Université Saad Dahleb de Blida 1
Faculté des Sciences

Thèse de doctorat

en Informatique

Spécialité : Informatique Répartie et Mobile

La cryptographie des courbes elliptiques pour les réseaux à ressources restreintes : cas des réseaux de capteurs sans fil

Par

Mohamed RAMDANI

Devant le jury composé de :

Mme S. Oukid	Pr, Université de Blida 1	Présidente
Mlle. N. Boustia	Pr, Université de Blida 1	Examinatrice
M. L. Sekhri	Pr, Université de Oran	Examinateur
M. M. Daoui	Pr, Université de Tizi Ouzou	Examinateur
M. M. Benmohammed	Pr, Université de Constantine 2	Directeur
Mlle. N. Benblidia	Pr, Université de Blida 1	co-Directrice

Blida, 09 Juillet 2020

Ecole doctorale Informatique Répartie et Mobile IRM
Laboratoire de Recherche pour le Développement des Systèmes Informatisés LRDSI
Université de Blida 1
Algérie

Laboratoire d'Informatique REpartie LIRE
Université de Constantine 2
Algérie

Résumé

Ces dernières années, la technologie a connu une grande évolution dans le domaine de l'électronique, nanotechnologies et les communications sans fil. Elle a donné naissance à une nouvelle génération de réseaux intitulés réseaux sans fil. Les nœuds formant un réseau sans fil sont souvent autonomes et faibles en ressources comme dans les réseaux de capteurs sans fil (RcSF). Un capteur est un véritable système embarqué alimenté par une batterie faible en capacité et un microcontrôleur faible en puissances de calcul et de stockage mémoire. Les RcSF constituent une technologie émergente à faible coût utilisée dans des applications stratégiques et sensibles nécessitant une sécurité accrue. Depuis l'émergence des réseaux de capteurs sans fil, utilisés principalement dans des applications stratégiques et sensibles et déployés dans des zones hostiles et inaccessibles, la protection des données s'est imposée comme une nécessité incontournable. L'utilisation de la cryptographie dans les opérations de protection de données permet d'exploiter des algorithmes plus complexes pour une sécurité (presque) optimale notamment pour la sécurité des échanges et l'authentification des utilisateurs. L'avantage d'opter pour la cryptographie des courbes elliptiques (ECC) est lié à la longueur réduite de ces clés pour un meilleur niveau de sécurité comparativement à d'autres systèmes comme RSA.

L'opération principale et centrale dans le calcul des clés et du texte chiffré dans les ECC est la multiplication d'un scalaire par un point de la courbe. Cette opération est trop complexe et nécessite beaucoup de calculs et par conséquent consomme énormément d'énergie. Plusieurs solutions ont été proposées pour réduire les calculs, améliorer la complexité et optimiser ainsi la consommation des ressources. Parmi les solutions proposées la parallélisation et la distribution des calculs. Toutes les solutions proposées jusqu'à présent permettent de distribuer et paralléliser les calculs sur un seul nœud. Ces techniques sont néanmoins efficaces pour réduire le temps de calcul mais non pas pour réduire la consommation d'énergie. Dans ce travail de thèse, nous avons proposé plusieurs solutions pour l'optimisation et la réduction des calculs d'une multiplication scalaire, en distribuant les traitements d'une multiplication scalaire sur plusieurs nœuds sans toutefois divulguer la clé privée aux nœuds participants. L'objectif du calcul distribué est de décomposer une tâche en plusieurs sous tâches qui peuvent être traitées indépendamment par plusieurs nœuds appartenant à un cluster d'un réseau. Les deux protocoles proposés permettent de réduire efficacement les coûts tout en assurant un très haut niveau de sécurité.

Mots clés : Réseaux de capteurs sans fil, Cryptographie des courbes elliptiques, Multiplication scalaire, Calculs distribués, complexité des calculs

Abstract

In recent years, technology has seen a great evolution in the field of electronics, nano technologies and wireless communications. It gave birth to a new generation of networks called wireless networks. Nodes forming a wireless network are often autonomous and low in resources such as Wireless sensor Networks (WsN). A sensor is a real embedded system powered by a battery low in capacity and a microcontroller low in computing power and storage. WsN is a low-cost emerging technology used in strategic and sensitive applications requiring increased security. Since the emergence of wireless sensor networks, used primarily in strategic and sensitive applications and deployed in hostile and inaccessible areas, data protection has emerged as an unavoidable necessity. The use of cryptography in data protection operations makes it possible to exploit more complex algorithms for (near) optimal security, in particular for the security of exchanges and the authentication of users. The advantage of opting for cryptography of elliptic curves (ECC) is related to the reduced length of these keys for a better level of security compared to other systems like RSA.

The main and central operation in the calculation of keys and ciphertext in ECC is the scalar point multiplication. This operation is too complex and requires a lot of calculations and therefore consumes a lot of energy. Several solutions have been proposed to reduce calculations, improve complexity and thus optimize the consumption of resources. Among the proposed solutions, the parallelization and distribution of calculations. All the solutions proposed so far make it possible to distribute and parallelize the calculations on a single node. These techniques are nonetheless effective in reducing computing time but not in reducing energy consumption. In this thesis work, we have proposed several solutions for the optimization and reduction of scalar multiplication calculations, by distributing scalar multiplication processing on several nodes without however disclosing the private key to the participating nodes. The purpose of distributed computing is to break down a task into several sub-tasks that can be handled independently by multiple nodes belonging to a cluster of a network. The two proposed protocols can effectively reduce costs while ensuring a very high level of security.

Keywords : Wireless sensor Networks, Elliptic Curve Cryptography, Scalar point Multiplication, Distributed Computations, computationnal complexity

ملخص

في السنوات الأخيرة ، مرت التكنولوجيا بتطور كبير في مجال الإلكترونيات وتكنولوجيا النانو والاتصالات اللاسلكية. لقد ولدت جيلاً جديداً من الشبكات يسمى الشبكات اللاسلكية. غالباً ما تكون العقد التي تشكل شبكة لاسلكية قائمة بذاتها و منخفضة الموارد كما هو الحال في شبكات الاستشعار اللاسلكية. المستشعر عبارة عن نظام داخلي حقيقي يتم تشغيله بواسطة بطارية منخفضة السعة ومتحكم ضعيف في طاقة الحوسبة وتخزين الذاكرة. شبكات الاستشعار اللاسلكية هي تقنية ناشئة منخفضة التكلفة تُستخدم في التطبيقات الحساسة والمهمة التي تتطلب أماناً متزايداً. منذ ظهور شبكات الاستشعار اللاسلكية ، التي تستخدم بشكل أساسي في التطبيقات الحساسة والمهمة للمهام ، والتي تم نشرها في مناطق معادية والتي يتعذر الوصول إليها ، ظهرت حماية البيانات كضرورة ملحة. استخدام التشفير في عمليات حماية البيانات يجعل من الممكن استغلال خوارزميات أكثر تعقيداً (تقريباً) للأمان الأمثل ، لا سيما لأمن التبادلات ومصادقة المستخدم. ترتبط ميزة اختيار تشفير المنحنى الإهليلجي بتقليل طول هذه المفاتيح للحصول على مستوى أمان أفضل مقارنة بالأنظمة الأخرى.

العملية الرئيسية والمركزية في حساب المفاتيح والنص المشفر في تشفير المنحنيات الإهليلجية هي مضاعفة العددية بنقطة من المنحنى. هذه العملية معقدة للغاية وتتطلب الكثير من الحسابات وبالتالي فهي تستهلك الكثير من الطاقة. تم اقتراح العديد من الحلول لتقليل الحسابات وتحسين التعقيد وبالتالي تحسين استهلاك الموارد. من بين الحلول المقترحة الموازاة وتوزيع الحسابات. تسمح جميع الحلول المقترحة حتى الآن بتوزيع الحسابات وموازنتها على عقدة واحدة. ومع ذلك ، فإن هذه التقنيات فعالة في تقليل وقت الحوسبة ، ولكن ليس في تقليل استهلاك الطاقة. في هذا البحث ، اقترحنا عدة حلول لتحسين وتقليل حسابات الضرب القياسي ، من خلال توزيع عمليات الضرب القياسي على عدة عقد دون الكشف عن المفتاح الخاص للعقد المشاركة. الهدف من الحوسبة الموزعة هو تقسيم المهمة إلى عدة مهام فرعية يمكن معالجتها بشكل مستقل عن طريق عدة عقد تنتمي إلى مجموعة من الشبكة. يتيح البروتوكولين المقترحين تقليل التكاليف بكفاءة مع ضمان مستوى عالٍ جداً من الأمان.

الكلمات المفتاحية: شبكات الاستشعار اللاسلكية ، تشفير المنحنيات الناقصية ، الضرب العددي ، الحسابات الموزعة ، تعقيد العمليات الحسابية

Remerciements

La thèse de doctorat est loin d'être un travail solitaire. Elle représente le fruit d'un travail s'inscrivant dans la durée. Pour cette raison, de nombreuses personnes se retrouvent de manière fortuite ou non entre l'auteur et sa thèse. Ce sont ces personnes que je souhaite mettre en avant dans ces remerciements.

Tout d'abord, je remercie chaleureusement toutes ces personnes qui m'ont aidé pendant l'élaboration de ma thèse et très particulièrement mon directeur de thèse Pr Mohamed Benmohammed ainsi que ma directrice de thèse Pr Nadjia Benblidia, pour leur intérêt et leur soutien, leur grande disponibilité et leurs nombreux conseils durant la préparation et la rédaction de ma thèse.

Je remercie chaleureusement, au terme de ce parcours, celles et ceux et me sont chers et que j'ai quelque peu délaissés ces derniers mois pour achever ma thèse. Leurs attentions et leurs encouragements m'ont accompagné tout au long de ces années. Je suis redevable à mes parents pour leur soutien moral et matériel et leur confiance indéfectible dans mes choix.

Également, je remercie infiniment ma chère épouse pour son soutien quotidien indéfectible et son enthousiasme contagieux à l'égard de mes travaux comme de la vie en général. Notre couple a grandi en même temps que mon projet scientifique, le premier servant de socle solide à l'épanouissement du second.

Mes remerciements vont aussi à ma famille et mes amis qui, avec cette question récurrentes "Quand est ce que tu la soutiens cette thèse", bien qu'angoissante en période de doutes, m'ont permis de ne jamais dévier de mon objectif final. Merci à mon frère et son épouse, mes sœurs, mes neveux Enzo, Ziane et Fouzi, et ma nièce Narimane, ainsi que les membres de famille de mon épouse, ses adorables parents, ses frères et ses oncles.

À mon cher fils, Yastene

Table des matières

Liste des figures

Liste des tableaux

Liste des Algorithmes

Liste d'abréviations

A : Addition

AES : Advanced Encryption Standard

CBC : Cipher Block Chaining

CH : Cluster Head

CPU : Central Processing Unit

CRC : Cyclic Redundancy Check

CTR : CounTeR

dBm : decibel-milliwatts

DES : Data Encryption Standard

DoS : Denial of Service

DPA : Differential Power Analysis

DPSM : Distributed and Parallel Scalar point Multiplication

ECB : Electronic Code Book

ECC : Elliptic Curve Cryptography

EEPROM : Electrically-Erasable Programmable Read-Only Memory

FDSM : Full Distributed Scalar point Multiplication

FIPS : Federal Information Processing Standards

Fp : Corps fini premier

GCC : GNU Compiler Collection

GHz : Giga Hertz

GO : Giga Octets

GPS : Global Positioning System

I : Inversion

IHM : Interface Homme-Machine

IdO : Internet des Objets

iIoT : industrial Internet of Things

IoT : Internet of Things

iWsN : industrial Wireless sensor Network

KB : KiloByte

Kbps : Kilo bits par seconde

Ko : Kilo Octets

LED : Light-Emitting Diode

LIPO : LIthium POlymer

LR : Left-to-Right

LR-WPAN : Low-Rate Wireless Personal Area Networks

M : Multiplication

mA : milli-Ampère

MANETs : Mobile Ad-hoc NETworks

MCU : Microcontroller Unit

MEMS : Micro-Electro-Mechanical Systems

Mhz : Mega Hertz

MIPS : Million d'Instructions Par Seconde

mJ : milliJoule

mW : milliWatt

Mo : Mega Octets

ms : milliseconde

NAF : Non-Adjacent Form

NIST : National Institute of Standards and Technologies

OFB : Output Feedback

PWM : Pulse Width Modulation

RAM : Random Access Memory

RC4 : Rivest Cipher 4

RcSF : Réseaux de Capteurs Sans Fil

RISC : Reduced Instruction Set Computer

RL : Right-to-Left

ROM : Read-Only Memory

S : Squaring

RSA : Rivest Shamir Adleman

Rx : Réception

SCA : Side-Channel Analysis

SIMD : Single Instruction Multiple Data

SPA : Simple Power Analysis

SRAM : Static Random Access Memory

Tx : Transmission

uA : microAmpère

UART : Universal Asynchronous Receiver Transmitter

u-ecc : micro ECC

us : microseconde

VANETs : Vehicular Ad hoc NETworks

WBANs : Wireless Body Area Networks

Wi-Fi : Wireless Fidelity

WPAN : Wireless Personal Area Networks

WsN : Wireless sensor Network

Introduction générale

Contexte général

Les développements récents dans le domaine des communications sans fil ont permis l'émergence de nouveaux types de réseaux autonomes, intelligents et multifonctionnels. L'essor également des systèmes micro-électro-mécaniques (MEMS - Micro-Electro-Mechanical Systems) ont facilité la réalisation de dispositifs petits (souvent minuscules), limités en ressources et autonomes installés dans des endroits souvent hostiles et parfois inaccessibles. Ces dispositifs embarqués sont regroupés pour former des réseaux dits réseaux à ressources restreintes ou bien systèmes embarqués en réseau (Networked Embedded Systems). Ces derniers sont composés généralement de dispositifs appelés systèmes embarqués.

Un système embarqué est un système informatique autonome, dédié généralement à des tâches bien précises, d'une taille limitée ayant des ressources faibles. On distingue deux contraintes importantes sur un système embarqué : une puissance de calcul définie au plus juste tout en respectant les contraintes temporelles et spatiales, et une sécurité indispensable pour assurer la confidentialité des données notamment pour des applications sensibles. Les systèmes embarqués sont regroupés pour former des réseaux autonomes et multifonctionnels couvrant un large champ d'application. Parmi ces réseaux, nous traiterons dans cette thèse le cas des réseaux de capteurs sans fil.

Les réseaux de capteurs sans fil (RcSF) est un domaine de recherche en pleine expansion qui a attiré beaucoup d'attentions ces dernières années dans les milieux universitaires et industriels. Un RcSF est formé d'un nombre important de petits appareils multifonctionnels autonomes et de tailles minuscules appelés capteurs ou micro-capteurs, interconnectés entre eux via des liaisons sans fil. Le déploiement, souvent aléatoire, des nœuds capteurs dans des zones stratégiques et sensibles, hostiles, inaccessibles aux humains et sans assistance a permis à la technologie des réseaux de capteurs d'avoir un champ d'application vaste et varié. Les domaines ciblés sont multiples : militaires, civils, environnementales, agricoles, médicales, domestiques, industriels, etc.

Les contraintes de conception d'un réseau de capteurs sans fil dépendent de leurs caractéristiques. En effet, les ressources limitées des capteurs en termes de capacités de calcul, de stockage et d'énergie, la collaboration et l'auto-organisation des nœuds pour réaliser des tâches complexes, la densité du réseau, l'absence d'infrastructures de base et de sécurité physique sont quelques caractéristiques imposant des contraintes de mise en place de ce type de réseaux. Deux de ces contraintes sont essentielles : i) une consommation raisonnable des ressources notamment d'énergie afin de prolonger la durée de vie des capteurs et celle du réseau et ii) la garantie des communications

sécurisées pour les échanges de données.

En effet, la nature de transmission sans fil et les ressources limitées et restreintes constituent les vulnérabilités des RcSF et augmentent, par conséquent, le risque d'attaques par des entités étrangères et malicieuses. Afin de sécuriser, authentifier, garantir la disponibilité et la confidentialité des informations émises et reçues, on a souvent besoin de recourir à des solutions de sécurité et très particulièrement à des mécanismes cryptographiques, considérés comme le meilleur moyen de protection pour ce genre de réseaux. La cryptographie représente les méthodes de chiffrement et de déchiffrement des données par les utilisateurs autorisés du réseau. La cryptographie moderne se divise en deux parties, symétrique et asymétrique.

Dans la cryptographie symétrique, les opérations de chiffrement et de déchiffrement se font avec une clé unique, tandis que dans les systèmes cryptographiques asymétriques, elles se font avec deux clés différentes liées par une fonction à sens unique. La clé de déchiffrement est difficilement calculable à partir de la clé de chiffrement et cette difficulté est dû à l'impossibilité de résoudre en temps polynomial des problèmes mathématiques complexes comme le problème de calcul du logarithme discret dans un groupe, exploité par la cryptographie des courbes elliptiques (ECC). Plusieurs solutions de sécurité sont proposées pour les réseaux où les contraintes sur les ressources restreintes des nœuds capteurs sont importantes.

La découverte du problème du logarithme discret en 1976 et encore la factorisation des entiers en 1977 ont donné encore plus d'élan et de progression pour les travaux déjà en cours sur la cryptographie des courbes elliptiques (ECC). C'est ainsi que KOBLITZ et MILLER ont été, indépendamment, les premiers à adapter les courbes elliptiques à la cryptographie en 1985 [1, 2]. L'intérêt de ces cryptosystèmes est leur niveau de sécurité élevé comparativement au cryptosystème RSA. En effet, pour un même niveau de sécurité, la cryptographie des courbes elliptiques utilisent des clés de plus petite taille. Il a été démontré qu'une clé du système ECC de 160 bits fournit le même niveau de sécurité qu'une clé de taille 1024 bits du système RSA [3].

Les systèmes de cryptographie à base de courbes elliptiques (ECC) sont considérés comme une nouvelle architecture cryptographique dite moderne, adoptés comme des systèmes standardisés de cryptographie asymétrique. Par conséquent, la cryptographie des courbes elliptiques est devenue une technologie indispensable et très adaptée aux implémentations sur les réseaux à ressources limitées comme les réseaux de capteurs sans fil. Les courbes elliptiques existent déjà depuis très longtemps (on parle du troisième siècle) pour résoudre les problèmes arithmétiques anciens. On a commencé leur étude en algèbre géométrique au milieu du 19eme siècle. En 1984, Hendrik LENSTRA présente la première description d'un algorithme de factorisation polynomiale sur les courbes elliptiques, ce qui a motivé les chercheurs à s'intéresser sérieusement sur l'aspect cryptographique de ces structures et dans le calcul théorique des nombres [4, 5].

Problématique et motivations

L'évolution continue du domaine de recherche sur les réseaux de capteurs sans fil et leur technologie rend possible leur utilisation dans de nombreux champs d'applications. Un grand nombre de capteurs sont déployés sur une zone à surveiller. Chaque capteur dispose d'une antenne radio et d'une batterie qui le rend autonome pour effectuer ses tâches dans des endroits hostiles, dangereux et inaccessibles où l'information transmise à l'utilisateur est souvent sensible et critique. Néanmoins, les contraintes de déploiement et de mise en place sont multiples. Les plus importantes concernent les capteurs eux-mêmes : la portée radio faible, les ressources limitées, l'autonomie de la source d'énergie, l'absence de la sécurité physique, l'absence de moyens de localisation...etc.

Les limites physiques et énergétiques des nœuds capteurs, l'accessibilité du réseau aux attaquants et la nature vulnérable des communications sans fil sont, entre autres, des facteurs qui peuvent augmenter le niveau du risque d'attaques sur les RcSF. Plusieurs domaines de recherche sont apparus récemment proposant des solutions de sécurité capables de remédier aux insuffisances des capteurs et aux vulnérabilités du médium sans fil utilisé dans les communications. Le recours aux mécanismes cryptographiques et principalement à la cryptographie à clé publique, longtemps considérée comme inapplicable aux RcSF car elle nécessite souvent l'emploi de primitives cryptographiques complexes et coûteuses en temps de calcul et par conséquent en consommation d'énergie, constitue actuellement un challenge intéressant pour la communauté scientifique du domaine. Beaucoup de travaux ont prouvé leur faisabilité dans les réseaux à ressources restreintes comme les réseaux de capteurs sans fil.

La cryptographie est le meilleur moyen pour sécuriser les communications dans les RcSF. Cependant, la compromission d'un nœud peut rapidement s'avérer fatal pour la sécurité du réseau entier, et la plupart des solutions proposées ne peuvent atténuer de telles attaques. Le problème d'emploi des primitives cryptographiques dans un RcSF est lié aux limites des nœuds capteurs en ressources physiques et énergétiques. Ces limites empêchent le nœud capteur à gérer les calculs nécessaires aux opérations cryptographiques. Les approches naïves utilisent les primitives classiques de protection des informations, engendrant un surcout de consommation de ressources. Beaucoup d'optimisations sont nécessaires pour réduire ces consommations et assurer un niveau de sécurité élevé.

En utilisant des clés de taille courte, la cryptographie des courbes elliptiques présente un avantage et un intérêt dans la rapidité des calculs. Cet avantage est adapté pour les systèmes qui disposent de capacités limitées et des ressources restreintes, notamment en mémoire et en puissance de calcul. L'opération la plus délicate, la plus gourmande en consommation de ressources et la plus complexe sur les courbes elliptiques est la multiplication scalaire, souvent utilisée dans les protocoles cryptographiques. Le

résultat d'exécution de cette opération influence d'une manière directe sur les performances des systèmes basés sur la cryptographie des courbes elliptiques. Plusieurs travaux d'optimisation sont en cours par les chercheurs du domaine afin d'accélérer le calcul et la multiplication scalaire.

Notre contribution consiste à assurer la sécurité des communications c'est-à-dire déterminer les moyens cryptographiques qui garantissent le couplage des exigences de sécurité avec les caractéristiques et les contraintes des nœuds capteurs. L'objectif de cette thèse est donc de fournir des solutions d'un déploiement sécurisé pour des types d'applications critiques, sensibles et stratégiques en définissant les outils nécessaires pour des communications efficaces, robustes et sécurisées.

Contributions de la thèse

La plupart des solutions, dites modernes, pour les réseaux à basses ressources utilisant des procédés cryptographiques à clés publique interviennent, souvent, pour la résolution du problème d'échange de clés entre chaque paire de nœuds après le déploiement. La sécurisation des liens est assurée par des procédés cryptographiques à clé privée reconnus pour leur consommation raisonnable de ressources. Par ailleurs, plusieurs études ont démontré l'adaptation des techniques de la cryptographie asymétrique à la sécurisation des communications à moindre coût en ressources. L'une de ces techniques est la cryptographie sur les courbes elliptiques (ECC).

Dans le cadre de cette thèse, nous avons étudié profondément la faisabilité de la cryptographie des courbes elliptiques (ECC) pour les réseaux à ressources limitées et nous nous sommes intéressés aux problèmes de sécurité dans les échanges de données dans les réseaux de capteurs sans fil. Nous avons proposé par la suite de nouvelles architectures et implémentations robustes et efficaces, basées sur les courbes elliptiques, pour la protection des communications entre n'importe quel nœud légitime et sa destination sur le réseau. Pour cela, nous avons évalué les différents algorithmes basés sur la cryptographie des courbes elliptiques en termes de complexité, puis nous avons implémenté des solutions algorithmiques adéquates pour les réseaux de capteurs sans fil (RcSF), sur un dispositif embarqué très limité en ressources, qui peuvent satisfaire conjointement aux ressources limitées de ce type de réseaux avec les exigences habituelles de sécurité. Nous avons proposé plusieurs solutions pour optimiser les calculs elliptiques en appliquant des solutions distribuées et parallèles.

Organisation de la thèse

La thèse est organisée en trois parties et six chapitres. La première partie est consacrée aux réseaux à ressources restreintes et les réseaux de capteurs sans fil, dans la

deuxième partie nous faisons une revue de l'état de l'art sur la cryptographie des courbes elliptiques et la dernière partie présente nos contributions.

La première partie introduit des généralités sur les réseaux à ressources restreintes en présentant des définitions sur les systèmes embarqués, quelques exemples de réseaux à ressources restreintes, leurs caractéristiques communes et leurs domaines d'applications. Ensuite, nous donnons des concepts et définitions sur les réseaux de capteurs sans fil. Après quelques définitions, l'architecture, les domaines d'applications et des contraintes de conception des RcSF, nous passons aux sections les plus importantes abordant les problèmes et les exigences de sécurité, quelques attaques sur les RcSF et quelques solutions proposées dans la littérature. Pour terminer, nous discutons la consommation d'énergie dans un réseau de capteurs sans fil.

La seconde partie comprend deux chapitres traitant exclusivement la cryptographie des courbes elliptiques. Au deuxième chapitre, nous donnons des généralités sur la cryptographie des courbes elliptiques. Nous commençons par présenter quelques préliminaires et définitions sur les courbes elliptiques avant d'aborder la cryptographie des courbes elliptiques et les cryptosystèmes basés sur les courbes elliptiques. Au troisième chapitre, nous présentons quelques optimisations de la multiplication scalaire sur courbes elliptiques. Nous commençons par la présentation de l'algorithme standard de calcul d'une multiplication scalaire puis nous abordons quelques techniques d'optimisation : techniques mathématiques, algorithmiques et celles basées sur la parallélisation des calculs.

La dernière partie comporte trois chapitres et elle est dédiée à la présentation de nos contributions. Au quatrième chapitre, nous proposons quelques combinaisons de techniques mathématiques et algorithmiques pour l'optimisation d'une multiplication scalaire. Nous entamons le chapitre par présenter quelques algorithmes optimisés pour l'accélération d'une multiplication scalaire puis nous évaluons et comparons la complexité des calculs. Pour évaluer les performances des solutions proposées, nous donnons une mise en œuvre logicielle et matérielle de l'environnement d'implémentation puis les résultats de nos expérimentations. Dans le cinquième chapitre, nous présentons notre approche distribuée pour le calcul d'une multiplication scalaire. Avant de présenter notre approche, nous présentons d'abord nos algorithmes distribués. Dans la partie analyse des performances, nous donnons les coûts en énergie des calculs elliptiques et nous évaluons les performances de notre approche distribuée. Enfin nous validons notre solution par une comparaison globale avec d'autres solutions de la littérature et une analyse de sécurité complète. Au dernier chapitre, nous proposons deux protocoles de sécurité distribués et parallèles basés sur les courbes elliptiques pour les réseaux de capteurs sans fil. Après avoir présenté le fonctionnement global, nous présentons nos deux protocoles avec et sans points préchargés. Nous appliquons ensuite notre protocole sans points préchargés pour un échange de secret dans un cluster. Afin de comparer

les performances de nos deux protocoles, nous donnons d'abord les paramètres d'implémentation puis nous évaluons leurs performances en termes de coûts de calcul, de stockage mémoire, des communications et de consommation d'énergie. Nous donnons par la suite les résultats des comparaisons avec d'autres solutions de la littérature.

Nous concluons par une conclusion générale et des perspectives.

Première partie

Les réseaux à ressources restreintes et les réseaux de capteurs sans fil

Chapitre 1
Généralités sur les réseaux à ressources restreintes et les réseaux de capteurs sans fil

L'émergence d'un certain type de dispositifs miniaturisés, intelligents et autonomes, appelés systèmes embarqués, a bouleversé le domaine des réseaux sans fil modernes. L'interconnexion d'un nombre suffisant de dispositifs embarqués permet d'interagir entre eux pour former des réseaux sans fil autonomes et intelligents. Cependant, les limites physiques des dispositifs embarqués en puissance de calcul, tributaire du type de l'application dédiée, du stockage mémoire mais également en ressources énergétiques, sont des contraintes influençant l'application des réseaux sans fil, connus alors sous l'appellation de réseaux à ressources restreintes.

Il existe différents types de réseaux à ressources restreintes couvrant un grand nombre de domaines sensibles et stratégiques comme la santé, la sécurité, l'environnement, la domotique, etc. Certains réseaux sont dédiés à des applications précises et d'autres peuvent être déployés pour un large champ d'application, comme les réseaux de capteurs sans fil. Les réseaux étudiés dans ce chapitre sont les réseaux ad hoc mobiles ou Mobile Ad-hoc NETworks (MANETs), les réseaux connus sous le nom de "Internet des objets ou Internet of Things (IoT)", les réseaux de capteurs sans fil (RcSF) ou Wireless sensor Network (WsN), les réseaux de capteurs industriels (iWsN) qui constitue une partie principale de l'internet industriel des objets (iIoT), les réseaux de capteurs multimédias, les réseaux corporels sans fil [6] ou Wireless Body Area Networks (WBANs), systèmes embarqués en réseau ou Networking Embedded Systems. Tous les réseaux sus cités ont des caractéristiques communes et des contraintes de conception proches. La problématique centrale étant la durée de vie du réseau qui dépend essentiellement des sources d'énergie limitées des entités qui forment un réseau.

Un réseau de capteurs sans fil est un système distribué formé par un nombre important de noeuds capteurs capables de surveiller leur environnement, de collecter des données et de les transmettre vers la station de base. Les RcSF peuvent être de topologie plate ou hiérarchique. Dans une topologie hiérarchique, le réseau est organisé en clusters. Chaque cluster possède son propre chef appelé souvent Cluster Head (CH). Les nœuds d'un cluster communiquent avec le CH qui à son tour communique avec d'autres CH et la station de base pour relayer les données reçues et récoltées vers la station de base. En raison des fonctionnalités variées des capteurs, les RcSF peuvent

avoir plusieurs familles, selon le type de chaque capteur. Ils sont classés donc en RcSF corporels, visuels, acoustiques, multimédias, souterrains, sous-marins, etc.

1.1 Définition d'un système embarqué

1.1.1 Définition

Un système embarqué est un système électronique et informatique, autonome et temps réel dans la plupart des applications, spécialisé dans une tâche bien précise, comprenant une partie matérielle et une partie logicielle. Un système embarqué traite de l'information provenant de son environnement (senseurs entre autres) et d'autres interactions provenant de l'utilisateur pour ensuite soit agir sur les processus externes via des actuateurs ou informer directement l'utilisateur. Le schéma ci-dessous présente le concept d'utilisation d'un système de climatisation de l'air complètement contrôlé par un système embarqué.

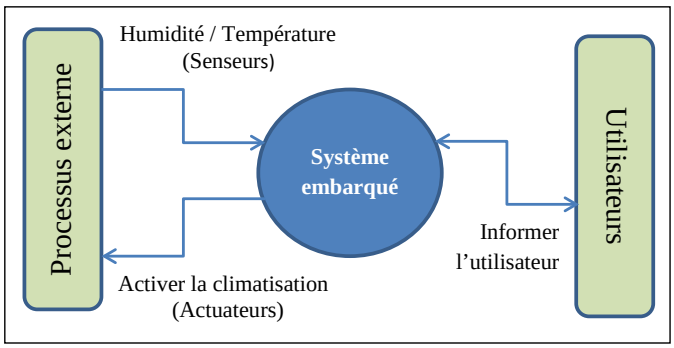

FIGURE 1.1 – Exemple d'utilisation d'un système embarqué

Sur la Figure 1.1, le système embarqué reçoit des informations provenant de l'extérieur via son unité d'entrée représentée par le senseur ou le capteur, les traite puis produit une action sur l'environnement extérieur via son unité de sortie l'actuateur ou actionneur, puis informe l'utilisateur.

Les systèmes embarqués sont présents partout. Cela est dû à leur fonctionnement autonome nécessitant peu d'interaction avec l'utilisateur en général. On les retrouve dans le domaine grand public, intégrés dans le système informatique des machines à laver, des microondes, dans les appareils photo et les caméras numériques, les téléviseurs, consoles de jeu, montres connectés, Smartphones, etc. On les retrouve également dans les dispositifs de communication comme les satellites et les antennes relais, dans le domaine de l'astronomie comme les fusées, les sondes spatiales, etc. De plus, ils sont très utilisés dans le domaine des transports comme les avions, trains, voitures, trains, tramways, métro, etc., mais aussi dans les équipements médicaux pour le diagnostic,

l'imagerie médicale, simulateur cardiaque, etc. Les systèmes embarqués sont présents également dans le monde industriel, utilisés dans les chaines de production, automates programmables industriels, stations de contrôle, comme pour surveiller le trafic aérien, les centrales nucléaires, les activités volcaniques, etc., dans l'électronique embarquée, dans des drones, etc.

1.1.2 Caractéristiques

Les caractéristiques d'un système embarqué sont nombreuses, nous citons entre autres :

- Ressources limitées notamment en mémoire et en puissance de calculs. La capacité mémoire est de l'ordre de quelques Giga Octets (Go).
- Les systèmes embarqués utilisent généralement des microprocesseurs ou des microcontrôleurs à basse consommation d'énergie.
- Les capacités physiques sont souvent adaptées à l'application. La puissance de calcul est généralement adaptée aux besoins temporels des tâches à exécuter en vue d'éviter une surconsommation d'énergie.
- Une taille très réduite.
- Consommation optimisée de l'énergie notamment sur batterie [7].
- Exécution souvent temps réel.
- Dédiés à une application spécifique contrairement aux systèmes traditionnels qui sont généralement à usage général.
- L'Interface Homme-Machine (IHM) est adapté selon l'application. L'affichage peut se faire par de simples LED jusqu'à sur un écran tactile.
- Disposent de périphériques et capteurs spécifiques comme les cartes SD, modules de communication Wi-Fi, Bluetooth, etc., capteur de température, humidité, etc.
- Les systèmes embarqués sont intégrés généralement au système qu'ils contrôlent, donc ils ne sont pas toujours indépendants.

1.1.3 Architecture d'un microcontrôleur embarqué

Un microcontrôleur, appelé aussi puce électronique, est un circuit intégré rassemblant les éléments essentiels d'un ordinateur : un microprocesseur, une mémoire Flash, une mémoire vive RAM et des interfaces d'Entrées/Sorties E/S. L'architecture d'un microcontrôleur ressemble à celle d'un ordinateur ordinaire avec d'énormes différences notamment la fréquence du processeur qui est de quelques MHz pour un microcontrôleur contre plusieurs GHz pour un ordinateur ordinaire, une consommation énergétique

trop faible et un fonctionnement autonome. La figure 1.2 ci-dessous présente l'architecture d'un microcontrôleur.

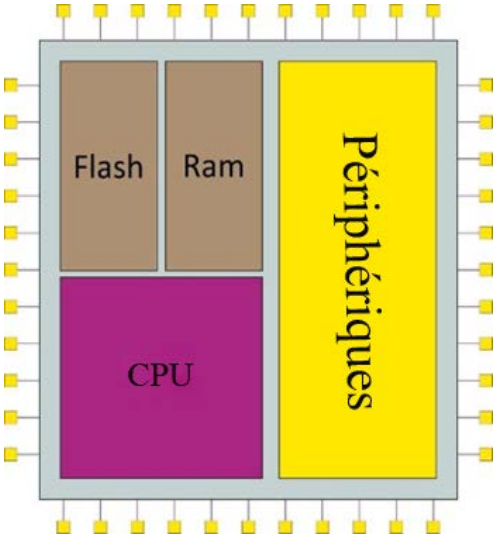

FIGURE 1.2 – Architecture générale d'un microcontrôleur

Un microcontrôleur peut être équipé d'un microprocesseur CPU à 8 bits, 16 bits ou 32 bits. Le choix du microcontrôleur dépend de l'application embarquée.

Les microcontrôleurs 8 bits présentent plusieurs avantages comme la faible consommation de ressources, le bas coût et une dimension réduite tandis que les microcontrôleurs 32 bits sont puissants avec des vitesses de calcul supérieures et un espace d'adressage plus grand. La taille de la mémoire Flash s'étend de quelques Ko à quelques centaines de Ko, et celle de la RAM de quelques octets à quelques centaines de Ko. Il existe plusieurs familles de microcontrôleurs dont les plus connues sont : Atmel AT91, Atmel AVR, C167 de Siemens/Infineon, Hitachi H8, Intel 8051, Intel 8085, Motorola 68HC11, PIC de Microchip, ST6 de STMicroelectronics. Le tableau 1.1 présente quelques caractéristiques des microcontrôleurs les plus connus et utilisés.

1.1.4 Exemple d'un microcontrôleur embarqué : ATmega328P

Le microcontrôleur ATmega 328P est de la famille AVR 8 bits fabriquée par Atmel. Ses principales caractéristiques sont : une mémoire Flash de 32 Ko pour les programmes, une mémoire SRAM de 2 Ko pour les données volatiles et une mémoire EEPROM de 1 Ko pour les données non volatiles, 28 broches en tout dont 23 broches pour les trois ports B, C et D, trois timers, le Timer 0 et le Timer 2 pour un comptage 8 bits et le Timer 1 pour un comptage 16 bits. Certaines broches sont multifonctions et peuvent avoir plusieurs fonctions différentes choisies par programmation.

TABLEAU 1.1 – Caractéristiques des microcontrôleurs les plus utilisés

Famille	Nom	Core CPU	RAM (Ko)	Rom (Ko)	Fréquence (Mhz)	CPU Clock
Microchip	PIC	PIC16F84	68	64	10	8
Atmel	AT91	SAM9263	80	128	200	32
	AVR	ATmega128L	4	128	16	8
		ATmega328P	2	32	8 - 16	8
Intel	80XX	8051	128	4096	12	8
Siemens/ Infineon	C16X	C167	2	8	20	16
Hitachi	H8	H8/534	2	32	10	16
Motorola	68HC11	MC68HC11A8	0.25	8192	4	8
STMicroelectronics	STM32F205	ARM® 32-bits Cortex	4	1024	120	32

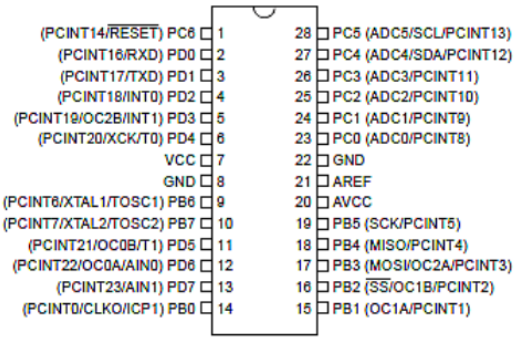

FIGURE 1.3 – Architecture du microcontrôleur ATmega 328P

Le microcontrôleur ATmega 328P fonctionne avec un voltage de 1.8 à 5.5 Volts et possède cinq modes d'économie d'énergie sélectionnables par programmation. C'est un MCU (abréviation de microcontrôleur en anglais) d'une haute performance et d'une faible consommation énergétique (1.5mA à 3V - 4MHz, 5.2mA à 5V - 8MHz et 9.2mA à 5V - 16MHz en mode actif et 0.25mA à 3V - 4MHz, 1mA à 5V - 8MHz et 1.9mA à 5V - 16MHz en mode veille). Il est basé sur l'architecture RISC (Reduced Instruction Set Computer) avancée avec une vitesse du processeur jusqu'à 16 Mhz. Pour plus d'informations sur le microcontrôleur ATmega 328P, le lecteur peut se référer au document datasheet [8] mis en ligne par le fabricant.

1.2 Exemples de quelques réseaux à ressources limitées

1.2.1 Réseaux ad hoc mobiles

Un réseau ad hoc est un système autonome constitué de nœuds mobiles. Les nœuds communiquent entre eux par des liaisons sans fil point à point. Quand deux nœuds en communication sont éloignés, appartenant à deux zones de couverture disjointes, les nœuds intermédiaires sont alors sollicités pour assurer la liaison entre ces deux nœuds communicants. Ils sont déployés dans de nombreux domaines et environnements, nous citons essentiellement leur utilisation dans les bases de données parallèles, l'enseignement à distance, les fichiers répartis, les applications de calculs distribués, et dans des applications tactiques comme les applications militaires, la gestion des catastrophes, les réseaux véhiculaires VANETs (Vehicular Ad hoc NETworks), etc.
Caractéristiques : un réseau ad hoc est différent d'un réseau doté d'une architecture fixe [9]. Les principales différences sont :

- La mobilité des nœuds provoque des modifications aléatoires de l'architecture du réseau.
- Les transmissions sans fil.
- Absence d'une infrastructure de base pour la gestion du réseau.
- Les nœuds sont alimentés par une source d'énergie autonome.
- La taille du réseau est illimitée.

L'une des contraintes majeures des réseaux ad hoc est l'absence d'une structure de contrôle de sécurité. Cette vulnérabilité permet d'augmenter le risque d'attaques sur le réseau. Un attaquant peut être interne ou externe au réseau [10]. Les mécanismes de sécurité déployés permettent de garantir les besoins habituels de sécurité [11, 12]

à savoir l'authentification, l'intégrité, la confidentialité, le contrôle d'accès, la non-répudiation, etc. Pour plus de détails sur les réseaux ad hoc, le lecteur peut consulter l'article de Loo et al [13] et le livre de Sarkar et al [14] ainsi que les travaux de [15, 16, 17].

1.2.2 Internet des objets

L'Internet des Objets (IdO ou IoT pour Internet of Things en anglais), certains lui préfèrent d'autres formulations comme objets connectés, désigne un ensemble d'objets physiques capables de collecter des données grâce à des capteurs et de les émettre vers des plateformes capables de les recueillir et de les analyser, ce qui constitue tout un écosystème. Une étude réalisée par Idate dans [18], en Octobre 2015, anticipe le nombre de 80 milliards d'objets connectés en 2020 tandis que Cisco dans son étude publiée dans [19] en Février 2015 envisage environ 50 milliards.

Caractéristiques : les principales caractéristiques des objets connectés sont :

- Un concentré de technologies pour former une application d'objets connectés.
- Des domaines d'application variés pour des besoins multiples.
- Considérée comme la technologie du futur (Web 3.0).
- Envisageable grâce au Big Data et aux bases de données No SQL et temps réel.
- Généralement les données sont très volumineuses, redondantes et déstructurées (sans relation entre elles).
- L'inter-connectivité des objets se fait par des antennes multifréquences de taille optimisée pouvant être gravée sur une puce.

Malgré des avancées considérables et parfois phénoménales, plusieurs problématiques demeurent d'actualité. Parmi elles, la problématique de sécurité où plusieurs objets font objet de vol ou de piratages, ainsi que le problème de standardisation avec le besoin d'un standard ouvert (comme TCP/IP pour l'internet). Leurs domaines d'application sont très nombreux. On peut citer les Smart Cities (villes intelligentes), Smart Home (Maisons intelligentes), en logistique (traçabilité), dans le domaine pharmaceutique (comme par exemple éviter les contrefaçons par l'utilisation de puces biodégradables), de la santé (dépistage), etc. Le lecteur peut se référer aux études faites par Li et al [20], Al-Fuqaha et al [21] et Ray [22] pour d'amples informations sur les objets connectés.

1.2.3 Internet des Objets industriels

L'avènement des réseaux sans fil intelligents, les capteurs à basse consommation d'énergie ainsi que les outils d'analyse du Big Data ont facilité le développement des

objets connectés industriels. L'association de ces technologies permet de placer d'innombrables capteurs intelligents et multifonctionnels partout et de les relier par des infrastructures de communication. Ainsi, il est désormais facile et possible de connaitre l'état, la position et la nature d'un objet qui peut être une machine, une pompe, un wagon, etc.

Caractéristiques : parmi les caractéristiques de ce genre de réseaux à ressources limitées, on peut citer :

- La priorité est donnée à la sécurité et à la fiabilité.
- Leur implantation nécessite une transition fluide entre les équipements neufs et anciens ainsi que pour le personnel.
- Les objets peuvent être des équipements de nature différente (hétérogènes).
- Le facteur temps est essentiel notamment dans des applications de production de biens et de disponibilité des données dans les délais impartis.
- Fonctionnement sans interruption pendant de nombreuses années.

Le domaine d'application est incontestablement le milieu industriel. D'autres challenges sont évoqués dans les travaux de Gungor et al [23] et Low et al [24].

1.2.4 Réseaux de capteurs sans fil

Les réseaux de capteurs sans fil constituent une technologie émergente à faible coût utilisée dans des applications stratégiques et sensibles comme la surveillance des zones de combats, le suivi temps réel de l'état des patients, la détection des phénomènes naturels, la sécurité alimentaire, les télécommunications, la robotique et dans des applications traditionnelles (automobile, aéronautiques, applications commerciales...etc). Les nœuds sont déployés soit aléatoirement ou d'une manière déterministe pour récolter des informations sur leur environnement et de les router par la suite vers la station de base.

Caractéristiques : un réseau de capteurs sans fil possède plusieurs caractéristiques [25] dont :

- Ressources limitées des noeuds capteurs en calcul, en mémoire et en énergie.
- Durée de vie limitée.
- Mode de communication direct ou en multi-sauts.
- Densité importante des noeuds capteurs qui peuvent atteindre des dizaines de millions pour certaines applications.
- Possibilité de découper le réseau en clusters et d'utiliser les noeuds capteurs comme calculateurs ou des agrégateurs.
- La coopération entre les nœuds capteurs pour les tâches complexes.

- Absence d'un identifiant global pour les noeuds capteurs.
- Deux modes de fonctionnement : « Un à plusieurs » où la station de base diffuse des informations aux différents noeuds capteurs ; et « Plusieurs à un » où les nœuds capteurs diffusent des informations à la station de base.

