Université Abou Bekr Belkaid
Tlemcen Algérie
جامعة أبي بكر بلقايد
تلمسان الجزائر

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Laboratoire STIC

THESE

Présentée

A L'UNIVERSITE DE TLEMCEN
FACULTE DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE ET ELECTRONIQUE

Pour l'obtention du diplôme de

DOCTORAT

Spécialité : " Systèmes et Réseaux de Télécommunications"

Par

MERAD BOUDIA Omar Rafik

AGREGATION DES DONNEES ET SECURITE DES RESEAUX DE CAPTEURS SANS FIL

Soutenue le Lundi 26 Mai 2014 devant le Jury:

KAMECHE Samir	MCA, Université de Tlemcen	Président
SENOUCI Sidi Mohammed	Pr, Université de Bourgogne, France	Examinateur
ALIPACHA Adda	Pr, Université UST-Oran	Examinateur
MHAMED Abdallah	MC-HDR, Institut de Télécom Sud-paris	Examinateur
LEHSAINI Mohamed	MCA, Université de Tlemcen	Examinateur
FEHAM Mohammed	Pr, Université de Tlemcen	Directeur de Thèse

« Les machines un jour pourront résoudre tous les problèmes,
mais jamais aucune d'entre elles ne pourra en poser un ! »
- Albert Einstein

A mon cher père Mohammed
Et ma chère mère Fatima Zohra
Pour l'éducation et le grand amour dont ils m'ont entouré depuis ma naissance.
Et pour leurs patiences et leurs sacrifices.

A mon cher frère Abdelilleh ;
A ma chère sœur Cherifa et son mari Abdelghani ;
A mon cher neveu Rachad ;
A tous ceux que j'aime.

Je dédie ce travail

Omar

REMERCIEMENTS

Grâce à Allah vers lequel vont toutes les louanges, ce travail s'est accompli.

Avant tout, j'adresse mes sincères remerciements à mon directeur de thèse le professeur FEHAM Mohammed pour son soutien tout au long de ces années de thèse. Je ne saurais dire combien nos échanges et ses nombreux conseils m'ont été précieux. Sa disponibilité, ses qualités pédagogiques et humaines, ses compétences et sa bonne humeur m'ont apporté un encadrement déterminant dans toutes les phases de ce travail.

Je tiens également à remercier le professeur SENOUCI Sidi Mohammed pour avoir su me guider avec attention et une gentillesse constante pendant mon stage en France à l'université de Bourgogne. Sa qualité scientifique et humaine, son encouragement et ses remarques ont largement contribué à l'aboutissement de cette thèse.

Je remercie Mr KAMECHE Samir pour avoir accepté de présider le jury de ma thèse, je tiens à remercier aussi les membres du jury, Mr SENOUCI Sidi Mohammed, Mr ALIPACHA Adda, Mr MHAMED Abdallah et Mr LEHSAINI Mohamed, pour l'honneur qu'ils m'ont fait en acceptant de juger ce travail.

Je souhaite également remercier les membres du laboratoire Systèmes et Technologie de l'Information et de la Communication (STIC) au sein duquel ce travail a été réalisé, en particulier Mme LABRAOUI Nabila, ses remarques et conseils, tous très constructifs, m'ont beaucoup aidé.

Je souhaite témoigner ma gratitude et vifs remerciements envers tous ceux qui ont contribué, de prés ou de loin à la réalisation de ce travail.

Bien sûr, je ne peux terminer sans remercier mes proches de tout cœur et notamment mes parents qui, au cours de ces années de thèse, m'ont toujours soutenu et encouragé.

Résumé

Les réseaux de capteurs sont des réseaux sans fil constitués d'un ensemble de dispositifs ayant des ressources informatiques limitées. Ce type de réseaux a attiré beaucoup d'attentions ces dernières années, non seulement dans le milieu universitaire mais aussi dans l'industrie, pour l'étude et le développement de plusieurs applications potentielles. Cependant, la contrainte des ressources représente la caractéristique la plus importante de ce réseau. En effet, les capteurs sans fil sont limités en termes de calcul, stockage, batterie, etc. Par conséquent, une solution qui vise à conserver ces ressources est largement souhaitée.

L'agrégation des données est une technique très utilisée car elle permet de réduire le nombre de message transmis dans le réseau et par conséquent réduire la consommation d'énergie, ainsi améliorer la durée de vie du réseau. Cependant, Les réseaux de capteurs sont généralement déployés dans des environnements, souvent très hostiles et sans assistance. Par conséquent, le concepteur devrait, non seulement considérer les menaces de sécurité pouvant survenir dans un réseau facilement accessible à l'attaquant, mais aussi faire face à des ressources informatiques très limitées.

Dans cette thèse, nous nous sommes intéressés aux problèmes de sécurité liés à l'agrégation des données dans les réseaux de capteurs sans fil. Nous avons proposé deux nouvelles approches ainsi qu'une implémentation efficace et sécurisée du chiffrement basé sur les courbes elliptiques. Notre première approche traite les problèmes des usurpations d'identités et le rejet total des données. Alors que la seconde fournit les services de sécurité de bout en bout. Les résultats présentés dans cette thèse sont issus non seulement de plusieurs simulations mais aussi d'expérimentations en implémentant nos algorithmes sur un réseau de capteur réel.

Mots clés : Réseaux de capteurs sans fil, agrégation des données, sécurité, courbes elliptiques.

Abstract

Sensor networks are wireless networks consisting of a set of small and resources-constrained devices. Wireless sensor networks have received much attention over the last few years, not only in academia, but also in industries for the study and development of a plethora of potential applications. However, the resource constraint is the most important feature of such network. In fact, the sensors are constrained in battery power, communication, and computation capability; therefore, every possible solution that aims to conserve these resources is extensively sought.

Data aggregation is one of the techniques that is actually considered as an essential paradigm for sensor networks since it tends to save computation and communication resources. Data aggregation allows in-network processing which leads to lesser packet transmissions and reduces redundancy, and therefore, helps in increasing the network's overall lifetime. However, sensor networks are usually deployed in unattended and hostile environments. Therefore, the designer should not only consider the security threats that can occur in an easily accessible network to the attacker, but also cope with the limited resources.

In this thesis, we are interested in security problems related to data aggregation in wireless sensor networks. We proposed two new approaches and a fast and secure implementation of Elliptic Curve based encryption. Our first approach addresses the problems of identity usurpation and the total data rejection, while the second provides security services in an end-to-end manner. The results presented in this thesis come not only from several simulations but also from real experiments by implementing our algorithms on a real sensor network.

Keywords: Wireless sensor networks, data aggregation, security, elliptic curve.

Sommaire

Introduction générale **6**

Chapitre 1 : Les réseaux de capteurs sans fil **9**

1. Introduction 9
2. Un réseau de capteurs sans fil 9
3. Le capteur sans fil 10
   3.1. Unité de captage 10
   3.2. Unité de traitement 10
   3.3. Unité de transmission 10
   3.4. Unité d'énergie 10
4. Les domaines d'applications des RCSFs 13
   4.1. Surveillance de l'habitat écologique 14
   4.2. Surveillance militaire et poursuite de cibles 14
   4.3. Surveillance structurelle et sismique 14
   4.4. Les réseaux de détection industriels et commerciaux 15
   4.5. Les applications médicales 15
5. Les challenges liés à la conception des RCSFs 15
   5.1. La durée de vie 15
   5.2. La réactivité 16
   5.3. La robustesse 16
   5.4. La scalabilité 16
   5.5. L'auto-configuration 16
   5.6. L'auto-optimisation et adaptation 17
   5.7. La sécurité 17
6. Agrégation des données et sécurité dans les RCSFs 17
   6.1. L'agrégation des données 17
   6.2. Les exigences de sécurité dans les RCSFs 19
   6.3. Les attaques contre les RCSFs 20
   6.4. La sécurité et l'agrégation dans les RCSFs 23
7. Conclusion 24

Chapitre 2 : La sécurité des données agrégées dans les RCSFs **25**

1. Introduction 25
2. La sécurité des données agrégées 25

2.1. Les solutions préventives 27
2.2. Les solutions curatives 39
3. Discussion 41
3.1. Sécurité 41
3.2. Performances 44
4. Conclusion 46
Chapitre 3 : Première contribution: La solution RSAED 48
1. Introduction 48
2. Motivations 48
3. Spécifications générales 49
3.1. Modèle du réseau 49
3.2. Modèle de sécurité 49
3.3. Modèle d'attaque 51
3.4. Objectifs de conception 51
4. Notre Approche: RSAED 51
4.1. Vue d'ensemble 52
4.2. Détail de l'approche 52
5. Analyse 53
5.1. Sécurité 53
5.2. Performances 54
5.3. Comparaison avec quelques solutions de la littérature 59
6. Conclusion 60
Chapitre 4 : Deuxième contribution: Implémentation efficace et sécurisée de l'algorithme ECEG pour une sécurité de bout en bout des RCSFs 62
1. Introduction 62
2. L'agrégation des données et la confidentialité 62
3. Cryptographie sur les courbes elliptiques 64
3.1. Définition 64
3.2. Systèmes de coordonnées 65
3.3. Les attaques par canaux cachés sur l'ECC 65
3.4. La Multiplication des Points par un Scalaire MPS 66
3.5. La décompression des points 69
3.6. TinyECC 70
4. Notre implémentation de l'ECEG pour les RCSFs 70

4.1. Implémentation de la multiplication des points 71
4.2. Implémentation de la décompression des points 73
5. Résultats et discussions 74
5.1. Performance de la MPS et l'ADP 74
5.2. Performances de l'ECEG pour les RCSFs 76
5.3. Comparaison avec l'implémentation d'Ugus et al. [59] 77
6. Résultats pour RSAED 78
7. Conclusion 78
Chapitre 5: Troisième contribution: Une nouvelle solution pour sécuriser les données agrégées en utilisant la cryptographie à clé publique avec état 80
1. Introduction 80
2. Motivations 80
3. Spécifications générales 81
3.1. Le chiffrement à clé publique avec état 81
3.2. Modèle du réseau 82
3.3. Modèle de sécurité 83
3.4. Modèle d'attaque 84
3.5. Objectifs de conception 84
4. Solution SASPKC 84
5. Analyse 88
5.1. Sécurité 88
5.2. Evaluation des performances 90
5.3. Comparaison avec d'autres solutions 98
6. Conclusion 99
Conclusion générale et perspectives 100
Liste des publications 103
Références 104

Liste des tableaux :

Tableau 1. Modèles de capteurs sans fil 11
Tableau 2. Comparaison des tailles des clés en bits 27
Tableau 3. Comparaison entre les C.P.H 35
Tableau 4. Services de sécurité fournis 41
Tableau 5. Vulnérabilité aux attaques 43
Tableau 6. Comparaison des performances 44
Tableau 7. Charges de transmission 45
Tableau 8. Temps d'exécution de nos opérations sur MicaZ 55
Tableau 9. Résultats sur TelosB 58
Tableau 10. Comparaison entre les solutions 59
Tableau 11. Coût des opérations 65
Tableau 12. Systèmes de coordonnée Vs Nombre de bits 69
Tableau 13. Implémentations de l'ECC dans les RCSFs 72
Tableau 14. Comparaison avec les implémentations précédentes 74
Tableau 15. Performances pour SECP160R1 76
Tableau 16. Performances de ECEG.Encrypt() 77
Tableau 17. Performances de ECEG.hom_add() 77
Tableau 18. Comparaison avec Ugus et al. [59] 78
Tableau 19. Résultats de RSAED 78
Tableau 20. Notations et leurs définitions 83
Tableau 21. Temps d'exécution des opérations 91
Tableau 22. Energie estimée pour nos opérations 95
Tableau 23. Energie totale consommée par les nouds CH et Non-CH dans un réseau de 100 nœuds 97
Tableau 24. Nombre L Vs λ 98
Tableau 25. Comparaison des implémentations 98
Tableau 26. Comparaison entre les solutions 99

Liste des figures :

Figure 1. Architecture d'un réseau de capteurs sans fil 9
Figure 2. Composants d'un capteur sans fil 10
Figure 3. Chronologie des capteurs sans fil 11
Figure 4. Pile protocolaire 12
Figure 5. Efficacité de l'agrégation des données 18
Figure 6. Agrégation basée sur les arbres 18
Figure 7. Agrégation basée sur les clusters 19
Figure 8. Classifications des solutions 26
Figure 9. Arbre de hachage de Merkle 28
Figure 10. Modèle réseau de la solution de Du et al. [48] 30
Figure 11. Modèle du RCSF considéré dans [78] 39
Figure 12. Charges totales de transmission 45
Figure 13. Energies totales de transmission 45
Figure 14. Modèle du réseau 49
Figure 15. Comparaison entre les algorithmes basés sur l'ECC [58] 50
Figure 16. Charge de calcul Vs Nombre de nœuds 56
Figure 17. Délai de bout en bout Vs Nombre de nœuds 56
Figure 18. Nombre de parquets reçus Vs nombre de nœuds 57
Figure 19. Charge de communication de RSAED et Albath et al. [73] 58
Figure 20. Tracé énergétique d'une séquence d'additions et de doublements de points sur F_p [105] 66
Figure 21. Le coût pour la courbe SECP160R1 73
Figure 22. Les tailles (en octets) du code ROM et de mémoire RAM 77
Figure 23. Modèle du réseau 82
Figure 24. La transmission des données dans SASPKC 85
Figure 25. Durée de vie d'un capteur dans SASPKC 87
Figure 26. Exemple du fonctionnement de SASPKC 88
Figure 27. Charge de communication pour différentes topologies 94
Figure 28. Energie totale estimée consommé par: 96

Introduction générale

Les récents progrès de la technologie MEMS (*Micro-Electro-Mechanical-Systems*) et les technologies des communications sans fil ont permis le déploiement des Réseaux de Capteurs Sans Fil (RCSFs) dans une pléthore d'applications, variant largement de la surveillance environnementale aux applications militaires, de l'assistance médicale à la gestion des appareils électroménagers, etc. Par conséquent, différents besoins émergent en fonction de l'application spécifique et le contexte où le réseau doit être déployé. Un RCSF est un ensemble de capteurs intégrés dans un environnement physique. Ces capteurs ont des ressources limitées en termes de calcul, stockage, communication, etc. Cependant, le fait que la communication est le processus le plus coûteux en termes de consommation d'énergie pour un nœud capteur, un objectif commun pour les applications des RCSFs est de réduire la quantité de données à transmettre autant que possible. Ces données sont généralement des mesures de capteurs qui doivent être collectées et acheminées vers un point de rassemblement au moyen d'une communication multi-saut. Une approche possible pour réduire cette quantité de données est d'utiliser l'agrégation des données, c'est à dire traiter les données afin de réduire leur taille. L'agrégation des données diminue considérablement la consommation d'énergie et, par conséquent, est un élément fondamental pour plusieurs applications des RCSFs.

Le support sans fil, cependant, est intrinsèquement dangereux, car il est accessible par tous les utilisateurs présents dans la portée de transmission. Aussi, Les réseaux de capteurs sont généralement déployés dans des environnements, souvent très hostiles et sans assistance (une application militaire par exemple). Un certain niveau de sécurité doit donc être assuré. Les besoins de sécurité en général concernent la confidentialité des données (l'attaquant ne doit pas être en mesure de comprendre le contenu du message), l'intégrité des données (les modifications accidentelles ou malveillantes des paquets doivent être détectées) et l'authentification (le récepteur devrait être en mesure de vérifier que les données proviennent bien de la source supposée).

Malheureusement, l'agrégation des données et la sécurité ne vont pas très bien ensemble car il convient d'observer qu'ils ont des objectifs opposés. En effet, la première essaie de minimiser la quantité de données transmises et la deuxième ajoute une charge de calcul et de communication non-négligeable afin d'assurer la vérification de quelques propriétés de la sécurité. D'autre part, puisque les capteurs sont dotés de ressources informatiques limitées, la mise en œuvre des primitives de sécurité est un challenge et ces limitations doivent être prises en considération. Par conséquent, les algorithmes qui sont simples et efficaces en termes d'énergie sont les plus appropriés, ou parfois le sacrifice d'un certain niveau de sécurité rend possible sa mise en œuvre sur un capteur. En résumé, les protocoles d'agrégation de données doivent être conçus conjointement avec les protocoles de sécurité, dans le but de donner un bon compromis entre la complexité du protocole global et le niveau de sécurité fourni, et cela, tout en maintenant une consommation d'énergie acceptable. Garantir la sécurité de l'agrégation constitue donc un grand défi.

L'objectif de cette thèse est lié justement à la sécurité de l'agrégation des données dans les RCSFs. Nous proposons deux nouvelles approches pour sécuriser l'agrégation i) l'approche

RSAED (*Robust and Secure Aggregation of Encrypted Data*) qui assure une confidentialité de bout en bout et une vérification des paquets saut à saut et ii) l'approche SASPKC (*Secure Aggregation using Stateful Public Key Cryptography*) qui assure les services de sécurité de bout en bout.

Cette thèse est organisée en cinq chapitres en plus d'une introduction générale et d'une conclusion générale :

Dans le premier chapitre nous présentons une description générale des réseaux de capteurs sans fil. Nous commençons d'abord par la définition des différentes notions et concepts gravitant autour de cette thématique tout en donnant quelques exemples d'applications, ensuite nous présentons les challenges liés à la conception de ces dispositifs minuscules. Nous abordons par la suite l'agrégation des données dans les RCSFs et les problèmes liés à la sécurité.

Le second chapitre détaille les multiples aspects de la sécurité afin d'assurer un transfert sécurisé des données dans les RCSFs. Ensuite, nous présentons un état de l'art sur la sécurité des données agrégées, en étudiant les forces et les faiblesses des solutions proposées dans la littérature à la fois basées sur la cryptographie et la surveillance.

Dans le troisième chapitre, nous proposons une solution robuste, efficace et sécurisée pour l'agrégation dans laquelle nous utilisons le protocole de chiffrement El Gamal ainsi que sa propriété homomorphique pour fournir une confidentialité de données de bout en bout. Le protocole proposé permet la vérification au niveau des nœuds intermédiaires, assure que les paquets reçus par la station de base (point de rassemblement) proviennent seulement des nœuds qui sont légitimes et améliore l'efficacité. Cette dernière est assurée à travers un calcul distribué exécuté par les nœuds intermédiaires.

Le quatrième chapitre est consacré à l'implémentation efficace et sécurisée de l'algorithme El Gamal sur un capteur sans fil pour une sécurisation de bout en bout. Une implémentation de l'analogue du protocole El Gamal sur les courbes elliptiques est proposée, et ce, en se basant sur des algorithmes non seulement efficaces mais aussi sécurisés contre les attaques par canaux cachés (en particulier l'attaque par analyse simple de la consommation d'énergie). Cette implémentation permet le maintien de la consommation d'énergie au minimum. Les résultats montrent que notre implémentation est considérablement meilleure par rapport à celles proposées dans la littérature.

Dans le cinquième chapitre, nous proposons un protocole qui assure les services de sécurité de bout en bout. Cette solution est basée sur le chiffrement à clé publique avec état; un chiffrement qui combine les meilleures caractéristiques des deux types de chiffrements connus (symétrique et asymétrique) et dans lequel l'émetteur maintient un état qui sera utilisé pour les futurs chiffrements. Nous avons appliqué ce type de chiffrements sur l'agrégation dans les RCSFs. La confidentialité et l'intégrité des données sont assurées de bout en bout en utilisant respectivement, le chiffrement homomorphique et l'agrégation des signatures. De plus, notre solution est versatile, c'est à dire elle permet à la station de base de calculer n'importe qu'elle fonction d'agrégation sur les données. Les résultats de simulation et d'expérimentation montrent que notre proposition offre un niveau de sécurité élevé avec

une charge minimale (calcul et communication), en comparaison avec les meilleures solutions proposées dans la littérature.

Enfin, dans la partie conclusion, les contributions de cette thèse sont résumées. Plusieurs problèmes et directions de recherche possibles sont également examinés.

Chapitre 1 : Les réseaux de capteurs sans fil

1. Introduction

Les récents progrès de la technologie MEMS (*Micro-Electro-Mechanical-Systems*) et les technologies des communications sans fil ont permis la conception et le développement de dispositifs à faible coût et à faible puissance, appelés capteurs sans fil. Les capacités de ces dispositifs minuscules, incluant la détection, le traitement et la communication, ont permis la réalisation des réseaux de capteurs sans fil. Un réseau de capteur est constitué d'un nombre très large de nœuds capteurs et quelques nœuds de contrôle puissants (appelés *stations de base*). Chaque nœud capteur possède un ou quelques composants de captage pour capter des grandeurs physiques tels que la température, l'humidité, la pression, etc. Dans l'environnement (ou zone) où il est déployé, un composant de traitement et de communication peut effectuer des calculs simples sur les données et communiquer avec les nœuds voisins. Les réseaux de capteurs sont généralement déployés à grande échelle et collaborent entre eux au moyen d'une communication sans fil. Les nœuds de contrôle peuvent traiter les données collectées, disséminer des commandes de contrôle aux nœuds capteurs, et connecter le réseau à un autre (Un réseau filaire par exemple). Dans ce chapitre, nous allons présenter brièvement les RCSFs et les questions liées à la sécurité de l'agrégation dans ce type de réseaux.

2. Un réseau de capteurs sans fil

Un RCSF est constitué de plusieurs nœuds capteurs dispersés dans un *champ de captage*. Chacun de ces nœuds est capable de collecter des données et les acheminer vers le nœud *puits (station de base)* [1]. Les données sont ensuite routées vers l'utilisateur final à travers le puits, qui permet l'interconnexion avec d'autres réseaux comme le montre la Figure 1.

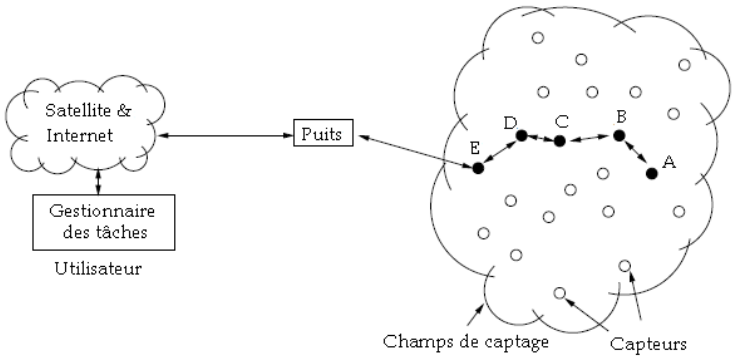


Figure 1. Architecture d'un réseau de capteurs sans fil

3. Le capteur sans fil

Un nœud capteur est composé de quatre unités principales: (i) unité de captage (ii) unité de traitement (iii) unité de transmission, et (iv) une batterie comme le montre la Figure 2. Ces quatre unités sont définies brièvement dans la suite:

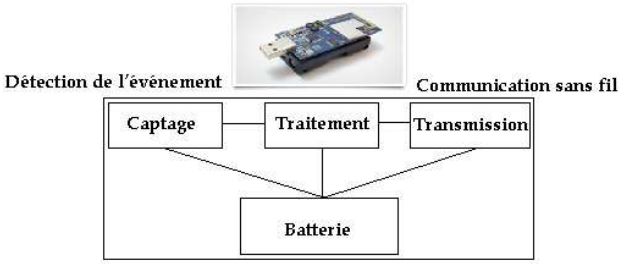


Figure 2. Composants d'un capteur sans fil

3.1. Unité de captage

Elle consiste en une série de capteurs qui peuvent mesurer les caractéristiques physiques de leur environnement, comme la température, la luminosité, la pression et autres. Chaque capteur est capable de relever ces caractéristiques via l'unité de captage et utilise ensuite un module *ADC* (*Analog to Digital Converter*) pour convertir l'information en données à transmettre à l'unité de traitement.

3.2. Unité de traitement

Elle est composée d'une mémoire interne pour stocker les données et les programmes de l'application, et un microcontrôleur pour traiter les données. Le microcontrôleur peut être considéré comme un ordinateur à capacité limitée comportant une mémoire et des interfaces nécessaires pour créer de simples applications. Cette unité devrait être en mesure de fonctionner avec une source limitée d'énergie et traiter efficacement les données délivrées par l'unité de captage.

3.3. Unité de transmission

Elle est responsable des émissions et réceptions des messages à travers un canal sans fil. En d'autres mots, elle permet à un capteur de communiquer avec d'autres nœuds capteurs et former un réseau Ad-Hoc.

3.4. Unité d'énergie

Elle fournit l'énergie nécessaire pour le fonctionnement de tous les composants du capteur. Cette énergie peut provenir soit d'une batterie ou bien d'une source d'énergie renouvelable.

Un nœud capteur peut avoir éventuellement une mémoire externe qui fonctionne comme une mémoire secondaire afin de stocker la quantité de données nécessaire pour l'application.

Les types de capteurs varient selon l'application envisagée. Le Tableau 1 présente les principales caractéristiques des capteurs les plus populaires conçus ces dernières années en termes de rapidité du processeur, mémoires, fréquence et débit. Une chronologie de ces plateformes est présentée dans la Figure 3. Ces capteurs peuvent être classés selon leurs capacités en deux catégories: (i) capteurs faibles, et (ii) capteurs puissants, les capteurs faibles sont caractérisés par leurs ressources limitées en termes de calcul, de mémoire et de communication. Ces capteurs son généralement déployés en grand nombre dans un RCSF afin d'accomplir leurs tâches de détection. Ils ont été utilisés dans le développement de plusieurs protocoles de communication. Parmi les capteurs de cette catégorie, on peut citer la famille des capteurs Mica (Mica, Mica2, MicaZ et IRIS), les capteurs Telos et Tmote [2] et le capteur EYES [3]. En plus du captage, du traitement local et de la communication multi-saut, les RCSFs nécessitent des fonctions additionnelles qui ne peuvent pas être effectuées efficacement par un capteur faible. Des tâches complexes telle que la gestion du réseau nécessite plus d'énergie et de mémoire. De plus, l'intégration des RCSFs avec d'autres infrastructures de réseaux nécessite plusieurs techniques de communication. Pour répondre à ces exigences, des capteurs puissants on été développés; on peut citer le capteur Stargate et les capteurs Imote et Imote2 développés par Intel [2].

Modèle du capteur	Rapidité CPU (MHz)	ROM (Octets)	RAM (Octets)	Freq. Radio (Hz)	Débit (kbps)
Mica [2]	6	128K	4K	868M	10/40
Mica2 [2]	16	128K	4K	433/868/916M	38.4
MicaZ [2]	16	128K	4K	2.4G	250
IRIS [2]	16	128K	8K	2.4G	250
Telos/Tmote [2]	16	48K	10K	2.4G	250
EYES [3]	8	60K	2K	868M	115
Stargate [2]	400	32M	64M	2.4G	Varie[1]
Imote [2]	12	512K	64K	2.4G	100
Imote2 [2]	13-416	32M	256K	2.4G	250

Tableau 1. Modèles de capteurs sans fil

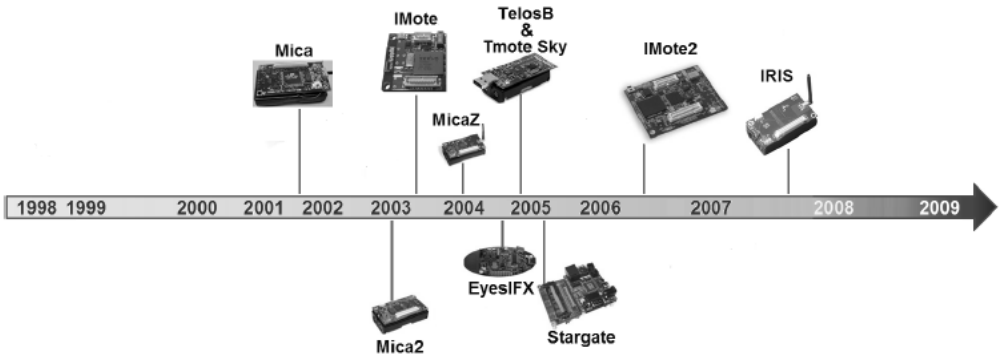


Figure 3. Chronologie des capteurs sans fil

[1] Le débit du capteur Stargate dépend du dispositif de communication avec lequel il est connecté (Capteur MicaZ, Carte WLAN, etc.).

La pile protocolaire utilisée par le puits et tous les nœuds capteurs est présentée dans la Figure 4. Cette pile consiste en une couche physique, une couche liaison, une couche réseau, une couche transport et une couche application, en plus d'un niveau de synchronisation, un niveau de localisation, un niveau de gestion de la topologie, un niveau de gestion de mobilité, un niveau de gestion de l'énergie et un niveau de gestion des tâches. Suivant la fonctionnalité des capteurs, de nombreuses applications peuvent être développées sur la couche application. La couche transport, quant à elle, sert à maintenir le flux de données en cas de nécessité dans les applications utilisées, particulièrement lors d'une connexion avec internet, tandis que la couche réseau est chargée de router les données fournies par la couche transport et cela d'une manière efficace. Puisque les RCSFs sont intrinsèquement non fiables et les nœuds peuvent être mobile, la couche liaison assure la fiabilité de la communication en utilisant les techniques de correction d'erreurs et aussi les méthodes d'accès au canal pour minimiser les collisions. Enfin, La couche physique est responsable de la sélection de fréquence, de la génération de la fréquence porteuse, de la détection de signal et de la modulation.

Le niveau de gestion d'énergie consiste à gérer comment le nœud capteur utilise son énergie, un capteur peut par exemple éteindre son interface de réception dès qu'il reçoit un message d'un nœud voisin afin d'éviter la réception des messages dupliqués. Le niveau de gestion de mobilité détecte et enregistre tous les mouvements des nœuds capteurs, d'une manière à leur permettre de garder toujours une route vers l'utilisateur final, et maintenir une image récente des nœuds voisins; cette image étant nécessaire pour pouvoir équilibrer l'exécution des tâches et la consommation d'énergie. Le niveau de gestion des tâches assure l'équilibrage et l'ordonnancement des tâches sur les différents nœuds du réseau. En effet, détecter avec le même rythme pour tous les capteurs n'est pas obligatoire. Ainsi, il est plus efficace si les capteurs peuvent collaborer les uns avec les autres, de sorte que la durée de vie du RCSF peut être prolongée.

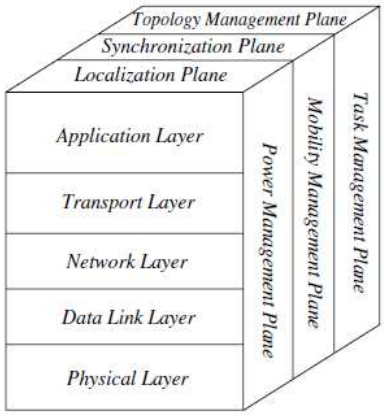


Figure 4. Pile protocolaire

En plus des fonctionnalités de communication de la pile protocolaire, les RCSFs ont également été équipés de plusieurs fonctionnalités facilitant la bonne exécution des solutions proposées. Dans un RCSF, chaque capteur est équipé de sa propre horloge nécessaire pour les opérations internes. En effet, chaque évènement lié au fonctionnement du capteur, à savoir la détection, le traitement et la communication, est associé à une synchronisation

temporelle commandé par l'horloge locale. Le RCSF devrait être en mesure d'ordonner correctement les événements détectés par les capteurs afin de modéliser avec précision l'environnement physique. Ces exigences temporelles ont conduit à l'élaboration de protocoles de synchronisation dans les RCSFs.

Les RCSFs sont étroitement associés aux phénomènes physiques dans leur environnement. Les informations recueillies doivent être associées à l'emplacement des nœuds capteurs afin de fournir une vue précise du champ de captage. Aussi, les RCSFs peuvent être utilisés pour le suivi de certains objets dans les applications de surveillance, ce qui nécessite l'intégration des informations de localisation dans les algorithmes de suivi. De plus, les services basés sur la localisation et les protocoles de communication nécessitent des informations de position. Par conséquent, les protocoles de localisation ont été incorporés dans la pile de communication.

Enfin, plusieurs solutions de gestion de topologie sont nécessaires pour maintenir la connectivité et la couverture du RCSF. Les algorithmes de gestion de la topologie fournissent des méthodes efficaces pour le déploiement du réseau, engendrant ainsi une durée de vie plus longue et une couverture efficace de l'information. De plus, les protocoles de contrôle de topologie permettent de déterminer les niveaux de puissance de transmission ainsi que les durées de l'activité des nœuds capteurs afin de minimiser la consommation d'énergie tout en assurant la connectivité réseau. Enfin, les protocoles de *clustering* sont utilisés pour organiser le réseau en *clusters* afin d'améliorer la scalabilité et améliorer la durée de vie du réseau.

L'intégration de chacun des composants pour un fonctionnement efficace, dépend des applications qui s'exécutent sur le RCSF. Cette dépendance définit plusieurs propriétés uniques. Bien que les débuts de recherche et le déploiement des RCSFs se focalisait uniquement sur le transfert des données sur un support sans fil, plusieurs nouveaux domaines pour les RCSFs ont émergés tels que les Réseaux de Capteurs et Acteurs Sans Fil (RCASFs) qui consistent en des actionneurs, en plus des capteurs qui convertissent les informations détectées en des actions agissant sur l'environnement, les Réseaux de Capteurs Multimédia Sans Fil (RCMSFs) qui supportent le trafic multimédia en termes d'information audiovisuelle en plus des données scalaires. De plus, récemment, le phénomène des RCSFs a été adopté dans des environnements contraints tels que les paramètres sous-marins et souterrains pour créer les Réseaux de Capteurs sous-Marins Sans Fil (RCMSFs) et les Réseaux de Capteurs sous-Terrains Sans Fil (RCTSFs). Ces nouveaux domaines de recherche posent des défis supplémentaires qui n'ont pas été étudiés par le grand nombre de solutions développées pour les réseaux de capteurs traditionnels.

4. Les domaines d'applications des RCSFs

Il existe plusieurs domaines d'applications des RCSFs, faisant l'objet de nombreux thèmes de recherche et développement, non seulement dans le milieu universitaire mais aussi dans l'industrie. Dans ce qui suit, nous décrivons brièvement quelques applications des RCSFs dans différents domaines:

4.1. Surveillance de l'habitat écologique

Les études scientifiques des habitats écologiques (animaux, plantes, micro-organismes) sont traditionnellement menées par des activités pratiques des enquêteurs sur le terrain. Un problème préoccupant dans ces études est ce que l'on appelle parfois «l'effet de l'observateur». En effet, la présence des enquêteurs sur le terrain peut affecter le comportement des organismes présents dans l'habitat surveillé et donc biaiser les résultats observés. Un RCSF peut être déployé pour la surveillance de l'habitat, dans lequel les nœuds capteurs jouent le rôle de l'observateur. De plus, les réseaux de capteurs, en raison de leur grande échelle et leur densité spatio-temporelle, peuvent fournir des données expérimentales d'une richesse inouïe. Une application pour la surveillance de l'habitat est présentée dans [4] et une autre pour la surveillance des serres est relatée dans [5].

4.2. Surveillance militaire et poursuite de cibles

Comme beaucoup d'autres technologies de l'information, les RCSFs proviennent originellement de la recherche liée au secteur militaire. Les réseaux de capteurs sont considérés comme un ingrédient clé dans la réalisation des systèmes de guerre. Ils peuvent être rapidement déployés pour la surveillance et utilisés pour fournir des renseignements sur le champ de bataille concernant l'emplacement, le nombre, le mouvement, et l'identité des troupes et des véhicules, et aussi pour la détection des armes nucléaires, chimiques et biologiques [6]. Un exemple intéressant est l'auto-guérison de la *DARPA* [7], un réseau de capteurs où la communication *peer-to-peer* entre les mines antichars est utilisée pour répondre aux attaques. La détection collaborative des cibles, basée sur les RCSFs à mobilité réactive, est présentée dans [8] où un algorithme d'ordonnancement de mouvements du capteur a été développé et son efficacité a été prouvée à travers des simulations.

4.3. Surveillance structurelle et sismique

Une autre classe d'applications des RCSFs concerne le suivi de l'état des structures civiles, les structures pouvant être des bâtiments, des ponts, des routes, ou même des appareils. A l'heure actuelle, la santé de ces structures est contrôlée principalement par le biais des inspections manuelles et visuelles ou occasionnellement par des technologies coûteuses, comme les rayons *X* et les ultrasons. Les techniques des RCSFs peuvent automatiser le processus, fournissant des informations riches et opportunes sur les fissures naissantes ou sur d'autres dégâts structurels. Les chercheurs envisagent de déployer des capteurs à forte densité sur les structures, soit littéralement incorporés dans le matériau de construction comme le béton, ou sur la surface. Ces réseaux de capteurs ont la capacité de surveiller l'usure à long terme des structures ainsi que leur état après des événements destructeurs, tels que les tremblements de terre ou les explosions. Une vision futuriste particulièrement convaincante pour l'utilisation des réseaux de capteurs implique le développement de structures contrôlables, contenant des actionneurs qui réagissent à l'information du capteur en temps réel pour effectuer "l'annulation d'écho" sur les ondes sismiques de sorte que la structure ne sera pas affectée par une perturbation externe. Des applications à long terme basées sur les RCSFs pour le traitement de l'information en ligne sont présentées dans [9, 10, 11].

4.4. Les réseaux de détection industriels et commerciaux

Dans les installations de fabrication industrielle, des capteurs et des actionneurs sont utilisés pour la surveillance et le contrôle des processus. Par exemple, dans une usine de transformation chimique, il peut y avoir des capteurs placés en différents points du processus afin de contrôler la température, la concentration du produit chimique, la pression, etc. L'information recueillie à partir de cette surveillance en temps réel peut être utilisée pour faire varier les contrôles du procédé, tel que le réglage de la quantité d'un ingrédient particulier ou de la température. Le principal avantage d'utiliser les RCSFs dans ces milieux est qu'ils peuvent améliorer le coût et la flexibilité associés à l'installation, l'entretien et la modernisation des systèmes câblés [12]. Parmi les applications commerciales des réseaux de capteurs: la surveillance des véhicules, la protection des biens et la détection d'événements. La détection des véhicules dans un parking à l'aide de capteurs à ultrasons et magnétiques est présentée dans [13]. Un système de stationnement intelligent basé sur les RCSFs (*SPARK*) est présenté dans [14], où le suivi du parking à distance et le mécanisme de réservation de parking et d'orientation automatisé, sont quelques unes des principales fonctionnalités offertes par le système.

4.5. Les applications médicales

La recherche médicale peut grandement bénéficier des RCSFs, la surveillance des signes vitaux et la reconnaissance d'accidents sont les applications les plus naturelles. Une question importante est la prise en charge des personnes âgées, un réseau de capteurs et d'actionneurs pourrait les surveiller et même les aider dans leurs tâches quotidiennes. Des appareils intelligents pourraient les aider à organiser leur vie en leur rappelant leurs repas et quand prendre leurs médicaments. Les capteurs peuvent être utilisés pour capturer les signes vitaux des patients en temps réel et relayer les données vers les ordinateurs portables du personnel médical, les nœuds capteurs peuvent aussi stocker des données sur les patients tels que l'identification, l'historique et les traitements. La mise en œuvre et l'analyse d'une application E-santé basée sur les RCSFs est présentée dans [15]. Une recherche qualitative sur la prise en charge des personnes âgées est décrite dans [16], ou encore [17, 18].

5. Les challenges liés à la conception des RCSFs

Les RCSFs sont intéressants d'un point de vue technique, parce qu'ils présentent un certain nombre de sérieux défis qui ne peuvent être traités de manière adéquate par les technologies existantes:

5.1. La durée de vie

Comme mentionné précédemment, les nœuds du RCSF sont généralement sévèrement restreints en termes d'énergie en raison des limitations des batteries. Une pile alcaline typique, par exemple, fournit environ 50 watt-heures d'énergie, ce qui peut se traduire par moins d'un mois de fonctionnement en continu pour chaque nœud en mode actif. Étant donné le coût et l'infaisabilité de la surveillance et du remplacement des batteries pour un grand réseau, des durées de vie plus longues sont souhaitées. En pratique, il est nécessaire

dans de nombreuses applications de fournir des garanties qu'un réseau de capteurs sans fil sans assistance peut rester opérationnel pendant plusieurs mois voire plusieurs années. Les améliorations matérielles concernant la conception des batteries et les techniques de récolte d'énergie n'offrent que des solutions partielles. C'est la raison pour laquelle la plupart des protocoles dans les RCSFs sont conçus en fixant l'efficacité énergétique comme objectif principal.

5.2. La réactivité

Une solution simple pour étendre la durée de vie du réseau est d'exploiter les nœuds d'une manière cyclique avec commutation périodique entre les modes de réveil et de sommeil, bien que la synchronisation des modes de sommeil soit un défi. En effet, une plus grande préoccupation est que les périodes arbitrairement longues de sommeil peuvent réduire la réactivité et l'efficacité des capteurs. Dans les applications où il est essentiel que certains événements dans l'environnement soient détectés et signalés rapidement, la latence induite par les modes de sommeil doit être maintenue dans des limites strictes, même en présence de congestion du réseau.

5.3. La robustesse

La vision des réseaux de capteurs sans fil est de fournir un réseau à grande échelle avec la meilleure couverture possible, ce qui motive l'utilisation d'un grand nombre de dispositifs peu coûteux. Cependant, les dispositifs peu coûteux peuvent souvent être peu fiables et sujets à des défaillances. Le taux de défaillance du dispositif varie selon la difficulté ou l'hostilité de l'environnement dans lequel il est déployé. Les protocoles doivent donc avoir des mécanismes intégrés pour fournir de la robustesse. Il est donc important de s'assurer que les performances globales du système ne sont pas sensibles au mauvais fonctionnement d'un ou plusieurs dispositifs.

5.4. La scalabilité

Plusieurs applications envisagent des RCSFs avec des centaines, voire des milliers de capteurs. Par conséquent, les protocoles devront être intrinsèquement distribués et les réseaux de capteurs doivent utiliser des architectures hiérarchiques afin de fournir une telle scalabilité. Cependant, les visions d'un grand nombre de nœuds resteront loin de la pratique puisque certains problèmes fondamentaux tels que la tolérance aux pannes et la programmation in-situ, doivent être traitées, même dans des petits réseaux impliquant des dizaines, voire des centaines de nœuds. Il y a aussi quelques limites fondamentales sur le débit et la capacité qui influent sur la scalabilité et les performances du réseau.

