Table des matières

Chapitre 3 : Elaboration de la solution

Liste des figures

Chapitre1

Chapitre2

Chapitre 3

Liste des Tableaux

Introduction Générale

Les avancées technologiques et techniques opérées dans les réseaux sans fil, dans la micro-fabrication et dans l'intégration des microprocesseurs ont fait naître une nouvelle génération : les réseaux de capteurs. Cette dernière promet de révolutionner notre vie, notre travail et notre façon d'interagir avec l'environnement physique qui nous entoure. Imaginons un ensemble de petits appareils électroniques, autonomes, forment un réseau de capteurs sans fil capable de superviser une région, un phénomène d'intérêt, ou encore fournir des informations utiles par la combinaison des mesures prises des différents capteurs et les communiquer via un support sans fil à un centre de contrôle distant.

Les nœuds capteurs composant le réseau 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. Mais l'absence de sécurité physique, et la nature vulnérable des communications radios sont des caractéristiques qui augmentent les risques d'attaque.

Les réseaux de capteur sans fil sont sujets à différents types de menaces telles que l'interception des données envoyées et reçues par un support sans fil permet de les modifier ou de les rejouer. L'intrus peut également injecter, saturer ou endommager les équipements du réseau. Dans cette optique, un protocole de sécurité doit pouvoir être établi avec peu d'influence sur la performance globale du réseau, tout en fournissant les différents services de sécurité pour chaque type d'application.

L'agrégation des données est une approche très intéressante pour réduire la charge des communications. Elle consiste à traiter les données recueillies par chaque capteur au niveau d'un nœud appelé agrégateur. Seulement le résultat produit sera transmis à la station de base. De cette manière la quantité de données communiquées dans le réseau peut être diminuée, ce qui réduit par conséquent la consommation de la bande passante et l'épuisement d'énergie des capteurs. Cependant, garantir la sécurité conjointement à des techniques d'agrégation est très difficile parce qu'un nœud capturé pose un double problème. Il compromet la confidentialité des données (possibilité d'écoute) et leur disponibilité (possibilité d'attaque du type déni de service). Egalement un nœud d'agrégation compromis met en danger toutes les mesures collectées qui font parti de l'agrégat dont le nœud est responsable.

Nous nous focalisons dans notre travail sur le problème de confidentialité pour lequel nous pensons qu'il manque encore des solutions efficaces. Nous avons proposé un algorithme nommé FTSA (Fault-Tolerant Secure data Aggregation in wireless sensor networks) qui combine deux algorithmes afin de tirer partie des avantages des deux algorithmes. Le but de notre travail est de résister contre l'attaque par capture, ainsi l'attaquant n'aura jamais accès aux données même s'il capture un ou plusieurs nœuds capteurs.

L'idée principale de notre proposition est de diviser la donnée captée de chaque capteur en plusieurs parts selon le principe de la cryptographie à seuil, de chiffrer chaque part et d'envoyer ces parts chiffrées via plusieurs chemins. De ce fait l'attaque même si elle capture un ou plusieurs nœuds ne peut jamais récupérer les données d'origines. L'utilisation du chiffrement homomorphique constitue un puissant outil puisque l'agrégateur va aggréger des données chiffrées dont il n'a pas accès, ce qui a comme avantage d'assurer la confidentialité même si l'agrégateur a été compromis. L'utilisation d'une architecture clusterisée du réseau contribue à la diminution de l'overhead (nombre de messages émis et reçus), ce qui a pour effet d'économiser l'énergie des capteurs et ainsi prolonger la durée de vie du réseau de capteurs.

Nous avons également implémenté notre algorithme sur un prototype d'un réseau de capteurs constitué de capteurs de type telosb. Ce qui nous a permis d'appréhender le système d'exploitation Tinyos, le langage NesC ainsi que d'autres outils de programmation.

Notre mémoire s'articule autour de trois chapitres :

Le premier chapitre présente une introduction aux réseaux de capteurs sans fil ainsi que leur fonctionnement, leurs applications potentielles et aussi leurs principales caractéristiques.

Le second chapitre survole les problèmes de sécurités des données agrégées dans les réseaux de capteurs sans fil, les solutions proposées et leurs classifications avec la description des publications existantes.

Le dernier chapitre constitue le cœur de notre travail. Nous proposons un nouvel algorithme nommé FTSA pour renforcer la confidentialité des données agrégées dans les réseaux de capteurs clustérisés.

Et enfin, nous terminons notre mémoire par une conclusion générale et perspective.

Chapitre1
Réseaux de capteurs sans fil

1. Introduction

Au cours de ces dernières années, la technologie des réseaux sans fil n'a cessé de croître grâce aux développements technologiques, de ce fait un nouveau domaine de recherche s'est créé : *les réseaux de capteurs sans fil (RCSF).*

Les réseaux de capteurs sans fil sont constitués de nœuds (capteurs) qui possèdent des capacités particulières comme l'auto-organisation, rapidité de déploiement, tolérance aux erreurs et leur faible cout.

2. Capteur sans fil

Un capteur ou *mote* en anglais est un petit dispositif électronique capable de mesurer une grandeur physique environnementale telle que la température, la pression, l'humidité, etc., et de la communiquer à un centre de contrôle via une station de base, ses inconvénients sont [1]:

- Faible capacité de calcul.
- Faible capacité de mémoire.
- Faible capacité d'énergie.
- Faible capacité de communication.

3. Composants d'un capteur sans fil

Un capteur sans fil est doté, principalement, de quatre unités (fig.1.1).

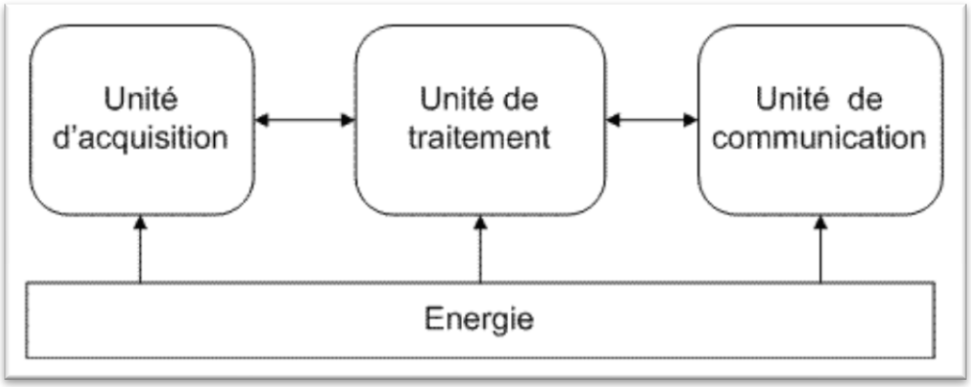


Figure1.1. Composants d'un capteur sans fil [13].

- **Unité d'acquisition** est composée d'un dispositif qui va obtenir des mesures analogiques sur les paramètres environnementaux et les transformer en signaux numériques, et d'un convertisseur Analogique/Numérique qui va convertir l'information relevée et la transmettre à l'unité de traitement.

- **Unité de traitement** est composée de deux interfaces, une interface pour l'unité d'acquisition et une interface pour l'unité de transmission.

Cette unité est également composée d'un processeur et d'un système d'exploitation spécifique. Elle acquiert les informations en provenance de l'unité d'acquisition et les envoie à l'unité de transmission.

- **Unité de transmission** est responsable de toutes les émissions et réceptions de données via « un médium sans fil » [8].

- **Unité de contrôle d'énergie** est responsable de la gestion de l'énergie et de l'alimentation de tous les composants du capteur. Elle consiste, généralement, en une batterie qui est limitée et irremplaçable, ce qui a rendu l'énergie comme principale contrainte pour un capteur [13].

4. Réseaux de capteur sans fil

4.1. Définition

Un réseau de capteur sans fil est constitué de centaines ou de milliers de nœuds afin de collecter ou transmettre des données et les envoyer vers un ou plusieurs points de collecte appelés « puits » ou « Sink ». Ces derniers sont alimentés par des piles et sont typiquement déployés de façon aléatoire dans des endroits ou l'accès est difficile appelé « champs de captage » ou « zone de couverture ».

