Résumé

Dans ce travail nous avons visé les applications qui nécessite de la sécurité et que l'information soit remontée rapidement au centre de contrôle. Pour cela nous avons utilisé la cryptographie basée sur les courbes elliptiques pour sécuriser l'information à acheminer et pour remonter rapidement l'information nous avons proposé une approche distribuée impliquant des algorithmes qui accélèrent les calculs.

La mise en place de cette approche exige des outils matériels et logiciels bien spécifiques puisque les capteurs sont des composants à ressources limitées. Pour cela nous avons utilisé le langage NesC qui fait minimiser l'utilisation de la mémoire et TinyOs qui est un système d'exploitation léger.

Mots clés : Réseaux de capteurs, Sécurité, Cryptographie, parallélisme, NesC, ECC.

Remerciements

«Et lorsque votre Seigneur proclama : "Si vous êtes reconnaissants, très certainement J'augmenterai [Mes bienfaits] pour vous» [Coran S14.V7]

J*e remercie Dieu le tout Puissant qui nous a donné la force et la volonté pour réaliser ce modeste travail.*

N*ous tenons à exprimer notre grande gratitude à notre encadreur Mr LAHSAINI, pour avoir accepté de nous encadrer tout au long de ce travail, pour sa disponibilité, son amabilité, ses conseils et suggestions et pour toute l'aide morale qu'il n'a cessé de nous prodiguer. Il nous a prodigué beaucoup de connaissances et conseils dans le domaine des réseaux de capteurs.*

N*ous tenons également à remercier Mme LABRAOUI pour l'honneur qu'elle nous fait de présider notre jury de soutenance nous lui exprimons notre gratitude profonde.*

N*os remerciements s'adressent ensuite à Mme DIDI et Mr BENMAMMAR qui ont aimablement accepté d'examiner et de juger notre modeste travail.*

N*ous remercions tous nos enseignants du département informatique.*

N*ous remercions profondément l'équipe RCSF (Zineb, Saadia, Abbes) pour leur collaboration et les moments innoubliables passés ensemble.*

S*ans oublier nos très chères familles et surtout parents pour leur contribution, leur soutien et leur patience.*

E*nfin, nous adressons nos plus sincères remerciements à tous nos proches et amis, qui nous ont toujours soutenus et encouragé au cours de la réalisation de ce mémoire. Merci à tous et à toutes.*

Dédicaces

Je dédie ce travail, à tous ceux qui me sont chers,

$\mathcal{A}$

Mes parents,

En ma très grande affectation et ma gratitude, Vous vous êtes dépensés pour moi sans compter. En reconnaissance de tous les sacrifices consentis par vous pour me permettre d'atteindre cette étape de ma vie. Vous représentez pour moi le symbole de la bonté par excellence, et l'exemple du dévouement. Aucune dédicace ne saurait exprimer l'amour, l'estime et le respect que j'ai toujours eu pour vous. "QUE DIEU VOUS GARDE et fasse qu'ils soient toujours fiers de moi"

$\mathcal{A}$

Mes frères ZIZIH et WIDOU qui ont su me guider et être un exemple pour moi,

$\mathcal{A}$

Ma poupée de nièce ZAZAR qui a fait de moi une "Taty", ma belle-sœur « nesrine » qui m'a toujours donnée de bons conseils,

$\mathcal{A}$

Mes cousins et cousines et toutes les familles TAHRAOUI, HAMIMED.

$\mathcal{A}$

Ma binôme AMINA avec qui j'ai partagé de belles années d'études et avec qui j'ai eu l'honneur de les finir. A toute sa famille BELHASSENA.

$\mathcal{A}$ *mes amis,*

Zinouba, Abbes, Saher pour tous les moments de fou rire en plein cours, Salah, Tarik, Saky, HBH, Walid, Amina, Téma, Saadia, sousou, selma,... et toutes les promos RSD, SIC, MID. Ainsi qu'à Djamel pour l'aide que tu nous as apporté pour ce modeste travaille.

TAHRAOUI Warda Ilhem

Dédicaces

Je dédie ce travail :

A mes parents qui ont éclairé mon chemin et qui m'ont encouragé et soutenue tout au long de mes études...

A mes frères : Sid Ahmed Et Houari

A mes soeurs : Houaria Et Fatima Zohra

A mes nièces : Louiza et Ayah

A mes chères amies Narimane, Amel , ma binôme Tahraoui Ilhem Warda , et a toute sa famille,

Enfin je dédie ce travail à mes chers amis et consultant et à tous ceux, qui de près ou de loin ont contribués à la réalisation de mon mémoire de fin d'études.

Amina Belhassena

Table des matières

Table des figures

Liste des tableaux

Abréviations

GPS : Global Positioning System.
WSN : Wireless Sensor Networks.
RCSF : Réseaux de Capteurs Sans Fil.
MAC : Media Access Control.
TDMA : Time Division Multiple Access.
S-MAC : Sensor MAC.
CSMA-CA : Carrier Sense Multiple Access with Collision Avoidance.
RTS/CTS : Request to Send / Clear to Send.
RC4 : Rivest Cipher 4.
DES : Data Encryption Standard.
AES : Advanced Encryption Standard.
CPU : Central Processing Unit.
SSL/TLS : Secure Sockets Layer / Transport Layer Security.
WM-RSA : Wireless sensor Motes RSA.
ECC : Elliptic Curve Cryptography.
TinyOS : Tiny Operating System.
ECDSA : Elliptic curve digital signature algorithm.
ECIES : Elliptic curve Integrated Encryption Scheme.
ECDH : Elliptic curve Diffie–Hellman.
MAC : Message Authentication Code.
DES-CBC : Data Encryption Standard Cipher-Block Chaininq.
TCP : Transmission Control Protocol.
RSA : Rivest Shamir Adleman.
PLD : Le Problème du Logarithme Discret.
PDHC : Le Problème de Diffie-Hellman Calculatoire.
NESC : National Electrical Safety Code.
LED : Light-Emitting Diode.
MIG : Message Interface Generator.

Introduction générale

Au cours de ces dernières années, le développement technologique des réseaux de communication sans fil, a connu un essor important grâce aux avancées technologiques dans divers domaines, telles que la micro-électronique et la miniaturisation. C'est ainsi que de nouvelles voies d'investigation ont été ouvertes avec l'émergence des réseaux de capteurs sans fil qui sont composés de petits dispositifs électroniques. Ces éléments sont autonomes, équipés de capteurs et capables de communiquer entre eux sans fil. En plus ils collaborent entre eux pour former un réseau de capteurs sans fil capable de superviser une région ou un phénomène d'intérêt, de fournir des informations utiles par la combinaison des mesures prises par les différents capteurs et de les communiquer ensuite via le support sans fil à un centre de contrôle distant.

Les nœuds capteurs sont conçus pour être déployés d'une manière dense dans des endroits hostiles et difficiles d'accès, d'où la nécessité de limiter au maximum leurs dimensions physiques qui s'obtiennent impérativement au détriment des capacités de calcul, de traitement et de ressources énergétiques.

En raison de leur déploiement en environnements ouverts, de leurs ressources limitées, et la nature broadcast du medium de transmission, les réseaux de capteurs doivent faire face à de nombreuses attaques. Sans mesures de sécurité, un agent malveillant peut lancer plusieurs types d'attaques qui peuvent nuire au travail des réseaux de capteurs sans fil (RCSF) et empêcher leur bon objectif de déploiement. La sécurité est donc une dimension importante pour ces réseaux.

Dans ce travail nous avons visé à sécuriser un réseau de capteurs par un moyen efficace et nous avons visé aussi à faire remonter l'information rapidement à un centre de contrôle distant. Pour atteindre cet objectif nous avons utilisé la cryptographie basée sur les courbes elliptiques car elle est considérée comme un moyen efficace et adaptable aux dispositifs présentant des ressources limitées.

En outre pour remonter l'information rapidement nous avons opté à des algorithmes qui permettent d'exécuter l'opération de chiffrement dans un temps raisonnable. En plus pour améliorer ce temps nous avons proposé une approche distribuée basée sur les courbes elliptiques.

Notre mémoire s'articule autour de quatre chapitres :

Le chapitre 1 : Est une introduction aux réseaux de capteurs sans fil, il présente aussi le fonctionnement général de ce type de réseau, ses applications potentielles et ses

principales caractéristiques.

Le chapitre 2 : Est une présentation des attaques connues dans ce type de réseau et des solutions envisageable.

Le chapitre 3 : Portera sur la définition des courbes elliptiques et une initiation au parallelisme.

Le chapitre 4 : Est une présentation des outils matériels et logiciels pour implémenter une application qui assurant la sécurité dans les réseaux de capteurs et réduisant le temps des calculs pour les opérations de chiffrement. Nous concluons ce mémoire par un bilan et des perspectives pour une amélioration future.

Chapitre 1

Concepts généraux sur les réseaux de capteurs

1.1 Introduction

Au cours des dernières décennies, nous avons assisté à une miniaturisation du matériel informatique. Cette tendance à la miniaturisation a apporté une nouvelle génération de réseaux informatiques et télécoms présentant des défis importants et produisant en masse des systèmes d'une taille extrêmement réduite et embarquant des unités de calcul et de communication sans fil pour un coût réduit. Les réseaux de capteurs sans fil sont l'une des technologies visant à résoudre les problèmes de cette nouvelle ère de l'informatique embarquée et omniprésente, ils sont capables de générer et d'échanger des données d'une manière autonome et complètement transparente pour les utilisateurs.

Les réseaux de capteurs représentent actuellement un nouveau domaine, en plein développement, émergeant des innovations des technologies de communication.

Dans ce chapitre nous allons introduire et faire une description synthétique des réseaux de capteurs sans fil en présentant leurs évolutions, architectures, caractéristiques, leurs domaines d'applications variés et le point sur le besoin de la sécurité dans les réseaux de capteurs[1].

1.2 Définition d'un capteur

Un capteur est un dispositif électronique ayant pour tâche de transformer une mesure physique environnementale observée en une mesure généralement électrique qui sera à son tour traduite en une donnée binaire exploitable et compréhensible par un système d'information et de la communiquer à un centre de contrôle via une station de base [2].

Parmi les différents types de mesures enregistrées par les capteurs, on peut citer entre autres : la température, l'humidité, la luminosité, l'accélération, la distance, les mouve-

ments, la position, la pression, la présence d'un gaz, la vision (capture d'image), le son, etc...

On pourrait vulgairement dire qu'un nœud capteur, comme on l'entend aujourd'hui, est un dispositif composé de plusieurs capteurs.

On fait souvent la confusion entre capteur et transducteur qui est un dispositif qui traduit une grandeur physique en une autre alors qu'un capteur au minimum il est constitué d'un transducteur [2].

1.3 Composant d'un nœud capteur

L'évolution de l'architecture des capteurs est un des facteurs qui a permis l'essor de solutions à base de réseaux de capteurs. En effet les capteurs des générations précédentes avaient une architecture qui se limitait au capteur proprement dit (dispositif capable de mesurer une grandeur physique) et une unité d'alimentation (batterie, piles, etc. . .).Le détecteur d'incendies est le parfait exemple de ce type de capteur, Celui-ci n'est composé que d'un système de capture alimenté par une pile, capable de détecter la fumée et de déclencher une alarme [2].

Les capteurs actuels ont su évoluer pour ajouter les fonctionnalités de traitement de l'information et de la communication de cette information . . .

La figure 1.3.1 est l'illustration la plus générale de l'architecture d'un capteur dit intelligent[3].

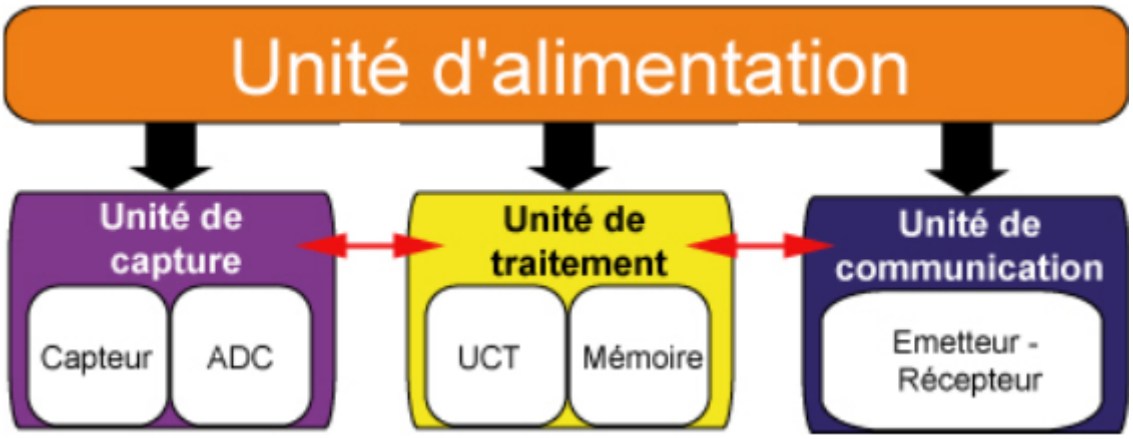


FIGURE 1.3.1: Architecture d'un nœud capteur sans fil

Cette architecture s'articule autour de quatre unités (figure 1.3.1) :

1. **L'unité d'acquisition de données**

   Elle permet la capture des données, c'est à dire la mesure des grandeurs physiques ou analogiques et leurs conversions en données numériques. Elle est composée du capteur lui-même et de convertisseurs analogique-numérique (ADCs). Le capteur est chargé de récupérer les signaux analogiques qu'il les transmet au ADC [1] qui les transforme à son tour en données numériques et les communiquer à l'unité de traitement [4].

2. **L'unité de traitement de données**

   C'est l'unité principale du capteur, généralement un processeur couplé à une mémoire vive et mémoire flash, Sur certains capteurs elle peut embarquer un système d'exploitation pour faire fonctionner le capteur. Elle peut aussi être couplée à une unité de stockage, qui servira par exemple à y enregistrer les informations transmises par l'unité d'acquisition de données [2].

3. **L'unité de communication de données (Transceiver)**

   Elle a pour fonction de transmettre et recevoir l'information. Elle est équipée d'un couple émetteur/récepteur, aussi appelé transceiver, terme anglophone issu de la contraction des mots transmitter et receiver. Elle permet la communication au sein du réseau, dans le cas qui nous intéresse par radiofréquence (ondes radios).

4. **l'unité d'alimentation**

   Elément principal de l'architecture du capteur, c'est elle qui fournit en énergie toutes les autres unités. Elle correspond le plus souvent à une batterie ou une pile alimentant le capteur, dont les ressources limitées en font une problématique propre à ce type de réseaux. La réalisation récente d'unité d'alimentation à base de panneaux solaires tente d'apporter une solution pour prolonger sa durée de vie.

   Par ailleurs, d'autres unités peuvent se greffer à cette architecture générale. Citons, entre autres, la possibilité d'ajouter une unité de localisation, tel qu'un GPS, un mobilisateur pour assurer la mobilité du capteur, ou une unité spécifique de capture comme une caméra pour l'acquisition vidéo.

   Dans le cas de l'utilisation d'un GPS ou d'une caméra de surveillance, il est intéressant de noter que leur utilisation dans les capteurs actuels a un coût non négligeable en termes de consommation énergétique, qui peut amener à une réduction drastique de la durée de vie d'un capteur équipé de tels dispositifs [4].

1. ADS : Convertisseur Analogique/Numérique

1.4 Type de capteurs

Il existe actuellement un grand nombre de capteurs, avec des fonctionnalités diverses et variées.

La plupart des capteurs dépendent de l'application pour lesquels ils ont été conçus (capteurs aquatiques, sous-terrain, etc. . .). Il est plus intéressant de décrire les capteurs les plus utilisés et leur évolution au cours du temps. La figure 1.4.1 illustre l'évolution des capteurs au cours de ces 20 dernières années. Cette représentation met en avant l'importance des travaux de recherche de l'université de Berkeley dans l'essor des réseaux de capteurs, surtout sachant que l'entreprise Xbow [2] (aussi appelé Crossbow) qui fait jusqu'à aujourd'hui office de référence dans la fabrication de capteurs est née au sein de la célèbre université californienne.

Les capteurs fabriqués par Xbow au cours des dix dernières années (famille de capteurs Mica et Telos) sont sans aucun doute les plus utilisés dans les expériences et travaux de recherche. Ces capteurs sont capables de mesurer plusieurs métriques et s'articulent pour la plupart d'entre eux autour du Chipcon CC2420 qui est devenu le standard au niveau des modules de transmission utilisant le protocole de communication IEEE 802.15.4. [4].

FIGURE 1.4.1: Evolution des capteurs

2. Xbow : entreprise de fabrication de capteurs appelés aussi CrossBow.

1.5 Les réseaux de capteurs

Un Réseau de Capteurs Sans Fil, Wireless Sensor Networks en anglais (WSN) est un système distribué de grande échelle mettant en communication un grand nombre de dispositifs très petits, autonomes, communément appelés "capteurs sans fil", ou simplement "capteurs".

Dans ces réseaux, chaque nœud est capable de surveiller son environnement et de réagir, en cas de besoins, en envoyant l'information collectée, à un ou plusieurs points de collecte, à l'aide d'une connexion sans fil.

Les capteurs sont des dispositifs de taille extrêmement réduite avec des ressources très limitées, autonomes, capables de traiter des informations et de les transmettre via les ondes radio (WiFi ou ZigBee [3] par exemple), à une autre entité (capteurs, unité de traitements...) sur une distance limitée à quelques mètres. Les nœuds ont la capacité de jouer le rôle de routeurs. Un capteur analyse son environnement et propage les données récoltées aux capteurs appartenant à sa zone de couverture.

Dans un scénario d'application classique, plusieurs nœuds capteurs sont déployés dans un certain environnement pour mesurer différents phénomènes physiques et faire remonter les informations collectées à une station de base distante, nommée aussi le nœud puits [5].

