonnées considérées pour représenter un point sur E.

Le premier système de coordonnées que nous avons vu précédemment est la représentation affine des points d'une courbe de Weierstrass. En analysant les formules d'addition et de doublement de ce système de coordonnées nous obtenons que 12 multiplications sont nécessaires pour une addition (1I + 2M + 1S) qu'un doublement.

Dans ces conditions, les performances des coordonnées affines obtenues, sont incompatibles pour des applications cryptographiques, dues à l'opération d'inversion qui est très coûteuse. Ainsi, un moyen d'améliorer les performances de l'addition et du doublement est de ne pas calculer cette inversion et de l'accumuler dans une troisième coordonnée Z qui sera inversée à la toute fin du calcul de kP. Ce système de coordonnées est dit projectif et représente un point P par trois coordonnées $(X : Y : Z)$, il peut être converti vers les coordonnées affines en appliquant la transformation suivante :

$$(X : Y : Z) \rightarrow \left(\frac{X}{Z}, \frac{Y}{Z}\right)$$

Dans ce système projectif un point P peut avoir plusieurs représentations qui correspondent à une seule et unique représentation affine : un point $P = (X : Y : Z)$ est égale à un point $P' = (\lambda X : \lambda Y : \lambda Z)$. Nous verrons qu'une telle propriété a son importance dans la protection des protocoles cryptographiques sur courbes elliptiques face aux attaques par canaux auxiliaires.

Finalement, l'adoption d'une telle représentation des points permet d'accélérer le doublement, mais ralentit l'addition. L'addition dans ce cas nécessitera 14 multiplications (14 M + 1 S) et le doublement 7 (7M + 1S). Malgré tout ce système accélère le calcul de la multiplication scalaire, car cette opération utilise essentiellement des doublements. Les détails des formules d'addition et de doublement sont donnés à la section 13.3.1 du livre [CFA+06].

Il existe de nombreuses autres méthodes de représentation des points qui permettent d'accélérer d'avantage les calculs. Nous pouvons citer les coordonnées jacobiennes ou les coordonnées de Lopez-Dahab. Nous nous attarderons sur un système particulier celui des coordonnées différentielles qui consiste à représenter un point uniquement par sa coordonnée affine x. Cette forme particulière de représentation du point utilisée avec des méthodes de calculs de kP spécifiques permet d'augmenter les performances [Sta02a]. Le doublement ne nécessite qu'une multiplication (1M + 3S) et l'addition 5 (5M + 3S). Les auteurs de [FLRV08a] montrent qu'un tel choix de représentation des coordonnées peut être un vecteur d'attaque par fautes.

In fine, le tableau 2.1 compare le coût en multiplications des différents systèmes de coordonnées sur les courbes de Weierstrass définie sur $\mathbb{F}_{2^d}$.

	Doublement	Addition
Affines	12M	12M
Projectives	7M	14M
Jacobiennes	4M	14M
Lopez-Dahab	3M	13M
Différentielles	1M	5M

TABLE 2.1 – Coût en multiplications de l'addition et du doublement de points dans différents systèmes de coordonnées sur les courbes de Weierstrass.

Multiplication scalaire

De l'addition de deux points sur une courbe elliptique, nous pouvons construire l'opération de multiplication scalaire d'un point qui est essentielle pour la construction des protocoles cryptographiques sur les courbes elliptiques. Cette dernière consiste à effectuer le calcul kP. Les algorithmes de multiplications scalaires vont être ressemblant de ceux de l'exponentiation, seule l'opération de base change : pour la multiplication scalaire, on aura une addition de points sur les courbes elliptiques tandis que pour l'exponentiation, on aura une multiplication entre des éléments d'un groupe multiplicatif.

Le premier algorithme de multiplication scalaire est celui de « *double and add* », algorithme 2.5.

Algorithme 2.5 Algorithme *double and add* de multiplication scalaire d'un point P sur une courbe elliptique E

Entrées k un scalaire, P un point de E
Sorties: kP
  $Q \leftarrow \mathcal{O}$
  $n = \log_2(k)$
  Pour $i = n - 1$ **à** 0 **Faire**
    $Q \leftarrow 2Q$
    Si $k_i = 1$ **alors**
      $Q \leftarrow Q + P$
    Fin Si
  Fin Pour
  Retourner $Q = kP$

Cependant, cet algorithme possède deux inconvénients majeurs.

Le premier est qu'il est très peu performant, la multiplication scalaire est effectuée en traitant le scalaire k bit à bit. Une amélioration consiste à utiliser un algorithme à fenêtre glissante qui permettra de parcourir plus rapidement le scalaire k en contrepartie de quelques pré-calculs qui dépendent de la taille de la fenêtre (Algorithme 2.6). Il existe des variantes à cet algorithme à pré-calculs tel que l'algorithme w-NAF qui va parcourir différemment le scalaire k. Bien que ces algorithmes améliorent les performances, ils ne résolvent pas le problème lié à la non-uniformité des segments des calculs qui dépend de la valeur de la clef.

Le second est relatif aux opérations effectuées à chaque étape qui dépendent directement du scalaire k. Si k est une clef privée alors cette différence d'exécution

Algorithme 2.6 Algorithme à fenêtre glissante de multiplication scalaire d'un point P sur une courbe elliptique E

Entrées k un scalaire, P un point de E, t une taille de fenêtre
Sorties: kP

```
T un tableau de 2^{t-1} points.
Pour i = 0 à 2^{t-1} Faire
    T[i] ← iP
Fin Pour
Q ← O
n = log_2(k)
Pour i = n − 1 à 0 Faire
    Si k_i = 0 alors
        Q ← 2Q
    Sinon
        Si i >= t alors
            w ← extraire les t prochains bits de k
            Pour j = 0 à t Faire
                Q ← 2Q
            Fin Pour
            Q ← Q + T[w]
        Sinon
            Appliquer l'algorithme 2.5 sur les bits de k restant
        Fin Si
    Fin Si
Fin Pour
Retourner Q = kP
```

pourra être exploitée au travers d'attaques par canaux auxiliaires afin d'extraire de l'information sur la clef. Par exemple, si le bit courant est un 1 alors une opération supplémentaire d'addition de point est réalisée, ce qui va modifier la trace de la consommation de courant ou du rayonnement électromagnétique acquise à l'issue du calcul. Nous verrons plus en détails cet aspect de la cryptographie sur les courbes elliptiques dans le chapitre 3. Une première possibilité pour unifier l'algorithme 2.5, de manière à avoir les mêmes opérations pour un bit 0 ou 1, est d'effectuer une addition fictive lorsque le bit courant est 0. Nous obtenons alors l'algorithme de *double and add always* (Algorithme 2.7). Le problème de cette approche est qu'elle impacte le problème de performance de l'algorithme 2.5. De plus, la protection apportée est d'une utilité limitée car dans ce cas l'algorithme 2.7 est sensible aux attaques par fautes mais aussi aux attaques par canaux auxiliaires qui permettent de distinguer les accès au point T. Si une faute est effectuée durant l'addition fictive un attaquant peut alors inférer que le bit courant est nul, car le résultat final n'aura pas été modifié.

Algorithme 2.7 Algorithme *double and add always* de multiplication scalaire d'un point P sur une courbe elliptique E

Entrées k un scalaire, P un point de E
Sorties: kP
  $Q \leftarrow \mathcal{O}$
  $n = \log_2(k)$
  Pour $i = n - 1$ **à** 0 **Faire**
    Si $k_i = 1$ **alors**
      $Q \leftarrow 2Q$
      $Q \leftarrow Q + P$
    Sinon
      $Q \leftarrow 2Q$
      $T \leftarrow Q + P$
    Fin Si
  Fin Pour
  Retourner $Q = kP$

Ainsi, nous devons considérer un algorithme de multiplication scalaire sur les courbes elliptiques qui devra être efficace et effectuant les mêmes opérations quel que soit l'état du bit courant. L'algorithme de l'Échelle de Montgomery [JY02], algorithme 2.8, est une méthode de multiplication scalaire qui va effectuer les mêmes opérations indépendamment de la valeur de la clef traitée. Contenu des critères de performance et de sécurité, notre choix s'est porté sur cet algorithme pour la suite de nos travaux.

2.3.3 Les modèles alternatifs de Courbes Elliptiques

Comme nous l'avons vu précédemment, le modèle de Weierstrass de courbes elliptiques, tel que présenté dans la définition 2.15, muni de la loi de groupe usuelle, admet des formules d'additions et de doublement de points différents. De plus, nous ne pouvons pas appliquer ces formules à tous les points de la courbe car certains points, tel que le point $\mathcal{O}$, ne vérifient pas les équations d'addition et de doublement.

Algorithme 2.8 Algorithme de l'Échelle de Montgomery pour la multiplication scalaire d'un point P sur une courbe elliptique E

Entrées k un scalaire, P un point de E
Sorties: kP
  $R_0 \leftarrow \mathcal{O}$
  $R_1 \leftarrow P$
  $n = \log_2(k)$
  Pour $i = n - 1$ **à** 0 **Faire**
    Si $k_i = 0$ **alors**
      $R_1 \leftarrow R_0 + R_1$
      $R_0 \leftarrow 2R_0$
    Sinon
      $R_0 \leftarrow R_0 + R_1$
      $R_1 \leftarrow 2R_1$
    Fin Si
  Fin Pour
  Retourner $Q = kP$

Cette particularité peut s'avérer problématique lorsque l'on considère une implémentation de la multiplication scalaire d'un point en temps constant, *i.e.* pour tout point d'entrée P et tout scalaire k, le temps d'exécution de l'algorithme calculant kP est constant. Comme les temps de calculs pour effectuer kP sont différents, alors cela impliquent l'utilisation de branchements conditionnels dont le temps d'exécution dépend des entrées. Ainsi, il est commun d'utiliser des modèles alternatifs de courbes elliptiques qui admettent une loi de groupe complète et uniforme. Notons que récemment Joost *et al.* [RCB16] ont construit une loi de groupe complète pour les courbes de Weierstrass.

En pratique, considérer la nécessité d'une loi de groupe uniforme et complète pour des applications cryptographiques des courbes elliptiques n'est pas un bon argument. En effet, pour toutes les lois de groupe sur courbe elliptique le calcul du doublement de point est beaucoup plus rapide que celui de l'addition, utiliser les formules d'additions pour calculer un doublement de point n'est pas efficace et risque de grandement ralentir les performances du calcul de kP. Dans ces conditions pour les applications réelles, nous n'utiliserons pas l'uniformité d'une loi de groupe. De plus, la complétude de la loi de groupe n'apporte aucune sécurité, car les points qui posent généralement problème, comme dans le cas de la loi de groupe sur les courbes de Weierstrass, seront dans tous les cas évités. Comme nous le verrons dans le chapitre 3, ces points peuvent être utilisés pour mener des attaques par canaux auxiliaires [GVY17a].

L'utilisation de modèles alternatifs permet cependant d'accélérer les calculs sur les courbes elliptiques, car les lois de groupe alors considérées sont plus efficaces comme nous le verrons par la suite. Le premier exemple en ce sens est celui des courbes de Montgomery [Mon87] qui ont été construites afin d'accélérer l'algorithme de factorisation utilisant les courbes elliptiques [Len87].

Comme nous l'avons vu à la section 1.1, l'IoT que nous considérons dans cette étude ne possède pas de bit de contrôle de la retenue. Ainsi, une implémentation basée sur les courbes elliptiques définies sur un corps de grande caractéristique risque

d'être lourdement pénalisée en termes de performances. En effet, l'arithmétique sur $\mathbb{F}_p$ avec p impair nécessite de propager la retenue tout au long du calcul d'addition ou de multiplication. Il s'en suit, que nous privilégierons les modèles en caractéristiques 2 afin d'éviter toutes les contraintes relatives à la propagation de la retenue. De plus, certaines attaques par canaux auxiliaires utilisent cette propagation de la retenue afin d'inférer de l'information sur la clef secrète k utilisée lors du calcul de kP, [DPN+16a]. Nous présenterons brièvement les modèles alternatifs dans le cas de la caractéristique impair, car la majeure partie des modèles alternatifs de courbes elliptiques sur les corps binaires sont des adaptations de modèles déjà existant sur $\mathbb{F}_p$.

Modèles alternatifs de courbes elliptiques sur $\mathbb{F}_p$

L'un des premiers modèle alternatif de courbes elliptiques a été introduit par Montgomery [Mon87], il a pour avantage de pouvoir être utilisé efficacement avec l'algorithme 2.8 de multiplication scalaire de l'Échelle de Montgomery. Il permet aussi de d'utiliser des formules d'additions et de doublement de points très efficaces dans le cas de l'utilisation de l'Échelle de Montgomery. Une courbe de Montgomery se présente sous la forme suivante :

$$E_{A,B} : By^2 = x^3 + Ax^2 + x \tag{2.9}$$

Les travaux de Joye [JQ01], de Doche-Icart-Kohel [DIK06] et d'Edwards [Edw07] qui présentent une forme alternative de courbes elliptiques admettant une loi de groupe complète et uniforme ont permis de considérer les modèles alternatifs pour des applications cryptographique. Le modèle d'Edwards est défini par l'équation suivante :

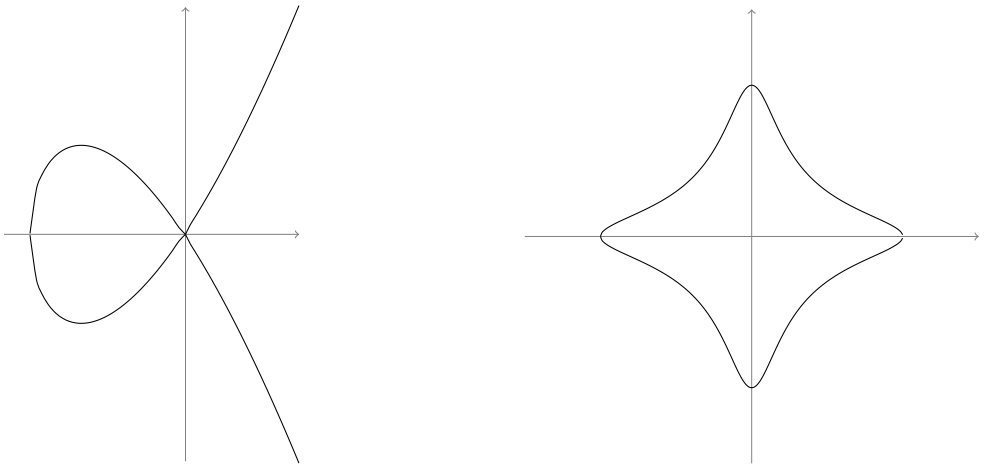
$$E_d : x^2 + y^2 = 1 + dxy \tag{2.10}$$

Les courbes d'Edwards et de Montgomery ont l'avantage d'être birationnelles à une courbe de Weierstrass, cette propriété est importante dans le cadre d'applications cryptographiques. En effet, une grande partie des IoT déployés utilisent le modèle de Weierstrass notamment au travers du standard du NIST [Gal13]. Ainsi, si nous voulons déployer un nouveau réseau IoT utilisant un modèle alternatif de courbes elliptiques, il est important que celui-ci puisse communiquer parfaitement avec un réseau préexistant utilisant quant à lui les courbes de Weierstrass. Finalement, le modèle de Weierstrass peut être considéré comme un invariant dans un écosystème IoT hétérogène, ce qui impose la nécessité de compatibilité entre les modèles.

La figure 2.5, donne la représentation des courbes de Montgomery et d'Edwards sur $\mathbb{R}$.

Les travaux de Bernstein *et al.* [BDL+11b, BCL14, BJL+15] ont permis d'adopter les courbes d'Edwards sur un corps à large caractéristique pour des applications cryptographiques. Les courbes d'Edwards Curve22519 [BDL+11b] et Curve448 [Ham15] font l'objet de normes RFC [LHT16, NJ16, JL17].

Il existe de nombreux modèles alternatifs de courbes elliptiques sur $\mathbb{F}_p$ tels que : la forme de Doche-Icart-Kohel [DIK06], les jacobiennes quadratiques [HWCD09], l'intersection de jacobienne [FNW10], les hessiennes [JQ01, FJ10] et les hessiennes



(a) $3y^2 = x^3 + 7x^2 + x$ (b) $x^2 + y^2 = 1 - 30xy$

FIGURE 2.5 – Courbe de Montgomery (A) et courbe d'Edwards (B) sur $\mathbb{R}$.

Modèles	Addition	Doublement	Différentielle
Weierstrass [OLAR13, Sta02b]	8M + 2S	3M + 4S + 2m	5M + 3S + 1m
Edwards [BLF08, KLN14, KAK15]	13M +3S + 2m	2M + 6S + 1m	5M + 4S + 1m
Hessienne [FJ10]	8M + 3S + 2m	6M + 3S + 1m	5M + 4S + 1m
Huff [DJ11]	14M	6M + 2m	5M + 1m
μ_4-normale [Koh17]	9M + 2S	2M +5S + 2m	4M + 4S + 3m

TABLE 2.2 – Nombres d'opérations nécessaire pour additionner et doubler un point sur les modèles alternatifs de courbes elliptiques en caractéristique 2.

tordues [BCKL15], les courbes d'Edwards [Edw07, BL07] et les courbes d'Edwards tordues [BBJ+08, HWCD08], les courbes de Huff [JTV10].

Modèles alternatifs de courbes elliptiques sur $\mathbb{F}_{2^d}$

Les travaux sur les modèles alternatifs de courbes elliptiques dans le cas de la caractéristique 2 sont bien moins importants. La majeure partie de ces modèles alternatifs sont une adaptation de modèles en grande caractéristiques préexistants. C'est le cas pour les courbes binaires d'Edwards [BLF08] que nous détaillerons dans la section 2.4 les propriétés mathématiques et arithmétiques.

Il existe d'autres modèles alternatifs de courbes elliptiques dans le cas de la caractéristique 2 tel que : les hessiennes [JQ01, FJ10], les courbes binaires de Huff [DJ11] et les formes μ_4-normales [Koh12, Koh17]. Ces différents modèles ont des propriétés mathématiques similaires utiles pour une application cryptographique : ces modèles sont birationnels à celui de Weierstrass et leur loi d'addition est complète et uniforme. Notons que le modèle des hessiennes a l'avantage d'être indépendant de la caractéristique du corps rendant l'implémentation de leur arithmétique compatible pour la caractéristique 2 et la caractéristique impaire.

La table 2.2 présente les meilleures performances d'addition et de doublement de points pour chaque modèle de courbe elliptique en caractéristique 2. La colonne différentielle de la table 2.2 donne la performance de calcul pour une étape de l'Échelle de Montgomery en coordonnées différentielles ou dans le cas du modèle de Weierstrass en utilisant seulement la coordonnée x d'un point.

Les performances des différents modèles sont très proches en coordonnées différentielles, exceptées le modèle μ_4-normal mais comme nous pourrons le voir par la suite nous atteindrons les mêmes performances avec le modèle binaire d'Edwards en optimisant le choix du point générateur, 4.3.3.

Notre choix s'est donc porté sur le modèle de courbe binaire d'Edwards. De plus, il existe déjà un grand nombre de travaux sur les implémentations matérielles de ce modèle, [KAK15, AR12, FSK16, RFS17, RSF17, FAB18, LDM19]. Les travaux de cette thèse portent sur les implémentations logicielles des courbes binaires d'Edwards, cependant le nouvel ensemble de courbes elliptiques qui sera présenté lors du chapitre 4 pourra parfaitement être utilisé pour des implémentations matérielles.

2.4 Courbes Binaires d'Edwards et leurs implémentations

Les courbes binaires d'Edwards (*Binary Edwards Curves* ou BEC) ont été introduites par Bernstein *et al.* [BLF08] suite aux travaux d'Edwards [Edw07] en caractéristique impaire. L'arithmétique de ces courbes a été améliorée par Kim *et al.* [KLN14] et Koziel *et al.* [KAK15]. La suite de la section présente en détails les propriétés mathématiques et arithmétiques propres à ce modèle de courbe elliptique.

2.4.1 Définitions et propriétés générales

Définition 2.22. *Soit K un corps de caractéristique 2 et soit d_1, d_2 deux éléments de K tels que $d_1 \neq 0$ et $d_2 \neq d_1^2 + d_1$. Nous définissons alors la courbe binaire d'Edwards E_{d_1,d_2} comme étant la courbe affine définie par :*

$$E_{d_1,d_2} : d_1(x+y) + d_2(x^2+y^2) = xy + xy(x+y) + x^2y^2$$

De cette définition, nous observons que la courbe est symétrique en x et y. Le théorème ci-dessous donne un résultat intéressant dans le cadre d'application cryptographique des BEC.

Théorème 2.23 (Non-singulière). *Toutes les courbes binaires d'Edwards sont non-singulières.*

Démonstration. Pour montrer que la courbe donnée par la définition 2.22 n'est pas singulière pour tout d_1 et d_2 tels que $d_1 \neq 0$ et $d_2 \neq d_1^2 + d_1$, il faut montrer que les dérivées partielles en x et y de cette courbe ne s'annulent pas simultanément.

Les dérivées partielles en x puis en y de E_{d_1,d_2} sont égales à $d_1 + y + y^2$ et $d_1 + x + x^2$. Dans ce cas, un point singulier (x_1, y_1) devra vérifier que $d_1 + y_1 + y_1^2 = 0$ et $d_1 + x_1 + x_1^2 = 0$, et donc que $(x_1 + y_1)^2 = x_1 + y_1$ ce qui implique que $x_1 = y_1$ ou que $x_1 = y_1 + 1$.

- Si $x_1 = y_1$, alors à partir de l'équation de la définition 2.22, nous avons $0 = x_1^2 + x_1^4$ et donc que $d_1^2 = x_1^2 + x_1^4 = 0$, ce qui est impossible vu la condition que $d_1 \neq 0$.
- Si $x_1 = y_1 + 1$, alors $d_1 + d_2 = y_1^2 + y_1^4$ et donc $d_1 = y_1^2 + y_1^4 = d_1 + d_2$, ce qui est aussi impossible car $d_2 \neq d_1^2 + d_1$

□

La clôture algébrique de E_{d_1,d_2} est donnée par :

$$d_1(X+Y)Z^3 + d_2(X^2+Y^2)Z^2 = XYZ^2 + XY(X+Y)Z + X^2Y^2$$

Les deux points à l'infini : $(1:0:0)$ et $(0:1:0)$ sont des points singuliers. Pour éviter ces singularités nous pouvons étudier le cas du point $(1:0:0)$ et par symétrie nous pouvons nous ramener au point $(0:1:0)$. Nous considérons alors la courbe affine $d_1(1+y)z^3 + d_2(1+y^2)z^2 = xyz^2 + xy(x+y)z + x^2y^2$. Les dérivées partielles $d_1z^3 + z^2 + z$ et $d_1(1+y)z^2 + y(1+y)$ s'annulent simultanément en $(0,0)$ ce qui nous donne effectivement une singularité en ce point. Pour éviter cette singularité nous pouvons poser $y = tz$ et en divisant par z^2 nous obtenons la courbe :

$$d_1(1+tz)z + d_2(1+t^2z^2) = tz + t(1+tz) + t^2$$

En substituant z par zéro nous obtenons alors l'équation $d_2 + t + t^2 = 0$ qui admet deux racines distinctes dans la clôture algébrique de K, qui sont exactement les points $(1:0:0)$ et $(0:1:0)$. Ainsi, ces deux points ne sont pas singuliers tant que la dérivée partielle $d_1z^2 + z + 1$ ne s'annule pas si $z = 0$. Cette manière d'éviter une singularité aux points $(1:0:0)$ et $(0:1:0)$ est définie sur la plus petite extension de K dans laquelle $d_2 + t + t^2 = 0$ admet ses racines.

Une propriété importante de ce modèle de courbe est son équivalence birationnelle avec le modèle de Weierstrass. Cette propriété permet donc de passer d'un modèle à l'autre et ainsi garantir une compatibilité des protocoles cryptographiques basés sur les BEC avec ceux basés sur le modèle de Weierstrass. Ainsi, le modèle de BEC donnée par la définition 2.22 est birationnel à la courbe de Weierstrass donnée par :

$$v^2 + uv = u^3 + a_2u^2 + a_6$$

avec $a_6 \neq 0$. Nous pouvons définir cette application $\Phi(x,y) \to (u,v)$ par

$$u = d_1(d_1^2 + d_1 + d_2)\left(\frac{x+y}{xy + d_1(x+y)}\right) \tag{2.11}$$

$$v = d_1(d_1^2 + d_1 + d_2)\left(\frac{x}{xy + d_1(x+y)} + d_1 + 1\right) \tag{2.12}$$

De plus la courbe de Weierstrass associée est donnée par :

$$v^2 + uv = u^3 + (d_1^2 + d_2)u^2 + d_1^4(d_1^4 + d_1^2 + d_2^2) \tag{2.13}$$

Il s'en suit que le j-invariant de la courbe est $1/(d_1^4(d_1^4 + d_1^2 + d_2^2))$ et que l'application inverse $\varphi^{-1}(u,v) \to (x,y)$ de φ est donnée par les formules suivantes :

$$x = \frac{d_1(u + d_1^2 + d_1 + d_2)}{u + v + (d_1^2 + d_1)(d_1^2 + d_1 + d_2)} \tag{2.14}$$

$$y = \frac{d_1(u + d_1^2 + d_1 + d_2)}{v + (d_1^2 + d_1)(d_1^2 + d_1 + d_2)} \tag{2.15}$$

Pour que ces deux applications φ et φ^{-1} soient pleinement définies pour tous les points de la courbe E_{d_1,d_2} nous devons les étendre au point $(0,0)$ de la manière suivante :

$$\varphi(0,0) \to \mathcal{O}, \quad \varphi^{-1}(\mathcal{O}) \to (0,0)$$

Où $\mathcal{O}$ est le point à l'infini et élément neutre du modèle de Weierstrass.

+	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
$(0,0)$	$(0,0)$	$(1,0)$	$(0,1)$	$(1,1)$
$(1,0)$	$(1,0)$	$(1,1)$	$(0,0)$	$(0,1)$
$(0,1)$	$(0,1)$	$(0,0)$	$(1,1)$	$(1,0)$
$(1,1)$	$(1,1)$	$(0,1)$	$(1,0)$	$(0,0)$

TABLE 2.3 – Table d'addition du sous-groupe d'ordre 4 généré par $(1,0)$ sur une courbe binaire d'Edwards avec $d_1 = d_2$

Loi de groupe

Le modèle de courbes elliptiques binaires d'Edwards admet aussi une loi de composition interne qui permet de donner une structure de groupe à l'ensemble des points de la courbe elliptique E_{d_1,d_2}. Nous pouvons alors définir l'addition entre deux points $P = (x_1, y_1)$ et $Q = (x_2, y_2)$, donnant le point $R = (x_3, y_3)$, par :

$$x_3 = \frac{d_1(x_1 + x_2) + d_2(x_1 + y_1)(x_2 + y_2) + (x_1 + x_1^2)(x_2(y_1 + y_2 + 1) + y_1y_2))}{d_1 + (x_1 + x_1^2)(x_2 + y_2)} \tag{2.16}$$

$$y_3 = \frac{d_1(y_1 + y_2) + d_2(x_1 + y_1)(x_2 + y_2) + (y_1 + y_1^2)(y_2(x_1 + x_2 + 1) + x_1x_2))}{d_1 + (y_1 + y_1^2)(x_2 + y_2)} \tag{2.17}$$

En posant $(x_1, y_1) = (0,0)$ ou $(x_2, y_2) = (0,0)$, il s'en suit que le point $(0,0)$ est l'élément neutre de cette loi de groupe. De même nous avons $(x_1, y_1) + (1,1) = (x_1 + 1, y_1 + 1)$ ce qui nous donne que $(1,1) + (1,1) = (0,0)$. Ainsi, le point $(1,1)$ est un point d'ordre deux.

Dans le cas particulier où $d_1 = d_2$ nous avons qu'un seul paramètre d pour définir la courbe binaire d'Edwards, nous avons alors la condition que pour t dans K : $d \neq t^2 + t$. Les formules de la loi de groupe donnent $2(1,0) = (1,1)$ et $2(0,1) = (1,1)$, les points $(1,0)$ et $(0,1)$ sont donc des points d'ordre 4. La table 2.3 détaille la loi d'addition dans le sous-groupe d'ordre 4 formé par les points : $(0,0)$, $(1,0)$, $(0,1)$ e $(1,1)$. Ainsi, quel que soit le paramètre d de E_d, ce sous-groupe d'ordre 4 sera présent et donc le cardinal d'une courbe binaire d'Edwards avec $d_1 = d_2$ sera toujours divisible par quatre. Cette loi de groupe possède certaines propriétés intéressantes.

Proposition 2.24. *La loi de groupe définie par les équations 2.16 et 2.17, sur la courbe binaire d'Edwards E_{d_1,d_2}, vérifie les propriétés suivantes :*

- *si $(x_1, y_1) + (x_2, y_2) = (0,0)$, alors $(x_1, y_1) = (x_2, y_2)$.*
- *si $\varphi(x_1, y_1) = \varphi(x_2, y_2)$, alors $(x_1, y_1) = (x_2, y_2)$.*
- *$\varphi(y_1, x_1) = -\varphi(x_1, y_1)$.*
- *si $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$, alors $\varphi(x_1, y_1) + \varphi(x_2, y_2) = \varphi(x_3, y_3)$*

Où (x_1, y_1), (x_2, y_2) et (x_3, y_3) sont des points de E_{d_1,d_2} et φ est l'application d'équivalence birationnelle entre le modèle des courbes binaires d'Edwards et celui de Weierstrass.

Pour les différentes preuves de ces différentes propriétés, nous renvoyons le lecteur à [BLF08].

Si la trace de d_2 est égale à 1, alors la loi de groupe est complète, *i.e.* que les dénominateurs des équations 2.16 et 2.17 ne s'annulent jamais. Dans le cas où le corps K est une extension de $\mathbb{F}_2$ de degré supérieur ou égal à 3 alors toute courbe binaire d'Edwards admet une équivalence birationnelle avec une courbe binaire d'Edwards munie d'une loi de groupe complète. Il s'en suit la définition d'une courbe binaire d'Edwards complète.

Définition 2.25 (Courbe binaire d'Edwards complète). *Soit K un corps de caractéristique 2 et soient deux éléments d_1 et d_2 de ce corps tels que $d_1 \neq 0$ et qu'il n'existe pas d'élément t de K vérifiant l'équation $t^2 + t = d_2$,* i.e. *la trace de d_2 est égale à 1. Nous pouvons alors définir la courbe binaire d'Edwards complète donnée par l'équation affine :*

$$E_{d_1,d_2} : d_1(x + y) + d_2(x^2 + y^2) = xy + xy(x + y) + x^2y^2$$

Dans la suite de ce chapitre, nous supposerons que toutes les courbes d'Edwards considérées sont complètes.

2.4.2 Systèmes de coordonnées

Le choix du système de coordonnées est essentiel dans le cadre d'une application cryptographique des courbes elliptiques car les performances de ce dernier vont grandement dépendre des performances du système de coordonnées pour additionner et doubler les points de la courbe. Idéalement, le système de coordonnées le plus pertinent est celui qui va effectuer le minimum d'opérations pour calculer une addition et un doublement. Ceci est dû à l'utilisation d'algorithme homogène pour effectuer la multiplication scalaire kP tel que l'algorithme 2.8 de l'Échelle de Montgomery. Ces algorithmes sont importants pour se prémunir d'attaques par canaux auxiliaires telles que les attaques exploitant la différence de temps de calcul observée entre différents paramètres d'entrées.

Nous évaluerons les différents systèmes de coordonnées en fonction du nombre d'opérations nécessaire dans le corps fini pour effectuer une addition et un doublement. Les opérations prises en compte seront : l'inversion (I), la multiplication (M), l'élévation au carré (S) et la multiplication par une constante (m) pouvant être choisie petite évitant ainsi d'effectuer une multiplication complète. Pour rappel toutes les opérations n'ont pas le même coût. Ainsi, nous éviterons d'utiliser l'inversion et nous réduirons au maximum le nombre de multiplications. L'élévation au carré et la multiplication par une petite constante n'influenceront que très peu les performances. Cependant, il est nécessaire de les considérer afin de départager d'éventuels systèmes de coordonnées nécessitant le même nombre de multiplications pour calculer une addition et un doublement.

Coordonnées affines

Le système de coordonnées affines peut être utilisé pour effectuer $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ ou $2(x_1, y_1) = (x_3, y_3)$. Les équations 2.16 et 2.17 utilisent le système de coordonnées affines et une addition de point s'effectue en 2I + 8M + 2S + 3m. Nous pouvons réduire ce calcul à 1I + 11M + 2S + 3m en utilisant la technique d'inversion de Montgomery [Mon87]. Le doublement peut être effectué en

1I + 2M + 4S + 2m et est donné par les équations suivantes :

$$x_3 = 1 + \frac{d_1 + d_2(x_1^2 + y_1^2) + y_1^2 + y_1^4}{d_1 + x_1^2 + y_1^2 + d_2 d_1^{-1}(x_1^4 + y_1^4)} \tag{2.18}$$

$$y_3 = 1 + \frac{d_1 + d_2(x_1^2 + y_1^2) + x_1^2 + x_1^4}{d_1 + x_1^2 + y_1^2 + d_2 d_1^{-1}(x_1^4 + y_1^4)} \tag{2.19}$$

Ce système de coordonnées nécessite plusieurs inversions (I) qui peuvent s'avérer très coûteuses en termes de performance.

Coordonnées projectives

Pour rappel le système de coordonnées projectives a pour but d'éviter le calcul d'inversion en utilisant une troisième coordonnée Z qui fait office de dénominateur. En effet, nous pouvons passer du système affine au système projectif par la transformation :

$$x = \frac{X}{Z}, \quad y = \frac{Y}{Z}$$

Un point P de la courbe E_{d_1,d_2} est alors représenté par trois coordonnées $(X : Y : Z)$. Nous pouvons alors effectuer $(X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2) = (X_3 : Y_3 : Z_3)$ ou $2(X_1 : Y_1 : Z_1) = (X_3 : Y_3 : Z_3)$ en évitant toutes inversions.

Algorithme 2.9 Addition de deux points P et Q en coordonnées projectives sur une courbe binaire d'Edwards.

Entrées $P = (X_1 : Y_1 : Z_1)$ et $Q = (X_2 : Y_2 : Z_2)$
Sorties: $R = (X_1 : Y_1 : Z_1) + (X_2 : Y_2 : Z_2)$

$A \leftarrow X_1 X_2$
$B \leftarrow Y_1 Y_2$
$C \leftarrow Z_1 Z_2$
$D \leftarrow d_1 Z_2$
$E \leftarrow C^2$
$F \leftarrow D^2$
$G \leftarrow (X_1 + Z_1)(X_2 + Z_2)$
$H \leftarrow (Y_1 + Z_1)(Y_2 + Z_2)$
$I \leftarrow A + G$
$J \leftarrow B + H$
$K \leftarrow (X_1 + Y_1)(X_2 + Y_2)$
$U \leftarrow C(F + d_1 K(K + I + J + C))$
$V \leftarrow U + DF + K(d_2(d_1 E + GH + AB) + (d_1 + d_2)IJ)$
$X_3 \leftarrow V + D(A + D)(G + D)$
$Y_3 \leftarrow V + D(B + D)(H + D)$
$Z_3 \leftarrow U + (d_1 + d_2)CK^2$
Retourner $R = (X_3 : Y_3 : Z_3)$

Les algorithmes 2.9 et 2.10 détaillent la manière d'additionner deux points et de doubler un point. Ainsi, dans le système projectif, nous pouvons effectuer une addition en 18M + 3S + 6m et 2M + 6S + 2m pour un doublement. Nous pouvons remarquer que dans le cas particulier où $d_1 = d_2$ les performances s'améliorent pour atteindre 16M + 2S + 4m pour l'addition et 2M + 6S +1m pour le doublement.

Algorithme 2.10 Doublement du point P en coordonnées projectives sur une courbe binaire d'Edwards.

Entrées $P = (X_1 : Y_1 : Z_1)$
Sorties: $R = 2(X_1 : Y_1 : Z_1)$

$A \leftarrow X_1^2$
$B \leftarrow A^2$
$C \leftarrow Y_1^2$
$D \leftarrow C^2$
$E \leftarrow Z_1^2$
$F \leftarrow E^2$
$G \leftarrow (d_2 d_1^{-1})(B + D)$
$H \leftarrow AE$
$I \leftarrow CE$
$J \leftarrow H + I$
$K \leftarrow G + d_2 J$
$Z_3 \leftarrow F + J + G$
$X_3 \leftarrow K + H + D$
$Y_3 \leftarrow K + I + B$
Retourner $R = (X_3 : Y_3 : Z_3)$

Coordonnées mixtes

Le système à coordonnées mixtes va à la fois utiliser les coordonnées projectives et affines. L'utilisation de ces deux systèmes de concert permet d'éviter une inversion lors d'une addition ou d'un doublement et aussi d'économiser un grand nombre de multiplications par rapport au système projectif. Cependant, lors d'une addition dans ce système le résultat $R = P + Q$ est sous forme projective. Ainsi, dans le cas du calcul d'une multiplication scalaire, il faut employer un algorithme qui va utiliser à chaque étape un point fixe qui sera représenté en coordonnées affines. Dans ce cas, ce système de coordonnées mixes ne pourra être utilisé qu'avec des algorithmes de type *double and add* (algorithmes 2.5, 2.7). L'algorithme 2.11 détaille les opérations pour l'addition de deux points dans un système de coordonnées mixtes sur une BEC.

Coordonnées différentielles

Les coordonnées différentielles regroupent un ensemble de systèmes de coordonnées : affines, projectifs, mixtes. L'idée de ces coordonnées est de changer la représentation d'un point à deux coordonnées (x, y) en un même point à une coordonnée $w = x + y$. Cependant, ce système de coordonnées impose une contrainte sur l'addition de points : pour pouvoir effectuer $R = P + Q$ en coordonnées différentielles nous devons connaître la différence $P - Q$. Cette contrainte limite grandement le choix des algorithmes de multiplications scalaires utilisables avec ce système de coordonnées. L'algorithme 2.8 de l'Échelle de Montgomery tire avantage de cette contrainte car tout au long du calcul la différence entre $R0$ et $R1$ est constante et égale à P, point de départ du calcul kP.

Dans le cas des BEC, la coordonnée différentielle d'un point $P = (x, y)$ notée $w(P)$, ou simplement w, est égale à $w(P) = x + y$. Dans [FH16], les auteurs présentent un nouveau système de coordonnées différentielles où $w(P) = \frac{x+y}{d_1(x+y+1)}$.

Algorithme 2.11 Addition de deux points $P = (X_1 : X_2 : Z_2)$ et $Q = (x_2, y_2)$ en coordonnées mixtes sur une courbe binaire d'Edwards.

Entrées $P = (X_1 : Y_1 : Z_1)$ et $Q = (x_2, y_2)$
Sorties: $R = (X_1 : Y_1 : Z_1) + (x_2, y_2)$
$W_1 \leftarrow X_1 + Y_1$
$w_2 \leftarrow x_2 + y_2$
$A \leftarrow x_2^2 + x_2$
$B \leftarrow y_2^2 + y_2$
$D \leftarrow W_1 Z_1$
$E \leftarrow d_1 Z_1^2$
$H \leftarrow (E + d_2 D) w_2$
$I \leftarrow d_1 Z_1$
$U \leftarrow E + AD$
$V \leftarrow E + BD$
$Z_3 \leftarrow UV$
$X_3 \leftarrow Z_3 y_2 + (H + X_1(I + A(Y_1 + Z_1)))V$
$Y_3 \leftarrow Z_3 x_2 + (H + Y_1(I + A(X_1 + Z_1)))V$
Retourner $R = (X_3 : Y_3 : Z_3)$

Cependant, ce système présente les mêmes performances que les coordonnées différentielles $w(P) = x + y$. Ainsi, nous concentrerons notre étude sur les coordonnées différentielles les plus simples : $w(P) = x + y$.

Dans la suite nous adopterons la notation suivante pour les coordonnées différentielles, ou w-coordonnées : $w(P)$ sera notée w_2 tandis que $w(Q)$ sera notée w_3, $w(P+Q) = w_5$ et $w(2P) = w_4$ et la différence $w(P-Q)$ sera quant à elle notée w_1.

Coordonnées différentielles affines : Les formules explicites pour calculer $w(P+Q)$ sont données par :

$$\begin{aligned} R &= w_2 w_3, \\ S &= R^2, \\ T &= R(1 + w_2 + w_3) + S, \\ w_5 &= T \frac{1}{d_1 + T + (d_2 d_1^{-1} + 1)S} + w_1 \end{aligned}$$

Ainsi, 1I+ 3M + 1S + 1m sont nécessaires pour effectuer une addition dans ce système de coordonnées. Dans le cas où $d_1 = d_2$, l'addition de point peut s'effectuer en 1I + 1M + 2S + 1m et est alors donnée par :

$$\begin{aligned} A &= w_2^2, \\ B &= A + w_2, \\ C &= w_3^2, \\ D &= C + w_3, \\ w_5 &= 1 + d_1 \frac{1}{d_1 + B + D} + w_1 \end{aligned}$$

Pour le doublement de point, nous avons :

$$\begin{aligned} A &= w_2^2, \\ J &= A^2, \\ K &= A + J, \\ w_4 &= K\frac{1}{d_1 + K + (d_2 d_1^{-1} + 1)J} \end{aligned}$$

1I + 1M + 2S + 1m sont alors nécessaire et dans le cas où $d_1 = d_2$ nous pouvons légèrement améliorer les performances :

$$\begin{aligned} A &= w_2^2, \\ B &= A + w_2, \\ w_4 &= 1 + d_1\frac{1}{d_1 + B^2} \end{aligned}$$

Il est a noté que certaines opérations sont communes entre l'addition et le doublement. Ainsi, lors d'une étape de l'Échelle de Montgomery certaines opérations peuvent être mutualisées, une addition et un doublement peuvent alors s'effectuer en 2I + 4M + 3S + 2m. De même, nous pouvons économiser une inversion grâce à la technique d'inversion de Montgomery [Mon87], mais qui va rajouter quelques multiplications : 1I + 7M + 3S + 2m. Dans le cas où $d_1 = d_2$ et en utilisant la technique d'inversion de Montgomery nous pouvons effectuer une addition et un doublement en 1I + 4M + 3S +2m. Malgré ces optimisations, les coordonnées différentielles affines restent coûteuses en terme de performance ce qui est dû à l'inversion.

Coordonnées différentielles projectives : De la même manière que dans le cas non-différentiel, nous pouvons représenter la coordonnée différentielle w d'un point P sous forme projective par $w = \frac{W}{Z}$. Nous obtenons alors des formules d'addition et de doublement ne nécessitant pas d'inversions. Supposons alors que w_1, w_2 et w_3 soient représentées sous forme projective par W_1/Z_1, W_2/Z_2 et W_3/Z_3. Alors, nous pouvons calculer w_4 et w_5 sous forme projective : W_4/Z_4 et W_5/Z_5. Les algorithmes 2.12 et 2.13 détaillent les différentes étapes de l'addition et du doublement en coordonnées différentielles projectives.

Algorithme 2.12 Addition de deux points sous forme différentielle $w(P) = W_2/Z_2$ et $w(Q) = W_3/Z_3$ en coordonnées projectives sur une courbe binaire d'Edwards. Nous avons que $w(P - Q) = W_1/Z_1$.

Entrées $w(P) = W_2/Z_2$, $w(Q) = W_3/Z_3$ et $w(P - Q) = W_1/Z_1$
Sorties: $w(R) = w(P) + w(Q)$
$A \leftarrow W_2 W_3$
$B \leftarrow Z_2 Z_3$
$C \leftarrow (W_2 + Z_2)(W_3 + Z_3)$
$W_5 \leftarrow Z_1(d_1(C + A + B)^2)$
$Z_5 \leftarrow W_1(AC + (\sqrt{d_1}B + \sqrt{d_2 d_1^{-1} + 1}A)^2)$
Retourner $w(R) = W_5/Z_5$

Ainsi, dans ce système de coordonnées différentielles nous pouvons effectuer une addition en 6M + 2S + 3m et si $d_1 = d_2$, il nous faut effectuer 6M + 2S + 2m. Le

Algorithme 2.13 Doublement d'un point sous forme différentielle $w(P) = W_2/Z_2$ en coordonnées projectives sur une courbe binaire d'Edwards.

Entrées $w(P) = W_2/Z_2$
Sorties: $w(R) = 2w(P)$
$C \leftarrow W_2(W_2 + Z_2)$
$W_4 \leftarrow C^2$
$Z_5 \leftarrow W4 + ((\sqrt[4]{d_1}Z_2 + \sqrt[4]{d_2 d_1^{-1} + 1}W_2)^2)^2$
Retourner $w(R) = W_4/Z_4$

doublement est particulièrement efficace et ne nécessite que 1M + 3S + 3m. Nous pouvons économiser une multiplication par une constante (m) si $d_1 = d_2$.

Coordonnées différentielles mixtes : comme pour le cas non-différentiel, nous pouvons user des avantages des coordonnées affines et projectives différentielles en les combinant. La différence avec le cas classique est que la coordonnée sous forme affine sera la coordonnée w de $P - Q$. Dans le cas d'une utilisation avec l'Échelle de Montgomery $P-Q$ est un point fixe, ce système de coordonnées est donc compatible avec un tel algorithme de multiplication scalaire. La coordonnée $w(P - Q)$ est alors donnée par w_1.

Le doublement est identique à celui du système de coordonnées différentielles projectives. Dans le cas de l'addition, l'algorithme 2.14 donne les détails des différentes étapes.

Algorithme 2.14 Addition de deux points sous la forme différentielle $w(P) = W_2/Z_2$ et $w(Q) = W_3/Z_3$ en coordonnées mixtes sur une courbe binaire d'Edwards. Nous avons que $w(P - Q) = w_1$.

Entrées $w(P) = W_2/Z_2$, $w(Q) = W_3/Z_3$ et $w(P - Q) = w_1$
Sorties: $w(R) = w(P) + w(Q)$
$C \leftarrow W_2(W_2 + Z_2)$
$D \leftarrow W_3(W_3 + Z_3)$
$E \leftarrow Z_2 Z_3$
$F \leftarrow W_2 W_3$
$V \leftarrow CD$
$U \leftarrow V + (\sqrt{d_1}E + \sqrt{d_2 d_1^{-1} + 1}F)^2$
$W_5 \leftarrow V + w_1 U$
$Z_5 \leftarrow U$
Retourner $w(R) = W_5/Z_5$

Ainsi, une addition peut s'effectuer en 6M + 1S + 2m et si $d_1 = d_2$, alors 5M + 1S + 1m sont nécessaires. Les mutualisations possibles d'opérations en coordonnées différentielles mixtes permettent d'effectuer une étape de l'Échelle de Montgomery en 5M + 4S + 2m.

Nous pouvons encore améliorer ces formules dans le cas où $d_1 = d_2$. En effet, comme expliqué dans [KLN14, KAK15], il est alors possible de modifier les formules de telle sorte que P et Q aient la même coordonnée Z en représentation différentielle projective. Il est alors possible de calculer simultanément $w(2P) = W_4/Z'$ et $w(P + Q) = W5/Z'$ étant donné $w(P) = W_2/Z$, $w(Q) = W_3/Z$ et $w(P - Q) = w_1$. Ce

calcul correspond exactement à une étape de l'Échelle de Montgomery. De plus, le fait d'avoir une coordonnée commune permet d'économiser de l'espace mémoire. Dans le cadre d'une implémentation logiciel sur un microcontrôleur, ceci n'a que peu d'influence, alors que dans le cas d'une implémentation matérielle, ce résultat peut permettre d'alléger les ressources matérielles notamment en taille de RAM nécessaire. L'algorithme 2.15 donne les détails des différentes opérations à effectuer.

Algorithme 2.15 Addition et doublement de deux points sous forme différentielle $w(P) = W_2/Z$ et $w(Q) = W_3/Z$ en coordonnées mixtes sur une courbe binaire d'Edwards. Nous avons que $\frac{1}{w(P-Q)} = \frac{1}{w_1}$.

Entrées $w(P) = W_2/Z$, $w(Q) = W_3/Z$ et $w(P-Q) = w_1$
Sorties: $w(2P)$ et $w(P+Q)$
$C \leftarrow (W_2 + W_3)^2$
$D \leftarrow Z^2$
$E \leftarrow \frac{1}{w_1} C$
$U \leftarrow E + C$
$V \leftarrow E + D$
$S \leftarrow (W_2(W_2 + Z))^2$
$T \leftarrow S + d_1 D^2$
$W_5 \leftarrow UT$
$W_4 \leftarrow VS$
$Z' \leftarrow VT$
Retourner $w(2P) = W_4/Z'$ et $w(P+Q) = W_5/Z'$

Il est important de noter que dans ce cas précis la loi de groupe perd sa complétude. La constante $1/w_1$ implique que w_1 ne soit pas nul. Ainsi, la différence $P - Q$ doit être différente du point $(0, 0)$ et du point $(1, 1)$. Si $P - Q = (0, 0)$, alors $P = Q$ et dans ce cas la somme $P + Q$ revient à effectuer un doublement. Si $P - Q = (1, 1)$, alors $P = Q + (1, 1)$ ce qui implique que $x_p = x_q + 1$ et $y_p = y_q + 1$ et donc $w(P) = w(Q)$ et dans ce cas cela revient toujours à effectuer un doublement au lieu d'une addition. Ce résultat nous montre que plusieurs points différents ont la même représentation différentielle. Nous verrons que ceci peut être problématique pour l'intégration des BEC au sein de protocoles de signatures tel que l'ECDSA (cf. le chapitre 4).

Retrouver les coordonnées x et y à partir de la coordonnée w : la représentation différentielle permet d'effectuer une addition de point et un doublement d'un point de manière efficace et ainsi permettre un calcul rapide de kP. Cependant, une fois la multiplication scalaire effectuée, le résultat est donné en coordonnées différentielles. Il faut alors retrouver les coordonnées affines qui serviront par la suite au sein de protocoles cryptographiques tel que la signature ECDSA. Nous pouvons retrouver les coordonnées affines de kP à partir de $w(kP)$, $w(kP + P)$ et $w(P)$. L'Échelle de Montgomery a la particularité de calculer à la fois kP et $kP + P$ ce qui nous permet de retrouver les coordonnées affines de kP en résolvant l'équation :

$$x_2^2 + x_2 = \frac{w_3(d_1 + w_0 w_2(1 + w_0 + w_2) + d_2 d_1^{-1} w_1^2 w_2^2) + d_1(w_1 + w_2) + (y_1^2 + y_1)(w_2^2 + w_2)}{w_1^2 + w_1} \tag{2.20}$$

Système	Addition	Doublement	Addition + Doublement
Affine	1I + 11M + 2S + 3m	1I + 2M + 4S + 2m	2I + 13M + 6S + 5m
Affine, $d_1 = d_2$	1I + 11M + 2S + 3m	1I + 2M + 4S + 1m	2I + 13M + 6S + 4m
Projectif	18M + 3S + 6m	2M + 6S + 2m	20M + 9S + 8m
Projectif, $d_1 = d_2$	16M + 2S + 4m	2M + 6S + 1m	18M + 8S + 5m
Mixte	13M + 3S + 3m	2M + 6S + 2m	15M + 9S + 5m
Mixte, $d_1 = d_2$	13M + 3S + 3m	2M + 6S + 1m	15M + 9S + 4m
Diff. affine	1I + 3M + 1S + 1m	1I + 1M + 2S + 1m	1I + 7M + 3S + 2m
Diff. affine, $d_1 = d_2$	1I + 1M + 2S + 1m	1I + 2S + 1m	1I + 4M + 3S + 2m
Diff. projectif	6M + 2S + 3m	1M + 3S + 3m	7M + 5S + 6m
Diff. projectif, $d_1 = d_2$	6M + 2S + 2m	1M + 3S + 2m	7M + 5S + 4m
Diff. mixte	5M + 1S + 1m	1M + 3S + 3m	5M + 4S + 4m
Diff. mixte, $d_1 = d_2$	5M + 1S + 1m	1M + 3S + 3m	5M + 4S + 1m

TABLE 2.4 – Performances des différents systèmes de coordonnées pour le modèle de courbes binaires d'Edwards.

Où $w(P) = w((x_1, y_1)) = w_1$, $w(kP) = w_2$ et $w(kP + P) = w_3$. Pour résoudre cette équation nous pouvons utiliser le calcul de la demi-trace, donné par les équations 2.2 et 2.3, qui nous permettrons d'obtenir x_2 ou $x_2 + 1$. Ainsi, les coordonnées affines trouvées sont (x_2, y_2) ou $(x_2 + 1, y_2 + 1) = (x_2, y_2) + (1, 1)$. Il est implicite que pour pouvoir retrouver les coordonnées affines de kP le dénominateur ne peut être nul donc $w_1^2 + w_1 \neq 0$ ce qui est possible uniquement si $4(kP) = (0, 0)$.

Synthèse des différents systèmes de coordonnées

Nous avons détaillé les différents systèmes de coordonnées relatifs au modèle de courbes binaires d'Edwards. Le tableau 2.4 résume les performances pour tous les systèmes de coordonnées présentés pour le modèle alternative BEC. De cette analyse nous pouvons conclure que le cas $d_1 = d_2$ permet d'économiser quelques opérations, ce gain est maximal dans le cas de coordonnées différentielles mixtes.

Cependant, il faut mettre en perspective ces performances au regard de l'algorithme de multiplication scalaire associée. Par exemple, le système de coordonnées différentielles mixtes est applicable si nous utilisons l'Échelle de Montgomery (algorithme 2.8) qui, *a priori*, est moins efficace qu'un algorithme de fenêtres glissantes tel que l'algorithme 2.6. Ainsi, un autre système de coordonnées, compatible avec l'algorithme de fenêtre glissante, aurait de meilleures performances.

Nous pouvons mesurer les performances de chaque algorithme pour calculer l'opération kP, pour une taille de clef k de 256 bits, ce qui correspond aux recommandations actuelles (cf. le tableau 1.1), nous avons :

- ***Double and Add*** : L'algorithme 2.5 est le plus efficace avec le système de coordonnées mixtes, combinant les coordonnées projectives et affines. Cependant, ce dernier va effectuer le calcul kP bit à bit ce qui va grandement ralentir les performances d'un tel algorithme. Ainsi, pour une clef de 256 bits, il nous faudra effectuer 256 doublements et un nombre d'additions égal au nombre de bits à 1 dans k. Dans le cas où k est tirée aléatoirement de façon uniforme, le nombre de bits à 1 et à 0 sont proches. Ainsi, en moyenne l'algorithme 2.5 effectuera 128 additions de points, l'opération kP nécessitera en moyenne : **2176M + 1920S + 640m**. Dans le cas où l'on utilise l'algorithme 2.7 qui prémunit l'algorithme 2.5 contre certaines attaques physiques en ajoutant une addition

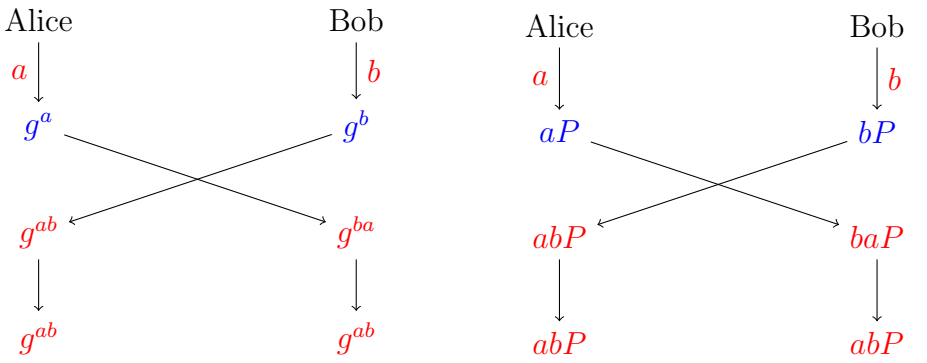
Taille de w	Performances	Tailles des précalculs
2	2176M + 1920S + 640m	2048
3	1630M + 1794S + 514m	4096
4	1344M + 1728S + 448m	8192
5	1174M + 1695S + 407m	16 384
6	1092M + 1692S + 392m	32 768
7	1014M + 1674S + 374S	65 536
8	928M + 1632S + 352S	131 072

TABLE 2.5 – Performances et tailles des précalculs pour un calcul de kP par fenêtres glissantes. Les tailles sont données en bits.

factice quand nécessaire, nous obtenons comme performances : **3840M + 2304S + 1024m**.

- **Fenêtres glissantes** : L'algorithme 2.6 devra utiliser les coordonnées mixtes pour être le plus efficace. Cette méthode de calcul de multiplication scalaire parcourt la clef k à l'aide d'une fenêtre glissante d'une taille w. L'algorithme effectuera à chaque pas de la fenêtre, w doublements et une addition de points avec un point pré-calculé. La taille des pré-calculs dépend alors directement de la taille w. Il est possible d'améliorer les performances de cet algorithme en remplaçant les séries de bits à 0 de la clef par une série de doublements au lieu d'une addition. Cependant, une telle optimisation rend le temps d'exécution de ce dernier directement dépendant de la clef secrète ce qui peut être un biais d'attaques par canaux auxiliaires, cf. chapitre 3. Ainsi, une telle solution n'est pas envisagée dans le cas d'une implémentation pour des applications IoT sensibles. De plus, nous excluons de cette mesure les pré-calculs qui peuvent être effectués préalablement lorsque le point P est un point fixe, ce qui est le cas pour le protocole de signature ECDSA. Les performances de cet algorithme sont données par le tableau 2.5 en fonction de la taille de w. La taille en bit des pré-calculs sont aussi exprimés en fonction de w.
- **Échelle de Montgomery** : Dans le cas de l'algorithme 2.8 nous pouvons utiliser les coordonnées différentielles mixtes comme expliqué précédemment. Le gain obtenu par ce système de coordonnées permet de pallier au fait que les calculs sont effectués bit à bit comme pour l'algorithme 2.5 ou 2.7. De plus, cette méthode de calcul de kP est uniforme, *i.e.*, les mêmes opérations sont effectuées pour tous les bits de la clef. Dans ces conditions, nous obtenons comme performance : **1280M + 1024S + 256m**. Nous verrons par la suite, cf. 4.3.3, que nous pouvons améliorer ces performances dans le cas où P est un point fixe en choisissant judicieusement le point P. Cette optimisation nous permettra d'échanger une multiplication longue par une multiplication courte à chaque étape : **1024M + 1024S + 512m**.

L'algorithme à fenêtre glissante de taille supérieure à 6 semble être le plus performant, cependant la taille des précalculs obtenus associée à une résistance faible aux attaques physiques ne permet pas son utilisation dans le cas d'un environnement IoT contraint. Au regard des différents résultats obtenus, l'Échelle de Montgomery est le meilleur compromis entre performance et sécurité.



(a) Dans un groupe fini G.

(b) Sur une courbe elliptique E.

FIGURE 2.6 – Protocole d'échange de clefs Diffie-Hellman. Les éléments en rouge sont privés, en bleu publics.

2.5 Protocoles cryptographiques sur les courbes elliptiques

L'utilisation des outils mathématiques tels que les courbes elliptiques, à des fins cryptographiques n'ont de sens que s'ils sont intégrés à des protocoles sécurisés. Les calculs de la cryptographie asymétrique sont moins performants comparé à la cryptographie symétrique. Ainsi, nous utilisons généralement la cryptographie asymétrique pour établir un canal sécurisé entre deux entités, Alice et Bob, qui par suite utiliseront la cryptographie symétrique pour s'échanger des messages. Un canal est considéré sécurisé s'il y eu authentification mutuelle ou non entre Alice et Bob, et construction de clefs de sessions qui seront utilisées par la cryptographie symétrique pour échanger des messages.

2.5.1 Protocoles d'échange de clefs

Les protocoles d'échange de clefs ont pour objectif l'établissement d'un secret partagé entre Alice et Bob sur un canal non-sécurisé. Celui-ci pourra être utilisé pour dériver des clefs de sessions qui serviront par suite à l'établissement d'une communication sécurisé entre ces deux entités.

Le standard NIST SP 800-57 [Bar16] donne les détails sur les différents protocoles d'échanges de clefs possibles. Ce standard repose essentiellement sur les travaux de Diffie et Hellman [DH76a, DH76b] qui ont construit un protocole d'échange de clefs basé sur le problème mathématique du logarithme discret dans un corps fini. Ce protocole peut s'appliquer à tous les groupes finis où le problème du logarithme discret est difficilement résolvable.

Alice et Bob ont respectivement une paire de clefs publiques/privées (a, g^a) et (b, g^b) où g est un élément d'un groupe fini prédéfini. Les clefs privées a et b sont des scalaires et les clefs publiques g^a et g^b sont des éléments du groupe. Le protocole de Diffie-Hellman (DH) est présenté par la figure 2.6a.

Notons que ce protocole est sensible aux attaques dites *Man-In-The-Middle* (MITN) qui consistent à usurper l'identité d'Alice ou Bob en interceptant g^a ou g^b qui sont envoyés en clair. Pour se prémunir de ce genre d'attaque, il est important d'authentifier l'émetteur de g^a ou g^b à l'aide d'un protocole de signature.

Dans le cas où le groupe fini G est le groupe formé par les points d'une courbe elliptique E, nous pouvons effectuer le même protocole dont les opérations seront notées additivement tel que détaillé sur la Figure 2.6b. Le protocole DH est alors appelé ECDH pour *Elliptic Curve Diffie-Hellman* où pour lequel le secret partagé entre Alice et Bob est constitué des coordonnées du point abP.

2.5.2 Protocoles de signatures

Les protocoles de signatures sont utilisés pour pouvoir authentifier une entité, garantir l'intégrité d'une information ou garantir la non-répudiation d'une information. Ainsi, une signature cryptographique est, *mutatis mutandis*, identique à la signature manuscrite que nous pouvons trouver sur un document papier. Un protocole de signature est divisé en deux parties :

- **Signature** : la première partie consiste en la construction d'une signature à partir d'une clef privée et du message à signer. Cette opération est sensible, car les calculs effectués manipulent des données secrètes telle que la clef privée de l'entité signataire. Si cette clef fuit, un attaquant pourra alors forger des signatures à la place de l'entité légitime.
- **Vérification** : la deuxième consiste à vérifier qu'une signature s associée à un message m est valide, *i.e.* valider que s est lié mathématiquement à m et à la clef publique associée à la clef privée qui a servi à signer m. Cette opération ne manipule que des données publiques, ainsi, la vérification d'une signature ne sera pas sensible à des attaques physiques, car la clef privée n'est pas utilisée.

Étant donnée une courbe elliptique E et un point P d'ordre n sur cette dernière, nous pouvons construire la paire de clefs privée/publique composée d'un nombre aléatoire a pour sa partie privée et du point aP pour sa partie publique. Cette paire de clefs est de la même forme pour tous les protocoles de signatures sur les courbes elliptiques que nous considérerons dans le reste de cette étude.

Il existe plusieurs protocoles de signature utilisant les courbes elliptiques tels que l'*Elliptic Curve Digital Signature Algorithm* (ECDSA) [Gal13] ou l'*Edwards Digital Signature Algoritm* (EdDSA) [BDL+11a]. Ces protocoles sont standardisés par le NIST [Gal13] pour l'ECDSA ou par une RFC [JL17] dans le cas de l'EdDSA.

Ces protocoles de signatures dérivent du protocole de signature de Schnorr [Sch90] qui a fait l'objet d'un brevet qui est passé dans le domaine public en 2008. Une signature construite à partir de ce protocole est un couple (R, s) et est générée à partir de :

- E : les paramètres définissant la courbe elliptique utilisée.
- P : un point de la courbe de grand ordre.
- n : l'ordre du point P
- $hash$: une fonction de hachage. Ces fonctions ne font pas l'objet de cette étude et ne seront donc pas décrites.