5.5. L'auto-configuration

En raison de leur grande échelle et la nature de leurs applications, les RCSFs sont par nature des systèmes distribués sans assistance. Le fonctionnement autonome du réseau est donc un enjeu clé de la conception. Dès le début, les nœuds d'un RCSFs doivent être en mesure de configurer leur propre topologie du réseau, se localiser, se synchroniser et se calibrer,

communiquer les coordonnées et déterminer d'autres paramètres importants liés au fonctionnement.

5.6. L'auto-optimisation et adaptation

Traditionnellement, la plupart des systèmes d'ingénierie sont optimisés a priori pour fonctionner efficacement face à des conditions prévues ou bien modélisées. Dans les RCSFs, il peut souvent y avoir une grande incertitude sur les conditions avant leur déploiement. Dans de telles conditions, il est important qu'il y ait des mécanismes intégrés pour apprendre de manière autonome des informations concernant le réseau et utiliser ces connaissances pour améliorer en permanence les performances. Aussi, l'environnement dans lequel le réseau fonctionne peut changer radicalement au fil du temps. Les protocoles des RCSFs devraient également être en mesure de s'adapter à ces dynamiques environnementales d'une manière spontanée.

5.7. La sécurité

Les réseaux de capteurs peuvent être utilisés dans des applications critiques, donc leur sécurité est une question essentielle et à considérer. Les attaques par déni de service par exemple peuvent être facilement exécutées sur un RCSF. De plus, les approches de communication en temps réel ne considèrent pas les questions de sécurité. Ainsi, certains intrus peuvent facilement exploiter ces failles de sécurité. Le grand dilemme est de savoir comment mettre en œuvre des techniques de sécurité qui ont besoin d'importantes ressources de calcul dans une technologie à ressources informatiques très limitées.

6. Agrégation des données et sécurité dans les RCSFs

6.1. L'agrégation des données

Dans plusieurs applications des RCSFs, un phénomène physique est d'abord détecté par des nœuds capteurs puis transmis à la station de base. Afin de réduire les dépenses énergétiques de communication, ces applications doivent minimiser le nombre de paquets transitant dans le réseau en éliminant les données redondantes. Ainsi, ces applications peuvent utiliser l'agrégation avant que les données atteignent la station de base.

L'agrégation des données est l'une des techniques considérée actuellement comme un paradigme essentiel pour les réseaux de capteurs. Dans l'agrégation des données, le réseau se compose de trois types de nœuds: (i) les nœuds capteurs, (ii) les nœuds d'agrégation, et (iii) la (les) station(s) de base, ce dernier peut être un capteur ou tout autre appareil et il est généralement un dispositif puissant. Les nœuds capteurs détectent l'information à partir d'une région cible et l'envoient aux nœuds d'agrégation. Les nœuds d'agrégation effectuent la fonction d'agrégation sur les données reçues. Une fonction d'agrégation peut être la moyenne, le maximum, le minimum, ou toute autre opération arithmétique. Enfin, les données agrégées sont transmises vers la station de base. Dans un réseau multi-saut, les données sont agrégées au niveau de chaque nœud intermédiaire qu'elles traversent avant d'atteindre la station de base. Considérons l'exemple présenté dans la Figure 5. Sans

agrégation, un total de 4+6+7=17 messages sont transmis dans le réseau. Avec agrégation, seulement 4+2+1=7 messages sont envoyés. Ainsi, à travers ce simple exemple, il est clair qu'avec l'agrégation, la quantité de données est réduite, ce qui conduit à une réduction de la consommation d'énergie et par conséquent la durée de vie du réseau est améliorée [19].

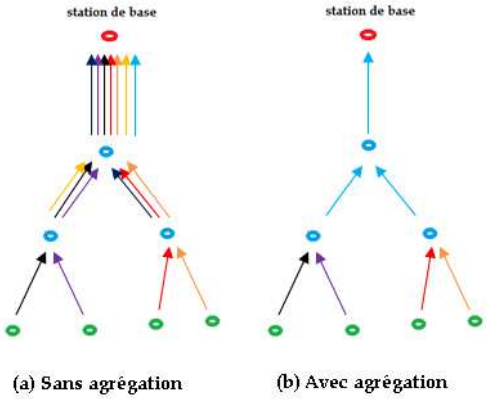


Figure 5. Efficacité de l'agrégation des données

Il existe deux types de topologies pour agréger les données dans les RCSFs [20], une topologie statique basée sur les arbres et une topologie dynamique basée sur les clusters (Voir Figures 6 et 7). La première consiste en une structure en arbre dans laquelle les nœuds feuilles sont des nœuds *capteurs* et tous les nœuds non-feuilles jouent le rôle d'*agrégateurs* et la *station de base* représente la racine de l'arbre. Un agrégateur peut aussi capter sa donnée, cette donnée est ensuite combinée avec celles reçues. Le calcul du chemin optimal pour acheminer les données dans l'arbre est un processus complexe et cela est souvent réalisé au niveau de la station de base. L'inconvénient majeur de ce type de topologies est le fait que si un paquet d'un nœud particulier est perdu (à cause du canal ou un mauvais fonctionnement) les données de l'ensemble du sous arbre associé à ce nœud seront perdues. Ce problème peut être plus sévère si ce nœud est proche de la racine. Plusieurs techniques d'agrégation sur les arbres ont été étudiées [21,22,23,24]. Dans une topologie basée sur les clusters, le réseau est divisé en groupe appelé *cluster*, un chef de groupe appelé *Cluster-Head* (*CH*) est élu dans chaque groupe, le *CH* effectue localement l'agrégation des données reçues par les autre nœuds du groupe et envoie le résultat à la station de base ou au chef du groupe le plus proche. [25] et [26] sont des exemples de protocoles basés sur les clusters.

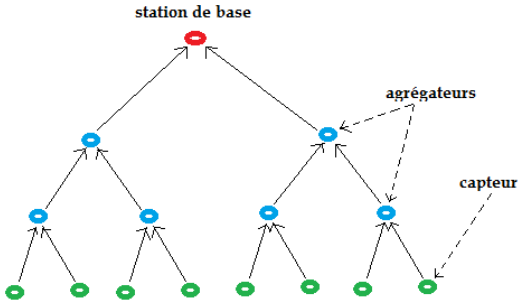


Figure 6. Agrégation basée sur les arbres

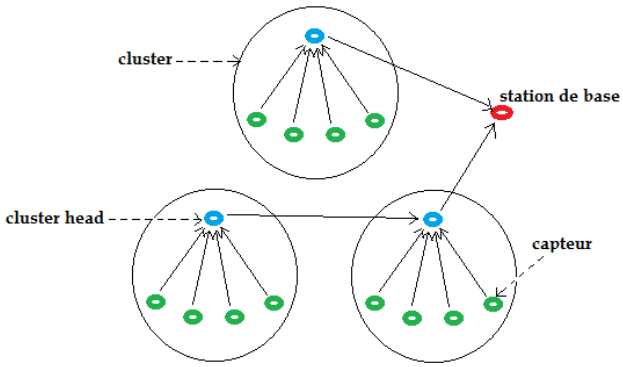


Figure 7. Agrégation basée sur les clusters

Dans les références [19,27,28], il a été montré que la transmission consomme plus d'énergie par rapport au calcul. Comme illustré dans la Figure 5, l'agrégation des données peut considérablement diminuer la consommation énergétique en réduisant la quantité de données transmise et prolonger ainsi la durée de vie du capteur. Cependant, le problème majeur de l'agrégation de données, c'est quand nous visons à fournir des services de sécurité. Les RCSFs possèdent quelques caractéristiques particulières différentes des autres types de réseau: (i) ils ont des contraintes de ressources, ce qui rend le choix des algorithmes de sécurité quelque peu difficiles, (ii) ils sont souvent déployés dans un environnement hostile et sans assistance, ce qui les rend soumis à plusieurs types d'attaques. Dans un scénario d'agrégation par exemple, le nœud d'agrégation effectue l'agrégation de données qui concernent plusieurs nœuds du réseau, ce qui le rend plus attrayant pour un adversaire que les autres nœuds, et (iii) ils utilisent un support sans fil, ce qui pourrait permettre à un attaquant d'intercepter les paquets et même participer au processus d'agrégation. En résumé, la sécurité est un enjeu crucial pour les RCSFs et pose plusieurs défis lors de la conception d'un protocole. Dans ce qui suit, les exigences de sécurité et les types d'attaques sont présentées.

6.2. Les exigences de sécurité dans les RCSFs

Plusieurs propriétés peuvent être exigées à un protocole de sécurité, et cela en fonction de l'application spécifique [29]. Les principales exigences de sécurité sont présentées dans la suite.

6.2.1. La confidentialité des données

La confidentialité des données garantit que les données ne sont jamais divulguées à des tiers non autorisés. Cela signifie que l'information transmise par chaque nœud capteur n'est lisible que par le nœud auquel cette information est destinée. Il existe deux points d'attaques dans un RCSF (i) les nœuds et (ii) le canal de communication. Un réseau sécurisé fournissant la confidentialité des données empêchera toutes fuites d'information à partir des nœuds et fournira un canal sécurisé pour ces nœuds afin de communiquer. Ceci est vital pour les RCSFs où la communication à travers le canal sans fil est sensible à l'écoute. Dans une application militaire par exemple, garantir la confidentialité des données signifie ne pas laisser l'ennemi se procurer les données du réseau. Le chiffrement est la technique

fondamentale utilisée pour assurer la confidentialité. La plupart des algorithmes fournissent une communication sécurisée en utilisant un mécanisme de chiffrement. Les algorithmes de chiffrement sont classés en deux catégories: le chiffrement symétrique et le chiffrement asymétrique.

6.2.2. L'authentification

Puisque les capteurs utilisent un support sans fil pour communiquer, les mécanismes d'authentification sont nécessaires pour détecter les paquets malicieusement injectés ou falsifiés, afin de s'assurer que les données sont effectivement fournies par la source supposée. Le manque d'une communication authentifiée pourrait conduire à plusieurs problèmes, tels que l'usurpation d'identité, les attaques sybilles, etc.

6.2.3. L'intégrité des données

Cette propriété garantit que le contenu d'un message n'a pas été altéré, que ce soit malicieusement ou accidentellement, pendant le processus de transmission. Dans les RCSFs, où la communication est souvent multi-sauts, un nœud intermédiaire compromis peut modifier un message entre l'émetteur et le récepteur. Un RCSF garantissant l'intégrité des données bloquera toute tentative d'injection de fausses données dans le réseau. Le *MAC* (*Message Authentication Code*) et les signatures numériques sont généralement utilisés pour vérifier l'intégrité.

6.2.4. La disponibilité des données

Si cette propriété est vérifiée, le réseau sera en mesure de garantir la disponibilité des ressources et des données mesurées, même en présence de quelques nœuds compromis. Il est pratiquement impossible de fournir 100% de disponibilité des données dans un RCSF, car les transmissions des capteurs sont de faible puissance et peuvent être facilement bloquées par un attaquant puissant. Cela peut conduire à des conséquences catastrophiques dans certaines applications comme le cas d'une application militaire où l'indisponibilité des données dans un temps voulu pourrait conduire par exemple à une invasion de l'ennemi. Les mécanismes de surveillance sont souvent utilisés pour assurer ce service.

6.2.5. La fraicheur des données

Afin de protéger le réseau contre les attaques par rejeu, les messages ayant le même contenu devrait être différent à chaque fois qu'ils sont transmis, de sorte que l'attaquant ne peut pas exploiter la confidentialité, l'authenticité ou l'intégrité d'un message à l'instant t pour les transmissions successives à l'instant $t+s$. La fraîcheur des données peut être assurée simplement par l'intégration d'un compteur ou bien d'un nombre pseudo-aléatoire (partagé entre l'émetteur et le récepteur) dans le paquet avant d'appliquer la primitive de chiffrement utilisée pour assurer la confidentialité ou l'authentification.

6.3. Les attaques contre les RCSFs

Les problèmes de sécurité proviennent principalement des attaques. Les stations de base dans les réseaux de capteurs sont généralement considérées comme dignes de confiance. En

effet, la plupart des études se concentrent sur la sécurité des nœuds capteurs. S'il n'existe aucune attaque, bien sûr il ne sera pas nécessaire de sécuriser. En général, la probabilité d'attaque contre les réseaux de capteurs est supérieure à celle de tous les autres types de réseaux, tels que les réseaux locaux sans fil, et cela en raison de leurs environnements de déploiement et les ressources limitées. Une classification exhaustive des attaques possibles contre les RCSFs est au-delà de l'objectif de cette thèse et, par conséquent, le lecteur intéressé par un aperçu global peut se référer à un travail spécifique, tel que celui présenté dans [30]. Dans ce qui suit, l'accent est principalement mis sur les attaques liées à l'agrégation des données.

6.3.1. Les attaques passives

Les attaques passives comprennent toutes les attaques qui n'obligent pas l'adversaire à interférer activement avec la connexion. Pour effectuer de telles attaques, l'adversaire ne doit rien faire d'autre qu'écouter les paquets transmis. L'information espionnée peut être évaluée et généralement cryptanalysée afin d'obtenir des données secrètes. Bien que le pire des cas d'une telle attaque soit généralement la déduction de la clé secrète, la plupart des attaques visent à révéler les textes en clair ou à recueillir des informations pour des actions futures. Les attaques passives peuvent être facilement exécutées. Compte tenu des caractéristiques du support, ces attaques sont indétectables, ce qui les rend très dangereuses. Par conséquent, il est primordial qu'un adversaire ne soit pas en mesure d'obtenir des informations par une simple écoute, surtout dans des applications où la confidentialité est largement désirée (une application militaire par exemple). Bien sûr, un solide algorithme de chiffrement peut assurer la sécurité à l'égard de ce type d'attaque, avec un coût en termes de consommation d'énergie.

6.3.1.1. L'analyse des chiffrés

L'attaque passive la plus fondamentale est l'analyse des paquets cryptés. Dans une telle analyse, l'adversaire vise à obtenir des informations seulement en interprétant des chiffrés. Le système doit donc s'assurer qu'il n'est pas possible d'obtenir toute information sensible (texte en clair, clé) à partir du chiffré. De plus, il doit s'assurer qu'un attaquant ne puisse pas décider si un paquet chiffré correspond à un texte en clair spécifique ou non, en particulier, dans les réseaux de capteurs caractérisés par un intervalle limité de valeurs mesurées, ce qui peut conduire à la déduction des textes en clairs.

6.3.1.2. Attaque à texte en clair connue

Dans ce type d'attaques, l'adversaire essaie de déterminer des informations secrètes en connaissant les textes en clair. La connaissance du texte en clair avec le chiffré correspondant est le but principal de l'adversaire où il peut révéler la clé secrète ou bien recueillir des informations supplémentaires pouvant être exploitées pour décrypter d'autres messages. Dans les RCSFs, une telle attaque est très probable car un adversaire peut obtenir les textes en clair correspondant aux chiffrés envoyés par diverses manières, par exemple (i) en devinant le texte en clair (par exemple la température), (ii) en utilisant son propre capteur qui détermine le texte en clair, (iii) en accédant physiquement au capteur, ou (iv) en faussant

les mesures du capteur (par exemple chauffer le capteur). En supposant que le système utilise la même clé secrète sur chaque nœud, ce genre d'attaques est une menace sérieuse si le système ne prévoit pas de résistance. La résistance à cette attaque signifie que, même avec un grand nombre de pairs (texte en clair, chiffré), il ne sera pas possible de déduire les clés secrètes ou bien d'autres pairs.

6.3.2. Les attaques actives

Contrairement aux attaques passives décrites précédemment, dans les attaques actives, l'adversaire est supposé être capable d'interférer activement avec la connexion, c'est-à dire obtenir, détruire, modifier et injecter des paquets. Un attaquant pourrait prendre un paquet, l'analyser, modifier son contenu et même le remplacer par un autre. Ces attaques obligent l'attaquant à avoir beaucoup plus de connaissances et d'instruments techniques. Bien que de telles attaques sont beaucoup plus compliquées et plus coûteuses que les attaques passives, leurs dommages potentiels peuvent être également beaucoup plus sévère. Une attaque qui permet à un adversaire de modifier ou de falsifier n'importe quel paquet peut rendre l'ensemble du réseau inutile.

6.3.2.1. Attaque par capture

Compromettre un nœud capteur permet à un attaquant d'accéder à l'ensemble de ses contenus (tels que les clés cryptographiques) et de gérer la communication avec ses voisins. Un nœud compromis peut modifier les données captées à sa propre volonté, sans être détecté comme malicieux après vérification par le récepteur. Cette attaque est très dangereuse. En effet, l'accès à la mémoire peut révéler des informations importantes pouvant compromettre l'ensemble du réseau. La capture du nœud peut également s'effectuer d'une manière furtive. En effet, les auteurs dans [31] ont élaboré une compromission furtive (attaque par canaux cachés) dans laquelle l'attaquant prend les tracés de la consommation énergétique du nœud sans le supprimer du réseau ou bien perturber son fonctionnement. Ces tracés peuvent permettre à l'adversaire de déduire la clé secrète.

6.3.2.2. Attaque par déni de service

Dans cette classe, un large éventail d'attaques possibles est inclus. Ces attaques pourraient être réalisées par un brouillage radio, mais aussi en déviant simplement le comportement d'un capteur, de manière à éviter que le système fonctionne correctement. Dans un scenario d'agrégation des données, un exemple de cette attaque peut être celui d'un agrégateur qui refuse des paquets reçus de ses nœuds membres, empêchant ainsi les données d'aller vers le prochain saut.

6.3.2.3. Envoi sélectif des paquets

Par cette attaque, un nœud compromis peut ignorer un ou plusieurs paquets reçus, empêchant ainsi le réseau de fonctionner correctement (Attaque par déni de service) ou tout simplement en modifiant le résultat de l'agrégation.

6.3.2.4. Attaque sybille

Cette attaque permet à l'adversaire de se faire passer pour plus d'un capteur au sein du même réseau, de cette façon, il peut modifier les protocoles d'agrégation de plusieurs manières. Par exemple, il peut créer des identités multiples pour générer des voix supplémentaires lors de la phase de l'élection de l'agrégateur (en supposant que les agrégateurs ne sont pas choisis à l'avance), permettant l'élection d'un nœud compromis. Sinon, l'attaquant pourrait décider de contribuer à plusieurs reprises dans le processus d'agrégation, en exploitant ses différentes identités.

6.3.2.5. Attaque par rejeu

Un attaquant pourrait enregistrer une partie du trafic du réseau, sans nécessairement comprendre son contenu (c'est à dire sans violer la confidentialité), et de retransmettre ces paquets plus tard, en exploitant leur authenticité et intégrité pour tromper le récepteur afin de modifier, par exemple, le résultat de l'agrégation.

6.3.2.6. Attaque furtive

Un attaquant vise à injecter des paquets dans le réseau sans révéler son existence. Dans un scénario d'agrégation des données, les fausses données injectées conduisent à un faux résultat d'agrégation, le but de l'attaquant est d'obliger le réseau à accepter des résultats malicieux.

6.4. La sécurité et l'agrégation dans les RCSFs

Dans les sections précédentes, l'agrégation des données et la sécurité dans les RCSFs ont été introduites. L'agrégation des données est nécessaire afin d'améliorer la durée de vie globale du réseau, qui est en effet un paramètre fondamental de conception pour un RCSF. Cependant, le concepteur devrait, en même temps, non seulement considérer les menaces de sécurité pouvant survenir dans un réseau facilement accessible à l'attaquant, mais aussi faire face à des ressources informatiques très limitées. Comme nous l'avons mentionné dans l'introduction générale, la sécurité et l'agrégation ont des objectifs opposés. Ainsi, on peut souligner plusieurs difficultés rencontrées lors de la mise en œuvre d'un protocole de sécurité pour l'agrégation: (i) Un risque majeur de l'agrégation des données est quand un agrégateur est compromis. En effet, un tel nœud traite des données concernant plusieurs capteurs, corrompant ainsi non seulement ses propres mesures, mais aussi celle de tous les nœuds participants à l'agrégat. En conséquence, un adversaire qui capte les nœuds de manière sélective et stratégique, peut corrompre tout le processus d'agrégation du réseau, tout en encourant un effort minime. Surveiller les nœuds les plus importants et intéressants pour l'attaquant reste une tâche très difficile surtout dans un tel contexte. (ii) L'agrégation de données nécessite le traitement de l'information au niveau de l'agrégateur, par conséquent, assurer la vérification de bout en bout des services de sécurité est extrêmement difficile. Ainsi, un assouplissement des exigences de sécurité et des hypothèses particulières sur l'attaquant ou sur les capacités du réseau pourrait être nécessaire afin de rendre le processus d'agrégation faisable et efficace et cela même en présence de mécanismes de sécurité. (iii) Les capteurs étant sévèrement limités en termes de ressources de calcul, de stockage et d'énergie;

chaque protocole pour les RCSFs devrait être conçu avec cette limitation à l'esprit. Quand il s'agit de sécurité, il est encore plus difficile de se conformer à ces contraintes, les solides mécanismes traditionnels sont trop coûteux pour la plupart des capteurs.

7. Conclusion

Dans ce chapitre, nous avons introduit le concept des réseaux de capteurs sans fil en donnant des définitions et quelques applications potentielles. Nous avons présenté par la suite les challenges liés à la conception de ce type de réseaux, où l'objectif principal de cette thèse est d'étudier l'un de ces challenges à savoir la sécurité. La sécurité constitue un défi intriguant pour les RCSFs et ceci est dû principalement aux caractéristiques spécifiques des RCSFs, différentes des autres types de réseaux. Finalement, nous avons défini l'agrégation des données et présenté les différentes attaques possibles contre cette technique.

Nous avons remarqué à travers nos lectures que d'une part, l'agrégation de données constitue un avantage considérable pour les RCSFs en termes de réduction de la consommation d'énergie. Et d'autre part, l'agrégation peut être sujet à plusieurs attaques d'où la nécessité de fournir de la sécurité. Cependant, cette dernière nécessite d'importantes ressources. Par conséquent, une solution pour sécuriser l'agrégation doit donc fournir un certain équilibre (compromis) entre les services de sécurité fournis et les performances du réseau. Cette problématique constitue les objectifs de cette Thèse.

Dans le chapitre suivant, un état de l'art sur les solutions proposées dans la littérature sera présenté. Cette étude nous a permis de proposer de nouvelles solutions qui seront présentées dans les prochains chapitres.

Chapitre 2 : La sécurité des données agrégées dans les RCSFs

1. Introduction

Un réseau de capteur sans fil apparait comme une technologie émergente qui se compose d'une centaine, voire un millier de nœuds capteurs à ressources limitées. Ces limitations concernent l'énergie, le calcul, le stockage et la communication. Comme nous l'avons souligné dans le premier chapitre, la communication entre les capteurs consomme une quantité considérable d'énergie ainsi, une solution qui vise à réduire cette consommation est largement souhaitée. L'agrégation des données est une solution très utilisée dans les RCSFs. Elle permet de réduire le nombre de messages transmis dans le réseau en combinant les données provenant de différents capteurs du réseau et par conséquent réduire la consommation d'énergie. Cependant, dans des environnements hostiles, les données agrégées peuvent être sujets à plusieurs types d'attaques, fournir la sécurité est donc primordial. Plusieurs protocoles ont été proposés pour sécuriser l'agrégation. Dans ce chapitre, nous allons passer en revue ces protocoles et identifier leurs avantages et inconvénients.

2. La sécurité des données agrégées

Dans le but de faire face aux risques de la sécurité, les protocoles doivent fournir les services de sécurité définis précédemment. Plusieurs solutions ont été proposées pour sécuriser l'agrégation, et plusieurs travaux ont comparé ces solutions par la suite. Dans [32], les auteurs ont analysé les vulnérabilités des solutions et ont comparé celles qui résistent aux attaques d'injection de données fausses. Cependant, ils ont seulement examiné quelques solutions. Dans [33], les solutions ont été classifiées en deux catégories: les solutions saut à saut et les solutions de bout en bout. Cependant, l'analyse de sécurité et des performances n'a pas été fournie. Dans [34], les auteurs ont étudié la plupart des solutions et ont donné quelques détails sur les services de sécurité fournis par chaque solution. Cependant, leur travail date de 2009 et plusieurs solutions ont été proposées depuis. En 2011, nous avions commencé par examiner quelques solutions basées spécialement sur la cryptographie à clé publique [35], puis nous avons étendu cette étude en analysant la plupart des solutions qui sont à la fois récentes et pertinentes. L'analyse de sécurité concerne la résistance aux attaques décrites dans le chapitre précédent ainsi que les services de sécurité fournis. L'analyse des performances concerne les charges de calcul et de communication introduites par la solution et ceci afin de déterminer la solution la plus gourmande.

Dans ce qui suit, nous présentons ces solutions en les classifiant en deux catégories *les solutions préventives,* et *les solutions curatives* (Voir Figure 8). Une solution préventive empêche les attaquants d'accéder au réseau en utilisant la cryptographie ou ce que l'on appelle la première ligne de défense. Une solution curative conduit à une guérison du réseau c'est-à-dire que l'attaque est déjà exécutée lorsqu'elle est détectée.

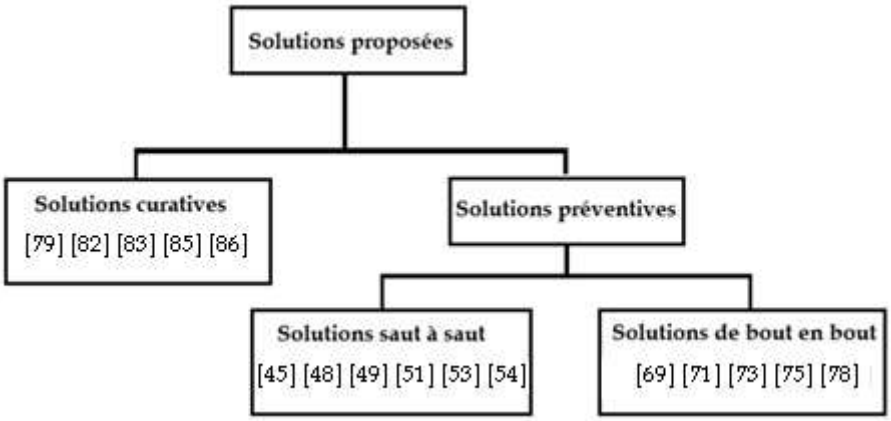


Figure 8. Classifications des solutions

Pour la cryptographie, il existe deux types de chiffrements, symétrique et asymétrique. Avant d'aborder les solutions, nous allons décrire brièvement ces deux types de chiffrement:

- Chiffrement symétrique:

Ce type de chiffrements considère que les deux nœuds impliqués dans la communication, par exemple N_1 et N_2, partagent une clé secrète, par exemple K_{12}. Cette clé permet aux deux nœuds à la fois de chiffrer et de déchiffrer les données:

Chiffrement: $E : K \times M \rightarrow C$, $c = E(K_{12}, m)$

Déchiffrement: $D : K \times C \rightarrow M$, $m = D(K_{12}, c)$

Où K est l'ensemble de toutes les clés possibles, M est l'ensemble de tous les messages possibles, C est l'ensemble de tous les chiffrés possibles, m et c sont respectivement le message choisi et le chiffré correspondant. Les algorithmes symétriques les plus utilisés sont AES [36], RC5 [37] et Skipjack [38].

- Chiffrement asymétrique:

Ce type de chiffrements utilise une paire de clé (K_{pr}, K_{pu}), c'est-à-dire, respectivement la clé privée et la clé publique. En utilisant la clé publique chaque nœud peut chiffrer des données qui ne peuvent être déchiffrées par la suite que par la clé privée correspondante:

Chiffrement: $E : K \times M \rightarrow C$, $c = E(K_{pu}, m)$

Déchiffrement: $D : K \times C \rightarrow M$, $m = D(K_{pr}, c)$

Où K, M, C, m et c ont les mêmes définitions que précédemment.

Les algorithmes asymétriques sont basés sur des problèmes mathématiques difficiles à résoudre, tels que la factorisation ou le calcul du logarithme discret. En considérant cette difficulté, la sécurité de ces algorithmes a été prouvée. Les algorithmes les plus connus sont: *RSA* [39], l'algorithme ElGamal [40], l'algorithme d'échange de clé *Diffie-Hellman* [41] et plusieurs techniques basées sur la cryptographie sur les courbes elliptiques ou en anglais *Elliptic Curve Cryptography* (*ECC*) [42]. Ce type de chiffrements est différent de celui du *RSA* puisqu'il peut atteindre le même niveau de sécurité avec une clé réduite; une clé de 160 bits

de l'*ECC* offre le même niveau de sécurité que celle du *RSA* avec 1024 bits (voir Tableau 2) [43]. Les détails de ce type de chiffrements seront fournis dans les prochains chapitres.

Le chiffrement asymétrique a un avantage considérable par rapport au chiffrement symétrique en termes de gestion de clés. En effet, le chiffrement symétrique requiert la même clé à chaque paire de nœud; par conséquent, si nous avons à sécuriser les communications dans un réseau qui comprend *N* nœuds, le nombre total de paire de clés nécessaires est de *N(N-1)/2*. Le chiffrement asymétrique n'aura besoin que de *N* clés publiques dans un tel réseau, conduisant ainsi à une réduction considérable de la charge (*overhead*). Par contre, le chiffrement symétrique est beaucoup moins gourmand par rapport au chiffrement asymétrique. Ceci veut dire que le chiffrement asymétrique a besoin de plus de ressources et d'énergie; ce qui constitue la raison principale de la difficulté de réaliser ce type de chiffrement en pratique, surtout dans des dispositifs à ressource limitées tels que des capteurs MicaZ et TelosB [2].

Symmetric Key Size	RSA Key Size	ECC Key Size
80	1024	160
112	2048	224
128	3072	256
192	8192	384
256	15360	512

Tableau 2. Comparaison des tailles des clés en bits

2.1. Les solutions préventives

On peut classer les solutions préventives en deux sous catégories à savoir *les solutions saut à saut* et *les solutions de bout en bout* (Voir Figure 8.). Dans la première catégorie, la cryptographie est appliquée saut à saut c'est-à-dire les services de sécurité sont vérifiés dans chaque étape, les nœuds intermédiaires déchiffrent chaque message reçu et avant de chiffrer ils calculent l'agrégat. Cette méthode permet une implémentation plus simple des fonctions d'agrégation, et elle n'impose aucune limite sur leur nature (la somme, la moyenne, la variance, le maximum, le minimum, etc.) et les deux types de chiffrements peuvent être utilisés. Dans les solutions saut à saut, l'agrégateur effectue la fonction d'agrégation sur les messages en clair et donc les données sont exposées à l'attaquant au niveau de chaque nœud agrégateur. Aussi, ces solutions encourent un délai important et ceci est dû à l'effort de chiffrement/déchiffrement effectué par les nœuds intermédiaires. Ces problèmes ont été résolus par les solutions de bout en bout basées sur une propriété spéciale des algorithmes de chiffrement (appelée *privacy homomorphism* [44]), permettant le calcul direct (addition et/ou multiplication) sur les données chiffrées. L'avantage principal de ces solutions est qu'elles permettent de fournir une confidentialité de bout en bout et elles n'ont pas besoins d'effectuer des opérations cryptographiques au niveau des nœuds intermédiaires, et jusqu'à maintenant seulement deux opérations sont possibles à savoir l'addition et la multiplication. Dans ce qui suit, quelques unes des principales propositions pour sécuriser l'agrégation sont présentées, tout en identifiant les avantages et les inconvénients de chaque solution.

2.1.1. Les solutions saut à saut

La solution de Przydatek et al.

Przydatek et al. [45] ont proposé une agrégation sécurisée de l'information. C'est l'une des premières solutions proposées par la communauté scientifique. Cette solution vise à calculer d'une manière sécurisée la fonction d'agrégation (moyenne, médiane, minimum, maximum, etc.), de telle sorte que non seulement l'utilisateur peut accepter les données avec une grande probabilité si le résultat agrégé est une bonne approximation du résultat réel, mais aussi l'utilisateur peut détecter l'attaque avec une forte probabilité et rejeter le résultat s'il est en dehors de la limite. Les auteurs considèrent l'attaque furtive «*Stealthy attack*» dans laquelle le but de l'attaquant est de tromper l'utilisateur en acceptant de faux résultats. En d'autres termes, l'utilisateur accepte des résultats qui sont significativement différents des vrais résultats déterminés par les valeurs mesurées.

Les auteurs proposent l'approche «*Aggregate-commit-prove*» dans laquelle l'agrégateur non seulement agrège les données mais aussi prouve qu'il effectue cette tâche correctement. Dans la première phase, l'agrégateur collecte les données à partir des nœuds capteurs puis calcule localement l'agrégat correspondant. Dans la seconde phase, l'agrégateur s'engage à fournir une preuve assurant que l'agrégat représente bien les donnés collectées, cette preuve sera vérifiée par le serveur. Pour cela, les auteurs suggèrent d'utiliser l'arbre de hachage de Merkle [46], un exemple est fourni dans la Figure 9. Dans la dernière phase, l'agrégateur et le serveur s'engagent dans un protocole dans lequel l'agrégateur communique l'agrégat et la preuve au serveur, puis prouve que le résultat rapporté est exact à l'aide d'un système de preuve interactive. Ce système est composé de deux étapes: (i) le serveur vérifie si l'agrégat est une bonne représentation des vraies valeurs dans le réseau et (ii) le serveur vérifie si l'agrégateur a triché dans le cas où l'agrégat est différent du résultat correct.

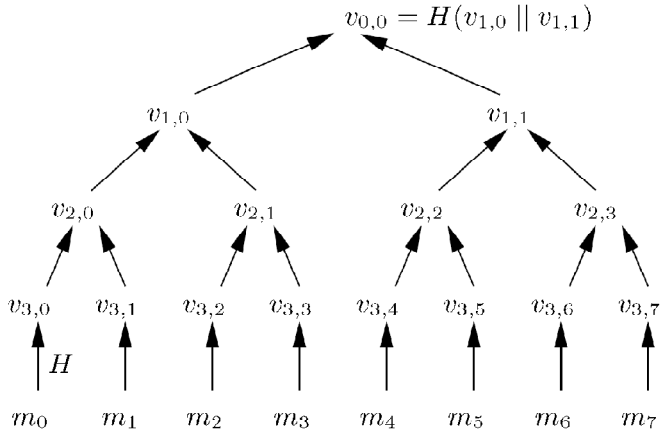


Figure 9. Arbre de hachage de Merkle

Dans l'exemple présenté dans la Figure 9, l'agrégateur construit l'arbre de hachage de Merkle à partir des mesures collectées $m_0,\ldots,m_7$. Pour diminuer la taille de l'information à vérifier, l'agrégateur commence par hacher les mesures avec une fonction de hachage cryptographique, par exemple, $V_{3,0} = H(m_0)$, en supposant que la taille du hash est plus petite que la taille des données. Pour construire l'arbre de hachage de Merkle, chaque valeur

interne de l'arbre est dérivée de ses deux nœuds enfants: $V_{i,j} = H(V_{i+1,2j} \| V_{i+1,2j+1})$. Le hash racine représente un résumé de tous les hash des mesures collectées, et étant donné ce résumé $V_{0,0}$, un vérificateur peut authentifier n'importe quelle valeur feuille en vérifiant que cette dernière est bien utilisée pour calculer le nœud racine $V_{0,0}$. Par exemple, pour authentifier la mesure m_5, l'agrégateur envoie m_5 avec $v_{3,4}$, $v_{2,3}$, $v_{1,0}$, et m_5 sera authentique que si l'égalité suivante est vérifiée: $V_{0,0} = H(V_{1,0} \| H(H(V_{3,4} H(m_5))\|V_{2,3}))$.

Les auteurs considèrent l'attaque furtive dans laquelle l'attaquant peut corrompre quelques nœuds et ensuite essaie de fausser les résultats. Cette solution est robuste contre ces attaques. Elle est aussi robuste contre les attaques sybilles puisque chaque nœud a son propre identifiant et une clé secrète temporaire. Cette clé assure la sécurité contre les attaques par rejeu. Cependant, un nœud compromis peut ignorer quelques messages. Ainsi, cette solution est vulnérable à l'attaque par envoi sélectif des paquets. La confidentialité des données n'a pas été abordée par les auteurs et ils n'ont utilisé aucun algorithme de chiffrement. Cependant, l'authentification et l'intégrité des données sont assurées en utilisant le *MAC* et l'arbre de hachage de Merkle. En outre, les auteurs suggèrent l'utilisation du *HMAC* [47]. La fraicheur des données est aussi assurée grâce aux clés dynamiques qui changent après chaque intervalle de temps bien défini.

Le calcul concerne spécialement la fonction de hachage. Des recherches ont montré que *HMAC* peut être efficacement utilisé dans les RCSFs. Cependant, dans cette solution, l'agrégateur aura à effectuer plus de calcul qu'un capteur normal (Merkle). Dans la première phase, chaque nœud envoie sa donnée captée et le *MAC* correspondant à l'agrégateur. Dans la deuxième phase, l'agrégateur envoie l'agrégat et le résumé «*the commitment*» issue de l'arbre de Merkle ainsi que le nombre de nœuds participants à l'agrégat. Par conséquent, le nombre total de bits transmis dans le réseau est $n(m+c+h) + (m+c+i+h)$, où n est le nombre de nœuds participants, m est la taille en bit de la donnée captée (on suppose aussi que l'agrégat est de la même taille que les données), c est la taille du *MAC* (on suppose aussi la même taille pour le résumé) et i, h représentent respectivement la taille en bit de n et du header du paquet.

La solution de Du et al.

Du et al. [48] ont proposé une fusion de données sécurisée en utilisant des témoins «*Witnesses*». Cette solution vise à résoudre le problème de sécurité entre les nœuds d'agrégation et la station de base. C'est une solution dans laquelle des nœuds autour de l'agrégateur sont sélectionnés pour être des témoins afin de surveiller les agrégats (Voir Figure 10). La méthode proposée garantit la validité des résultats de fusion de données reçus par la station de base.

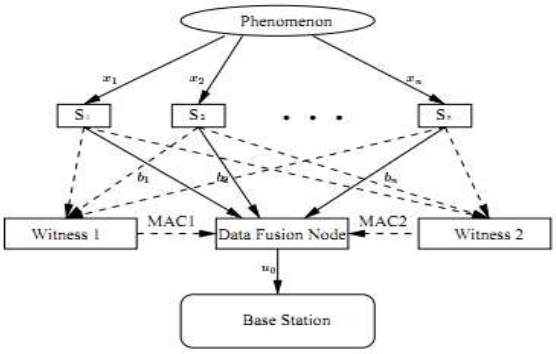


Figure 10. Modèle réseau de la solution de Du et al. [48]

Les auteurs suggèrent que l'agrégateur «*data fusion node*» doit fournir des preuves à partir de plusieurs témoins. Tout comme un agrégateur, un témoin est un nœud qui effectue également la fusion de données, mais ne transmet pas son résultat à la station de base. Chaque témoin calcule le *MAC* correspondant au résultat (l'agrégat) et puis transmet ce *MAC* (la preuve) à l'agrégateur qui, par la suite, doit fournir ces preuves à la station de base. Si un agrégateur est compromis et veut envoyer un agrégat invalide à la station de base, il devra modifier les preuves correspondantes. Les auteurs indiquent que si au moins *n* des (*m+1)* (*m* témoins et un agrégateur) *MACs* correspondent, le résultat est accepté sinon, le résultat est rejeté parce qu'il n'est pas approuvé par au moins *(n-1)* des témoins.

Les auteurs considèrent les attaques contre l'agrégateur et les témoins. Cependant, ils supposent qu'il y a au moins (*n-1)* témoins honnêtes sur les *m* témoins de l'agrégateur. Cette solution est vulnérable à plusieurs attaques. En effet, un attaquant peut affecter l'agrégat en s'emparant de quelques nœuds et utiliser leurs identités parce que l'authentification n'est pas assurée entre les nœuds feuilles et l'agrégateur «attaque sybille». Un agrégateur compromis peut choisir de transmettre l'agrégat et les preuves ou non «envoie sélectif des paquets». Il peut aussi retransmettre des agrégats valides avec les preuves correspondantes pour tromper la station de base «attaque par rejeu». Toutefois, la sécurité contre les attaques furtives est assurée (mais partiellement). La confidentialité des données n'a pas été abordée par les auteurs, ils n'ont utilisé aucun algorithme de chiffrement, tout comme la fraicheur et l'authentification des nœuds feuilles. Cependant, l'authentification et l'intégrité de l'agrégat sont assurées grâce aux preuves fournies par l'agrégateur.

Le calcul concerne spécialement les preuves (*MAC*) fournies. Le calcul s'effectue seulement au niveau de l'agrégateur et les témoins correspondants. Aucune primitive de sécurité n'est prise en charge par les nœuds feuilles. Les auteurs considèrent que les nœuds feuilles sont honnêtes, donc les données reçues par l'agrégateur et les témoins sont supposées être valides. Ainsi, chaque nœud envoie seulement sa donnée *m*, et cette dernière est envoyée à l'agrégateur et aux *w* témoins correspondants $n((m+h)(w+1))$. Ensuite, chaque témoin calcule l'agrégat et envoie le résultat ainsi que la preuve à l'agrégateur $(w(m+c+h))$. Enfin, l'agrégateur envoie l'agrégat et toutes les preuves ainsi que la sienne à la station de base $((m+(w+1)c+h))$. Donc, le nombre total de bits transmis dans le réseau est : $n((m+h)(w+1)) + (w(m+c+h)) + ((m+(w+1)c+h))$.

La solution de Mahimkar et al.

Mahimkar et al. [49] ont proposé *SDAV* (*Secure Data Aggregation and Verification*), une agrégation sécurisée avec vérification. Les auteurs utilisent la cryptographie sur les courbes elliptiques *(ECC)*, en raison de sa clé réduite et son efficacité en termes de calcul et de bande passante par rapport aux algorithmes asymétriques traditionnels comme *RSA* [39]. En effet, l'*ECC* permet à un capteur de calculer efficacement une signature alors que la vérification (qui est plus complexe) est dévolue à la station de base (qui n'a pas généralement les limitations d'un capteur). Tout comme [45], les auteurs considèrent l'attaque furtive. Un protocole de gestion de clés (*CKE*) et un autre de vérification des données agrégées ont été proposés (*SDAV*).