Dans le cas le plus simple, les capteurs communiquent directement avec la station de base. Cependant, dans le cas d'un réseau à grande échelle, les capteurs ne sont pas tous dans le voisinage du puits et les messages seront acheminés du nœud source vers le puits en transitant par plusieurs nœuds, selon un mode de communication multi-sauts comme l'illustre la Figure 1.5.1

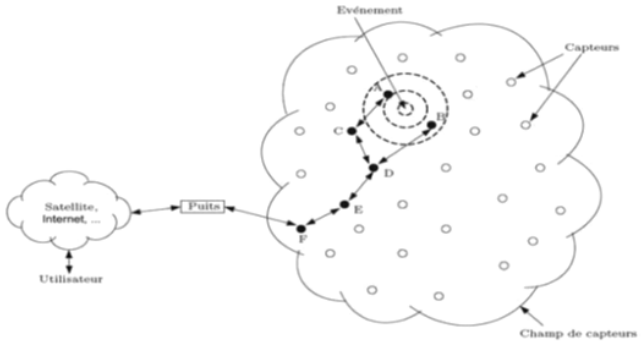


FIGURE 1.5.1: Schéma d'un réseau de capteurs sans fil

3. ZigBee : Un protocole de haut niveau permettant la communication de petites radios, à consommation réduite.

1.5.1 Architecture

Il existe deux types d'architectures pour les réseaux de capteurs : les réseaux de capteurs plats et les réseaux de capteurs hiérarchiques [1] :

Les réseaux de capteurs sans fil plats

Un réseau de capteurs sans fil plat est un réseau homogène, où tous les nœuds sont identiques en termes de batterie et des fonctions, excepté le « Sink » [6]. Ce dernier joue le rôle d'une passerelle et est responsable de la transmission de l'information collectée à l'utilisateur final comme le montre la figure 1.5.2

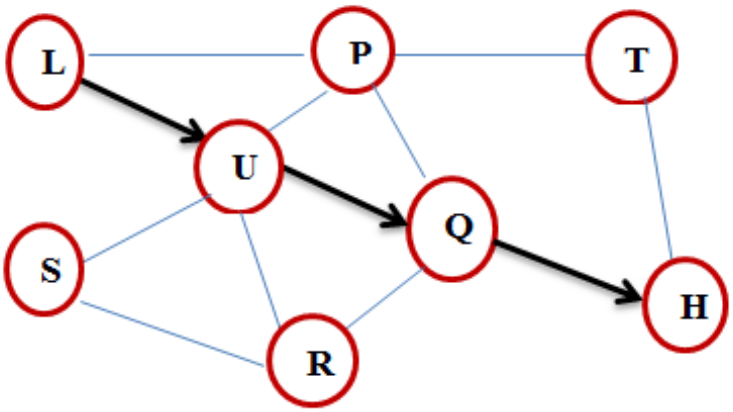


FIGURE 1.5.2: Routage Plat

Les réseaux de capteurs hiérarchiques

Une architecture hiérarchique a été proposée pour réduire le coût et la complexité de communications dans les RCSF. Elle consiste à introduire un ensemble de nœuds plus coûteux et plus puissants, en créant une infrastructure qui décharge la majorité des nœuds simples à faible coût de plusieurs fonctions du réseau. L'architecture hiérarchique est composée de plusieurs couches : une couche de capteur, une couche de transmission et une couche de point d'accès, la figure 1.5.3 le montre.

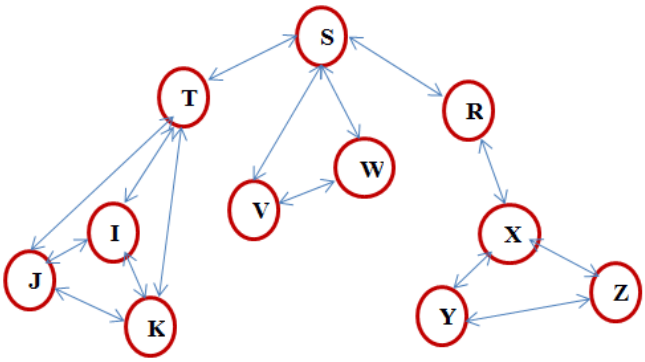


FIGURE 1.5.3: Routage hiérarchique

1.6 Les différents facteurs de conception

La conception des réseaux de capteurs est influencée par de nombreux facteurs comme la tolérance aux pannes, les coûts de production, la consommation d'énergie, l'environnement ou la topologie du réseau. Ces facteurs représentent la base de la conception de protocoles ou d'algorithmes pour les réseaux de capteurs.

1.6.1 Tolérance aux pannes

Les nœuds peuvent être sujets à des pannes dues à leur fabrication (ce sont des produits de série bon marché, il peut donc y avoir des capteurs défectueux) ou plus fréquemment à un manque d'énergie. Les interactions externes (chocs, interférences,. . .) peuvent aussi être la cause de leur dysfonctionnement. Afin que les pannes n'affectent pas la tâche première du réseau, Il faut mettre en place des mécanismes qui permettent d'augmenter le taux de disponibilité d'un réseau de capteurs [5].

1.6.2 Topologie du réseau

En raison de leur forte densité dans la zone à observer, il faut que les nœuds capteurs soient capables d'adapter leur fonctionnement afin de maintenir la topologie souhaitée[5].

On distingue généralement trois phases dans la mise en place et l'évolution d'un réseau :

- Déploiement : Les nœuds sont soit répartis de manière prédéfinie soit de manière aléatoire (largués en masse depuis un avion). Il faut alors que ceux-ci s'organisent de manière autonome.

- Post-Déploiement - Exploitation : Durant la phase d'exploitation, la topologie du réseau peut être soumise à des changements dus à des modifications de la position des

nœuds ou bien à des pannes.

- Redéploiement : L'ajout de nouveaux capteurs dans un réseau existant implique aussi une remise à jour de la topologie.

1.6.3 Consommation d'énergie

L'économie d'énergie est une des problématiques majeures dans les réseaux de capteurs. En effet, la recharge des sources d'énergie est souvent trop coûteuse et parfois impossible. Il faut donc que les capteurs économisent au maximum l'énergie afin de pouvoir fonctionner.

1.7 Caractéristiques des RCSF

Les principaux caractéristiques des réseaux de capteurs se résument dans ce qui suit :

1.7.1 Densité importante des nœuds

Les réseaux de capteurs se composent généralement d'un nombre très important des nœuds pour garantir une couverture totale de la zone surveillée. Ceci engendre un niveau de surveillance élevé et assure une transmission plus fiable des données sur l'état du champ de capteur.

1.7.2 Topologie dynamique

L'instabilité de la topologie des réseaux de capteurs est le résultat des trois facteurs essentiels suivants :

- La mobilité des nœuds : les nœuds capteurs peuvent être attachés à des objets mobiles qui se déplacent librement et arbitrairement, introduisant ainsi une topologie instable du réseau.
- La défaillance des nœuds : du fait de l'autonomie énergétique limitée des nœuds, la topologie du réseau n'est pas fixée (les nœuds qui épuisent leur énergie, sont considérés comme des nœuds inexistants).
- L'ajout de nouveaux nœuds : de nouveaux nœuds peuvent facilement être rajoutés. Il suffit de placer un nouveau capteur qui soit dans la portée de communication d'au moins un autre nœud capteur du réseau déjà existant.

1.7.3 Auto-organisation

L'auto organisation s'avère très nécessaire pour ce type de réseau afin de garantir sa maintenance. Vu les différentes conséquences résultant de l'instabilité de la topologie du

réseau de capteur, ce dernier devra être capable de s'auto-organiser pour continuer ses applications.

1.7.4 Scalabilité

Les réseaux de capteurs peuvent contenir des centaines voire des milliers de nœuds capteurs [7]. Un nombre aussi important engendre beaucoup de transmissions internodales et nécessite que le nœud « Sink » soit équipé d'une mémoire importante pour stocker les formations reçues.

1.8 Consommation d'énergie d'un nœud-capteur

1.8.1 Formes de dissipation d'énergie

Les nœuds capteurs sont alimentés principalement par des batteries. Ils doivent donc fonctionner avec un bilan énergétique frugal. En outre, ils doivent le plus souvent avoir une durée de vie de l'ordre de plusieurs mois, voire de quelques années, puisque le remplacement des batteries n'est pas une option envisageable pour des réseaux avec des milliers de nœuds.

Afin de concevoir des solutions efficaces en énergie, il est extrêmement important de faire d'abord une analyse des différents facteurs provoquant la dissipation de l'énergie d'un nœud-capteur [8].

Cette dissipation d'énergie se fait de manière générale selon plusieurs modes :

La radio : opère dans quatre modes de fonctionnement : émission, réception, « idle », et sommeil. Une observation importante dans le cas de la plupart des radios est que le mode « idle » induit une consommation d'énergie significative, presque égale à la consommation en mode réception. Ainsi, il est plus judicieux d'éteindre complètement la radio plutôt que de passer en mode « idle » quand l'on a ni à émettre ni à recevoir de données. Un autre facteur déterminant est que, le passage de la radio d'un mode à un autre engendre une dissipation d'énergie importante due à l'activité des circuits électroniques. Par exemple, quand la radio passe du mode sommeil au mode émission pour envoyer un paquet, une importante quantité d'énergie est consommée pour le démarrage de l'émetteur lui-même. Un autre point important est que les données des constructeurs sous-estiment assez régulièrement ces différentes consommations, en particulier concernant la consommation dans le mode « idle ».

Le MCU *« Unité du microcontrôleur ou Micro Controller Unit »* : Généralement les MCUs possèdent divers modes de fonctionnement : actif, « idle », et sommeil, à des fins de gestion d'énergie. Chaque mode est caractérisé par une quantité différente de consommation d'énergie. Par exemple, le MSP430 [4] de TelosB consomme 3mW en

4. MSP430 : Microcontrôleur injecté dans des capteurs de type TmoteSky.

mode actif, 98 µW dans le mode «idle» et seulement 15 µW dans le mode sommeil. Toutefois, la transition entre les modes de fonctionnement implique un surplus d'énergie et de latence. Ainsi, les niveaux de consommation d'énergie des différents modes, les coûts de transition entre les modes et encore le temps passé par le MCU dans chaque mode ont une incidence importante sur la consommation totale d'énergie d'un nœud-capteur [5].

Le détecteur ou le capteur proprement dit : il y a plusieurs sources de consommation d'énergie par le module de détection, notamment l'échantillonnage et la conversion des signaux physiques en signaux électriques, le conditionnement des signaux et la conversion analogique-numérique.

Étant donné la diversité des capteurs, il n'y a pas de valeurs typiques de l'énergie consommée.

En revanche, les capteurs passifs (température, sismiques, ...) consomment le plus souvent peu d'énergie par rapport aux autres composants du nœud-capteur. Notons, les capteurs actifs tels que les sonars, les capteurs d'images, etc. peuvent consommer beaucoup d'énergie.

En outre, il existe d'autres formes de dissipation d'énergie telles que les lectures et les écritures mémoire.

Un autre aspect non négligeable est le phénomène d'autodécharge de la batterie. En effet, cette dernière se décharge d'elle-même et perd de sa capacité au fil du temps.

Il est difficile d'apporter ici une étude quantitative et comparative précise de la consommation de chaque composant d'un nœud-capteur en raison du grand nombre de plates-formes commerciales existantes. Cependant, des expérimentations ont montré que c'est la transmission de données qui est la plus consommatrice en énergie. Le coût d'une transmission d'un bit d'information est approximativement le même que le coût nécessaire au calcul d'un millier d'opérations. La consommation du module de détection dépend du type spécifique du nœud-capteur.

1.8.2 Sources de surconsommation d'énergie

Nous appelons surconsommation d'énergie toute consommation inutile que l'on peut éviter afin de conserver l'énergie d'un nœud-capteur [9].

Les causes majeures de la perte d'énergie sont :

a. Les collisions

elles sont la première source de perte d'énergie. Quand deux trames sont émises en même temps et se heurtent, elles deviennent inexploitables et doivent être abandonnées. Ce que nécessite de les retransmettre de nouveau. Les protocoles MAC essayent à leur manière d'éviter les collisions. Les collisions concernent plutôt les protocoles MAC avec contention.

b. L'écoute à vide (idlelistening)

un nœud dans l'état « idle » est prêt à recevoir un paquet, mais il n'est pas actuellement en train de recevoir quoi que ce soit. Ceci est coûteux et inutile dans le cas des réseaux à faible charge de trafic. Plusieurs types de radios présentent un coût en énergie significatif pour le mode idle. Eteindre la radio est une solution, mais le coût de la transition entre les modes consomme également de l'énergie, la fréquence de cette transition doit alors rester raisonnable.

c. L'écoute abusive (overhearing)

cette situation se présente quand un nœud reçoit des paquets qui ne lui sont pas destinés. Le coût de l'écoute abusive peut être un facteur dominant de la perte d'énergie quand la charge de trafic est élevée et la densité des nœuds est grande, particulièrement dans les réseaux mostly-on.

d. L'overmitting

un nœud envoie des données et le nœud destinataire n'est pas prêt à les recevoir.

e. L'overhead des paquets de contrôle

l'envoi, la réception, et l'écoute des paquets de contrôle consomment de l'énergie. Comme les paquets de contrôle ne transportent pas directement des données, ils réduisent également le débit utile effectif.

1.8.3 Mécanismes de conservation de l'énergie

C'est la transmission de données qui se révèle extrêmement consommatrice par rapport aux tâches du nœud-capteur. Cette caractéristique conjuguée à l'objectif de maximisation de la durée de vie du réseau a suscité de nombreux travaux de recherche. Nous introduisons dans ce point certains mécanismes de base :

a. Mode d'économie d'énergie

Ce mode est possible quelle que soit la couche MAC adoptée. Cela consiste à éteindre le module de communication dès que possible. Par exemple, des protocoles MAC fondés sur la méthode TDMA offrent une solution implicite puisqu'un nœud n'échange des messages que dans les intervalles de temps qui lui sont attribués.

Il peut alors garder sa radio éteinte durant les autres slots. Il faut toutefois veiller à ce que le gain d'énergie obtenu en mettant en veille le module radio ne soit pas inférieur au surcoût engendré par le redémarrage de ce module.

Afin d'optimiser le gain d'énergie, dans le cas d'une présence d'un évènement pertinent ou même d'une agrégation de données on va mettre en place un nœud passif qui ne sera actif que lors d'un dépassement d'un certain seuil.

b. Traitement local

L'idée de cette technique est que la source peut se censurer. Ainsi une programmation événementielle semble bien adaptée aux réseaux de capteurs. Seuls les changements significatifs de l'environnement devrait provoquer un envoi de paquets le réseau.

Dans le même contexte, une grande collaboration est attendue entre les capteurs d'une même région en raison de leur forte densité et dans la mesure où les observations ne varient presque pas entre des voisins très proches. Ainsi les données pourront être confrontées localement et agrégées au sein d'un seul et unique message. Cette stratégie de traitement local permet de réduire sensiblement le trafic [5].

c. Organisation des échanges

Ce procédé revient à limiter les problèmes de retransmission dus aux collisions. La solution extrême consiste à utiliser la technique d'accès au medium TDMA. Les collisions sont ainsi fortement réduites. Cette solution présente l'inconvénient d'être peu flexible et de demander une synchronisation fine des capteurs. Des solutions intermédiaires ont vu le jour, par exemple S-MAC (Sensor MAC) [10] qui est une méthode d'accès au canal de type CSMA-CA avec le mécanisme RTS/CTS qui permet d'éviter les collisions et le problème de la station cachée. La principale innovation, apportée par ce protocole, est d'avoir un mécanisme de mise en veille distribué sur chaque nœud du réseau dans le but de réduire la consommation d'énergie. La principale difficulté de S-MAC est également de synchroniser les nœuds entre eux pour que la communication soit toujours possible.

d. Limitation des accusés de réception

L'acquittement systématique est mal adapté à des réseaux denses : il provoque une surcharge du réseau et donc des collisions et des interférences avec les données utiles échangées dans le réseau. Les acquittements par « piggy-backing » seront à privilégier.

1.8.4 Ordonnancement d'activité des capteurs

Le comportement global de la communication dans un réseau de capteurs sans fil est axé sur l'application.

Par conséquent, nous trouvons plusieurs types de réseaux en fonction du modèle de délivrance de données qu'impose l'application (continu, à la demande ou hybride) [11].

Ce modèle influe sur l'activité radio des nœuds, ce qui fait que plusieurs types de réseaux existent, en l'occurrence les réseaux « Mostly-off » et les réseaux « Mostly-on ».

Les réseaux de capteurs « mostly-off » se réveillent que pour envoyer leurs données à un collecteur.

Par antonymie, le « mostly-on » sont des réseaux de nœuds dont les radios sont la plupart du temps allumées.

1.9 Différentes problématiques dans les RCSF

Les recherches dans le domaine des réseaux de capteurs ont révélé plusieurs problématiques, parmi ces problématiques, nous citons [4] :

1.9.1 Routage

Concevoir un protocole de routage performant en termes de minimisation de la consommation de l'énergie, du choix des routes optimales pour l'acheminement de l'information d'un capteur à la station de base et vice versa, de réduction du délai de délivrance des paquets...Ainsi le réseau doit passer à l'échelle sans que ses performances se dégradent.

1.9.2 Couche MAC

La spécificité des réseaux de capteurs sans fil mobiles nécessite le développement de nouveaux protocoles MAC qui s'adaptent aux contraintes imposées par ces réseaux. Ceci est dans le but d'améliorer le débit, minimiser la consommation d'énergie, optimiser le partage du médium ainsi que minimiser le délai de délivrance des paquets.

1.9.3 Qualité de service

Des protocoles au niveau de la couche MAC devraient être capables d'établir des priorités entre les flux, limiter les pertes de paquets vitaux pour la gestion du réseau, ou du moins en restreindre l'impact.

1.9.4 Cross-layer

Dans les modèles classiques, les concepteurs essaient d'optimiser les performances au niveau d'une couche indépendamment des autres couches. Par exemple, le routage et les fonctions de la couche MAC sont optimisés indépendamment les uns des autres. Cependant, une telle indépendance est communément considérée comme trop onéreuse pour les réseaux de capteurs. Par conséquent, une coopération permettant un compromis entre performances, dépendance et flexibilité doit être proposée pour optimiser les capacités du réseau en général.

1.9.5 Diffusion de l'information

les protocoles de diffusion conçus pour les réseaux de capteurs doivent tenir compte de leurs spécificités ainsi que de leurs contraintes intrinsèques imposées. Ainsi, pour concevoir un protocole efficace, il faudrait assurer une couverture maximale des capteurs composant le réseau (taux d'accessibilité supérieur 90%), minimiser le nombre des réémetteurs et des réceptions redondantes ainsi que la consommation d'énergie.

1.9.6 Sécurité

pour les applications qui exigent un niveau de sécurité assez élevé telles que les applications militaires, des mécanismes d'authentification, de confidentialité, et d'intégrité doivent être mis en place au sein de leur communauté. Les algorithmes de cryptographie conçus pour les réseaux de capteurs doivent tenir compte des ressources limitées que présentent ces réseaux.

1.10 Domaines d'applications des RCSF

La miniaturisation, l'adaptabilité, le faible coût et la communication sans fil permettent aux réseaux de capteurs d'envahir plusieurs domaines d'applications. Ils permettent aussi d'étendre le domaine des applications existantes. Parmi ces domaines où ces réseaux se révèlent très utiles et peuvent offrir de meilleures contributions, on peut noter le militaire, la santé, l'environnemental, et les maisons intelligentes...

1.10.1 Applications militaires

Les nœuds capteurs devraient fournir les services suivants :
- Surveillance des champs de bataille.
- Reconnaissance des forces d'opposition.
- Repérage des cibles.
- Évaluation des dommages de la bataille.
- Détection et reconnaissance d'attaque nucléaire, biologique et chimique.

Toutefois il faut noter que l'utilisation des RCSF à des fins militaires requiert une sécurité supérieure à tout autre domaine. En effet elle peut avoir une incidence voir mettre en jeu des vies humaines si des informations stratégiques sont récupérées par un ennemi [7].

1.10.2 Applications de santé

Certaines applications de santé des RCSF sont :
- Fourniture d'interfaces pour les handicapés.
- Repérage et surveillance des médecins et des patients dans les hôpitaux.
- Télésurveillance des données physiologiques humaines.