Une signature d'un message m est alors calculée à l'aide d'une clef privée a :

$$\begin{aligned} R &= rP \\ s &= (r + hash(R, m)a)[n] \end{aligned}$$

La vérification de la signature (R, s) du message m s'effectue à partir de la clef publique Q associée à la clef privée a.

Ainsi, nous devons calculer $R' = sP - hash(R, m)Q$ et vérifier que :

$$hash(R', m) = hash(R, m)$$

Ce protocole est correct car :

$$\begin{aligned} R' &= sP - hash(R, m)Q \\ &= sP - hash(R, m)aP \\ &= (r + hash(R, m)a)P - hash(R, m)aP \\ &= rP \\ &= R \end{aligned}$$

L'ECDSA a l'avantage d'offrir des signatures aléatoires, *i.e.* que pour un même message m et une même paire de clefs publique/privée le calcul d'une signature ECDSA donnera deux signatures différentes mais valides. Les algorithmes 2.16, 2.17 de signature et vérification vont prendre en paramètre le même ensemble de données publiques que le protocole de Schnorr :

Algorithme 2.16 Signature ECDSA.

Entrées une clef privée a, un message m
Sorties: (r, s) la signature associée à m

```
H(m) ← hash(m)
Faire
    Faire
        k ← un aléa entre 1 et n − 1
        (x, y) ← kP
    Tant que x = 0
    r ← x[n]
    s ← k^{-1}(H(m) + ar)[n]
Tant que s = 0
Retourner (r, s)
```

Algorithme 2.17 Vérification ECDSA.

Entrées m un message, (r, s) la signature associée, Q la clef publique associée à la clef privée qui a servi à signer le message.
Sorties: Retourne vrai si la signature est correcte, faux sinon

```
H(m) ← hash(m)
u = H(m)s^{-1}[n]
v = rs^{-}1[n]
(x, y) = uP + vQ
Si x ≠ r alors
    Retourner Faux
Fin Si
Retourner Vrai
```

Nous pouvons montrer qu'une signature produite par l'algorithme 2.16 sera bien validée par l'algorithme 2.17.

$$\begin{aligned} uP + vQ &= H(m)s^{-1}G + rs^{-1}Q \\ &= H(m)s^{-1}G + rs^{-1}aG \\ &= (H(m) + ar)s^{-1}G \\ &= (H(m) + ar)k(H(m) + ar)^{-1}G \\ &= kG \\ &= (x, y) \end{aligned}$$

L'EdDSA quant à lui est un protocole de signature publié en 2011 par Bernstein *et al.*, [BDL+11b, BJL+15], il a pour vocation initiale pour vocation d'être utilisé avec la courbe elliptique Curve25519, mais peut très bien être utilisé avec tout autre courbe elliptique. Les algorithmes 2.18 et 2.19 présentent respectivement la signature et la vérification de signatures effectuées par ce protocole.

Algorithme 2.18 Signature EdDSA.

Entrées une clef privée a, la clef publique associée Q, un message m
Sorties: (R, s) la signature associée à m
$H(a, m) \leftarrow hash(a, m)$
$R \leftarrow H(a, m)P$
$H(R, Q, m) \leftarrow hash(R, Q, m)$
$s \leftarrow (H(a, m) + H(R, Q, m)a)[n]$
Retourner (R, s)

Algorithme 2.19 Vérification EdDSA.

Entrées m un message, (R, s) la signature associée, Q la clef publique associée à la clef privée qui a servi à signer le message.
Sorties: Retourne vrai si la signature est correcte, faux sinon
$H(R, Q, m) \leftarrow hash(R, Q, m)$
$U \leftarrow 8sP$
$V \leftarrow 8R + 8H(R, Q, m)Q$
Si $U \neq V$ **alors**
    Retourner Faux
Fin Si
Retourner Vrai

De la même façon, ce protocole de signature est correct car :

$$\begin{aligned} R + H(R, Q, m)Q &= H(a, m)P + H(R, Q, m)aP \\ &= (H(a, m) + H(R, Q, m)a)P \\ &= sP \end{aligned}$$

La grande différence entre l'ECDSA et l'EdDSA est le caractère entièrement déterministe de la signature EdDSA alors que l'ECDSA produit des signatures aléatoires. Cette différence peut conduire à des attaques physiques sur l'EdDSA ce qui

n'est pas le cas pour l'ECDSA. La signature EdDSA étant déterministe, nous pourrons la recalculer autant de fois que désiré le même calcul, les attaques physiques nécessitant un grand nombre de mesures pourront alors être menées. Les auteurs de ce protocole justifient le choix du déterminisme par le fait qu'il est difficile de tirer un nombre aléatoire et que si ce nombre n'a pas une entropie suffisante, alors des attaques peuvent être menées contre les protocoles de signatures comme l'ECDSA. Cependant, le nombre aléatoire k utilisé pour le calcule kP de la signature ECDSA peut être haché afin d'éviter une entropie plus faible. Ceci est identique au cas de la clef privée d'une système cryptographique basé sur les courbes elliptiques qui est généralement hachée pour éviter un manque d'entropie dans cette dernière.

2.5.3 Sécurité des protocoles

Les différents protocoles que nous avons vus dans cette section reposent tous sur la résolution de l'ECDLP. Ce problème mathématique se définit comme suit :

Définition 2.26 (ECDLP). *Soit E une courbe elliptique définie sur un corps fini $\mathbb{F}_{p^d}$ et soient P et Q deux points de E, alors il est difficile de trouver a, s'il existe, tel que $Q = aP$.*

Actuellement, les meilleurs algorithmes connus pour résoudre ce problème sont les méthodes ρ-Pollard [Pol75] et *Baby-Step Giant-Step* [Sha71] (BSBG). Ces techniques reposent sur le paradoxe des anniversaires afin de trouver une collision au sein d'un ensemble de calculs de kP, où k varie. Le paradoxe des anniversaires donne la complexité de telles méthodes et peut être formulé comme suit :

Théorème 2.27 (Paradoxe des anniversaires). *Soit E un ensemble fini. La probabilité $P(n)$ que pour n éléments de E tirés uniformément, deux éléments au moins soient identique est égale à :*

$$P(n) = 1 - \frac{|E|!}{([E| - n)!\,|E|^n}\frac{1}{}$$

Lorsque le cardinal de E est grand il est difficile de calculer exactement cette probabilité. Ainsi, nous pouvons l'approximer par :

$$P(n) \approx 1 - e^{\frac{n(n-1)}{2|E|}}$$

Si l'on veut connaître une approximation de n en fonction de la probabilité P :

$$n(P) \approx \sqrt{2|E| \ln\left(\frac{1}{1-P}\right)}$$

Ainsi, cette dernière approximation nous permet de calculer la taille minimale des clefs privées et donc du cardinal de la courbe elliptique. Il s'en suit que pour assurer les 128 bits de sécurité recommandés, cf. tableau 1.1, il nous faut considérer des clefs privées de 256 bits et ainsi des courbes elliptiques dont le cardinal est de l'ordre de 2^{256}. En effet, nous considérons qu'un système cryptographique asymétrique est sécurisé si la probabilité P de collision est proche de 0, donc le terme $\sqrt{\ln\left(\frac{1}{1-P}\right)}$ est proche de 1 et ainsi négligeable dans l'approximation de $n(P)$. De plus, les recommandations de sécurité estiment que $n(P)$ doit être supérieur à 2^{128}, ce qui nous permet de calculer une borne majorée du cardinal de E pour un tel niveau de sécurité : $n(P) \approx \sqrt{|E|}$.

Calcul d'index

Il existe une autre catégorie d'algorithmes permettant de résoudre l'ECDLP : les méthodes par calcul d'index. Ces méthodes ont été développées initialement par Odlyzko [Odl84]. La stratégie adoptée est différente de celle des méthodes de type ρ-Pollard ou BSBG, l'idée est de construire directement une solution, a, à l'équation $Q = aP$.

Cette méthode est divisée en deux parties. La première partie consiste en la construction d'un grand nombre de relations linéaires sur une « base » de points de la courbe E. La seconde partie est la résolution de ce système linéaire dont a serait une solution. En détail la méthode du calcul d'index se construit comme suit :

1. Définir une base $\mathcal{F}$ du groupe de la courbe elliptique E appelée : base de facteurs.
2. Collecter des relations linéaires sur cette base :
   (a) Prendre des nombres aléatoires x et y et calculer : $R = xQ + yP$
   (b) Essayer de décomposer R sur $\mathcal{F}$
   (c) En cas de succès : $xQ + yP = \sum_{G_i \in \mathcal{F}} e_i G_i$ est une relation linéaire et est sauvegardée.
   (d) Répéter ces étapes jusqu'à avoir au moins autant de relations qu'il y a de points dans $\mathcal{F}$
3. Résoudre le système linéaire alors construit afin de trouver une relation $\lambda Q + \mu P = 0$
4. Si λ est inversible alors on a $a = \frac{\mu}{\lambda}$

Malgré tout, cette méthode est difficile à mettre en place due à la complexité de la construction de relations linéaires et à la résolution d'un système linéaire large. Dans [GG16], les auteurs reviennent en détails sur les méthodes du calcul d'index et leurs avancées et concluent que cette stratégie n'atteint pas les performances des méthodes aléatoires (ρ-Pollard et BSBG) et ce, même dans le cas où la courbe elliptique E est définie sur un corps fini de caractéristique 2.

En 2019, le record de la résolution de l'ECDLP est établi par à Bernstein *et al.* datant de 2016 n'a pas été battu. La courbe utilisée est définie sur le corps fini $\mathbb{F}_{2^{127}}$, la clef privée a a quant à elle une taille de 117 bits. Notons que cette résolution a nécessité une implémentation dédiée sur FPGA et 6 mois de calculs sur un ensemble de 64 à 576 FPGA. Dans le cas particulier de courbes elliptiques de Barreto-Naehrig [BN05], le record de résolution quant à lui date de 2017 et est dû à Kusaka *et al.* [KJI+18]. Dans ce cas, la taille de la courbe utilisée est de 114 bits, et 200 CPU travaillant en continu pendant 6 mois ont été nécessaire pour en venir à bout. Ces deux résolutions de l'ECDLP utilisent l'algorithme ρ-Pollard.

Chapitre 3

Attaques physiques contre la cryptographie sur les courbes elliptiques

L'invincibilité se trouve dans la défense, la possibilité de victoire dans l'attaque.

Sun Tzu, *L'Art de la guerre*

Sommaire

Dans ce chapitre, nous abordons les attaques par canaux auxiliaires dans le cas de la cryptographie sur les courbes elliptiques. Ces attaques sont basées sur les manipulations de l'objet physique exécutant un algorithme cryptographique manipulant des données secrètes telles qu'une clef privée. Ces mesures physiques peuvent laisser fuir de l'information sur les clefs utilisées. Ainsi, un attaquant qui observerait ces canaux auxiliaires serait à même de pouvoir retrouver une clef secrète sans attaquer la sécurité mathématique de l'algorithme cryptographique. Cependant, ce type d'attaques physiques n'est pas le seul moyen pour arriver à retrouver une clef secrète lors de l'exécution d'un calcul cryptographique. L'injection de fautes lors d'une opération de chiffrement peut permettre de retrouver tout ou partie de la clef utilisée.

Dans ce chapitre, nous présenterons les principes généraux relatifs aux attaques physiques. Nous donnerons un état de l'art exhaustif des attaques par canaux auxiliaires et par fautes sur les implémentations cryptographiques basées sur les courbes elliptiques. Nous conclurons ce chapitre par un exposé des contremesures existantes permettant de se prémunir de ce genre d'attaques.

3.1 Les différentes attaques physiques

À l'heure du tout connecté, les circuits intégrés sont omniprésents. Nous les retrouvons aussi bien dans les appareils communicants basiques tel qu'un badge d'accès, que dans les appareils communicants plus complexes tels que les ordinateurs. Quelle que soit leur complexité, ces circuits intégrés sont basés sur la même brique élémentaire : le transistor. Ces transistors sont conçus en technologie *Complementary Metal-Oxide-Semiconductor* ou CMOS.

Les transistors CMOS sont associés les uns aux autres afin de construire des fonctionnalités plus complexes telles que des portes logiques ou des cellules mémoires permettant de stocker des données. Ainsi, la charge/décharge des transistors dans une porte logique induit un changement d'état. Il s'en suit que pour passer d'un bit à 0 vers un bit à 1, une capacité est chargée. À l'inverse, le passage de 1 à 0 est dû à la décharge d'une capacité. Par exemple, dans la Figure 3.1, cela revient à piloter grâce à V_{in} la connexion de la capacité C_l à V_{cc} ce qui implique une consommation de courant (passage de 0 à 1) tandis que connecter cette même capacité à la masse la décharge et n'induit pas de consommation de courant (passage de 1 à 0). Le passage d'un état 1-1 ou 0-0 n'engendre aucun changement d'état de la capacité, il en résulte aucune consommation ni décharge de la capacité.

Plusieurs moyens sont alors disponibles pour extraire de l'information basée sur l'état des transistors.

- **Attaques invasives** : ce type d'attaques consiste à analyser et modifier physiquement un circuit intégré [AK97]. Ces attaques permettent, par exemple, de reproduire la fonctionnalité de tout ou partie du circuit intégré. Pour ce faire, il faut retrouver chaque élément qui compose le circuit (CPU, mémoires, lignes de bus etc.) ce qui nécessite très souvent de retirer chimiquement ou par plasma toutes les couches du circuit à l'aide d'équipements spécifiques d'imagerie, il est alors possible d'identifier les différents blocs voire de reconstituer le layout du circuit.
- **Attaques passives** : à l'inverse des attaques invasives, ces attaques ne nécessitent pas une destruction du circuit, elles exploitent les différences observées

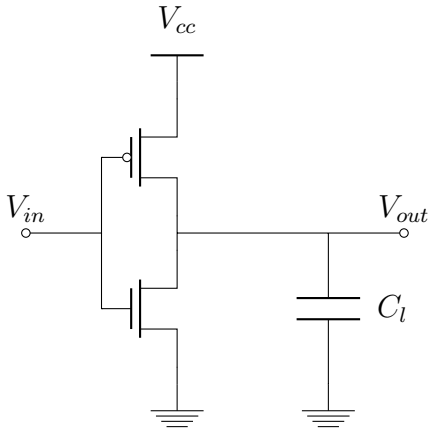


FIGURE 3.1 – Inverseur CMOS

lors des mesures de caractéristiques physiques. Ces mesures peuvent être la consommation de courant [KJJ99], le rayonnement électromagnétique [QS01], le temps de calcul [Koc96] ou tout autre paramètre évoluant en fonction du changement d'état des transistors. Nous pouvons par ce biais là effectuer une analyse de rétro-ingénierie logicielle afin de retrouver, par exemple, des tables de substitutions d'algorithme cryptographique [Nov03, Cla04].

Dans la suite de cette thèse, nous nous concentrerons sur les attaques passives, aussi appelées attaques par canaux auxiliaires, en vue d'extraire des clefs secrètes dans des applications cryptographiques, sur courbes elliptiques.

- **Attaques semi-invasives** : ce troisième type d'attaques appelées injections de fautes consiste à modifier le comportement d'un composant sans pour autant détruire ce dernier. Ainsi, en modifiant le comportement d'un composant nous pouvons obtenir de l'information sur le secret manipulé. Ce type d'attaques contre des implémentations cryptographiques a été introduit par Boneh *et al.* [BDL01], l'attaque présentée ciblait une implémentation de RSA. Nous pouvons générer une faute de différentes manières : par rayonnements électromagnétiques [GMO01], par lumière [SA03], par perturbation de l'horloge ou de la tension [AK97]. Dans cette thèse, nous ne traiterons que de l'aspect théorique de l'influence de ces attaques sur la cryptographie basée sur les courbes elliptiques.

Classification des attaques : Afin de faire un état de l'art exhaustif des attaques physiques contre les implémentations cryptographiques sur les courbes elliptiques, nous pouvons nous référer à plusieurs publications [FGM+10, FV12, DGH+13, Mur14, ACL19] qui synthétisent toutes les attaques existantes. Nous nous conformerons à la classification suivante :

- **Attaques par observations** : aussi appelée *Side Channel Analysis* (SCA), exploite une mesure physique telle que la consommation de courant, le rayonnement électromagnétique ou le temps de calcul, via une analyse statistique qui dépend de l'attaque menée et de l'implémentation visée.
  - **Attaques temporelles** : cette sous-catégorie d'attaques consiste en l'exploitation d'un temps de calcul des opérations cryptographiques qui dépendrait de la clef secrète.

  - **Attaques simples** : cet ensemble d'attaques tente de mettre en avant des faiblesses simples et immédiates dans les calculs sur les ECC.
  - **Attaques verticales** : cette sous-famille va utiliser plusieurs traces d'une mesure physique afin de mettre en évidence soit des collisions entre les calculs, soit des corrélations au sein du calcul.
  - **Attaques horizontales** : tout comme pour les attaques verticales, celles-ci s'appuient des collisions ou des corrélations au sein d'une seule et même trace d'une mesure physique.
  - **Attaques par profilage** : les attaques par profilage regroupent toutes les attaques élaborées en deux phases : une première d'apprentissage essayant de caractériser des fuites à partir d'un ensemble de traces et une seconde d'attaque à proprement dite qui consiste en l'application de l'apprentissage (ou *Template*) à des traces dont la clef utilisée est inconnue.
- **Attaques par injections de fautes** : Cette famille d'attaques actives consiste en l'injection d'une ou plusieurs fautes lors d'un calcul cryptographique afin d'opérer par suite à une analyse mathématique sur le résultat final du calcul.
  - **Attaques par fausses erreurs** : l'objectif de ces attaques est de provoquer des fautes qui n'auront aucune influence sur le résultat final afin d'identifier des opérations inutiles dans le calcul dépendant de la clef.
  - **Attaques par courbe faibles** : ces attaques vont exploiter une faiblesse mathématique en injectant une faute qui va affaiblir les paramètres cryptographiques permettant ainsi d'effectuer une attaque mathématique.
  - **Attaques différentielles** : à l'image des attaques verticales, ces attaques vont exploiter une multitude de fautes injectées à différents instants lors de l'exécution de plusieurs calculs utilisant chacun les mêmes données en entrée.

3.2 État de l'art des attaques par observation

Les attaques par canaux auxiliaires permettent de récupérer une clef secrète utilisée par un algorithme cryptographique, même si ce dernier est considéré mathématiquement mathématiquement sûr. En effet, chaque étape du calcul cryptographique induit un état interne du circuit intégré qui peut être capturé et analysé via des mesures physiques. Ainsi, un ensemble de matériel est nécessaire pour mener une telle attaque. La Figure 3.2 présente un schéma du dispositif nécessaire afin de tester un circuit intégrer (*Device Under Test* ou DUT) face à des attaques par canaux auxiliaires. Ce dispositif permet à la fois de récupérer la consommation de courant du DUT et le rayonnement électromagnétique grâce à une sonde électromagnétique, cf. la figure 3.3.

3.2.1 Attaques temporelles

Les attaques temporelles ou *Timing Attacks* (TA) ont été introduites par Kocher [Koc96] et améliorées par Dhem *et al.* [DKL+98]. Elles exploitent des variances dans le temps d'exécution d'un algorithme cryptographique dépendant de la clef manipulée.

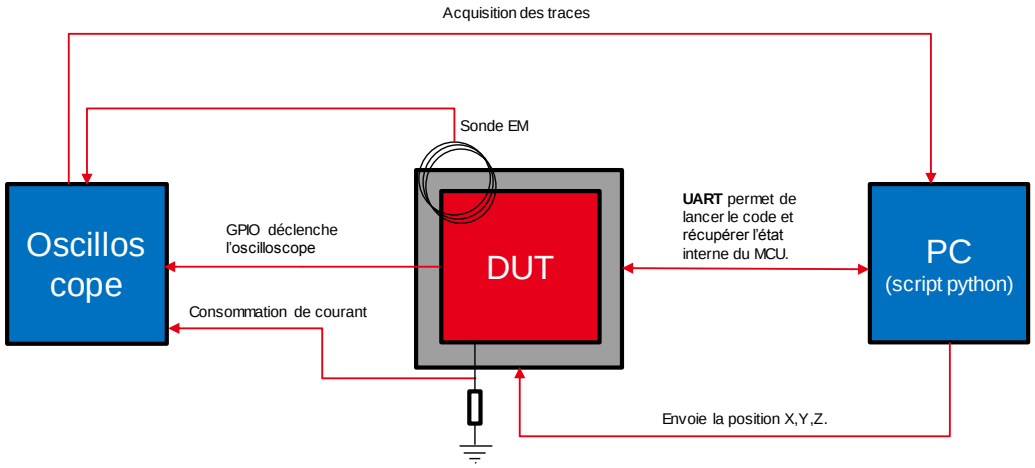


FIGURE 3.2 – Dispositif d'attaques par canaux auxiliaires d'un circuit intégré (DUT).

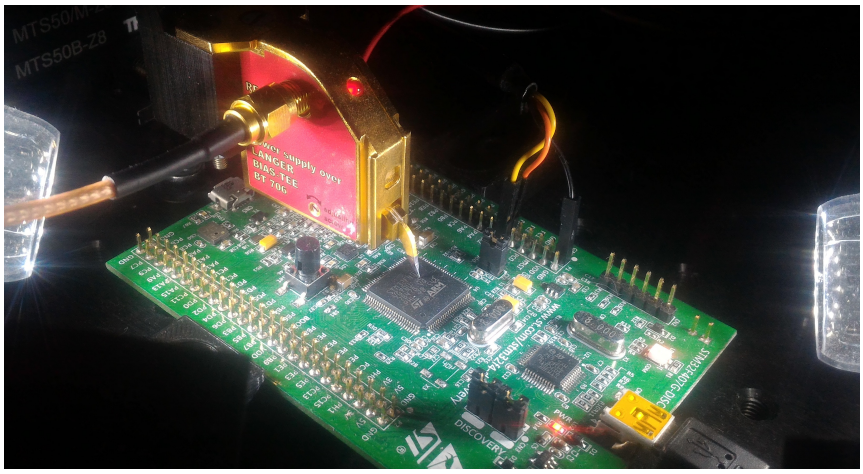


FIGURE 3.3 – Sonde électromagnétique au dessus d'un DUT.

Ces attaques ont été initialement construites pour attaquer une implémentation cryptographique de type RSA mais ont été aisément appliquées aux ECC.

Ces attaques sont récursives, *i.e.* que pour cibler le i-ème bit de la clef, les $n-i-1$ bits de poids fort doivent être connus, où n est le nombre total de bits de la clef attaquée. Ainsi, une clef k est formé de n bits : $k = (k_{n-1}k_{n-2}...k_0)_2$, et l'attaquant essaie de trouver le bit k_i. Pour effectuer l'attaque, nous devons collecter différents temps d'exécutions pour $k_i = 0$ et $k_i = 1$ pour différents points de base sur une implémentation identique à celle qui est attaquée. Chaque temps consiste en deux sous-mesures de temps l'une à la dernière opération effectuée pour le calcul de k_{i-1} et l'autre à la fin de l'opération effectuée pour le calcul de k_i. La différence entre ces deux mesures de temps est alors comparée afin de trouver la bonne hypothèse pour le bit k_i.

Il s'en suit que pour se prémunir de ce genre d'attaques, nous pouvons utiliser une loi de groupe unifiée ainsi qu'une arithmétique sur le corps fini en temps constant. Ceci n'étant pas suffisant, il faut aussi s'assurer que l'algorithme pour effectuer kP soit en temps constant. Ainsi, l'algorithme 2.8 de l'Échelle de Montgomery [JY02] permet de facilement se prémunir de ce genre d'attaque.

3.2.2 Attaques simples

Les attaques simples ou *Simple Side Channel Attack* (SSCA) ont été introduites pour le cas des courbes elliptiques par Coron [Cor99]. Elles exploitent la non-homogénéité de certains algorithmes de multiplication scalaire comme l'algorithme 2.5 de *Double and Add*. Comme nous l'avons vu dans la section 2.3.2, cet algorithme ne va pas effectuer les mêmes calculs si le bit courant est 0 ou 1. Ainsi, sur une trace de consommation de courant ou de rayonnement électromagnétique nous pouvons directement lire la clef utilisée en distinguant les étapes où seul un doublement est effectué (bit 0) et les étapes où un doublement et une addition sont présents (bit 1).

Afin de se prémunir de ce genre d'attaques, nous pouvons utiliser un algorithme de multiplication scalaire homogène comme l'algorithme 2.7 de *Double and Add Always* qui va exécuter une addition fictive afin d'effectuer les mêmes calculs quel que soit le bit courant. Cependant, une telle contremesure ralentit grandement les performances du calcul cryptographique et l'implémentation devient alors sensible aux attaques par fautes, cf. section 3.3.1. Ainsi, il vaut mieux privilégier un algorithme tel que l'Échelle de Montgomery [JY02] qui a une arithmétique homogène sans pour autant ajouter de fausses opérations au sein du calcul.

RSCA : Cette catégorie d'attaque, dite *Refinied Side Channel Attack* (ZSCA), introduite par Goubin [Gou03] améliore les attaques SSCA en utilisant des points spéciaux au sein du calcul.

Cette attaque est aussi récursive : un attaquant doit connaître les $n - i - 1$ bits pour attaquer le i-ème bit. De plus, l'attaquant doit pouvoir soumettre un point spécial P_s en entrée de la multiplication scalaire de telle sorte que le point temporaire en entrée du calcul du bit k_i soit $P_0 = (0, y)$. Dans ce cas là, si $k_i = 1$, alors P_0 sera doublé à l'étape $i - 1$ ce qui sera visible sur une trace d'une mesure physique, car la coordonnée x est nulle.

Pour effectuer une telle attaque, nous connaître l'existence de tels points sur la

courbe utilisée. Dans le cas des BEC, l'équation donnée par la définition 2.22, le seul point existant P_0 est le point donnée par les coordonnées $(0, d_2/d_1)$ et $(0, 0)$. En effet, nous avons :

$$\begin{aligned} d_1 y + d_2 y^2 &= 0 \\ y(d_1 + d_2 y) &= 0 \end{aligned}$$

Dans le cas où $d_1 = d_2$, nous avons pour solution les deux points $(0, 0)$ qui est l'élément neutre et $(0, 1)$ qui est un point d'ordre 4. De même :

$$\begin{aligned} d_1(y + y^2) &= 0 \\ y &= y^2 \end{aligned}$$

Ainsi, ce type d'attaque est impossible sur ce modèle de courbes elliptiques si $d_1 = d_2$, car pour qu'au i-ème bit du calcul nous ayons le point $P_0 = (0, 0)$ ceci implique que P_s soit un point d'ordre $(k_{n-1}..k_{i+1})_2^{-1}$. Ceci n'est généralement pas possible dans le cas de courbes elliptiques pour des applications cryptographiques. En effet, le cardinal de la courbe E est choisi de telle sorte qu'il soit proche d'un nombre premier, *i.e.* $|E| = hp$ où p est premier et le cofacteur h est petit, cf section 4.2. De plus, si au i-ème bit nous avons $P_0 = (0, 1)$ alors P_s est un point d'ordre 4, *i.e.* soit $(0, 1)$, soit $(1, 0)$, ainsi en vérifiant que le point d'entrée de la multiplication scalaire n'est pas d'ordre 4 nous écartons ce genre d'attaques. De plus, l'utilisation de points de petit ordre est exploitée par une attaque plus efficace comme nous le verrons ci-dessous.

ZSCA : Akishita et Takagi [AT03] ont amélioré la RSCA en *Zero Side Channel Attack* (ZSCA).

Cette variante suit la logique que la RSCA mais utilise des points $P = (x_p, y_p)$ afin qu'une valeur intermédiaire lors du calcul du bit k_i s'annule. Les auteurs donnent l'exemple du point P de telle sorte que $3x_p + a = 0$ sur le modèle de Weierstrass donné par l'équation de la définition 2.15. Si ce point est utilisé en coordonnées jacobiennes, alors la valeur intermédiaire $C = 3X_p^2 + aZ_p^4$ s'annule pour $X_p = x_p Z_p^2$. Ainsi, nous avons plus de points particuliers qui peuvent amener à l'exploitation d'une telle attaque.

Par conséquent, la faisabilité de cette attaque va dépendre du système de coordonnées utilisé. Dans le cas de notre implémentation des BEC, nous utilisons le système de coordonnées différentielles mixtes avec la coordonnée Z commune. Comme nous l'avons vu à la section 2.4.2 et résumé dans le tableau 2.4, c'est le système le plus efficace. L'algorithme 2.15 présente les différentes étapes nécessaires pour le calcul calculer $w(2P)$ et $w(P + Q)$ à partir de $w(P)$, $w(Q)$ et $w(P - Q)$. Il nous faut alors étudier dans quelles conditions l'un de ces calculs intermédiaires s'annule.

Nous savons que W_4 et W_5 ne peuvent pas être nuls sinon $w(2P)$ et $w(P + Q)$ seraient soit l'élément neutre $(0, 0)$, soit le point d'ordre 2 $(1, 1)$. Ainsi les seules possibilités pour P et Q sont les points du sous-groupe de E d'ordre 4 formé par $(0, 0)$, $(1, 1)$, $(1, 0)$ et $(0, 1)$. Ainsi, en vérifiant que le point de base P du calcul kP n'est pas un point d'ordre 4, nous pouvons exclure cette possibilité.

Il s'en suit que pour $W_4 = UT$ et $W_5 = VS$ sont non-nuls et donc S, T, U et V le sont également, car ce sont des éléments du corps finis $\mathbb{F}_{2^d}$. Ainsi, les seuls calculs intermédiaires pouvant être nuls sont C, E ou D.

Or, $C = (W_2 + W_3)^2$ et $E = w_1^{-1}C$ ne peuvent pas être nuls car nous aurions dans ce cas $P = Q$. Ceci est impossible dans le cas de l'utilisation de ce système de coordonnées différentielles avec l'algorithme 2.8 de l'Échelle de Montgomery. En effet, si nous avons $P = Q$, alors $w(P - Q) = w((0, 0)) = 0 = w(P)$. En vérifiant que le point d'entrée du calcul de kP n'est pas l'élément neutre $(0, 0)$ nous nous assurons qu'à chaque étape de l'algorithme 2.8, C et E sont non-nuls.

De plus, $D = Z^2$ ne peut être nul sans quoi les points P et Q sont des points à l'infini qui n'appartiennent pas à la courbe E. Ainsi, en vérifiant que le point d'entrée du calcul kP est bien sur la courbe nous pouvons éviter ce cas.

Finalement, en vérifiant que le point de départ de la multiplication scalaire n'est pas un point d'ordre 4, nous pouvons nous prémunir de ce genre d'attaque. Ceci est valable si E est choisi de telle sorte que son cardinale $|E| = 4p$, où p est un nombre premier, ce qui est toujours le cas si $d_1 = d_2$.

SVSCA : Cette attaque, dite *Same Value Side Channel Attack*, présentée par Murdica *et al.* [MGD+12] est une variante des attaques RSCA et ZSCA qui consiste à utiliser des valeurs qui se répètent au sein d'un calcul de $2P$ ou $P + Q$.

La stratégie est la même que pour les attaques RSCA et ZSCA. Nous cherchons à construire un point spécial de sorte qu'à la i-ème étape du calcul kP nous ayons deux valeurs intermédiaires identiques si $k_i = 1$.

Dans le cas des BEC nous utilisons les coordonnées différentielles mixtes dont l'addition et le doublement sont effectués via l'algorithme 2.15. De plus, nous utilisons comme algorithme pour effectuer le calcul kP, l'Échelle de Montgomery où à chaque étape du calcul nous avons une relation entre P et Q, *i.e.* $Q = ((k_{n-1}...k_{i+1})_2 + 1)P$. Ainsi, trouver des valeurs intermédiaires entre le calcul de $2P$ et $P+Q$ semblent difficile vu les conditions imposées à ces deux points. Cependant, les calculs intermédiaires $D = Z^2$ et $S = (W_2(Z + W_2))^2$ ne dépendent que du point P. Nous pouvons alors imposer une condition à P de sorte que $S = D$:

$$\begin{aligned}(W_2(Z + W_2))^2 &= Z^2\\ W_2(Z + W_2) &= Z\\ W_2Z + W_2^2 &= Z\end{aligned}$$

En effectuant le changement de variable $W_2 \leftarrow ZW_2$, nous obtenons la condition suivante :

$$W_P + W_P^2 = \frac{1}{Z} \tag{3.1}$$

Ainsi, pour un $1/Z$ de trace nulle donnée nous pouvons construire un point P en résolvant l'équation précédente.

Il s'en suit que si P est doublé alors $S = D$, donc dans le cas de l'Échelle de Montgomery si le bit courant k_i est 0 alors nous avons $S = D$ sinon dans le cas contraire, c'est le point Q qui est doublé. Une fois le point spécial P construit nous pouvons construire un point $G = ((k_{n-1}...k_{i+1})_2^{-1}[|E|])P$ qui sera le point de base du calcul kG.

Cette attaque nécessite autant de traces que le scalaire k a de bits. Une manière simple de se prémunir contre ce genre d'attaque est de changer la représentation

projective du point de base via un nombre aléatoire. Ces méthodes sont présentées dans la section 3.4. En effet, en changeant la représentation du point de base nous changeons la représentation du point P à l'étape i et donc l'équation 3.1 n'est alors pas nécessairement valide.

SSCA et points de petit ordre : Nous pouvons affiner les attaques manipulant le point base pour effectuer une SSCA grâce aux attaques utilisant les points d'une courbe elliptique ayant un petit ordre [FGV11, GVY17b, USK+18].

La stratégie de ces attaques est de changer le point de base par un point de petit ordre qui va induire une trace de consommation de courant ou de rayonnement électromagnétique particulier. Pour effectuer cela, la courbe utilisée doit alors admettre un point de petit ordre ce qui n'est pas toujours le cas. Par exemple, la courbe P256 du NIST [Gal13] n'admet pas de tels points et est donc insensible à ce type d'attaque. La courbe d'Edwards Curve25519 [NJ16], quant à elle, admet des points d'ordres 4 qui peuvent être utilisés pour mener à bien ce genre d'attaque [GVY17b, USK+18].

Dans [GVY17b], les auteurs attaquent une implémentation de la multiplication scalaire sur la Curve25519 utilisant l'Échelle de Montgomery. L'attaque du bit k_i se fait à partir la connaissance des $n - i - 1$ bits de points forts de la clef k. Ainsi, les auteurs expliquent que si k_i et k_{i+1} sont identiques alors la réduction modulaire employée dans l'implémentation attaquée va être plus courte. Nous pouvons alors extraire la clef k en contrôlant la longueur de la réduction modulaire pour chaque bit de la clef. Ainsi, cette attaque exploite un défaut d'implémentation. En effet, la réduction modulaire employée dans l'implémentation attaquée n'est pas en temps constant, *i.e.* le temps de calcul de cette opération va directement dépendre des données d'entrée.

L'attaque présentée par les auteurs de [USK+18] utilise les points d'ordre 4 de la courbe Curve25519 afin que certaines valeurs intermédiaires du calcul d'une étape de l'Échelle de Montgomery s'annulent si le bit courant est 1 ou 0. La force de cette attaque est de pouvoir être plus facilement transposable à une autre implémentation contrairement à l'attaque présentée dans [GVY17b]. Cependant, les auteurs de [USK+18] n'ont pu effectuer celle-ci en une seule trace de consommation de courant, mais en moyennant 80 traces.

Par conséquent, la cryptographie sur les BEC est sensible à ce genre d'attaque si $d_1 = d_2$. En effet, dans ce cas là, la courbe E admet deux points d'ordre 4 : $(1, 0)$ et $(0, 1)$.

Une manière simple de se prémunir contre ce genre d'attaque est de vérifier, avant le calcul de kP, que le point de base n'est pas un point de petit ordre. De plus, choisir une courbe elliptique E dont le cardinal est de la forme $|E| = hp$ avec p un nombre premier et h un petit cofacteur permet de limiter les vérifications afin d'éviter les points de petit ordre. Dans le cas des BEC, nous choisirons $h = 4$ et il nous suffira alors que le point de base ne soit pas l'un des points $(0, 0), (1, 0), (0, 1), (1, 1)$ (cf. section 2.4.1).

3.2.3 Attaques verticales

Les attaques verticales sont une catégorie d'attaques par canaux auxiliaires qui consistent à effectuer une analyse statistique sur un ensemble de traces de consom-

mation ou de rayonnement électromagnétique afin d'inférer de l'information sur la clef utilisée. La principale différence avec les SSCA est que ces dernières utilisent une seule courbe pour trouver des bits de la clef, même s'il faut une trace de mesures physiques par bit de la clef. Les attaques verticales vont utiliser un ensemble de traces pour retrouver une partie des bits formant la clef secrète.

Certaines attaques comme la CSCA (*Correlation Side Channel Attack*) ou la DSCA (*Differential Side Channel Attack*) vont nécessiter un grand nombre de traces afin d'atteindre un taux de succès significatif. D'autres attaques, comme l'attaque par doublement, ont besoin d'un nombre fixe de traces pour être menées à bien. Malgré tout, ces attaques font parties de la catégorie des attaques verticales, car elles nécessitent au minimum plusieurs traces pour inférer un bit de la clef utilisée.

CSCA, DSCA : Les attaques dites *Correlation Side Channel Attack* (CSCA) [Cor99] et *Differential Side Channel Attack* (DSCA) [KJJ99] utilisent un grand nombre de traces afin de désigner la bonne clef parmi un ensemble d'hypothèses. Dans la suite de cette thèse, nous concentrerons sur la CSCA. En effet, les auteurs de [LCC+06] ont montré que la CSCA est une forme spéciale de la DSCA normalisée par l'écart type du signal mesuré.

La stratégie de cette attaque est de construire un ensemble d'hypothèses pour chaque clef possible en fonction d'un modèle de fuites. Ces hypothèses sont définies pour un ensemble de points de base. Une fois ces hypothèses posées, un attaquant pourra calculer la corrélation entre les hypothèses et l'ensemble des traces de consommation ou EM mesurées pour chaque point précédemment choisi. Ainsi, l'hypothèse avec la corrélation la plus élevée a une forte probabilité d'être la clef.

Plusieurs paramètres vont influer sur l'efficacité d'une telle attaque :

- **Modèle de fuites** : un modèle de fuites nous permet de calculer les hypothèses pour chaque point considéré. Ce modèle doit être choisi de telle sorte qu'il y ait une corrélation entre l'hypothèse donnée par ce modèle et une trace d'une mesure physique correspondant au calcul cryptographique de la clef attaquée. Il existe deux modèles de fuites généralement considérés :
  - **Poids de Hamming** : ce modèle de fuites correspond au nombre de bits à 1 pour une données de longueur de n bits.
  - **Distance de Hamming** : ce modèle de fuites correspond au poids de Hamming de la différence entre deux données de même longueur.

  Comme nous l'avons vu en introduction de ce chapitre, ce sont les transistors qui vont induire une fuite. Ainsi, le poids de Hamming correspond au nombre de capacités C_l chargées, cf. 3.1, alors que la distance de Hamming correspond au nombre de changements d'état de la capacité C_l entre deux données.
- **Nombre d'hypothèses** : dans le cas d'une attaque par corrélation sur les ECC, la clef privée k est bien trop grande pour être attaquée en une seule CSCA. En effet, le nombre d'hypothèses est bien trop important pour pouvoir appliquer l'attaque. Ainsi, il est nécessaire de scinder l'attaque de la clef k en plusieurs attaques CSCA dont la taille des hypothèses va dépendre du nombre t de bits qui sera attaqué en une seule fois. De plus, le calcul de kP est structuré de façon séquentielle (cf section 2.3.2), *i.e.* les bits de poids forts (ou poids faibles) sont utilisés en premier. Ainsi, pour attaquer les bits $(k_{i+t-1}...k_i)_2$ l'attaquant doit alors connaître les $n-i-t$ premiers bits de la clef.

Les hypothèses sont alors disposées au sein de la matrice suivante :

$$H = \begin{pmatrix} h(0P_1) & \cdots & h((2^t-1)P_1) \\ \vdots & \ddots & \vdots \\ h(0P_N) & \cdots & h((2^t-1)P_N) \end{pmatrix} \tag{3.2}$$

Où N est le nombre de points considérés, t le nombre de bits attaqués et h le modèle de fuites considéré.

- **Nombre de points** : pour chaque point de base, nous devrons calculer les hypothèses pour toutes les clefs possibles et capturer une trace de consommation ou de rayonnement EM pour kP où k est la clef attaquée. La quantité de points considérée influera grandement pour réaliser avec succès l'attaque. En effet, un trop petit nombre de points ne permettra pas de discriminer par corrélation une hypothèse correspondant à la clef. Par contre, si ce nombre de points est suffisant, les corrélations calculées pour la bonne hypothèse seront significatives.

  Chaque trace correspondant à un point de base forme une ligne d'une matrice de traces S. Ainsi, la matrice S contiendra N lignes, correspondant au nombre de points considérés, et s colonnes correspondant au nombre de mesures physiques effectuées pour un point P au cours du temps.

Une fois les matrices H et S construites, il suffit de calculer la corrélation de Pearson entre ces deux matrices. Cette corrélation est donnée par :

$$C = \frac{\mathrm{cov}(S, H)}{\sigma_S \sigma_T} \tag{3.3}$$

Où cov est la covariance et σ l'écart type. Ainsi, les lignes de C correspondent à la probabilité de corrélation à chaque instant des traces capturées (s éléments) et les colonnes aux différentes clefs possibles. Afin de sélectionner la bonne clef, il suffit de sélectionner l'hypothèse de clef ayant le plus grand taux de corrélation.

Une contremesure proposée par Coron [Cor99] est d'ajouter de l'aléatoire dans la représentation des points. Cet aléa permet de décorréler les hypothèses de H et les traces capturées pour chaque point de base. Une autre manière de casser cette corrélation est de cacher la clef k avec un nombre aléatoire. Les détails de ces différentes contremesures seront donnés dans la section 3.4.

La force de ces attaques réside dans leur indépendance à un modèle de courbes elliptiques ou à une représentation particulière des points. Cette attaque permet de faire fuir une clef privée utilisée par la cryptographie basée sur les BEC par exemple.

Bits d'adresse DSCA : Cette attaque, proposée par Itoh *et al.* [IIT02], peut être interprétée comme une variante d'une DSCA. En effet, un attaquant va effectuer une attaque différentielle non pas sur les données manipulées, mais sur les adresses mémoires manipulées.

Cette stratégie repose sur l'observation suivante : une grande majorité des algorithmes de multiplication scalaire n'utilisent pas les mêmes registres de données et donc d'adresses si le bit courant est 1 ou 0. Par exemple, l'algorithme 2.8 de l'Échelle de Montgomery va effectuer un doublement de R_1 si $k_i = 1$. Ainsi, l'attaque consiste

à repérer si l'adresse du point doublé correspond à la même adresse que le point doublé pour un bit de référence que nous supposerons être égal à 1. Si cette hypothèse s'avère fausse, il suffira alors de considérer le conjugué de la clef retrouvée.

Cette attaque peut être déjouée en utilisant des adresses aléatoires pour les registres de points utilisés par le calcul de kP. Cette contremesure a été proposée par Itoh *et al.* [IIT03] puis améliorée dans [IISO10] par Izumi *et al.*.

Nous pouvons aussi protéger une implémentation du calcul de kP contre ce genre d'attaque en fixant les adresses utilisées à chaque étape du calcul quel que soit le bit courant. Ainsi, le succès d'une telle attaque va principalement dépendre de l'implémentation et non du modèle de courbes considérées ou du système de coordonnées utilisée.

Il existe une variante de cette attaque [KDKL17, KDKL18] qui compare les adresses des registres de points utilisées au sein d'un même calcul de kP. Une seule trace est donc nécessaire pour effectuer une telle attaque. Ce type d'attaque combine à la fois les attaques des bits d'adresse et les attaques horizontales. Cette version d'attaque permet de passer outre les contremesures ajoutant de l'aléatoire dans les données et les adresses. Cependant, si les adresses utilisées lors d'une étape du calcul ne dépendent pas du bit courant, alors cette attaque reste impossible.

Attaque par doublement : L'attaque par doublement introduite par Fouque et Valette [FV03] provoque des collisions entre deux calculs de kP en changeant le point de base. Le premier calcul dont nous récupérons la trace est kP et le second est $k(2P)$.

Les auteurs démontrent qu'entre ces deux multiplications scalaires des calculs intermédiaires sont similaires si l'algorithme 2.5 de *Double & Add* est utilisé. En effet, si $k_i = 0$, alors l'étape i de kP est identique à l'étape $i - 1$ du calcul $k(2P)$. Ainsi, en identifiant les collisions au sein de ces deux calculs, nous pouvons retrouver les bits 0 de la clef k.

Une contremesure simple pour éviter ce genre de fuites est d'utiliser l'algorithme 2.8 de l'Échelle de Montgomery. Il s'en suit que la cryptographie basée sur les BEC utilisant l'algorithme 2.8 n'est pas sensible à ce genre d'attaque.

Attaque relative par doublement : Il existe une variante de l'attaque par doublement proposée par Yen *et al.* [YKMH05] qui permet d'attaquer l'Échelle de Montgomery.

La stratégie reste la même : nous devons mesurer les traces pour le calcul de kP et $k(2P)$. Cette fois-ci, une collision apparaît entre les étapes i et $i - 1$ des calculs kP et $k(2P)$ si les bits k_i et k_{i-1} sont identiques (indépendamment de 0 et 1). Il s'en suit que l'attaque ne permet pas de retrouver directement les bits de la clef k mais seulement les séries de bits identiques. Ainsi, celle-ci permet de réduire l'espace des clefs possibles à deux clefs.

Les auteurs appliquent cette attaque contre une implémentation RSA. Cependant, dans les cas des ECC elle reste applicable à une Échelle de Montgomery sur les courbes elliptiques.

Une contremesure à l'attaque proposée par les auteurs [YKMH05] est l'utilisation d'un algorithme de multiplication scalaire alternatif ne permettant pas ce genre de collisions.

Attaque par propagation de la retenue : L'attaque par propagation de la retenue publiée par Fouque *et al.* [FRVD08] cible la contremesure consistant à cacher le scalaire k avec un nombre aléatoire r. Généralement, nous remplaçons k par $k' = k + r|E|$.

Cependant, la propagation de la retenue de cette opération dépend des bits de poids forts de chaque mot formant k et $r|E|$. Ainsi, en moyennant la trace de la consommation du canal auxiliaire mesuré (courant ou du rayonnement électromagnétique) de l'addition $k' = k+r|E|$, nous pouvons inférer s'il y a ou non propagation de la retenue entre les mots des opérandes grâce à l'amplitude de la trace moyennée.

Cette attaque nous permet d'inférer quelques bits de la clef k, pour les bits restants nous pouvons résoudre l'ECDLP à l'aide de l'algorithme BSBG.

Pour se protéger de ce genre d'attaque, nous pouvons utiliser la contremesure de Joye *et al.* [CJ01] qui consiste à séparer le scalaire k de façon aléatoire. Cette contremesure sera présentée dans la section 3.4. Cependant, elle double le temps de calcul de kP. Afin de prévenir ce genre d'attaque et de limiter le surcoût induit pour les contremesures ajoutées, il est conseillé de ne pas utiliser la contremesure qui cache la clef mais d'en privilégier d'autres. Citons par exemple, l'ajout d'aléatoire dans la représentation du point de base.

Dugardin *et al.* [DPN+16b] montrent que la propagation de la retenue peut être utilisée pour mener des attaques par profilage. En optant pour un modèle de courbes elliptiques sur un corps fini de caractéristique 2, nous prévenons toute forme de fuite due à la propagation de la retenue. Les auteurs de l'attaque précisent que l'on peut aussi s'en prémunir en ajoutant de l'aléatoire dans les coordonnées du point de base.

3.2.4 Attaques horizontales

Le principal inconvénient des attaques verticales est leur verticalité. En effet, plusieurs mesures avec la même clef sont nécessaires afin de pouvoir mener à bien une attaque. Dans certains cas, il est impossible de mener à bien de telles attaques : un protocole cryptographique utilisant des clefs éphémères empêche d'effectuer plusieurs mesures avec la même clef, tels que les protocoles d'échange de clefs ECDH ou celui de signatures ECDSA. De plus, les attaques verticales nécessitent de choisir le point de base lors d'une mesure or certains protocoles (ECDSA, EdDSA) n'offrent pas cette possibilité.

Ainsi, il est donc difficile de mener une attaque verticale sur le calcul de kP contre de nombreux protocoles cryptographiques à clefs publiques. Il s'en suit le développement d'attaques dites horizontales qui vont utiliser une seule et unique trace et vont effectuer une analyse statistique entre différentes parties de cette trace.

Ces attaques sont redoutables de par leur conception. Cependant, elles sont difficiles à mettre en œuvre. Leur succès peut directement dépendre des choix d'implémentations faits pour calculer kP. Ainsi, leur complexité et leur efficacité théorique en font un domaine de recherche à part entière. Dans cette thèse, nous n'effectuerons pas d'attaques horizontales, laissant cet aspect des attaques physiques pour de nouvelles perspectives de recherche. Malgré tout, nous proposons d'effectuer un état de l'art de celles-ci car dans notre construction et notre implémentation d'une cryptographie nouvelle sur les courbes elliptiques, il nous faut avoir à l'esprit les stratégies utilisées par ce genre d'attaque afin de limiter leur impact d'un point de vue théorique. Toutes les hypothèses de sécurité que nous énoncerons pour la cryp-

tographie sur les BEC resterons très spéculatives et nécessitent dans l'avenir une étude complète.

De par la spécificité de ces attaques, certains chercheurs ont commencé à adopter une approche systématique afin d'évaluer leur impact sur une implémentation cryptographique sur les courbes elliptiques, [PZS17, APS19].

Attaque Big Mac : Cette première attaque horizontale fut introduite par Walter [Wal01] sur une implémentation RSA. La stratégie de cette attaque est d'identifier des multiplications modulaires ayant un opérande commun au cours du calcul. Pour repérer des opérandes identiques entre deux multiplications, Walter utilise la distance euclidienne. Si celle-ci est petite entre deux multiplications alors un des opérandes est identique et à l'inverse, si cette distance est grande alors les opérandes sont différents. Cette attaque sera améliorée par Clavier *et al.* [CFG+12] par l'utilisation de la corrélation de Pearson au lieu de la distance euclidienne.

Cette attaque fut adaptée par Bauer *et al.* [BJPW13] dans le cadre de la cryptographie sur les courbes elliptiques. La stratégie adoptée est de repérer des opérandes pouvant être communs entre deux calculs intermédiaires de la multiplication scalaire. Si ces deux calculs possèdent un opérande commun en fonction du bit courant de la multiplication scalaire, nous pouvons retrouver ce bit sous l'hypothèse que nous pouvons distinguer des opérandes communs entre deux multiplications scalaires.

Dans le cas des BEC, l'utilisation des coordonnées différentielles mixtes avec la coordonnée Z commune est permise par l'algorithme 2.15. Ce système de coordonnées est utilisé avec l'algorithme 2.8 de l'Échelle de Montgomery. Ainsi, aucune multiplication intermédiaire ne comporte un opérande commun avec une autre multiplication qui dépendrait du bit courant : *a priori* ce modèle de courbes elliptiques couplé avec le système de coordonnées de l'algorithme 2.15 ne serait pas sensible à ce type d'attaque.

Attaque horizontale par corrélation HCSCA : L'*Horizontal Correlation Side Channel Attack* a été introduite par Clavier *et al.* [CFG+10]. L'attaque nécessite une seule et unique trace de consommation ou rayonnement électronique du calcul de kP. Cependant, l'attaquant doit connaître le point de base utilisé contrairement à une attaque Big Mac.

L'objectif de cette attaque est de trouver récursivement les bits de k via des hypothèses de clefs qui seront corrélées avec la trace. Cette stratégie est similaire à celle de l'attaque verticale CSCA. Ainsi, un attaquant va calculer les valeurs intermédiaires μ_i d'une multiplication entre plusieurs opérandes supposés qui dépendent du bit courant. En connaissant le poids de Hamming des μ_i un attaquant peut les corréler avec les traces correspondant au calcul de μ_i. L'attaquant pourra inférer le bit de la clef en sélectionnant l'hypothèse ayant la plus forte corrélation.

Cette attaque a été étendue à plusieurs algorithmes de multiplication scalaire par Bauer *et al.* [BJPW13].

Nous avons vu que pour construire les hypothèses μ_i un attaquant doit connaître le point de base. Ainsi, une contremesure d'ajout d'aléa dans la représentation projective du point de base permet de se prémunir de cette attaque. De même, l'ajout d'aléas dans les paramètres de la courbe à l'aide d'isomorphisme permet de se protéger contre ce genre d'attaque.

Attaque horizontale par valeurs identiques HSVSCA : L'*Horizontal Same Value Side Channel Attack* est une extension de l'attaque *Same Value Side Channel Attack* publiée par Murdica [MGD+12].

Tout comme l'attaque SVSCA, la stratégie est d'utiliser un point de base spécial qui va permettre d'avoir des valeurs intermédiaires identiques en fonction des bits de la clef. Les auteurs utilisent le calcul de l'élévation du carré afin d'avoir des valeurs intermédiaires identiques. Ces collisions peuvent apparaître soit à la même étape du calcul de kP, soit à l'étape suivante du calcul.

Nous pouvons nous prémunir de cette attaque en ajoutant de l'aléa dans le scalaire k. Malgré tout, un attaquant peut attaquer plusieurs bits de la clef à la fois ce qui réduit l'efficacité de la contremesure.

3.2.5 Attaques par profilages

Les attaques par profilages ou *Template Attack* ont été introduite par Chari *et al.* [CRR02]. Les attaques de cette famille utilisent une stratégie qui est organisée en deux phases :

1. **Profilage** : la première étape consiste à construire un ensemble de profils (*Template*) caractérisant la fuite induite pour des hypothèses de clefs. Cette construction se fait en collectant un grand nombre de traces de consommation ou de rayonnements électromagnétiques pour chaque valeur de clef possible. La difficulté de cette phase d'apprentissage est d'obtenir des *templates* qui seront efficaces, *i.e.* qui permettront de retrouver la clef secrète en un minimum de traces d'attaques, une trace dans le cas idéal. Pour ce faire, nous employons des méthodes de réductions de dimensions permettant de sélectionner l'information pertinente au sein d'une trace d'une mesure physique. Nous verrons par la suite les différentes méthodes de réduction de dimension que nous considérerons dans cette thèse.
2. **Attaques** : la seconde étape consiste à appliquer les *templates* à un ensemble de traces dites d'attaque correspondant à la même clef secrète à trouver. Ainsi, nous pouvons inférer que le *template* le plus proche correspond à la clef utilisée par les traces d'attaques. Pour trouver quel *template* est le plus proche nous utiliserons un modèle de gaussiennes multivariées, proposé par Chari *et al.* [CRR02].

La force de cette attaque est son indépendance à un modèle de courbes elliptiques particulier. Afin d'en augmenter son efficacité, il est opportun de cibler une sous-partie du calcul kP correspondant exactement à l'opération qui va manipuler la donnée secrète k. Nous verrons au chapitre 6 quelle opération du calcul kP sera ciblée dans le cadre des BEC. Il est à noter que cette attaque est indépendante du modèle BEC et qu'elle peut s'appliquer à d'autres modèles.

Ces attaques imposent qu'un attaquant ait un accès complet à une copie de l'implémentation ciblée et du système embarqué cible. Il nous faut donc une copie de tout le système sur lequel nous pouvons soumettre nos propres clefs en entrée. En effet, la phase d'apprentissage demande une grande quantité de données correspondant à chaque clef possible. Pour que le *template* soit efficace, nous devons le construire sur un système identique à celui attaqué.

Malgré tout, ce type d'attaque est très peu portable, *i.e.* les traces de profilages et d'attaques sont généralement effectuées sur le même circuit. La difficulté

de pouvoir appliquer un *template* entre différents matériels embarqués a été étudié par Choudary *et al.* [CK14]. L'idée des auteurs est d'utiliser différents exemplaires d'un même système embarqué pour la construction du *template*. De plus, l'utilisation de méthodes avancées de construction des *templates* (cf. section 6.2.2) permet d'améliorer la portabilité des *templates*. Les auteurs ont confirmé leurs travaux dans [CK18].

Dans le cadre de la cryptographie sur les courbes elliptiques la clef k est utilisée de façon séquentielle, bit à bit, dans le cas de l'utilisation d'algorithmes de multiplications scalaire tels que le *Double & Add* (algorithme 2.5), *Double & Add always* (algorithme 2.7) ou l'Échelle de Montgomery (algorithme 2.8). Nous pouvons donc attaquer indépendamment chaque bit et donc les clefs possibles sont 0 et 1 dans le cas des attaques par profilages. Nous avons alors seulement deux *template* à construire.

Construction des profils : Pour cette première phase, l'attaquant doit pouvoir collecter un grand nombre de traces d'une mesure physique (consommation, rayonnement électromagnétique) correspondant au calcul kP pour différents P. Chacune de ces traces du calcul kP est séparée en sous-traces correspondant au traitement d'un bit du scalaire k. Il faut alors sélectionner les points d'intérêts de cet ensemble de traces. Ces points correspondent aux points temporels des traces qui vont dépendre essentiellement de la clef. En effet, pour que le *template* soit le plus performant, il suffit de sélectionner les points comportant l'information relatives à la clef.

Il existe différentes méthodes de sélection de points d'intérêts (PoI). Dans le cas de la cryptographie sur les courbes elliptiques le t-test permet facilement de sélectionner les points d'intérêt. En effet, nous n'avons que deux clefs possibles : 0 ou 1. Ainsi, nous pouvons comparer la différence des moyennes pour l'ensemble des traces correspondant au bit 0 et correspondant au bit 1. Nous supposons que les traces X_0 pour le bit 0 sont indépendantes de celles X_1 pour le bit 1. Nous pouvons alors appliquer le t-test :

$$t = \frac{\bar{X}_0 - \bar{X}_1}{\sqrt{\frac{\sigma_{X_0}}{n_0} + \frac{\sigma_{X_1}}{X_1}}} \tag{3.4}$$

Où n_0 et n_1 sont le nombre de points de X_0 et X_1, σ_{X_0} et σ_{X_1} est l'écart-type de X_0 de X_1.

Ce test nous permet de mesurer temporellement la différence des moyennes. Cette différence est importante, car elle désigne les points temporels dont la mesure physique acquise dépend, *a priori*, de la clef. Il est courant de prendre une valeur de seuil de 4,5 [tte18]. Dans le cas des attaques par profilages, le résultat du t-test nous permet de sélectionner des PoI. Pour un nombre n de PoI, nous sélectionnons les n points maximums. Cet ensemble de PoI est alors utilisé pour la construction du *template* pour une clef possible $k_b \in \{0, 1\}$. Un *template* est alors construit de la manière suivante :

$$\mu_{k_b} = \begin{pmatrix} \mu_{k_b,1} \\ \mu_{k_b,2} \\ \mu_{k_b,3} \\ \vdots \end{pmatrix}, \qquad \Sigma_{k_b} = \begin{pmatrix} \upsilon_1 & c_{1,2} & c_{1,3} & \cdots \\ c_{2,1} & \upsilon_2 & c_{2,3} & \cdots \\ c_{3,1} & c_{3,2} & \upsilon_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} \tag{3.5}$$

Où les $\mu_{k_b,i}$ sont les moyennes des traces t relatives à clef k_b au i-ème PoI. Ainsi,

pour n_{k_b} traces pour la clef k_b et s_i le i-ème point d'intérêt, nous avons :

$$\mu_{k_b,i} = \frac{1}{n_{k_b}} \sum_{j=1}^{n_{k_b}} t_{j,s_i} \tag{3.6}$$

De plus, pour construire Σ_{k_b}, nous avons l'écart-type υ_i à chaque point d'intérêt et la covariance $c_{i,i'}$ entre les points d'intérêts i et i' :

$$\upsilon_i = \frac{1}{n_{k_b}} \sum_{j=1}^{n_{k_b}} (t_{j,s_i} - \mu_{k_b,i})^2 \tag{3.7}$$

$$c_{i,i'} = \frac{1}{n_{k_b}} \sum_{j=1}^{n_{k_b}} (t_{j,s_i} - \mu_{k_b,i})(t_{j,s_{i'}} - \mu_{k_b,i'}) \tag{3.8}$$

Ainsi, dans le cas de la cryptographie sur les courbes elliptiques nous avons seulement deux *templates* : (μ_0, Σ_0) et (μ_1, Σ_1).

Application des profils : Une fois tous les *templates* construits nous pouvons les appliquer à un ensemble de traces d'attaques A formé de n_A traces a_i. Toutes ces traces ont été obtenues avec la même clef k_b. Pour appliquer les *templates* il suffit d'extraire temporellement les mêmes points d'intérêts qui ont servi à construire les *templates* pour chaque trace a_i. Nous avons alors un nouvel ensemble A' de traces d'attaques $a_i^{'}$ qui sont la concaténation des points d'intérêts.

Pour chaque *template* et chaque trace $a_i^{'}$, nous calculons la fonction de densité de probabilité (PDF) $p_{k_b,i} = f_{k_b}(a_i^{'})$, avec :

$$f_{k_b}(a_i^{'}) = \frac{1}{\sqrt{(2\pi)^{n_{s_i}} |\Sigma_{k_b}|}} e^{-((a_i^{'} - \mu_{k_b,i})^T \Sigma_{k_b}^{-1} (a_i^{'} - \mu_{k_b,i})/2} \tag{3.9}$$

Afin de sélectionner la clef k_b avec la plus forte probabilité, nous pouvons appliquer le principe de maximum de vraisemblance qui consiste à calculer le logarithme de la probabilité P_{k_b} pour chaque k_b.

$$\log P_{k_b} = \sum_{j=1}^{n_A} \log p_{k_b,j} \tag{3.10}$$

Pour lequel le plus grand logarithme $\log P_{k_b}$ donne la clef de l'ensemble A des traces d'attaques.

L'application des attaques par profilages à la cryptographie sur les courbes elliptiques a été introduite par Medwed et Oswald [MO08]. Les auteurs ciblent une implémentation du protocole ECDSA en effectuant au préalable une DSCA sur le calcul kP pour sélectionner les points d'intérêts.

Il est commun de se protéger de cette attaque en ajoutant de l'aléatoire dans le point de base ou la clef k utilisée.

3.3 État de l'art des attaques par injections de fautes

Les attaques par injections de fautes sont actives. La stratégie de ces attaques est de perturber le fonctionnement de l'unité de calcul durant une opération cryptographique. De cette erreur injectée au sein du calcul, l'attaquant pourra en déduire

de l'information sur la clef secrète utilisée. Les multiples manières d'inférer de l'information à partir d'une faute déterminent les différentes attaques par fautes.

La première attaque par injection de fautes contre les ECC fut introduite par Biehl *et al.* [BMM00] qui consiste à effectuer une attaque différentielle par injection de fautes. Depuis, les attaques se sont complexifiées et deviennent de plus en plus pertinentes dans un cas réel. D'elles dépendent fortement les critères de sécurité que doit vérifier une courbe elliptique pour être utilisée à des fins cryptographiques comme nous le verrons dans la section 4.2.

Dans cette section, nous verrons également les différentes familles d'attaques par fautes existantes contre les implémentations cryptographiques sur les courbes elliptiques. Dans cette thèse, nous n'effectuerons aucune attaque par faute, nous les considérerons uniquement d'un point de vue théorique. Dans le cadre d'une implémentation des ECC, il est important de considérer les attaques par fautes afin d'éviter des erreurs d'implémentations qui les faciliteraient.