4.2 Architecture d'un RCSF

4.2.1. Composition

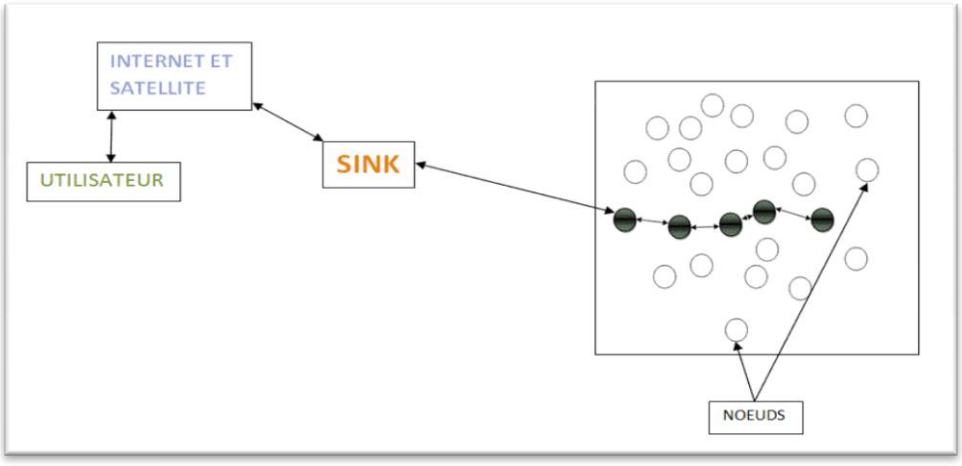


Figure1. 2. Composition d'un RCSF [3].

4.2.2. Comment collecter les informations

Il y a deux méthodes pour collecter les informations d'un réseau de capteurs [16] :

a. A la demande

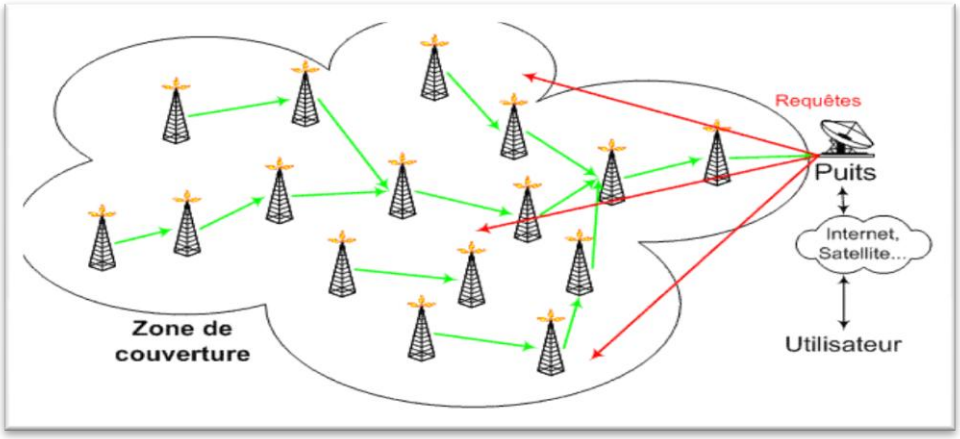


Figure1. 3. Collecte d'information à la demande.

Lorsque nous souhaitons avoir l'état de la zone de couverture à un moment T, le puits émet des brodcasts vers toute la zone pour que les capteurs remontent leur dernier relevé vers le puits. Les informations sont alors acheminées par le biais d'une communication multi-sauts (fig.1.3).

b. Suite à un événement

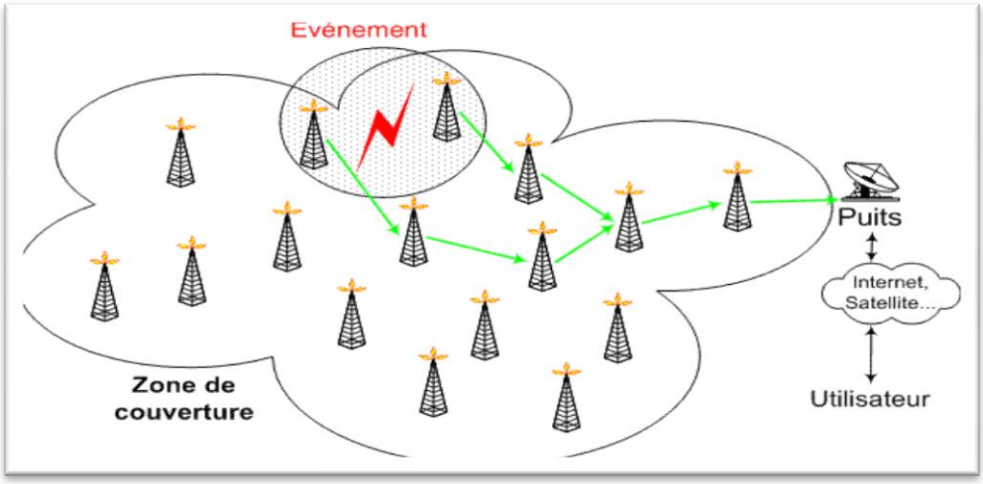


Figure1. 4. Collecte d'information suite à un événement.

Un événement se produit en un point de la zone de couverture (changement brusque de température, mouvement...), les capteurs situés à proximité remontent alors les informations relevées et les acheminent jusqu'au puits (fig.1.4).

5. Fonctionnement des RCSF

Les données captées par les nœuds sont acheminées grâce à un routage multi saut à un nœud considéré comme un "point de collecte". Ce dernier peut être connecté à l'utilisateur du réseau (via Internet, un satellite ou un autre système).

En outre, le déploiement d'un réseau de capteurs exige la *fidélité d'acquisition* (sensing fidelity) c'est-à-dire s'il y aura un événement il faut qu'il soit détecté par au moins un capteur et la *fidélité de routage* (routing fidelity) c'est-à-dire qu'il doit exister au moins un chemin entre le capteur qui a détecté l'événement et la station de base [2].

6. Applications des RCSF

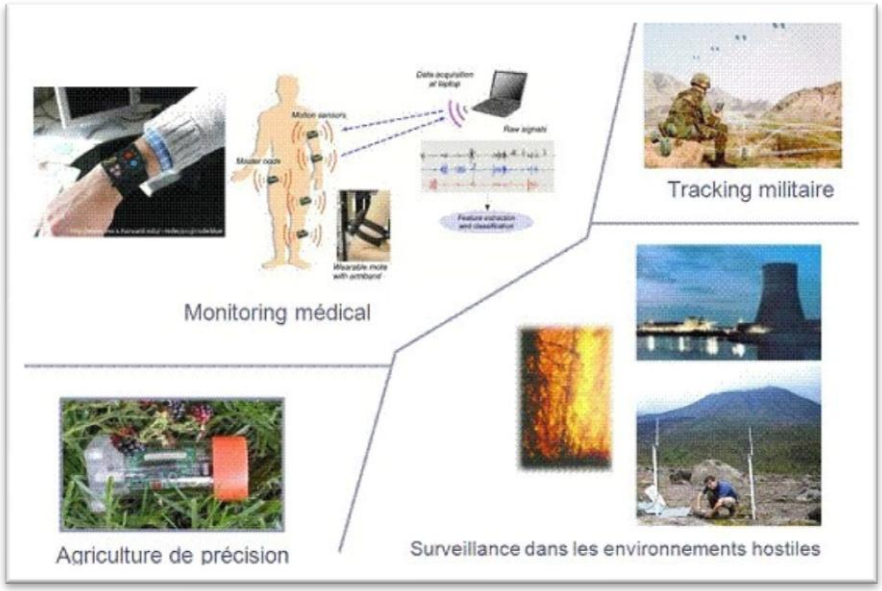


Figure1. 5. Application des RCSF [4].