Les ressources physiques et énergétiques limitées des noeuds capteurs, l'absence de la sécurité physique et la nature vulnérable des communications sans fil sont, entre autres, des caractéristiques qui peuvent augmenter le niveau du risque d'attaques [26] sur le réseau et leurs vulnérabilités découlent de leurs propriétés. Des études sur les réseaux de capteurs sont menées par beaucoup de chercheurs de la communauté scientifique d'autant que ce domaine constitue l'un des domaines de recherche les plus actifs de ces dernières années. Le lecteur peut se référer aux travaux de Akyildiz et al [27], Rawat et al [28], Oliveira et al [29] pour plus de détails sur ce type de réseaux très en vogue ces dernières années.

1.2.5 Réseaux de capteurs multimédias

Les réseaux de capteurs multimédias sont des réseaux de capteurs dont les nœuds peuvent gérer et véhiculer des données non scalaires dites multimédias comme l'image, la vidéo et le son. La procédure d'acheminement des données dans un RcSF multimédia diffère d'un RcSF classique et ceci est dû au volume de données récoltées, traitées et acheminées, ce qui complique leur stockage.

Caractéristiques : les RcSF multimédias ont pratiquement les mêmes caractéristiques que les RcSF classiques, à la différence de :

- Les données récoltées et échangées sont non scalaires.
- La taille des données récoltées et traitées est très importante.
- La taille des données récoltées et traitées implique des besoins énormes en ressources notamment le stockage, la batterie et le microprocesseur.
- La lenteur des communications.
- Le besoin insistant pour l'agrégation des données.
- Des modules de caméras et de micros facilement intégrables.
- Le type de capteur utilisé détermine le type des données collectées.

Cette dernière caractéristique permet de classifier les RcSF multimédias en plusieurs catégories. Ainsi, si c'est uniquement un micro qui est utilisé, on parle de RcSF acoustique, si c'est une caméra donc c'est un Réseau de capteurs visuel sans fil, divisé en réseau de capteurs image sans fil et réseau de capteurs vidéo sans fil. Leurs domaines d'applications sont multiples. On cite le projet Smart Santander [30], pour la recherche de places de stationnement dans un parking (Smart Parking), le projet Safecast [31]

pour la détection des radiations chimiques, etc. Le lecteur peut consulter les études de Akyildiz et al [32, 33] et de Usman et al [34] pour plus de détails sur les RcSF multimédias.

1.2.6 Réseaux corporels sans fil

Un réseau de capteur corporel sans fil (ou WBAN en anglais pour Wireless Body Area Network) est un réseau constitué d'un ensemble de capteurs portables de taille minuscule implantés dans le corps humain afin de surveiller en continu des signes physiologiques comme la température, la fréquence cardiaque, la pression artérielle, etc. Les données obtenues sont ensuite envoyées soit à une équipe médicale pour un diagnostic en temps réel du patient, ou bien à un équipement émetteur d'alertes d'urgence, ou stockées dans une base de données pour une utilisation ultérieure.

Caractéristiques : les principales caractéristiques d'un réseau de capteur corporel sont :

- La taille des capteurs est miniaturisée.
- Le domaine d'utilisation est strictement médical.
- L'architecture d'un réseau de capteurs corporel est décomposée en trois tiers : communication intra-WBAN, communication inter-WBAN et communication extra-WBAN.
- La communication intra-WBAN se fait autour du corps humain, soit entre les noeuds capteurs eux-mêmes ou entre les noeuds capteurs capteurs et le point de collecte.
- La communication inter-WBAN est entre le point de collecte et les points d'accès.
- La communication extra-WBAN est entre le point d'accès et la destination finale (base de données, équipe médicale et/ou équipement d'alerte).

La contrainte, considérée comme une exigence principale, dans les réseaux de capteurs corporels est la sécurité des données récoltées et échangées. Le lecteur peut s'approfondir dans le sujet en consultant les travaux de Movassaghi et al [35] et Ghamari et al [36].

1.3 Caractéristiques communes et comparaisons

Les réseaux à ressources limitées sont caractérisés essentiellement par une faible alimentation en énergie où les nœuds qui les composent sont souvent alimentés par des batteries limitées en capacité, non rechargeables et irremplaçables dans la plupart des applications. Ce sont des réseaux implantés généralement pour des applications critiques, sensibles et stratégiques où les besoins et les exigences en sécurité sont très

importants. Dans la majorité des applications, les réseaux à ressources limitées n'ont pas d'architecture fixe au déploiement ou pendant leur utilisation. Les nœuds s'auto-organisent et créent des routes afin de remonter l'information depuis la source vers la destination en se basant généralement sur des modes de communication multi sauts. Le nombre de nœuds déployés pour certaines applications peut atteindre des dizaines de millions avec une taille illimitée du réseau. Le mode de communication se base sur des échanges d'informations par un média sans fil (Zigbee, Wi-Fi, Bluetooth, etc). La taille des nœuds varie de petite taille à miniaturisée. Les réseaux sont souvent homogènes. L'objectif premier est d'assurer un fonctionnement sans interruption durant plusieurs mois à plusieurs années. Pour la plupart, les données traitées sont scalaires et d'une taille petite (de l'ordre de quelques Kilo octets au plus). Lors du déploiement de ce type de réseaux, aucune importance n'est accordée pour l'Interface Homme Machine, IHM, qui peut être assurée uniquement via des LED ou des boutons. Le tableau 1.2 ci-dessous présente quelques caractéristiques communes et comparaisons des réseaux à ressources limitées vus dans cette partie.

TABLEAU 1.2 – Caractéristiques communes et comparaisons des réseaux à ressources limitées

Caractéristique	**Ad hoc**	**IoT**	**WBAN**	**RcSF multimédia**	**RcSF classique**
Densité	Moyenne	Forte	Faible	Faible	Forte
Déploiement	Grand public	Grand public	Corps humain	Grand public	Endroits difficilement accessibles
Taille des objets	Illimitée	Illimitée	Miniaturisée	Illimitée	Petite
Ressources	Acceptables	Limitées	Très limitées	Limitées	Limitées
Communication	Diffusion	Point à point	Point à point	Point à point	Point à point
Source d'énergie	Remplaçable	Peut-être remplaçable	Irremplaçable	Peut-être remplaçable	Irremplaçable
Remplacement des objets	Remplaçables	Peuvent-être remplaçables	Possible mais difficile	Remplaçables	Peuvent-être remplaçables
Rechargement de batterie	Possible	Possible	Impossible	Possible	Rarement possible
Mobilité	Oui	Oui	Oui	Oui	Oui
Sécurité	Nécessaire	Selon l'application	Très élevée	Très élevée	Selon l'application
Interférences	Moyennes	Moyennes	Rares	Faibles	Grandes
Redondance des informations	Peu	Forte	Très peu	Forte	Forte

1.4 Domaines d'application

Les domaines d'application des réseaux à ressources restreintes et limitées sont pour la plupart sensibles et stratégiques touchant un très vaste choix de domaines comme la santé, maison/bâtiment, cités intelligentes, industrie, militaire, agriculture, environnement, etc. Nous citerons ici quelques domaines jugés comme les plus intéressants.

1.4.1 E-Santé

On utilise les capteurs par exemple pour assurer un service d'aide aux patients, compatibles avec les réseaux de capteurs corporels. Un exemple d'application est l'utilisation de gélules multi-capteurs pour la transmission d'images de l'intérieur du corps humain sans recourir à la chirurgie [37]. Ils sont également efficaces pour la détection de comportements anormaux chez des personnes âgées ou handicapées comme les chutes ou les chocs.

1.4.2 Smart Cities

L'objectif des projets Smart Cities est de rendre les villes intelligentes et par conséquent offrir à ses habitants une qualité de vie agréable avec une consommation minimale de ressources. Pour cela, l'optimisation de ressources passe par une combinaison intelligente de différentes infrastructures aux différents niveaux hiérarchiques, allant du bâtiment à la ville en passant par le quartier. Les données récoltées par les objets installés grâce à des capteurs intégrés dans l'IoT sont utilisées pour améliorer le cadre de vie. Un exemple d'application serait d'exploiter les données de géolocalisation de chaque Smartphone pour avertir d'un bouchon sur la route.

1.4.3 Domotique

Autrement appelée maison intelligente ou maison connectée, la domotique représente l'usage de l'Internet des Objets et les autres réseaux à ressources limitées comme les réseaux de capteurs sans fil dans une maison. Ainsi, l'écosystème installé permettra d'optimiser la consommation d'énergie et d'eau, commander des appareils à distance comme la climatisation, les lampes, le téléviseur, le chauffage, la machine à laver, etc., ouvrir et fermer la porte du garage, mais aussi et surtout sécuriser la maison par l'installation de caméras et de capteurs de présence connectés.

1.4.4 Automobile 4.0

La voiture de demain serait autonome ! les centaines voire les milliers de capteurs intégrés dans un véhicule permettent de récolter des données et de les exploiter par des

entreprises. Comme exemple d'application, les données récoltées peuvent être exploitées par les compagnies d'assurance afin de proposer des offres préférentielles suivant la conduite de l'usager. Également, l'écran installé à l'intérieur du véhicule pourra être en mesure de proposer des publicités ou d'autres contenus selon, par exemple, le lieu visité, un embouteillage sur la route détecté, etc. Un exemple de voiture autonome est Google Car, déjà en circulation depuis 2015.

1.4.5 Industrie 4.0

Les réseaux de capteurs et l'Internet des Objets est une vraie et grande révolution. Ces deux types de réseaux sont appliqués dans le domaine industriel où ils permettent, entre autres, d'optimiser le processus logistique et la consommation d'énergie afin de réduire les coûts et les risques. Les capteurs intégrés, sur un emballage par exemple, permettent de tracer les déplacements des objets, s'ils ont subi des températures extrêmes, l'humidité, les chocs, etc. On peut citer comme exemple d'application, la maintenance prédictive, c'est-à-dire la capacité de prévoir et d'anticiper les pannes ou toute autre défaillance.

1.4.6 Objets connectés et marketing

La plupart des applications installées sur un Smartphone permettent de créer de nouveaux usages, notamment le service des publicités push envoyées à des clients potentiels. Parmi leurs usages, le paiement mobile en recevant directement sur son Smartphone une offre alléchante et régler instantanément et directement l'achat en deux clics ! Comme exemple d'application, à présent plusieurs centres commerciaux et grandes surfaces sont équipés d'objets connectés qui permettent de repérer des clients potentiels par la géolocalisation et de leur envoyer des offres commercialisées.

1.5 Concepts et définitions sur les réseaux de capteurs sans fil

Les réseaux à ressources limitées sont des réseaux autonomes couvrant un large champ d'applications. La majorité de leurs applications sont dédiés à des tâches précises où les dispositifs embarqués qui les composent ont des fonctionnalités très restreintes. Toutefois, d'autres réseaux, comme les réseaux de capteurs sans fil (RcSF), sont à usage général grâce aux capteurs multifonctionnels qui les composent. Un noeud capteur sans fil est un dispositif de taille minuscule, multifonctionnel et autonome, considéré comme un véritable système embarqué. Il est capable d'accomplir beaucoup de tâches parfois

dans des environnements hostiles. Un ensemble de noeuds capteurs s'auto-organisent pour former un réseau de capteurs sans fil (RcSF).

En tenant compte de leurs caractéristiques et de leur champ d'application varié, la conception d'un réseau de capteurs est très contraignante. Ainsi, leur déploiement dans des zones inaccessibles et hostiles, leur topologie dynamique et l'absence d'une infrastructure de base, entre autres, augmente le risque d'attaques sur le réseau, ce qui nécessite une protection renforcée et qui engendre une surconsommation d'énergie. Les besoins donc en sécurité et en énergie de chacune des familles de RcSF sont importants. Les solutions de sécurité existantes pour les réseaux classiques ne s'adaptent pas pour les RcSF en raison des caractéristiques de ces derniers. Dans ce chapitre, nous présenterons quelques définitions sur les RcSF, leur architecture et leurs contraintes de conception, puis nous aborderons les deux volets sécurité et énergie où nous détaillerons intrinsèquement quelques attaques et quelques besoins en sécurité et en énergie. Nous présenterons quelques solutions de sécurité adaptées aux RcSF, respectant à la fois les exigences habituelles de sécurité et les limites en puissance de calcul, de stockage et de ressources énergétiques des noeuds capteurs.

1.6 Quelques définitions

Nous présenterons dans cette section quelques définitions et concepts les plus utilisés dans les réseaux de capteurs.

1.6.1 Noeud Capteur

Un noeud capteur est un dispositif équipé principalement de quatre unités : i) unité de captage responsable de la collecte de données, ii) unité de traitement composée d'un microprocesseur et d'une mémoire de stockage ayant pour rôle d'effectuer des traitements sur les données captées, iii) unité de communication responsable des émission/réception de données et iv) unité d'alimentation composée d'une ou de plusieurs sources d'énergie comme une batterie et/ou un module solaire, responsable de gérer l'alimentation en énergie de tous les autres composants du noeud capteur. Notons que certains nœuds capteurs sont équipés par d'autres unités supplémentaires comme un système de localisation (un module GPS) et un mobilisateur pour assurer le mouvement du noeud capteur au besoin. La Figure 1.4 [27] présente les principaux composants d'un noeud capteur. Les données captées, comme la température, le taux d'humidité, le mouvement, la pression, etc., sont des grandeurs physiques converties en signaux numériques.

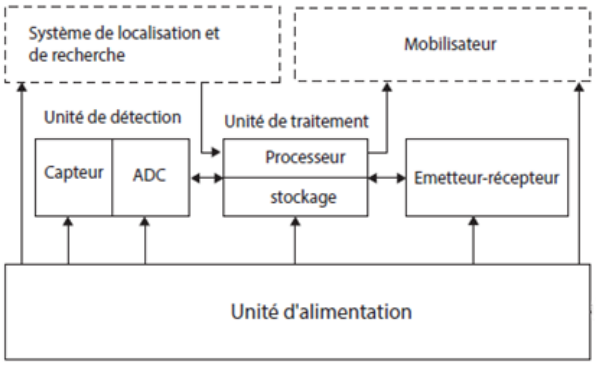

FIGURE 1.4 – Les principaux composants d'un noeud capteur

1.6.2 Capteur intelligent ou Smart Sensor

Le terme intelligent est désigné dans le domaine de l'industrie pour une sorte de capteurs n'ayant pas uniquement pour mission de fournir des mesures. Ainsi, un capteur intelligent permet également de traiter localement les données collectées ou reçues en lui intégrant de nouveaux modules électroniques additionnels et d'autres unités programmables [38]. Plus largement, le concept de capteur intelligent se décompose [39] en, en plus d'un capteur classique, un ou plusieurs conditionneurs spécifiques, un ou plusieurs transducteurs, un module intelligent interne permettant de traiter localement des données ainsi qu'une interface de communication. La Figure 1.5 montre un exemple d'un capteur de pression intelligent.

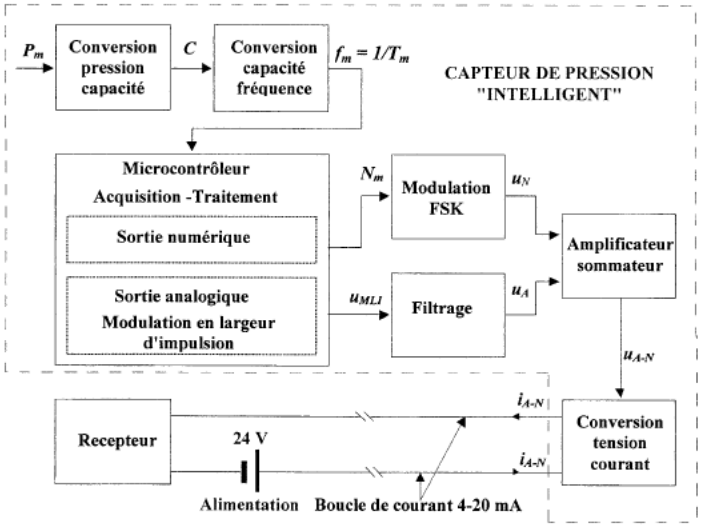

FIGURE 1.5 – Exemple d'un noeud capteur intelligent

1.6.3 Topologie plate

Dans une topologie plate, on considère que tous les nœuds sont égaux et possèdent le même rôle et les mêmes ressources. La Figure 1.6 montre un exemple d'une topologie

plate.

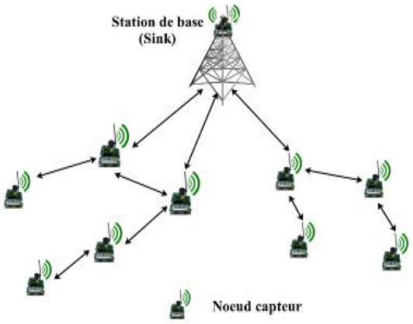

FIGURE 1.6 – Exemple d'une topologie plate

1.6.4 Topologie hiérarchique

Le principe du déploiement hiérarchique est de diviser le réseau en plusieurs sous-réseaux et attribuer aux nœuds plusieurs niveaux de responsabilités. Dans les réseaux de capteurs, la technique employée est le clustering où le réseau est partitionné en clusters. Chaque cluster possède son chef appelé Cluster Head et des membres ordinaires. La Figure 1.7 [40] présente le principe d'une topologie hiérarchique.

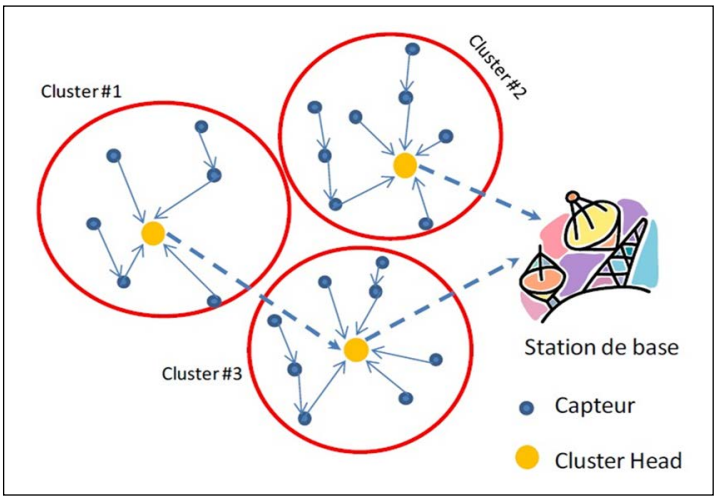

FIGURE 1.7 – Exemple d'une topologie hiérarchique

1.6.5 Norme IEEE 802.15.4

La norme de communication IEEE 802.15.4 permet de garantir la fiabilité des transmissions de données tout en respectant les limites des nœuds capteurs en énergie [41]. La couche IEEE 502.15.4 est composée de deux couches : la couche physique et la couche liaison de données. Elle est caractérisée par une puissance de transmission à une distance maximale de 100 mètres et un débit au choix de 20, 40 et 250 Kbps. Le

Tableau 1.3 présente les caractéristiques principales de la norme IEEE 802.15.4. L'auteur de [42] présente un résumé complet sur les caractéristiques principales de la norme 802.15.4.

TABLEAU 1.3 – Caractéristiques principales de la norme 802.15.4

Fréquence	Low-Band (BPSK Modulation)	868 MHz	1 Canal	20 Kb/s
		915 MHz	10 Canaux	40 Kb/s
	High-Band (O-QPSK Modulation)	2.4 GHz	16 canaux	250 Kb/s
Accès au canal	CSMA-CA ou Slotted CSMA-CA			
Portée	Moyenne : 10 à 20 mètres	Maximale : 100 mètres		
Mode d'adressage	Short 8-bit ou 64-bit IEEE			
Consommation d'énergie	Très faible			

1.6.6 Protocole Zigbee

C'est un protocole de communication multi-sauts développé au niveau de la couche réseau en se basant sur les couches physiques et liaison de données de la norme 802.15.4 (voir la Figure 1.8 [43]). Pour plus d'informations sur les spécifications et l'architecture du protocole Zigbee, le lecteur peut se référer au document de Somani et Patel [44].

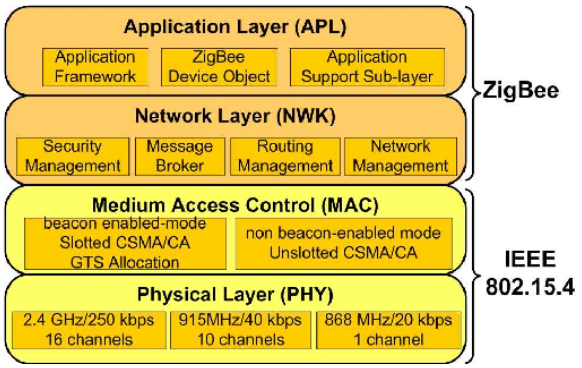

FIGURE 1.8 – Pile de protocoles 802.15.4/Zigbee

1.7 Architecture d'un RcSF

Dans un réseau de capteurs sans fil, un nœud diffère d'un autre nœud selon le type de déploiement du réseau, sa topologie et son application. Un nœud ordinaire est un nœud équipé de toutes les unités d'un noeud capteur : traitement, transmissions, énergie, capture et quelques autres unités optionnelles comme le système de géolocalisation, un

système mobile pour les déplacements, un module solaire comme générateur d'énergie, etc. Un nœud source est un nœud destiné uniquement à la capture de données et leur transmission vers le prochain saut. Le nœud puits ou Sink Node est un nœud ordinaire équipé d'une seconde unité de communication, comme le Wi-Fi, 3G, 4G, WiMax, afin de relayer les données reçues vers un autre réseau ou vers l'utilisateur final.

Les nœuds d'un réseau de capteurs sont déployés arbitrairement dans un environnement sans infrastructures et sans aucune information sur la topologie globale du réseau. Généralement, une phase d'initialisation est toujours nécessaire après un déploiement. Durant cette phase, les nœuds établissent une infrastructure de communication qui leur permet d'interagir entre eux et la station de base (appelée également le puits) afin de transmettre les données captées jusqu'à la destination. Le principe est donné par la Figure 1.9. Il existe deux types d'architectures pour les RcSF : réseaux en topologie plate et réseaux en topologie hiérarchique.

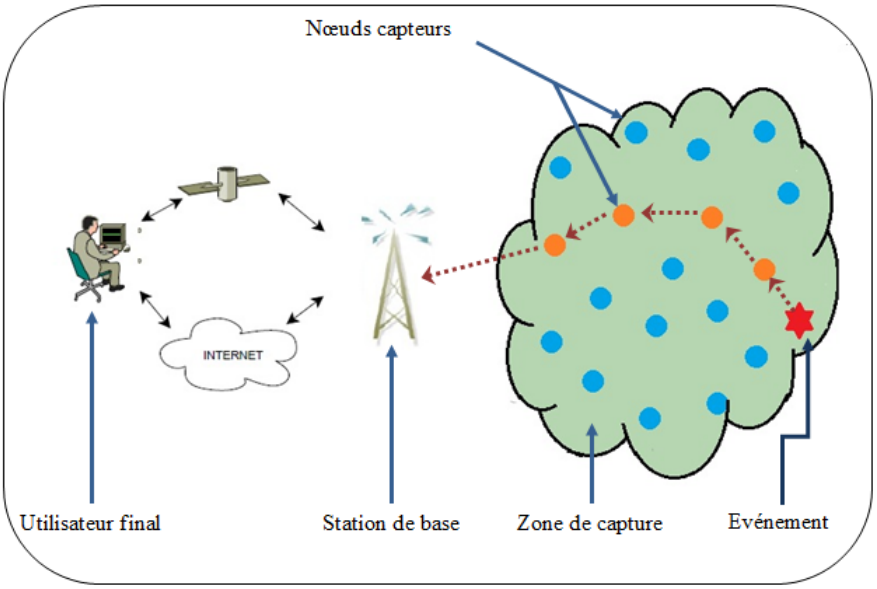

FIGURE 1.9 – Exemple d'un réseau de capteurs sans fil

Un réseau de capteurs sans fil plat est un réseau homogène composé d'un nombre très important de nœuds communicants via un mode de communication sans fil multi-sauts. Ces deux caractéristiques rendent le passage à l'échelle très critique et une consommation de ressources importante. Tous les nœuds déployés ont les mêmes ressources notamment en énergie, mémoire et puissance de calcul.

Un réseau de capteurs sans fil hiérarchique est un réseau hétérogène où les nœuds n'ont pas nécessairement les mêmes rôles et les mêmes ressources. Certains nœuds sont considérés comme puissants et sont déployés pour des tâches complexes et énergivores, déchargeant ainsi les nœuds ordinaires et limités en ressources de ce type de tâches. Les nœuds ordinaires sont déployés et utilisés pour des tâches de base comme la capture.

Il existe différentes familles de réseaux de capteurs, la Figure 1.10 présente les principales d'entre elles.

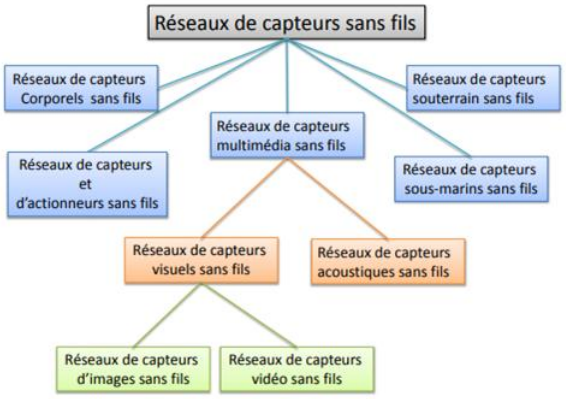

FIGURE 1.10 – Les différentes familles de réseaux de capteurs sans fil

1.8 Applications des RcSF

Les réseaux de capteurs sans fil sont déployés et utilisés dans divers domaines d'application dû essentiellement au large choix de capteurs disponibles : thermiques, sismiques, visuels, infrarouges, acoustiques, radars, vibrations, ultrasoniques, etc. Ceci les rend capables d'accomplir plusieurs tâches et phénomènes ambiants tels que la mesure de la température, de l'humidité, la pression, le taux de bruit, la vitesse, la direction, le son ainsi que le contrôle du mouvement des objets, etc. Un noeud capteur autonome peut fonctionner sans interruption permettant ainsi une surveillance continue de son environnement. L'autonomie des noeuds capteurs et des réseaux de capteurs sans fil, notamment en énergie (utilisation d'une batterie comme source d'énergie) et en communications (utilisation des médiums de communication sans fil) constitue un avantage pour beaucoup d'applications dans divers domaines comme le domaine de la santé, l'environnement, militaire, commerciale, agriculture, domotique, etc. Beaucoup d'autres applications peuvent intéresser le lecteur et bien détaillées dans les travaux de [29, 45, 46, 47, 48, 49].

1.9 Contraintes de conception

Les limites sur les ressources des nœuds qui composent un réseau de capteurs sans fil, la nature vulnérable des communications, l'environnement de déploiement, le type d'applications, les besoins en sécurité, le grand nombre de nœuds et l'absence d'une

infrastructure physique centrale sont entre autres des contraintes qui rendent le déploiement de ce type de réseaux très difficile et parfois problématique. Ajoutons à cela :

— Contraintes matérielles : les noeuds capteurs sont des composants de très petites tailles munis de capacités restreintes en ressources notamment énergétiques, de calcul et de mémoire.

— Le challenge de sécurité : la sécurité est un domaine très important pour les réseaux de capteurs sans fil surtout que la majorité des applications sont sensibles et stratégiques. La mise en place d'une stratégie de sécurité dépend de plusieurs facteurs : le type d'application déployée, la puissance de calcul des noeuds capteurs, leur niveau d'énergie, la contrainte temps réel, etc.

— Défis de conservation d'énergie : les nœuds capteurs sont dans la majorité alimentés par des batteries inamovibles, non rechargeables et difficilement remplaçables voire parfois impossible à remplacer dans certaines applications déployées dans des zones hostiles et inaccessibles. Toutes les tâches effectuées par un noeud capteur requièrent l'optimisation de l'énergie, surtout pour des fonctions importantes comme le routage et la sécurité des communications.

— Une topologie dynamique : Les RcSF sont déployés aléatoirement sur un champ à surveiller sans aucune information au préalable sur leur déploiement. Les nœuds s'auto-organisent par la suite pour permettre à chaque nœud de connaitre sa situation dans le réseau. Cette organisation permet de créer un champ de surveillance et des routes pour transporter l'information recueillie par chaque nœud vers la destination. En cas d'absence d'un nœud pour différentes raison comme le changement de position ou une extinction suite à l'épuisement de ses ressources énergétiques, un autre nœud doit assurer ses fonctions afin de maintenir le fonctionnement normal du réseau.

Les différentes contraintes de conception d'un RcSF sont résumées par la Figure 1.11 ci-dessous :

1.10 Quelques attaques sur les RcSF

La vulnérabilité majeure d'un réseau de capteurs est l'utilisation d'une communication sans fil pour l'échange de données. Plusieurs attaques sur le médium de communication sans fil existent, elles peuvent nuire aux protocoles de communication et de sécurité afin de perturber les communications, altérer les données ou épuiser les ressources physiques des nœuds. Nous citons dans cette section quelques types d'attaques :

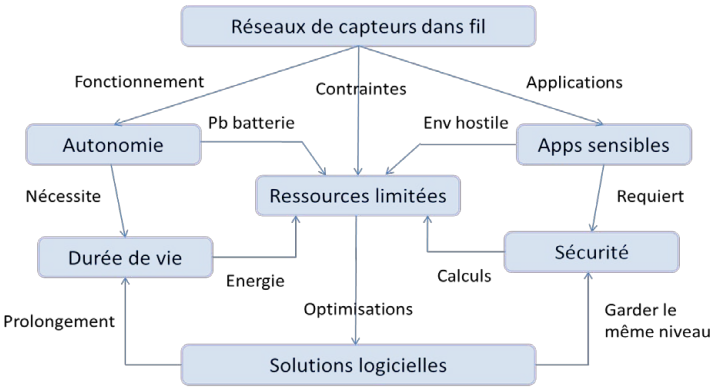

FIGURE 1.11 – La relation entre les différentes contraintes des RcSF

1.10.1 Attaques passives/Actives

Une attaque passive est une attaque "silencieuse" qui vise uniquement à écouter le canal de communication dans le but de récolter les données qui y circulent sans les altérer. Ces données seront ensuite utilisées pour fabriquer des attaques actives visant à insérer de nouvelles données, modifier ou altérer les données légitimes du réseau [50, 51].

1.10.2 Analyse du trafic

C'est une attaque qui se base sur l'écoute passive du canal afin de récupérer la destination des paquets de données envoyés. L'objectif de cette attaque est d'étudier la topologie du réseau et d'identifier les nœuds importants comme les Clusters-Head, les nœuds agrégateurs, les nœuds calculateurs, les nœuds maitres, etc. Cette technique permet de choisir efficacement le type et la cible de l'attaque.

1.10.3 Attaques internes

Une attaque interne est une attaque amie. Elle survient quand un nœud du réseau, capturé, reprogrammé puis réinséré dans le réseau, participe directement à une attaque visant son propre réseau.

1.10.4 Brouillage radio

Cette attaque, connue sous le terme anglophone radio jamming, vise à perturber les communications en plaçant quelques nœuds malicieux dans la zone de communication. Le rôle des nœuds intrus est de brouiller le signal et d'inonder le réseau (connue sous le terme anglophone flooding) avec des messages inutiles entrainant sa saturation. C'est un type d'attaques de DoS pour déni de services ou Denial of Service [52].

1.10.5 Inondation

Le but est de saturer le réseau en envoyant des messages de type hello inutilement et de façon ininterrompue. En effet, quand un nœud légitime est réveillé ou inséré sur le réseau, il envoie périodiquement un message hello aux autres nœuds pour se manifester. Un nœud d'une puissance radio plus élevée, lui permettant d'atteindre un nombre très important de nœuds, exploite ce principe pour lancer sans cesse des messages de type hello. Tous les destinataires de ce message essayent de lui répondre pour lancer la procédure d'intégration. Par conséquent, ces nœuds consomment une partie de leur énergie et en même temps, ils deviennent indisponibles car ils seront très occupés à répondre aux messages inutiles d'intégration.

1.10.6 Usurpation d'identité

On vise dans ce type d'attaque l'authenticité des entités du réseau. Un nœud malicieux se déguise en une entité légitime du réseau en utilisant son identité, ce qui lui permettra de récupérer des informations précieuses circulant sur le réseau.

Beaucoup d'autres attaques existent dans la littérature et que nous ne détaillerons pas dans ce document. Nous citons [53, 54] : l'attaque Blackhole [55], Sinkhole [56], Wormhole [57], Sybil [58]. Le tableau 1.4 résume certains types d'attaques sur certains services importants des réseaux de capteurs sans fil.

TABLEAU 1.4 – Quelques types d'attaques sur les RcSF

Service visé	Couche concernée	Type d'attaque
Collection de données	Application	- Analyse du trafic - Déni de service (DoS) - Ecoute clandestine (Eavesdropping) [59] - Attaques internes
Perturbation du trafic	Physique	- Jamming - Brouillage radio
	Liaison	- Collision [60]
	Réseau	- Blackhole - Sinkhole - Wormhole
	Transport	- Flooding
Epuisement de ressources	Liaison	- Empêcher la mise en veille [61] - Technique d'interrogation [62]
	Transport	- Désynchronisation [63]
Agrégation de données	Réseau	- Sybil - Blackhole

1.11 Exigences en sécurité dans un RcSF

Les vulnérabilités des réseaux de capteurs sans fil découlent de leur nature, des caractéristiques propres aux nœuds capteurs et du médium de communication. Nous distinguons deux types de vulnérabilités : physiques et technologiques. Les vulnérabilités physiques sont liées principalement à la nature de l'environnement de déploiement où la majorité des RcSF sont déployés dans des zones hostiles et souvent accessibles à l'ennemi. Un attaquant peut facilement récupérer un ou plusieurs nœuds déployés pour les analyser et formuler une attaque. Les vulnérabilités technologiques découlent essentiellement de la technologie sans fil sous-jacente. Ainsi, plusieurs attaques connues [64, 65, 66] sur les médiums sans fil sont redoutables.

Les principaux objectifs de sécurité pour les réseaux de capteurs sans fil sont la disponibilité, l'intégrité, la confidentialité, la fraicheur de données, mais également l'authentification des utilisateurs, la non répudiation et le contrôle d'accès. Afin d'assurer un fonctionnement normal du réseau et une meilleure protection des données échangées, plusieurs mécanismes sont mis en place, soit avant ou après le déploiement, comme la détection d'intrusion, la gestion de clés, la tolérance aux intrusions, prévention contre le déni de service, les primitives cryptographiques, etc. Dans cette thèse, nous allons nous intéresser uniquement à la cryptographie, considérée comme un mécanisme préventif qui permet aux nœuds capteurs de se prémunir des écoutes, garantissant la confidentialité, l'intégrité, l'authenticité et la fraicheur des données.

1.12 Solutions de sécurité pour les RcSF

Dans cette section, nous allons proposer quelques solutions de sécurité convenables aux réseaux à ressources limitées et restreintes. A cause de la diversité des attaques et les limites sur les ressources des nœuds, il est difficile de proposer une solution globale pour tous les types d'attaques. Les solutions qu'on propose dans ce qui suit sont conçues pour respecter les limites des noeuds capteurs, notamment la puissance de calcul, le volume de stockage, la taille des paquets échangés et par conséquent la consommation d'énergie.

1.12.1 Clé d'authentification dynamique

Le principe est de renouveler les clés périodiquement en les régénérant et les distribuant par la station de base [67]. Cette nouvelle clé est utilisée par les nœuds pour prouver leur appartenance au réseau. Cette solution est efficace contre les intrusions et les nœuds malicieux. Cependant, elle nécessite une distribution sécurisée et fiable des clés. Le problème du type d'attaque Man in the Middle se pose sérieusement et les

nœuds du réseau ne peuvent pas commencer à communiquer tant que l'opération de renouvellement n'est pas complètement terminée. Toute forme de perturbation ralentit l'opération. Un autre inconvénient concerne l'insertion des nouveaux nœuds car on ne connait pas au préalable la nouvelle clé à utiliser.

1.12.2 Réseaux de confiance

L'idée est d'évaluer la réputation de chaque nœud. Un indice de réputation est distribué par chaque nœud à ses voisins [67]. A partir de là, chaque nœud surveille ses voisins et à chaque preuve de confiance d'un voisin on augmente son indice de réputation. Au cas contraire, on le diminue. Lors du routage, on sélectionne les nœuds ayant l'indice de réputation le plus élevé. L'inconvénient de cette solution est la difficulté de détecter les attaques passives car on ne peut pas savoir si un nœud avec un indice élevé communique correctement ses informations et que ses données échangées sont parfaitement chiffrées.

1.12.3 Sténographie

Cette technique consiste à cacher les informations sensibles dans des paquets de contrôle ou dans des paquets de données en exploitant certains champs ou octets non utilisés. Un attaquant ne pourra pas connaitre où sont cachées ses informations. L'inconvénient, si cet emplacement est dévoilé, toutes les informations cachées peuvent être aisément divulguées.

1.12.4 Cryptographie

Elle est considérée comme la technique de sécurité la plus sûre. L'idée est d'utiliser des méthodes mathématiques (des problèmes connus appartenant à la classe des problèmes NP-Complet ou NP-Difficile) pour transformer un message en clair en un autre message crypté qui ne peut être lu que par une entité autorisée et légitime. La cryptographie assure la confidentialité, l'intégrité et l'authenticité. L'inconvénient de la cryptographie est qu'elle nécessite des calculs complexes et un volume de données important et un accès rapide en mémoire. Actuellement, les travaux menés par les chercheurs du domaine s'orientent sur la recherche de solutions qui permettent d'optimiser les calculs et de réduire les quantités de données utilisées pour adapter ces protocoles et algorithmes cryptographiques aux réseaux de capteurs sans fil.

1.13 La cryptographie dans les RcSF

Les contraintes physiques des réseaux de capteurs sans fil (RcSF) constituent un problème de taille pour le développement d'architectures de sécurité complètes et robustes respectant les limites matérielles des nœuds capteurs et les caractéristiques des RcSF. La plupart des attaques sur la couche réseau présentées à la section 1.10 de ce document se basent sur l'écoute passive des communications afin de collecter un maximum d'informations, puis formuler des attaques actives visant à perturber les communications et à altérer les données échangées. Plusieurs solutions sont proposées dans la littérature pour contrecarrer ce type d'attaques. Dans ce document, nous nous intéresserons aux solutions cryptographiques.

1.13.1 Définition

La cryptographie est un terme générique qui désigne l'ensemble des techniques permettant de chiffrer un texte et de le rendre inintelligible sans aucune action spécifique. Elle est basée essentiellement sur l'arithmétique. Les opérations de chiffrement, c'est-à-dire la transformation du texte en clair en un texte chiffré, et de déchiffrement, c'est-à-dire retrouver le texte en clair à partir du texte chiffré, reposent sur l'utilisation de clés comme montré par la Figure 1.12. Un texte en clair est alors transformé en une succession de caractères incompréhensibles par n'importe quelle entité illégitime du réseau. Seul le destinataire peut déchiffrer le texte en utilisant le secret de la trappe (une clé de déchiffrement).

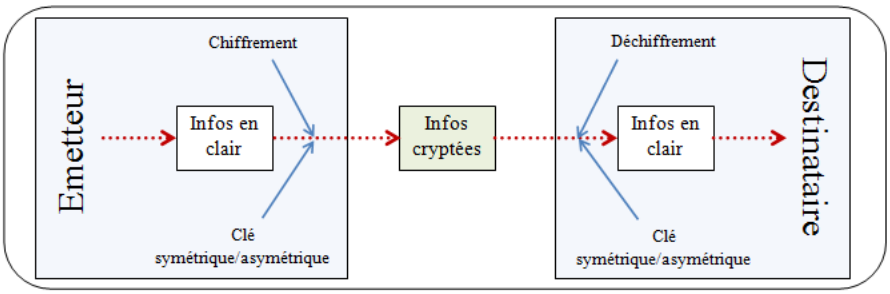

FIGURE 1.12 – Principe global d'un système cryptographique

Pour assurer un système cryptographique fiable, deux nœuds communicants doivent choisir soigneusement leur clé de chiffrement/déchiffrement et doivent appliquer les principes suivants :

- La sécurité doit se reposer sur le secret de la clé et non pas sur la sécurité de l'algorithme.
- Le déchiffrement sans connaissance préalable de la clé doit être impossible en un temps raisonnable.

- Calculer la clé à partir du texte en clair et du texte chiffré doit être impossible en un temps raisonnable.

1.13.2 Outils de la cryptographie

Les outils cryptographiques sont un ensemble de techniques, méthodes et algorithmes utilisant des clés afin de garantir un fonctionnement sûr du cryptosystème par la protection des données échangées et l'authentification des utilisateurs. Nous présentons dans cette section quelques outils indispensables à la sécurité des échanges de données.

Chiffrement/Déchiffrement

Le chiffrement de données permet de garantir la confidentialité des données avec des clés identiques dans le cas des systèmes de chiffrement symétrique ou bien une paire de clés privée/publique dans les systèmes de chiffrement asymétriques.

— Chiffrement symétrique : les nœuds communicants utilisent une même clé pour le chiffrement et le déchiffrement des données, comme montré sur la Figure 1.13. On distingue deux types de chiffrement symétrique : chiffrement par flots où les données sont fractionnées en bit à bit, et le chiffrement par blocs où les données sont fractionnées en blocs de tailles fixes. Les algorithmes de chiffrement par blocs sont les plus utilisés dans les systèmes symétriques. Les données sont décomposées en blocs de tailles généralement égales, comprises entre 64 et 512 bits et chiffrés successivement selon différents modes : ECB, CBC, OFB, CTR, etc.

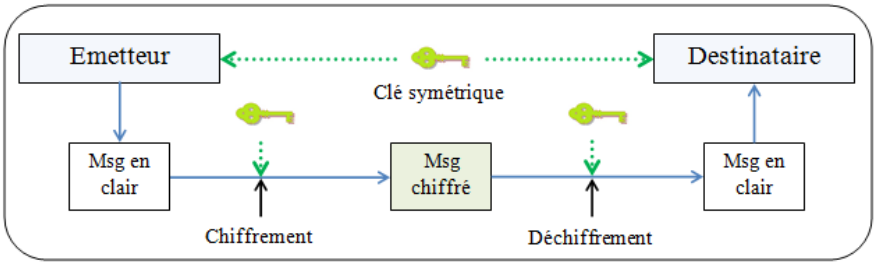

FIGURE 1.13 – Principe du chiffrement symétrique

Avantages : les algorithmes de chiffrement symétriques ne nécessitent aucune opération mathématique complexe pour les calculs durant les phases de chiffrement et de déchiffrement. Les clés sont relativement courtes (128 à 256 bits).

Inconvénients : la difficulté du système à distribuer les clés symétriques aux nœuds d'un réseau dont la topologie est dynamique ou aléatoire et la difficulté de gérer les $n(n-1)/2$ clés symétriques d'un réseau de n nœuds. Certaines propriétés sont difficiles à réaliser (exemple : la signature).

Exemples d'algorithmes symétriques : AES (Advanced Encryption Standard) [68], DES (Data Encryption Standard) [69], RC4 (Rivest Cipher 4) [70].

— Chiffrement asymétrique : les algorithmes de chiffrement asymétrique utilisent une paire de clés publique/privée pour le chiffrement/déchiffrement. Ainsi, la clé publique est générée à partir de la clé privée et diffusée à tous les nœuds émetteurs du réseau, tandis que la clé privée est maintenue uniquement par le nœud récepteur. La clé publique sert à chiffrer les données et la clé privée correspondante à les déchiffrer. Le principe est donné par la Figure 1.14.

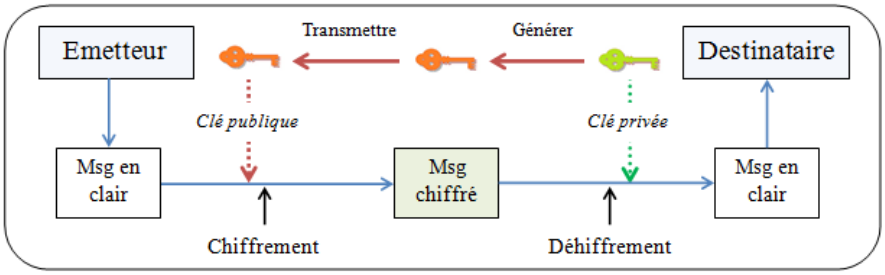

FIGURE 1.14 – Principe du chiffrement asymétrique

Avantages : les algorithmes de chiffrement asymétriques ont plusieurs avantages, on cite : permettent la scalabilité, efficaces contre la capture physiques des nœuds, la facilité de la distribution des clés, une signature facile des messages, nombre réduit de clés à distribuer.

Inconvénients : parmi les inconvénients des algorithmes asymétriques, on cite : la taille des clés, calculs complexes et trop lents, gourmands en ressources physiques et énergétiques, vulnérables aux attaques de type déni de service.

Exemples d'algorithmes asymétriques : RSA (Rivest Shamir Adleman) [71], ECC (Elliptic Curve Cryptography) [1, 2].

Signature numérique

C'est un système cryptographique qui repose sur des clés asymétriques pour assurer la non-répudiation de la source. L'émetteur produit une signature digitale et signe son message avec sa clé privée. Le récepteur tente de déchiffrer le message reçu en utilisant sa clé publique. Si le message est déchiffré, l'émetteur ne pourra plus nier l'émission de ces données par la suite. Le principe étant décrit par la Figure 1.15.