Le premier protocole génère une clé secrète pour chaque groupe de nœuds «*Cluster*», chaque nœud possède une clé partielle de ce secret afin de générer une signature partielle en utilisant l'*ECDSA* (*Elliptic Curve Digital Signature Algorithm*) [50]. L'agrégateur «*Cluster Head*» collecte toutes ces signatures et les combine en une signature complète et l'envoie ainsi que l'agrégat à la station de base. Cette dernière, qui connaît les clés publiques correspondantes, peut vérifier l'authenticité. Un attaquant, qui ne connaît pas la clé du groupe, ne peut donc générer la signature globale. En effet, les auteurs ont prouvé que la corruption de t nœuds d'un cluster (où $t < n/2$, n représente le nombre total de nœuds par cluster) ne permet pas à l'attaquant de générer cette signature. Une fois ces clés générées, ils seront utilisées pour assurer la sécurité des données agrégées. Dans *SDAV*, l'agrégateur collecte les données cryptées de ses membres, décrypte, calcule la moyenne puis renvoie le résultat à ses membres, chacun de ces derniers compare sa mesure avec la moyenne reçue, si la différence dépasse un certain seuil, il génère une signature partielle en utilisant la clé générée pendant le *CKE*, et l'envoie à l'agrégateur. L'agrégateur génère et envoie la signature complète à la station de base. En cas de rejet, l'intégrité des mesures est vérifiée en utilisant l'arbre de hachage de Merkle, similairement à [45]. La solution n'accepte que la moyenne comme fonction d'agrégation.

Cette solution est robuste contre les attaques furtives. Elle est robuste aussi contre les attaques sybilles puisque chaque nœud a son propre identifiant et partage une clé secrète avec le *CH*. Cependant, cette solution est vulnérable à l'attaque par envoi sélectif des paquets et aussi à l'attaque par rejeu. En effet, un agrégateur compromis peut ignorer des paquets. Aussi, les auteurs n'ont pas considéré la dynamicité des clés. La confidentialité des données est assurée puisque tous les messages sont chiffrés, tout comme l'authentification et l'intégrité en utilisant la signature. Cependant, la fraicheur des données n'est pas assurée, un attaquant peut envoyer des paquets déjà envoyés.

Le calcul est très considérable. En effet, les services de sécurité sont assurés à travers l'*ECC*, qui est un chiffrement asymétrique et qui introduit beaucoup plus de calcul que le symétrique mais moins que *RSA*. Dans le chapitre suivant, nous reviendrons sur l'efficacité de ce type de chiffrements dans les RCSFs. Les auteurs considèrent que les nœuds feuilles envoient à l'agrégateur leurs messages cryptés *mc* et aussi le hash *s*, soit *n(mc+s+h)*, ensuite l'agrégateur envoie (*broadcast*) l'agrégat crypté à ses membres *mc+s+h*, une fois reçu, les capteurs génèrent la signature partielle, soit *n(s+h)*, enfin l'agrégateur combine et envoie

l'agrégat crypté et la signature complète, soit *mc+s+h*. Le nombre total de bit transmis dans *SDAV* est donc *((n+2)(mc+s+h))+n(s+h)*.

La solution d'Ozagur et al.

Ozagur et al. [51] ont proposé *SRDA* (*Secure Reference based Data Aggregation*), une agrégation sécurisée basée sur les références. Les auteurs suggèrent l'utilisation des références, c'est-à-dire au lieu d'envoyer la valeur captée, le capteur envoie la différence entre cette valeur et la valeur référence. Et cela afin de réduire le nombre de bits transmis dans le réseau. Les auteurs ont indiqué que les messages interceptés par les niveaux supérieurs représentent un résumé de plusieurs transmissions des niveaux inferieurs. Par conséquent, ils pensent que le niveau de sécurité doit augmenter graduellement. Pour cela, un algorithme de chiffrement spécifique a été utilisé.

Les auteurs suggèrent qu'au lieu d'envoyer les données captées, un nœud capteur fait la différence entre sa donnée et la valeur référence[2]. Par exemple, la température captée par un nœud est 95° et la température référence est 90°, le nœud n'aura besoin d'envoyer que la différence qui est 5° à l'agrégateur; ce qui réduit ainsi le nombre de bits transmis dans le réseau. Ensuite, il envoie le chiffré du résultat à l'agrégateur. Ce dernier déchiffre et calcule l'agrégat, ensuite il détermine la distance par rapport à la station de base en nombre de sauts *S*. Enfin il chiffre l'agrégat en utilisant *RC6* avec le nombre de rounds *R* et envoie le résultat à la station de base, *R* est calculé comme suit :

$$R = 1+(1/S) * 100$$

Le nombre de round qui est nécessaire à *RC6* pour effectuer un chiffrement, est ajusté en fonction de la position de l'agrégateur dans la hiérarchie par rapport à la station de base, plus l'agrégateur est proche de la station de base (*S* diminue), plus le nombre *R* augmente.

Cette solution est robuste contre les attaques passives, puisque tous les messages sont chiffrés. Cependant, elle est vulnérable aux attaques actives parce qu'elles n'ont pas été considérées par les auteurs, sauf l'attaque par rejeu parce que les clés sont mis à jour. La confidentialité des données est assurée puisque tous les messages sont chiffrés, la fraicheur aussi puisque il y a une mise à jour des clés. Cependant, l'intégrité et l'authentification ne le sont pas.

Le calcul concerne le chiffrement symétrique en utilisant *RC6*. Chaque nœud utilise cet algorithme pour chiffrer ses données. Cependant, le temps d'exécution des opérations augmente avec le nombre de rounds *R*. Les auteurs de [52] ont étudié la faisabilité des chiffrements par bloc dans les RCSFs et ils ont montré que *RC6* n'est pas efficace en termes d'énergie et de mémoire pour un capteur. Cette solution est efficace en termes de communication, puisque l'objectif principal est de réduire le nombre de bits transmis.

La solution de Vu et al.

Vu et al. [53] ont proposé *THIS* (*THreshold security for Information aggregation in Sensor networks*), une sécurité à seuil pour l'agrégation. Ce protocole permet à la station de base

[2] La valeur référence représente la moyenne de toutes les mesures précédentes N, tel que N > 1.

d'accepter l'agrégat seulement si la majorité des nœuds envoient des données similaires. Plus précisément, les données générées ou forgées collectivement par *t* capteurs compromis seront rejetées, où *t* est le seuil représentant le nombre de nœuds compromis considéré par le protocole dans le réseau. Le protocole est composé de trois phases: une phase pour la gestion des clés, une deuxième pour la sécurité de l'agrégation et une dernière pour l'authentification.

Dans la première phase, chaque nœud partage une clé avec l'agrégateur correspondant, une fois que ce dernier aura une clé avec chacun de ces membres, il fait un *broadcast* d'une clé de groupe temporaire pour tous ses membres. Dans la deuxième phase, chaque nœud capte, chiffre puis transmet sa donnée à l'agrégateur, ce dernier calcule et chiffre l'agrégat en utilisant la clé de groupe, puis fait un *broadcast* du chiffré à tous ses membres. Ces derniers comparent l'agrégat reçu avec leurs valeurs captées, si la différence n'est pas signifiante, ils génèrent une signature *MAC* en utilisant la clé partagée avec la station de base, sinon ils refusent de la générer. L'agrégateur transmet à la station de base le chiffré d'un message contenant l'agrégat et toutes les signatures. La station de base déchiffre et vérifie l'agrégat. Si au moins $t + 1$ signatures sont conformes avec la valeur finale, alors l'agrégat sera accepté, sinon il sera rejeté.

Cette solution est similaire à [49], avec une différence concernant le type de chiffrements utilisé. Elle est robuste contre les attaques passives puisque tous les messages sont chiffrés, et aussi contre l'attaque par rejeu car les clés de groupes sont mises à jour. Cette solution est également sécurisée contre l'attaque sybille puisque les nœuds sont authentifiés par la station de base. Cependant, elle reste vulnérable aux attaques par envoi sélectif des paquets.

La confidentialité des données est assurée puisque tous les messages sont chiffrés. La fraicheur l'est aussi, puisque il y a une mise à jour de la clé du groupe et l'authentification est assurée en utilisant le *MAC*. Cependant, l'intégrité des données individuelles n'est pas assurée puisque l'attaquant peut facilement modifier les données entre les nœuds feuilles et l'agrégateur.

Les auteurs de la référence [53], utilisent un algorithme de chiffrement symétrique sans préciser sa nature. Cependant, même si ce type de chiffrement est efficace par rapport au chiffrement asymétrique, l'effort de chiffrement/déchiffrement aux niveaux des agrégateurs peut conduire à l'épuisement de leurs batteries. Les auteurs considèrent que les nœuds feuilles envoient à l'agrégateur leurs messages cryptés *mc*, ensuite l'agrégateur envoie (*broadcast*) l'agrégat crypté à ses membres $mc+h$, une fois reçu, les capteurs génèrent le *MAC*, soit $n(c+h)$, enfin l'agrégateur envoie le tout ainsi que l'agrégat. Les auteurs supposent que ce message est 10% plus grand qu'un message normal, soit $1{,}1*q*(mc+c+h)$, pour tous les nœuds *q* intermédiaires jusqu'à la station de base. Le nombre total de bits transmis dans *THIS* est donc $n*(mc+h)+ n(c+h)+ 1{,}1*q*(mc+c+h)$.

La solution de Yang et al.

Yang et al. [54] ont proposé *SDAP* (*Secure hop-by-hop Data Aggregation Protocol*), un protocole de sécurité saut à saut pour l'agrégation des données. Cette solution essaie de répondre à la question: comment la station de base peut obtenir une bonne approximation de l'agrégat en

présence de quelques nœuds compromis? Les auteurs proposent diverses solutions pour des problèmes collatéraux impliqués dans la sécurité de l'agrégation tels que la construction de l'arbre et la distribution des clés. Ils se basent sur l'idée que dans une approche hiérarchique, les nœuds placés près de la station de base calculent un agrégat qui concerne un nombre considérable de capteurs, et une fois compromis, sa donnée erronée aura plus d'impact que d'autres nœuds du réseau. Pour éviter de faire plus de confiance à ce type de nœuds, le système est composé de deux phases: (i) diviser et régner «*divide-and-conquer*» et (ii) garantir et certifier «*commit-and-attest*».

Dans la première phase, le système utilise une approche probabiliste qui divise l'arbre en groupes de même taille, et un chef est élu dans chaque groupe. Dans la deuxième phase, une agrégation est effectuée dans chaque groupe. La station de base identifie les groupes suspects sur la base d'un ensemble d'agrégats de groupes. Chaque groupe soupçonné participe à un processus d'attestation pour prouver la validité de son agrégat. Elle commence par authentifier les chefs de groupes, en vérifiant que le *MAC* reçu provient bien du chef supposé, ensuite elle utilise le test de Grubbs [55] pour détecter les groupes suspects. Comme dans [45], la station de base interagit avec le groupe soupçonné afin de prouver la validité de son agrégat.

Cette solution est robuste contre les attaques passives puisque tous les messages sont chiffrés, contre l'attaque par rejeu aussi due à l'utilisation d'un nombre de séquence. Cette solution est sécurisée contre l'attaque sybille puisque les nœuds sont authentifiés par la station de base via *µTESLA*. Cependant, elle est vulnérable aux attaques par envoie sélectif des paquets. La confidentialité des données est assurée puisque tous les messages sont chiffrés, en utilisant le *MAC*, l'authentification et l'intégrité le sont aussi. Le nombre de séquences garantit la fraicheur.

Le calcul concerne les primitives symétriques exécutées avant chaque transmission. Cependant, l'effort de chiffrement/déchiffrement aux niveaux des nœuds intermédiaires peut conduire à une consommation d'énergie considérable. Les auteurs considèrent que les nœuds feuilles envoient à l'agrégateur leurs messages cryptés *mc* et les *MACs* correspondants *c*, ensuite l'agrégateur envoie un paquet similaire à la station de base, soit *mc+c+h*. Le nombre total[3] de bits transmis dans *SDAP* est donc *n(mc+c+h)*.

2.1.2. Les solutions de bout en bout

Avant d'aborder les solutions proposées, une brève description du Chiffrement Homomorphique *(C.H)* est présentée.

Un algorithme de chiffrement E ( ) est dit homomorphique s'il vérifie l'équation suivante:

$$D\ (E(x)\ \Diamond_c\ E(y)) = D\ (E(x\ \Diamond_m\ y))$$

Où les opérations sur les groupes $\Diamond_c$ et $\Diamond_m$ sont effectuées respectivement dans le groupe des chiffrés *C* et le groupe des messages en clair *M* [44]. L'équation précédente implique que le chiffrement homomorphique est effectué sur les données chiffrées, dans laquelle la somme

[3] Ce nombre concerne uniquement la phase d'agrégation. Après vérification, les agrégats invalides peuvent conduire à un coût de communication très important.

des chiffrés de deux données est égale à la version chiffrée de la somme de ces deux données. Le chiffrement homomorphique peut être additif et/ou multiplicatif. Le *C.H* qui prend en charge n'importe quelle fonction sur les chiffrés est un Chiffrement Complètement Homomorphique *(C.C.H)*, et qui constitue l'avancée la plus importante en cryptographie au cours des dernières années, d'abord introduite par Gentry [56]. Le *C.C.H* est prometteur, mais la complexité en temps de ses algorithmes est encore trop élevée pour une utilisation pratique. L'autre classe des *C.H* est le Chiffrement Partiellement Homomorphique *(C.P.H)*, qui comprend les systèmes de chiffrement permettant une seule fonction sur les chiffrés.

Le procédé (*C.H*) a été introduit pour la première fois en 1978 par les chercheurs Rivest, Adleman et Dertouzos pour réponde à la fameuse question: «*Can we do arbitrary computations on data while it remains encrypted, without ever decrypting it ?*» [57]. Cette question a donné la possibilité fantastique d'effectuer des calculs sur des données cryptées sans être en mesure de les «voir». Dans leur article, ils proposent les fonctions d'exponentiation et *RSA* respectivement comme homomorphisme additif et multiplicatif. Depuis cette date, plusieurs applications s'intéressent à ce type de chiffrement comme, l'agrégation des données dans les RCSFs [58,59], le vote électronique [60], la biométrie [61], le réseau intelligent de distribution d'électricité (*smart grid*) [62] ou encore la confidentialité dans l'exploration de données [63]. Plusieurs *C.P.H* ont été proposés [64-71], des *C.P.H* qui acceptent soit l'addition soit la multiplication. En 2005, Boneh et al [72] ont proposé un *C.P.H* permettant une multiplication suivie de plusieurs additions. Le Tableau 3 illustre une comparaison entre ces *C.P.H*. Ensuite, l'année 2009 a vu l'avènement de la percée théorique de Gentry en proposant le premier *C.C.H* non cassé. Cependant, il reste inefficace en pratique à cause du coût très important des calculs. Depuis, l'implémentation du *C.C.H* a fait l'objet de plusieurs recherches.

C.P.H	**Homomorphisme**	**Calcul**
Goldwasser Micali [64]	Xor	Exp.mod dans Z_{pq}
Benaloh [65]	Additif	Exp.mod dans Z_{pq}
Naccache-Stern [66]	Additif	Exp.mod dans Z_{pq}
Pailler [67]	Additif	Exp.mod dans $Z_{(pq)^2}$
Okamoto-Uchiyama [68]	Additif	Exp.mod dans Z_{p^2q}
El Gamal [69]	Multiplicatif	Exp.mod dans GF(p)
Elliptic Curve El Gamal [58]	Additif	Mult. scalaire (Courbes elliptiques)
Castelluccia [71]	Additif et multiplicatif	Operations modulo M
Boneh-Goh-Nissim [72]	Formules 2-DNF	Exp.mod dans $Z_{(pq)^2}$, Groupes bilinéaires

Tableau 3. Comparaison entre les C.P.H

Dans les RCSFs, la communauté scientifique se penche sur la première classe dûe à l'efficacité des calculs. En général, le *C.P.H* symétrique est plus efficace en termes de calcul, mais la plupart des solutions ont été cryptanalysées. La solution proposée dans [71] est la seule qui n'a pas été cassée [44]. Le *C.P.H* asymétrique a été utilisé [58] pour combler les lacunes du chiffrement symétrique, mais l'inconvénient principal de ces solutions demeure le temps de calcul très important et qui conduit à une consommation dramatique de l'énergie. Ce qui a fait de ces solutions, des solutions impraticables pour plusieurs applications. Dans ce qui suit, nous allons présenter quelques solutions basées sur ce type de chiffrement.

La solution de Castelluccia et al.

Castelluccia et al. [71] ont proposé une agrégation efficace des données chiffrées. Cette solution vise à fournir l'efficacité et la confidentialité de bout en bout dans les RCSFs, plus précisément, un mélange de techniques de chiffrements efficaces et de méthodes d'agrégation très simples est utilisé pour réaliser cet objectif. Les auteurs proposent un chiffrement homomorphique additif tout en fournissant une preuve de sécurité. L'idée de base est de remplacer l'opération Xor[4] (traditionnellement utilisée dans le chiffrement par flux) par une simple addition modulaire.

Ce nouveau C.P.H est défini dans l'Algorithme 1 suivant :

Algorithme 1: Le C.P.H de Castelluccia et al. [71]

Chiffrement :

1. Représenter le message m dans l'intervalle [0, M-1], où M est un nombre très grand.
2. Générer une clé de chiffrement k, où $k \in [0, M-1]$
3. $c=Enc(m,k,M)= m+k \pmod M$

Déchiffrement :

1. $Dec(c,k,M)=c-k \pmod M$

Addition des chiffrés :

1. Soit, $c_1=Enc(m_1,k_1,M)$ et $c_2=Enc(m_2,k_2,M)$
2. Pour $k=k_1+k_2$, $Dec(c_1+c_2,k,M)=m_1+m_2$

Les nœuds feuilles chiffrent leurs messages et envoient le résultat à l'agrégateur. Ce dernier calcule directement sur les chiffrés en effectuant une addition modulaire. De cette manière, non seulement les agrégateurs n'ont pas besoins de déchiffrer pour agréger les données mais aussi ils n'ont pas accès aux données en clair, ce qui assure une confidentialité de bout en bout. Les auteurs considèrent que la clé *k* est générée en utilisant un chiffrement par flux tel que RC4, la clé secrète partagée avec la station de base et un identifiant du message. La station de base par la suite aura besoin de toutes les clés utilisées lors du chiffrement c'est-à-dire correspondant à tous les participants à l'agrégat. Les auteurs suggèrent que chaque nœud ajoute son identifiant avec le chiffré.

Cette solution est robuste contre les attaques passives, la sécurité contre les attaques actives n'a pas été considérée. Le système proposé est probabiliste[5]. Cette caractéristique rend la cryptanalyse compliquée puisque la mesure captée est couverte par la valeur aléatoire injectée. Cependant, la source de cette valeur constitue la plus grande menace contre l'analyse des chiffrés. En effet, s'il est possible de prédire la valeur aléatoire suivante, cela impliquerait une rupture totale du système. La sécurité contre l'attaque par rejeu est aussi assurée puisqu'une nouvelle clé est associée à chaque nouveau message. Aussi, le protocole est malléable, un attaquant peut facilement ajouter une valeur à un chiffré, une modification qui ne peut être détectée par le système. Par exemple, un attaquant qui veut augmenter la mesure par 10, n'a qu'à faire ceci:

$$((m+10)+k) \bmod M = ((m+k)+10) \bmod M$$

[4] Xor représente la fonction OU-exclusif.

[5] Si un même message est chiffré plusieurs fois, le résultat sera aléatoire.

La confidentialité et la fraicheur des données sont assurées puisque tous les messages sont chiffrés en utilisant des clés différentes, mais à la différence des solutions saut à saut, la confidentialité est fournie de bout en bout.

Cette solution est efficace en termes de calcul, puisque ce dernier concerne seulement une addition modulaire pour le chiffrement. Les auteurs considèrent que les nœuds feuilles envoient à l'agrégateur leurs messages cryptés *mc* et leurs identifiants *id*, ensuite l'agrégateur envoie à la station de base *mc+(n*id)+h*. Le nombre total de bits transmis est donc *n*(mc+h)+n*id*.

La solution d'Albath et al.

Albath et al. [73] ont proposé une agrégation hiérarchique des données chiffrées dans les RCSFs. Cette solution traite les services de sécurité de bout en bout, la confidentialité est assurée en utilisant l'*ECEG* (*Elliptic Curve El Gamal*), un *C.P.H* asymétrique proposé dans [58], et l'intégrité est fournie par un *ECDSA*[6] modifié. L'objectif est de s'assurer que les données reçues par la station de base proviennent seulement des nœuds participants à l'agrégat.

Dans cette solution, chaque capteur génère une mesure et cette mesure est signée par l'*ECDSA* modifié en utilisant sa clé privée. Cette modification consiste en un remplacement dans l'algorithme original de la fonction de hachage par le message lui-même et ceci afin de permettre l'addition sur les signatures ainsi que la vérification. Ensuite la mesure est chiffrée par l'*ECEG* en utilisant la clé publique de la station de base. Le capteur envoie un paquet à l'agrégateur contenant le chiffré, la signature et la clé publique correspondante. Après réception des messages provenant de ses membres, l'agrégateur combine, en utilisant la propriété homomorphique, les chiffrés, les signatures et les clés publiques, respectivement en un seul chiffré, une seule signature et une seule clé. Ce processus est répété par chaque parent jusqu'à atteindre la station de base. Cette dernière commence par déchiffrer l'agrégat reçu en utilisant sa clé privée, ensuite vérifie la somme des signatures étant donné la somme des clés publiques. L'agrégation des signatures garantit que seules les mesures des capteurs légitimes sont incluses dans l'agrégat.

Cette solution est robuste contre les attaques passives et actives puisque tous les messages sont chiffrés et signés. Cependant, l'inconvénient majeur de cette solution est l'utilisation par tous les nœuds, de la même valeur de *k* pour la signature, ce qui pourrait permettre à l'attaquant de perturber tous le réseau en corrompant un seul nœud. La confidentialité et l'intégrité des données sont assurées de bout en bout, l'agrégateur n'a pas accès aux données en clair, et la vérification se fait seulement au niveau de la station de base.

Cette solution est gourmande en termes de calcul, puisque le chiffrement asymétrique est utilisé. Certes, il a été montré que la cryptographie sur les courbes elliptiques est la plus adaptée aux RCSFs, mais demeure impraticable pour un nombre considérable d'applications. Plus de détails seront fournis dans le troisième chapitre. Les auteurs dans [74] indiquent dans leur implémentation que le paquet à une taille de 107 octets contenant un chiffré (42 octets), une signature (21 octets) et la clé publique correspondante (42 octets) et enfin 2 octets,

[6] ECDSA et ECEG sont les analogues du DSA et le cryptosystème El Gamal sur les courbes elliptiques, respectivement.

un pour indiquer le nombre de nœuds participants à l'agrégat et un autre pour l'adresse source. Le nombre total de bits transmis est donc (*n*(h+107)*8)*.

La solution de Sun et al.

Sun et al. [75] ont proposé une agrégation de données sécurisée et vérifiable au niveau de la station de base. Ces auteurs combinent le chiffrement *ECEG* et un protocole d'agrégation des signatures proposé dans [76]. Tout comme la solution précédente, l'objectif est d'assurer et la confidentialité et l'intégrité de bout en bout. Leurs résultats de simulations et expérimentaux sont présentés dans [77].

Cette solution est composée de quatre phases: *déploiement, chiffrement, agrégation* et *vérification*. Dans la première phase, chaque nœud et préchargé par les clés nécessaires ensuite les nœuds sont déployés et organisés en clusters. Dans la deuxième phase, les nœuds chiffrent et signent leurs mesures et envoient le résultat au *CH* correspondant. Ce dernier effectue la fonction d'agrégation sur les données reçues en exécutant l'opération homomorphique sur les chiffrés et les signatures[7], et envoie les deux résultats à la station de base. La phase finale concerne la récupération et la vérification, les mesures individuelles sont d'abord extraites à partir de l'agrégat puis vérifiées.

Tous les messages sont chiffrés par l'*ECEG*, ainsi le protocole est sécurisé contre les attaques passives. Il est aussi sécurisé contre les attaques actives dans lesquelles l'adversaire tente d'envoyer de faux résultats dans le but de tromper la station de base. L'avantage majeur de cette solution est que la station de base peut identifier le nœud malicieux dans la dernière phase. Les messages sont chiffrés et signés par les nœuds capteurs et sont vérifiés seulement par la station de base, donc la confidentialité et l'intégrité de bout en bout sont bien assurées.

La charge de calcul introduite par cette solution concerne le chiffrement et la signature pour tous les nœuds, et aussi l'opération homomorphique pour l'agrégateur. Pour la charge de communication, les mêmes auteurs indiquent dans leur implémentation [77] que la taille des messages est de 476 bits, dont 161*2=322 bits pour le chiffré et 154 bits pour la signature. Le nombre total de bits envoyés dans le réseau est de *n*(476+h)*.

La solution d'Ozdemir et al.

Ozdemir et al. [78] ont proposé *IPHCDA* (*Integrity Protecting Hierarchical Concealed Data Aggregation*), une sécurité hiérarchique des données agrégées cryptées permettant l'agrégation de données chiffrées par des clés différentes. Dans cette solution, la station de base peut classifier les données chiffrées agrégées en se basant sur les clés du chiffrement, les auteurs utilisent un algorithme basé sur la cryptographie sur les courbes elliptiques proposé dans [72] afin d'assurer la confidentialité et l'intégrité est fournie en utilisant le *MAC*.

Les auteurs considèrent un réseau à plusieurs groupes (Voir Figure 11), chaque groupe est muni d'une clé publique différente des autres groupes pour chiffrer les données. Chaque membre chiffre sa mesure avec la clé publique correspondante à son groupe et l'envoie à son agrégateur. Après la réception de tous les messages provenant des membres de sa région,

[7] L'opération homomorphique consiste en une addition sur la courbe pour les chiffrés, et en une simple addition modulaire pour les signatures.

l'agrégateur agrège les chiffrés selon la propriété homomorphique du chiffrement utilisé [72] et calcule un *MAC* sur le résultat, ce *MAC* est calculé en utilisant la clé qu'il partage avec la station de base. Par la suite, il envoie son paquet à la station de base ou bien l'agrégateur le plus proche. Ce dernier, une fois qu'il reçoit des paquets provenant d'autres agrégateurs, agrège les chiffrés, et exécute la fonction *Xor* pour les *MACs*. Ce processus est répété jusqu'à atteindre la station de base.

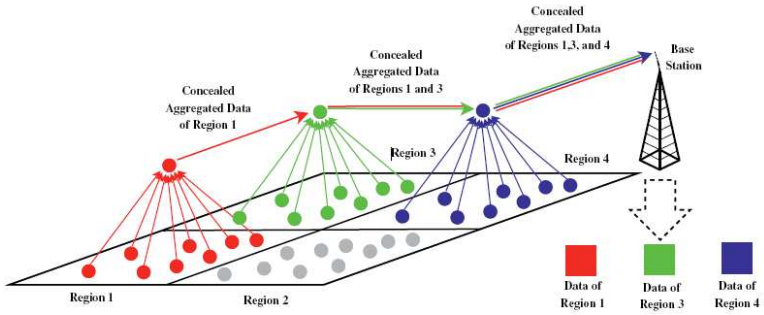


Figure 11. Modèle du RCSF considéré dans [78]

Tous les messages sont chiffrés par l'algorithme de [72], et tout comme l'*ECEG*, ce chiffrement est aussi probabiliste, donc la sécurité contre les attaques passives telles que l'analyse des chiffrés et l'attaque par message clair connu, est assurée. Afin d'empêcher les modifications et les fabrications des agrégats, les signatures *MACs* sont considérées entre les agrégateurs et la station de base. Donc, cette solution assure une confidentialité de bout en bout et une intégrité saut à saut entre les agrégateurs.

Des multiplications modulaires à 1024-bits sont nécessaires pour le calcul, et ce dernier représente 20 fois le calcul de l'*ECEG* si seulement 4 régions sont considérées, ce qui est considérable. Aussi, les messages sont trop longs à cause de l'expansion des chiffrés. Selon [78], la taille des chiffrés est de $(k+1)*q$ avec k le nombre de régions et q le modulo, ce qui conduit à une taille de 1705 bits. Pour la transmission, les auteurs divisent les paquets en blocs et envoie chaque bloc séparément. Enfin, certes, cette solution permet l'agrégation de données chiffrées différentes (chiffrées avec des clés publiques différentes), cependant, l'algorithme utilisé introduit une lourde charge de communication et de calcul. Pour un RCSF, elle est loin d'être la meilleure.

2.2. Les solutions curatives

Ozdemir et al. [79] ont proposé *SELDA* (*SEcure and reLiable Data Aggregation*), une agrégation de données sécurisée et fiable. Les auteurs indiquent que la cryptographie seule ne peut garantir une sécurité suffisante pour les RCSFs, et proposent un réseau de confiance (*web of trust*). L'idée est que chaque nœud capteur observe le comportement de ses voisins pour développer un niveau de confiance. Pour cela, des mécanismes de surveillance sont utilisés pour détecter la disponibilité du nœud et le mauvais comportement des voisins. La méthode utilisée est la *beta reputation function* [80], des informations de confiances sont échangées entre les nœuds pour créer un réseau de confiance qui leur permet de trouver un chemin de confiance vers l'agrégateur. *SELDA* peut détecter si un agrégateur est l'objet d'une attaque du type déni de service. Les auteurs affirment que *SELDA* implique une charge de

communication tolérable, tout en augmentant considérablement la sécurité dans le processus d'agrégation. Une amélioration de *SELDA* dans [81] a été proposée en introduisant le concept de réputations fonctionnelles (*Functional reputations*) qui assure une évaluation plus efficace du comportement des capteurs.

Sedjelmaci et al. [82] ont proposé une solution sécurisée qui consiste en un système de détection d'intrusion basé sur la *SVM* (*Support Vector Machine)*. Dans cette solution, les auteurs ont proposé une détection à plusieurs niveaux, Le premier niveau consiste en un ensemble d'agents *IDS* (*Intrusion Detection System*) qui applique les politiques de détection basées sur des règles pour la modélisation du comportement normal d'un nœud. Dans le niveau intermédiaire, un système de classification binaire à base des *SVM* s'exécute dans chaque cluster-head, ce dernier utilise un système de réputation afin d'évaluer le niveau de confiance de ses agents *IDSs*. Puisque le cluster-head est la cible la plus attrayante pour l'attaquant, les auteurs ont introduit un niveau supérieur. Dans ce niveau chaque cluster-head surveille ses frères voisins, lorsque l'un d'eux présente un comportement malicieux, une alarme est envoyée à la station de base pour une meilleure confirmation du caractère malicieux du nœud soupçonné. Les résultats de simulation montrent que leur système est efficace en termes de taux de détection, temps de détection et consommation d'énergie.

Sun et al. [83] ont proposé une agrégation sécurisée à travers un système de détection d'intrusion et un système de surveillance, le premier pour détecter si les nœuds surveillés sont malicieux et le deuxième pour surveiller les événements d'urgence. *Le filtre de kalman étendu* (*KEF*) a été proposé afin de prédire et estimer les futurs comportements des voisins. Aussi, grâce à KEF, un mécanisme de détection local, capable de faire la différence entre les événements urgents et les événements malicieux, est créé. Par conséquent, le taux de fausses alarmes sera réduit. Les auteurs ont montré que la solution proposée permet de détecter les injections des fausses données. Plus de résultats ont été fournis par la suite dans [84].

Gursel et al. [85] ont proposé une solution pour surveiller les nœuds d'agrégation. Ces auteurs combinent les statistiques avec l'Intelligence Artificielle (*IA*), une combinaison dans laquelle un nœud peut détecter si l'agrégateur transmet des agrégats incorrects. Pour cela, les auteurs ont développé un mécanisme qui permet aux nœuds d'enregistrer un score de réputation pour leurs parents respectifs. Chaque nœud connait seulement sa valeur et la valeur agrégée par l'agrégateur, et en utilisant des statistiques chaque nœud calcule ensuite la probabilité que l'agrégateur transmet un agrégat correct. Cette probabilité est utilisée pour mettre à jour la fiabilité du nœud parent en utilisant un algorithme d'apprentissage.

Labraoui et al. [86] ont proposé une agrégation sécurisée basée sur une surveillance hiérarchique et adaptative. La solution est basée sur deux principes: agrégation indépendante et exactitude. Les auteurs utilisent deux niveaux de surveillance pour assurer l'intégrité et l'exactitude du résultat de l'agrégation. Dans le premier niveau un capteur, jouant le rôle de superviseur principal (*PSUP_L1*), surveille le comportement de l'agrégateur. Dans le second niveau, les autres capteurs membres jouent le rôle de superviseurs secondaires et surveillent l'agrégateur et *PSUP_L1*. Un deuxième superviseur principal (*PSUP_L2*) est utilisé pour gérer la surveillance dans le second niveau. La solution n'introduit aucune charge additionnelle dans une situation normale. Cependant, cette charge pourrait être très considérable dans la présence d'une attaque contre le second niveau.

3. Discussion

Dans cette section, nous allons comparer entre les solutions étudiées précédemment tout en fournissant une analyse qui nous permettra de tirer des conclusions pertinentes. Nous analysons la sécurité de ces solutions et les différentes charges (Communication et calcul).

3.1. Sécurité

3.1.1. Services de sécurité

Chaque solution fournit différents services de sécurité pour contrer l'attaquant pris en considération (Voir Tableau 4), cette section résume les services fournis par chaque solution.

Solution	C	I	A	D	F
Sia et al. [45]		•	•		•
Du et al. [48]		•	•		
Mahimkar et al. [49]	•	•	•		
Ozagur et al. [51]	•				•
Vu et al. [53]	•	•	•		•
Yang et al. [54]	•	•	•		•
Castelluccia et al. [71]	•				•
Albath et al. [73]	•	•	•		•
Sun et al. [75]	•	•	•		•
Ozdemir et al. [78]	•	•	•		•
Ozdemir et al. [79]		•	•	•	•
Sedjelmaci et al. [82]				•	
Sun et al. [83]				•	
Gursel et al. [85]				•	
Labraoui et al. [86]		•	•	•	•

C : Confidentialité - I : Intégrité - **A** : Authentification - **D** : Disponibilité - **F** : Fraicheur - • : Fourni.

Tableau 4. Services de sécurité fournis

La confidentialité des données est assurée par toutes les solutions qui utilisent un algorithme de chiffrement. Parmi ces solutions, on peut citer [49,51,73,75]. Castelluccia et al. [71] avaient comme objectif la sécurisation des données contre les adversaires passifs. Ainsi, ils se sont focalisés seulement sur la confidentialité. Aussi, contrairement aux solutions [49,51,53,54], cette confidentialité avec celles de [73,75,78] sont assurées de bout en bout, et comme nous l'avons souligné précédemment, ce meilleur service ne donne pas accès aux données en clair à l'agrégateur et les nœuds intermédiaires. Ainsi, seule la station de base qui détient la (les) clé(s) de chiffrement peut accéder aux données. L'assurance ou non de la confidentialité dépend généralement de l'application dans laquelle les capteurs sont déployés. L'intégrité des données est fournie par les solutions dans lesquelles les attaques actives sont considérées dans la zone de déploiement. Un nœud compromis peut modifier les données qui le traversent, si l'intégrité n'est pas fournie, cette modification ne pourra être détectée par les nœuds intermédiaires. Les solutions [45,48,49] assurent l'intégrité tandis que dans [51,71],

elle n'a pas été considérée. Dans [78], l'intégrité est fournie mais partiellement, le *MAC* est seulement considéré entre les agrégateurs et la station de base. Dans les solutions [73,75], l'intégrité est fournie de bout en bout. Ainsi, ces solutions empêchent un nœud intermédiaire compromis de tromper la station de base par un agrégat invalide en utilisant sa clé légitime. L'authentification est fournie dans toutes les solutions qui utilisent des signatures (*MAC*, *ECDSA*,…). On peut constater à partir du Tableau 4, que tous les protocoles qui assurent l'intégrité assurent aussi l'authentification. Ceci parce que la primitive de signature utilisée fournie généralement les deux services. Dans [48], l'authentification est assurée partiellement, seulement entre l'agrégateur et la station de base. Les communications (Capteurs-Agrégateur) ne sont pas authentifiées. La fraicheur des données est assurée dans toutes les solutions qui utilisent des mises à jour des clés ou des nonces associés à chaque message. Les solutions [53,54,79,86], garantissent la fraicheur. Ainsi, elles empêchent l'attaquent d'envoyer des paquets valides déjà envoyés auparavant. En effet, si ce service n'est pas fourni, l'agrégat peut facilement et considérablement être affecté sans être détecté. Une attaque qui peut être exécutée avec succès dans [48,49]. Enfin, la disponibilité des données est assurée dans [79,82,83,85,86], à travers les mécanismes de surveillances utilisés qui leur permettent de détecter des attaques de type déni de service et d'assurer la survivabilité du réseau.

3.1.2. Vulnérabilité aux attaques

Dans cette section, nous allons étudier la résistance des solutions contre les attaques décrites dans le chapitre précédent. La comparaison est présentée dans le Tableau 5 :

3.1.2.1. Attaques passives

Les solutions [45,48] sont vulnérables à une attaque passive, ce n'est pas parce qu'elles utilisent des algorithmes non sécurisés mais parce qu'elles ne considèrent pas l'attaquant passif. Les algorithmes symétriques proposés dans [49,51,53,54] sont sécurisés et résistent bien aux attaques *A.C* et *A.T.C.* Les chiffrements homomorphiques utilisés dans [73,75,78] sont des algorithmes probabilistes donc ils ne sont pas vulnérables à ces attaques. Cependant, la solution de [71] peut être vulnérable à l'attaque *A.T.C* à cause de sa simplicité. Ainsi, si l'attaquant connait le texte en clair, il peut facilement en déduire la clé à partir du chiffré. Aussi, Contrairement à [73,75,78], cet algorithme ne peut pas atteindre le plus haut niveau de sécurité contre les attaques passives à savoir *IND-CPA*[8] (*INDistinguishability under a Chosen Plaintext Attack*), l'indistinguabilité sous une attaque à texte clair choisi [44]. Selon [87], pour atteindre ce niveau de sécurité, il faut utiliser des *PRF* (*Pseudo Random Function*), des fonctions pseudo aléatoires, impliquant ainsi, une clé selon le niveau de sécurité et la *PRF* utilisée, en conséquence, l'avantage majeur de la solution d'origine sera perdu. Pour résoudre ce problème, les mêmes auteurs ont proposé (*Hashed CMT*) d'utiliser une deuxième fonction dont l'objectif est de réduire la taille de la sortie à une taille similaire à celle du maximum possible d'une valeur agrégée, et ceci afin de préserver leur avantage. Nous reviendrons sur ce point dans le cinquième chapitre.

[8] Le niveau IND-CPA est équivalent à la sécurité sémantique [44].

Solution	Attaques passives		Attaques actives					
	A.C	A.T.C	A.C.A	A.D.S	A.E.S	A.S	A.R	A.F
Sia et al. [45]	■		■	■	■			
Du et al. [48]	■		■	■	■	■	■	■
Mahimkar et al. [49]			■	■	■		■	
Ozagur et al. [51]			■	■				
Vu et al. [53]			■	■	■			
Yang et al. [54]			■	■	■			
Castelluccia et al. [71]		■	■	■				■
Albath et al. [73]			■	■	■			
Sun et al. [75]			■	■	■			
Ozdemir et al. [78]			■	■	■	■		
Ozdemir et al. [79]	■		■					
Sedjelmaci et al. [82]	■					■	■	■
Sun et al. [83]	■				■		■	
Gursel et al. [85]	■					■	■	■
Labraoui et al. [86]	■		■					

A.C : L'Analyse des Chiffrés - **A.T.C** : Attaque à Texte en Clair connue - **A.C.A** : Attaque par Capture - **A.D.S** : Attaque par Déni de Service - **A.E.S** : Attaque par Envoi Sélectif des paquets - **A.S** : Attaque Sybille – **A.R** : Attaque par Rejeu – **A.F** : Attaque Furtive.
■ : Vulnérable.

Tableau 5. Vulnérabilité aux attaques

3.1.2.2. Attaques actives

Un nœud compromis permet à l'attaquant d'accéder à sa mémoire et peut ainsi extraire toutes les clés disponibles. Dans le contexte des RCSFs, la capture des nœuds est facile [31]. Ainsi, toutes les solutions basées sur la cryptographie sont vulnérables à l'attaque par capture. Pour les attaques par déni de service, les solutions curatives assurent la disponibilité donc ils ne sont pas vulnérables. L'attaque sybille est possible si l'authentification n'est pas fournie. Dans [48] par exemple, les nœuds feuilles ne sont pas authentifiés, un attaquant peut facilement utiliser une ou plusieurs identités pour tromper l'agrégateur.

Une fois un capteur est compromis, l'attaque par envoi sélectif peut être exécutée, ainsi toutes les solutions basées sur la cryptographie sont vulnérables à cette attaque. Les solutions curatives résistent bien à cette attaque sauf dans [83], l'attaque par envoi sélectif n'a pas été considérée. Dans l'attaque par rejeu, l'attaquant essaie de rejouer ou réinjecter des paquets valides déjà envoyés sans même comprendre le contenu. Toutes les solutions qui assurent la fraicheur des données peuvent contrer cette attaque. Cependant, les solutions [48,71,79] sont vulnérables à l'attaque par rejeu. Finalement, les attaques furtives ne peuvent s'exécuter avec succès sur les solutions qui assurent l'intégrité et l'authentification. A partir du tableau, on peut constater que les solutions [48,71,82,85] sont vulnérables à ces attaques. Le chiffrement homomorphique est intrinsèquement malléable, n'importe qui peut modifier les données sans même comprendre le contenu. Sans une sécurité additionnelle, la station de base peut accepter le résultat erroné comme valide.

3.2. Performances

Le Tableau 6 présente une comparaison entre les solutions étudiées en termes de charges générées, scalabilité et versatilité.