Cependant ces applications se heurtent à un problème de taille, à savoir la sécurité des informations transitant sur le réseau.

D'un point de vue législatif, il est primordial que ces informations d'une part ne permettent pas d'authentifier le patient au regard du secret médical, d'autre part il faut pouvoir s'assurer que les données ne puissent être falsifiées. Ainsi une personne mal intentionnée pourraient envoyer de fausses informations sur une application sans protection, ce qui pourrait causer une mauvaise réaction du personnel médical et causer de graves troubles, voir la mort du patient [7].

1.10.3 Applications environnementales

Ces applications incluent :

- Le repérage des mouvements des oiseaux des petits animaux et des insectes.
- La surveillance des conditions environnementales qui affectent les récoltes et le bétail.
- L'exploration planétaire.
- Alertes des catastrophes (incendie, séisme...).
- La détection d'inondation.
- L'étude de pollution.

Le niveau de sécurité nécessaire dans des applications environnementales utilisant les RCSF peut paraître faible voir nul. Cependant ce serait oublier les risques de sabotage qui peuvent être le fait de personnes agissant dans un but gratuit (comme peut l'être la dégradation de bien publics) ou mercantile (les mesures environnementales peuvent impliquer des organisations et les mettre en porte-à-faux avec des conséquences économiques non négligeables) [7].

1.11 Conclusion

Dans ce chapitre nous avons procédé à l'étude des réseaux de capteurs sans fil.

Nous avons posé les briques de base et fédéré quelques concepts nécessaires à la compréhension de la problématiques dans la suite de ce mémoire.

La flexibilité, la tolérance aux fautes, le prix réduit et les caractéristiques rapides de déploiement des réseaux de capteurs offrent des possibilités infinies de développement dans tous les domaines d'application.

Ceci nous permet de penser que les réseaux de capteurs feront bientôt partie intégrante de nos vies et satisferont sûrement les plus grands projets.

Dans le prochain chapitre nous allons définir un enjeu majeur des technologies numériques modernes qui est la sécurité.

Chapitre 2

La sécurité dans les réseaux de capteurs

2.1 Introduction

Le degré d'utilisation des systèmes et réseaux d'informations et l'environnement des technologies de l'information dans son ensemble ont évoluée de façon spectaculaire depuis 1992. Ces évolutions offrent des avantages significatifs mais requirent également que le gouvernement, les entreprises, les autres organisations et les utilisateurs individuels qui développent, possèdent, fournissent, gèrent, maintiennent et utilisent les systèmes et réseaux d'informations, portent une bien plus grande attention à la sécurité [12].

La sécurité est un enjeu majeur des technologies numériques modernes. Infrastructure de télécommunication, réseau sans fils, Internet, systèmes d'informations, routeurs, systèmes d'exploitation, applications informatiques, toutes ces entités présentent des vulnérabilités : faille de sécurité, défaut de conception ou de configuration. Ces systèmes tombent en panne, subissent des erreurs d'utilisation et sont attaqués de l'extérieur ou par des pirates, des cybercriminels.

D'après le constat fait dans le chapitre 1 les RCSF nécessitent plus de sécurité.

Dans ce chapitre nous allons définir l'enjeu majeur des technologies numériques qui est la sécurité.

2.2 Objectifs de sécurité dans les RCSF

Comme évoqué précédemment les réseaux de capteurs sans fil nécessitent dans de nombreuses applications des solutions qui assurent la sécurité des informations circulant sur le réseau. La sécurité des informations transitant dans les RCSF doivent répondre à plusieurs prérequis [13] :

2.2.1 Disponibilité du réseau

Le réseau doit pouvoir être disponible à tout instant, c'est-à- dire que l'envoi d'information ne doit pas être interrompu, de même que la circulation de l'information ne doit pas être stoppée. Dans le cas d'un réseau de capteurs réactif, il faut qu'un capteur qui détecte un événement puisse transmettre à tout instant cette information vers la station de base.

2.2.2 Authentification

L'authentification des capteurs est nécessaire pour s'assurer que l'identité déclarée par un capteur est bien celle du capteur déclarant. En l'absence d'un mécanisme permettant d'authentifier clairement un nœud du réseau, de nombreuses attaques peuvent se mettre en place.

2.2.3 Intégrité des données

Les données circulant sur le réseau ne doivent pas pouvoir être altérées au cours de la communication. Il faut donc s'assurer que personne ne puisse capturer et modifier les données du réseau. De la même manière, il faut vérifier que les données n'ont pas subi d'altération due à un disfonctionnement du matériel, qui est un risque important sur des capteurs sensibles aux altérations d'état.

2.2.4 Confidentialité des données

le réseau doit s'assurer que les données transmises soient confidentielles et ne puissent être lues par des dispositifs ou personnes autres que ceux ayant droit de le faire. Les données doivent être cachées ou cryptées de telle manière que personne ne puissent y accéder. La confidentialité des données est prépondérante dans des applications de type médicales où les informations du patient ne doivent pas être divulguées. Il en est de même pour des applications militaires où ces informations peuvent avoir une conséquence stratégique sur des actions en cours.

2.2.5 Fraîcheur des données

la fraîcheur des données permet de savoir si la donnée est récente ou non. Cela signifie qu'il faut s'assurer que la donnée transmise corresponde à un état présent. La fraîcheur des données garantit ainsi que ces données ne reflètent pas un état passé qui n'a plus cours. Sans mécanisme de sécurité vérifiant que les données transmises sont récentes ou non, un attaquant pourrait capturer des informations circulant sur le réseau à une date T, puis les retransmettre à une date T+1 pour tromper le réseau et faire circuler de fausses informations. On peut prendre pour exemple un réseau de capteurs censé de

détecter les incendies, L'attaquant enregistrerait les informations envoyées à l'occurrence de cet événement. Il pourrait alors plus tard renvoyer ces mêmes données pour déclencher une fausse alerte.

2.2.6 Auto-organisation

Les capteurs du réseau doivent être capables, après avoir été déployés, de s'auto-organiser et surtout de se sécuriser eux-mêmes, sans autres interventions extérieures. Ce besoin d'auto-organisation se retrouve dans l'établissement automatique de la distribution des clés de cryptages entre les nœuds du réseau et la gestion de ses clés ou bien encore dans le développement des relations de confiance entre capteurs du réseau. Pour cela les capteurs doivent avoir été munis au préalable des outils qui leur permettent de telles fonctionnalités.

2.2.7 Localisation sécurisée

Le besoin de se localiser et de connaître la position des autres nœuds peut être primordial dans de nombreux cas pour déjouer d'éventuelles attaques jouant sur les distances.

2.2.8 Temps synchronisé

De nombreuses solutions de sécurité nécessitent des capteurs synchronisés pour qu'elles soient effectives. Il faut ainsi s'assurer que les capteurs du réseau ont une horloge commune afin par exemple d'éviter des attaques de type rejeu de paquets.

2.3 Taxonomie des attaques dans les RCSF

Dans cette section, nous décrivons une liste non exhaustive mais représentative des attaques les plus courantes et connues, actives ou passives, que nous pouvons trouver dans les RCSF [2].

a. Ecoute passive : Consiste à écouter sans modifier les données ou le fonctionnement du réseau. Elle est généralement indétectable mais une prévention est possible.

b. Analyse du trafic : L'attaquant analyse les chemins empruntés par les paquets sur le réseau pour récupérer des informations précieuses sur les vulnérabilités de ce réseau. L'analyse du trafic peut permettre à un attaquant de connaître la position des stations de base ou des nœuds d'agrégation de données en repérant les lieux où le plus grand nombre de paquets transitent.

c. Brouillage radio : Un attaquant utilise les mêmes fréquences qu'un RCSF ce qui conduit à empêcher les nœuds de communiquer car le médium sera saturé par le brouillage radio.

d. Flooding : Dans ce type d'attaque, un attaquant utilise un ou plusieurs nœuds malicieux ou un dispositif particulier avec dans certains cas une puissance d'émission forte, pour envoyer régulièrement des messages sur le réseau afin de le saturer. On est en présence d'une attaque active qui est de même type que les attaques de type déni de service dans les réseaux classiques.

e. Hello Flooding : Les protocoles de découverte dans les réseaux ad-hoc utilisent des messages de type "HELLO" pour découvrir ses nœuds voisins et pour s'insérer dans un réseau. Dans une attaque dite de HELLO Flooding, un attaquant utilise ce mécanisme pour consommer l'énergie des capteurs et empêcher leurs messages d'être routés, comme l'illustre la figure 2.3.1 ci-dessous.

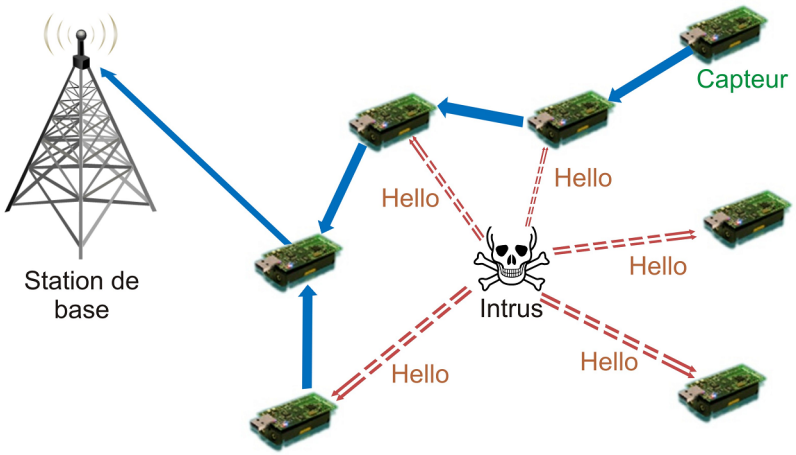


FIGURE 2.3.1: Hello Flooding

f. Injection de messages : L'attaquant cherche par divers moyens (utilisation de nœuds malicieux, envoi de paquets sur la même fréquence radio, etc. . .) à injecter des messages dans le réseau. Cette attaque permet de saturer, perturber le réseau, ou le tromper en envoyant de fausses informations.

g. Réplication des données : Si les paquets envoyés dans le réseau peuvent être lus et enregistrés par un attaquant. Ce dernier peut renvoyer ces mêmes paquets à une date ultérieure pour tromper le réseau. Pour illustrer cette attaque, on peut prendre pour exemple un réseau de capteurs qui a pour mission de détecter un incendie : si un premier incendie est détecté et qu'un capteur envoie un paquet d'alerte pour en informer le centre de contrôle, l'attaquant pourra enregistrer ce paquet, même s'il est chiffré et qu'il ne peut le déchiffrer, puis l'émettre à une autre date postérieure et faire croire qu'à un nouvel incendie s'est produit [2].

Cette attaque est réalisable si le paquet ne contient pas d'informations concernant la date de l'envoi ou si cette date est accessible et facilement modifiable par un attaquant.

j. Nœud compromis : La plupart des RCSF sont déployés dans des zones hostiles qui ne permettent pas une surveillance humaine de l'ensemble des capteurs. Il est alors

tout à fait possible pour un attaquant de compromettre un capteur. Cette attaque physique peut permettre à un attaquant d'extraire par exemple les clés cryptographiques contenues dans le capteur, remplacer le programme que contient par un autre, afin que le capteur devienne un nœud compromis. Ce nœud contrôlé par l'attaquant va lui permettre de s'intégrer dans le réseau pour récupérer des informations ou de lancer d'autres attaques à partir de ce nœud [2].

k. Attaque du trou noir *(black holeattack)* : L'attaque du trou noir consiste tout d'abord à insérer un nœud malicieux par divers moyens dans le réseau. Ce nœud va modifier les tables de routage pour obliger le maximum de nœuds voisins à faire transiter leurs informations par lui. Ensuite, toutes les informations qui passent par ce nœud ne seront jamais retransmises, comme le montre la figure 2.3.2

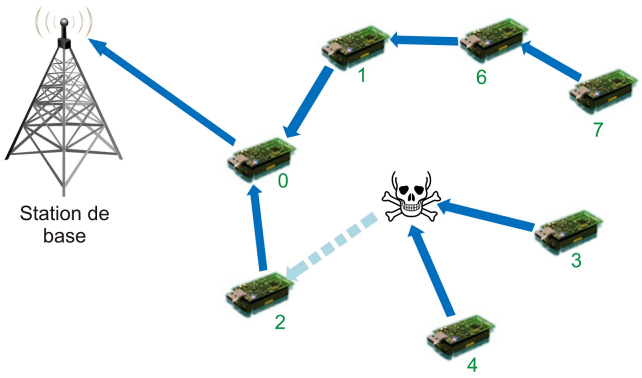


FIGURE 2.3.2: Attaque du trou noir

l. Attaque du trou gris : L'attaque du trou gris est une variante plus subtile de l'attaque du trou noir. Tout comme le trou noir, il s'agit d'un nœud malicieux qui va être inséré dans le réseau et qui va modifier les tables de routage pour faire transiter un maximum d'informations par lui. Contrairement au trou noir, le trou gris relaye certaines informations. Par exemple, le trou gris peut relayer toutes les informations concernant le routage, mais ne le fera pas pour des informations critiques. Ce type d'attaque est ainsi plus difficile à détecter que l'attaque du trou noir, car le capteur malicieux tant qu'il se comporte de manière normale ne peut être facilement détecté.

m. Attaque du trou de ver *(worm hole attack)* : L'attaque du trou de ver nécessite l'insertion dans le réseau de capteurs d'au moins deux nœuds malicieux. Ces deux nœuds sont reliés entre eux par une connexion puissante qui peut être filaire ou radio, ce qui permet de tromper les autres nœuds sur les distances les séparant [2].

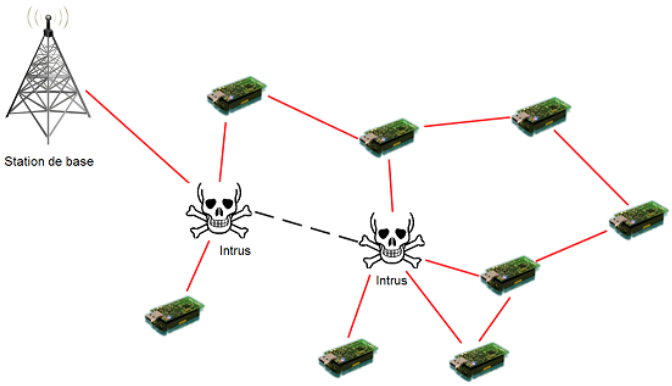


FIGURE 2.3.3: Attaque du trou de ver

n. Attaque sybile : Une attaque de type "Sybil attack" consiste à ce qu'un capteur malicieux se fasse passer pour plusieurs capteurs. Il va ainsi pouvoir modifier la table de routage qui deviendra caduque. Un nœud malicieux qui peut se faire passer pour plusieurs nœuds peut gagner un avantage important pour une élection de nœud maître par exemple [2].

o. Altération des messages : Un nœud malicieux va récupérer un message et l'altérer, en lui ajoutant des fausses informations (sur le destinataire, l'émetteur, l'information en elle-même) ou en le modifiant.

p. Attaque spécifique au type de capteur : Dans cette attaque, un attaquant modifie de manière physique le comportement du capteur. Il peut par exemple allumer une flamme devant un capteur thermique ou bien allumer une lampe devant un capteur de luminosité pour tromper un capteur. A cet effet, ce dernier va remonter l'information continuellement jusqu'à l'épuisement son énergie ou enregistrer de fausses informations sur le réseau.

q. Attaque sur les Pacemakers : L'attaque sur les pacemakers est un parfait exemple d'attaques spécifique qui pourra y avoir une conséquence dramatique sur la santé du patient. Ceci montre la dangerosité d'une absence de sécurité pour des capteurs de type médical. Dans [14], les auteurs ont montré qu'il est facile de modifier à distance les réglages du pacemaker d'un patient avec un matériel à moindre frais.

2.4 Mécanismes de sécurité

Plusieurs mécanismes, sont mis en place afin de répondre à la question de la sécurité dans les RCSF. Nous présentons dans ce qui suit quelques solutions de sécurité les plus répandues et les protocoles de sécurité sous-jacents aux RCSF.

2.4.1 Le partitionnement des données

Propose une solution pour empêcher la récupération d'information dans les RCSF en partitionnant des données.

Si un capteur cherche à envoyer une information, celui-ci va la découper en plusieurs paquets de taille fixe. Ces paquets seront envoyés sur des chemins différents à la station de base qui les rassemble pour pouvoir reproduire l'information. Ce mécanisme oblige un attaquant à récupérer l'ensemble des paquets s'il veut reproduire l'information. Il doit aussi être capable d'écouter l'ensemble du réseau, pour récupérer les différents paquets qui circulent sur des chemins différents. Cependant cette solution augmente considérablement la consommation d'énergie car elle implique un grand nombre de nœuds pour acheminer un paquet à la station de base [15].

Un exemple de cette solution est représentée par la figure 2.4.1, où un capteur A divise un message en trois paquets qui vont suivre trois chemins différents.

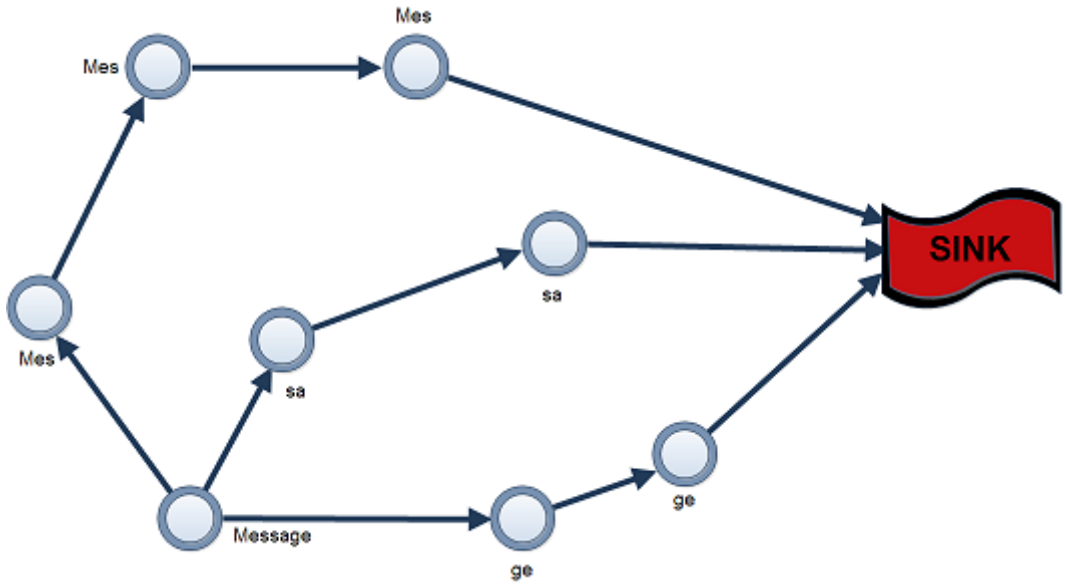


FIGURE 2.4.1: Exemple de partitionnement de données

2.4.2 Agrégation des données

Il a été montré dans plusieurs publications scientifiques que la transmission d'un bit est équivalente, en termes d'énergie, à l'exécution d'environ 1000 instructions. Cette valeur augmente avec la portée de la radio. Plus le capteur qui devra transmettre, est loin, et par conséquent il devra augmenter sa puissance d'émission pour atteindre la station de base, plus il consomme plus de l'énergie, ce qui affecte sa durée de vie. Il convient donc d'agréger les données avant les acheminer à la station de base.