1.13.3 Quelques attaques cryptographiques

Beaucoup d'attaques visant à reconstruire le texte initial à l'aide de méthodes mathématiques sont utilisées par les attaquants. La technique de reconstruction du texte initial est appelé la cryptanalyse. On distingue plusieurs méthodes de cryptanalyse, on cite :

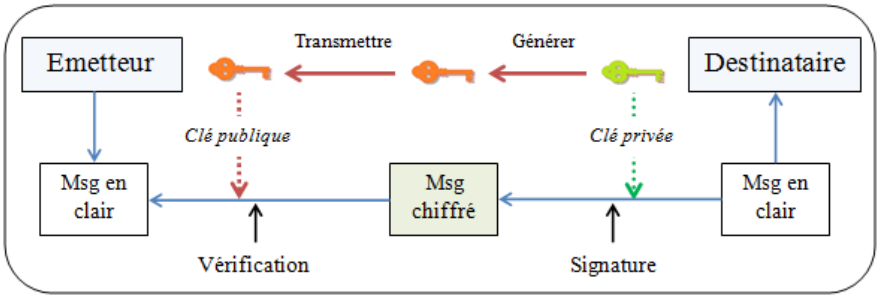

FIGURE 1.15 – Principe de la signature numérique

- Attaque sur un texte chiffré choisi : elle consiste à retrouver la clé de déchiffrement à l'aide de textes chiffrés générés à partir de textes en clair.
- Attaque sur un texte clair choisi : elle consiste à retrouver la clé de déchiffrement à l'aide de textes en clair générés à partir de textes chiffrés.
- Attaque adaptative sur un texte en clair choisi : c'est un cas particulier de l'attaque précédente où un attaquant peut adapter le choix des textes en clair en fonction des textes chiffrés déjà essayés.
- Attaque sur texte chiffré seulement : elle consiste à retrouver la clé de déchiffrement à partir de plusieurs textes chiffrés.
- Attaque sur texte clair connu : elle consiste à retrouver la clé de déchiffrement à l'aide de textes chiffrés correspondants à leurs textes en clair connus.
- Attaque sur une clé choisie : elle consiste à retrouver la clé de déchiffrement en utilisant quelques relations entre différentes clés déjà connues.

1.14 La consommation d'énergie dans un RcSF

Les nœuds capteurs sont des composants microélectroniques équipés par des sources d'énergie limitées. La batterie est la source principale d'énergie d'un noeud capteur, elle a une capacité moyenne de 2 Ampère-Heure et une tension d'alimentation entre 1,2 Volts et 5,5 Volts (pour la plupart des noeuds capteurs commercialisés actuellement). Cette capacité demeure insuffisante pour faire fonctionner un noeud capteur pour des durées longues allant de quelques mois à quelques années. De plus, les batteries des nœuds capteurs sont souvent irremplaçables et non rechargeables dans la majorité des applications. De ce fait, la consommation d'énergie représente une métrique de performance importante. L'énergie d'un noeud capteur est utilisée essentiellement pour la capture, les communications et le traitement de données.

L'unité d'énergie d'un noeud capteur est un dispositif essentiel. Son rôle est de répartir l'énergie disponible aux autres modules, comme l'antenne, le processeur, l'unité

de capture, le mobilisateur, le GPS, etc., d'une manière optimale. L'optimisation énergétique est un défi important lancé par la communauté scientifique. Elle dépend de trois facteurs essentiels : i) le type du module électronique (certains modules sont énergivores), ii) les types de communications effectuées et iii) la topologie globale du réseau. Ainsi beaucoup de travaux de recherche [72, 73] ont été réalisés afin de réduire la consommation d'énergie d'un noeud capteur. Les plus importants ont porté sur les axes suivants :

— Optimisation de la consommation radio : l'idée a été lancée par Akyildiz dans ses premiers travaux dans [74] en proposant une méthode pour réduire la taille des paquets (Payload) échangés, puis d'autres travaux plus récents ont traité d'autres problématiques comme la gestion efficace de la mise en veille du module radio [75] et l'optimisation des canaux dans les systèmes de communication Half-Duplex [76], etc.

— Routage efficace de l'information : un routage efficace permet de minimiser les risques de collision des données, le choix optimal de routes selon plusieurs métriques comme la distance, l'énergie des noeuds capteurs disponible ou l'énergie globale de la route choisie. Dans [77], une étude complète sur les protocoles de routage économes en énergie est effectuée.

— Réduction de la complexité des calculs : les calculs dans les RcSF sont énergivores notamment pour les calculs cryptographiques. Ainsi, la complexité doit être prise en considération lors de l'implémentation d'un protocole de sécurité. Le challenge est de trouver un compromis entre le maintien du niveau de sécurité exigé et la réduction de la complexité des méthodes utilisées. Ce compromis peut être assuré par l'emploi de techniques mathématiques souples garantissant le même niveau de sécurité. Dans cette thèse, nous avons travaillé sur ce dernier point où nous avons proposé plusieurs solutions permettant la réduction de la complexité des calculs cryptographiques tout en gardant le niveau de sécurité exigé, que nous détaillerons dans la partie contribution de ce document.

1.15 Conclusion

L'apparition des systèmes embarqués a révolutionné les systèmes informatiques traditionnels. Un système embarqué est composé de dispositifs miniaturisés et autonomes capables d'effectuer des opérations complexes. Ces dispositifs sont souvent équipés de ressources limitées notamment en puissance de calcul, de stockage mais surtout en énergie. De plus, l'une des contraintes majeures des systèmes embarqués est la sécurité. Les systèmes embarqués composent généralement les entités des réseaux sans fils appelés réseaux à ressources restreintes.

Les réseaux à ressources restreintes ou limitées sont des réseaux sans fil multifonctionnels assurant un fonctionnement autonome à l'application dédiée. Leurs limites découlent des limites des dispositifs embarqués qui les composent. Sauf que dans la pratique, leur intérêt dépasse leurs contraintes, en particulier pour des applications sensibles dans des domaines stratégiques comme la santé, la sécurité, le domaine militaire, etc.

On distingue plusieurs types de réseaux à ressources restreintes, certains sont dédiés à des tâches spécifiques et d'autres sont à usage général. Dans ce premier chapitre, nous avons mis l'accent sur les caractéristiques de chacun des réseaux étudiés. Dans le prochain chapitre, nous étudierons les réseaux de capteurs sans fil (RcSF), un cas particulier des réseaux ad hoc en pleine expansion, déployés pour un usage général pour des applications qui peuvent toucher des centaines de domaines.

Les caractéristiques intéressantes des réseaux de capteurs sans fil, comme le coût faible des noeuds capteurs, leur taille réduite, la large gamme des capteurs, le media de communication sans fil, etc. ont permis leur émergence et leur utilisation dans un champ d'application vaste et varié. Cependant, certaines contraintes de conception, comme la durée de vie limitée des noeuds capteurs et les besoins importants en sécurité, ont rendues leur développement difficile. En effet, les limites des nœuds capteurs en ressources énergétiques, en capacités mémoire et en puissance de calcul rendent l'application des algorithmes habituels de sécurité impossible.

Plusieurs solutions adaptées au contexte des RcSF ont été proposées afin de contrer les attaques les plus redoutables comme les attaques actives qui visent à perturber le fonctionnement global du réseau et à altérer les données y circulant. Parmi les solutions robustes et adaptées aux RcSF se distinguent les systèmes de cryptographie asymétriques. Dans le chapitre suivant, nous étudierons l'une des plus puissantes techniques de chiffrement asymétrique, la cryptographie des courbes elliptiques (ECC), à la fois sûre, respectant les contraintes physiques des nœuds capteurs, garantissant les exigences habituelles de sécurité comme la confidentialité, l'intégrité, la non-répudiation, l'authenticité et la disponibilité.

Deuxième partie

État de l'art sur la cryptographie des courbes elliptiques

Chapitre 2
Généralités sur la cryptographie des courbes elliptiques

La cryptographie est toujours considérée comme l'un des mécanismes dominants pour la protection des données. Son principe est de rendre une information illisible à un utilisateur illégitime. Elle emploie un ensemble de méthodes mathématiques, souvent complexes, ce qui pose un problème à leur application dans des systèmes à ressources restreintes comme les réseaux de capteurs sans fil. Les solutions cryptographiques symétriques ne s'adaptent pas bien aux RcSF en raison de leur nature de déploiement. Ainsi, les solutions asymétriques conviennent mieux à ce type de réseaux mais la complexité des opérations arithmétiques rendent leur application problématique. La longueur des clés dans les systèmes traditionnels, comme RSA (Rivest, Shamir, Adleman) [71], nécessite des calculs encore plus complexes et plus gourmands en ressources. Une alternative à RSA est ECC (Elliptic Curve Cryptography) [1, 2], utilisant des clés plus courtes pour un même niveau de sécurité [78]. La cryptographie des courbes elliptiques repose sur le problème du logarithme discret elliptique.

Les courbes elliptiques est un sujet très à la mode en mathématiques. La théorie de ces objets mathématiques complexes est liée à la géométrie algébrique [79] et à la théorie des nombres [80]. Elles sont utilisées dans la cryptographie (d'où l'appellation Elliptic Curve Cryptography ECC) pour sécuriser les communications avec des clés supposées inviolables. Les courbes elliptiques interviennent dans différents domaines : en théorie des nombres [80] pour prouver certains théorèmes comme celui de Fermat [81] et en cryptologie, entre autres. En cryptologie, on utilise les courbes elliptiques dans les opérations de chiffrement/déchiffrement, l'échange de clés et la signature numérique. La cryptographie basée sur les courbes elliptiques (ECC) utilise des primitives de cryptographie asymétrique considérées actuellement comme un standard.

Dans ce chapitre, nous présenterons d'abord quelques préliminaires sur la géométrie algébrique, puis quelques définitions sur les courbes elliptiques avant d'étudier la cryptographie des courbes elliptiques et nous présenterons quelques cryptosystèmes basés sur les ECC.

2.1 Préliminaires

Dans cette section et afin de comprendre mieux les courbes elliptiques, nous donnerons quelques notions mathématiques sur lesquelles se base une courbe elliptique.

2.1.1 Loi de groupe

Un groupe est un couple formé d'un ensemble E et d'une loi de composition interne $(\cdot)$ qui combine deux éléments x et y de E pour obtenir un troisième élément $(x \cdot y)$ tel que $(x, y) \to (x.y)$ sur E satisfaisant les quatre axiomes suivants :

- $\forall (x, y) \in E \mid x \cdot y \in E$ (Fermeture)
- $\forall (x, y) \in E \mid (x \cdot y) \cdot c = x \cdot (y \cdot c)$ (Associativité)
- $\exists e \in E \mid x \cdot e = e \cdot x = x$ (Elément neutre)
- $\forall x \in E, \exists y \in E \mid x \cdot y = y \cdot x = e$ (Symétrie)

2.1.2 Loi de groupe abélien

Si la loi de groupe est commutative alors le groupe est appelé groupe abélien ou groupe commutatif. On utilise alors la notation suivante :

$$\forall (x, y) \in E \mid x \cdot y = y \cdot x$$

2.1.3 Anneau

On appelle un anneau sur un ensemble E muni de deux lois de composition notées respectivement $(+, \cdot)$ qui combine deux éléments x et y de E pour obtenir deux autres éléments $(x \cdot y)$ et $(x + y)$ tel que $(x, y) \to (x \cdot y)$ et $(x, y) \to (x + y)$ sur E satisfaisant les trois axiomes suivants :

- $(E, +)$ forme un groupe commutatif.
- La loi de composition $(x \cdot y)$ est associative dont l'élément neutre est 1.
- $\forall (x, y, z) \in E : x(y + z) = (x \cdot y)(y \cdot z)$ (Distributivité).

2.1.4 Anneau commutatif

Si la loi de composition $(x, y) \to (x \cdot y)$ est commutative alors l'anneau est appelé anneau commutatif. On note que l'ensemble des entiers munis des lois de composition habituelles (adition et multiplication) forment un anneau commutatif.

2.1.5 Corps

Un corps est défini sur un ensemble E muni des deux lois de composition notées respectivement $(+, \cdot)$ satisfaisant les conditions suivantes :

- $(E, +)$ forme un groupe abélien.
- Il existe un élément neutre, noté 0, tel que $\forall x \in E \mid (x + 0) = (0 + x) = x$.

- $(E \setminus \{0\}, \cdot)$ forme également un groupe abélien.
- Il existe un élément neutre, noté 1, tel que $\forall x \in E \mid (x \cdot 1) = (1 \cdot x) = x$.
- $\forall (x, y, z) \in E \mid x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ et $(y + z) \cdot x = (y \cdot x) + (z \cdot x)$ (Distributivité de la multiplication $(\cdot)$ pour l'addition $(+)$.

Autrement dit, un corps est un anneau dont les éléments non nuls sont inversibles et qui forment un groupe abélien pour la multiplication. Les anneaux Q et R, appelés respectivement corps des nombres premiers et corps des nombre réels, sont des corps.

2.1.6 Corps fini

Un corps fini est un corps commutatif dont le nombre d'éléments est fini. Soit p un nombre premier, appelé la caractéristique du corps, et $n \in Z^+$, l'ordre du corps, noté q et qui représente le nombre d'éléments de ce corps, est défini comme suit : $q = p^n$.

2.1.7 Corps fini premier

Un corps fini est premier, noté F_P, s'il est constitué d'un nombre fini d'éléments entiers $\{0, 1, 2, p-2, p-1\}$ dont l'ordre $q = p^n$, p est un nombre premier et $\forall x \in Z, x \ mod \ p = r$ tel que $r \in [\![0, p-1]\!]$.

2.1.8 Corps fini binaire

Un corps binaire est un corps fini de l'ordre 2^n, noté F_{2^n}, construit en utilisant une représentation polynomiale où les éléments du corps sont des polynômes binaires dont les coefficients $a_i \in \{0, 1\}$ et les degrés sont inférieurs à n. le corps F_{2^n} est représenté comme suit : $F_{2^n} = \{a_{n-1}Z^{n-1} + a_{n-2}Z^{n-2} + \cdots + a_1 Z + a_0\}$.

2.1.9 Arithmétique modulaire sur les corps finis

Elle comprend un ensemble d'entiers sur lequel des opérations arithmétiques comme l'addition, la soustraction, la multiplication, l'inversion et le carré sont exécutées. Ces opérations sont calculées en modulo un nombre premier p.

— Addition : soient $a, b \in F_p$, $(a + b) \bmod p = r$, où r est le reste de la division entière de $(a + b)$ par p, $0 \leq r \leq p - 1$. Elle est dénotée par A.

— Soustraction : soient $a, b \in F_p$, $(a - b) \bmod p = r$, où r est le reste de la division entière de $(a - b)$ par p, $0 \leq r \leq p - 1$. Cette opération peut être remplacée par une addition de a et de $(-b)$. La soustraction a le même coût qu'une addition, elle est dénotée également par A.

— Multiplication : soient $a, b \in F_p$, $(a \cdot b) \bmod p = r$, où r est le reste de la division entière de $(a \cdot b)$ par p, $0 \leq r \leq p-1$. Elle est dénotée par M.

— Inversion : soit $a \neq 0 \in F_p$, $(a^{-1}) \bmod p = r$, est l'unique entier $r \in F_p$ pour lequel $(a \cdot r) \bmod p = 1$. Elle est dénotée par I.

— Carré : soit $a \in F_p$, $(a^2) \bmod p = r$, où r est le reste de la division entière de a^2 par p, $0 \leq r \leq p-1$. Dans certaines implémentations, cette opération est remplacée par la multiplication $(a \cdot a)$. Elle est dénotée par S.

2.2 Présentation des courbes elliptiques

Le choix d'une courbe elliptique est à la fois difficile et judicieux relativement à la large gamme de courbes disponibles, on cite les courbes obtenues à l'aide du théorème de Weil [80], les courbes super singulières, les courbes d'Edwards, de Montgomery, de Weierstrass, etc. Dans ce document, nous nous intéresserons uniquement aux courbes elliptiques de Weierstrass définies sur un corps fini premier. Notre choix s'est porté sur ce type de courbes car elles offrent la possibilité de les utiliser dans la cryptographie pour la mise en œuvre de schémas cryptographiques asymétriques basés sur le problème du logarithme discret.

2.2.1 Définition de courbe elliptique

Une courbe elliptique est une courbe cubique non singulière définie sur un corps K satisfaisant la propriété suivante : Soit $P(X, Y, Z) \in K^3$ tel que $E : F(x, y, z) = 0$ alors $\frac{\partial F}{\partial x}(X, Y, Z), \frac{\partial F}{\partial y}(X, Y, Z), \frac{\partial F}{\partial z}(X, Y, Z)$ n'est pas un vecteur nul. Ceci démontre qu'on peut définir une tangente à la courbe au point $P(X, Y, Z)$. Il existe plusieurs types de courbes elliptiques définies sur un corps fini premier, nous citons :

— Les courbes de Weierstrass que nous détaillerons dans la suite de cette section.

— Les courbes d'Edwards : $x^2 + y^2 = c^2(1 + dx^2y^2)$.

— Les courbes tordues d'Edwards : $ax^2 + y^2 = 1 + dx^2y^2$ (twisted Edwards curves en anglais).

2.2.2 Equation de Weierstrass

Une courbe elliptique définie sur un corps K est l'ensemble des solutions à l'équation de Weierstrass suivante :

$$E : y^2z + a_1xyz + a_3yz^2 = x^3 + a_2x^2z + a_4xz^2 + a_6z^3 \tag{2.1}$$

avec $a_1, a_2, a_3, a_4, a_6 \in F$.

L'équation 2.1 peut être simplifiée avec des coordonnées non homogènes. On pose $X = \frac{x}{z}$ et $Y = \frac{y}{z}$. L'équation 2.1 devient alors :

$$E : Y^2 + a_1XY + a_3 = X^3 + a_2X^2 + a_4X + a_6 \tag{2.2}$$

Dans le contexte de la cryptographie des courbes elliptiques, nous travaillons sur des entiers et nous gardons uniquement les points dont l'abscisse et l'ordonnée sont des entiers. Afin de limiter l'abscisse et l'ordonnée d'un point on utilise l'arithmétique modulaire. A cet effet, on ajoute un nombre premier p et chacune des coordonnées d'un point sera exprimée en modulo p. De plus, si la caractéristique du corps K est différente de 2 et de 3 ($car(K) \neq 2$ et $car(K) \neq 3$), l'équation simplifiée 2.2 qui définit la courbe elliptique sur un corps fini premier F_p est donnée par :

$$E : Y^2(mod\ p) = X^3 + aX + b(mod\ p) \tag{2.3}$$

avec a, b deux entiers $\in F_p$.

Tout point $P(X, Y)$ appartenant à la courbe elliptique définie sur E doit posséder la propriété de lisseté tel que $\frac{\partial P(X,Y)}{\partial X} = 0$ et $\frac{\partial P(X,Y)}{\partial Y} = 0$ [3]. Nous munissons les points de la courbe définie sur E par une loi de groupe $+$ et un point à l'infini O tel que $P(X, Y) + O = P(X, Y)$.

2.2.3 Représentation graphique sur un corps fini premier

Exemple : $E : Y^2 = X^3 - X$ avec $a = 1$ et $b = 0 \in F_p$. La courbe elliptique qui satisfait cette équation est donnée par la Figure 2.1.

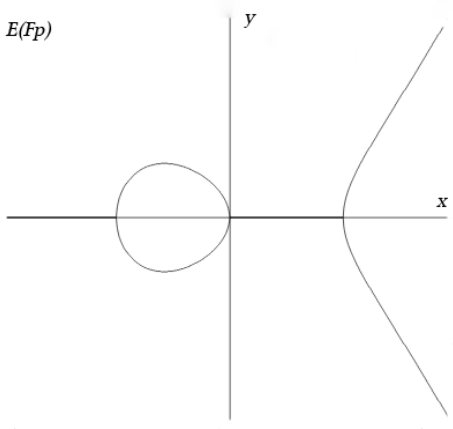

FIGURE 2.1 – Exemple de courbe elliptique sur $E(F_p)$ donnée par l'équation $Y^2 = X^3 - X$

2.2.4 Addition et doublement de points

L'addition et le doublement de points, un cas particulier de l'addition d'un point avec lui-même, s'effectuent entre les points d'une même courbe elliptique définie sur un corps et le résultat est un autre point sur la même courbe. L'addition et le doublement de points peuvent être calculés dans différents systèmes de coordonnées : Affines (cas général), projectives, Jacobéennes, co-Z et mixtes. Dans cette thèse, nous travaillerons et nous présenterons les formules uniquement sur les corps finis premiers (F_p) étant les plus adaptés pour la cryptographie.

Approche géométrique

Soit E une courbe elliptique définie sur un corps K, et trois points P, Q et R de la courbe $E(K)$ tel que :

- L'opposé d'un point $P = -R$: on trace une droite parallèle à l'axe des ordonnées passant par le point P. Le point R est le point d'intersection de la droite et la courbe (voir Figure 2.2).
- L'addition de deux points $P + Q = R$: on trace une droite passant par les deux points P et Q et qui coupe la courbe en un troisième point R. L'opposé du point R définie alors l'addition des deux points P et Q (voir Figure 2.3).
- Le doublement d'un point $2P = R$: on trace une droite tangente à la courbe passant par le point P et qui coupe la courbe en un deuxième point R. L'opposé du point R définie alors le doublement du point P (voir Figure 2.4).

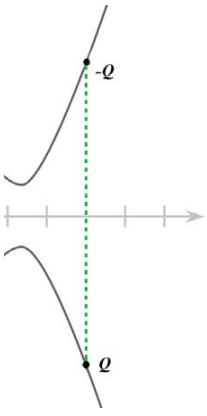

FIGURE 2.2 – Opposé d'un point

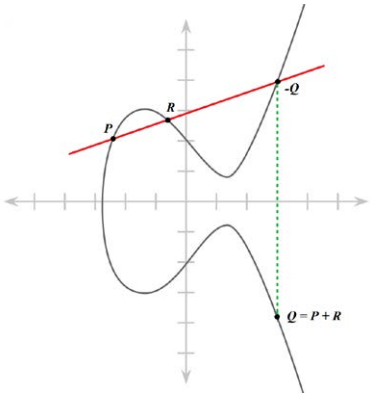

FIGURE 2.3 – Addition de points

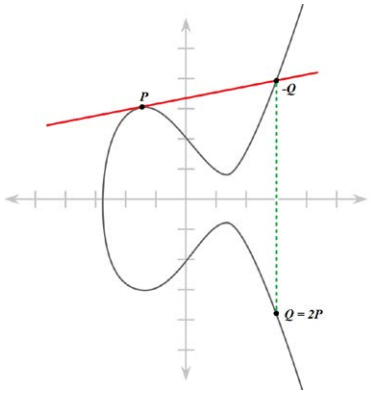

FIGURE 2.4 – Doublement d'un point

Représentation en coordonnées affines

Soient $P(X_P, Y_P)$ et $Q(X_Q, Y_Q)$ deux points de la courbe elliptique définie sur F_p tel que $P \neq Q$. Le doublement du même point P donne un autre point de la courbe $R(X_w, Y_w)$ tel que $2P(X_P, Y_P) = W(X_w, Y_w)$ calculé comme suit :

$$\lambda = (3X_P^2 + a)/(2Y_P)$$
$$X_w = \lambda^2 - 2X_P$$
$$Y_w = \lambda(X_P - X_w) - Y_P$$

L'addition des deux points P et Q donne un autre point de la courbe $R(X_w, Y_w)$ tel que $P(X_P, Y_P) + Q(X_Q, Y_Q) = R(X_w, Y_w)$ calculée comme suit :

$$\lambda = (Y_Q - Y_P)/(X_Q - X_P)$$
$$X_w = \lambda^2 - X_P - X_Q$$
$$Y_w = \lambda(X_P - X_w) - Y_P$$

2.2.5 Problème du logarithme discret elliptique

Soient E, F, P, Q, k des paramètres elliptiques tel que E est une courbe elliptique définie sur le corps fini premier F, P et Q deux points de la courbe $E(F)$ et $k \in N$, le problème du logarithme discret consiste à trouver l'entier k, s'il existe, tel que $Q = k \cdot P$. Si F est un corps fini, ce problème est réputé difficile. La solution la plus naïve pour résoudre ce problème est de calculer exhaustivement $1P$, $2P$, $3P \cdots$ jusqu'à ce que nous trouvions Q, mais le calcul peut devenir extrêmement long si la valeur de k est suffisamment grande. Réciproquement, connaissant E, F, P et k, il est facile de calculer $Q = k \cdot P$ en utilisant un algorithme d'exponentiation rapide (voir l'algorithme 1).

Algorithme 1 Algorithme d'exponentiation rapide pour la résolution du logarithme discret

Require: Une courbe elliptique E définie sur F_p, un entier n et un point $G \in E(F_p)$
Ensure: Q = n·G

```
P ← O
if n = 0 then
  Return P
else
  if n < 0 then
    N ← −n
    Q ← −G
  else
    N ← n
    P ← G
  end if
  while n est impair do
    Q ← Q + P
    N ← N/2
    if n = 0 then
      Return P
    else
      P ← 2P
    end if
  end while
end if
```

L'algorithme connu comme étant le plus efficace pour résoudre le problème du logarithme discret est à temps exponentiel, contrairement au système RSA pour lequel il existe des algorithmes a temps sous-exponentiel. Un algorithme est sous-exponentiel si le logarithme du temps d'exécution croît asymptotiquement moins vite que tout polynôme donné [82].

2.2.6 Multiplication scalaire

Une multiplication scalaire consiste à multiplier un point de la courbe par un scalaire et le résultat est un autre point de la courbe. Soit un point P de la courbe elliptique. Il est possible d'ajouter le point P à lui-même ($P + P = 2P = Q$) pour réaliser un doublement et on peut également ajouter le point P à un autre point Q de la courbe elliptique pour réaliser une addition de points. On peut ensuite calculer $3P, 4P, \cdots, kP$ avec k un scalaire. Par exemple, $3P$ est calculé comme suit : $3P = P + P + P = (P + P) + P = Q + P$. Pour calculer $3P$, on a réalisé le doublement du point P, on a obtenu le point Q qu'on a ensuite additionné au point P. En généralisant cette multiplication, on obtient : $Q = kP = P + P + \cdots + P$ (k fois) calculé par la formule 2.4. Il existe plusieurs algorithmes de calcul d'une multiplication scalaire, nous les

détaillerons dans le chapitre suivant de ce document.

Soit $k = (k_{n-1}, \cdots, k_1, k_0)_2 \in N$, $k_i \in \{0, 1\}$:

$$\left\{ \begin{array}{rcl} k \cdot P & = & (\sum_0^{n-1} 2^i k_i \cdot P) \\ & = & (k_0 + 2(k_1 + 2(k_2 + 2(\cdots + 2k_{n-1}) \cdots))) \cdot P \\ & = & (k_0 \cdot P + 2(k_1 \cdot P + 2(k_2 \cdot P + 2(\cdots + 2k_{n-1} \cdot P) \cdots))) \end{array} \right\} \quad (2.4)$$

En cryptographie, on associe un point générateur à chaque courbe elliptique. Le rôle du générateur, comme son nom le suggère, est de générer une famille de points par la loi d'addition. Dans un corps fini, le nombre de points est fini et appelé l'ordre du générateur. Le point générateur sert donc à dériver des clés publiques à partir des clés privées. Ainsi, une clé publique, représentée par un point Q, est obtenue par la multiplication d'une clé privée, représentée par un scalaire k, par le point générateur P. Le calcul du point Q connaissant k et P est très facile mais retrouver k à partir de P et Q est très difficile voire impossible pour un k assez grand. C'est le secret de la cryptographie sur les courbes elliptiques pour résoudre le problème du logarithme discret dans les réseaux de capteurs sans fil. Cette technique permet, entre autres, le partage de clés, le chiffrement/déchiffrement et la signature des paquets de données.

2.3 Cryptographie des courbes elliptiques

Les courbes elliptiques est un ensemble d'opérations mathématiques adaptées à la cryptographie à clé publique. La grande majorité des courbes elliptiques utilisées en pratique pour ECC sont définies sur des corps finis premiers F_p. La cryptographie sur les courbes elliptiques (ECC) utilise le même principe que RSA pour garantir la sécurité des opérations cryptographiques [78]. La sécurité est obtenue par la résolution du problème du logarithme discret sur les courbes elliptiques qui consiste à retrouver un entier k à partir de deux points (P et Q) d'une courbe elliptique E définie sur le corps fini (F_p) à p éléments, tel que $Q = k \cdot P$ et $k < p$.

ECC est présentée indépendamment par [1] et [2] au milieu des années 80 mais son utilisation dans le cadre des réseaux à ressources restreintes fut alors impossible en raison de la complexité des opérations de traitement et de mémoire qui nécessitaient une consommation de ressources importante. La Figure 2.5 montre la complexité d'une seule opération de multiplication scalaire par un point, qui fait intervenir des milliers d'opérations arithmétiques et des centaines d'opérations de doublement et d'addition de points. Les différentes optimisations faites récemment sur les opérations de calcul notamment la multiplication scalaire ont permis d'attirer l'attention des chercheurs sur leur utilisation. Une étude de la cryptographie asymétrique pour les réseaux de

capteurs est présentée ici [83].

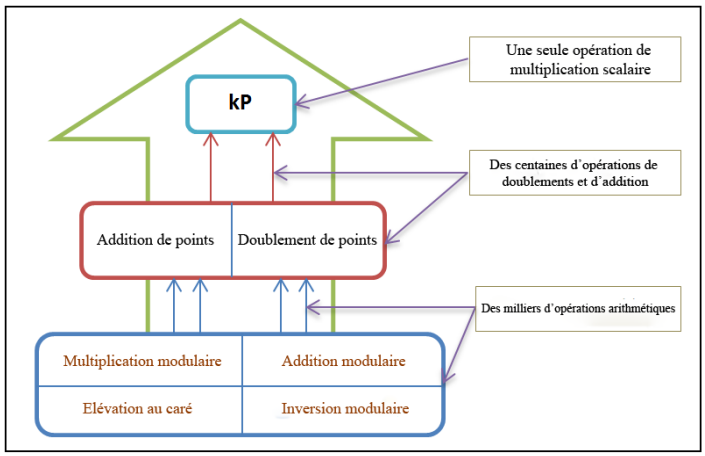

FIGURE 2.5 – Complexité des calculs d'une multiplication scalaire

La cryptographie des courbes elliptiques possède plusieurs limites et leur utilisation directe dans le cadre des réseaux à ressources restreintes comme les réseaux de capteurs sans fil est très problématique pour les raisons suivantes :

- Complexité des calculs notamment la multiplication d'un point par un scalaire. La complexité mémoire d'une multiplication scalaire est égale à $\log_2(n)$ où n est l'ordre du groupe choisi, et la complexité spatiale est $\ominus(\log(n))^u$ avec un $u >= 1$ petit.
- Lenteur d'exécution. Le temps de calcul d'une multiplication scalaire dépend en général de trois facteurs : i) La taille des données (taille des opérandes), ii) Le nombre d'opérations de doublement de points et iii) Le nombre d'opérations d'addition de points. La lenteur d'exécution constitue un problème pour les applications temps réel et un inconvénient pour les applications d'agrégations de données et de calculs.
- L'analyse du temps de calcul favorise les attaques basées sur l'analyse de la fréquence des cycles d'horloge.
- La sécurité des échanges et la complexité sur laquelle repose le logarithme discret dépend de l'entier p choisi pour le corps Z/pZ.
- Ressources limitées des capteurs (systèmes embarqués en général) notamment en mémoire et en puissance de calcul.

2.4 Cryptosystèmes basés sur les courbes elliptiques

Les communications dans les réseaux informatiques en général et les réseaux de capteurs en particulier emploient des protocoles pour sécuriser les données échangées

et authentifier les entités communicantes. La plupart des protocoles implémentés et intégrés dans les applications sur internet ou dans les applications des réseaux embarqués et celles des réseaux de capteurs sans fil, sont basés sur un outil central : la clé. Une clé est utilisée pour le chiffrement et le déchiffrement des données échangées ainsi que pour l'authentification des entités légitimes du réseau. On distingue deux types de clés. La clé privée connue uniquement par l'entité qui la génère et la clé publique connue par tous. Dans ce qui suit, nous allons présenter quelques cryptosystèmes pour l'échange de clés, le chiffrement et la signature numérique, basés sur le logarithme discret et les courbes elliptiques.

2.4.1 Échange de clés

Dans le cadre de la cryptographie symétrique, deux entités communicantes sur un réseau partagent une même clé commune pour chiffrer et déchiffrer les messages échangés. Cette clé doit être renouvelée périodiquement. Dans des applications de réseaux de capteurs basées sur la cryptographie symétrique, et après le déploiement, le renouvellement des clés constitue un sérieux problème pour différentes raisons, nous citons quelques unes : i) le canal de communication non protégé, ii) les attaques redoutables de type Man in the Middle (Homme du milieu), iii) la nature du déploiement, iv) la topologie dynamique du réseau et v) l'absence d'une autorité centrale. Diffie et Hellman ont proposé en 1976 un protocole [84] qui répond au problème d'échange de clés.

L'échange de clés se réalise comme suit. Soit un grand nombre g appelé générateur, un grand nombre premier p. Le générateur g et le nombre premier p sont rendus publics. Deux nœuds A et B, ayant les deux clés privées x et y respectivement, peuvent désormais échanger leurs clés publiques P_A, P_B et calculer une clé secrète commune k comme illustré sur la Figure 2.6 ci-dessous.

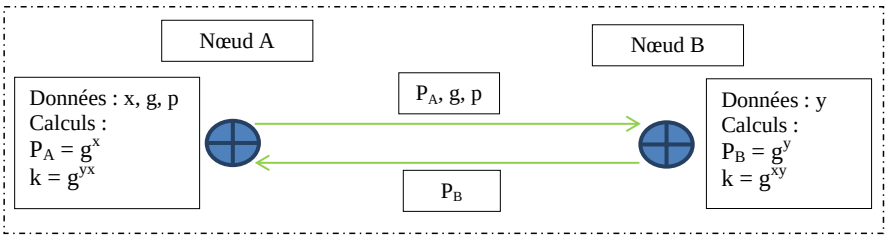

FIGURE 2.6 – Algorithme d'échange de clés basé sur le problème du logarithme discret

L'attaquant, un nœud intrus sur le réseau et qui écoute sur le canal de communication partagé par A et B, peut récupérer publiquement les données g, p, g^x et g^y. Bien qu'il possède toutes les données lui permettant de retrouver les clés privées x et y, l'attaquant ne peut pas calculer, en un temps polynomial et raisonnable, x ou y à partir de g^x ou g^y respectivement car ce calcul correspond à la résolution du problème du

logarithme discret. Cependant, ce système ne présente aucun moyen d'authentification pour A et B et demeure très vulnérable aux attaques de type Man in the Middle.

L'algorithme de Diffie et Hellman peut être appliqué sur les courbes elliptiques. Soit un grand nombre premier p et deux entiers a, b satisfaisant l'équation 2.3. a et b définissent la courbe elliptique $E(F_p)$ d'ordre n très élevé. On choisit au hasard un point G comme point générateur de $E(F_p)$. Deux nœuds A et B, ayant les deux clés privées k_A et k_B respectivement, peuvent calculer une clé commune comme montré par la Figure 2.7 ci-dessous.

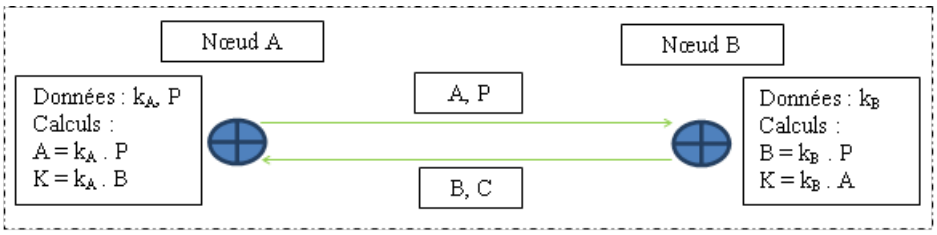

FIGURE 2.7 – Algorithme d'échange de clés sur les courbes elliptiques

2.4.2 Chiffrement/Déchiffrement d'El Gamal

El Gamal a proposé en 1985 un système de chiffrement [85] asymétrique basé sur le logarithme discret dans un corps fini. Deux nœuds A et B, ayant les deux clés privées x et y respectivement, souhaitent échanger un message m, détenu par A, en toute sécurité. Le nœud B génère le générateur g et le nombre premier p. Le générateur g et le nombre premier p sont rendus publics. Le principe est illustré sur la Figure 2.8 ci-dessous.

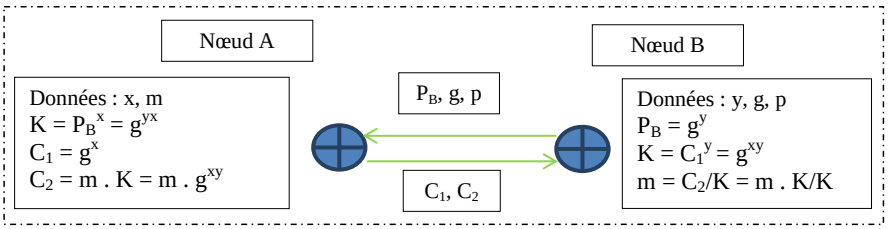

FIGURE 2.8 – Algorithme de chiffrement d'El Gamal basé sur le problème du logarithme discret

Pour les mêmes raisons que pour l'algorithme Diffie-Hellman, un attaquant peut récupérer toutes les données échangées par A et B mais ne pourra pas, en un temps polynomial ou raisonnable, retrouver les clés secrètes x et y et par conséquent le message m. L'algorithme de chiffrement d'El Gamal peut être appliqué sur les courbes elliptiques. Soit un grand nombre premier p et deux entiers a, b satisfaisant l'équation 2.3 et définissant la courbe elliptique $E(F_p)$ d'ordre n très élevé. On choisit au hasard

un point G comme point générateur de $E(F_p)$. Deux nœuds A et B, ayant les deux clés privées k_A et k_B respectivement, peuvent créer un canal de communication sécurisé pour échanger des messages comme montré par la Figure 2.9 ci-dessous.

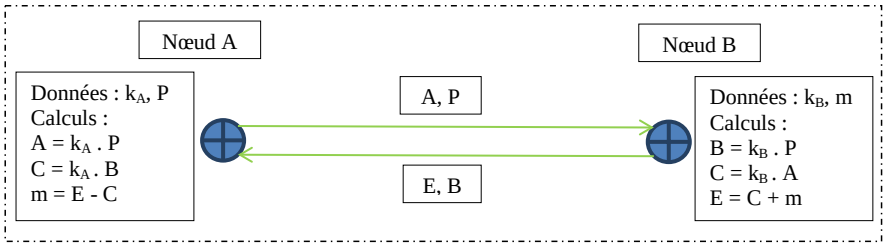

FIGURE 2.9 – Algorithme de chiffrement d'El Gamal sur les courbes elliptiques

2.4.3 Signature numérique d'El Gamal

Les algorithmes présentés ci-dessus, par les Figures 2.6, 2.7, 2.8 et 2.9, ne permettent pas aux deux nœuds A et B d'avoir une preuve qu'ils communiquent entre eux sans aucune intrusion d'une tierce partie ou si le message a été modifié ou altéré pendant la transmission. Ce problème est résolu par la technique de la signature numérique d'un message échangé afin d'identifier son émetteur par le récepteur et de garantir son intégrité.

El Gamal a proposé en 1985 un algorithme pour la signature et la vérification des messages échangés [85] basé sur le logarithme discret dans un corps fini. Le nœud A muni de la clé privée x est le nœud signataire du message. A génère le générateur g, le nombre premier p et nombre entier $k \in [\![0; p-1]\!]$ tel que PGCD $(k, p-1) = 1$. Le générateur g et le nombre premier p sont rendus publics. Le principe est donné par la Figure 2.10 ci-dessous.

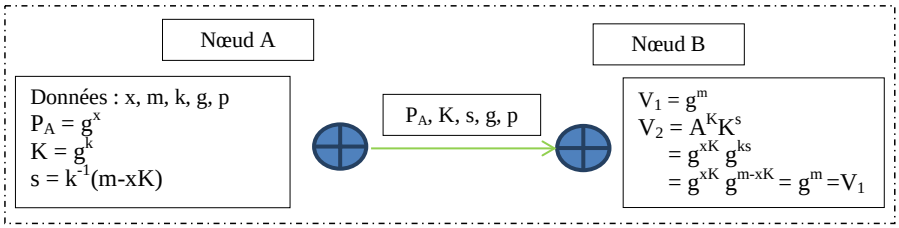

FIGURE 2.10 – Algorithme de signature numérique d'El Gamal basé sur le problème du logarithme discret

L'algorithme de signature numérique d'El Gamal peut être appliqué sur les courbes elliptiques. Soit un grand nombre premier p et deux entiers a, b satisfaisant l'équation 2.3 et définissant la courbe elliptique $E(F_p)$ d'ordre n très élevé. On choisit au hasard

un point G comme point générateur de $E(F_p)$. Le nœud A muni de la clé privée k_A peut signer un message avant de l'envoyer au nœud B qui le vérifie par la suite comme expliqué par la Figure 2.11 ci-dessous :

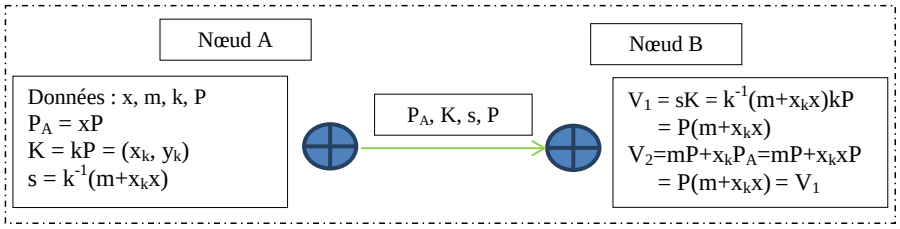

FIGURE 2.11 – Algorithme de signature numérique d'El Gamal sur les courbes elliptiques

2.5 Conclusion

Les courbes elliptiques sont des objets mathématiques complexes offrant de nombreuses applications dans plusieurs domaines des mathématiques. Leurs applications dépendent beaucoup du corps de définition choisi. Les courbes elliptiques définies dans un corps fini premier s'adaptent à la cryptographie à clé publique, appelée cryptographie des courbes elliptiques, pour fabriquer des codes performants. Ainsi, elles interviennent pour le chiffrement/déchiffrement, signature numérique, échange de clés, entre autres.

La sécurité des courbes elliptiques repose sur la résolution du problème du logarithme discret où aucune méthode efficace pour le résoudre en un temps raisonnable n'est connue, pour des nombres suffisamment grands. Cependant, les calculs elliptiques font intervenir des opérations de multiplication scalaire par un point, très complexes et qui exigent une puissance de calcul considérable pour des dispositifs à ressources limitées. Toutefois, malgré les efforts de la communauté scientifique, les fonctions elliptiques demeurent très complexes et nécessitent beaucoup d'optimisations, très particulièrement sur la multiplication scalaire considérée comme l'opération centrale et complexe. Dans le chapitre suivant, nous présenterons quelques solutions d'optimisations d'une multiplication scalaire.

Chapitre 3
Optimisations de la multiplication scalaire sur courbe elliptique

La cryptographie des courbes elliptiques est longtemps considérée comme inappropriée pour les réseaux à ressources restreintes. Les récents travaux de la communauté scientifique ont changé cette hypothèse qui considérait que la réduction de la complexité des calculs elliptiques est une tâche insurmontable. Les efforts considérables de Montgomery, Joye et d'autres ont permis de réduire les calculs par différentes techniques mathématiques et algorithmiques. Ces travaux se sont intéressés très exactement à la multiplication scalaire par un point, une opération très complexe et centrale dans les calculs elliptiques. L'algorithme Double-and-Add est la première solution proposée pour opérer une multiplication scalaire. Malgré ses performances, Double-and-Add présente plusieurs vulnérabilités comme le nombre important d'opérations de doublement et d'addition de points ainsi que ses failles de sécurité.

La complexité des calculs dans une multiplication scalaire par un point et le nombre important de doublements et d'additions de points exécutés pour effectuer cette opération a nécessité beaucoup de travaux d'optimisation pour la réduction des calculs elliptiques. Les premiers travaux d'optimisation étaient orientés vers la réduction du nombre de doublements et d'additions dans une multiplication scalaire. D'autres ont pu réduire la complexité des calculs par le changement du système de coordonnées en passant des coordonnées affines aux coordonnées mixtes et Jacobéennes [86]. Ce type de coordonnées permet d'éviter les opérations d'inversion de points trop coûteuse en temps de calcul et en consommation d'énergie. Enfin, quelques travaux ont pu réduire cette complexité par l'introduction d'algorithmes et de techniques distribuées et parallèles. Nous présentons dans cette section quelques travaux portés sur ces trois types d'optimisation.

Dans ce chapitre, nous allons présenter d'abord l'algorithme standard de calcul d'une multiplication scalaire Double-and-Add, ses limites et quelques solutions pour contrer les deux attaques SPA [87] et DPA [88], puis quelques techniques mathématiques et algorithmiques d'optimisation. Ensuite, nous allons présenter quelques techniques de parallélisation permettant l'accélération des calculs dans une multiplication scalaire. Dans cette thèse, nous considérons uniquement les algorithmes sur les courbes elliptiques de Weierstrass définies sur un corps fini premier, détaillées à la section 2.2.2 ci-dessus.