a. Application militaire

Les réseaux de capteurs sont utilisés pour la surveillance, la détection des intrusions, la détection des substances dangereuses, la communication, la reconnaissance et le ciblage, Ils peuvent être rapidement déployés et utilisés pour la surveillance des champs de bataille afin de fournir des renseignements concernant l'emplacement, le nombre, le mouvement, et l'identité des soldats et des véhicules, ou bien encore pour la détection des agents chimiques, biologiques et nucléaires [13].

b. Application environnementale

les applications d'environnement qui peuvent bénéficier de la technologie des réseaux de capteurs sans fil, on peut citer par exemple, le cheminement des mouvement

d'oiseaux; les macro-instruments utilisés pour la surveillances des terrains à grande échelle et les explorations planétaires; la détection chimique et biologique [13].

c. Application médicale

Les réseaux de capteurs ont aussi des développements dans le domaine de diagnostic médical. Par exemple, des micro-caméras sont capables, sans avoir recours à la chirurgie, de transmettre des images de l'intérieur d'un corps humain [6]. On peut aussi surveiller des malades à distance et intervenir le plus rapidement possible.

d. Application agricole

Dans les champs agricoles, les capteurs peuvent être semés avec les graines. Ainsi, les zones sèches seront facilement identifiées et l'irrigation sera donc plus efficace [6].

7. Système d'exploitation pour RCSF

De nombreux systèmes d'exploitation pour capteurs existent, parmi lesquels :

- **Contiki** est un système développé et soutenu par l'Institut d'Informatique Suédoise. Adam Dunkels est l'initiateur du projet.
- **RETOS** est un système développé par l'Université Yonsei en Corée.
- **LiteOS** est un système développé et soutenu par l'Université américaine de l'Illinois à Urbana-Champaign.

Nous nous intéressons au TinyOS qui est un système d'exploitation open-source conçu pour des réseaux de capteurs sans fil. Nous parlerons de ce système dans le chapitre 3.

8. Conclusion

Dans ce chapitre nous avons présenté les réseaux de capteurs sans fil qui représentent un domaine très utile et une technologie récente. Ces derniers ont conduit à de nombreuses nouvelles applications réalisées dans des différents domaines tels que le domaine militaire, environnemental, médical, etc. Ces applications incluent souvent le suivi des informations sensibles telles que les mouvements de l'ennemi sur le champ de bataille ou la détection des feux dans des grandes zones forestières ou encore la détection de cancer par exemple. La sécurité est donc importante dans les réseaux de capteurs.

Dans le chapitre suivant nous aborderons les problèmes de sécurité liés aux wireless sensor network (WSN), en particulier nous nous intéressons à sécurité de l'agrégation de données

Chapitre2
Sécurité de l’agrégation des données

1. Introduction

La sécurité a une très grande importance pour plusieurs applications intéressantes des réseaux de capteurs sans fil telles que la surveillance des champs de bataille, la détection des feux de forêts ou la mesure de la température et de la pression dans les canalisations d'huile. La fiabilité de ces systèmes est un enjeu majeur, et l'existence des failles de sécurité représente un risque non toléré.

Les réseaux de capteurs sont plus vulnérables aux attaques par rapport aux autres types de réseau du fait qu'ils fonctionnent sous diverses contraintes et dans différents environnements (hostile, non surveillé, …). Ainsi que les différentes caractéristiques des réseaux de capteurs (densité importante, une capacité limitée de calcul et de stockage, une communication sans fil, une ressource d'énergie limitée, ….) font que l'utilisation des mesures classiques de sécurité soit restreinte.

2. Services de sécurité

Les RCSF, étant des entités fournissant des informations, doivent s'appuyer sur un certain nombre de services afin d'assurer un transfert sécurisé des données. Parmi ces services :

- Confidentialité
- Intégrité
- Authentification
- Non répudiation

3. Classification des attaques

Les attaques sur les réseaux de capteurs peuvent être classifiées dans les catégories suivantes [12] :

3.1 Attaques passives

Les attaques passives "eavesdropping" se limitent à l'écoute et l'analyse du trafic échangé. Ce type d'attaque est plus facile à réaliser et il est difficile de le détecter puisque l'attaquant n'apporte aucune modification sur les informations échangées. L'intention de l'attaquant peut être la connaissance des informations confidentielles ou bien la connaissance des nœuds importants dans le réseau. En analysant les informations de routage, l'attaquant va se préparer à mener ultérieurement une action précise.

3.2. Attaques actives

Dans les attaques actives, un attaquant tente de supprimer ou modifier les messages transmis sur le réseau. Il peut aussi injecter son propre trafic ou rejouer d'anciens messages pour perturber le fonctionnement du réseau ou provoquer un déni de service.

3.3. Attaques externes

Dans le cas de l'attaque externe, le nœud attaquant n'est pas autorisé à participer dans le réseau de capteurs. Des techniques de cryptographie et d'authentification protègent l'accès au réseau à ce type d'attaquant. Cependant ce dernier peut uniquement déclencher des attaques passives telles que l'écoute clandestine, ou l'attaque par rejeu.

3.4. Attaques internes

L'attaque interne est considérée comme la plus *dangereuse* du point de vue sécurité. Puisque l'attaquant qui capture un nœud, peut lire sa mémoire et avoir accès à son matériel cryptographique tel que les clés et par conséquent peut s'authentifier comme un nœud légitime et émettre des messages aléatoires erronés sans qu'il soit identifié comme intrus, puisqu'il utilise des clés valides. Les méthodes cryptographiques s'avèrent donc inefficaces pour ce genre d'attaque. Il est donc nécessaire d'utiliser d'autres méthodes complémentaires telles que les systèmes de monitoring et les systèmes de réputations.

4. Obstacles de la sécurité

La sécurisation des réseaux de capteurs reste un problème difficile pour les raisons suivantes :

- **Ressources limitées en mémoire, espace de stockage et puissance de calcul**

Le capteur est un composant miniature avec un espace mémoire et de stockage limité, et avec une faible vitesse de calcul. Sur ce capteur, nous devons installer le code du système d'exploitation et les applications. Donc le code de la sécurité et les données relatives doivent être très petits. Afin d'établir un mécanisme efficace de sécurité, il est nécessaire de limiter le nombre d'instructions de l'algorithme [13].

- **Ressources limitées en énergie**

Il faut retenir que l'opération de transmission de données est extrêmement gourmande en énergie et donc le moindre ajout de MAC, de numéro de séquence, de

vecteur d'initialisation…etc., dans les paquets de données a un coût très élevé qui va affecter la durée de vie des capteurs [12].

- **Communication non fiable**

Certainement, la communication est un autre obstacle pour la sécurité des capteurs :

- **Transfert non fiable**

Les paquets peuvent être endommagés en raison des erreurs de transmission ou supprimés dans les nœuds fortement encombrés. D'une manière primordiale, le protocole doit disposer d'une gestion d'erreur appropriée sinon il serait possible de perdre des paquets critiques de sécurité tels que les paquets contenant les clés cryptographiques [14].

- **Collisions**

Même si le canal est fiable, la communication ne peut pas toujours l'être. Ceci est dû à la nature d'émission des paquets dans les réseaux de capteurs sans fil (*broadcast*).

Si les paquets se rencontrent lors du transfert, les collisions se produisent et le transfert lui même échouera [14].

5. Solution proposée

Dans un capteur, la problématique principale concerne la consommation d'énergie. En effet, ces derniers doivent rester opérationnels le plus longtemps possible, dans des conditions parfois difficiles (ex : lâchés par avion sur les parois d'un volcan). Comme il n'est pas possible de recharger leur énergie ni changer les piles par exemple (on ne sait pas toujours où se trouvent les capteurs), il est nécessaire d'économiser au maximum l'énergie consommée par ces derniers. Nous estimons que la transmission des données d'un capteur représente environ 70% de sa consommation d'énergie. De plus, les réseaux de capteurs étant assez denses en général, cela signifie que des nœuds assez proches en terme de distance (voisins) peuvent capter les mêmes données (température, pression, humidité équivalentes par exemple) et donc il apparaît nécessaire d'introduire le mécanisme d'agrégation de données afin d'éviter la duplication d'information au sein du réseau de capteurs et donc de préserver leur énergie et pour enfin augmenter la durée de vie du réseau [6].