Les techniques d'agrégation des données, permettent de réduire le nombre de messages redondants et par conséquent réduire la consommation en énergie. Par exemple, si un réseau est déployé pour mesurer la température et que le puits n'est intéressé que par la moyenne des températures, un nœud intermédiaire pourra additionner les valeurs reçues de ses membres et envoyer le résultat au nœud relais dans la direction de la station de base. Le puits recevra alors qu'un seul message, contenant la somme des données au lieu de n messages (ou n est le nombre de capteurs).

Ces techniques d'agrégation sont souvent utilisées. Elles sont cependant difficiles à mettre en œuvre lorsque les données sont chiffrées car le traitement des données devient alors très délicat.

2.4.3 La cryptographie

La cryptographie est sans doute la technique la plus utilisée dans la plupart des mécanismes de sécurisation actuelle.

Les outils cryptographiques

Le chiffrement : Le chiffrement des données permet d'empêcher l'écoute des données transitant dans un réseau sans fil et de garantir la confidentialité des données. Pour cela, il utilise des clés. On distingue de classes de chiffrement : symétrique ou asymétrique.

Le chiffrement symétrique

Le principe de chiffrement symétrique se base sur le partage d'une même clé K de chiffrement entre deux entités, communément nommé Alice et Bob pour chiffrer et déchiffrer les données en utilisant un algorithme de chiffrement symétrique, un exemple est illustré dans la figure 2.4.2

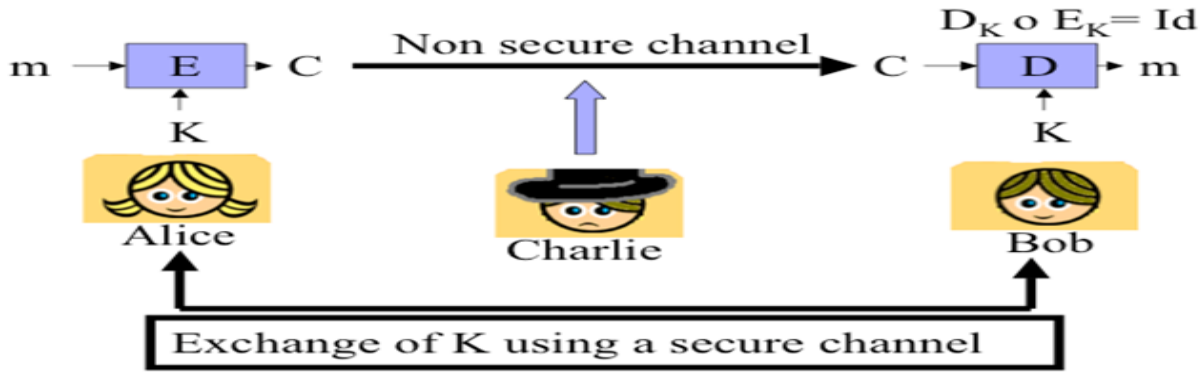


FIGURE 2.4.2: Cryptographie symétrique

Les algorithmes de chiffrement symétriques sont décomposés en deux catégories :

– Chiffrement par flux

Dans cette catégorie, le chiffrement se fait bit par bit sans attendre la réception entière des données. Par exemple, RC4 *(RivestCipher 4)* utilise cette technique.

Dans cette classe, la technique utilisée consiste à chiffrer le message à transmettre en effectuant un XOR avec la clé de chiffrement.

Soit M le message à chiffrer, K la clé de chiffrement, et $\bigoplus$ l'opération booléenne XOR, le chiffrement correspondant est :

$$M \bigoplus K = Mk \text{ où } Mk \text{ est le message chiffré.}$$

Le déchiffrement se fera alors par :

$$Mk \bigoplus K = M \bigoplus K \bigoplus K = M$$

– Chiffrement par blocs

Le chiffrement par blocs consiste à fractionner un message M en blocs de n bits. Ces blocs seront ensuite chiffrés par un algorithme de chiffrement F et une clé k extraite d'une clé maître K.

Soient M le message à chiffrer et K la clé de chiffrement dont sont extraites les clés k_i et F la fonction de chiffrement. M sera découpé en r blocs de n bits. Pour chaque bloc b_x de M, le chiffrement se fera de la manière suivante :

$$C_1 = F(k_1, b_x)$$

F est ensuite itérée avec une nouvelle clé extraite de la clé maître K pour garantir la sécurité de l'algorithme de chiffrement, ainsi :

$$C_2 = F(k_2; C_1)$$

$$C_y = F(k_y; C_{y-1})$$

Le déchiffrement se fait avec une fonction G, inverse de la fonction F et les différentes clé k_i partagées extraites de la clé commune K, de la manière suivante :

$$C_{y-1} = G(k_y, C_y) = G(k_y, F(k_y, C_{y-1}))$$

$$b_x = G(k_1, C_1)$$

Les algorithmes les plus utilisés sont : DES, AES.

Le chiffrement asymétrique [16]

Deux clés différentes sont générées par le récepteur : une clé publique diffusée à tous les nœuds servant au chiffrement de données qu'ils vont émettre au récepteur, et, une clé privée maintenue secrète chez le récepteur servant pour le déchiffrement de ces données lorsque ce dernier les reçoit. Le point fondamental sur lequel repose la sécurité du chiffrement asymétrique est l'impossibilité de déduire la clé privé à partir de la clé publique.

Formellement le chiffrement et le déchiffrement de données d'un message entre deux nœuds Alice et Bob correspondent au mécanisme suivant : soit K_p la clé publique et K_s la clé privée d'Alice, F la fonction de chiffrement et G la fonction de déchiffrement. Alice diffuse sa clé publique dans le réseau. Soit M le message que souhaite transmettre Bob à Alice, alors

$$Mk = F(K_p, M) \text{ où } Mk \text{ est le message chiffré,}$$

$$M \neq G(K_p, Mk)$$

Bob envoie le message à Alice qui va ensuite pouvoir le déchiffrer avec sa clé privée :

$$M = G(K_s, Mk)$$

Par ailleurs Bob peut demander à Alice de prouver son identité avec le mécanisme de la signature numérique. Pour cela Alice cryptera un message avec sa clé privée, Bob déchiffrera alors le message d'Alice avec la clé publique d'Alice. Comme seul Alice possède la clé privée correspondant à sa clé publique, si le message est déchiffrable avec sa clé publique, Bob est normalement assuré que le message provient effectivement d'Alice.

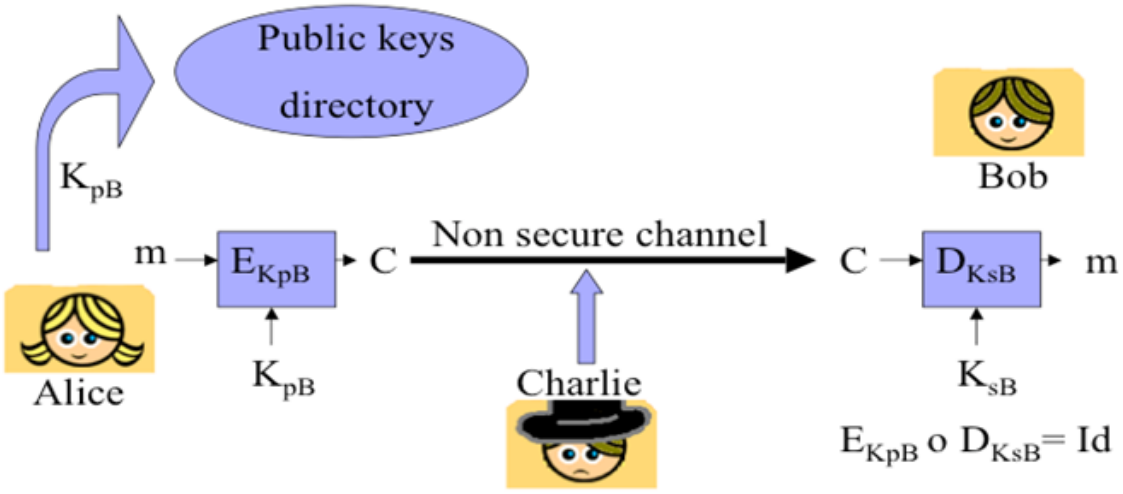


FIGURE 2.4.3: Cryptographie à clé publique

Les mécanismes de cryptographie publique sont potentiellement de bonnes solutions pour sécuriser les réseaux de capteurs en fournissant d'une part la confidentialité des données et d'autre part le mécanisme d'authentification qui leur fait défaut. Cependant le besoin de puissance CPU pour l'exécution d'algorithmes de chiffrement fait défaut.

Le tableau 2.1 montre les temps d'exécution nécessaires à l'exécution du protocole sécurisé d'authentification SSL/TLS utilisant l'algorithme de cryptographie RSA sur différents types de capteurs les plus répandus [17].

Type de capteur	RSA -1024
MICA2DOT	22.00 s
MICAZ	12.00 s
TelosB	5.70 s

TABLE 2.1: Temps d'exécution sur capteurs du protocole d'authentification SSL/TLS

A la lumière de ce tableau, on peut voir très clairement que les temps d'exécution de quelques unes des fonctions nécessaires à l'application de la cryptographie publique dans les réseaux de capteurs posent de gros problèmes de latence et de consommation énergétique.

Dans cette optique, des travaux de recherche essaient d'optimiser les algorithmes de chiffrement à clé publique comme présenté dans [16] où les auteurs proposent une version optimisée de RSA, appelée WM-RSA pour capteurs de type MicaZ. Bien que les résultats soient meilleurs que ceux de la version originale de RSA, les temps d'exécution restent toujours de l'ordre de la seconde. Un temps qui n'est pas envisageable dans des réseaux de capteurs qui nécessitent une intervention rapide. De plus la taille des clés nécessaires avec l'algorithme RSA à savoir 1024 bits peut poser des problèmes de stockage. Dans cette direction, des travaux récents tentent d'apporter une réponse avec l'utilisation de cryptographie à clé publique basée sur les courbes elliptiques (ECC) qu'on détaillera dans le chapitre suivant.

La cryptographie utilisant les courbes elliptiques nécessitent des tailles de clés inférieures à RSA. Par exemple, le niveau de sécurité d'une clé de 160 bits utilisée avec des algorithmes de chiffrement à courbes elliptiques correspond à un chiffrement avec une clé de 2046 bits avec l'algorithme de chiffrement RSA. Dans [19], les auteurs ont proposé la librairie cryptographique TinyECC pour le système d'exploitation TinyOS, qui offre la possibilité de chiffrer et d'authentifier des données avec des algorithmes à base de courbes elliptiques (ECDSA, ECIES, ECDH). Le temps d'exécution de cette solution dépasse la minute à quelques millisecondes selon le type d'architecture utilisé.

Cette solution ne peut donc pas être envisagée sur tout type de capteur. De plus la librairie TinyECC a été écrite pour la version 1.x de TinyOS et n'est plus compatible avec la version 2.x de TinyOS.

2.5 Comment faire face aux attaques

2.5.1 Génération de clés

Une solution proposée dans [19] consiste à utiliser une clé de génération. A chaque période ou génération, la station de base envoie une nouvelle clé à l'ensemble du réseau. Cette clé sert de certificat à chacun des nœuds, pour prouver son appartenance au réseau. Si un nœud non identifié tente de rentrer dans le réseau et qu'il ne possède pas cette clé de génération, il ne pourra être accepté en son sein.

Cette technique permet aussi de limiter les attaques de substitution d'un capteur et de sa reprogrammation pour être réinjecté dans le réseau. Si ce nœud est subtilisé à l'instant 0 avec la clé de génération K_0, le temps qu'un attaquant le reprogramme pour le remettre dans le réseau il se sera écoulé un temps "x". Quand le capteur sera repositionné dans le réseau, la nouvelle clé de génération sera alors K_x. Le nœud malicieux demandera à ses nœuds voisins de rentrer dans le réseau avec la clé K_0 et non pas K_x, car il n'a pas pu recevoir la nouvelle clé. Comme $K_0! = K_x$, les nœuds voisins n'accepterons pas sa requête et le nœud malicieux ne pourra pas rentrer dans le réseau.

2.5.2 Localisation

Un mécanisme utilisé pour détecter les nœuds malicieux et particulièrement des attaques de type trou de ver, consiste à utiliser une technique de localisation géographique. Pour cette solution, le RCSF doit être équipé de capteurs balises, qui sont des capteurs qui connaissent leur position géographique, par exemple au moyen d'un équipement GPS.

Avec la localisation, si un capteur demande à entrer dans le réseau, les capteurs balises qui vont recevoir cette demande vont pouvoir estimer sa localisation par rapport à son domaine d'écoute. Les capteurs balises vont ensuite quadriller leur zone d'écoute respective, et chaque nœud qui a reçu la demande d'insertion dans le réseau va voter pour une zone du quadrillage qu'il est capable d'entendre. La zone qui obtiendra le plus grand nombre de voix sera considérée comme la zone où est censé se trouver le nouveau capteur.

2.5.3 L'indice de confiance et la réputation

Des chercheurs ont proposé une solution qui consiste à utiliser les mécanismes de confiance et de réputation que l'on peut trouver dans les réseaux de communauté ou

bien encore dans les sites commerciaux comme ebay, les réseaux pair à pair. Dans ce type de réseau tout comme dans les RCSF, il est difficile de savoir, quel nœud peut être un nœud malicieux. Pour le détecter et conserver l'intégrité du réseau, chaque nœud va surveiller ses nœuds voisins et leurs actions au cours du temps. En fonction des actions réalisées par ses nœuds voisins, un nœud va augmenter une note de l'indice de confiance de ces nœuds, basée sur sa réputation. Si un nœud ne répond jamais à une requête, son indice de confiance va diminuer, de la même manière que si ce nœud retransmet toujours correctement l'information qu'on lui a demandé de transmettre, son indice de confiance va augmenter.

A l'aide de ces indices de confiance, un nœud va alors choisir le routage le plus adapté pour transmettre son information. Contrairement à des protocoles classiques de routage où le nœud chercherait le chemin le plus rapide en nombre de sauts ou de distance géographique, il va choisir ici de transmettre son information via les nœuds avec les indices de confiance les plus élevés, la route qui lui semble la plus sûre. Ces techniques permettent d'éliminer du routage traditionnel les nœuds qui sont potentiellement dangereux, et empêcher ainsi l'information de passer par ces nœuds.

Cette solution peut être jumelée avec un mécanisme de surveillance entre voisins proches nommé mécanisme du chien de garde *(watchdog)*, où pour chaque communication entre deux nœuds A et B, un nœud intermédiaire C, situé dans la zone de communication, est chargé de surveiller que cette communication a bien été effectuée, comme représentée dans la figure 2.5.3

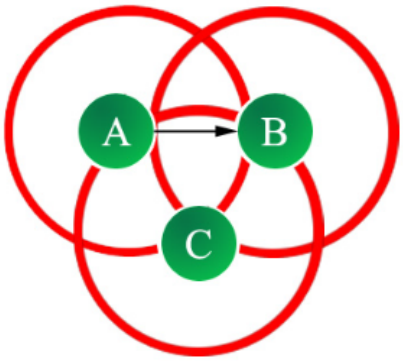


FIGURE 2.5.1: Exemple de chien de garde

Les solutions basées sur l'indice de confiance sont peu coûteuses en termes d'énergie et permettent, selon le type de sécurité voulu, de ne pas avoir recours à la cryptographie. Cependant pour des réseaux qui demandent une sécurité maximale, elles ne sont pas toujours adaptées. Ainsi un nœud malicieux qui enregistrerait des informations sur le réseau et, par ailleurs, se comporterait de manière normale, est difficilement détectable.

2.6 Protocoles de sécurité

Plusieurs protocoles ont été proposés pour sécuriser les RCSF. Certains d'entre eux visent seulement à détecter les attaques de type trou noir ou trou de ver. Toutefois, les protocoles de sécurité SPINS et TinySEC sont eux considérés comme des solutions de sécurité générales et apportent des solutions pour la confidentialité des données et leur authentification. Nous décrivons dans cette section les mécanismes utilisés par ces deux protocoles pour sécuriser les réseaux de capteurs.

2.6.1 SPINS (Security Protocols for Sensor Networks)

SPINS est un protocole basé sur deux blocs de sécurité que sont SNEP et µTESLA.

1. **SNEP**

   SNEP utilise deux mécanismes de sécurité, le premier consiste à chiffrer les données pour assurer leur confidentialité et le second de calculer un code MAC(Message Authentication Code) pour assurer l'authentification et l'intégrité des données entre deux entités.

   Dans SNEP durant un premier échange de données entre deux nœuds, le nœud émetteur précède le message d'une chaîne de bits aléatoires, appelée vecteur initial, avant de l'encrypter avec une fonction de chiffrement de type DES-CBC. Puis le message chiffré sera ajouté au bloc suivant et ainsi de suite. Cette technique empêche un attaquant qui écoute le réseau et qui a en sa possession le même message chiffré précédemment, de pouvoir en déduire que le même message a été envoyé[2].

   Les deux nœuds partagent ensuite un compteur qui leur permet d'utiliser des chiffrements par bloc en mode compteur (CTR) et de ne plus utiliser de vecteur initial. A chaque bloc échangé, le compteur est incrémenté. Or un attaquant ne peut décrypter l'information que s'il peut voir le même message plusieurs fois encryptés. L'utilisation d'un vecteur initial aléatoire et d'un compteur empêche cette possibilité, puisque un même message sera suivi en clair soit d'une chaîne de bits, soit d'un compteur incrémenté qui une fois chiffrée sera à chaque fois différent. L'utilisation de ce compteur permet d'éviter les attaques par rejeu de paquet car chaque message est numéroté, et donc, garantit la fraicheur des données

2. **µTESLA**

   µTESLA est une version adaptée aux RCSF. Elle permet l'utilisation du broadcast authentifié.