3.1 Algorithme standard de calcul d'une multiplication scalaire

Plusieurs algorithmes ont été proposés pour le calcul d'une multiplication scalaire et au même temps optimiser la complexité. Le premier algorithme proposé [3] est l'algorithme doubling-and-addition (connu sous le nom de Double-and-Add) équivalent à l'algorithme multiply-and-square en exponentiation modulaire.

L'algorithme standard Double-and-Add (voir algorithme 2) présente plusieurs inconvénients malgré ses bonnes performances dans le calcul d'une multiplication scalaire. Le premier inconvénient est sa vulnérabilité aux attaques SPA [87] et DPA [88]. Cette vulnérabilité est corrigée par plusieurs travaux, nous présentons ici quelques-uns. La seconde vulnérabilité est le nombre d'additions et de doublements effectués pour chaque opération de multiplication scalaire. Plusieurs travaux ont été effectués pour réduire le nombre d'opérations de doublements et d'addition, nous présentons ici quelques-uns.

3.1.1 Présentation de l'algorithme Double-and-Add

La multiplication scalaire est une suite d'addition de points consécutifs $Q = k \cdot P = P + P + \cdots + P$ (k fois). P et Q sont deux points de la courbe elliptique définie sur un corps fini premier $E(F_p)$, k est un nombre entier positif de longueur l bits représenté en binaire par $k = \sum_{i=0}^{l-1}(k_i 2^i)$. Deux opérations sont nécessaires pour effectuer une multiplication scalaire : le doublement de points $P + P = 2P = R$ et l'addition de points $P + Q = R$. Ainsi, pour calculer $k \cdot P$, nous parcourons le scalaire k du bit de poids faible au bit de poids fort (ou inversement). Un doublement est effectué pour chaque bit k_i de k, suivi d'une addition pour chaque bit non nul ($k_i \neq 0$) (voir l'algorithme 2). L'algorithme Double-and-add nécessite en moyenne l doublements et $l/2$ additions.

Algorithme 2 Algorithme standard Double-and-Add de calcul $Q = k \cdot P$ bit faible/bit fort

Require: $k = (k_{l-1}, \cdots, k_1, k_0)$, $P \in E(F_p)$
Ensure: $Q = k \cdot P$
  $Q \leftarrow \infty$
  for $i \leftarrow 0$ *to* $l - 1$ **do**
    if $k_i = 1$ **then**
      $Q \leftarrow Q + P$
    end if
    $Q \leftarrow 2Q$
  end for
  return Q;

3.1.2 Failles de sécurité de l'algorithme Double-and-Add

Les algorithmes binaires sont efficaces mais ne sont pas sûrs dans des applications où ils sont exposés à certaines types d'attaques comme les SCA (Side-Channel Analysis) appelés Attaques par canaux cachés. Les attaques SCA exploitent la fuite de l'information physique lors des calculs cryptographiques tels que sa consommation de ressources ou ses rayonnements électromagnétiques. Dans le cadre de la cryptographie des courbes elliptiques et particulièrement la multiplication scalaire par un point, les traitements sont très vulnérables à deux types d'attaques SCA : l'attaque SPA (Simple Power Analysis) et l'attaque DPA (Differential Power Analysis).

L'attaque SPA [87] consiste à identifier l'ordre d'enchainement de deux motifs dans la trace de consommation d'un calcul d'une multiplication scalaire en observant un canal caché. Cette information permet à l'attaquant de retrouver l'ordre d'exécution des opérations de doublement et d'additions de points. L'attaque DPA [88] repose sur le même principe que SPA à la différence que l'attaque DPA consiste plutôt à observer plusieurs courbes du canal caché pour pouvoir retrouver le secret de la multiplication scalaire (appelé également le secret de la trappe) à l'aide d'outils statistiques. Un exemple d'une telle attaque est donné par la Figure 3.1.

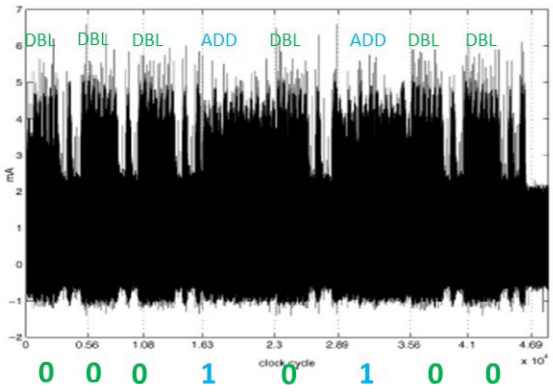

FIGURE 3.1 – Exemple d'attaque de type SCA

Sur la Figure 3.1, on remarque qu'un attaquant peut facilement reconstituer le scalaire k utilisé pour la multiplication scalaire en analysant la courbe de consommation de ressources du capteur. Ainsi, il est clair qu'une opération de doublement consomme moins de ressources qu'une opération d'additions de points. En comparant la trace de consommation de chacune des deux opérations et en s'appuyant sur le fonctionnement de l'algorithme Double-and-Add, un attaquant peut aisément retrouver le secret k. En effet, l'algorithme Double-and-Add présente un énorme inconvénient. Un attaquant qui écoute le canal peut facilement retrouver le scalaire k en analysant la fréquence d'exécution des opérations de calcul. Ainsi, si le bit est égal à 0, l'algorithme met moins de temps pour calculer une seule opération de doublement contrairement à un

bit égal à 1 où on calcule un doublement suivi d'une addition. Ce type d'attaques est connu sous le nom de SPA [87] et DPA [88]. Plusieurs solutions ont été proposées afin de corriger cette vulnérabilité. Nous citerons dans la section suivante quelques-unes.

3.1.3 Contrer les attaques SPA et DPA

Un grand nombre d'algorithmes et de méthodes [89] sont proposées pour contrer les attaques de type SCA. Nous présentons succinctement quelques-unes ici. Le principe appliqué par les solutions aux attaques de type SCA est de rendre la multiplication scalaire régulière, c'est-à-dire exécuter les opérations constamment quelle que soit la valeur du scalaire. L'objectif étant de faire des opérations de doublement et d'addition de points des motifs indiscernables.

Multiplication scalaire par la méthode 2^t-ère

La méthode 2^t-ère [3] repose sur le calcul de valeurs particulières. Quand la valeur de $t = 1$, la méthode 2^t-ère et la méthode binaire Double-and-Add se valent d'où la première est une généralisation de la seconde. La méthode 2^t-ère requiert le pré-calcul des valeurs xP avec $x \in [\![2; 2^t - 1]\!]$. Le principe est détaillé par l'algorithme 3 et un exemple concret est donné dans [90].

Algorithme 3 Calcul d'une multiplication scalaire avec la méthode 2^t-ère

Require: $k = (k_{l-1}, \cdots, k_1, k_0)$, $P \in E(F_p)$
Ensure: $Q = k \cdot P$
  Calculer : $d_i = k_{it} + 2k_{it+1} + 2^2 k_{it+2} + \cdots 2^{t-1} k_{it+t-1}$
  Calculer : $k = d_0 + 2^t d_1 + 2^{2t} d_2 + \cdots + 2^{2l} d_l$
  Calculer : $P, 2P, 3P, \cdots, (2^{-1}P)$
  $Q \leftarrow O$
  for $i \leftarrow l - 1$ *to* 0 *par pas de* $-t$ **do**
    $Q \leftarrow 2^t Q$
    if $d_i > 0$ **then**
      $Q \leftarrow Q + d_i P$
    end if
  end for
  return Q;

Algorithme Double-and Add Always

L'algorithme Double-and-Add Always [91] comme son nom l'indique exécute à chaque itération de la boucle et successivement les deux opérations de doublement et d'addition de points indépendamment du bit du scalaire k. Si le bit est nul, l'addition est ignorée. L'idée est d'effectuer des opérations inutiles et plus précisément forcer une addition à chaque tour de boucle afin d'avoir un temps de calcul constant. L'algorithme

Double-and-Add Always, décrit par l'algorithme 4 a une complexité de l doublements et l additions. Il est beaucoup plus lent que l'algorithme standard Double-and-Add mais résiste mieux aux attaques SCA.

Algorithme 4 Algorithme Double-and-Add Always pour contrer les attaques SCA

Require: $k = (k_{l-1}, \cdots, k_1, k_0)$, $P \in E(F_p)$
Ensure: $Q = k \cdot P$
  $Q_0 \leftarrow O$
  $Q_1 \leftarrow O$
  for $i \leftarrow l-1$ *to* 0 **do**
    $Q_0 \leftarrow 2Q_0$
    $Q_1 \leftarrow Q_0 + P$
    $Q_0 \leftarrow Q_{k_i}$
  end for
  return Q_0 ;

Echelle de Montgomery

Montgomery Ladder [92] a présenté une autre alternative pour contrer les attaques SPA et DPA appelée Echelle de Montgomery. A chaque itération et quelle que soit la valeur du bit k_i de k, une opération de doublement et une autre d'addition de points sont exécutées. Cette propriété est partagée avec l'algorithme Double-and-Add Always. Cependant, l'Echelle de Montgomery possède une relation entre les quantités manipulées de telle sorte qu'il existe un invariant à chaque tour de boucle $R_1 - R_0 = P$. L'Echelle de Montgomery est détaillé par l'algorithme 6.

Algorithme 5 Echelle de Montgomery pour le calcul d'une multiplication scalaire

Require: $k = (k_{l-1}, \cdots, k_1, k_0)$, $P \in E(F_p)$
Ensure: $Q = k \cdot P$
  $Q_0 \leftarrow O$
  $Q_1 \leftarrow P$
  for $i \leftarrow l-1$ *to* 0 **do**
    if $k_i = 0$ **then**
      $Q_1 \leftarrow Q_0 + Q_1$
      $Q_0 \leftarrow 2Q_0$
    else
      $Q_0 \leftarrow Q_0 + Q_1$
      $Q_1 \leftarrow 2Q_1$
    end if
  end for
  return Q_0 ;

Multiplication scalaire hautement régulière de Joye

Joye a proposé une autre solution [93] qui permet de construire un algorithme régulier dans lequel aucune opération n'est inutile. En termes de complexité, l'algorithme de Joye nécessite l doublements et l additions de points.

Algorithme 6 Méthode Double-and-Add de Joye pour le calcul d'une multiplication scalaire

Require: $k = (k_{l-1}, \cdots, k_1, k_0)$, $P \in E(F_p)$
Ensure: $Q = k \cdot P$
  $Q_0 \leftarrow O$
  $Q_1 \leftarrow P$
  for $i \leftarrow 0$ *to* $l-1$ **do**
    $b \leftarrow 1 - k_i$
    $Q_b \leftarrow 2Q_b + Q_{k_j}$
  end for
  return Q_0 ;

D'autres solutions pour contrer les attaques SPA et DPA sont proposées dans la littérature [94, 95]. Une étude complète des attaques par canaux cachés et contremesures est donnée dans [89].

3.2 Techniques mathématiques

3.2.1 Optimisation des calculs arithmétiques

Tous les calculs pour la cryptographie des courbes elliptiques se basent sur l'arithmétique du corps fini F_p. L'opération de multiplication doit donc être efficace. Le multiplieur utilisé est de type : $c = a \times b \mod p$ avec a, b, c et p sont des entiers de taille entre 160 et 512 bits. La constante p représente le nombre premier associé à la courbe elliptique utilisée. Son choix est donc primordial pour la réduction de la complexité des opérations arithmétiques notamment la réduction suivant une multiplication.

A cet effet, des nombres premiers spéciaux ont été proposés et recommandés par le NIST (National Institute of Standards & Technologies) sur les corps finis premiers (F_p) introduits dans le standard FIPS [96] et qui décrit trois types de courbes P-256, P-384 et P-521. Par exemple, le p utilisé pour la courbe P-256 est le nombre premier de Mersenne : $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. La réduction modulaire avec les nombres premiers spéciaux est réalisée plus rapidement. D'autres formes de nombres ont été également proposées comme les nombres premiers aléatoires qui permettent d'optimiser la réduction modulo p.

Une autre méthode d'optimisation consiste à mixer la réduction et la multiplication pour le calcul efficace d'une multiplication modulo p, elle est appelée la méthode du

multiplieur de Montgomery [97]. La différence entre les multiplieurs classiques et le multiplieur de Montgomery est donnée par la Figure 3.2.

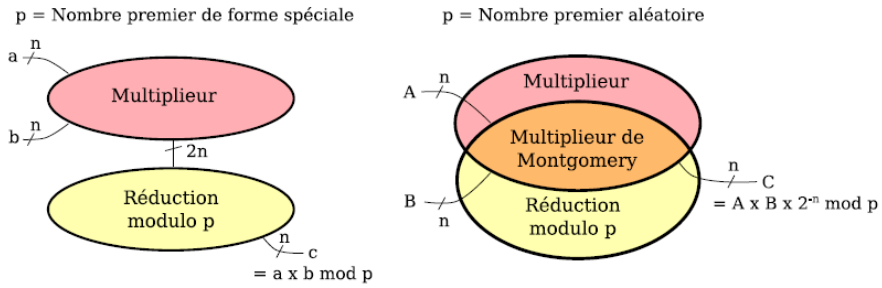

FIGURE 3.2 – Le multiplieur de Montgomery

Les travaux de Pontié dans [98] peuvent permettre au lecteur d'avoir plus de détails sur l'optimisation des calculs arithmétiques.

3.2.2 Système de coordonnées

Dans le système de coordonnées affines, les formules d'addition et de doublement font intervenir des opérations d'inversion de points dans F_p considérées comme des traitements très couteux sur les corps finis. Afin d'éviter l'inversion de points, nous allons présenter d'autres systèmes de coordonnées. Les coordonnées Jacobéennes sont plus souples, en termes de complexité (voir le tableau 3.1), où les calculs ne nécessitent aucune opération d'inversion de points. D'autres optimisations sont faites également sur les systèmes de coordonnées Jacobéennes et que nous allons présenter dans cette section. Il s'agit des coordonnées co-Z proposée par Meloni [82] où il considère que deux points d'entrée partageant la même coordonnée Z. Après avoir présenté le système de coordonnées affines à la section 2.2.4, nous présentons également dans cette section le système de coordonnées mixtes (un point en coordonnées Jacobéennes et un point en coordonnées affines) et nous allons clore par une comparaison des performances de chaque système de coordonnées.

En coordonnées Jacobiennes

Les formules d'addition et de doublement en coordonnées affines font intervenir des inversions dans F_p, une opération compliquée et coûteuse dans les corps finis. Dans ce travail, nous avons éliminé les inversions de points en choisissant de travailler sur les systèmes de coordonnées Jacobiennes. Avec le système de coordonnées Jacobiennes, un point est représenté par 3 coordonnées $(X : Y : Z)$ qui correspondent au point affine $(\frac{X}{Z^2}, \frac{Y}{Z^3})$. Si on remplace x par $\frac{X}{Z^2}$ et y par $\frac{Y}{Z^3}$ dans l'équation 2.2 on obtient la forme projective de l'équation de Weierstrass :

$$Y^2Z = X^3 + aXZ^2 + bZ^3 \tag{3.1}$$

Soient $P = (X_P, Y_P, Z_P)$ et $Q = (X_Q, Y_Q, Z_Q)$, les formules de calcul d'un doublement et d'une addition de points en coordonnées Jacobiennes sont données comme suit :

— **Doublement de points** : Un doublement en coordonnées jacobiennes est calculé comme suit : $2P(X_P : Y_P : Z_P) = Q(X_Q : Y_Q : Z_Q)$ avec :

$$X_Q = -2(4X_PY_P^2) + (3X_P^2 + aZ_P^4)^2$$
$$Y_Q = -8Y_P^4 + (3X_P^2 + aZ_P^4)(4X_PY_P^2 - X_Q^3)$$
$$Z_Q = 2Y_PZ_P$$

— **Addition de points** : Une addition en coordonnées jacobiennes est calculée comme suit : $P(X_P : Y_P : Z_P) + Q(X_Q : Y_Q : Z_Q) = R(X_W : Y_W : Z_W)$ avec :

$$X_W = -(X_QZ_P^2 - X_PZ_Q^2)^3 - 2(X_PZ_Q^2)(X_QZ_P^2 - X_PZ_Q^2)^3 + (Y_QZ_P^3 - Y_PZ_Q^3)^2$$
$$Y_W =$$
$$-(Y_PZ_Q^3)(X_QZ_P^2 - X_PZ_Q^2)^3 + (Y_QZ_P^3 - Y_PZ_Q^3)(X_PZ_Q^2)(X_QZ_P^2 - X_PZ_Q^2)^2 - X_W^4)$$
$$Z_W = Z_PZ_Q(X_QZ_P^2 - X_PZ_Q^2)$$

En coordonnées co-Z

co-Z addition, proposée par [82] est une optimisation supplémentaire de l'addition en coordonnées jacobiennes. Meloni, dans [82], considère que deux points d'entrée partageant la même coordonnée Z. Soit $P = (X_P, Y_P, Z)$ et $Q = (X_Q, Y_Q, Z)$, les formules de calcul d'un doublement et d'une addition de points en coordonnées co-Z sont données comme suit :

— **Doublement de points** : Un doublement en coordonnées co-Z est calculé comme suit : $2P(X_P : Y_P : Z) = Q(X_Q : Y_Q : Z)$ avec :

$$X_Q = (3X_P^2 + a)^2 - 2(4X_PY_P^2)$$
$$Y_Q = (3X_P^2 + a)((4X_PY_P^2) - X_Q) - 8Y_P^4$$

— **Addition de points avec mise à jour des coordonnées** : Une addition en coordonnées co-Z avec mise à jour des coordonnées est calculée comme suit : $P(X_P : Y_P : Z) + Q(X_Q : Y_Q : Z) = R(X_W : Y_W : Z)$ avec :

$$X_W = (Y_Q - Y_P)^2 - (X_P(X_Q - X_P)^2 + X_Q(X_Q - X_P)^2)$$
$$Y_W = (Y_Q - Y_P)(X_P(X_Q - X_P)^2 - X_W) - Y_P(X_Q(X_Q - X_P)^2 - X_P(X_Q - X_P)^2)$$
$$Z = Z(X_Q - X_P)$$

— **Addition-soustraction de points simplifiée** : Une addition et une soustraction en coordonnées co-Z avec des formules d'addition et de soustraction simplifiées sont calculées respectivement comme suit : $P(X_P : Y_P : Z) + Q(X_Q : Y_Q : Z) = W(X_W : Y_W : Z)$ et $P(X_P : Y_P : Z) - Q(X_Q : Y_Q : Z) = R(X_R : Y_R : Z)$ avec :

$$X_W = (Y_Q - Y_P)^2 - (X_P(X_Q - X_P)^2 + X_Q(X_Q - X_P)^2)$$
$$Y_W = (Y_Q - Y_P)(X_P(X_Q - X_P)^2 - X_W) - Y_P(X_Q(X_Q - X_P)^2 - X_P(X_Q - X_P)^2)$$
$$X_R = (Y_P + Y_Q)^2 - (X_P(X_Q - X_P)^2 + X_Q(X_Q - X_P)^2)$$
$$Y_R = (Y_P - Y_Q)(X_R - X_P(X_Q - X_P)^2) - Y_P(X_Q(X_Q - X_P)^2 - X_P(X_Q - X_P)^2)$$
$$Z = Z(X_Q - X_P)$$

— **Doublement et addition** : Un doublement suivi d'une addition en coordonnées co-Z est calculé comme suit : $P(X_P : Y_P : Z) + Q(X_Q : Y_Q : Z) = R(X_W : Y_W : Z)$ avec :

$$X_W = (Y_P(X_Q((X_Q - X_P)^2 - X_P((X_Q - X_P)^2)) - (((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2))((X_P((X_Q - X_P)^2 - ((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2)))^2) + X_P((X_Q - X_P)^2((X_P((X_Q - X_P)^2 - ((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2)))^2))$$
$$Y_W = (Y_P(X_Q((X_Q - X_P)^2 - X_P((X_Q - X_P)^2) - ((Y_Q - Y_P)(X_P((X_Q - X_P)^2 - ((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2))) - Y_P(X_Q((X_Q - X_P)^2 - X_P((X_Q - X_P)^2)))((((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2))(X_P((X_Q - X_P)^2 - ((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2)))^2) - X_W) - ((Y_Q - Y_P)(X_P((X_Q - X_P)^2 - ((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2))) - Y_P(X_Q((X_Q - X_P)^2 - X_P((X_Q - X_P)^2))(X_P((X_Q - X_P)^2(X_P((X_Q - X_P)^2 - ((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2)))^2 - ((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2))((X_P((X_Q - X_P)^2 - ((Y_Q - Y_P)^2 - (X_P((X_Q - X_P)^2 + X_Q((X_Q - X_P)^2)))^2)))$$

En coordonnées mixtes

Soit $P(X_P : Y_P : Z_P)$ un point elliptique en coordonnées jacobiennes et $Q(x_Q, y_Q)$ un point elliptique en coordonnées affines, l'addition $P + Q = R$ est un point elliptique en coordonnées jacobiennes $R(X_W : Y_W : Z_W)$ calculé comme suit :

$$X_W = (Y_Q Z_P^3 - Y_P)^2 - (X_Q Z_P^2 - X_P)^3 + 2X_P(X_Q Z_P^2 - X_P)^2$$
$$Y_W = (Y_Q Z_P^3 - Y_P)(X_P(X_Q Z_P^2 - X_P)^2 - X_W) - Y_P(X_Q Z_P^2 - X_P)^3$$
$$Z_W = Z_P(X_Q Z_P^2 - X_P)$$

Le tableau 3.1 résume les opérations de doublement et d'addition de points dans les différents types de coordonnées.

TABLEAU 3.1 – Doublement et addition de points dans les différents types de coordonnées

	Jacobiennes		co-Z					Mixtes
Opération	P + Q	2P	2P	P + Q	P+Q | P-Q		2P + Q	P + Q
A	$X_P Z_Q^2$	$4X_P Y_P^2$	$4X_P Y_P^2$	$(X_Q - X_P)^2$	$(X_Q - X_P)^2$		$(X_Q - X_P)^2$	$X_Q Z_P^2$
B	$X_Q Z_P^2$	$3X_P^2 + AZ_P^4$	$3X_P^2 + a$	$X_P A$	$X_P A$		$X_P A$	$Y_Q Z_P^3$
C	$Y_P Z_Q^3$	$A - X_W$	/	$X_Q A$	$X_Q A$		$X_Q A$	$A - X_P$
D	$Y_Q Z_P^3$	/	/	$(Y_Q - Y_P)^2$	$(Y_Q - Y_P)^2$		$(Y_Q - Y_P)^2$	$B - Y_P$
X_W	$-(B-A)^3 - 2A(B-A)^3 + (D-C)^2$	$-2A + B^2$	$B^2 - 2A$	$D-(B+C)$	X_W	$D - (B - C)$	$(Y_P(C - B) - ((D - (B + C))((B - (D - (B + C)))^2) + B((B - (D - (B + C)))^2))$	$D^2 - (C^3 + 2X_P C^2)$
					X_R	$(Y_P + Y_Q)^2 - (B + C)$		
Y_W	$-C(B-A)^3 + (D-C)(A(B-A)^2 - X_W)$	$-8Y_P^4 + BC$	$B(A - X_W) - 8Y_P^4$	$(Y_Q - Y_P)(B - X_W) - Y_P(C - B)$	Y_W	$((Y_P - Y_Q)(B - X_W)) - (Y_P(C - B))$	$(Y_P(C-B) - ((Y_Q - Y_P)(B - (D - (B + C))) - Y_P(C - B)))(((D-(B+C))(B-(D-(B+C)))^2) - X_W) - ((Y_Q - Y_P)(B - (D - (B + C))) - Y_P(C-B))(B(B-(D-(B+C)))^2 - (D-(B+C))((B-(D-(B+C)))^2)))$	
					Y_R	$(Y_P - Y_Q)(X_R - B) - Y_P(C - B)$		
Z_W	$Z_P Z_Q (B - A)$	$2Y_P Z_P$	Z	$Z(X_Q - X_P)$	Z		$2Z$	$Z_P C$
Coût	12M+4S+7A	4M+6S+8A	2M+4S+10A	4M+2S+7A	5M+3S+11A		8M+6S+31A	9M+3S+6A

3.2.3 Réduction du scalaire

Le recodage du scalaire k est un domaine important. La nouvelle valeur de k est utilisée dans beaucoup de méthodes de calculs de multiplication scalaire comme NAF, w-NAF, etc., afin de contrer d'un côté les attaques de type SCA et de l'autre accélérer les calculs cryptographiques en réduisant le nombre d'opérations de doublement et d'addition de points. Le recodage de k permet de se protéger contre les attaques SCA en proposant une représentation régulière du scalaire k en rendant aléatoire son écriture. Comme exemple, Chabrier dans ses travaux de thèse dans [99] propose de recoder le scalaire k comme suit : $k = \sum_{i=0}^{j} a_{(i,j)} \cdot 2^i 3^j$.

3.3 Techniques algorithmiques

L'utilisation de clés courtes constitue l'un des avantages de la cryptographie des courbes elliptiques (ECC), et permet des calculs rapides sur des dispositifs limitées en ressources mémoire et puissance de calcul tels que les systèmes embarqués. Cependant, la complexité des calculs d'une multiplication scalaire, considérée comme l'opération centrale dans les calculs elliptiques, ralentit les performances des protocoles cryptographiques et par conséquent celles de l'ensemble du cryptosystème. Afin de réduire encore davantage cette complexité, nous allons présenter dans cette section quelques techniques algorithmiques utilisées pour la réduction du nombre de doublement et d'additions de points, et pour l'accélération des calculs cryptographiques par changement de coordonnées.

3.3.1 Réduction du nombre de doublements et d'additions de points

Dans cette section, nous allons présenter quelques techniques algorithmiques permettant, principalement, la réduction du nombre d'opérations de doublement et d'addition de points. La plupart de ces techniques reposent sur le principe du recodage du scalaire k, une technique mathématique déjà présenté à la section 3.2.3.

Méthode NAF

L'une des méthodes efficaces pour réduire le nombre de doublements et d'additions de points est la méthode NAF (Non-Adjacent Form) [100], l'idée étant de réduire le nombre de bits positifs du scalaire k. La méthode NAF permet d'effectuer un doublement suivi d'une addition si le bit est égal à 1 ou une soustraction si le bit est égal à -1. Le nouveau scalaire k, noté $NAF(k)$, est généré comme suit : soit k un entier tel que $k = 2^i - 1$ qui s'écrit en binaire $(k)_2 = 1(11\ldots1)$ (i fois). Le scalaire k peut

donc s'écrire : $NAF(k) = 1(00\cdots00-1)$ (i+1 bits) et qui s'écrit : pour $k \in N$: $k = (k_{l-1}, \cdots, k_1, k_0)$ avec $k_i \in \{0,1\}$ et $NAF(k) = \sum_{i=0}^{l-1} k'_i 2^i$ avec $k'_i \in \{-1,0,1\}$. Généralement, et sans faire de démonstration dans ce document, le nombre de bits non nuls est $l/3$. L'algorithme 7 présente la méthode de calcul de $NAF(k)$.

Algorithme 7 Conversion de n'importe quel entier en NAF(k)

Require: un entier k
Ensure: NAF(k)

1. Soit $h_{n-1}h_{n-2}\ldots h_0$ la représentation binaire de 3k
2. Soit $k_{n-1}k_{n-2}\ldots k_0$ la représentation binaire de k
3. Pour $i = 1$ à $l-1$ Faire
4. $\quad g_{i-1} = h_i - k_i$
5. finPour

return $g = g_{n-2}g_{n-3}\ldots g_0$;

L'algorithme 8 calcule la multiplication scalaire en utilisant Double-and-Add avec la méthode NAF [100]. Le coût de la soustraction $Q = Q - P$ est équivalent au coût de l'addition $Q = Q + (-P)$ où $(-P)$ est le point inverse de P avec $P = (X, Y)$ et $-P = (X, -Y)$.

Algorithme 8 Algorithme Double-and-Add avec la méthode NAF

Require: $NAF(k) = \sum_{i=0}^{l-1}(k_i 2^i)$, P $\in$ E(F_p)
Ensure: Q = k·P

```
Q = ∅
for i ← l − 1 downto 0 do
  Q ← 2Q
  if k_i = 1 then
    Q ← Q + P
  end if
  if k_i = −1 then
    Q ← Q − P
  end if
end for
return Q ;
```

L'algorithme Double-and-add avec la méthode NAF nécessite en moyenne l doublements et $l/3$ additions de points ainsi que le temps de conversion du scalaire k en $NAF(k)$, jugé faible [100]. La méthode NAF peut être généralisée avec une méthode de fenêtrage. Au lieu de traiter bit par bit pour effectuer un doublement si le bit vaut 0 et un doublement et une addition si le bit vaut 1, on traite le scalaire k bloc par bloc (c'est-à-dire w-bits par w-bits). Ainsi, k est divisé en m blocs tel que chaque bloc i correspond à un entier Nb_i sur w bits. Les blocs peuvent être de taille fixe ou variable.

La méthode w-NAF

En généralisant la méthode NAF par le découpage du scalaire k, au lieu de limiter le codage de k à $\{-1, 0, 1\}$, on utilise $NAF_w(k)$ en m blocs de tailles fixes, représentant le scalaire k par $\{-2^{w-1}+1, \cdots, -5, -3, -1, 0, 1, 3, 5, ..., 2^{w-1}-1\}$, calculés par l'équation 3.2. $NAF_w(k)$ est une nouvelle méthode par fenêtres utilisant une forme non adjacente du scalaire k. Pour $w = 2$, les deux méthodes NAF et w-NAF sont équivalentes et $NAF(k) = NAF_2(k)$.

$$NAF_w(k) = \sum_{0}^{l-1} k_i 2^i P \quad avec \mid k_i \mid \; < 2^{w-1} \tag{3.2}$$

L'algorithme 9 décrit le fonctionnement de la méthode w-NAF [101].

Algorithme 9 Méthode Window NAF (w-NAF)

Require: $w, NAF_w(k) = \sum_{i=0}^{l-1}(k_i 2^i)$, P $\in$ E(F_p)
Ensure: Q = k·P
  Calculer $P_i = iP$, pour $i = \{1, 3, 5, \ldots, 2^{w-1} - 1\}$
  $Q = \emptyset$
  for $i \leftarrow l - 1$ downto 0 **do**
    $Q \leftarrow 2Q$
    if $k_i \neq 0$ **then**
      if $k_i > 0$ **then**
        $Q \leftarrow Q + P_{ki}$
      else
        $Q \leftarrow Q - P_{ki}$
      end if
    end if
  end for
  return Q;

L'algorithme 9 effectue en moyenne $(l - 1)$ doublements et $l/(w + 1)$ additions.

Méthode par fenêtrage signé

Une autre solution permet d'utiliser un système de fenêtrage signé (Regular signed window) [86]. Cette solution se base sur le même principe que w-NAF. Une clé $k(k_0, k_1, \cdots, k_{l-1})_2$ de longueur $n \in N$ est découpée en $k(d_{l-1}, d_{l-2}, \cdots, d_0)_{2w}$ avec $l = (n/w)$ et $d_{l-1} \neq 0$ d'où $k = \sum_i d_i 2^{iw}$ et $d_i \in (0, 1, \cdots, 2^w - 1)$. Le fonctionnement est détaillé par l'algorithme 10.

Le coût des calculs en nombre de doublements et d'additions pour une multiplication scalaire par un point des trois techniques algorithmiques est donné par le tableau 3.2.

Algorithme 10 Méthode par un fenêtrage signé (Regular signed window)

Require: k = $(k_{l-1},..., k_1, k_0)_{2^w}$, P ∈ E($F_p$)
Ensure: Q = k . P
  for all $d \in B^+$ **do**
    $R_d = d.P$
  end for
  $d \leftarrow d_{l-1}$
  $Q \leftarrow R_d$
  for $i \leftarrow l-2$ downto 0 **do**
    $d \leftarrow\mid d_i \mid$
    $Q \leftarrow 2^w.Q$
    if $d_i \geq 0$ **then**
      $Q \leftarrow Q + R_d$
    else
      $Q \leftarrow Q - R_d$
    end if
  end for
  return (Q) ;

TABLEAU 3.2 – coût des calculs en nombre de doublements et d'additions pour une multiplication scalaire pour les méthodes Double-and-Add, NAF, w-NAF et Regular Signed Window

Opération	Double-and-Add	NAF	Regular signed window	w-NAF
Doublement	l	l	$w(l-1)$	$l-1$
Addition	$l/2$	$l/3$	$l/(w+1)$	$l/(w+1)+2w-2$

3.3.2 Accélération de calculs par le système de coordonnées Jacobiennes

Dans cette section, nous allons présenter quelques algorithmes reposant sur le système de coordonnées jacobiennes pour l'accélération des calculs elliptiques. Nous nous intéresserons très particulièrement aux travaux de Montgomery et ceux de Joye.

Algorithmes de Montgomery

Les auteurs de [102] ont présenté dans leurs travaux sur l'accélération des calculs elliptiques un nouvel algorithme rapide et efficace basé sur l'algorithme de Montgomery [92] pour les courbes elliptiques courtes de Weierstrass. Le fonctionnement est détaillé par l'algorithme 11.

Algorithme 11 Nouvelle variante de l'algorithme Montgomery ladder pour l'accélération des calculs elliptiques

Require: k = $(k_{l-1},..., k_1, k_0)_2$, P $\in$ E(F_p)
Ensure: Q = k·P
  $R_0 = \emptyset$
  R_1 = P
  for $i \leftarrow l-1$ downto 0 **do**
    $b \leftarrow k_i$
    $R_{1-b} \leftarrow R_{1-b} + R_b$
    $R_b \leftarrow 2 \times R_b$
  end for
  return R_0 ;

La complexité de l'algorithme 11 est de 16M + 10S + 15A et peut être améliorée pour n'exécuter que 6M + 6S + 20 A par bit. Cette nouvelle optimisation est le résultat d'une nouvelle méthode de calcul de doublement-addition de points appelée LADD [102] et un nouveau système de coordonnées appelé on-the-flyadaptivecoordinates [102] appliqué à l'algorithme initial de Montgomery [92].

Algorithmes de Joye

Joye dans [93] propose différentes optimisations faites sur l'algorithme standard Double-and-Add. Nous présentons dans cette section une alternative basée sur le système de coordonnées jacobiennes standards détaillée par l'algorithme 12, avec une complexité de 11M + 7S + 27A.

Algorithme 12 Algorithme de Joye pour accélérer les calculs de l'algorithme Double-and-add en coordonnées jacobiennes standards

Require: k = $(k_{l-1},..., k_1, k_0)_2$, P ∈ E($F_p$)
Ensure: Q = k·P
  $R_0 = \emptyset$
  R_1 = P
  for $i \leftarrow 0$ to $l-1$ **do**
    $b \leftarrow k_i$
    $R_{1-b} \leftarrow 2 \times R_{1-b} + R_b$
  end for
  return R_0 ;

3.4 Techniques de parallélisation des calculs

L'objectif primaire de la parallélisation est de diminuer les temps de calcul et de réduire la complexité de certaines tâches qui peuvent être découpées en sous tâches. L'asynchronisme permet de minimiser les contraintes de la synchronisation, les délais de communications ainsi que la flexibilité des calculs. Dans les traitements coopératifs effectués par des nœuds multifonctionnels, le choix du parallélisme asynchrone permet à un nœud d'accomplir plusieurs tâches (calculs, capture, radio, etc) sans interruption et sans délais d'attente. Dans cette section, nous nous intéresserons uniquement à la parallélisation des calculs elliptiques.

3.4.1 Parallélisation avec des points précalculés

L'idée consiste à découper le scalaire k en plusieurs fragments. Soit k un entier binaire de longueur l exprimé sous la forme :

$$k = \sum_{i=0}^{l-1} k_i 2^i = k_{l-1}2^{l-1} + \ldots + k_1 2^1 + k_0 \tag{3.3}$$

Ainsi, le calcul de la multiplication scalaire est réalisé comme suit :

$$Q = k \cdot P = \sum_{i=0}^{l-1} k_i 2^i = (k_{l-1}2^{l-1} + \cdots + k_1 + k_0)P \tag{3.4}$$

Le point Q peut-être découpé en n segments, chaque segment est calculé par la suite soit par un cœur d'un processeur dans une architecture multi-cœurs ou bien par d'autres capteurs, comme suit :

$$Q = \underbrace{\sum_{i=0}^{l/n-1} k_i P^i}_{C_1} + \underbrace{\sum_{i=l/n}^{2l/n-1} k_i P^i}_{C_2} + \cdots + \underbrace{\sum_{i=l-l/n}^{l-1} k_i P^i}_{C_n} \tag{3.5}$$

Cette méthode permet de distribuer les calculs d'une seule multiplication scalaire $Q = k \cdot P$ en stockant l/n points précalculés, ce qui engendre des besoins importants en ressources mémoire [103]. Elle est conçue pour la multiplication de point fixe et convient ainsi à la cryptographie des courbes elliptiques car le point générateur de la courbe est souvent prédéfini et ne change pas durant toute la durée de vie du cryptosystème.

Une autre méthode basée sur les points précalculés en coordonnées jacobiennes est proposée dans [104]. La longueur du scalaire k est fixée à 160 bits. A l'étape d'initialisation, deux tableaux A et B contenant les 62 points précalculés sont générés comme suit :

$$A[s] = \sum_{j=0}^{4} a_{s,j} 2^{32j} P$$
$$B[s] = \sum_{j=0}^{4} a_{s,j} 2^{16+32j} P$$

Où $s \in [\![1; 31]\!]$ et $a_{s,j}$ avec $j \in [\![0; 4]\!]$ est la représentation binaire de $s = \sum_{j=0}^{4} a_{s,j} 2^j$. L'algorithme 13 montre le principe de fonctionnement de cette méthode.

Algorithme 13 Multiplication scalaire parallèle basée sur des points précalculés

Require: k $= (k = \sum_{i=0}^{l-1}(k_i 2^i)$, $P \in E(F_p)$
Ensure: $Q = k \cdot P$
  for $j \leftarrow 0$ to 15 **do**
    $u_j = \sum_{i=0}^{4} k_{32i+j} 2^i$
    $v_j = \sum_{i=0}^{4} k_{32i+16+j} 2^i$
  end for
  $A[0] \leftarrow \infty$
  $B[0] \leftarrow \infty$
  $T \leftarrow \infty$
  for $i \leftarrow 15$ to 0 **do**
    $T \leftarrow 2T$
    $T \leftarrow T + A[u_i] + B[v_i]$
  end for
  return T;

Les calculs dans la boucle peuvent être parallélisés et distribués sur des processeurs différents.

3.4.2 Parallélisation sans points précalculés

L'inconvénient de la méthode avec des points précalculés, décrite précédemment à la section 3.4.1 ci-dessus, est le stockage important d'une énorme quantité de données

dans la mémoire des dispositifs participants aux calculs parallèles d'une multiplication scalaire. Dans cette section, nous allons proposer une autre méthode de parallélisation de calculs sans la sauvegarde et le calcul de points au préalable. Le principe est le suivant.

Soient P et Q deux points en coordonnées jacobiennes d'une courbe elliptique définie sur un corps fini et k un scalaire de longueur l qu'on peut découper en n segments S_i. Chaque segment S_i doit être multiplié avec un point $G_i = 2^{ib}P$ où i est l'indice du segment et $b = l/n$ sa longueur. Par exemple, si on veut paralléliser les calculs entre 4 processeurs, il faudrait calculer les points $P_1 = 2^{40}P$, $P_2 = 2^{80}P$ et $P_3 = 2^{120}P$ utilisés par $P_4 = 2^{160}P = k \cdot P$ pour un scalaire de longueur 160 bits.

Néanmoins, cette méthode peut s'avérer gourmande en consommation mémoire si le nombre de dispositifs participants est insuffisant et/ou la longueur du scalaire est très importante. Cette méthode peut être améliorée en utilisant l'algorithme de quadruplement proposé par [105] qui permet le calcul direct de 2^xP en se basant sur l'algorithme standard Double-and-Add. Le principe de l'algorithme du quadruplement est montré par l'algorithme 14.

Algorithme 14 Algorithme de quadruplement en coordonnées jacobiennes

Require: $G\prime(X_1, Y_1, Z_1)$, $x \in Z^+$
Ensure: $2^xG\prime(X_{2^x}, Y_{2^x}, Z_{2^x})$
  $\alpha_1 \leftarrow X_1$
  $\beta \leftarrow 3X_1^2 + a$
  $\gamma \leftarrow -Y_1$
  for $i \leftarrow 2$ to x **do**
    $\alpha_i \leftarrow \beta_{i-1}^2 - 8\alpha_{i-1}\gamma_{i-1}^2$
    $\beta_i \leftarrow 3\alpha_i^2 + 16^{i-1}a(\prod_{j=1}^{i-1}\gamma_j)^4$
    $\gamma_i \leftarrow -8\gamma_{i-1}^4 - \beta_{i-1}(\alpha_i - 4\alpha_{i-1}\gamma_{i-1}^2)$
  end for
  $\omega_x \leftarrow 12\alpha_x\gamma_x^2 - \beta_x^2$
  $X_{2^x} \leftarrow \beta_x^2 - 8\alpha_x\gamma_x^2$
  $Y_{2^x} \leftarrow 8\gamma_x^4 - \beta_x\omega_x$
  $Z_{2^x} \leftarrow 2x\prod_{i=1}^{x}\gamma_i$
  return $(X_{2^x}, Y_{2^x}, Z_{2^x})$;

3.4.3 Parallélisation des opérations arithmétiques

Dans [106], une nouvelle méthode de parallélisation des opérations de doublement et d'addition d'une multiplication scalaire $(2P, P+Q)$ est proposée. Soient $P(x_P, y_P) = (X_P : Y_P : Z_P)$ et $Q = (x_Q, y_Q) = (X_Q : Y_Q : Z_Q)$ deux points en coordonnées respectivement affines et jacobiennes d'une courbe elliptique définie sur un corps fini. $P\prime(x_P\prime, y_P\prime) = (X_P\prime : Y_P\prime : Z_P\prime) = 2P$ et $Q\prime = (x_Q\prime, y_Q\prime) = (X_Q\prime : Y_Q\prime : Z_Q\prime) = P + Q$ sont calculés comme suit :

$$\begin{cases} X_{P\prime} &= 2(X_P Z_Q + X_Q Z_P)(X_P X_Q + a Z_P Z_Q) + 4b Z_P^2 Z_Q^2 - x_p (X_P Z_Q - X_Q Z_P)^2 \\ Z_{P\prime} &= (X_P Z_Q - X_Q Z_P)^2 \\ X_{Q\prime} &= (X_Q^2 - a Z_Q^2)^2 - 8b X_Q Z_Q^3 \\ Z_{Q\prime} &= 4(X_Q Z_Q (X_Q^2 + a Z_Q^2) + b Z_Q^4) \end{cases}$$

en ignorant les Y-coordonnées. L'algorithme parallèle de calcul de (2P, P + Q) est donné par l'algorithme 15.

Algorithme 15 Algorithme de calcul parallèle de $(P\prime, Q\prime) = (P + Q, 2Q)$

Require: (X_P, Z_P, X_Q, Z_Q)
Ensure: $(X_{P\prime}, Z_{P\prime}, X_{Q\prime}, Z_{Q\prime})$
$R_0 \leftarrow X_P, R_1 \leftarrow Z_P, R_2 \leftarrow X_Q, R_3 \leftarrow Z_Q$

1.	$R_6 \leftarrow R_2 \cdot R_1$	2.	$R_7 \leftarrow R_3 \cdot R_0$
3.	$R_4 \leftarrow R_7 + R_6$	4.	$R_5 \leftarrow R_7 - R_6$
5.	$R_5 \leftarrow R_5 \cdot R_5$	6.	$R_7 \leftarrow R_1 \cdot R_3$
7.	$R_1 \leftarrow a \cdot R_7$	8.	$R_6 \leftarrow R_7 \cdot R_7$
9.	$R_0 \leftarrow R_0 \cdot R_2$	10.	$R_6 \leftarrow b \cdot R_6$
11.	$R_0 \leftarrow R_0 + R_1$	12.	$R_6 \leftarrow R_6 + R_6$
13.	$R_0 \leftarrow R_0 \cdot R_4$	14.	$R_1 \leftarrow x_P \cdot R_5$
15.	$R_4 \leftarrow R_0 + R_6$		
16.	$R_4 \leftarrow R_4 + R_4$	17.	$R_6 \leftarrow R_2 + R_2$
18.	$R_4 \leftarrow R_4 - R_1$	19.	$R_7 \leftarrow R_3 + R_3$
20.	$R_0 \leftarrow R_6 \cdot R_7$	21.	$R_1 \leftarrow R_3 \cdot R_3$
22.	$R_2 \leftarrow R_2 \cdot R_2$	23.	$R_3 \leftarrow a \cdot R_1$
24.	$R_6 \leftarrow R_2 - R_3$	25.	$R_7 \leftarrow R_2 + R_3$
26.	$R_1 \leftarrow R_1 + R_1$		
27.	$R_2 \leftarrow b \cdot R_1$	28.	$R_7 \leftarrow R_7 \cdot R_0$
29.	$R_1 \leftarrow R_2 \cdot R_1$	30.	$R_0 \leftarrow R_0 \cdot R_2$
31.	$R_6 \leftarrow R_6 \cdot R_6$		
32.	$R_6 \leftarrow R_6 - R_0$	33.	$R_7 \leftarrow R_7 + R_1$

$X_{P\prime} \leftarrow R_4, Z_{P\prime} \leftarrow R_5, X_{Q\prime} \leftarrow R_6, Z_{Q\prime} \leftarrow R_7$

3.5 Techniques de parallélisation des calculs dans les réseaux de capteurs sans fil

Les processeurs sont constitués de plusieurs cœurs que l'on retrouve dans différents dispositifs comme les laptops, les ordinateurs de bureau, etc. mais aussi dans des systèmes embarqués comme les smartphones, les tablettes, les capteurs, etc. Cette technologie a permis d'améliorer les performances du dispositif en lançant et en exécutant simultanément plusieurs tâches. Dans le domaine de la cryptographie, la parallélisation des calculs demeure possible, notamment pour la multiplication scalaire, et plusieurs

solutions ont été proposées dans la littérature. Nous pencherons dans cette section sur les plus adaptés aux réseaux de capteurs sans fil.