Il existe trois grandes catégories d'attaques par injection de fautes :

- la première vise à injecter des erreurs qui n'auront aucune influence sur le calcul final, et dont le manque d'effet est une information permettant de déduire des bits de la clef secrète.
- la deuxième vise à affaiblir mathématiquement les ECC en changeant les paramètres cryptographiques utilisés.
- la troisième consiste à cumuler de l'information sur la clef via l'utilisation de plusieurs fautes injectées et du résultat modifié qu'elles induisent.

3.3.1 Attaques par fausses erreurs ou *Safe-error*

Cette catégorie d'attaques par fautes cherche à identifier au sein d'un calcul de kP les opérations qui ne participent pas au résultat final. Ces opérations sont généralement ajoutées afin de protéger un algorithme de multiplication scalaire tel que l'algorithme 2.5 de *Double & Add* face aux attaques SPA, cf. section 3.2.

***C-safe Error* :** Cette catégorie introduite par Yen *et al.* [YKLM01] utilise les faiblesses de la contremesure protégeant l'algorithme 2.5 de *Double & Add always* face aux SSCA en ajoutant de fausses opérations. En effet, en injectant une faute lors de l'addition effectuée par l'algorithme 2.7, nous pouvons en déduire si le bit courant est 0 ou 1, en vérifiant si cette faute modifie le résultat kP.

Il est possible de se protéger contre ces attaques en utilisant un algorithme tel que l'Échelle de Montgomery ou en vérifiant si le résultat se trouve bien sur la courbe.

***M-safe Error* :** La stratégie adoptée par les *M-safe Error* introduite par Yen et Joye [YJ00], consiste à injecter une erreur en mémoire sur un vecteur de calculs. Si l'erreur persiste une fois le vecteur sauvegardé, alors le bit courant peut être déduit. Un algorithme comme celui de l'Échelle de Montgomery peut être affecté par ce type d'attaque comme expliqué dans [YJ00].

Ainsi, dans le cas de la cryptographie sur les BEC, une contremesure simple à ce genre d'attaque serait de vérifier qu'une faute n'a pas été injectée sur les registres R_0 et R_1 de l'Échelle de Montgomery avant de passer au traitement du bit suivant.

3.3.2 Attaques par courbes faibles

Cette famille d'attaque consiste à changer la courbe sur laquelle est effectué le calcul de kP de manière à effectuer ce calcul sur une courbe faible ce qui permet de résoudre l'ECDLP pour retrouver k.

Attaque par point invalide : Cette attaque a été introduite par Bhiel *et al.* [BMM00] pour le modèle de Weierstrass. La stratégie de l'attaque consiste à injecter une faute dans le point de base de telle manière qu'il corresponde à un point sur une courbe faible d'équation $y^2 = x^3 + ax + \tilde{b}$. Cette attaque est rendue possible car le paramètre b dans le modèle de Weierstrass n'est pas utilisé par les formules d'addition et de doublement. Un attaquant peut alors résoudre l'ECDLP sur cette courbe faible afin de retrouver k.

Nous pouvons facilement nous prémunir de cette attaque en vérifiant que le point de base se trouve sur la courbe avant d'effectuer le calcul de kP.

Dans le cas des BEC nous avons un seul paramètre d, si $d_1 = d_2$, qui est utilisé par les formules d'addition et de doublement de l'algorithme 2.15. Ainsi, une telle attaque est impossible sur ce modèle de courbe elliptique.

Attaque par courbe invalide : Cette attaque a été introduite par Ciet et Joye [CJ05] pour le modèle de Weierstrass et est basée sur la même stratégie que celle de l'attaque par point invalide. Toutefois, l'attaquant va directement injecter la faute dans le paramètre a de la courbe.

Attaque par la tordue : Cette attaque publiée par Fouque *et al.* [FLRV08b] va spécifiquement adresser les implémentations de la multiplication scalaire utilisant l'Échelle de Montgomery et l'utilisation de la seule la coordonnée x d'un point. Dans ce cas, la stratégie de l'attaque consiste à injecter une faute dans la coordonnée x du point de base la transformant ainsi en $\tilde{x}$. Nous avons alors :

- $\tilde{x}^3 + a\tilde{x} + b$ est un carré. Dans ce cas, la solution $\tilde{y}$ à $\tilde{y}^2 = \tilde{x}^3 + a\tilde{x} + b$ dans $\mathbb{F}_p$ donne le point $\tilde{P}$ sur la courbe E. Aucune autre déduction n'est alors possible et l'attaque échoue.
- $\tilde{x}^3 + a\tilde{x} + b$ n'est pas un carré. Dans ce cas la solution $\tilde{y}$ à $\tilde{y}^2 = \tilde{x}^3 + a\tilde{x} + b$ dans $\mathbb{F}_{p^2}$ donne le point $\tilde{P}$ sur la tordue E^{tw} de E. Si l'ECDLP est résoluble sur la tordue nous pouvons alors retrouver la clef k.

Ainsi, nous pouvons nous protéger de ce genre d'attaque en choisissant une courbe elliptique E dont la tordue E^{tw} a le même niveau de sécurité. Cette contrainte ajoute un critère de sécurité à la construction de nouvelles courbes elliptiques pour la cryptographie, cf. section 4.2.

3.3.3 Attaques différentielles

Les attaques différentielles ou *Differential Fault Attack* (DFA) vont retrouver de façon récursive la clef k en comparant le résultat $Q = kP$ avec un résultat $\tilde{Q}$ sur lequel une faute a été injectée.

Attaque classique par fautes différentielles : Cette attaque a été introduite par Bhiel *et al.* [BMM00]. Sa stratégie est d'effectuer plusieurs calculs de $Q = kP$ et d'injecter une faute sur une valeur intermédiaire Q_i pour certains calculs.

Nous avons $Q_i = (k_{i-1}...k_0)_2 P$ et $\tilde{Q}_i$ qui est la valeur Q_i avec quelques bits faux. Ainsi, $Q = Q_i + (2^i(k_{n-1}...k_i)_2)P$ et $\tilde{Q} = \tilde{Q}_i + (2^i(k_{n-1}...k_i)_2)P$, un attaquant peut alors essayer toutes les possibilités pour $(k_{n-1}...k_i)_2$ de telle sorte que $Q_i = Q - (2^i(k_{n-1}...k_i)_2)P$ et $\tilde{Q}_i = \tilde{Q} - (2^i(k_{n-1}...k_i)_2)P$ n'aient que quelques bits de différence. Cette attaque peut être alors effectuée de façon récursive afin de retrouver tous les bits de la clef k.

Il est possible de se protéger contre ce type d'attaque en utilisant les coordonnées projectives ou en ajoutant de l'aléatoire dans la représentation des points à chaque calcul de kP. Dans ce cas, l'attaquant ne pourra pas calculer les valeurs intermédiaires Q_i et $\tilde{Q}_i$. Nous pouvons aussi vérifier que le point de sortie se trouve bien sur la courbe.

Pour les BEC, nous utilisons les coordonnées projectives ce qui nous protègent de ce genre d'attaque.

Attaque par changement de signe : Cette attaque introduite par Blömer *et al.* [BOS06] adopte la même stratégie que l'attaque de Bhiel *et al.* [BMM00]. La différence réside dans le type de fautes injectées. Pour cela, nous devons changer le signe d'un point intermédiaire. En coordonnée projective classique, nous avons $P = (X : Y : Z)$ et $-P = (X : -Y : Z)$. Ainsi, un attaquant aura la possibilité de changer l'addition effectuée lors d'une étape du calcul de kP en une soustraction.

L'avantage de cette attaque contrairement à la DFA classique est de pouvoir attaquer des implémentations utilisant les coordonnées projectives. Cependant, dans le cas de la cryptographie sur les courbes elliptiques binaires, les vecteurs de coordonnées ne sont pas signés, il est ainsi impossible de changer le signe d'un vecteur. De plus, dans le cas des BEC, nous savons que pour $P = (X : Y : Z)$: $-P = (Y : X : Z)$. Il est ainsi, impossible de changer le signe d'un point sur le modèle de courbes binaires d'Edwards. À cela s'ajoute le fait que l'on utilise ce modèle en coordonnées différentielles, où $w(P) = w(-P)$, nous ne pouvons donc pas mener une telle attaque sur ce modèle de courbes elliptiques.

Attaque sur la conversion du système de coordonnée : Cette catégorie présentée dans [MMNT13] par Maimut *et al.* injecte une faute lors de la conversion des coordonnées jacobiennes aux coordonnées affines pour appliquer l'attaque mathématique de Naccache *et el.* [NSS04].

Cependant, nous ne pouvons écarter l'éventualité d'une telle attaque dans notre implémentation de la cryptographie sur les BEC. Les contremesures envisagées par [MMNT13] est l'ajout d'aléatoire dans la représentation projective des points. Cette contremesure est identique à celle envisagée pour protéger notre implémentation des attaques CSCA et DSCA.

3.4 Contremesures face aux attaques physiques

Dans cette section, nous détaillons les différentes contremesures envisageables pour se protéger des différentes attaques que nous avons énumérées aux sections

3.2 et 3.3. Notre analyse sera effectuée dans le cadre de la cryptographie sur les BEC. En effet, certaines contremesures intrinsèques à ce modèle de courbe nous permettent d'éviter l'ajout de contremesures additionnelles. Cependant, l'analyse faite dans cette section est purement théorique et étudié en pratique dans le chapitre chapitre 3 pour le cas des attaques par canaux auxiliaires.

Nous pouvons alors synthétiser les différents résultats et analyses précédentes pour évaluer la sécurité intrinsèque d'une implémentation sur les BEC. Nous considérons l'implémentation sur les courbes elliptiques suivante :

- **Modèle** : Courbe binaire d'Edwards avec $d_1 = d_2$, cf. définition 2.22.
- **Corps de base** : $\mathbb{F}_{2^d}$.
- **Système de coordonnées** : Coordonnées w-différentielles mixtes avec Z commun, cf. algorithme 2.15.
- **Algorithme** kP : L'Échelle de Montgomery, cf. algorithme 2.8.

Nous pouvons aussi prendre en compte le déploiement de cette cryptographie au sein de protocoles de signatures ou d'échange de clef. En effet, ces protocoles représentent la majeure partie des applications cryptographiques des ECC. Nous considérons les deux protocoles de signatures ECDSA et EdDSA, cf. la section 2.5.2, ainsi que le protocole d'échange de clefs ECDH, cf. section 2.5.1. Seule la version éphémère de l'ECDH est ici considérée.

Analyser autant de protocoles à de l'intérêt car chacun ne permet pas d'effectuer le même nombre de calculs de kP pour une même clef k. De plus, seul l'ECDH permet de choisir son point de base ce qui n'est pas le cas avec l'ECDSA ou l'EdDSA. Ainsi, l'ECDH nous permet d'effectuer deux calculs de multiplications scalaires avec une même clef k et de choisir le point de base pour un seul de ces calculs. L'ECDSA est le protocole le plus contraignant, car il n'autorise qu'un seul calcul de kP par scalaire k. L'EdDSA est le protocole le plus permissif, il permet d'effectuer autant de calculs kP pour une même clef k. Cependant, ni l'ECDSA, ni l'EdDSA n'autorisent de choisir le point d'entrée.

Il s'en suit que les contraintes protocolaires vont grandement réduire les possibilités d'attaques dans un cas réel. Nous effectuons notre analyse exclusivement sur l'opération de multiplication scalaire sur les courbes elliptiques. En effet, nous excluons les attaques visant les autres opérations des protocoles de signature.

Le tableau 3.1 résume les différentes sécurités intrinsèques dues au modèle BEC et de l'implémentation considérée, ainsi que les protections apportées par les différentes contremesures existantes. Il est à noter que la contremesure qui consiste à cacher la clef amène une nouvelle faiblesse. Ainsi, elle sera écartée afin d'éviter l'ajout de nouvelles faiblesses dans l'implémentation comme expliqué dans la section 3.4.4 et dans [FRVD08].

Dans la suite de cette section, nous allons détailler les différentes contremesures et leurs mises en œuvre. Certaines, comme la séparation de la clef, ont plusieurs versions que nous présenterons dans cette section.

3.4.1 Coordonnées aléatoires

Cette première contremesure proposée par Coron [Cor99] a pour but de changer aléatoirement la représentation projective du point de base afin d'endiguer toutes fuites d'information induite par le point de base entre plusieurs calculs de kP. Cette stratégie consiste

	Attaques par canaux auxiliaires																						Attaques par fautes									
	Attaques temporelles	SSCA						Verticales						HSCA				TA		Fausses erreurs		Courbe faible				DFA					
		Classiques	RSCA	ZSCA	SVSCA	Points de petit ordre	CSCA, DSCA	Bits d'adresse DSCA	Doublement	Doublement relatif	Propagation de retenue	*Big Mac*	HCSCA	HSVSCA	Classiques	Duguardin *et al.* [DPN+16b]	*C-Safe Error*	*M-Safe Error*	Point invalide	Courbe invalide	Courbe tordue	Classique	Changement de signe	Conversion des coordonnées
Corps binaires																															
Coordonnées												✓											✓								
Courbes binaires d'Edwards	✓	✓	✓									✓											✓								
Échelle de Montgomery	✓	✓							✓									✓													
Implémentation	✓	✓						✓										✓		✓		✓									
Critères de sécurité			✓																			✓									
ECDH	✓		✓	✓	✓		✓	✓											✓				✓								
ECDSA	✓		✓	✓	✓	✓	✓	✓	✓	✓				✓					✓				✓								
EdDSA	✓		✓	✓	✓	✓	✓	✓	✓	✓				✓																	
Coordonnées aléatoires	✓		✓	✓			✓		✓	✓		✓	✓		✓	✓							✓								
Isomorphisme aléatoire	✓						✓	✓	✓	✓		✓	✓		✓	✓							✓								
Point caché	✓		✓	✓	✓	✓	✓					✓	✓	✓	✓								✓								
Clef cachée	✓		✓	✓	✓		✓	✓			×			✓					✓				✓								
Séparation de la clef	✓		✓	✓	✓		✓	✓						✓					✓				✓								
Registres aléatoires								✓											✓												
Multiplication aléatoire												✓	✓	✓																	
Validation du point				✓		✓																	✓								
Validation de la cohérence			✓	✓																			✓								

TABLE 3.1 – Sécurités intrinsèques et contremesures pour une implémentation de la multiplication scalaire basée sur le modèle de courbes binaires d'Edwards. ✓ : la contremesure protège contre l'attaque. × : la contremesure amène une faiblesse contre cette attaque.

à choisir un nombre aléatoire r de la même taille que les coordonnées projectives de P et de le multiplier à chacune des coordonnées de P :

$$(rX : rY : rZ) \leftarrow (X : Y : Z) \tag{3.11}$$

La manière de changer la représentation du point P dépend du système de coordonnées projectives considéré. En effet, dans le cas du modèle de Weierstrass et des coordonnées jacobiennes, le point P sera remplacé par $(r^2X : r^3Y : rZ)$.

Les coordonnées différentielles projectives du modèle de BEC peuvent être ainsi changées par l'équation suivante :

$$(rW : rZ) \leftarrow (X : Z) \tag{3.12}$$

Cette contremesure a l'avantage d'avoir un coût négligeable par rapport à un calcul de kP, tout en offrant un ensemble de protections conséquent face aux attaques physiques. Cette contremesure sera utilisée dans notre implémentation de la cryptographie sur les BEC.

3.4.2 Isomorphismes aléatoires

L'idée d'utiliser des isomorphismes aléatoires comme contremesure fut introduite par Joye et Tymen [JT01]. Pour cela, en utilisant une courbe E' isomorphe à E, nous pouvons nous protéger des attaques nécessitant plusieurs calculs de kP. Sous forme de Weierstrass, deux courbes $E : y^2 = x^3 + ax + b$ et $E' : y^2 = x^3 + a'x + b'$ sont isomorphes s'il existe un $u \in \mathbb{F}_p^*$ tel que $u^4a' = a$ et $u^6b' = b$.

Il n'existe pas d'équivalent construit pour le modèle BEC. Malgré tout, une telle contremesure n'est pas envisageable, car elle changerait le paramètre d de la courbe utilisée lors du calcul de kP. Or, d est choisi volontairement petit (petit degré et creux) de façon à accélérer les calculs effectués à chaque étape de la multiplication scalaire, cf. section 4.3.2.

3.4.3 Point caché

L'idée de cacher le point de base en tant que contremesure est due à Coron [Cor99]. La stratégie de cette protection est de cacher les coordonnées du point de base en lui ajoutant un point aléatoire R. Ce point aléatoire est pré-calculé et fixe, car son coût de construction, non négligeable en termes de temps de calcul, est à prendre en considération. Nous effectuons alors :

$$Q = k(P + R) \tag{3.13}$$

Afin de retrouver le bon résultat, nous effectuons $Q - kR$, où $S = kR$ est pré-calculé. De plus, pour que cette contremesure soit efficace entre chaque calcul de kP, nous devons mettre à jour R et S en tirant un bit aléatoire t et en effectuant :

$$S \leftarrow (-1)^t 2S, \quad R \leftarrow (-1)^t 2S$$

Le coût de cette contremesure est réduit. De plus, cette contremesure a été améliorée dans [IIT04, MMM04, Mur14] en modifiant les formules permettant de cacher le point P.

Les protections considérées sont proches de celles apportées par l'ajout d'aléa dans les coordonnées. De plus, le changement du point de base par un point aléatoire est incompatible avec les accélérations induites par une sélection d'un point de base spécial, comme nous le verrons dans la section 4.3.3.

3.4.4 Clef cachée

Cette contremesure introduite par Coron [Cor99] ajoute de l'aléa dans la clef secrète k. Elle utilise le fait que la clef k est généralement modulo le cardinal de la courbe E. Ainsi, nous prenons le plus petit représentant de k modulo $|E|$. Cependant, ce représentant n'est pas unique et nous pouvons choisir aléatoirement un autre représentant k' de telle sorte que :

$$k' = k + r|E| \tag{3.14}$$

Où r est un nombre aléatoire. Le surcoût en temps de calcul de cette contremesure dépend de la taille de r. Nous prenons généralement r un nombre aléatoire de 32 bits.

Cette contremesure est construite pour endiguer toutes les fuites d'information possibles entre différents calculs de kP pour la même entrée k. Cependant, Fouque *et al.* [FRVD08] ont montré que cette contremesure est attaquable grâce à la propagation de la retenue. Ainsi, une telle contremesure ne peut pas être envisagée dans une implémentation ECC à moins d'y ajouter un nouveau biais de fuite d'information.

3.4.5 Séparation de la clef

Séparer la clef en deux est une contremesure introduite par Clavier et Joye [CJ01]. La stratégie est d'utiliser un nombre aléatoire r afin de séparer le calcul de $Q = kP$ en deux exponentiations dont les scalaires seront aléatoires.

Séparation additive : Cette version de la séparation de la clef fut la première introduite par Clavier et Joye [CJ01]. Elle utilise l'équation suivante :

$$Q = (k - r)P + rP \tag{3.15}$$

Où r est un nombre aléatoire de la taille de k afin de pouvoir cacher tous les bits de k. La conséquence de cette protection est de doubler le temps de calcul d'une multiplication scalaire. Cette version de la séparation de la clef a été attaquée de nombreuses fois ce qui rend la contremesure obsolète, [FRVD08, MV06].

Séparation Multiplicative : Cette variante est due à Trichina et Bellezza [TB02]. Ici, le calcul de $Q = kP$ est remplacé par $Q = k'S$, où :

$$S = rP \tag{3.16}$$

$$k' = kr^{-1}[|E|] \tag{3.17}$$

L'avantage de cette contremesure est de limiter la taille de l'aléa r.Il est courant de prendre un r de 32 bits. Dans ce cas, le scalaire k est entièrement caché car il est multiplié par l'inverse de r. Le surcoût de cette contremesure se réduit à une multiplication scalaire par un nombre de 32 bits ce qui équivaut au surcoût induit par la contremesure consistant à cacher la clef, cf. équation 3.15.

Séparation euclidienne : Cette autre version de la séparation de la clef est due à Ciet et Joye [CJ03]. Dans ce cas, le calcul $Q = kP$ est changé par $Q = k_1P + k_2S$, avec :

$$S = rP \tag{3.18}$$

$$k_1 = k[r] \tag{3.19}$$

$$k_2 = \left\lfloor \frac{k}{r} \right\rfloor \tag{3.20}$$

L'aléa r est ici choisi de manière à diviser par deux la taille de k. Ainsi, le surcoût de cette version est de 50% par rapport au calcul de kP. Cependant, nous pouvons réduire ce surcoût de temps de calcul en utilisant la technique de multiplication scalaire de Shamir [CJ03].

Ces différentes versions de la séparation de la clef provoquent un surcoût dans le calcul de kP où l'utilisation d'algorithme de multiplication scalaire est non-compatible avec le système de coordonnées différentielles des BEC. Cette contremesure n'est pas envisageable, d'autant que les protections apportées par celle-ci sont comparables à celle des coordonnées aléatoires (cf. section 3.4.1), gratuite en termes de surcoût de temps de calcul et d'algorithme.

3.4.6 Registres aléatoires

Cette contremesure introduite par Izu et Takagi [IT03] à pour but d'éviter les fuites exploitées par l'attaque visant les bits d'adresses par DSCA. L'idée de cette protection est d'ajouter de l'aléatoire dans les bits des adresses mémoires utilisées entre chaque calcul kP. Pour ce faire, nous avons besoin d'un point supplémentaire et d'un nombre aléatoire r de longueur égale à celle de la clef k. Cet aléa désigne alors les registres utilisés tout au long du calcul de kP. Cette contremesure a été étendue à l'Échelle de Montgomery par Izumi *et al.* [IISO10].

Dans le cas d'une implémentation de la cryptographie sur les courbes binaires d'Edwards, nous devrions intégrer ce genre de contremesure à notre implémentation. Cependant, cet ajout peut être évité en contrôlant, à la compilation, les adresses utilisées pour les points temporaires utilisés tout au long du calcul de la multiplication scalaire. De plus, nous pouvons faire que ces adresses ne dépendent pas de la clef et qu'elles restent fixes tout au long du calcul.

3.4.7 Multiplication aléatoire

La multiplication aléatoire a été introduite par Clavier *et al.* [CFG+10] afin de se prémunir des attaques horizontales. La stratégie de cette contremesure autorise la permutation aléatoire des différentes sous-multiplications effectuées lors d'une multiplication entre deux éléments de $\mathbb{F}_p$. Cette contremesure a été améliorée par la suite par Bauer *et al.* [BJPW13].

Elle est basée sur les algorithmes de multiplications des éléments de $\mathbb{F}_p$ utilisant l'instruction élémentaire de multiplication d'un processeur. Dans le cas des courbes en caractéristique 2, nous ne pouvons pas utiliser ce genre d'instruction. En effet, les algorithmes de multiplications ont alors des formes différentes comme l'algorithme de López-Dahab, cf. algorithme 2.2. Il paraît donc difficile d'appliquer le même type de contremesure sur un tel algorithme. De plus, les fuites exploitées par les attaques horizontales sur ces sortes de courbes elliptiques et d'algorithme n'ont pas été étudiées dans le cas des courbes binaires.

3.4.8 Validation du point

Une contremesure simple pour se prémunir des injections de fautes au sein du calcul kP ou sur les paramètres d'entrée du calcul, est de vérifier que le point considéré appartient à la courbe. Cette vérification doit être effectuée en amont du calcul sur le point de base et à la fin du calcul sur kP. Cette contremesure a été introduite par Biehl *et al.* [BMM00].

Dans le cadre des BEC nous pouvons utiliser l'équation de la définition 2.22, si nous avons les coordonnées affines ou projectives du point. Si le point est donné sous forme différentielle, nous pouvons utiliser la condition donnée par Bernstein dans [Ber09] étant

donné la coordonnée w du point et le paramètre d de la courbe :

$$\mathrm{Tr}\left(\frac{d}{w+w^2}\right) = 0 \tag{3.21}$$

$$\mathrm{Tr}\left((w+w^2)HT\left(\frac{d}{w+w^2}\right)\right) = 0 \tag{3.22}$$

Où, HT est la demi-trace. Malgré tout, cette vérification du point n'est pas parfaite. En effet, soit E une courbe binaire d'Edwards définie sur $\mathbb{F}_{2^k}$, avec $d_1 = d_2$. Si E est choisie pour des applications cryptographiques, alors $|E| = 4p$ avec p un nombre premier de l'ordre de 2^k. En effet, comme nous l'avons vu à la section 2.4 : si $d_1 = d_2$, alors E à un groupe d'ordre 4. De plus, E est choisie de telle sorte que $|E| = hp$ avec h petit et p un grand nombre premier. Ainsi, soit un point $P = (x, y)$ de $(\mathbb{F}_{2^k})^2$, alors la probabilité que P soit sur E est de l'ordre de $1/2^k$. Si $P = w$ un élément de $\mathbb{F}_{2^k}$, alors la probabilité que P soit un élément de E est de l'ordre de $1/4$.

Il s'en suit qu'une faute injectée sur la représentation w d'un point P a 25% de chance de ne pas être détectée du faite de cette contremesure. Cependant, le point P' fauté et non détecté par la contremesure est bien un point de E, mais ne correspond pas au point de base ou au point kP selon le moment d'injection de la faute. Ainsi, nous ne pouvons pas résoudre l'ECDLP avec ce point fauté. À ce stade, la faculté de l'exploitation du résultat par une attaque reste inconnue.

Une implémentation cryptographique basée sur les BEC, avec $d_1 = d_2$, devra aussi vérifier que le point de base n'est pas un point d'ordre 4. Cette condition équivaut à s'assurer que P est différent de $(0, 0)$, $(1, 1)$, $(1, 0)$ et $(0, 1)$.

3.4.9 Validation de la cohérence

Cette contremesure a été introduite par Giraud [Gir06] pour une implémentation RSA utilisant l'Échelle de Montgomery, mais peut être généralisée aux courbes elliptiques. En effet, lors d'un calcul de kP via l'algorithme 2.8 de l'Échelle de Montgomery, nous avons une relation qui lie les points temporaires R_0 et R_1 : $P = R_0 - R_1$, où P est le point de base. Nous pouvons utiliser cette relation pour vérifier la cohérence entre les points R_0 et R_1 tout au long du calcul de kP, ou seulement à la fin du calcul, ceci afin de s'assurer qu'aucune faute n'a été injectée lors de la multiplication scalaire.

Dans le cas des BEC, nous pouvons appliquer cette contremesure pour pallier à la faiblesse de la contremesure vérifiant la validité d'un point dans le cas des coordonnées différentielles, cf. section 3.4.8.

3.5 Synthèse théorique

Nous avons vu, tout au long de cette première partie de la thèse, les aspects théoriques relatifs à la cryptographique sur les courbes elliptiques utilisant le modèle alternatif de courbes binaires d'Edwards. De plus, dans ce chapitre, nous avons pu analyser, les faiblesses et les forces d'une telle cryptographie face aux attaques par canaux auxiliaires et par injections de fautes.

Il est important de noter que nous avons de nombreuses protections intrinsèques à cette cryptographie comme présenté dans le tableau 3.1. Ces protections sont essentiellement rendues possibles grâce à l'association de plusieurs propriétés inhérentes aux BEC, à leurs implémentations ou aux protocoles cryptographiques dans lesquels cette cryptographie sera employée. Ainsi, d'un point de vue théorique, les seules contremesures à ajouter à une implémentation des BEC seraient : la validation de la cohérence des points et l'ajout d'aléa au sein de la représentation projective des points.

Cette analyse de la protection des BEC et de leur implémentation est purement théorique et devra faire l'objet d'une vérification minutieuse de chaque propriété de sécurité. Nous effectuerons une partie de ces vérifications au chapitre 5 et nous développerons plus en profondeur les attaques par profilage au chapitre 6. Nous verrons alors qu'une mise à jour du tableau théorique 3.1 devra être faite.

Chapitre 4

Génération d'un ensemble de Courbes Binaires d'Edwards pour la cryptographie

Les mathématiques sont les ennemies acharnées de la mémoire, excellente par ailleurs, mais néfaste, arithmétiquement parlant !

Eugène Ionesco, *Théatre complet*

Sommaire

Dans ce chapitre, nous construisons un nouvel ensemble de courbes binaires d'Edwards. En effet, il existe peu de BEC construites pour tirer avantage des propriétés mathématiques propres à ce modèle. Le seul travail connu est celui de Bernstein [Ber09] qui a proposé des paramètres de courbes binaires d'Edwards. Cependant, le seul niveau de sécurité adressé est celui des 128 bits de sécurité équivalents à la courbe du NIST P256 ou la Curve25519 pour le modèle d'Edwards en large caractéristique développée par Bernstein *et al.* [BDL+11b].

Nous proposerons différents paramètres de BEC afin d'adresser différents niveaux de sécurité. Ainsi, un utilisateur pourra adapter le niveau de sécurité en fonction de la sensibilité de l'information à protéger et des contraintes de performance. Cette approche est notamment adoptée par la norme du NIST [Gal13].

Nous verrons que nous pouvons optimiser les paramètres de ces nouvelles courbes à différents niveaux. Cependant, une courbe elliptique prise au hasard ne peut être utilisée pour des applications cryptographiques. En effet, l'analyse de l'ECDLP a amené à l'élaboration d'un ensemble de critères de sécurité que doit vérifier une courbe elliptique pour des applications cryptographiques. Nous exposerons ces critères dans ce chapitre.

Le modèle de courbes binaires d'Edwards nécessite une étude particulière pour les intégrer au sein des protocoles standards d'échange de clefs et de signatures. Nous aborderons cette question dans ce chapitre. Ainsi, nous étudierons leur mis en œuvre pour des applications cryptographiques.

Nous conclurons ce chapitre par l'exposé des performances de notre implémentation de ces nouvelles courbes elliptiques.

4.1 Niveau de sécurité

Dans le chapitre 1, nous avons abordé la notion de niveau de sécurité. De plus, la section 2.5.3 expose comment ce niveau de sécurité peut être déduit de la taille des paramètres de la courbe elliptique considérée.

Il nous faut alors définir les différents niveaux de sécurité que nous considérerons par la suite dans notre génération et sélection de nouvelles courbes elliptiques. Il est important d'avoir une échelle de sécurité progressive afin de pouvoir adapter au mieux la sécurité et les performances en fonction de la sensibilité de l'information à protéger et des contraintes de l'environnement IoT. De plus, proposer différents niveaux de sécurité, nous permet aussi de faire évoluer la sécurité de la cryptographie basée sur ces nouvelles courbes en fonction de l'évolution des recommandations de différents organismes, cf. tableau 1.1.

Le tableau 4.1 présente les différents niveaux de sécurité que nous considérons par la suite. Cependant, les contraintes imposées à la sécurité des BEC et les optimisations proposées ne permettent pas d'être au plus proche de ces niveaux de sécurité. Certaines des

Niveaux de sécurité	Tailles des ECC
112	224
128	256
192	384
224	448
256	512

TABLE 4.1 – Niveaux de sécurité considérés et l'ordre de grandeur de la taille des paramètres des BEC et de la clef privée. Ces niveaux de sécurité et ces tailles sont donnés en bits.

courbes que nous proposerons auront un niveau de sécurité réel bien supérieur au niveau de sécurité qu'il lui sera associé.

Notons que nous avons ajouté le niveau de sécurité de 112 bits qui est inférieur au niveau de sécurité actuellement recommandé, cf. tableau 1.1. Nous avons fait le choix de proposer une BEC en deçà du niveau de sécurité recommandé afin d'offrir la possibilité aux applications IoT très contraintes en termes de performances d'avoir une solution performante avec un niveau de sécurité légèrement inférieur aux recommandations.

4.2 Critères de sécurité

Le choix des paramètres d'une courbe elliptique pour des applications cryptographiques est important dans les performances finales qu'aura notre protocole cryptographique. Cependant, ce choix de paramètres va aussi grandement influencer la sécurité inhérente à notre protocole cryptographique. Ainsi, les critères de sécurité que nous devrons vérifier permettent de garantir une certaine robustesse dans la résolution de l'ECDLP.

Il existe différents critères de sécurité et leur considération dans les différentes normes sur les courbes elliptiques n'est pas la même. En effet, dans le cas de la norme du NIST [Gal13], les critères de sécurité considérés doivent être améliorés afin de suivre l'évolution de nos connaissances sur la résolution de l'ECDLP.

Ces différents critères de sécurité sont publiés sur le site [saf]. Dans le cas des courbes binaires, Bernstein donne l'équivalent de ces critères dans [Ber09]. Il a noté que certains critères diffèrent, car l'utilisation des courbes binaires impose d'autres contraintes telles que le degré d'extension du corps de base.

Dans la suite de cette thèse, nous séparerons ces critères en deux catégories : les principaux et les secondaires. En effet, nous verrons que les critères principaux sont plus difficiles à vérifier. Tandis que, les critères de sécurités secondaires ne sont pas discriminants pour la majorité des courbes elliptiques.

4.2.1 Critères principaux de sécurités

Cet ensemble de critères de sécurité regroupe les différentes propriétés que doit vérifier une courbe elliptique pour des applications cryptographiques et qui ont un fort impact sur la sécurité de ces dernières.

- **Degré premier de l'extension de corps** : Les courbes elliptiques binaires sont définies sur une extension de corps $\mathbb{F}_{2^n}$ de $\mathbb{F}_2$. Le degré n de cette extension doit être premier. En effet, Gaudry, Hess et Smart [GHS02] ont montré que la primalité de ce degré permet de se prémunir d'attaques mathématiques utilisant les couplages sur les courbes elliptiques. De plus, n doit être choisi assez grand afin d'adresser les différents niveaux de sécurité définis dans le tableau 4.1.
- **Nombre de points** : Ce critère impose une forme particulière sur la friabilité du nombre de points de la courbe elliptique. Celui-ci doit être de la forme $|E| = hp$, avec p un grand nombre premier et h un petit cofacteur. Dans le cas des BEC, nous prendrons $h = 4$. Nous avons vu à la section 2.4.1 que si $d_1 = d_2$, alors la courbe E admet un groupe d'ordre 4. Ainsi, les BEC que nous générerons, auront que deux sous-groupes de torsions : celui de 4-torsion et celui de p-torsion.

  Cette contrainte est difficilement vérifiable par une courbe aléatoire. Ainsi, nous devrons vérifier ce critère pour un grand nombre de courbes avant de pouvoir en sélectionner une.

  Ce critère permet d'éviter des attaques qui tenteraient de résoudre l'ECDLP sur tous les groupes de torsions d'ordre premier afin de recomposer la clef k utilisée

dans le calcul de kP.

- **Nombre de points de la tordue** : Nous avons la même contrainte sur le nombre de points de la tordue de E. En effet, l'attaque par faute de Fouque *et al.* [FLRV08b], cf. section 3.3.2, permet de transporter l'ECDLP de la courbe E à la tordue E^{tw}. Ainsi, nous avons le même critère de sécurité de telle sorte que l'ECDLP soit difficilement à résoudre sur E^{tw}.

  Notons que la proposition 2.21 qui lie le cardinal d'une courbe elliptique à celui de sa tordue, permet de montrer que $|E^{tw}| = 2(2(2^{n-1} - p) + 1)$, avec $|E| = 4p$. Ainsi, le cofacteur de la tordue de E est égale à 2, et nous sélectionnerons alors les courbes E de telle sorte que $|E^{tw}| = 2p'$.

4.2.2 Critères secondaires de sécurité

Certains critères sont plus simples à être vérifiés par une courbe elliptique aléatoire. Cet ensemble de propriétés de sécurité a un impact moindre sur la sécurité de l'ECDLP. Malgré tout, ces indicateurs restent importants pour identifier une anomalie au sein d'une courbe elliptique qui peut potentiellement la sécurité de cette dernière.

- **j-invariant** : Ce critère concerne le j-invariant de la courbe E, celui-ci doit être générateur du groupe multiplicatif de l'extension de corps sur lequel est défini E. Dans le cas des BEC, le j-invariant est $1/d_1^8$, si $d_1 = d_2$.
- **Éviter les petits discriminants** : Le discriminant $\Delta_E = \mathrm{Tr}(E)^2 - 4q$ de la multiplication complexe, où $q = 2^n$, doit être divisible par un grand nombre premier. De plus, la multiplicité de ce diviseur doit être égale à 1. Il est habituel de considérer ce nombre premier comme grand s'il est de l'ordre de 2^{100} ou plus.
- **Grand degré d'embarquement** : Le degré d'embarquement de la courbe E doit être grand. Ce degré k est défini par le nombre premier p tel que $|E| = hp$ et 2^n, où n est le degré de l'extension de corps utilisé. k est alors l'ordre multiplicatif de 2^n dans $\mathbb{F}_p$. Le NIST recommande d'avoir un k supérieur à 20, [Gal13]. Cependant, les récentes améliorations de résolution de l'ECDLP sur les courbes à petit degré d'embarquement nous obligent à augmenter ce seuil [GG16, KJI+17]. Le standard définissant les courbes de Brainpool [LM10] utilise comme seuil $\frac{p-1}{100}$. Nous adopterons cette borne dans la génération de nouvelles courbes elliptiques.

4.3 Algorithme de génération de Courbes Elliptiques pour la cryptographie

Dans cette section, nous exposerons la méthode employée pour générer et sélectionner ce nouvel ensemble de courbes elliptiques binaires d'Edwards qui sera présenté dans la section 4.4.

Nous séparons en trois parties distinctes la construction de nouvelles courbes elliptiques.

La première étape est la sélection des polynômes irréductibles utilisés pour construire les différentes extensions de corps sur lesquelles seront définies les nouvelles BEC. Nous pourrions utiliser les modules définis par la norme du NIST [Gal13], mais nous pouvons améliorer les performances de l'arithmétique du corps fini en utilisant les travaux de Scott [Sco07].

La deuxième étape est le choix des paramètres d définissant les nouvelles BEC selon la définition 2.22, où $d = d_1 = d_2$. Pour sélectionner un d, nous vérifions que la BEC définie par ce paramètre est en accord avec les critères de sécurités principaux, cf. section 4.2.1.

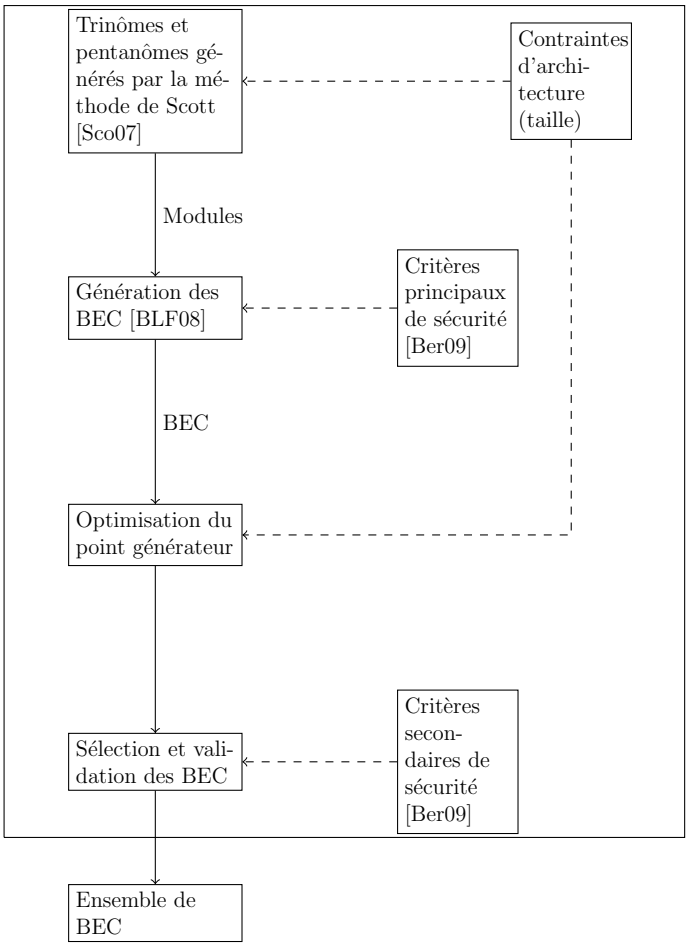


FIGURE 4.1 – Vu d'ensemble de la méthode de génération de nouvelles BEC.

Cette étape implique de considérer un grand nombre de d avant de pourvoir en sélectionner un. En effet, les critères de sécurités principaux sont discriminants dans la sélection des d.

La troisième et dernière étape consiste à choisir, pour chaque courbe sélectionnée à l'étape précédente, un point générateur qui sera par la suite utilisé dans les différents protocoles cryptographiques. Nous pouvons choisir un point générateur afin d'optimiser les calculs de kP quand il est utilisé, cf. section 4.3.3.

Pour les étapes 1 et 3, nous pouvons directement intégrer la contrainte de la taille d'une architecture d'un processeur afin d'optimiser les performances.

Les critères de sécurité secondaires sont vérifiés sur les différentes courbes finalement sélectionnées. Ces critères n'étant pas discriminants dans la sélection de nouvelles courbes, nous pouvons les vérifier en dernier afin d'économiser un temps de calcul important lors de la génération de nouvelles courbes.

La figure 4.1 synthétise les différentes étapes de la génération d'un nouvel ensemble de courbes binaires d'Edwards.

Dans la suite de cette section, nous considérerons que la taille de l'architecture que nous adressons est de 32 bits.

4.3.1 Optimisation de l'arithmétique $\mathbb{F}_{2^m}$

La sélection des polynômes irréductibles est essentielle. Ils serviront à la construction des extensions des corps sur lesquels seront construits les BEC. En effet, si nous prenons un

polynôme irréductible aléatoire, l'opération de réduction modulaire sera difficile et proche de celle de l'inversion. Ainsi, pour réduire le coût de la réduction modulaire, nous pouvons utiliser des polynômes irréductibles creux qui permettront d'avoir une arithmétique efficace.

Les recommandations du NIST [Gal13] basées sur celle de l'ANSI et de la norme X9.62 ou de Certicom [JMV01] préconisent d'utiliser des trinômes ou pentanômes. Les trinômes et pentanômes sont définis comme suit :

$$f(x) = x^n + x^a + 1, \qquad n > a > 0 \tag{4.1}$$

$$f(x) = x^n + x^a + x^b + x^c + 1, \qquad n > a > b > c > 0 \tag{4.2}$$

Si au moins un trinôme irréductible existe dans $\mathbb{F}_{2^n}[X]$, alors le NIST préconise de sélectionner celui avec le plus petit a. En effet, minimiser le degré du monôme intermédiaire permet de réduire le nombre d'opérations nécessaires à l'opération de réduction modulaire. Si aucun trinôme irréductible n'existe dans $\mathbb{F}_{2^n}[X]$, alors nous devons sélectionner un pentanôme dont les coefficients intermédiaires sont les plus petits possible. Pour les mêmes raisons que pour les trinômes, cette condition permet de diminuer le coût d'une réduction modulaire.

Dans le cas du NIST, les polynômes retenus sont :

$$\begin{aligned}
&\mathbb{F}_{2^{163}} : x^{163} + x^7 + x^6 + x^5 + 1\\
&\mathbb{F}_{2^{233}} : x^{233} + x^{74} + 1\\
&\mathbb{F}_{2^{283}} : x^{283} + x^{12} + x^7 + x^5 + 1\\
&\mathbb{F}_{2^{409}} : x^{409} + x^{87} + 1\\
&\mathbb{F}_{2^{571}} : x^{571} + x^{10} + x^5 + x^2 + 1
\end{aligned}$$

L'annexe A.1 associe les différents algorithmes de réduction modulaire par les polynômes du NIST. Ces algorithmes sont donnés pour une architecture de 32 bits. De plus, l'entrée pour chacun de ces algorithmes est un vecteur du double de la taille des éléments de $\mathbb{F}_{2^n}$, ceci correspond au résultat donné par l'algorithme 2.2 de la multiplication dans $\mathbb{F}_{2^n}$.

Cependant, Scott [Sco07] propose une stratégie différente afin de définir les polynômes irréductibles. Le but est d'économiser un grand nombre d'opérations de décalage imposées par le choix des polynômes du NIST. En effet, ce nombre de décalages va directement dépendre de la taille des registres l'architecture considérée. Chaque élément de $\mathbb{F}_{2^n}$ est représenté sur des registres de taille w qui correspond à la taille de l'architecture. Scott définit alors deux catégories de trinômes et pentanômes.

Définition 4.1 (Trinômes chanceux). *Un trinôme chanceux, ou* Lucky Trinomial*, est un trinôme $x^n + x^a + 1$ avec :*

$$n - a \equiv 0 \mod w$$

Pour un w choisit.

Définition 4.2 (Pentanômes chanceux). *Un pentanôme chanceux, ou* Lucky Pentanomial*, est un pentatnôme $x^n + x^a + x^b + x^c + 1$ avec*

$$n - a \equiv 0 \mod w, \quad n - b \equiv 0 \mod w, \quad n - c \equiv 0 \mod w$$

Pour un w choisi.

Un trinôme chanceux est rare et n'existe pas si 4 divise n ou que $n \equiv \pm 3 \mod 8$. Ce type de polynôme permet d'éviter un grand nombre de décalages dans leur algorithme de réduction si w est égal à la taille de l'architecture considérée.

Nous cherchons tous les trinômes et pentanômes chanceux entre 163 et 571. Ces bornes correspondent à la fenêtre dans laquelle sont définies les courbes elliptiques pour des applications cryptographiques. À l'aide d'un script Sage [The19], nous pouvons trouver tous ces polynômes spéciaux pour chaque n premier. Il s'en suit qu'il existe 6 trinômes chanceux dans cet intervalle pour un $w = 32$. Tandis qu'il existe un très grand nombre de pentanômes chanceux dans ce même intervalle.

La sélection finale de ces polynômes se fait en sélectionnant en priorité les trinômes ou les pentanômes avec des monômes intermédiaires de petits degrés. Nous sélectionnons un polynôme par niveau de sécurité défini dans le tableau 4.1. Ainsi, les polynômes sélectionnés sont :

$$\begin{aligned}
&\mathbb{F}_{2^{223}} : x^{223} + x^{159} + 1 \\
&\mathbb{F}_{2^{257}} : x^{257} + x^{65} + 1 \\
&\mathbb{F}_{2^{313}} : x^{313} + x^{121} + 1 \\
&\mathbb{F}_{2^{431}} : x^{431} + x^{303} + x^{239} + x^{111} + 1 \\
&\mathbb{F}_{2^{479}} : x^{479} + x^{255} + 1 \\
&\mathbb{F}_{2^{487}} : x^{487} + x^{295} + x^{167} + x^{39} + 1 \\
&\mathbb{F}_{2^{521}} : x^{521} + x^{489} + 1 \\
&\mathbb{F}_{2^{569}} : x^{569} + x^{441} + x^{313} + x^{121} + 1
\end{aligned}$$

Il est à noter que nous avons sélectionné un polynôme de degré 257 et 313 correspondant au niveau de sécurité de 128 bits. L'ajout du trinôme de degré 313 a pour but de proposer une taille intermédiaire entre 257 et 431. En effet, ce choix est motivé par l'idée de proposer une cryptographie adaptative à un grand nombre d'IoT contraints. De plus, deux polynômes sont proposés pour les niveaux de sécurité de 224 et 256 bits. En effet, les polynômes de degré 479 et 521 ne sont pas optimaux pour les architectures de 64 bits. Ainsi, dans l'objectif d'être aussi compatible pour ces architectures nous proposons deux alternatives.

L'annexe A.2 cite les différents algorithmes de réduction pour ces nouveaux polynômes. Nous pouvons observer que le nombre d'opérations de décalages a grandement baissé.

Le tableau 4.2 compare le coût en instruction logique (ou exclusif, et logique, décalages) pour effectuer une réduction modulaire avec les polynômes du NIST et ceux de Scott que nous avons sélectionnés.

4.3.2 Optimisation du paramètre de la Courbe Binaire d'Edwards

La seconde étape de la génération consiste à trouver des paramètres d de telle sorte que les BEC définies par ce paramètre vérifient les différents critères de sécurité.

Nous avons vu que le critère sur le nombre de points de la courbe est très discriminant et implique de tester un grand nombre de d. Le choix de d importe aussi dans l'optimisation du calcul de kP sur les BEC. En effet, l'algorithme 2.15 qui effectue une addition et un doublement entre deux points en coordonnées différentielles possède une multiplication par d qui est donc effectuée à chaque étape de la multiplication scalaire. Ainsi, en choisissant un d creux et de petit degré nous pouvons optimiser cette multiplication en la réduisant à un algorithme spécifique et peu coûteux.

Il s'en suit que pour trouver ces d, nous cherchons exhaustivement parmi tous les d possibles en commençant par les d de petit degré. De plus, pour calculer le nombre

Niveau de sécurité	Polynômes	Num. d'opérations logiques
80	$\boldsymbol{x^{163} + x^7 + x^6 + x^3 + 1}$	**74**
112	$x^{223} + x^{159} + 1$	38
	$\boldsymbol{x^{233} + x^{74} + 1}$	**62**
128	$x^{257} + x^{65} + 1$	44
	$\boldsymbol{x^{283} + x^{12} + x^7 + x^5 + 1}$	**152**
	$x^{313} + x^{121} + 1$	53
192	$\boldsymbol{x^{409} + x^{87} + 1}$	**108**
	$x^{431} + x^{303} + x^{239} + x^{111} + 1$	99
224	$x^{479} + x^{255} + 1$	78
	$x^{487} + x^{295} + x^{167} + x^{39} + 1$	114
256	$x^{521} + x^{489} + 1$	79
	$x^{569} + x^{441} + x^{313} + x^{121} + 1$	135
	$\boldsymbol{x^{571} + x^{10} + x^5 + x^2 + 1}$	**296**

TABLE 4.2 – Comparaison (en terme de nombre d'opérations logiques) entre différents polynômes comme module. Les polynômes en gras sont ceux du NIST.

de points de la BEC induite par d, nous utilisons l'algorithme 2.4 AGM permettant de compter le nombre de points d'une courbe elliptique. Si d vérifie le critère sur le nombre de points de la courbe elliptique, alors nous testons si la tordue vérifie aussi ce critère. En cas de succès, la courbe est sélectionnée pour l'étape suivante.

L'implémentation de l'algorithme AGM a été effectuée en langage C à l'aide de la bibliothèque de multiprécision FLINT [HJP13]. La recherche a été parallélisée de manière à profiter d'une ferme de calcul de 80 cœurs (Intel Xeon X5650 2.67GHz). Cet algorithme de génération de BEC a permis de trouver 48 nouvelles courbes elliptiques pour tous les corps finis définis précédemment dans la section 4.3.1. Il aura fallu un mois et demi de calculs pour aboutir à ce résultat.

Toutes les courbes vérifient les différents critères de sécurités. Citons ici à titre d'exemple de la courbe BEC313 qui est la courbe sélectionnée pour le corps défini par le polynôme de Scott de degré 313.

$$\begin{aligned}
\mathbb{F}_{2^{313}}[t] &= \frac{\mathbb{F}_2[X]}{(X^{313} + X^{121} + 1)} \\
d &= t^{38} + t^{33} + t^{28} + 1 \\
|E| &= 2^2 \times (2^{311} - 2380103563917833578062289719146201818615624 1745) \\
|E^{tw}| &= 2 \times (2^{312} + 4760207127835667156124579438292403637231248 3491)
\end{aligned}$$

- 313 est un nombre premier.
- Le nombre de points de E sur $\mathbb{F}_{313}$ est $4p_1$ où p_1 est le nombre premier

$$2^{311} - 2380103563917833578062289719146201818615624 1745$$

- Le nombre de points de la tordue de E, E^{tw}, est $2p_2$ où p_2 est le nombre premier

$$2^{312} + 4760207127835667156124579438292403637231248 3491$$

- Le j-invariant $1/d^8$ génère $\mathbb{F}_{2^{313}}$.
- $(2^{313} + 1 - 4p_1)^2 - 2^{315}$ est divisible par le nombre premier :

$$((2^{313} + 1 - 4p_1)^2 - 2^{315})/(-188793540488065899314670684485974 6511)$$

- L'ordre multiplicatif de 2^{313} modulo p_1 est $(p_1 - 1)/2$. L'ordre multiplicatif de 2^{313} modulo p_2 est $p_2 - 1$.

L'annexe E donne les paramètres des 48 BEC générées.

4.3.3 Optimisation du point générateur

La dernière étape de la génération d'un nouvel ensemble de courbes elliptiques binaires d'Edwards est la construction d'un point générateur optimisé.

D'après l'algorithme 2.15 d'addition et de doublement de deux points en coordonnées différentielles, à chaque étape du calcul de kP, nous devrons effectuer une multiplication par $w(P - Q)^{-1}$. Or, le point $P - Q$ est constant tout au long du calcul de kP si l'on utilise l'algorithme 2.8 de l'Échelle de Montgomery. De plus, ce point est exactement égal au point de base utilisé pour le calcul. Ainsi, en choisissant un point générateur G de telle sorte que $w(G)^{-1}$ soit creux et de petit degré nous pouvons accélérer la multiplication par cette constante.

Il résulte que nous pouvons effectuer une étape de l'Échelle de Montgomery en seulement 4M + 4S + 2m au lieu de 5M + 4S + 1m comme indiqué dans le tableau 2.4. Cette accélération correspond à près de 20% du calcul de la multiplication scalaire. Cependant, cette optimisation est profitable seulement aux protocoles cryptographiques utilisant un point fixe comme l'ECDSA ou l'EdDSA. Dans le cas, de l'ECDH, nous pouvons utiliser cette optimisation dans la première multiplication scalaire calculée par A ou par B.

Pour construire un tel point, il nous suffit de choisir un $w(G)^{-1}$ creux et de petit degré. De cette coordonnée différentielle, nous pouvons reconstruire les coordonnées affines du point G en réécrivant l'équation des BEC en fonction de w et x.

$$d(w + w^2) = x^4 + (1 + w + w^2)x^2 + (w + w^2)x \tag{4.3}$$

Ainsi, la résolution de cette équation permet de trouver les coordonnées x des points dont la représentation différentielle est w. En choisissant un x, nous pouvons retrouver la coordonnée y en utilisant l'équation générale des BEC de la définition 2.22.

En parcourant les $w(G)^{-1}$ de manière exhaustive nous pouvons retrouver tous les générateurs creux et de petits degrés. Nous pouvons améliorer la sélection de ce générateur en ajoutant une contrainte apportée par la taille de l'architecture du processeur considéré. En effet, on choisit $w(G)^{-1}$ tel que tous les monômes du polynôme soient d'un degré multiple de 32 ou 64, nous pouvons alors améliorer la multiplication par un tel polynôme.

Dans l'exemple de la BEC313 définie sur $\mathbb{F}_{2^{313}}$, avec comme paramètre $d = t^{38} + t^{33} + t^{28} + 1$, nous construisons par la méthode décrite précédemment le générateur donné par les coordonnées suivantes :

$$\begin{aligned} G_x =& \texttt{0x15c67e3024c7c27466e72a3391256e9a729fc158092053d89087c0f} \\ & \texttt{38408b214b0ade57363ea938} \\ G_y =& \texttt{0x15c67e3024c7c27446e72a3391256e9a529fc158092053d8b087c0f} \\ & \texttt{38408b214b0ade57363ea938} \\ w(G)^{-1} =& \texttt{0x10000000000000001} \end{aligned}$$

Il s'en suit que la multiplication par $w(G)^{-1} = t^{64} + 1$ nécessite de seulement 9 opérations `XOR`.

Nom	$\mathbb{F}_{2^n}$	Paramètre d	$w(G)^{-1}$
BEC223	$x^{223}+x^{159}+1$	$t^{64}+t^{36}+t^5+1$	$t^{32}+1$
BEC257	$x^{257}+x^{65}+1$	$t^{65}+t^{31}+t^{14}+1$	t^{192}
BEC313	$x^{313}+x^{121}+1$	$t^{38}+t^{33}+t^{28}+1$	$t^{64}+1$
BEC431	$x^{431}+x^{303}+x^{239}+x^{111}+1$	$t^{83}+t^{66}+t^{17}+1$	$t^{64}+1$
BEC479	$x^{479}+x^{255}+1$	$t^{73}+t^{29}+t^3+1$	$t^{64}+1$
BEC487	$x^{487}+x^{295}+x^{167}+x^{39}+1$	$t^{69}+t^{33}+t^{15}+1$	$t^{64}+1$
BEC521	$x^{521}+x^{489}+1$	$t^{66}+t^{29}+t^{28}+1$	$t^{32}+1$
BEC569	$x^{569}+x^{441}+x^{313}+x^{121}+1$	$t^{56}+t^{45}+t^{41}+1$	$t^{64}+1$

TABLE 4.3 – Nouvel ensemble de courbes binaires d'Edwards.

4.4 Un nouvel ensemble de Courbes Binaires d'Edwards

Finalement, nous obtenons une liste de BEC pour chaque corps défini dans la section 4.3.1. De plus, pour chacune de ces courbes, nous obtenons une liste de générateurs dont la représentation $w(G)^{-1}$ est optimisée comme expliquer dans la section 4.3.3. De ces listes nous devons sélectionner une et une seule courbe et générateur pour chacun des corps choisis.

Cette dernière sélection se fait en fonction des degrés des monômes du paramètre d et du générateur $w(G)^{-1}$. En choisissant des polynômes dont les monômes ont des degrés petits, nous pouvons accélérer la réduction du produit d'un élément de $\mathbb{F}_{2^n}$ et de d ou $w(G)^{-1}$. Le tableau 4.3 donne les différents paramètres du nouvel ensemble de courbes binaires d'Edwards. Ces courbes ont été construites à partir de la méthode présentée dans la figure 4.1.

L'annexe C présente tous les paramètres de ces nouvelles courbes elliptiques. Tandis que l'annexe D détaille les différents critères de sécurité pour ce nouvel ensemble de BEC.

4.5 Intégration du modèle de Courbes Binaires d'Edwards dans les protocoles standards

Dans la section 4.4, nous avons pu construire un nouvel ensemble de courbes elliptiques. De plus, nous avons identifié les différents éléments qui composent l'implémentation future de ces nouvelles courbes elliptiques : arithmétique dans $\mathbb{F}_{2^n}$, modèle de BEC, coordonnées différentielles mixtes avec Z commun et Échelle de Montgomery. À ce stade, nous pouvons implémenter et calculer kP.

Cependant, cette opération ne suffit pas pour utiliser ces courbes au sein d'applications cryptographiques. Pour cela, nous devons les intégrer dans des protocoles cryptographiques. Dans le cas de l'échange des clefs, nous devons les intégrer dans un protocole comme l'ECDH. Tandis que pour authentifier une entité, une signature sera nécessaire. Les protocoles ECDSA ou EdDSA définissent la construction et la vérification de signatures.

Le calcul de kP comme nous l'avons exposé utilise les coordonnées différentielles. Ainsi, un point $P=(x,y)$ sera représenté par $w(P)=x+y$. Il s'en suit que le résultat du calcul de kP sera connu sous forme différentielle. La résolution de l'équation 2.20 permet de retrouver la coordonnée du point kP à partir de $w(P)$, $w(kP)$ et $w(kP+P)$. Nous avons vu que le résultat final obtenu est (x,y) ou $(x,y)+(1,1)$.

Il subsiste donc une incertitude sur les coordonnées affines du point kP. Il s'en suit

que l'utilisation de ces coordonnées est, *a priori*, impossible à intégrer avec les formules des protocoles de signatures.

4.5.1 Coordonnées différentielles pour l'échange de clef

L'intégration des coordonnées différentielles pour l'ECDH est simple. En effet, nous pouvons nous abstenir de retrouver les coordonnées (x, y) du point secret partagé abP, cf. figure 2.6b. Habituellement, la coordonnée x est par la suite utilisée pour dériver les clefs de sessions lors de l'établissement d'une communication sécurisée via un protocole comme TLS [Res18].

Dans le cas de l'utilisation des coordonnées différentielles, le secret partagé est $w(abP)$. De plus, les données transférées par Alice et Bob sont normalement les coordonnées affines du point aP et bP. Dans le cas des coordonnées différentielles, nous pouvons seulement transférer $w(aP)$ et $w(bP)$. Ainsi, nous avons seulement un vecteur de données à transférer au lieu de deux.

Ce genre d'application est déjà utilisé, par exemple le protocole Curve25519 [LHT16], qui emploie la courbe éponyme sous forme de Montgomery, utilise seulement la coordonnée x dans l'établissement d'un secret partagé entre Alice et Bob.

4.5.2 Coordonnées différentielles pour la signature

Le cas des protocoles de signatures est plus délicat. Nous étudierons deux protocoles : l'ECDSA et l'EdDSA. Les spécificités de ses protocoles peuvent être trouvées dans la section 2.5.2. Dans la suite de cette section, nous considérerons exclusivement le cas de l'ECDSA : nous pouvons aisément étendre l'analyse à l'EdDSA car celui-ci a sensiblement la même structure que l'ECDSA.

Nous devons différencier les cas d'utilisation du système de coordonnées utilisé par Alice qui signe et Bob qui vérifie. Nous avons plusieurs cas :

- **S1** : Le premier scénario est le cas où Alice et Bob utilisent les coordonnées projectives pour calculer, respectivement, kP et $uP + vQ$.
- **S2** : Le deuxième scénario est le cas où Alice utilise les coordonnées différentielles et Bob les coordonnées projectives.
- **S3** : Le troisième scénario est le cas où Alice utilise les coordonnées projectives et Bob les coordonnées différentielles.
- **S4** : Le quatrième scénario est le cas où Alice et Bob utilisent les coordonnées différentielles. Dans le cas de Bob, les deux multiplications scalaires sont effectuées indépendamment. L'addition finale doit être faite en coordonnées projectives, car Bob ne connaît pas la différence entre uP et vQ.

Le premier scénario S1 est trivial. En effet, c'est le cas d'usage classique des protocoles de signatures.

Le deuxième scénario S2 ne pose pas de problèmes. En effet, si Alice utilise les coordonnées différentielles, alors le fait de retrouver les coordonnées affines de kP implique que $r = x[n]$ ou $r = x + 1[n]$. Ce résultat est dû à la perte d'information des coordonnées différentielles comme expliqué au début de ce chapitre. Ainsi, Alice obtient kP ou $kP + (1, 1)$. Par conséquent, la vérification effectuée par Bob est correcte dans les deux cas. Le cas où

Alice obtient kP est le cas habituel. Si Alice trouve $kP + (1, 1)$, alors $r = x + 1[n]$:

$$\begin{aligned} uP + vQ &= H(m)s^{-1}P + rs^{-1}Q \\ &= s^{-1}(H(m) + ar)P \\ &= k(H(m) + ar)^{-1}(H(m) + ar)P \\ &= kP \end{aligned}$$

Ainsi, la preuve de la cohérence du protocole de signature reste la même.

Le troisième scénario S3 peut entraîner une incohérence du protocole de signature. En effet, si Bob utilise les coordonnées différentielles alors il obtient uP ou $uP+(1, 1)$ et vQ ou $vQ+(1, 1)$. Dans le cas où Bob obtient uP et vQ ou $uP+(1, 1)$ et $vQ+(1, 1)$, nous sommes dans le cas habituel du protocole de signature, car $(1, 1) + (1, 1) = (0, 0)$. Les cas où Bob obtient uP et $vQ + (1, 1)$ ou $uP + (1, 1)$ et vQ sont identiques, nous traiterons le premier. Il s'en suit que Bob calcule $uP + vQ + (1, 1) = kP + (1, 1)$ d'après la démonstration de la cohérence de la signature ECDSA de la section 2.5.2. Ainsi, nous perdons la cohérence de la signature sur le bit de poids faible de r ce qui invalide la signature alors que cette dernière est correcte.

Le quatrième scénario S4 est équivalent au scénario S3. Nous obtenons ainsi une incohérence de la signature sur le bit de poids faible de r.

Finalement, la perte d'information des coordonnées différentielles peut induire une incohérence du protocole de signatures ECDSA si l'entité qui vérifie, ici Bob, utilise les coordonnées différentielles pour effectuer uP et vQ.