Solution	Charge		Sca	Ver
	Com	Cal		
Sia et al. [45]	+	++		•
Du et al. [48]	++	++		•
Mahimkar et al. [49]	++	+++		
Ozagur et al. [51]	+	+	•	•
Vu et al. [53]	+	++		
Yang et al. [54]	++	++		•
Castelluccia et al. [71]	+	+		
Albath et al. [73]	+++	+++		
Sun et al. [75]	+++	+++		•
Ozdemir et al. [78]	+++	+++		
Ozdemir et al. [79]	++	++		•
Sedjelmaci et al. [82]	++	++		•
Sun et al. [83]	++	++		•
Gursel et al. [85]	++	++		•
Labraoui et al. [86]	++	++		•

Com : Charge de communication – **Cal** : Charge de calcul
Sca : Scalabilité – **Ver** : Versatilité

+++: Grande charge
++ : Moyenne charge
+ : Petite charge
• : Fourni

Tableau 6. Comparaison des performances

3.2.1. Les charges générées

Le Tableau 7 résume les solutions en termes de type de sécurité fourni (BB: Bout en bout, SS: Saut à Saut), du type du chiffrement utilisé et du nombre total de bits transmis dans le réseau nécessaire pour qu'un agrégat atteint la station de base. Pour donner un sens à cette comparaison, nous allons utiliser des nombres. Nous sélectionnons pour le message en clair m, l'identifiant i, le header h, le nombre de sauts r, le nombre de nœuds n, le nombre de witnesses w, le *MAC* c et le hash s, soit respectivement 2 octets, 2 octets, 10 octets, 1 octet, 20 octets, 4 octets, 10 octets et 10 octets. Pour les chiffrés, nous utilisons les tailles correspondantes aux algorithmes de chiffrements considérés. Les Figures 12 et 13 illustrent respectivement, le nombre de bits transmis et le coût de communication estimé en termes d'énergie pour chaque solution. Le modèle d'énergie utilisé est de Meulenaer et al. [88] pour les plateformes les plus populaires à savoir MicaZ et TelosB [2]. Ces figures montrent que l'utilisation du chiffrement symétrique est efficace en termes de communication et par conséquent en consommation d'énergie. En effet, le chiffrement à clé publique utilisé dans [49,73,75] conduit à des paquets de tailles considérables surtout celui utilisé dans [73]. Le chiffrement symétrique homomorphique utilisé dans [71] est le plus efficace, cependant, il est inefficace dans les réseaux à grande échelle à cause du problème d'identifiants. Pour la charge du calcul, un chiffrement symétrique tel que Skipjack nécessite seulement 0.22ms et

0.35ms pour un chiffrement/déchiffrement [89] respectivement sur MicaZ et TelosB. Tandis que lorsque l'asymétrique est employé nous ne parlons plus de millisecondes mais de secondes. Selon [90] le chiffrement de l'*ECEG* dure 1.29s, et puisque le temps d'exécution est traduit directement en consommation d'énergie [88], le coût introduit par les solutions [49,73,75] est très important. Certes, le chiffrement à clé publique offre un niveau de sécurité meilleure que le chiffrement symétrique. Mais cette supériorité a un impact très important sur la batterie du capteur. Finalement, une solution de sécurité pour l'agrégation doit mettre en évidence l'avantage d'utiliser l'agrégation.

Solution	Sécurité	Chiffrement	Coût de transmission
Sia et al. [45]	SS	Sym	$n(m+c+h) + (m+c+1+h)$
Du et al. [48]	SS	Sym	$n((m+h)*(w+1)) + (w*(m+c+h)) + ((m+(w+1)c+h))$
Mahimkar et al. [49]	SS	Asym et Sym	$(n+2)* (mc+s+h))+n(s+h)$
Vu et al. [53]	SS	Sym	$n*(mc+h)+ n(c+h)+ 1,1*r*(mc+c+h).$
Yang et al. [54]	SS	Sym	$n(mc+c+h)$
Castelluccia et al. [71]	BB	Sym	$n*(mc+h) +n*id$
Albath et al. [73]	BB	Asym	$n*(h+107)$
Sun et al. [75]	BB	Asym	$n*(60+h)$

Tableau 7. Charges de transmission

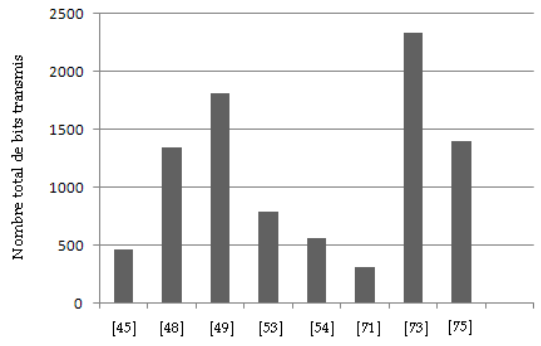


Figure 12. Charges totales de transmission

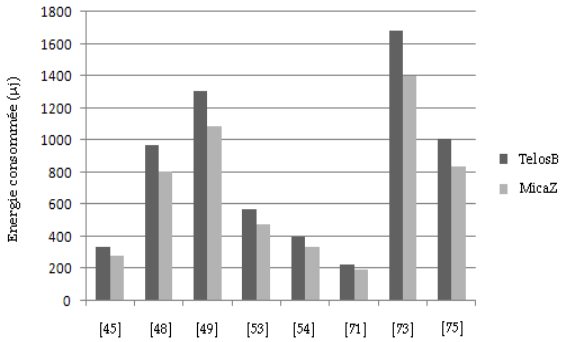


Figure 13. Energies totales de transmission

3.2.2. Scalabilité

Comme nous l'avons souligné dans le premier chapitre, plusieurs applications exploitent non seulement un grand nombre de capteurs mais aussi avec une forte densité. Ainsi, un protocole de sécurité pour l'agrégation doit aussi permettre le passage à l'échelle. Des travaux ont montré que l'utilisation des architectures à base de clusters est la meilleure solution pour permettre cet aspect. Cependant, la scalabilité demeure difficile d'être fournie dans la pratique surtout quand la sécurité est fournie et cela est principalement dû aux ressources limitées des capteurs sans fil. Les solutions de bout en bout offrent une meilleure scalabilité par rapport aux solutions saut à saut. En effet, le chiffrement homomorphique permet aux nœuds intermédiaires d'agréger directement les données chiffrées, évitant ainsi

non seulement l'effort chiffrement/déchiffrement mais aussi le délai correspondant introduit par les solutions saut à saut. Parmi les solutions étudiées, Ozagur et al. [51] offre un bon passage à l'échelle et l'utilisation des valeurs références permet une réduction considérable de la charge de communication.

3.2.3. Versatilité

La versatilité est un aspect très important pour la plupart des applications. Une solution est versatile lorsqu'elle accepte n'importe qu'elle fonction d'agrégation sur les données individuelles. Ce service est bien assuré dans les solutions saut à saut puisque l'agrégateur a en sa possession tous les messages en clair. Cependant, dans les solutions de bout en bout ce service dépend du chiffrement homomorphique utilisé. Comme indiqué dans [91], les solutions de bout en bout actuelles peuvent seulement assurer un nombre limité de fonctions (associées à l'opération d'addition) telles que la moyenne, la variance ou encore la détection des mouvements. Les solutions [45,48,51] assurent la versatilité, tandis que seulement la moyenne est considérée dans [49]. [73] est la seule solution de bout en bout qui fournit ce service à cause de la fonction d'encodage utilisée avant le chiffrement. Cette fonction permet à la station de base de récupérer tous les messages individuels.

4. Conclusion

Dans ce chapitre, on a mis l'accent sur la plupart des solutions, les plus étudiées et citées dans la littérature. Nous avons analysé et identifié les avantages et les inconvénients de chaque solution, ensuite une comparaison globale a été effectuée.

L'agrégation des données est très bénéfique pour les RCSFs puisqu'elle permet de réduire le nombre de messages transmis dans le réseau. La consommation d'énergie augmente linéairement avec le nombre de données transmises. En conséquence, l'agrégation est efficace pour un RCSF. Cependant, la sécurité constitue un problème majeur pour l'agrégation, chacune des solutions étudiées à ses avantages et ses limites. Nous avons classé ces solutions en deux grandes classes à savoir préventives et curatives. Les solutions préventives, utilisant la cryptographie pour contrer les attaques, se concentrent sur la confidentialité et l'intégrité des données, tandis que les solutions curatives se penchent sur la disponibilité des données. Concernant les sous classes des solutions préventives, les solutions de bout en bout fournissent une sécurité de bout en bout et celle-ci peut être considérée comme une propriété plus solide par rapport à la sécurité saut à saut. Les avantages sont: (i) l'agrégateur n'a pas besoin d'accéder aux données en clair ni d'avoir des clés de chiffrement. Dans les solutions saut à saut, un agrégateur compromis pourrait révéler ces informations. (ii) l'énergie totale du réseau peut être réduite. En effet, dans les solutions SS, l'agrégateur doit déchiffrer, agréger puis chiffrer le résultat. Cependant, les solutions BB réduisent considérablement le coût énergétique puisqu'il n'y a pas de déchiffrement et chiffrement au niveau de l'agrégateur. Il est primordial de fournir une solution efficace en termes d'énergie pour les agrégateurs, puisque ces nœuds représentent la pièce maitresse de la durée de vie et la connectivité des RCSFs. (iii) ils sont plus flexibles. Dans les solutions SS, seuls les nœuds possédant les clés correspondantes peuvent effectuer le déchiffrement. Cependant, dans les

solutions BB, n'importe quel nœud peut devenir un agrégateur, puisque les nœuds d'agrégation n'ont pas besoin de stocker des clés pour opérer sur les chiffrés reçus. Ainsi, l'élection de l'agrégateur peut seulement dépendre de l'énergie disponible. Autrement dit, les solutions BB fournissent une confidentialité sans imposer des restrictions sur les algorithmes d'élection, et ceci améliore la robustesse et la fiabilité des RCSFs. (iv) le niveau de sécurité du RCSFs augmente. Le chiffrement symétrique tel que Skipjack, *RC5* ou *AES* introduit un compromis entre la flexibilité et la sécurité. En effet, l'utilisation d'une seule clé globale implique une flexibilité de l'algorithme d'élection mais avec presqu'aucune sécurité. Avec une clé de groupe ou même des clés par paire (*pairwise key*), la sécurité augmente mais non seulement impose des chemins de routages statiques ou presque mais aussi un ensemble fixe d'agrégateurs. Cette observation vient du fait que dans les RCSFs ou la capture des nœuds est généralement facile, le composant le plus faible n'est pas le système de sécurité lui-même, mais la politique de stockage des données sensibles.

Aussi, les solutions curatives ont besoin d'une considérable charge de communication afin de détecter l'attaquant. Cependant, cette charge dépend du chiffrement utilisé dans les solutions préventives. En effet, le chiffrement asymétrique conduit à des chiffrés de tailles considérables, ainsi le nombre de bits transmis dans le réseau devient important. En termes de calcul, il est clair que le chiffrement symétrique est le plus approprié pour la sécurisation, mais ses inconvénients cités précédemment ont conduit à l'utilisation de l'asymétrique. Actuellement, l'implémentation efficace du chiffrement asymétrique basé sur les courbes elliptiques constitue l'objet de plusieurs recherches.

Tous ces observations nous ont conduit à apporter plusieurs contributions en proposant les solutions *RSAED* et *SASPKC* ainsi qu'une implémentation efficace et sécurisée de la cryptographie sur les courbes elliptiques. Ces contributions seront détaillées dans les chapitres suivants.

Chapitre 3 : Première contribution: La solution RSAED

1. Introduction

La détection des usurpations d'identité et les injections des données fausses est l'une des principales préoccupations dans les RCSFs. Comme nous l'avons souligné dans le chapitre précèdent, il existe deux approches pour contrer ces attaques, à savoir les approches de bout en bout et saut à saut. Dans les solutions de bout en bout, la détection de ces attaques peut seulement être effectuée au niveau de la station de base, une détection qui peut conduire à une perte très importante de données légitimes [73]. Puisque le puits est le seul endroit où la vérification est effectuée, les activités malveillantes au cours du processus d'agrégation ne peuvent être détectées qu'après réception de l'agrégat final. Ceci non seulement est inefficace mais aussi conduit à un taux important de pertes de données.

Dans ce chapitre, nous présentons notre première contribution nommée *RSAED* [92], qui fournit une confidentialité de bout en bout et permet une détection précoce de l'attaque à travers une vérification saut à saut, réduisant ainsi la nécessité de s'appuyer entièrement sur le puits pour la vérification.

2. Motivations

La confidentialité des données est un service d'une importance primordial pour un bon nombre d'applications (par exemple les applications militaires). Dans notre état l'art, nous avons étudié plusieurs solutions qui fournissent ce service [71,73,75,78]. Nous avons vu qu'une confidentialité de bout en bout, impliquant le chiffrement homomorphique, offre une meilleure sécurité que celle de saut à saut. Cependant, il est connu que cette propriété implique la malléabilité. En d'autres termes, étant donné un texte chiffré c, un attaquant peut facilement générer un texte chiffré c' afin de tromper la station de base en acceptant le m' correspondant qui est lié au message clair original m, et ce dernier n'est pas nécessairement connu par l'attaquant [44]. L'intégrité des données est donc nécessaire afin d'empêcher cette malléabilité de se produire. Cependant, les solutions existantes souffrent de deux inconvénients majeurs à savoir l'inefficacité due à la charge de calcul et de communication introduite et le rejet total des données. Ce dernier est un problème qui se produit lorsqu'une activité malveillante est détectée dans une branche particulière de sorte que toutes les données reçues de la branche infectée seront rejetées par le puits y compris les données importantes et correctes.

Dans ce qui suit, nous allons présenter notre algorithme *RSAED* dans lequel nous utilisons le protocole de chiffrement El Gamal ainsi que sa propriété homomorphique pour fournir une confidentialité de données de bout en bout. Notre solution permet aussi la vérification au niveau des nœuds intermédiaires, assure que les paquets reçus par la station de base proviennent seulement des nœuds qui sont légitimes et améliore l'efficacité.

3. Spécifications générales

Dans cette section, nous spécifions les hypothèses sur le réseau et la sécurité ainsi que les objectifs visés et l'attaquant considéré.

3.1. Modèle du réseau

Nous supposons que le réseau est divisé en *j* clusters statiques (Voir Figure 14), et chaque cluster contient *n* nœuds capteurs ayant des ressources limitées tels que MicaZ [2] pouvant communiquer entre eux d'une manière sécurisée. Chaque cluster est géré par deux Cluster-Head (*CH*). Notons que l'utilisation de deux *CHs* est pour des fins d'efficacité. L'algorithme d'élection peut être basé sur l'énergie disponible sur le capteur, plus l'énergie est disponible, plus la probabilité de devenir agrégateur augmente. En d'autres termes, notre solution n'impose aucune restriction sur cet algorithme. Chaque capteur envoie les données à l'agrégateur correspondant, ensuite les agrégateurs frères interagissent pour produire l'agrégat et enfin, le résultat est envoyé à la station de base ou bien l'agrégateur correspondant le plus proche. La station de base est supposée avoir des ressources informatiques illimitées.

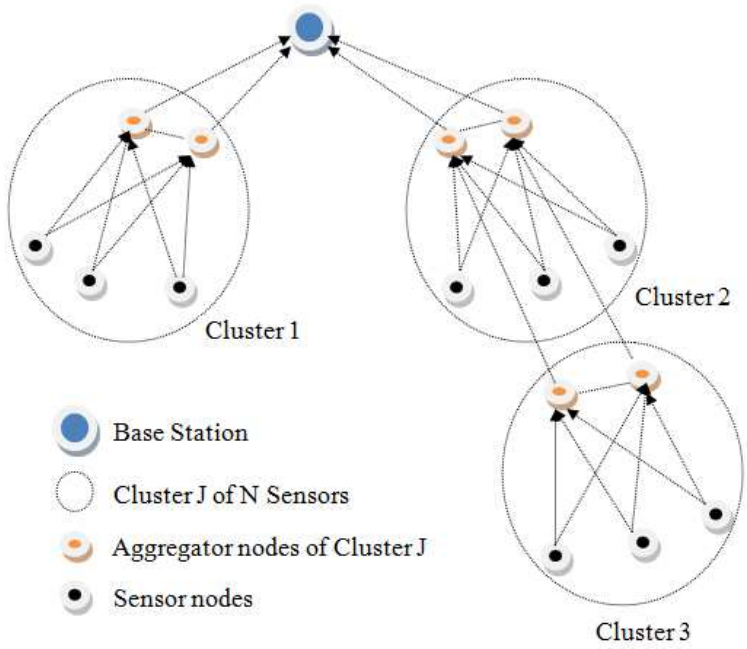


Figure 14. Modèle du réseau

3.2. Modèle de sécurité

Afin d'assurer une confidentialité de bout en bout, le chiffrement homomorphique est utilisé, ce qui permet le calcul sur les données dans leurs versions chiffrées sans la nécessité de déchiffrer, empêchant ainsi les nœuds intermédiaires d'accéder au texte en clair [44]. Dans notre étude des solutions proposées, nous avons vu que les deux types de chiffrement peuvent satisfaire le chiffrement homomorphique, mais récemment [93], ceux basés sur la cryptographie symétrique ont été cryptanalysés [94], et donc, pour des raisons de sécurité, il est préférable d'utiliser la Cryptographie à Clé Publique (*CCP*). Il est connu que l'utilisation

de la *CCP* entraîne un surcoût très élevé du calcul, mais les résultats récents ont montré que la cryptographie sur les courbes elliptiques est possible, et peut être mise en œuvre dans des dispositifs à ressources limitées [95]. Dans [58], les auteurs ont étudié et analysé un ensemble sélectionné de ces algorithmes pour une confidentialité de bout en bout dans les RCSFs. Cet ensemble comprend l'*EC-NS* (*Elliptic Curve Naccache-Stern*), l'*EC-OU* (*Elliptic Curve Okamoto-Uchiyama*), l'*EC-P* (*Elliptic Curve Pailler*) et l'*ECEG*. Leurs résultats présentés dans la Figure 15 montrent que l'*ECEG* est bien l'algorithme le plus adapté aux RCSFs.

Scheme	Encryption		Addition		Bandwidth	
EC-NS	$\frac{15}{2}\|n\|$	1800	$5\|n\|$	5	$\|n\|+2$	1026
EC-OU	$5+\frac{15}{2}\|m\|+\frac{15}{2}\|2k\|$	690	$5\|n\|$	5	$\|n\|+2$	1026
EC-P	$\frac{15}{2}\|n^2\|$	33105	5	20	$2\|n\|+4$	2052
EC-ElGamal	$2\frac{15}{2}\|p\|+\frac{15}{2}\|m\|$	38	$10\|p\|$	1	$2(\|p\|+1)$	328

Figure 15. Comparaison entre les algorithmes basés sur l'ECC [58]

Le chiffrement est représenté dans l'Algorithme 2. La fonction *map()* est une fonction déterministe qui transforme une valeur (le message en clair *m*) en un point sur la courbe tel que l'égalité suivante est vérifiée:

$$map\ (m_1 + m_2 + \ldots + m_n) = map\ (m_1) + map\ (m_2) + \ldots + map\ (m_n)$$

Cette conversion est seulement nécessaire pour permettre l'addition sur la courbe, autrement dit, la sécurité de l'*ECEG* est indépendante de cette fonction. La fonction inverse *rmap()* nécessite la résolution de l'*ECDLP* (*Elliptic Curve Discret Logarithm Problem*) sur une courbe elliptique et par conséquent, un inconvénient de cet algorithme. Cependant, puisque cette fonction inverse s'effectue seulement au niveau de la station de base, supposée généralement à ressources illimitées, ce point faible ne peut influencer sur les performances du réseau.

Algorithme 2 : Elliptic Curve El Gamal [58]

Clé privée : $x \in GF(p)$
Clé publique : E,p,G,Y avec Y=xG
Chiffrement :
Pour un message m dans l'intervalle [0, p-1], générer une clé aléatoire k, où $k \in [0, n-1]$, n représente l'ordre de la courbe E

1. M = map (m)= mG
2. C= (R, S) = (kG, kY+mG)

Déchiffrement :

1. M = S – xR
2. m =rmap(M)

Addition des chiffrés :

1. $C' = (k_1G, k_1Y+m_1G)+(k_2G, k_2Y+m_2G)+\ldots+(k_nG, k_nY+m_nG)$
2. $C' = ((k_1+k_2+..k_n)G, (m_1+m_2+m_n)G+(k_1+k_2+..k_n)Y)$

La sécurité de L'*ECEG* est basée sur l'*ECDLP*. L'*ECEG* est probabiliste, c'est-à-dire le chiffrement du même message conduit avec une grande probabilité à un chiffré différent. Aussi, le chiffrement de deux messages différents est indistinguable, ainsi il peut atteindre le plus haut niveau de sécurité contre les attaques passives à savoir *IND-CPA* [44].

Afin d'assurer l'intégrité, nous utilisons le *MAC*. Il est clair que l'intégrité de bout en bout est impossible puisque le *MAC* ne peut pas satisfaire la propriété homomorphique additive:

$$MAC\ (a + \beta) \neq MAC\ (a) + MAC\ (\beta)$$

Dans *RSAED*, la vérification saut à saut est adoptée pour assurer à la fois l'intégrité et l'authentification de la source. Nous utilisons *HMAC*, un *MAC* basé sur le hachage, qui a les mêmes propriétés que la fonction de hachage à sens unique, y compris une clé [47].

Nous supposons que les paramètres de la courbe sont préchargés dans chaque capteur avant le déploiement ainsi qu'une clé master *K*. Aussi, juste après le déploiement, chaque capteur calcule une clé unique avec chaque membre de son cluster comme suit: chaque nœud génère un nombre aléatoire, puis il fait un *broadcast* de ce nombre. Nous supposons que ce message peut être reçu par tous les membres du cluster. Ensuite le nœud calcule $K_{i,j}$= *HMAC (K,* $R_i \| R_j$*)* avec $i>j$, après le calcul de toutes les clés, tous les nœuds effacent la clé *K* de la mémoire, pour la simple raison que le maintien de cette clé dans la mémoire du nœud pourrait en cas de capture, compromettre toutes les communications. La suppression de la clé *K* peut être effectuée après quelques secondes une fois que le réseau est déployé. Donc, il y a une très faible probabilité de présence de l'adversaire dans ce court intervalle. En conséquence, chaque nœud maintient une liste de clés avec ses membres du cluster et peut communiquer de manière sécurisée.

3.3. Modèle d'attaque

Nous considérons toutes les attaques décrites dans [91] où l'attaquant peut écouter et surveiller les données transmises à savoir, l'attaque par analyse des chiffrés et les attaques à texte en clair connu. L'attaquant peut aussi produire un ensemble de données illégales, modifier les paquets et rejouer des paquets valides déjà transmis, ce qui correspond respectivement à la falsification des paquets, à la malléabilité et à l'attaque par rejeu.

3.4. Objectifs de conception

Dans notre approche, nous visons les objectifs suivants:

- Sécurité

Pour faire face aux risques de sécurité, une solution de sécurité doit fournir les services de sécurité cités dans les chapitres précédents.

- Efficacité

Un protocole de sécurité doit être efficace en termes de charge de calcul et de communication afin de préserver l'énergie et prolonger la durée de vie du réseau.

- Robustesse

Un mécanisme de sécurité doit assurer la disponibilité des paquets, même avec la présence de quelques nœuds compromis ou défectueux.

4. Notre Approche: RSAED

Dans cette section, nous allons présenter notre algorithme, nous commençons par une vue d'ensemble puis nous donnons les détails de notre proposition.

4.1. Vue d'ensemble

Notre solution est composée de trois étapes : *transmission, filtrage et agrégation,* puis *vérification*. Dans la première étape, chaque nœud capteur génère un chiffré et la signature correspondante en utilisant respectivement l'*ECEG* et le *MAC*, puis envoie le résultat de chaque portion du chiffré à l'agrégateur correspondant. Dans la seconde étape, les agrégateurs frères vérifient d'abord la validité des données reçues, ensuite ils interagissent afin de filtrer les paquets invalides ou malicieux. Par la suite, les deux agrégateurs envoient l'agrégat crypté avec sa signature à la station de base ou à l'agrégateur le plus proche. Enfin, la station de base reçoit et déchiffre les agrégats cryptés provenant de différents *CHs* et vérifie ensuite leur validité.

4.2. Détail de l'approche

Les étapes de notre proposition sont détaillées dans ce qui suit:

Transmission : Chaque nœud capteur chiffre sa mesure en utilisant l'*ECEG* et envoie les deux portions (dans leur forme compressée) du chiffré à savoir *R* et *S* aux agrégateurs, accompagnées d'une signature générée en utilisant la clé unique partagée entre les deux nœuds communicants, le nonce *N* et le texte chiffré. La valeur à usage unique *N* est un numéro de séquence qui ne doit être utilisé qu'une seule fois pour la fraîcheur des données. Enfin, le paquet, comprenant la donnée chiffrée et le *MAC*, est envoyé au *CH*. Dans l'Algorithme 3, nous décrivons l'algorithme exécuté par le capteur *SNi*, qui est un membre du cluster *j*.

Algorithme 3: RSAED pour le capteur SN_i

1. Utiliser l'algorithme 2 pour produire la paire $C_i=(R_i,S_i)$
2. Compresser les points R_i et S_i
3. Calculer $Sig_{i1}=HMAC(K_{i,CHJ1} , R_i \mid N_{i1})$
   $Sig_{i2}=HMAC(K_{i,CHJ2} , S_i \mid N_{i2})$
4. Envoyer (R_i, Sig_{i1}) à CH_{j1}
5. Envoyer (S_i, Sig_{i2}) à CH_{j2}
6. Incrémenter N_{i1} et N_{i2}

Filtrage et agrégation : Une fois les données reçues, chaque *CH* vérifie d'abord la signature Sig_i reçue puis décompresse le point. Dans l'autre cas, si la vérification échoue, alors le paquet est rejeté. Cela signifie que le paquet a été généré, soit par malveillance, soit par inadvertance, l'identifiant du nœud correspondant sera maintenu dans une liste. Ensuite, chaque agrégateur envoie à son frère agrégateur un message de vérification contenant le Nombre de Paquets Légitimes (*NPL*) et la Liste des Nœuds Malveillants (*LNM*). Ce message sera le même s'il n'y a pas d'anomalies, sinon les deux agrégateurs filtrent les nœuds malicieux en tenant compte du message reçu. Dans l'Algorithme 4, nous décrivons les opérations exécutées par l'agrégateur CH_{j1}, membre du cluster *j*. Le même algorithme est exécuté par le CH_{j2} correspondant à la portion *S* du chiffré. Le processus est répété entre les *CH* intermédiaires jusqu'à ce que l'agrégat final arrive à la station de base.

Algorithme 4: RSAED pour l'agrégateur CH_{j1}

Pour i=1 à n-2 **faire**

1. Calculer Sig_{i1}'=HMAC ($K_{i, CHj1}$, R_i | N_{i1})
2. Comparer Sig_{i1}' et Sig_{i1} ; **si** $Sig_{i1}' = Sig_{i1}$ **alors** aller à 3 et NPL= NPL+1
   sinon rejeter le paquet et ajouter SN_i à LNM
3. Décompresser R_i

Fin pour

4. Envoyer NPL | LNM | HMAC ($K_{CHj1, CHj2}$, | NPL | LNM | $N_{CHj1-CHj2}$) à CH_{j2}
5. Comparer avec les résultats de CH_{j2} et filtrer les nœuds malicieux
6. Utiliser l'algorithme 2 pour combiner les R_i valides et produire R_{agg}
7. Compresser R_{agg} et l'envoyer à SB ou le CH le plus proche
8. Incrémenter tous les Nonces

Vérification : Une fois les données reçues, la station de base vérifie d'abord l'intégrité de tous les paquets entrants provenant des *CHs*, puis décompresse les points. Enfin, le processus de déchiffrement est appliqué en utilisant sa clé privée. Pour récupérer le texte en clair agrégé $m = m_1 + m_2 + ... + m_n$, le *ECDLP* sur *M* doit être résolu et puisque la station de base est supposée avoir des ressources informatiques illimitées, le *ECDLP* peut être calculé de manière efficace en utilisant la méthode *Pollard-λ* sur les courbes elliptiques [96]. Dans l'Algorithme 5, nous décrivons l'algorithme exécuté par la station de base.

Algorithme 5: RSAED pour la station de base

Pour i=1 à p **faire** //p est le nombre de CH connectés avec la SB

1. Calculer Sig_p'=HMAC ($K_{SB, CHp}$, (S_{agg} or R_{agg}) | N_{CHp})
2. Comparer Sig_p' et Sig_p ; **si** $Sig_p' = Sig_p$ **alors** aller à 3
   sinon rejeter le paquet
3. Décompresser le point

Fin pour

4. M = S_{agg} – x R_{agg} // x est la clé privée de la SB
5. m =rmap(M)
6. Incrémenter les Nonces

5. Analyse

Dans cette section, nous analysons la sécurité et les performances de notre solution

5.1. Sécurité

Il est clair qu'en utilisant un chiffrement homomorphique, les données sont chiffrées seulement par les nœuds capteurs et déchiffrées seulement au niveau de la station de base, ainsi la confidentialité de bout en bout est assurée. Un attaquant passif ne peut rien comprendre en analysant les chiffrés et cela est dû à la propriété probabiliste que possède l'ECEG. Cependant, quand nous considérons les attaques actives, la vérification de l'intégrité est nécessaire afin de garantir que toutes les données ont été transmises avec succès. Dans notre solution, chaque nœud envoie une signature au nœud intermédiaire, ce dernier vérifie et si celle-ci échoue le paquet sera perdu. Le processus se répète jusqu'à la station de base. Analysons quelques attaques contre l'intégrité:

- Un attaquant qui veut envoyer des paquets déjà envoyés et valides, ce qu'on appelle l'attaque par rejeu, ne peut perturber le réseau, parce que même s'il est valide, il n'est pas

frais. L'utilisation du Nonce empêche l'attaque par rejeu, ainsi notre solution assure la fraicheur des données.

- Les attaques contre le chiffrement homomorphique (Malléabilité) est l'une des attaques les plus étudiées actuellement par la communauté scientifique. L'utilisation de la signature *HMAC* empêche la modification des chiffrés. Ainsi, un attaquant ne peut produire une signature valide d'un message modifié sans avoir une clé valide. Aussi, à cause des propriétés du chiffrement à clé publique n'importe qui peut produire des chiffrés, cette attaque ne peut aboutir sans la génération des signatures valides correspondantes.

- Si un nœud intermédiaire est physiquement compromis, l'intégrité des données sera affectée. Cependant, la confidentialité de bout en bout sera toujours maintenue.

- Dans notre système, il y a les deux agrégateurs qui exécutent la fonction d'agrégation sur les données chiffrées, chacun calcule une portion de l'agrégat, et l'envoie à la station de base ou l'agrégateur le plus proche. Le moyen le plus simple pour un attaquant de modifier le résultat de l'agrégation est de mettre certains nœuds malveillants dans le réseau et essayer d'injecter des paquets erronés ou même rejouer des paquets valides déjà transmis. Cette attaque ne peut réussir que si l'algorithme *ECEG* est utilisé seul, parce que le système est malléable, et un seul nœud compromis est capable de modifier le résultat correct. Notre système permet la vérification saut à saut et ces nœuds malveillants ne possèdent pas la clé *MAC*, donc ils ne peuvent pas perturber le réseau. Ainsi les paquets malicieux sont détectés et rejetés au niveau suivant. En outre, lors de la vérification dans la deuxième étape, les agrégateurs frères se mettent d'accord sur une liste spécifique qui contient les mêmes nœuds malveillants et regroupent les portions (points) qui proviennent des mêmes nœuds légitimes.

- Généralement, les données gérées par les agrégateurs concerne plusieurs nœuds du réseau, il est donc plus intéressant pour un attaquant de compromettre ce type de capteurs au lieu d'une simple mesure d'un capteur feuille. Notre système utilise le chiffrement homomorphique et, par conséquent, les textes en clair ne sont pas visibles au niveau de ces nœuds. Aussi, nous utilisons la vérification saut à saut, qui empêche les paquets malicieux d'aboutir et améliore la robustesse du RCSF. Par exemple, si l'un des agrégateurs d'un cluster est compromis, et puis détecté par le niveau supérieur, le paquet correspondant est rejeté. Aussi, au cours de la vérification entre les agrégateurs du niveau supérieur, le paquet reçu du frère du nœud compromis sera également rejeté, et par conséquent, notre système assure la transmission des paquets provenant seulement des nœuds légitimes.

- Dans *RSAED*, la vérification saut à saut est adoptée et peut ainsi tolérer la défaillance des nœuds. En effet, les paquets légitimes sont disponibles et peuvent bien atteindre la station de base, même avec la présence de quelques dysfonctionnements dans le réseau. Cela se fait simplement en sautant le point de défaillance, et par conséquent, la robustesse du RCSF est améliorée.

5.2. Performances

Notre solution vise à réduire la charge au niveau des nœuds d'agrégation en impliquant deux agrégateurs dans chaque cluster et diviser l'opération homomorphique entre eux. Dans

ce qui suit, nous évaluons notre système en termes de charges de calcul et de communication.

5.2.1. Charge de calcul

Nous avons implémenté notre solution en considérant le capteur MicaZ de Crossbow [2] équipé d'un processeur 8 bits et une horloge de 7.3728 MhZ, 4KB de RAM et 128KB de mémoire flash. NesC comme le langage spécifique pour les systèmes embarqués et aussi la bibliothèque des opérations *ECC* et TinyECC [97] sous TinyOS [98] sont utilisés. Pour notre application la courbe standard *secp160r1* recommandée par SEC (*Standards for Elliptic Curve*) [99] est utilisée. La simulation est effectuée en utilisant TOSSIM et AVRORA (version 1.7x [100]) qui simulent respectivement les applications des RCSFs et le temps d'exécution de nos opérations cryptographiques.

Le module *RSAEDM.nc* implémente l'initialisation, le cryptage, la fonction homomorphique, la signature *MAC* à savoir respectivement *RSEAD.init(), RSAED.encrypt(), RSAED.hom_add()* et *RSAED.sign()*. Dans la cryptographie sur les courbes elliptiques, l'opération la plus coûteuse est la multiplication des points par un scalaire. TinyECC fournit un certain nombre d'optimisations, qui peuvent être activées ou désactivées en fonction des besoins de l'application, y compris la méthode des fenêtres glissantes appelée *SWM* (*Sliding Windows Method*) qui améliore considérablement le temps d'exécution de la multiplication surtout quand le point de base est fixe et connu à priori. Dans notre code, nous utilisons *SWM*, et *HMAC* fournis par TinyECC. D'après [97], la consommation d'énergie *P* de chaque opération arithmétique peut être calculée en utilisant la formule $P=U \times I \times t$, où *U* désigne la tension, *I* représente le courant, alors que le temps d'exécution est représenté par *t*, la tension et le courant sont supposés être respectivement 3V et 8 mA. Le temps d'exécution et l'énergie estimée correspondante sont présentés dans le Tableau 8.

Opération	Temps d'exécution	Consommation
RSAED.encrypt()	2.844s	68.256 mj
RSAED.hom_add()	0.796s	19.104 mj
RSAED.sign()	0.028s	0.672 mj

Tableau 8. Temps d'exécution de nos opérations sur MicaZ

Un agrégateur dans *RSAED* calcule une signature pour chaque paquet reçu, puis envoie un paquet de vérification à son frère agrégateur, ensuite il effectue la décompression des points et enfin l'addition des points sur la courbe elliptique. L'opération homomorphique est calculée d'une manière distribuée et cela diminue le délai de bout en bout par rapport à la solution sans calcul distribué que nous allons appelé *S-ECEG*. Le temps pris par la fonction homomorphique est 0.796s et consomme 19.104mj pour deux points (deux portions de chiffrés). La Figure 16 montre l'effet de l'augmentation du nombre de nœuds sur la consommation d'énergie au niveau des nœuds d'agrégation à la fois dans *RSAED* et *S-ECEG*. Il est clair que l'utilisation de deux agrégateurs réduit considérablement la surcharge de calcul au niveau des nœuds d'agrégation et donc la durée de vie du réseau est améliorée. La Figure

17 illustre les avantages de l'utilisation d'un calcul distribué sur le délai de bout en bout (on suppose que les nœuds d'agrégation communiquent directement avec la station de base). Par exemple, pour 20 capteurs, la station de base doit attendre environ 19 secondes pour recevoir les données chiffrées agrégées en utilisant *S-ECEG*, alors qu'elle doit attendre seulement 9,5 secondes avec *RSAED*. En outre, la fonction homomorphique enregistre 140 octets moins de mémoire, par rapport à celle utilisée dans *S-ECEG*.

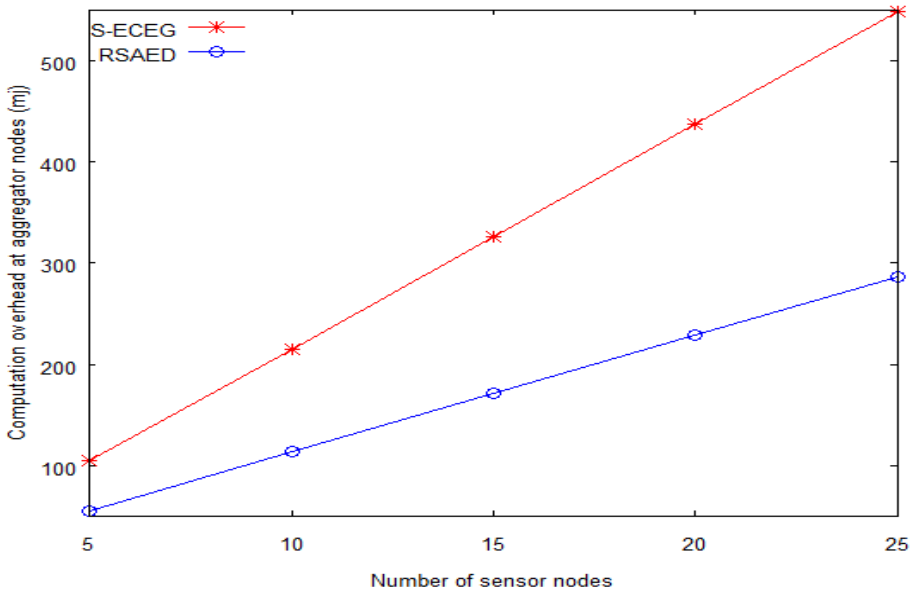


Figure 16. Charge de calcul Vs Nombre de nœuds

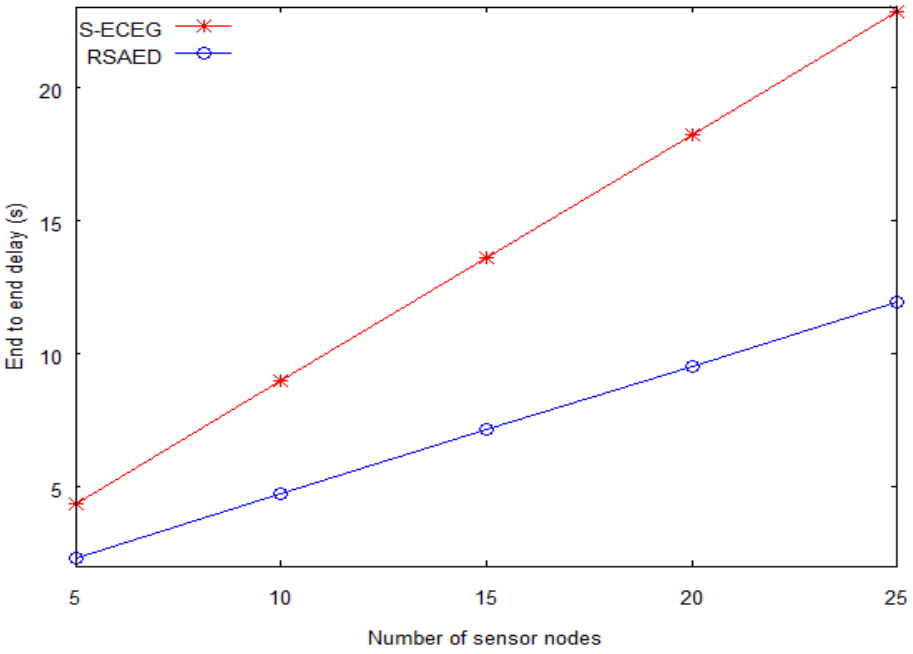


Figure 17. Délai de bout en bout Vs Nombre de nœuds

5.2.2. Charge de communication

En termes de charge de communication, chaque capteur envoie 31 octets, à savoir la portion du chiffré (le point compressé = 21 octets) et une signature à l'agrégateur (10 octets). Par conséquent, notre système évite les problèmes de division en blocs entraînant des délais importants et l'utilisation de longs paquets diminuant la fiabilité du RCSF. Si nous supposons que chaque cluster est composé de *N* nœuds, y compris les deux agrégateurs, le nombre de paquets reçus par les nœuds d'agrégation dans notre régime est (*N-1*), où (*N-2*) paquets provenant des nœuds de capteurs et un paquet de l'agrégateur frère à des fins de vérification. Sur la Figure 18, on compare notre solution *RSAED* avec *S-ECEG* où chaque capteur divise le texte chiffré en blocs et envoie chaque bloc séparément.