   µTESLA utilise une authentification symétrique liée à une méthode asymétrique où les clés symétriques sont divulguées au cours du temps. Pour permettre cette authentification, il est nécessaire que la station de base et les différents nœuds soient vaguement synchronisés. La station de base a pour rôle d'ajouter au paquet

à envoyer un code MAC calculé à partir d'une clé qui reste secrète à cet instant. Un nœud qui reçoit ce paquet peut vérifier que la clé pour déchiffrer le code MAC n'a pas encore été divulguée grâce à son horloge de synchronisation. Si elle n'a pas été encore divulguée, il peut en déduire que seule la station de base connaît la clé MAC et qu'aucun attaquant n'a pu altérer le message pendant son transit. Il peut alors stocker le paquet dans son buffer en attendant la prochaine divulgation de la clé. Quand la clé sera divulguée il pourra décrypter le message et vérifier son authenticité. Chaque clé K est une clé issue d'une chaîne de clé générée par une fonction à sens unique F, de telle manière que :

$$K_i = F(K_i + 1)$$

Cette clé de chiffrement pour code MAC est générée à des intervalles de temps réguliers de telle manière que si un capteur ne reçoit pas tous les paquets et donc toutes les clés, il sera capable de retrouver les anciennes clés en fonction de la dernière reçue. Ainsi, si un nœud du réseau possède la clé initiale K_0 et la clé K_2, mais n'a pas reçu la clé K_1, d'une part il peut vérifier que la clé K_2 est bien celle envoyée par la station de base car :

$$K_0 = F(F(K_2))$$

Et d'autre part il peut retrouver K_1 car :

$$K_1 = F(K_2)$$

Critiques de SPINS

Si SPINS a été l'un des premiers protocoles qui a proposé une solution garantissant la confidentialité et l'authenticité des données. Son approche n'est pas sans faille parce qu'il utilise un algorithme de chiffrement DES, qui n'est plus réputé sûr. En outre, µTesla nécessite un envoi de données permanent aux capteurs qui a un coût non négligeable pour la durée de vie du réseau. En plus, il est à noter que ce protocole a été défini mais jamais implémenté, et donc nous ne pouvons pas savoir son efficacité en termes de consommation d'énergie et latence sur des cas réels.

2.6.2 TinySec

TinySEC est une librairie de sécurité intégrée dans le système d'exploitation TinyOS-1.x. L'objectif de cette librairie est de pouvoir détecter les paquets non autorisés lorsqu'ils sont injectés pour la première fois dans le réseau et éviter leur propagation dans le réseau qui amènerait par les communications engendrées, à une perte d'énergie. Pour cette raison, TinySEC met en place des mécanismes d'authentification basés sur le code MAC, de chiffrement des informations et une protection contre les redondances d'informations.

Pour permettre une plus grande liberté d'actions, TinySEC supporte deux options de sécurité différentes :

- **TinySEC-Auth :** La sécurité concerne seulement l'authentification des données. Les données ne sont pas chiffrées, contrairement au code MAC qui est calculé à partir de l'entête du paquet pour assurer l'authenticité de l'expéditeur.
- **TinySEC-AE :** La sécurité porte à la fois sur l'authentification et l'encryptage des données. Les données sont chiffrées et envoyées avec un code MAC généré à partir des données chiffrées et de l'entête du paquet.

Pour l'authentification et l'encryptage des données TinySEC utilise un chiffrement par blocs de type CBC-MAC. De la même manière que pour SNEP, TinySEC utilise un vecteur initial pour le chiffrement du premier bloc et des chaînes de bits aléatoires ajoutés au message pour empêcher un attaquant d'analyser le trafic par comparaison des paquets. Cependant TinySEC n'utilise pas de compteur pour chaque chiffrement, ce qui empêche de garantir la fraîcheur des données et laisse possible les attaques de type rejeu de paquets. Il est aussi à noter que TinySEC n'est pas adapté à TinyOs-2.x.

2.7 Conclusion

Les solutions proposées pour protéger les réseaux de capteurs sans fil permettent de limiter le nombre des attaques possibles sur ce type de réseau. Malheureusement chacune des solutions n'est capable de contrer qu'une partie des attaques et parfois leur utilisation a un coût énergétique non supportable par les réseaux de capteurs. En plus, la latence de l'ensemble des solutions est consistante ce qui rend ces solutions non adaptables aux RCSF orientés « Event-driven ». D'où, il s'avère nécessaire de mettre en place un mécanisme de sécurité performant et d'accélérer les calculs. Pour ces exigences, nous proposons d'utiliser les ECC comme approche digne d'être sûre et la technique de parallélisation de l'opération de chiffrement pour minimiser la latence.

Dans le prochain chapitre, nous proposons une approche parallélisée basée sur les courbes elliptiques.

Chapitre 3

Parallélisation du chiffrement basée sur les ECC

3.1 Introduction

Les applications de type Event-Driven dans les RCSF nécessitent que l'information devrait être remontée rapidement au centre de contrôle ainsi que dans les applications orientées surveillance conçues pour les catastrophes où les capteurs ont éventuellement quelques secondes pour envoyer l'information à la station de base avant qu'ils soient détruits.

Pour répondre à ces deux challenges, il s'avère nécessaire d'utiliser d'une part un moyen sûr pour remonter l'information à la station de base et d'autre part un mécanisme permettant de la remonter aussi vite que possible.

Pour répondre au premier défi, nous avons fait appel à la cryptographie basée sur les courbes elliptiques (ECC) qui est considérée comme un moyen plus efficace qu'au RSA car ECC pourrait offrir le même niveau de sécurité en utilisant des clés plus courtes que RSA. En outre, pour accélérer les calculs et en particulier la multiplication d'un scalaire par un point, nous avons opté d'une part à des algorithmes qui permettent de minimiser le temps de calcul et d'autre part à impliquer plusieurs voisins d'un capteur dans ce calcul. Nous avons adopté cette approche de parallélisation de calculs car les capteurs sont contraints en termes de calcul, mémoire et énergie ; en plus nous avons besoin de remontrer rapidement l'information à la station de base.

Dans ce chapitre nous présentons les concepts de base des courbes elliptiques et la cryptographie sous-jacente à cette technique. Puis les algorithmes qui permettent d'accélérer la multiplication d'un scalaire par un point car cette opération est la plus gourmande en termes de temps de calcul et elle est impliquée dans le chiffrement. En plus, nous présentons l'apport de la parallélisation en termes de calcul et de sécurité.

3.2 Les concepts de base des courbes elliptiques

Dans cette section, nous présentons les concepts de base des courbes elliptiques qui sont définies sur un corps et qui permettent de les utiliser dans le domaine de la cryptographie.

La cryptographie basée sur les courbes elliptiques (ECC) a été proposée par [29][30] dans les années 80. Récemment, ECC a attiré beaucoup l'attention des chercheurs en raison de son exigence de clés de plus courte taille comparativement à d'autres techniques de cryptographie à clé publique à l'instar de RSA en particulier dans le domaine des systèmes embarqués où les dispositifs ont une puissance de calcul très limitée. Dans [31], ont démontré qu'ECC peut atteindre le même niveau de sécurité en utilisant une clé plus courte que celle de RSA (une clé de 160 bits dans ECC est équivalente à une clé de 1024 bits dans RSA).

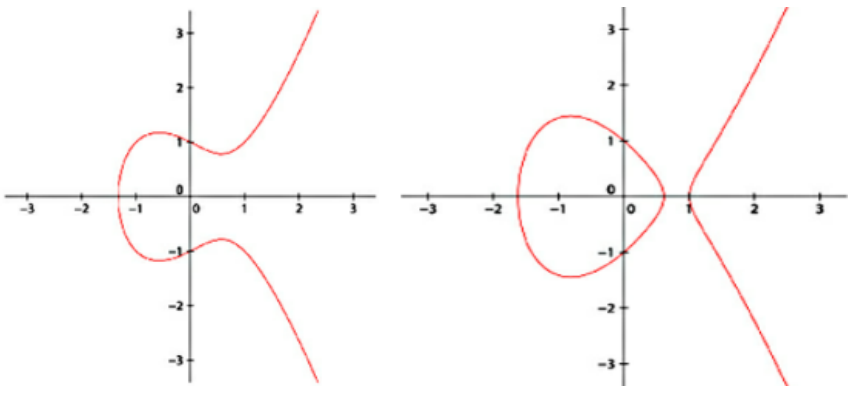
Dans le domaine de la cryptographie, les courbes elliptiques sont définies sur un corps fini représenté soit par F_p où p est un nombre premier ou F_2^m. Dans notre contexte, nous adoptons la première représentation et nous définissons une courbe elliptique E_p sur un corps fini F_p où p est un nombre premier et $E_p(F_p)$ l'ensemble de solutions (x, y) à l'équation :

$$y^2 = x^3 + ax + b \, mod \, p \qquad 4a^3 + 27b^2 \, mod \, p \neq 0$$

Et un point à l'infini, noté O [32].

L'ensemble des points d'une courbe elliptique peut être muni d'une structure de groupe abélien dont l'élément neutre est le point à l'infini, dénoté O. L'opération du groupe est une addition qui permet d'additionner deux points de la courbe et qui donne un troisième point qui appartient aussi au groupe. La loi de groupe peut être interprétée géométriquement grâce à la fameuse méthode dite corde et tangente.

Une courbe elliptique qui est définie sur un corps fini et représentée par les paramètres suivants : a, b et p. la Figure 3.2.1 illustre deux exemples de courbes elliptiques .



(a) : $y^2 = x^3 + x - 1$ (b) : $y^2 = x^3 + 2x - 1$

FIGURE 3.2.1: exemples sur les courbes elliptique

3.3 Arithmétiques sur les courbes elliptiques

Cette section est une présentation synthétique des éléments de base nécessaires à la compréhension de l'arithmétique des courbes elliptiques. Soit $E(K)$:$y^2 = x^3+ax+b$ une courbe elliptique, alors $E(K)$ est un groupe pour la loi de composition suivante [32] :

1. $P + O = O + P = P$ pour tout $P \in E(K)$.
2. Soit $P(x, y) \in E(K)$, on définit $-P\,(o\grave{u}\,\bar{P})\,par : -P = (x, -y)$.
3. Soient $P1(x1, y1)$ et $P2(x2, y2)$ deux points sur la courbe tels que $P1 \neq -P2$,

Alors $P1 + P2 = P3\,(x3; y3)$ avec :

$x_3 = \lambda - x_1 - x_2$

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

où $\lambda = \dfrac{y_2 - y_1}{x_2 - x_1}$ si $P1 \neq P2$, et $\lambda = \dfrac{3x_1^2 + a}{2y_1}$ sinon.

Pour la somme de deux points, Il est possible, comme l'ont démontré [29][30], de coder avec cette opération au lieu de travailler avec l'addition usuelle. Il en résulte une plus grande complexité des calculs qui fait dire aux spécialistes que les ECC avec une clef de 192 bits assurent le même niveau de sécurité qu'une clef de 1024 bits pour la méthode RSA.

Il est donc probable que cette méthode sera de plus en plus utilisée pour transmettre les clefs.

3.3.1 Addition de points

L'addition de point est l'additionnement de deux points J et K sur une courbe elliptique pour obtenir un autre point L sur la même courbe elliptique.

Considérez deux points J et K sur une courbe elliptique.

- *1^{ér} Cas*

Si $K \neq -J$ alors la droite passant par ces deux points dessinée croisera la courbe elliptique à exactement encore un point $-L$. Le symétrique du point -L en ce qui concerne l'axe des abscisses donne le point L, qui est le résultat de l'addition de points J et K, comme l'illustre la figure 3.3.1.

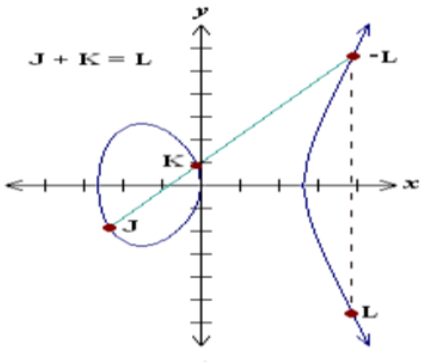


FIGURE 3.3.1: Représentation du 1er cas de l'addition de J et K

- *2ième Cas*

Si $K = -J$ la droite passant par ces deux points dessinée se croise à un point à l'infini O. D'où $J + (-J) = O$. Un négatif d'un point est le symétrique de ce point en ce qui concerne l'axe des abscisses, La figure 3.3.2 est l'illustration de ce cas.

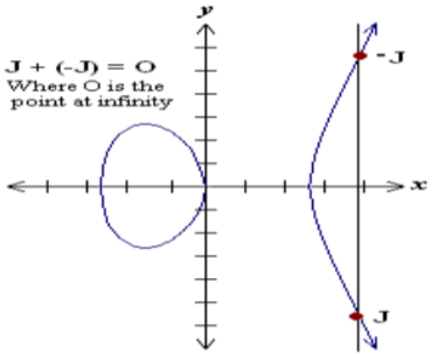


FIGURE 3.3.2: Représentation du 2ième cas de l'addition de J et K.

3.3.2 Doublement de point

Le doublement de point est l'addition d'un point J sur la courbe elliptique à lui-même pour obtenir un autre point L sur la même courbe elliptique.

- *1ér Cas*

Si la coordonnée du point J n'est pas le zéro alors la ligne de tangente à J croisera la courbe elliptique à exactement encore un point $-L$. Le symétrique du point L. En ce qui concerne l'axe des abscisses donne le point L, qui est le résultat du doublement du point J. Ainsi $L = 2J$. Comme le montre la figure 3.3.3.

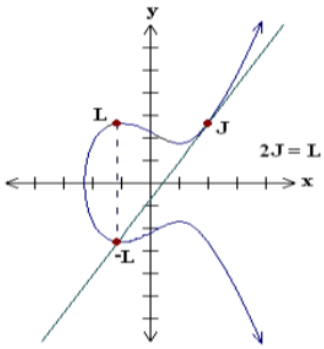


FIGURE 3.3.3: Représentation du 1$^{\text{ér}}$ cas du doublement de point.

- 2$^{\text{ième}}$ Cas

Si la coordonnée du point J est le zéro alors la tangente à ce point se croise à un point à infini O. De là $2J = O$ quand $Y_J = 0$. La figure 3.3.4 illustre le doublement de point dans ce cas-là.

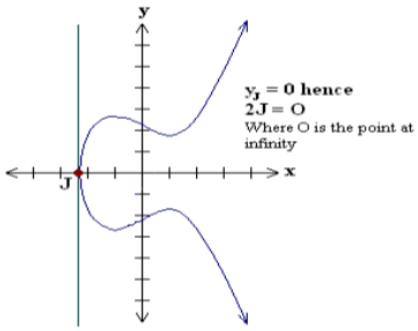


FIGURE 3.3.4: Représentation du 2$^{\text{ième}}$ cas du doublement de point.

3.4 Courbes elliptiques et cryptographie

Le principe de la cryptographie à clé publique repose sur un couple de clés, l'une publique, l'autre privée. Retrouver la clé privée à partir de la clé publique doit revenir à résoudre un problème considéré comme difficile (en termes de temps de calcul). Dans le cas des courbes elliptiques, le problème en question est celui du logarithme discret (ou PLD).

Nous allons donc le définir, analyser son niveau de sécurité dans le cadre les courbes elliptiques, puis faire une rapide comparaison avec RSA [32].

3.4.1 Définition du logarithme discret

Soit G un groupe (note additivement) cyclique fini d'ordre n engendré par un élément P.

Soit $H =< P >$ le sous-groupe engendré par P, alors :

$\forall Q \in H$, $\exists n \in \mathbb{N} : Q = n\,P$

Cet entier n est appelé le logarithme discret de Q en base P et nous le noterons $log_p(Q)$. Le problème du logarithme discret dans un groupe consiste donc à retrouver l'entier n à partir de la donnée publique (H, P, Q). La sécurité des protocoles basés sur les courbes elliptiques repose sur la résolution de ce problème.

Usuellement ce problème est plutôt présenté pour un groupe noté multiplicativement ce qui donne :

Soit G un groupe (noté multiplicativement) cyclique fini d'ordre N engendré par un élément g.

Soit h un élément de G. Comme G est un groupe cyclique engendré par g, il existe un unique entier n compris entre 1 et N tel que $h = gn$. Cet entier n est appelé le logarithme discret de h en base g et nous le noterons $log_g(h)$.

Maintenant nous faisons la correspondance entre les courbes elliptiques et le logarithme discret tel que : Soit E une courbe elliptique définie sur Fp. Soit P et Q deux points de $E(Fp)$ tels que $Q = nP$

En résumé, le problème du logarithme discret revient donc à déterminer un entier n tel que $Q = nP$.

Exemple : Échange de clés de Diffie-Hellman

Supposons que deux capteurs A et B veuillent partager un secret commun mais ces derniers ne peuvent pas communiquer par un canal sécurisé. L'échange de clés de Diffie-Hellman se déroule de la manière suivante :

1. A et B choisissent publiquement un groupe G et un point $P \in G$ appelé point générateur,
2. A choisit secrètement un entier k_A et calcule $(k_A * P)$,
3. B choisit secrètement un entier k_B et calcule $(k_B * P)$,
4. A et B échangent publiquement les données $(k_A * P)$ et $(k_B * P)$,
5. A peut calculer $[k_A * (k_B * P)]$,
6. B peut calculer $[k_B * (k_A * P)]$,
7. A et B possèdent tous deux la quantité $(k_A * k_B * P)$ qui sera leur secret commun.

Un attaquant éventuel n'aura quant à lui en sa possession que les données G, P, $k_A * P$, $k_B * P$ pour retrouver $k_A * k_B * P$. C'est ce que l'on appelle le problème de Diffie-Hellman calculatoire (PDHC) [32].

3.4.2 Multiplication d'un point par un scalaire

Une fois la loi d'addition définie sur les courbes, nous pouvons définir, pour tout entier $k \in \mathbb{N}$, le morphisme de multiplication scalaire par k :

$[k] : E \longrightarrow E$

$P \longrightarrow [k]P = \underbrace{P + P + ... + P}_{K fois}$

Le temps nécessaire pour réaliser la multiplication d'un point par un scalaire est très consistant. De ce fait, nous proposons d'utiliser des algorithmes qui permettent d'alléger ce calcul pour qu'un capteur n'épuise pas toute son énergie dans ce calcul.

Il existe de nombreux algorithmes permettant d'effectuer une multiplication de point par un scalaire. Ils découlent en général directement de la façon dont on représente ledit scalaire.

L'algorithme qui suit est un algorithme classique basé sur la représentation binaire du scalaire. C'est principalement une série de doublements entrecoupée d'additions (dépendantes du nombre de 1 dans la représentation du scalaire).

Algorithme 3.1 Doublement et addition

Données : $P \in E(K)$, $k = (k_{l-1} \dots k_0)_2 \in \mathbb{N}$.
Résultat : $[k]P \in E(K)$.
Début
  $Q \longleftarrow P$
  Pour $i = l - 2$ à 0 faire
    $Q \longleftarrow [2]Q$
    Si $k_i = 1$ alors
      $Q \longleftarrow Q + P$
    Fin
  Fin
Fin
Retourner Q

Pour effectuer la multiplication scalaire d'une manière originale on fait appelle à un algorithme moins connu dû à Montgomery [33].

Algorithme 3.2 Échelle de Montgomery

Données : $P \in E(K)$,$k = (k_{l-1} \dots k_0)_2$
Résultat :$[k]P$.
Début
  $(P_1, P_2) \longleftarrow (O, P)$
  Pour$i = l - 1$ à 0 faire
    Si $k_i = 0$ alors
      $(P_1, P_2) \longleftarrow ([2]P_1,\ P_1 + P_2)$
    Fin
    Si $k_i = 1$ alors
      $(P_1, P_2) \longleftarrow (P_1 + P_2,\ [2]P_2)$
    Fin
  Fin
Fin
Retourner P_1

3.4.3 Comparaison entre RSA et ECC

Le tableau ci-dessous compare la taille des clés utilisées en cryptographie dans RSA et ECC [31] :

Taille de la clé (bits)	
RSA	**ECC**
1024	160
2048	224
3072	256

TABLE 3.1: comparaison entre ECC et RSA

Nous remarquons que la cryptographie basée sur les courbes elliptiques permet d'utiliser des clés de taille moyenne comparativement à celles du RSA tout en fournissant les mêmes performances.