Première solution à l'incohérence de la signature

L'incohérence de la signature des scénarios S3 et S4 est problématique dans le cadre de l'intégration des BEC et de leur arithmétique dans les protocoles cryptographiques standards.

Une solution naïve serait d'employer les coordonnées projectives pour les opérations de la vérification de la signature. En effet, cette solution est équivalente aux scénarios S1 et S2. Cependant, l'utilisation des coordonnées projectives ralentit de façon drastique les performances de la multiplication scalaire. En effet, selon le tableau 2.4, 18M + 8S + 5m sont nécessaires pour effectuer une étape de l'Échelle de Montgomery au lieu de 5M + 4S + 4m pour les coordonnées différentielles mixtes. Nous pouvons réduire cette perte de performance en changeant l'algorithme de l'Échelle de Montgomery par un algorithme plus rapide avec les coordonnées projectives comme l'algorithme 2.6 à fenêtre glissante. Les opérations de l'étape de vérification ne manipulent que des données publiques, ainsi une sécurisation de la multiplication scalaire n'est pas nécessaire. Malgré tout, la quantité de mémoire nécessaire pour un tel algorithme peut le rendre inutilisable dans un environnement très contraint, cf. tableau 2.5. Il s'en suit que cette solution est fonctionnelle, mais coûteuse en termes de mémoire ainsi qu'en termes de coût d'implémentation : un autre algorithme de multiplication scalaire est nécessaire.

Une première solution satisfaisante est de lever l'incertitude. Ceci est rendu possible par le fait que cette dernière ne touche que le bit de poids faible des coordonnées affines d'un point. Ainsi, en fixant ce bit de poids faible à 0 ou à 1, nous levons les incertitudes des scénarios S3 et S4. Il s'en suit que lors du calcul de kP de la signature, nous fixons le bit de poids faible de x à 0 ou 1. Nous effectuons la même opération sur les coordonnées de uP et vQ.

Cette solution a pour conséquence de baisser le niveau de sécurité de 1 bit, car le bit de poids faible de r est connu quel que soit k. Une telle perte de sécurité est négligeable par rapport aux niveaux de sécurité actuels. En effet, 127 bits de sécurité ou 128 bits ne vont pas impliquer de grands changements dans la possibilité de la résolution de l'ECDLP.

Seconde solution l'incohérence de la signature

Une seconde solution pour remédier à l'incohérence de la signature est d'éviter les coordonnées projectives ou affines. La stratégie est d'utiliser entièrement les coordonnées différentielles sans retourner dans un système affine. Il s'en suit que r est donné par la coordonnée w et non par la coordonnée x.

Cependant, le calcul de $uP + vQ$ lors de la vérification pose un problème. En effet, nous ne connaissons pas la différence entre uP et vQ ce qui nous empêche d'effectuer l'addition finale. La stratégie est alors d'utiliser un algorithme de multiplication scalaire multidimensionnelle et compatible avec les contraintes des coordonnées différentielles. Un tel algorithme est proposé par Bernstein et Lange [BL17].

Algorithme de l'Echelle de Montgomery multidimensionnelle

L'Échelle de Montgomery (algorithme 2.8) a pour particularité d'avoir une différence constante entre les points temporaires R_1 et R_2. Cette différence est essentielle dans l'utilisation des coordonnées différentielles, car elle permet de s'affranchir de la principale contrainte de ce système de coordonnées. De plus, la différence entre R_1 et R_2 est exactement égale au point de base P.

L'idée d'une Échelle de Montgomery multidimensionnelle compatible avec les coordonnées différentielles implique l'utilisation de points temporaires dont les différences deux à deux sont constantes et connues. Un tel algorithme est rendu possible par l'utilisation des chaînes d'additions différentielles appelées chaînes de Lucas.

La construction d'une chaîne d'additions différentielles est un problème difficile. Akishita [Aki01] et Stam [Sta03] ont proposé différentes solutions pour effectuer une multiplication scalaire multiple. Nous porterons notre étude sur les travaux de Bernstein qui sont actuellement les plus aboutis en termes de chaîne d'additions différentielles uniformes.

L'idée de Bernstein est de calculer $uP + vG$ en t étapes où $t = \max(\log_2(u), \log_2(v))$. Chaque étape se décompose en trois parties effectuées les unes après les autres toujours dans le même ordre : `ADD` + `DBL` + `ADD`. Les entrées de chaque addition et du doublement dépendent de u et de v et de l'étape précédente. De plus, la différence entre chaque entrée d'une addition est exactement égale à $\mathcal{O}$ (cas particulier représentant un doublement), P, Q, $P - Q$ ou $P + Q$. L'algorithme a alors en sorties trois points parmi les quatre suivant : $nP + mQ$, $(n+1)P + mQ$, $nP + (m+1)Q$ et $(n+1)P + (m+1)Q$. On peut forcer l'une des sorties de telle sorte qu'elle soit égale à $nP + mQ$.

La figure 4.2 donne un exemple de multiplication scalaire multiple où l'on veut calculer $74P + 58Q$.

Pour obtenir une telle chaîne d'addition Bernstein définit une chaîne binaire, $C_D(A, B)$ avec $D \in \{0, 1\}$, de façon récursive. $C_D(A, B)$ est exactement égale à $C_d(a, b)$ à laquelle on a ajouté :

$$\begin{array}{rcl} (A + (A + 1 \mod 2))P & + & (B + (B + 1 \mod 2))Q \\ (A + (A \mod 2))P & + & (B + (B \mod 2))Q \\ (A + (A + D \mod 2)P & + & (B + (B + D + 1 \mod 2))Q \end{array}$$

où $a = \lfloor \frac{A}{2} \rfloor$, $b = \lfloor \frac{B}{2} \rfloor$, et :

$$d = \begin{cases} 0 & si(a + A, b + B) \mod 2 = (0, 1) \\ 1 & si(a + A, b + B) \mod 2 = (1, 0) \\ D & si(a + A, b + B) \mod 2 = (0, 0) \\ 1 - D & si(a + A, b + B) \mod 2 = (1, 1) \end{cases}$$

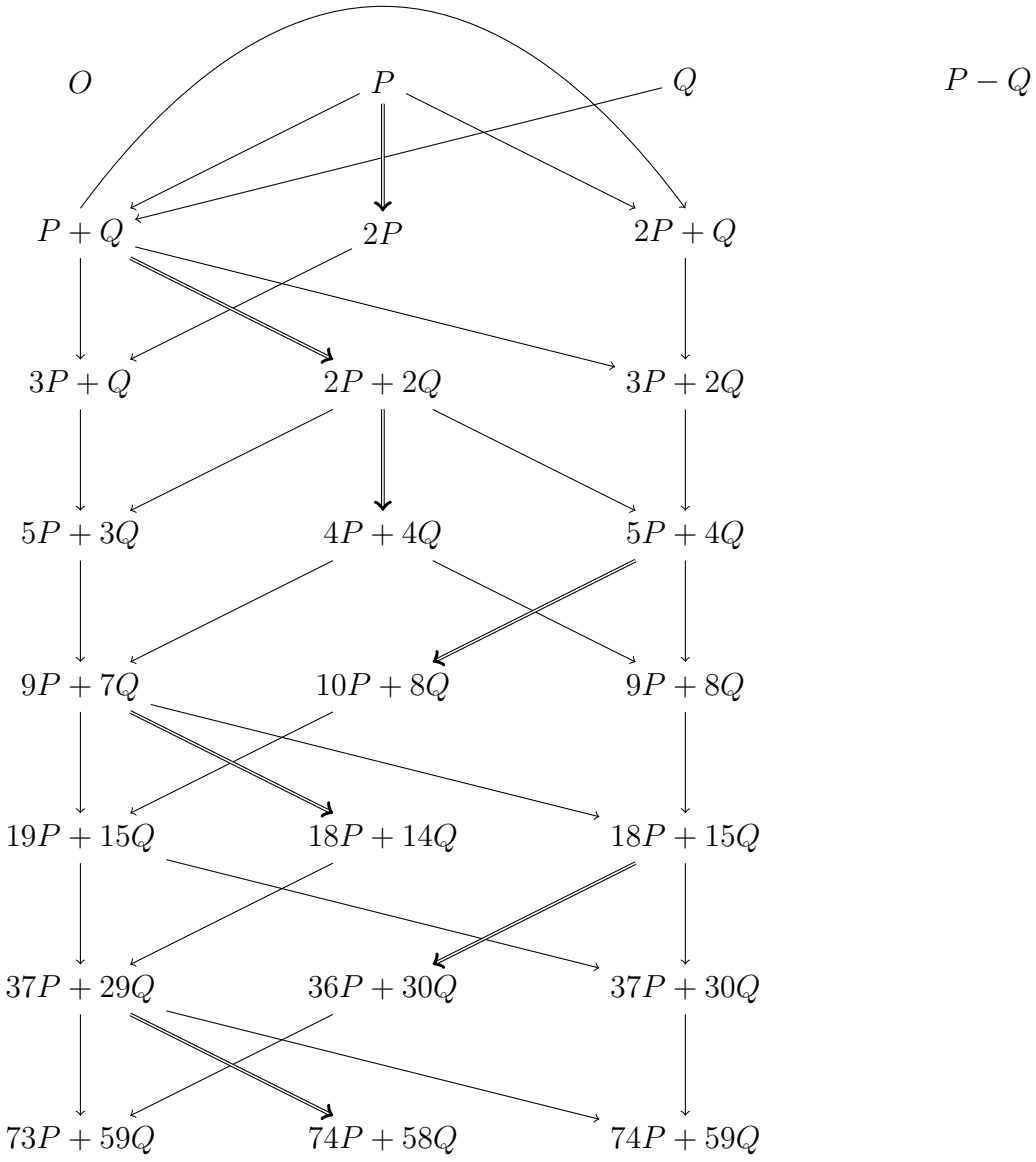


FIGURE 4.2 – Exemple d'une multiplication scalaire à deux dimensions.

L'initialisation est :

$$C_0(0,0) = C_1(0,0) = (O, P, Q, P - Q)$$

Arrangement de l'algorithme pour les systèmes embarqués :

L'algorithme tel que décrit ci-dessus est inutilisable dans un système embarqué. En effet, une implémentation naïve d'un tel algorithme implique qu'à l'étape i de l'algorithme, nous devons considérer les i bits de poids forts de chaque scalaire u et v. Il faut alors décomposer cet algorithme en étapes.

La première étape de l'algorithme est la construction de la série des $D \in \{0, 1\}$. Cette construction se fait en commençant par le bit de poids faible alors que la partie multiplication scalaire s'effectue à partir du bit de poids fort. Nous appellerons C_D la série de bits D, cette série peut être vue comme un nombre, dont le bit de poids fort, correspond au dernier bit D construit et le bit de poids faible correspond au premier D construit. Afin de forcer l'une des sorties de la multiplication scalaire à $nP + mQ$, le premier bit D est choisi comme étant égal au bit de poids faible de u. L'algorithme 4.1 expose la construction de C_D. Dans notre exemple $74P + 58Q$, on a $74 = 0b1001010$ et $58 = 0b0111010$ et en appliquant l'algorithme 4.1, nous obtenons $C_D = 0b0111010$.

Algorithme 4.1 Construction de C_D. n et m sont avec le bit de poids fort en premier, *i.e.* le n_0 de n est le bit de poids fort de celui-ci.

Entrées n, m
1: $C_D \leftarrow n_0$
2: $t \leftarrow \max(\log_2(n), \log_2(m))$
3: **Pour** $j = 1$ **à** $t - 1$ **Faire**
4: **Si** $n_j + n_{j+1} = 0$ **et** $m_j + m_{j+1} = 1$ **alors**
5: $C_D \leftarrow C_D | 0 \ll j$
6: **Sinon Si** $n_j + n_{j+1} = 1$ **et** $m_j + m_{j+1} = 0$ **alors**
7: $C_D \leftarrow C_D | 1 \ll j$
8: **Sinon Si** $n_j + n_{j+1} = 0$ **et** $m_j + m_{j+1} = 0$ **alors**
9: $C_D \leftarrow C_D | C_{D_j} \ll j$
10: **Sinon Si** $n_j + n_{j+1} = 1$ **et** $m_j + m_{j+1} = 1$ **alors**
11: $C_D \leftarrow C_D | (1 - C_{D_j}) \ll j$
12: **Fin Si**
13: **Fin Pour**
14: **Retourner** C_D

Le calcul de $uP + vQ$ est difficilement utilisable dans un système embarqué, *a priori*. En effet à chaque étape, nous devons ajouter et doubler des points, mais les entrées correspondantes dépendent directement des sorties de l'étape précédente. Il est ainsi difficile de savoir comment ajouter les points entre eux, car à chaque étape nous avons à disposition trois des éléments $aP + bQ$, $(a + 1)P + bQ$, $aP + (b + 1)Q$ et $(a + 1)P + (b + 1)Q$ qui correspondent aux sorties de l'étape précédente. Il faut ainsi connaître à chaque étape quels sont les éléments à disposition.

Pour ce faire, nous utilisons 4 registres mémoires, $R_0, ..., R_3$, qui vont contenir les points calculés durant la multiplication scalaire. Chaque registre contiendra le même type d'élément parmi $aP + bQ$, $(a + 1)P + bQ$, $aP + (b + 1)Q$ et $(a + 1)P + (b + 1)Q$. À chaque étape, seulement trois de ces registres seront mis à jour. Afin de savoir quels sont les registres à jour, nous pouvons coder l'état des registres sur 4 bits, un bit à 1 indique que le registre a été mis à jour à l'étape précédente, un bit 0 indique que la dernière mise à jour

du registre est vieille et qu'il n'a pas été mis à jour à l'étape précédente. Par exemple le codage $0b1011$ indique que seuls les registres R_0, R_2 et R_3 sont à jour. Ainsi, chaque entrée et sortie des sous-étapes d'addition et de doublement peuvent être directement identifiées en fonction des registres à jour et des trois bits courants de n, m et C_D. Nous présentons le codage de chaque étape en fonction des bits courants et des registres à jour de la manière suivante :

$$0bR_0R_1R_2R_3, 0bXYZ \rightarrow 0bR_0R_1R_2R_3 \rightarrow 0bR_0R_1R_2R_3$$

Où $0bR_0R_1R_2R_3, 0bXYZ$ indique respectivement quels sont les registres à jour et les 3 bits courants de u, v et C_D. Le second $0bR_0R_1R_2R_3$ désigne les registres alors utilisés en entrées de l'étape. Finalement, le troisième $0bR_0R_1R_2R_3$ montre dans quels registres doit être mis la sortie de l'étape.

Addition 1 :

$$\begin{aligned}
&0b0111, 0b00Z \rightarrow 0b0110 \rightarrow 0b0001\\
&0b0111, 0b10Z \rightarrow 0b0110 \rightarrow 0b0010\\
&0b0111, 0b01Z \rightarrow 0b0110 \rightarrow 0b0100\\
&0b0111, 0b11Z \rightarrow 0b0110 \rightarrow 0b1000\\
&0b1110, 0b00Z \rightarrow 0b0110 \rightarrow 0b0001\\
&0b1110, 0b10Z \rightarrow 0b0110 \rightarrow 0b0010\\
&0b1110, 0b01Z \rightarrow 0b0110 \rightarrow 0b0100\\
&0b1110, 0b11Z \rightarrow 0b0110 \rightarrow 0b1000\\
&0b1011, 0b00Z \rightarrow 0b1001 \rightarrow 0b0001\\
&0b1011, 0b10Z \rightarrow 0b1001 \rightarrow 0b0010\\
&0b1011, 0b01Z \rightarrow 0b1001 \rightarrow 0b0100\\
&0b1011, 0b11Z \rightarrow 0b1001 \rightarrow 0b1000
\end{aligned}$$

Doublement :

$$\begin{aligned}
&0bR_0R_1R_2R_3, 0b00Z \rightarrow 0b1000 \rightarrow 0b1000\\
&0bR_0R_1R_2R_3, 0b10Z \rightarrow 0b0100 \rightarrow 0b0100\\
&0bR_0R_1R_2R_3, 0b01Z \rightarrow 0b0010 \rightarrow 0b0010\\
&0bR_0R_1R_2R_3, 0b11Z \rightarrow 0b0001 \rightarrow 0b0001
\end{aligned}$$

Addition 2 :

$$\begin{aligned}
&0bR_0R_1R_2R_3, 0b000 \rightarrow 0b1010 \rightarrow 0b0010\\
&0bR_0R_1R_2R_3, 0b100 \rightarrow 0b0101 \rightarrow 0b0001\\
&0bR_0R_1R_2R_3, 0b010 \rightarrow 0b1010 \rightarrow 0b1000\\
&0bR_0R_1R_2R_3, 0b110 \rightarrow 0b0101 \rightarrow 0b0100\\
&0bR_0R_1R_2R_3, 0b001 \rightarrow 0b1100 \rightarrow 0b0100\\
&0bR_0R_1R_2R_3, 0b101 \rightarrow 0b1100 \rightarrow 0b1000\\
&0bR_0R_1R_2R_3, 0b011 \rightarrow 0b0011 \rightarrow 0b0001\\
&0bR_0R_1R_2R_3, 0b111 \rightarrow 0b0011 \rightarrow 0b0010
\end{aligned}$$

Algorithme 4.2 Échelle de Montgomery à 2 dimensions. Partie 1.

Entrées $n, m, C_dP, Q, P-Q, P+Q$

1: $R_0 \leftarrow O; R_1 \leftarrow O; R_2 \leftarrow O; R_3 \leftarrow O;$
2: $R'_0 \leftarrow O; R'_1 \leftarrow O; R'_2 \leftarrow O; R'_3 \leftarrow O;$
3: $CurReg = 0b0000; NewReg = 0b0000;$
4: **Si** $n_0 = 1$ **et** $m_0 = 0$ **et** $C_{D_0} = 0$ **alors**
5: $\quad R_2 = P+Q; R_1 = DBL(P); R_3 = ADD(P+Q, P); CurReg = 0b0111;$
6: **Sinon Si** $n_0 = 1$ **et** $m_0 = 0$ **et** $C_{D_0} = 1$ **alors**
7: $\quad R_2 = P+Q; R_1 = DBL(P); R_0 = P; CurReg = 0b1110;$
8: **Sinon Si** $n_0 = 0$ **et** $m_0 = 1$ **et** $C_{D_0} = 0$ **alors**
9: $\quad R_1 = P+Q; R_2 = DBL(Q); R_0 = Q; CurReg = 0b1110;$
10: **Sinon Si** $n_0 = 0$ **et** $m_0 = 1$ **et** $C_{D_0} = 1$ **alors**
11: $\quad R_1 = P+Q; R_2 = DBL(Q); R_3 = ADD(P+Q, Q); CurReg = 0b0111;$
12: **Sinon Si** $n_0 = 1$ **et** $m_0 = 1$ **et** $C_{D_0} = 0$ **alors**
13: $\quad R_0 = P+Q; R_3 = DBL(P+Q); R_1 = ADD(P+Q, P); CurReg = 0b1101;$
14: **Sinon Si** $n_0 = 1$ **et** $m_0 = 1$ **et** $C_{D_0} = 1$ **alors**
15: $\quad R_0 = P+Q; R_3 = DBL(P+Q); R_2 = ADD(P+Q, Q); CurReg = 0b1011;$
16: **Fin Si**
17: $t \leftarrow \max(\log_2(n), \log_2(m))$
18: **Pour** $j = 1$ **à** $t-1$ **Faire**
19: $\quad NewReg = 0b0000$
20: $\quad$ **Si** $CurReg = 0b1011$ **ou** $CurReg = 0b1101$ **alors**
21: $\quad\quad$ **Si** $n_j = 0$ **et** $m_j = 0$ **alors**
22: $\quad\quad\quad R'_3 = ADD(R_0, R_3); NewReg = 0b0001;$
23: $\quad\quad$ **Sinon Si** $n_j = 1$ **et** $m_j = 0$ **alors**
24: $\quad\quad\quad R'_2 = ADD(R_0, R_3); NewReg = 0b0010;$
25: $\quad\quad$ **Sinon Si** $n_j = 0$ **et** $m_j = 1$ **alors**
26: $\quad\quad\quad R'_1 = ADD(R_0, R_3); NewReg = 0b0100;$
27: $\quad\quad$ **Sinon Si** $n_j = 1$ **et** $m_j = 1$ **alors**
28: $\quad\quad\quad R'_0 = ADD(R_0, R_3); NewReg = 0b1000;$
29: $\quad\quad$ **Fin Si**
30: $\quad$ **Sinon Si** $CurReg = 0b0111$ **ou** $CurReg = 0b1110$ **alors**
31: $\quad\quad$ **Si** $n_j = 0$ **et** $m_j = 0$ **alors**
32: $\quad\quad\quad R'_3 = ADD(R_1, R_2); NewReg = 0b0001;$
33: $\quad\quad$ **Sinon Si** $n_j = 1$ **et** $m_j = 0$ **alors**
34: $\quad\quad\quad R'_2 = ADD(R_1, R_2); NewReg = 0b0010;$
35: $\quad\quad$ **Sinon Si** $n_j = 0$ **et** $m_j = 1$ **alors**
36: $\quad\quad\quad R'_1 = ADD(R_1, R_2); NewReg = 0b0100;$
37: $\quad\quad$ **Sinon Si** $n_j = 1$ **et** $m_j = 1$ **alors**
38: $\quad\quad\quad R'_0 = ADD(R_1, R_2); NewReg = 0b1000;$
39: $\quad\quad$ **Fin Si**
40: $\quad$ **Fin Si**

Algorithme 4.3 Échelle de Montgomery à 2 dimensions. Partie 2.

41: **Si** $n_j = 0$ **et** $m_j = 0$ **alors**
42: $R'_0 = DBL(R_0); NewReg = NewReg|0b1000;$
43: **Sinon Si** $n_j = 1$ **et** $m_j = 0$ **alors**
44: $R'_1 = DBL(R_1); NewReg = NewReg|0b0100;$
45: **Sinon Si** $n_j = 0$ **et** $m_j = 1$ **alors**
46: $R'_2 = DBL(R_2); NewReg = NewReg|0b0010;$
47: **Sinon Si** $n_j = 1$ **et** $m_j = 1$ **alors**
48: $R'_3 = DBL(R_3); NewReg = NewReg|0b0001;$
49: **Fin Si**
50: **Si** $C_{D_j} = 0$ **alors**
51: **Si** $n_j = 0$ **et** $m_j = 0$ **alors**
52: $R'_1 = ADD(R_0, R_2); NewReg = NewReg|0b0010;$
53: **Sinon Si** $n_j = 1$ **et** $m_j = 0$ **alors**
54: $R'_3 = ADD(R_1, R_3); NewReg = NewReg|0b0001;$
55: **Sinon Si** $n_j = 0$ **et** $m_j = 1$ **alors**
56: $R'_0 = ADD(R_0, R_2); NewReg = NewReg|0b1000;$
57: **Sinon Si** $n_j = 1$ **et** $m_j = 1$ **alors**
58: $R'_1 = ADD(R_1, R_3); NewReg = NewReg|0b0100;$
59: **Fin Si**
60: **Sinon Si** $C_{D_j} = 1$ **alors**
61: **Si** $n_j = 0$ **et** $m_j = 0$ **alors**
62: $R'_1 = ADD(R_0, R_1); NewReg = NewReg|0b0100;$
63: **Sinon Si** $n_j = 1$ **et** $m_j = 0$ **alors**
64: $R'_0 = ADD(R_0, R_1); NewReg = NewReg|0b1000;$
65: **Sinon Si** $n_j = 0$ **et** $m_j = 1$ **alors**
66: $R'_3 = ADD(R_2, R_3); NewReg = NewReg|0b0001;$
67: **Sinon Si** $n_j = 1$ **et** $m_j = 1$ **alors**
68: $R'_2 = ADD(R_2, R_3); NewReg = NewReg|0b0010;$
69: **Fin Si**
70: **Fin Si**
71: $CurReg \leftarrow NewReg; R_0 \leftarrow R'_0; R_1 \leftarrow R'_1; R_2 \leftarrow R'_2; R_3 \leftarrow R'_3;$
72: **Fin Pour**
73: **Retourner** R_0

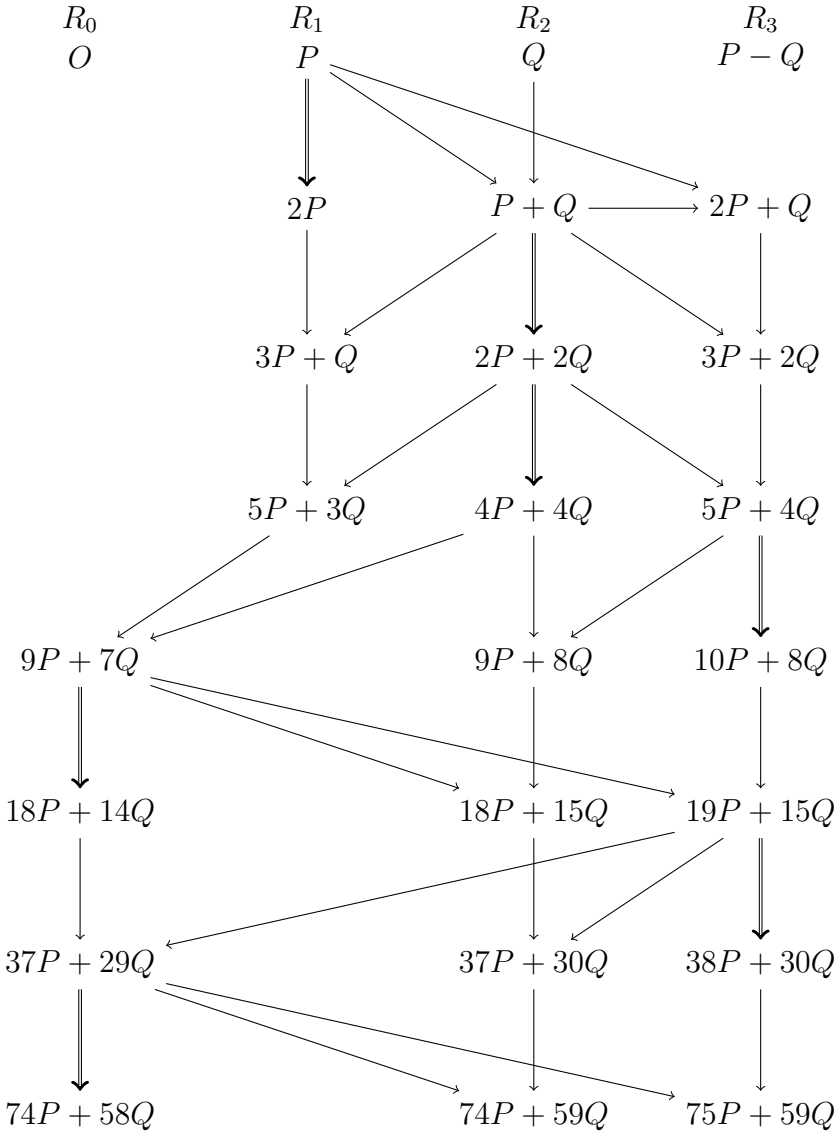


FIGURE 4.3 – Exécution d'une Échelle de Montgomery à 2 dimensions.

Ce codage est traduit par l'algorithme 4.3. Un exemple d'exécution pour $74P + 58Q$ avec $C_D = 0b0111010$ est donné par la figure 4.3.

La particularité d'un tel algorithme de multiplication scalaire multiple est la différence constante entre deux registres de points à jour. Ainsi, les différences possibles sont $\mathcal{O}, P, Q, P - Q$ et $P + Q$, nous pouvons donc utiliser un tel algorithme pour calculer $nP + mQ$ avec seulement leur représentation en coordonnées différentielles, w.

Nous pouvons alors intégrer les coordonnées différentielles dans un protocole de signature comme l'ECDSA où les étapes clefs sont données par l'algorithme 2.16 pour la signature et l'algorithme 2.17 pour la vérification. L'élément bloquant à l'utilisation des coordonnées différentielles est la multiplication scalaire multiple lors de la vérification. En effet, $uP + vQ$ doit être calculé où P est le point de base et Q la clef publique du signataire. Pour calculer cette multiplication scalaire multiple avec un algorithme tel que l'Échelle de Montgomery à deux dimensions en utilisant les coordonnées différentielles, nous devons connaître la différence $P - Q$ qui est exactement égale à $(a-1)P$. Ainsi, lors de la génération de la paire de clefs privée/publique nous devons aussi calculer $(a-1)P$ et sa représentation différentielle afin de pouvoir transmettre cette coordonnée en plus de la coordonnée de la clef publique. Cette opération peut être précalculée et donc n'influe pas sur les performances du protocole ECDSA. Les algorithmes 4.4 et 4.5 présentent l'ECDSA utilisant les coordonnées différentielles.

Algorithme 4.4 Signature ECDSA en coordonnées différentielles.

Entrées une clef privée a, un message m
Sorties: (r, s) la signature associée à m

```
H(m) ← hash(m)
Faire
    Faire
        k ← un aléa entre 1 et n − 1
        w ← w(kP)
    Tant que x = 0
    r ← w[n]
    s ← k^{-1}(H(m) + ar)[n]
Tant que s = 0
Retourner (r, s)
```

Algorithme 4.5 Vérification ECDSA en coordonnées différentielles.

Entrées m un message, (r, s) la signature associée, $w(Q)$ la clef publique associée à la clef privée qui a servi à signer le message, $w(P - Q)$.
Sorties: Retourne vrai si la signature est correcte, faux sinon

```
H(m) ← hash(m)
u = H(m)s^{-1}[n]
v = rs^{-}1[n]
w = w(uP + vQ)
Si w ≠ r alors
    Retourner Faux
Fin Si
Retourner Vrai
```

Caractéristiques	Microcontrôleur
Architecture	RISC V
Taille	32 bits
Cadence	100 MHz
Mémoire d'instructions	256 Ko
Mémoire des données	64 Ko
Technologie	28 nm (GF28SLP)

TABLE 4.4 – Caractéristiques du microcontrôleur RISC V.

4.6 Implémentation et performances

Dans ce chapitre, nous avons construit une nouvelle cryptographie basée sur les courbes binaires d'Edwards. De plus, nous avons étudié l'intégration de cette cryptographie au sein des protocoles cryptographiques standards d'échange de clefs (ECDH) et de signatures (ECDSA, EdDSA).

Le nouvel ensemble de courbes elliptiques développé au chapitre 4 et son utilisation au sein de l'ECDH et de ECDSA a fait l'objet d'une implémentation logicielle complète. Cette dernière a été effectuée sur un microcontrôleur dont les caractéristiques sont présentées dans le tableau 4.4.

Le cœur de ce microcontrôleur est basé sur le jeu d'instruction RISC V [WAD17]. De plus, nous avons vu que dans la section 1.1 que l'architecture RISC V est modulaire, sur cette implémentation les jeux d'instructions suivants ont été embarqués :

- Instructions sur les entiers (RV32I).
- Instructions compressées (RV32C).
- Instructions de multiplication et division d'entier (RV32M).

L'implémentation de cette cryptographie basée sur les BEC a été organisée comme suit :

- Arithmétique sur $\mathbb{F}_{2^n}$ (addition, élévation au carré, multiplication, division, réduction).
- Arithmétique sure les BEC (coordonnées différentielles mixtes avec Z commun, conversion des coordonnées, Échelle de Montgomery, Échelle de Montgomery à 2 dimensions).
- Protocoles cryptographiques (signature ECDSA, vérification ECDSA, arithmétiques des entiers)
- Contremesures (ajout d'aléa dans les coordonnées projectives, validation du point, trace, demi-trace)

La figure 4.4 schématise l'architecture de notre implémentation logicielle du nouvel ensemble de BEC du tableau 4.3.

L'implémentation a été faite à la fois en Assembleur et en langage C. L'intérêt de l'assembleur est de pouvoir parfaitement contrôler la gestion des registres du cœur et des instructions utilisées. La partie assembleur concerne les primitives relatives à l'arithmétique binaire.

Les opérations clefs de la cryptographie sur les courbes elliptiques sont la multiplication scalaire kP que l'on retrouve dans l'ECDH et la signature ECDSA, et la multiplication scalaire à deux dimensions $uP + vQ$ qui est l'opération principale de la vérification de signature ECDSA.

Ainsi, nous donnerons les performances de notre implémentation en termes de performance du calcul de kP et $uP + vQ$. Le tableau 4.5 indique les différents temps de calcul

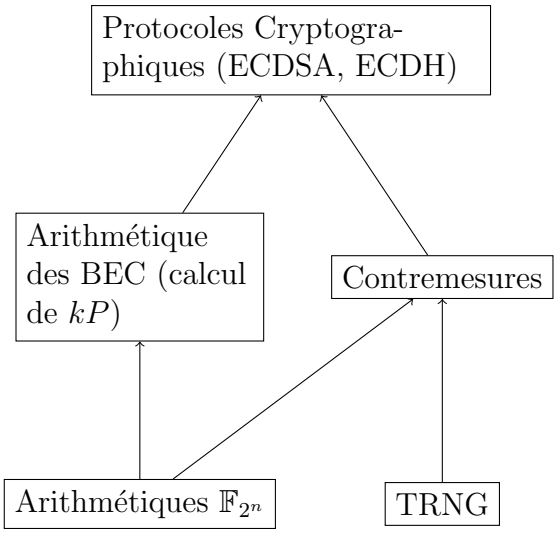


FIGURE 4.4 – Vu d'ensemble de l'architecture de notre implémentation logicielle des BEC du tableau 4.3.

Courbes	kP			$uP + vQ$
	Point de base optimisé	Point de base aléatoire		
	w-coor.		proj. coor.	alg. 4.3
BEC223	32 ms	39 ms	137 ms	95 ms
BEC257	46 ms	57 ms	201 ms	138 ms
BEC313	79 ms	96 ms	342 ms	229 ms
BEC431	188 ms	231 ms	821 ms	538 ms
BEC479	242 ms	299 ms	1064 ms	693 ms
BEC487	264 ms	326 ms	1156 ms	757 ms
BEC521	316 ms	390 ms	1388 ms	904 ms
BEC569	396 ms	489 ms	1744 ms	1266 ms

TABLE 4.5 – Performances de l'implémentation des BEC sur un composant réel d'une architecture RISC V 32 bits cadencée à 100 MHz.

nécessaires pour effectuer ces calculs pour les différentes courbes binaires d'Edwards que nous avons générées au chapitre 4. Ces performances ont été mesurées à l'aide d'un oscilloscope et d'une GPIO (*General Pin In Out*) qui est levée au début du calcul de kP et baissée à la fin de ce même calcul.

Les figures 4.5 et 4.6 donnent l'évolution des performances du tableau 4.5 en fonction de la taille de la courbe elliptique. Il est à noter que les performances de l'opération $uP+vQ$ sont bien meilleures que celles de kP en coordonnées projectives. Cependant, les temps de calcul pour effectuer $uP+vQ$ avec l'Échelle de Montgomery à 2 dimensions sont supérieurs au double du temps de calcul nécessaire pour effectuer kP en coordonnées différentielles avec un point de base aléatoire. Ceci s'explique par le fait que pour effectuer une étape de kP il nous faut 5M + 4S + 1m et donc 10M + 8S + 2m pour effectuer une étape de $uP + vQ$ si nous effectuons les multiplications scalaires indépendamment. Tandis que pour effectuer une étape de $uP + vQ$ en coordonnées différentielles, 11M + 5S + 3m sont nécessaires pour une étape du calcul.

Il s'en suit que la solution d'intégration des coordonnées différentielles de la section

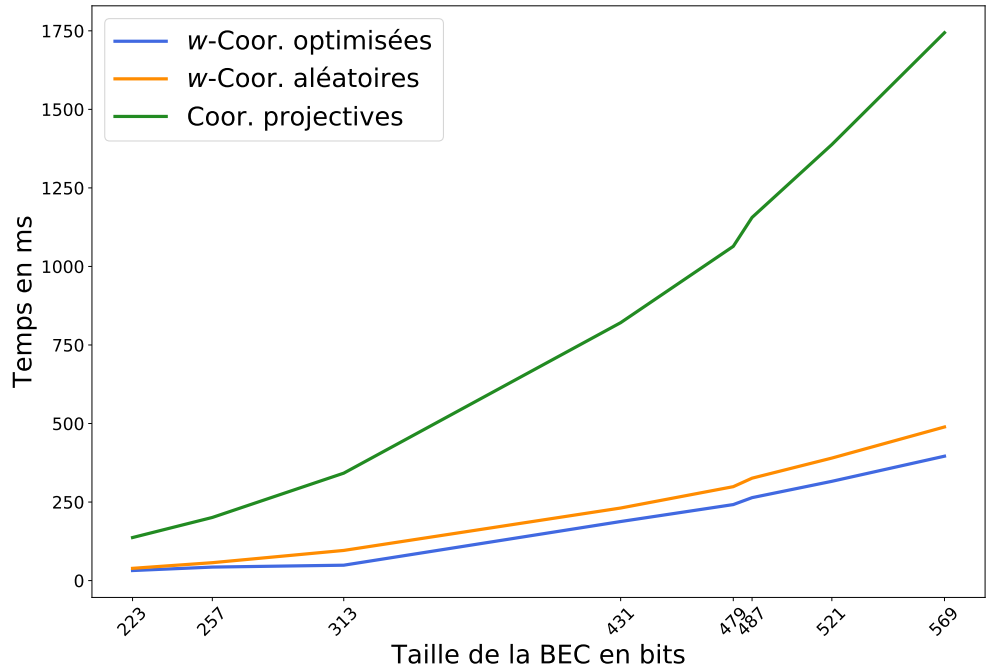


FIGURE 4.5 – Évolution des performances du tableau 4.5 pour le calcul de kP. En bleu les coordonnées différentielles avec le point de base optimisé, en orange les coordonnées différentielles avec un point de base aléatoire, en vert les coordonnées projectives et le point de base aléatoire.

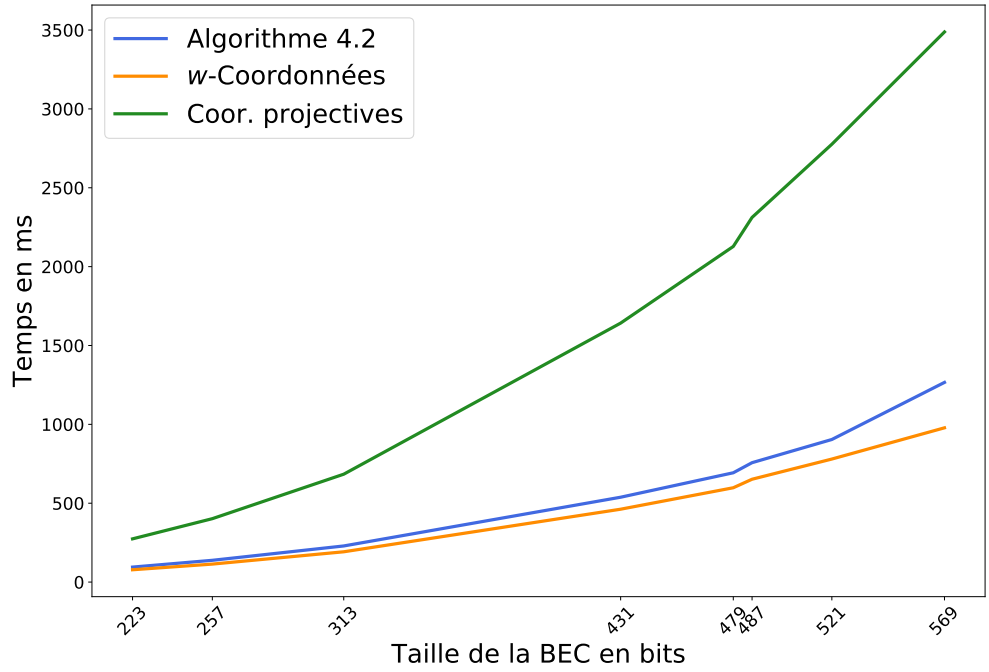


FIGURE 4.6 – Évolution des performances du tableau 4.5 pour le calcul de $uP + vQ$. En bleu les coordonnées différentielles avec l'algorithme 4.3, en orange les coordonnées différentielles avec deux calculs de kP, en vert les coordonnées projectives avec deux calculs de kP.

Tailles	BEC	[TPG15]	[Mac17]	[WUW13]	[dCUVHV14]	[DHH+15]	[dG15]	[NM16]	[FGLCLT08]	[LSC+19]
159	-	-	-	-	-	-	-	-	1 104	-
160	-	-	1 868	-	-	-	-	-	-	-
191	-	-	-	-	-	-	-	-	1 690	-
192	-	6 336	2 369	-	-	-	-	-	-	-
223	3 191	-	-	-	-	-	-	-	2 445	-
224	-	8 160	3 569	-	-	-	-	-	-	-
233	-	-	-	-	1 864	-	-	-	-	-
255	-	-	-	-	-	3 590	2 661	6 245	3 396	-
256	-	11 712	6 817	10 730	-	-	-	-	-	16 900
257	4 641	-	-	-	-	-	-	-	-	-
313	7 827	-	-	-	-	-	-	-	-	-
384	-	19 392	-	-	-	-	-	-	-	52 251
431	18 694	-	-	-	-	-	-	-	-	-
479	24 189	-	-	-	-	-	-	-	-	-
487	26 330	-	-	-	-	-	-	-	-	-
521	31 482	33 696	-	-	-	-	-	-	-	84 833
569	39 503	-	-	-	-	-	-	-	-	-

TABLE 4.6 – Performances de l'état de l'art en millier de cycles en fonction de la taille de la courbe elliptique.

4.5.2 est moins coûteuse en temps que la solution proposée dans cette section. Malgré tout, l'algorithme de l'Échelle de Montgomery à deux dimensions peut être amélioré en utilisant une chaîne d'addition plus performante. Dans [Ber06, BL17], Bernstein propose une solution alternative.

Performances de l'état de l'art :

Afin de comparer les performances de notre implémentation à l'état de l'art, nous considérerons le nombre de cycles nécessaires pour effectuer une multiplication scalaire. En effet, les différences de fréquences de microcontrôleurs de la littérature rendent la comparaison en seconde du temps de calcul non significatif. De plus, la taille du chemin de données l'architecture considérée influence grandement les performances finales de l'opération kP, ainsi nous considérerons exclusivement les implémentations sur une architecture de 32 bits.

De nombreux articles présentent des performances d'une implémentation du calcul de kP, [TPG15, Mac17, WUW13, dCUVHV14, DHH+15, dG15, NM16, FGLCLT08, LSC+19]. Dans ces études, différents modèles et tailles de courbes elliptiques sont considérés. Le tableau 4.5 permet de comparer les performances d'un calcul de kP pour une taille de courbes données quels que soit le modèle ou les algorithmes utilisés. La figure 4.7 représente le positionnement de notre implémentation par rapport à l'État de l'art.

Il est à noter que notre implémentation est la meilleure dans de nombreux cas. Les implémentations ayant de meilleures performances utilisent des algorithmes de multiplication scalaire plus rapides, mais moins sécurisés. De plus, l'architecture RISC V autorise l'utilisation d'instructions personnalisées. Ainsi, les performances de notre implémentation pourront être grandement améliorées avec une instruction de multiplication sur les corps binaires. Une telle instruction permettrait de se rapprocher des implémentations des courbes définies sur un corps à large caractéristique.

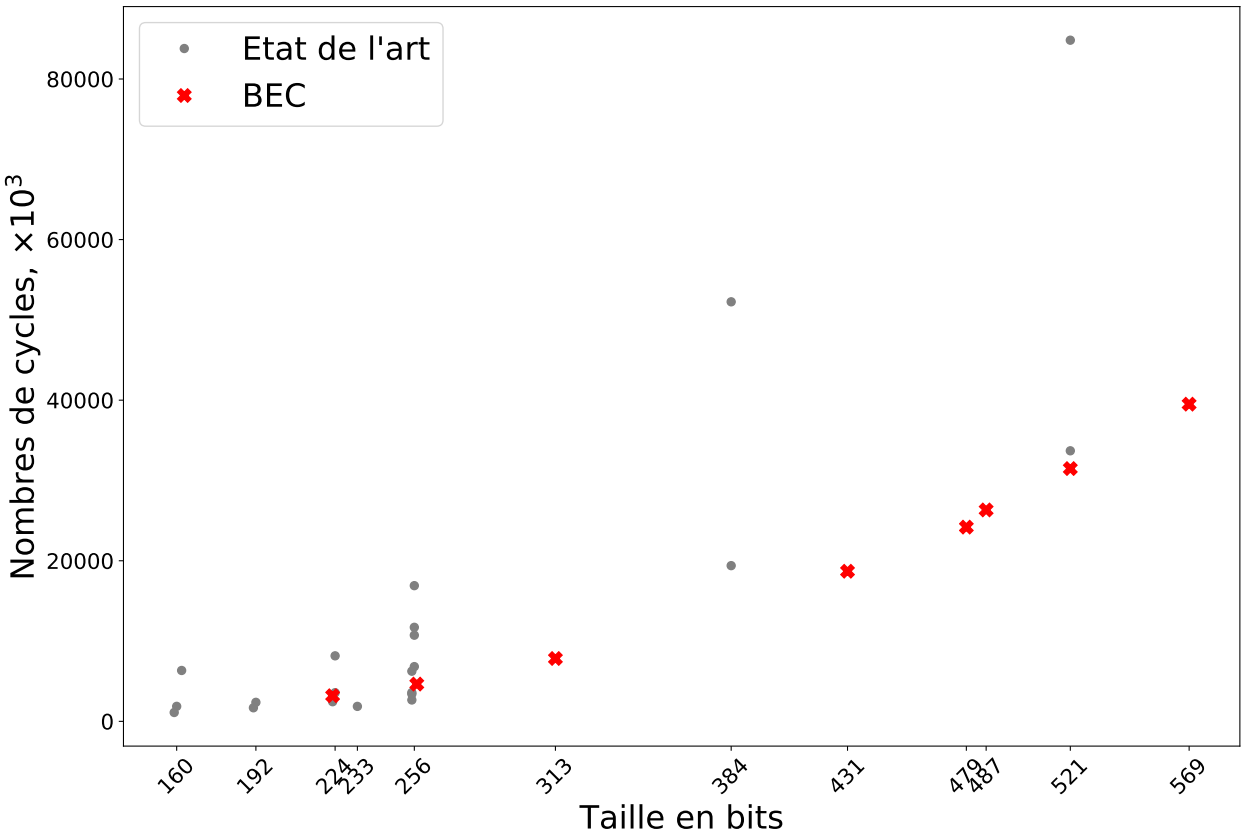


FIGURE 4.7 – Positionnement de l'implémentation des BEC par rapport à l'État de l'art, tableau 4.6.

4.7 Conclusion

Dans ce chapitre, nous avons construit un nouvel ensemble de courbes elliptiques utilisant le modèle de courbes binaires d'Edwards. Pour générer un tel ensemble, nous avons considéré les critères de sécurité de la section 4.2. De plus, les différents paramètres de cet ensemble intègrent plusieurs optimisations : au niveau du polynôme irréductible du corps de base (section 4.3.1), du choix de paramètre d creux (section 4.3.2) et du point de base utilisé (section 4.3.3). Il s'en suit un nouvel ensemble de courbes binaires d'Edwards pour différents niveaux de sécurité (tableau 4.3). Ces travaux ont fait l'objet d'un brevet (FR3071081) et d'une publication à Secrypt 2018 [LF18].

Nous avons étudié l'intégration de ces nouvelles courbes au sein des protocoles cryptographiques standards : ECDH pour l'échange de clefs, ECDSA et EdDSA pour l'authentification. Il nous est apparu que l'utilisation des coordonnées différentielles peut amener une impossibilité de vérifier une signature ECDSA ou EdDSA. Ceci est dû à l'approximation faite lors du passage des coordonnées affines aux coordonnées différentielles. Dans ce cas, retrouver les coordonnées (x, y) à partir de $w = x + y$ donne comme résultat le point (x, y) ou $(x, y) + (1, 1)$. Cette approximation peut rendre une signature correcte impossible à vérifier. Nous proposons deux solutions pour pallier ce problème.

La première vient du fait que cette approximation concerne seulement les bits de poids faible de x et y car $(x, y) + (1, 1) = (x + 1, y + 1)$. Ainsi, en forçant le bit de poids faibles des coordonnées affines du point à 0 ou 1, nous pouvons fixer l'incertitude ce qui permet de vérifier toutes les signatures ECDSA et EdDSA.

La seconde solution consiste à utiliser les coordonnées différentielles tout le long du protocole de signature. Dans ce cas, w est utilisé à la place de x pour construire la signature. Cependant, la vérification d'une signature générée par cette méthode peut être problématique. Nous avons vu à la section 4.5.2 que le calcul $uP + vQ$ de la vérification nécessite une Échelle de Montgomery à deux dimensions. Bernstein et Lange proposent un tel algorithme dans [BL17]. Ce dernier a été choisi afin d'effectuer notre implémentation de ce nouvel ensemble de courbes elliptiques.

Dans la section 4.6, nous décrivons notre implémentation et la comparons à l'état de l'art. Cette comparaison nous a permis de conclure que notre implémentation est performante. L'ajout d'une instruction de multiplication polynomiale dans $\mathbb{F}_{2^n}$ augmenterait les performances de celle-ci.

Chapitre 5

Évaluation de la sécurité des BECs face aux attaques par canaux auxiliaires

L'expérience ne nous offre au premier coup d'œil qu'un chaos suivi d'un autre chaos.

John Stuart Mill,

Sommaire

Dans le chapitre 3, nous avons détaillé les différentes familles d'attaques connues par canaux auxiliaires contre les implémentations sur les courbes elliptiques. De plus, dans le chapitre 4 nous avons construit et implémenté une cryptographie basée sur les courbes binaires d'Edwards. Dans ce chapitre, nous nous proposons d'effectuer une première évaluation de notre implémentation de la cryptographie basée sur les BEC. Nous nous concentrerons sur les attaques par canaux auxiliaires. Nous vérifierons les différentes propriétés de sécurités que nous avons émises dans le tableau 3.1.

Il est à noter que dans cette évaluation, nous ne détaillerons pas entièrement les attaques par profilages qui seront traitées dans le chapitre 6. Nous ne traiterons pas les attaques horizontales et les attaques par fautes qui devront être étudiées en profondeur dans de futurs travaux.

Toutes les attaques sont effectuées sur le microcontrôleur dont les caractéristiques sont inscrites dans le tableau 4.4. De plus, la mesure physique considérée pour effectuer les attaques est la consommation de courant. Pour effectuer ces mesures, nous utiliserons le dispositif d'évaluation décrit dans la figure 3.2. L'attaque présentée dans ce chapitre a été faite en observant le rayonnement EM du microcontrôleur. Le dispositif d'attaque est présenté par la figure 3.2.

5.1 Évaluation de l'implémentation face aux attaques par canaux auxiliaires

Cette section présente les différents résultats que nous avons pu obtenir sur la mise en place d'attaques par canaux auxiliaires sur notre implémentation de la cryptographie des courbes binaires d'Edwards. Nous évaluerons aussi l'impact des contremesures que nous intégrons face aux attaques par canaux auxiliaires.

Pour mémoire, les deux contremesures considérées sont :

- La validation du point (comportant la vérification que le point de base n'est pas d'ordre 4), cf. section 3.4.8.
- L'ajout d'aléatoire dans la représentation des coordonnées projectives, cf. section 3.4.1.

Précisons que cette évaluation n'est pas absolue, car il est difficile de démontrer qu'une implémentation ou une contremesure est entièrement sécurisée face à une attaque par canaux auxiliaires. En effet, nous passons un temps donné pour mener une attaque, si elle échoue, nous ne pouvons garantir qu'en passant plus de temps sur cette dernière, l'attaque est impossible. Malgré tout, nous pouvons avoir un degré de confiance si nous n'arrivons pas à mener l'attaque. Nous nous efforcerons donc de garantir un degré de confiance maximum dans notre évaluation.

Toutes les mesures de courant ont été effectuées avec le même oscilloscope dont les caractéristiques sont présentées dans le tableau 5.1. La fréquence d'échantillonnage utilisée est par défaut de 10 Gs sauf mention du contraire. De plus toutes les attaques ont été effectuées sur la BEC223 définie dans le tableau 4.3. L'intérêt de mener les attaques sur la plus petite courbe elliptique est de limiter la taille des traces de consommations capturées et ainsi limiter la taille des données manipulées. Les résultats obtenus restent valides sur les BEC de plus grandes tailles.

5.1.1 Attaques temporelles

Ce type d'attaque consiste à exploiter les différences de calculs qui dépendent de la clef secrète. Nous devons vérifier que notre implémentation est en temps constant. Pour

Marque	Rohde & Schwarz
Modèle	RTO 2024
Bande passante	2 GHz
Fréquence d'échantillonnage	10 Gs

TABLE 5.1 – Caractéristiques de l'oscilloscope.

permettre une telle implémentation, nous utilisons un algorithme de multiplication scalaire uniforme : l'Échelle de Montgomery, cf. algorithme 2.8.

Cependant, le branchement conditionnel de cet algorithme dépend de la clef et peut induire une légère variation dans l'exécution du calcul de kP en fonction de la clef k. Ainsi, nous devons éviter ce branchement conditionnel et appliquer exactement les mêmes instructions, quelle que soit la clef. Notons que selon le bit courant de la clef, les opérations effectuées sur les points R_0 et R_1 sont inversées. Il s'en suit qu'il suffit d'échanger en temps constant ces deux points en fonction du bit de la clef afin d'éviter un branchement conditionnel. Pour cela, il est usuel d'utiliser une fonction d'échange conditionnel qui va appliquer les mêmes opérations, quelles que soient les entrées de celle-ci. Cet échange conditionnel est effectué par une équation qui va échanger deux registres si le bit de condition est 1, sinon l'équation n'effectue aucun échange, mais les opérations sont tout de même effectuées. Il suffit alors d'appliquer celle-ci sur tous les registres composant les points R_0 et R_1.

L'échange conditionnel est effectué à l'aide des équations suivantes :

$$R_{0,i} = R_{0,i} + b(R_{1,i} - R_{0,i}) \tag{5.1}$$

$$R_{1,i} = R_{1,i} - b(R_{1,i} - R_{0,i}) \tag{5.2}$$

où $R_{0,i}$ est le i-ème registre formant R_0 et b est le bit de condition correspondant au bit courant de la clef. Nous pouvons alors réécrire en temps constant l'Échelle de Montgomery, cf. algorithme 5.1.

Algorithme 5.1 Algorithme de l'Échelle de Montgomery en temps constant.

Entrées $k = (k_{n-1}...k_1k_0)_2$ un scalaire, P un point de E
Sorties: kP
  $R_0 \leftarrow \mathcal{O}$
  $R_1 \leftarrow P$
  $n = \log_2(k)$
  Pour $i = n - 1$ **à** 0 **Faire**
    `cswap`$(R_0,\ R_1,\ k_i)$
    $R_1 \leftarrow R_0 + R_1$
    $R_0 \leftarrow 2R_0$
    `cswap`$(R_0,\ R_1,\ k_i)$
  Fin Pour
  Retourner $Q = kP$

Pour vérifier l'exécution en temps constant de notre implémentation, il nous faudrait tester toutes les clefs possibles et vérifier que chaque temps d'exécution est identique. Il est impossible d'effectuer une telle vérification. Ainsi, nous allons utiliser seulement trois clefs différentes qui doivent en théorie maximiser les variances d'exécutions si elles existent. Nous considérons les deux clefs extrêmes : `0x00...00` et `0xFF..FF`. Si nous avons une variance entre une étape de l'algorithme 5.1 où $k_i = 0$ et $k_i = 1$, alors ces deux clefs

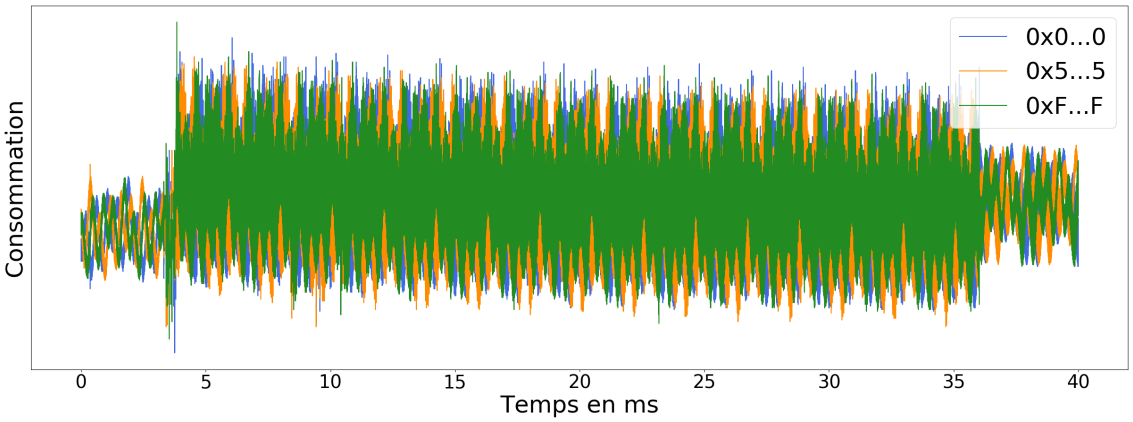


FIGURE 5.1 – Temps d'exécution pour trois calculs de kP sur la BEC223. Les trois clefs utilisées sont 0x00...00, 0x11..11 et 0x55...55.

Clefs	`0x00...00`	`0xFF..FF`	`0x55...55`
Nbe de cycles	3 219 929	3 219 929	3 219 929

TABLE 5.2 – Nombre de coups d'horloges nécessaires pour effectuer kP sur la BEC223.

maximisent la variance du temps d'exécution de kP. De plus, nous considérons la clef `0x55...55` afin de vérifier qu'une variance n'existe pas si $k_{i-1} = 0$ et $k_i = 1$.

La figure 5.1 donne la sortie brute de la consommation de courant pour trois calculs de kP. Aucune différence n'est visible entre ces trois traces de consommation. Cependant, la variation induite par des clefs différentes pourrait être non visible sur ce genre de représentation. Ainsi, une mesure au cycle près de l'exécution du calcul kP est nécessaire. Le tableau 5.2 donne le nombre de coups d'horloges exact nécessaires pour effectuer le calcul de kP. Ces mesures ont été faites sur microcontrôleur dont les caractéristiques sont données par le tableau 4.4. Nous avons utilisé le *timer* interne du microcontrôleur.

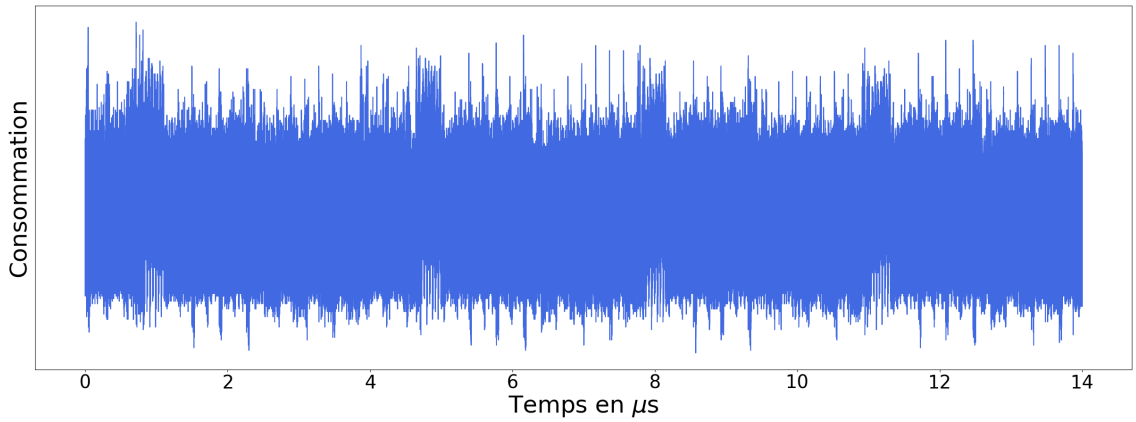
Nous avons donc un temps de calcul exactement identique pour différentes clefs. Nous en déduisons à ce stade que notre implémentation n'est pas sensible aux attaques temporelles.

5.1.2 Attaques simples

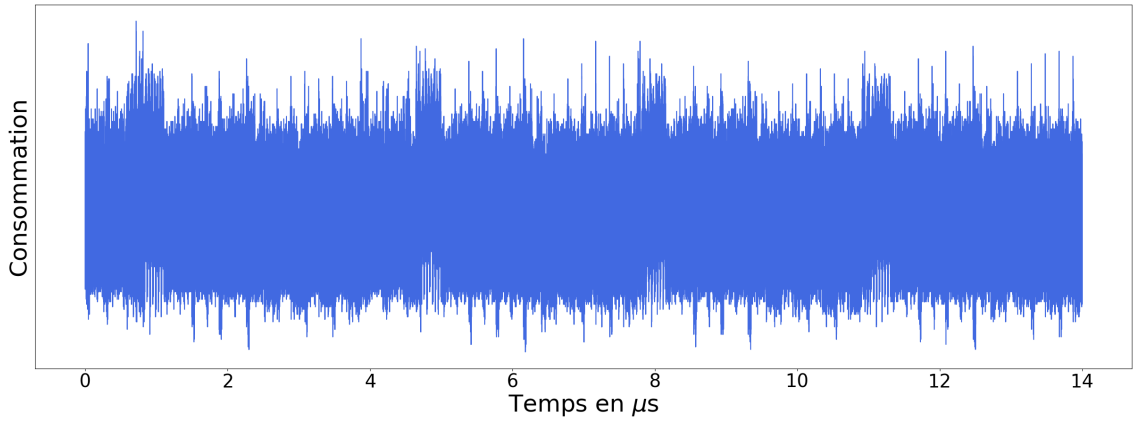
Les attaques simples cherchent à mettre en évidence un biais dans le profil d'une trace d'une mesure physique. Si ce biais dépend de la clef utilisée, alors nous pouvons inférer de l'information sur cette dernière. Cette catégorie d'attaque peut s'appliquer à un cas réel. Malgré tout, les attaques avancées de cette famille ne peuvent que rarement s'appliquer aux protocoles usuels. Ces attaques nécessitent de soumettre un point de base spécifique ce qui est impossible lors d'une signature ECDSA ou EdDSA. De plus, elles nécessitent d'effectuer un calcul kP pour inférer un bit du scalaire k, or dans les protocoles d'échanges de clefs (ECDH) ou de signature (ECDSA) ce scalaire k est éphémère et donc change à chaque exécution du protocole.

Simple Side Channel Attack (SSCA)

Cette attaque consiste à comparer deux étapes d'une trace du calcul de kP. Si l'on trouve une différence dans le profil des traces entre deux étapes du calcul, alors nous pouvons en déduire que chacune de ces étapes correspond au traitement de deux bits de



(a) Bit 0



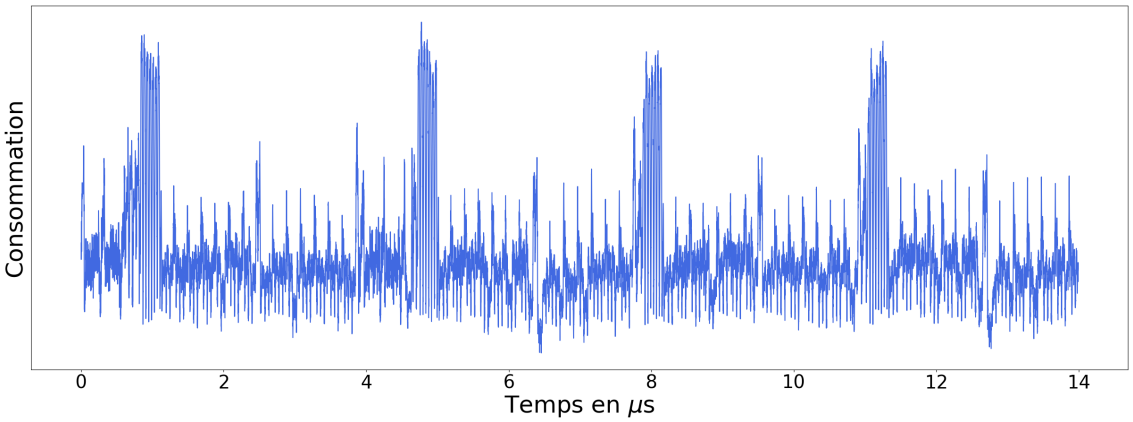
(b) Bit 1

FIGURE 5.2 – Traces de consommation pour une étape de l'Échelle de Montgomery pour les bit 0 et 1.