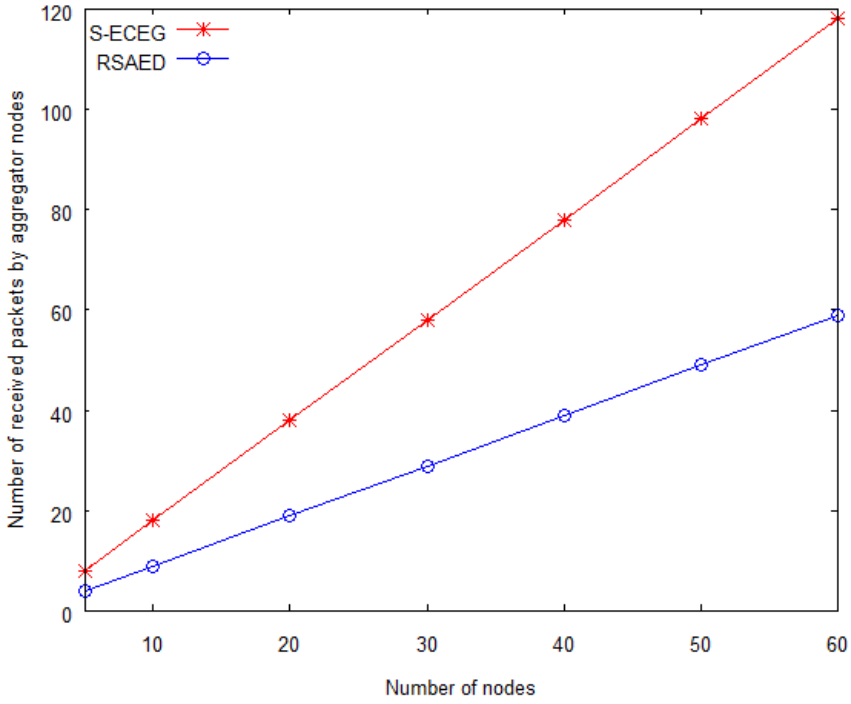


Figure 18. Nombre de parquets reçus Vs nombre de nœuds

En utilisant TOSSIM, plus précisément TOSSIM-CC2420, fourni dans la distribution TinyOS (il modélise la radio CC2420), nous comparons notre solution avec celle proposée dans [73]. La raison de choisir cette solution pour comparaison est qu'elle fournit la confidentialité et l'intégrité et rencontre le problème du rejet total des données. Dans trois scénarios de simulation, les capteurs sont déployés de façon aléatoire à l'intérieur d'une zone carrée avec une seule station de base placée au centre. Nous considérons 4, 8 et 12 clusters pour respectivement 50, 100 et 150 nœuds. Chaque exécution de la simulation dure 500s, et le résultat de chaque solution correspond à la moyenne des résultats de plusieurs simulations. Nous utilisons un simple algorithme de *clustering* à base de *TDMA* pour transmettre les données à la station de base. Dans notre application, les nœuds capteurs envoient (toutes les 30 secondes) un paquet aux *CHs* correspondants, les *CHs* interagissent entre eux afin de filtrer les nœuds malveillants. Les données sont ensuite agrégées puis envoyées à la station de base. La surcharge de communication est mesurée comme étant le nombre total de bits transmis sur le réseau. Les résultats sont présentés sur la Figure 19.

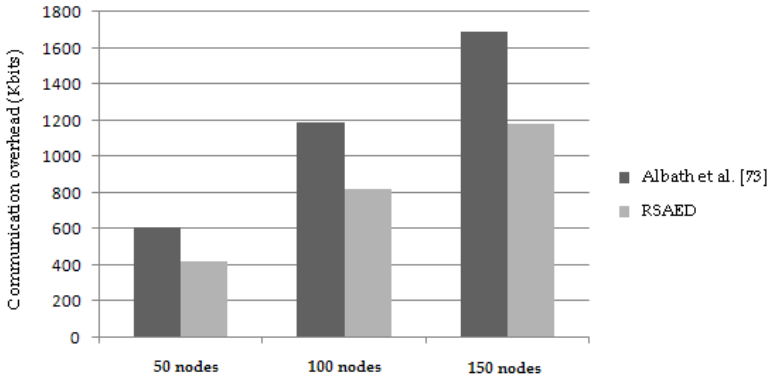


Figure 19. Charge de communication de RSAED et Albath et al. [73]

Les résultats montrent que *RSAED* introduit moins de charge par rapport au travail d'Albath et al. [73]. Ceci est principalement dû au schéma de signature utilisé. Albath et al. [73] ont utilisé un *ECDSA* modifié exigeant l'inclusion de la signature et la clé publique dans le paquet, et ceci afin de permettre à la station de base de vérifier les données. Aussi, dans leur solution, une modification se produisant au cours du processus d'agrégation conduira à un rejet total de l'agrégat final. En d'autres termes, elle ne peut arrêter la propagation des fausses données. Ce problème empêche certaines données correctes d'être validées par la station de base et surtout mène à un gaspillage important des ressources du réseau. Notre système prend en charge la vérification dans le réseau (*in-network verification*), qui permet une vérification locale des données, sans avoir le besoin de se référer à la station de base pour la vérification, et ce en révoquant immédiatement les données ou les nœuds malveillants au cours du processus d'agrégation. En d'autres termes, seuls les paquets provenant d'une inadvertance ou d'une malveillance seront rejetés. Ainsi, *RSAED* préserve les ressources du réseau.

5.2.3. Résultats pratiques

Les résultats présentés jusqu'ici concernent uniquement une évaluation par simulation. Afin de donner plus de valeur à notre évaluation, nous avons implémenté notre solution [101] sur des capteurs TelosB disponibles au laboratoire STIC. TelosB est un dispositif à 16 bits muni du microcontrôleur MSP430F1611 avec 48 Ko de mémoire flash, 10 Ko de RAM et fonctionnant à une fréquence d'horloge de 8 MHz.

Nous avons utilisé les mêmes outils à savoir TinyOS et TinyECC et aussi les mêmes paramètres pour la courbe. Les résultats obtenus, présentés dans le Tableau 9, montrent que le temps pris par un capteur TelosB pour les mêmes opérations est supérieur à celui de MicaZ. Cependant, l'énergie consommée par MicaZ est supérieure à celle de TelosB. Ceci peut être expliqué par le fait que TelosB est intrinsèquement efficace en énergie par rapport à MicaZ.

Opération	Temps d'exécution	Consommation
RSAED.encrypt()	6.11s	32.994 mj
RSAED.hom_add()	1.61s	8.694 mj
RSAED.sign()	0.032s	0.172 mj

Tableau 9. Résultats sur TelosB

5.3. Comparaison avec quelques solutions de la littérature

Le Tableau 10 présente une comparaison entre les différentes solutions en considérant les critères les plus pertinents.

Solution	Charge		Sca	Ver
	Com	Cal		
Sia et al. [45]	+	++		•
Mahimkar et al. [49]	++	+++		
Castelluccia et al. [71]	+	+		
Albath et al. [73]	+++	+++		
Sun et al. [75]	+++	+++		•
RSAED	++	++		

Solution	Attaques passives		Attaques actives						Services				
	A.C	A.T.C	A.C.A	A.D.S	A.E.S	A.S	A.R	A.F	C	I	A	D	F
Sia et al. [45]	■	■	■	■						•	•		•
Mahimkar et al. [49]			■	■	■		■		•	•	•		
Castelluccia et al. [71]		■	■	■	■			■	•				•
Albath et al. [73]			■	■	■				•	•	•		•
Sun et al. [75]			■	■	■				•	•	•		•
RSAED			■		■				•	•	•	•	•

A.C : L'Analyse des Chiffrés - **A.T.C** : Attaque à Texte en Clair connue - **A.C.A** : Attaque par Capture - **A.D.S** : Attaque par Déni de Service - **A.E.S** : Attaque par Envoi Sélectif des paquets - **A.S** : Attaque Sybille – **A.R** : Attaque par Rejeu – **A.F** : Attaque Furtive.
Com : Charge de communication – **Cal** : Charge de calcul
Sca : Scalabilité – **Ver** : Versatilité

+++: Grande charge
++ : Moyenne charge
+ : Petite charge
• : Fourni - ■ : Vulnérable

Tableau 10. Comparaison entre les solutions

Le protocole proposé dans [45] fournit une agrégation sécurisée de l'information. Cette solution vise à calculer d'une manière sécurisée la fonction d'agrégation (moyenne, médiane, minimum, maximum, etc.) en considérant l'attaque furtive dans laquelle le but de l'attaquant est de tromper l'utilisateur en acceptant de faux résultats. Même si le régime proposé prévoit l'intégrité, la confidentialité des données n'est pas disponible parce que les données sont envoyées en clair. Dans notre système, nous utilisons un schéma de chiffrement homomorphique qui assure la confidentialité de bout en bout, puisque le chiffrement est effectué au niveau des capteurs et le décryptage est effectué uniquement au niveau de la station de base.

Les auteurs de [49] ont proposé la première agrégation des données sécurisée basée sur la cryptographie sur les courbes elliptiques. Dans leur solution les données sont cryptées saut à saut, et une fois reçu le *CH* calcule la moyenne et envoie le résultat à ses membres. Chaque membre (capteur) compare ensuite le résultat avec sa propre valeur et si la différence dépasse un certain seuil, la signature partielle sur la moyenne est effectuée et envoyée au *CH*. Le *CH* combine ensuite toutes les signatures en une signature complète et transmet le résultat à la station de base. Ce régime entraîne des coûts élevés de communication afin de

valider les données, et ne peut supporter que la fonction moyenne comme fonction d'agrégation. Notre système permet de réduire considérablement la consommation d'énergie due à la réduction du nombre de messages envoyés nécessaires pour valider les données, et peut prendre en charge toutes les fonctions d'agrégation, décrites dans [91], que peut supporter un chiffrement homomorphique additif.

Dans [71], les auteurs ont proposé un chiffrement homomorphique symétrique dans lequel une addition du message en clair à la clé en cours modulo un nombre très large est réalisée pour le cryptage. Pour le décryptage, la station de base a besoin exactement des mêmes clés utilisées lors du chiffrement pour obtenir l'agrégat des messages en clair. Leur solution assure la confidentialité de bout en bout, mais l'intégrité des données n'avait pas été abordée. Aussi, il y a un problème d'identité des capteurs parce que la station de base doit savoir exactement quel sont les capteurs qui font partie de l'agrégat afin de soustraire les clés correspondantes. Dans notre solution, nous fournissons à la fois la confidentialité et l'intégrité des données, et en raison de l'utilisation du chiffrement à clé publique l'*ECEG*, notre régime n'a pas de problème d'identifiant du chiffrement symétrique.

Le travail dans [73] fournit à la fois la confidentialité et l'intégrité de bout en bout. Pour la confidentialité, les auteurs utilisent l'*ECEG* et pour l'intégrité, ils utilisent un algorithme de signatures basée sur les groupes bilinéaires [25], les caractéristiques de cet algorithme sont que le vérificateur final doit connaître non seulement les données individuelles, mais également la clé publique utilisée pour la signature afin de vérifier la signature globale finale. A cet effet, les auteurs utilisent une fonction d'encodage permettant à la station de base d'extraire les données individuelles. Le problème de cette fonction est que si le nombre de nœuds augmente, la taille du message à chiffrer augmente aussi et conduit à une surcharge de calcul. Dans notre système, la vérification est effectuée saut à saut en utilisant *HMAC* qui conduit à moins de calculs par rapport à un tel algorithme de signature. Aussi, la taille du message à chiffrer est toujours la même dans *RSAED*.

Dans [75], les auteurs ont proposé l'intégrité de bout en bout en utilisant un *ECDSA* modifié (une signature *ECDSA* sans fonction de hachage) afin de permettre l'opération d'addition sur les signatures. Leur système conduit à une importante charge de calcul et de communication parce que le paquet inclut le chiffré, la signature et la clé publique. De plus, leur système ne peut vérifier que l'agrégat final. Si la vérification échoue, alors un nombre important de paquets légitimes sera perdu, ce qui conduit à un gaspillage important des ressources du réseau. Dans notre système, la vérification saut à saut économise la bande passante et aussi le calcul, et permet au vérificateur de rejeter les paquets malicieux. Par conséquent, les paquets légitimes peuvent atteindre la station de base et la robustesse du RCSF est améliorée.

6. Conclusion

Dans ce chapitre, nous avons proposé une nouvelle solution appelée *RSAED* pour sécuriser les données agrégées dans les RCSFs. *RSAED* est basée sur un algorithme de chiffrement homomorphique asymétrique et sur les signatures *MAC*. Cette solution assure une

confidentialité de bout en bout et une intégrité saut à saut. Par conséquent, la robustesse du réseau est améliorée et l'énergie des capteurs est préservée.

La confidentialité de bout en bout à été assurée à travers l'*ECEG*, un chiffrement homomorphique permettant le traitement direct sur les données chiffrées. L'agrégateur exécute l'opération homomorphique et n'a pas accès aux données en clair. Cette propriété est largement désirée dans des applications militaires ou toutes autres applications où la confidentialité des informations échangées est d'une importance primordiale. L'intégrité saut à saut empêche les attaquants de participer malicieusement à la communication mais surtout résout le problème du rejet total des données; un problème qui conduit à un gaspillage considérable des précieuses ressources du capteur.

Les propriétés de l'algorithme *ECEG* nous ont permis de proposer un calcul distribué de l'opération homomorphique dans *RSAED*. Le nœud d'agrégation constitue la pierre angulaire d'un RCSF, améliorer sa durée de vie améliore la durée de vie de tout le réseau. Les résultats de l'implémentation ont montré que ce calcul a conduit à une réduction de la charge de calcul et de communication au niveau des agrégateurs et aussi le délai de bout en bout par un facteur de 1/2.

L'implémentation montre que la multiplication des points par un scalaire demeure l'opération principale des algorithmes basés sur l'*ECC*. Enfin, la décompression des points est une opération trop coûteuse pour les capteurs. Cependant, dans les réseaux peu fiables cette technique demeure une alternative.

Dans le chapitre suivant, nous allons nous focaliser sur l'implémentation de l'*ECEG*, utilisé par *RSAED*, son efficacité et sa sécurité contre les attaques par canaux cachés.

Chapitre 4 : Deuxième contribution: Implémentation efficace et sécurisée de l'algorithme ECEG pour une sécurité de bout en bout des RCSFs

1. Introduction

Dans le secteur des réseaux de capteurs sans fil, la recherche sur la sécurité des données agrégées a augmenté de façon spectaculaire au cours des dernières années. Cette recherche est motivée principalement par les caractéristiques très spéciales que possède ce type de réseaux. Les solutions de bout en bout connues sous le nom de *CDA* (*Concealed data aggregation*) ont été proposées afin de fournir une confidentialité entre les nœuds capteurs et la station de base. Des recherches [58] ont montré que l'*ECEG* est le cryptosystème le plus approprié pour les réseaux de capteurs fournissant un haut niveau de sécurité basé sur la cryptographie sur les courbes elliptiques. Cependant, le temps d'exécution demeure l'inconvénient majeur en raison de la complexité des opérations de l'*ECC* et les ressources informatiques limitées du capteur. Par conséquent, une mise en œuvre efficace de ces opérations est alors cruciale.

Dans ce chapitre, nous présentons une mise en œuvre efficace et sécurisée de l'*ECEG* sur le capteur MicaZ en proposant une version améliorée de la bibliothèque TinyECC, dans laquelle une multiplication par un scalaire, rapide et sécurisée contre les attaques par canaux cachés (en particulier l'analyse simple de l'énergie), est utilisée. Aussi, un algorithme de décompression de points nécessaire pour l'agrégation est utilisé afin de calculer les racines carrées de manière efficace, satisfaisant $p = 3 \bmod 4$. Nos résultats [90] montrent qu'un chiffrement sécurisé de l'*ECEG* ne prend que 1.29s, ce qui est nettement meilleur par rapport aux implémentations de la littérature sur une plate forme 8 bits.

2. L'agrégation des données et la confidentialité

Comme nous l'avons souligné dans les précédents chapitres, le problème majeur de l'agrégation est posé quand on veut fournir certains services de sécurité telle que la confidentialité. Les réseaux de capteurs ont des caractéristiques particulières qui sont différentes des autres types de réseaux: (i) ils ont des contraintes de ressources, ce qui rend le choix des algorithmes de sécurité quelque peu difficile; (ii) ils sont souvent déployés dans des zones hostiles et sans surveillance, ce qui les rend sujets à plusieurs types d'attaques. Dans un scénario d'agrégation par exemple, les nœuds d'agrégation effectuent l'agrégation des données concernant plusieurs nœuds du réseau, ce qui les rendent plus attrayants pour un adversaire que les autres nœuds. Ces nœuds utilisent une liaison sans fil, ce qui peut permettre à un attaquant d'intercepter les paquets et même participer au processus d'agrégation [33,34,35]. Par conséquent, la conception d'un protocole de sécurité pour l'agrégation des données avec une mise en œuvre efficace, constitue donc un challenge.

Pour fournir une confidentialité pour l'agrégation dans les RCSFs, toutes les données transmises doivent être chiffrées en utilisant un algorithme solide et efficace. Traditionnellement, le chiffrement est effectué saut à saut c'est-à-dire chaque nœud

intermédiaire déchiffre, exécute la fonction d'agrégation puis chiffre le résultat. Cependant, cette approche qui est utilisée dans [49,49,54] a quelques limitations. En effet, les données sont agrégées dans leurs formes en clair, ainsi un agrégateur compromis peut accéder à ces données. Aussi, l'effort de chiffrement/déchiffrement aux niveaux des nœuds intermédiaires introduit un délai important. Dans une application militaire par exemple, le temps de délivrance et la protection des données contre les attaques passives sont d'une importance primordiale, en conséquence, une confidentialité de bout en bout est largement souhaitée. Au cours de la dernière décennie, plusieurs solutions offrant ce solide service ont été proposées [71,73,75]. Des solutions dans lesquelles les nœuds intermédiaires exécutent la fonction d'agrégation directement sur les données chiffrées et n'ont pas accès aux données en clair. Ces solutions impliquent l'utilisation du chiffrement homomorphique. Plusieurs algorithmes existent dans la littérature [44], mais ils ne sont pas tous adaptables pour les RCSFs. Un exemple très connu est *RSA*, un chiffrement homomorphique asymétrique dans un groupe multiplicatif. Cependant, *RSA* est un algorithme déterministe qui ne peut atteindre le plus haut niveau de sécurité contre les attaques passives, à savoir *IND-CPA*. Aussi, l'expansion des chiffrés du *RSA* entraînerait une charge de communication non-négligeable en raison de la taille de la clé. Le chiffrement homomorphique symétrique est certainement la meilleure solution pour les RCSFs à cause de sa simplicité et son efficacité en termes de ressources utilisées. Cependant, la plupart des algorithmes ont été cryptanalysés [44]. Par conséquent, d'un point de vue sécurité, il est plus intéressant d'employer des primitives plus solides tels que les systèmes de chiffrement à clé publique.

Comme *RSA*, il y a plusieurs algorithmes asymétriques qui possèdent la propriété homomorphique et contrairement à *RSA*, ils sont non seulement efficaces mais aussi ils peuvent atteindre le niveau *IND-CPA*. Les auteurs de [58] ont étudié et analysé la pertinence de plusieurs systèmes de cryptage homomorphes asymétriques pour une sécurité de bout en bout dans les RCSFs. Ils ont montré que l'*ECEG* est le plus adaptable. Récemment, l'implémentation efficace des opérations de l'*ECC* a fait l'objet de plusieurs recherches. Les auteurs de [95] ont démontré la faisabilité de l'*ECC* sur un capteur sans fil, et ceux de [59] ont montré que l'*ECEG* est praticable dans un RCSFs. Les avantages principaux de l'*ECEG* sont (i) il possède la propriété homomorphique nécessaire pour la sécurité de bout en bout, (ii) il peut contrer les *CPA*, (iii) il produit des chiffrés réduit par rapport à d'autres algorithmes asymétriques, (iv) il est scalable et efficace en termes de mémoire, et (v) il est résistant à l'attaque par capture vue les spécificités du chiffrement à clé publique. Cependant, l'inconvénient majeur de l'*ECEG* est la charge de calcul. En effet, les opérations de l'*ECC* sont un peu complexes et un capteur sans fil est alimenté par une batterie. Sachant que le temps d'exécution est traduit directement en consommation d'énergie, la réduction de la complexité en temps de ces opérations est donc cruciale afin de préserver l'énergie du capteur et par conséquent prolonger sa durée de vie.

Durant la dernière décennie, une nouvelle classe d'attaques appelée attaque par canaux cachés (*Side Channel Attack*) est apparue. Cette attaque permet à un attaquant d'exploiter les fuites d'informations telles que le temps et la consommation d'énergie durant les opérations cryptographiques, qui sont fortement corrélés avec les bits de la clé utilisée. Les attaques contre l'implémentation constituent une réelle menace pour les RCSFs. Dans [31], les auteurs

ont montré par la pratique la faisabilité de ces attaques dans le contexte des RCSFs. Ils ont souligné que des contremesures doivent être prévues sur le capteur afin de compliquer la récupération de la clé. Par conséquent, sécuriser les réseaux de capteurs ne nécessite pas seulement des algorithmes efficaces, mais également une mise en œuvre sécurisée de ces algorithmes contre ces attaques. En particulier, ces auteurs ont indiqué que l'implémentation de l'*ECC* sur un capteur est vulnérable à l'attaque par analyse simple de l'énergie.

Dans ce chapitre, nous présentons une implémentation logicielle efficace et sécurisée de l'*ECEG* sur un capteur MicaZ [2]. Notre contribution surpasse les implémentations logicielles existantes de l'*ECC* en général tels que [97,102,103], et la mise en œuvre de l'*ECEG* en particulier [59], par ce qui suit: d'abord, dans notre implémentation, nous utilisons une multiplication des points rapide afin de réduire le temps d'exécution de l'*ECC*, et donc de l'*ECEG*. Deuxièmement, nous accélérons l'opération homomorphique de l'*ECEG* aux niveaux des nœuds d'agrégation en utilisant un algorithme de décompression de points rapide. Enfin, grâce à la multiplication utilisée, notre mise en œuvre est sécurisée contre les fuites de clé, et par conséquent, elle permettra aux développeurs de mettre en œuvre en toute sécurité d'autres services de sécurité tels que l'intégrité des données de bout en bout, ce qui peut impliquer l'utilisation de l'algorithme *ECDSA*, voir par exemple la référence [73].

3. Cryptographie sur les courbes elliptiques

Dans cette section, nous allons présenter une brève description de l'*ECC* et nous étudions la multiplication des points par un scalaire, ainsi que son efficacité et sa sécurité contre les attaques par canaux cachés.

3.1. Définition

La cryptographie sur les courbes elliptiques introduite dans [104] est une approche à clé publique basée sur des concepts algébriques liés aux courbes elliptiques sur les corps finis. Il existe deux types de corps dans lesquelles les courbes elliptiques sont définies, les corps premiers F_p et les corps binaires F_{2m}. Dans ce manuscrit, nous allons nous intéresser seulement aux corps premiers. Soit F_p un corps premier, avec $p > 3$, une courbe elliptique non super-singulière E sur F_p est définie comme la solution de $(x, y) \in F_p * F_p$ de l'équation:

$$y^2 = x^3 + ax + b \bmod p$$

Avec $a, b \in F_p$ tel que $4a^3 + 27b^2 \neq 0 \pmod p$, et un point spécial ∞ appelé *point à l'infini*. Le groupe des points forme un groupe abélien muni de l'opération d'addition. L'addition de deux points quelconques de la courbe donne un autre point appartenant à la même courbe. Cette opération est définie comme suit: Soit $P_1 = (x_1, y_1)$ et $P_2 = (x_2, y_2)$, le résultat d'addition est $P_3 = (x_3, y_3)$ avec:

$$x_3 = \lambda^2 - x_1 - x_2 \bmod p$$

$$y_3 = \lambda (x_1 - x_3) - y_1 \bmod p$$

Et

$$\lambda = ((y_2 - y_1)/(x_2 - x_1)) \bmod p, \quad si\ P_1 \neq P_2 \text{ (Addition)}$$

$$\lambda = ((3x_1^2 + a)/2y_1) \bmod p, \text{ si } P_1 = P_2 \text{ (Doublement)}$$

Notons que l'inverse d'un point P_1 est $-P_1 = (x_1, -y_1)$ et $P_1 + \infty = \infty + P_1 = P_1$. Le groupe permet *l'addition des points* (*ECADD : Elliptic Curve ADDition*) et aussi *le doublement des points* (*ECDBL : Elliptic Curve Doubling*). Le produit $Q = k.P$ d'un point de la courbe avec un scalaire k, appelé *Multiplication de Point par un Scalaire* (*MPS*), consiste en une séquence d'additions et de doublements de points. Ces derniers appellent à leur tour les opérations arithmétiques du corps correspondant F_p. La sécurité de tous les algorithmes *ECC*, basée sur l'*ECDLP* défini précédemment, reste un problème difficile à résoudre surtout quand les paramètres de la courbe sont rigoureusement choisis [105].

3.2. Systèmes de coordonnées

L'addition et le doublement des points nécessitent une inversion qui est une opération trop coûteuse. Cependant, l'utilisation des coordonnées projectives peut éviter cette opération. Dans ce système, chaque point est représenté par trois coordonnées (X, Y, Z) correspondant à la représentation affinée $(X/Z, Y/Z)$ avec $Z \neq 0$. L'équation projective est:

$$Y^2Z = X^3 + aXZ^2 + bZ^3$$

Les coordonnées projectives jacobiennes, un cas particulier des coordonnées projectives peuvent donner des formules plus efficaces, où la représentation correspond au point $(X/Z^2, Y/Z^3)$. L'équation projective devient:

$$Y^2 = X^3 + aXZ^4 + bZ^6$$

Avec cette représentation, un doublement de points nécessite seulement 4 multiplications et 4 élévations au carré (Voir Tableau 11). Tandis que l'addition peut s'effectuer en utilisant les coordonnées mixtes. Cette méthode requiert seulement l'exécution de 8 multiplications et 3 élévations au carré [105].

Addition		**Doublement**	
A+A -> A	*1I, 2M, 1S*	*2A -> A*	*1I, 2M, 2S*
P+P -> P	*12M, 2S*	*2P -> P*	*7M, 3S*
J+J -> J	*12M, 4S*	*2J -> J*	*4M, 4S*
J+A -> J	*8M, 3S*		

A - Affine, ***P*** - Standard projective, ***J*** - Jacobian,
I - Field inversion, ***M*** - Multiplication, ***S*** - Squaring.

Tableau 11. Coût des opérations

3.3. Les attaques par canaux cachés sur l'ECC

Les attaques par canaux cachés profitent des fuites d'information involontairement divulguées par un dispositif de cryptage. Ces fuites peuvent être utilisées pour lancer différents types d'attaques telles que l'analyse simple de l'énergie (*SPA : Simple Power Analysis*) et l'analyse différentielle de l'énergie (*DPA : Differential Power Analysis*). *SPA* analyse un seul tracé de la consommation d'énergie durant l'exécution de l'algorithme, tandis que *DPA* est un ensemble d'analyses statistiques à travers plusieurs tracés issus de

l'exécution. Tous les algorithmes *ECC* impliquent l'exécution d'une *MPS* et généralement un scalaire pseudo-aléatoire est utilisé (Par exemple *ECEG*, *ECDSA*...). Donc, l'attaque *DPA* n'est pas faisable cependant l'attaque *SPA* est possible sur la *MPS*. Dans [31], les auteurs prouvent par la pratique que cette attaque est faisable dans le contexte des RCSFs, et ils ont souligné que des contremesures doivent être prévues sur le capteur afin de compliquer la révélation de la clé. L'objectif d'une simple analyse de l'énergie est de déduire la séquence d'adition et de doublement de points à partir du tracé énergétique durant l'exécution de la *MPS*. Comme nous pouvons le constater à partir du Tableau 11, ces deux opérations ont des coûts différents, où une opération d'addition prend plus de temps qu'un doublement (Voir Figure 20).

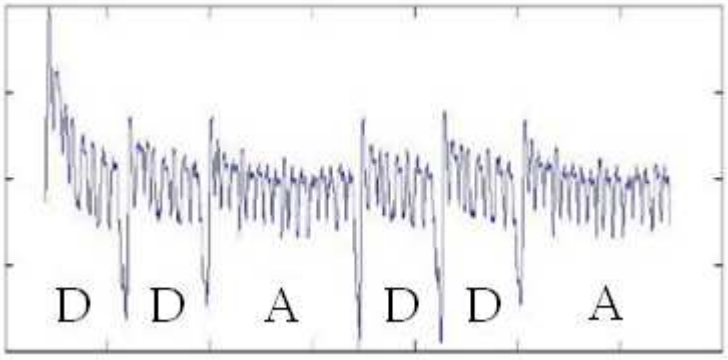


Figure 20. Tracé énergétique d'une séquence d'additions et de doublements de points sur F_p [105]

Une attaque telle que *SPA*, basée sur l'observation d'un seul tracé, peut facilement déduire les bits de la clé utilisée. Dans le but de faire face à *SPA*, des contremesures doivent être incorporées dans l'implémentation afin de rendre le temps de traitement de la *MPS* indépendants des opérandes. Dans la section suivante, nous examinons certains des algorithmes de la *MPS* et nous discutons leur résistivité à *SPA*.

3.4. La Multiplication des Points par un Scalaire MPS

$$Q = P + \ldots + P \text{ (k times)}$$

Par définition, la *MPS* c'est l'addition d'un point à lui-même *k* fois. La *MPS* est l'analogue de l'exponentiation modulaire dans les groupes multiplicatifs. L'algorithme de base pour effectuer cette opération, appelé *double-and-add algorithm* (Voir Algorithme 6), est l'équivalent à l'algorithme *Square-and-multiply algorithm* dans les groupes multiplicatifs.

Algorithme 6 : Double-and-add
Entrée: point P sur E (Fp); k = $(k_{l-1},\ldots,k_0)_2$
Sortie: Q = kP
1. Q = P
2. Pour i = l - 2 jusqu'à 0 faire
2.1 Q= 2Q
2.2 Si k_i= 1 alors Q= Q + P
3. Retourner Q

Le nombre d'additions nécessaires est un de moins que le poids de hamming[9] *h* de *k* noté *h(k)*, en moyenne *h(k)=l/2*, tel que *l* représente le nombre de bits de *k*. Tandis que le nombre de doublements est un de moins que la taille binaire de *k*. Soit un coût total de:

$$(h(k)-1)\ ECADD + (l-1)\ ECDBL$$

Comme mentionné précédemment, ces deux opérations différent en termes de coût. L'analyse du tracé de la consommation d'énergie révélera la séquence des opérations exécutée par l'Algorithme 6 (attaque *SPA*), ainsi que les bits de *k*. Par exemple la séquence *ECDBL-ECDBL* veut dire $k_i = 0$ et *ECDBL-ECADD* veut dire $k_i = 1$.

La seconde approche est la représentation signée du scalaire appelée *NAF* (*Non Adjacent Form*). Une méthode pour améliorer significativement le temps d'exécution de la *MPS*, consiste à réduire le poids de hamming de *k*, *h(k)*. *NAF*, une représentation dans laquelle il n'existe par de bits non nuls adjacents, a l'avantage de réduire *h(k)* à *l/3* en moyenne; ce qui engendre moins d'additions par rapport à l'algorithme 6. Malgré cette amélioration, cette méthode est vulnérable à l'attaque *SPA*. Ainsi, un attaquant qui suit la consommation d'énergie lors de l'exécution de la *MPS* peut déduire les bits nuls de *k*, et le fait que seule *l/3* bits de *k* sont non nuls en moyenne, une recherche exhaustive peut facilement s'effectuer sur l'intervalle calculé.

Plusieurs approches ont été proposées pour contrer *SPA* [106-108]. Ces contremesures impliquent une modification de la *MPS* de telle sorte que la même séquence d'opérations est effectuée quel que soit le bit courant, égal à 0 ou 1. Malheureusement, cela signifie ralentir la performance globale de la multiplication et un compromis doit être trouvé entre le temps d'exécution et la sécurité fournie. L'Algorithme 7 suivant est la version sécurisée de l'Algorithme 6.

Algorithme 7 : Double-and-add always

Entrée: point P sur E (Fp); $k = (k_{l-1},\ldots,k_0)_2$
Sortie: Q = kP
1. $Q_0 = P$
2. Pour i = l - 2 jusqu'à 0 faire
   2.1 $Q_0 = 2Q_0$
   2.2 $Q_1 = Q_0 + P$
   2.3 $Q_0 = Q_{ki}$
3. Retourner Q_0

Il est clair qu'avec cet algorithme, la *MPS* effectue exactement un doublement et une addition de points dans chaque itération, et ceci grâce à l'opération fictive incorporée dans la boucle principale. Le coût de l'Algorithme 7 est alors:

$$(l-1)ECADD + (l-1)\ ECDBL$$

La troisième approche pour améliorer le temps d'exécution est d'utiliser les ressources matérielles en utilisant des précalculs (offline pour une multiplication à base fixe, online pour une multiplication à base variable). Le temps d'exécution de l'Algorithme 6 peut être

[9] Le poids de hamming représente le nombre de bits non nuls dans une représentation binaire.

réduit en utilisant une méthode de fenêtre, appelée WM (*The Window Method*) permettant de traiter la représentation de k, w bits à la fois (Voir Algorithme 8).

Algorithme 8 : The window method

Entrée: point P sur E (Fp),w, d = [l/w], k = $(k_{d-1},\ldots,k_0)_2$w
Sortie: Q = kP
1. $P_0 = P$
2. Précalculer: Pour i = 1 à 2^w-1 faire: $P_i = P_{i-1} + P$
3. Q=0
4. Pour i = d - 1 jusqu'à 0 faire
   4.1 $Q= 2^wQ$
   4.2 $Q= Q + P_{kd}$
5. Retourner Q

Notons que l'Algorithme 6 est le cas particulier de l'Algorithme 8 avec $w = 1$. Le nombre d'additions requis est $2^w-1 / 2^w (d-1)$ en moyenne et le nombre de doublements est $w (d-1)$ si nous supposons que $k_{d-1} \neq 0$. Par conséquent, le coût moyen de cette opération est d'environ:

$$2^w-1/2^w (d-1) ECADD + w (d-1) ECDBL$$

A noter également que cette durée concerne seulement l'étape d'évaluation, le coût de l'étape du précalcul peut être négligée si le point de base est fixe, sinon le temps de précalcul doit être considérée. Même si la fréquence des termes nuls dans l'expansion de k est réduite en comparaison avec l'algorithme 6, WM n'est pas à l'abri des attaques SPA. L'analyse du tracé de la consommation d'énergie peut révéler à l'attaquant où $k_d = 0$, ce qui constitue une bonne information pour la recherche de la clé.

L'une des solutions efficaces est celle proposée par Lim et Lee dans [109] pour l'exponentiation modulaire, qui peut aussi s'appliquer efficacement sur la *MPS* (Voir Algorithme 9). Pour représenter un entier, la méthode Comb utilise une matrice binaire de w lignes et d colonnes.

Algorithme 9 : The Comb method

Entrée: point P sur E (Fp), une fenêtre w>2 et un entier k
Sortie: Q = kP
L'étape de précalcul
1. P_0 = *Calculer* $[b_{(w-1)}, \ldots, b_1, b_0]*P$ *pour tout* $(b_{(w-1)}, \ldots, b_1, b_0)$ *dans* Z_2^w.
2. Ecrire $K = (K^{(w-1)} \| \ldots \| K^1 \| K^0)$ ajouter des 0 à gauche si nécessaire, où chaque $|K^j| = d$.
   Soit K_i^j le i-ème bit de K^j
   Définir $K_i = [K_i^{w-1}, \ldots, K_i^1, K_i^0]$
3. Q=0
L'étape d'évaluation
4. Pour i = d - 1 jusqu'à 0 faire
   4.1 Q= 2Q
   4.2 $Q= Q + K_iP$
5. Retourner Q

Le nombre d'additions nécessaires est de $2^w-1/2^w(d-1)$ en moyenne et le nombre de doublements est $d-1$ si nous supposons que $k_{d-1} \neq 0$. Le coût moyen de la méthode Comb est donc:

$$2^w-1/2^w\ (d-1)\ ECADD + (d-1)\ ECDBL$$

La méthode Comb effectue une addition et un doublement si $K_i \neq 0$, et seulement un doublement dans l'autre cas. L'analyse de la consommation d'énergie permet à l'attaquant de savoir si le bloc de bits K_i est nul ou pas. Par conséquent, la méthode Comb est aussi vulnérable à *SPA*.

A noter que pour les Algorithmes 8 et 9, la stratégie de précalcul et le choix de la fenêtre w varie, en fonction de l'application cible.

3.5. La décompression des points

Dans l'équation $y^2=x^3+ax+b\ mod\ p$, les points de la courbe sont représentés en coordonnées affines. Comme nous avons vu précédemment, plusieurs systèmes de coordonnée existent. Afin d'optimiser la transmission et le stockage, la compression des point a été proposée et standardisée dans *IEEE P1363* [110]. Le Tableau 12 présente le nombre de bits nécessaires pour la transmission et le stockage des points représentés dans différents systèmes de coordonnées. Par exemple, pour stocker un point coordonnées affines, seul $(log2\ p)+1$ bits sont nécessaires. Dans ce scénario de compression, seule la coordonnée x est stockée conjointement avec un bit de la coordonnée y qui définit l'une des deux racines carrées possibles calculées selon l'équation de la courbe elliptique.

Système de coordonnées	Nombre de bits
Affine	2 ($\log_2$ p)
Projective	3 ($\log_2$ p)
Jacobian	3 ($\log_2$ p)
Modified Jacobian	4 ($\log_2$ p)
Chudnovsky Jacobian	5 ($\log_2$ p)

Tableau 12. Systèmes de coordonnée Vs Nombre de bits

Il s'ensuit que pour la conversion de la forme compressée d'un point à sa forme non compressée, il est nécessaire de trouver la racine carrée de $x^3 + ax + b$ dans $F(p)$. L'efficacité de la racine carrée dans $F(p)$ dépend fortement du nombre premier p choisi. Nous nous concentrons sur le cas $p \equiv 3\ (mod\ 4)$, parce que, pour les courbes standardisées, $p \equiv 3\ (mod\ 4)$ est le plus souvent le cas.

Généralement, les algorithmes *ECC* tels que *ECDSA*, *ECDH (Elliptic Curve Diffie-Helman)*, etc, le cas aussi pour l'*ECEG* (Voir Algorithme 2), produisent en sortie des points, et ces points doivent être transmis à un récepteur. Les capteurs sans fil ont une bande passante radio limitée, de longs paquets causeraient ainsi un taux important de pertes dans un réseau sans fil non fiable tel que le RCSF. Aussi, la consommation d'énergie augmente linéairement avec la taille des données transmises. Par conséquent, réduire la taille de la sortie en compressant ces points (Voir Algorithme 10) peut donc être très bénéfique.

L'Algorithme de Décompression des Points (*ADP*) requiert le calcul d'une racine carrée modulo *p*, et en considérant les ressources informatiques limitées du capteur, ce calcul doit être effectué de manière efficace. Le calcul nécessite une exponentiation modulaire ($Y_p=Z^{(p+1)/4} \ mod \ p$), un calcul similaire à la *MPS* parce que tous les algorithmes utilisés pour améliorer la MPS peuvent s'appliquer sur l'exponentiation, mais les deux opérations sont un peu différentes. En effet, le point de base *G* et le modulo *p* sont connus à l'avance, ainsi le précalcul est possible puisque les multiples de *G* peuvent être précalculés et sauvegardés. Dans l'algorithme de l'exponentiation, l'exposant $e = (p+1)/4$ et le modulo *p* sont connus à l'avance mais pas la base. Donc, de telles optimisations ne sont pas envisageables. Pour plus de détails, consulter la référence [111].

Algorithme 10 : Compression et décompression des points

Compression
Entrée: un point P (X_p,Y_p) dans E
Sortie: le point compressé P (X_p,Y'_p)
1. $Y'_p = Y_p \bmod 2$
2. Retourner (X_p,Y'_p)

Décompression
Entrée: un point compressé P (X_p,Y'_p)
Sortie: le point correspondant P (X_p,Y_p) dans E
1. $Z= X_p^3+aX_p +b \bmod p$
2. $Y_p =Z^{(p+1)/4} \bmod p$
3. Si $(Y_p = Y'_p \bmod 2)$ alors $Y_p =Y'_p$
   sinon $Y_p = p -Y'_p$.
4. Retourner (X_p,Y_p).

3.6. TinyECC

La bibliothèque TinyECC fournit toutes les opérations *ECC* dans le domaine F_p, des algorithmes bien connus tels que *ECDH* et *ECDSA* et aussi un nombre important d'optimisations pouvant être utilisé selon l'application. Ces optimisations concernent la réduction de Barrett pour une réduction modulaire efficace, la multiplication et l'élévation au carré hybrides, le système de coordonnées projectives pour des additions et doublement de points rapides et enfin la *SWM* pour la multiplication. Plus de détails sont fournis dans [97].

4. Notre implémentation de l'ECEG pour les RCSFs

Le chiffrement dans l'*ECEG* implique le calcul de deux *MPS*. Aussi, le chiffré contient deux points appartenant à la courbe, qui doivent être compressés. Dans ce qui suit, nous présentons notre implémentation efficace des deux opérations les plus coûteuses de l'*ECEG* à savoir *MPS* et *ADP*, nécessaires pour la confidentialité de bout en bout dans les RCSFs. Nous soulignons que le temps d'exécution et la résistance à *SPA* sont les critères les plus importants dans ce travail, tandis que la consommation de la mémoire est la troisième priorité. Dans notre implémentation, nous utilisons TinyECC ainsi que ces optimisations et nous nous focalisons sur la *MPS* et l'*ADP*.

4.1. Implémentation de la multiplication des points

Les points *G* et *Y* sont connus à l'avance et, par conséquent, les précalculs sont possibles si éventuellement la mémoire est disponible. Une des multiplications les plus efficaces à base fixe est la méthode Comb introduite dans [109] pour calculer l'exponentiation, définie précédemment. La méthode Comb nécessite le stockage de $2^{w}-1$ points dans la phase de précalcul. Pour un point de base fixe, le précalcul est effectué une seule fois et son coût est négligeable, et pour une base variable ce coût doit être considéré. Nous soulignons que la force de cet algorithme est dans la phase de précalcul.

Hedabou et al. [112], ont proposé une méthode Comb sécurisée contre l'analyse simple de l'énergie qui, non seulement améliore les performances, mais aussi évite cette attaque (Voir Algorithme 11). Ces auteurs ont proposé une nouvelle représentation du scalaire *k*, traduite par une séquence de bits non nul avec des 1 et -1. Leur méthode nécessite seulement le stockage de 2^{w-1} points au lieu des $2^{w}-1$ pour la méthode Comb originale, et ce, sans influence sur le temps d'exécution. L'algorithme calcule la nouvelle représentation uniquement pour les entiers impairs. Pour calculer la multiplication $k*P$ pour un entier k pair, il suffit de générer la nouvelle représentation pour l'entier impair $k-1$ puis calculer $(k-1)*P$ en utilisant la méthode Comb [113]. La valeur de la multiplication peut être trouvée en calculant $k*P = (k-1)*P+P$.