Ce constat permet de favoriser l'utilisation des courbes elliptiques pour les systèmes possédant des ressources limitées en termes de mémoire et CPU tels que les capteurs. Dans ce contexte, nous pourrons parler de cryptographie légère.

La problématique réside dans la multiplication d'un scalaire par un point de la courbe elliptique, pour cela on a fait appel au parallélisme.

3.5 Introduction au parallélisme

Si le parallélisme existe depuis un demi-siècle, il apparaît dans les architectures grand public en 2005 avec les premiers processeurs multi-cœurs, qui bénéficient ainsi de 50 ans de recherches dans ce domaine.

Le parallélisme a pour but d'augmenter la vitesse d'applications en multipliant les ressources de calcul d'un système. Pour en bénéficier, l'application doit être structurée en tâches indépendantes qui pourront être exécutées de manière concurrente par les différentes unités de calcul [21].

3.5.1 Parallélisations avec des points pré-calculés

On suppose que dans notre système on utilise des nombres entiers de longueur 160 bits qui peuvent être exprimés en hexadécimal sous forme :
$a_{39}.16^{39} + a_{38}.16^{38} + \ldots + a_1.16^1 + a_0.16^0$. Et $a_i \in [0, F]$
Pour calculer la multiplication scalaire :
$R = m.P$
$=(a_{39}.16^{39} + a_{38}.16^{38} + \ldots + a_1.16^1 + a_0.16^0).P$
$=\sum_{i=0}^{39} a_i.16^i.P$
L'idée est de paralléliser le calcul en sacrifiant la mémoire de stockage [22].
On calcule le stocke 40 points $\{i \in [0, 39] \mid Q_i = 16^i.P\}$Ensuite il suffit de découper le nombre m en plusieurs segments et de les diffuser aux autres capteurs.

Exemple :

Un capteur C1 veut calculer $m.P$, il a trouvé trois voisins (C2, C3, C4) autour qui sont disponible. Il découpe donc la valeur m en 4 segments, et chacun calcule une partie :

$$R = \underbrace{\sum_{i=0}^{9} a_i.Q^i}_{C1} + \underbrace{\sum_{i=10}^{19} a_i.Q^i}_{C2} + \underbrace{\sum_{i=20}^{29} a_i.Q^i}_{C3} + \underbrace{\sum_{i=30}^{39} a_i.Q^i}_{C4}$$

Cette solution est conçue pour la multiplication de point fixe, c'est à dire que le point P est connu à l'avance et ses coordonnées ne changent jamais. Elle est donc applicable à l'ECC car le point générateur de la courbe est souvent prédéfini et on ne le change pas pendant la durée de vie du crypto système [22].

Exemple :

Prenons un exemple en décimal. On veut calculer 620413*2, on parcourt tous les chiffres de gauche à droite et on calcule le produit en suivant les étapes ci-dessous :

$$6 * 2 = 12$$

$$12 * 10 + 2 * 2 = 124$$

$$124 * 10 + 0 * 2 = 1240$$

$$1240 * 10 + 4 * 2 = 12408$$

$$12408 * 10 + 1 * 2 = 124082$$

$$124082 * 10 + 3 * 2 = 1240826$$

Pour paralléliser le calcul, on découpe le nombre 620413 en 2 parties qui sont respectivement 620 et 413, et on peut calculer les 2 parties séparément :

$$6 * 2 * 10^3 = 12 * 10^3$$

$$12 * 10^4 + 2 * 2 * 10^3 = 12,4 * 10^4$$

$$12,4 * 10^5 + 0 * 2 * 10^3 = 12,4 * 10^5$$

$$= P1$$

$$4 * 2 = 8$$

$$80 + 1 * 2 = 82$$

$$820 + 3 * 2 = 826$$

$$= P2$$

Maintenant il suffit d'additionner $P1$ et $P2$ pour obtenir le résultat final 124826. On remarque que le segment contient les chiffres de l'ordre 10^5 à 10^3, le multiplicateur 2 doit encore être multiplié par 10^3.

Et le deuxième segment contient des chiffres de l'ordre 10^2 à 10^0, le multiplicateur 2 est donc multiplié par $100 = 1$.

Et on peut remarquer que pour accélérer le calcul de la première partie, il suffit d'éviter recalculer $2 * 10^3$.

Pour la multiplication scalaire sur les courbes elliptiques. On calcule $K * P$, tel que k est un nombre entier de 160 bits et P est un point sur la courbe.

Pour paralléliser le calcule, il suffit que l'on pré calculer un seul point$2^{40}.P$.

Et si on paralléliser le calcule sur 4 capteurs, il suffit de pré calculer 3 points qui sont :

$$2^{120}.P,\ 2^{80}.P \text{ et } 2^{40}.P$$

$$k.P = (K_1.2^{120} + K_2.2^{80} + K_3.2^{40} + K_4.2^0).P$$

$$k.P = K_1.2^{120}.P + K_2.2^{80}.P + K_3.2^{40}.P + K_4.2^0.P$$

3.5.2 Réduction de points pré-calculés basée sur l'algorithme d'Elgamal

Dans cette section, nous présentons l'algorithme d'Elgamal, et nous montrons comment nous pouvons l'adapter à l'ECC puis, nous accélérons les calculs en utilisant des points pré-calculés. Nous avons opté pour cette technique car le problème du logarithme discret est difficile à résoudre.

Tout d'abord, nous présentons à partir d'un exemple le fonctionnement de l'algorithme d'Elgamal. Supposons qu'Alice veut envoyer un message secret à Bob. Tout d'abord, Bob fabrique une clé publique de la manière suivante. Il choisit une courbe elliptique E définie sur un corps fini Fq de telle manière que le problème du logarithme discret soit plus difficile à résoudre. Il choisit aussi un point P sur E dit point générateur tel que l'ordre de P soit un grand nombre premier. Il choisit un nombre entier secret s et calcule $B = sP$. La courbe E, la taille du corps fini Fq et les points P et B sont les informations publiques de Bob et s est sa clé secrète. Pour envoyer le message, Alice fait comme suit [23] :

1. Elle télécharge la clé publique de Bob.
2. Elle transforme le message en un point $M \in E(Fq)$.
3. Elle choisit un nombre entier secret k et calcule $M_1 = kP$.
4. Elle calcule $M_2 = M + kB$.
5. Elle envoie M_1 et M_2 à Bob.

Bob déchiffre le message en calculant :$M = M_2 - sM_1$. Nous avons cette égalité parce que :

Nous avons cette égalité parce que :

$$\begin{aligned} M_2 - sM_1 &= (M + kB) - s(kP) \\ &= M + k(sP) - skP \\ &= M + s(kP) - skP \\ &= M \end{aligned}$$

Un espion connait les informations publiques et les points M_1 et M_2. Si l'espion savait résoudre le problème du logarithme discret, il pourrait utiliser P et B pour trouver s et ainsi calculer $M_2 - sM_1$. L'espion pourrait aussi utiliser P et M_1 pour trouver k et calculer : $M = M_2 - kB$.

Actuellement, on ne connait pas de moyen plus rapide pour retrouver le message initial en ne sachant que ce qui est rendu public du crypto-système. Donc, apriori, la fiabilité de ce genre de crypto-systèmes dépend fortement des progrès faits en matière de résolution du logarithme discret.

Pour optimiser le temps d'exécution de l'algorithme d'Elgamal, nous pouvons paralléliser le calcul en effectuant des opérations parallèlement et indépendamment. Dans l'algorithme d'Elgamal, on fait distinguer deux opérations indépendantes. Il s'agit des multiplications à faire : $K * G$ et $K * P$. De ce fait il suffit de doubler le nombre de points pré-calculés. Pour pouvoir paralléliser ce calcul de chiffrement, il faut pré-calculer et stocker donc un certain nombre de points.

La technique de parallélisation permet de réduire le temps d'exécution de l'algorithme d'Elgamal. Cependant, elle engendre une consommation de mémoire de stockage très importante car les courbes sont souvent définies dans un corps d'une taille d'une centaine de bits

3.6 Conclusion

Dans ce chapitre nous avons présenté les courbes elliptiques (ECC) et leurs apports dans la cryptographie. Nous avons constaté que les ECC représentent un moyen efficace de sécurité et ils sont adaptables aux dispositifs présentant des ressources limitées en termes de calcul, mémoire et énergie, tels que les capteurs.

Nous avons de même mis le point sur les algorithmes qui accélèrent la multiplication d'un point par un scalaire dans les ECC puisque cette opération est utilisée dans le chiffrement et le déchiffrement et il nécessite un temps de calcul assez grand. Pour optimiser le temps de calcul nous avons opté à une technique de parallélisation.

Dans notre contexte nous avons visé à réduire le temps de réponse au maximum même au détriment de la consommation de l'énergie puisque nous nous sommes intéressés aux applications orientées évènements (Event-Driven).

Dans le chapitre suivant nous présentons les outils matériels et logiciels pour implémenter une application qui assure la sécurité dans les réseaux de capteurs et de réduire le temps des calculs pour les opérations de chiffrement.

Chapitre 4

Contribution et implémentation

4.1 Introduction

La partie implémentation de notre projet consiste à créer une application qui aura pour but de remonter l'information le plus rapidement possible au centre de contrôle et en toute sécurité.

Ainsi pour le bon développement de notre application notre choix c'est porté sur l'utilisation des capteurs TelosB qui sont très répondus dans le secteur de la recherche et TinyOs comme environnement de développement avec le langage NesC.

Dans ce chapitre nous allons définir chacun d'eux tout en donnant un aperçu de notre application.

4.2 Architecture générale

L'architecture générale est constituée d'un ensemble de capteurs TelosB (Sender/Receiver), d'une station de base et d'un ordinateur.

Chacun des capteurs TelosB Sender mesure une grandeur physique et la communique aux capteurs TelosB Receiver via une transmission radio. Ces derniers la communiquent à la station de base via la même transmission. La station de base transmet alors cette grandeur au centre de contrôle via le câble USB.

La figure 4.2.1 décrit le fonctionnement d'une architecture générale.

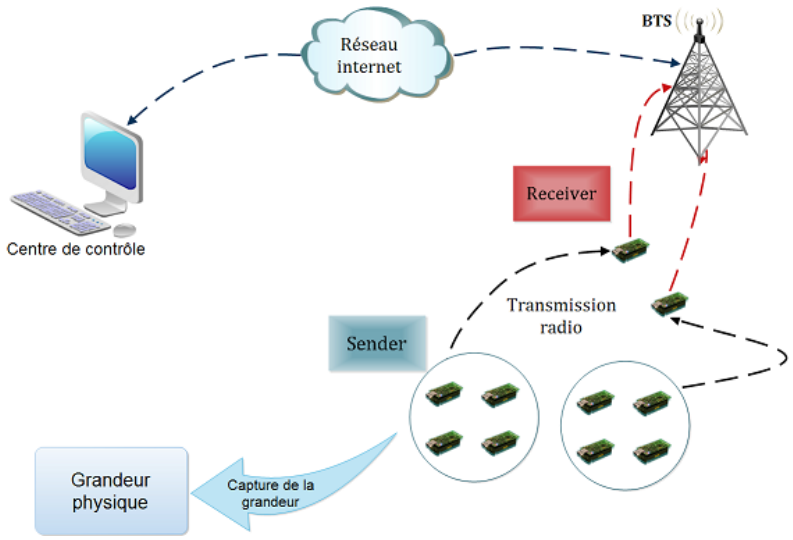


FIGURE 4.2.1: Schéma d'une architecture générale.

4.3 Les choix techniques

L'implémentation de notre application nécessite des moyens logiciels et matériels.

4.3.1 Système d'exploitation « TinyOs »

On fait distinguer plusieurs systèmes d'exploitation qui sont dédiés le plus souvent à un ou plusieurs types de capteurs spécifiques et sont sujets à des révisions fréquentes.

Par ailleurs, ils ne sont pas comme les systèmes d'exploitation que l'on retrouve sur ordinateur qui supportent un système de fichier et les possibilités d'exécution multitâches.

Ces systèmes d'exploitation permettent de développer un peu plus le caractère intelligent des capteurs.

TinyOs : [34] est un système d'exploitation orienté événement (event-driven) conçu pour les réseaux de capteurs. Il est implémenté entièrement en Nesc et il respecte une architecture basée sur une association de composants, permettant de réduire la taille du code nécessaire à sa mise en place. Il occupe un espace mémoire très faible dans sa distribution minimale (512 octets). Par conséquent, il s'adapte aux capteurs pourvus de ressources mémoires très limitées.

Pour autant, la bibliothèque de composants de TinyOs est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données.

Une application s'exécutant sur TinyOs est constituée d'une sélection de composants systèmes et de composants développés spécifiquement pour l'application. Parmi les principaux composants systèmes, on trouve :

– l'ordonnanceur TinyOs : Cet élément est responsable de la gestion des tâches et des évènements du système. Son choix déterminera le fonctionnement global du système et le dotera de propriétés précises telles que la capacité à fonctionner en évènementiel.
– Communication et Synchronisation : Les notions de synchronisation et de communication sont étroitement liées et l'existence de l'une ne peut s'envisager sans faire référence à l'autre. En général, les processus d'une application n'évoluent pas de façon indépendante. La spécification de l'application fixe les relations logiques et temporelles entre ses processus. Dans le but de synchroniser ces processus, on fait communiquer ces processus à l'aide de mécanismes offerts par le système. Parmi les mécanismes existants pour les systèmes Linuxien et spécialement TinyOs, tels que par Messages, zone commune....

FIGURE 4.3.1: Logo de TinyOs.

4.3.2 MIG (Message Interface Generator)

TinyOs fournit des outils pour produire automatiquement des objets message des descriptions de paquet. Ainsi, au lieu d'analyser les formats de paquet manuellement, l'outil de MIG établit une interface Java à la structure de message. En effet, pour une séquence d'octets donnée, le générateur de code MIG devra analyser automatiquement chacun des champs des paquets et fournit un ensemble d'accédant et de mutators standards pour visualiser les paquets reçus ou produire des nouveaux paquets [4].

4.3.3 Langages utilisés

NesC

C'est un langage de programmation dérivé du langage C, orienté composant, conçu pour incarner les concepts de structuration et d'exécution du système TinyOs. Il propose une architecture basée sur des composants, permettant de réduire la taille mémoire du système TinyOs et de ses applications. Un composant peut être un concept abstrait ou un élément matériel tel que LEDs, timer,... et il peut être réutilisé dans différentes applications. Ces applications sont des ensembles de composants associés entre eux dans un but précis.

En NesC, les composants présentent des similarités avec des objets. Les états sont encapsulés et on peut y accéder par des interfaces. L'implémentation des composants

est réalisée par la déclaration des tâches, des commandes ou des événements. En outre, l'ensemble des composants et leurs interactions sont fixés à la compilation ce qui permet d'optimiser l'application pour une exécution plus performante.

NesC permet de déclarer deux types de fichiers : les configurations et les modules.

Configuration : Est la définition des composants qui seront utilisés par l'application déployée sur le capteur.

```
Configuration nomConfig{}

Implémentation {
        //liste des modules et configurations utilisées
        Composants Main, Module1, ……ModuleN, Config1……..ConfigM
        //description des liaisons
        Interfaces requises->Interfaces fournies
}
```

Module : Module est un composant dans lequel on met toutes les implémentations des interfaces qu'il doit fournir et les implémentations des événements s'il utilise aussi des interfaces.

```
Module nomModule{
Provides {
        //liste des interfaces fournies
        Interface nomInterface }
Uses {
        //liste des interfaces requises
        Interface nomInterface  }
Implémentation {
        //déclaration des variables
        //implémentation des fonctions décrites par les interfaces fournies
} }
```

JAVA

Pour réaliser l'interface graphique notre choix s'est porté sur JAVA car il permet la création des interfaces graphiques grâce à sa librairie Swing et aussi parce que c'est le seul langage dont le code exécutable est portable.

4.3.4 Choix du matériel

Pour la réalisation de notre application nous avons utilisé les capteurs TelosB. Ces capteurs ont été développés par les chercheurs de l'université de Berkeley et sont commercialisés par CrossBow. Ils sont généralement utilisés pour l'éxpérimentation dans le domaine de recherche. La figure 4.3.2 illustre l'architecture de ces derniers.

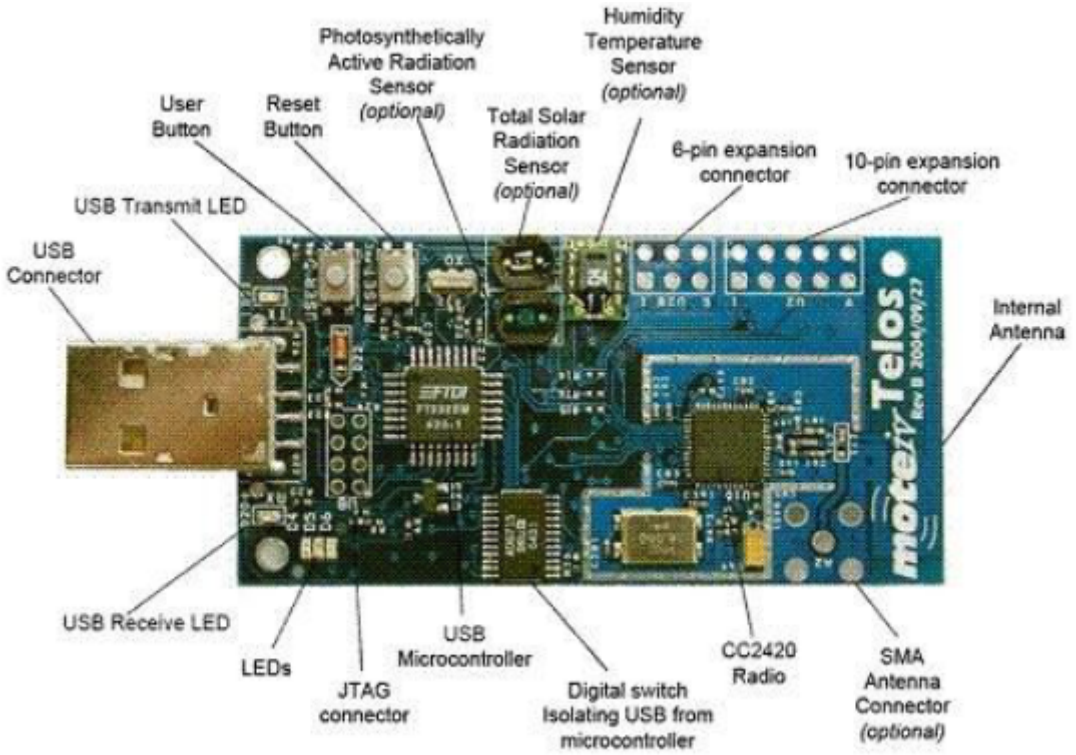


FIGURE 4.3.2: Recto d'un TelosB.