6. Agrégation dans les RCSF

L'agrégation est l'une des mécanismes utilisées pour réduire la charge du trafic acheminé dans le réseau ainsi que la consommation d'énergie. Pour bien comprendre la notion d'agrégation, nous allons citer deux exemples :

a. Sans agrégation

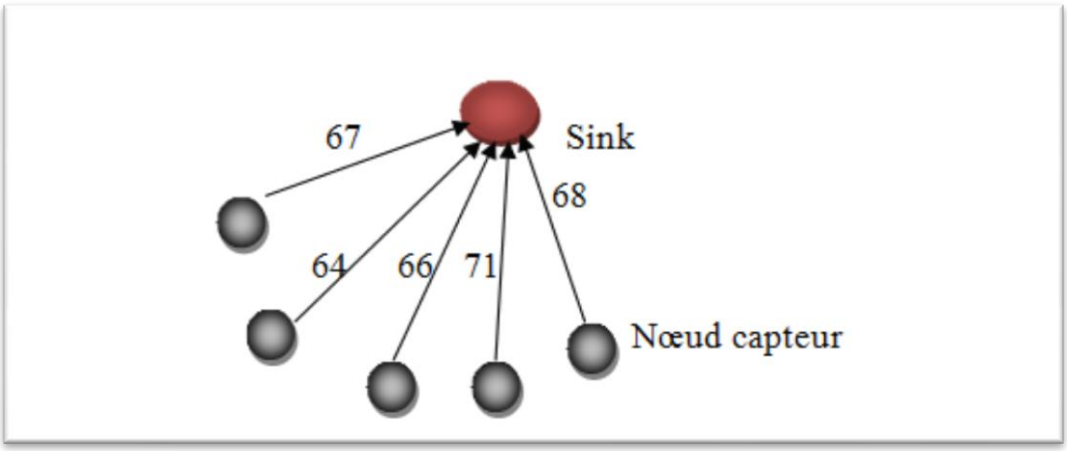


Figure 2. 1. Envoi des températures vers le Sink.

Prenons l'exemple d'un réseau qui est déployé pour mesurer la température moyenne dans une zone géographique donnée. Chaque capteur détecte la température de sa zone et envoie périodiquement ses données vers le puits. Par conséquent, si le réseau est constitué de n nœuds, le puits recevra à chaque période de mesure n messages! (fig.2.1). Ensuite le Sink enverra ces mesures vers la SB (fig.2.2) [5].

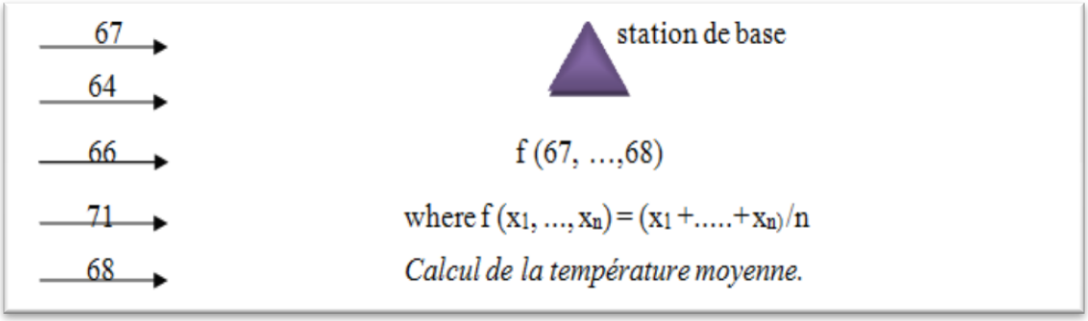


Figure 2. 2. Réception des températures par la station de base.

b. Avec agrégation

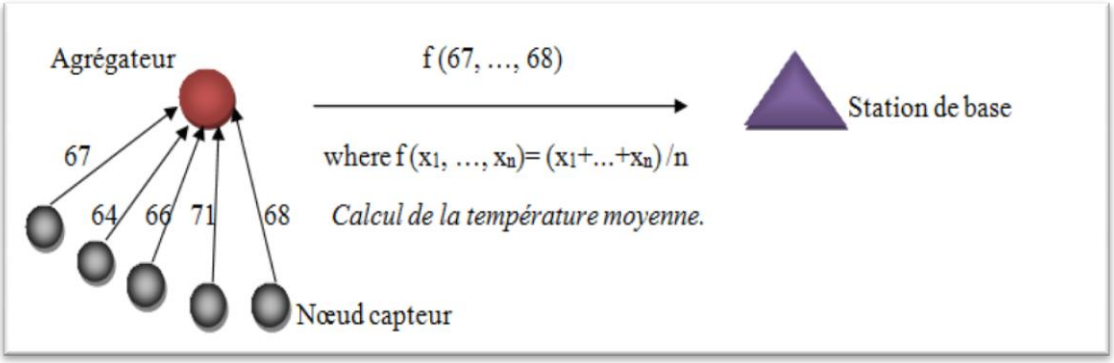


Figure 2. 3. Utilisation de l'agrégation.

Pour réduire le nombre de messages, nous utilisons le même principe que la figure2.2, mais au lieu que l'agrégateur n'envoie chaque mesure à la SB, il calcule la moyenne de ses valeurs grâce à la formule :

Agrégation = somme des données collectées / nombre de capteurs

Et donc la SB recevra un seul message qui contiendra la somme des messages au lieu de n messages (fig.2.3).

7. Sécurisation des données agrégées

Nous avons deux approches:

7.1. Approche saut à saut

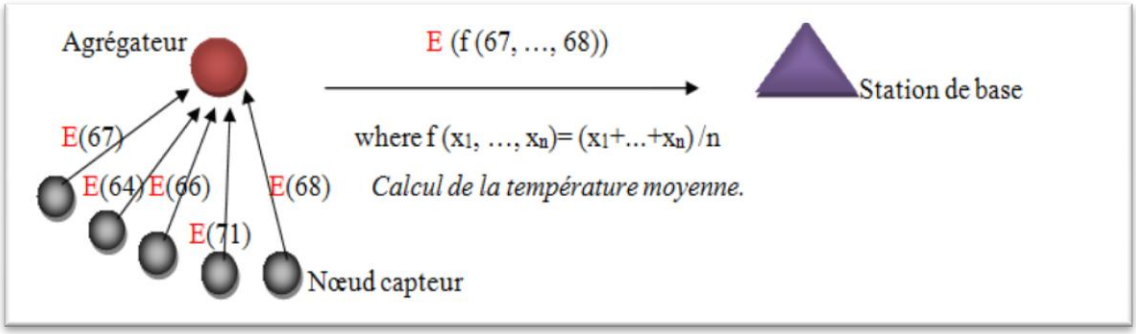


Figure 2. 4. La sécurité saut à saut.

- Chaque nœud chiffre et authentifie ses données avec une clef qu'il partage avec un agrégateur local.
- Chaque agrégateur déchiffre et vérifie authentification, calcule la moyenne et chiffre le résultat!!
- **Problème de cette approche**

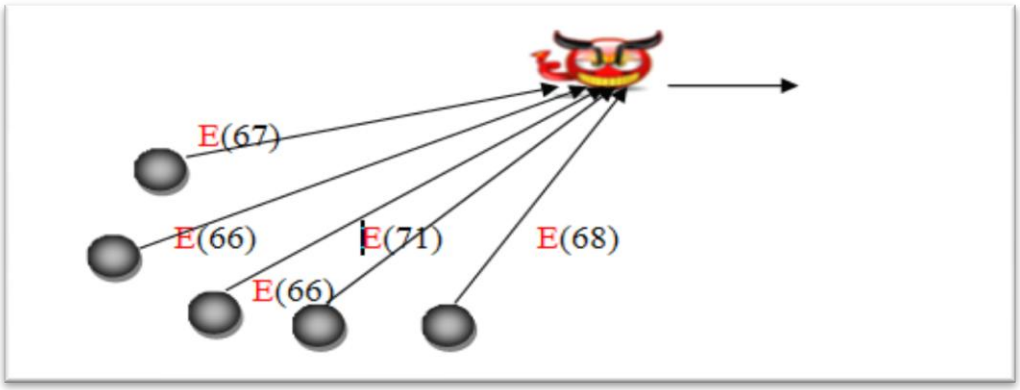


Figure 2. 5. Inconvénient de l'approche saut à saut.