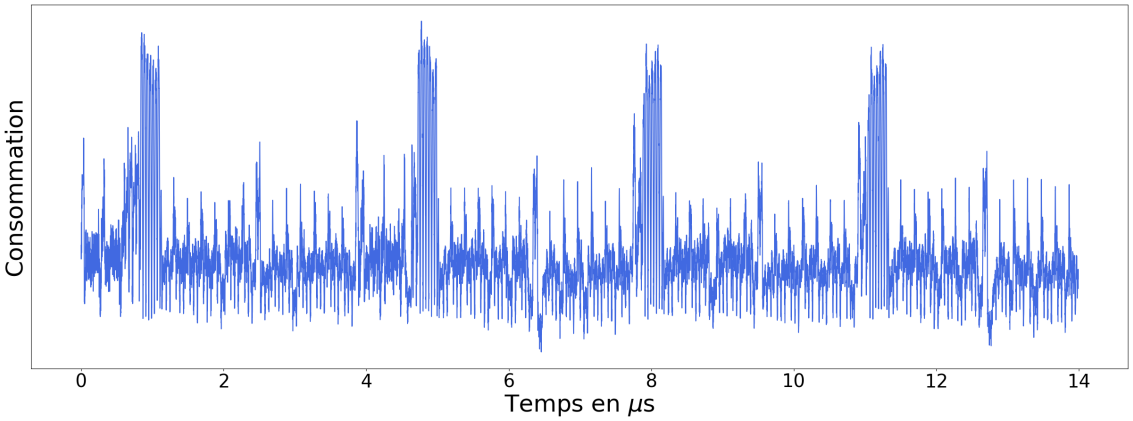
valeur différente de la clef. Pour vérifier la robustesse de notre implémentation face à cette attaque, il nous faut comparer deux étapes du calcul de kP, la première pour le bit 0 et la seconde pour le bit 1.

La figure 5.2 présente deux traces de consommation de courant, chacune étant une étape du calcul de kP soit pour le bit 0, soit pour le bit 1. Les mesures ont été faites à une fréquence d'échantillonnage de 10 Gs qui correspond à la fréquence maximale permise par l'oscilloscope utilisé. Par conséquent comparer les traces brutes ne permet pas de statuer, car aucun schéma est explicitement distinguable. Ainsi, nous devons appliquer un post-traitement afin de faire ressortir les schémas correspondant aux différents calculs d'une étape de l'Échelle de Montgomery. Pour cela, nous appliquons différents filtres qui auront comme objectif de faire ressortir les différents schémas pertinents.

- **Moyenne glissante** : ce post-traitement consiste à effectuer une moyenne glissante pour une fenêtre donnée fixe. L'intérêt de ce traitement est de faire ressortir une information utile qui serait diffuse tout au long du calcul et ainsi éliminer le bruit local.
- **Compression par la somme** : ce post-traitement consiste à compresser une fenêtre donnée en sommant toutes les valeurs. Dans ce cas, nous supposons que l'information utile est locale.
- **Compression par le maximum** : ce post-traitement consiste à compresser une fenêtre donnée en prenant le maximum. Dans ce cas, nous supposons que l'information



(a) Bit 0



(b) Bit 1

FIGURE 5.3 – Traces de consommation pour une étape de l'Échelle de Montgomery pour les bits 0 et 1. Post-traitement : moyenne glissante.

utile se trouve dans les maximums de la trace de consommation.

- **Compression par le minimum** : ce post-traitement consiste à compresser une fenêtre donnée en prenant le minimum. Dans ce cas, nous supposons que l'information utile se trouve dans les minimums de la trace de consommation.

L'application de ces différents filtres permet de faire apparaître des profils au sein d'une étape de l'Échelle de Montgomery pour le bit 0 et 1. Les quatre filtres donnent sensiblement le même résultat. Ainsi, la figure 5.3 donne le résultat obtenu en filtrant les traces de consommation de la figure 5.2 avec une moyenne glissante de taille 1000 points. La taille de cette fenêtre est arbitraire et a été choisie en raison de sa capacité à dévoiler les différents paternes du calcul.

Le traitement effectué nous permet de clairement distinguer visuellement les 4 multiplications sures $\mathbb{F}_{2^n}$ effectuées à chaque étape du calcul de kP. Il s'en suit que nous ne pouvons distinguer de différences significatives dans les paternes entre les bits 0 et 1. En calculant le coefficient de corrélation entre les traces de consommations de courant correspondant au bit 1 et au bit 0, nous pouvons affiner notre analyse. Les résultats sont donnés dans le tableau 5.3.

Nous ne pouvons donc pas opérer une attaque simple sur notre implémentation. Cependant, ce résultat est à nuancer. En effet, avec une analyse plus poussée via un post-traitement avancé et/ou en focalisant notre analyse sur une partie de la trace, nous pourrions, *a priori*, pouvoir distinguer les étapes pour le bit 0 et le bit 1. Par exemple, avec des

Post-traitement	Coefficients de corrélation
Aucun	98%
Moyenne glissante	99%
Somme	99%
Maximum	93%
Minimum	96%

TABLE 5.3 – Coefficients de corrélation entre deux traces d'une étape de l'Échelle de Montgomery pour les bits 1 et 0.

techniques de classification, nous pourrions distinguer un échange conditionnel utilisant le bit 0 ou le bit 1. Ce type d'attaque avancé est catégorisé dans les attaques horizontales et pourrait faire l'objet de travaux futurs.

RSCA, ZSCA et SVSCA

Les attaques Refined Side Channel Attack (RSCA), Zero Side Channel Attack (ZSCA) et Same Value Side Channel Attack (SVSCA) sont une amélioration des attaques simples classiques. Les deux premières peuvent être écartées, car la contremesure excluant les points de petits ordres permet d'exclure ce genre d'attaque. De même, l'attaque simple utilisant les points de petits ordres n'est pas applicable pour ces mêmes raisons.

Dans le cas de la SVSCA nous ne pouvons pas l'appliquer dans un cas réel d'application. En effet, comme nous l'avons vu à la section 3.2.2, cette attaque ne peut pas être appliquée dans le cas de l'utilisation dans les protocoles ECDH, ECDSA ou EdDSA.

5.1.3 Attaques verticales

Les attaques verticales consistent en l'utilisation d'un ensemble de traces ayant les mêmes entrées, sur lesquelles une analyse statistique est réalisée afin d'inférer de l'information sur la clef utilisée. Notons que cette famille d'attaque n'est que rarement applicable dans un contexte d'applications cryptographiques au travers de protocoles tels que ECDH, ECDSA ou EdDSA. Ces protocoles utilisent soit des scalaires éphémères k pour le calcul kP, soit l'utilisation d'un point de base fixe.

Correlation Side Channel Attack (CSCA)

Cette attaque, équivalente à la Differential Side Channel Attack (DSCA), consiste à corréler un ensemble d'hypothèses de clefs selon un modèle de fuite, à un ensemble de traces de consommation. Il s'en suit que l'hypothèse de clef exacte aura un fort taux de corrélation si les paramètres de l'attaque sont suffisants pour faire fuir de l'information. Les détails de l'attaque peuvent être trouvés à la section 3.2.3.

Notons que les hypothèses de cette attaque, notamment celles consistant à choisir le point de base, rendent l'attaque impraticable sur un protocole cryptographique basé sur les courbes elliptiques. Malgré tout, il est intéressant de mener celle-ci afin de valider que nous avons correctement appliqué la contremesure d'ajout d'aléatoire dans la représentation du point de base, cf. section 3.4.1. Cette attaque nous permet de mettre en évidence une fuite dans notre implémentation qui sera plus finement exploitée grâce aux attaques par profilages du chapitre 6.

Ainsi, nous appliquons l'attaque décrite dans la section 3.2.3. Il est à noter que nous construisons nos hypothèses d'attaque sur le huitième bit du calcul. Nous avons alors 256 hypothèses de clefs possibles. De plus, nous concentrons la recherche de fuites sur

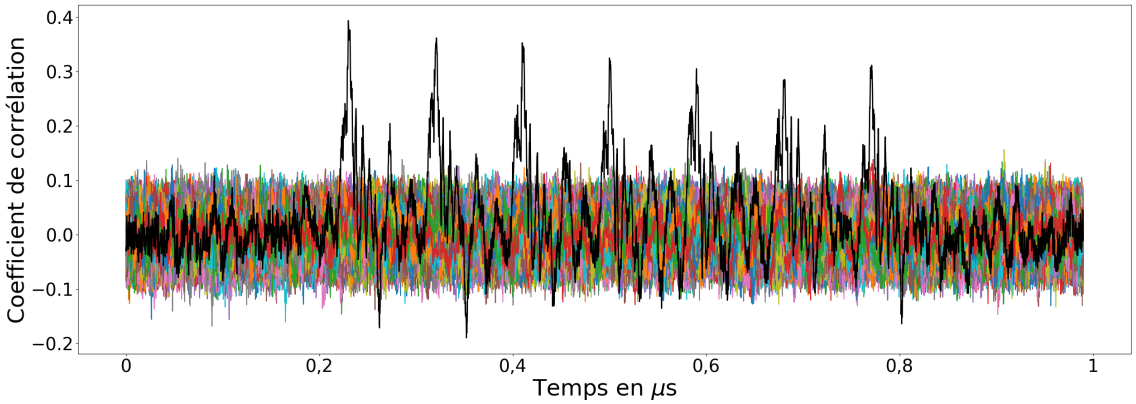


FIGURE 5.4 – Coefficients de corrélation en fonction du temps pour toutes les hypothèses de clefs sur 8 bits. Implémentation sans contremesure. La bonne hypothèse de clef correspond à la courbe noire.

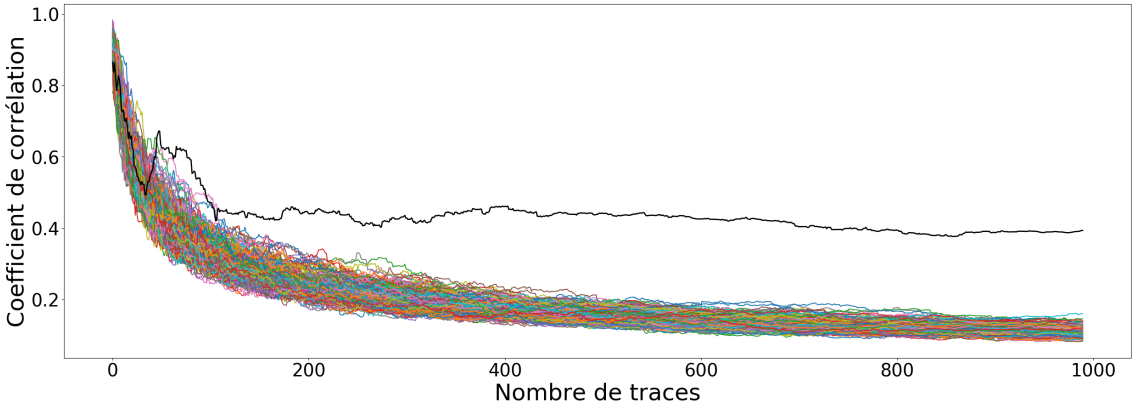


FIGURE 5.5 – Coefficients de corrélation maximum en fonction du nombre de trace pour chaque hypothèse. Implémentation sans contremesure. La bonne hypothèse de clef correspond à la courbe noire.

l'opération d'échange conditionnel de points de l'algorithme 5.1. En effet, les hypothèses de clefs seront calculées à partir de la sortie de cette opération pour la coordonnée projective W du point.

En calculant la corrélation de Pearson (équation 3.3) entre les hypothèses émises et les traces de consommations de l'opération d'échange conditionnel. Les coefficients de corrélations en fonction du temps pour toutes les hypothèses sont donnés par la figure 5.4. Nous avons utilisé 1000 traces différentes et donc 1000 points de bases différents pour cette attaque.

Les coefficients de corrélation de l'hypothèse de clef correcte sont supérieurs à ceux correspondant aux autres hypothèses. De plus, l'écart entre les coefficients maximums de la bonne hypothèse et les autres est important ce qui nous indique que notre attaque est pertinente. Nous sommes alors en droit de nous interroger sur le nombre minimum de traces nécessaires par hypothèse pour retrouver la bonne clef. La figure 5.5 donne le maximum des coefficients de corrélation en fonction du nombre de traces utilisé pour toutes les hypothèses de clefs. Ainsi, il suffit de 200 traces par hypothèse pour retrouver la bonne clef.

Nous pouvons alors répéter de façon récursive cette attaque sur tous les octets de la clef en attaquant en premier les octets de poids fort.

Nous pouvons appliquer cette même attaque sur une implémentation utilisant la contre-

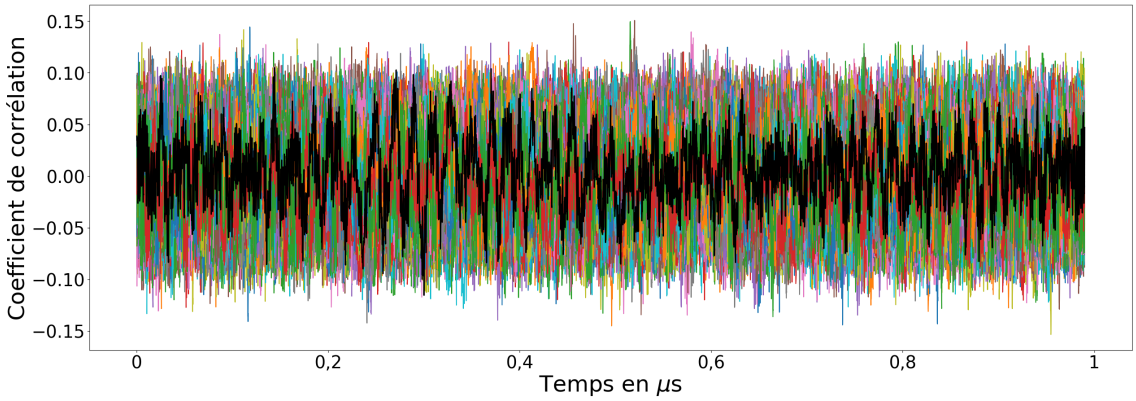


FIGURE 5.6 – Coefficients de corrélation en fonction du temps pour toutes les hypothèses de clefs sur 8 bits. Implémentation avec contremesure. La bonne hypothèse de clef correspond à la courbe noire.

mesure d'ajout d'aléatoire dans la représentation des coordonnées du point, cf. section 3.4.1. Le résultat de cette attaque est donné par la figure 5.6. Ainsi, cette contremesure permet de se protéger contre ce genre d'attaque.

Attaque par doublement

Cette attaque détaillée dans la section 3.2.3 cherche à identifier les bits 0 de la clef en sollicitant des collisions entre les calculs de kP et $k(2P)$. Comme nous l'avons expliqué, cette attaque n'est pas applicable si la multiplication scalaire est calculée grâce à l'Échelle de Montgomery. En effet, nous n'observons aucune collision de ce type lors de ce calcul. Cependant, il existe une variante qui va retrouver les transitions entre les bits 0 et 1 au sein de la clef et nous pouvons alors chercher des collisions au sein du calcul de l'Échelle de Montgomery. Ces dernières apparaissent lorsque, aux étapes i de kP et $i-1$ de $k(2P)$, nous avons deux bits de clefs différents. Ainsi, cette variante permet d'identifier les transitions de bits au sein de la clef et donc réduire l'espace de clefs possibles à seulement deux clefs.

En théorie, cette attaque est faisable sur un algorithme tel que l'Échelle de Montgomery et il est alors nécessaire d'ajouter une contremesure afin de se prémunir de ce genre d'attaque. Cependant, nous utilisons les coordonnées différentielles mixtes avec la coordonnée Z commune. Ainsi, à l'étape i du calcul de kP, nous n'aurons pas nécessairement la même représentation projective avec le calcul de $k(2P)$, des points intermédiaires R_1 et R_0. Or, nous cherchons précisément une collision entre ces points. Finalement, nous pouvons conclure que cette attaque n'est pas applicable à notre implémentation.

En considérant l'exemple de la courbe BEC223 dont les paramètres sont donnés par le tableau 4.3 et en utilisant comme point de base celui donné par ce même tableau, nous pouvons trouver des collisions au sein du calcul de kP et $k(2P)$ pour $k = 0x5555$ en coordonnées affines. Tandis qu'en coordonnées projectives, nous n'en avons aucune. À l'étape 16 du calcul kP, nous avons comme entrées pour les points R_1 et R_0 :

$$\begin{aligned} W(R_0) &= 0x1077C60C6019C83AA98BF78F695DF4CA91A349CACA9031A634B6534B \\ W(R_1) &= 0x5F3775C1EBC31825626B898600988848809A2523F0591E231ED5BD1B \\ Z &= 0x0B9C6616A079EA99B86214AFDC62FF5FEF3A6361571287948763 24D \end{aligned}$$

En coordonnées affines, nous avons :

$$\begin{aligned} w(R_0) &= 0x0C48E46EF29552D6442CA698D189DF26A1A5FCA9D36C9AEE48872927 \\ w(R_1) &= 0x4CEDD47ED91D5FB7E0B86B0FAE750ADBFC5013C7857B5F501B7EF629 \end{aligned}$$

Dans le cas du calcul de $k(2P)$, nous obtenons à l'étape 15 les coordonnées suivantes :

$$\begin{aligned} W(R_0) &= 0x2176F00C2C46801C3D9DC463106AC19DAF4F0A15097CAF929FF792A4 \\ W(R_1) &= 0x55E82CBCF59D511B3CB03A6A621669C911680106DEBB8A8A0208A0B8 \\ Z &= 0x7A42EDD73C8459D8B22D970A663B2164CC9BFF0E30E0BB6FB66E6518 \\ w(R_0) &= 0x6B81D5E640066CDA863DF0B81C88F8D791AA69683ED1938613EB54D8 \\ w(R_1) &= 0x4CEDD47ED91D5FB7E0B86B0FAE750ADBFC5013C7857B5F501B7EF629 \end{aligned}$$

Ainsi, nous avons effectivement une collision entre les coordonnées affines, mais celle-ci ne se retrouve pas lors du calcul sur les coordonnées projectives, car nous avons une représentation des points différentes. Nous pouvons donc conclure que cette attaque est inapplicable aux implémentations sur les courbes elliptiques utilisant une représentation projective des points, mais possible si nous utilisons les coordonnées affines.

5.1.4 Attaque par profilage

Les attaques par profilages décrites à la section 3.2.5 cherchent à construire un profil de fuite pour chaque clef possible. Dans le cas d'une implémentation de la cryptographie sur les courbes elliptiques, la clef secrète est utilisée bit à bit comme le montre l'algorithme 5.1. Ainsi, nous pouvons appliquer une attaque par profilage indépendamment à chaque bit de la clef secrète k. Il s'en suit que nous pouvons alors considérer seulement deux clefs possibles, à savoir le bit 0 et 1, et appliquer l'attaque avec ces mêmes profils à tous les bits formant k.

Finalement, cette attaque consiste à classifier chaque étape de la boucle « Pour ... Faire » de l'algorithme 5.1 afin d'inférer si $k_j = 0$ ou 1. Nous pouvons raffiner cette attaque en observant que le bit de la clef secrète k_j est utilisé seulement par l'opération de `cswap` au début et à la fin de chaque étape de la boucle de l'algorithme 5.1. Nous pouvons isoler cette opération précise afin de construire nos profils d'attaque car l'information sensible recherchée, k_j, est uniquement utilisée à cette étape. Cette opération consiste en l'application des équations 5.1 et 5.2 comme expliquée à la section 5.1.1.

Notons que cette attaque n'est pas spécifique au choix du modèle de courbes elliptiques et peut être appliquée à d'autres implémentations et modèles de courbes elliptiques. Nous avons voulu démontrer ce non-attachement de cette attaque à un modèle particulier de courbes elliptiques. Pour cela, le chapitre 6 détaille entièrement la mise en œuvre de cette attaque dans un cas moins spécifique que la cryptographie sur les BEC que nous avons décrite jusqu'à présent. Nous exposerons cette attaque au chapitre suivant dans le cas de l'implémentation du standard RFC 7748 [LHT16] qui décrit l'implémentation de la courbe Curve22519 qui est une courbe elliptique utilisée sous la forme de Montgomery [Mon87].

Dans le cas de notre implémentation sur les BEC cette attaque a été menée en isolant l'opération de `cswap` du reste de la multiplication scalaire. Nous avons donc classifié des opérations de `cswap` en fonction du bit de condition. En faisant varier le nombre de traces à la construction des profils pour les bits 0 et 1, nous avons atteint un taux de succès de classification de 98%. Ainsi, nous pouvons, *a priori*, inférer 98% d'une clef sur notre implémentation des BEC. Les différentes méthodes considérées de construction des profils sont décrites à la section 6.2.2. Cette préétude des attaques par profilages nous a permis de montrer leur faisabilité. Dans le chapitre 6, nous détaillerons les différentes étapes de cette attaque et comment elles peuvent être mises en place dans une implémentation standardisée de la cryptographie sur les courbes elliptiques.

5.2 Attaque hybride par fautes différentielles

Nous avons vu que l'attaque par doublement et sa variante pour l'Échelle de Montgomery ne sont en réalité pas applicables aux implémentations sur les courbes elliptiques dont les calculs sont effectués en coordonnées projectives. Cependant, l'idée de trouver des collisions entre deux exécutions peut être utilisée. Nous proposons dans cette section une nouvelle attaque exploitant ces collisions en provoquant une variance au sein du calcul kP en y injectant une faute.

Nous supposons que l'attaquant peut distinguer, en analysant une mesure physique, si deux étapes de l'Échelle de Montgomery sont identiques, *i.e.* effectuent les mêmes opérations et utilisent donc les mêmes entrées. De plus, l'attaquant peut effectuer plusieurs fois le même calcul kP pour la même clef k et ainsi obtenir différentes mesures physiques pour le calcul kP. Cependant, l'attaquant ne peut pas modifier les entrées de la multiplication scalaire et ainsi choisir un point de base particulier comme pour l'attaque par doublement. À cela, nous ajoutons l'impossibilité pour l'attaquant de récupérer le résultat final du calcul.

La stratégie de l'attaque est d'injecter une faute lors d'une étape du calcul de kP et d'observer par canaux auxiliaires comment cette faute se propage au sein du calcul. En effet, si une faute est injectée sur l'addition $R_1 + R_0$ de l'étape i et si k_i et k_{i+1} sont identiques, alors l'erreur injectée touche seulement l'addition de l'étape $i + 1$ et le doublement est correct. L'attaquant acquiert alors une trace témoin du calcul kP sans injecter de faute. Par la suite, il cherche des collisions entre cette trace et des traces du calcul kP où une faute aura été injectée. En effet, en capturant une trace où une faute a été injectée à l'étape i, l'attaquant peut chercher les collisions entre la trace témoin et la trace fausse sur les doublements des étapes $i + 1, ..., i + n$. Tant qu'une collision est observable entre les doublements, alors $k_{i+1}, ..., k_{i+n}$ sont identiques à k_i. Il s'en suit que l'attaquant peut identifier de façon récursive les séries de bits identiques au sein de la clef k et donc réduire l'espace de clef à seulement deux clefs possibles, comme l'attaque relative par doublement.

Nous nommerons cette attaque Attaque hybride par fautes différentielles ou HDFA (*Hybrid Differential Fault Attack*).

Pour illustrer notre attaque, prenons l'exemple du calcul de $199P$. Nous avons $199 = 0b11000111$, donc le calcul de l'Échelle de Montgomery aura 8 sous-étapes. L'exécution normale de ce calcul est donnée par la figure 5.7a. En injectant une faute à la première étape de ce calcul et en observant les collisions entre cette trace et la trace témoin nous pouvons identifier que les bits 1 et 2 sont les mêmes, ceci est illustré par la figure 5.7b. De même, en injectant une faute à l'étape 3 et 6, nous pouvons respectivement identifier les bits 3, 4, 5 et 6, 7, 8. Les figures 5.7c et 5.7d illustrent ces deux étapes de l'attaque.

Finalement, nous pouvons retrouver la clef en injectant seulement 3 fautes et en obtenant 4 traces du calcul $199P$. De manière générale, il faut injecter n fautes pour retrouver une clef k, où n est le nombre de séries de bits identiques au sein de k. Il faudra alors $n+1$ traces du calcul de kP.

Il est à noter que les fautes injectées ne nécessitent pas d'être connues. En effet, nous n'utilisons pas le résultat du calcul comme lors d'une DFA classique, cf. la section 3.3.3.

L'utilisation de protocoles comme l'ECDH ou l'ECDSA empêche la mise en œuvre d'une telle attaque. Dans ce cas précis, les scalaires utilisés pour le calcul de kP sont éphémères et varient à chaque exécution du calcul. Il s'en suit qu'un protocole tel que l'EdDSA est sensible à ce genre d'attaque, car les calculs effectués sont déterministes. Malgré tout, nous pouvons nous prémunir de ce type d'attaque par l'emploi de la contremesure qui ajoute de l'aléatoire dans la représentation des coordonnées projectives. Dans ce cas, il est impossible de trouver une collision entre la trace témoin et les autres traces.

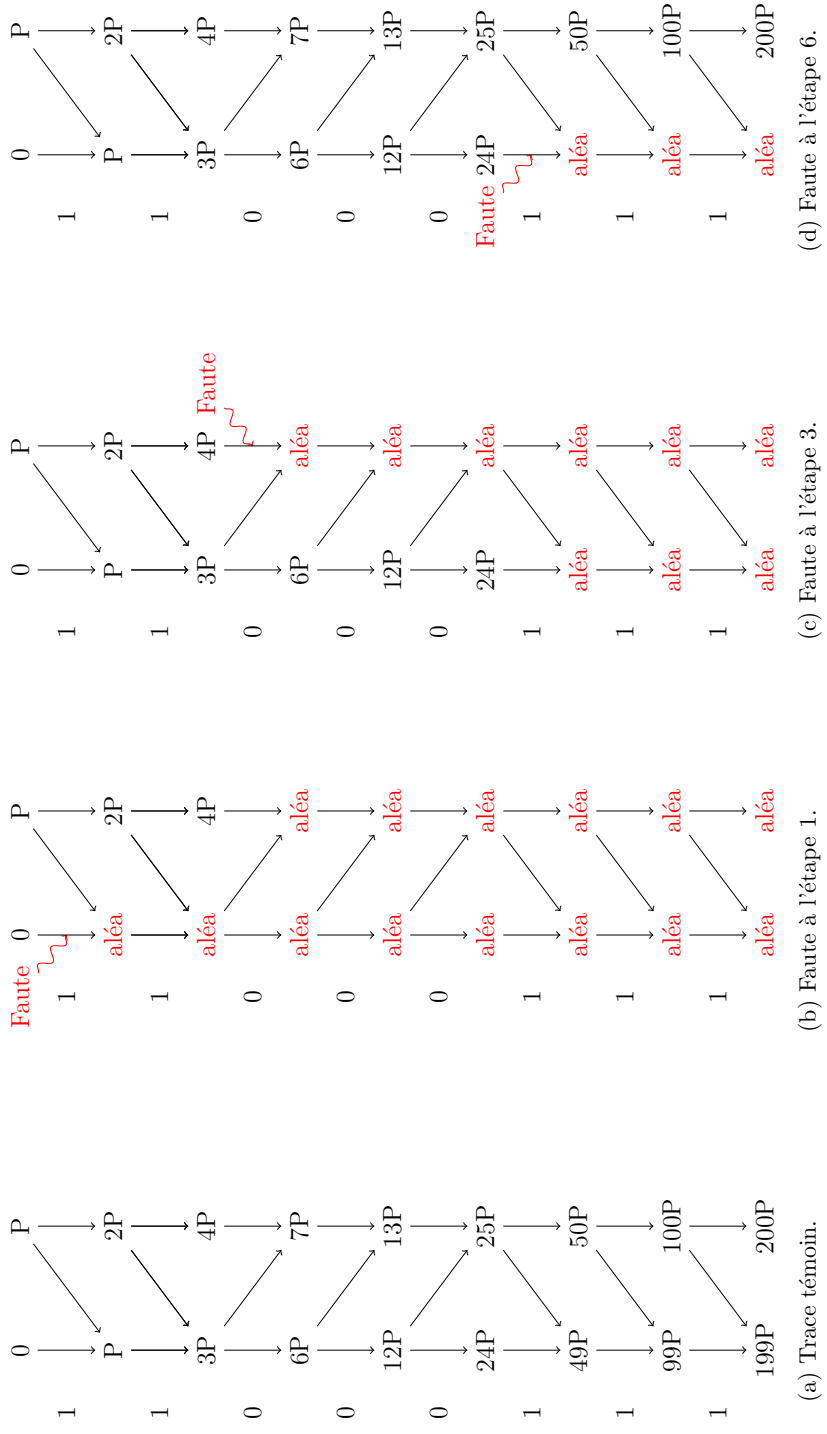


FIGURE 5.7 – Exemple de l'application d'une HDFA sur le calcul de $199P$.

Ce nouveau schéma d'attaque n'a pas été mené en pratique et pourra faire l'objet de travaux futurs. La nécessité d'obtenir différentes traces pour un même calcul rend cette attaque difficilement applicable aux standards ECDH et ECDSA car les scalaires utilisés sont éphémères. Tandis que le calcul de multiplication scalaire effectué par l'EdDSA permet, *a priori*, d'appliquer ce schéma d'attaque à ce protocole car le scalaire utilisé est déterministe.

5.3 Conclusion

Dans ce chapitre, nous avons vérifié la majeure partie des propriétés de sécurité de notre implémentation face aux attaques par canaux auxiliaires. Pour ce faire nous avons mené les attaques suivantes sur un microcontrôleur implémentant le jeu d'instructions RISC V :

- Attaques temporelles.
- *Simple Side Channel Attack* (SSCA)
- *Refined Side Channel Attack* (RSCA), *Zero Side Channel Attack* (ZSCA) et *Same Value Side Channel Attack* (SVSCA)
- *Correlation Side Channel Attack* (CSCA)
- Attaque par doublement et attaque relative par doublement

Il est difficile de prouver formellement la sécurité d'une implémentation face à une attaque. Nous pouvons seulement avoir un degré de confiance sur la sécurité de notre implémentation. De notre étude, nous avons confirmé les protections intrinsèques de notre implémentation. De plus, nous avons mis en avant l'impossibilité d'effectuer une attaque relative par doublement (section 5.1.3) sur les implémentations ECC utilisant les coordonnées projectives. Cette protection est due à la non-unicité de la représentation des points en coordonnées projectives.

Nous avons aussi proposé un nouveau schéma d'attaque visant l'Échelle de Montgomery. Celui-ci utilise l'injection de fautes pour perturber le calcul de la multiplication scalaire et une analyse par canaux auxiliaires pour observer la propagation de cette faute au sein du calcul. Cette attaque nécessite plusieurs itérations du calcul de kP pour être effectuée ce qui la rend impraticable dans le cas de protocoles tels que l'ECDH ou ECDSA. Cependant, l'EdDSA reste sensible à ce type d'attaque. Malgré tout, la contremesure qui ajoute de l'aléa dans la représentation des coordonnées protège, *a priori*, de ce biais d'attaque.

Chapitre 6

Sécurisation des implémentations ECCs face aux attaques par profilages

Il y a des choses que l'intelligence seule est capable de chercher, mais que par elle-même elle ne trouvera jamais. Ces choses, l'instinct seul les trouverait, mais il ne les cherchera jamais.

Henri Bergson,

Sommaire

Le cas des attaques par profilages n'a pas été traité dans le chapitre 5. Les résultats que nous avons obtenus pour ces attaques et qui sont présentés dans ce chapitre demandent une étude toute particulière. Ceci est en partie dû aux avancées récentes en termes d'attaques par profilage utilisant des techniques de construction avancées des profils amènent une nouvelle manière d'appréhender ce genre d'attaque, [APSQ06, SA08, BHvW12, CDP15].

Dans ce chapitre nous détaillerons la méthode d'application des attaques par profilages sur une implémentation cryptographique basée sur les courbes elliptiques. Nous verrons l'influence des méthodes de réductions de dimensions de type *Principal Component Analysis* (PCA) [Jol11] et *Linear Discriminant Analyis* (LDA) [Fis38] sur la cryptographie à base de courbes elliptiques. De plus, nous verrons que ce genre de méthode permet d'outrepasser des contremesures classiques comme l'ajout d'aléatoire dans la représentation de coordonnées ou toutes les contremesures cachant la clef comme celles présentées dans les sections 3.4.4 ou 3.4.5. Nous proposons alors une nouvelle contremesure afin de pallier aux fuites pouvant être exploitées lors du calcul de kP.

Il est à noter que l'attaque menée est faite sur l'implémentation de la Curve25519 [LHT16] de mbedTLS [mbe19]. Les choix faits par cette implémentation sont proches de ceux que nous avons faits. Elle utilise l'algorithme de l'Échelle de Montgomery en temps constant, cf. algorithme 5.1. De plus, nous montrons que cette attaque n'est pas liée à un modèle particulier et peut être applicable à un grand nombre d'implémentations sur les courbes elliptiques.

Finalement, nous mènerons notre attaque sur l'implémentation de mbedTLS [mbe19] du calcul de kP pour la courbe Curve25519 [LHT16] sous forme de Montgomery. La version de la bibliothèque considérée est la version 2.14.1. Cette bibliothèque est compilée avec gcc [gcc19].

Caractéristiques	Microcontrôleur
Processeur	Cortex M4 168 MHz
RAM	192 Ko
Flash	512 Ko

TABLE 6.1 – Caractéristiques du microcontrôleur utilisé.

6.1 Faiblesse de la fonction d'échange conditionnel de vecteurs

La section 5.1.1 présente la manière dont est implémenté l'algorithme 2.8 de l'Échelle de Montgomery en temps constant. En effet, l'algorithme 5.1 utilise les équations 5.1 et 5.2 afin d'échanger en temps constant les registres représentant les points de la courbe elliptique. Cet échange, ou `cswap`, dépend directement du bit courant de la clef.

En analysant ces équations, nous notons que si le bit b est nul, alors le résultat de la multiplication des équations 5.1 et 5.2 est aussi nul et a donc un poids de Hamming nul. Tandis que si $b = 1$ alors le poids de Hamming du résultant est différent de 0. Ce constat dans le poids de Hamming résultat peut être exploité grâce à une attaque par profilage.

Cependant, cette méthode d'échange conditionnel n'est pas unique. Par exemple, OpenSSL [ope19] emploie des équations différentes afin d'opérer ce changement. La première étape est de changer le bit conditionnel b en 0 si $b = 0$ ou en `0xF...F` si $b = 1$. La seconde étape

est d'appliquer les formules suivantes :

$$R_{0,i} = R_{0,i} \oplus b \wedge (R_{1,i} \oplus R_{0,i}) \tag{6.1}$$
$$R_{1,i} = R_{1,i} \oplus b \wedge (R_{1,i} \oplus R_{0,i}) \tag{6.2}$$

Dans le cas de mbedTLS [mbe19], d'autres équations sont utilisées :

$$R_{0,i} = bR_{1,i} + (1-b)R_{0,i} \tag{6.3}$$
$$R_{1,i} = bR_{0,i} + (1-b)R_{1,i} \tag{6.4}$$

Les équations 6.3 et 6.4 ont l'avantage d'effectuer une multiplication par 0 et 1 quel que soit le bit de condition b. Malgré tout, nous avons toujours une différence en fonction de b dans l'ordre d'exécution de ces multiplications. Cette différence peut donc être également exploitée avec une attaque par profilage.

De plus, la contremesure ajoutant de l'aléatoire dans la représentation des coordonnées du point de base, cf. section 3.4.1, a pour effet d'ajouter de l'aléatoire dans un seul opérande de ces multiplications. Ainsi, *a priori*, cette contremesure ne permettrait pas de se prémunir de ce genre de fuites.

6.2 Attaque par profilage sur le cswap

La stratégie de cette attaque consiste à effectuer l'attaque récursivement sur tous les bits de la clef. Ainsi, nous avons seulement deux clefs possibles : le bit 0 et le bit 1. De plus, nous nous plaçons dans un cas applicatif de la cryptographie sur les courbes elliptiques, *i.e.* nous considérons les contraintes induites par les protocoles ECDH, ECDSA et EdDSA. Nous distinguons alors plusieurs scénarios :

- **S1** : La clef ciblée est celle utilisée lors d'une session du protocole d'échange de clef ECDH, cf. figure 2.6b. La clef k est alors utilisée deux fois. Ainsi, l'attaquant pourra obtenir deux traces du calcul kP. Notons qu'entre chacun de ces calculs le point de base P ne sera pas le même. Cependant, ceci n'a pas d'impact sur l'attaque, car nous considérons le point de base comme aléatoire grâce à l'utilisation de la contremesure d'ajout d'aléatoire dans la représentation des points.
- **S2** : Dans le cas d'une signature ECDSA, la clef privée du signataire n'est pas utilisée lors du calcul kP. Le nombre aléatoire utilisé pour ce calcul est néanmoins sensible, car si ce dernier est connu nous pouvons retrouver la clef privée du signataire, cf. Algorithme 2.16. Entre chaque calcul d'une signature, le scalaire k est donc différent. Il s'en suit qu'un attaquant aura une seule et unique trace du calcul de kP pour inférer k.
- **S3** : La signature construite par le protocole EdDSA est complètement déterministe. Un attaquant aura accès à autant de calculs de kP que souhaité. Ce déterminisme dans la signature peut permettre d'augmenter significativement les taux de succès de notre attaque.

Le modèle d'attaquant que nous considérons est le suivant :

- L'attaquant peut obtenir,selon les scénarios S1, S2 ou S3, un certain nombre de traces d'une mesure physique du calcul kP ciblé.
- L'attaquant possède une copie de l'implémentation visée (logicielle et matérielle) sur laquelle il peut accomplir l'apprentissage et la construction des profils pour chaque clef possible. De plus, durant cet apprentissage :
  - L'attaquant peut soumettre une clef k choisie.

— L'attaquant ne peut pas changer le point de base (S2, S3).

— L'attaquant utilise la contremesure d'ajout d'aléatoire dans la représentation des coordonnées.

— L'attaquant n'utilise pas de contremesures changeant la clef aléatoirement. Ceci empêcherait la labélisation des traces nécessaire à l'attaque.

— L'attaquant peut effectuer autant de traces d'apprentissage que nécessaire.

Toutes ces conditions sont atteignables dans un cas réel ce qui rend notre attaque réalisable sur application cryptographique.

6.2.1 Synchronisation des cswap

La première étape pour mener une telle attaque est de sélectionner seulement les parties d'une trace du calcul kP correspondant à l'application du `cswap`.

En étudiant le code de la bibliothèque mbedTLS, nous observons que la fonction `ecp_mul_mxz` calcule kP pour les courbes elliptiques sous forme de Montgomery comme la Curve25519. Cette fonction fait appel à la fonction `mbedtls_mpi_safe_ cond_swap` qui applique le `cswap` selon les équations 6.3 et 6.4. Cette dernière est appliquée à chaque coordonnée représentant les points R_0 et R_1. En l'occurrence, les points sont représentés par les coordonnées X et Z, ainsi l'application du `cswap` est effectuée deux fois et cet échange également :

- une première fois au début de l'étape de l'Échelle de Montgomery pour le bit k_i
- une seconde fois à la fin de cette même étape

Il s'en suit que pour chaque bit k_i, nous avons quatre appels à la fonction `mbedtls_mpi_safe_cond_swap`.

Cette opération est formée d'une première application des formules de `cswap` pour échanger conditionnellement le signe des coordonnées considérées. Ensuite, ces mêmes équations sont appliquées à tous les registres composant la donnée entière de la coordonnée.

Dans le cas de la Curve25519, les formules 6.3 et 6.4 sont appliquée 9 fois en tout. Cette opération a donc un profil atypique au sein de la trace kP. Nous pouvons alors repérer visuellement une première instance de ce `cswap`. La figure 6.1 donne le profil de quatre `cswap` successifs. Notons que les deux premiers `cswap` de cette figure correspondent à l'étape $i-1$, tandis que les deux derniers font partie de l'étape i.

De ces traces, nous pouvons effectuer une corrélation croisée sur toute la longueur de la trace afin de retrouver toutes les occurrences de `cswap`. Cette corrélation croisée est directement effectuée avec la trace extraite de la figure 6.1. La figure 6.2 présente quant à elle le résultat de cette corrélation croisée. Notons que les 254 intervalles ainsi trouvés sont en accord avec les 254 bits formant la clef lors de la multiplication scalaire sur la Curve25519. De plus, les `cswap` à chaque extrémité de la trace (cf. Figure 6.2) ont des corrélations croisées plus faibles car nous avons seulement deux appels à l'opération `mbedtls_mpi_safe_cond_swap`.

La figure 6.2 nous permet de conclure que chaque étape du calcul kP n'est pas en temps constant, car l'écart entre chaque série de 4 `cswap` n'est pas constant. Cela est dû, *a priori*, à la mémoire cache utilisée par le microcontrôleur.

Une fois toutes les opérations `cswap` identifiées, nous pouvons les recomposer en une trace de 4 `cswap` correspondants à la même étape de l'Échelle de Montgomery. Chaque pic de la figure 6.2 correspond à 4 `cswap`, les deux premiers pour le bit k_{i-1} et les deux derniers pour le bit k_i. Nous pouvons alors reconstituer des traces correspondant à 4 `cswap` pour un même bit k_i du scalaire. À chaque trace nous associons un label désignant le bit conditionnel k_i utilisé. Cet ensemble de traces de 4 `cswap` forme les données d'apprentissage qui serviront à construire les profils de fuites utiliser pour mener l'attaque par profilage.

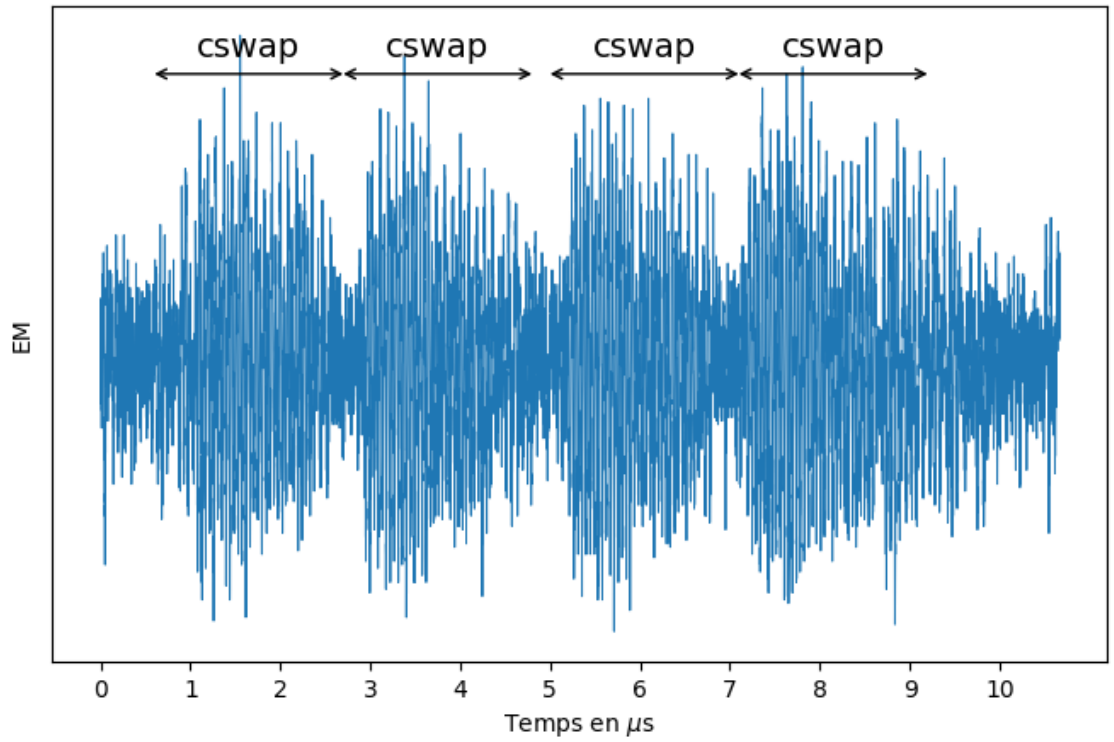


FIGURE 6.1 – Trace électromagnétique de 4 cswap consécutifs.

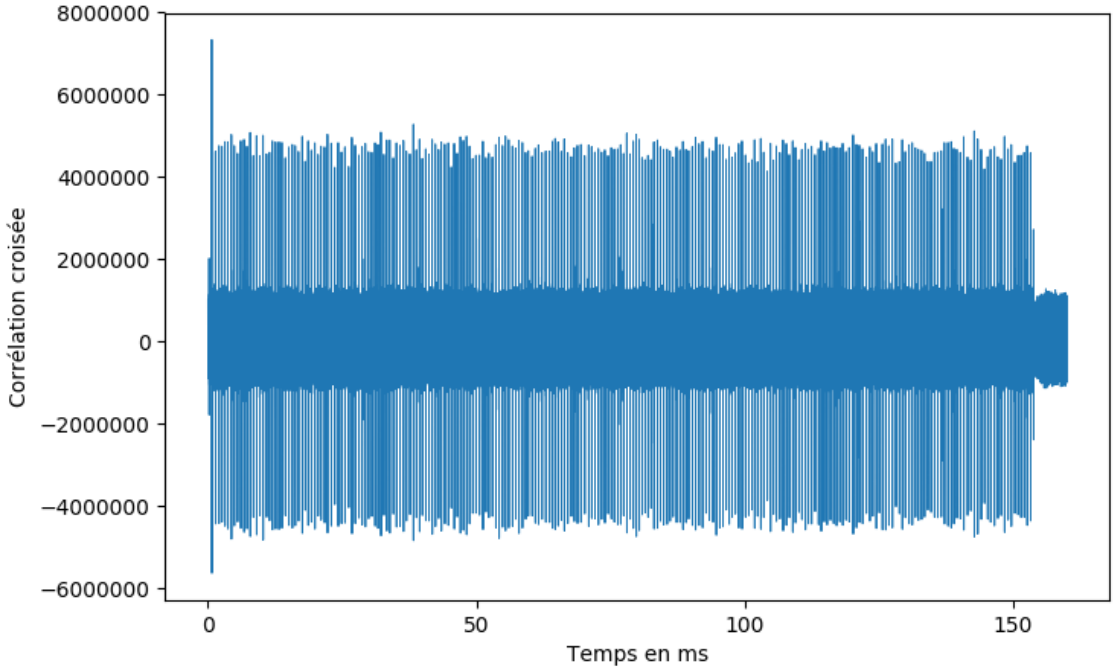


FIGURE 6.2 – Corrélation croisée entre la trace complète du calcul kP et la trace extraite de la figure 6.1.

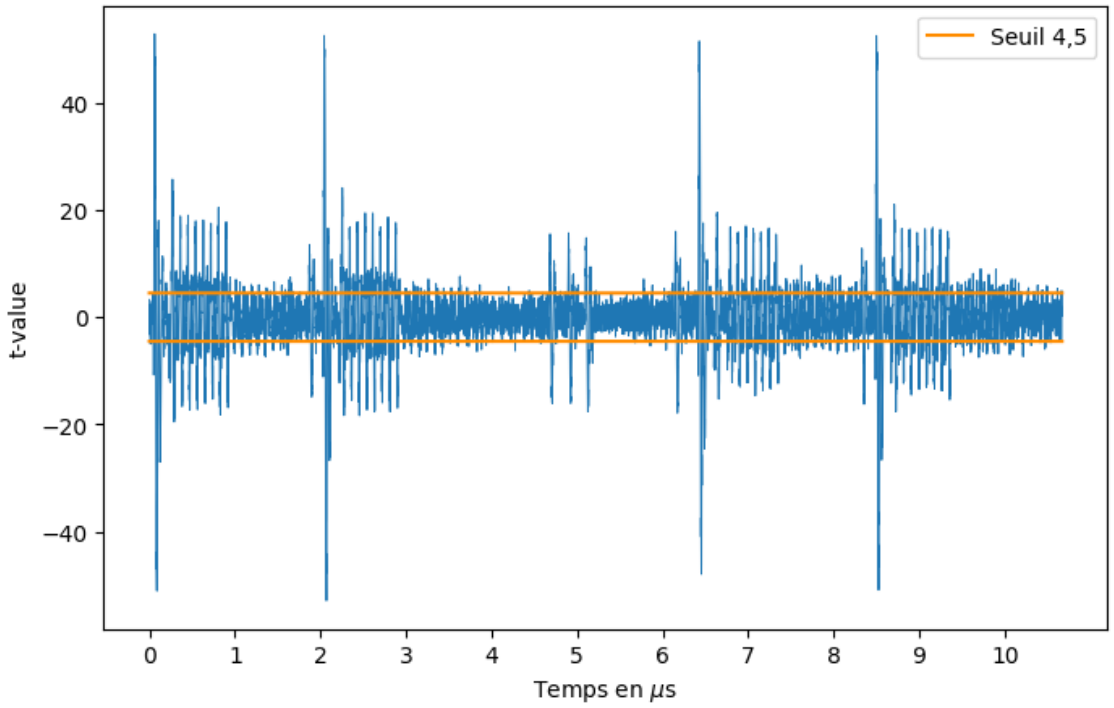


FIGURE 6.3 – Résultat du t-test entre les cswap (figure 6.1) pour le bit 0 et le bit 1. 50000 traces ont été utilisées dans chaque classe.

6.2.2 Construction des profils

Nous avons vu à la section 3.2.5, que la première étape de création des profils est un problème de réduction de dimensions, car l'information pertinente est très localisée dans une trace. Pour sélectionner les points d'intérêts les plus opportuns nous pouvons utiliser le t-test comme nous l'avons expliqué à la section 3.2.5.

Ainsi, la première méthode de réduction de dimensions que nous appliquerons dans notre attaque est celle de sélection des points d'intérêts dans les maximums locaux d'un t-test. Le résultat du t-test sur l'ensemble des traces de profilage des `cswap` est donné par la figure 6.3. On retrouve clairement les différentes étapes composants le calcul de `cswap`. De plus, la valeur de seuil de 4,5 [tte18, SM15] est dépassée ce qui témoigne d'une fuite lors de ces opérations.

Pour la construction des profils, nous utilisons 50 points d'intérêts. Cette caractéristique est empirique, nous avons eu les meilleurs résultats pour ce nombre de points d'intérêts.

Cependant, des méthodes avancées de réduction de dimensions peuvent être utilisées pour notre attaque. Nous considérons ici la PCA [Jol11] et la LDA [Fis38]. Ces méthodes sont notamment utilisées pour les attaques par profilages sur la cryptographie symétrique [APSQ06, SA08, BHvW12, CDP15].

Principal Components Analysis (PCA)

Cette méthode de réduction a été introduite par Joliffe [Jol11]. Sa stratégie consiste à projeter l'ensemble des données sur un sous-espace composé d'un nombre fini de composantes. L'intérêt de cette méthode est de maximiser la variance interclasses dans les premières composantes.

Pour ce faire, nous considérons la matrice de covariance empirique :

$$\bar{S} = \sum_{k=1}^{|\mathcal{K}|} N_t(\bar{\mu}_k - \hat{\mu})(\bar{\mu}_k - \hat{\mu})^T \tag{6.5}$$

Où, $\mathcal{K}$ est l'espace des clefs possibles, dans notre cas nous avons $\mathcal{K} = \{0, 1\}$. N_t représente le nombre de traces pour chaque classe de clef. De plus, $\hat{\mu}$ est la moyenne des traces moyennées $(\bar{\mu}_k)_{k=1,...,N}$.

Soit r le rang de la matrice $\bar{S}$ et $\lambda_1, ..., \lambda_r$ les valeurs propres. Nous avons les vecteurs propres associés : $\alpha_1, ..., \alpha_r$. Nous pouvons alors lister de façon décroissante les valeurs propres λ_i. Ainsi, chaque valeur propre λ_i correspond à la variance des données projetées sur la composante α_i.

Il s'en suit que nous pouvons projeter l'ensemble de nos données de profilage sur les n premières composantes données par la PCA. Ce nombre n est choisi empiriquement en fonction des résultats que nous obtenons. Par exemple, pour $n = 1$, nous utilisons la première composante qui correspond à la variance engendrée par le bruit.

Pour ce faire, nous considérons la matrice de projection W composée des n premiers α_i. Dans ce cas, le profil est composé différemment de celui donné dans la section 3.2.5 :

$$\bar{x}_k = W^T \bar{\mu}_k, \qquad S_k = W^T S_k W \tag{6.6}$$

Où, S_k est la matrice de covariance des traces pour la clef k. Dans notre cas, les clefs possibles sont 0 ou 1.

Linear Discriminant Analysis (LDA)

La LDA introduite par Fisher [Fis38] applique la même stratégie que la PCA. Cependant, dans ce cas-ci, nous maximisons la distance interclasses au lieu de la variance. Ce problème est lié à la maximisation du quotient de Rayleigh :

$$\alpha_1 = \text{argmax}_\alpha \frac{\alpha^T \bar{S} \alpha}{\alpha^T \hat{S} \alpha} \tag{6.7}$$

Où, la matrice $\bar{S}$ est la même matrice de covariance définie par l'équation 6.5 et $\hat{S}$ est la matrice définie par :

$$\hat{S} = \sum_{k \in \mathcal{K}} \sum_{i=1}^{N_t} (\mu_{k,i} - \bar{\mu}_z)(\mu_{k,i} - \bar{\mu}_z)^T \tag{6.8}$$

La LDA est alors réduite à un problème de réduction de la matrice $\hat{S}^{-1}\bar{S}^T$. En effet, pour chaque vecteur propre α_i et valeur propre λ_i associés de $\hat{S}^{-1}\bar{S}^T$, nous avons :

$$\frac{\alpha_i^T \bar{S} \alpha_i}{\alpha_i^T \hat{S} \alpha_i} = \lambda_i \tag{6.9}$$

Par conséquent, le plus grand λ_i maximise l'équation 6.7 de Rayleigh. Finalement, nous construisons le profil pour chaque clef de la même manière que pour la PCA.

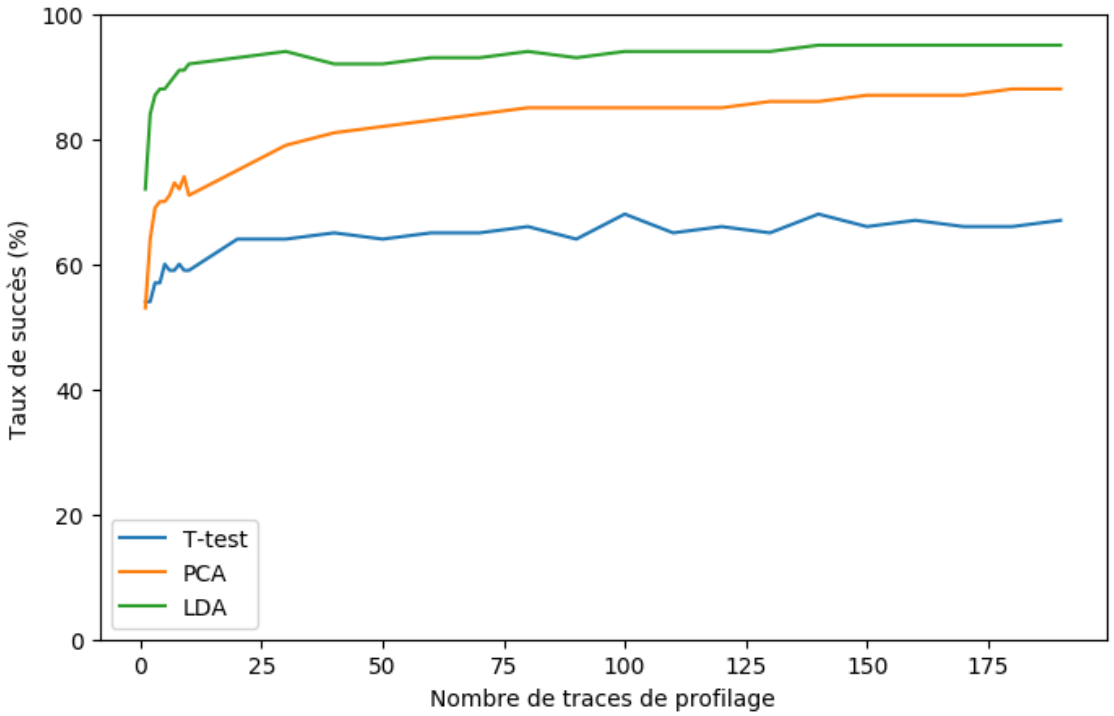


FIGURE 6.4 – Taux de succès d'inférer un bit de la clef en fonction du nombre de traces kP utilisées pour le profilage.

6.2.3 Résultats de l'attaque

Pour appliquer les méthodes de réduction PCA et LDA, nous utilisons la bibliothèque Python sklearn [PVG+11].

Nous avons collecté 200 traces du calcul de kP pour une clef k fixe. Cet ensemble de mesures a été séparé en deux :

1. les 190 premières ont été utilisées pour la construction des profils.
2. les 10 autres traces sont utilisées afin de mener la phase d'attaque pour retrouver k.

Notons que pour les traces de profilage ou d'attaque, nous appliquons la recherche des `cswap` décrite à la section 6.2.1. Ainsi, toutes les traces sont découpées en 253 sous-traces correspondant aux opérations de `cswap` pour chaque bit de la clef. Nous excluons les 4 premiers `cswap`, et donc la première étape du calcul de kP. Ces premières opérations sont biaisées car l'initialisation du calcul se fait avec les points P et l'élément neutre $\mathcal{O}$ dont les coordonnées $(X : Z)$ sont $(0 : 0)$. De plus, ce premier bit de la clef est connu, car le standard [LHT16] impose que le bit 254 de la clef soit 1.

Il s'en suit que nous avons jusqu'à 48070 traces de 4 opérations `cswap` et donc 24035 traces pour construire le profil de fuite pour chaque clef possible, à savoir 0 et 1. De plus, nous avons 2530 traces sur lesquelles est menée l'attaque. Nous moyennons les résultats afin d'obtenir le pourcentage moyen du taux de succès pour inférer le bon bit de clef. La figure 6.4 représente le taux de succès de retrouver un bit de la clef.

Le taux de succès de l'approche simple par t-test avoisine les 60% ce qui est légèrement mieux que de deviner aléatoirement un bit de la clef. Ce résultat témoigne d'une fuite, et ce malgré l'utilisation de la contremesure d'ajout d'aléatoire dans la représentation des données. Cet aléa est différent à chaque calcul de kP. Cette fuite est mise en exergue par la PCA et surtout par la LDA. Dans ce cas précis, nous atteignons 95% de réussite. Nous pouvons alors inférer les bits inconnus par force brute car seuls 14 bits sont à retrouver.

6.2.4 Attaque « *online* »

Notre attaque est possible car nous pouvons labéliser les traces qui servent à construire les différents profils de fuite. Ceci est permis grâce à la bibliothèque mbedTLS [mbe19] est une bibliothèque dont les fichiers sont en libre accès (*opensource*). Cependant, cette hypothèse est forte, un attaquant n'a pas forcément accès à l'implémentation visée. Une implémentation propriétaire pourrait ne pas être, *a priori*, sensible à ce genre d'attaque.

L'attaque présentée dans ce chapitre peut être améliorée pour être applicable à ces cas. Pour cela, nous utilisons une particularité imposée par la norme [LHT16]. Certains bits de la clef sont fixes, le bit 254 est égal à 1, tandis que les bits 2, 1 et 0 sont nuls. Nous pouvons alors obtenir des traces du calcul kP et utiliser seulement les bits dont nous connaissons les valeurs afin de labelliser les traces pour l'étape de profilage.

Notre attaque peut alors être effectuée entièrement « en ligne ». De plus, notre modèle d'attaquant est très permissif et peut s'appliquer à de nombreux cas d'usages. L'attaquant peut ne pas avoir à interagir avec la cible, il peut seulement observer une grandeur physique lors du calcul de kP.

L'inconvénient de ce modèle d'attaquant est d'allonger la phase de profilage, car il devra obtenir un grand nombre de traces du calcul kP avant de pouvoir mener l'attaque. Cette différence provient du nombre de traces de `cswap` que nous pouvons extraire d'un calcul de kP. Dans la version classique de l'attaque, nous pouvons extraire 253 sous-traces pour l'apprentissage. Tandis que dans le cas restreint présenter dans cette section, seulement 4 sous-traces peuvent être extraites.

Cette version de l'attaque n'a pas été menée car elle diffère seulement sur la manière d'obtenir et de labéliser les traces qui serviront à la construction de nos profils. Le déroulement de l'attaque reste le même. L'allongement de la phase d'apprentissage rend cette version de l'attaque plus difficilement testable.

6.3 Masquage de l'opération d'échange conditionnel de vecteurs

Nous avons vu qu'une contremesure comme l'ajout d'aléatoire dans la représentation des coordonnées ne permet pas de se protéger de ce genre d'attaques par profilage. Les autres contremesures présentées dans la section 3.4 n'apportent aucune sécurité face à cette attaque :

- **Isomorphisme aléatoire** : cette contremesure a pour effet de changer la courbe utilisée et donc les coordonnées des points utilisés. Ainsi, cette contremesure est équivalente à celle des coordonnées aléatoires et ne permet pas de se prémunir de cette attaque.
- **Point caché** : cette protection est semblable à celle des coordonnées aléatoires et n'apporte donc aucune protection face à cette attaque.
- **Clef cachée** : les contremesures protégeant la clef n'apporteront non plus aucune protection, car nous avons besoin d'une seule trace d'attaque pour retrouver entièrement la clef. Cependant, lors de la phase de profilage cette contremesure peut empêcher la labellisation des traces de `cswap`. C'est pourquoi l'attaquant doit pouvoir contrôler l'utilisation de celle-ci lors de la phase de profilage. Ceci est possible lorsque l'attaquant a librement accès aux sources de l'implémentation ciblée. Ainsi, cette contremesure protège contre la version « en ligne » de notre attaque (section 6.2.4).
- **Séparation de la clef** : cette contremesure a le même impact sur notre attaque que la contremesure consistant à cacher la clef car elle a pour effet de changer

aléatoirement les bits de la clef secrète.

Ainsi, les contremesures de la littérature ne permettent pas de se prémunir de notre attaque par profilage. Il nous faut donc en construire une nouvelle permettant de se protéger.

La fuite exploitée par cette attaque découle des multiplications particulières faites lors d'un `cswap`. Il s'en suit qu'en masquant les multiplications effectuées nous devrions pouvoir nous prémunir de ce type d'attaque. Cependant, ce masque va dépendre de la formule utilisée pour effectuer un `cswap`.

Nous utilisons deux nombres aléatoires r_0 et r_1 d'une longueur en bit égale à celle des mots utilisés par l'opération de `cswap`. Ces nombres aléatoires ou masques sont tirés au début de chaque opération `cswap`.