Les modules essentiels intégrés dans ces capteurs sont les suivants [25] :

- **Un microcontrôleur MSP430 :** fabriqué par Texas Instrument, cadencé à 8MHz. Il dispose de 10 KOctets de RAM et de 48 KOctets de mémoire flash. Ce microcontrôleur est optimisé pour répondre aux contraintes d'économie d'énergie des réseaux de capteurs, d'où son cadencement faible, et de ce fait, il a une capacité de calcul assez limitée.
- **Un module radio CC2420 :** conforme à la norme IEEE 802.15.4, travaillant sur une bande de fréquences comprise entre 2,4 GHz et 2,4835 GHz. Il permet de travailler sur 16 canaux étanches différents
- **Un port USB :** permettant de programmer les cartes, ou de faire remonter des informations et de les récupérer ensuite à l'aide d'un hyper terminal à partir d'un ordinateur.

– **Des LEDs :** outil visuel indispensable de vérification du fonctionnement du code intégré. Chaque carte dispose de trois LEDs (rouge, jaune et bleu). Elles permettent de suivre le changement d'état de fonctionnement de la carte (réception, envoi, etc.) avec une programmation adéquate.

4.4 Les étapes de la contribution

Dans la contribution on présente la partie installation matérielle et l'architecture de l'application.

4.4.1 Installation matérielle

Une fois l'installation logicielle terminée, il a fallu installer le matériel : une station de base reliée à l'ordinateur via un câble USB, deux capteurs TelosB pour la maquette (figure 4.4.1). Chacun des TelosB communique avec la station de base via une liaison sans fil et la station de base communique avec l'ordinateur via le câble USB.

FIGURE 4.4.1: Environnements de travail.

4.4.2 Maquette

L'architecture de l'application s'articule autour de trois capteurs TelosB : un Sender, un Receiver, une station de base et aussi un centre de contrôle représenté par un ordinateur. La figure 4.4.2 illustre cette architecture :

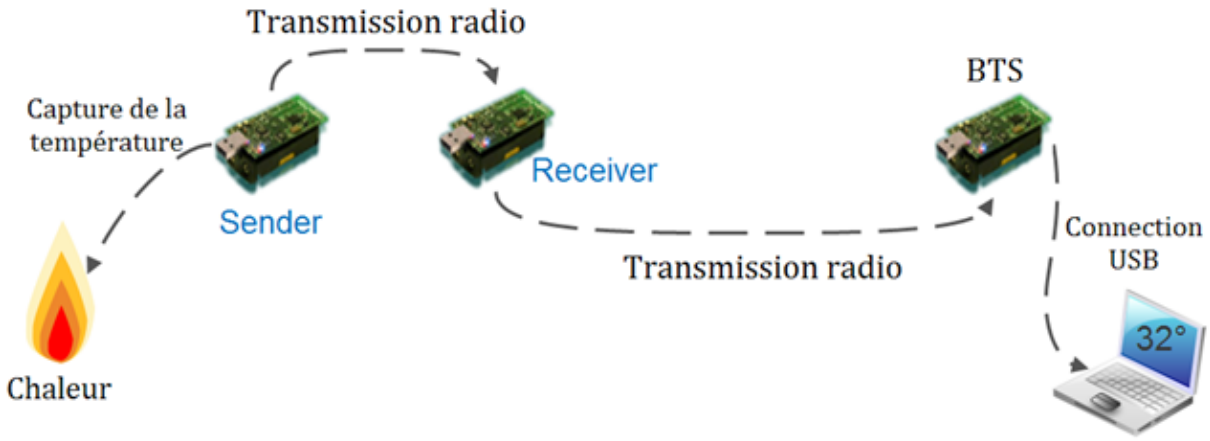


FIGURE 4.4.2: schéma de l'architecture.

Les étapes suivantes expliquent le fonctionnement de l'architecture proposée :
- Le Sender mesure périodiquement les températures qui l'entourent.
- Fait le chiffrement de cette température en se basant sur les courbes elliptiques.
- Transmet le résultat du chiffrement qui sera un point de la courbe choisie au Receiver qui jouera le rôle d'un nœud relai dans l'architecture.
- Le Receiver à son tour retransmet le message à la station de base rattaché au centre de contrôle où réside l'application de déchiffrement.
- L'application déchiffre le message reçu et affiche la température correspondante.
- Pour que la température remonte le plus rapidement au centre de contrôle et en toute sécurité, le Sender envoie une partie de la clé privée aux nœuds voisins qu'ils puissent la calculer parallèlement.

Bien que le schéma puisse paraitre simple, une analyse détaillée des éléments de cette architecture et de leurs fonctions attachées met en évidence certains points essentiels.

4.4.3 Implémentation

Dans cette partie nous allons présenter la démarche qu'on a suivie pour établir notre approche distribuée :

Application côté SENDER

Le Sender nous permet la capture de la température et le chiffrement de cette dernière à l'aide d'ECC pour cela nous devons passer par les étapes suivantes :

1. La lecture de la température.
2. Le choix d'une courbe elliptique pour la génération de points.
3. La représentation de la température capturée sous forme d'un point appartenant à la courbe.

4. Le choix du point générateur.
5. Le chiffrement du point représentant la température :
   - Le choix de la clé privé K_A (assez grande).
   - La réception de la clé publique Q_B ($Q_B = K_B * P$) de la part de la station de base.
   - Faire le calcul de $Q_A = K_A * P$ et l'envoyer au Receiver.
   - Calculez $Q_{xy} = K_A * Q_B$
   - Faire l'addition entre le point représentant la température et Q_{XY}

   (Exemple : une température de 30° ⇢ point (2,1) ⇒ point (2,1) + Q_{XY}) L'addition qui représentera le point chiffré.

   Les clés K_A et K_B sont assez grandes donc on est passé par une optimisation pour diminuer le temps de calcul et aussi faire une parallélisation du calcul en impliquant les nœuds voisins.

   On a utilisé un algorithme pour accélérer la multiplication d'un point par un scalaire. Cet algorithme se base sur la division binaire d'un nombre. Dans notre cas il s'agit des clés privées K_A et K_B, et un point de la courbe
6. Enfin il transmet le message chiffré au Receiver.

   Le point de la courbe représentant une information est donnée par la structure de données contenues dans **«Message.h »** qui est un fichier d'entête contenant la structure des messages à envoyer.

```
#ifndef MESSAGE_H
#define MESSAGE_H
1- typedef nx_struct Temperature_Msg{
2- nx_uint16_t temperature;
3- } Temperature_Msg;
4- enum {AM_TEMPERATURE_MSG = 6};
5- typedef nx_struct Point_Msg{
6- nx_uint16_t x;
7- nx_uint16_t y;
8- nx_uint16_t z;
9- nx_uint16_t w;
10- nx_uint16_t K;
11- nx_uint16_t K1;
12- nx_uint16_t Tx;
13- nx_uint16_t Ty;
14- nx_uint16_t temps;
15- } Point_Msg;
16-enum {AM_POINT_MSG = 6};
#endif
```

FIGURE 4.4.3: Exemple d'un fichier d'en-tète

Le fichier Message.h contient deux définitions importantes.

Lignes 1-3 et 5-15 : La définition de la structure du message que l'on va envoyer. Les types de données sont précédés d'un préfixe nx_. Ce sont des types que l'on utilise uniquement en communication radio et série.

Lignes 4 et 16 : Type AM de paquet. On peut demander à un capteur d'envoyer plusieurs types de paquets : paquet de température, paquet d'humidité etc. Pour que le récepteur puisse les différencier et les traiter différemment, il faut donner à chaque paquet une étiquette (le type AM). Le type AM est toujours <AM_> + <NOM DE STRUCTURE EN MAJUSCULE>.

On crée d'abord un module et une configuration, nommés respectivement **SenderC** et **SenderAppC** où **SenderAppC** implémente les différents composants.

```
configuration SenderAppC{}
implementation{
        components SenderC as App;
        components MainC, LedsC;
        components new TimerMilliC() as Timer;
        components ActiveMessageC;
        components new SensirionSht11C() as Sense;
        App.Boot -> MainC;
        App.Leds -> LedsC;
        App.Timer -> Timer;
        App.Read -> Sense.Temperature;
        App.Packet -> ActiveMessageC;
        App.AMSend -> ActiveMessageC.AMSend[AM_TEMPERATURE_MSG];
        App.RadioControl -> ActiveMessageC;
        }
```

FIGURE 4.4.4: Exemple d'une configuration

```
module SenderC {
provides {interface Arith ;}
uses interface Boot ;
uses interface Timer<TMilli> ;
uses interface Leds ;
uses interface Read<uint16_t> ;
uses interface Packet ;
uses interface AMSend ;
uses interface SplitControl as RadioControl ;
implementation {
event void Boot.booted() { /*** IMPLÉMENTATION***/ }
event void RadioControl.startDone { /***IMPLÉMENTATION***/ }
event void RadioControl.stopDone {}
event void Timer.fired() { /*** IMPLÉMENTATION***/ }
event void Read.readDone(error_t err, uint16_t val) { /*** IMPLÉMENTATION***/
}
event void AMSend.sendDone(message_t* msg, error_t err) { /*** IMPLÉMENTA-
TION***/
command int pgcd (int, int) { /*** IMPLÉMENTATION***/ }
command int inverse (int, int) {/*** IMPLÉMENTATION***/ }
command int power (int, int) { /*** IMPLÉMENTATION***/ }
command Point_Msg calcul_pt (int, Point_Msg) {/***IMPLÉMENTATION***/ }
command Point_Msg calcul_pt _pt (Point_Msg, Point_Msg) { /*** IMPLÉMENTA-
TION***/ }
```

FIGURE 4.4.5: Le module SenderC

Pour définir les déclarations des différentes méthodes utilisées, on a du utilisé une interface nommée ARITH.nc définie ci-dessous :

```
interface Arith {
        command Point_Msg calcul_pt(int ,Point_Msg) ;
        command Point_Msg calcul_pt_pt(Point_Msg ,Point_Msg) ;
        command int pgcd(int,int) ;
        command int inverse(int,int) ;
        command int power (int,int) ;
        command Point_Msg optimisation (int, Point_Msg) ;
        command float produit (int,int) ;
        command int modulo (int,int) ;
        }
```

FIGURE 4.4.6: Interface ARITH.nc

Maintenant nous détaillons les fonctions exécutées par le Sender.

La lecture de la température

- Dans la configuration du SenderAppC { } On a implémenté les composants suivants :

```
components SenderC as App ;
components new SensirionSht11C() as Sense ;
App.Read -> Sense.Temperature ;
```

- Dans le composant SenderC on a utilisé l'interface Read :

```
Uses interface Read<uint16_t> ;
```

- Dans l'implémentation de ce module quand le timer déclenche, on allume le LED 0 et commence la lecture de température.

```
Event void Timer.fired(){
        Call Leds-led0Toggle() ;
        Call Read.read() ;
        }
```

Le choix de la courbe elliptique

Pour ce qui est du choix de la courbe elliptique, on choisit a et b tel que le discriminant delta soit différent de 0($Delta = 4a^3 + 27b^2$). Exemple : soient $a = 2$, $b = 3$, $p = 71$ (avec ces paramètres 87 points vont être générés).

```
aa = a*a*a ;
aa = 4*a ;
bb = b*b ;
bb = 27*b ;
delta = aa + bb ;
delta = delta%p ;
if (delta == 0) { printf ("erreur") ; } else {
        nb_pts = 0 ;
        x = 0 ; while (x< p){ y=0 ;
                while (y < p) { xx = x*x*x+ a*x + b ;
                        xx = xx%p ;
                        yy = y*y ;
                        if (yy==xx){tab_pt[nb_pts].x= x ;
                                tab_pt[nb_pts].y= y ;
                                nb_pts++ ; }
                        y++ ; }
                x++ ;}
        }
```

FIGURE 4.4.7: Algorithme de generation de points

Chiffrement :

Contient les méthodes suivantes :

1. Le calcul de $Q_A = K_A * P$: on a utilisé la méthode calcul_pt(int, point) défini précédemment dans l'interface Arith.nc.

Dans cette partie le Sender calcule sa clé publique représentée par :

$$E_1 = K_A * P$$

Ce point (E_1) sera envoyé au centre de contrôle pour l'aider à déchiffrer l'information envoyée par le Sender.

2. Le calcul de Q_{AB}+ le point correspondant à la température a chiffré : on a utilisé la méthode clacule_pt_pt Dans cette partie le Sender :

- Calcule Q_{AB} qui représente le secret commun et faire correspondre la température détectée à un point de la courbe (soit M ce point).
- Calcule le point E2 qui véhicule l'information détectée :

$$E_2 = M + Q_{AB}$$

Ce point (E_2) va être envoyé en même temps qu'avec E_1.

Optimisation

Quand les clés sont assez grandes la multiplication d'un scalaire par un point prend beaucoup de temps. C'est la raison pour laquelle on a essayé d'améliorer ce temps en deux reprises. Dans la première le chiffrement se fait seulement par un seul capteur en utilisant un algorithme qui accélère la multiplication d'un scalaire par un point nommé « shift-add ». Dans la seconde, on a fait appel à une approche parallélisée pour améliorer un peu plus le temps. Dans cette approche plusieurs capteurs voisins sont impliqués dans le calcul. Par exemple, si un capteur a quatre voisins, il divise sa clé en quatre parties qui ne sont pas égales.

```
command Point_Msg Arith.optimisation(int n,Point_Msg P)
{
int re,r ; //int binxy ;
re=1 ;
i=0 ;
binxy=0 ;
while( re != 0 ) {
                re=n/2 ;
                r=n%2 ;
                n=re ;
                tab_bin[i++]=r ;
                binxy=i ;
                }
if (tab_bin[binxy-1]==1) {
                        R.x= P.x ;
                        R.y= P.y ;
                        }
else {
      R.x=0 ;
      R.y=0 ;
      }
for(j=binxy-2 ;j>=0 ;j–)//j= 3 ; bin=5 ; {
                                R= call Arith.calcul_pt_pt(R,R) ;
                                if (tab_bin[j]==1) {R=call Arith.calcul_pt_pt(R,P) ;}
                                }
Qxy.x = R.x ;
Qxy.y = R.y ;
return (Qxy) ;
}
```

FIGURE 4.4.8: Algorithme shift-add

L'envoie au Receiver

SenderC utilise les méthodes Send pour l'envoi du paquet.

```
sndE = (Point_Msg*)(call Packet.getPayload(&pkt, sizeof(Point_Msg))) ;
```

Receiver

Le capteur qui représente le Receiver joue le rôle d'un nœud relai qui reçoit les résultats du chiffrement et les retransmet à la station de base. Ce type de nœuds est impliqué dans

l'architecture dans le but de la hiérarchiser. Il reçoit les paquets provenant de Sender et les envoie à la station de base :

```
E1 = (Point_Msg*)call Packet.getPayload(&pkt, sizeof(Point_Msg)) ;
E1-> x = rcvPayload->x ;
E1-> y= rcvPayload->y ;
E1-> w= rcvPayload->w ;
E1-> z= rcvPayload->z ;
```

Station de base

Reliée au centre de contrôle, contenant un programme prédéfini dans tinyOs.

Centre de contrôle

Exécute une application développée en Java Au niveau du centre de contrôle, on a un programme Java qui reçoit et affiche les paquets reçus, mais pour que l'application puisse interpréter les informations reçus, il faut qu'il sache la structure du contenu, c'est-à-dire la structure que l'on a défini dans le fichier « Message.h ».

TinyOs fournit un outil : MIG, qui est capable d'analyser un « fichier.h » et convertir toutes les structures en classes Java disposant des méthodes nécessaires qui nous permettent d'accéder aux valeurs des attributs.

Maintenant pour afficher les valeurs reçus sur PC. On lance le programme Java en exécutant la commande suivante dans une console :

java net.tinyos.tools.MsgReader Message -comm serial@/dev/ttyUSB0 :telosb

Le centre de contrôle reçoit :
- la clé publique Q_A du Sender représentée par E_1.
- le message chiffré représenté par le point E_2

Le centre de contrôle effectue le déchiffrement à l'aide de ce qu'il a reçu (E_1, E_2). Pour cela il doit retrouver le secret commun $(K_A * Q_B)$ à l'aide de sa clé K_B. Il suffit de multiplier $K_B * Q_A$

Le message déchiffré : $M = E_2 - K_B * (Q_A)$
$= E_2 - K_B * E_1$

4.4.4 Exemple d'exécution

Pour illustrer l'apport de l'algorithme qui accélère le processus chiffrement / déchiffrement nous avons évalué le temps nécessaire pour exécuter ces deux opérations comme suit :

1. **Sans optimisation :** Dans la première version nous avons exécuté la multiplication d'un point par un scalaire sous forme d'additions successives dont le nombre est égal à ce scalaire. On remarque que le temps de calcule est assez grand il a atteint 11.78 Secondes

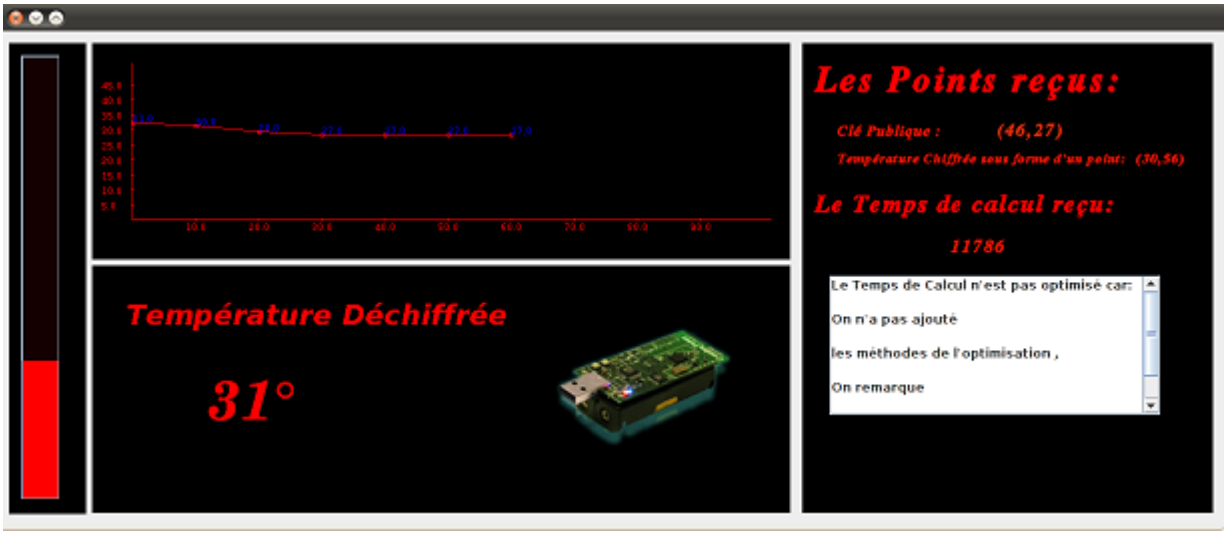


FIGURE 4.4.9: Calcul du temps sans optimisation

2. **Avec optimisation :** Dans la deuxième version nous avons fait une optimisation en utilisant l'algorithme shift-add. Nous avons pu minimiser le temps de calcul jusqu'à arriver à 0.126 Secondes une différence assez importante.