Algorithme 11 : La MPS sécurisée de Hedabou et al. [112]

Entrée: P La représentation binaire $(k_{l-1},\ldots,k_0)$ de k, P et w>2
Sortie: Q = kP
1. d=l/w
2. Si k est pair faire k = k-1
3. Rajouter w*d-l bits nuls à la gauche de la représentation binaire de k
4. $k_0' = k_0$
5. Pour i = 1 à l – 2 faire
   5.1 Si $k_i = 0$ faire $k'_i = k'_{i-1}$ et $k'_{i-1} = -k'_{i-1}$
   5.2 Sinon faire $k'_i = k_i$ et $k'_{i-1} = k'_{i-1}$
6. $k'_{l-1} = k_{l-1}$
7. Calculer kP en utilisant l'algorithme 9
8. Si k est impair retourner Q, sinon retourner Q + P

Exemple:

Soit $k = 110$ et $w = 3$, le scalaire est pair, donc on utilise $k=109=(1101101)_2$, on a aussi $d=3$. On commence par rajouter *2* zéros à gauche *(001101101)* puis on calcule la nouvelle représentation pour obtenir *(1-1-11-11-1-1)*, la matrice qui représente le scalaire est donc:

$$\begin{Bmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ 1 & -1 & -1 \end{Bmatrix}$$

Ensuite, on calcule en utilisant la méthode Comb:

$[K_2^2, K_2^1, K_2^0]$ $P = [1, 1, 1]$ $P = 2^6P + 2^3P + P = 73P$

$[K_1^2, K_1^1, K_1^0]$ $P = [-1, 1, 1]$ $P = -2^6P + 2^3P + P = -55P$

$[K_0^2, K_0^1, K_0^0]\ P = [-1, -1, -1]\ P = -2^6P - 2^3P - P = -73P$

Et enfin on vérifie que:

$k'P = 2^2\ [K_2^2, K_2^1, K_2^0]\ P + 2\ [K_1^2, K_1^1, K_1^0]\ P + [K_0^2, K_0^1, K_0^0]\ P = 4*73P+2*(-55P)-73P = 109P$

Le résultat trouvé est pour k'. Afin de trouver kP il suffit d'ajouter un point à la fin de l'exécution, soit $k'P+P = 109P+P = 110P = kP$.

Dans notre implémentation, nous avons créé deux optimisations à savoir *COMB_MUL* et *SEC_COMB*, la première fait référence à la méthode Comb originale (Algorithme 9) et la seconde à sa version sécurisée proposée dans [112] (Algorithme 11). Pour les deux méthodes nous avons utilisé le système de coordonnées mixtes pour une *ECADD* efficace (Voir Tableau 11). Dans l'Algorithme 12, nous montrons comment calculer la *MPS* de [112] en utilisant les coordonnées mixtes. Soit *(w,d)*, la matrice représentant *k*, avec *w* la fenêtre utilisée et *d* le nombre de colonnes calculé en fonction de *l* et *w*.

Algorithme 12 : La MPS de [112] en utilisant les coordonnées mixtes

Entrée: Système de coordonnées Jacobienne pour le doublement des points et le système de coordonnées mixtes pour l'addition des points. Un entier positif k et 2^{w-1} points ($\mathbb{K}_n$) représenté en coordonnées affines.
Sortie: le point Q représenté en coordonnées affines tel que Q = kP
Pour i = d-2 à 0 par -1 faire:
Q = 2Q // J = 2*J
Q = Q + $\mathbb{K}_n$P // J = J+A
Convertir Q en coordonnées affines
Retourner Q.

Puisque tous les K_i^j sont non nuls, les opérations exécutées par l'algorithme 12 consistent en une séquence alternative d'additions et de doublement de points à savoir DADA…DADA. Ainsi le coût sera de *(d-1) ECADD + (d-1) ECDBL*. Par conséquent, cet algorithme ne permet pas des fuites d'informations sur le scalaire *k* pour l'attaque *SPA*. Le Tableau 13 présente les implémentations de l'*ECC* qu'on peut trouver dans la littérature.

Implémentation	P.p	Coût	MPS	SPA
[95]	-	*((l/2)−1)PADD + (l−1)PDBL*	D&A	Oui
[59]	*(t-1)+t(2^{w-1}-1)*	*(l/w+1)PADD + (l/t)PDBL*	I&M	Oui
[97,102]	2^w-1	2^w-1/2^w*(d-1)PADD + w(d-1)PDBL*	WM	Oui
[103]	2^w-1	2^w-1/2^w *(d-1)PADD + (d-1)PDBL*	Comb	Oui
Notre travail	2^{w-1}	*(d-1)PADD + (d-1)PDBL*	[112]	Non

Tableau 13. Implémentations de l'ECC dans les RCSFs

Nous pouvons constater que toutes les implémentations existantes présentées dans le Tableau 13 sont vulnérables à l'attaque *SPA* en raison des différences des coûts. Dans la section suivante, nous analysons et discutons le temps d'exécution de notre mise en œuvre des deux méthodes (sécurisée et non sécurisée) en utilisant TinyECC et le capteur MicaZ.

4.2. Implémentation de la décompression des points

Le texte chiffré de l'*ECEG* comme mentionné précédemment, contient deux points compressés, et une fois reçu par les nœuds d'agrégation, les points doivent être décompressés car l'opération homomorphique ne peut être effectuée sur la forme compressée des points. La décompression implique le calcul d'une exponentiation, et en raison des ressources limitées des capteurs, cette opération doit être soigneusement mise en œuvre. Une optimisation telle que la méthode Comb peut également être appliquée, mais contrairement à la multiplication, les précalculs, dans ce cas, sont exécutés online et conduisent à un temps d'exécution supplémentaire.

Récemment, les auteurs de [114] ont proposé une méthode de décompression pour les courbes satisfaisant $p = 3 \bmod 4$. Leur méthode est basée sur deux algorithmes, l'exponentiation des *repunits*[10] et la méthode des fenêtres glissantes (*SWM: Sliding Windows Method*) en utilisant la plus courte séquence. Dans la méthode proposée l'exposant est considéré comme k fenêtres de repunit de la forme $2^{Ni}-1$, et les Ni distincts forment l'ensemble W. Le but est de trouver la plus courte séquence de W et de précalculer tous les $B^a \bmod p$ en utilisant l'exponentiation des repunits avec $a = 2^{Ni}-1$. Enfin, *SWM* est utilisée pour calculer l'exponentiation finale. Leur algorithme est très efficace dans le cas des nombres premiers de Mersenne parce que k est très petit et encore plus si la plus grande fenêtre apparaît dans les bits les plus significatifs de l'exposant.

Dans notre implémentation, nous avons utilisé les paramètres du domaine *SECP160R1* recommandé par [99] qui satisfait $p = 3 \bmod 4$. La courbe *SECP160R1* possède les avantages cités dans le paragraphe précédent. En effet, nous avons un ensemble contenant une seule fenêtre $W = \{129\}$, ainsi la séquence la plus courte de W est $\{1, 2, 4, 8, 16, 32, 64,128,129\}$. Ensuite, nous calculons B^a où $a = 2^{129}-1$ en utilisant l'exponentiation des repunits qui coûte 8 multiplications (M) et 128 élévations au carré (S), et enfin, 29 élévations au carré supplémentaires sont nécessaires pour les bits nuls restants de l'expansion de k (voir Figure 21). Pour montrer l'efficacité de cette méthode, nous avons implémenté également la méthode *Square and multiply*, TinyECC utilise la méthode quaternaire pour la décompression dans *ECIES* (*Elliptic Curve Integrated Encryption Scheme*).

$$\text{Secp160r1: e} = (2^{129}-1)2^{29}$$

$$w_1 \xrightarrow{\cdot 2} w_2 \xrightarrow{\cdot 2^2} w_4 \xrightarrow{\cdot 2^4} w_8 \xrightarrow{\cdot 2^8} w_{16} \xrightarrow{\cdot 2^{16}} w_{32} \xrightarrow{\cdot 2^{32}} w_{64} \xrightarrow{\cdot 2^{64}} w_{128} \xrightarrow{\cdot 2} w_{129}$$

$$\text{Cost: 8 M} + 128\ \text{S and 29 S}$$

Figure 21. Le coût pour la courbe SECP160R1

[10] Les repunits sont les nombres ayant que des 1 dans leur représentation binaire.

5. Résultats et discussions

5.1. Performance de la MPS et l'ADP

Pour toutes les *MPS*, le tableau des points est calculé offline et le temps d'exécution est calculé à partir de plusieurs exécutions en utilisant une clé secrète aléatoire de 160 bits. Dans le Tableau 14, une comparaison avec les implémentations précédentes de la *MPS* est présentée. *P.p* et *w* font référence respectivement au nombre de points précalculés et à la fenêtre utilisée. La comparaison montre que notre implémentation logicielle est plus rapide par rapport aux précédentes sur MicaZ.

Implémentation	Alg.	P.p	W	Temps	Mémoire (octets)		SPA
					ROM	RAM	
Ugus et al. [59]	I&M	2	2	0.57s	3536	543	Oui
Gura et al. [95]	D&A	-	-	0.81s	3682	282	Oui
Wang et al. [102]	SWM	15	4	1.24s	n/a	n/a	Oui
Szczechowiak et al. [103]	Comb	15	4	1.27s	46100	1800	Oui
Liu et al. [97]	SWM	15	4	1.31s	9112	827	Oui
Notre travail	Comb	15	4	0.63s	9780	827	Oui
Notre travail	Sec_Comb	8	4	0.65s	10286	575	Non
Notre travail	Sec_Comb	16	5	0.52s	10834	869	Non

Tableau 14. Comparaison avec les implémentations précédentes

Dans [97], les auteurs ont implémenté une multiplication de point en utilisant la méthode des fenêtres *SWM*. Notre implémentation non sécurisée, en utilisant la méthode Comb et la même fenêtre, effectue une multiplication de points *52%* plus rapide, avec une consommation de RAM comparable et une légère augmentation de 7% de la taille du code. Le meilleur temps d'exécution s'explique par le nombre réduit des *ECDBL* nécessaires dans la phase d'évaluation de la méthode Comb. En effet, la vitesse est réduite de *wd -w- d +1* à savoir *117 ECDBL* dans notre cas. Nous notons que la surcharge dû au précalcul de la méthode Comb peut expliquer la légère augmentation de la taille du code. Notre implémentation sécurisée de la méthode Comb est presque aussi efficace que l'originale avec le même *w*, seules quelques additions font la différence. Comme le montre le tableau 14, le principal avantage de cette *MPS* est le nombre réduit de points précalculés nécessaires à savoir 2^{w-1} qui se traduit par une utilisation de mémoire *31%* plus petite que les implémentations non sécurisées avec une fenêtre semblable *w*. Aussi, lorsque nous augmentons la fenêtre *w* par *1*, le nombre de points précalculés est presque le même que les implémentations non sécurisées, l'utilisation de la mémoire augmente légèrement à savoir *5%*. L'avantage majeur est que le temps d'exécution est respectivement *60%* et *18%* plus rapide que *SWM* et la méthode Comb non sécurisée. La charge de précalcul explique le gain à l'étape de l'évaluation où seuls *d - 1* additions et *d - 1* doublements sont nécessaires, 31 *ECADD* et 31 *ECDBL* dans notre cas, la même charge explique également la légère augmentation de la taille du code. Cependant, comme les points sont fixes, l'augmentation

de la taille du code peut être évitée en effectuant les précalculs *offline* et distribuer les points à chaque capteur avant le déploiement.

Notre implémentation sécurisée effectue, par rapport à [102], une *MPS* 48% et 58% plus rapide, respectivement lorsque 8 et 16 points précalculés sont utilisés. Les auteurs dans [102] n'ont pas fournis des informations concernant la consommation ROM et RAM de leur mise en œuvre. Toutefois, étant donné que dans leur mise en œuvre, à peu près les mêmes techniques d'accélération que dans TinyECC ont été employées, nous supposons que notre mise en œuvre est similaire en termes de taille de code et de consommation de mémoire. Si nous comparons notre implémentation à [103], nous pouvons trouver que notre *MPS* sécurisée est 36% plus rapide, quand 16 points précalculés sont utilisés, tandis que la taille du code et l'utilisation de la mémoire sont respectivement 77% et 52% plus petit. L'augmentation du temps d'exécution de notre travail provient certainement des techniques d'accélération utilisées dans TinyECC, en particulier la méthode barrett pour la réduction modulaire.

Gura et al. [95], ont utilisé l'algorithme de base pour la *MPS*, un algorithme qui est vulnérable à *SPA*. Notre implémentation est non seulement sécurisée contre *SPA* mais aussi elle est 20% et 36% plus rapide, respectivement lorsque 8 et 16 points précalculés sont utilisés. Ce gain de rendement est dû à l'utilisation de la version sécurisée de la méthode Comb [112]. Par conséquent, beaucoup moins d'addition et de doublement sont effectuées. Malgré l'utilisation de l'algorithme de base, le temps d'exécution obtenu par [95] peut être expliqué par une implémentation en assembleur. Dans notre travail, basé sur TinyECC, l'assembleur est utilisé uniquement pour les opérations arithmétiques du corps fini.

Ugus et al. [59], ont utilisé une *MPS* en combinant la méthode entrelacée (*Interleaved*) et *Mutual Opposite Form* (I&M), la première méthode est utilisée pour réduire les doublements de points et la deuxième pour réduire les additions. Leur *MPS* reste vulnérable aux attaques *SPA*, la représentation signée utilisée avec 1, 0, -1 n'élimine pas la vulnérabilité à *SPA*. Une seule observation du tracé peut révéler à un attaquant où le bit de k est 0, et le fait que seulement un tiers des bits scalaires sont non nuls en moyenne, signifie que la recherche exhaustive pour le scalaire est à portée de main. Notre mise en œuvre quand 16 points précalculés sont utilisées n'est pas seulement sécurisée contre les attaques *SPA*, mais aussi 9% plus rapide. Cependant, l'utilisation de la ROM et la RAM sont accrus en raison de la surcharge de précalcul et les optimisations de bibliothèque TinyECC. Notre meilleur temps d'exécution peut être expliqué par les 43% de moins d'opérations d'addition et de doublement nécessaires à l'exécution de la multiplication, environ 23 *ECADD* et 23 *ECDBL* dans notre cas. Malgré les différences importantes en termes de nombre d'additions et de doublements, le temps d'exécution obtenu par [59] peut être expliqué par les techniques d'optimisation utilisées dans leur mise en œuvre.

Notre implémentation peut être utilisée dans tous les systèmes basés sur l'*ECC* en général, et où la résistivité *SPA* est primordial en particulier. Dans *ECDSA* [50] par exemple, le calcul de *r* fait usage de la clé *k* éphémère et le point de base *G*. L'entier *k* change pour chaque génération de signature et il est inconnu pour l'attaquant. Par conséquent, les attaques *DPA* ne sont pas possibles sur cette opération. Par contre les attaques *SPA* sont possibles, ainsi, la protection des clés éphémères est un aspect très important qui ne peut être négligé. En

conséquence, une mise en œuvre non protégée peut permettre à un attaquant la récupération de la clé privée via *SPA* (comme dans [31]). Finalement, la mise en œuvre de la *MPS* doit être sécurisée contre les attaques *SPA*, même si elle n'est utilisée que pour les signatures. Notre mise en œuvre de la multiplication est idéale pour cette application puisque les points précalculés sont calculés une seule fois et appelés lors de chaque génération de signature. Dans les RCSFs, le temps d'exécution influence directement sur la batterie du capteur. En effet, la consommation énergétique augmente linéairement avec le temps d'exécution [88,89]. Toujours suivant le modèle utilisé dans [97], nos *0.52s* pour la *MPS* se traduisent en une consommation de *23.48mj* tandis que les *1.31s* de TinyECC consomment *31.44mj*. Ainsi, notre implémentation contribue considérablement à l'amélioration de la durée de vie du capteur.

Dans le Tableau 15, nous montrons que notre implémentation de l'algorithme *Rep.exp&SWM* sur MicaZ est respectivement 42% et 26% plus rapide que l'algorithme de base et la méthode quaternaire. L'amélioration est due au nombre réduit de multiplications effectuées par l'algorithme à savoir respectivement 90% et 87% de moins que *S&M* et la méthode quaternaire, soit environ 71M et 53M dans notre cas. La taille du code est presque la même que la méthode quaternaire utilisée dans TinyECC pour *ECIES*. Notre application est plus efficace et par conséquent, elle permet une exécution plus rapide de la fonction homomorphique au niveau des nœuds intermédiaires.

Implémentation	Algorithme	Temps	Mémoire (octets)	
			ROM	RAM
Liu et al. [97]	S&M	0.548s	7572	237
Liu et al. [97]	Quaternary	0.426s	7735	237
Notre travail	Rep.exp&SWM	0.316s	7744	237

Tableau 15. Performances pour SECP160R1

5.2. Performances de l'ECEG pour les RCSFs

Nous avons développé une application TinyOS pour l'*ECEG* et sa propriété homomorphique. Le chiffrement ECEG implique deux MPS avec un scalaire de *n* bits, généré de façon aléatoire, et également une *MPS* pour mapper le message *m* en un point sur la courbe et ceci afin de permettre le fonctionnement au niveau du groupe de la courbe elliptique. Pour le test, on définit *m* d'une taille de 8 bits, le tableau des points est calculé *offline* dans l'initialisation de l'ECEG "*ECEG.init ()*", le tableau 16 montre la performance de "*ECEG.encrypt ()*" avec différentes *MPS*. L'opération homomorphique est effectuée au niveau des nœuds intermédiaires. Dans ce niveau, le nœud décompresse d'abord les textes chiffrés entrants puis effectue l'agrégation sur les données cryptées en utilisant l'opération d'addition sur les courbes elliptiques. Le tableau 17 montre la performance de "*ECEG.hom_add ()*" avec différents algorithmes d'exponentiation en considérant l'addition de deux chiffrés. Les tailles du code et de mémoire de notre application, présentées dans la Figure 22, sont obtenues par différentes *MPS*, tout en considérant l'algorithme *Rep.exp&SWM* comme algorithme d'exponentiation. Des travaux comme [73,75,78] pourront donc directement bénéficier de nos résultats.

Implémentation	Algorithme	W	Temps
Liu et al. [97]	SWM	15	2.84s
Notre travail	Comb	15	1.47s
Notre travail	Sec_Comb	8	1.53s
Notre travail	Sec_Comb	16	1.29s

Tableau 16. Performances de ECEG.Encrypt()

Implémentation	Algorithme	Temps
Liu et al. [97]	S&M	1.892s
Notre travail	Quaternary	1.499s
Notre travail	Rep.exp&SWM	1.243s

Tableau 17. Performances de ECEG.hom_add()

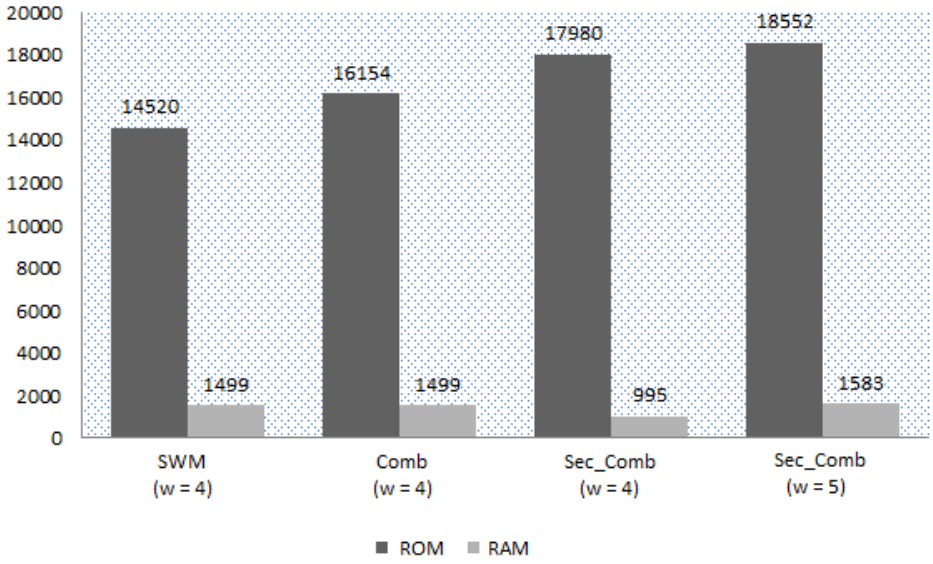


Figure 22. Les tailles (en octets) du code ROM et de mémoire RAM

5.3. Comparaison avec l'implémentation d'Ugus et al. [59]

Ugus et al. [59] ont présenté une implémentation optimisée de l'*ECEG* en utilisant un algorithme de multiplication scalaire basé sur la méthode entrelacée et *MOF* (*I&M*) qui reste comme mentionné précédemment vulnérables à l'attaque *SPA*. Si une attaque comme dans [31] s'exécute sur leur *MPS* avec succès, les chiffrements passés seront compromis. Notre mise en œuvre de l'ECEG est non seulement 9% plus rapide mais aussi sécurisée contre *SPA* (voir Tableau 18), et cela est dû à l'utilisation de bits non nuls dans l'expansion de *k*. Aussi, dans leur travail, les points sont représentés et transmis en coordonnées projectives. Piotrowski et al. [115] ont montré que l'énergie nécessaire à la transmission d'un seul bit d'information est équivalente à effectuer plusieurs centaines, voire des milliers de cycles d'horloge de calcul, de sorte que, l'envoi de trois coordonnées est plus coûteux que la

transmission de deux coordonnées en termes de consommation d'énergie. Dans notre application, les points de sortie sont en coordonnées affines qui sont représentés par deux coordonnées. Aussi, on utilise la compression des points et un algorithme de décompression efficace, ce qui conduit à des points avec une seule coordonnée avec un bit supplémentaire pour récupérer la coordonnée y. Notre implémentation est efficace en termes de rapidité et offre une sécurité contre *SPA*. Par conséquent, les principaux objectifs de ce travail ont été atteints.

Implémentation	Algorithme	P.p	Temps	Taille du chiffré	k_i	SPA
Ugus et al. [59]	I&M	2	1.42s	2(3|p|) bits	{1, 0, -1}	Oui
Notre travail	Sec_Comb	16	1.29s	2(3|p|+1) bits	{1, -1}	Non

Tableau 18. Comparaison avec Ugus et al. [59]

6. Résultats pour RSAED

L'algorithme *RSAED* que nous avons présenté dans le chapitre 3 peut bénéficier de ces résultats. Dans ce cadre, nous avons évalué par la suite *RSAED* avec notre nouvelle implémentation de la *MPS* et l'*ADS* sur les deux plateformes MicaZ et TelosB. Les résultats sont présentés dans le Tableau 19.

Opération		Temps d'exécution		Consommation	
		MicaZ	TelosB	MicaZ	TelosB
Chiffrement	SWM	2.844s	6.110s	68.256 mj	32.994 mj
	Sec_comb	1.290s	2.531s	30.96 mj	13.667 mj
Agrégation	Quaternary	0.796s	1.610s	19.104 mj	8.694 mj
	Rep.exp&SWM	0.653s	1.329s	15.672 mj	7.176 mj

Tableau 19. Résultats de RSAED

7. Conclusion

Dans ce chapitre, le chiffrement homomorphique *ECEG* a été implémenté d'une manière efficace et sécurisée. Nous avons utilisé la courbe *SECP160R1* et notre implémentation nécessite seulement 1.29 secondes pour effectuer un chiffrement sécurisé. Dans ce travail, nous avons utilisé d'abord la méthode Comb pour améliorer la multiplication dans la bibliothèque TinyECC, puis nous avons sécurisé la bibliothèque contre les attaques par canaux cachés et en particulier l'attaque par analyse simple de l'énergie, et enfin nous avons prouvé que le coût de la résistivité n'est pas important et peut être réalisé sur des capteurs sans fil. Finalement, un algorithme de décompression efficace a été employé afin de permettre aux nœuds intermédiaires d'exécuter efficacement la fonction homomorphique. Beaucoup d'autres algorithmes tels que l'*ECDH* et l'*ECDSA*, nécessitant une multiplication par un scalaire rapide et sécurisée, pourront donc directement bénéficier de nos résultats. Les résultats expérimentaux, présentés dans ce chapitre, permettent de conclure que l'utilisation

de la cryptographie asymétrique est réalisable dans les réseaux de capteurs sans fil à travers l'*ECC*. Cependant, les algorithmes nécessaires pour la mise en œuvre doivent non seulement être analysés et sélectionnés, mais aussi implémentés en toute sécurité.

Dans le chapitre suivant, nous allons présenter une nouvelle solution pour sécuriser l'agrégation en utilisant un chiffrement hybride basé sur le chiffrement à clé publique avec état.

Chapitre 5: Troisième contribution: Une nouvelle solution pour sécuriser les données agrégées en utilisant la cryptographie à clé publique avec état

1. Introduction

Dans les RCSFs, l'agrégation des données joue un rôle très important dans la préservation de l'énergie des capteurs. Cependant, ces réseaux sont généralement déployés dans des environnements hostiles et sans assistance (par exemple un champ de bataille dans une application militaire), et dans lesquels la confidentialité et l'intégrité des données sont largement souhaitées. Ce chapitre présente un nouveau protocole, combinant les meilleurs caractéristiques des deux types de chiffrements (symétrique et asymétrique), pour sécuriser efficacement les données agrégées. Ce protocole est basé sur le Chiffrement à Clé Publique avec Etat (*CCPE*) qui produit des chiffrés avec des tailles réduites tout en offrant un haut niveau de sécurité. Aussi, notre solution est versatile, car elle permet à la station de base de calculer n'importe quelle fonction d'agrégation sur les données individuelles. Nous présentons dans la suite notre algorithme implémenté sur TelosB et MicaZ. Nous avons mesuré le temps d'exécution de nos opérations cryptographiques. Des simulations ont été aussi effectuées pour montrer comment notre solution peut atteindre un niveau de sécurité élevé (en offrant les deux services de sécurité) avec une charge réduite (en termes de calcul et de communication) dans des scenarios à grande échelle.

2. Motivations

En général, les RCSFs sont déployés dans une zone géographique afin de collecter de l'information utile. Cependant, dans un environnement hostile, sans assistance et sans sécurité, le RCSF est menacé. En raison de leur design, les capteurs sans fil peuvent être aisément capturés. Aussi, les nœuds qui effectuent la fonction d'agrégation sont les plus attractifs aux yeux des attaquants. Par conséquent, afin de faire face à ces menaces de sécurité, la recherche sur la sécurité des données agrégées est primordiale. Nous avons vu dans notre état de l'art que dans les protocoles traditionnels, le chiffrement est effectué saut à saut, une approche qui non seulement introduit une charge de calcul importante mais aussi un délai considérable. En outre, l'agrégateur peut accéder aux données en clair, donc la confidentialité de bout en bout n'est pas fournie. Dans un bon nombre d'applications, cette propriété est hautement souhaitée. Le développement de solutions avec une sécurité plus solide est donc crucial.

Fournir un tel service requiert l'utilisation du chiffrement homomorphique c'est-à-dire le calcul sur les données chiffrées aux niveaux des nœuds intermédiaires. Dans les algorithmes homomorphiques, on peut trouver les deux types symétrique et asymétrique. Ceux basés sur l'asymétrique ont été étudiés et analysés dans [58] où les auteurs ont montré que l'*ECEG* est la solution la plus prometteuse. Dans le chapitre précédent, nous avions proposé une implémentation efficace de l'*ECEG*, où un nouveau record a été atteint dans l'implémentation de la *MPS* [90]. En effet, un capteur a besoin seulement de 1.29s pour

effectuer un chiffrement sécurisé de l'*ECEG*. Cependant, dans une application où la station de base à besoin de collecter d'une manière continue les informations concernant la zone à surveiller, par exemple chaque 20 secondes, une telle solution est impraticable est conduit à l'épuisement de l'énergie du capteur. Les algorithmes basés sur le chiffrement symétrique ont été aussi proposés mais la plupart ont été cryptanalysés [44]. Un autre service de sécurité, à savoir l'intégrité de bout en bout est une problématique intéressante, devenue un challenge à la communauté scientifique. En effet, une vérification saut à saut ne peut assurer que l'agrégateur effectue correctement l'agrégation des données. Un agrégateur compromis peut produire un agrégat malicieux et l'authentifie en utilisant sa clé légitime. Ainsi, l'intégrité de bout en bout est un autre service largement désiré. Vu les spécificités des RCSFs, il est très difficile de fournir efficacement ces deux services en même temps.

Le chiffrement à clé publique avec état proposé par Bellare et al. [116] utilise un chiffrement hybride avec un état afin de réduire la charge de calcul. Ce type de chiffrement permet à l'émetteur de maintenir un état qui sera utilisé lors de différentes sessions et en conséquence préserver l'énergie. Dans [117], les auteurs ont réalisé le *CCPE* sur un *RCSF* et ont montré son applicabilité sur une plateforme de 8 bits. Cependant, les auteurs ont considéré une communication directe avec la station de base (1 saut).

Dans ce chapitre, nous proposons *SASPKC* [118], une solution dans laquelle les deux services de sécurité seront adressés. Notre solution profite des avantages des deux types de chiffrements (symétrique et asymétrique) afin de réduire les charges de calcul et de communication à travers le *CCPE*. Aussi, nous avons implémenté notre protocole sur TelosB mais aussi sur MicaZ en utilisant la version améliorée et sécurisée de TinyECC proposée dans le chapitre précédent.

3. Spécifications générales

Notre solution adopte le *CCPE* pour une sécurité efficace. Dans cette section, nous commençons par une brève description du *CCPE*, ensuite nous spécifions les hypothèses sur le réseau et la sécurité, ainsi que les objectifs visés et l'attaquant considéré.

3.1. Le chiffrement à clé publique avec état

Le coût calculatoire du Chiffrement à Clé Publique (*CCP*) est dominé par les exponentiations modulaires ou la *MPS* dans le cas des *ECC*, et qui influence dramatiquement sur la consommation énergétique du dispositif exécutant ces opérations. Dans [116], les auteurs ont proposé un chiffrement avec état qui peut considérablement améliorer le coût calculatoire des *CCP* traditionnels (Voir Algorithme 13). L'état maintenu par l'émetteur sera utilisé lors de différentes sessions.

Algorithme 13: Chiffrement à clé publique avec état
Paramètres publiques: Groupe du DSA ou groupe des courbes elliptiques avec un ordre premier p et un générateur "g" Clé publique < $Y=g^x$, H >, Clé privée < x > **Chiffrement**: r ←Zq, $C_1=g^r$, $K=H(C_1,Y,Y^r)$, $C_2=E_{sym}(K,M)$ Chifré: (C_1, C_2) **Déchiffrement**: $K=H(C_1, Y, C_1^x)$, $M= D_{sym}(K,C_2)$

L'état (C_1, r) est maintenu par l'émetteur, ce dernier utilise ce même état pour les futurs chiffrements, ce qui réduit significativement le coût de calcul puisque C_1 est calculé une seule fois. La sécurité est basée sur l'hypothèse de *Diffie-Hellman* (étant donné g^a et g^b, il est calculatoirement infaisable de calculer g^{ab}, voir [116] pour plus de détails).

3.2. Modèle du réseau

Nous considérons un réseau avec un grand nombre *NR* de capteurs et une Station de Base (*BS*). Les capteurs sont des dispositifs dotés des ressources limitées (par exemple TelosB [2]). Ils sont déployés dans une zone géographique afin d'effectuer certaines fonctions spécifiques de surveillance. Ensuite, ils sont organisés en plusieurs clusters (Voir Figure 23). Les clusters sont statiques après leurs formations et un Cluster-Head (*CH*) est élu dans chaque groupe suivant des algorithmes tels que [25] et [26]. Nous supposons aussi que la station de base soit un dispositif fiable et puissant. Le *CH* effectue la fonction d'agrégation sur les données en provenance de ses membres et transmet le résultat à la *BS*. Toutes les notations utilisées dans le présent chapitre ainsi que leurs définitions sont résumées dans le Tableau 20.

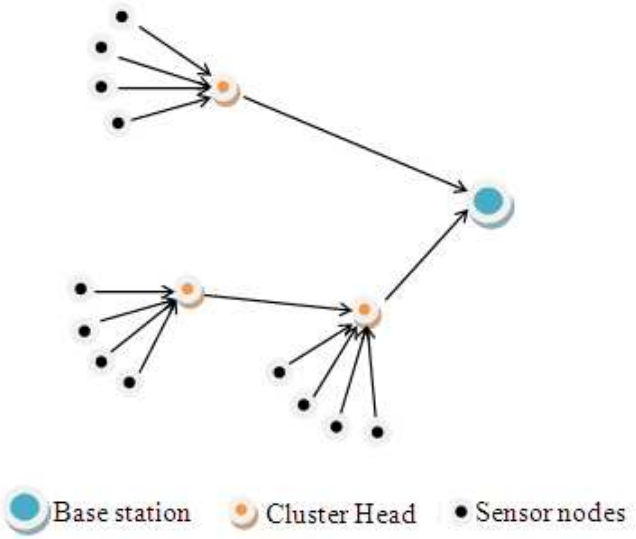


Figure 23. Modèle du réseau

Notation	Définition
NR	Nombre de capteurs
R	Nombre d'agrégateurs dans le réseau
CH_j	CH_j, $j \in \{1,\ldots,R\}$
L	Nombre maximum de nœuds par cluster
S_{ij}	Sensor node i belong CH_j, $i \in \{1,\ldots,L\}$
BS	La station de base
Y	La clé publique de BS
$\|\|0^r$	Concaténation *r* fois avec des bits nuls
K_{ij}^{BS}	La clé partagée entre S_{ij} et BS
N_{ij}	Un Nonce
H	Fonction de hachage
HMAC(M, K)	La signature HMAC du message M via la clé K
Enc (K, M)	Chiffrement symétrique du message M via la clé K

Tableau 20. Notations et leurs définitions

3.3. Modèle de sécurité

Pour la confidentialité, nous utilisons l'algorithme de Castelluccia et al. [71] décrit dans le Chapitre 2. Dans [58] plusieurs *C.P.H* ont été présentés. Cependant, les auteurs indiquent que l'Algorithme 1 (Voir Chapitre 2) reste l'algorithme le plus efficace en termes de communication et calcul. Il permet le calcul sur les chiffrés et fournit une meilleure confidentialité par rapport aux approches saut à saut.

Pour l'intégrité, nous adoptons les agrégations *MAC* (*Aggregate MAC*). Il est clair que le *MAC* ne peut satisfaire la propriété homomorphique. Dans [119], les auteurs ont suggéré une idée où ils ont prouvé que les signatures *MAC* peuvent être agrégées en utilisant l'opération *Xor* et que le résultat permet la vérification de l'intégrité et l'authentification avec la condition d'avoir toutes les données individuelles.

$$MAC_{agg} = MAC_1 \oplus MAC_2 \oplus \ldots \oplus MAC_n$$

L'*ECC* a hérité de beaucoup d'attention au cours des dernières années. En effet, ce type de chiffrement peut atteindre le même niveau de sécurité du *RSA* avec des clés réduites. Une clé de 160 bits *ECC* fournit le même niveau de sécurité qu'une clé 1024 bits *RSA*. Par conséquent, *ECC* est la solution la plus adéquate pour les capteurs sans fil puisque des clés réduites signifient moins de temps, moins de stockage, moins de bande passante et une consommation d'énergie réduite [95]. En plus, il est recommandé par [116] pour le *CCPE*. On suppose que, avant le déploiement, la *BS* génère son couple de clés (x, Y) où $Y = xG$, et maintient la clé privée x. Chaque capteur est préchargé, par une clé K_{ij}^{BS} partagée seulement avec la *BS*, par les paramètres de la courbe elliptique (Y, E, P, G, n, où Y est la clé publique, E la courbe elliptique sur un corps premier p, le point de base G d'ordre n), par une fonction de dérivation de clé sécurisée (*KDF : Key Derivation Fonction*) et enfin par un *MAC* infalsifiable

(par exemple, respectivement, *NIST SP800 HKDF* (*HMAC based KDF*) et *HMAC-SHA-1-160* avec des clés de 160 bits, qui sont utilisés dans notre implémentation logicielle dans la section 5.2.).

3.4. Modèle d'attaque

Nous considérons toutes les attaques décrites dans [91] où l'attaquant peut écouter et surveiller les données transmises ainsi il peut effectuer des attaques passives telles que l'attaque par analyse des chiffrés et les attaques à texte en clair connu. Il peut aussi exécuter des attaque actives dans lesquelles il peut produire un ensemble de données illégales, modifier les paquets et rejouer des paquets valides déjà transmis ce qui conduit respectivement à la falsification de paquets, à la malléabilité et à l'attaque par rejeu. Nous considérons aussi l'attaque *SPA* contre les implémentations *ECC* décrite dans [31].

3.5. Objectifs de conception

- Sécurité :

Dans les RCSFs utilisant l'agrégation des données, les deux services de sécurité de bout en bout, à savoir la confidentialité et l'intégrité sont largement souhaités. Dans notre approche nous visons à fournir ces deux services.

- Efficacité :

Nous visons aussi à proposer une solution efficace en termes de charges de calcul et de communication. En effet, les opérations cryptographiques doivent être efficaces, surtout quand la *BS* à besoin de collecter les données d'une manière continue. Aussi, une solution de sécurité pour l'agrégation doit mettre en évidence l'avantage d'utiliser l'agrégation et s'assurer que le protocole n'affecte pas cet avantage.

- Versatilité :

Cet aspect permet à la *BS* de calculer n'importe quelle fonction sur les données captées. C'est un aspect très important puisqu'il contribue à servir un nombre très large d'applications.

4. Solution SASPKC

SASPKC est composée de deux phases principales à savoir *la phase de transmission* et *la phase d'agrégation*. Dans la première, tous les *NR* capteurs envoient leurs états qui seront utilisés dans la phase d'agrégation, dans laquelle les capteurs chiffrent et signent leurs données captées en utilisant leurs états partagés avec la *BS*. Ensuite, le *CH* combine tous les chiffrés et les signatures en un seul chiffré et une seule signature en utilisant respectivement l'opération homomorphique et l'opération *Xor*. Enfin, la *BS* vérifie les données agrégées, elle commence par déchiffrer et extraire les données en clair, puis vérifie l'intégrité des paquets et authentifie les expéditeurs. Les détails de ces deux procédures ci-dessus sont présentés dans ce qui suit.

La phase de transmission

Dans cette phase, les *NR* capteurs envoient leurs états qui seront utilisés pour générer les sous clés nécessaires pendant la phase d'agrégation, les clés d'authentifications sont obtenues via *HKDF* (Voir Algorithme 14). Chaque capteur envoie son état et une signature *MAC* à l'agrégateur *CHj* correspondant. Afin de pouvoir extraire tous les états par la *BS*, tous les paquets doivent être transmis. Par conséquent, dans cette phase les *CHs* agissent comme des transitaires de données et non pas comme des agrégateurs (Voir Figure 24). L'algorithme exécuté par les capteurs est le suivant:

Algorithme 14: Phase de transmission (Capteurs)
1. Générer un nombre aléatoire $r_{ij} \in [1, n-1]$
2. Calculer $St_{ij}=r_{ij}.G$
3. Calculer $K_{ij}=H(N_{ij}\|St_{ij}\|r_{ij}Y\|K_{ij}^{BS})$ via HKDF
4. Calculer $MAC_{ij}=MAC(St_{ij}, K_{ij})$
5. Envoyer (St_{ij}, MAC_{ij}) à CH_j

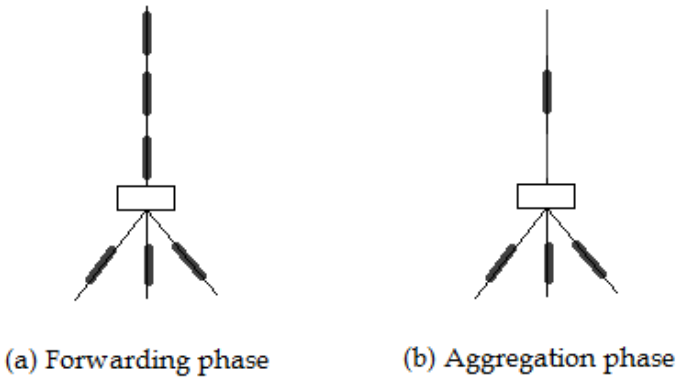


Figure 24. La transmission des données dans SASPKC

Chaque capteur maintient $<r_{ij}, St_{ij}>$ comme état et utilise cet état pour les sessions futures. Cet état est transmis au *CH* correspondant, qui le retransmet à la *BS* ou au *CH* le plus proche. La *BS* vérifie ensuite l'intégrité et authentifie tous les expéditeurs en utilisant sa clé privée *x* (Voir Algorithme 15). Notons que l'exactitude «*correctness*» est garantie puisque $xSt_{ij}=xr_{ij}G$.

Algorithme 15: Phase de transmission (Station de base)
1. Pour chaque $i \in \{1,\ldots,L\}$ et $j \in \{1,\ldots,R\}$ Calculer $K_{ij}= H(N_{ij}\|St_{ij}\|xSt_{ij}\|K_{ij}^{BS})$ via HKDF
2. Vérifier tous les paquets reçus

La phase d'agrégation

Cette phase est composée de trois étapes: *Chiffrement, agrégation* et *vérification*. Ces étapes fonctionnent comme suit:

- Chiffrement

Dans cette étape, les données sont encodées de manière similaire à [75]. La différence est que le texte en clair est chiffré en utilisant le chiffrement symétrique qui est non seulement plus efficace mais aussi produit des chiffrés réduits par rapport à l'asymétrique (Voir Algorithme 16). Aussi, les techniques telles que introduites dans [51] peuvent être utilisées pour réduire

de manière significative le nombre de bits (λ) nécessaires pour représenter les données capturées. Contrairement à la phase de transmission, la fonction *HKDF* produit en sortie deux clés dans la phase d'agrégation à savoir, K_{ij1} et K_{ij2}. Chaque capteur chiffre le texte en clair encodé et calcule le *MAC* correspondant respectivement à l'aide de K_{ij1} et K_{ij2}. Chaque membre du cluster exécute l'algorithme suivant:

Algorithme 16: Phase d'agrégation (Capteurs)

1. Encoder m_{ij} à $e_{ij}=m_{ij}||0^r$, avec $r=\lambda*(i-1)$.
2. Calculer l'actuel $K_{ij}= H(N_{ij}||St_{ij}||r_{ij}Y||Y)$ via *HKDF* avec $K_{ij} = K_{ij1}|| K_{ij2}$
3. Calculer:
   $C_{ij}= K_{ij1}+ e_{ij} \bmod M$
   $MAC_{ij}= MAC (C_{ij}, K_{ij2})$
4. Envoyer (C_{ij}, MAC_{ij}) à CH_j

Le chiffrement est effectué suivant [71], *M* doit être supérieur à $e_{agg} = \sum_{i=1...L}^{j} e_{ij}$ sinon l'exactitude ne sera pas garantie. Si cette propriété est vérifiée alors le déchiffrement produira un message e_{agg} qui est inferieur à *M*. Le nonce N_{ij} assure la dynamicité des clés pour la sécurité des chiffrements [71]. Une signature *MAC* est par la suite calculée sur le chiffré (*encrypt-then-MAC*). Enfin, le chiffré et le *MAC* correspondant sont envoyés au *CH*.