L'inconvénient de cette approche est que si l'agrégateur sera compromis alors l'utilisateur recevra de faux résultats.

7.2. Approche bout en bout

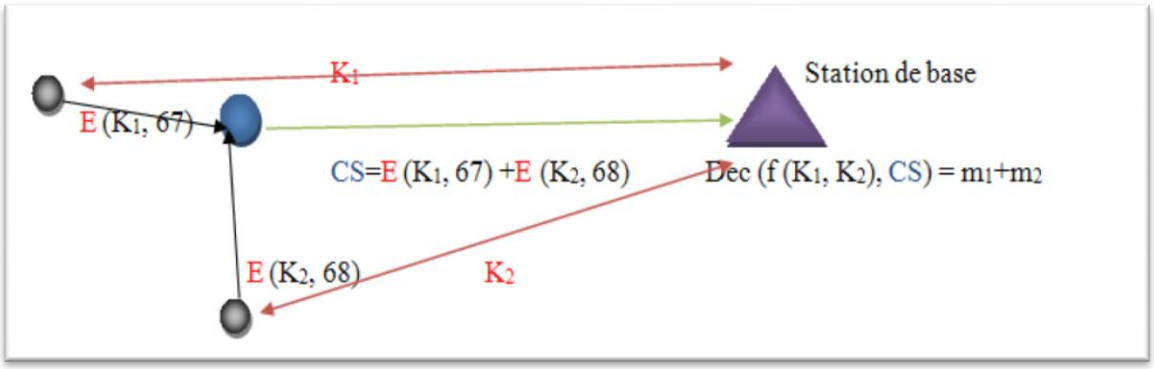


Figure 2. 6. La sécurité bout en bout.

- Chaque nœud chiffre ses données avec une clef qu'il partage avec la station de base (pas l'agrégateur!!).
  - Gestion de clefs simplifiée et sécurité forte.
- Agrégateur manipule les données chiffrées, et n'a jamais accès aux données en clairs.
  - Nous concluons qu'avec cette approche, l'agrégateur additionne les données chiffrées et à la station de base, déchiffre et retrouve la somme des données! Donc l'agrégateur compromis n'a plus accès aux données!
  - Cette approche est efficace car l'agrégateur ne doit pas déchiffrer et chiffrer les résultats! [18].

8. Exigences de sécurité dans l'agrégation des données

Le but de l'agrégation des données est de [9] :

- **Overhead :** Réduire les communications.
- **Scalabilité :** Il faut maintenir la sécurité dans les petits et les grands réseaux.
- **Flexibilité :** Les techniques d'agrégation doivent fonctionner dans n'importe quel type de réseau.
- **Efficacité :** Il faut que le résultat final soit exact.
- **Généralité :** Le schéma d'agrégation doit s'appliquer sur différentes fonctions : MAX, MIN, AVG,…etc.
- Pour renforcer la sécurité dans l'agrégation des données il faut assurer:

- Intégrité des données

Il faut s'assurer que les données ne sont ni altérées ni supprimées soit volontairement ou accidentellement.

Supposons qu'un schéma d'agrégation de données sécurisé se concentre uniquement sur la confidentialité des données. Un adversaire à proximité du point agrégateur sera en mesure de changer le résultat agrégé envoyé à la station de base en ajoutant quelques fragments ou de manipuler le contenu du paquet sans détection. En outre, même en l'absence d'un adversaire, les données pourraient être endommagées ou perdues à cause de l'environnement sans fil [15].

- Confidentialité des données

Il faut s'assurer que « les données ne sont compréhensibles que par les entités qui partagent un même secret. » [7]. Si par exemple, un réseau est chargé de surveiller les attaques surprises alors il est nécessaire d'assurer la confidentialité de l'information pour réussir l'attaque. Pour cela, le réseau de capteur qui utilise l'agrégation doit protéger les données agrégées.

- Authentification des données

Il faut vérifier l'identité d'un envoyeur de données sur le réseau (identification) afin d'éviter l'injection des données non autorisées [6].

Dans l'agrégation de données sécurisées, à la fois l'identification et l'authentification sont importantes pour assurer le transfert des données entre les capteurs légitimes [10].

« L'authentification est un mécanisme qui permet de séparer les amis des ennemis » [7].

- Disponibilité et fraicheur des données

Étant donné que les réseaux de capteurs sont utilisés pour surveiller les événements sensibles au facteur du temps, il est donc important d'assurer que les données fournies par le réseau soient à jour (fraîches) et disponibles tout le temps. Cela signifie que l'adversaire ne peut pas rejouer les anciens messages à l'avenir. Cependant, les exigences de sécurité des données agrégées doivent être soigneusement mises en œuvre pour éviter la consommation de beaucoup d'énergie. S'il n'y a plus d'énergie, les données ne seront plus disponibles.

9. Attaques contre l'agrégation de données

Les RCSF sont vulnérables à différents types d'attaques en raison de la nature du support de transmission (*broadcast*), donc l'agrégation des données doit se faire en

toute sécurité afin d'éviter une lecture trompeuse de l'état de l'environnement à surveiller. L'objectif d'un attaquant est soit de falsifier le résultat généré par l'agrégation de chaque groupe, et de faire accepter les résultats par la station de base, soit d'accéder à des résultats confidentiels. Le moyen le plus facile pour un attaquant de réaliser l'attaque par falsification, serait de compromettre le nœud agrégateur, puis générer un résultat arbitraire. L'autre solution plus complexe serait de compromettre une partie importante de capteurs afin de générer une quantité suffisante de fausses lectures. Dans cette section, nous discutons sur les attaques qui pourraient affecter l'agrégation dans les réseaux de capteurs [17] :

9.1. Compromission d'un nœud

Nous pouvons avoir physiquement accès aux capteurs du fait de leur dispersion en pleine nature bien souvent. Un attaquant peut donc éventuellement extraire des informations de ces capteurs puisqu'ils sont à sa disposition physique [6].

9.2. Attaque par déni de service (DoS)

Un nœud compromis peut arrêter d'agréger et d'acheminer les données, ce dernier empêche la station de base de récupérer des informations au sujet de plusieurs nœuds dans le réseau. Si l'attaquant continue d'échanger des messages de routage, ce problème peut se révéler difficile à résoudre. De cette façon, un agrégateur malintentionné peut rendre le réseau inutilisable [12].

9.3. Attaque sybil

Dans l'attaque Sybille, de nouveaux capteurs (malicieux) peuvent prendre l'identité d'autres capteurs légitimes dans le réseau [10]. Cette attaque affecte les algorithmes d'agrégation de différentes manières [11]. Premièrement, un adversaire pourra créer des identités multiples pour générer des votes additionnels dans l'élection d'un agrégateur. Deuxièmement, le résultat de l'agrégat peut être affecté si l'adversaire est capable de générer des entrées multiples avec des valeurs collectées différentes. Troisièmement, quelques algorithmes utilisent des nœuds témoins pour valider l'agrégation des données et ces dernières sont valides si seulement n témoins acceptent le résultat de l'agrégat. Cependant, un adversaire peut déclencher une attaque Sybille et génère n identités de témoins ou plus pour modifier le vote et persuader la station de base pour qu'elle accepte le résultat de l'agrégat [12].

9.4. Attaque par renvoi sélectif

Dans le renvoi sélectif, un nœud malicieux agit comme un trou noir (l'attaquant falsifie les informations de routage d'un nœud pour forcer le passage des données par lui-même) et refuse de relayer les parquets. L'adversaire utilise les nœuds compromis pour relayer les messages. Dans le contexte de l'agrégation, tout nœud intermédiaire

compromis a la capacité de déclencher l'attaque du relai sélectif et par conséquent affecter le résultat de l'agrégation [12].