- **Équations 5.1 et 5.2** : la version masquée de ces formules est donnée par :

$$R_{0,i} = R_{0,i} + (r_0 + b)(R_{1,i} - R_{0,i}) - (r_0 + r_1)(R_{1,i} - R_{0,i}) + r_1(R_{1,i} - R_{0,i}) \quad (6.10)$$
$$R_{0,i} = R_{0,i} - (r_0 + b)(R_{1,i} - R_{0,i}) + (r_0 + r_1)(R_{1,i} - R_{0,i}) - r_1(R_{1,i} - R_{0,i}) \quad (6.11)$$

- **Équations 6.3 et 6.4** : la version masquée de ces formules est donnée par :

$$R_{0,i} = (b + r_0)R_{1,i} + (1 - (b + r_1))R_{0,i} - r_0R_{1,i} - r_1R_{0,i} \quad (6.12)$$
$$R_{0,i} = (b + r_0)R_{0,i} + (1 - (b + r_1))R_{1,i} - r_0R_{0,i} - r_1R_{1,i} \quad (6.13)$$

- **Equations 6.1 et 6.2** : la version masquée de ces formules est donnée par :

$$R_{0,i} = R_{0,i} \oplus (r_0 \oplus b) \wedge (R_{1,i} \oplus R_{0,i}) \oplus (r_0 \oplus r_1) \wedge (R_{1,i} \oplus R_{0,i}) \oplus r_1 \wedge (R_{1,i} \oplus R_{0,i}) \quad (6.14)$$
$$R_{1,i} = R_{1,i} \oplus (r_0 \oplus b) \wedge (R_{1,i} \oplus R_{0,i}) \oplus (r_0 \oplus r_1) \wedge (R_{1,i} \oplus R_{0,i}) \oplus r_1 \wedge (R_{1,i} \oplus R_{0,i}) \quad (6.15)$$

Ces formules de `cswap` modifiées ont l'avantage d'effectuer des multiplications ou des « et » logiques différents quel que soit le bit conditionnel considéré. Ainsi, aucun lien n'associe ces opérations. Il est alors difficile de trouver le bit de condition manipulé par le `cswap`. Les résultats de notre attaque sur la version masquée du `cswap` utilisé par mbedTLS (équations 6.12 et 6.13) sont exposés dans la section 6.3.1.

6.3.1 Résultats de l'attaque sur une implémentation masquée

Pour vérifier l'efficacité de notre contremesure, nous avons modifié la fonction `mbedtls_mpi_safe_con` de mbedTLS qui effectue un échange conditionnel de vecteurs à partir des équations 6.3 et 6.4. Ce changement permet d'utiliser notre contremesure qui est donnée par les équations 6.12 et 6.13. Nous appliquons alors exactement la même attaque que décrite dans la section 6.2. Les résultats sont donnés par la Figure 6.7.

La figure 6.5 présente une trace de 4 `cswap` consécutifs. Le résultat du t-test sur l'ensemble des traces de profilages des `cswap` masqués est donné par la figure 6.6.

Le taux de succès d'inférer un bit avoisine les 50% pour toutes les méthodes de construction des profils. Ainsi, il est impossible de faire mieux que le hasard pour retrouver un bit de la clef. Nous pouvons conclure que notre contremesure présentée dans la section 6.3 protège l'implémentation contre notre attaque par profilage.

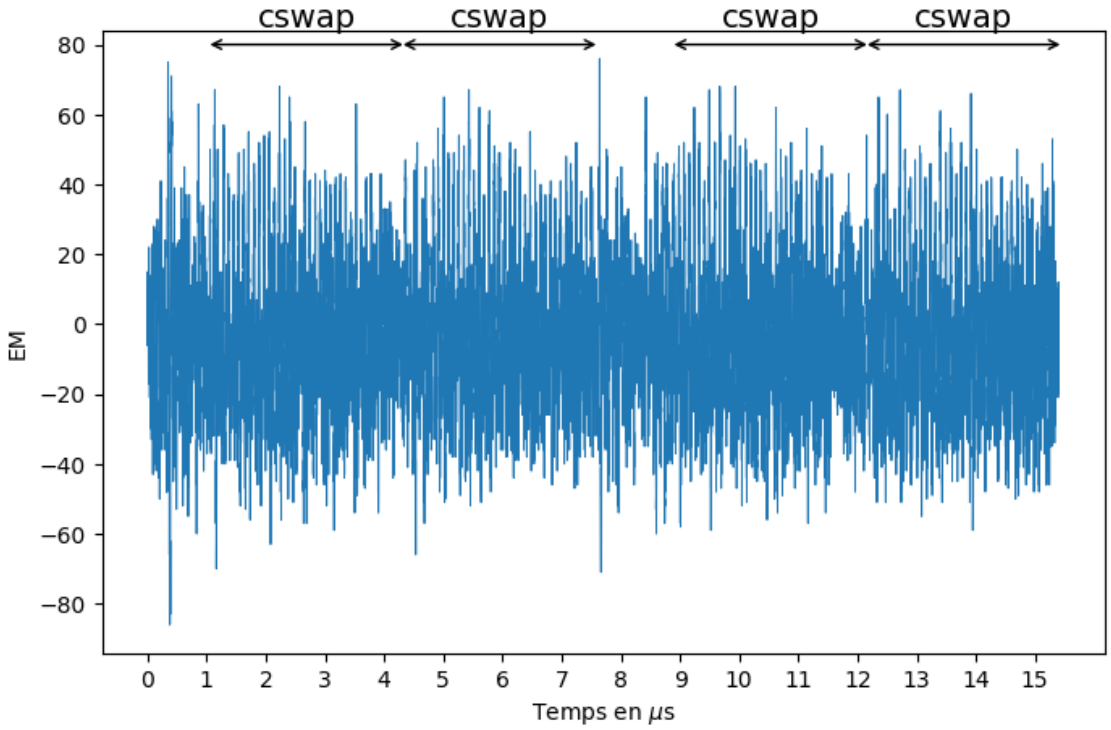


FIGURE 6.5 – Trace électro-magnétique de 4 cswap consécutifs utilisant la contre-mesure décrite par les équations 6.12 et 6.13.

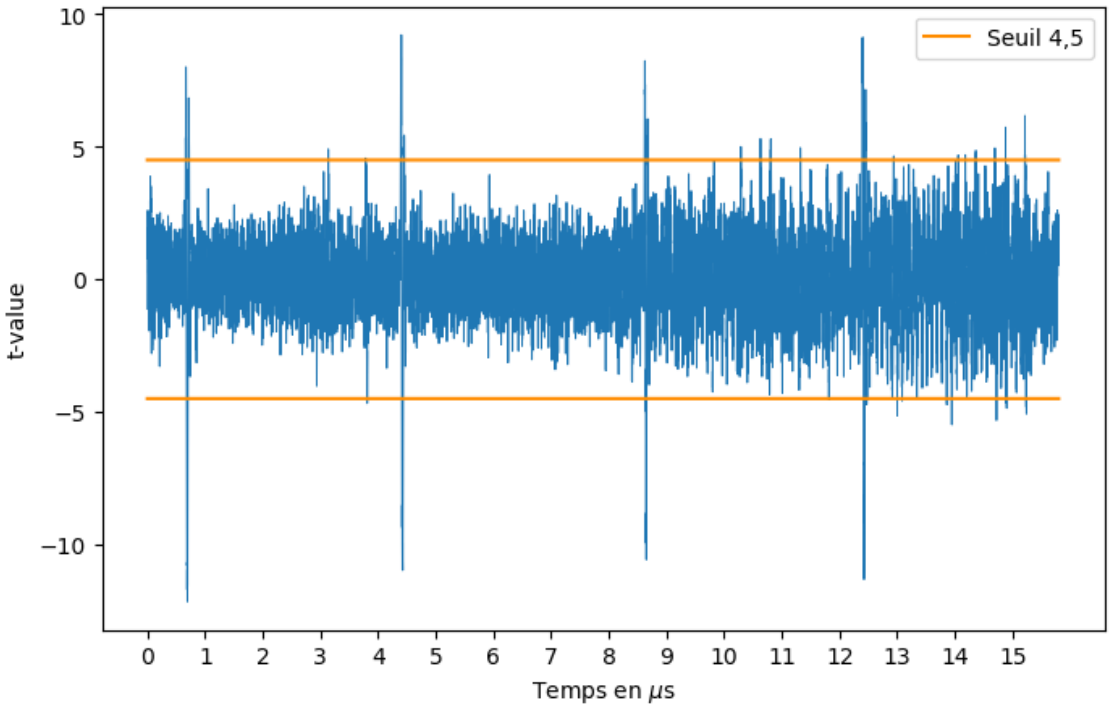


FIGURE 6.6 – Résultat du t-test entre les cswap (figure 6.5) pour le bit 0 et le bit 1. 50000 traces ont été utilisées dans chaque classe.

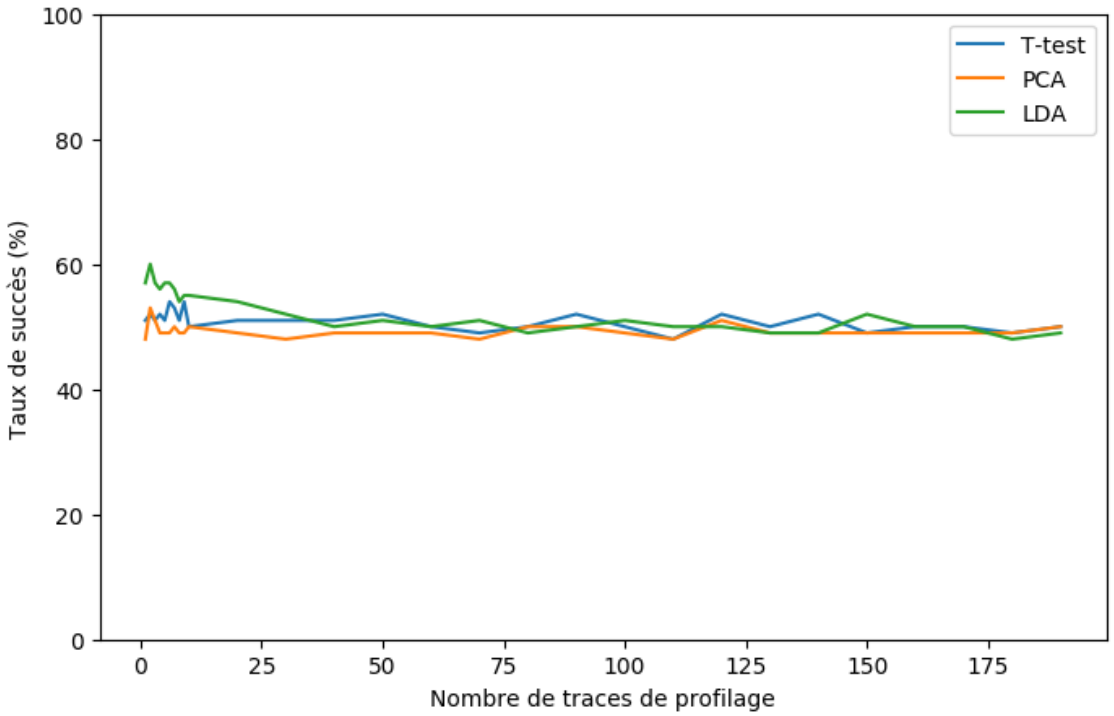


FIGURE 6.7 – Taux de succès d'inférer un bit de la clef en fonction du nombre de traces kP utilisées pour le profilage. La version du cswap est celle des formules 6.12 et 6.13.

6.4 Conclusion

Ce chapitre étudie la mise en œuvre et l'efficacité des attaques par profilage. Nous avons vu que l'opération d'échange conditionnel de vecteurs, couramment employée dans les implémentations cryptographiques, est sensible aux attaques par profilage. En isolant cette opération du reste du calcul kP, nous sommes en mesure de retrouver jusqu'à 95% des bits d'une clef ECC.

De plus, l'efficacité de cette attaque permet d'outrepasser la contremesure qui ajoute de l'aléa dans la représentation projective des points et est efficace contre celles qui cachent la clef.

Cette attaque a été menée sur l'implémentation mbedTLS de la Curve25519 sur un microcontrôleur possédant une architecture 32 bits ARM. Cependant, cette attaque n'est pas inhérente à l'implémentation de mbedTLS et touche toutes les bibliothèques cryptographiques utilisant la même stratégie pour effectuer kP. Nous avons obtenu des résultats similaires sur notre implémentation des BEC sur RISC V.

Pour pallier cette faiblesse, nous avons développé une nouvelle contremesure qui a été validée dans le cas de l'implémentation de mbedTLS. Cette contremesure masque l'opération d'échange conditionnel à l'aide de nombres aléatoires. Cette protection a fait l'objet d'un brevet (FR1904667). De plus, cette attaque ainsi que la contremesure de la section 6.3 font l'objet d'une publication soumise à HOST 2020.

Il est a noter que cette attaque a déjà été menée par Nascimento *et al.* [NCOS16]. Dans cette étude les auteurs présentent une attaque par profilage sur l'opération de « et » logique des équations 6.1 et 6.2. Cette attaque a été menée en isolant spécifiquement l'instruction de « et » logique. De plus, l'architecture considérée à un chemin de donnée de 8 bits et le microcontrôleur utilisé est cadencé à 7 MHz. Dans notre cas nous considérons une implémentation commerciale (mbedTLS [mbe19]) des équations 6.3 et 6.4. Le microcontrôleur considéré est une architecture ARM 32 bits cadencée à 168 MHz ce qui rend difficile l'isolation précise d'une instruction au sein d'un calcul complet de kP. Ainsi, notre attaque considère que l'attaquant ne possède que des traces complètes du calcul kP, à la fois pour la phase de construction des profils d'attaque et la phase d'application de ces derniers. Notre attaque est proche d'un cas d'usage réel de l'implémentation de la Curve25519 de mbedTLS **??**

Chapitre 7

Conclusion et perspectives

Les travaux exposés dans ce manuscrit abordent la problématique des implémentations de cryptographie sur les courbes elliptiques efficaces et sécurisées pour l'IoT. Ces travaux ont été impulsés par différentes considérations :

- le constat que le principal standard [Gal13] de la cryptographie ne prend pas en compte toutes les avancées mathématiques de ces 20 dernières années sur l'arithmétique des courbes elliptiques, en particulier pour les systèmes embarqués.
- l'émergence de nouveaux modèles alternatifs à celui de Weierstrass, améliorant les implémentations de cette cryptographie.
- le nombre croissant d'objets connectés contraints, ou IoT, accentuent le besoin d'avoir un meilleur compromis entre sécurité et performance qui n'étaient pas au centre de l'élaboration des premiers standards de cryptographie sur les courbes elliptiques.
- le besoin d'avoir des solutions déployables dès à présent.

Une des problématiques majeures de sécurité de ces nouveaux objets connectés est liée aux attaques dites physiques. Ce biais d'attaque dans le domaine de la cryptographie sur les courbes elliptiques amène de nouveaux enjeux lors d'une implémentation sur les courbes elliptiques. Or, il est très difficile d'ajouter des protections à une implémentation ou une arithmétique déjà existante sans en dégrader les performances. Malgré tout, l'ajout de contremesures est nécessaire pour garantir la sécurité d'une implémentation d'un algorithme cryptographique sur les courbes elliptiques dans un environnement IoT. Une cryptographie prenant en compte ces problématiques dès sa conception permettrait de garantir plus aisément une sécurité physique tout en garantissant des performances acceptables pour des objets IoT contraint.

Bernstein *et al.* [BDL+11b, BJL+15] proposent de nouvelles courbes elliptiques qui ont conduit à l'élaboration d'un nouveau standard [LHT16]. Cependant, nous pouvons aller plus loin en termes de conception de courbes elliptiques adaptées aux contraintes de l'IoT.

Dans cette thèse, nous avons donc étudié les contraintes de sécurité physique rendue nécessaire par le fait que les objets IoT sont facilement accessibles, afin d'aboutir à une analyse fine des faiblesses apparaissant dans un algorithme cryptographique basé sur les courbes elliptiques. De cette analyse nous pouvons en extraire le modèle de courbes elliptiques le plus adapté à l'IoT et l'implémentation sous-jacente. Néanmoins, il subsiste des faiblesses d'implémentations face aux attaques physiques ne pouvant être résolues au regard du choix particulier d'un modèle de courbes elliptiques ou du choix d'implémentation : par exemple, les attaques par profilages nécessitent l'intégration d'une contremesure spécifique.

Nous avons opté pour le modèle de courbes binaires d'Edwards (BEC) défini sur une extension de $\mathbb{F}_2$, introduit par Bernstein *et al.* [BLF08]. Les points de la courbe sont représentés sous forme différentielle, ce qui permet d'effectuer une addition et un doublement en seulement 5 multiplications, 4 élévations au carré et une multiplication courte par une constante.

L'autre choix que nous avons fait est porté sur l'utilisation de l'algorithme de l'Échelle de Montgomery pour effectuer la multiplication scalaire nécessaire à un grand nombre de protocoles standards de la cryptographie sur les courbes elliptiques. Cette méthode de calcul de kP a des propriétés de sécurité intrinsèque face aux attaques physiques évitant le rajout de contremesures coûteuses et offrant ainsi un bon compromis entre sécurité et performance.

Les implémentations que nous avons faites de primitives cryptographiques basées sur ces nouvelles courbes et choix de paramètres nous ont permis de valider la sécurité du modèle de courbes elliptiques, des algorithmes utilisés, des spécificités d'implémentations et de l'intégration dans les standards (ECDH, ECDSA et EdDSA) et de démontrer leurs efficacité en matière de performances. Cette implémentation permet d'obtenir des protections intrinsèques face aux attaques simples et verticales, ainsi qu'à la plupart des attaques par fautes bien que des contremesures doivent être ajoutées pour pallier, *a priori*, des faiblesses sur la sécurité des courbes elliptiques face aux attaques physiques. Les contremesures implémentées et validées en pratique sont :

- l'ajout d'aléatoire dans la représentation des coordonnées
- la vérification du point d'entrée et de sortie. Cette validation doit porter sur l'appartenance du point de base à la courbe et elle doit aussi vérifier que le point de base est différent d'un point de petit ordre.

Une fois le modèle de courbe elliptique choisi, nous avons choisi les paramètres qui définissent les courbes elliptiques que nous utiliserons. Nous aurions pu utiliser les courbes standards définies par le NIST [Gal13], cependant, les paramètres choisis par le NIST ne sont pas adaptés au modèle BEC pour pouvoir exploiter au mieux l'arithmétique de ce modèle de courbes elliptiques. Ainsi, nous avons fait le choix de construire nos propres courbes elliptiques afin de profiter au mieux des performances du modèle de BEC. Nous avons construit de nouvelles courbes elliptiques utilisant le modèle BEC et adressant différents niveaux de sécurité destinés à différents cas d'usages en matière de niveau de sécurité, à l'image du standard du NIST [Gal13] où plusieurs paramètres de courbes elliptiques sont proposés pour différents niveaux de sécurité.

La construction d'un nouvel ensemble de courbes elliptiques à des fins d'applications cryptographiques doit prendre en compte un certain nombre de critères de sécurité. Le principal critère concerne le cardinal de la courbe elliptique, qui doit être de la forme $n = hp$ avec h petit et p premier. Nous avons le même critère pour la tordue de la courbe. Ces critères sont très discriminants et la plupart de courbes elliptiques ne les vérifient pas.

Nous décomposons les implémentations cryptographiques sur les BEC en trois niveaux distincts qui peuvent être optimisés à tour de rôle :

1. Le premier est le choix de l'extension de corps de $\mathbb{F}_2$ et du module. Nous aurions pu choisir d'utiliser les polynômes définis par le NIST [Gal13]. Cependant, les travaux de Scott [Sco07] ont démontré la possibilité d'intégrer les contraintes de l'architecture matérielle sur laquelle l'implémentation est faite, dans la construction des modules. Ces modules alternatifs permettent d'optimiser, pour une taille du chemin de données et des registres, l'opération de réduction modulaire.
2. Le second choix concerne celui du paramètre d définissant notre courbe BEC. Ce dernier doit vérifier tous les critères de sécurité mathématiques. Nous pouvons le

sélectionner creux et de petit degré afin de réduire le nombre d'opérations nécessaires pour la multiplication par d.

3. Le troisième et dernier critère à choisir est celui du point générateur qui sera utilisé dans les différents protocoles cryptographiques. Nous pouvons choisir aléatoirement un générateur. Cependant, comme les formules d'addition et doublement différentielles couplées à l'Échelle de Montgomery impliquent l'exécution d'une multiplication par l'inverse de la coordonnée différentielle de ce générateur à chaque étape du calcul de kP, en choisissant cet inverse creux et de petit degré, la multiplication par ce dernier aura une complexité et donc un temps d'exécution négligeable par rapport à une multiplication complète. Nous avons donc choisi des inverses de la coordonnée différentielle du générateur de manière à simplifier la multiplication par celui-ci. Ainsi nous avons considéré la taille du chemin de données pour contraindre un choix qui sera optimisé pour ce dernier.

Les travaux sur la génération de ce nouvel ensemble de BEC ont fait l'objet d'un brevet (FR3071081), d'une publication à Secrypt 2018 [LF18] et d'une présentation au séminaire SoSySec en Septembre 2018.

Par la suite, nous avons étudié l'intégration de ce nouvel ensemble de BEC au sein des protocoles standards de signature et d'échange de clefs. Nous avons démontré que nous avons une compatibilité avec les protocoles standards (ECDH, ECDSA et EdDSA). Cette intégration au sein des protocoles de signature n'est pas triviale en raison de la représentation différentielle qui amène une incertitude lorsque l'on essaie de retrouver les coordonnées affines. Dans certains cas, une signature correctement générée sera invérifiable. Deux solutions de contournement au problème ont été proposées :

1. La première vient du constat que cette incertitude est cantonnée au bit de poids faible des coordonnées affines. Ainsi, en fixant arbitrairement celui-ci, nous pouvons lever toute incertitude et vérifier n'importe quelle signature.
2. La seconde est d'utiliser les coordonnées différentielles tout au long du protocole de signature. Pour cela, nous utilisons un algorithme d'Échelle de Montgomery à deux dimensions tel que proposé par Bernstein et Lange [BL17]. Cependant, cette solution s'avère moins performante que la première. En améliorant la chaîne d'addition utilisée, nous pouvons si nécessaire améliorer les performances de cette solution.

Une fois l'ensemble de BEC construit, nous avons fait les implémentations logicielles associées sur un processeur RISC V. La comparaison à l'état de l'art à notre disposition a démontré que notre implémentation est la plus performante en termes de temps de calcul pour les niveaux élevés de sécurité (au-delà de 256 bits) et talonne les implémentations de la Curve25519. Ce dernier résultat montre que même sans instruction de multiplication dans $\mathbb{F}_{2^n}$, notre solution est aussi performant que des implémentations sur la Curve25519 utilisant l'instruction de multiplication dans les entiers. La complexité de l'opération de multiplication sur les entiers ou sur les polynômes est essentielle, car elle représente l'opération la plus impactant en matière de temps de calcul des opérations effectuées lors du calcul de kP.

Nous avons vérifié les propriétés de sécurité de notre implémentation, propriétés théoriquement attendues de sécurité face aux attaques physiques. Cette phase de validation de la sécurité face aux attaques par canaux auxiliaires nous a conduit à élaborer une nouvelle attaque. Celle-ci consiste à injecter une faute lors du calcul de kP et à l'analyse par canal auxiliaire de la propagation de celle-ci au sein du calcul. La méthode utilisée est basée sur la recherche de collisions potentielles entre une trace d'une grandeur physique n'ayant pas subi de fautes et une autre ayant subi une faute à une étape précise du calcul. Nous expliquons aussi comment nous pouvons nous prémunir de ce type d'attaque en utilisant

la contremesure qui ajoute de l'aléatoire dans la représentation des coordonnées.

Nous nous sommes concentrés plus précisément sur les attaques par profilage car celles-ci nécessitent une étude toute particulière de par leur efficacité et versatilité. Notre attaque par profilage cible les étapes d'échange conditionnel de vecteurs ou `cswap`. Cette étape que l'on retrouve dans la majeure partie des implémentations de l'Échelle de Montgomery manipule directement les bits de la clef. Ainsi, nous pouvons construire une attaque par profilage afin de trouver quel bit est utilisé. Afin d'améliorer les performances de notre attaque, nous avons utilisé des méthodes de réduction de dimension telles que la PCA et la LDA. Ces dernières nous ont permis d'augmenter significativement le taux de succès de notre attaque. Nous sommes passés d'un taux de succès de 65% à 95% en utilisant la LDA. La force de celle-ci réside dans la possibilité d'attaquer des cas réels de mis en œuvre de la cryptographie sur les courbes elliptiques au sein de protocoles standards (ECDH, ECDSA et EdDSA). Il nous suffit d'une trace du calcul kP pour retrouver la clef k. De plus, cette attaque fonctionne malgré l'utilisation de la contremesure qui consiste à ajouter de l'aléatoire dans la représentation des points. L'attaque n'est pas inhérente à un modèle particulier de courbes elliptiques, c'est pourquoi nous l'avons menée sur l'implémentation de mbedTLS [mbe19] de la Curve25519 sur un microcontrôleur basé sur une architecture ARM 32 bits.

La contrainte inhérente à ce genre d'attaque est de pouvoir labelliser les traces qui serviront pour la phase de construction des profils. Ainsi, si l'attaquant n'a pas accès à une copie de l'implémentation attaquée dont il peut contrôler les entrées, alors l'attaque ne peut pas être menée. Nous avons pu contourner ce problème en utilisant une spécificité du standard [LHT16] qui force certains bits de la clef à des valeurs spécifiques. En utilisant cette connaissance nous pouvons ajouter des labels aux traces d'apprentissage que nous obtenons et donc mener complètement l'attaque en ligne.

L'analyse des contremesures classiques de la cryptographie sur les courbes elliptiques nous a montré que de toutes celles proposées à ce jour dans la littérature, aucune ne pouvait nous protéger contre cette dernière. Ainsi, nous avons élaboré une nouvelle contremesure qui consiste à masquer les opérations effectuées lors de l'étape de `cswap`. Nous avons validé notre contremesure en l'intégrant dans notre implémentation en vérifiant que le taux de succès de notre attaque était satisfaisant.

Cette attaque par profilage ainsi que la nouvelle contremesure proposée font l'objet d'une publication soumise à HOST 2020. De plus, la contremesure a fait l'objet d'un brevet (FR1904667).

Perspectives

Les travaux exposés dans ce manuscrit traitent de la construction de la cryptographie sur les courbes elliptiques pour l'IoT et de la sécurisation de leur implémentation face aux attaques par canaux auxiliaires. Cependant, à ce stade, plusieurs points peuvent encore être approfondis et développés.

Le premier point concerne l'évolution du jeu d'instructions RISC V pour favoriser l'implémentation logicielle des BEC. Pour mémoire, l'implémentation décrite et réalisée dans ces travaux a été faite sur un microcontrôleur embarquant un processeur RISC V de 32 bits. Nous pourrions améliorer grandement les performances de notre implémentation en dotant notre processeur RISC V d'une instruction de multiplication dans $\mathbb{F}_{2^n}$. Cet ajout

permettrait de rapprocher notre implémentation de celles effectuées pour les courbes elliptiques définies sur un corps à grande caractéristique. Par conséquent, en comparant une implémentation ECC sure $\mathbb{F}_p$ et notre implémentation des BEC sure $\mathbb{F}_{2^n}$, nous pourrions quantifier les accélérations dues au choix du modèle de courbe elliptique ou aux différentes optimisations menées lors de la génération des paramètres. De manière générale, l'identification des opérations coûteuses en termes de temps de calcul permettrait de faire du « co-design » matériel-logiciel pour définir un jeu d'instructions « minimaliste » accélérant les performances. Cette approche nous permettrait de profiter d'une accélération matérielle en termes de performance de calcul sans pour autant construire un crypto-processeur dédié coûteux en termes de surface.

De même, une implémentation entièrement matérielle de ces nouvelles courbes elliptiques permettrait d'estimer l'impact des choix effectués sur les performances d'une implémentation matérielle. De plus, une implémentation compacte pourrait être utilisée au sein des architectures des processeurs afin de sécuriser notamment pour des mécanismes de « secure boot » nécessitant une vérification de signature du logiciel avant de l'exécuter [SHEM19].

Le deuxième point adresse notre vérification des propriétés de sécurité contre les attaques physiques de notre implémentation dont nous avons exclu les attaques par injection de fautes. Cette étude n'a pu être menée durant cette thèse par manque de temps. Il serait intéressant d'effectuer cette évaluation sur notre implémentation. De plus, la nouvelle attaque que nous avons élaborée et qui combine attaque par faute et canaux auxiliaires pourrait être menée sur un cas réel afin de valider sa faisabilité et de définir quelle contremesure est le plus efficace.

Le troisième point porte sur l'évaluation des attaques par profilage. Notre attaque est indépendante du modèle de courbes elliptiques utilisé. L'étude de cas des attaques par profilage sur l'arithmétique des BEC reste à étudier. Un des objectifs pourrait être de voir si les opérations effectuées lors d'une étape de l'Échelle de Montgomery laissent-elles fuir de l'information et sont-elles exploitables par une attaque par profilage. Les opérations d'addition et de doublement de points effectuées à chaque étape dépendent indirectement des bits de la clef. Ainsi, pouvons-nous utiliser cette dépendance pour en déduire de l'information sur la clef utilisée ? La difficulté de cette approche réside dans la construction des profils. Une telle attaque reviendrait à effectuer une classification des différentes clefs possibles afin de pouvoir inférer des bits de la clef.

Dans ces travaux, nous avons mis l'accent sur les attaques par profilage. Ainsi, un autre point qui pourrait être étudié concerne les attaques horizontales. Ces dernières peuvent s'avérer dangereuses pour une implémentation cryptographique, car elles ne demandent qu'une seule et unique trace du calcul cryptographique. Ces attaques sont très dépendantes des choix d'implémentation effectués. Ainsi, nous devons étudier les faiblesses de notre implémentation face à ce genre d'attaques. La manière de mener ces attaques dépend aussi des choix algorithmiques de notre implémentation. Par exemple, la recherche d'opérandes communs entre deux sous-opérations du calcul de kP peut être difficile car lors d'une multiplication de Lòpez-Dahab les opérandes d'entrée sont traités de manière asymétrique. De même, les choix du modèle de BEC et son arithmétique peuvent, peut-être, être utilisés afin de construire un nouveau biais d'attaque. Une telle attaque serait propre aux BEC et serait impossible sur les modèles de Weierstrass ou d'Edwards. Ces interrogations, *a priori*, sont à confirmer ou infirmer *a posteriori*. Il serait donc important d'étudier tous ces aspects.

De plus, une attaque horizontale peut être vue comme un problème de classification qui consiste à distinguer si une étape du calcul kP correspond au bit 0 ou 1. Ainsi, les techniques d'attaques utilisant l'apprentissage profond peuvent amener à un nouveau biais d'attaque. Dans le cas de la cryptographie symétrique, de nombreux travaux ont montré l'efficacité de ces techniques [CDP17, PSB+18]. Cependant, dans le cadre de la cryptographie sur les courbes elliptiques, aucune étude à notre connaissance n'utilise ce genre de techniques. Cette perspective permettrait d'augmenter l'efficacité des attaques connues en termes de temps de mise en œuvre mais aussi en termes d'applicabilité des attaques. Le but de cette étude serait alors de construire de nouvelles contremesures dans la mesure où les contremesures actuellement connues ne suffiraient pas à se protéger de ce genre d'attaque.

Références

[ABB+17] Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, and Gilles Zémor. BIKE : Bit Flipping Key Encapsulation. 2017. Submission to the NIST post quantum standardization process.

[ACL19] Rodrigo Abarzúa, Claudio Valencia Cordero, and Julio López. Survey for performance & security problems of passive side-channel attacks countermeasures in ECC. *IACR Cryptology ePrint Archive*, 2019 :10, 2019.

[AK97] Ross J. Anderson and Markus G. Kuhn. Low cost attacks on tamper resistant devices. In *Security Protocols, 5th International Workshop, Paris, France, April 7-9, 1997, Proceedings*, pages 125–136, 1997.

[Aki01] Toru Akishita. Fast simultaneous scalar multiplication on elliptic curve with montgomery form. In *Selected Areas in Cryptography, 8th Annual International Workshop, SAC 2001 Toronto, Ontario, Canada, August 16-17, 2001, Revised Papers*, pages 255–267, 2001.

[ALH10] Diego F. Aranha, Julio López, and Darrel Hankerson. Efficient software implementation of binary field arithmetic using vector instruction sets. In *International Conference on Cryptology and Information Security in Latin America*, pages 144–161. Springer, 2010.

[ans14] Référentiel Génŕal de Sécurité version 2.0 - Annexe B1. Technical report, ANSSI, 2014.

[APS19] Melissa Azouaoui, Romain Poussier, and François-Xavier Standaert. Fast side-channel security evaluation of ECC implementations - shortcut formulas for horizontal side-channel attacks against ECSM with the montgomery ladder. In *Constructive Side-Channel Analysis and Secure Design - 10th International Workshop, COSADE 2019, Darmstadt, Germany, April 3-5, 2019, Proceedings*, pages 25–42, 2019.

[APSQ06] Cédric Archambeau, Eric Peeters, François-Xavier Standaert, and Jean-Jacques Quisquater. Template attacks in principal subspaces. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pages 1–14, 2006.

[AR12] Reza Azarderakhsh and Arash Reyhani-Masoleh. Efficient FPGA implementations of point multiplication on binary edwards and generalized hessian curves using gaussian normal basis. *IEEE Trans. VLSI Syst.*, 20(8) :1453–1466, 2012.

[AT03] Toru Akishita and Tsuyoshi Takagi. Zero-value point attacks on elliptic curve cryptosystem. In *Information Security, 6th International Conference,*

ISC 2003, Bristol, UK, October 1-3, 2003, Proceedings, pages 218–233, 2003.

[Atk88] AOL Atkin. The Number of Points an an Elliptic Curve Modulo a Prime, manuscript. *Chicago IL, January*, 1, 1988.

[Bar16] Elaine Barker. Recommendation for Key Management Part 1 : General. Rapport technique NIST SP 800-57pt1r4, National Institute of Standards and Technology, 2016.

[BBJ+08] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted Edwards Curves. In *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, pages 389–405, 2008.

[BCKL15] Daniel J. Bernstein, Chitchanok Chuengsatiansup, David Kohel, and Tanja Lange. Twisted Hessian Curves. In *Progress in Cryptology - LATINCRYPT 2015 - 4th International Conference on Cryptology and Information Security in Latin America, Guadalajara, Mexico, August 23-26, 2015, Proceedings*, pages 269–294, 2015.

[BCL14] Daniel J. Bernstein, Chitchanok Chuengsatiansup, and Tanja Lange. Curve41417 : Karatsuba revisited. *IACR Cryptology ePrint Archive*, 2014 :526, 2014.

[BCO04] Eric Brier, Christophe Clavier, and Francis Olivier. Correlation power analysis with a leakage model. In *Cryptographic Hardware and Embedded Systems - CHES 2004 : 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 16–29, 2004.

[BCR+03] Elaine Barker, Lily Chen, Allen Roginsky, Apostol Vassilev, Richard Davis, and Scott Simon. Recommendation for pair-wise key establishment using integer factorization cryptography, 2019-03.

[BDK+18] Joppe Bos, Leo Ducas, Eike Kiltz, T Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehle. CRYSTALS - Kyber : A CCA-Secure Module-Lattice-Based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367, 2018.

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 14(2) :101–119, Mar 2001.

[BDL+11a] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 124–142. Springer, 2011.

[BDL+11b] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-Speed High-Security Signatures. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 124–142, 2011.

[Ber06] Daniel J. Bernstein. Differential addition chains. *http ://cr.yp.to/ecdh/diffchain-20060219.pdf*, 2006.

[Ber09] Daniel J. Bernstein. Batch binary edwards. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 317–336, 2009.

[BHvW12] Lejla Batina, Jip Hogenboom, and Jasper G. J. van Woudenberg. Getting more from PCA : first results of using principal component analysis for

extensive power analysis. In *Topics in Cryptology - CT-RSA 2012 - The Cryptographers' Track at the RSA Conference 2012, San Francisco, CA, USA, February 27 - March 2, 2012. Proceedings*, pages 383–397, 2012.

[BJL+15] Daniel J. Bernstein, Simon Josefsson, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. EdDSA for more curves. *IACR Cryptology ePrint Archive*, 2015 :677, 2015.

[BJPW13] Aurélie Bauer, Éliane Jaulmes, Emmanuel Prouff, and Justine Wild. Horizontal collision correlation attack on elliptic curves. In *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, pages 553–570, 2013.

[BL07] Daniel J. Bernstein and Tanja Lange. Inverted Edwards Coordinates. In *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 17th International Symposium, AAECC-17, Bangalore, India, December 16-20, 2007, Proceedings*, pages 20–27, 2007.

[BL17] Daniel J. Bernstein and Tanja Lange. Montgomery curves and the montgomery ladder. *IACR Cryptology ePrint Archive*, 2017 :293, 2017.

[BLF08] Daniel J. Bernstein, Tanja Lange, and Reza Rezaeian Farashahi. Binary Edwards Curves. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 244–265, 2008.

[BMM00] Ingrid Biehl, Bernd Meyer, and Volker Müller. Differential fault attacks on elliptic curve cryptosystems. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 131–146, 2000.

[BN05] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, pages 319–331, 2005.

[BOS06] Johannes Blömer, Martin Otto, and Jean-Pierre Seifert. Sign change fault attacks on elliptic curve cryptosystems. In *Fault Diagnosis and Tolerance in Cryptography, Third International Workshop, FDTC 2006, Yokohama, Japan, October 10, 2006, Proceedings*, pages 36–52, 2006.

[BR19] Elaine Barker and Allen Roginsky. Transitioning the use of cryptographic algorithms and key lengths. Technical Report NIST SP 800-131Ar2, National Institute of Standards and Technology, 2019.

[CDP15] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Enhancing dimensionality reduction methods for side-channel attacks. In *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, pages 15–33, 2015.

[CDP17] Eleonora Cagli, Cécile Dumas, and Emmanuel Prouff. Convolutional neural networks with data augmentation against jitter-based countermeasures - profiling attacks without pre-processing. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 45–68, 2017.

[CFA+06] Henri Cohen, Gerhard Frey, Roberto Avanzi, Christophe Doche, Tanja Lange, Kim Nguyen, and Frederik Vercauteren. *Handbook of elliptic and hyperelliptic curve cryptography*. Discrete mathematics and its applications. Chapman & Hall/CRC, 2006.

[CFG+10] Christophe Clavier, Benoit Feix, Georges Gagnerot, Mylène Roussellet, and Vincent Verneuil. Horizontal correlation analysis on exponentiation. In *Information and Communications Security - 12th International Conference, ICICS 2010, Barcelona, Spain, December 15-17, 2010. Proceedings*, pages 46–61, 2010.

[CFG+12] Christophe Clavier, Benoit Feix, Georges Gagnerot, Christophe Giraud, Mylène Roussellet, and Vincent Verneuil. ROSETTA for single trace analysis. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 140–155, 2012.

[CFMR+17] A Casanova, J.-C. Faugère, G Macaio-Rat, J Patarin, L Perret, and J Ryckeghem. GeMSS : A Great Multivariate Short Signature. 2017. Submission to the NIST post quantum standardization process.

[CJ01] Christophe Clavier and Marc Joye. Universal exponentiation algorithm. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, number Generators, pages 300–308, 2001.

[CJ03] Mathieu Ciet and Marc Joye. (virtually) free randomization techniques for elliptic curve cryptography. In *Information and Communications Security, 5th International Conference, ICICS 2003, Huhehaote, China, October 10-13, 2003, Proceedings*, pages 348–359, 2003.

[CJ05] Mathieu Ciet and Marc Joye. Elliptic curve cryptosystems in the presence of permanent and transient faults. *Des. Codes Cryptography*, 36(1) :33–43, 2005.

[CJL+16] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. Report on post-quantum cryptography. Rapport technique NIST IR 8105, National Institute of Standards and Technology, 2016.

[CK14] Omar Choudary and Markus G. Kuhn. Template attacks on different devices. *IACR Cryptology ePrint Archive*, 2014 :459, 2014.

[CK18] Marios O. Choudary and Markus G. Kuhn. Efficient, portable template attacks. *IEEE Trans. Information Forensics and Security*, 13(2) :490–501, 2018.

[Cla04] Christophe Clavier. Side channel analysis for reverse engineering (scare) – an improved attack against a secret a3/a8 gsm algorithm. *IACR Cryptology ePrint Archive*, page 49, 01 2004.

[Cor99] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, pages 292–302, 1999.

[CRR02] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 13–28, 2002.

[CRR03] Suresh Chari, Josyula R. Rao, and Pankaj Rohatgi. Template Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002*, volume 2523, pages 13–28. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[CTV17] Earl T. Campbell, Barbara M. Terhal, and Christophe Vuillot. Roads towards fault-tolerant universal quantum computation. In *Nature*, volume 549, page 172, 2017.

[DC06] Christophe De Cannière. Trivium : A stream cipher construction inspired by block cipher design principles. In Sokratis K. Katsikas, Javier López, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *Information Security*, volume 4176, pages 171–186. Springer Berlin Heidelberg, 2006.

[DCP05] Christophe De Cannière and Bart Preneel. TRIVIUM specications. Rapport technique, ECRYPT Stream Cipher Project, 2005.

[dCUVHV14] Ruan de Clercq, Leif Uhsadel, Anthony Van Herrewege, and Ingrid Verbauwhede. Ultra Low-Power implementation of ECC on the ARM Cortex-M0+. In *Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference - DAC '14*, pages 1–6. ACM Press, 2014.

[DES77] Data Encryption Standard. Standard, National Bureau of Standards, 1977.

[DFJP14] Luca De Feo, David Jao, and Jérôme Plût. Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies. 8(3) :209–247, 2014.

[dG15] Wouter de Groot. A Performance Study of X25519 on Cortex-M3 and M4. 2015. Master Thesis.

[DGH+13] Jean-Luc Danger, Sylvain Guilley, Philippe Hoogvorst, Cédric Murdica, and David Naccache. A synthesis of side-channel attacks on elliptic curve cryptography in smart-cards. *J. Cryptographic Engineering*, 3(4) :241–265, 2013.

[DH76a] Whitfield Diffie and Martin E. Hellman. Multiuser cryptographic techniques. In *American Federation of Information Processing Societies : 1976 National Computer Conference, 7-10 June 1976, New York, NY, USA*, pages 109–112, 1976.

[DH76b] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Trans. Information Theory*, 22(6) :644–654, 1976.

[DHH+15] Michael Düll, Björn Haase, Gesine Hinterwälder, Michael Hutter, Christof Paar, Ana Helena Sánchez, and Peter Schwabe. High-speed Curve25519 on 8-bit, 16-bit, and 32-bit microcontrollers. 77(2) :493–514, 2015.

[DIK06] Christophe Doche, Thomas Icart, and David R. Kohel. Efficient Scalar Multiplication by Isogeny Decompositions. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, pages 191–206, 2006.

[DJ11] Julien Devigne and Marc Joye. Binary huff curves. In *Topics in Cryptology - CT-RSA 2011 - The Cryptographers' Track at the RSA Conference 2011, San Francisco, CA, USA, February 14-18, 2011. Proceedings*, pages 340–355, 2011.

[DKL+98] Jean-François Dhem, François Koeune, Philippe-Alexandre Leroux, Patrick Mestré, Jean-Jacques Quisquater, and Jean-Louis Willems. A practical implementation of the timing attack. In *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*, pages 167–182, 1998.

[DKL+18] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-dilithium : A lattice-based digital signature scheme. In *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018.

[DLR15] Jean-Guillaume Dumas, Pascal Lafourcade, and Patrick Redon. *Architectures PKI et communications sécurisées*. 2015.

[DPN+16a] Margaux Dugardin, Louiza Papachristodoulou, Zakaria Najm, Lejla Batina, Jean-Luc Danger, and Sylvain Guilley. Dismantling real-world ECC with horizontal and vertical template attacks. In François-Xavier Standaert and Elisabeth Oswald, editors, *Constructive Side-Channel Analysis and Secure Design*, volume 9689, pages 88–108. Springer International Publishing, 2016.

[DPN+16b] Margaux Dugardin, Louiza Papachristodoulou, Zakaria Najm, Lejla Batina, Jean-Luc Danger, and Sylvain Guilley. Dismantling real-world ECC with horizontal and vertical template attacks. In *Constructive Side-Channel Analysis and Secure Design - 7th International Workshop, COSADE 2016, Graz, Austria, April 14-15, 2016, Revised Selected Papers*, pages 88–108, 2016.

[dV18] B. de Vigenere. *Traicte Des Chiffres, Ou Secretes Manieres d'Escrire, (Ed.1586)*. HACHETTE LIVRE, 2018.

[ecr18] Algorithms, Key Size and Protocols Report. Rapport technique, 2018.

[Edw07] Harold Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44(3) :393–422, 2007.

[elk91] Explicit isogenies. *Draft*, 1991.

[FAB18] Luckas A. Farias, Bruno C. Albertini, and Paulo S. L. M. Barreto. A class of safe and efficient binary Edwards curves. *J. Cryptographic Engineering*, 8(4) :271–283, 2018.

[FGLCLT08] Christian Franck, Johann Grozschadl, Yann Le Corre, and Cyrille Lenou Tago. Energy-Scalable Montgomery-Curve ECDH Key Exchange for ARM Cortex-M3 Microcontrollers. In *2018 6th International Conference on Future Internet of Things and Cloud Workshops (FiCloudW)*, pages 231–236. IEEE, 2018-08.

[FGM+10] Junfeng Fan, Xu Guo, Elke De Mulder, Patrick Schaumont, Bart Preneel, and Ingrid Verbauwhede. State-of-the-art of secure ECC implementations : A survey on known side-channel attacks and countermeasures. In *HOST 2010, Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 13-14 June 2010, Anaheim Convention Center, California, USA*, pages 76–87, 2010.

[FGV11] Junfeng Fan, Benedikt Gierlichs, and Frederik Vercauteren. To infinity and beyond : Combined attack on ECC using points of low order. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 143–159, 2011.

[FH16] Reza Rezaeian Farashahi and Seyed Gholamhossein Hosseini. Differential addition on binary elliptic curves. In *Arithmetic of Finite Fields - 6th International Workshop, WAIFI 2016, Ghent, Belgium, July 13-15, 2016, Revised Selected Papers*, pages 21–35, 2016.

[Fis38] Ronald A Fisher. The statistical utilization of multiple measurements. *Annals of eugenics*, 8(4) :376–386, 1938.

[FJ10] Reza R. Farashahi and Marc Joye. Efficient Arithmetic on Hessian Curves. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography – PKC 2010*, volume 6056, pages 243–260. Springer Berlin Heidelberg, 2010.

[FLRV08a] Pierre-Alain Fouque, Reynald Lercier, Denis Réal, and Frédéric Valette. Fault attack on elliptic curve montgomery ladder implementation. In *Fault Diagnosis and Tolerance in Cryptography, 2008. FDTC'08. 5th Workshop on*, pages 92–98. IEEE, 2008.

[FLRV08b] Pierre-Alain Fouque, Reynald Lercier, Denis Réal, and Frédéric Valette. Fault attack on elliptic curve montgomery ladder implementation. In *Fifth International Workshop on Fault Diagnosis and Tolerance in Cryptography, 2008, FDTC 2008, Washington, DC, USA, 10 August 2008*, pages 92–98, 2008.

[FNW10] Rongquan Feng, Menglong Nie, and Hongfeng Wu. Twisted Jacobi Intersections Curves. In *Theory and Applications of Models of Computation, 7th Annual Conference, TAMC 2010, Prague, Czech Republic, June 7-11, 2010. Proceedings*, pages 199–210, 2010.

[FRVD08] Pierre-Alain Fouque, Denis Réal, Frédéric Valette, and M'hamed Drissi. The carry leakage on the randomized exponent countermeasure. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 198–213, 2008.

[FSK16] Apostolos P. Fournaris, Nicolas Sklavos, and Christos Koulamas. A High Speed Scalar Multiplier for Binary Edwards Curves. In *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems, CS2@HiPEAC, Prague, Czech Republic, January 20, 2016*, pages 41–44, 2016.

[FTPN14] Jacques Fournier, Assia Tria, Florian Pebay, and Dominique Noguet. Security Challenges facing the IoT. In *Pan European Networks Science & Technology*, pages 66–67, November 2014.

[FV03] Pierre-Alain Fouque and Frédéric Valette. The doubling attack - *Why Upwards Is Better than Downwards*. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, pages 269–280, 2003.

[FV12] Junfeng Fan and Ingrid Verbauwhede. An updated survey on secure ECC implementations : Attacks, countermeasures and cost. In *Cryptography and Security : From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, pages 265–282, 2012.

[Gal13] Patrick Gallagher. Digital signature standard (dss). Rapport technique, Federal Information Processing Standards Publications, volume FIPS 186-4, 2013.

[gcc19] GCC. https ://gcc.gnu.org/, 2019.

[GG16] Steven D. Galbraith and Pierrick Gaudry. Recent progress on the elliptic curve discrete logarithm problem. *Designs, Codes and Cryptography*, 78(1) :51–72, 2016.

[GHS02] Pierrick Gaudry, Florian Hess, and Nigel P. Smart. Constructive and destructive facets of weil descent on elliptic curves. *J. Cryptology*, 15(1) :19–46, 2002.

[Gir06] Christophe Giraud. An RSA implementation resistant to fault attacks and to simple power analysis. *IEEE Trans. Computers*, 55(9) :1116–1120, 2006.

[GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis : Concrete results. In Çetin K. Koç, David Naccache, and Christof

Paar, editors, *Cryptographic Hardware and Embedded Systems — CHES 2001*, pages 251–261, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Gou03] Louis Goubin. A refined power-analysis attack on elliptic curve cryptosystems. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, pages 199–210, 2003.

[GVY17a] Daniel Genkin, Luke Valenta, and Yuval Yarom. May the fourth be with you : A microarchitectural side channel attack on several real-world applications of curve25519. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security - CCS '17*, pages 845–858. ACM Press, 2017.

[GVY17b] Daniel Genkin, Luke Valenta, and Yuval Yarom. May the fourth be with you : A microarchitectural side channel attack on several real-world applications of curve25519. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 845–858, 2017.

[Ham15] Mike Hamburg. Ed448-goldilocks, a new elliptic curve. *IACR Cryptology ePrint Archive*, 2015 :625, 2015.

[HJP13] W. Hart, F. Johansson, and S. Pancratz. FLINT : Fast Library for Number Theory, 2013. Version 2.4.0, http ://flintlib.org.

[HWCD08] Huseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Twisted Edwards curves revisited. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 326–343. Springer, 2008.

[HWCD09] Hüseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Jacobi Quartic Curves Revisited. In *Information Security and Privacy, 14th Australasian Conference, ACISP 2009, Brisbane, Australia, July 1-3, 2009, Proceedings*, pages 452–468, 2009.

[IISO10] Masami Izumi, Jun Ikegami, Kazuo Sakiyama, and Kazuo Ohta. Improved countermeasure against address-bit DPA for ECC scalar multiplication. In *Design, Automation and Test in Europe, DATE 2010, Dresden, Germany, March 8-12, 2010*, pages 981–984, 2010.

[IIT02] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. Address-bit differential power analysis of cryptographic schemes OK-ECDH and OK-ECDSA. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 129–143, 2002.

[IIT03] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. A practical countermeasure against address-bit differential power analysis. In *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, pages 382–396, 2003.

[IIT04] Kouichi Itoh, Tetsuya Izu, and Masahiko Takenaka. Efficient countermeasures against power analysis for elliptic curve cryptosystems. In *Smart Card Research and Advanced Applications VI, IFIP 18th World Computer Congress, TC8/WG8.8 & TC11/WG11.2 Sixth International Conference on Smart Card Research and Advanced Applications (CARDIS), 22-27 August 2004, Toulouse, France*, pages 99–113, 2004.

[iot15] Internet of Things : Privacy and Security in a Connected World. Rapport technique, Federal Trade Commission, 2015.

[IT88] Toshiya Itoh and Shigeo Tsujii. A Fast Algorithm for Computing Multiplicative Inverses in GF(2^m) Using Normal Bases. In *Information and Computation*, volume 78, pages 171–177, 1988.

[IT03] Tetsuya Izu and Tsuyoshi Takagi. Exceptional procedure attack on elliptic curve cryptosystems. In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, pages 224–239, 2003.

[JL17] S. Josefsson and I. Liusvaara. Edwards-Curve Digital Signature Algorithm (EdDSA). Standard RFC8032, RFC Editor, 2017.

[JMV01] Don Johnson, Alfred Menezes, and Scott A. Vanstone. The Elliptic Curve Digital Signature Algorithm (ECDSA). *Int. J. Inf. Sec.*, 1(1) :36–63, 2001.

[Jol11] Ian T. Jolliffe. Principal component analysis. In *International Encyclopedia of Statistical Science*, pages 1094–1096, 2011.

[JQ01] Marc Joye and Jean-Jacques Quisquater. Hessian Elliptic Curves and Side-Channel Attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, number Generators, pages 402–410, 2001.

[JT01] Marc Joye and Christophe Tymen. Protections against differential analysis for elliptic curve cryptography. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, pages 377–390, 2001.

[JT17] Marc Joye and Michael Tunstall. *Fault Analysis in Cryptography*. Springer, 2017.

[JTV10] Marc Joye, Mehdi Tibouchi, and Damien Vergnaud. Huff's Model for Elliptic Curves. In *Algorithmic Number Theory, 9th International Symposium, ANTS-IX, Nancy, France, July 19-23, 2010. Proceedings*, pages 234–250, 2010.

[JY02] Marc Joye and Sung-Ming Yen. The montgomery powering ladder. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 291–302. Springer, 2002.

[Kah96] David Kahn. *The Codebreakers : The comprehensive history of secret communication from ancient times to the internet*. Simon and Schuster, 1996.

[KAK15] Brian Koziel, Reza Azarderakhsh, and Mehran Mozaffari Kermani. Low-Resource and Fast Binary Edwards Curves Cryptography. In *Progress in Cryptology - INDOCRYPT 2015 - 16th International Conference on Cryptology in India, Bangalore, India, December 6-9, 2015, Proceedings*, pages 347–369, 2015.

[KDKL17] Ievgen Kabin, Zoya Dyka, Dan Kreiser, and Peter Langendörfer. Horizontal address-bit DPA against montgomery kp implementation. In *International Conference on ReConFigurable Computing and FPGAs, ReConFig 2017, Cancun, Mexico, December 4-6, 2017*, pages 1–8, 2017.

[KDKL18] Ievgen Kabin, Zoya Dyka, Dan Kreiser, and Peter Langendörfer. Horizontal address-bit DEMA against ECDSA. In *9th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2018, Paris, France, February 26-28, 2018*, pages 1–7, 2018.

[Ker83] Auguste Kerckhoffs. La Cryptographie Militaire. *Journal des Sciences Militaires*, 1883.

[KJI+17] Takuya Kusaka, Sho Joichi, Ken Ikuta, Md. Al-Amin Khandaker, Yasuyuki Nogami, Satoshi Uehara, Nariyoshi Yamai, and Sylvain Duquesne. Solving 114-bit ECDLP for a barreto-naehrig curve. In *Information Security and Cryptology - ICISC 2017 - 20th International Conference, Seoul, South Korea, November 29 - December 1, 2017, Revised Selected Papers*, pages 231–244, 2017.

[KJI+18] Takuya Kusaka, Sho Joichi, Ken Ikuta, Md. Al-Amin Khandaker, Yasuyuki Nogami, Satoshi Uehara, Nariyoshi Yamai, and Sylvain Duquesne. Solving 114-bit ECDLP for a barreto-naehrig curve. In Howon Kim and Dong-Chan Kim, editors, *Information Security and Cryptology – ICISC 2017*, volume 10779, pages 231–244. Springer International Publishing, 2018.

[KJJ99] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential Power Analysis. In *Proceedings of the 19th International Advances in Cryptology Conference (CRYPTO'99)*, number 1666 in LNCS, pages 388–397. Springer-Verlag, 1999.

[KLN14] Kwang Ho Kim, Chol Ok Lee, and Christophe Nègre. Binary Edwards Curves Revisited. In *Progress in Cryptology - INDOCRYPT 2014 - 15th International Conference on Cryptology in India, New Delhi, India, December 14-17, 2014, Proceedings*, pages 393–408, 2014.

[KO63] A. Karatsuba and Yuri Petrovich Ofman. Multiplication of many-digital numbers by automatic computers. 1963.

[Kob87] N. Koblitz. Elliptic curve cryptosystems. *Mathematics of Computation*, 48(177) :243–264, 1987.

[Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, pages 104–113, 1996.

[Koh12] David Kohel. Efficient arithmetic on elliptic curves in characteristic 2. In *Progress in Cryptology - INDOCRYPT 2012, 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings*, pages 378–398, 2012.

[Koh17] David Kohel. Twisted μ_4-normal form for elliptic curves. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, pages 659–678, 2017.

[LCC+06] Thanh-Ha Le, Jessy Clédière, Cécile Canovas, Bruno Robisson, Christine Servière, and Jean-Louis Lacoume. A proposition for correlation power analysis enhancement. In *Cryptographic Hardware and Embedded Systems - CHES 2006, 8th International Workshop, Yokohama, Japan, October 10-13, 2006, Proceedings*, pages 174–186, 2006.

[LD00] Julio López and Ricardo Dahab. High-Speed Software Multiplication in GF(2^m). In *Proceedings of the First International Conference on Progress in Cryptology*, INDOCRYPT '00, pages 203–212, London, UK, UK, 2000. Springer-Verlag.

[LDM19] Carlos Andres Lara-Nino, Arturo Diaz-Perez, and Miguel Morales-Sandoval. Energy/Area-Efficient Scalar Multiplication with Binary Edwards Curves for the IoT. *Sensors*, 19(3) :720, 2019.

[Len87] H. W. Lenstra. Factoring integers with elliptic curves. *The Annals of Mathematics*, 126(3) :649, 1987.

[LF18] Antoine Loiseau and Jacques J. A. Fournier. Binary Edwards Curves for Intrinsically Secure ECC Implementations for the IoT. In *Proceedings of the 15th International Joint Conference on e-Business and Telecommunications, ICETE 2018 - Volume 2 : SECRYPT, Porto, Portugal, July 26-28, 2018.*, pages 625–631, 2018.

[LHT16] A. Langley, M. Hamburg, and S. Turner. Elliptic curves for security. Standard RFC7748, RFC Editor, 2016.

[LM10] Manfred Lochter and Johannes Merkle. Elliptic Curve Cryptography (ECC) Brainpool Standard Curves and Curve Generation. *RFC 5639*, 2010.

[LSC+19] Zhe Liu, Hwajeong Seo, Aniello Castiglione, Kim-Kwang Raymond Choo, and Howon Kim. Memory-Efficient Implementation of Elliptic Curve Cryptography for the Internet-of-Things. 16(3) :521–529, 2019.

[Mac17] K. MacKey. MicroECC : ECDH and ECDSA for 8-bit, 32-bit, and 64-bitprocessors. https ://github.com/kmackay/micro-ecc, 2017.

[mbe19] mbedtls. https ://tls.mbed.org/, 2019.

[Mes00a] Jean-Fraçois Mestre. Application de l'AGM au calcul du nombre de points d'une courbe de genre 1, 2000. Exposé donné à Rennes.

[Mes00b] Jean-Fraçois Mestre. Lettre adressé à Gaudry et Harley. https ://webusers.imj-prg.fr/ jean-francois.mestre/, 2000.

[MGD+12] Cédric Murdica, Sylvain Guilley, Jean-Luc Danger, Philippe Hoogvorst, and David Naccache. Same values power analysis using special points on elliptic curves. In *Constructive Side-Channel Analysis and Secure Design - Third International Workshop, COSADE 2012, Darmstadt, Germany, May 3-4, 2012. Proceedings*, pages 183–198, 2012.

[Mil86] VS Miller. Use of Elliptic-Curves in Cryptography. *Lecture Notes in Computer Science*, 218 :417–426, 1986.

[MMM04] Hideyo Mamiya, Atsuko Miyaji, and Hiroaki Morimoto. Efficient countermeasures against rpa, dpa, and SPA. In *Cryptographic Hardware and Embedded Systems - CHES 2004 : 6th International Workshop Cambridge, MA, USA, August 11-13, 2004. Proceedings*, pages 343–356, 2004.

[MMNT13] Diana Maimut, Cédric Murdica, David Naccache, and Mehdi Tibouchi. Fault attacks on projective-to-affine coordinates conversion. In *Constructive Side-Channel Analysis and Secure Design - 4th International Workshop, COSADE 2013, Paris, France, March 6-8, 2013, Revised Selected Papers*, pages 46–61, 2013.

[MO08] Marcel Medwed and Elisabeth Oswald. Template attacks on ECDSA. In *Information Security Applications, 9th International Workshop, WISA 2008, Jeju Island, Korea, September 23-25, 2008, Revised Selected Papers*, pages 14–27, 2008.

[Mon87] Peter L Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48 :243–264, 1987.

[Mur14] Cédric Murdica. *Physical security of elliptic curve cryptography. (Sécurité physique de la cryptographie sur courbes elliptiques)*. PhD thesis, Télécom ParisTech, France, 2014.

[MV06] Frédéric Muller and Frédéric Valette. High-order attacks against the exponent splitting protection. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, pages 315–329, 2006.

[NCOS16] Erick Nascimento, Lukasz Chmielewski, David Oswald, and Peter Schwabe. Attacking embedded ECC implementations through cmov side channels. In *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, pages 99–119, 2016.

[NJ16] Y. Nir and S. Josefsson. Curve25519 and Curve448 for the Internet Key Exchange Protocol Version 2 (IKEv2) Key Agreement. Standard RFC8031, RFC Editor, 2016.

[NM16] Toshifumi Nishinaga and Masahiro Mambo. Implementation of μNaCl on 32-bit ARM Cortex-M0. *IEICE Transactions on Information and Systems*, E99.D(8) :2056–2060, 2016.

[noa19] IBM-Q. https ://www.research.ibm.com/ibm-q/, 2019.

[Nov03] Roman Novak. Side-channel attack on substitution blocks. In Jianying Zhou, Moti Yung, and Yongfei Han, editors, *Applied Cryptography and Network Security*, pages 307–318, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[NSS04] David Naccache, Nigel P. Smart, and Jacques Stern. Projective coordinates leak. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 257–267, 2004.

[Odl84] Andrew M. Odlyzko. Discrete logarithms in finite fields and their cryptographic significance. In *Advances in Cryptology : Proceedings of EUROCRYPT 84, A Workshop on the Theory and Application of of Cryptographic Techniques, Paris, France, April 9-11, 1984, Proceedings*, pages 224–314, 1984.

[OLAR13] Thomaz Oliveira, Julio López, Diego F. Aranha, and Francisco Rodríguez-Henríquez. Lambda coordinates for binary elliptic curves. In *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, pages 311–330, 2013.

[ope19] OpenSSL. https ://www.openssl.org/, 2019.

[Pol75] J. M. Pollard. A monte carlo method for factorization. *BIT Numerical Mathematics*, 15(3) :331–334, Sep 1975.

[PSB+18] Emmanuel Prouff, Remi Strullu, Ryad Benadjila, Eleonora Cagli, and Cécile Dumas. Study of deep learning techniques for side-channel analysis and introduction to ASCAD database. *IACR Cryptology ePrint Archive*, 2018 :53, 2018.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[PZS17] Romain Poussier, Yuanyuan Zhou, and François-Xavier Standaert. A systematic approach to the side-channel analysis of ECC implementations with

worst-case horizontal attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, pages 534–554, 2017.

[QS01] Jean-Jacques Quisquater and David Samyde. ElectroMagnetic Analysis (EMA) : Measures and Counter-measures for Smart Cards. In Isabelle Attali and Thomas Jensen, editors, *Smart Card Programming and Security*, pages 200–210, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Qui00] J.-J. Quisquater. A new tool for non-intrusive analysis of smart cards based on electro-magnetic emissions. The SEMA and DEMA methods. *Eurocrypt2000 Rump Session*, 2000.