3.5.1 Parallélisation par décomposition des données

La méthode proposée par [107] pour le calcul parallèle dans un réseau de capteurs sans fil se base sur, et n'est pas très différente de, la méthode de calcul sans points précalculés présentée à la section 3.4.2 ci-dessus. L'objectif étant de décomposer une tâche en plusieurs sous tâches par un nœud maitre puis affecter chaque sous tâche à un esclave participant aux calculs parallèles d'une multiplication scalaire $Q = k \cdot P$ où P est un point générateur d'une courbe elliptique définie sur un corps fini et k un scalaire de longueur l.

Dans un premier temps, le nœud maitre décompose le scalaire k en n segments S_i de longueur $b = l/n$ où n est le nombre de nœuds esclaves déployés à proximité du nœud maitre et participants aux calculs parallèles. S_i est calculé comme suit :

$$S_i = \sum_{j=ib}^{ib+b-1} k_j 2^j$$

Le calcul de $Q = k \cdot P = Q_0 + Q_1 + \cdots + Q_{n-1}$ peut alors être décomposé en :

$$\begin{cases} Q_0 &= S_0 P \\ Q_1 &= S_1 2^b P \\ \vdots & \\ Q_{n-1} &= S_{n-1} 2^{b(n-1)} P \end{cases}$$

Chaque Q_i peut être calculé indépendamment par un nœud esclave vu que le générateur G est connu d'avance et ne change pas au cours du cycle de vie du cryptosystème.

3.5.2 Parallélisation par décomposition des opérations de doublement et d'addition

Une multiplication scalaire est une succession d'opérations d'addition et de doublement de points d'une courbe elliptique définie sur un corps fini. L'enchainement des deux opérations est important dans des algorithmes connus sous le nom Left-to-Right où le scalaire k est balayé de gauche à droite, du bit de poids fort au bit de poids faible. Ainsi, le résultat d'un doublement est directement utilisé par une addition. Cependant, dans les algorithmes connus sous le nom de Right-to-Left où le scalaire k est balayé de droite à gauche, du bit de poids faible au bit du poids fort, une addition de points n'utilise pas directement le résultat d'un doublement. Ainsi, les doublements de points peuvent être calculés indépendamment et en parallèle avec les additions de points. Le principe est décrit par les trois algorithmes 16, 17 et 18 qui constituent des alternatives parallèles respectivement aux algorithmes 2, 8 et 10.

Algorithme 16 Algorithme Double-and-Add de droite-à-gauche

Require: $k = (k_{l-1}, \cdots, k_1, k_0)_2, P \in E(F_p)$
Ensure: $Q = k \cdot P$
  $R_0 \leftarrow \emptyset$
  $R_1 \leftarrow P$
  for $i \leftarrow 1$ to $l - 1$ **do**
    if $k_i = 1$ **then**
      $R_0 \leftarrow R_0 + R_1$
    end if
    $R_1 \leftarrow 2R_1$
  end for
  return R_0 ;

Algorithme 17 Méthode NAF de droite-à-gauche

Require: $k \in N$, P, R $\in E(F_p)$
Ensure: $Q = k \cdot P$
  $Q = \emptyset$
  $R = P$
  while $k \geq 1$ **do**
    if $k \pmod 2 = 1$ **then**
      $u \leftarrow 2 - (k \pmod 4)$
      $k \leftarrow k - u$
      if $u = 1$ **then**
        $Q \leftarrow Q + R$
      else
        $Q \leftarrow Q - R$
      end if
    end if
    $k \leftarrow k/2$
    $R \leftarrow 2R$
  end while
  return Q

Algorithme 18 Méthode de fenêtrage signé (signed window algorithm) de droite-à-gauche

Require: $k = (k_{l-1}, \cdots, k_1, k_0)_{2^w}, P \in E(F_p)$
Ensure: $Q = k \cdot P$
  for all $d \in B^+$ **do**
    $R_d \leftarrow \emptyset$
  end for
  $Q \leftarrow P$
  for $i \leftarrow 0$ to $l-2$ **do**
    $d \leftarrow |d_i|$
    if $d_i \geq 0$ **then**
      $R_d \leftarrow R_d + Q$
    else
      $R_d \leftarrow R_d - Q$
    end if
    $Q \leftarrow 2^w.Q$
  end for
  return $(\sum_d d.R_d)$;

La complexité d'un doublement de points est beaucoup plus faible que celle d'une addition de points (voir tableau 3.1). De ce fait, le temps d'exécution d'un doublement est beaucoup plus rapide qu'une addition de points (chose que nous allons démontrer ci-après), ce qui nous a amené à lancer un certain nombre d'opérations de doublement avant de lancer les additions, puis lancer les additions en même temps que les doublements restants. Le principe étant décrit par l'algorithme 19.

Algorithme 19 Algorithme multithreads de calcul parallèle de $k \cdot P$

Require: $k = (k_{l-1}, \cdots, k_1, k_0)_{2^w}, P \in E(F_p)$
Ensure: $Q = k \cdot P$

Thread 1 : Calculer les doublements	Thread 2 : Calculer les additions
1. $D[0] \leftarrow P$	1. $Q \leftarrow \emptyset$
2. For i=1 to 21 do	2. Attendre le signal du Thread 1
3. $D[i] \leftarrow 2D[i-1]$	
4. EndFor	
5. Envoyer un signal au Thread 2	
6. For i=22 to $l-1$ do	6. For i = 0 to $l-1$ do
7. $D[i] \leftarrow 2D[i-1]$	7. If $k_i = 1$ Then
8. endFor	8. $Q \leftarrow Q + D[i]$
	9. EndIf
	10. EndFor

Return (Q)

3.6 Conclusion

L'algorithme de calcul d'une multiplication scalaire, Double-and-Add, présente plusieurs inconvénients malgré ses bonnes performances. Les deux principaux inconvénients de Double-and-Add sont sa vulnérabilité aux attaques SCA (Side-Channel Analysis) et le nombre important d'opérations de doublement et d'addition de points opérées pour les calculs elliptiques. Beaucoup de travaux d'optimisation ont été réalisés pour corriger la première vulnérabilité citée, comme la méthode 2^t-ère, l'Echelle de Montgomery, la multiplication scalaire hautement régulière de Joye etc.

Plusieurs techniques ont été proposées afin d'optimiser les performances de l'algorithme Double-and-Add. Nous avons présenté dans ce chapitre les techniques mathématiques, algorithmiques, de parallélisation de calculs, et celles adaptées aux réseaux de capteurs sans fil. Les techniques mathématiques réduisent les calculs arithmétiques et font intervenir d'autres systèmes de coordonnées, comme les coordonnées jacobiennes, afin d'éviter des opérations très couteuses telle que l'inversion de points. Les techniques algorithmiques se basent sur la réduction du nombre d'opérations de doublement et d'addition de points ainsi que l'accélération des calculs par l'emploie du système de coordonnées jacobiennes.

Les techniques de parallélisation de calculs et de traitements sont considérées très performantes car elles permettent de diminuer le temps de calcul et de réduire la complexité de certaines tâches en les découpant en sous tâches et en les distribuant sur d'autres nœuds. Une première approche consiste à découper le scalaire k en plusieurs fragments. Cette solution est peu recommandée car elle présente l'inconvénient de la violation du secret de la clé en divulguant une partie de la clé aux autres nœuds, ce qui peut profiter à des attaquants sur le réseau. Une seconde approche consiste à paralléliser les opérations de doublement et d'addition de points, rendu possible par la méthode Right-to-Left permettant à des traitements de doublement et d'addition de points de s'exécuter indépendamment.

Partant de ce dernier principe, nous allons proposer dans les prochains chapitres nos contributions pour l'optimisation des calculs elliptiques d'une multiplication scalaire. La première optimisation est décrite dans le chapitre suivant et présentera quelques algorithmes améliorés basés sur les systèmes de coordonnées jacobiennes pour le calcul d'une multiplication scalaire. Le chapitre d'après présentera une nouvelle approche distribuée et parallèle pour le calcul d'une multiplication scalaire et le dernier chapitre sera consacré à un nouveau protocole de sécurité distribué basé sur la cryptographie des courbes elliptiques pour les réseaux de capteurs sans fil.

Troisième partie

Contributions

Chapitre 4
Combinaison de techniques mathématiques et algorithmiques pour l'optimisation d'une multiplication scalaire

La cryptographie des courbes elliptiques (ECC) s'est distinguée par la taille réduite de ses clés comparativement à d'autres systèmes asymétriques comme RSA [78]. Ainsi, une clé ECC de 192 bits offre un même niveau de sécurité qu'une clé RSA de 2048 bits [78]. En conséquent, ECC est plus appropriée pour les réseaux à ressources restreintes. Cependant, la complexité des calculs d'une multiplication scalaire constitue un inconvénient de taille à leur développement sur des dispositifs faibles en ressources comme les systèmes embarqués. L'algorithme standard Double-and-Add requièrt beaucoup d'optimisations. Nous avons présenté au chapitre 3 précédent quelques techniques mathématiques et algorithmiques pour contrer à la fois les attaques de type SCA et réduire la complexité des calculs.

Dans ce chapitre, nous allons combiner les techniques mathématiques basées sur les systèmes de coordonnées aux techniques algorithmiques réduisant le nombre de doublements et d'additions de points ainsi que d'autres techniques pour accélérer les calculs elliptiques. Nous avons donc travaillé sur quatre méthodes principales et largement utilisées pour le calcul d'une multiplication scalaire : Double-and-Add, Montgomery, Joye et NAF, et nous les avons combinées à trois systèmes de coordonnées : coordonnées jacobiennes standards, coordonnées mixtes affines-jacobiennes et coordonnées co-Z jacobiennes. Nous avons obtenu plusieurs variantes que nous en avons évalué la complexité et le coût. Nous les avons ensuite implémenté sur une carte Arduino Embedded System considéré comme un vrai système embarqué de par ses caractéristiques proches de celles des capteurs et autres dispositifs implantés sur les réseaux à ressources restreintes.

Les résultats obtenus montrent que les coordonnées co-Z offrent de meilleures performances, comparativement aux autres systèmes évalués dans ce travail, combinées à la méthode de Montgomery pour le calcul d'une multiplication scalaire. Pour cela, nous avons présenté d'abord les différentes variantes obtenues des différentes combinaisons de techniques mathématiques/algorithmiques, et nous avons évalué et comparé leurs complexités théoriques. Ensuite, nous avons présenté l'environnement des expérimentations suivi de l'évaluation des performances de toutes les opérations utilisées dans le calcul d'une multiplication scalaire et nous en avons terminé par une conclusion.

4.1 Algorithmes optimisés pour l'accélération de la multiplication scalaire

La multiplication scalaire est l'opération centrale, la plus coûteuse et la plus complexe, de la cryptographie des courbes elliptiques. C'est une opération qui fait intervenir des milliers de calculs arithmétiques et des centaines de doublement et d'addition de points. Nous avons présenté au chapitre 3 plusieurs techniques pour accélérer les calculs elliptiques d'une multiplication scalaire. Nous avons étudié les techniques mathématiques qui consistent à optimiser les calculs arithmétiques, à éviter certaines opérations coûteuses comme l'inversement de points par le changement du système de coordonnées et à réduire le scalaire k. Nous avons également étudié quelques techniques algorithmiques pour la réduction du nombre de doublements et d'additions de points. Certes, les solutions proposées sont efficaces mais requièrent encore des optimisations car les calculs demeurent complexes par rapports aux dispositifs utilisés de nos jours, de plus en plus miniaturisés et faibles en ressources, ajoutons à cela les besoins croissants en sécurité.

Dans cette section, nous allons présenter quelques combinaisons de quelques algorithmes d'accélération des calculs dans une multiplication scalaire avec les différents systèmes de coordonnées [108]. Ainsi, plusieurs améliorations sont apportées aux algorithmes présentés à la section 3.3 combinés aux techniques mathématiques présentées à la section 3.2, sont proposées, réduisant davantage la complexité des calculs d'une multiplication scalaire.

4.1.1 Optimisations en coordonnées jacobiennes standard

L'algorithme standard d'une multiplication scalaire, Double-and-Add, nécessite en moyenne l doublements et $l/2$ additions de points comprenant $(3l-2)/2$ inversion de points en coordonnées affines. Une inversion de points est une opération très couteuse en termes de calculs. Afin de l'éviter, d'autres systèmes de coordonnées ont été proposés. Dans cette section, nous présenterons deux alternatives : une variante de l'algorithme Double-and-Add, présenté à la section 3.1 par l'algorithme 2, et une autre variante de l'algorithme de Montgomery, présenté à la section 3.3.2 par l'algorithme 11, en coordonnées jacobiennes standards, détaillées respectivement par les algorithmes 22 et 23.

Les deux algorithmes 22 et 23 utilisent les deux méthodes StdJDbl pour le calcul d'un doublement de points et la méthode StdJAdd pour le calcul d'une addition de points en coordonnées jacobiennes standards, détaillées respectivement par les algorithmes 20 et 21 ci-dessous. Soient $P(X_1 : Y_1 : Z_1)$ et $Q(X_2 : Y_2 : Z_2)$ deux points d'une courbe elliptique définie sur un corps fini premier F_P, le doublement $2P(X_3 : Y_3 : Z_3)$

est donné par l'algorithme 20.

Algorithme 20 StdJDbl : Doublement de points (2P) en coordonnées jacobiennes standards

Require: X_1, Y_1, Z_1
Ensure: 2P
$A = X_1 Y_1^2$
$B = (3X_1^2 + aZ_1^4)/2$
$X_3 = B^2 - 2A$
$Y_3 = B(A - X_3) - Y_1^4$
$Z_3 = Y_1 Z_1$
return (X_3, Y_3, Z_3);

La méthode stdJDbl nécessite 4M + 6S + 8A. L'addition $P + Q = (X_3 : Y_3 : Z_3)$ est donnée par l'algorithme 21.

Algorithme 21 StdJAdd : Addition de points (P+Q) en coordonnées jacobiennes standards

Require: X_1, Y_1, Z_1, X_2, Y_2, Z_2
Ensure: P+Q
$A = X_1 Z_2^2$
$B = X_2 Z_1^2$
$C = Y_1 Z_2^3$
$D = Y_2 Z_1^3$
$X_3 = (C - D)^2 - (A - B)^2 - 2B(A - B)^2$
$Y_3 = (C - D)(B(A - B)^2 - X_3) - D(A - B)^3$
$Z_3 = Z_1 Z_2 (A - B)$
return (X_3, Y_3, Z_3);

La méthode StdJAdd nécessite 12M + 4S + 7A.

L'alternative à l'algorithme standard Double-and-Add détaillée par l'algorithme 22 ci-dessous a un coût de calcul de 10M + 8S + 12A si le bit courant est égal à 1 et 4M + 6S + 8A sinon.

L'autre alternative, une variante de l'algorithme de Montgomery détaillée par l'algorithme 23 ci-dessous a un coût de calcul de 16M + 10S + 5A.

4.1.2 Optimisations en coordonnées mixtes

Les coordonnées mixtes, proposées par [109], peuvent être très efficaces quand le point générateur est défini en coordonnées affines. Le principe étant d'additionner deux points, le premier en coordonnées affines et le second en coordonnées jacobiennes et le résultat serait défini en coordonnées jacobiennes. La méthode d'addition en coordonnées mixtes MixedJAdd, décrite par l'algorithme 24 où $P(X_1 : Y_1 : Z_1) + Q(x_2, y_2) = (X_3 : Y_3 : Z_3)$ est en coordonnées jacobiennes tandis que P et Q sont respectivement en

Algorithme 22 Algorithme Double-and-Add en coordonnées jacobiennes standards

Require: k = $(k_{l-1},..., k_1, k_0)_2$, P $\in$ E(F_p), $R_0(X_0, Y_0, Z_0)$, $R_1(X_1, Y_1, Z_1)$
Ensure: Q = k·P
  $R_0 \leftarrow \emptyset$
  $R_1 \leftarrow P$
  for $i \leftarrow 1$ to $l-1$ **do**
    if $k_i = 1$ **then**
      $R_0 \leftarrow$ StdJAdd(R_0, R_1)
    end if
    $R_1 \leftarrow$ StdJDbl(R_1)
  end for
  return R_0 ;

Algorithme 23 Algorithme Montgomery ladder en coordonnées jacobiennes standards

Require: k = $(k_{l-1},..., k_1, k_0)_2$, P $\in$ E(F_p), $R_0(X_0, Y_0, Z_0)$, $R_1(X_1, Y_1, Z_1)$
Ensure: Q = k·P
  $R_0 \leftarrow P$
  $R_1 \leftarrow \emptyset$
  for $i \leftarrow l-1$ downto 0 **do**
    if $k_i = 1$ **then**
      $R_1 \leftarrow$ StdJAdd(R_0, R_1)
      $R_0 \leftarrow$ StdJDbl(R_0)
    end if
    if $k_i = 0$ **then**
      $R_0 \leftarrow$ StdJAdd(R_1, R_0)
      $R_1 \leftarrow$ StdJDbl(R_1)
    end if
  end for
  return (R_0) ;

coordonnées jacobiennes et affines définies sur une courbe elliptique $E(F_P)$, a une complexité de calculs de 8M + 3S + 7A.

Algorithme 24 MixedJAdd : Addition de points (P+Q) en coordonnées mixtes Jacobian-affine

Require: X_1, Y_1, Z_1, x_2, y_2
Ensure: P+Q
$A = x_2 Z_1^2$
$B = y_2 Z_1^3$
$C = X_1 - A$
$D = Y_1 - B$
$X_3 = D^2 - C^2 - 2AC^2$
$Y_3 = D(AC^2 - X_3) - BC^3$
$Z_3 = Z_1 C$
return (X_3, Y_3, Z_3);

D'après [110], le doublement de points est plus rapide en coordonnées jacobiennes tandis qu'une addition est plus rapide en coordonnées mixtes. Dans cette section, nous présentons également deux alternatives aux algorithmes standard Double-and-Add et de Montgomery, détaillées respectivement par les algorithmes 25 et 26. Nous avons implémenté ces deux alternatives en utilisant des opérations de doublement en coordonnées jacobiennes et des additions en coordonnées mixtes afin d'accélérer les calculs.

Algorithme 25 Algorithme Double-and-Add en coordonnées mixtes Jacobian-affine

Require: k = $(k_{n-1},..., k_1, k_0)_2$, P $\in$ E(F_p), $R_0(X_0, Y_0, Z_0)$, $R_1(X_1, Y_1, Z_1)$, $T(x, y)$
Ensure: Q = k·P
$R_0 \leftarrow \emptyset$
$R_1 \leftarrow P$
for $i \leftarrow$ 1 to n-1 **do**
  if $k_i = 1$ **then**
    $x = X_1/Z_1^2$
    $y = Y_1/Z_1^3$
    $R_0 \leftarrow$ MixedJAdd(R_0, T)
  end if
  $R_1 \leftarrow$ StdJDbl(R_1)
end for
return R_0;

L'algorithme 25 a un coût de calculs de 12M + 9S + 15A si le bit courant est égal à 1 et 4M + 6S + 8A sinon.

La complexité de calculs de l'algorithme 26 est de 15M + 10S + 15A.

Algorithme 26 Algorithme Montgomery ladder en coordonnées Mixtes Jacobian-affine

Require: k = $(k_{n-1},..., k_1, k_0)_2$, P $\in$ E(F_p), $R_0(X_0, Y_0, Z_0)$, $R_1(X_1, Y_1, Z_1)$, $T(x, y)$
Ensure: Q = k·P
  $R_0 \leftarrow P$
  $R_1 \leftarrow \emptyset$
  for $i \leftarrow n-1$ downto 0 **do**
    if $k_i = 1$ **then**
      $x = X_1/Z_1^2$
      $y = Y_1/Z_1^3$
      $R_1 \leftarrow$ MixedJAdd(R_0, T)
      $R_0 \leftarrow$ StdJDbl(Q)
    end if
    if $k_i = 0$ **then**
      $x = X_0/Z_0^2$
      $y = Y_0/Z_0^3$
      $R_0 \leftarrow$ MixedJAdd(R_1, T)
      $R_1 \leftarrow$ StdJDbl(R_1)
    end if
  end for
  return (R_0);

4.1.3 Optimisations en coordonnées co-Z

Nous avons vu à la section 3.2.2 du chapitre 3 de ce document que les nouvelles coordonnées co-Z, proposées par [82], sont très optimisées pour les opérations de doublement et d'addition de points. L'addition de points est proposée en deux variantes : l'addition avec mise à jour des coordonnées et l'addition-soustraction de points simplifiée, décrites à la section 3.2.2 ci-dessus. L'addition avec mise à jour de coordonnées, que nous notons coZAddU et décrite par l'algorithme 27, permet de calculer $(P + Q)$ et $P\prime$ la mise à jour de P. Le coût de l'algorithme coZAddU est de 4M + 2S + 7A.

Algorithme 27 coZAddU : Addition de points (P+Q) en coordonnées co-Z

Require: X_1, Y_1, X_2, Y_2, Z
Ensure: P+Q, P$\prime$
  $A = (X_2 - X_1)^2$
  $B = X_1 A$
  $C = X_2 A$
  $D = (Y_2 - Y_1)^2$
  $X_3 = D - (B + C)$
  $Y_3 = (Y_2 - Y_1)(B - X_3) - Y_1(C - B)$
  $X_4 = B$
  $Y_4 = Y_1(C - B)$
  return (X_3, Y_3, X_4, Y_4);

La méthode addition-soustraction de points, que nous notons coZAddC et décrite

par l'algorithme 28, permet le calcul d'une addition $(P+Q)$ et d'une soustraction $(P-Q)$ en même temps. Son coût en calculs est de 5M + 3S + 11A.

Algorithme 28 coZAddC : Addition (P+Q) et soustraction de points (P-Q) en coordonnées co-Z

Require: X_1, Y_1, X_2, Y_2, Z
Ensure: P+Q, P-Q
$A = (X_2 - X_1)^2$
$B = X_1 A$
$C = X_2 A$
$D = (Y_2 - Y_1)^2$
$X_3 = D - (B + C)$
$Y_3 = (Y_2 - Y_1)(B - X_3) - Y_1(C - B)$
$X_4 = (Y_1 + Y_2)^2 - (B + C)$
$Y_4 = (Y_1 - Y_2)(X_4 - B) - Y_1(C - B)$
return (X_3, Y_3, X_4, Y_4) ;

Le doublement de points en coordonnées co-Z, que nous notons coZDbl et décrite par l'algorithme 29, est obtenu par la multiplication du point $P(x_1, y_1, 1) \in E(F_P)$ en coordonnées affines par lui-même $2P = (X_3 : Y_3 : Z)$ en coordonnées co-Z. Le coût de calculs de l'algorithme 29 est 2M + 4S + 10A.

Algorithme 29 coZDbl : Doublement de points (2P) en coordonnées co-Z

Require: X_1, Y_1, Z
Ensure: 2P, $P\prime$
$A = 4x_1 y_1^2$
$B = 3x_1^2 + a$
$X_3 = B^2 - 2A$
$Y_3 = B(A - X_3) - 8y_1^4$
$Z = 2y_1$
return (X_3, Y_3, Z, X_4, Y_4) ;

Une autre méthode proposée dans les calculs des doublements et des additions de points en coordonnées co-Z est le Doubling-Addition, que nous notons par coZDA et décrite par l'algorithme 30. La méthode coZDA permet d'effectuer un doublement et une addition de points en même temps $(2P + Q) = (X_3 : Y_3 : Z)$ avec $P(X_1 : Y_1 : Z)$ et $Q(X_2 : Y_2 : Z) \in E(FP)$ en coordonnées co-Z. le coût de l'algorithme 30 est 9M + 5S + 18A.

Le Doubling-Addition est obtenu par une première addition avec mise à jour des coordonnées $(R = P + Q, P\prime)$ puis une deuxième addition-soustraction pour calculer $(R+P, R-P) = (P+Q+P, P+Q-P) = (2P+Q, Q) = (X_4 : Y_4 : Z)$ en coordonnées co-Z.

Dans cette section, nous présentons plusieurs variantes des algorithmes présentés au chapitre 3, tous implémentés en coordonnées co-Z. Nous donnons une troisième variante

Algorithme 30 coZDA : Doubling-addition (2P+Q) en coordonnées co-Z

Require: X_1, Y_1, X_2, Y_2, Z
Ensure: 2P+Q
$A = (X_2 - X_1)^2$
$B = X_1 A$
$C = X_2 A$
$D = (Y_2 - Y_1)^2$
$X_3 = D - (B + C)$
$Y_3 = (Y_2 - Y_1)(B - X_3) - Y_1(C - B)$
$E = (B - X_3)^2$
$F = X_3 E$
$G = BE$
$H = (Y_1(C - B) - Y_3)^2$
$X_4 = H - (F + G)$
$Y_4 = (Y_1(C - B) - Y_3)(F - X_4) - Y_3(G - F)$
return (X_4, Y_4);

de l'algorithme standard, après celles données en coordonnées jacobiennes standards et mixtes, décrite par l'algorithme 31 en coordonnées co-Z.

Algorithme 31 Algorithme Double-and-Add en coordonnées co-Z

Require: k $= (k_{n-1},..., k_1, k_0)_2$, P $\in$ E(F_p), $R_0(X_0, Y_0, Z_0)$, $R_1(X_1, Y_1, Z_1)$, $W(X_w, Y_w, Z_w)$
Ensure: Q = k·P
$R_0 \leftarrow \emptyset$
$R_1 \leftarrow P$
for $i \leftarrow$ 1 to n-1 **do**
  if $k_i = 1$ **then**
    $(R_0, R_1) \leftarrow$ coZAddU(R_0, R_1)
  end if
  $(R_1, W) \leftarrow$ coZDbl(R_1)
end for
return R_0;

Le coût de l'algorithme 31 est de 6M + 6S + 17A si le bit courant est égal à 1 et 2M + 4S + 10A sinon. L'algorithme Doubling-and-Add est très rapide mais très vulnérable aux attaques SCA présentées à la section 3.1.2. Quatre autres variantes de l'algorithme de Montgomery ont corrigé ce problème et sont données en coordonnées co-Z et détaillées par les algorithmes 32, 34, 35 et 36. La première variante de l'algorithme de Montgomery, détaillée par l'algorithme 32, a un coût de 6M + 6S + 7A. L'addition de points est effectuée avec la méthode coZAddU et le doublement avec coZDbl.

Une deuxième variante de l'algorithme de Montgomery, détaillée par l'algorithme 34, se basant sur la technique (X, Y)-only, détaillée dans [86], fait intervenir des additions de points uniquement dans les calculs itératifs. A l'initialisation, une seule

Algorithme 32 Algorithme Montgomery ladder en coordonnées co-Z

Require: k $= (k_{n-1},..., k_1, k_0)_2$, P $\in$ E(F_p), $R_0(X_0, Y_0, Z_0)$, $R_1(X_1, Y_1, Z_1)$, $W(X_w, Y_w, Z_w)$
Ensure: Q = k·P
  $R_0 \leftarrow P$
  $R_1 \leftarrow \emptyset$
  for $i \leftarrow n-1$ downto 0 **do**
    if $k_i = 1$ **then**
      $(R_0, R_1) \leftarrow$ coZAddU(R_0, R_1)
      $(R_0, W) \leftarrow$ coZDbl(R_0)
    end if
    if $k_i = 0$ **then**
      $(R_0, R_1) \leftarrow$ coZAddU(R_1, R_0)
      $(R_0, W) \leftarrow$ coZDbl(R_1)
    end if
  end for
  return (R_0);

opération de doublement de points est effectuée, et pour obtenir le résultat final, une opération d'inversion du point Z, décrite par l'algorithme 33, est exécutée.

Algorithme 33 FinalInvZ : Calculer l'inversement du point Z (1/Z)

Require: X_0, Y_0, X_1, Y_1, Z, X_P, Y_P, b
Ensure: 1/Z
  $A = X_1 - X_0$
  $B = Y_b A$
  $C = X_P B$
  $D = 1/C$
  $E = Y_P D$
  return $(X_b E)$;

L'inversement du point Z : $R_d \leftarrow (-1)^s R_d$ est calculée comme suit :

$$R_0 \leftarrow Y; \quad R_1 \leftarrow -Y; \quad Y \leftarrow R_s + R_s - R_{s\oplus 1} \tag{4.1}$$

Le coût d'une inversion de points FinalInvZ est de 1I + 4M + 1A.

L'algorithme 34 a un coût global de l(9M + 5S + 18A) + 1I + 18M + 10S + 29A) et un coût par bit de 9M + 5S + 18A.

La troisième variante de l'algorithme de Montgomery, détaillée par l'algorithme 35, utilisant la technique (X, Y)-only [86] fait intervenir des Doubling-Addition uniquement sur les calculs itératifs. A l'initialisation, un doublement de points et une addition-soustraction sont effectuées, et pour obtenir le résultat final, un inversement du point Z puis une addition avec mise à jour des coordonnées sont exécutés. L'algorithme 35 a un coût global de l(9M + 5S + 22A) + (1I + 18M + 10S + 29A) et un coût par bit

Algorithme 34 Algorithme Montgomery ladder en coordonnées (X,Y)-only co-Z addition

Require: k = $(k_{n-1},..., k_1, k_0)_2$, P $\in$ E(F_p)
Ensure: Q = k·P
  (R_0,R_1) $\leftarrow$ coZDbl(P)
  for $i \leftarrow n-2$ to 1 **do**
    b $\leftarrow k_i$
    (R_{1-b},R_b) $\leftarrow$ coZAddU(R_b,R_{1-b})
    (R_b,R_{1-b}) $\leftarrow$ coZAddC(R_{1-b},R_b)
  end for
  b $\leftarrow k_0$
  (R_{1-b},R_b) $\leftarrow$ coZAddU(R_b,R_{1-b})
  $\lambda \leftarrow$ FinalInvZ(R_0,R_1,P,b)
  (R_b,R_{1-b}) $\leftarrow$ coZAddC(R_{1-b},R_b)
  return ($X_0\lambda^2$,$Y_0\lambda^3$)

de 9M + 5S + 22A.

Algorithme 35 Algorithme Montgomery ladder en coordonnées (X,Y)-only co-Z Doubling-Addition

Require: k = $(k_{n-1},..., k_1, k_0)_2$, P $\in$ E(F_p)
Ensure: Q = k·P
  (R_1,R_0) $\leftarrow$ coZDbl(P)
  b $\leftarrow k_{n-2}$
  (R_{1-b},R_b) $\leftarrow$ coZAddC(R_b,R_{1-b})
  for $i \leftarrow n-2$ downto 1 **do**
    b $\leftarrow k_i$
    d $\leftarrow k_{i-1}$
    s $\leftarrow d \oplus b$
    (R_{1-d},R_d) $\leftarrow$ coZDA(R_{1-b},R_b)
    $R_d \leftarrow (-1)^s R_d$
  end for
  b $\leftarrow k_0$
  $\lambda \leftarrow$ FinalInvZ(R_0,R_1,P,b)
  (R_b,R_{1-b}) $\leftarrow$ coZAddU(R_{1-b},R_b)
  return ($X_0\lambda^2$,$Y_0\lambda^3$)

Une dernière variante de l'algorithme de Montgomery, détaillée par l'algorithme 36, fait intervenir des additions de points uniquement. Son coût est de 9M + 5S + 18A.

Nous présentons également dans cette section deux variantes de l'algorithme de Joye, présenté à la section 3.3.2 et décrit par l'algorithme 12, en coordonnées co-Z, détaillées par les algorithmes 37 et 38. La première variante de l'algorithme de Joye (Algorithme 37) utilise uniquement des additions de points en recalculant les coordonnées du point P en ($X_{3P} : Y_{3P} : Z_{3P}$). Son coût est de 9M + 5S + 18A.

La seconde variante de l'algorithme de Joye (Algorithme 38) fait intervenir la mé-

Algorithme 36 Algorithme Montgomery ladder en coordonnées co-Z Addition

Require: k = $(k_{n-1},..., k_1, k_0)_2$ with $k_{n-1} = 1$, P $\in$ E(F_p)
Ensure: Q = k·P
  $(R_1) \leftarrow (X_{2P} : Y_{2P} : Z_{2P})$
  $(R_0) \leftarrow (xZ_{2P^2} : yZ_{2P^3} : Z_{2P})$
  for $i \leftarrow n-2$ downto 1 **do**
    $(R_{1-k_i},R_{k_i}) \leftarrow$ coZAddC(R_{k_i},R_{1-k_i})
    $(R_{k_i},R_{1-k_i}) \leftarrow$ coZAddU(R_{1-k_i},R_{k_i})
  end for
  return (R_0)

Algorithme 37 Algorithme Joye en coordonnées co-Z Addition

Require: k = $(k_{n-1},..., k_1, k_0)_2$ with $k_0 = 1$, P $\in$ E(F_p)
Ensure: Q = k·P
  $(R_{1-k_i}) \leftarrow (X_{3P} : Y_{3P} : Z_{3P})$
  $(R_{k_i}) \leftarrow (xZ_{3P^2} : yZ_{3P^3} : Z_{3P})$
  for $i \leftarrow 2$ to $n-1$ **do**
    $(R_{k_i},R_{1-k_i}) \leftarrow$ coZAddU(R_{1-k_i},R_{k_i})
    $(R_{1-k_i},R_{k_i}) \leftarrow$ coZAddC(R_{k_i},R_{1-k_i})
  end for
  return (R_0)

thode coZDA uniquement aux calculs itératifs. A l'initialisation, un doublement coZDbl et une addition avec mise à jour des coordonnées coZAddU sont effectuées. Le coût global de l'algorithme 38 est de l(9M + 5S + 18A) + (6M + 6S + 17A) et un coût par bit de 9M + 5S + 18A.

Algorithme 38 Algorithme de Joye en coordonnées co-Z Doubling-Addition

Require: k = $(k_{n-1},..., k_1, k_0)_2$ with $k_0 = 1$, P $\in$ E(F_p)
Ensure: Q = k·P
  $b \leftarrow k_1$
  $(R_b,R_{1-b}) \leftarrow$ coZDbl(R_b)
  $(R_{1-b},R_b) \leftarrow$ coZAddU(R_b, R_{1-b})
  for $i \leftarrow 2$ to $n-1$ **do**
    $b \leftarrow k_i$
    $(R_{1-b}, R_b) \leftarrow$ coZDA$(R_{1-b} + R_b)$
  end for
  return (R_0);

La méthode NAF [100] décrite à la section 3.3.1 présente des résultats intéressants en coordonnées affines mais nécessite tout de même des optimisations. Dans cette section, nous présentons une variante de la méthode NAF, détaillée par l'algorithme 39, en coordonnées co-Z. Le coût par bit de l'algorithme 39 est de 16M + 10S + 15A si le bit courant est différent de 0 et 4M + 6S + 8A sinon.

Algorithme 39 Algorithme NAF en coordonnées co-Z

Require: $k \in N$, P, R $\in$ E(F_p), $R_0(X_0, Y_0, Z_0)$, $R_1(X_1, Y_1, Z_1)$, $W(X_w, Y_w, Z_w)$
Ensure: Q = k·P
  $R_0 \leftarrow \emptyset$
  $R_1 \leftarrow P$
  while $k \geq 1$ **do**
    if $k \pmod 2 = 1$ **then**
      $u \leftarrow 2 - (k \pmod 4)$
      $k \leftarrow k - u$
      if $u = 1$ **then**
        $(R_0, W) \leftarrow$ coZAddC(R_0, R_1)
      else
        $(W, R_0) \leftarrow$ coZAddC(R_0, R_1)
      end if
    end if
    $k \leftarrow k/2$
    $R_1 \leftarrow coZDbl(R_1)$
  end while
  return R_0

4.2 Évaluation de la complexité des calculs et comparaisons

Le tableau 4.1 présente un récapitulatif des différentes caractéristiques des algorithmes présentés dans ce document pour l'optimisation d'une multiplication scalaire. Nous avons proposé quatre méthodes de calcul : Double-and-Add, Montgomery, Joye et NAF (Non-Adjacent Form). Nous avons proposé plusieurs variantes pour chaque méthode dans différents systèmes de coordonnées : A (Affines), J (Jacobiennes standards), M (Mixtes J-A), co-Z Addition, co-Z Doubling-Addition et (X, Y)-only co-Z. Nous avons évalué la complexité de chaque algorithme en termes du nombre d'opérations d'Addition de points (Add), Doublement de points (Dbl) et Inversion de points (Inv), en fonction de la longueur l du scalaire k. Ensuite, nous avons calculé le coût par bit et le coût global de chaque algorithme en fonction du nombre des opérations arithmétiques : M (Multiplication modulaire), S (Elévation au carré), A (Addition modulaire) et I (Inversion modulaire). Enfin, nous avons évalué la résistance de chacun des algorithmes proposés à l'attaque SCA (Side-Channel Analysis).

Le coût d'une opération arithmétique d'addition modulaire est négligeable par rapport à la multiplication et à l'élévation au carré. Nous avons donc représenté sur la Figure 4.1 que la multiplication modulaire et l'élévation au carré pour avoir une vue plus globale sur la complexité des algorithmes présenté jusqu'à maintenant dans ce document.

TABLEAU 4.1 – Tableau global de la complexité et du coût de chaque algorithme présenté dans ce document

Méthode	Variante	Complexité	Coût/bit		Coût global	Résistance à SCA
			$k_i = 0$	$k_i \neq 0$		
Double-and-Add	A (Algo 2)	$\frac{1}{2}l$ Add + l Dbl	I+2M+2S	2I+4M+3S	$\frac{1}{2}l$(3I+6M+5S)	Non
	J (Algo 22)	$\frac{1}{2}l$ Add + l Dbl	4M+6S+8A	16M+10S+15A	l(10M+8S+12A)	Non
	M (Algo 25)	$\frac{1}{2}l$ Add + l Dbl	4M+6S+8A	12M+9S+15A	$\frac{1}{2}l$(19M+16S+30A)	Non
	co-Z (Algo 31)	$\frac{1}{2}l$ Add + l Dbl	2M+4S+10A	6M+6S+17A	l(4M+5S+13A)	Non
Montgomery	J (Algo 11)	l Add + l Dbl	16M+10S+15A		l(16M+10S+15A)	Oui
	J (Algo 23)	l Add + l Dbl	16M+10S+15A		l(16M+10S+15A)	Oui
	M (Algo 26)	l Add + l Dbl	15M+10S+15A		l(15M+10S+15A)	Oui
	co-Z (Algo 32)	l Add + l Dbl	6M+6S+17A		l(6M+6S+17A)	Oui
	(X, Y)-only (Algo 34)	$2l$ Add + 1 Dbl + 1 Inv	9M+5S+18A		l(9M+5S+18A)+(I+18M+10S+29A)	Oui
	(X, Y)-only (Algo 35)	(l-2) DA + 1 Dbl + 2 Add + 1 Inv	9M+5S+22A		l(9M+5S+22A)+(I+18M+10S+29A)	Oui
	co-Z (Algo 36)	$2(l-2)$ Add	9M+5S+18A		l(9M+5S+18A)	Oui
Joye	J (Algo 12)	l DA	11M+7S+27A		l(11M+7S+27A)	Oui
	co-Z (Algo 37)	$2(l-2)$ Add	9M+5S+18A		l(9M+5S+18A)	Oui
	co-Z (Algo 38)	$(l-2)$ DA + 1 Add + 1 Dbl	9M+5S+18A		l(9M+5S+18A)+(6M+6S+17A)	Oui
NAF	A (Algo 7)	$\frac{1}{3}l$ Add + l Dbl	I+2M+2S	2I+4M+13S	$\frac{1}{3}l$(4I+8M+7S)	Oui
	co-Z (Algo 39)	$\frac{1}{3}l$ Add + l Dbl	4M+6S+8A	16M+10S+15A	$\frac{1}{3}l$(11M+15S+41A)	Oui

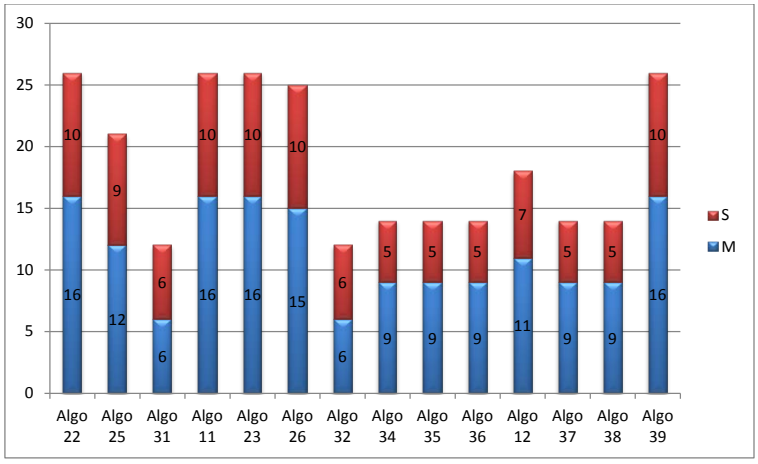

FIGURE 4.1 – Représentation globale du coût par bit des algorithmes proposés

En termes de coût, l'algorithme 31 présente de meilleures performances. Cependant ses vulnérabilités aux attaques de type SCA (Side-Channel Analysis) rendent son utilisation inadéquate pour les réseaux de capteurs sans fil. Les algorithmes 32, 34, 35, 36, 37 et 38 présentent d'excellentes performances en coordonnées co-Z. Par contre, le coût par bit des variantes 11, 23 et 26 de la méthode Montgomery en coordonnées jacobiennes, et celui de la méthode NAF en coordonnées co-Z (Algorithme 39 est élevé et sont donc inappropriés pour les réseaux à ressources restreintes. En résumé, les coordonnées co-Z fournissent de meilleures performances pour la multiplication scalaire. Ces résultats seront confirmés par l'évaluation de leurs performances effectuée dans la section 4.3 suivante.

4.3 Évaluation des performances

Nous avons implémenté les algorithmes proposés dans ce chapitre et le chapitre précédent sur une carte Arduino Embedded System. Nous donnerons dans cette section les coût des opérations arithmétiques, le coût des opérations d'addition et de doublement de points et le coût de tous les algorithmes proposés. Les paramètres elliptiques utilisés concernant le choix des courbes, le choix de la bibliothèque logicielle et l'architecture matérielle du système embarqué utilisés sont donnés en annexe (voir l'Annexe 6.6).

4.3.1 Coût des opérations arithmétiques

Une multiplication scalaire est une suite d'opérations d'additions et de doublements de points. Une addition et un doublement de points est une suite d'opérations arithmétiques modulaires sur le corps fini premier F_P définies à la section 2.1.9 ci-dessus. Le coût de ces opérations est donné par le tableau 4.2.

On remarque que sur un corps fini premier Fp, les temps d'exécution des opérations

TABLEAU 4.2 – Le temps d'exécution des opérations arithmétiques Multiplication modulaire, élévation au carré modulaire et addition modulaire

Paramètre		Mul	Sqrt	Add
P-192	Time	0.584	0.584	0.026
	Cycles	9344	9344	416
P-224	Time	0.788	0.788	0.027
	Cycles	12608	12608	432
P-256	Time	1.132	1.132	0.029
	Cycles	18112	18112	464

de multiplication modulaire et d'élévation au carré modulaire sont identiques. Le temps d'exécution d'une addition modulaire est négligeable par rapport au temps d'exécution d'une multiplication modulaire.

4.3.2 Coût des opérations de doublement et d'addition de points dans différents systèmes de coordonnées

Le tableau 4.3 montre le temps d'exécution des différents algorithmes de calcul d'un doublement et d'addition de points en coordonnées jacobiennes, mixtes et co-Z. D'après les résultats obtenus, on remarque qu'une addition avec mise à jour de coordonnées co-Z offre un meilleur temps de calcul avec un gain d'environ 50% par rapport à une addition de points en coordonnées mixtes, et environ 65% par rapport à une addition de points en coordonnées jacobiennes standards. L'addition-soustraction en coordonnées co-Z offre également un gain intéressant mais moins important que la précédente mais demeure plus efficace que dans les deux autres systèmes de coordonnées mixtes et jacobiennes standards.

TABLEAU 4.3 – Temps d'exécution (en ms) de doublement, d'addition et de Doubling-Addition de points

Opération	Algorithme	Coordonnées	P-192	P-224	P-256
2P	StdJDbl	Standard	6.064	6.599	11.732
	coZDBL	co-Z	9.708	13.002	18.592
P+Q	StdJAdd	Standard	10.056	12.759	19.332
	MixedJAdd	Mixtes	7.177	9.634	13.743
	coZAddU	co-Z	3.628	4.911	6.916
	coZAddC	co-Z	4.958	6.601	9.375
2P+Q	coZDA	co-Z	10.916	14.684	20.808

4.3.3 Coût des algorithmes proposés de multiplication scalaire

Dans cette section nous allons implémenter les algorithmes proposés dans ce chapitre et le chapitre précédent pour le calcul d'une multiplication scalaire, à savoir le

Double-and-Add, Montgomery, Joye et NAF et leurs variantes dans les différents systèmes de coordonnées. Les résultats d'exécution de ces algorithmes sur la plateforme Arduino Uno R3 sont donnés sur le tableaux 4.4.