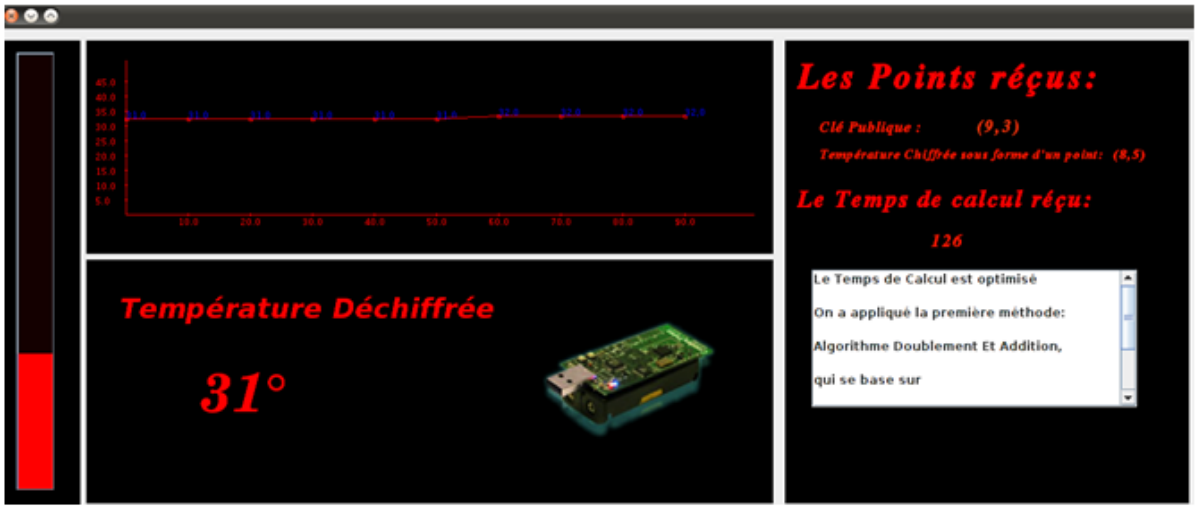


FIGURE 4.4.10: Calcul du temps avec shift-add

3. **Après parallélisation :** dans la troisième version nous avons fait appel à une architecture parallélisée, en impliquant deux capteurs dans la partie chiffrement.

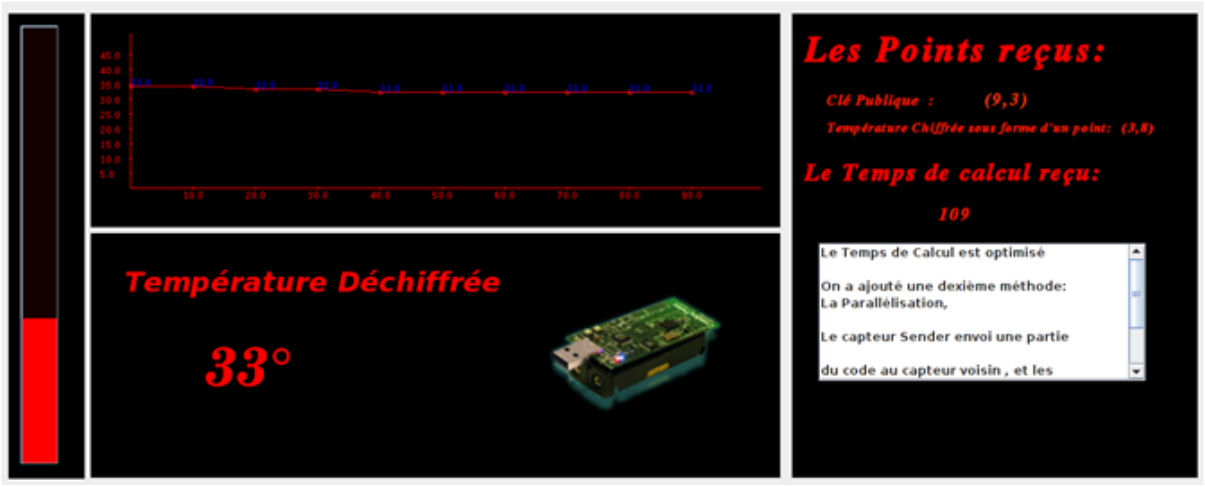


FIGURE 4.4.11: Calcul du temps après parallélisation

Pour illustrer l'apport de la parallélisation nous avons généralisé l'architecture en impliquant un grand nombre de capteurs et en utilisant une clé de l'ordre de 160 bits. Pour chaque configuration nous avons fait trois tests et calculé la moyenne. Le tableau ..... Montre les résultats obtenus.

Test	1 nœud	2 nœuds	3 nœuds	4 nœuds
1	2912	1520	1069	873
2	2911	1509	1063	876
3	2910	1514	1072	872
Moyenne	2911	1514.33	1068	873.66
Gain	0%	47.98%	63.31%	69.98%

TABLE 4.1: Performance du mécanisme de parallélisation

A la lumière de ce tableau on remarque qu'une approche parallélisée permet d'améliorer énormément le temps de réponse. Ce qui rend cette approche bénéfique pour les applications « event-driven » qui exige de remonter l'information le plus rapidement possible au centre de contrôle.

4.4.5 Conclusion

Dans ce chapitre nous avons proposé un crypto-système basé sur les courbes elliptiques, garantit un niveau de sécurité très élevé tout en utilisant une clé de petite taille relativement à RSA.

En outre nous avons mis en place un algorithme qui permet d'accélérer les opérations de chiffrement / déchiffrement. En plus pour mieux améliorer le temps de réponse nous avons implémenté un mécanisme basé sur une approche parallélisée.

Les résultats obtenus sont prometteurs et permettent de penser à appliquer une approche parallélisée pour les réseaux de capteurs orientés événement.

Conclusion générale

Ce projet nous a appris la manière de mener les projets et les différentes façons de traiter les problèmes rencontrés. De plus nous avons acquis des connaissances dans les domaines techniques et nous avons appris à tirer le meilleur de la théorie qui existe afin d'atteindre notre but.

En plus de tout ce que nous avons appris, nous avons développé notre sens du travail collectif et le développement des idées ce qui nous facilitera l'immersion dans le domaine professionnel.

Ce projet nous a permis d'acquérir des connaissances en programmation évènementielle. Il nous a aussi fait découvrir un nouveau langage de programmation, le NesC ainsi que la plateforme de programmation adéquate qui est TinyOs.

Dans notre projet nous avons établi une distribution de calcul sur un réseau de capteurs pour traiter le problème de sécurité dans ce type de réseaux. Ceci est dans le but de remonter l'information rapidement au centre de contrôle et en toute sécurité. Pour faire face à cette problématique, nous avons utilisé un chiffrement basé sur les courbes elliptiques « ECC » tout en impliquant des algorithmes qui accélèrent l'opération de multiplication d'un scalaire par un point. Pour améliorer plus ces calculs nous avons opté pour une architecture parallélisée dans laquelle plusieurs capteurs voisins sont impliqués dans les calculs.

En perspectives nous proposons de mettre en place cette distribution de calcul sur un réseau de capteurs de grande taille tout en utilisant des clés de grandes tailles aussi pour voir l'apport de la parallélisation dans ce type de réseau qui est caractérisé par des ressources limitées.

Annexes

Annexe A

Généralités mathématiques

Dans cette section, nous présentons quelques notions mathématiques qui s'avèrent utiles pour comprendre le mécanisme de sécurité basé sur les courbes elliptiques.

Définition d'un groupe

Un groupe est un ensemble non vide muni d'une loi de composition interne appelée multiplication qui permet de multiplier deux éléments de l'ensemble $(G, *)$ tels que :

– $*$ est associative,

– $*$ admet un neutre eG,

– tout élément de G admet un symétrique pour.$*$

Si $*$ est commutative, on dit que $(G, *)$ est commutatif.

Définition d'un anneau

Un anneau est un ensemble muni de deux lois de composition interne multiplication et addition telles que la multiplication soit distributive par rapport à l'addition et qui permettent de multiplier et d'additionner des éléments de l'ensemble $(A, +, *)$ tels que :

- $(A, +)$est un groupe commutatif de neutre noté 0_A.

- La loi $*$est une loi de composition interne sur A associative et distributive à gauche et à droite par rapport à $+$:

$\forall x,\ y,\ z \in A, \qquad x * (y + z) = x * y + x * z$ et$(x + y) * z = x * z + y * z$

- La loi $*$ admet un neutre différent de 0_A, noté 1_A.

Si la loi $*$ est commutative, l'anneau est dit commutatif.

Définition d'un sous-anneau

Soit $(A, +, *)$ un anneau. Une partie non vide A_1 de A est un sous-anneau de A lorsque :

- $1_A \in A_1$,

- les lois $+$ et $*$ induisent des lois de composition interne, et, muni de ces lois, $(A_1, +, *)$ est un anneau.

Remarques Contrairement aux sous-groupes, on ne peut pas se passer de la condition $1_A \in A_1$, qui ne découle pas des autres conditions. En outre, on pourra montrer qu'une partie A_1 de A est un sous-anneau si et seulement si :

- $(A1, +)$ est un sous-groupe de $(A, +)$,

- $1_A \in A_1$,

- $*$ induit une loi de composition interne sur A_1.

Définition d'un corps

Un corps est un anneau commutatif dans lequel tout élément non nul est inversible.

Si $(K, +, *)$est un corps, un sous-corps de K est un sous-anneau K_1 de K tel que pour tout élément non nul x de K_1, on a $x^{-1} \in K_1$; $(K_1, +, *)$est alors un corps

Annexe B

Installation de TinyOs 2.1.1 sous Linux

Il y a deux façons de faire une installation propre de TinyOs. La première façon est d'installer une VM qui a une installation complète de TinyOs. La deuxième façon est d'installer TinyOs sur votre système d'exploitation hôte. Lors de l'installation sur un système d'exploitation hôte, vous pouvez soit utiliser un paquet Debian ou installer manuellement avec RPM, nous allons nous intéresser à une installation sur un Os avec un paquet Debian.

Installation en deux étapes sur votre système d'exploitation hôte avec les paquets Debian :

Si vous exécutez une version de Linux qui prend en charge les paquets Debian, alors vous voudrez peut-être utiliser le dépôt de paquets TinyOs.

1. Retirez tout ancien référentiel TinyOs du fichier : / etc / apt / sources.list et ajoutez la ligne suivante :
   Un ensemble commun prend en charge toutes les distributions basées sur Ubuntu Debian Squeeze. Spécifiant la version lucide devrait bien fonctionner :

   deb `http://tinyos.stanford.edu/tinyos/dists/ubuntu` **lucid main**

2. Mettez à jour votre cache de dépôts :

   sudo apt-get update

3. Exécutez la commande suivante pour installer la dernière version de TinyOs et tous ses outils pris en charge :

   sudo apt-get install tinyos

Cela vous donnera probablement un message vous invitant à choisir entre les deux versions disponibles. Un exemple à exécuter ensuite est :

sudo apt-get install tinyos-2.1.1

Ajoutez la ligne suivante à votre fichier ~ /. Bashrc ou ~ /.profil dans votre répertoire home afin de mettre en place l'environnement pour le développement TinyOS lors de la connexion

#Sourcing the tinyos environment variable setup script

source /opt/tinyos-2.1.1/tinyos.sh

Si vous exécutez généralement TinyOS à partir de CVS et nécessitent seulement l'installation de **toolchain**, vous pouvez installer le paquet **tinyos-required** au lieu de **tinyos**.

Pour l'instant il est préférable de jouer la sécurité et de supprimer tous les anciens paquets **TinyOs** avant d'installer les nouveaux.

Aussi, vous avez utilisé le référentiel TinyOs Debian dans le passé, garder à l'esprit que tous les outils ont été mis à jour pour TinyOs-2.1.1, mais toujours travailler avec toutes les anciennes versions de TinyOs ainsi.

Programmation :

Branchez le capteur Telos B à votre ordinateur avec un câble USB et ouvrez une console sous UBUNTU . Avant de compiler et d'installer le programme, il faut tester si le capteur a été bien détecté et reconnu par le système. Entrez la commande suivante pour lister tous les capteurs branchés :

root@amina :~$ motelist

Si tout se passe bien, le système va vous retourner les informations du capteur branché :

```
Reference   CommPort        Description
----------- --------------- -------------------------
UCC89MXV    /dev/ttyUSB2    Telos (Rev B 2004-09-27)
```

Voilà, il a bien détecté qu'un capteur Telos B est branché sur le port USB numéro 2. Quand vous testez cette commande sur votre machine, le numéro pourrait être différent (0, 1, 3 etc.). Maintenant on peut compiler et installer le programme à l'aide de Makefile fourni :

root@amina :~$ make telosb install /dev/ttyUSB2

La traduction de cette commande est : Compiler le programme pour la plateforme Telos B, et l'installer vers le port /dev/ttyUSB2.

Bibliographie

[1] Dhib Aya, " Routage avec QOS temps réel dans les réseaux de capteurs ", Master, Ecole supérieure de communication de TUNIS, 2006-2007

[2] David Martins, "Sécurité dans les réseaux de capteurs sans fil stéganographie et réseaux de confiance", Thèse de doctorat, Université de FRANCHE-COMTÉ, 2010.

[3] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. "Wireless sensor networks : a survey". Computer Networks 38, Elsevier Science, pp. 393–422, 2002.

[4] Lehsaini Mohamed. " Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique ", Thèse de doctorat, Université A.B Tlemcen et Université de Franche-Comté U.F.R Sciences et Techniques École Doctorale SPIM, 2009.

[5] Kacimi Rahim. " Techniques de conservation d'énergie pour les réseaux de capteurs sans fil ", Thèse de Doctorat, Institut National Polytechnique de Toulouse, Septembre 2009.

[6] A.Bharathidasan,V.Anad Sau Ponduru," Sensor networks : An overview ", département d'informatique de Californie.

[7] L.Raileanu,F.Nastaran, " les réseaux de senseurs ", haute ecole d'ingénieurie et de géstion du Couton de Vaud, 10/01/2006.

[8] Vijay Raghunathan, Curt Schurgers, " energy-aware wireless microsensor networtks ", mars 2002.

[9] Cesare Alippi, Giuseppe Anastasi, Cristian Galperti, Francesca Mancini, and Manuel Roveri. " Adaptive sampling for energy conservation in wireless sensor networks for snow monitoring applications ". In Proceedings of the 4th IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS'07), pages 1#6, Pisa, Italy, October 2007.

[10] Wei Ye, John Heidemann, and Deborah Estrin. " Medium access control with coordinated adaptive sleeping for wireless sensor networks ". IEEE/ACM Transactions on Networking, 12(3) :493-506, 2004.

[11] Sameer Tilak, Nael.B, Abu-Ghazaleh, and Wendi Heinzelman." A taxonomy of wireless micro-sensor network models ". SIGMOBILE Mobile Computing and Communications Review, 28_36, 2002. 12, 21.

[12] Samir Othmani, " Protocole de sécurité pour les réseaux de capteurs sans fil ", Magistère, université de BATNA, 15/07/2010.

[13] Kaci Bader, " Détection d'intrusion sans les réseaux de capteurs dans fils " ; Master, université de RENNES, 2010.

[14] D.Halperin, T.S.Heydt Benjamin, B.Ransford, S.S.Clask, " Pacemakers and implantable cardiac defibrilators ", Okland, 2008.

[15] Jing Deng, Richard Han, and Shivakant Mishra. " Countermeasures against trafic analysis attacks in wireless sensor networks ". In SECURECOMM '05 : Proceedings of the First International Conference on Security and Privacy for Emerging Areas in Communications Networks, pages 113_126, Washington, DC, USA, 2005. IEEE Computer Society.

[16] H. Wang and Q. " Li. Efficient implementation of public key cryptosystems on mote Sensors ". In International Conference on Information and Communication Security, LNCS 4307, pp.519-528, 2006.

[17] A. Liu and P. 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, pp.245-256, Washington, USA, 2008.

[19] Chakib Bekara and Maryline Laurent-Maknavicius. " A new resilient key management protocol for wireless sensor networks ". In Damien Sauveron, Constantinos Markantonakis, Angelos Bilas, and Jean-Jacques Quisquater, editors, WISTP, volume 4462 of Lecture.Notes in Computer Science, pages 14_26. Springer, 2007.

[19] Victor Miller, Neal Koblitz, " Securitu of using elliptic curves ", 1985.

[20] Abderahman Nitay, " Introduction Au courbes Elliptique ", France-Rabat, 29 octobre 2002.

[21] S.Dupensne, " Cryptographie sur les courbes elliptiques " ; école des jeunes chercheurs, 5 avril 2005.

[22] Thomas Izard, " Opérateurs arithmétiques parallèles pour la cryptographie asymétrique ", Thèse de Doctorat, Université de Montpelier II, Décembre 2011.

[23] Yambo Shou, " Parallelisation de la multiplication scalaire par la cryptographie sur les courbes elliptiques dans les réseaux de capteurs ", 1ér février 2012.

[24] Hagler Michael, " Courbe elliptiques et cryptographie ", 19 février 2006.

[25] Gérard CHALHOUB, " Les réseaux de capteurs sans fil ", Clermont Université, IUT de Clermont-Ferrand France,chalhoub@iut.u-clermont1.fr.

[26] Messai Mohamed Lamine, " Sécurité dans les Réseaux de Capteurs Sans-Fil ", Magistère, Université Abderrahmane Mira de Bejaia, 2007

[27] `www.wikipedia.org/wiki/NesC`

[28] Yanbo shou, Hervé Guyennet, " Mini tutoriel de la programmation en NesC ",17 novembre 2011.

[29] V.S. Miller, " Uses of elliptic curves in cryptography ". In Advances in Cryptology-CRYPTO, volume 218 of LNCS, pages 417_428. Springer, 1986.

[30] N. Koblitz." Elliptic curve cryptosystems ". Mathematics of Computation, 48 :203_209, 1987.

[31] N. Gura, A. Patel, A. Wander, H. Eberle, and S.C. Shantz. " Comparing elliptic curve cryptography and rsa on 8-bit cpus ". In Cryptographic hardware and embedded systems CHES 2004 : 6th international workshop, Cambridge, MA, USA, August 11-13, 2004 : proceedings, volume 6, page 119. Springer-Verlag New York Inc, 2004.

[32] Nicolas Méloni, " Arithmétique pour la Cryptographie basée sur les Courbes Elliptiques ", thèse de doctorat, 24 septembre 2007.

[33] P. Montgomery. " Speeding the pollard and elliptic curve methods of factorization. Mathematics of Computation ", 48 :243_264, 1987.

[34] `www.tinyos.net/tinyos-1.x/doc/`