- Agrégation

Dans cette étape, le *CH* combine les n chiffrés, y compris le sien en un seul chiffré, et fait la même chose pour les signatures *MAC* (Voir Algorithme 17).

Algorithme 17: Phase d'agrégation (Agrégateurs)

1. Pour n chiffrés (C_{1j}…C_{nj})
   Calculer $C_{agg}= \sum_{i=1...L}^{j} C_{ij}$ suivant [71]
2. Pour n signatures (MAC_{ij}…MAC_{nj})
   Calculer $MAC_{agg}= \oplus MAC_{ij}$
3. Envoyer (C_{agg}, MAC_{agg}) à la BS ou le CH le plus proche

Le *CH* effectue l'opération d'addition modulo *M* sur les chiffrés et les signatures sont combinées en utilisant la fonction *Xor*. Pour un *CH* qui reçoit un agrégat à partir d'un autre *CH*, il n'a qu'à transmettre le message à la *BS*. Aussi, le *CH* peut notifier à la *BS*, les capteurs n'ayant pas envoyés leurs données, soit malicieusement ou à cause d'un mal fonctionnement.

- Vérification

Une fois les agrégats de tous les clusters reçus par la *BS*, elle exécute les processus de déchiffrement et de vérification selon l'Algorithme 18.

Algorithme 18: Phase d'agrégation (Station de base)

1. Calculer tous les actuels $K_{ij}= H(N_{ij}||S_{ij}||xS_{ij}|| Y)$
2. Calculer $e_{agg}= C_{agg}- \sum_{i=1...L}^{j} K_{ij1} \bmod M$
3. Décoder (e_{agg},L, λ): $m_i= e_{agg}\ [(i-1)* \lambda, \lambda*i-1]$, avec i=1,…,L
4. Vérifier MAC_{agg}

La station de base commence par calculer toutes les clés courantes correspondantes à tous les nœuds du réseau. Le calcul se fait en se basant sur les états correspondants sauvegardés lors

de la phase de transmission. Une fois cette phase achevée, la *BS* aura en sa possession toutes les clés de chiffrement et d'authentification. Ensuite, le déchiffrement des données est effectué en soustrayant de l'agrégat la somme des clés de chiffrement courantes. Le résultat, correspondant à l'agrégat des messages en clair, est utilisé par la suite comme entrée dans la fonction de décodage afin d'obtenir les données individuelles. Enfin, après le calcul de toutes les paires (Chiffré, Signature), la *BS* vérifie l'intégrité de bout en bout. Si la vérification échoue l'agrégat sera rejeté. La *BS* pourra identifier l'anomalie en notifiant au *CH* correspondant d'envoyer toutes les paires, ces dernières lui permettront de vérifier la validité de chaque paire et ainsi trouver l'anomalie. Dans l'autre cas, l'agrégat est accepté. Les deux services de sécurité sont donc fournis. Un autre avantage de notre solution est qu'il n'est pas nécessaire de transmettre la liste des capteurs participants puisqu'ils sont tous concernés. Dans *SASPKC*, un capteur produit un chiffré et une signature même s'il n'a pas de données à transmettre. Par conséquent, puisque la *BS* aura en sa possession toutes les données individuelles, elle peut effectuer n'importe quelle fonction d'agrégation sur eux, une propriété qui constitue l'avantage principal des solutions saut à saut. *SASPKC* est donc versatile et n'impose aucune limite sur la nature de cette fonction.

La phase d'agrégation est effectuée plusieurs fois jusqu'à ce que l'état St_{ij} expire. La date d'expiration est désignée par les développeurs de l'application. L'expiration d'une clé est une mesure de sécurité très importante, permettant le rafraichissement des clés, ce qui implique une nouvelle phase de transmission et la sécurité est donc améliorée. Dans *SASPKC*, la durée de vie du capteur peut être vue comme une séquence d'époques où chaque époque est constituée des deux phases de transmission et d'agrégation (Voir Figure 25).

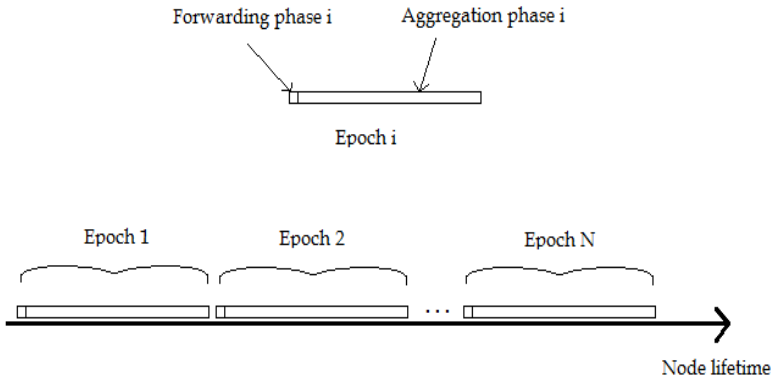


Figure 25. Durée de vie d'un capteur dans SASPKC

Exemple:

Dans ce qui suit, nous donnons un exemple (Voir Figure 26) pour montrer comment *SASPKC* fournit les deux services de sécurité et la versatilité.

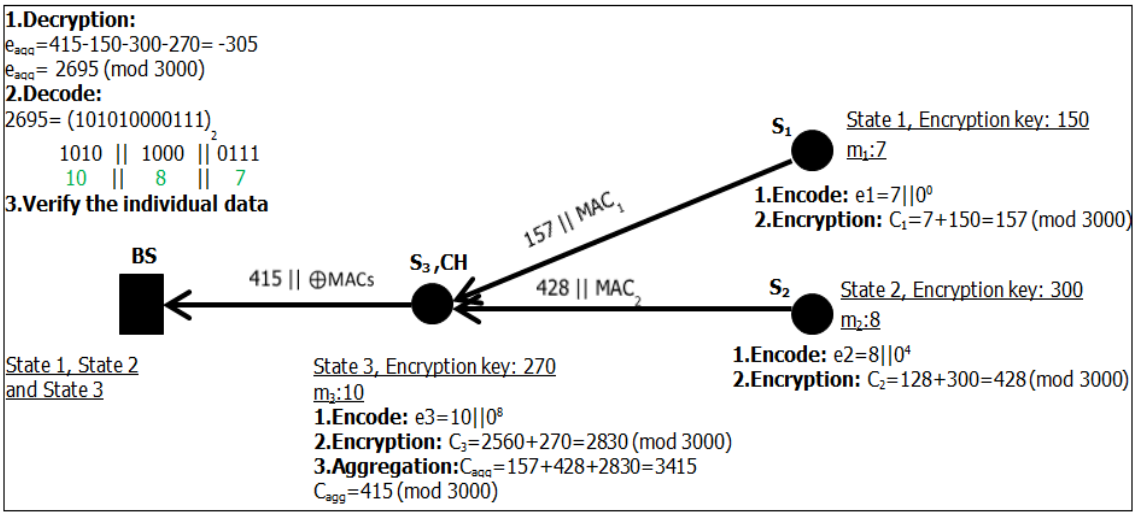


Figure 26. Exemple du fonctionnement de SASPKC

Comme le montre la Figure 26, nous supposons un réseau qui consiste en une station de base et 3 nœuds capteurs S_1, S_2 et S_3. Le capteur S_3 est sélectionné comme un agrégateur. Supposons que la phase de transmission est achevée donc chaque capteur partage un état avec la station de base. Nous supposons aussi que les données captées par ces nœuds sont respectivement 7, 8, 10, ainsi que le nombre λ=4 nécessaire pour représenter les données. Chaque capteur encode le texte en clair, puis effectue un cryptage via une addition avec la clé courante modulo un grand nombre M (Soit M=3000). Enfin, il envoie le chiffré, ainsi que le *MAC* correspondant au *CH*. Nous notons que, dans cet exemple, de petits nombres sont utilisés dans le but de simplifier la compréhension, toutefois de très grands nombres peuvent être utilisés dans la pratique. Après cela, l'agrégateur effectue l'opération homomorphique sur les chiffrés et transmet le résultat agrégé avec le *MAC* à la *BS*. Enfin, la *BS* extrait les clés, décrypte et décode le résultat agrégé pour obtenir les données individuelles (7, 8 et 10) et enfin appelle le processus de vérification.

5. Analyse

5.1. Sécurité

Nous commençons par la phase de transmission où tous les états sont transmis à la *BS*. La sécurité dans cette phase est fournie en supposant que le problème *Gap Diffie-Hellman*[11] (*GDH*) est difficile et le *MAC* est infalsifiable. Dans *SASPKC*, le *MAC* est calculé en utilisant la clé K_{ij} générée via *HKDF* à partir de $r_{ij}Y$ (qui est inconnu à l'attaquant en supposant que le *GDH* est difficile) et K_{ij}^{BS} , la clé préchargée avant le déploiement. Ensuite, la *BS* peut vérifier l'intégrité de tous les messages et authentifie les émetteurs en utilisant sa clé privée x et K_{ij}^{BS}. Contrairement à St_{ij} qui expire après une date prédéfinie, K_{ij}^{BS} est une clé qui n'expire jamais. En effet, le capteur utilise la même clé et un nonce pour générer le nouvel état.

[11] Le *GDH* est un problème calculatoire où l'attaquant, étant donné (G,aG,bG) pour des inconnus (a,b), essaie de calculer la clé abG [120].

Dans la phase d'agrégation, la sécurité est assurée car *GDH* est considéré comme un problème difficile, les clés sont dynamiques et le *MAC* est infalsifiable. Notre protocole vise à fournir les données capturées avec la confidentialité et l'intégrité de bout en bout.

La confidentialité de bout en bout est assurée en utilisant le *C.P.H* symétrique de Castelluccia et al. [71]. Le *C.P.H* permet une addition modulaire sur les données chiffrées au niveau des nœuds intermédiaires, en conséquence, le nœud effectuant cette opération n'a pas accès au message en clair. La sécurité de ce service repose sur la sécurité du *C.P.H* utilisé. Les auteurs de [71] ont affirmé que pour assurer la sécurité, les clés utilisées pour le cryptage doivent être dynamiques c'est-à-dire les clés changent d'un message à un autre. Dans le but de fournir cette sécurité, nous utilisons un nonce. En effet, le dispositif utilise en plus de l'état, un nonce pour le cryptage. La *BS* utilise le même état et le même nonce pour le déchiffrement. Ce nonce est une valeur publique, qui ne doit pas être cachée à l'adversaire, mais la seule exigence est que la paire (état, nonce) soit utilisée pour crypter un seul message. En d'autres termes, le couple (état, nonce) doit changer d'un message à un autre. Il y a deux façons de les changer: (i) un nouvel état pour chaque message ou (ii) continuer à utiliser le même état, mais choisir un nouveau nonce pour chaque message. Dans *SASPKC*, l'état est calculé une seule fois et un nouveau nonce est généré pour chaque message. Le nonce utilisé est un numéro de séquence implicite qui est initialisé et incrémenté d'un paquet à un autre. Dans notre système, le hachage (en utilisant *HKDF*) de l'état actuel avec un nonce est utilisé pour générer des clés différentes pour différents messages dans la phase d'agrégation. Cette propriété offre la sécurité contre l'adversaire passif décrit dans [91].

Dans [87], les auteurs ont affirmé que pour atteindre le niveau de sécurité *IND-CPA*, il est nécessaire d'utiliser directement la sortie de la *PRF*, cela signifie que l'avantage principal de l'algorithme sera perdu. En effet, le principe de l'Algorithme 1 est d'utiliser une clé similaire au message en termes de taille (appartenant à [1, M-1]), ainsi pour une sécurité *IND-CPA*, le choix du modulo *M* est mis en question puisqu'il doit dépendre aussi de la taille de la clé pour des fins d'exactitude. Dans le même article les auteurs ont proposé (*Hashed CMT*), un algorithme qui emploie une deuxième fonction de hachage. Le rôle de cette fonction est de réduire la taille de la sortie *PRF* en une taille similaire à celle du maximum possible d'une valeur agrégée. Dans [121], les mêmes auteurs ont indiqué que même le choix de la taille de la sortie de la deuxième fonction doit être raisonnable afin d'assurer une probabilité de succès raisonnablement faible pour une estimation aléatoire. Dans *SASPKC*, la sortie de la fonction est directement utilisée et ceci grâce à la fonction d'encodage employée. Autrement dit le choix du modulo *M* dépend toujours de e_{agg} dans notre cas même si la taille originale de la sortie est utilisée. Ainsi, le niveau *IND-CPA* est atteint sans la nécessité d'utiliser une quelconque autre fonction.

L'intégrité de bout en bout est assurée en utilisant l'agrégation des *MAC*. Dans [91], les auteurs ont indiqué que l'attaquant actif peut effectuer plusieurs attaques surtout dans le cas d'un chiffrement homomorphique. En effet, il est connu que ce type de chiffrement est très vulnérable aux attaques actives en raison de sa propriété homomorphique. La non-malléabilité constitue un défi pour la communauté scientifique. Dans ce qui suit, nous analysons la résistivité de *SASPKC* aux attaques actives décrites dans [91].

- L'attaque par rejeu vise à envoyer des paquets valides déjà transmis afin de produire une action non autorisée. *SASPKC* utilise différentes clés de chiffrement et d'authentification d'un paquet à un autre. Par exemple, après un nombre de tours *t*, l'attaquant rejoue le paquet du *i-ème* tour de la phase d'agrégation. Il est évident que le processus de vérification échoue parce que la station de base utilise la clé du *(t+i)-ème* tour (la clé de *l'i-ème* tour étant expirée) pour vérifier les données.

- La malléabilité permet à l'attaquant d'altérer le contenu valide d'un chiffré afin de tromper la station de base. En effet, l'attaquant peut modifier les données chiffrées sans nécessairement connaitre le contenu et ceci peut être effectué par un simple ajout d'un nombre à un chiffré dans notre cas. Par exemple, *((m+10)+k) mod M = ((m+k)+10) mod M*. *SASPKC* fournit l'intégrité de bout en bout permettant à la station de base à la fois de vérifier les données individuelles et authentifier les émetteurs. Par conséquent, si un chiffré est altéré, la vérification échouera et l'agrégat correspondant sera rejeté. La station de base enclenche la procédure d'identification du nœud malicieux.

- Un attaquant peut aussi injecter des données erronées en remplaçant les données actuelles par les données forgées. *SASPKC* utilise des clés différentes pour chaque message avec une signature *MAC*. Par conséquent, l'attaquant ne peut produire un chiffré et une signature *MAC* valides sans connaitre les clés courantes.

- Un attaquant, qui capture un nœud, peut l'utiliser pour (i) agir comme un nœud légitime (ii) forger un message qui est peu différent du légitime et (iii) forger un message qui est très diffèrent du légitime. Pour les deux premiers cas, il n'y a pas actuellement de solutions permettant la détection de telles actions. Cependant, pour le dernier cas la station de base dans *SASPKC* est en mesure de détecter le nœud malicieux grâce au processus d'identification des nœuds compromis. Si le nœud capturé est un *CH*, ce dernier ne peut ni accéder aux données en clair de ses membres, ni tromper la station de base par un message malicieux parce que les clés correspondantes sont inconnues pour l'attaquant. En d'autres termes, un *CH* compromis peut uniquement modifier son propre message pour essayer de tromper la *BS*.

- l'attaquant peut aussi effectuer une capture furtive à travers l'analyse de l'énergie consommée (*SPA*) [122]. En effet, comme le montre le travail pratique [31], la clé peut être récupérée à partir de l'observation d'un seul tracé énergétique, en conséquence, les communications passées et futures peuvent être compromises. Dans *SASPKC*, l'opération *ECC* vulnérable à ce type d'attaque à savoir la *MPS* est soigneusement implémentée (voir la section suivante).

5.2. Evaluation des performances

Notre solution vise à réduire les charges de calcul et de communication. Dans cette section, nous analysons les performances de notre solution *SASPKC*.

5.2.1. Charge de calcul

L'implémentation est effectuée sur des capteurs TelosB [2]. Dans la phase de transmission, chaque capteur génère un état, ce dernier implique deux *MPS* sur la courbe elliptique et un *MAC*. Dans la phase d'agrégation, chaque capteur produit un chiffré et un *MAC*. Pour l'implémentation, nous avons utilisé TinyOS et la librairie TinyECC. La courbe standardisée *SECP160R1* est utilisée. Nous avons développé une application qui assure le calcul de l'état, la génération des clés courantes, le chiffrement symétrique et le calcul du *MAC*. Pour ce dernier, le *HMAC* [47] fourni par la librairie est utilisé. La fonction *HKDF* utilisée est *NIST SP800 HKDF* (*HMAC based KDF*) avec l'option SHA-1 qui produit en sortie une clé de 20 octets dans la phase de transmission et deux clés de 20 octets dans la phase d'agrégation (pour le chiffrement et le *MAC*). Avant d'évaluer la charge de calcul de nos opérations cryptographiques, nous nous focalisons dans ce qui suit sur la MPS, son efficacité et sa sécurité contre les attaques par canaux cachés.

La *MPS* est l'opération majeure de l'*ECC*, elle représente 80% du temps de calcul d'une clé [95]. La librairie TinyECC fournit des optimisations qui peuvent être activées ou désactivées selon les besoins de l'application, y compris la méthode *SWM*. Cette méthode utilise des précalculs à l'avance afin d'améliorer le temps d'exécution de la *MPS*, en particulier lorsque le point de base est fixe et connu à l'avance. Dans *SASPKC*, la *MPS* est effectuée avec les deux points *G* et *Y*, à savoir respectivement le point de base et la clé publique de la BS, et qui sont fixes et connus à l'avance. L'exécution de la *MPS* implique l'exécution d'un *ECADD* et un *ECDBL* si le bit est non nul et seulement un *ECDBL* dans l'autre cas. Comme mentionné précédemment, avec une implémentation non sécurisée, l'attaque *SPA* peut s'effectuer avec succès et ainsi toutes les communications passées et futures de l'époque courante seront compromises. Dans notre implémentation, nous avons utilisé la version améliorée et sécurisée de TinyECC proposée dans le chapitre précédent pour l'*ECEG* [90]. Une version dans laquelle la *MPS* employée est non seulement efficace mais aussi sécurisée contre *SPA* [112]. Une nouvelle représentation pour le scalaire avec des *1* et *-1*, a besoin seulement de 2^{w-1} points précalculés par rapport aux 2^w-1 points de la méthode originale [109]. Le Tableau 21 présente le temps d'exécution de nos fonctions cryptographiques implémentées sur un capteur TelosB. Les tableaux précalculés des deux points *Y* et *G* sont calculés offline dans la fonction *StPKE.init()*. Le temps d'exécution est obtenu en moyennant plusieurs exécutions avec des scalaires aléatoires de 160 bits. Puisque les points *Y* et *G* sont fixes et connus à l'avance, la charge introduite par cette fonction d'initialisation peut être négligée en effectuant les précalculs offline et distribuer les tableaux correspondants à chaque capteur avant le déploiement.

Opération		Temps	SPA
Stpke.state()	SWM (w=4)	5.82s	Oui
	Comb (w=4)	2.71s	Oui
	Sec_Comb (w=4)	2.85s	Non
	Sec_Comb (w=5)	2.29s	Non
Stpke.encrypt()		0.081s	
Hom.add()		0.002s	

Tableau 21. Temps d'exécution des opérations

Nos résultats présentés dans le Tableau 21 montrent que la méthode Comb peut significativement améliorer le temps d'exécution de la *MPS*, nécessaire dans la phase de transmission. On remarque aussi que la version sécurisée proposée dans [112] est non seulement sécurisée mais aussi 61% et 15% respectivement plus rapide que *SWM* (utilisée dans TinyECC) et Comb, vulnérables à *SPA*. Dans *SASPKC*, la technique de compression des points, adoptée pour transmettre les états, permet la représentation des points en utilisant le nombre minimum possible de bits. L'algorithme de décompression nécessite un calcul de la racine carrée [90], mais cette opération est effectuée seulement par la station de base, supposée avoir des ressources illimitées en termes de capacité de calcul, ressources énergétiques, de mémoire, etc. Cette opération peut être efficacement exécutée en utilisant l'algorithme [115] décrit précédemment.

Pour les phases d'agrégation et de chiffrement, le capteur fait 0.081s pour chiffrer et générer un *MAC*. Si nous comparons ce coût avec celui des solutions existantes, nous trouverons que notre algorithme est le plus efficace. En effet, les travaux [73] et [75] utilisent l'*ECEG* pour le chiffrement qui prend approximativement le même temps d'exécution de notre phase de transmission. L'avantage majeur de notre solution, basée sur le *CCPE*, est que deux calculs de la *MPS* sont évités pour chaque chiffrement. Au niveau intermédiaire, l'agrégateur exécute l'opération homomorphique sur les chiffrés qui consiste en une adition modulaire dans notre cas. Autrement dit, il a besoin seulement d'une petite charge de calcul pour effectuer l'agrégation. Dans [73], [75] et [78] l'agrégation consiste en un calcul sur la courbe elliptique qui non seulement cause une charge importante mais aussi augmente le délai de bout en bout où la *BS* doit attendre avant de recevoir l'agrégat. Notre système prend avantages des deux types de chiffrement afin de réduire le calcul. Il utilise l'*ECEG* pour générer un état calculé une seule fois (partagé seulement avec la *BS*) permettant le cryptage pendant toutes les sessions de la phase d'agrégation. Cet état est actualisé après une date prédéfinie. Notre système utilise aussi les meilleures fonctionnalités du chiffrement symétrique à savoir le calcul rapide et les chiffrés réduits.

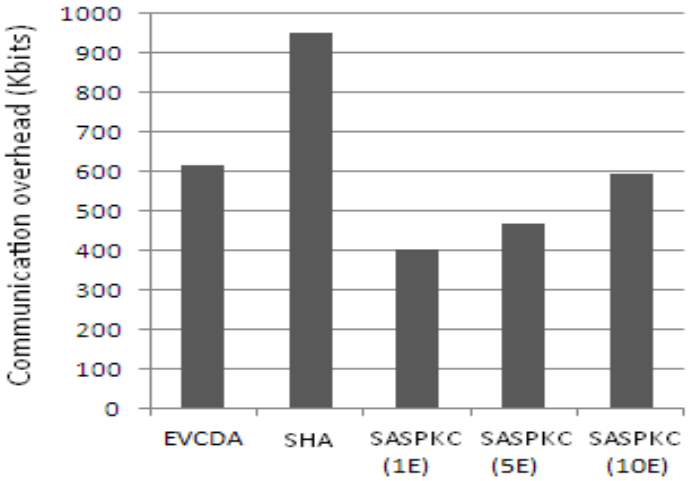
5.2.2. Charge de communication

Dans cette section, nous présentons de nouveaux résultats, relatifs à la charge de communication et à la consommation énergétique, qui compléteront les études expérimentale et analytique présentées dans la section précédente. Ces résultats confirment pleinement la pertinence de notre protocole pour assurer un haut niveau de sécurité avec une charge minimale. Nous avons sélectionné TOSSIM, un simulateur largement utilisé pour les réseaux de capteurs. Nous avons utilisé le simulateur TOSSIM-CC2420, fourni dans la distribution de TinyOS. Il modélise la radio CC2420. Nous avons simulé les trois solutions, Albath et al. [73] (*SHA*) et Sun et al. [75] (EVCDA), et *SASPKC* afin d'établir une nette comparaison et obtenir des résultats précis. La raison du choix de ces deux autres travaux pour la comparaison, repose sur un niveau de sécurité fourni (confidentialité et l'intégrité de bout en bout) comparable avec le nôtre. Trois scénarios de simulation, relatifs à 50, 100 et 150 nœuds capteurs déployés de manière aléatoire dans une zone carrée avec une seule station de base placée au centre, sont analysés.

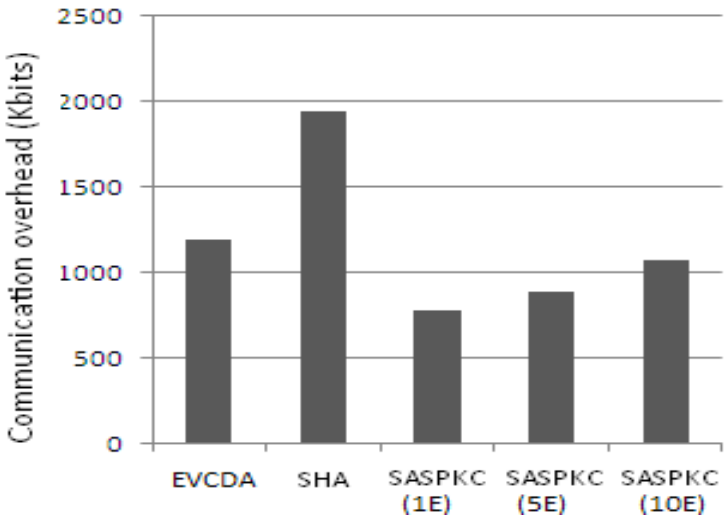
Dans *SASPKC*, les états sont transmis dans leur forme compressée qui nécessite seulement $1+[\log_2 p]$ *bits*, ce qui signifie 21 octets dans notre cas (en considérant la courbe *SECP160R1*). Les

sorties *HMAC* ont une taille de 160 bits (20 octets), pouvant être tronquée à 10 octets suivant [47]. Dans la phase d'agrégation, nous utilisons un texte en clair de 20 octets. Par conséquent, notre proposition utilise un paquet réduit afin de fournir les deux services de sécurité de bout en bout.

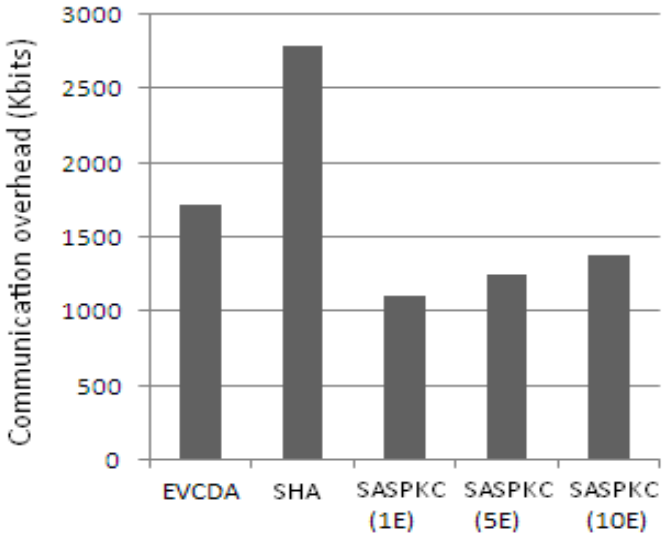
La sécurité de bout en bout proposée dans [73], [75] et [78] encourt des charges considérables de communication par rapport à notre proposition *SASPKC*. Cette charge peut être calculée selon la manière utilisée pour envoyer les paquets. Il existe deux façons: (i) envoyer des messages plus longs (puisque 802.15.4 prend en charge jusqu'à 127 octets), ce qui conduit à augmenter le taux d'erreurs binaires et à diminuer la fiabilité du réseau. Cette approche est utilisée dans [73], et (ii) diviser le paquet en blocs et les envoyer séparément, qui non seulement introduit des délais importants, mais également implique des headers supplémentaires nécessaires pour la transmission des blocs de paquets. Cette approche est utilisée dans [78]. Néanmoins, quel que soit le moyen utilisé dans ces systèmes, notre solution offre davantage de réduction de la consommation d'énergie en raison du nombre et de la taille des messages nécessaires pour la validation des données. Pour donner un sens à cela, nous effectuons plusieurs simulations des trois solutions, durant 500s chacune. Le résultat de chaque solution correspond à la moyenne des résultats de plusieurs simulations. Nous utilisons un simple algorithme de *clustering* à base de *TDMA* pour transmettre les données à la station de base. Le nombre de CHs est de 4, 8 et 12, sélectionnés respectivement pour les trois topologies de 50, 100 et 150 capteurs. Ainsi, le nombre de nœuds par cluster L ∈ [8, 13]. Dans notre application, le nœud non-*CH* envoie (toutes les 20 secondes) un paquet au *CH* correspondant, le *CH* agrège et envoie le résultat à la *BS*. Pour notre application, les nœuds exécutent d'abords la phase de transmission ensuite ils envoient en toute sécurité leur rapport des informations (à l'aide de l'état) sur la région où ils sont déployés à la station de base. La surcharge de communication est mesurée comme étant le nombre total de bits transmis sur le réseau. Les résultats de cette étude sont présentés dans la Figure 27.



(a)



(b)



(c)

Figure 27. Charge de communication pour différentes topologies (a) 50 nœuds, (b) 100 nœuds et (c) 150 nœuds

Les résultats montrent que notre proposition introduit une moindre charge de communication par rapport aux solutions [73,75]. Ceci est dû au *CCPE* dans lequel les données sont cryptées avec un chiffrement symétrique, ce dernier produit des chiffrés réduit et par conséquent des paquets réduits. Les résultats montrent aussi que, même si la sécurité est améliorée dans notre système (en impliquant plusieurs époques pour la même durée de temps), la charge reste acceptable. En outre, l'avantage d'utiliser le *CCPE* peut être observé dans le cas où la densité des nœuds augmente dans le réseau, voir la Figure 27 (b) et (c).

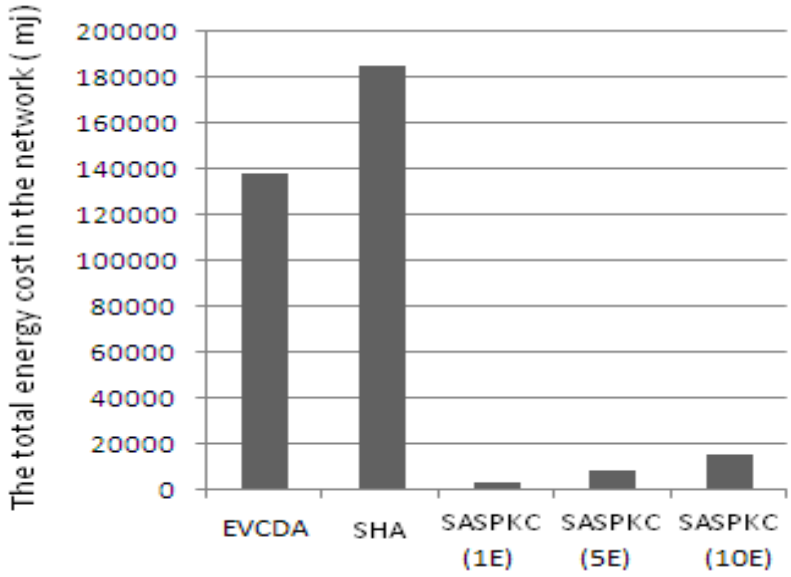
La consommation d'énergie est la problématique centrale dans les réseaux de capteurs sans fil. Le calcul et la communication sont deux aspects qui ont un impact direct sur la consommation d'énergie et par conséquent la durée de vie du nœud. TOSSIM-CC2420 Intègre PowerTOSSIM [123], une extension qui modélise l'énergie. PowerTOSSIM comprend un modèle de la consommation d'énergie des capteurs TelosB. Cependant, TOSSIM ne

modélise pas le temps d'exécution du CPU, il ne peut pas fournir des renseignements exacts sur le calcul de la consommation d'énergie du CPU. Pour ce faire, nous avons ajouté pour tous les paquets transmis/reçus la charge correspondante de calcul. En se basant sur les modèles utilisés dans [102] et [97], la consommation d'énergie *E* peut être calculée par la formule $E = U \times I \times t$, où U désigne la tension, *i* représente le courant et t traduit le temps d'exécution. Pour 2 piles AA, la tension est de 3.0V. Comme indiqué dans [97], le montant de la consommation du courant est 1,8 mA pour TelosB MCUOn/RadioOff. Par conséquent, la consommation d'énergie de nos fonctions cryptographiques peuvent être calculées et sont présentées dans le Tableau 22.

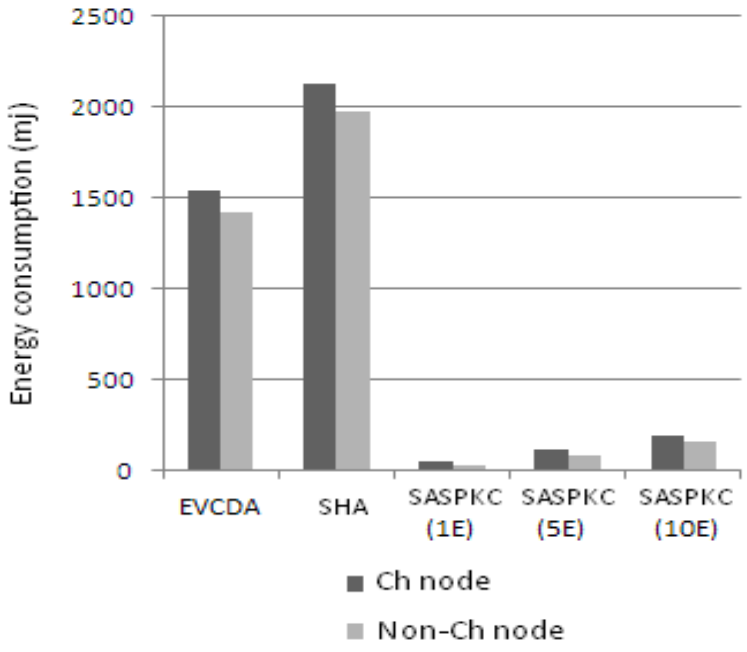
Opération	Consommation
Stpke.state()	12.366 mj
Stpke.encrypt()	0.437 mj
Hom.add()	0.009 mj

Tableau 22. Energie estimée pour nos opérations

Ces mesures sont ajoutées à notre modèle d'énergie et nous effectuons nos simulations pour les trois solutions étudiées. Notons que pour la solution *SHA* [73], les résultats présentés dans [74] ont été utilisés et nous considérons pour *EVCDA* un coût de chiffrement de 3 *MPS*, 1 *ECADD* et un calcul d'un haché de la fonction *SHA*-1, et un coût d'agrégation de 2(n-1) *ECADD* comme indiqué dans [77]. Chaque simulation dure 500s. Les estimations des énergies consommées par tout le réseau et par les nœuds sont présentées respectivement dans les Figures 28(a) et 28(b). Il est apparent de la Figure 28(a) que notre solution offre une forte réduction de la consommation d'énergie par rapport aux autres travaux. Ce gain peut être expliqué par le fait que beaucoup moins de charge de calcul est engagée dans notre système en raison de l'utilisation des primitives symétriques et la mise en œuvre efficace des opérations asymétriques (calcul de l'état). Par conséquent, pour le même niveau de sécurité fourni, la durée de vie du réseau est largement améliorée. Dans la Figure 28(b), nous montrons que la quantité d'énergie moyenne consommée par les nœuds *CH* et non-*CH* est considérablement réduite dans notre système. Cela est dû à, d'une part, les nœuds effectuent en permanence (à chaque tour d'agrégation) des calculs intensifs dans *EVCDA* et *SHA*, tandis que dans notre solution, les opérations coûteuses sont uniquement effectuées dans les phases de transmission. D'autre part, en plus de participer dans les processus d'agrégation, les nœuds *CH* effectuent la fonction d'agrégation (opération homomorphique). Dans *SHA* et *EVCDA*, cette opération nécessite l'opération *ECADD* sur la courbe elliptique tandis que cela nécessite seulement une petite quantité de calcul dans notre proposition (addition modulaire).



(a)



(b)

Figure 28. Energie totale estimée consommé par:
(a) tous le réseau et (b) les nœuds (CH et Non-CH)

Des travaux récents [88] et [115] ont montré que le coût énergétique des communications peut être négligé par rapport aux calculs asymétriques. Les auteurs ont toutefois indiqué que si l'énergie nécessaire pour les calculs cryptographiques est réduite, le coût des communications devient important. La *CCPE* améliore le coût de calcul du *CCP* traditionnel en évitant deux exponentiations (deux *MPS* dans notre cas). En conséquence, il nous donne l'opportunité et la possibilité de considérer le coût des communications et ainsi, de mettre en

évidence l'avantage d'utiliser l'agrégation des données dans les réseaux de capteurs. Dans le Tableau 23, les coûts de l'énergie totale estimée de *SHA*, *SASPKC* et *EVCDA* au niveau des nœuds sont présentés. Les résultats montrent en effet que, contrairement à *SASPKC*, les calculs composent presque le coût total de *SHA* et *EVCDA* sur TelosB.

Protocole		CH (coût en mj)	Non-CH (coût en mj)
SASPKC(1E)	Comm	33.446 (60%)	15.029 (48%)
	Comp	21.999 (40%)	16.649 (52%)
	Total	55.465	31.678
SASPKC(5E)	Comm	35.711 (32%)	15.191 (20%)
	Comp	77.461 (68%)	63.038(80%)
	Total	113.172	78.229
SPSPKC(10E)	Comm	38.103 (21%)	15.284 (11%)
	Comp	141.145 (79%)	128.369 (89%)
	Total	179.248	143.653
EVCDA [75]	Comm	39.007 (3%)	16.748 (1%)
	Comp	1497.143 (97%)	1408.692 (99%)
	Total	1536.15	1425.44
SHA [73]	Comm	90.744 (4%)	27.488 (1%)
	Comp	2038.263 (96%)	1950.672 (99%)
	Total	2129.007	1978.16

Tableau 23. Energie totale consommée par les nœuds CH et Non-CH dans un réseau de 100 nœuds

5.2.3. Scalabilité

Notre solution proposée est scalable et permet le passage à l'échelle en ajoutant autant de clusters que nous voulons. La seule condition est que, le nombre de nœuds par cluster L ne doit pas dépasser le nombre maximal de nœuds qui peut soutenir la fonction d'encodage. Dans le Tableau 24, nous montrons comment ce nombre peut changer avec le niveau de sécurité et λ (le nombre de bits nécessaires pour représenter les données). Comme mentionné précédemment, la technique de référence utilisée dans [51] peut considérablement réduire le nombre λ. Pour les réseaux multi-saut, le *CH* qui reçoit un texte chiffré agrégé provenant d'un autre *CH* dans le réseau n'a qu'à transmettre le paquet correspondant à la *BS* ou au *CH* le plus proche.

Niveau de sécurité	λ	L
80 bits	1	80
	2	40
	4	20
	8	10
160 bits	1	160
	2	80
	4	40
	8	20

Tableau 24. Nombre L Vs λ

5.2.4. Portabilité

Pendant un stage effectué à l'*ISAT* (l'Institue Supérieur de l'Automobile et des Transports de l'Université de Bourgogne, France), nous avons eu la chance d'implémenter notre solution *SASPKC* sur une autre plateforme populaire à savoir le capteur MicaZ disponible au laboratoire *DRIVE*. Le MicaZ est un dispositif avec des ressources limitées (Voir Chapitre 1). Les résultats obtenus sont très encourageants. En effet, les fonctions *StPKE.State ()* et *StPKE.Encyrt ()* prennent respectivement seulement 0.057s et 0.0012s. Suivant le modèle utilisé dans [97], l'énergie consommée est calculée et présentée dans le Tableau 25. Puisque TelosB est plus efficace que MicaZ en termes d'énergie, sa consommation d'énergie est plus faible. En fait, les résultats montrent que, même si le temps d'exécution sur MicaZ est 34% plus rapide que sur TelosB, ce dernier nécessite seulement 27% de l'énergie consommée par MicaZ pour la même durée de temps.

Operation		MicaZ	TelosB
Stpke.state()	Temps	1.48s	2.29s
	Energie	44.24mj	12.366mj
Stpke.encrypt()	Temps	0.057s	0.081s
	Energie	1.7mj	0.437mj
Hom.add()	Temps	0.0012s	0.002s
	Energie	0.036mj	0.009mj

Tableau 25. Comparaison des implémentations

5.2.5. Comparaison avec d'autres solutions

Une comparaison, en termes de sécurité, d'efficacité (calcul, communication) et de versatilité, de notre solution avec quelques travaux de la littérature étudiés dans le chapitre 2, est présentée dans le Tableau 26. Les principaux avantages de notre protocole sont: (i) toutes les données captées ne peuvent être vérifiées qu'au niveau de la *BS* et (ii) les primitives symétriques employées conduisent à une efficacité en termes de calcul et de communication.