TABLEAU 4.4 – Temps d'exécution par bit (en ms)

Algorithme	192-bit		224-bit		256-bit	
	$K_i = 0$	$K_i \neq 0$	$K_i = 0$	$K_i \neq 0$	$K_i = 0$	$K_i \neq 0$
22 et 39	7.92	15.574	8.096	20.893	11.552	29.867
12	11.214		14.913		21.159	
11 et 23	15.574		20.893		29.867	
25	7.92	12.654	8.096	16.953	11.552	24.211
26	14.99		20.105		28.739	
31	3.794	7.45	4.998	9.915	7.082	14.077
32	7.45		9.915		14.077	
34, 36, 37 et 38	8.698		11.518		16.37	
35	8.814		11.626		16.486	

Nous remarquons que l'algorithme de Montgomery en coordonnées co-Z (Algorithme 32) donne de meilleurs résultats comparativement aux autres algorithmes et permet de réduire d'environ 15% les calculs sur une multiplication scalaire par rapport au deuxième meilleur algorithme et 52% par rapport au plus pire algorithme.

4.4 Synthèse des comparaisons

Les algorithmes optimisés obtenus sont le résultat de combinaisons de quelques algorithmes d'accélération des calculs avec les différents systèmes de coordonnées [108]. Ainsi, plusieurs améliorations sont apportées à des algorithmes combinés aux techniques mathématiques présentées ci-dessus, réduisant davantage la complexité des calculs d'une multiplication scalaire.

En coordonnées Jacobéennes, les deux algorithmes Double-and-add et Montgomery utilisent les deux méthodes StdJDbl pour le calcul d'un doublement de points et la méthode StdJAdd pour le calcul d'une addition de points en coordonnées jacobiennes standards. En coordonnées mixtes, nous avons implémenté ces deux alternatives en utilisant des opérations de doublement en coordonnées jacobiennes et des additions en coordonnées mixtes afin d'accélérer les calculs. En coordonnées co-Z, nous avons présenté plusieurs variantes des algorithmes ci-dessus, tous implémentés en coordonnées co-Z. Nous donnons une troisième variante de l'algorithme standard, après celles données en coordonnées jacobiennes standards et mixtes.

En termes de coût, l'algorithme Double-and-add en coordonnées co-Z présente de meilleures performances. Cependant ses vulnérabilités aux attaques de type SCA (Side-Channel Analysis) rendent son utilisation inadéquate pour les réseaux de capteurs

sans fil. Les algorithmes Montgomery en coordonnées co-Z Addition et co-Z Doubling-Addition, et les algorithmes de Joye en coordonnées co-Z Addition et co-Z Doubling-Addition présentent d'excellentes performances en coordonnées co-Z.

4.5 Conclusion

Dans ce chapitre, nous avons implémenté sur une carte Arduino Embedded System les différents algorithmes largement utilisés dans le calcul d'une multiplication scalaire, sur une courbe elliptique définie sur un corps fini premier, dans différents systèmes de coordonnées. Nous avons ensuite comparé leurs performances et les performances des opérations arithmétiques et de doublement et d'addition de points, en utilisant des longueurs différentes du scalaire.

Ainsi, nous avons implémenté différentes techniques d'accélération de calculs étudiées dans la section 3.2 du chapitre 3 précédent sur quatre méthodes très utilisées : Double-and-Add, Montgomery, Joye et NAF. Les résultats montrent que l'algorithme Double-and-Add en coordonnées co-Z donne de meilleurs résultats mais demeure vulnérable aux attaques de type SCA et par conséquent inapproprié pour les réseaux de capteurs sans fil. Dans les autres résultats, l'algorithme de Montgomery en coordonnées co-Z offre un meilleur temps de calcul avec un gain de 15% sur ses variantes 34 et 36, et sur les algorithmes 37 et 38 de Joye.

Toutes les solutions proposées jusqu'ici pour le calcul d'une multiplication scalaire sur un corps fini premier sont centralisées, c'est-à-dire elles s'exécutent sur un seul dispositif embarqué. Dans le chapitre suivant, nous allons proposer une nouvelle approche de calculs distribués d'une multiplication scalaire basée sur le traitement coopératifs des nœuds d'un réseau de capteurs.

Chapitre 5
Approche distribuée pour le calcul d'une multiplication scalaire sur courbe elliptique

L'un des points forts des réseaux de capteurs sans fil est leur capacité à réaliser des traitements coopératifs entre capteurs comme pour le routage ou l'agrégation de données. Comme nous l'avions discuté à la section 3.4, il est possible de paralléliser des traitements sur plusieurs threads d'un même capteur. Cette technique est efficace pour la réduction des temps de calcul mais pas pour économiser de l'énergie. L'application des techniques de virtualisation dans des réseaux à ressources restreintes, notamment les RcSF, est quasiment impossible vu les limites des processeurs embarqués sur les nœuds capteurs.

Afin d'optimiser la consommation et ainsi prolonger la durée de vie des capteurs, nous proposons dans ce travail une solution efficace qui permet des traitements parallèles et coopératifs pour les calculs complexes, réduisant ainsi considérablement le temps de calcul et la consommation d'énergie. Ainsi, les calculs sont répartis sur plusieurs nœuds. Les calculs couteux seront confiés à des (super) capteurs qui n'ont pas de contraintes d'énergie comme par exemple des capteurs équipés de module de gestion des récupérateurs d'énergie [111].

5.1 Algorithmes distribués pour le calcul d'une multiplication scalaire

Dans les sections précédentes, nous avons présenté plusieurs optimisations de l'algorithme standard Double-and-Add pour le calcul d'une multiplication scalaire sur un seul dispositif. Tous les calculs sont donc effectués par un seul processeur ou bien répartis sur plusieurs cœurs d'un même processeur. La consommation de ressources dans ce cas, notamment les ressources énergétiques et de stockage, demeure importante. Dans cette section, nous allons proposer trois variantes distribués des trois algorithmes présentés à la section 3.5.2 ci-dessus connus sous le nom de Right-to-Left [112], où le scalaire k est balayé du bit de poids faible au bit du poids fort et où les deux opérations de doublement et d'addition de points peuvent s'exécuter indépendamment.

Les trois algorithmes proposées, Right-to-Left Distributed binary algorithm (RL-

Dba), Right-to-Left Distributed Non-Adjacent Form (RL-DNAF) et Right-to-Left Distributed Signed window algorithm (RL-Dswa), consistent à distribuer les calculs d'une multiplication scalaire sur deux nœuds ; un nœud calculateur nommé SensorCalc et un nœud ordinaire. Le nœud calculateur effectue des doublements de points tandis qu'un nœud ordinaire opère uniquement des additions de points. Le principe étant que les doublements de points sont généralement des traitements redondants effectués par différents nœuds que nous pourrons centraliser au niveau d'un même nœud appelé nœud calculateur (sensorCalc). Les trois algorithmes RL-Dba, RL-DNAF et RL-Dswa seront ensuite appliqués à notre approche distribuée pour les calculs elliptiques dans un cluster formé par plusieurs nœuds. Le coût des communications radio entre le nœud ordinaire et le nœud calculateur ainsi que le coût des calculs seront détaillés à la section 5.5.

5.1.1 Right-to-Left Distributed binary algorithm (RL-Dba)

L'algorithme RL-Dba est une variante distribuée de l'algorithme Double-and-Add décrit à la section 3.1. La complexité de RL-Dba est de $\frac{l}{2}$ additions de points pour le nœud ordinaire et l doublements de points pour le nœud calculateur. Le nœud calculateur stocke tous les points obtenus dans un tableau $Q[l]$ qu'il envoie au nœud ordinaire. Le nœud ordinaire reçoit le tableau de points $Q[l]$ qu'il utilise pour le calcul des additions de points. Le principe de l'algorithme RL-Dba est donné par l'algorithme 40.

Algorithme 40 Algorithme RL-Dba en coordonnées jacobiennes standards

Nœud calculateur	**Nœud ordinaire**
Require: P $\in$ E(F_p)	**Require:** k = $(k_{l-1},..., k_1, k_0)_2$, $Q[l]$
Ensure: 2^iP where $i = [0, l-1]$	**Ensure:** Q = k·P
$Q[0] \leftarrow P$	$Q \leftarrow \emptyset$
for $i \leftarrow 1$ to $l-1$ **do**	**for** $i \leftarrow 0$ to $l-1$ **do**
$Q[i] \leftarrow 2 \times Q[i-1]$	**if** $k_i = 1$ **then**
end for	$Q \leftarrow Q + (k_i \times Q[i])$
	end if
Envoyer $Q[l]$ au nœud ordinaire	**end for**
	return Q

5.1.2 Right-to-Left Distributed Non-Adjacent Form (RL-DNAF)

L'algorithme RL-DNAF est une variante distribuée de la méthode NAF décrite à la section 3.5.2. La complexité de RL-DNAF est de $\frac{l}{3}$ additions de points pour le nœud ordinaire et l doublements de points pour le nœud calculateur. Le nœud calculateur stocke tous les points obtenus dans un tableau $Q[l]$ qu'il envoie au nœud ordinaire. Le

nœud ordinaire reçoit le tableau de points $Q[l]$ qu'il utilise pour le calcul des additions de points. Le principe est donné par l'algorithme 41.

Algorithme 41 Algorithme RL-DNAF en coordonnées jacobiennes standards

Nœud calculateur

Require: P $\in$ E(F_p), $k \in N$
Ensure: $2^i P$ où $i = [0, l-1]$
  $Q_0 \leftarrow P$
  $i \leftarrow 1$
  while $k \geq 1$ **do**
    $k \leftarrow k/2$
    $Q[i] \leftarrow 2 \times Q[i-1]$
    $i \leftarrow i+1$
  end while

  Envoyer $Q[l]$ au nœud ordinaire)

Nœud ordinaire

Require: $k \in N$, $Q[l]$
Ensure: Q = k·P
  $Q \leftarrow \emptyset$
  $i \leftarrow 0$
  while $k \geq 1$ **do**
    if $k \pmod 2 = 1$ **then**
      $u \leftarrow 2 - (k \pmod 4)$
      $k \leftarrow k - u$
      if $u \neq 0$ **then**
        $Q \leftarrow Q + u \cdot Q[i]$
      end if
    end if
    $i \leftarrow i+1$
  end while
  return Q

5.1.3 Right-to-Left Distributed Signed window algorithm (RL-DSwa)

L'algorithme 42 décrit l'algorithme RL-Dswa, une variante distribuée de la méthode RLSwa décrite à la section 3.5.2. La complexité de RL-Dswa est de $\frac{l-1}{w} + \frac{l}{w+1} + 2^{w-2}$ additions de points pour le nœud ordinaire et $(l-1+d)$ doublements de points pour le nœud calculateur. Le nœud calculateur stocke tous les points obtenus dans un tableau $Q[l-1+d]$ qu'il envoie au nœud ordinaire. Le nœud ordinaire reçoit le tableau de points $Q[l-1+d]$ qu'il utilise pour le calcul des additions de points. Le principe est donné par l'algorithme 42.

5.2 Présentation de notre approche distribuée

Plusieurs optimisations ont été proposées pour alléger les calculs coûteux de la cryptographie des courbes elliptiques en réduisant la complexité des calculs de multiplication scalaire par l'introduction de nouveaux systèmes de coordonnées comme les coordonnées jacobiennes co-Z, et d'autres optimisations par la minimisation du nombre de doublements et d'additions, entre autres. En plus de ces travaux, beaucoup de chercheurs ont proposé d'autres méthodes de calcul se basant principalement sur les traitements coopératifs entre les nœuds d'un même cluster ou d'un même petit réseau. La

Algorithme 42 Algorithme RL-DSwa en coordonnées jacobiennes standards

Nœud calculateur

Require: P $\in$ E(F_p)
Ensure: 2^iP where $i = [0, l-1]$
  $Q_0 \leftarrow P$
  for $i \leftarrow 1$ to $l-1$ **do**
    $Q[i] \leftarrow 2 \times Q[i-1]$
  end for

  Envoyer $Q[l]$ au nœud ordinaire

Nœud ordinaire

Require: k = $(k_{l-1},..., k_1, k_0)_{2^w}$, $Q[l]$
Ensure: Q = k · P
  for all $d \in B^+$ **do**
    $R_d \leftarrow \emptyset$
  end for
  for $i \leftarrow 0$ to $l-2$ **do**
    $d \leftarrow |d_i|$
    if $d_i \neq 0$ **then**
      $R_d \leftarrow R_d + k_i \cdot Q[i]$
    end if
  end for
  return $(Q = \sum_d d.R_d)$;

distribution de calculs repose sur la technique Right-to-Left présentée précédemment à la section 3.5.2 de ce document.

La technique Right-to-Left permet de distribuer et paralléliser les calculs sur un seul nœud. Cette technique est efficace dans la réduction des temps de calcul mais non pas pour économiser de l'énergie car tous les traitements sont effectués par le même nœud. Dans la section 5.1 ci-dessus, nous avons proposé trois variantes des algorithmes Right-to-Left Double-and-Add, NAF et signed window algorithm, allouant la distribution de calculs sur plusieurs nœuds d'un même cluster. Dans cette section, nous appliquons les trois variantes distribuées sur un cluster de deux à dix nœuds pour l'échange de clés et de secrets.

5.2.1 Principe de fonctionnement

L'objectif de notre approche distribuée est de répartir les calculs sur plusieurs nœuds d'un même cluster. Chaque cluster possède au minimum un nœud calculateur. Le nœud calculateur peut être un nœud ordinaire du réseau n'ayant pas de contraintes d'énergie ou bien un super nœud sans contraintes ni limitations, équipés par exemple de module de gestion des récupérateurs d'énergie et de la recharge de batterie [111]. Les calculs coûteux seront confiés aux nœuds calculateurs. Dans un réseau homogène, la sélection de ces nœuds calculateurs se fait par rapport à leurs réserves en énergie. Un algorithme de sélection et de choix de ces nœuds calculateurs dans un réseau homogène est donné par l'algorithme 43.

Notre solution consiste à distribuer les calculs sur les doublements et les additions. Le nœud calculateur effectue les doublements et les enregistre dans un tableau de taille l (l étant la taille du scalaire k). Le nœud ordinaire utilise les résultats du tableau de points précalculés pour effectuer des additions. Nous avons testé notre solution sur

Algorithme 43 Algorithme de sélection d'un nœud calculateur dans un réseau homogène

Require: Cluster de n nœuds
Ensure: TOS(SensorCalc)
1: Formation d'un cluster
2: Décider de la valeur du seuil : seuil = valeur
3: Désignation d'un capteur calculateur
4: sensorCalc = TOS //Adresse du capteur calculateur
5: **while** sensorCalc → Energie > seuil **do**
6: Lire niveau énergie de chaque capteur à chaque échange avec sensorCalc
7: **if** sensorCalc → Energie ≤ seuil **then**
8: **if** ∀(Nœuds → Energie) ≤ seuil **then**
9: Aller à 1
10: **else**
11: Désigner un capteur calculateur
12: sensorCalc = TOS
13: Informer tous les capteurs du cluster
14: **end if**
15: **end if**
16: **end while**
17: **return** TOS ;

trois algorithmes de la littérature : un algorithme standard (Right-to-Left Double-and-Add) et deux algorithmes optimisés (Right-to-Left Non-Adjacent Form et Right-to-Left signed window scalar multiplication). Right-to-Left signed window scalar multiplication est considéré comme l'un des algorithmes les plus économes en énergie [86]. Nous avons choisi d'évaluer ces algorithmes car ils permettent la parallélisation des calculs.

La communication entre les nœuds s'effectue via des liaisons sans fil par l'échange de messages publics ne nécessitant aucune opération de chiffrement/déchiffrement. Le nombre de messages échangés durant toutes les phases du protocole est très limité ce qui ne constitue aucunement un inconvénient aux calculs distribués. L'échange de données se fait d'une manière séquentielle où la tache j de n'importe quel nœud ordinaire est tributaire de la tache i du nœud calculateur. Si un nœud calculateur tombe en panne, le nœud master (éventuellement un Cluster Head CH) intervient pour le remplacer par un autre nœud suivant l'algorithme 43 donné ci-dessus.

Les résultats obtenus sont très intéressants notamment pour les réseaux denses comme les réseaux de capteurs sans fil. Pour un cluster de 10 nœuds, le nombre maximal de clés à générer est de 45 clés nécessitant jusqu'à 18 opérations de multiplication scalaire par nœud et 180 opérations au total. Le nombre maximal de clés est calculé comme suit : soit m le nombre de nœuds du cluster, $S = \sum_{i=3}^{m} S+i-1$ avec S initialisé à 1.

5.2.2 Chiffrement/Déchiffrement distribué

Le processus de synchronisation et d'exécution de notre approche distribuée est donné sur la Figure 5.1. A noter que D[X] est le tableau de points obtenus après les opérations de doublement du point X et Add(D[X]) représente le résultat d'additions des points du tableau D[X].

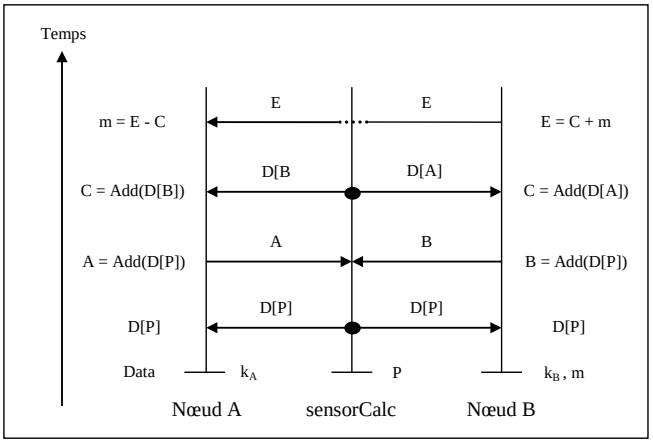

FIGURE 5.1 – Processus de synchronisation et d'exécution de notre solution pour le partage d'un secret m

Le schéma du processus de distribution des tâches pour le partage du secret m est donné sur la Figure 5.2 ci-dessous :

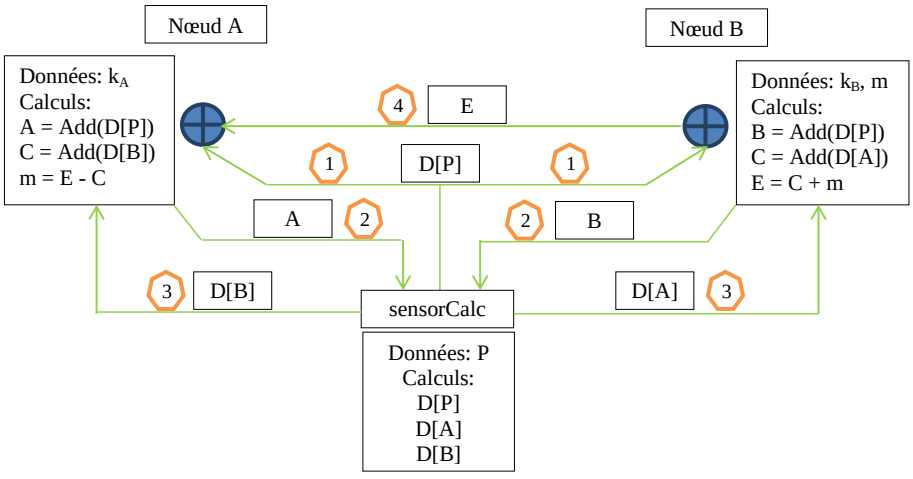

FIGURE 5.2 – Processus de distribution des tâches pour le partage d'un secret m

Ce schéma illustre l'exécution parallèle et synchronisée d'un partage de secret m entre deux nœuds capteurs.

5.2.3 Échange de clés distribué

Le processus d'échange de clés ne diffère pas trop de celui de partage de secret. Un secret C est partagé par les deux nœuds A et B est calculé comme illustré sur la Figure 5.3.

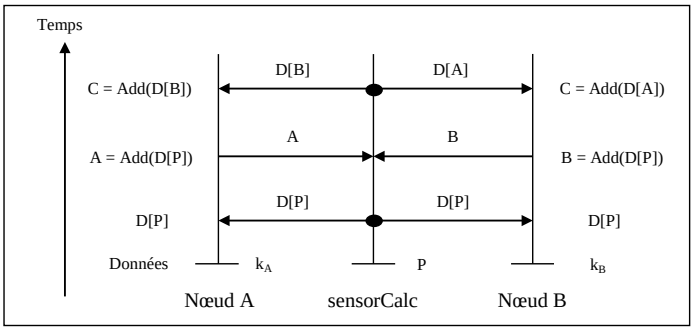

FIGURE 5.3 – Processus de synchronisation et d'exécution de notre solution pour le partage d'un secret m

Le schéma du processus de distribution des tâches pour le partage du point C est donné sur la Figure 5.4 ci-dessous :

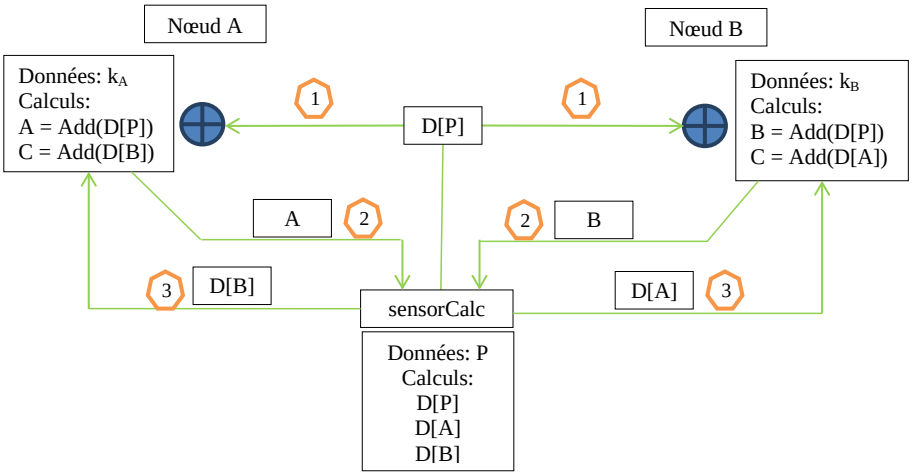

FIGURE 5.4 – Processus de distribution des tâches pour le partage d'une clé C

— Cas général : Exemple de n = 4

5.3 Comparaison de la complexité des calculs

Le tableau 5.1 montre la complexité des algorithmes de calcul d'une multiplication scalaire utilisés dans ce chapitre. La complexité est exprimée en nombre de doublements (D), d'additions (A) et de Doubling-Addition (DA) en fonction de la longueur l du scalaire k.

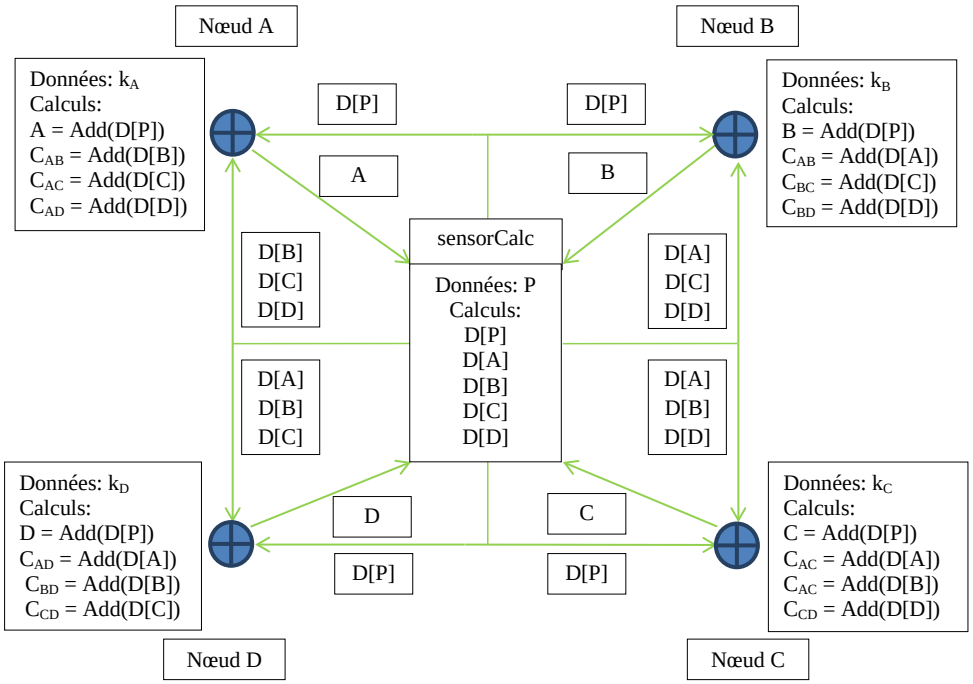

FIGURE 5.5 – Processus de distribution des tâches pour le partage d'une clé C pour $n = 4$

TABLEAU 5.1 – Complexité des algorithmes de multiplication scalaire utilisés dans ce chapitre

Algorithme	**Complexité**
L-R Double-and-Add	$lD + \frac{l}{2}A$
L-R NAF	$lD + \frac{l}{3}A$
L-R 5-NAF	$(l-1)D + \frac{l}{6} + 8A$
L-R signed window algorithm	$\frac{l}{w} + (l-1)D + \frac{l-1}{w} + \frac{l}{w+1} + 2^{w-2}A$
RL-Dba	$\frac{l}{2}A$
RL-DNAF	$\frac{l}{3}A$
RL-Dswa	$\frac{l-1}{w} + \frac{l}{w+1} + 2^{w-2}A$

5.4 Environnement d'implémentation

Dans ce qui suit, nous donnons les résultats de nos expérimentations. Les algorithmes sont implémentés sur une carte Arduino Uno équipée d'un microcontrôleur ATMega328P très limité en ressources et semblable à la plupart des microcontrôleurs des capteurs utilisés sur le marché. Une comparaison de performances est donnée par le tableau 5.2.

TABLEAU 5.2 – Comparaison des performances de différents systèmes embarqués

Caractéristiques		**Telos B**	**MicaZ**	**Arduino Uno**	
Microcontrôleur		MSP430	ATmega 128	ATmega 328P	
Fréquence		1 Mz	8 Mz	8/16 Mhz	
Voltage		1.8 - 3.6 V	2.7 – 3.3 V	1.8 – 5.5 V	
(S)RAM		10 Ko	4 Ko	2 Ko	
Mémoire Flash		48 Ko	512 Ko	32 Ko	
Energie	Radio + CPU	75 mW	63 mW	8 Mhz	58.4 mW
				16 Mhz	81.6 mW
	Mode veille	140 uW	30 uW	16 Mhz	14 uW

Nous commençons par présenter les paramètres elliptiques utilisés dans nos expérimentations, le matériel utilisé dans les communications radio et ainsi que son modèle de consommation d'énergie, puis les paramètres généraux des expérimentations.

5.4.1 Mise en œuvre logicielle et matérielle

Pour le besoin de nos expérimentations, nous avons utilisé les paramètres requis et recommandés par SEC2-V2 [113] pour un meilleur niveau de sécurité et définis en Annexe A. Nous avons choisi 192-bit Elliptic Curve domain parameters secp192r1, 224-bit Elliptic Curve domain parameters secp224r1 et 256-bit Elliptic Curve domain parameters secp256r1 pour comparer nos différents résultats fournissant respectivement un niveau de sécurité de 96 bits, 112 bits et 128 bits.

Le matériel utilisé dans ce travail est composé de deux cartes Arduino Uno R3 équipées de microcontrôleurs ATmega328P, alimentées par une pile alcaline 9V non rechargeable pour le nœud capteur et une LIPO rechargeable 7.4V pour le nœud calculateur, et de deux cartes Xbee Serie 2 (Zigbee) identiques alimentées par une tension de 3.3V. Le microcontrôleur utilisé pour un nœud capteur est un ATmega328P, une plateforme à ressources à basse consommation d'énergie [114]. Il est cadencé à une fréquence de 16Mhz sous une tension de 5V, équipé d'une RAM de 2 Ko et d'une mémoire flash 32 Ko avec une puissance théorique de 81.6 mW. Le microcontrôleur est programmé sous le langage C [115].

5.4.2 Modèle de communication radio

Le module utilisé pour les communications est une carte Xbee Serie 2 [116] fonctionnant avec le protocole Zigbee connu sous le nom de réseau personnel sans fil à faible débit (LR-WPAN). Le baud est fixé à 20 Kbps à 868 MHz. L'énergie consommée pour envoyer un point $P(X : Y : Z)$ en coordonnées jacobiennes de taille 3 octets est de 1.064 mJ. L'envoi d'un tableau de n points nécessite $4n$ octets avec $3n$ octets comme taille des points et n octets comme données du tableau. Pour $n = 192$ bits, la taille du tableau est de 768 octets nécessitant environ 7 paquets de données. Le coût total pour recevoir les 7 paquets de données est 50.176 mJ.

5.4.3 Paramètres des expérimentations

La carte Arduino du nœud capteur est branchée à une pile alcaline non rechargeable de 9V alimentant à la fois le microcontrôleur et le module Xbee Serie 2. La tension sur les PIN est de U = 5V pour le microcontrôleur et U = 3.3V pour le module Xbee. L'intensité du ATmega328P est de I = 15.15 mA engendrant une puissance P = 75.7 mW. L'intensité du module radio est de I = 42.424 mA fournissant une puissance P = 139 mW. Le temps d'envoi d'un paquet de 128 Octets est de 51.2 ms et l'énergie consommée, calculée comme suit : $E = \int^t P(t)d(t)$ avec $P = U \times I$, est de E = 7.168 mJ. Un paquet d'une taille de 128 Octets est composé d'un header de 12 Octets, d'un CRC de 2 Octets et du message à envoyer sur 114 Octets.

5.5 Comparaison des performances

Dans cette section, nous donnerons une comparaison de performances de nos trois variantes RL-Dba, RL-DNAF et RL-Dswa avec les solutions les plus appropriées pour le calcul d'une multiplication scalaire. Nous appliquons nos solutions sur deux types de clusters : un cas particulier (cluster à 2 nœuds) et un cas généralisé (un cluster à 10 nœuds). Pour cela, nous présenterons dans un premier temps l'énergie nécessaire pour les traitements arithmétiques et les différentes opérations de doublement et d'addition de points en coordonnées jacobiennes standards et co-Z. Ensuite, nous calculerons l'énergie consommée par notre approche distribuée pour les trois variantes RL-Dba, RL-DNAF et RL-Dswa, puis nous comparerons nos résultats à d'autres solutions optimisées pour le calcul d'une multiplication scalaire. L'énergie consommée comprend l'énergie de calcul et l'énergie de communication. L'énergie consommée pour le partage d'un secret (un message) et celle consommée pour le partage d'un point (une clé) sont équivalentes car l'énergie dissipée pour le calcul de E = C + m et m = C – E est négligeable.

5.5.1 Coût en énergie des calculs elliptiques

Le tableau 5.3 présente la consommation en énergie, en milli-Joules (mJ), des opérations arithmétiques modulaires pour une longueur du scalaire de 192, 224 et 256 bits. Nous constatons qu'une multiplication consomme autant qu'une élévation au carré dans un corps fini premier.

TABLEAU 5.3 – Énergie consommée dans les calculs arithmétiques (en mJ)

Opération	P-192R1	P-224R1	P-256R1
Multiplication	0.044	0.059	0.086
Elévation au carré	0.044	0.059	0.086
Addition	0.0013	0.0018	0.0028
Inversion	1.357	1.755	2.283

Le tableau 5.4 suivant présente l'énergie consommée, en milli-Joules, par les opérations de doublement et d'addition de points en coordonnées J (jacobiennes standards) et en coordonnées co-Z.

TABLEAU 5.4 – Energie consommée dans les opérations de doublement et d'addition de points en coordonnées J (jacobiennes standards) et co-Z (en mJ)

Opération	Coordonnées	P-192R1	P-224R1	P-256R1
StdJDbl	J	0.457	0.613	0.892
coZDbl	co-Z	0.281	0.373	0.531
StdJAdd	J	0.724	0.971	1.411
coZAddC	co-Z	0.381	0.502	0.720
coZAddU	co-Z	0.284	0.373	0.531
FinalInv (1/Z)	co-Z	1.539	1.997	2.570

Nous constatons que l'énergie dissipée dans les opérations de doublements et d'addition de points en coordonnées co-Z est nettement inférieure en coordonnées jacobiennes standards.

5.5.2 Évaluation des performances de RL-Dba et RL-DNAF

Nous évaluons dans cette section nos deux variantes RL-Dba et RL-DNAF et nous comparons leurs performances aux deux algorithmes 16 et 17. Nous commençons par donner le coût en énergie dans les calculs puis dans les communications et enfin le coût global dans des clusters de deux nœuds et dix nœuds. Le tableau 5.5 présente le coût en énergie, en milli-Joule, dans les calculs elliptiques pour un scalaire de longueur 192, 224 et 256 bits.

Nous constatons que les nœuds ordinaires des deux solutions proposées consomment moins de ressources que ceux des algorithmes 16 et 17. Cette différence s'explique par le

TABLEAU 5.5 – Énergie nécessaire pour le calcul d'une multiplication scalaire (en mJ)

Nœud	Algorithme	P-192R1	P-224R1	P-256R1
sensorCalc	RL-Dba	87.90	137.47	228.51
	RL-DNAF	87.90	137.47	228.51
Ordinaire	RL-Dba	69.56	108.79	180.67
	Algo 16	157.46	246.26	409.18
	RL-DNAF	46.37	72.52	120.44
	Algo 17	134.27	209.99	348.95

fait que nos deux solutions n'exécutent que des additions de points sur des doublements de points précalculés par le nœud calculateur (SensorCalc). Le coût de transmission des points précalculés est donné par le tableau 5.6.

TABLEAU 5.6 – Énergie de réception (Rx) des points précalculés (en mJ)

Opération	Métriques	P-192R1	P-224R1	P-256R1
Réception Rx	Nombre de paquets	7	8	9
	Temps d'envoi (ms)	358	409	460
	Energie de réception (mJ)	50.1	57.2	64.4
Emission Tx	Nombre de paquets	1		
	Temps d'envoi (ms)	8.8		
	Energie d'émission (mJ)	1.23		

L'exécution de l'algorithme d'échange de secret entre deux nœuds coûte deux multiplications scalaires par nœud. Dans un cluster de 10 nœuds, chaque nœud peut effectuer jusqu'à 18 multiplications scalaires pour établir une paire de clés ou échanger un secret avec chacun des nœuds du cluster. Le tableau 5.7 résume la consommation d'énergie par nœud pour un cluster de deux nœuds et un autre cluster de 10 nœuds.

Soit n le nombre de nœuds ordinaires formant le cluster, sans prendre en compte le ou les nœuds calculateurs. Le nombre de clés (#keys) est au maximum $n(n-1)/2$, le nombre de multiplications scalaire kP par nœud (#kP/node) est au maximum n, le nombre total de multiplications scalaires à effectuer (#kP) est n^2. Le coût en consommation énergétique des deux algorithmes proposés RL-Dba et RL-DNAF est calculé comme suit : $nE(Add)+nE(Rx)+1E(Tx)$, avec E(Add) : Energie d'addition de points, E(Rx) : Energie de réception et E(Tx) : Energie d'émission de paquets.

Nous constatons que la consommation d'énergie par nœud et par l'ensemble des nœuds du réseau dans notre approche est meilleure que dans les autres algorithmes. Notre approche Right-to-Left Distributed NAF (RL-DNAF) permet d'économiser jusqu'à 55%, 47% et 25% pour un cluster de 10 nœuds ayant un niveau de sécurité 128 bits (avec une clé de 256 bits) par rapport respectivement à l'algorithme 16, 17 et RL-Dba. L'énergie par nœud dans notre approche comprend l'énergie de calcul, d'émission et de réception de paquets de résultats intermédiaires fournis par le nœud calculateur.

TABLEAU 5.7 – Energie consommée par les algorithmes proposés et d'autres algorithmes pour l'échange d'un secret dans un cluster de deux et de dix nœuds (en Joule)

Cluster	#keys	#kP / node	#kP	Algorithme	P-192R1	P-224R1	P-256R1
2	1	2	4	Algo 16	0.314	0.490	0.816
				RL-Dba	0.240	0.333	0.491
				Algo 17	0.268	0.419	0.697
				RL-DNAF	0.194	0.260	0.371
10	< 45	< 10	< 100	Algo 16	1.574	2.462	4.091
				RL-Dba	1.197	1.661	2.451
				Algo 17	1.342	2.099	3.489
				RL-DNAF	0.965	1.298	1.849

Par exemple, dans notre approche un nœud muni d'une clé de taille 192 bits consomme 69.56 mJ pour les calculs, 1.23 mJ pour envoyer des données et 50.1 mJ pour la réception des paquets pour chaque opération de multiplication scalaire. Il exécute en moyenne 96 additions, 0 doublements, émis 1 paquets de données et reçoit 7 paquets. Pour l'algorithme 17, une opération de multiplication scalaire kP coûte 46.37 mJ pour les additions et 87.9 mJ pour les doublements de points nécessitant un coût global de 134.27 mJ pour une seule multiplication scalaire. Le coût de notre algorithme RL-DNAF est de 97.7 mJ calculé comme suit : E(Add) = 46.37 mJ, E(Rx) = 50.1 mJ et E(Tx) = 1.23 mJ. Pour un simple échange de clés entre deux capteurs nécessitants 4 opérations kP, notre approche distribuée consomme 194 mJ contre 268 mJ pour le meilleur des autres algorithmes, pour une clé de 192 bits.

5.5.3 Évaluation des performances de RL-Dswa

La deuxième solution que nous proposons permet de paralléliser l'algorithme Right-to-Left signed window algorithm (algorithme 42). Notre solution permet de réduire de $w(l-1)$ doublements pour chaque opération kP. L'énergie consommé dans les opérations de précalculs est négligeable pour $w = 2$, 3 et 4, comme démontré dans [117] et [118] et n'est pas prise dans les calculs dans ce travail.

TABLEAU 5.8 – Comparaison du nombre d'opérations de doublement et d'addition de points des deux algorithmes 18 et RL-Dswa

Operation	Algorithme 18	RL-Dswa
Doubling	$w(l-1)+l$	0
Addition	$(l-1)+\frac{l}{w+1}+2^{w-2}$	

Le tableau 5.9 ci-dessous présente l'énergie de réception des paquets intermédiaires de données calculées et transmises par le nœud calculateur pour $w = 2$, 3 et 4.

L'énergie de réception et le nombre de paquets de données intermédiaires dépendent

TABLEAU 5.9 – L'énergie de réception des paquets intermédiaires de données

w	Operation	P-192R1	P-224R1	P-256R1
2	Nombre de paquets	4		5
	Temps d'envoi (ms)	204		256
	Energie (mJ)	28.6		35.8
3	Nombre de paquets	3		
	Temps d'envoi (ms)	153		
	Energie (mJ)	21.4		
4	Nombre de paquets	2		3
	Temps d'envoi (ms)	102		153
	Energie (mJ)	14.3		21.4

du nombre de fenêtres. Pour $w = 4$, le nombre de paquets à recevoir est de 2 paquets pour secp192r1 et secp224r1 contre 3 pour secp256r1.

TABLEAU 5.10 – L'énergie (en mJ) de calcul d'une multiplication scalaire pour l'algorithme 18 et RL-Dswa

	Algorithme	P-192R1	P-224R1	P-256R1
w = 2	Algo 18	223	261	298
	RL-Dswa	136	159	182
w = 3	Algo 18	174	204	272
	RL-Dswa	87	103	156
w = 4	Algo 18	152	177	202
	RL-Dswa	66	77	87

Le tableau 5.10 ci-dessus montre le temps de calcul et la consommation d'énergie pour $w = 2, 3$ et 4 pour l'algorithme 18 et l'algorithme proposé RL-Dswa. Globalement, le temps et la consommation diminue pour un w plus grand. Ainsi, pour un niveau de sécurité de 96 bits et un $w = 4$, notre approche consomme 86 mJ de moins pour chaque opération kP. Pour une opération de partage de secret, notre algorithme consomme 216 mJ de moins que l'algorithme 18 comme le montre le tableau 5.11 ci-dessous.

5.5.4 Comparaison globale

Cette comparaison est faite sur l'ensemble des résultats obtenus dans ce chapitre. Nous remarquons que pour n'importe quel niveau de sécurité, les deux algorithmes que nous avons proposés (RL-DNAF et RL-Dswa) consomment moins d'énergie que ce soit par nœud ou dans un cluster de 10 nœuds comme le montre la Figures 5.6 ci-dessous. L'algorithme proposé 4-RL-Dswa, pour une clé de 256 bits, offre un meilleur résultat.

5.6 Analyse de sécurité

L'attaque par canaux auxiliaires (Side Channel Attack SCA) demeure la principale menace qui tente à exploiter les failles dans l'implémentation de l'algorithme de calcul

TABLEAU 5.11 – Énergie consommée par l'algorithme RL-Dswa et par l'algorithme 18 pour l'échange d'un secret dans un cluster de deux et de dix nœuds (en Joule)

Cluster	#keys	#kP/ nœud	#kP	w	Algorithme	P-192R1	P-224R1	P-256R1
2	1	2	4	2	Algo 18	0.446	0.522	0.596
					RL-Dswa	0.301	0.346	0.399
				3	Algo 18	0.348	0.408	0.544
					RL-Dswa	0.195	0.227	0.333
				4	Algo 18	0.304	0.354	0.404
					RL-Dswa	0.146	0.168	0.195
10	< 45	< 18	< 180	2	Algo 18	4.014	4.698	5.364
					RL-Dswa	2.505	2.904	3.598
				3	Algo 18	3.132	3.672	4.896
					RL-Dswa	1.758	2.046	3.001
				4	Algo 18	2.736	3.186	3.636
					RL-Dswa	1.316	1.578	1.758

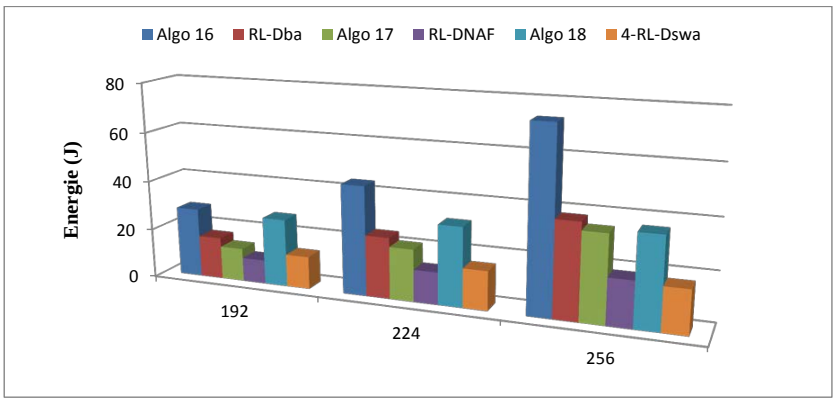

FIGURE 5.6 – Comparaison globale de la consommation d'énergie dans un cluster de 10 nœuds pour l'échange de secrets

de la multiplication d'un scalaire par un point. L'objectif étant d'utiliser les propriétés physiques du matériel pour reproduire la clé. Ces attaques peuvent être algorithmiques et physiques. Dans le cadre de la cryptographie des courbes elliptiques, tous les messages sont en clair et la difficulté de retrouver la clé dépend de la capacité de l'attaquant à résoudre le problème du logarithme discret sur lequel repose la sécurité des échanges. Nous nous intéressons aux attaques physiques et très particulièrement aux attaques temporelles basées sur une comparaison du temps mis pour effectuer les opérations et les attaques par analyse de consommation d'énergie qui consistent à analyser la consommation en courant.

Les deux algorithmes que nous avons présenté dans ce travail permettent de contrer efficacement ces deux attaques : pour l'algorithme RL-Dba, un nœud capteur effectue un test de bit et si c'est un bit de 1 on effectue une addition de points sinon on incrémente la boucle. Un test de bit prend 4 μs et consomme 3.03×10^{-4} mJ contre une addition de points consommant respectivement 9566 μs (0.724 mJ), 12824 μs (0.971 mJ) et 18634 μs (1.411 mJ) pour une clé de taille 192 bits, 224 bits et 256 bits.

La consommation en temps et en énergie pour le test de bit demeurent négligeables et représentent environ 0.03 % du temps et de l'énergie globale consommée pour chaque itération, ce qui ne permettrait pas à un attaquant de connaitre la valeur du test de bit. Pour l'algorithme RL-Dswa, quel que soit la valeur du bit, une opération d'addition de points est donc effectuée. Dans le cas où le bit vaut 0, l'algorithme utilise une inversion de points. Le coût de cette opération est négligeable en coordonnées Jacobéennes car il suffit d'inverser le signe de la coordonnée Y du point $P(X : Y : Z)$ pour obtenir son inverse $P\prime(X : -Y : Z)$. L'opération de l'inversion s'effectue en moins de 1 μs et par conséquent négligeable.