9.5. Attaque par rejeu

Dans ce cas, un attaquant enregistre une partie du trafic du réseau, sans même en comprendre le contenu et le rejouer plus tard pour induire l'erreur d'agrégation et par conséquent, les résultats de l'agrégation seront affectés [9].

9.6 Attaque furtive

Le but de l'adversaire est que l'utilisateur accepte de faux résultats d'agrégation, qui sont significativement différents des résultats réels déterminés par les valeurs mesurées, tout en n'étant pas détectées par l'utilisateur [9].

10. Classification des systèmes d'agrégation sécurisés

Les systèmes d'agrégations sécurisés sont classés selon deux grandes familles : algorithmes basées sur la cryptographie et algorithmes basés sur la confiance:

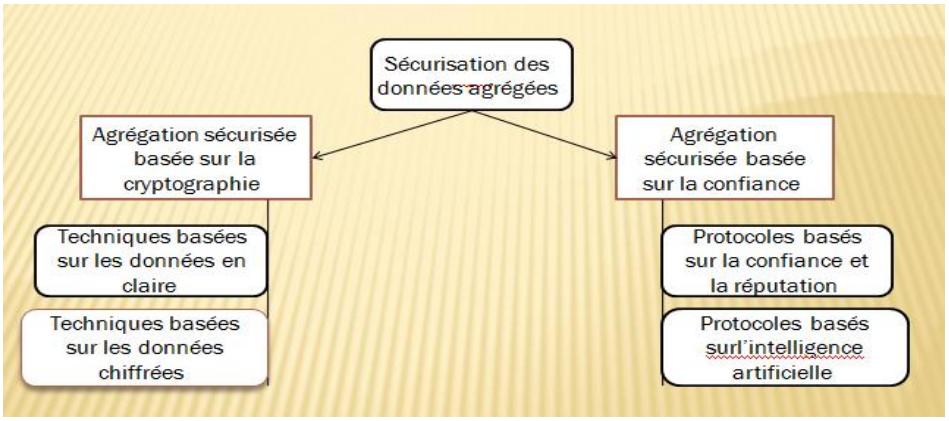


Figure 2. 7. Classification des algorithmes d'agrégation sécurisée [9].

10.1. Agrégation sécurisée basée sur la cryptographie

:Les besoins de sécurité tels que la confidentialité et l'intégrité dans l'agrégation des données deviennent vitaux lorsqu'un réseau de capteur est déployé dans un environnement hostile. La plupart des recherches dans ce domaine se focalisent sur des algorithmes basés sur la cryptographie. Dans certaines applications, il est essentiel de s'assurer que les informations qui sont transmises sur le réseau ne puissent être interceptées et lues que par des personnes autorisées, donc elles doivent donc être chiffrées.

Nous distinguons deux techniques de l'agrégation sécurisée: les techniques basées sur les données en clairs (confidentialité saut par saut) et les techniques basées sur les données chiffrées (confidentialité de bout en bout) [9].

10.1.1. Techniques basées sur les données en claire

Dans les techniques de l'agrégation sécurisée basées sur les données en clair, la fonction d'agrégation est effectuée par l'agrégateur sur des données en clair. Cela veut dire que l'agrégateur doit déchiffrer les données de ses fils (dans une architecture plate) ou des membres de son cluster (dans une architecture clustérisée) avant d'effectuer la fonction d'agrégat tels que la somme ou la moyenne. Plusieurs algorithmes existent tel que:

- **SIA: Secure information aggregation in sensor networks: 2003**

Przydatek et al. [19] ont proposé un algorithme (SIA) qui permet à l'agrégateur d'accepter les données avec une grande probabilité si le résultat de l'agrégat est dans une limite acceptable, ou de rejeter le résultat s'il est hors limite en utilisant des fonctions d'agrégations telles que la médiane, le minimum, le maximum, le comptage et la moyenne.

- **WDA: Secure a witness-based approach for data fusion assurance in wireless sensor networks: 2003**

Du et al. [21] ont proposé (WDA) un protocole d'agrégation de données basée sur des témoins (*Witness*) pour assurer la validation des données envoyées du nœud agrégateur jusqu'à la station de base. Pour qu'on s'assure du résultat de l'agrégat, il faut que l'agrégateur fournisse des preuves qui les obtiennent de la part de plusieurs nœuds témoins. Un nœud témoin est un nœud dédié pour la surveillance et effectue lui aussi l'agrégation, mais ne transmet pas son résultat à la station de base. Le rôle de chaque nœud témoin est de calculer le code d'authentification du message (MAC) du résultat et le transmet au nœud agrégateur comme preuve.

10.1.2. Techniques basées sur les données chiffrées

Une solution de bout-en-bout de l'agrégation sécurisée serait préférable car la compromission d'un nœud ne fournirait aucune information sur l'agrégat ou les données envoyées par les autres nœuds du système. Cette solution est appelée agrégation des données cachée (concealed data aggregation : CDA). Contrairement aux techniques basées sur les données en clair, CDA consiste à effectuer la fonction d'agrégation sur des données chiffrées. Parmi les algorithmes qui existent :

- **HSC: Efficient Aggregation of Encrypted Data Wireless Sensor Network: 2005**

Castellucia et al. [22] ont proposé un protocole simple basé sur le chiffrement par flot et l'homomorphique par l'addition (HSC) qui permet une agrégation de données

efficace. Le nouveau chiffrement remplace l'opération du xor (OUexclusif) par une addition modulaire qui est très appropriée pour les capteurs avec une capacité de calcul réduite. L'agrégation basée sur ce chiffrement peut être utilisée de manière efficace pour le calcul des fonctions statistiques telles que la moyenne, la variance et la déviation standard des données collectées ; tout en achevant un gain considérable de la bande passante. Les inconvénients de ce protocole sont l'overhead important généré dans un réseau non fiable et le passage à l'échelle.

- **Securing wireless sensor networks against aggregator compromises: 2008**

Claveirole et al. [24] ont propose trois nouvelles techniques :

- Agrégation Multi-chemins Secrète (AMS) découper chaque mesure en un nombre donné de parts et envoyer une part par chemin. Pour plus de détail, nous allons améliorer cet algorithme dans le prochain chapitre.

- Agrégation Multi-chemins Dispersée (AMD) accumuler plusieurs mesures dans une mémoire interne avant de l'éparpiller dans plusieurs parts, et envoyer une seule part (c'est-à-dire Une part contient plusieurs sous mesure) par chemin.

- Agrégation Multi-chemins Dispersée avec Authentification (AMD-A) accumuler plusieurs mesures dans une mémoire interne puis ajouter une valeur d'authentification à cette mémoire avant de l'éparpiller dans plusieurs parts.

L'idée principale de ces approches est d'exploiter plusieurs chemins jusqu'à la station de base, tout en s'assurant que les nœuds intermédiaires n'auront pas une connaissance complète des données. En fonction de l'application ou du scénario, une approche peut présenter plus ou moins d'avantages sur l'autre.

10.1.3 Discussion

Le champ de la cryptographie dans le traitement de données dans RCSF est un axe de recherche très prometteur, et introduit beaucoup de challenges. Pour l'instant, l'approche la plus simple dans les WSN est celle qui achève une confidentialité saut-par-saut et une intégrité des données. Cependant, les nœuds capteurs peuvent être capturés et la révélation de leur matériel cryptographique peut mener à la révélation des données originales et des résultats partiaux des agrégats.

Pour surmonter ce problème, des techniques ont été proposées qui exploitent la confidentialité de bout en bout conjointement aux distributions de clés particulières, au chiffrement homomorphique ou au chiffrement à clé publique [9].