[RCB16] Joost Renes, Craig Costello, and Lejla Batina. Complete Addition Formulas for Prime Order Elliptic Curves. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 403–428, 2016.

[RD01] Vincent Rijmen and Joan Daemen. Advanced Encryption Standard. *Proceedings of Federal Information Processing Standards Publications, National Institute of Standards and Technology*, 2001.

[Res18] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, August 2018.

[RFS17] Bahram Rashidi, Reza Rezaeian Farashahi, and Sayed Masoud Sayedi. High-speed hardware implementations of point multiplication for binary edwards and generalized hessian curves. *IACR Cryptology ePrint Archive*, 2017 :5, 2017.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2) :120–126, 1978.

[RSF17] Bahram Rashidi, Sayed Masoud Sayedi, and Reza Rezaeian Farashahi. Full-custom hardware implementation of point multiplication on binary edwards curves for application-specific integrated circuit elliptic curve cryptosystem applications. *IET Circuits, Devices & Systems*, 11(6) :568–578, 2017.

[Rub14] Paul Rubens. Internet of Things a Potential Security Disaster. https ://www.esecurityplanet.com/network-security/internet-of-things-a-potential-security-disaster.html, September 04 2014.

[SA03] Sergei P. Skorobogatov and Ross J. Anderson. Optical fault induction attacks. In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 2–12, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[SA08] François-Xavier Standaert and Cédric Archambeau. Using subspace-based template attacks to compare and combine power and electromagnetic information leakages. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 411–425, 2008.

[saf] SafeCurves. https ://safecurves.cr.yp.to/index.html. 2019-05-14.

[Sch85] René Schoof. Elliptic curves over finite fields and the computation of square roots mod *p*. *Mathemathics of Computation*, 44, 1985.

[Sch90] C. P. Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *Advances in Cryptology — CRYPTO' 89 Proceedings*, pages 239–252, New York, NY, 1990. Springer New York.

[Sco07] Michael Scott. Optimal Irreducible Polynomials for GF(2^m) Arithmetic. *IACR Cryptology ePrint Archive*, 2007 :192, 2007.

[Sha49a] Claude E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 1949.

[Sha49b] Claude E. Shannon. A Mathematical Theory of Cryptography. *Bell System Technical Journal*, 1949.

[Sha71] Daniel Shanks. Class Number, a Theory of Factorization and Genera. *Procedings of symposia in pure mathematics*, 20, 1971.

[SHEM19] Olivier Savry, Thomas Hiscock, and Mustapha El Majihi. *Sécurité matérielle des systèmes*. Dunod, 2019.

[Sho94] P.W. Shor. Algorithms for quantum computation : discrete logarithms and factoring. In *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE Comput. Soc. Press, 1994.

[Sil09] J.H. Silverman. *The Arithmetic of Elliptic Curves*. Graduate Texts in Mathematics. Springer New York, 2009.

[Sim81] GJ Simmons. Preliminary report on a theory of authentication. Rapport technique, Sandia National Labs., Albuquerque, NM (USA), 1981.

[Sim85] Gustavus J. Simmons. Authentication theory/coding theory. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pages 411–431, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[SM14] Antonio Skarmeta and M.Victoria Moreno. Internet of Things. In Willem Jonker and Milan Petkovic, editors, *Secure Data Management*, Lecture Notes in Computer Science, pages 48–53. Springer International Publishing, 2014.

[SM15] Tobias Schneider and Amir Moradi. Leakage assessment methodology - A clear roadmap for side-channel evaluations. In *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, pages 495–513, 2015.

[Sta02a] Martijn Stam. On Montgomery-Like Representations for Elliptic Curves over GF(2^k). In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 240–254, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[Sta02b] Martijn Stam. On montgomery-like representations for elliptic curves over $gf(2^k)$. In Yvo G. Desmedt, editor, *Public Key Cryptography — PKC 2003*, pages 240–254, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

[Sta03] Martijn Stam. On montgomery-like representationsfor elliptic curves over gf(2^{k}). In *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, pages 240–253, 2003.

[Sta14] J.A. Stankovic. Research directions for the internet of things. *IEEE Internet of Things Journal*, 1(1) :3–9, 2014.

[TB02] Elena Trichina and Antonio Bellezza. Implementation of elliptic curve cryptography with built-in counter measures against side channel attacks. In *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, pages 98–113, 2002.

[The19] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*, 2019. `https://www.sagemath.org`.

[TPG15] Hannes Tschofenig and Manuel Pégourié-Gonnard. Performance investigations. Présentation, 2015.

[tte18] Understanding leakage detection. Workshop, Reasure H2020 (CARDIS), 2018.

[USK+18] Yoshinori Uetake, Akihiro Sanada, Takuya Kusaka, Yasuyuki Nogami, Leo Weissbart, and Sylvain Duquesne. Side-channel attack using order 4 element against curve25519 on atmega328p. In *International Symposium on Information Theory and Its Applications, ISITA 2018, Singapore, October 28-31, 2018*, pages 618–622, 2018.

[Van92] Scott Vanstone. Responses to NIST's Proposal. *Communication of the ACM*, 35 :50–52, 1992.

[WAD17] Andrew Waterman, Krste Asanovic, and CS Division. The RISC-V Instruction Set ManualVolume i : User-level ISA. Rapport technique, RISC-V Foundation, 2017.

[Wal01] Colin D. Walter. Sliding windows succumbs to big mac attack. In *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, number Generators, pages 286–299, 2001.

[WTS+85] Charles C. Wang, Trieu-Kien Truong, Howard M. Shao, Leslie J. Deutsch, Jim K. Omura, and Irving S. Reed. VLSI architectures for computing multiplications and inverses in $GF(2^m)$. *IEEE Trans. Computers*, 34(8) :709–717, 1985.

[WUW13] Erich Wenger, Thomas Unterluggauer, and Mario Werner. 8/16/32 Shades of Elliptic Curve Cryptography on Embedded Processors. In Goutam Paul and Serge Vaudenay, editors, *Progress in Cryptology – INDOCRYPT 2013*, volume 8250, pages 244–261. Springer International Publishing, 2013.

[YJ00] Sung-Ming Yen and Marc Joye. Checking before output may not be enough against fault-based cryptanalysis. *IEEE Trans. Computers*, 49(9) :967–970, 2000.

[YKLM01] Sung-Ming Yen, Seungjoo Kim, Seongan Lim, and Sang-Jae Moon. A countermeasure against one physical cryptanalysis may benefit another attack. In *Information Security and Cryptology - ICISC 2001, 4th International Conference Seoul, Korea, December 6-7, 2001, Proceedings*, pages 414–427, 2001.

[YKMH05] Sung-Ming Yen, Lee-Chun Ko, Sang-Jae Moon, and JaeCheol Ha. Relative doubling attack against montgomery ladder. In *Information Security and Cryptology - ICISC 2005, 8th International Conference, Seoul, Korea, December 1-2, 2005, Revised Selected Papers*, pages 117–128, 2005.

Annexe A

Algorithmes de réduction modulaire dans $\mathbb{F}_{2^n}$

A.1 Polynômes irréductibles du NIST

Algorithme A.1 Réduction par $x^{163} + x^7 + x^6 + x^3 + 1$

Entrées Registres $r_{10}, .., r_0$ du polynôme devant être réduit

1: $a_5 \leftarrow r_5 \oplus r_{10} \oplus (r_{10} \ll 3) \oplus (r_9 \gg 29) \oplus (r_{10} \ll 4) \oplus (r_9 \gg 28) \oplus (r_{10} \gg 3)$
2: $a_4 \leftarrow r_4 \oplus r_9 \oplus (r_9 \ll 3) \oplus (r_8 \gg 29) \oplus (r_9 \ll 4) \oplus (r_8 \gg 28) \oplus (r_9 \gg 3) \oplus (r_{10} \ll 29)$
3: $a_3 \leftarrow r_3 \oplus r_8 \oplus (r_8 \ll 3) \oplus (r_7 \gg 29) \oplus (r_8 \ll 4) \oplus (r_7 \gg 28) \oplus (r_8 \gg 3) \oplus (r_9 \ll 29)$
4: $a_2 \leftarrow r_2 \oplus r_7 \oplus (r_7 \ll 3) \oplus (r_6 \gg 29) \oplus (r_7 \ll 4) \oplus (r_6 \gg 28) \oplus (r_7 \gg 3) \oplus (r_8 \ll 29)$
5: $a_1 \leftarrow r_1 \oplus r_6 \oplus (r_6 \ll 3) \oplus (a_5 \gg 29) \oplus (r_6 \ll 4) \oplus (a_5 \gg 28) \oplus (r_6 \gg 3) \oplus (r_7 \ll 29)$
6: $A_5 \leftarrow a_5 \wedge \texttt{0xFFFFFFF8}$
7: $a_0 \leftarrow r_0 \oplus A_5 \oplus (A_5 \ll 3) \oplus (A_5 \ll 4) \oplus (a_5 \gg 3) \oplus (r_6 \ll 29)$
8: $a_5 \leftarrow a_5 \wedge \texttt{0x00000007}$
9: **Retourner** $a_5, ..., a_0$

Algorithme A.2 Réduction par $x^{233} + x^{74} + 1$

Entrées Registres $r_{14}, .., r_0$ du polynôme devant être réduit

1: $a_9 \leftarrow r_9 \oplus (r_{13} \gg 31) \oplus (r_{14} \ll 1)$
2: $a_8 \leftarrow r_8 \oplus (r_{12} \gg 31) \oplus (r_{13} \ll 1)$
3: $a_7 \leftarrow r_7 \oplus (r_{14} \gg 9) \oplus (r_{11} \gg 31) \oplus (r_{12} \ll 1)$
4: $a_6 \leftarrow r_6 \oplus (r_{13} \gg 9) \oplus (r_{14} \ll 23) \oplus (r_{10} \gg 31) \oplus (r_{11} \ll 1)$
5: $a_5 \leftarrow r_5 \oplus (r_{12} \gg 9) \oplus (r_{13} \ll 23) \oplus (a_9 \gg 31) \oplus (r_{10} \ll 1)$
6: $a_4 \leftarrow r_4 \oplus (r_{11} \gg 9) \oplus (r_{12} \ll 23) \oplus (a_8 \gg 31) \oplus (a_9 \ll 1)$
7: $a_3 \leftarrow r_3 \oplus (r_{10} \gg 9) \oplus (r_{11} \ll 23) \oplus (a_7 \gg 31) \oplus (a_8 \ll 1)$
8: $a_2 \leftarrow r_2 \oplus (a_9 \gg 9) \oplus (r_{10} \ll 23) \oplus ((a_7 \wedge \texttt{0xFFFFFC00}) \ll 1)$
9: $a_1 \leftarrow r_1 \oplus (a_8 \gg 9) \oplus (a_9 \ll 23)$
10: $a_0 \leftarrow r_0 \oplus (a_7 \gg 9) \oplus (a_8 \ll 23)$
11: $a_7 \leftarrow a_7 \wedge \texttt{0x000001FF}$
12: **Retourner** $a_7, ..., a_0$

Algorithme A.3 Réduction par $x^{283} + x^{12} + x^7 + x^5 + 1$

Entrées Registres $r_{17}, .., r_0$ du polynôme devant être réduit

1: $a_9 \leftarrow r_9 \oplus (r_{17} \gg 15) \oplus (r_{17} \gg 20) \oplus (r_{17} \gg 22)$
2: $a_8 \leftarrow r_8 \oplus (r_{16} \gg 27) \oplus (r_{17} \ll 5) \oplus (r_{16} \gg 15) \oplus (r_{17} \ll 17) \oplus (r_{16} \gg 20) \oplus (r_{17} \ll 12) \oplus (r_{16} \gg 22) \oplus (r_{17} \ll 10)$
3: $a_7 \leftarrow r_7 \oplus (r_{15} \gg 27) \oplus (r_{16} \ll 5) \oplus (r_{15} \gg 15) \oplus (r_{16} \ll 17) \oplus (r_{15} \gg 20) \oplus (r_{16} \ll 12) \oplus (r_{15} \gg 22) \oplus (r_{16} \ll 10)$
4: $a_6 \leftarrow r_6 \oplus (r_{14} \gg 27) \oplus (r_{15} \ll 5) \oplus (r_{14} \gg 15) \oplus (r_{15} \ll 17) \oplus (r_{14} \gg 20) \oplus (r_{15} \ll 12) \oplus (r_{14} \gg 22) \oplus (r_{15} \ll 10)$
5: $a_5 \leftarrow r_5 \oplus (r_{13} \gg 27) \oplus (r_{14} \ll 5) \oplus (r_{13} \gg 15) \oplus (r_{14} \ll 17) \oplus (r_{13} \gg 20) \oplus (r_{14} \ll 12) \oplus (r_{13} \gg 22) \oplus (r_{14} \ll 10)$
6: $a_4 \leftarrow r_4 \oplus (r_{12} \gg 27) \oplus (r_{13} \ll 5) \oplus (r_{12} \gg 15) \oplus (r_{13} \ll 17) \oplus (r_{12} \gg 20) \oplus (r_{13} \ll 12) \oplus (r_{12} \gg 22) \oplus (r_{13} \ll 10)$
7: $a_3 \leftarrow r_3 \oplus (r_{11} \gg 27) \oplus (r_{12} \ll 5) \oplus (r_{11} \gg 15) \oplus (r_{12} \ll 17) \oplus (r_{11} \gg 20) \oplus (r_{12} \ll 12 \oplus (r_{11} \gg 22) \oplus (r_{12} \ll 10))$
8: $a_2 \leftarrow r_2 \oplus (r_{10} \gg 27) \oplus (r_{11} \ll 5) \oplus (r_{10} \gg 15) \oplus (r_{11} \ll 17) \oplus (r_{10} \gg 20) \oplus (r_{11} \ll 12) \oplus (r_{10} \gg 22) \oplus (r_{11} \ll 10)$
9: $a_1 \leftarrow r_1 \oplus (a_9 \gg 27) \oplus (r_{10} \ll 5) \oplus (a_9 \gg 15) \oplus (r_{10} \ll 17) \oplus (a_9 \gg 20) \oplus (r_{10} \ll 12) \oplus (a_9 \gg 22) \oplus (r_{10} \ll 10)$
10: $A_8 \leftarrow a_8 \wedge$ `0xF8000000`
11: $a_0 \leftarrow r_0 \oplus (a_8 \gg 27) \oplus (a_9 \ll 5) \oplus (A_8 \gg 15) \oplus (a_9 \ll 17) \oplus (A_8 \gg 20) \oplus (a_9 \ll 12) \oplus (A_8 \gg 22) \oplus (a_9 \ll 10)$
12: $a_8 \leftarrow a_8 \wedge$ `0x07FFFFFF`
13: **Retourner** $a_8, ..., a_0$

Algorithme A.4 Réduction par $x^{409} + x^{87} + 1$

Entrées Registres $r_{25}, .., r_0$ du polynôme devant être réduit

1: $a_{15} \leftarrow r_{15} \oplus (r_{25} \gg 2)$
2: $a_{14} \leftarrow r_{14} \oplus (r_{24} \gg 2) \oplus (r_{25} \ll 30)$
3: $a_{13} \leftarrow r_{13} \oplus (r_{23} \gg 2) \oplus (r_{24} \ll 30)$
4: $a_{12} \leftarrow r_{12} \oplus (r_{24} \gg 25) \oplus (r_{25} \ll 7) \oplus (r_{22} \gg 2) \oplus (r_{23} \ll 30)$
5: $a_{11} \leftarrow r_{11} \oplus (r_{23} \gg 25) \oplus (r_{24} \ll 7) \oplus (r_{21} \gg 2) \oplus (r_{22} \ll 30)$
6: $a_{10} \leftarrow r_{10} \oplus (r_{22} \gg 25) \oplus (r_{23} \ll 7) \oplus (r_{20} \gg 2) \oplus (r_{21} \ll 30)$
7: $a_9 \leftarrow r_9 \oplus (r_{21} \gg 25) \oplus (r_{22} \ll 7) \oplus (r_{19} \gg 2) \oplus (r_{20} \ll 30)$
8: $a_8 \leftarrow r_8 \oplus (r_{20} \gg 25) \oplus (r_{21} \ll 7) \oplus (r_{18} \gg 2) \oplus (r_{19} \ll 30)$
9: $a_7 \leftarrow r_7 \oplus (r_{19} \gg 25) \oplus (r_{20} \ll 7) \oplus (r_{17} \gg 2) \oplus (r_{18} \ll 30)$
10: $a_6 \leftarrow r_6 \oplus (r_{18} \gg 25) \oplus (r_{19} \ll 7) \oplus (r_{16} \gg 2) \oplus (r_{17} \ll 30)$
11: $a_5 \leftarrow r_5 \oplus (r_{17} \gg 25) \oplus (r_{18} \ll 7) \oplus (a_{15} \gg 2) \oplus (r_{16} \ll 30)$
12: $a_4 \leftarrow r_4 \oplus (r_{16} \gg 25) \oplus (r_{17} \ll 7) \oplus (a_{14} \gg 2) \oplus (a_{15} \ll 30)$
13: $a_3 \leftarrow r_3 \oplus (a_{15} \gg 25) \oplus (r_{16} \ll 7) \oplus (a_{13} \gg 2) \oplus (a_{14} \ll 30)$
14: $a_2 \leftarrow r_2 \oplus (a_{14} \gg 25) \oplus (a_{15} \ll 7) \oplus ((a_{12} \wedge$ `0xFE000000`$) \gg 2) \oplus (a_{13} \ll 30)$
15: $a_1 \leftarrow r_1 \oplus (a_{13} \gg 25) \oplus (a_{14} \ll 7)$
16: $a_0 \leftarrow r_0 \oplus (a_{12} \gg 25) \oplus (a_{13} \ll 7)$
17: $a_{12} \leftarrow a_{12} \wedge$ `0x01FFFFFF`
18: **Retourner** $a_{12}, ..., a_0$

Algorithme A.5 Réduction par $x^{571} + x^{10} + x^5 + x^2 + 1$

Entrées Registres $r_{35}, .., r_0$ du polynôme devant être réduit

1: $a_{18} \leftarrow r_{18} \oplus (r_{35} \gg 17) \oplus (r_{35} \gg 22) \oplus (r_{35} \gg 25)$
2: $a_{17} \leftarrow r_{17} \oplus (r_{34} \gg 27) \oplus (r_{35} \ll 5) \oplus (r_{34} \gg 17) \oplus (r_{35} \ll 15) \oplus (r_{34} \gg 22) \oplus (r_{35} \ll 10) \oplus (r_{34} \gg 25) \oplus (r_{35} \ll 7)$
3: $a_{16} \leftarrow r_{16} \oplus (r_{33} \gg 27) \oplus (r_{34} \ll 5) \oplus (r_{33} \gg 17) \oplus (r_{34} \ll 15) \oplus (r_{33} \gg 22) \oplus (r_{34} \ll 10) \oplus (r_{33} \gg 25) \oplus (r_{34} \ll 7)$
4: $a_{15} \leftarrow r_{15} \oplus (r_{32} \gg 27) \oplus (r_{33} \ll 5) \oplus (r_{32} \gg 17) \oplus (r_{33} \ll 15) \oplus (r_{32} \gg 22) \oplus (r_{33} \ll 10) \oplus (r_{32} \gg 25) \oplus (r_{33} \ll 7)$
5: $a_{14} \leftarrow r_{14} \oplus (r_{31} \gg 27) \oplus (r_{32} \ll 5) \oplus (r_{31} \gg 17) \oplus (r_{32} \ll 15) \oplus (r_{31} \gg 22) \oplus (r_{32} \ll 10) \oplus (r_{31} \gg 25) \oplus (r_{32} \ll 7)$
6: $a_{13} \leftarrow r_{13} \oplus (r_{30} \gg 27) \oplus (r_{31} \ll 5) \oplus (r_{30} \gg 17) \oplus (r_{31} \ll 15) \oplus (r_{30} \gg 22) \oplus (r_{31} \ll 10) \oplus (r_{30} \gg 25) \oplus (r_{31} \ll 7)$
7: $a_{12} \leftarrow r_{12} \oplus (r_{29} \gg 27) \oplus (r_{30} \ll 5) \oplus (r_{29} \gg 17) \oplus (r_{30} \ll 15) \oplus (r_{29} \gg 22) \oplus (r_{30} \ll 10) \oplus (r_{29} \gg 25) \oplus (r_{30} \ll 7)$
8: $a_{11} \leftarrow r_{11} \oplus (r_{28} \gg 27) \oplus (r_{29} \ll 5) \oplus (r_{28} \gg 17) \oplus (r_{29} \ll 15) \oplus (r_{28} \gg 22) \oplus (r_{29} \ll 10) \oplus (r_{28} \gg 25) \oplus (r_{29} \ll 7)$
9: $a_{10} \leftarrow r_{10} \oplus (r_{27} \gg 27) \oplus (r_{28} \ll 5) \oplus (r_{27} \gg 17) \oplus (r_{28} \ll 15) \oplus (r_{27} \gg 22) \oplus (r_{28} \ll 10) \oplus (r_{27} \gg 25) \oplus (r_{28} \ll 7)$
10: $r_9 \leftarrow r_9 \oplus (r_{26} \gg 27) \oplus (r_{27} \ll 5) \oplus (r_{26} \gg 17) \oplus (r_{27} \ll 15) \oplus (r_{26} \gg 22) \oplus (r_{27} \ll 10) \oplus (r_{26} \gg 25) \oplus (r_{27} \ll 7)$
11: $a_8 \leftarrow r_8 \oplus (r_{25} \gg 27) \oplus (r_{26} \ll 5) \oplus (r_{25} \gg 17) \oplus (r_{26} \ll 15) \oplus (r_{25} \gg 22) \oplus (r_{26} \ll 10) \oplus (r_{25} \gg 25) \oplus (r_{26} \ll 7)$
12: $a_7 \leftarrow r_7 \oplus (r_{24} \gg 27) \oplus (r_{25} \ll 5) \oplus (r_{24} \gg 17) \oplus (r_{25} \ll 15) \oplus (r_{24} \gg 22) \oplus (r_{25} \ll 10) \oplus (r_{24} \gg 25) \oplus (r_{25} \ll 7)$
13: $a_6 \leftarrow r_6 \oplus (r_{23} \gg 27) \oplus (r_{24} \ll 5) \oplus (r_{23} \gg 17) \oplus (r_{24} \ll 15) \oplus (r_{23} \gg 22) \oplus (r_{24} \ll 10) \oplus (r_{23} \gg 25) \oplus (r_{24} \ll 7)$
14: $a_5 \leftarrow r_5 \oplus (r_{22} \gg 27) \oplus (r_{23} \ll 5) \oplus (r_{22} \gg 17) \oplus (r_{23} \ll 15) \oplus (r_{22} \gg 22) \oplus (r_{23} \ll 10) \oplus (r_{22} \gg 25) \oplus (r_{23} \ll 7)$
15: $a_4 \leftarrow r_4 \oplus (r_{21} \gg 27) \oplus (r_{22} \ll 5) \oplus (r_{21} \gg 17) \oplus (r_{22} \ll 15) \oplus (r_{21} \gg 22) \oplus (r_{22} \ll 10) \oplus (r_{21} \gg 25) \oplus (r_{22} \ll 7)$
16: $a_3 \leftarrow r_3 \oplus (r_{20} \gg 27) \oplus (r_{21} \ll 5) \oplus (r_{20} \gg 17) \oplus (r_{21} \ll 15) \oplus (r_{20} \gg 22) \oplus (r_{21} \ll 10) \oplus (r_{20} \gg 25) \oplus (r_{21} \ll 7)$
17: $a_2 \leftarrow r_2 \oplus (r_{19} \gg 27) \oplus (r_{20} \ll 5) \oplus (a_{19} \gg 17) \oplus (r_{20} \ll 15) \oplus (a_{19} \gg 22) \oplus (r_{20} \ll 10) \oplus (a_{19} \gg 25) \oplus (r_{20} \ll 7)$
18: $a_1 \leftarrow r_1 \oplus (a_{18} \gg 27) \oplus (r_{19} \ll 5) \oplus (a_{18} \gg 17) \oplus (a_{19} \ll 15) \oplus (a_{18} \gg 22) \oplus (a_{19} \ll 10) \oplus (a_{18} \gg 25) \oplus (a_{19} \ll 7)$
19: $A_{17} \leftarrow a_{17} \wedge \texttt{0xF8000000}$
20: $a_0 \leftarrow r_0 \oplus (a_{17} \gg 27) \oplus (a_{18} \ll 5) \oplus (A_{17} \gg 17) \oplus (a_{18} \ll 15) \oplus (A_{17} \gg 22) \oplus (a_{18} \ll 10) \oplus (A_{17} \gg 25) \oplus (a_{18} \ll 7)$
21: $r_{17} \leftarrow r_{17} \wedge \texttt{0x07FFFFFF}$
22: **Retourner** $a_{17}, ..., a_0$

A.2 Polynômes irréductibles de la section 4.3.1

Algorithme A.6 Réduction par $x^{223} + x^{159} + 1$

Entrées Registres $r_{13}, .., r_0$ du polynôme devant être réduit

1: $a_{11} \leftarrow r_{11} \oplus r_{13}$
2: $a_{10} \leftarrow r_{10} \oplus r_{12}$
3: $a_9 \leftarrow r_9 \oplus a_{11}$
4: $a_8 \leftarrow r_8 \oplus a_{10}$
5: $a_7 \leftarrow r_7 \oplus a_9$
6: $a_6 \leftarrow r_6 \oplus (r_{12} \gg 31) \oplus (r_{13} \ll 1) \oplus a_8$
7: $a_5 \leftarrow r_5 \oplus (a_{11} \gg 31) \oplus (r_{12} \ll 1) \oplus a_7$
8: $a_4 \leftarrow r_4 \oplus (a_{10} \gg 31) \oplus (a_{11} \ll 1) \oplus (a_6 \wedge \texttt{0x80000000})$
9: $a_3 \leftarrow r_3 \oplus (a_9 \gg 31) \oplus (a_{10} \ll 1)$
10: $a_2 \leftarrow r_2 \oplus (a_8 \gg 31) \oplus (a_9 \ll 1)$
11: $a_1 \leftarrow r_1 \oplus (a_7 \gg 31) \oplus (a_8 \ll 1)$
12: $a_0 \leftarrow r_0 \oplus (a_6 \gg 31) \oplus (a_7 \ll 1)$
13: $a_6 \leftarrow a_6 \wedge \texttt{0x7FFFFFFF}$
14: **Retourner** $a_6, ..., a_0$

Algorithme A.7 Réduction par $x^{257} + x^{65} + 1$

Entrées Registres $r_{16}, .., r_0$ du polynôme devant être réduit

1: $a_{10} \leftarrow r_{10} \oplus r_{16}$
2: $a_9 \leftarrow r_9 \oplus r_{15}$
3: $a_8 \leftarrow r_8 \oplus (r_{16} \gg 1) \oplus r_{14}$
4: $a_7 \leftarrow r_7 \oplus (r_{15} \gg 1) \oplus (r_{16} \ll 31) \oplus r_{13}$
5: $a_6 \leftarrow r_6 \oplus (r_{14} \gg 1) \oplus (r_{15} \ll 31) \oplus r_{12}$
6: $a_5 \leftarrow r_5 \oplus (r_{13} \gg 1) \oplus (r_{14} \ll 31) \oplus r_{11}$
7: $a_4 \leftarrow r_4 \oplus (r_{12} \gg 1) \oplus (r_{13} \ll 31) \oplus a_{10}$
8: $a_3 \leftarrow r_3 \oplus (r_{11} \gg 1) \oplus (r_{12} \ll 31) \oplus a_9$
9: $a_2 \leftarrow r_2 \oplus (a_{10} \gg 1) \oplus (r_{11} \ll 31) \oplus (a_8 \wedge \texttt{0xFFFFFFFE})$
10: $a_1 \leftarrow r_1 \oplus (a_9 \gg 1) \oplus (a_{10} \ll 31)$
11: $a_0 \leftarrow r_0 \oplus (a_8 \gg 1) \oplus (a_9 \ll 31)$
12: $a_8 \leftarrow a_8 \wedge \texttt{0x00000001}$
13: **Retourner** $a_8, ..., a_0$

Algorithme A.8 Réduction par $x^{313} + x^{121} + 1$

Entrées Registres $r_{19}, .., r_0$ du polynôme devant être réduit

1: $a_{13} \leftarrow r_{13} \oplus r_{19}$
2: $a_{12} \leftarrow r_{12} \oplus r_{18}$
3: $a_{11} \leftarrow r_{11} \oplus r_{17}$
4: $a_{10} \leftarrow r_{10} \oplus r_{16}$
5: $a_9 \leftarrow r_9 \oplus (r_{18} \gg 25) \oplus (r_{19} \ll 7) \oplus r_{15}$
6: $a_8 \leftarrow r_8 \oplus (r_{17} \gg 25) \oplus (r_{18} \ll 7) \oplus r_{14}$
7: $a_7 \leftarrow r_7 \oplus (r_{16} \gg 25) \oplus (r_{17} \ll 7) \oplus a_{13}$
8: $a_6 \leftarrow r_6 \oplus (r_{15} \gg 25) \oplus (r_{16} \ll 7) \oplus a_{12}$
9: $a_5 \leftarrow r_5 \oplus (r_{14} \gg 25) \oplus (r_{15} \ll 7) \oplus a_{11}$
10: $a_4 \leftarrow r_4 \oplus (a_{13} \gg 25) \oplus (r_{14} \ll 7) \oplus a_{10}$
11: $a_3 \leftarrow r_3 \oplus (a_{12} \gg 25) \oplus (a_{13} \ll 7) \oplus (a_9 \wedge \texttt{0xFE000000})$
12: $a_2 \leftarrow r_2 \oplus (a_{11} \gg 25) \oplus (a_{12} \ll 7)$
13: $a_1 \leftarrow r_1 \oplus (a_{10} \gg 25) \oplus (a_{11} \ll 7)$
14: $a_0 \leftarrow r_0 \oplus (a_9 \gg 25) \oplus (a_{10} \ll 7)$
15: $a_9 \leftarrow r_9 \wedge \texttt{0x01FFFFFF}$
16: **Retourner** $a_9, ..., a_0$

Algorithme A.9 Réduction par $x^{431} + x^{303} + x^{239} + x^{111} + 1$

Entrées Registres $r_{26}, .., r_0$ du polynôme devant être réduit

1: $a_{22} \leftarrow r_{22} \oplus r_{26}$
2: $a_{21} \leftarrow r_{21} \oplus r_{25}$
3: $a_{20} \leftarrow r_{20} \oplus r_{26} \oplus r_{24}$
4: $a_{19} \leftarrow r_{19} \oplus r_{25} \oplus r_{23}$
5: $a_{18} \leftarrow r_{18} \oplus r_{24} \oplus a_{22}$
6: $a_{17} \leftarrow r_{17} \oplus r_{23} \oplus a_{21}$
7: $a_{16} \leftarrow r_{16} \oplus r_{26} \oplus a_{22} \oplus a_{20}$
8: $a_{15} \leftarrow r_{15} \oplus r_{25} \oplus a_{21} \oplus a_{19}$
9: $a_{14} \leftarrow r_{14} \oplus r_{24} \oplus a_{20} \oplus a_{18}$
10: $a_{13} \leftarrow r_{13} \oplus (r_{26} \gg 15) \oplus r_{23} \oplus a_{19} \oplus a_{17}$
11: $a_{12} \leftarrow r_{12} \oplus (r_{25} \gg 15) \oplus (r_{26} \ll 17) \oplus a_{22} \oplus a_{18} \oplus a_{16}$
12: $a_{11} \leftarrow r_{11} \oplus (r_{24} \gg 15) \oplus (r_{25} \ll 17) \oplus a_{21} \oplus a_{17} \oplus a_{15}$
13: $a_{10} \leftarrow r_{10} \oplus (r_{23} \gg 15) \oplus (r_{24} \ll 17) \oplus a_{20} \oplus a_{16} \oplus a_{14}$
14: $a_9 \leftarrow r_9 \oplus (a_{22} \gg 15) \oplus (r_{23} \ll 17) \oplus a_{19} \oplus a_{15} \oplus (a_{13} \wedge \texttt{0xFFFF8000})$
15: $a_8 \leftarrow r_8 \oplus (a_{21} \gg 15) \oplus (a_{22} \ll 17) \oplus a_{18} \oplus a_{14}$
16: $a_7 \leftarrow r_7 \oplus (a_{20} \gg 15) \oplus (a_{21} \ll 17) \oplus a_{17} \oplus (a_{13} \wedge \texttt{0xFFFF8000})$
17: $a_6 \leftarrow r_6 \oplus (a_{19} \gg 15) \oplus (a_{20} \ll 17) \oplus a_{16}$
18: $a_5 \leftarrow r_5 \oplus (a_{18} \gg 15) \oplus (a_{19} \ll 17) \oplus a_{15}$
19: $a_4 \leftarrow r_4 \oplus (a_{17} \gg 15) \oplus (a_{18} \ll 17) \oplus a_{14}$
20: $a_3 \leftarrow r_3 \oplus (a_{16} \gg 15) \oplus (a_{17} \ll 17) \oplus (a_{13} \wedge \texttt{0xFFFF8000})$
21: $a_2 \leftarrow r_2 \oplus (a_{15} \gg 15) \oplus (a_{16} \ll 17)$
22: $a_1 \leftarrow r_1 \oplus (a_{14} \gg 15) \oplus (a_{15} \ll 17)$
23: $a_0 \leftarrow r_0 \oplus (a_{13} \gg 15) \oplus (a_{14} \ll 17)$
24: $a_{13} \leftarrow a_{13} \wedge \texttt{0x00007FFF}$
25: **Retourner** $a_{13}, ..., a_0$

Algorithme A.10 Réduction par $x^{479} + x^{255} + 1$

Entrées Registres $r_{29}, .., r_0$ du polynôme devant être réduit

1: $a_{22} \leftarrow r_{22} \oplus r_{29}$
2: $a_{21} \leftarrow r_{21} \oplus r_{28}$
3: $a_{20} \leftarrow r_{20} \oplus r_{27}$
4: $a_{19} \leftarrow r_{19} \oplus r_{26}$
5: $a_{18} \leftarrow r_{18} \oplus r_{25}$
6: $a_{17} \leftarrow r_{17} \oplus r_{24}$
7: $a_{16} \leftarrow r_{16} \oplus r_{23}$
8: $a_{15} \leftarrow r_{15} \oplus a_{22}$
9: $a_{14} \leftarrow r_{14} \oplus (r_{28} \gg 31) \oplus (r_{29} \ll 1) \oplus a_{21}$
10: $a_{13} \leftarrow r_{27} \oplus (r_{27} \gg 31) \oplus (r_{28} \ll 1) \oplus a_{20}$
11: $a_{12} \leftarrow r_{12} \oplus (r_{26} \gg 31) \oplus (r_{27} \ll 1) \oplus a_{19}$
12: $a_{11} \leftarrow r_{11} \oplus (r_{25} \gg 31) \oplus (r_{26} \ll 1) \oplus a_{18}$
13: $a_{10} \leftarrow r_{10} \oplus (r_{24} \gg 31) \oplus (r_{25} \ll 1) \oplus a_{17}$
14: $a_9 \leftarrow r_9 \oplus (r_{23} \gg 31) \oplus (r_{24} \ll 1) \oplus a_{16}$
15: $a_8 \leftarrow r_8 \oplus (a_{22} \gg 31) \oplus (r_{23} \ll 1) \oplus a_{15}$
16: $a_7 \leftarrow r_7 \oplus (a_{21} \gg 31) \oplus (a_{22} \ll 1) \oplus (a_{14} \wedge \texttt{0x80000000})$
17: $a_6 \leftarrow r_6 \oplus (a_{20} \gg 31) \oplus (a_{21} \ll 1)$
18: $a_5 \leftarrow r_5 \oplus (a_{19} \gg 31) \oplus (a_{20} \ll 1)$
19: $a_4 \leftarrow r_4 \oplus (a_{18} \gg 31) \oplus (a_{19} \ll 1)$
20: $a_3 \leftarrow r_3 \oplus (a_{17} \gg 31) \oplus (a_{18} \ll 1)$
21: $a_2 \leftarrow r_2 \oplus (a_{16} \gg 31) \oplus (a_{17} \ll 1)$
22: $a_1 \leftarrow r_1 \oplus (a_{15} \gg 31) \oplus (a_{16} \ll 1)$
23: $a_0 \leftarrow r_0 \oplus (a_{14} \gg 31) \oplus (a_{15} \ll 1)$
24: $a_{14} \leftarrow a_{14} \wedge \texttt{0x7FFFFFFF}$
25: **Retourner** $a_{14}, ..., a_0$

Algorithme A.11 Réduction par $x^{487} + x^{295} + x^{167} + x^{39} + 1$

Entrées Registres $r_{30}, .., r_0$ du polynôme devant être réduit

1: $a_{24} \leftarrow r_{24} \oplus r_{30}$
2: $a_{23} \leftarrow r_{23} \oplus r_{29}$
3: $a_{22} \leftarrow r_{22} \oplus r_{28}$
4: $a_{21} \leftarrow r_{21} \oplus r_{27}$
5: $a_{20} \leftarrow r_{20} \oplus r_{30} \oplus r_{26}$
6: $a_{19} \leftarrow r_{19} \oplus r_{29} \oplus r_{25}$
7: $a_{18} \leftarrow r_{18} \oplus r_{28} \oplus a_{24}$
8: $a_{17} \leftarrow r_{17} \oplus r_{27} \oplus a_{23}$
9: $a_{16} \leftarrow r_{16} \oplus r_{30} \oplus r_{26} \oplus a_{22}$
10: $a_{15} \leftarrow r_{15} \oplus (r_{30} \gg 7) \oplus r_{29} \oplus r_{25} \oplus a_{21}$
11: $a_{14} \leftarrow r_{14} \oplus (r_{29} \gg 7) \oplus (r_{30} \ll 25) \oplus r_{28} \oplus a_{24} \oplus a_{20}$
12: $a_{13} \leftarrow r_{27} \oplus (r_{28} \gg 7) \oplus (r_{29} \ll 25) \oplus r_{27} \oplus a_{23} \oplus a_{19}$
13: $a_{12} \leftarrow r_{12} \oplus (r_{27} \gg 7) \oplus (r_{28} \ll 25) \oplus r_{26} \oplus a_{22} \oplus a_{18}$
14: $a_{11} \leftarrow r_{11} \oplus (r_{26} \gg 7) \oplus (r_{27} \ll 25) \oplus r_{25} \oplus a_{21} \oplus a_{17}$
15: $a_{10} \leftarrow r_{10} \oplus (r_{25} \gg 7) \oplus (r_{26} \ll 25) \oplus a_{24} \oplus a_{20} \oplus a_{16}$
16: $a_9 \leftarrow r_9 \oplus (a_{24} \gg 7) \oplus (r_{25} \ll 25) \oplus a_{23} \oplus a_{19} \oplus (a_{15} \wedge \texttt{0xFFFFFF80})$
17: $a_8 \leftarrow r_8 \oplus (a_{23} \gg 7) \oplus (a_{24} \ll 25) \oplus a_{22} \oplus a_{18}$
18: $a_7 \leftarrow r_7 \oplus (a_{22} \gg 7) \oplus (a_{23} \ll 25) \oplus a_{21} \oplus a_{17}$
19: $a_6 \leftarrow r_6 \oplus (a_{21} \gg 7) \oplus (a_{22} \ll 25) \oplus a_{20} \oplus a_{16}$
20: $a_5 \leftarrow r_5 \oplus (a_{20} \gg 7) \oplus (a_{21} \ll 25) \oplus a_{19} \oplus (a_{15} \wedge \texttt{0xFFFFFF80})$
21: $a_4 \leftarrow r_4 \oplus (a_{19} \gg 7) \oplus (a_{20} \ll 25) \oplus a_{18}$
22: $a_3 \leftarrow r_3 \oplus (a_{18} \gg 7) \oplus (a_{19} \ll 25) \oplus a_{17}$
23: $a_2 \leftarrow r_2 \oplus (a_{17} \gg 7) \oplus (a_{18} \ll 25) \oplus a_{16}$
24: $a_1 \leftarrow r_1 \oplus (a_{16} \gg 7) \oplus (a_{17} \ll 25) \oplus (a_{15} \wedge \texttt{0xFFFFFF80})$
25: $a_0 \leftarrow r_0 \oplus (a_{15} \gg 7) \oplus (a_{16} \ll 25)$
26: $a_{15} \leftarrow a_{15} \wedge \texttt{0x0000007F}$
27: **Retourner** $a_{15}, ..., a_0$

Algorithme A.12 Réduction par $x^{521} + x^{489} + 1$

Entrées Registres $r_{32}, .., r_0$ du polynôme devant être réduit

1: $a_{31} \leftarrow r_{31} \oplus r_{32}$
2: $a_{30} \leftarrow r_{30} \oplus a_{31}$
3: $a_{29} \leftarrow r_{29} \oplus a_{30}$
4: $a_{28} \leftarrow r_{28} \oplus a_{29}$
5: $a_{27} \leftarrow r_{27} \oplus a_{28}$
6: $a_{26} \leftarrow r_{26} \oplus a_{27}$
7: $a_{25} \leftarrow r_{25} \oplus a_{26}$
8: $a_{24} \leftarrow r_{24} \oplus a_{25}$
9: $a_{23} \leftarrow r_{23} \oplus a_{24}$
10: $a_{22} \leftarrow r_{22} \oplus a_{23}$
11: $a_{21} \leftarrow r_{21} \oplus a_{22}$
12: $a_{20} \leftarrow r_{20} \oplus a_{21}$
13: $a_{19} \leftarrow r_{19} \oplus a_{20}$
14: $a_{18} \leftarrow r_{18} \oplus a_{19}$
15: $a_{17} \leftarrow r_{17} \oplus a_{18}$
16: $a_{16} \leftarrow r_{16} \oplus (r_{32} \gg 9) \oplus a_{17}$
17: $a_{15} \leftarrow r_{15} \oplus (a_{31} \gg 9) \oplus (r_{32} \ll 23) \oplus (a_{16} \wedge \texttt{0xFFFFFE00})$
18: $a_{14} \leftarrow r_{14} \oplus (a_{30} \gg 9) \oplus (a_{31} \ll 23)$
19: $a_{13} \leftarrow r_{13} \oplus (a_{29} \gg 9) \oplus (a_{30} \ll 23)$
20: $a_{12} \leftarrow r_{12} \oplus (a_{28} \gg 9) \oplus (a_{29} \ll 23)$
21: $a_{11} \leftarrow r_{11} \oplus (a_{27} \gg 9) \oplus (a_{28} \ll 23)$
22: $a_{10} \leftarrow r_{10} \oplus (a_{26} \gg 9) \oplus (a_{27} \ll 23)$
23: $r_9 \leftarrow r_9 \oplus (a_{25} \gg 9) \oplus (a_{26} \ll 23)$
24: $a_8 \leftarrow r_8 \oplus (a_{24} \gg 9) \oplus (a_{25} \ll 23)$
25: $a_7 \leftarrow r_7 \oplus (a_{23} \gg 9) \oplus (a_{24} \ll 23)$
26: $a_6 \leftarrow r_6 \oplus (a_{22} \gg 9) \oplus (a_{23} \ll 23)$
27: $a_5 \leftarrow r_5 \oplus (a_{21} \gg 9) \oplus (a_{22} \ll 23)$
28: $a_4 \leftarrow r_4 \oplus (a_{20} \gg 9) \oplus (a_{21} \ll 23)$
29: $a_3 \leftarrow r_3 \oplus (a_{19} \gg 9) \oplus (a_{20} \ll 23)$
30: $a_2 \leftarrow r_2 \oplus (a_{18} \gg 9) \oplus (a_{19} \ll 23)$
31: $a_1 \leftarrow r_1 \oplus (a_{17} \gg 9) \oplus (a_{18} \ll 23)$
32: $a_0 \leftarrow r_0 \oplus (a_{16} \gg 9) \oplus (a_{17} \ll 23)$
33: $r_{16} \leftarrow r_{16} \wedge \texttt{0x000001FF}$
34: **Retourner** $a_{16}, ..., a_0$

Algorithme A.13 Réduction par $x^{569} + x^{441} + x^{313} + x^{121} + 1$

Entrées Registres $r_{35}, .., r_0$ du polynôme devant être réduit

1: $a_{31} \leftarrow r_{31} \oplus r_{35}$
2: $a_{30} \leftarrow r_{30} \oplus r_{34}$
3: $a_{29} \leftarrow r_{29} \oplus r_{33}$
4: $a_{28} \leftarrow r_{28} \oplus r_{32}$
5: $a_{27} \leftarrow r_{27} \oplus r_{35} \oplus a_{31}$
6: $a_{26} \leftarrow r_{26} \oplus r_{34} \oplus a_{30}$
7: $a_{25} \leftarrow r_{25} \oplus r_{33} \oplus a_{29}$
8: $a_{24} \leftarrow r_{24} \oplus r_{32} \oplus a_{28}$
9: $a_{23} \leftarrow r_{23} \oplus a_{31} \oplus a_{27}$
10: $a_{22} \leftarrow r_{22} \oplus a_{30} \oplus a_{26}$
11: $a_{21} \leftarrow r_{21} \oplus r_{35} \oplus a_{29} \oplus a_{25}$
12: $a_{20} \leftarrow r_{20} \oplus r_{34} \oplus a_{28} \oplus a_{24}$
13: $a_{19} \leftarrow r_{19} \oplus r_{33} \oplus a_{27} \oplus a_{23}$
14: $a_{18} \leftarrow r_{18} \oplus r_{32} \oplus a_{26} \oplus a_{22}$
15: $a_{17} \leftarrow r_{17} \oplus (r_{34} \gg 25) \oplus (r_{35} \ll 7) \oplus a_{31} \oplus a_{25} \oplus a_{21}$
16: $a_{16} \leftarrow r_{16} \oplus (r_{33} \gg 25) \oplus (r_{34} \ll 7) \oplus a_{30} \oplus a_{24} \oplus a_{20}$
17: $a_{15} \leftarrow r_{15} \oplus (r_{32} \gg 25) \oplus (r_{33} \ll 7) \oplus a_{29} \oplus a_{23} \oplus a_{19}$
18: $a_{14} \leftarrow r_{14} \oplus (a_{31} \gg 25) \oplus (r_{32} \ll 7) \oplus a_{28} \oplus a_{22} \oplus a_{18}$
19: $a_{13} \leftarrow r_{13} \oplus (a_{30} \gg 25) \oplus (a_{31} \ll 7) \oplus a_{27} \oplus a_{21} \oplus (a_{17} \wedge \texttt{0xFE000000})$
20: $a_{12} \leftarrow r_{12} \oplus (a_{29} \gg 25) \oplus (a_{30} \ll 7) \oplus a_{26} \oplus a_{20}$
21: $a_{11} \leftarrow r_{11} \oplus (a_{28} \gg 25) \oplus (a_{29} \ll 7) \oplus a_{25} \oplus a_{19}$
22: $a_{10} \leftarrow r_{10} \oplus (a_{27} \gg 25) \oplus (a_{28} \ll 7) \oplus a_{24} \oplus a_{18}$
23: $r_9 \leftarrow r_9 \oplus (a_{26} \gg 25) \oplus (a_{27} \ll 7) \oplus a_{23} \oplus (a_{17} \wedge \texttt{0xFE000000})$
24: $a_8 \leftarrow r_8 \oplus (a_{25} \gg 25) \oplus (a_{26} \ll 7) \oplus a_{22}$
25: $a_7 \leftarrow r_7 \oplus (a_{24} \gg 25) \oplus (a_{25} \ll 7) \oplus a_{21}$
26: $a_6 \leftarrow r_6 \oplus (a_{23} \gg 25) \oplus (a_{24} \ll 7) \oplus a_{20}$
27: $a_5 \leftarrow r_5 \oplus (a_{22} \gg 25) \oplus (a_{23} \ll 7) \oplus a_{19}$
28: $a_4 \leftarrow r_4 \oplus (a_{21} \gg 25) \oplus (a_{22} \ll 7) \oplus a_{18}$
29: $a_3 \leftarrow r_3 \oplus (a_{20} \gg 25) \oplus (a_{21} \ll 7) \oplus (a_{17} \wedge \texttt{0xFE000000})$
30: $a_2 \leftarrow r_2 \oplus (a_{19} \gg 25) \oplus (a_{20} \ll 7)$
31: $a_1 \leftarrow r_1 \oplus (a_{18} \gg 25) \oplus (a_{19} \ll 7)$
32: $a_0 \leftarrow r_0 \oplus (a_{17} \gg 25) \oplus (a_{18} \ll 7)$
33: $a_{17} \leftarrow a_{17} \wedge \texttt{0x01FFFFFF}$
34: **Retourner** $a_{17}, ..., a_0$

Annexe B

Algorithmes d'inversion

Cette annexe présente les différents algorithmes d'inversion dans $\mathbb{F}_{2^n}$ utilisés pour notre implémentation des BEC. Les inverses dans $\mathbb{F}_{2^n}$ sont calculés grâce à la méthode d'Itoh-Tsujii [IT88]. Cette dernière utilise une propriété des éléments de $\mathbb{F}_{2^n}$ issue du Petit Théorème de Fermat :

$$\forall a \in \mathbb{F}_{2^n}, a^{2^n-2} = a^{-1}$$

La chaîne d'additions utilisée pour chaque inversion est donnée en commentaire au fil de l'algorithme. De plus chaque algorithme utilise un certain nombre de variables temporaires qui sont notées : $t_0, ..., t_6$.

Algorithme B.1 Inversion dans $\mathbb{F}_{2^{223}}$

Entrées $a \in \mathbb{F}_{2^{223}}$

1: $t_0 \leftarrow a^2$ /* 2^1 */
2: $t_0 \leftarrow t_0 \times a$ /* $2^2 - 1$ */
3: $t_1 \leftarrow t_0^2$ /* $2^3 - 2$ */
4: $t_1 \leftarrow t_1 \times a$ /* $2^3 - 1$ */
5: $t_2 \leftarrow t_1^2$ /* $2^4 - 2$ */
6: $t_2 \leftarrow t_2^2$ /* $2^5 - 2^2$ */
7: $t_2 \leftarrow t_2^2$ /* $2^6 - 2^3$ */
8: $t_1 \leftarrow t_2 \times t_1$ /* $2^6 - 1$ */
9: $t_2 \leftarrow t_1^2$ /* $2^7 - 2$ */
10: **Pour** $i = 7$ **à** 12 **Faire**
11: $t_2 \leftarrow t_2^2$
12: **Fin Pour** /* $2^{12} - 2^6$ */
13: $t_2 \leftarrow t_2 \times t_1$ /* $2^{12} - 1$ */
14: $t_3 \leftarrow t_2^2$ /* $2^{12} - 2$ */
15: **Pour** $i = 13$ **à** 24 **Faire**
16: $t_3 \leftarrow t_3^2$
17: **Fin Pour** /* $2^{24} - 2^{12}$ */
18: $t_2 \leftarrow t_3 \times t_2$ /* $2^{24} - 1$ */
19: $t_3 \leftarrow t_2^2$ /* $2^{25} - 2$ */
20: **Pour** $i = 25$ **à** 48 **Faire**
21: $t_3 \leftarrow t_3^2$
22: **Fin Pour** /* $2^{48} - 2^{24}$ */
23: $t_3 \leftarrow t_3 \times t_2$ /* $2^{48} - 1$ */
24: $t_4 \leftarrow t_3^2$ /* $2^{49} - 2$ */
25: **Pour** $i = 49$ **à** 96 **Faire**
26: $t_4 \leftarrow t_4^2$
27: **Fin Pour** /* $2^{96} - 2^{48}$ */
28: $t_3 \leftarrow t_4 \times t_3$ /* $2^{96} - 1$ */
29: $t_4 \leftarrow t_3^2$ /* $2^{97} - 2$ */
30: **Pour** $i = 97$ **à** 192 **Faire**
31: $t_4 \leftarrow t_4^2$
32: **Fin Pour** /* $2^{192} - 2^{96}$ */
33: $t_3 \leftarrow t_4 \times t_3$ /* $2^{192} - 1$ */
34: $t_4 \leftarrow t_3^2$ /* $2^{193} - 2$ */
35: **Pour** $i = 193$ **à** 216 **Faire**
36: $t_4 \leftarrow t_4^2$
37: **Fin Pour** /* $2^{216} - 2^{24}$ */
38: $t_3 \leftarrow t_4 \times t_2$ /* $2^{216} - 1$ */
39: $t_4 \leftarrow t_3^2$ /* $2^{217} - 2$ */
40: **Pour** $i = 217$ **à** 222 **Faire**
41: $t_4 \leftarrow t_4^2$
42: **Fin Pour** /* $2^{222} - 2^6$ */
43: $t_3 \leftarrow t_4 \times t_1$ /* $2^{222} - 1$ */
44: $b \leftarrow t_3^2$ /* $2^{223} - 2$ */
45: **Retourner** $b = a^{-1}$

Algorithme B.2 Inversion dans $\mathbb{F}_{2^{257}}$

Entrées $a \in \mathbb{F}_{2^{257}}$

1: $t_0 \leftarrow a^2$ /* 2^1 */
2: $t_0 \leftarrow t_0 \times a$ /* $2^2 - 1$ */
3: $t_1 \leftarrow t_0^2$ /* $2^3 - 2$ */
4: $t_1 \leftarrow t_1^2$ /* $2^4 - 2^2$ */
5: $t_1 \leftarrow t_1 \times t_0$ /* $2^4 - 1$ */
6: $t_1 \leftarrow t_0^2$ /* $2^5 - 2$ */
7: **Pour** $i = 5$ **à** 8 **Faire**
8: $\quad t_1 \leftarrow t_1^2$
9: **Fin Pour** /* $2^8 - 2^4$ */
10: $t_0 \leftarrow t_1 \times t_0$ /* $2^8 - 1$ */
11: $t_1 \leftarrow t_0^2$ /* $2^9 - 2$ */
12: **Pour** $i = 8$ **à** 16 **Faire**
13: $\quad t_1 \leftarrow t_1^2$
14: **Fin Pour** /* $2^{16} - 2^8$ */
15: $t_0 \leftarrow t_1 \times t_0$ /* $2^{16} - 1$ */
16: $t_1 \leftarrow t_0^2$ /* $2^{17} - 2$ */
17: **Pour** $i = 17$ **à** 32 **Faire**
18: $\quad t_1 \leftarrow t_1^2$
19: **Fin Pour** /* $2^{32} - 2^{16}$ */
20: $t_0 \leftarrow t_1 \times t_0$ /* $2^{32} - 1$ */
21: $t_1 \leftarrow t_0^2$ /* $2^{33} - 2$ */
22: **Pour** $i = 33$ **à** 64 **Faire**
23: $\quad t_1 \leftarrow t_1^2$
24: **Fin Pour** /* $2^{64} - 2^{32}$ */
25: $t_0 \leftarrow t_1 \times t_0$ /* $2^{64} - 1$ */
26: $t_1 \leftarrow t_0^2$ /* $2^{65} - 2$ */
27: **Pour** $i = 65$ **à** 128 **Faire**
28: $\quad t_1 \leftarrow t_1^2$
29: **Fin Pour** /* $2^{128} - 2^{64}$ */
30: $t_0 \leftarrow t_1 \times t_0$ /* $2^{128} - 1$ */
31: $t_1 \leftarrow t_0^2$ /* $2^{129} - 2$ */
32: **Pour** $i = 129$ **à** 256 **Faire**
33: $\quad t_1 \leftarrow t_1^2$
34: **Fin Pour** /* $2^{256} - 2^{128}$ */
35: $t_0 \leftarrow t_1 \times t_0$ /* $2^{256} - 1$ */
36: $b \leftarrow t_0^2$ /* $2^{257} - 2$ */
37: **Retourner** $b = a^{-1}$

Algorithme B.3 Inversion dans $\mathbb{F}_{2^{313}}$

Entrées $a \in \mathbb{F}_{2^{313}}$

1: $t_0 \leftarrow a^2$ /* 2^1 */
2: $t_0 \leftarrow t_0 \times a$ /* $2^2 - 1$ */
3: $t_1 \leftarrow t_0^2$ /* $2^3 - 2$ */
4: $t_0 \leftarrow t_1 \times a$ /* $2^3 - 1$ */
5: $t_1 \leftarrow t_0^2$ /* $2^4 - 2$ */
6: **Pour** $i = 4$ **à** 6 **Faire**
7: $t_1 \leftarrow t_1^2$
8: **Fin Pour** /* $2^6 - 2^3$ */
9: $t_0 \leftarrow t_1 \times t_0$ /* $2^6 - 1$ */
10: $t_1 \leftarrow t_0^2$ /* $2^7 - 2$ */
11: **Pour** $i = 7$ **à** 12 **Faire**
12: $t_1 \leftarrow t_1^2$
13: **Fin Pour** /* $2^{12} - 2^6$ */
14: $t_0 \leftarrow t_1 \times t_0$ /* $2^{12} - 1$ */
15: $t_1 \leftarrow t_0^2$ /* $2^{13} - 2$ */
16: **Pour** $i = 13$ **à** 24 **Faire**
17: $t_1 \leftarrow t_1^2$
18: **Fin Pour** /* $2^{24} - 2^{12}$ */
19: $t_0 \leftarrow t_1 \times t_0$ /* $2^{24} - 1$ */
20: $t_1 \leftarrow t_0^2$ /* $2^{25} - 2$ */
21: **Pour** $i = 25$ **à** 48 **Faire**
22: $t_1 \leftarrow t_1^2$
23: **Fin Pour** /* $2^{48} - 2^{24}$ */
24: $t_1 \leftarrow t_1 \times t_0$ /* $2^{48} - 1$ */
25: $t_2 \leftarrow t_1^2$ /* $2^{49} - 2$ */
26: **Pour** $i = 49$ **à** 96 **Faire**
27: $t_2 \leftarrow t_2^2$
28: **Fin Pour** /* $2^{96} - 2^{48}$ */
29: $t_1 \leftarrow t_2 \times t_1$ /* $2^{96} - 1$ */
30: $t_2 \leftarrow t_1^2$ /* $2^{97} - 2$ */
31: **Pour** $i = 97$ **à** 192 **Faire**
32: $t_2 \leftarrow t_2^2$
33: **Fin Pour** /* $2^{192} - 2^{96}$ */
34: $t_2 \leftarrow t_2 \times t_1$ /* $2^{192} - 1$ */
35: $t_3 \leftarrow t_2^2$ /* $2^{193} - 2$ */
36: **Pour** $i = 193$ **à** 288 **Faire**
37: $t_3 \leftarrow t_3^2$
38: **Fin Pour** /* $2^{288} - 2^{96}$ */
39: $t_2 \leftarrow t_3 \times t_1$ /* $2^{288} - 1$ */
40: $t_3 \leftarrow t_2^2$ /* $2^{289} - 2$ */
41: **Pour** $i = 289$ **à** 312 **Faire**
42: $t_3 \leftarrow t_3^2$
43: **Fin Pour** /* $2^{312} - 2^{24}$ */
44: $t_2 \leftarrow t_3 \times t_1$ /* $2^{312} - 1$ */
45: $b \leftarrow t_2^2$ /* $2^{313} - 2$ */
46: **Retourner** $b = a^{-1}$

Algorithme B.4 Inversion dans $\mathbb{F}_{2^{431}}$. Partie 1.

Entrées $a \in \mathbb{F}_{2^{431}}$

1: $t_0 \leftarrow a^2$ /* 2^1 */
2: $t_0 \leftarrow t_0 \times a$ /* $2^2 - 1$ */
3: $t_1 \leftarrow t_0^2$ /* $2^3 - 2$ */
4: $t_1 \leftarrow t_1^2$ /* $2^4 - 2^2$ */
5: $t_1 \leftarrow t_1 \times t_0$ /* $2^4 - 1$ */
6: $t_2 \leftarrow t_1^2$ /* $2^5 - 2$ */
7: **Pour** $i = 5$ **à** 8 **Faire**
8: $\quad t_2 \leftarrow t_2^2$
9: **Fin Pour** /* $2^8 - 2^4$ */
10: $t_2 \leftarrow t_2 \times t_1$ /* $2^8 - 1$ */
11: $t_3 \leftarrow t_2^2$ /* $2^9 - 2$ */
12: **Pour** $i = 9$ **à** 16 **Faire**
13: $\quad t_3 \leftarrow t_3^2$
14: **Fin Pour** /* $2^{16} - 2^8$ */
15: $t_3 \leftarrow t_3 \times t_2$ /* $2^{16} - 1$ */
16: $t_4 \leftarrow t_3^2$ /* $2^{17} - 2$ */
17: **Pour** $i = 17$ **à** 32 **Faire**
18: $\quad t_4 \leftarrow t_4^2$
19: **Fin Pour** /* $2^{32} - 2^{16}$ */
20: $t_3 \leftarrow t_4 \times t_3$ /* $2^{32} - 1$ */
21: $t_4 \leftarrow t_3^2$ /* $2^{33} - 2$ */
22: **Pour** $i = 33$ **à** 64 **Faire**
23: $\quad t_4 \leftarrow t_4^2$
24: **Fin Pour** /* $2^{64} - 2^{33}$ */
25: $t_4 \leftarrow t_4 \times t_3$ /* $2^{64} - 1$ */
26: $t_5 \leftarrow t_4^2$ /* $2^{65} - 2$ */
27: **Pour** $i = 65$ **à** 128 **Faire**
28: $\quad t_5 \leftarrow t_5^2$
29: **Fin Pour** /* $2^{128} - 2^{64}$ */
30: $t_4 \leftarrow t_5 \times t_4$ /* $2^{128} - 1$ */
31: $t_5 \leftarrow t_4^2$ /* $2^{129} - 2$ */
32: **Pour** $i = 129$ **à** 256 **Faire**
33: $\quad t_5 \leftarrow t_5^2$
34: **Fin Pour** /* $2^{256} - 2^{128}$ */
35: $t_5 \leftarrow t_5 \times t_4$ /* $2^{256} - 1$ */
36: $t_5 \leftarrow t_5^2$ /* $2^{257} - 2$ */
37: **Pour** $i = 257$ **à** 384 **Faire**
38: $\quad t_5 \leftarrow t_5^2$
39: **Fin Pour** /* $2^{256} - 2^{128}$ */
40: $t_5 \leftarrow t_5 \times t_4$ /* $2^{384} - 1$ */
41: $t_5 \leftarrow t_5^2$ /* $2^{385} - 2$ */
42: **Pour** $i = 385$ **à** 416 **Faire**
43: $\quad t_5 \leftarrow t_5^2$
44: **Fin Pour** /* $2^{416} - 2^{32}$ */
45: $t_5 \leftarrow t_5 \times t_3$ /* $2^{416} - 1$ */

Algorithme B.5 Inversion dans $\mathbb{F}_{2^{431}}$. Partie 2.

46: $t_5 \leftarrow t_5^2$ /* $2^{417} - 2$ */
47: **Pour** $i = 417$ **à** 424 **Faire**
48: $\quad t_5 \leftarrow t_5^2$
49: **Fin Pour** /* $2^{424} - 2^8$ */
50: $t_5 \leftarrow t_5 \times t_2$ /* $2^{416} - 1$ */
51: $t_5 \leftarrow t_5^2$ /* $2^{425} - 2$ */
52: **Pour** $i = 425$ **à** 428 **Faire**
53: $\quad t_5 \leftarrow t_5^2$
54: **Fin Pour** /* $2^{428} - 2^4$ */
55: $t_5 \leftarrow t_5 \times t_1$ /* $2^{428} - 1$ */
56: $t_5 \leftarrow t_5^2$ /* $2^{428} - 2$ */
57: $t_5 \leftarrow t_5^2$ /* $2^{430} - 2^2$ */
58: $t_5 \leftarrow t_5 \times t_0$ /* $2^{430} - 1$ */
59: $b \leftarrow t_5^2$ /* $2^{431} - 2$ */
60: **Retourner** $b = a^{-1}$

Algorithme B.6 Inversion dans $\mathbb{F}_{2^{479}}$. Partie 1.