5.7 Conclusion

Dans ce chapitre, nous avons proposé une nouvelle approche basée sur des traitements parallèles et distribués accomplis par plusieurs nœuds d'un même cluster ou d'un même réseau sur des calculs complexes. C'est une solution destinée à des réseaux denses effectuant un nombre important d'opérations de partage de secrets comme les réseaux de capteurs sans fil.

Dans ce cadre, nous avons développé trois solutions qui permettent de parallèliser et de distribuer les calculs : RL-Dba est une variante de l'algorithme standard Right-to-Left Double-and-Add, RL-DNAF, une variante de l'algorithme Right-to-Left NAF et RL-Dswa une variante de l'algorithme Right-to-Left signed window algorithm. Ces trois variantes permettent d'éliminer les calculs de doublement de points sur un nœud du réseau et de les confier à des nœuds calculateurs (sensCalc). Nous avons constaté qu'avec notre solution, nous avons pu réduire jusqu'à 125% la consommation d'énergie

d'un nœud capteur ce qui rallonge la durée de vie du réseau.

Globalement, les calculs redondants ont été éliminés comme le calcul des doublements exécutés lors d'une multiplication scalaire par un point kP quelle que soit la valeur de la clé k multipliée par un même point P partagé par deux nœuds A et B. Ceci permet de réduire le nombre de doublements de l (l étant la taille du scalaire k) pour chaque opération de multiplication scalaire par un point.

De plus, la sécurité des communications et des opérations de partage de secrets est garantie. Notre solution résiste aux attaques par canaux cachés et par analyse de temps de calculs.

Chapitre 6
Protocoles de sécurité distribués basés sur les courbes elliptiques pour les réseaux de capteurs sans fil

La majorité des solutions existantes pour l'optimisation de la performance de multiplication scalaire sont appliquées sur un seul nœud ou bien sur plusieurs nœuds mais pour effectuer une seule multiplication scalaire. La solution que nous proposons dans ce chapitre s'applique à plusieurs nœuds à la fois et permet d'effectuer une multiplication scalaire sur chaque nœud. Elle permet également d'équilibrer les charges entre les nœuds afin d'accélérer les calculs tout en optimisant la consommation de ressources. Ainsi, un groupe de nœuds formant un cluster collaborent entre eux pour effectuer des calculs communs sur des données différentes de type SIMD (Single Intsruction Multiple Data). Chaque nœud exécute un sous ensemble d'opérations et obtient un résultat partiel qu'il partage avec les autres nœuds. A la fin de l'opération, chaque nœud rassemble tous les résultats partiels reçus pour lancer le calcul d'une multiplication scalaire.

La communication entre les nœuds s'effectue via des liaisons sans fil par l'échange de messages publics ne nécessitant aucune opération de chiffrement/déchiffrement. Le nombre de messages échangés durant toutes les phases du protocole est très limité ce qui ne constitue aucunement un inconvénient aux calculs distribués. L'échange de données se fait d'une manière séquentielle où la tâche j est tributaire de la tâche i. Si un nœud tombe en panne, le nœud master intervient pour le remplacer.

Deux solutions sont proposées dans ce chapitre. La première solution utilise des points précalculés par d'autres nœuds, dits esclaves, et préchargés dans la mémoire du nœud, soit avant le déploiement ou reçus après le déploiement, et la seconde sans les points préchargés. La première solution est plus efficace en terme de communications où chaque nœud envoi un seul message contre deux messages reçus quel que soit le nombre de nœuds du cluster. Son inconvénient est le nombre important de points à stocker qui est de $(l-1)$ points pour une clé de l bits. La seconde solution est plus efficace en terme de mémoire où chaque nœud stocke au plus deux points quel que soit le nombre de nœuds. Cette solution nécessite beaucoup de communications où chaque nœud doit recevoir $(m-1)$ messages avec m est le nombre de nœuds du cluster. Dans la suite de ce chapitre, nous présenterons nos deux protocoles avec points préchargés DPSM (Distributed and Parallel Scalar point Multiplication) et sans points préchargés FDSM (Full Distributed Scalar point Multiplication) ainsi que les deux algorithmes

de calculs parallèles de doublements de points DblP et d'addition de points AddP que nous détaillerons dans la suite.

6.1 Fonctionnement global

Notre travail s'appuie sur les travaux de parallélisation des calculs d'une multiplication scalaire, notamment ceux de [119], mais sur un corps premier fini et la possibilité de paralléliser les calculs en deux sous tâches indépendantes : une sous tâche pour les doublements de points et une autre pour les additions de points. En effet, les calculs d'une multiplication scalaire sont distribués sur les nœuds du cluster. Tous les nœuds effectuent des sous tâches de doublement de points puis s'échangent les résultats entre eux pour calculer chacun de son côté une multiplication scalaire. Sur la Figure 6.1, on peut déduire qu'un doublement peut être réalisé sans connaissance préalable du secret ce qui est hyper important pour la sécurité du réseau. Les additions de points sont effectuées suivant la valeur du scalaire k. Les deux protocoles proposés ne fournissent aucune information sur le scalaire k en partageant uniquement des paramètres elliptiques, ce qui accroit leur sécurité.

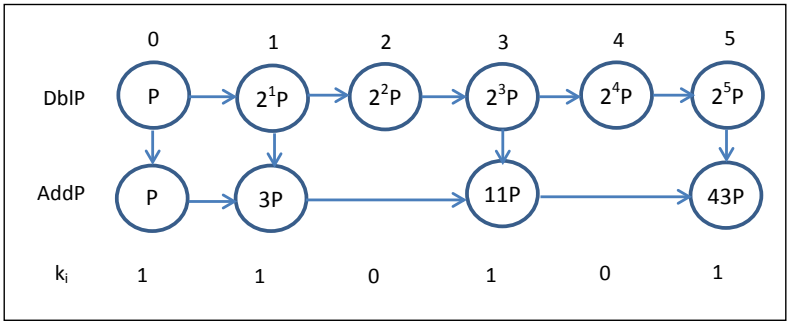

FIGURE 6.1 – Exécution parallèle des deux opérations de doublement et d'addition de points avec $k = 110101$

Pour distribuer une multiplication scalaire $Q = k \cdot P$ avec P, Q deux points de la courbe elliptique E définie sur un corps fini premier F_P et k un entier de longueur l représenté en binaire par : $k = \sum_{i=0}^{l-1}(k_i 2^i)$

Le nœud qui initialise et distribue les données et les sous tâches est le nœud maitre, et les autres nœuds participants sont des nœuds esclaves. Le nœud maitre décompose les calculs en m segments selon le nombre de nœuds du cluster. La longueur d'un segment S est $t = \frac{l}{m}$. chaque nœud i calcule t doublements du point reçu par le nœud $i-1$. La formule de calcul de $S_i = t \cdot S_{i-1}$ est donnée par : $S_i = 2^j \cdot S_{i-1}$ avec $j = [\![t_{1+1}; t_1 + t]\!]$. t_1 est fourni par S_{i-1} et représente sa borne supérieure de calcul. L'algorithme de doublement est décrit par l'algorithm 44.

Algorithme 44 Algorithme de doublement de points DblP

Require: t_1, t_2, P $\in$ E(F_P)
Ensure: 2^iP avec $i = [t_1, t_2 - 1]$
  $Q_{t_{1-1}}$ = P
  for $i \leftarrow t_1$ to $t_2 - 1$ **do**
    $Q_i \leftarrow 2 \times Q_{i-1}$
  end for
  return Q_{t_2-1} ;

Algorithme 45 Algorithme d'addition de points AddP

Require: $k = \sum_{i=0}^{l-1}(k_i 2^i)$, $2^iP = \{P, 2P, 4P, 8P, \ldots, 2^{l-1}P\}$ pour $i = [\![0, l-1]\!]$
Ensure: Q = k·P
  $Q = \emptyset$
  for $i \leftarrow l - 1$ downto 0 **do**
    if $k_i \neq 0$ **then**
      $Q \leftarrow Q + (k_i.2^iP)$
    end if
  end for
  return Q ;

L'addition de points reçus par un nœud et qui représentent les doublements du point P pour le calcul d'une multiplication scalaire $Q = k \cdot P$ est décrit par l'algorithme 45.

Soit $k = ([k_{l-1}, \ldots, k_m], [k_{m-1}, \ldots, k_j] \ldots [k_{b-1}, \ldots, k_a], [k_{a-1}, \ldots, k_0])$ tel que $k_i \in \{0, 1\}$ avec m nœuds participants, chaque nœud calcule une multiplication scalaire comme suit : $Q = \sum_{u=1}^{m} \sum_{i=0}^{l-1}(k_i S_{uv})$, tel que $v = \{1, \ldots, t\}$. Les points S_{uv} calculés par les nœuds u sont donnés par :

$$\left\{ \begin{array}{rcl} S_0 & = & k_0 S_{01} + k_1 S_{02} + \ldots + k_{(a-1)} S_{0t} \\ S_a & = & k_a S_{a1} + k_{a+1} S_{a2} + \ldots + k_{(b-1)} S_{at} \\ \ldots & & \\ S_j & = & k_j S_{j1} + k_{j+1} S_{j2} + \ldots + k_{(m-1)} S_{jt} \\ S_m & = & k_m S_{m1} + k_{m+1} S_{m2} + \ldots + k_{(l-1)} S_{mt} \end{array} \right\}$$

Ainsi, $Q = S_0 + S_a + \ldots + S_j + S_m$

Dans ce qui suit, nous présentons nos deux protocoles. Le protocole Distributed and Parallel Scalar point Multiplication (DPSM) utilise des points préchargés reçus par le nœud maître et calculés par les nœuds esclaves. L'opération de doublement dans DPSM est distribuée tandis que l'addition est parallèle. Le protocole Full Distributed Scalar point Multiplication (FDSM) est complétement distribué et ne nécessite aucun point préchargé. Les additions sont calculées au fur et à mesure de la réception des points doublés. Leur fonctionnement est décrit dans les sections suivantes.

6.2 Calcul distribué d'une multiplication scalaire

Dans cette section, nous présentons les deux protocoles distribués pour la multiplication scalaire. Le protocole DPSM utilise un ensemble de points précalculés et préchargés pour exécuter les calculs d'une multiplication scalaire tandis que le protocole FDSM les exécute sans aucune donnée préchargée. De ce fait, DPSM est gourmand en ressources mémoire et FDSM en communications. Leur application dépend du domaine d'utilisation et des caractéristiques des appareils utilisés. Les deux algorithmes sont décrits respectivement par les algorithmes 47 et 48 et l'algorithme 49.

6.2.1 Avec points préchargés

Le protocole Distributed and Parallel Scalar point Multiplication (DPSM) est un protocole à la fois distribué et parallèle. Les doublements de points sont distribués sur les nœuds du cluster et les additions sont calculées en parallèle. Le principe est donné sur la Figure 6.2 et décrit par l'algorithme 46.

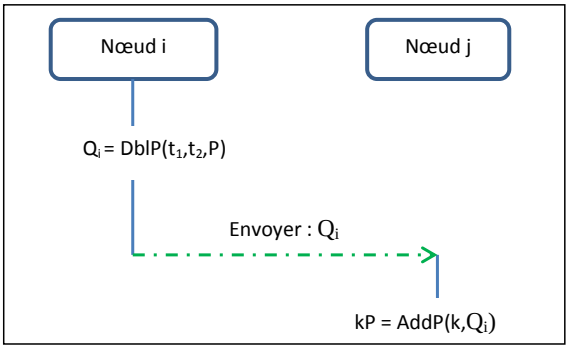

FIGURE 6.2 – Principe de fonctionnement global du protocole DPSM

Algorithme 46 Fonctionnement global de l'algorithme DPSM

Require: $k = \sum_{i=0}^{l-1}(k_i 2^i)$, $P \in E(F_p)$
Ensure: Q = k·P

1. Attendre() ;
2. Réception $(2^i P)$;
3. Q = AddP$(k, 2^i P)$;

return Q ;

La Figure 6.3 est une généralisation du principe de calcul d'une multiplication scalaire présenté sur la Figure 6.2. Le protocole DPSM est appliqué sur un cluster de $(m-1)$ nœuds esclaves et d'un seul nœud maître. Le nœud maître génère les paramètres elliptiques, fixe le nombre de nœuds participants m, calcule la longueur de

chaque segment t égale à la taille de l'entier k divisé par le nombre de nœuds $m (t = \frac{l}{m})$ et fixe la valeur de départ de w à 0 $(w = 0)$. Au lancement du round de calculs, le nœud master envoie un message à un nœud ayant le plus petit id_a, désigné par SNode-a, avec les paramètres P, t et w. A la réception du message, SNode-a calcule la borne supérieure $a = t + w$ et exécute l'algorithme $DblP(w, a, P)$ pour calculer tous les doublements de points $2^x P$ avec $x \in [w, a]$ et les stocker dans un tableau S_a d'une taille a. Le tableau S_a est envoyé au prochain nœud SNode-b ayant un $id_b = id_a + 1$ et au nœud maître.

SNode-b fait de même que SNode-a jusqu'à atteindre SNode-j (avec $j = m-1$). Une fois que tous les calculs sont réalisés, le nœud master envoie un message contenant tous les doublements du point P à tous les nœuds esclaves pour calculer en parallèle $Q = k_i \cdot P$ ou k_i est le scalaire du nœud SNode-i en utilisant l'algorithme $AddP(k_i, 2^i P)$. A la fin de l'exécution du protocole DPSM, le nombre de multiplications scalaires obtenues est de m contrairement à d'autres solutions distribuées où les m nœuds collaborent pour calculer une seule multiplication scalaire comme dans [107]. Le protocole est décrit par les deux algorithmes 47 pour le nœud maître (Master Node) et par l'algorithme 48 pour le nœud esclave (Slave Node).

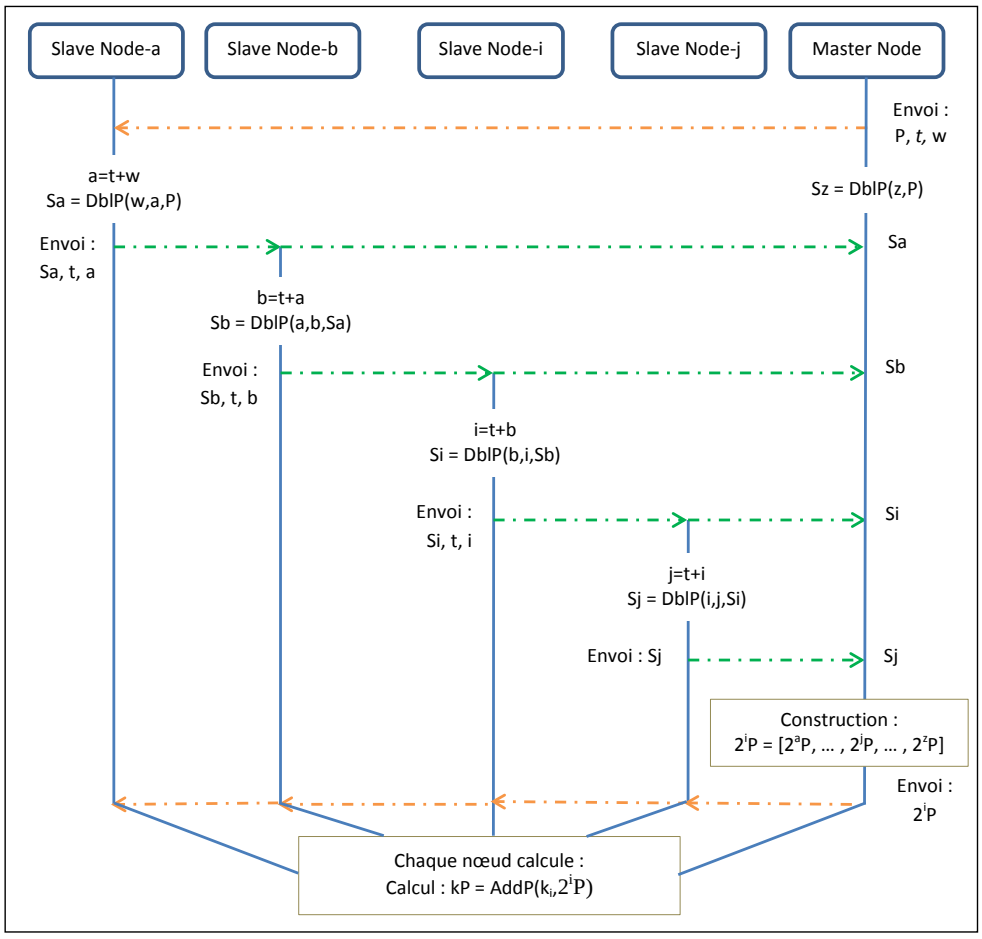

FIGURE 6.3 – Principe de fonctionnement du protocole DPSM

Algorithme 47 Algorithme DPSM - Calcul de kP par le nœud maître

Require: $k = \sum_{i=0}^{l-1}(k_i 2^i)$, $P \in E(F_p)$
Ensure: Q = k·P

1. Génération des paramètres elliptiques ;
2. $t = l/m$;
3. $m = l - t$;
4. $Q_x = \mathrm{DblP}(m, l, R)$;
5. $\mathrm{Envoyer}(Q_x, t, 0, id_{SlaveNode-a})$;
6. Attendre() ;
7. Réception $(2^x P)$; // A partir du nœud esclave
8. Construction $2^i P = [2^a P, \ldots, 2^j P, \ldots, 2^m P]$;
9. $\mathrm{Envoyer}(2^i P, *)$;
10. Q = $\mathrm{AddP}(k, 2^i P)$

return Q ;

Algorithme 48 Algorithme DPSM - Calcul de kP par le nœud esclave

Require: $k = \sum_{i=0}^{l-1}(k_i 2^i)$, $P \in E(F_p)$
Ensure: Q = k·P

1. Attendre() ;
2. Réception(R, t, w) ;
3. $x = t + w$;
4. $Q_x = \mathrm{DblP}(w, x, R)$;
5. $\mathrm{Envoi}(Q_x, t, a, id_{MasterNode}, id_{NextNode})$;
6. Attendre() ;
7. $\mathrm{Réception}(2^i P)$; // A partir du nœud maître
8. Q = $\mathrm{AddP}(k, 2^i P)$

return Q ;

6.2.2 Sans points préchargés

Le protocole DPSM nous permet de distribuer puis paralléliser les calculs des multiplications scalaires sur les courbes elliptiques mais les nœuds participants doivent stocker en mémoire $t \times m$ points. Dans ce qui suit, nous présentons un autre protocole complètement distribué sans points préchargés. Son principe est donné par la Figure 6.4. Le protocole FDSM est appliqué sur un cluster de m nœuds esclaves et le nœud ayant le plus grand id jouera le rôle du nœud maitre en générant les paramètres elliptiques, fixe le nombre de nœuds participants m, calcule la longueur de chaque segment t égale à la taille de l'entier k divisé par le nombre de nœuds m ($t = \frac{l}{m}$) et fixe la valeur de départ de w à 0 ($w = 0$).

Au lancement du round de calculs, le nœud master envoie un message à un nœud ayant le plus petit id_a, désigné par SNode-a, avec les paramètres P, t et w. A la réception du message, SNode-a calcule la borne supérieure $a = t + w$ et exécute l'algorithme $\mathrm{DblP}(w, a, P)$ pour calculer tous les doublements de points $2^x P$ avec $x \in \{w, a\}$ et les stocker dans un tableau S_a d'une taille a. Le tableau S_a est envoyé à tous les nœuds du cluster. A la réception du message de SNode-a, chaque nœud i exécute l'algorithme d'addition de points $\mathrm{AddP}(a, a + t, S_a)$ pour calculer $Q_a = k_{ix}.\ S_a$ où k_i est le scalaire du nœud SNode-i et $x \in [a, a+t]$. SNode-b fait de même que SNode-a jusqu'à atteindre SNode-m. Une fois que tous les calculs sont réalisés, chaque nœud du cluster calcule $Q = \sum_{i=1}^{m} Q_i$. A la fin de l'exécution du protocole FDSM, comme pour DPSM, le nombre de multiplications scalaires obtenues est de m. Le protocole est décrit par l'algorithme 49.

6.3 Échange d'un secret sans points préchargés

Nous présentons dans cette section un cas pratique de l'algorithme proposé FDSM pour l'échange d'un secret entre deux nœuds d'un réseau soit en mode Master/Slave (cluster de 2 nœuds) ou bien en mode cluster (cluster de plusieurs nœuds).

6.3.1 Master / Slave

Dans ce mode, deux nœuds, Master Node et Slave Node veulent échanger un secret. Le nœud Master effectue les principales opérations en calculant le point public M ainsi que les doublements des points P et M, puis envoie tous les résultats au nœud Slave. Ce dernier effectue uniquement des additions de points pour le calcul des deux points publics S et R qu'il transmet par la suite au nœud Master. Le nœud Master retrouve le secret en calculant $m = k_M S - R$. Le principe est décrit par la Figure 6.5.

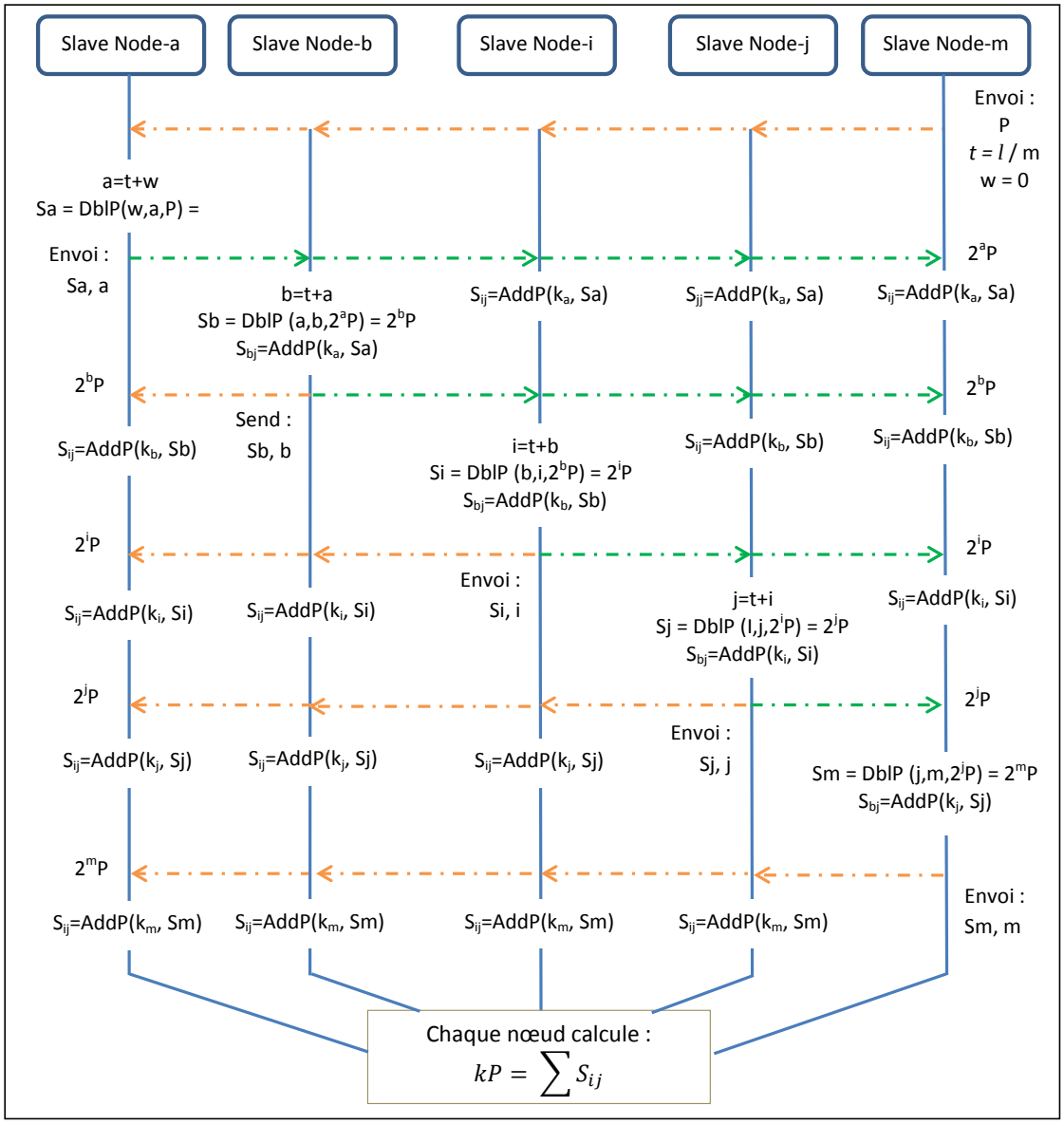

FIGURE 6.4 – Principe de fonctionnement du protocole FDSM

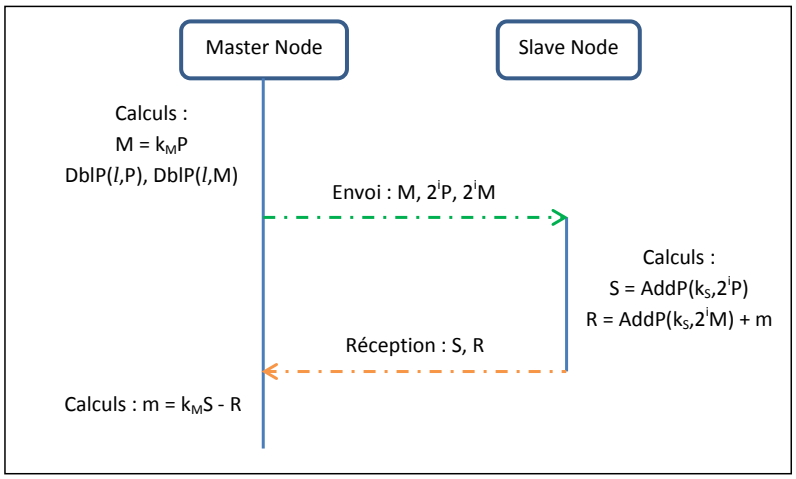

FIGURE 6.5 – Echange de secrets entre deux nœuds avec le protocole FDSM

Algorithme 49 Algorithme FDSM - Calcul de kP

Require: $k = \sum_{i=0}^{l-1}(k_i 2^i)$, $P \in E(F_p)$
Ensure: Q = k·P

1. Attendre() ;
2. Réception (R, t, w) ;
3. $x = t + w$;
4. $Q_x = \mathrm{DblP}(w, x, R)$;
5. Envoi$(Q_x, t, x, *)$;
6. Attendre() ;
7. $j = 0$;
8. $S_j = \emptyset$;
9. TantQue $(j < m)$
10. Réception (Q_j, t, w) ;
11. $S_j = S_j + Q_j$;
12. Supprimer Q_j ;
13. $++j$;
14. FinTantQue
15. Q = AddP(k, Q_x) ;

return Q ;

6.3.2 Cluster

Dans un cluster de n nœuds, deux nœuds Slave 1 et Slave 2 souhaitent échanger un secret m. Les calculs intermédiaires sont confiés au nœud Master. Le nœud Master calcule d'abord les doublements du point P qu'il transmet simultanément aux deux nœuds Slave 1 et Slave 2. Les deux nœuds calculent chacun de son côté les points publics S_1 et S_2 qu'ils envoient au nœud Master. Ce dernier calcule les doublements de S_1 et S_2 et les transmet respectivement à Slave 2 et Slave 1. Le nœud ayant le secret m (Slave 2 dans notre cas) calcule le point public R et l'envoi au nœud Master. Le nœud Master calcule les doublements du point R et les envois au Slave 1. Le nœud Slave 1 retrouve le secret en calculant $m = \mathrm{AddP}(k_1, 2^i S_2) - \mathrm{AddP}(k_1, 2^i R)$. Le principe est décrit par la Figure 6.6.

Dans la section suivante, nous allons comparer les performances de nos deux protocoles avec d'autres solutions de la littérature très efficaces pour les calculs scalaires d'un point.

6.4 Paramètres d'implémentation

Dans cette section nous évaluons les deux protocoles DPSM et FDSM en temps d'exécution, exigences en stockage mémoire et en consommation d'énergie. Nous com-

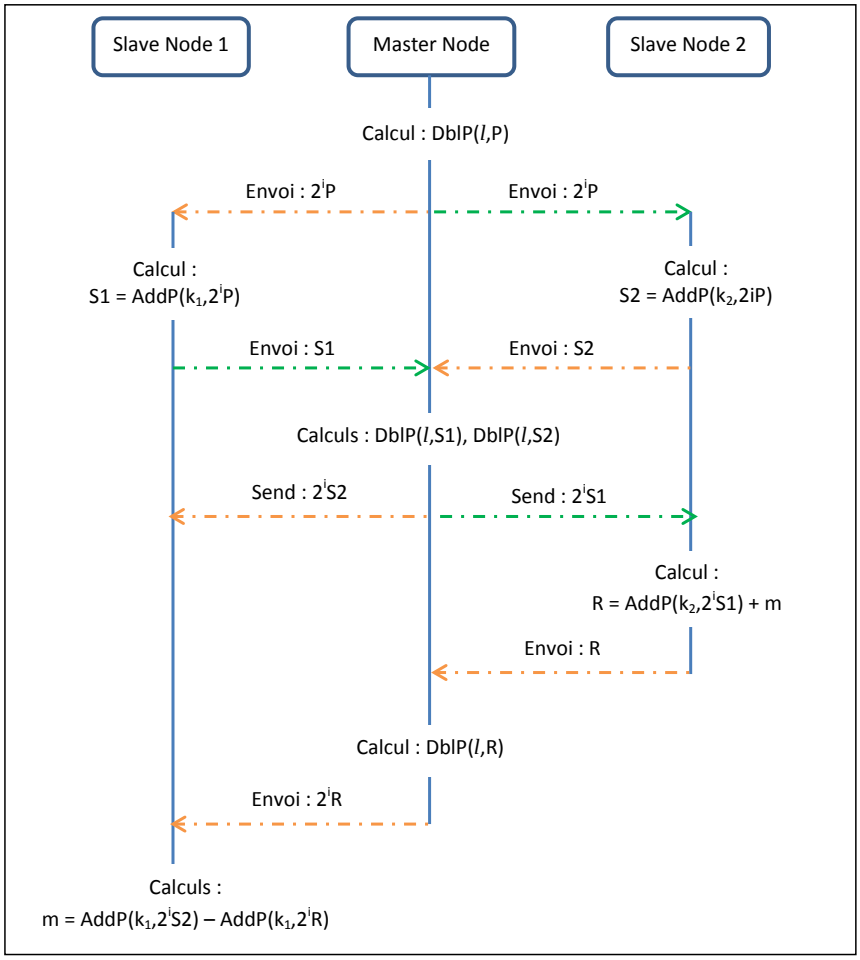

FIGURE 6.6 – Echange de secrets dans un cluster de n nœuds avec le protocole FDSM

mençons par présenter les paramètres elliptiques utilisés dans nos expérimentations, l'architecture de notre Arduino Embedded System, les paramètres de la carte sans fil utilisée dans les communications radio, puis le coût des deux protocoles proposés DPSM et FDSM.

6.4.1 Paramètres elliptiques

Les paramètres elliptiques choisis pour nos expérimentations sont ceux recommandés par NIST [113] et décrit à la section A de ce document, d'une courbe elliptique E définie sur un corps fini premier F_P spécifiés par le sextuple $(p, a, b, G, n, 1)$ représentés sur les tableaux A.1, A.2 et A.3.

6.4.2 Architecture matérielle

Le système embarqué utilisé pour les expérimentations est la carte Arduino Uno R3 [120]. Arduino est une plate-forme open source qui offre un bon environnement pour l'informatique embarquée. Elle est largement utilisée dans les applications de l'Internet des objets et ses caractéristiques sont presque identiques à celles des capteurs d'un réseau de capteurs sans fil, comme indiqué dans le tableau 6.1. Arduino Uno R3, est une carte à microcontrôleur basée sur un microcontrôleur amovible ATmega328P AVR [8, 120], fonctionnant à une fréquence d'horloge de 16 Mhz, basée sur une architecture ATMEL AVR RISC, 32 Ko de mémoire Flash, 1 KB EEPROM. et 2 Ko SRAM, et fonctionnant entre 1,8 et 5,5 Volts.

6.4.3 Modèle de consommation d'énergie

Dans cette section nous mesurons et nous calculons l'énergie consommée dans les calculs et les communications radio. Nous avons mené nos expérimentations sur la carte Arduino Uno R3 équipé du module radio CC2530 System-on-Chip [121] pour IEEE 802.15.4/ZigBee, qui sont des plates-formes populaires pour les systèmes embarqués tels que IoT et RcSF. Les principaux composants gourmands en énergie sont le microcontrôleur ATmega328P et l'antenne émetteur-récepteur sans fil à faible puissance CC2530 [121]. Notre code est écrit en langage C optimisé pour mettre en œuvre une ECC normalisée sur des corps finis F_P tel que présenté dans [122]. Le tableau 6.2 présente les caractéristiques pour la plate-forme Arduino Uno R3.

Nous notons que la transmission d'un bit équivaut à l'exécution d'environ 76 cycles d'horloge sur le microcontrôleur ATmega328P. Cela confirme que le coût en énergie de la transmission des données n'est pas excessif par rapport au coût des calculs. Le temps et l'énergie nécessaires pour transmettre une trame de données de 133 octets

TABLEAU 6.1 – Caractériqtiques techniques de quelques capteurs largement déployés dans la pratique

Matériel	Tmote Sky	Mica2	MicaZ	Telos	Telos B	Uno R3
Microcontrôleur						
Type	MSP430	ATmega128	ATmega128	MSP430	MSP430	ATmega328P
RAM (KB)	10	4	4	10	10	2
Flash (KB)	48	128	512	48	48	32
Communication						
Radio Type	CC2420	CC1000	MPR2400CA	CC2420	CC2420	CC2530
Fréquence (MHz)	2400	868 / 916	2400	2400	2400	2400
Débit (Kbps)	250	38.4	250	250	250	250
Consommation énergétique						
Voltage Min (V)	1.8	2.7	2.7	1.8	1.8	1.8
Active Mode (mA)	0.5	8	8	0.5	1.8	0.2
Mode Veille (uA)	2.6	< 15	< 15	2.6	5.1	0.1
Radio Voltage (V)	2.1	2.7	2.7	2.1	2.1	2
Energie de Réception (mA)	21.8	10	19.7	21.8	23	24
Energie de Transmission à 0 dBm (mA)	19.8	27	17.4	19.5	19.8	29

TABLEAU 6.2 – Caractéristiques de la carte Arduino Uno R3

Caractéristique		Valeur
ATmega328P	Active Mode	75.75 mW
	MIPS / Joule	211
	Joule / MIPS	4.73 mJ
	Voltage	5.112 V
CC2530	Technologie	IEEE 802.15.4/Zigbee
	Périmètre (mètre)	100 m
	Taille paquet	120 Bytes
	Temps de Transmission / Byte	32 uS
	Temps de Transmission / Trame	4.25 mS
	Energie de Transmission	2.88 uJ / Byte
	Energie de Réception	2.304 uJ / Byte
	Voltage	3.3 V

est respectivement de 4,25 ms et 0,383 mJ, tandis que la réception des mêmes données coûte 0,306 mJ.

6.5 Évaluation des performances, résultats et comparaisons

Cette section est dédiée à nos résultats. Dans un premier temps, nous discutons des coûts de calcul, de mémoire et de communication des deux protocoles DPSM et FDSM. Ensuite, nous présenterons leur coût énergétique global et nous comparerons nos résultats aux autres résultats des différentes solutions de l'état de l'art décrites dans ce travail. Le nombre de nœuds participants dans un cluster est de $m = \{3, 6, 8, 12\}$ ce qui représente environ 5 à 10% de la taille du réseau. Le nombre de points précalculés dans chaque configuration est de $(\frac{l}{m})$. Le résultat est 2^i doublements de points avec $i = [\![1, \frac{l}{m}]\!]$.

6.5.1 Coût des calculs elliptiques

Le coût des calculs d'une multiplication scalaire est donné en temps d'exécution et en consommation d'énergie. Les résultats sont présentés sur le tableau 6.3.

TABLEAU 6.3 – Le coût des calculs des deux protocoles DPSM et FDSM en temps d'exécution (ms) et en énergie (mJ)

Bit-Key	m		3	6	8	12
P-192	Point calculé		2^{64}P	2^{32}P	2^{24}P	2^{16}P
	w		64	32	24	16
	DPSM	Temps	969.72	810.91	771.20	731.50
		Energie	75.07	62.77	59.70	56.63
	FDSM	Temps	1030.86	872.04	853.77	852.72
		Energie	79.80	67.51	66.09	66.01
P-224	Point calculé		2^{74}P	2^{37}P	2^{28}P	2^{18}P
	w		74	37	28	18
	DPSM	temps	1495.91	1278.97	1219.58	1153.59
		Energie	115.80	99.01	94.41	89.30
	FDSM	Temps	1545.70	1360.67	1328.51	1316.99
		Energie	119.66	105.33	102.84	101.95
P-256	Point calculé		2^{85}P	2^{42}P	2^{32}P	2^{21}P
	w		85	42	32	21
	DPSM	Temps	2443.14	2032.59	1938.28	1834.53
		Energie	189.13	157.35	150.05	142.02
	FDSM	Temps	2501.59	2149.48	2094.13	2068.32
		Energie	193.66	166.40	162.12	160.12

Nous constatons que le temps de calcul et la consommation d'énergie par nœud

diminue en fonction de la taille du cluster pour le protocole DPSM contrairement au protocole FDSM où l'énergie consommée se stabilise dès qu'on dépasse 8 nœuds dans un cluster, comme le montre la Figure 6.7.

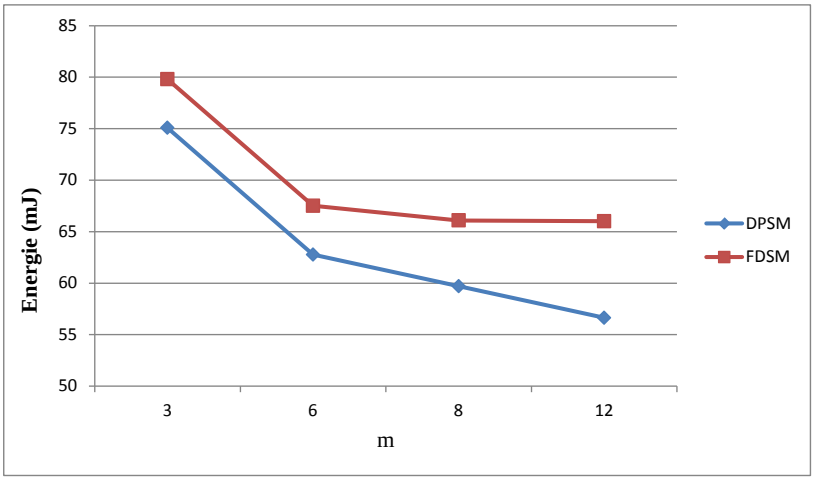

FIGURE 6.7 – Coût des calculs elliptiques des protocoles DPSM et FDSM

6.5.2 Coût de stockage mémoire

Le tableau 6.4 présente la taille mémoire allouée nécessaire pour calculer une multiplication scalaire. Les deux protocoles diffèrent selon la méthode de calcul. Le protocole DPSM requiert des points préchargés pour exécuter les opérations d'additions de points contrairement au protocole FDSM qui ne nécessite aucun point préchargé.

TABLEAU 6.4 – Coût en mémoire des deux protocoles DPSM et FDSM

	P-192				P-224				P-256			
m	3	6	8	12	3	6	8	12	3	6	8	12
Point	2^{64}P	2^{32}P	2^{24}P	2^{16}P	2^{74}P	2^{37}P	2^{28}P	2^{18}P	2^{85}P	2^{42}P	2^{32}P	2^{21}P
w	64	32	24	16	74	37	28	18	85	42	32	21
DPSM	13824				18816				24576			
FDSM	216	288	576	864	252	504	672	1008	288	576	768	1152

La Figure 6.8 montre que la quantité mémoire utilisée par le protocole DPSM est très importante. Elle représente environ 75.56% de la mémoire Flash du dispositif utilisé dans nos expérimentations, les autres comparaisons sont données sur le tableau 6.5.

Nous remarquons que dans des dispositifs récents pour RcSF comme MicaZ la consommation mémoire ne représente qu'environ 5% et 1% pour les périphériques IoT comme MKR VIDOR 4000. La consommation mémoire du protocole FDSM est insignifiante, il est recommandé pour des dispositifs faibles en capacités de stockage.

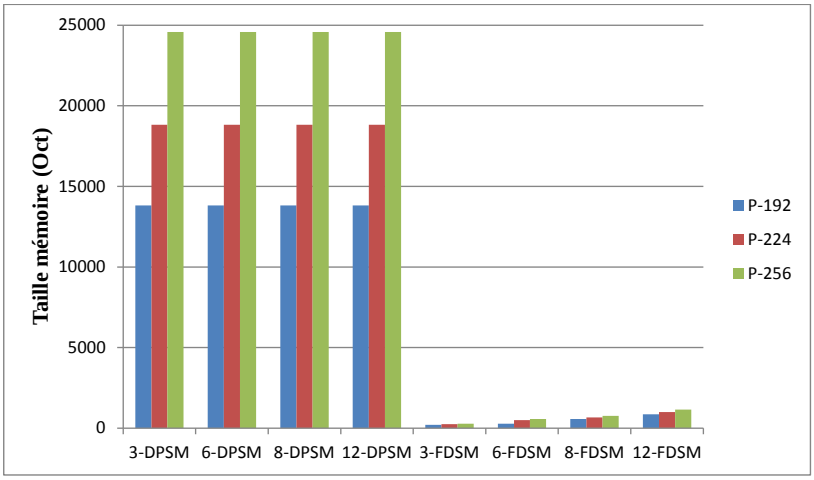

FIGURE 6.8 – Coût mémoire des protocoles DPSM et FDSM

TABLEAU 6.5 – Comparaison en coût mémoire entre la carte Arduino Uno R3 et les autres périphériques

Périphérique	Mica2	MicaZ	Telos B	Uno R3	MKR VIDOR 4000
Flash (KB)	128	512	48	32	2048
% Utilisé	18.88	4.72	50.36	75.56	1.18

6.5.3 Coût des communications

Les communications sont gérées par le module intégré CC2530 [121], un émetteur-récepteur RF conforme à la norme IEEE 802.15.4/Zigbee à 2,4 GHz. L'interface de sortie est compatible avec les broches XBee. Il prend en charge l'envoi de gros paquets de données et la diffusion rapide via UART. Le nombre de paquets envoyés et reçus par chaque nœud ainsi que la taille totale des messages échangés est donné par les tableaux 6.6 et 6.7.

On remarque que le nombre global de paquets émis par les nœuds utilisant les deux protocoles est largement inférieur au nombre de paquets reçus. La quantité de données échangée par DPSM est nettement supérieure à celle échangée par le protocole FDSM. La Figure 6.9 montre que la consommation d'énergie dans la réception est nettement supérieure à la quantité consommée pour l'émission en fonction du nombre de paquets échangés.

6.5.4 Coût global en énergie

Le tableau 6.8 résume les données de consommation d'énergie par nœud des deux métriques sur le calcul et les communications. $E_{Totale} = E_{Calculs} + E_{Tx} + E_{Rx}$.

Les résultats du tableau 6.8 sont représentés graphiquement sur la Figure 6.10 pour un scalaire de 224 bits.