10.2. Agrégation sécurisée basée sur la confiance

Dans cette section, nous allons étudier les protocoles proposés basés sur la confiance et la réputation pour sécuriser les données agrégées dans les WSN. Nous les classifions en deux catégories : les protocoles basés sur la confiance et la réputation, et les protocoles basés sur l'intelligence artificielle [9].

10.2.1. Protocoles basés sur la confiance et la réputation

- **Trust-based aggregation in wireless sensor networks: 2005**

Hur et al. [25] ont proposé un protocole d'agrégation basé sur l'évaluation de la confiance locale (LTE) dans les WSN. Le degré de confiance d'un nœud est calculé sur la base de plusieurs facteurs d'évaluation de confiance, tels que la durée de vie de la batterie, le taux de communication, le résultat du captage et le niveau de consistance.

10.2.2. Protocoles basés sur l'intelligence artificielle

- **Robust Trust Mechanisms for Monitoring Aggregator Nodes in Sensor Networks: 2008**

Dans Gursel et al [27], les auteurs ont proposé le premier mécanisme qui combine les statistiques et les techniques d'intelligence artificielles (AIF) pour la détection robuste des nœuds malicieux dans les réseaux de capteurs.

10.2.3 Discussion

Les systèmes basés sur la confiance et la réputation ont récemment été suggérés comme mécanismes efficaces de sécurité pour les environnements ouverts tels qu'Internet. Une recherche considérable a été menée pour modeler et gérer le concept de confiance et la réputation. Quelques travaux de recherches ont démontré que l'estimation de la confiance et la réputation d'un nœud ont une approche efficace dans les environnements distribués pour améliorer la sécurité.

L'évaluation du degré de confiance des nœuds capteur est basée par exemple sur la théorie des probabilités. L'information de confiance est donc utilisée pour déterminer si la donnée reportée par un capteur est correcte ou bien erronée [12].

11. Conclusion

Dans ce chapitre nous avons vu les services de sécurité, la classification des attaques, les obstacles de la sécurité, les solutions proposées ainsi que l'agrégation des données dans les RCSF, la sécurité, les différents types d'attaques, ainsi que la classification des systèmes sécurisés.

Dans le chapitre qui suit, nous présenterons AMS, le chiffrement additif homomorphique, ensuite nous présenterons notre modèle de réseau, ainsi que l'attaque choisie, et enfin, notre solution et ses détails.

Chapitre3
Elaboration de la solution

1. Introduction

Les réseaux de capteurs sont souvent déployés dans des zones non surveillées, ce qui les rend vulnérables à plusieurs attaques dans lesquelles l'intrus peut accéder aux données transitant par un ou plusieurs nœuds capteurs.

Dans le contexte de l'agrégation des données, un nœud capteur compromis peut attribuer les données lisibles à ses voisins. Par conséquent, le résultat partial ou final de l'agrégat peut être lu.

Après une revue d'ensemble de l'état de l'art, ce chapitre présente notre proposition assemblée à deux algorithmes pour sécuriser l'agrégation des données dans les réseaux de capteurs à architecture hiérarchique, surtout pour résoudre le problème de « confidentialité ». En premier lieu nous présenterons les motivations de cette proposition, et par la suite, les détails de notre algorithme et l'analyse de sécurité ainsi que l'implémentation du protocole, et enfin conclusion.

2. Motivations

Dans ce travail, nous nous sommes intéressées aux problèmes de confidentialité du résultat final de l'agrégat causés par le rassemblement des données au niveau des nœuds agrégateurs du réseau.

Parmi les algorithmes utilisés pour assurer la confidentialité dans les RCSF cités précédemment, AMS [24] (Agrégation Multi-chemins Secrète) présente l'avantage de séparer la donnée en plusieurs parts qui sont envoyées immédiatement par plusieurs chemins. Cependant, l'utilisation de ce type d'algorithme permet la tolérance de perte de messages vu qu'il a besoin simplement d'un certain nombre de parts pour reconstruire le message d'origine. En parallèle, il peut causer un problème de confidentialité si l'attaquant capture suffisamment de parts.

L'objectif de notre contribution sera de pallier à cette faille grâce au chiffrement des données qui sera utilisé dès la création des mesures. Pour cela, nous proposons un nouvel algorithme nommé FTSA (Fault- Tolerant Secure data Aggregation in wireless sensor networks), qui se base sur AMS en l'améliorant avec l'utilisation du cryptage additif homomorphique. Nous appliquerons cet algorithme dans un réseau clusterisé à l'encore de AMS utilisé dans une architecture plate, et ceci afin de minimiser la dispersion des parts de données pour une meilleure économie d'énergie et minimiser l'overhead. Ainsi les messages multiples seront acheminés uniquement au niveau du cluster et non dans tout le réseau. Ce qui donne à notre protocole une résistance aux *pertes de messages* (grâce aux parts multiples) et un renforcement de *confidentialité* (grâce au chiffrement homomorphique).

3. Préliminaires

3.1. Notations utilisées

Tableau 3. 1. Notation

NOTATION	DESCRIPTION
SB	Station de base
CH	Agrégateur (Cluster-Head)
CHc	Cluster-Head centralisé (Agrégateur central)
Enc (x)	Cryptage de la donnée x
Dec (x)	Décryptage de la donnée x
K_i^{SB}	Clé partagée entre capteur i et SB
M	Grande valeur avec M > k
n	Nombre de parts à créer
t	Nombre de parts nécessaires pour la reconstruction
Nb_{cap}	Nombre de capteurs
Moy	Moyenne

3.2. Agrégation Multi-chemins Secrète (AMS)

Cette approche utilise la cryptographie à seuil pour créer des parts, ce qui est une approche courante lorsqu'il faut assurer la sécurité malgré des possibilités de capture. Son point fort est qu'elle assure la confidentialité.

Pour créer des parts, chaque capteur divise sa mesure en *n* messages et les envoie sur *p* chemins distincts vers la station de base. Cette dernière peut tolérer jusqu'à *t*-1 nœuds compromis (c.-à-d. qu'un nœud doit disposer d'au moins *t* parts pour reconstruire les mesures). Lorsque le capteur mesure une valeur r_i, il tire aléatoirement un polynôme $P_i(x)$ de degré *t*-1 tel que $P_i(0) = r_i$. Un tel polynôme peut se construire en tirant aléatoirement ses coefficients d'ordre plus grand que zéro : $a_{i,k}$, $\forall k \in [1, t\text{-}1]$ en utilisant ensuite :

$$P_i(x) = r_i + a_{i,1}*x + a_{i,2}*x^2 + \ldots + a_{i,t-1}*x^{t-1}.$$

Chacune des parts est ensuite constituée par les valeurs $P_i(q)$ ($1 <= q <= n$). Le nœud *i* envoie ensuite un message contenant $P_i(q)$ sur chaque chemin *q* [28].

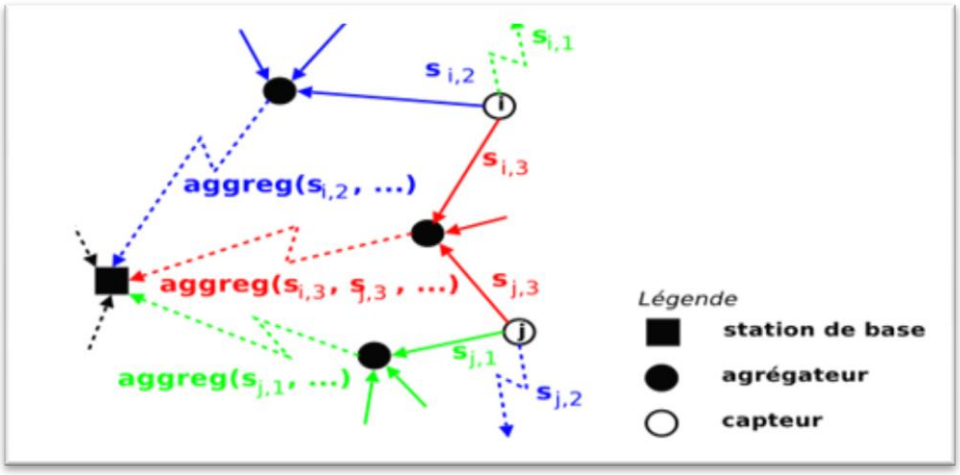


Figure 3. 1 : Principe de base d'AMS [28].