Solution	Charge		Sca	Ver
	Com	Cal		
Sia et al. [45]	+	++		•
Mahimkar et al. [49]	++	+++		
Castelluccia et al. [71]	+	+		
Albath et al. [73]	+++	+++		
Sun et al. [75]	+++	+++		•
RSAED	++	++		
SASPKC	+	+	•	•

Solution	Attaques passives		Attaques actives						Services				
	A.C	A.T.C	A.C.A	A.D.S	A.E.S	A.S	A.R	A.F	C	I	A	D	F
Sia et al. [45]	■	■	■	■						•	•		•
Mahimkar et al. [49]			■	■	■		■		•	•	•		
Castelluccia et al. [71]		■	■	■	■			■	•				•
Albath et al. [73]			■	■	■				•	•	•		•
Sun et al. [75]			■	■	■				•	•	•		•
RSAED			■		■				•	•	•	•	•
SASPKC			■	■	■				•	•	•		•

A.C : L'Analyse des Chiffrés - **A.T.C** : Attaque à Texte en Clair connue - **A.C.A** : Attaque par Capture - **A.D.S** : Attaque par Déni de Service - **A.E.S** : Attaque par Envoi Sélectif des paquets - **A.S** : Attaque Sybille – **A.R** : Attaque par Rejeu – **A.F** : Attaque Furtive.
Com : Charge de communication – **Cal** : Charge de calcul
Sca : Scalabilité – **Ver** : Versatilité

+++: Grande charge
++ : Moyenne charge
+ : Petite charge
• : Fourni - ■ : Vulnérable

Tableau 26. Comparaison entre les solutions

6. Conclusion

Dans ce chapitre, nous avons proposé un nouveau protocole de sécurité, nommé *SASKPC*, pour l'agrégation des données dans les réseaux de capteurs, en utilisant le cryptage à clé publique et le chiffrement homomorphique. Dans ces réseaux, l'association de l'agrégation des données avec la confidentialité et l'intégrité des données reste un défi. Notre solution proposée utilise un cryptage homomorphe et la fonction *Xor* pour fournir respectivement la confidentialité de bout en bout et l'intégrité de bout en bout. Les résultats expérimentaux et de simulation ont confirmé l'efficacité de notre proposition en termes d'énergie et montrent que notre système assure un niveau de sécurité élevé avec des frais généraux minimes par rapport à d'autres modèles proposés dans la littérature actuelle. Au meilleur de notre connaissance, ce travail est le premier à aborder les questions de sécurité de l'agrégation des données dans les réseaux de capteurs sans fil en utilisant le *CCPE*. Notre modèle peut être très utile dans une application militaire, où la station de base a besoin de recueillir en continu des données pertinentes de chaque nœud capteur sur la zone cible, et ce, tout en offrant une forte sécurité contre l'adversaire qui vise à perturber le réseau.

Conclusion générale et perspectives

L'agrégation des données est un paradigme essentiel dans les RCSFs, permettant une réduction considérable de l'énergie utilisée. Cependant, les RCSFs sont sujets à plusieurs types d'attaques malicieuses. Ainsi, il est important de fournir un certain niveau de sécurité. Dans cette thèse nous avons traité plusieurs questions liées à la sécurité des données agrégées. Les travaux de recherche, menés de cette thèse, sont résumés dans ce qui suit:

Dans le premier chapitre, nous avons introduit le concept des RCSFs en donnant un aperçu général sur les types de dispositifs actuels pouvant constituer un tel réseau et aussi un bon nombre d'application potentielle. Ensuite nous avons présenté quelques challenges liés à la conception de ce type de réseaux. Enfin l'accent est mis sur l'agrégation des données, son avantage et les problèmes liés à la sécurité. Pour faire face à ces problèmes, les besoins de sécurité dans un RCSFs ont été présentés par la suite.

Dans le deuxième chapitre, nous avons présenté un état de l'art détaillé, une étude et une analyse de plusieurs solutions à la fois récentes et pertinentes. Ces solutions ont été classifiées en deux catégories, à savoir préventives et curatives. En se basant sur la cryptographie, les solutions préventives permettent de fournir des services de sécurité tels que la confidentialité et l'intégrité, ces solutions ont été par la suite classifiées en deux sous catégories à savoir les solutions de bout en bout et les solutions saut à saut. Après notre analyse, nous avons déduit que celles de bout en bout offre une meilleure sécurité et efficacité. Cependant, les algorithmes permettant de fournir cette sécurité sont généralement coûteux et non versatiles. Les solutions curatives visent quant à elle la disponibilité des données et la sécurité contre les dénis de service, et ceci en se basant sur des algorithmes de surveillance. Cependant, ces algorithmes encourent généralement des charges de communications importantes et parfois même des calculs intensifs dus aux algorithmes d'apprentissage. Pour fournir une analyse de la sécurité et des performances, ces solutions ont été comparées selon (i) les services de sécurité fournis (ii) les attaques auxquelles elles sont vulnérables et (iii) les charges introduites en termes de calcul et de communication afin d'accomplir une agrégation sécurisée.

Le troisième chapitre présente notre première contribution qui adresse les problèmes du gaspillage des ressources et du rejet total des données agrégées. Cette contribution, appelée *RSAED*, améliore la fiabilité et aussi la robustesse du réseau. L'objectif principal de cette solution est d'offrir une sécurité additionnelle et efficace à un chiffrement homomorphique (*ECEG*). Ce dernier est intrinsèquement malléable et plusieurs attaques actives peuvent être exécutées. Pour faire face, nous avons proposé une vérification saut à saut, qui non seulement permet une détection précoce de l'attaque et le rejet du paquet malicieux, mais aussi une disponibilité pour les données correctes. Aussi, et pour des fins d'efficacité, un calcul distribué de la fonction homomorphique à été proposé en employant deux agrégateurs dans chaque cluster. Cette méthode nous a permis d'alléger les charges de calcul et de communication sur l'agrégateur mais aussi le délai de bout en bout par un facteur ½.

Nous nous sommes focalisés dans le quatrième chapitre sur l'implémentation efficace et sécurisée de l'*ECEG* pour une sécurité de bout en bout dans les RCSFs. Nous avons commencé par une brève introduction de la cryptographie sur les courbes elliptiques. Les opérations de base au niveau de la courbe sont l'addition de points et le doublement de points. Puisque la multiplication par un scalaire, qui est l'opération la plus importante dans l'*ECC*, est basée sur ces deux opérations, nous avons examiné les méthodes qui permettent d'améliorer leur performance. Notre analyse a montré que ces deux opérations sont efficacement effectuées, quand elles sont représentées en coordonnées mixtes. Ensuite, une analyse de plusieurs algorithmes pour une *MPS* efficace et sécurisée contre *SPA* a été achevée. L'algorithme *Sec_comb* permet une *MPS* en 0.52s et un chiffrement sécurisé de l'*ECEG* en 1.29s sur le capteur MicaZ, ce qui constitue un nouveau record pour l'implémentation logicielle de l'*ECC*. Aussi, nous avons étudié une autre opération coûteuse à savoir la décompression des points. L'algorithme *Rep.Exp&SWM* permet un calcul de l'exponentiation en 0.316s, traduisant un temps très appréciable par rapport à l'utilisation des algorithmes traditionnels.

La troisième contribution de cette thèse, présentée dans le cinquième chapitre, consiste en une nouvelle solution pour sécuriser l'agrégation dans les RCSFs. Une solution appelée *SASPKC* basée sur un nouveau type de chiffrements à clé publique avec état. Ce type de chiffrements permet un calcul efficace et une charge de communication réduite par rapport aux algorithmes asymétriques traditionnels. *SAPSKC* est composé des deux phases de transmission et d'agrégation. L'analyse de sécurité montre que *SASPKC* assure les deux services (confidentialité et intégrité) de bout en bout, et permet aussi une identification du nœud malicieux grâce à la fonction d'encodage employée. L'analyse des performances montre quant à elle que *SASPKC* peut atteindre un niveau de sécurité comparable aux solutions proposées dans la littérature mais avec une réduction considérable de la charge en termes de calcul et de communication.

Les travaux présentés dans cette thèse traitent les problèmes liés à la sécurité des données agrégées dans les RCSFs et conduisent à plusieurs perspectives envisageables.

Il est clair que la cryptographie est le meilleur moyen pour sécuriser les communications dans les réseaux de capteurs. Toutefois, les solutions préventives y compris *SASPKC* et *RSAED*, souffrent d'un problème lorsque l'agrégateur est compromis. En effet, un agrégateur compromis, par l'attaque de renvoi sélectif des paquets, peut ignorer certains paquets et transmettre des paquets de son choix. Malheureusement, la cryptographie ne peut pas atténuer de telles attaques, donc plus de travaux de recherche doivent être menés dans ce sens.

La mobilité est un aspect très important permettant l'extension du domaine d'application des RCSFs. Par exemple, des applications intéressantes utilisent des capteurs pouvant se rapprocher de la cible ou pouvant être associées à des personnes, à des voitures ou à toute entité mobile. Actuellement, la plupart des travaux de recherche supposent des réseaux statiques alors que de nombreuses applications peuvent obtenir des résultats significatifs si

la mobilité est prévue. Les solutions *RSAED* et *SASPKC* ont l'avantage d'utiliser l'*ECEG* qui est flexible, il est donc intéressant d'étendre ces travaux pour supporter la mobilité.

Aussi, l'implémentation logicielle des *ECC* est jusqu'à maintenant limitée et ne peut servir qu'un nombre limité d'applications. L'implémentation matérielle conduit à des calculs très efficaces. Cependant, elle est coûteuse et le coût devient considérable dans les réseaux à grande échelle. Donc, la recherche d'algorithmes plus efficaces permettant de contrer d'autres types d'attaques *SCA* [113] constitue une orientation intéressante pour l'avenir.

Le chiffrement homomorphique permet, soit l'addition, soit la multiplication comme opérations sur les données chiffrées. Des algorithmes qui permettent n'importe quelle fonction sur les chiffrés sont largement souhaités. La percée théorique de Gentry [56] est très intéressante mais elle est loin de l'être par la pratique. Concevoir de tels algorithmes est l'une des perspectives les plus attrayantes par la communauté scientifique.

Les solutions curatives permettent de fournir la disponibilité et de contrer les attaques internes. Cependant, la plupart des solutions utilisent une quantité considérable de ressources afin de trouver l'attaque. La recherche d'algorithmes de surveillance et d'apprentissage plus efficaces est intéressante pour l'avenir. Cette efficacité peut permettre à envisager l'utilisation des primitives de chiffrement afin de fournir plus de services de sécurité.

Liste des publications

- Conférences internationales avec comité de lecture

1- Merad Boudia Omar Rafik and Mohammed Feham. Fast and secure implemenattion of ECC based concealed data aggregation in wireless sensor networks In : *The 5th Global Information Infrastructure and Networking Symposium (GIIS), 2013*. IEEE, 2013. 1-7. Trento, Italy.

ISBN : 978-1-4799-2969-6
DOI : 10.1109/GIIS.2013.6684371

2- Merad Boudia Omar Rafik and Mohammed Feham. The impact of ECC's scalar multiplication on wireless sensor networks. In : *11th International Symposium on Programming and Systems (ISPS), 2013*. IEEE, 2013. p. 17-23. Algiers, Algeria.

ISBN : 978-1-4799-1152-3
DOI : 10.1109/ISPS.2013.6581488

3- Merad Boudia Omar Rafik and Mohammed Feham. SA-SPKC: Secure and efficient aggregation scheme for wireless sensor networks using stateful public key cryptography. In: *11th International Symposium on Programming and Systems (ISPS), 2013*. IEEE, 2013. p. 96-102. Algiers, Algeria.
Prix pour le meilleur article.

ISBN : 978-1-4799-1152-3
DOI : 10.1109/ISPS.2013.6581500

4- Merad Boudia Omar Rafik and Mohammed Feham. The impact of point representation on wireless sensor networks. In: *International Conference on Artificial Intelligence and Information Technology (ICAIIT)*, Mars 2014. Ouargla, Algeria.

- Conférences nationales

1- Merad Boudia Omar Rafik and Mohammed Feham. Performance evaluation on TelosB mote of a secure data aggregation protocol using ECC In : *Conférence Nationale sur les Technologies de l'Information et les Télécommunications*, Tlemcen, Algeria.

- Revues internationales

1- Merad Boudia Omar Rafik, Feham Mohammed and Senouci Sidi Mohammed. "A Novel Secure Aggregation Scheme for Wireless Sensor Networks Using Stateful Public Key Cryptography "– *Wiley SCN*, Soumis.

2- Merad Boudia Omar Rafik and Feham Mohammed. «RSAED: Robust and Secure Aggregation of Encrypted Data in Wireless Sensor Networks », *International Journal of Network Security & Its Applications (IJNSA)*, Vol.4, No.6, November 2012.

ISSN : 0974-9330
DOI : 10.5121/ijnsa.2012.4601

3- Merad Boudia Omar Rafik and Feham Mohammed. "Secure Data Aggregation in Wireless Sensor Networks: State of the art and future works ", *(IJCSIS) International Journal of Computer Science and Information Security*,Vol. 9, No. 8, August 2011.

ISSN : 1947-5500

REFERENCES

[1] Akyildiz, I. F. and Vuran, M. C. Introduction, In : *Wireless Sensor Networks*, *John Wiley & Sons, Ltd, Chichester*, UK. 2010.

[2] Crossbow technology. http://www.xbow.com.

[3] HAVINGA, Paul, ETALLE, Sandro, KARL, Holger, et al., "Eyes–energy efficient sensor networks", In : *Personal Wireless Communications. Springer Berlin Heidelberg*, pp. 198-201, 2003. Venice, Italy.

[4] Ahonen, Teemu, Reino Virrankoski, and Mohammed Elmusrati. "Greenhouse monitoring with wireless sensor network". In *IEEE/ASME International Conference on Mechtronic and Embedded Systems and Applications*, pp. 403-408, 2008. Beijin, China.

[5] YingMing, G., & RenCheng, J., "A Novel Wireless Sensor Networks Platform for Habitat Surveillance", In: *International Conference on Computer Science and Software Engineering*, pp. 1028-1031. 2008. Wuhan, Hubei.

[6] N. Alsharabi, L. R. Fa, F. Zing, and M. Ghurab, "Wireless sensor networks of battlefields hotspot: challenges and solutions", In : *Sixth Int. Symposium on Modeling and Optimisation in Mobile adhoc and Wireless Networks and Workshops*, pp.192-196, April 2008. Berlin, Germany.

[7] ROLADER, Glenn E., ROGERS, John, et BATTEH, Jad. "Self-healing minefield". In : *Defense and Security. International Society for Optics and Photonics*, p. 13-24, 2004.

[8] R. Tan, G. Xing, J. Wang, and H. C. So, "Collaborative target detection in wireless sensor networks with reactive mobility". In: *16th Int. Workshop on Quality of Service*, pp. 150-159. 2008. Enschede. Hollande.

[9] Q. Ling, Z. Tian, Y. Yin, and Y. Li. "Localized structural health monitoring using energy efficient wireless sensor networks" *IEEE Sensors Journal*, vol. 9, no.11, pp.1596 – 1604, 2009.

[10] D. D. L. Mascaranes, E. B. Flynn, M. D. Todd, T. G. Overly, K. M. Farinholt, G. Park, and C. R. Farrar, "Development of capacitance based and impedance based wireless sensors and sensor nodes for structural health monitoring applications", *Journal of Sound and Vibration*, vol. 329, pp. 2410-2420, June 2010.

[11] C. C. Song, Y. C. Hsu, C. F. Feng, and Y. K. Chen, "Construction of a wireless sensor networking platform with vibration sensing and GPS positioning", *ICROS-SICE Intl. Joint Conf*, , pp. 5570-5575, 2009. Fukuoka, Japan.

[12] A. Flemmini, P. Ferrari, D. Marioli, E. Sisinni, and A. Taroni, "Wired and wireless sensor networks for industrial applications", *Microelectronics Journal*, vol. 40, pp. 1322-1336, September 2009.

[13] S. Lee, D. Yoon, and A. Ghosh, "Intelligent parking lot application using wireless sensor networks", In : *Symposium on Collaborative Technologies and Systems*, pp. 48-57, 2008. California, USA.

[14] S.V. Srikanth, P. J. Pramod, K. P. Dileep, S. Tapas, M. U. Patel, S. C. Babu, "Design and implementation of a prototype smart PARKing (SPARK) system using wireless sensor networks", In : *Intl. Conf. on Advanced Information Networking and Applications Workshop*, pp. 401-406, 2009. Bradford, England.

[15] H. Yan, Y. Xu, and M. Gidlund, "Experimental e-health applications in wireless sensor networks", In : *Intl Conf. on Communications and Mobile Computing*, pp. 563-567, 2009. Yunnan, China.

[16] R. Steele, A. Lo, C. Secombe, and Y. K. Wong, "Elderly persons' perception and acceptance of using wireless sensor networks to assist healthcare", *International Journal of Medical Informatics*, vol. 78, pp.788-801, December 2009.

[17] H. Huo, Y. Xu, H. Yan, S. Mubeen, and H. Zhang, "An elderly health care system using wireless sensor networks at home", In : *Third Intl. Conf. on Sensor Technologies and Application*, pp. 158-163, 2009. Athens, Glyfada.

[18] J. F. Martinez, M. S. I. Familiar, , Corredor, A. B. Garcia, S. Bravo, and L. Lopez, "Composition and deployment of e-health services over wireless sensor networks", *Mathematical and Computer Modelling*, vol. 53, no. 3-4, pp. 485-503, February 2011.

[19] K. Akkaya, M.Demirbas, R.S. Aygun, "The Impact of Data Aggregation on the performance of Wireless Sensor Networks", *Wiley Wireless Communication Mobile Computing (WCMC)*, J(8), 171-193, 2008.

[20] Rajagopalan, R., Varshney, P.K. "Data-aggregation techniques in sensor networks: a survey". *IEEE Communications Surveys & Tutorials* 8(4), 48–63 (2006) 31.

[21] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, F. Silva. "Directed diffusion for wireless sensor networking", in: *IEEE/ACM Transactions on Networking*, vol. 11, pp. 2–16, 2003.

[22] S. Madden et al., TAG: "A Tiny AGgregation Service for Ad Hoc Sensor Networks", In : *5th symposium on Operating systems design and implementation OSDI*, pp. 131–146, 2002. Boston, USA.

[23] M. Lee, V.W.S. Wong, "An Energy-Aware Spanning Tree Algorithm for Data AggrAegation in Wireless Sensor Networks", In : *IEEE Pacific Rim Conference on Communications, Computers and signal Processing,* pp. 300–303. 2005. Victoria, Canada.

[24] S. Lindsey, C. Raghavendra, K.M. Sivalingam, "Data gathering algorithms in sensor networks using energy metrics", *IEEE Trans.parallel Distrib*. Sys. 13 (9). 924–935. 2002.

[25] W.B. Heinzelman, A.P. Chandrakasan, H. Balakrishnan, "An application-specific protocol architecture for wireless microsensor networks", In : *IEEE Trans. Wireless Commun*. 1 (4) 660–670. 2002.

[26] O. Younis, S. Fahmy, "HEED: a hybrid, energy-efficient distributed clustering approach for ad hoc sensor networks", *IEEE Trans. Mobile Comput*. 3 (4). 366–379. 2004.

[27] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. "The impact of data aggregation in wireless sensor networks". In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, *ICDCSW'02*, pp. 575-578, July 2-5, 2002. Vienna, Austria.

[28] David Wagner. "Resilient aggregation in sensor networks". In *Sanjeev Setia and Vipin Swarup, editors, Proceedings of the 2nd ACM Workshop on Security of ad hoc and Sensor Networks*, SASN'04, pp. 78-87, October 25, 2004. Washington, DC, USA.

[29] Xiangqian Chen, Kia Makki, Kang Yen, Niki Pissinou, "Sensor network security: a survey". *IEEE Communications Surveys and Tutorials* 11(2): 52-73 (2009).

[30] T. Roosta, S. Shieh, S. Sastry , "Taxonomy of security attacks in sensor networks", In: *The First IEEE International Conference on System Integration and Reliability Improvements*, *IEEE*, pp. 13-15. 2006, Washington, DC, USA.

[31] Giacomo de Meulenaer, François-Xavier Standaert, "Stealthy Compromise of Wireless Sensor Nodes with Power Analysis Attacks", *MOBILIGHT* 2010: pp. 229-242. Barcelona, Spain.

[32] Sanjeev Setia, Sankardas Roy, and Sushil Jajodia, "Secure data aggregation in wireless sensor networks",In : *Javier Lopez and Jianyin. Zhou, editors, Wireless Sensor Network Security*, chapter 8, pages 204-222. IOS press, 2008.

[33] Yingpeng Sang, Hong Shen, Yasushi Inoguchi, Yasuo Tan, and Naixue Xiong, "Secure data aggregation in wireless sensor networks: A survey", In : *The 7th international Conference on Parallel and Distributed Computing, Applications and Technologies, PDCAT'06*, pp. 315-320, 2006. Taipei, Taiwan.

[34] Suat Ozdemir and Yang Xiao, "Secure data aggregation in wireless sensor networks: A comprehensive overview", *Computer Networks*, 53(12):2022-2037, 2009.

[35] Merad Boudia Omar Rafik, Feham Mohammed, "Secure Data Aggregation in Wireless Sensor Networks: State of the art and future works", *(IJCSIS) International Journal of Computer Science and Information Security*, Vol. 9, No. 8, August 2011.

[36] Joan Daemen and Vincent Rijmen, "The Design of Rijndael: AES – The Advanced Encryption Standard" , *Springer-Verlag*, 2002. ISBN 3540425802.

[37] R.L. Rivest, "The RC5 Encryption Algorithm", In : *The Second International Workshop on Fast Software Encryption (FSE)*. pp. 86-96. 1994. Belgium.

[38] NIST 1998. Skipjack and KEA Algorithm Specifications Version 2.0. NIST.

[39] R. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems", *Communications of the ACM*, Vol. 21 (2), pp.120–126. 1978. New York, USA.

[40] Taher ElGamal, "A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms", In : *IEEE Transactions on Information Theory*, n. 4, 1985, pp.469-472 or *CRYPTO 84*, pp.10-18, *Springer-Verlag*.

[41] W. Diffie and M. E. Hellman, "New Directions in Cryptography", *IEEE Transactions on Information Theory*, vol. IT-22, pp: 644-654. Nov.1976.

[42] Roberto M. Avanzi, Henri Cohen, Christophe Doche, Gerhard Frey, Tanja Lange, Kim Nguyen, Frederik Vercauteren, "Handbook of Elliptic and Hyperelliptic Curve Cryptography".2006.

[43] A. K. Lenstra and E. R. Verheul, "Selecting cryptographic key sizes", In : *Journal of Cryptology*, 14(4):255-293, 2001.

[44] Fontaine and Galand, "A Survey of Homomorphic Encryption for Nonspecialists", In : *EURASIP Journal on Information Security*, volume 2007, pages 1-15, 2007.

[45] Bartosz Przydatek, Dawn Xiaodong Song, Adrian Perrig, "SIA: secure information aggregation in sensor networks", In : *Conference On Embedded Networked Sensor Systems*, SenSys 2003: 255-265. L.A, California, USA.

[46] Ralph C. Merkle, "Protocols for public key cryptosystems", In : *IEEE Symposium on Security and Privacy*, pp. 122-134, April 14-16, 1980. Oakland, California, USA.

[47] H. Krawczyk, M. Bellare, R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", *RFC editor*. February 1997.

[48] Wenliang Du, Jing Deng, Yunghsiang S. Han, and Pramod Varshney, "A witness-based approach for data fusion assurance in wireless sensor networks", In : *The IEEE Global Communications Conference, GLOBECOM'03*, volume 3, pages 1435 - 1439, December 1-5, 2003. San Francisco, USA.

[49] Mahimkar, A., Rappaport, T.S, "SecureDAV: A secure data aggregation and verification protocol for sensor networks", In : *The IEEE Global Telecommunications Conference*, pp. 2175-2179. 2004. Dallas, Texas, USA.

[50] D. Johnson, A. Menezes and S. Vanstone, "The Elliptic Curve Digital Signature Algorithm (ECDSA)", *Springer-Verlag*, 2001.

[51] H. Ozagur Sanli, Suat Ozdemir, and Hassan Cam, "SRDA: secure reference-based data aggregation protocol for wireless sensor networks", In : *The 60th IEEE Vehicular*

Technology Conference, *VTC'04*, volume 7, pages 4650-4654, September 26-29, 2004. Los Angeles, USA.

[52] Yee Wei Law, Jeroen Doumen, and Pieter H. Hartel, "Survey and benchmark of block ciphers for wireless sensor networks", *ACM Transactions on Sensor Networks (TOSN)*, 2(1):65-93, 2006.

[53] Vu, Hai, Neeraj Mittal, and Subbarayan Venkatesan, "THIS: THreshold security for Information aggregation in Sensor networks", In: *Fourth International Conference on Information Technology, ITNG'07. IEEE*, pp. 89-95. 2007. Las Vegas, Nevada, USA.

[54] Yang, Yi, et al, "SDAP: A secure hop-by-hop data aggregation protocol for sensor networks" *ACM Transactions on Information and System Security (TISSEC)* 11.4 (2008): 18.

[55] FE Grubbs, "Procedures for detecting outlying observations in samples", *Technometrics*, 11(1):1-21, 1969.

[56] C. Gentry, "Fully homomorphic encryption using ideal lattices", *Symposium on the Theory of Computing (STOC)*, 2009, pp. 169-178. Washinton, DC, USA.

[57] Rivest RL, Adleman L, Detrouzos ML, "On Data Banks and Privacy Homomorphism". *Foundations of Secure Computation, Academia Press*. 1978.

[58] E. Mykletun, J. Girao, and D. Westhoff, "Public Key Based Cryptoschemes for Data Concealment in Wireless Sensor Networks", In : *IEEE Int. Conference on Communications ICC*, pp. 2288 - 2295 June 2006, Istanbul, Turkey.

[59] Osman Ugus, Dirk Westhoff, Ralf Laue, Abdulhadi Shoufan, Sorin A. Huss, "Optimized Implementation of Elliptic Curve Based Additive Homomorphic Encryption for Wireless Sensor Networks", *WESS '07*, Salzburg, Austria, 2007.

[60] Aggelos Kiayias, Moti Yung, "Tree-Homomorphic Encryption and Scalable Hierarchical Secret-Ballot Elections", *Financial Cryptography* 2010: pp. 257–271.

[61] Julien Bringer, Herv´e Chabanne, Malika Izabach´ene, David Pointcheval, Qiang Tang and S´ebastien Zimmer, "An Application of the Goldwasser-Micali Cryptosystem to Biometric Authentication", *Information Security and Privacy, LNCS* 4586, pp. 96–106, 2007.

[62] Li, F., Luo, B., & Liu, P., "Secure information aggregation for smart grids using homomorphic encryption. In : *First IEEE International Conference on Smart Grid Communications (SmartGridComm)*, pp. 327-332. 2010. Gaithersburg, MD

[63] M. Kantarcioglu, "Privacy-preserving distributed data mining and processing on horizontally partitioned data", *PhD. dissertation, Department of Computer Science, Purdue University*, 2005.

[64] S. Goldwasser, S. Micali, "Probabilistic Encryption", *J. Comp. Sys. Sci.*, 28, pp. 270–299, 1984.

[65] Josh Benaloh, "Dense Probabilistic Encryption", *SAC 94*, pages 120–128, 1994.

[66] D. Naccache, J. Stern., "A New Public Key Cryptosystem Based on Higher Residues", In : *The 5th ACM CCS*, pages 59–66, 1998. NY. USA

[67] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes", In : *Advances in Cryptology EUROCRYPT'99*, LNCS 1592, pp. 223–238. 1999, *Springer*, New York, NY, USA.

[68] T. Okamoto and S. Uchiyama., "A New Public-Key Cryptosystem as Secure as Factoring". In : *Eurocrypt*, *LNCS 1403*, pp. 308-318, 1998. Finland.

[69] Taher El Gamal. "A public key cryptosystem and a signature scheme based on discrete logarithms", In : *CRYPTO 84 on Advances in Cryptology, Springer-Verlag*, Inc., pp.10–18. 1985. New York.

[70] Dirk Westhoff, Joao Girão, Mithun Acharya, "Concealed Data Aggregation for Reverse Multicast Traffic in Sensor Networks: Encryption, Key Distribution, and Routing Adaptation". *IEEE Trans. Mob. Comput*. 5(10): 1417-1431. 2006.

[71] Castelluccia, C., Mykletun, E., Tsudik, G.: "Efficient Aggregation of Encrypted Data in Wireless Sensor Networks", In: *MobiQuitous,* pp. 109 - 117. 2005. San Diego, USA.

[72] D. Boneh, E. Goh, K. Nissim, "Evaluating 2-DNF Formulas on Ciphertexts", *TCC '05, LNCS* 3378, pp. 325-341, 2005.

[73] Albath, J., Madria, S.: "Secure Hierarchical Aggregation in Sensor Networks", In : *The IEEE Wireless Communications and Networking Conference*. pp. 196-201 .2009. Bengaluru, Karnataka.

[74] KUMAR, Vimal et MADRIA, Sanjay Kumar., "Secure Hierarchical Data Aggregation in Wireless Sensor Networks: Performance Evaluation and Analysis". In : *IEEE 13th International Conference on Mobile Data Management (MDM),*. IEEE, 2012. p. 196-201. Bengaluru, Karnataka.

[75] Sun, H.-M., Hsiao, Y.-C., Lin, Y.-H., Chen, C.-M., "An Efficient and Verifiable concealed Data Aggregation Scheme in Wireless Sensor Networks", In : *The 2008 International Conference on Embedded Software and Systems*, pp. 19–26. 2008. Sichuan.

[76] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps", In : *EUROCRYPT 2003*, pp. 416-432. 2003.

[77] CHEN, Chien-Ming, LIN, Yue-Hsun, LIN, Ya-Ching, et al. "RCDA: recoverable concealed data aggregation for data integrity in wireless sensor networks", *IEEE Transactions on Parallel and Distributed Systems*, 2012, vol. 23, no 4, p. 727-734.

[78] Suat Ozdemir, Yang Xiao, "Integrity protecting hierarchical concealed data aggregation for wireless sensor networks", *Computer Networks* 55(8): 1735-1746. 2011.

[79] S. Ozdemir, "Secure and reliable data aggregation for wireless sensor networks", In: *H. Ichakawa et al. (Eds.), LNCS* 4836, pag. 102-109. 2007

[80] A. Josang, E. Ismail, "The beta reputation system", In: *The 15th Bled Conference Electronic Commerce*, 2002.

[81] Suat Ozdemir, "Functional Reputation Based Reliable Data Aggregation and Transmission for Wireless Sensor Networks", *Computer Communications, Elsevier*, vol. 31, no. 17, pp. 3941-3953, Nov. 2008.

[82] Sedjelmaci, H., Senouci, S. M. and Feham, M. (2013), "An efficient intrusion detection framework in cluster-based wireless sensor networks", *Security Comm. Networks*, 6: 1211–1224. 2013

[83] B. Sun, X. Jin, K. Wu, Y. Xiao, "Integration of secure in-network aggregation and system monitoring for wireless sensor networks", In: *Proceedings of IEEE International Conference on Communications (IEEE ICC'07)*, 2007, pp. 1466–1471. Glasgow, Switzerland.

[84] Sun, B., X. Shan, K. Wu, Y. Xiao,. "Anomaly detection based secure in-network aggregation for wireless sensor networks", *Systems Journal, IEEE,* 7: 13-25. 2013.

[85] A. Gursel, O. Mistry, S. Sandip, " Robust Trust Mechanisms for Monitoring Aggregator Nodes in Sensor Networks", *Int. Workshop on Agent Technology for Sensor Networks (ATSN-08)*, May 2008. Estoril, Portugal.

[86] LABRAOUI, Nabila, GUEROUI, Mourad, ALIOUAT, Makhlouf., "RAHIM: Robust Adaptive Approach Based on Hierarchical Monitoring Providing Trust Aggregation for Wireless Sensor Networks". *J. UCS, 2011*, vol. 17, no 11, p. 1550-1571.

[87] CHAN, Aldar C.-F. et CASTELLUCCIA, Claude., "On the privacy of concealed data aggregation", In : *Computer Security–ESORICS 2007. Springer Berlin Heidelberg*, 2007. p. 390-405.

[88] DE MEULENAER, Giacomo, GOSSET, François, STANDAERT, F.-X., et al., "On the energy cost of communication and cryptography in wireless sensor networks", In : *WIMOB'08. IEEE International Conference on Wireless and Mobile Computing Networking and Communications. IEEE*, 2008. p. 580-585. Avignon, France.

[89] LEE, Jongdeog, KAPITANOVA, Krasimira, et SON, Sang H. The price of security in wireless sensor networks. *Computer Networks*, 2010, vol. 54, no 17, p. 2967-2978.

[90] Merad Boudia Omar Rafik and Feham Mohammed. Fast and secure implementation of ECC-based concealed data aggregation in WSN. In : *Global Information Infrastructure Symposium, 2013. IEEE*, 2013. p. 1-7. Trento, Italy

[91] PETER, Steffen, WESTHOFF, Dirk, et CASTELLUCCIA, Claude, "A survey on the encryption of convergecast traffic with in-network processing", In : *IEEE Transactions on Dependable and Secure Computing, 2010*, vol. 7, no 1, p. 20-34.

[92] Merad Boudia Omar Rafik and Feham Mohammed, "RSAED: Robust and Secure Aggregation of Encrypted Data in Wireless Sensor Networks", *International Journal of Network Security & Its Applications (IJNSA)*, Vol.4, No.6, November 2012.

[93] J. Domingo-Ferrer, "A provably secure additive and multiplicative privacy homomorphism", In : *Proceedings of the Information Security Conference*, 2002, pp. 471-483.

[94] D. Wagner, "Cryptanalysis of an Algebraic Privacy Homomorphism", In: *Proceeding of the 6th Information Security Conference (ISC03)*, pp. 234-239. October 2003. Bristol, UK

[95] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, and Sheueling Chang Shantz, "*Comparing elliptic curve cryptography and rsa on 8-bit cpus*", In *CHES,* pp. 119–132, 2004. Boston.

[96] Boston, J. Pollard, "Monte Carlo methods for index computation mod p", *Mathematics of Computation*,Volume 32, 1978.

[97] An Liu, Peng Ning, "TinyECC: A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks", In : *Proceedings of the 7th International Conference on Information Processing in Sensor Networks (IPSN 2008), SPOTS Track,* pages 245--256, April 2008.

[98] http://www.tinyos.net.

[99] Certicom Research, "Standards for efficient cryptography-SEC 2:Recommended elliptic curve domain parameters", http://www.secg.org/collateral/sec2_final.pdf, September 2000.

[100] Avrora, The AVR simulation and analysis framework.
http://compilers.cs.ucla.edu/avrora/index.html.

[101] Merad Boudia Omar Rafik and Mohammed Feham, "Performance evaluation on TelosB mote of a secure data aggregation protocol using ECC", In : *Conférence Nationale sur les Technologies de l'Information et les Télécommunications*, Tlemcen, Algeria.

[102] Hadong Wang and Qun Li., "Efficient Implementation of Public Key Cryptosystems on MICAz and TelosB Motes". *Technical report, College of William and Mary*, October 2006.

[103] Szczechowiak, P., Oliveira, L.B., Scott, M., Collier, M., Dahab, R.:NanoECC, "Testing the Limits of Elliptic Curve Cryptography in Sensor Networks", *In EWSN(2008)* 305-320.

[104] N. Koblitz, "Elliptic curve cryptosystems". *Mathematics of Computation*, 48:203–209,1987.

[105] Darrel Hankerson, Alfred J. Menezes, and Scott Vanstone," Guide to Elliptic Curve Cryptography", *Springer*, New York, 2004.

[106] P.L. Montgomery, "Speeding up the Pollard and elliptic curve methods of factorization", *Mathematics of Computation*, 48(177), pp. 243-264, January 1987.

[107] J.S.Coron. "Resistance against differential power analysis for elliptic curve cryptosystems". In: *Cryptography Hardware and Embedded Systems-CHES'99*, C.K. Koc and C.Paar, editors, vol 1717, LNCS, pp. 292-302, 1999.

[108] B. Moller, "Securing elliptic curve point multiplication against side-channel attacks", In: *Information Security, G.I Davida and Y. Frankel, editors,* vol 2200,LNCS, pp. 324-334, 2001.

[109] C. Lim and P. Lee, "More Flexible Exponentiation with Precomputation", *Advances in Cryptology – CRYPTO'94*, LNCS 839, pp. 95–107, Springer-Verlag, 1994.

[110] IEEE P1363: Standard Specifications for Public-Key Cryptography. (2000), http://grouper.ieee.org/groups/1363/P1363/draft.html.

[111] Daniel M. Gordon. "A Survey of Fast Exponentiation Methods", *J. Algorithms* 27(1): pp. 129-146. 1998.

[112] Hedabou, M., Beneteaus, L., & Pinel, P., "Some ways to secure Elliptic Curve Cryptosystem", *Advances in Applied Clifford Algorithm*, 18, pp. 677–688. 2008.

[113] Mustapha Hedabou, "Amélioration et sécurisation des calcus arithmetiques pour la cryptographie basée sur les courbes elliptiques, *PhD. dissertation, Department of Computer Science*, *Toulouse University*, 2006.

[114] Billy Bob Brumley, Kimmo U. Järvinen, "Fast Point Decompression for Standard Elliptic Curves", *EuroPKI 2008*: 134-149. Trondheim, Norway.

[115] Krzysztof Piotrowski, Peter Langendörfer, Steffen Peter, "How public key cryptography influences wireless sensor node lifetime", *SASN 2006*: 169-176.

[116] M. Bellare, T. Kohno and V. Shoup, "Stateful Public-Key Cryptosystems: How to Encrypt with One 160-bit Exponentiation", *In ACM-CCS*, pp. 380–389, 2006.

[117] J.Baek, H.Chiang Tan, J.Zhou and J.W.Wong: "Realizing Stateful Public Key Encryption in Wireless Sensor Network", *23rd International Information Security Conference,* pp. 95–107. 2008, Boston. USA.

[118] Merad Boudia Omar Rafik and Mohammed Feham, "SA-SPKC: Secure and efficient aggregation scheme for wireless sensor networks using stateful public key cryptography", In : *11th International Symposium on Programming and Systems (ISPS)*, 2013. IEEE, 2013. p. 96-102. Algiers, Algeria.

[119] J. Katz and Y. Lindell, "Aggregate Message Authentication Codes", In : *CT-RSA, Springer-Verlag* (LNCS 4964), pages 155-169, 2008.

[120] T. Okamoto and D. Pointcheval. The Gap Problems, "A New Class of Problems for the Security of Cryptographic Schemes", *PKC '01, Lecture Notes in Computer Science* Vol. 1992, K. Kim ed., Springer-Verlag, 2001.

[121] Chan, A. C., & Castelluccia, C., "A security framework for privacy-preserving data aggregation in wireless sensor networks", *ACM Transactions On Sensor Networks (TOSN)*, 7(4), 29. 2011.

[122] Merad Boudia Omar Rafik and Mohammed Feham, "The impact of ECC's scalar multiplication on wireless sensor networks", In : *11th International Symposium on Programming and Systems (ISPS)*, 2013. IEEE, 2013. p. 17-23. Algiers, Algeria

[123] Victor Shnayder, Mark Hempstead, Bor-rong Chen, and Matt Welsh, Harvard University, "PowerTOSSIM: Efficient Power Simulation for TinyOS Applications", *SenSys2004*, November 2004.

Résumé

Les réseaux de capteurs sont des réseaux sans fil constitués d'un ensemble de dispositifs ayant des ressources informatiques limitées. Ce type de réseaux a attiré beaucoup d'attentions ces dernières années, non seulement dans le milieu universitaire mais aussi dans l'industrie, pour l'étude et le développement de plusieurs applications potentielles. Cependant, la contrainte des ressources représente la caractéristique la plus importante de ce réseau. L'agrégation des données est une technique très utilisée car elle permet de réduire le nombre de message transmis dans le réseau et par conséquent réduire la consommation d'énergie, ainsi améliorer la durée de vie du réseau. Cependant, Les réseaux de capteurs sont généralement déployés dans des environnements, souvent très hostiles et sans assistance. Par conséquent, le concepteur devrait, non seulement considérer les menaces de sécurité pouvant survenir dans un réseau facilement accessible à l'attaquant, mais aussi faire face à des ressources informatiques très limitées. Dans cette thèse, nous nous sommes intéressés aux problèmes de sécurité liés à l'agrégation des données dans les réseaux de capteurs sans fil. Nous avons proposé deux nouvelles approches ainsi qu'une implémentation efficace et sécurisée du chiffrement basé sur les courbes elliptiques.

Mots clés : Réseaux de capteurs sans fil, agrégation des données, sécurité, courbes elliptiques.

Abstract

Sensor networks are wireless networks consisting of a set of small and resources-constrained devices. Wireless sensor networks have received much attention over the last few years, not only in academia, but also in industries for the study and development of a plethora of potential applications. However, the resource constraint is the most important feature of such network. Data aggregation is one of the techniques that is actually considered as an essential paradigm for sensor networks since it tends to save computation and communication resources. Data aggregation allows in-network processing which leads to lesser packet transmissions and reduces redundancy, and therefore, helps in increasing the network's overall lifetime. However, sensor networks are usually deployed in unattended and hostile environments. Therefore, the designer should not only consider the security threats that can occur in an easily accessible network to the attacker, but also cope with the limited resources. In this thesis, we are interested in security problems related to data aggregation in wireless sensor networks. We proposed two new approaches and a fast and secure implementation of Elliptic Curve based encryption.

Keywords: Wireless sensor networks, data aggregation, security, elliptic curve.

ملخص

شبكات الاستشعار اللاسلكية تتكون من عدد وافر من الأجهزة ذات الموارد المحدودة. وقد اجتذب هذا النوع من الشبكات اهتماما كبيرا في السنوات الأخيرة، ليس فقط في الأوساط الأكاديمية ولكن أيضا في الصناعة، وذلك لدراسة و تطوير العديد من التطبيقات المحتملة.القيود المفروضة على الموارد هي أهم سمة من سمات هذه الشبكة. التجميع هو أسلوب يستخدم على نطاق واسع لأنه يقلل من عدد البيانات المرسلة في الشبكة، وبالتالي خفض استهلاك الطاقة و تحسين عمر الشبكة. يتم نشر شبكات الاستشعار عادة في بيئات معادية جدا في كثير من الأحيان و غير مراقبة. لذا ، ينبغي للمصمم النظر ليس فقط في التهديدات الأمنية التي يمكن أن تحدث، ولكن أيضا التعامل مع موارد محدودة للغاية . في هذه الأطروحة ، نحن مهتمون في القضايا الأمنية المتعلقة بتجميع البيانات في شبكات الاستشعار اللاسلكية. اقترحنا نهجين جديدين .يتناول النهج الأول مشاكل سرقة الهوية و الرفض التام من البيانات. بينما ينص الثاني خدمات الأمن من البداية للنهاية.

كلمات البحث: شبكات الاستشعار اللاسلكية، تجميع البيانات، الأمن