Tableau 6.6 – Coût des communications en énergie (mJ) du protocole DPSM

m	Clé	DPSM					
		Tx			Rx		
		#Paquets	Taille (Byte)	Energie	#Paquets	Taille (Byte)	Energie
3	P-192	39	4644	13.37	116	13898	32.02
	P-224	53	6252	18.01	159	18992	43.75
	P-256	69	8196	23.60	207	24784	57.10
6	P-192	20	2340	6.73	116	13898	32.02
	P-224	27	3144	9.05	159	18992	43.75
	P-256	34	4068	11.71	207	24784	57.10
8	P-192	15	1764	5.08	116	13898	32.02
	P-224	20	2388	6.87	159	18992	43.75
	P-256	26	3108	8.95	207	24784	57.10
12	P-192	10	1188	3.42	116	13898	32.02
	P-224	13	1548	4.45	159	18992	43.75
	P-256	18	2052	5.91	207	24784	57.10

Tableau 6.7 – Coût des communications en énergie (mJ) du protocole FDSM

m	Clé	FDSM					
		Tx			Rx		
		#Paquets	Taille (Byte)	Energie	#Paquets	Taille (Byte)	Energie
3	P-192	39	4642	13.36	79	9396	21.64
	P-224	53	6250	18.00	106	12644	29.13
	P-256	69	8194	23.59	139	16564	38.16
6	P-192	20	2338	6.73	99	11808	27.21
	P-224	27	3142	9.04	133	15860	36.54
	P-256	34	4066	11.71	171	20512	47.26
8	P-192	15	1762	5.07	104	12456	28.69
	P-224	20	2386	6.87	141	16856	38.83
	P-256	26	3106	8.94	183	21928	50.52
12	P-192	10	1186	3.41	110	13176	30.35
	P-224	13	1546	4.45	144	17168	39.55
	P-256	18	2050	5.90	190	22744	52.40

Tableau 6.8 – Coût global en énergie des deux protocoles DPSM et FDSM

		m			
Clé	Protocole	3	6	8	12
P-192	DPSM	120.46	101.51	96.80	92.07
	FDSM	114.80	101.45	99.77	99.85
P-224	DPSM	177.56	151.81	145.03	137.50
	FDSM	166.79	150.91	148.54	145.95
P-256	DPSM	269.83	226.16	216.10	205.03
	FDSM	255.41	225.37	221.58	218.42

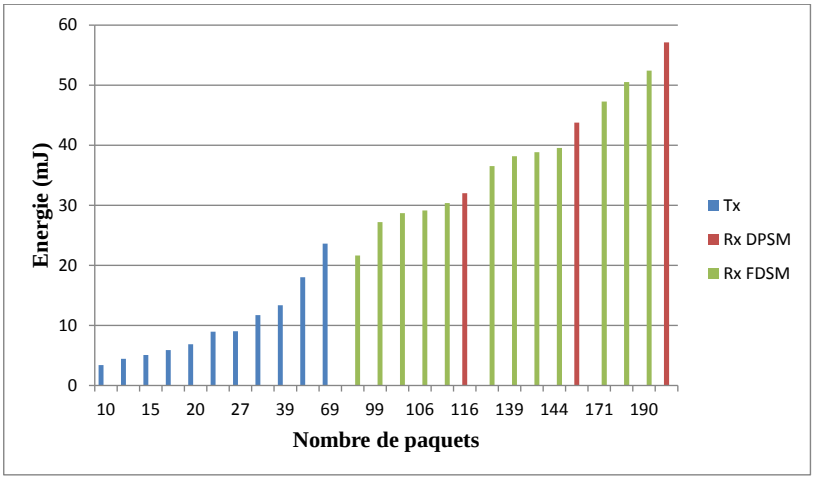

FIGURE 6.9 – Coût en communication des protocoles DPSM et FDSM

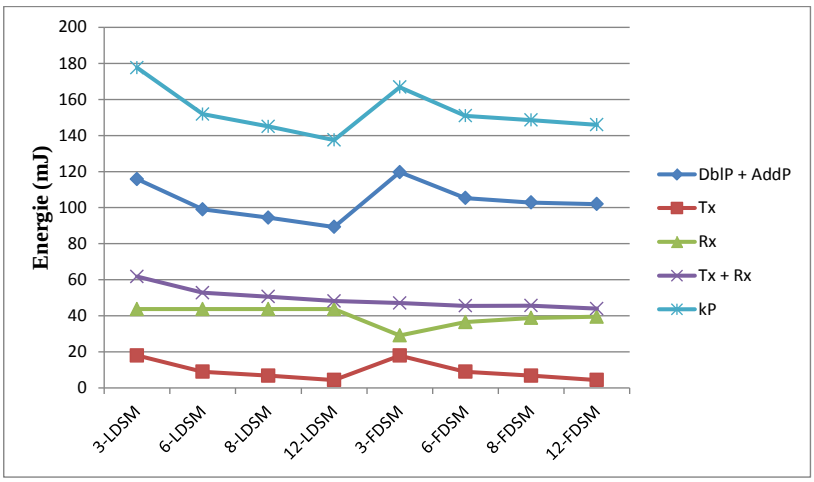

FIGURE 6.10 – Coût global en énergie des protocoles DPSM et FDSM

La Figure 6.10 montre que l'énergie consommée, pour une opération de calcul de multiplication scalaire, par les deux algorithmes DblP et AddP est supérieure à l'énergie des communications Tx et Rx. L'énergie d'une multiplication scalaire kP est calculée par $E_{kP} = E_{Tx} + E_{Rx} + E_{DblP} + E_{AddP}$.

6.5.5 Comparaisons avec d'autres solutions

Dans cette section nous comparons nos résultats aux résultats obtenus pour l'implémentation de trois autres méthodes de calculs d'une multiplication scalaire. La complexité des calculs est donnée par le tableau 6.9. Contrairement à toutes les optimisations faites jusqu'à maintenant sur la réduction de la complexité dans les calculs d'une multiplication scalaire, où c'est le nombre d'additions de points qui est optimisé, notre solution permet d'optimiser le nombre de doublements de points.

TABLEAU 6.9 – Comparaison globale des complexités de calculs des différentes solutions présentées et proposées

	Double-and-Add	**NAF**	**w-NAF**	**DPSM**	**FDSM**
DBL	l	l	$l-1$	$\frac{l}{m}$	$\frac{l}{m}$
ADD	$\frac{l}{2}$	$\frac{l}{3}$	$\frac{l}{w+1} + 2^{w-2}$	$\frac{l}{3}$	$\frac{l}{3} + (m-1)$

Les résultats des comparaisons globales faites sur toutes les solutions proposées et présentées dans ce travail sont représentés graphiquement sur la Figure 6.11.

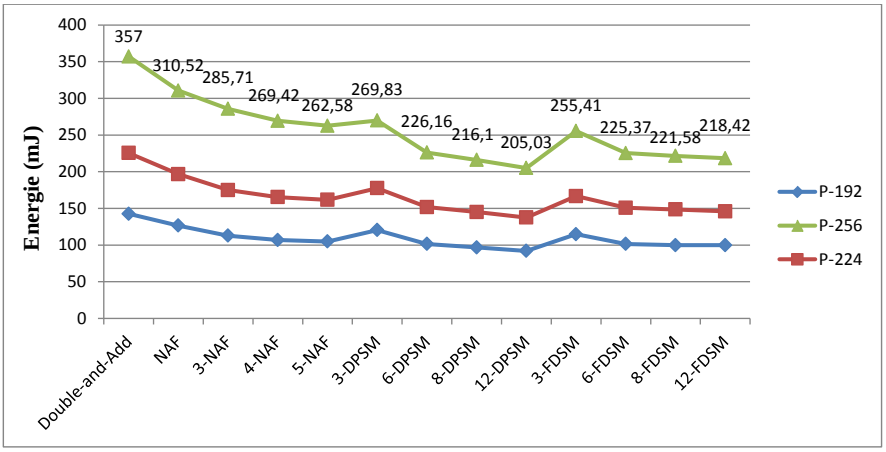

FIGURE 6.11 – Coût global en énergie de toutes les solutions proposées

Les résultats montrent que notre solution DPSM appliquée sur un cluster de 12 nœuds participants donne des résultats meilleurs que toutes les autres solutions implémentées. Ce gain est représenté sur le tableau 6.10.

TABLEAU 6.10 – Gain en energie du protocole 12-DPSM comparé aux autres solutions

Double-and-Add	NAF	3-NAF	4-NAF	5-NAF	12-FDSM
35.47%	27.28%	18.39%	13.81%	12.26%	7.79%

6.6 Conclusion

Dans ce chapitre, nous avons proposé deux protocoles distribués DPSM et FDSM pour le calcul d'une multiplication scalaire. Nos deux solutions distribuées offrent des résultats intéressants avec un temps d'exécution plus rapide et une consommation d'énergie plus faible pour les calculs elliptiques sur un corps fini premier. L'optimisation proposée est basée sur un équilibrage des charges entre les nœuds d'un même cluster afin de réduire la consommation de ressources dans un réseau de capteurs. Par conséquent, les deux nouveaux protocoles de traitement répartis, que nous avons proposé dans ce chaiptre, nommés Distributed and Parallel Scalar point Multiplication (DPSM) et Full Distributed Scalar point Multiplication (FDSM) ont permis d'améliorer significativement les performances des systèmes embarqués et des réseaux de capteurs utilisés dans le déploiement d'un réseau de capteurs.

Le protocole DPSM donne de meilleurs résultats que le protocole FDSM. Le protocole FDSM est recommandé pour les périphériques à faible stockage et le protocole DPSM est recommandé aux systèmes faibles en communications radio. Deux algorithmes sont également proposés pour effectuer indépendamment et en parallèle le doublement et l'addition de points, appelés respectivement DblP et AddP. Notre protocole proposé 12-DPSM s'exécute en un nombre de cycles par seconde de 19,44 MIPS pour une clé P-192 de 192 bits, 29.04 MIPS pour une clé P-224 de 224 bits et 43.3 MIPS pour une clé P-256 de 256 bits. Le protocole DPSM offre une amélioration des performances de 12.26%, 27.28% et 35.47% par rapport respectivement à 5-NAF , NAF et Double-and-Add. Notre implémentation du 12-DPSM nécessite 92.07 mJ en P-192, 137.5 mJ en P-224 et 205.03 mJ en P-256, alors que 5-NAF, considéré comme le mieux comparé aux autres résultats des travaux obtenus dans le cadre de ce travail, nécessite 104.93 mJ en P-192, 161.7 mJ en P-224 et 262.58 en P-256.

Conclusion générale et perspectives

Les réseaux de capteurs sans fil, un cas particulier des réseaux à ressources restreintes, sont de plus en plus utilisés dans des applications sensibles où les communications doivent être protégées, les nœuds du réseau authentifiés et les données échangées disponibles et confidentielles. Certaines caractéristiques des réseaux de capteurs sans fil, comme les limites physiques des nœuds capteurs, les restrictions en consommation d'énergie et le médium de communication sans fil, favorisent des problèmes de sécurité et rendent les nœuds capteurs vulnérables à des attaques et des menaces extérieures par des nœuds malicieux et des entités étrangères au réseau.

Les mécanismes cryptographiques à clé publique sont considérés comme les plus adéquats pour les RcSF en raison du niveau de sécurité assuré. Cependant, la limite en puissance de calcul et en capacité mémoire des nœuds et les restrictions sur l'utilisation de l'énergie, unique source de survie, sont les obstacles de taille pour le développement de tels mécanismes, coûteux en calculs, en mémoire et en consommation d'énergie. Une solution pour satisfaire ces différentes contraintes et assurer en conséquence des communications sécurisées est l'utilisation de la cryptographie des courbes elliptiques.

Les cryptosystèmes basés sur la cryptographie des courbes elliptiques (ECC) utilisent des problèmes mathématiques difficiles pour chiffrer et signer les messages échangés sur le réseau. Ils utilisent deux types de clés : une clé privée pour le chiffrement et une clé publique pour le déchiffrement. La clé privée est dérivée à partir de la clé publique et la régénération de cette clé revient à résoudre un problème de calcul difficilement résoluble selon le niveau de sécurité choisi. La difficulté de résolution de ce problème repose sur ce qu'on appelle les fonctions à sens unique avec trappe. Ces fonctions sont facilement calculables dans un sens et difficilement inversibles si on ne connait pas le secret (trappe). Du point de vue mathématique, ces cryptosystèmes sont censés être les plus sûrs. Cependant, ils peuvent faire objet d'attaques matérielles pour récupérer le secret de la trappe (clé secrète) et ainsi déchiffrer les messages.

Le problème mathématique sur lequel repose la cryptographie des courbes elliptiques est le problème du logarithme discret elliptique. L'opération complexe et centrale de la cryptographie des courbes elliptiques est la multiplication scalaire par un point. C'est une opération qui nécessite des milliers de calculs arithmétiques engendrant une consommation importante de ressources physiques et énergétiques des capteurs. Plusieurs solutions sont proposées pour réduire sa complexité en calculs. Les plus adaptées reposent sur des techniques mathématiques, algorithmiques et de parallélisation. Ainsi, plusieurs algorithmes sont proposés pour optimiser les calculs arithmétiques, réduire le nombre d'opérations de doublements et d'addition de points, accélérer les calculs par l'introduction de nouveaux systèmes de coordonnées, parallélisation des calculs par

la décomposition des données et par décomposition des opérations de doublement et d'additions de points.

L'objectif de cette thèse est d'étudier l'application de la cryptographie des courbes elliptiques aux réseaux de capteurs sans fil. Au cours de ce travail de thèse, nous avons tout d'abord étudié les réseaux à ressources restreintes en général et les réseaux de capteurs en particulier, puis nous avons présenté une étude détaillée sur la cryptographie des courbes elliptiques et les différentes optimisations proposées pour l'accélération des calculs d'une multiplication scalaire. Nous avons par la suite présenté quelques solutions de sécurité économes en consommation énergétiques basées essentiellement sur les techniques de parallélisation de calculs sur plusieurs nœuds d'un réseau sans toutefois échanger le secret de la trappe ou divulguer des informations sur la clé.

En effet, la parallélisation par décomposition des opérations de doublement et d'addition de points peuvent être utilisées pour réduire efficacement la consommation de ressources énergétiques. Dans ce cadre, nous avons présenté plusieurs solutions distribuées et parallèles pour le calcul d'une multiplication scalaire. Le principe étant de distribuer les calculs d'addition de points sur plusieurs nœuds ordinaires du réseau et de confier les calculs de doublements de points à des nœuds calculateurs. Cette technique permet d'éliminer la redondance de calculs de doublements, en calculant une seule fois les doublements d'une multiplication de la clé k par le point générateur P au lieu d'être calculés m fois par les m nœuds du réseau. Elle permet également de réduire la complexité des calculs sur les nœuds ordinaires en effectuant uniquement des additions de points sur des doublements précalculés. Nous avons opté pour cette solution distribuée car nous avons conclu que la réception de l points précalculés consomme moins de ressources que de les calculer.

Nous avons combiné dans un premier temps des techniques mathématiques d'accélération de calculs à des techniques algorithmiques pour le calcul d'une multiplication scalaire. Les algorithmes obtenus sont ensuite implémentés sur une carte Arduino Embedded System. Les résultats de nos implémentations montrent que l'algorithme de Montgomery combiné au système de coordonnées co-Z offre de meilleures performances et permet de réduire le temps de calcul d'environ 15% comparativement à ses deux autres variantes en coordonnées standards et mixtes ainsi que sur les variantes de l'algorithme de Joye.

Dans la deuxième partie de nos contributions, nous avons proposé une nouvelle approche distribuée pour un échange de clé sécurisé dans un cluster formé par 2 et 10 nœuds. Notre solution proposée repose sur trois variantes distribuées des algorithmes Double-and-Add, NAF et signed window algorithm. Les trois variantes sont appelées respectivement Right-to-Left Distributed binary algorithm (RL-Dba), Right-to-Left Distributed Non-Adjacent Form (RL-DNAF) et Right-to-Left Distributed signed window algorithm (RL-Dswa). Les résultats obtenus pour notre approche distribuée sont

très satisfaisants et offrent une meilleure consommation de ressources par rapport aux autres solutions. Nous avons également présenté deux autres protocoles de sécurité, le premier distribué et parallèle nommé Distributed and Parallel Scalar point Multiplication (DPSM) et le second entièrement distribué nommé Full Distributed Scalar point Multiplication (FDSM), pour le calcul d'une multiplication scalaire. Les deux protocoles sont très efficaces en termes de consommation de ressources énergétiques et permettent de réduire la consommation jusqu'à 12% par rapport à la meilleure des autres solutions proposées pour le calcul d'une multiplication scalaire.

Durant tout ce travail de thèse, nous avons opté pour les courbes elliptiques courtes de Weierstrass définies sur un corps fini premier car elles sont adaptées à la cryptographie. Nous avons implémenté toutes nos solutions sur une carte Arduino Uno R3 équipée d'un microcontrôleur ATmega328P très limité en ressources, considéré comme un véritable système embarqué et semblable à la plupart des capteurs utilisés dans la pratique. Les communications sont assurées par des cartes Xbee serie 2, très économes en ressources. Nous concluons que les résultats obtenus sont très satisfaisants et ont permis de réduire considérablement la consommation de ressources notamment énergétiques sur un nœud capteur. De ce fait, bien que la cryptographie des courbes elliptiques demeure un mécanisme cryptographique très compliquée sur des nœuds limités, les différentes optimisations encore possibles sur ces opérations elliptiques comme la multiplication scalaire peuvent faire des ECC le choix le mieux adapté et le plus sûr pour des réseaux à ressources restreintes et les réseaux de capteurs sans fil.

En perspectives, tous nos travaux sont centrés sur la réduction de la complexité des calculs d'une multiplication scalaire et par conséquent la réduction de la consommation de ressources. A cause de la diversité des attaques et les limites sur les ressources des nœuds, il est difficile de proposer une solution globale pour tous les types d'attaques. Nos implémentations peuvent être plus complètes en proposant une solution cryptographique complète. Nous allons également implémenter nos différentes solutions sur d'autres types de courbes comme les courbes de Montgomery et d'Edwards ainsi que sur d'autres corps comme les corps finis binaires. Nous comptons également implémenter nos solutions sur d'autres dispositifs comme FPGA, mais également sur d'autres processeurs matériels.

Annexe A
Paramètres elliptiques

Dans cette section nous spécifierons les paramètres elliptiques utilisés pour nos expérimentations. Rappelons qu'une courbe elliptique est un cas particulier d'une courbe algébrique [79] définie sur un corps K munie d'une addition géométrique sur ses points. Nous avons choisi de travailler sur les courbes non singulières [79] définies sur un corps fini premier, ce type de courbes ont des applications en algorithmique notamment en cryptographie. La cryptographie des courbes elliptiques est un ensemble de primitives cryptographiques qui utilisent les propriétés des courbes elliptiques.

Le choix d'une courbe elliptique dépend de l'application envisagée. Nous suggérons que nos solutions soient destinées à des applications sensibles et stratégiques d'où le choix des paramètres doit garantir une sécurité optimale. Pour cela nous avons choisi les paramètres recommandés par NIST [113]. Les paramètres elliptiques de la courbe définie sur le corps fini premier P sont spécifiés par le sextuple $(p, a, b, G, n, 1)$ et donnés par les tables A.1, A.2 et A.3 pour un scalaire de 192 bits, 224 bits et 256 bits respectivement.

Tableau A.1 – Paramètres recommandés pour un scalaire de 192 bits

Paramètre	Paramètres recommandés pour un scalaire de 192 bits
p	$2^{192} - 2^{64} - 1$
a	-3
b	0x 64210519 E59C80E7 0FA7E9AB 72243049 FEB8DEEC C146B9B1
$G.x$	0x 188DA80E B03090F6 7CBF20EB 43A18800 F4FF0AFD 82FF1012
$G.y$	0x 07192B95 FFC8DA78 631011ED 6B24CDD5 73F977A1 1E794811
n	0x FFFFFFFF FFFFFFFF FFFFFFFF 99DEF836 146BC9B1 B4D22831

Tableau A.2 – Paramètres recommandés pour un scalaire de 224 bits

Paramètre	Paramètres recommandés pour un scalaire de 224 bits
p	$2^{224} - 2^{96} - 1$
a	0x FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE FFFFFFFF FFFFFFFF FFFFFFFE
b	0x B4050A85 0C04B3AB F5413256 5044B0B7 D7BFD8BA 270B3943 2355FFB4
$G.x$	0x B70E0CBD 6BB4BF7F 321390B9 4A03C1D3 56C21122 343280D6 115C1D21
$G.y$	0x BD376388 B5F723FB 4C22DFE6 CD4375A0 5A074764 44D58199 85007E34
n	

Tableau A.3 – Paramètres recommandés pour un scalaire de 256 bits

Paramètre	Paramètres recommandés pour un scalaire de 256 bits
p	$2^{224}(2^{32} - 1) + 2^{192} + 2^{96} - 1$
a	0x FFFFFFFF 00000001 00000000 00000000 00000000 FFFFFFFF FFFFFFFF FFFFFFFC
b	0x 5AC635D8 AA3A93E7 B3EBBD55 769886BC 651D06B0 CC53B0F6 3BCE3C3E 27D2604B
$G.x$	0x 6B17D1F2 E12C4247 F8BCE6E5 63A440F2 77037D81 2DEB33A0 F4A13945 D898C296
$G.y$	0x 4FE342E2 FE1A7F9B 8EE7EB4A 7C0F9E16 2BCE3357 6B315ECE CBB64068 37BF51F5
n	0x FFFFFFFF 00000000 FFFFFFFF FFFFFFFF BCE6FAAD A7179E84 F3B9CAC2 FC632551

Annexe B
Bibliothèque logicielle

Il existe plusieurs librairies fournissant des outils aux systèmes asymétriques basés sur la cryptographie des courbes elliptiques et qui peuvent être configurés de manière flexible et intégrées dans des applications de réseau de capteurs. Parmi ces bibliothèques on a choisi de travailler sur la bibliothèque Micro-ecc (u-ecc) [122], petite et rapide pour les processeurs 8 bits, 32 bits et 64 bits, pour plusieurs raisons, nous citons :

- Résiste aux attaques latérales connues.
- Écrite en C, avec un assemblage en ligne GCC en option pour les plates-formes AVR, ARM et Thumb.
- Prend en charge les architectures 8, 32 et 64 bits.
- Le code produit est de petite taille.
- Aucune allocation de mémoire dynamique n'est nécessaire.
- Prend en charge 5 courbes standards : secp160r1, secp192r1, secp224r1, secp256r1 et secp256k1.
- Licence BSD à 2 clauses.

Annexe C
Architecture matérielle du système embarqué

Les nœuds d'un réseau à ressources restreintes, comme les réseaux de capteurs sans fil, sont des appareils équipés de microcontrôleurs économes en consommation de ressources mais qui sont très faibles en puissance de calcul et en capacité mémoire. Nous avons choisi d'implémenter les algorithmes présentés dans cette thèse sur une plateforme Arduino Embedded System [120]. Les calculs sont opérés par un microcontrôleur ATmega328P ayant les caractéristiques suivantes :

- Fréquence du CPU : 16 Mhz
- Architecture : 8 bits
- Voltage : 5.038 V
- SRAM : 2 KB
- Mémoire Flash : 32 KB

Production scientifique

1. RAMDANI, Mohamed, BENMOHAMMED, Mohamed, et BENBLIDIA, Nadjia. Comparison of scalar point multiplication algorithms in a low resource device. Journal of King Saud University-Computer and Information Sciences, 2019.
2. RAMDANI, Mohamed, BENMOHAMMED, Mohamed, et BENBLIDIA, Nadjia. Distributed solution of scalar multiplication on elliptic curves over F_p for resource-constrained networks. In : Proceedings of the 2nd International Conference on Future Networks and Distributed Systems. ACM, 2018. p. 63.
3. RAMDANI, Mohamed, BENMOHAMMED, Mohamed, et BENBLIDIA, Nadjia. Comparison of Scalar Multiplication Algorithms in a Low Resource Device. In : ICAASE. 2018. p. 70-75.

Bibliographie

[1] Neal Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177) :203–209, 1987. 20, 52, 57, 65

[2] Victor S Miller. Use of elliptic curves in cryptography. In *Conference on the theory and application of cryptographic techniques*, pages 417–426. Springer, 1985. 20, 52, 57, 65

[3] Darrel Hankerson, Alfred J Menezes, and Scott Vanstone. Guide to elliptic curve cryptography. *Computing Reviews*, 46(1) :13, 2005. 20, 61, 72, 74

[4] Hendrik Willem Lenstra et al. Elliptic curves and number-theoretic algorithms, 1986. 20

[5] Hendrik W Lenstra Jr. Factoring integers with elliptic curves. *Annals of mathematics*, pages 649–673, 1987. 20

[6] S Macwan, N Gondaliya, and N Raja. Survey on wireless body area network. *International Journal of Advanced Research in Computer and Communication Engineering*, 5 :107–110, 2016. 26

[7] Matthew Malewski, David MJ Cowell, and Steven Freear. Review of battery powered embedded systems design for mission-critical low-power applications. *International Journal of Electronics*, 105(6) :893–909, 2018. 28

[8] Microchip. Atmega48a/pa/88a/pa/168a/pa/328/p - megaavr® data sheet. http://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061A.pdf, 2018 (accédé le 30 Juillet 2019). 31, 139

[9] Mohamed Amine Ferrag, Mehdi Nafa, and Salim Ghanemi. Security and privacy for routing protocols in mobile ad hoc networks. *Security for Multihop Wireless Networks*, 19, 2014. 31

[10] Mohamed Ali Ayachi. *Contributions à la détection des comportements malhonnêtes dans les réseaux ad hoc AODV par analyse de la confiance implicite*. PhD thesis, Université de Rennes 1, 2011. 31

[11] Al-Sakib Khan Pathan. *Security of self-organizing networks : MANET, WSN, WMN, VANET*. CRC press, 2016. 31

[12] Ali Dorri, Seyed Reza Kamel, and Esmaeil Kheirkhah. Security challenges in mobile ad hoc networks : A survey. *arXiv preprint arXiv :1503.03233*, 2015. 31

[13] Shafiullah Khan, Jonathan Loo, J Mauri, and J Ortiz. Mobile ad hoc networks : Current status and future trends. *CRC PressINC*, 2011. 32

[14] Subir Kumar Sarkar, Tiptur Gangaraju Basavaraju, and C Puttamadappa. *Ad hoc mobile wireless networks : principles, protocols, and applications*. CRC Press, 2016. 32

[15] Muhammad Safdar, Izaz Ahmad Khan, Farman Ullah, Fazlullah Khan, and Syed Roohullah Jan. Comparative study of routing protocols in mobile adhoc networks. *International Journal of Computer Science Trends and Technology*, 4(2) :2347–8578, 2016. 32

[16] Ashish Kumar Jain and Vrinda Tokekar. Mitigating the effects of black hole attacks on aodv routing protocol in mobile ad hoc networks. In *2015 International Conference on Pervasive Computing (ICPC)*, pages 1–6. IEEE, 2015. 32

[17] Tie Qiu, Ning Chen, Keqiu Li, Daji Qiao, and Zhangjie Fu. Heterogeneous ad hoc networks : Architectures, advances and challenges. *Ad Hoc Networks*, 55 :143–152, 2017. 32

[18] Institut de l'audiovisuel et des télécommunications en Europe (France). *Internet of Things : A Key Pillar of Digital Transformation*. IDATE, 2015. 32

[19] Mahdi H Miraz, Maaruf Ali, Peter S Excell, and Rich Picking. A review on internet of things (iot), internet of everything (ioe) and internet of nano things (iont). In *2015 Internet Technologies and Applications (ITA)*, pages 219–224. IEEE, 2015. 32

[20] Shancang Li, Li Da Xu, and Shanshan Zhao. The internet of things : a survey. *Information Systems Frontiers*, 17(2) :243–259, 2015. 32

[21] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of things : A survey on enabling technologies, protocols, and applications. *IEEE communications surveys & tutorials*, 17(4) :2347–2376, 2015. 32

[22] Partha Pratim Ray. A survey on internet of things architectures. *Journal of King Saud University-Computer and Information Sciences*, 30(3) :291–319, 2018. 32

[23] Vehbi C Gungor and Gerhard P Hancke. Industrial wireless sensor networks : Challenges, design principles, and technical approaches. *IEEE Transactions on industrial electronics*, 56(10) :4258–4265, 2009. 33

[24] Kay Soon Low, Win Nu Nu Win, and Meng Joo Er. Wireless sensor networks for industrial environments. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 2, pages 271–276. IEEE, 2005. 33

[25] Mohamed Ramdani. Problèmes de sécurité dans les réseaux de capteurs avec prise en charge de l'énergie, mémoire de magister, université de blida, 2013. 33

[26] Sahabul Alam and Debashis De. Analysis of security threats in wireless sensor network. *arXiv preprint arXiv :1406.0298*, 2014. 34

[27] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks : a survey. *Computer networks*, 38(4) :393–422, 2002. 34, 39

[28] Priyanka Rawat, Kamal Deep Singh, Hakima Chaouchi, and Jean Marie Bonnin. Wireless sensor networks : a survey on recent developments and potential synergies. *The Journal of supercomputing*, 68(1) :1–48, 2014. 34

[29] Luís ML Oliveira and Joel JPC Rodrigues. Wireless sensor networks : A survey on environmental monitoring. *JCM*, 6(2) :143–151, 2011. 34, 44

[30] Libelium. Smart parking, technical guide. http://www.libelium.com/uploads/2013/02/smart-parking-sensor-board_eng.pdf, 2013 (accédé le 30 Juillet 2019). 34

[31] Sean Bonner Azby Brown, Pieter Franken. The safecast report. http://safecast.org/downloads/safecastreport2015.pdf, 2015 (accédé le 30 Juillet 2019). 34

[32] Ian F Akyildiz, Tommaso Melodia, and Kaushik R Chowdhury. A survey on wireless multimedia sensor networks. *Computer networks*, 51(4) :921–960, 2007. 35

[33] Ian F Akyildiz, Tommaso Melodia, and Kaushik R Chowdhury. Wireless multimedia sensor networks : Applications and testbeds. *Proceedings of the IEEE*, 96(10) :1588–1605, 2008. 35

[34] Muhammad Usman, Mian Ahmad Jan, Xiangjian He, and Priyadarsi Nanda. Data sharing in secure multimedia wireless sensor networks. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 590–597. IEEE, 2016. 35

[35] Samaneh Movassaghi, Mehran Abolhasan, Justin Lipman, David Smith, and Abbas Jamalipour. Wireless body area networks : A survey. *IEEE Communications surveys & tutorials*, 16(3) :1658–1686, 2014. 35

[36] Mohammad Ghamari, Balazs Janko, R Sherratt, William Harwin, Robert Piechockic, and Cinna Soltanpur. A survey on wireless body area networks for ehealthcare systems in residential environments. *Sensors*, 16(6) :831, 2016. 35

[37] David M Davenport, Budhaditya Deb, and Fergus J Ross. Wireless propagation and coexistence of medical body sensor networks for ambulatory patient monitoring. In *2009 Sixth International Workshop on Wearable and Implantable Body Sensor Networks*, pages 41–45. IEEE, 2009. 37

[38] Jennifer Yick, Biswanath Mukherjee, and Dipak Ghosal. Wireless sensor network survey. *Computer networks*, 52(12) :2292–2330, 2008. 40

[39] Michel Robert. *Capteurs intelligents et méthodologie d'évaluation*. Hermès Paris, 1993. 40

[40] Kamal Beydoun. *Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs*. PhD thesis, Université de Franche-Comte, 2009. 41

[41] LAN/MAN Standards Committee et al. Part 15.4 : wireless medium access control (mac) and physical layer (phy) specifications for low-rate wireless personal area networks (lr-wpans). *IEEE Computer Society*, 2003. 41

[42] Sinem Coleri Ergen. Zigbee/ieee 802.15. 4 summary. *UC Berkeley, September*, 10 :17, 2004. 42

[43] André Cunha, Anis Koubaa, Ricardo Severino, and Mário Alves. Open-zb : an open-source implementation of the ieee 802.15. 4/zigbee protocol stack on tinyos. In *2007 IEEE International Conference on Mobile Adhoc and Sensor Systems*, pages 1–12. IEEE, 2007. 42

[44] Nisha Ashok Somani and Yask Patel. Zigbee : A low power wireless technology for industrial applications. *International Journal of Control Theory and Computer Modelling (IJCTCM)*, 2(3) :27–33, 2012. 42

[45] Etimad Fadel, Vehbi C Gungor, Laila Nassef, Nadine Akkari, MG Abbas Malik, Suleiman Almasri, and Ian F Akyildiz. A survey on wireless sensor networks for smart grid. *Computer Communications*, 71 :22–33, 2015. 44

[46] Mustafa Kocakulak and Ismail Butun. An overview of wireless sensor networks towards internet of things. In *2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 1–6. IEEE, 2017. 44

[47] Milica Pejanović Đurišić, Zhilbert Tafa, Goran Dimić, and Veljko Milutinović. A survey of military applications of wireless sensor networks. In *2012 Mediterranean conference on embedded computing (MECO)*, pages 196–199. IEEE, 2012. 44

[48] Moshaddique Al Ameen, Jingwei Liu, and Kyungsup Kwak. Security and privacy issues in wireless sensor networks for healthcare applications. *Journal of medical systems*, 36(1) :93–101, 2012. 44

[49] Hande Alemdar and Cem Ersoy. Wireless sensor networks for healthcare : A survey. *Computer networks*, 54(15) :2688–2710, 2010. 44

[50] Dr G Padmavathi, Mrs Shanmugapriya, et al. A survey of attacks, security mechanisms and challenges in wireless sensor networks. *arXiv preprint arXiv :0909.0576*, 2009. 46

[51] Al-Sakib Khan Pathan and Choong Seon Hong. Security attacks and challenges in wireless sensor networks. In *Encyclopedia on ad hoc and ubiquitous computing : theory and design of wireless ad hoc, sensor, and mesh networks*, pages 397–425. World Scientific, 2010. 46

[52] David R Raymond and Scott F Midkiff. Denial-of-service in wireless sensor networks : Attacks and defenses. *IEEE Pervasive Computing*, 1(1) :74–81, 2008. 46

[53] Abdul Aziz Mugheri, Murtaza Ahmed Siddiqui, and Mohammad Khoso. Analysis on security methods of wireless sensor network (wsn). *Sukkur IBA Journal of Computing and Mathematical Sciences*, 2(1) :52–60, 2018. 47

[54] Al-Sakib Khan Pathan, Hyung-Woo Lee, and Choong Seon Hong. Security in wireless sensor networks : issues and challenges. In *2006 8th International Conference Advanced Communication Technology*, volume 2, pages 6–pp. IEEE, 2006. 47

[55] Hanane Kalkha, Hassan Satori, and Khalid Satori. Preventing black hole attack in wireless sensor network using hmm. *Procedia computer science*, 148 :552–561, 2019. 47

[56] Benjamin Jack Culpepper and H Chris Tseng. Sinkhole intrusion indicators in dsr manets. In *First International Conference on Broadband Networks*, pages 681–688. IEEE, 2004. 47

[57] Yih-Chun Hu, Adrian Perrig, and David B Johnson. Packet leashes : A defense against wormhole attacks in wireless ad hoc networks. In *Proceedings of INFOCOM*, volume 2003, 2003. 47

[58] James Newsome, Elaine Shi, Dawn Song, and Adrian Perrig. The sybil attack in sensor networks : analysis & defenses. In *Third international symposium on information processing in sensor networks, 2004. IPSN 2004*, pages 259–268. IEEE, 2004. 47

[59] Hong-Ning Dai, Qiu Wang, Dong Li, and Raymond Chi-Wing Wong. On eavesdropping attacks in wireless sensor networks with directional antennas. *International Journal of Distributed Sensor Networks*, 9(8) :760834, 2013. 47

[60] Loren J Rittle, Timothy J Spets, Guy G Romano, and Kenneth T Crisler. Method and system for data packet collision avoidance in a wireless communication system, July 20 2004. US Patent 6,765,882. 47

[61] Michael Brownfield, Yatharth Gupta, and Nathaniel Davis. Wireless sensor network denial of sleep attack. In *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pages 356–364. IEEE, 2005. 47

[62] Hemanta Kumar Kalita and Avijit Kar. Wireless sensor network security analysis. *International Journal of Next-Generation Networks (IJNGN)*, 1(1) :1–10, 2009. 47

[63] K Venkatraman, J Vijay Daniel, and G Murugaboopathi. Various attacks in wireless sensor network : Survey. *International Journal of Soft Computing and Engineering (IJSCE)*, 3(1) :208–212, 2013. 47

[64] Michael Healy, Thomas Newe, and Elfed Lewis. Security for wireless sensor networks : A review. In *2009 IEEE Sensors Applications Symposium*, pages 80–85. IEEE, 2009. 48

[65] Song Han, Elizabeth Chang, Li Gao, and Tharam Dillon. Taxonomy of attacks on wireless sensor networks. In *EC2ND 2005*, pages 97–105. Springer, 2006. 48

[66] Randall K Nichols, Panos Lekkas, and Panos C Lekkas. *Wireless security*. McGraw-Hill Professional Publishing, 2001. 48

[67] Chakib Bekara and Maryline Laurent-Maknavicius. A new resilient key management protocol for wireless sensor networks. In *IFIP International Workshop on Information Security Theory and Practices*, pages 14–26. Springer, 2007. 48, 49

[68] Joan Daemen and Vincent Rijmen. Announcing the advanced encryption standard (aes). *Federal Information Processing Standards Publication*, 197, 2001. 52

[69] Alex Biryukov and Christophe De Cannière. *Data encryption standard (DES)*, pages 129–135. Springer US, 2005. 52

[70] Scott Fluhrer, Itsik Mantin, and Adi Shamir. Weaknesses in the key scheduling algorithm of rc4. In *International Workshop on Selected Areas in Cryptography*, pages 1–24. Springer, 2001. 52

[71] Ronald L Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2) :120–126, 1978. 52, 57

[72] Tifenn Rault, Abdelmadjid Bouabdallah, and Yacine Challal. Energy efficiency in wireless sensor networks : A top-down survey. *Computer Networks*, 67 :104–122, 2014. 54

[73] Najmeh Kamyab Pour. Energy efficiency in wireless sensor networks. *arXiv preprint arXiv :1605.02393*, 2016. 54

[74] Yore Sankarasubramaniam, Ian F Akyildiz, and SW McLaughlin. Energy efficiency based packet size optimization in wireless sensor networks. In *Proceedings of the First IEEE International Workshop on Sensor Network Protocols and Applications, 2003.*, pages 1–8. IEEE, 2003. 54

[75] Raja Jurdak, Antonio G Ruzzelli, and Gregory MP O'Hare. Radio sleep mode optimization in wireless sensor networks. *IEEE Transactions on Mobile Computing*, 9(7) :955–968, 2010. 54

[76] Brian Hinman, David Gurevich, Miika Keskinen, Vishakan Ponnampalam, and Chiu Ngok Eric Wong. Channel optimization in half duplex communications systems, April 7 2015. US Patent 9,001,689. 54

[77] Nikolaos A Pantazis, Stefanos A Nikolidakis, and Dimitrios D Vergados. Energy-efficient routing protocols in wireless sensor networks : A survey. *IEEE Communications surveys & tutorials*, 15(2) :551–591, 2012. 54

[78] Dindayal Mahto, Danish Ali Khan, and Dilip Kumar Yadav. Security analysis of elliptic curve cryptography and rsa. In *Proceedings of the World Congress on Engineering*, volume 1, pages 419–422, 2016. 57, 65, 95

[79] Robin Hartshorne. *Algebraic geometry*, volume 52. Springer Science & Business Media, 2013. 57, 151

[80] Joseph H Silverman. *The arithmetic of elliptic curves*, volume 106. Springer Science & Business Media, 2009. 57, 60

[81] Charles J Mozzochi. *The Fermat Diary*. American Mathematical Soc., 2000. 57

[82] Nicolas Meloni. New point addition formulae for ecc applications. In *International Workshop on the Arithmetic of Finite Fields*, pages 189–201. Springer, 2007. 64, 77, 78, 100

[83] MK Senthilkumar and U Senthilkumaran. Review of asymmetric key cryptography in wireless sensor networks. *International Journal of Engineering and Technology*, 8(2) :859–862, 2016. 66

[84] Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6) :644–654, 1976. 67

[85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, 31(4) :469–472, 1985. 68, 69

[86] Matthieu Rivain. Fast and regular algorithms for scalar multiplication over elliptic curves. *IACR Cryptology ePrint Archive*, 2011 :338, 2011. 71, 83, 102, 103, 116

[87] Rita Mayer-Sommer. Smartly analyzing the simplicity and the power of simple power analysis on smartcards. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 78–92. Springer, 2000. 71, 72, 73, 74

[88] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In *Annual International Cryptology Conference*, pages 388–397. Springer, 1999. 71, 72, 73, 74

[89] Jean-Marc Robert. *Contrer l'attaque Simple Power Analysis efficacement dans les applications de la cryptographie asymétrique, algorithmes et implantations*. PhD thesis, Perpignan, 2015. 74, 76

[90] Jérémy Métairie. *Contribution aux opérateurs arithmétiques GF (2m) et leurs applications à la cryptographie sur courbes elliptiques*. PhD thesis, Université de Rennes1, 2016. 74

[91] Jean-Sébastien Coron. Resistance against differential power analysis for elliptic curve cryptosystems. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 292–302. Springer, 1999. 74

[92] Peter L Montgomery. Speeding the pollard and elliptic curve methods of factorization. *Mathematics of computation*, 48(177) :243–264, 1987. 75, 85

[93] Marc Joye. Highly regular right-to-left algorithms for scalar multiplication. In *International Workshop on Cryptographic Hardware and Embedded Systems*, pages 135–147. Springer, 2007. 76, 85

[94] Benoît Chevallier-Mames, Mathieu Ciet, and Marc Joye. Low-cost solutions for preventing simple side-channel analysis : Side-channel atomicity. *IEEE Transactions on computers*, 53(6) :760–768, 2004. 76

[95] Andrew Byrne, Nicolas Meloni, Francis Crowe, William P Marnane, Arnaud Tisserand, and Emanuel M Popovici. Spa resistant elliptic curve cryptosystem using addition chains. In *Fourth International Conference on Information Technology (ITNG'07)*, pages 995–1000. IEEE, 2007. 76

[96] PUB Fips. 186-2. digital signature standard (dss). *National Institute of Standards and Technology (NIST)*, 20 :13, 2000. 76

[97] Peter L Montgomery. Modular multiplication without trial division. *Mathematics of computation*, 44(170) :519–521, 1985. 77

[98] Simon Pontie. *Sécurisation matérielle pour la cryptographie à base de courbes elliptiques*. PhD thesis, Université Grenoble Alpes, 2016. 77

[99] Thomas Chabrier. *Arithmetic recodings for ECC cryptoprocessors with protections against side-channel attacks*. PhD thesis, Université de Rennes 1, 2013. 81

[100] François Morain and Jorge Olivos. Speeding up the computations on an elliptic curve using addition-subtraction chains. *RAIRO-Theoretical Informatics and Applications*, 24(6) :531–543, 1990. 81, 82, 105

[101] Bodo Möller. Improved techniques for fast exponentiation. In *International Conference on Information Security and Cryptology*, pages 298–312. Springer, 2002. 83

[102] Kwang Ho Kim, Junyop Choe, Song Yun Kim, Namsu Kim, and Sekung Hong. Speeding up elliptic curve scalar multiplication without precomputation. *IACR Cryptology ePrint Archive*, 2017 :669, 2017. 85

[103] Thomas Izard. *Opérateurs arithmétiques parallèles pour la cryptographie asymétrique*. PhD thesis, Université de Montpellier 2, 2011. 87

[104] Henri Cohen, Atsuko Miyaji, and Takatoshi Ono. Efficient elliptic curve exponentiation using mixed coordinates. In *International Conference on the Theory*

and Application of Cryptology and Information Security, pages 51–65. Springer, 1998. 87

[105] Yasuyuki Sakai and Kouichi Sakurai. Efficient scalar multiplications on elliptic curves without repeated doublings and their practical performance. In *Australasian Conference on Information Security and Privacy*, pages 59–73. Springer, 2000. 88

[106] Wieland Fischer, Christophe Giraud, Erik Woodward Knudsen, and Jean-Pierre Seifert. Parallel scalar multiplication on general elliptic curves over fp hedged against non-differential side-channel attacks. *IACR Cryptology ePrint Archive*, 2002 :7, 2002. 88

[107] Yanbo Shou, Herve Guyennet, and Mohamed Lehsaini. Parallel scalar multiplication on elliptic curves in wireless sensor networks. In *International Conference on Distributed Computing and Networking*, pages 300–314. Springer, 2013. 90, 133

[108] Mohamed Ramdani, Mohamed Benmohammed, and Nadjia Benblidia. Comparison of scalar point multiplication algorithms in a low resource device. *Journal of King Saud University-Computer and Information Sciences*, 2019. 96, 110

[109] Tetsuya Izu, Bodo Möller, and Tsuyoshi Takagi. Improved elliptic curve multiplication methods resistant against side channel attacks. In *International Conference on Cryptology in India*, pages 296–313. Springer, 2002. 97

[110] Pritam Gajkumar Shah. *Australian Journal of Wireless Technologies, Mobility and Security*. Ganesh Traders., 2012. 99

[111] JF Christmann, E Beigné, C Condemine, and J Willemin. An innovative and efficient energy harvesting platform architecture for autonomous microsystems. In *Proceedings of the 8th IEEE International NEWCAS Conference 2010*, pages 173–176. IEEE, 2010. 112, 115

[112] Mohamed Ramdani, Mohamed Benmohammed, and Nadjia Benblidia. Distributed solution of scalar multiplication on elliptic curves over fp for resource-constrained networks. In *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, page 63. ACM, 2018. 112

[113] Certicom Research. Standards for efficient cryptography 2 (sec 2), 2010. 120, 139, 151

[114] Ali Mohammed AH Al-Kuwari, Cesar Ortega-Sanchez, Atif Sharif, and Vidyasagar Potdar. User friendly smart home infrastructure : Beehouse. In *5th IEEE International Conference on Digital Ecosystems and Technologies (IEEE DEST 2011)*, pages 257–262. IEEE, 2011. 120

[115] Julien Bayle. *C programming for Arduino*. Packt Publishing Ltd, 2013. 120

[116] Charles Bell. Tiny talking modules : an introduction to xbee wireless modules. In *Beginning sensor networks with arduino and raspberry pi*, pages 19–50. Springer, 2013. 121

[117] Ravi Kishore Kodali, Kashyapkumar H Patel, and Narasimha Sarma. Energy efficient elliptic curve point multiplication for wsn applications. In *2013 National Conference on Communications (NCC)*, pages 1–5. IEEE, 2013. 124

[118] Xu Huang, Pritam Shah, and Dharmendra Sharma. Fast algorithm in ecc for wireless sensor network. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 2, pages 17–19, 2010. 124

[119] Christophe Negre and Jean-Marc Robert. New parallel approaches for scalar multiplication in elliptic curve over fields of small characteristic. *IEEE Transactions on Computers*, 64(10) :2875–2890, 2015. 130

[120] Steven F Barrett. Arduino microcontroller processing for everyone ! *Synthesis Lectures on Digital Circuits and Systems*, 8(4) :1–513, 2013. 139, 154

[121] Texas Instruments. Chipcon products from texas instruments cc2530 datasheet, 2009. 139, 143

[122] Ken Mackay. micro-ecc–ecdh and ecdsa for 8-bit, 32-bit, and 64-bit processors, 2016. 139, 153