Entrées $a \in \mathbb{F}_{2^{479}}$
1: $t_0 \leftarrow a^2$ /* 2^1 */
2: $t_0 \leftarrow t_0 \times a$ /* $2^2 - 1$ */
3: $t_1 \leftarrow t_0^2$ /* $2^3 - 2$ */
4: $t_1 \leftarrow t_1^2$ /* $2^4 - 2^2$ */
5: $t_1 \leftarrow t_1 \times t_0$ /* $2^4 - 1$ */
6: $t_2 \leftarrow t_1^2$ /* $2^5 - 2$ */
7: **Pour** $i = 5$ **à** 8 **Faire**
8: $\quad t_2 \leftarrow t_2^2$
9: **Fin Pour** /* $2^8 - 2^4$ */
10: $t_2 \leftarrow t_2 \times t_1$ /* $2^8 - 1$ */
11: $t_3 \leftarrow t_2^2$ /* $2^9 - 2$ */
12: **Pour** $i = 9$ **à** 16 **Faire**
13: $\quad t_3 \leftarrow t_3^2$
14: **Fin Pour** /* $2^{16} - 2^8$ */
15: $t_3 \leftarrow t_3 \times t_2$ /* $2^{16} - 1$ */
16: $t_4 \leftarrow t_3^2$ /* $2^{17} - 2$ */
17: **Pour** $i = 17$ **à** 32 **Faire**
18: $\quad t_4 \leftarrow t_4^2$
19: **Fin Pour** /* $2^{32} - 2^{16}$ */
20: $t_4 \leftarrow t_4 \times t_3$ /* $2^{32} - 1$ */
21: $t_5 \leftarrow t_4^2$ /* $2^{33} - 2$ */
22: **Pour** $i = 33$ **à** 64 **Faire**
23: $\quad t_5 \leftarrow t_5^2$
24: **Fin Pour** /* $2^{64} - 2^{32}$ */
25: $t_4 \leftarrow t_5 \times t_4$ /* $2^{64} - 1$ */

Algorithme B.7 Inversion dans $\mathbb{F}_{2^{479}}$. Partie 2.

26: $t_5 \leftarrow t_4^2$ /* $2^{65} - 2$ */
27: **Pour** $i = 65$ **à** 128 **Faire**
28: $\quad t_5 \leftarrow t_5^2$
29: **Fin Pour** /* $2^{128} - 2^{64}$ */
30: $t_5 \leftarrow t_5 \times t_4$ /* $2^{128} - 1$ */
31: $t_6 \leftarrow t_5^2$ /* $2^{129} - 2$ */
32: **Pour** $i = 129$ **à** 256 **Faire**
33: $\quad t_6 \leftarrow t_6^2$
34: **Fin Pour** /* $2^{256} - 2^{128}$ */
35: $t_6 \leftarrow t_6 \times t_5$ /* $2^{256} - 1$ */
36: $t_6 \leftarrow t_6^2$ /* $2^{257} - 2$ */
37: **Pour** $i = 257$ **à** 384 **Faire**
38: $\quad t_6 \leftarrow t_6^2$
39: **Fin Pour** /* $2^{384} - 2^{128}$ */
40: $t_6 \leftarrow t_6 \times t_5$ /* $2^{384} - 1$ */
41: $t_6 \leftarrow t_6^2$ /* $2^{385} - 2$ */
42: **Pour** $i = 385$ **à** 448 **Faire**
43: $\quad t_6 \leftarrow t_6^2$
44: **Fin Pour** /* $2^{448} - 2^{64}$ */
45: $t_6 \leftarrow t_6 \times t_4$ /* $2^{448} - 1$ */
46: $t_6 \leftarrow t_6^2$ /* $2^{449} - 2$ */
47: **Pour** $i = 449$ **à** 464 **Faire**
48: $\quad t_6 \leftarrow t_6^2$
49: **Fin Pour** /* $2^{464} - 2^{16}$ */
50: $t_6 \leftarrow t_6 \times t_3$ /* $2^{464} - 1$ */
51: $t_6 \leftarrow t_6^2$ /* $2^{465} - 2$ */
52: **Pour** $i = 465$ **à** 472 **Faire**
53: $\quad t_6 \leftarrow t_6^2$
54: **Fin Pour** /* $2^{472} - 2^8$ */
55: $t_6 \leftarrow t_6 \times t_2$ /* $2^{472} - 1$ */
56: $t_6 \leftarrow t_6^2$ /* $2^{473} - 2$ */
57: **Pour** $i = 473$ **à** 476 **Faire**
58: $\quad t_6 \leftarrow t_6^2$
59: **Fin Pour** /* $2^{476} - 2^4$ */
60: $t_6 \leftarrow t_6 \times t_1$ /* $2^{476} - 1$ */
61: $t_6 \leftarrow t_6^2$ /* $2^{477} - 2$ */
62: $t_6 \leftarrow t_6^2$ /* $2^{478} - 2^2$ */
63: $t_6 \leftarrow t_6 \times t_0$ /* $2^{478} - 1$ */
64: $b \leftarrow t_6^2$ /* $2^{479} - 2$ */
65: **Retourner** $b = a^{-1}$

Algorithme B.8 Inversion dans $\mathbb{F}_{2^{487}}$

Entrées $a \in \mathbb{F}_{2^{487}}$

1: $t_0 \leftarrow a^2$ /* 2^1 */
2: $t_0 \leftarrow t_0 \times a$ /* $2^2 - 1$ */
3: $t_1 \leftarrow t_0^2$ /* $2^3 - 2$ */
4: $t_1 \leftarrow t_1^2$ /* $2^4 - 2^2$ */
5: $t_1 \leftarrow t_1 \times t_0$ /* $2^4 - 1$ */
6: $t_2 \leftarrow t_1^2$ /* $2^5 - 2$ */
7: **Pour** $i = 5$ **à** 8 **Faire**
8: $\quad t_2 \leftarrow t_2^2$
9: **Fin Pour** /* $2^8 - 2^4$ */
10: $t_2 \leftarrow t_2 \times t_1$ /* $2^8 - 1$ */
11: $t_3 \leftarrow t_2^2$ /* $2^9 - 2$ */
12: **Pour** $i = 9$ **à** 16 **Faire**
13: $\quad t_3 \leftarrow t_3^2$
14: **Fin Pour** /* $2^{16} - 2^8$ */
15: $t_2 \leftarrow t_3 \times t_2$ /* $2^{16} - 1$ */
16: $t_3 \leftarrow t_3^2$ /* $2^{17} - 2$ */
17: **Pour** $i = 17$ **à** 32 **Faire**
18: $\quad t_3 \leftarrow t_3^2$
19: **Fin Pour** /* $2^{32} - 2^{16}$ */
20: $t_3 \leftarrow t_3 \times t_2$ /* $2^{32} - 1$ */
21: $t_3 \leftarrow t_2^2$ /* $2^{33} - 2$ */
22: **Pour** $i = 33$ **à** 64 **Faire**
23: $\quad t_3 \leftarrow t_3^2$
24: **Fin Pour** /* $2^{64} - 2^{32}$ */
25: $t_3 \leftarrow t_3 \times t_3$ /* $2^{64} - 1$ */
26: $t_4 \leftarrow t_3^2$ /* $2^{65} - 2$ */
27: **Pour** $i = 65$ **à** 128 **Faire**
28: $\quad t_4 \leftarrow t_4^2$
29: **Fin Pour** /* $2^{128} - 2^{64}$ */
30: $t_4 \leftarrow t_4 \times t_3$ /* $2^{128} - 1$ */
31: $t_5 \leftarrow t_4^2$ /* $2^{129} - 2$ */
32: **Pour** $i = 129$ **à** 256 **Faire**
33: $\quad t_5 \leftarrow t_5^2$
34: **Fin Pour** /* $2^{256} - 2^{128}$ */
35: $t_5 \leftarrow t_5 \times t_4$ /* $2^{256} - 1$ */
36: $t_5 \leftarrow t_5^2$ /* $2^{257} - 2$ */
37: **Pour** $i = 257$ **à** 384 **Faire**
38: $\quad t_5 \leftarrow t_5^2$
39: **Fin Pour** /* $2^{384} - 2^{128}$ */
40: $t_5 \leftarrow t_5 \times t_4$ /* $2^{384} - 1$ */
41: $t_5 \leftarrow t_5^2$ /* $2^{385} - 2$ */
42: **Pour** $i = 385$ **à** 448 **Faire**
43: $\quad t_5 \leftarrow t_5^2$
44: **Fin Pour** /* $2^{448} - 2^{64}$ */
45: $t_5 \leftarrow t_5 \times t_3$ /* $2^{448} - 1$ */

Algorithme B.9 Inversion dans $\mathbb{F}_{2^{487}}$. Partie 2.

46: $t_5 \leftarrow t_5^2$ /* $2^{449} - 2$ */
47: **Pour** $i = 449$ **à** 480 **Faire**
48: $\quad t_5 \leftarrow t_5^2$
49: **Fin Pour** /* $2^{480} - 2^{32}$ */
50: $t_5 \leftarrow t_5 \times t_2$ /* $2^{480} - 1$ */
51: $t_5 \leftarrow t_5^2$ /* $2^{481} - 2$ */
52: **Pour** $i = 481$ **à** 484 **Faire**
53: $\quad t_5 \leftarrow t_5^2$
54: **Fin Pour** /* $2^{484} - 2^4$ */
55: $t_5 \leftarrow t_5 \times t_1$ /* $2^{484} - 1$ */
56: $t_5 \leftarrow t_5^2$ /* $2^{485} - 2$ */
57: $t_5 \leftarrow t_5^2$ /* $2^{486} - 2^2$ */
58: $t_5 \leftarrow t_5 \times t_0$ /* $2^{486} - 1$ */
59: $b \leftarrow t_5^2$ /* $2^{487} - 2$ */
60: **Retourner** $b = a^{-1}$

Algorithme B.10 Inversion dans $\mathbb{F}_{2^{521}}$

Entrées $a \in \mathbb{F}_{2^{521}}$

1: $t_0 \leftarrow a^2$ /* 2^1 */
2: $t_0 \leftarrow t_0 \times a$ /* $2^2 - 1$ */
3: $t_1 \leftarrow t_0^2$ /* $2^3 - 2$ */
4: $t_1 \leftarrow t_1^2$ /* $2^4 - 2^2$ */
5: $t_0 \leftarrow t_1 \times t_0$ /* $2^4 - 1$ */
6: $t_1 \leftarrow t_0^2$ /* $2^5 - 2$ */
7: **Pour** $i = 5$ **à** 8 **Faire**
8: $\quad t_1 \leftarrow t_1^2$
9: **Fin Pour** /* $2^8 - 2^4$ */
10: $t_0 \leftarrow t_0 \times t_1$ /* $2^8 - 1$ */
11: $t_1 \leftarrow t_0^2$ /* $2^9 - 2$ */
12: **Pour** $i = 9$ **à** 16 **Faire**
13: $\quad t_1 \leftarrow t_1^2$
14: **Fin Pour** /* $2^{16} - 2^8$ */
15: $t_1 \leftarrow t_1 \times t_0$ /* $2^{16} - 1$ */
16: $t_2 \leftarrow t_1^2$ /* $2^{17} - 2$ */
17: **Pour** $i = 17$ **à** 32 **Faire**
18: $\quad t_2 \leftarrow t_2^2$
19: **Fin Pour** /* $2^{32} - 2^{16}$ */
20: $t_1 \leftarrow t_2 \times t_1$ /* $2^{32} - 1$ */
21: $t_2 \leftarrow t_1^2$ /* $2^{33} - 2$ */
22: **Pour** $i = 33$ **à** 64 **Faire**
23: $\quad t_2 \leftarrow t_2^2$
24: **Fin Pour** /* $2^{64} - 2^{32}$ */
25: $t_1 \leftarrow t_2 \times t_1$ /* $2^{64} - 1$ */
26: $t_2 \leftarrow t_1^2$ /* $2^{65} - 2$ */
27: **Pour** $i = 65$ **à** 128 **Faire**
28: $\quad t_2 \leftarrow t_2^2$
29: **Fin Pour** /* $2^{128} - 2^{64}$ */
30: $t_1 \leftarrow t_2 \times t_1$ /* $2^{128} - 1$ */
31: $t_2 \leftarrow t_1^2$ /* $2^{129} - 2$ */
32: **Pour** $i = 129$ **à** 256 **Faire**
33: $\quad t_2 \leftarrow t_2^2$
34: **Fin Pour** /* $2^{256} - 2^{128}$ */
35: $t_1 \leftarrow t_2 \times t_1$ /* $2^{256} - 1$ */
36: $t_2 \leftarrow t_1^2$ /* $2^{257} - 2$ */
37: **Pour** $i = 257$ **à** 512 **Faire**
38: $\quad t_2 \leftarrow t_2^2$
39: **Fin Pour** /* $2^{512} - 2^{256}$ */
40: $t_1 \leftarrow t_2 \times t_1$ /* $2^{512} - 1$ */
41: $t_2 \leftarrow t_1^2$ /* $2^{513} - 2$ */
42: **Pour** $i = 513$ **à** 520 **Faire**
43: $\quad t_2 \leftarrow t_2^2$
44: **Fin Pour** /* $2^{520} - 2^8$ */
45: $t_1 \leftarrow t_2 \times t_0$ /* $2^{520} - 1$ */
46: $b \leftarrow t_1^2$ /* $2^{521} - 2$ */
47: **Retourner** $b = a^{-1}$

Algorithme B.11 Inversion dans $\mathbb{F}_{2^{569}}$

Entrées $a \in \mathbb{F}_{2^{569}}$

1: $t_0 \leftarrow a^2$ /* 2^1 */
2: $t_0 \leftarrow t_0 \times a$ /* $2^2 - 1$ */
3: $t_1 \leftarrow t_0^2$ /* $2^3 - 2$ */
4: $t_1 \leftarrow t_1^2$ /* $2^4 - 2^2$ */
5: $t_0 \leftarrow t_1 \times t_0$ /* $2^4 - 1$ */
6: $t_1 \leftarrow t_0^2$ /* $2^5 - 2$ */
7: **Pour** $i = 5$ **à** 8 **Faire**
8: $\quad t_1 \leftarrow t_1^2$
9: **Fin Pour** /* $2^8 - 2^4$ */
10: $t_0 \leftarrow t_0 \times t_1$ /* $2^8 - 1$ */
11: $t_1 \leftarrow t_0^2$ /* $2^9 - 2$ */
12: **Pour** $i = 9$ **à** 16 **Faire**
13: $\quad t_1 \leftarrow t_1^2$
14: **Fin Pour** /* $2^{16} - 2^8$ */
15: $t_1 \leftarrow t_1 \times t_0$ /* $2^{16} - 1$ */
16: $t_2 \leftarrow t_1^2$ /* $2^{17} - 2$ */
17: **Pour** $i = 17$ **à** 32 **Faire**
18: $\quad t_2 \leftarrow t_2^2$
19: **Fin Pour** /* $2^{32} - 2^{16}$ */
20: $t_2 \leftarrow t_2 \times t_1$ /* $2^{32} - 1$ */
21: $t_3 \leftarrow t_2^2$ /* $2^{33} - 2$ */
22: **Pour** $i = 33$ **à** 64 **Faire**
23: $\quad t_3 \leftarrow t_3^2$
24: **Fin Pour** /* $2^{64} - 2^{32}$ */
25: $t_3 \leftarrow t_3 \times t_2$ /* $2^{64} - 1$ */
26: $t_4 \leftarrow t_3^2$ /* $2^{65} - 2$ */
27: **Pour** $i = 65$ **à** 128 **Faire**
28: $\quad t_4 \leftarrow t_4^2$
29: **Fin Pour** /* $2^{128} - 2^{64}$ */
30: $t_3 \leftarrow t_4 \times t_3$ /* $2^{128} - 1$ */
31: $t_4 \leftarrow t_3^2$ /* $2^{129} - 2$ */
32: **Pour** $i = 129$ **à** 256 **Faire**
33: $\quad t_4 \leftarrow t_4^2$
34: **Fin Pour** /* $2^{256} - 2^{128}$ */
35: $t_3 \leftarrow t_4 \times t_3$ /* $2^{256} - 1$ */
36: $t_4 \leftarrow t_3^2$ /* $2^{257} - 2$ */
37: **Pour** $i = 257$ **à** 512 **Faire**
38: $\quad t_4 \leftarrow t_4^2$
39: **Fin Pour** /* $2^{512} - 2^{256}$ */
40: $t_3 \leftarrow t_4 \times t_3$ /* $2^{512} - 1$ */

Algorithme B.12 Inversion dans $\mathbb{F}_{2^{569}}$. Partie 2.

41: $t_4 \leftarrow t_3^2$ /* $2^{513} - 2$ */
42: **Pour** $i = 513$ **à** 544 **Faire**
43: $\quad t_4 \leftarrow t_4^2$
44: **Fin Pour** /* $2^{544} - 2^{32}$ */
45: $t_3 \leftarrow t_4 \times t_2$ /* $2^{544} - 1$ */
46: $t_4 \leftarrow t_3^2$ /* $2^{545} - 2$ */
47: **Pour** $i = 545$ **à** 560 **Faire**
48: $\quad t_4 \leftarrow t_4^2$
49: **Fin Pour** /* $2^{560} - 2^{16}$ */
50: $t_3 \leftarrow t_4 \times t_1$ /* $2^{560} - 1$ */
51: $t_4 \leftarrow t_3^2$ /* $2^{561} - 2$ */
52: **Pour** $i = 561$ **à** 568 **Faire**
53: $\quad t_4 \leftarrow t_4^2$
54: **Fin Pour** /* $2^{568} - 2^8$ */
55: $t_3 \leftarrow t_4 \times t_1$ /* $2^{568} - 1$ */
56: $b \leftarrow t_1^2$ /* $2^{569} - 2$ */
57: **Retourner** $b = a^{-1}$

Annexe C

Nouvel ensemble de courbes binaires d'Edwards

Ce nouvel ensemble de courbes binaires d'Edwards est donné selon la nomenclature suivante :

- m : le degré de l'extension de corps.
- f : le polynôme irréductible définissant l'extension de corps.
- d : le paramètre d définissant la BEC.
- a : le coefficient a de la BEC sous forme de Weierstrass courte.
- b : le coefficient b de la BEC sous forme de Weierstrass courte.
- G_x : la coordonnée x du point générateur.
- G_y : la coordonnée y du point générateur.
- $G_{1/w}$: la coordonnée $w(G)^{-1}$ du point générateur.
- G_{Wx} : la coordonnée x du point générateur sous forme de Weierstrass.
- G_{Wy} : la coordonnée y du point générateur sous forme de Weierstrass.
- n : le cardinal de la courbe.
- p : l'ordre du point générateur.
- h : le cofacteur.

$m =$ 223

$f =$ $x^{223} + x^{159} + 1$

$d =$ $t^{64} + t^{36} + t^5 + 1$

$a =$ 1 00000000 00000101 00000010 00000420

$b =$ 1 00000003 00000000 00000002 00000006 00000102 00000003

$G_x =$ 205bfedd 71b0b0fd feb3345a f71cc721 790e83c4 b88094e9 a63f6d43

$G_y =$ 205bfedd f1b0b0fd 7eb3345a f71cc721 790e83c4 b88094e9 a63f6d43

$G_{1/w} =$ 1 00000001

$G_{Wx} =$ 9cfeebd fd48636d 380b581f 30d1e365 1da70b3d daf9b960 d73b0dca

$G_{Wy} =$ 27cf176a ed7af61b 699461bb a31f279e 152e208f 29308e52 e5d21f8b

$n =$ 13479973333575319897333507543509817898877363662863184062623242975476

$p =$ 33699933333938299743333768858774544747193409157157960156558107438 69

$h =$ 4

$m =$ 257
$f =$ $x^{257} + x^{65} + 1$
$d =$ $t^{65} + t^{31} + t^{14} + 1$
$a =$ 4 00000000 00000002 40000000 90004000
$b =$ 1000000 00000000 00000000 00000100 00010000 00000000 00000000 00000041
$G_x =$ 1 6b46e24a a4b12ab2 289fcd34 17615387 810f083f 43419d8c ae38ad9a c640d960
$G_y =$ 1 6b46e24a a4b12aba 289fcd34 17615383 810f083f 43419d8e ae38ad9a c640d968
$G_{1/w} =$ 1 00000000 00000000 00000000 00000000 00000000 00000000
$G_{Wx} =$ ffa37ca8 84a96447 546394f4 7489f1cd 0c1426ce e7f12f5b 5e448c93 d053c6f8
$G_{Wy} =$ e8fe8598 c155ab7f 1884aeb4 6ff3713d 4e5a4b22 9416d230 6cdf68bf db0703f4
$n =$ 231584178474632390847141970017375815706332616967362709021140632923291797618908
$p =$ 57896044618658097711785492504343953926583154241840677255285158230822949404727
$h =$ 4

$m =$ 313
$f =$ $x^{313} + x^{121} + 1$
$d =$ $t^{38} + t^{33} + t^{28} + 1$
$a =$ 1004 01000042 10000000
$b =$ 10000 00000100 00000001 00000000 00000000 00000000 00000000 00000000 00000000 00000001
$G_x =$ 15c67e3 024c7c27 466e72a3 391256e9 a729fc15 8092053d 89087c0f 38408b21 4b0ade57 363ea938
$G_y =$ 15c67e3 024c7c27 446e72a3 391256e9 a529fc15 8092053d 8b087c0f 38408b21 4b0ade57 363ea938
$G_{1/w} =$ 1 00000000 00000001
$G_{Wx} =$ 1de91b7 1a6213ec e5d54374 426b2130 9ecb988d 9e24ed7e 8e678976 7612dee3 efbf26d4 6667825e
$G_{Wy} =$ 163387d 236bc500 a3fbd65b 0edabcfb 2d3c86f5 0947f5c0 80e22336 d47e67d1 236fea37 63f3577e
$n =$ 166873987181321100187111070794496258953336290808161456226545492179886000188954062243097663372 12
$p =$ 41718496795330275046777767698624064738334072702040364056636373044971500047238515560774415843 03
$h =$ 4

$m =$ 431
$f =$ $x^{431} + x^{303} + x^{239} + x^{111} + 1$
$d =$ $t^{83} + t^{66} + t^{17} + 1$
$a =$ 40 00000010 00000000 00080004 00000004 00020000
$b =$ 1010000 00000000 01010000 00000000 01000000 00000200 01010000
00000000 01000100 00000202 00000000 00000200 00000001
$G_x =$ 4e17 65c1f2f6 140db17d 5ef2f14c 59a38a93 e5b65ba9 acca547b
f2cc34f3 d55bd85c cf4daeaf 7ca1beca a8ee877b 01f8d8ac ae12b210
$G_y =$ 4e17 65c1f2f6 140d317d 5ef2f14c 59a30a93 e5b65ba9 acca547b
f2cc34f3 d55b585c cf4daeaf 7ca13eca a8ee877b 01f8d8ac ae12b210
$G_{1/w} =$ 1 00000000 00000001
$G_{Wx} =$ 5a49 3ad63647 7019431f e09a9104 109949b7 dd371177 74f64988
4ae17680 605404f3 a5cba956 1cea5ac2 fe230062 35caa6ac a6423cf8
$G_{Wy} =$ 700b 7089da12 d2aae147 a7e26a75 4eb98ac0 c8de8fb3 86d425f1
7bbd3917 8d4f902d e4711444 e02fbeae f16b2256 27e582f3 ed81d15a
$n =$ 554533938824162971915682836828616740687287415075163315034095916131188222853620417205051641025857549800325003969881916422248862069 2
$p =$ 138633484706040742978920709207154185171821853768790828758523979032797055713405104301262910256464387450081250992470479105562215517 3
$h =$ 4

$m =$ 479
$f =$ $x^{479} + x^{255} + 1$
$d =$ $t^{73} + t^{29} + t^3 + 1$
$a =$ 40000 00000000 00000200 04000000 20000048
$b =$ 100 00000000 00000000 00000000 00000100 00000000 00000000
00000000 00000200 00000000 00000000 01000001
$G_x =$ 7bdd9f19 e11e888e 80d7c093 092d208b 4fe996e8 fcbdffa2 8cc90173
ece2c436 73f1372e 975ba9dc d3a06332 abf15dbe 9b679f6c 63e30b88
4ab93272
$G_y =$ 3bdd9f19 a11e888e 40d7c093 492d208b 8fe996e8 bcbdffa2 4cc90173
ece2c436 f3f1372e 175ba9dc 53a06332 2bf15dbe 1b679f6c e3e30b88
cab93272
$G_{1/w} =$ 1 00000000 00000001
$G_{Wx} =$ 7eed3566 e13a0e74 db25e511 2bfc6a32 f06666dc 3c903ac5 2b844204
08fb51c3 8d17db15 31cec46f 11c2a862 8acff220 721dd408 9923eeb8
6dd567d0
$G_{Wy} =$ 24e5bc7a 0e745503 7215bcb3 c5109db2 d8a230ec 593f4994 c7572645
68e68b13 c1436774 5220265e 97d22bbf a046b8049 17aea89b b7da1e7
a1e1631d
$n =$ 156087427515799611569079861489658315287429907133248557542957847981268586941544801971795445881886763046934698032411395977889616430979594599455835 6
$p =$ 390218568789499028922699653724145788218574767833121393857394619953171467353862004929488614704716907617336745081028489944724041077448986498639589
$h =$ 4

$m =$ 487

$f =$ $x^{487} + x^{295} + x^{167} + x^{39} + 1$

$d =$ $t^{69} + t^{33} + t^{15} + 1$

$a =$ 400 00000000 00000024 00000002 40008000

$b =$ 100 00000000 00000000 00000100 00000100 00000000 00000000
00000000 01000100 00000002 00000000 00000001

$G_x =$ 33 9b843c53 c409543f 396d39e5 7efde813 f06e3099 735004b9
99b15776 a75a4c3a 22dcaf1e 91e261fe 479b89a6 4d651039 28195d72
7bd3d157 735b2071

$G_y =$ 33 9b843c53 c40954bf 396d39e5 7efde893 f06e3099 73500439
99b15776 a75a4c3a 22dcaf1e 91e261fe 479b89a6 4d6510b9 28195d72
7bd3d1d7 735b2071

$G_{1/w} =$ 1 00000000 00000001

$G_{Wx} =$ 3e 06f269fa 101a7419 1851936f e1e56013 07d3ccf2 82133d7b
a7bc16eb 2c88cd07 097c135f f369cddb 0375230b 291c7e7a f8f4962d
6a9ad18d 4458ca54

$G_{Wy} =$ 31 ff52175c da7cc5ed 982283f5 5cb239c5 475e90af 85922767
0c9dc63c b00cf27b 294d781c dfc6b453 7dfb0af9 c7231a3d 7d0088c8
506e54d4 a45a7f0a

$n =$ 39958381444044700561684444541352528713582056226111630730997209083204758260181862138226693854521544466825345498675533707766156971976964541332597784 4

$p =$ 99895953610111751404211111353381321783955140565279076827493022708011895650454655345566734636303861167063363746688834269415392429942411353331494461

$h =$ 4

$m =$ 521

$f =$ $x^{521} + x^{489} + 1$

$d =$ $t^{66} + t^{29} + t^{28} + 1$

$a =$ 10 00000000 00000004 05000000 30000000

$b =$ 10000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000101 00000000 00000000 00000000 00000000 00000000
00000000 00000081

$G_x =$ 16b 369b497b 805e6199 a342909a a4608cdc ecb10e09 88ba73eb
1f118603 9c8b1f6d 2a9db39b 1302d29d 9d449b9a a459cc5d 6bbb4e33
a1eb8fcc 056ce724 cde5aaa8

$G_y =$ 16b 369b4b7b 805e6199 a342909a a4608cdc ecb10e09 88ba73eb
1f118603 9c8b1f6d 2a9db39b 1302d29d 9d449b9a a459cc5d 6bbb4e33
a1eb8fcc 056ce724 cde5aaa8

$G_{1/w} =$ 1 00000001

$G_{Wx} =$ 5e 28e2104e 3a1cfbbc f0852a88 489b969d b4461052 2cc589ae
6a5f0308 204bba01 98e5230c 6951caff 23dd3a9d 36b2e2b0 3e6b3a0b
4285ffb8 25db0dd1 7290fd64

$G_{Wy} =$ 1fe 5a7ef516 a46af41e 85ed35a7 9b30d82d d78b399a 7f96624a
7e36e30e 7016a933 d91e4351 ce5671bd 0ee8e8e8 6b8f286f 3cb6169f
6f191599 0434c0d9 ecc8dfcc

$n =$ 686479766013060971498190079908139321726943530014330540939446345918554318339765708943950328983162243962653943419778613112216264068985797800513224032860278220 4

$p =$ 171619941503265242874547519977034830431735882503582635234861586479638579584941427235987582245790560990663485854944653278054066017246449450128306008215069555 1

$h =$ 4

$m =$ 569

$f =$ $x^{569} + x^{441} + x^{313} + x^{121} + 1$

$d =$ $t^{56} + t^{45} + t^{41} + 1$

$a =$ 10000 04040000 01002200 00000000

$b =$ 1 00000000 00000000 00000100 00000100 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000001

$G_x =$ 195b22b 2864ee08 dd456bab 1a95cdd8 c7e3fd33 0fddf630 f9c3bb5c
33f062b3 41c919c6 bb4cbf1d 4335a344 ed023b31 9585ea0e 16f03453
cc5ba9a8 6a4b28b1 6e1c72ad 75f1141f

$G_y =$ 195b22b 2864ee08 df456bab 1a95cdd8 c5e3fd33 0fddf630 f9c3bb5c
33f062b3 41c919c6 bb4cbf1d 4135a344 ed023b31 9785ea0e 16f03453
ce5ba9a8 6a4b28b1 6e1c72ad 75f1141f

$G_{1/w} =$ 1 00000000 00000001

$G_{Wx} =$ 563c09 2af2a539 855667eb 3dd9ff07 1406267f 89154f79 260a1a76
5411b025 afe2fb1f e8b10dee a8b95ff7 0f3470d1 9e4e1f14 4c4da268
2fb5b923 97e843b8 621b7e7e 76bcea12

$G_{Wy} =$ 1e0812b d37eab10 4731760f 67015c45 c9322467 fa186977 2a41585c
ee455c39 7a5bb1ad 136c8c8b ce83915c 2d9b0bed dbaf2593 65151ba1
61750704 113558ee 4826bc8a 3417fd93

$n =$ 193226876150862917234767594546599367214946366485321749932861762572575957114478021226803243121352042256909941058335965554184749327731790508484922465034002522088006719930908

$p =$ 48306719037715729308691898636649841803736591621330437483215440643143989278619505306700810780338010564227485264583991388546187331932947627121230616258500630522001679982732 7

$h =$ 4

Annexe D

Détails sur les critères de sécurité

D.1 BEC223

$$\mathbb{F}_{2^{223}}[t] = \frac{\mathbb{F}_2[X]}{(X^{223} + X^{159} + 1)}$$
$$d = t^{64} + t^{36} + t^5 + 1$$

— 223 is prime.

— $|E| = 4p_1 = 2^2 \times (2^{221} - 640514697862898224455517859462717)$

— $|E^{tw}| = 2p_2 = 2 \times (2^{222} - 1281029395725796448911035718925433)$

— $\mathrm{Ord}(1/d^8) = 2^{223} - 1$

— $\Delta_E = -1 \times 7 \times 431 \times 4314911 \times 1126746703 \times 309404525859287$
$\times 104345227684848889894980821992990 49$

— $\mathrm{Ord}_{\mathbb{F}_{p_1}}(2^{223}) = (p_1 - 1)/3$

— $\mathrm{Ord}_{\mathbb{F}_{p_2}}(2^{223}) = (p_2 - 1)/2$

D.2 BEC257

$$\mathbb{F}_{2^{257}}[t] = \frac{\mathbb{F}_2[X]}{(X^{257} + X^{65} + 1)}$$
$$d = t^{65} + t^{31} + t^{14} + 1$$

— 257 is prime.

— $|E| = 4p_1 = 2^2 \times (2^{255} - 5183809097960476444363377313361541 5241)$

— $|E^{tw}| = 2p_2 = 2 \times (2^{256} + 1036761819592095288872675462672308304 83)$

— $\mathrm{Ord}(1/d^8) = 2^{257} - 1$

— $\Delta_E = -1 \times 5 \times 101 \times 4481 \times 916310512011856696478796469734 53$

— $\mathrm{Ord}_{\mathbb{F}_{p_1}}(2^{223}) = (p_1 - 1)/2$

— $\mathrm{Ord}_{\mathbb{F}_{p_2}}(2^{223}) = p_2 - 1$

D.3 BEC313

$$\mathbb{F}_{2^{313}}[t] = \frac{\mathbb{F}_2[X]}{(X^{313} + X^{121} + 1)}$$
$$d = t^{38} + t^{33} + t^{28} + 1$$

— 313 is prime.

— $|E| = 4p_1 = 2^2 \times (2^{311} - 2380103563917833578062289719146201818615624$
$1745)$

— $|E^{tw}| = 2p_2 = 2 \times (2^{312} + 4760207127835667156124579438292403637231248$
$3491)$

— $\mathrm{Ord}(1/d^8) = 2^{313} - 1$

— $\Delta_E = -1 \times 31 \times 1671713 \times 12880999 \times 2828226393551610769063$
$\times 30554946934858664591129288801673358518014327708172493826537$

— $\mathrm{Ord}_{\mathbb{F}_{p_1}}(2^{313}) = (p_1 - 1)/2$

— $\mathrm{Ord}_{\mathbb{F}_{p_2}}(2^{313}) = p_2 - 1$

D.4 BEC431

$$\mathbb{F}_{2^{431}}[t] = \frac{\mathbb{F}_2[X]}{(X^{431} + X^{303} + X^{239} + X^{111} + 1)}$$
$$d = t^{83} + t^{66} + t^{17} + 1$$

— 431 is prime.

— $|E| = 4p_1 = 2^2 \times (2^{429} + 2065990323123823149264189951257408545211690444400300 5141135076261)$

— $|E^{tw}| = 2p_2 = 2 \times (2^{430} - 4131980646247646298528379902514817090423380888800601 0282270152521)$

— $\mathrm{Ord}(1/d^8) = 2^{431} - 1$

— $\Delta_E = -1 \times 23 \times 761 \times 6422498333397737 \times 1108155675780127087$
$\times 12323928288128339636438796910313802399759008692$
$0099532058848225292873725497120370713605562799$

— $\mathrm{Ord}_{\mathbb{F}_{p_1}}(2^{431}) = p_1 - 1$

— $\mathrm{Ord}_{\mathbb{F}_{p_2}}(2^{431}) = (p_2 - 1)/2$

D.5 BEC479, BEC487, BEC521 and BEC569

La vérification de tous les critères de sécurité est difficile. En effet, pour chacun des critères nous avons à factoriser un grand nombre, *e.g.* pour calculer l'ordre du j-invariant et de 2^n, nous devons factoriser $2^n - 1$, $p_1 - 1$ et $p_2 - 1$. De plus, le critère sur le discriminant Δ_E est, explicitement, une factorisation d'un grand nombre. Cependant, ces critères ne sont pas primaires. En effet, l'impacte de ces spécifications sur l'ECDLP n'est pas clair ou très restreint. LEs principaux critères sont ceux de la primalité de l'extension du corps binaire et la décomposition du cardinal de la courbe et de sa tordue en $4p$, avec p premier.

D.6 BEC479

$$\mathbb{F}_{2^{479}}[t] = \frac{\mathbb{F}_2[X]}{(X^{479} + X^{255} + 1)}$$
$$d = t^{73} + t^{29} + t^3 + 1$$

— 479 is prime

— $|E| = 4p_1 = 2^2 \times (2^{477} + 139130241445035182178041122235837869085266208279304411753795083535331 7)$

— $|E^{tw}| = 2p_2 = 2 \times (2^{478} - 278260482890070364356082244471675738170532416558608823507590167070663 3)$

D.7 BEC487

$$\mathbb{F}_{2^{487}}[t] = \frac{\mathbb{F}_2[X]}{(X^{487} + X^{295} + X^{167} + X^{39} + 1)}$$
$$d = t^{69} + t^{33} + t^{15} + 1$$

— 487 is prime

— $|E| = 4p_1 = 2^2 \times (2^{485} + 822215550171693933827211808994890195080382233676105720820518622353020882 9)$

— $|E^{tw}| = 2p_2 = 2 \times (2^{486} - 164443110034338786765442361798978039016076446735221144164103724470604176 57)$

D.8 BEC521

$$\mathbb{F}_{2^{521}}[t] = \frac{\mathbb{F}_2[X]}{(X^{521} + X^{489} + 1)}$$
$$d = t^{66} + t^{29} + t^{28} + 1$$

— 521 is prime

— $|E| = 4p_1 = 2^2 \times (2^{519} + 259329235912292541971162310780701576318271260163172535333548139553009371931263)$

— $|E^{tw}| = 2p_2 = 2 \times (2^{520} - 518658471824585083942324621561403152636542520326345070667096279106018743862525)$

D.9 BEC569

$$\mathbb{F}_{2^{569}}[t] = \frac{\mathbb{F}_2[X]}{(X^{569} + X^{441} + X^{313} + X^{121} + 1)}$$
$$d = t^{56} + t^{45} + t^{41} + 1$$

— 569 is prime

— $|E| = 4p_1 = 2^2 \times (2^{567} - 161130193602166030330993580978582617087615133$
$008971857555574215983787883671058)$

— $|E^{tw}| = 2p_2 = 2 \times (2^{568} + 322260387204332060661987161957165234175230266$
$017943715111148431967575767342116040512 03)$

Annexe E

Courbes binaires d'Edwards générées

E.1 $\mathbb{F}_{2^{223}}$

$$
\begin{aligned}
d &= t^{155} + t^{111} + 1 \\
|E| &= 13479973333575319897333507543509819749158693850776165028497683523796 \\
|E^{tw}| &= 13479973333575319897333507543509810924478450571764407452605926725422
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{44} + t^{35} + t^{26} + 1 \\
|E| &= 13479973333575319897333507543509816530278689606961238374321955643092 \\
|E^{tw}| &= 13479973333575319897333507543509814143358454815579334106781654606126
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{53} + t^{23} + t^{7} + 1 \\
|E| &= 13479973333575319897333507543509810121567870467157732590923977333204 \\
|E^{tw}| &= 13479973333575319897333507543509820552069273955382839890179632916014
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{54} + t^{42} + t^{5} + 1 \\
|E| &= 13479973333575319897333507543509818796860131293223914962192880564052 \\
|E^{tw}| &= 13479973333575319897333507543509811876777013129316657518910729685166
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{55} + t^{46} + t^{26} + 1 \\
|E| &= 13479973333575319897333507543509817938875988644895454756237508818004 \\
|E^{tw}| &= 13479973333575319897333507543509812734761155777645117724866101431214
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{57} + t^{44} + t^{2} + 1 \\
|E| &= 13479973333575319897333507543509809360552125166769231720403748790356 \\
|E^{tw}| &= 13479973333575319897333507543509821313085019255771340760699861458862
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{58} + t^{48} + t^{13} + 1 \\
|E| &= 1347997333357531989733350754350981500963337399910176999880257\\
&\quad 4262804 \\
|E^{tw}| &= 1347997333357531989733350754350981566400377042343880248230103\\
&\quad 5986414
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{64} + t^{36} + t^5 + 1 \\
|E| &= 1347997333357531989733350754350981789887736366286318406262324\\
&\quad 2975476 \\
|E^{tw}| &= 1347997333357531989733350754350981277475978075967738841848036\\
&\quad 7273742
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{64} + t^{55} + t^{46} + 1 \\
|E| &= 1347997333357531989733350754350982022723945420554483790586265\\
&\quad 2108244 \\
|E^{tw}| &= 1347997333357531989733350754350981044639769021699573457524095\\
&\quad 8140974
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{65} + t^{37} + t^{35} + 1 \\
|E| &= 1347997333357531989733350754350981479450291587349555853466954\\
&\quad 2443444 \\
|E^{tw}| &= 1347997333357531989733350754350981587913422854904501394643406\\
&\quad 7805774
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{65} + t^{61} + t^{52} + 1 \\
|E| &= 1347997333357531989733350754350981784482333780996085990330229\\
&\quad 6925236 \\
|E^{tw}| &= 1347997333357531989733350754350981282881380661257971257780131\\
&\quad 3323982
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{66} + t^{44} + t^{19} + 1 \\
|E| &= 1347997333357531989733350754350981794306700543404979932766278\\
&\quad 7905684 \\
|E^{tw}| &= 1347997333357531989733350754350981273057013898849077315344082\\
&\quad 2343534
\end{aligned}
$$

E.2 $\mathbb{F}_{2^{257}}$

$$
\begin{aligned}
d &= t^{41} + t^{29} + t^{10} + 1 \\
|E| &= 2315841784746323908471419700173758157067737370174016352628894\\
&\quad 83902151088501148 \\
|E^{tw}| &= 2315841784746323908471419700173758157063062016451606208949408\\
&\quad 52129501430058598
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{56} + t^{46} + t^{45} + 1 \\
|E| &= 2315841784746323908471419700173758157062852610420168714334878\\
&\quad 69204397597479452 \\
|E^{tw}| &= 2315841784746323908471419700173758157067946776205453847243424\\
&\quad 66827254921080294
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{65} + t^{31} + t^{14} + 1 \\
|E| &= 2315841784746323908471419700173758157063326169673627090211406\\
&\quad 32923291797618908 \\
|E^{tw}| &= 2315841784746323908471419700173758157067473216951995471366897\\
&\quad 03108360720940838
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{66} + t^{47} + t^{29} + 1 \\
|E| &= 231584178474632390847141970017375815705888786078697671446106955659703225464572 \\
|E^{tw}| &= 231584178474632390847141970017375815707191152583864584711723380371949293095174
\end{aligned}
$$

E.3 $\mathbb{F}_{2^{313}}$

$$
\begin{aligned}
d &= t^{38} + t^{33} + t^{28} + 1 \\
|E| &= 166873987181321100187111070794496258953336290808161456226545492179886000188954062243097663372 12 \\
|E^{tw}| &= 166873987181321100187111070794496258953336290810065539077679759042335831964271023697990162711 74 \\
d &= t^{53} + t^{49} + t^{17} + 1 \\
|E| &= 166873987181321100187111070794496258953336290809267337665573007392078310398934533967243728481 88 \\
|E^{tw}| &= 166873987181321100187111070794496258953336290808959657638652243830143521754290551973844097601 98 \\
d &= t^{60} + t^{33} + t^{20} + 1 \\
|E| &= 166873987181321100187111070794496258953336290809003138126087402449603758795030293020359665795 48 \\
|E^{tw}| &= 166873987181321100187111070794496258953336290809223857178137848772618073358194792920728160288 38 \\
d &= t^{60} + t^{33} + t^{20} + 1 \\
|E| &= 166873987181321100187111070794496258953336290809003138126087402449603758795030293020359665795 48 \\
|E^{tw}| &= 166873987181321100187111070794496258953336290809223857178137848772618073358194792920728160288 38 \\
d &= t^{76} + t^{66} + t^{56} + 1 \\
|E| &= 166873987181321100187111070794496258953336290808161456226545492179886000188954062243097663372 12 \\
|E^{tw}| &= 166873987181321100187111070794496258953336290810065539077679759042335831964271023697990162711 74 \\
d &= t^{87} + t^{58} + t^{42} + 1 \\
|E| &= 166873987181321100187111070794496258953336290808659423008727767728935906480327628728922802 40188 \\
|E^{tw}| &= 166873987181321100187111070794496258953336290809567572295497483493285925672897457212165018681 98 \\
d &= t^{151} + t^{117} + t^{34} + 1 \\
|E| &= 166873987181321100187111070794496258953336290809698141264558695887280957483615074141367436302 68 \\
|E^{tw}| &= 166873987181321100187111070794496258953336290808528854039666555334940874669610011799720389781 18 \\
d &= t^{196} + t^{137} + t^{59} + 1 \\
|E| &= 166873987181321100187111070794496258953336290809353297361277236173593960903084941612073645932 44 \\
|E^{tw}| &= 166873987181321100187111070794496258953336290808873697942948015048627871250140144329014180151 42
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{268} + t^{199} + t^{69} + 1 \\
|E| &= 1668739871813211001871110707944962589533362908072699315843381 \\
& 9312350843110793321750722399763004 \\
|E^{tw}| &= 1668739871813211001871110707944962589533362908109570637198870 \\
& 5809871340104529186843386382845382
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{302} + t^{234} + t^{68} + 1 \\
|E| &= 1668739871813211001871110707944962589533362908096981412645586 \\
& 9588728095748361507414136743630268 \\
|E^{tw}| &= 1668739871813211001871110707944962589533362908085288540396665 \\
& 5533494087466961001179972038978118
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{35} + t^{24} + t^{23} + t^{12} + 1 \\
|E| &= 1668739871813211001871110707944962589533362908070638615711831 \\
& 4337623713626113549760757974821692 \\
|E^{tw}| &= 1668739871813211001871110707944962589533362908111631337330421 \\
& 0784598469589208958833350807786694
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{37} + t^{27} + t^{24} + t^{17} + 1 \\
|E| &= 1668739871813211001871110707944962589533362908086890321737498 \\
& 5690002825142345949443144587086908 \\
|E^{tw}| &= 1668739871813211001871110707944962589533362908095379631304753 \\
& 9432219358072976559150964195521478
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{264} + t^{229} + t^{227} + t^{143} + 1 \\
|E| &= 1668739871813211001871110707944962589533362908079531785703240 \\
& 8491816720622524016103424180036604 \\
|E^{tw}| &= 1668739871813211001871110707944962589533362908102738167339011 \\
& 6630405462592798492490684602571782
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{274} + t^{200} + t^{118} + t^{79} + 1 \\
|E| &= 1668739871813211001871110707944962589533362908093532973612772 \\
& 3617359396090308494161207364593244 \\
|E^{tw}| &= 1668739871813211001871110707944962589533362908088736979429480 \\
& 1504862787125014014432901418015142
\end{aligned}
$$

E.4 $\mathbb{F}_{2^{431}}$

$$
\begin{aligned}
d &= t^{81} + t^{66} + t + 1 \\
|E| &= 5545339388241629719156828368286167406872874150751633150340959 \\
& 1612475777683135028401309655794910946027673101871452241766769 \\
& 38291412 \\
|E^{tw}| &= 5545339388241629719156828368286167406872874150751633150340959 \\
& 1612109074629089996520289320449254637096222546567003901106389 \\
& 58339886
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{83} + t^{66} + t^{17} + 1 \\
|E| &= 5545339388241629719156828368286167406872874150751633150340959 \\
& 1613118822285362041720505164102585754980032500396988191642224 \\
& 88620692 \\
|E^{tw}| &= 5545339388241629719156828368286167406872874150751633150340959 \\
& 1611466030026862983201093812141579828143863148041467951230934 \\
& 08010606
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{109} + t^6 + t^4 + 1 \\
|E| &= 554533938824162971915682836828616740687287415075163315034095916130337092806146971602038044953833877734193566615352677244483537547 6 \\
|E^{tw}| &= 554533938824162971915682836828616740687287415075163315034095916115511430316103277613951717487821953504762917769208751487106125582 2
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{112} + t^{48} + t^{21} + 1 \\
|E| &= 554533938824162971915682836828616740687287415075163315034095916110451164310799855379520610103512388056476818478075591818468717877 2 \\
|E^{tw}| &= 554533938824162971915682836828616740687287415075163315034095916135397358811450393836469152338143443182479665906485836913120945252 6
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{42} + t^{24} + t^3 + t + 1 \\
|E| &= 554533938824162971915682836828616740687287415075163315034095916116688841041277751286888333041679135935846746256 8419478975289296532 \\
|E^{tw}| &= 554533938824162971915682836828616740687287415075163315034095916129159682080972497929101429399976695303109738128719480834060733476 6
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{42} + t^{41} + t^{28} + t^6 + 1 \\
|E| &= 554533938824162971915682836828616740687287415075163315034095916108796606670151812312841618861919214976819259469164450675018519176 4 \\
|E^{tw}| &= 554533938824162971915682836828616740687287415075163315034095916137051916452098436903148143579736616262137224915396978056571143953 4
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{43} + t^{32} + t^{10} + t^7 + 1 \\
|E| &= 554533938824162971915682836828616740687287415075163315034095916127803781958770790178996334428856711468055227269765859335325385915 6 \\
|E^{tw}| &= 554533938824162971915682836828616740687287415075163315034095916118044741163479459036993428012799119770901257114795569396264277214 2
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{45} + t^{37} + t^{22} + t^4 + 1 \\
|E| &= 554533938824162971915682836828616740687287415075163315034095916114991878189528752633662396905971077180989666471486048307835853461 2 \\
|E^{tw}| &= 554533938824162971915682836828616740687287415075163315034095916130856644932721496582327365535684754057966817913075380423753809668 6
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{46} + t^{28} + t^{24} + t^3 + 1 \\
|E| &= 554533938824162971915682836828616740687287415075163315034095916125581618652337531022264339970615967474141042292107806174396372693 2 \\
|E^{tw}| &= 554533938824162971915682836828616740687287415075163315034095916120266904469912718193725422471039863764815442092453622557193290436 6
\end{aligned}
$$

E.5 $\mathbb{F}_{2^{479}}$

$$
\begin{aligned}
d &= t^{73} + t^{29} + t^3 + 1 \\
|E| &= 1560874275157996115690798614896583152874299071332485575429578 \\
&\quad 4798126858694154480197179544588188676304693469803241139597788 \\
&\quad 9616430979594599455835 6 \\
|E^{tw}| &= 1560874275157996115690798614896583152874299071332485575429578 \\
&\quad 4798126858694043176004023516442446243406904799507972926631165 \\
&\quad 5181136949233931173182 2
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{94} + t^{43} + t^{41} + 1 \\
|E| &= 1560874275157996115690798614896583152874299071332485575429578 \\
&\quad 4798126858708108608628131342254525521084844155094535193191262 \\
&\quad 5764905711113509109781 2 \\
|E^{tw}| &= 1560874275157996115690798614896583152874299071332485575429578 \\
&\quad 4798126858680089047573071718776109398626754114216678873037691 \\
&\quad 9032662217715021519236 6
\end{aligned}
$$

E.6 $\mathbb{F}_{2^{487}}$

$$
\begin{aligned}
d &= t^{55} + t^{30} + t^3 + 1 \\
|E| &= 3995838144404470056168444454135252871358205622611163073099720 \\
&\quad 9083204758255556091869563226657629826030629212337010898152703 \\
&\quad 2474566073020526864748532 \\
|E^{tw}| &= 3995838144404470056168444454135252871358205622611163073099720 \\
&\quad 9083204758258229908005516609580795568431062357099697111393420 \\
&\quad 2207207824780511545536526
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{65} + t^{13} + t^7 + 1 \\
|E| &= 3995838144404470056168444454135252871358205622611163073099720 \\
&\quad 9083204758256043727688599021503197386577846540517766587158717 \\
&\quad 5064475961767438144863476 \\
|E^{tw}| &= 3995838144404470056168444454135252871358205622611163073099720 \\
&\quad 9083204758257742272186480814735228007883845028918941422387405 \\
&\quad 9617297936033600265421582
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{69} + t^{33} + t^{15} + 1 \\
|E| &= 3995838144404470056168444454135252871358205622611163073099720 \\
&\quad 9083204758260181862138226693854521544466825345498675533707766 \\
&\quad 1569719769645413325977844 \\
|E^{tw}| &= 3995838144404470056168444454135252871358205622611163073099720 \\
&\quad 9083204758253604137736853142383903849994866223938032475838357 \\
&\quad 3112054128155625084307214
\end{aligned}
$$

$$
\begin{aligned}
d &= t^{74} + t^{48} + t^{27} + 1 \\
|E| &= 3995838144404470056168444454135252871358205622611163073099720 \\
&\quad 9083204758256735425657484801447733975998430652059371552968318 \\
&\quad 2169360538780459219284852 \\
|E^{tw}| &= 3995838144404470056168444454135252871358205622611163073099720 \\
&\quad 9083204758257050574217595034790691418463260917377336456577805 \\
&\quad 2512413359020579191000206
\end{aligned}
$$

E.7 $\mathbb{F}_{2^{521}}$

$$
\begin{aligned}
d &= t^{66} + t^{29} + t^{28} + 1 \\
|E| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976570894395032898316224396265394341977861311221 \\
&\quad 626406898579780051322403286027822 04 \\
|E^{tw}| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976550148056159914912866703280531885851755849520 \\
&\quad 8133530957530962001581625362733210 2 \\
d &= t^{75} + t^{62} + t^{6} + 1 \\
|E| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976564432561837162033013678226829199034898897360 \\
&\quad 2935351114141960457442165751243321 2 \\
|E^{tw}| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976556609889355651196077421319097028794718263382 \\
&\quad 1462248829186802057363492471768109 4 \\
d &= t^{103} + t^{56} + t + 1 \\
|E| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976528033712159057973545582840523383252706840843 \\
&\quad 8328505837298857563826672713504802 8 \\
|E^{tw}| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976593008739033755255545516705402844576910319898 \\
&\quad 6069094106029904950978985509506627 8 \\
d &= t^{37} + t^{35} + t^{33} + t^{4} + 1 \\
|E| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976517164407917265121660278630159284760578277825 \\
&\quad 1650489209112087036858053848931065 2 \\
|E^{tw}| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976603878043275548107430820915766943069038882917 \\
&\quad 2747110734216675477947604374080365 4 \\
d &= t^{50} + t^{31} + t^{28} + t^{19} + 1 \\
|E| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976566721945235445951060107371029241756511300875 \\
&\quad 5882793179045428414368021001871698 8 \\
|E^{tw}| &= 6864797660130609714981900799081393217269435300143305409394463 \\
&\quad 4591855431833976554320505957367278030992174896986073105859866 \\
&\quad 8514806764283334100437637221139731 8
\end{aligned}
$$

E.8 $\mathbb{F}_{2^{569}}$

$$
\begin{aligned}
d &= t^{56} + t^{45} + t^{41} + 1 \\
|E| &= 1932268761508629172347675945465993672149463664853217499328617 \\
&\quad 6257257595711447802122680324312135204225690994105833596555418 \\
&\quad 4749327731790508484922465034002522088006719930 9308 \\
|E^{tw}| &= 1932268761508629172347675945465993672149463664853217499328617 \\
&\quad 6257257595711447802122681613353684021553933642054481425216355 \\
&\quad 1758538372508257089368402312705552781691361551 4118
\end{aligned}
$$

Liste des Acronymes

AES *Advanced Encryption Standard*. Standard de chiffrement symétrique. 7, 8, 10

AGM *Arithmetic-Geometric Mean*. Algorithme pour compter le nombre de points d'une courbe elliptique sur un corps fini de caractéristique 2. 32, 93

ANSI *American National Standard*. 91

ANSSI Agence National de la Sécurité des Systèmes d'Information. 11

BEC *Binary Edwards Curves*. Courbes Binaires d'Edwards. iv, vi, 17, 41, 42, 46, 50, 51, 65–67, 69, 70, 72, 73, 76–79, 81, 83–85, 87–90, 92–95, 97, 106, 107, 109, 110, 113, 121, 137, 139–142, 168, 180

BSBG *Baby-Step Giant-Step*. Méthode de résolution du DLP. 57, 58, 71

CA *Certification Autority*. Autorité de certification. 9, 10

CBC *Cipher Block Chaining*. 7

CMOS *Complementary Metal-Oxide-Semiconductor*. iv, 60, 61

CPA *Correlation Power Attack*. 12

CSCA *Correlation Side Channel Attacks*. 68, 72, 78, 80, 118, 124

DES *Data Encryption Standard*. Ancien standard de chiffrement symétrique. 7

DFA *Differential Fault Attack*. Attaque par fautes différentielles. 77, 78, 80, 122

DH Protocole d'échange de clefs Diffie-Hellman. 53, 54

DLP *Discret Logarithm Problem*. Problème du logarithme discret. 8

DPA *Differential Power Attack*. 12

DSCA *Differential Side Channel Attacks*. 68, 69, 75, 78, 80, 83, 118

DUT *Device Under Test*. iv, 62, 63

ECB *Electronic Cipher Block*. 7

ECC *Elliptic Curves Cryptography*. 12, 27, 62, 64, 68, 70, 76, 79, 82, 87, 142

ECDH *Elliptic Curve Diffie-Hellman*. Protocole d'échange de clefs Diffie-Hellman sur courbes elliptiques. 33, 54, 71, 79, 80, 94–96, 106, 107, 111, 115, 118, 122, 124, 127, 139–141

ECDLP *Elliptic Curves Discret Logarithm Problem*. Problème du Logarithme Discret sur Courbes Elliptiques. 8, 9, 12, 27, 57, 58, 71, 77, 84, 87–89, 97, 187

ECDSA *Elliptic Curves Digital Signature Algorithm*. vii, viii, 27, 33, 50, 52, 54–57, 71, 75, 79, 80, 94–97, 105–107, 111, 115, 118, 122, 124, 127, 139–141

ECRYPT-CSA *Coordination & Support Action ECRYPT*. Programme européen H2020. 11

EdDSA *Edwards Digital Signature Algorithm*. vii, 54, 56, 57, 71, 79, 80, 94–96, 106, 111, 115, 118, 122, 124, 127, 139–141

EM Électromagnétique. 68, 69, 113

FCSR *Feedback with Carry Shift Register*. 7
FLINT *Fast Library for Number Theory*. Bibliothèque de multiprécision. 93
FPGA *Field-Programmable Gate Array*. Circuit logique programmable. 13, 58

GPIO *General Pin In Out*. 107

HCSCA *Horizonal Correlation Side Channel Attacks*. 72, 80
HDFA *Hybrid Differential Fault Attack*. iv, 122, 123
HSCA *Horizonal Side Channel Attacks*. 80
HSVSCA *Horizontal Same Value Side Channel Attacks*. 73, 80

IoT *Internet of Things*. 2, 12–14, 22, 27, 38, 39, 52, 87, 88, 92, 138, 141

LDA *Linear Discriminant Analysis*. Méthode de réduction de dimensions. 126, 130–132, 141
LFSR *Linear Feedback Shift Register*. 7

MITN *Man-In-The-Middle*. 53

NBS *National Bureau of Standards*. 7
NIST *National Institue of Standards and Technology*. vi, 7, 9, 11, 27, 39, 53, 54, 67, 87–89, 91–93, 139
NLFSR *NonLinear Feedback Shift Register*. 7

PCA *Principal Component Reduction*. Méthode de réduction de dimensions. 126, 130–132, 141
PDF *Probability Density Function*. Fonction de densité de probabilité. 75
PKI *Public Key Infrastructure*. Infrastructure à Clef Publique. 9, 10
PoI *Points of Interest*. Points d'intérêts. 74

RFC *Request For Comments*. Spécifications. 39, 54, 121
RISC *Reduced Instruction Set Computer*. vi, 12, 14, 22, 24, 106, 107, 109, 124, 137, 140, 141
RSA *Rivest, Shamir, Aldleman*. Primitive cryptographique de Rivest, Shamir et Aldleman. 8, 9, 27, 61, 64, 70, 72, 84
RSCA *Refined Side Channel Attacks*. 64–66, 80, 118, 124

SCA *Side Channel Attacks*. 61
SEA *Schoof-Elkies-Atkin*. Algorithme pour compter le nombre de points d'une courbe elliptique sur un corps fini. 32
SPA *Simple Power Attack*. 12, 76
SSCA *Simple Side Channel Attacks*. 64, 67, 68, 76, 80, 115, 124
SVSCA *Same Value Side Channel Attacks*. 66, 73, 80, 118, 124

TA *Timing Attacks*. 62, 80
TDES *Triple Data Encryption Standard*. Ancien standard de chiffrement symétrique. 7
TLS *Transport Layer Socket*. Protocole de communications sécurisées. 96
TRNG *True Random Number Generator*. Générateur de nombres aléatoires. 107

ZSCA *Zero Side Channel Attacks*. 64–66, 80, 118, 124

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : 2019LYSEM028

Antoine LOISEAU

Lightweight and secure implementation for elliptic curve cryptography for the Internet of Things.

Speciality : Microelectronic

Keywords : Cryptography, Elliptic Curves, Side Channel Attacks, IoT

Abstract : This thesis deals with issues related to the implementation of cryptography on elliptic curves in a constrained environment such as the Internet of Things (IoT). Elliptic curves are widely deployed in cryptographic applications. These implementations are generally based on the NIST standard. However, since the development of this standard several advances in terms of performance and safety have made.

In this thesis, we propose to construct a new set of elliptic curves for the Edwards binary curve model. These new curves will incorporate a number of optimizations specific to 32-bit processor architectures. The efficiency of this curve model lies in the use of differential coordinates. However, we will see that this representation is difficult to integrate into signature protocols. We will then propose two solutions to overcome this problem.

In addition, the strong security constraints against side channel attacks will lead us to evaluate our implementation against a set of physical attacks. This evaluation will validate the intrinsic security properties inherent in this elliptic curve model and the implementation choices we have made. From this study, we will propose a new physical attack combining fault attack and side channel attack.

Template attacks will be the subject of a special study. We will show how we can use these attacks to find a key in a single attack trace, despite the use of countermeasure of point randomization. To do this, we will use dimension reduction methods (PCA, LDA). We will then propose a new countermeasure to protect against this kind of attack.

École Nationale Supérieure des Mines
de Saint-Étienne

NNT : 2019LYSEM028

Antoine LOISEAU

Implémentation légère et sécurisée pour la cryptographie sur Courbes Elliptiques pour l'Internet des Objets.

Spécialité: Microélectronique

Mots clefs : Cryptographie, Courbes Elliptiques, Attaques par canaux auxiliaires, IoT

Résumé : Cette thèse traite des problématiques relatives aux implémentations de la cryptographie sur les courbes elliptiques dans un environnement contraint tel que l'Internet des Objets (IoT). Les courbes elliptiques sont largement déployées au sein d'applications cryptographiques. Ces implémentations sont généralement basées sur le standard du NIST. Cependant, de nombreuses avancées en terme de performance et de sécurité ont été faites depuis l'élaboration de ce standard.

Nous nous proposons dans cette thèse de construire un nouvel ensemble de courbes elliptiques pour le modèle de courbes binaires d'Edwards. Ces nouvelles courbes intègreront dès leur conception un certain nombre d'optimisations spécifiques aux architectures de processeurs de 32 bits. L'efficacité de ce modèle de courbes réside dans l'utilisation de coordonnées différentielles. Cependant, nous verrons que cette représentation est difficilement intégrable au sein des protocoles de signatures. Nous proposerons alors deux solutions pour pallier à cette problématique.

Par ailleurs, les contraintes fortes en terme de sécurité face aux attaques par canaux auxiliaires nous mèneront à l'évaluation de notre implémentation face à un ensemble d'attaques physiques. Cette évaluation permettra de valider les propriétés de sécurité intrinsèques inhérentes à ce modèle de courbe elliptique et aux choix d'implémentations que nous aurons fait. De cette étude, nous proposerons une nouvelle attaque physique mêlant attaque par faute et attaque par canaux auxiliaires.

Les attaques par profilages feront l'objet d'une étude particulière. Nous montrerons comment nous pouvons utiliser ces attaques pour retrouver une clef en une seule trace d'attaque, malgré l'utilisation de la contremesure ajoutant de l'aléatoire dans la représentation des points. Pour cela, nous utiliserons les méthodes de réduction de dimensions (PCA, LDA). Nous proposerons alors une nouvelle contremesure pour se prémunir de ce genre d'attaque.