Les nœuds *i* et *j* découpent respectivement leurs mesures dans les parts $s_{i,1}$, $s_{i,2}$, $s_{i,3}$ et $s_{j,1}$, $s_{j,2}$, $s_{j,3}$. Les parts sont ensuite envoyées et agrégées sur différents chemins.

Pour faire la reconstruction, il faut d'abord retrouver P_i grâce à une interpolation polynomiale et ensuite calculer $r_i = P_i(0)$. Cette opération nécessite au moins *t* parts distinctes. Il y a une infinité de polynômes de degré *t*-1 qui passent à travers *t*-1 points donnés. Dès lors, *t*-1 nœuds compromis ne peuvent déduire quoi que ce soit au sujet de P_i et r_i. Egalement, la station de base peut tolérer jusqu'à *p-t* nœuds muets et être toujours capable de reconstituer r_i. En conséquence, cette technique fournit la confidentialité et une robustesse contre les dénis de service, même en présence de quelques nœuds mal intentionnés [28].

- ***Remarque***

Un nœud d'agrégation le long d'un chemin *q* doit fusionner les mesures des nœuds *i* et *j*, soit $r_i = P_i(0)$ et $r_j = P_j(0)$. Etant sur le chemin *q*, les seules données qu'il reçoit sont $P_i(q)$ et $P_j(q)$. Il retransmet $P_i(q) + P_j(q) = (P_i + P_j)(q)$. La même opération est effectuée sur les autres parts de ces nœuds sur les différents chemins. En recevant *t* échantillons, la station de base peut alors reconstruire $P_i + P_j$ et ensuite $(P_i + P_j)(0) = r_i + r_j$ [28].

AMS a l'avantage d'offrir une confidentialité très forte grâce à la cryptographie à seuil.

Les avantages et les inconvénients d'AMS se résument dans le tableau suivant (tableau3.2) :

Tableau 3. 2. Avantages et inconvénients.

AVANTAGES	**INCONVÉNIENTS**
Très forte confidentialité.	Surcharge de communication.
Avec moins de t parts, la reconstruction et la compréhension n'est pas évidente.	Chaque mesure envoyée a la même taille que la mesure d'origine.[1]
Code une seule mesure par part.[2]	Pas efficace contre les nœuds compromis.[3]
Peut tolérer la perte des parts.	N'est pas résistante contre l'attaque par rejeu.[4]
Protection contre les dénis de service.[5]	Pas d'authentification.
Homomorphisme (point clef).	La SB détecte qu'il y a des données falsifiées mais ne peut pas savoir quelles sont les reconstructions justes.
Sécurité assurée même s'il existe des captures.	*P-1* messages envoyés par mesure.

3.3. Chiffrement additif homomorphique

Un cryptage *Enc* est homomorphe, si à partir de *Enc* (*a*) et Enc (*b*) il est possible de calculer Enc (f (a, b)), où f peut être l'opération *d'addition* ou de *multiplication*, ce qui permet de distinguer le chiffrement homomorphique *additif* et le chiffrement homomorphique *multiplicatif* [23].

L'idée principale du cryptage additif homomorphique est que chaque nœud chiffre ses données m_i avec une clé qu'il partage avec la station de base pour obtenir $Enc_{ki}(m_i)$. Les nœuds intermédiaires manipulent des données chiffrées sans jamais accéder aux données en clair. Seule la station de base peut déchiffrer le résultat crypté : $Dec_{ki}(m_i)$.

Le principe de ce chiffrement est présenté dans l'algorithme ci-dessous [22]:

[1] C'est un résultat bien connu de la cryptographie à seuil qui se démontre facilement grâce à la théorie de l'information.
[2] L'attaquant ne connaît pas la mesure.
[3] Supposons qu'un nœud reçoit toutes les parts.
[4] Enregistrer les messages sans les comprendre et les rejouer plus tard.
[5] Dans le cas ou un agrégateur refuse d'agréger les données, on peut faire la reconstruction avec un autre sous-ensemble.

Algorithme 3.1. Chiffrement additif homomorphique

Données

m: donnée entier avec $m \in [0, M-1]$
M: l'ensemble du texte clair.
k : clef aléatoire.

Chiffrement

$c_1 = \text{Enc}_{k1}(m_1) = m_1 + k_1 \bmod M$
$c_2 = \text{Enc}_{k2}(m_2) = m_2 + k_2 \bmod M$

Agrégation

$C_a = \text{Enc}_A(m_1+m_2) = c_1 + c_2$ avec $A = k_1 + k_2$

Déchiffrement

$\text{Dec}_A(C_a) = C_a - A \bmod M$

4. Spécifications générales

4.1. Modèle du réseau

Nous considérons un réseau de capteurs à architecture clustérisée constituée de *n* nœuds capteurs statiques et une station de base (SB) statique. Chaque nœud capteur a un identificateur Id_i unique dans le réseau, où $1<=i<=n$.

Le réseau est divisé en groupes nommés clusters, et chaque cluster est géré par un cluster-Head centralisé (CHc), qui jouera le rôle d'agrégateur central. Parmi les nœuds capteurs ordinaires, *d* capteurs sont dédiés pour jouer le rôle d'agrégateurs partiaux, avec *d* inférieur à la taille du cluster.

Nous supposons que chaque agrégateur partial reçoit un nombre de parts égales et que seul le CHc peut communiquer avec la SB.

Nous supposons que les clés échangées entre les nœuds et la SB sont définies au début, et qu'une grande valeur fixe *M* est aussi prédéfinie.

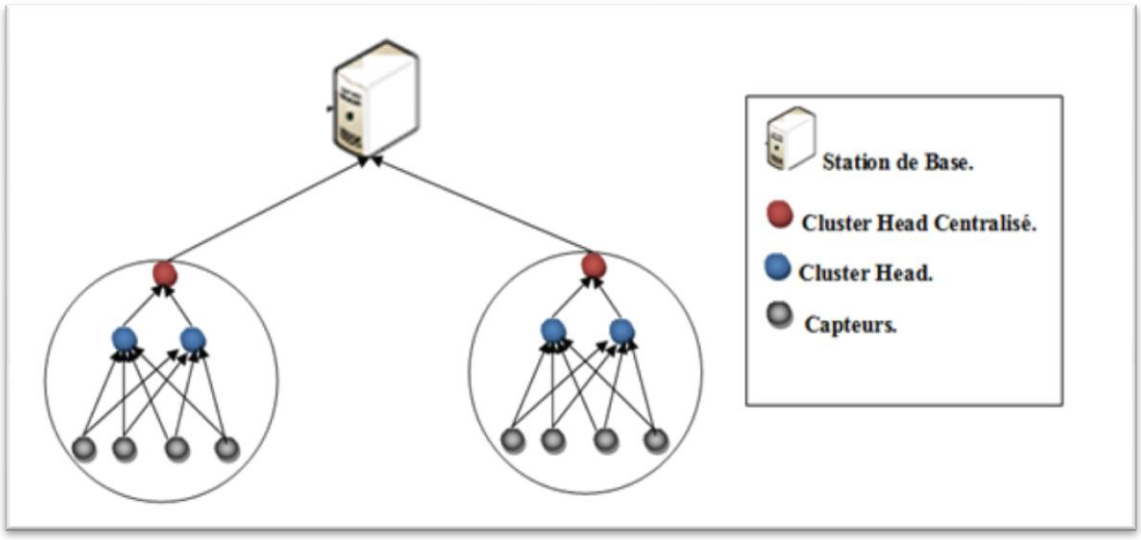


Figure 3. 2 : Modèle du réseau.

4.2. Modèle d'attaque

Nous supposons que l'attaquant tente de compromettre tous les agrégateurs (central et partiaux) dans le seul but de déclencher une attaque passive, c'est-à-dire de capter la donnée et la lire. En d'autres termes l'attaquant tente de casser la confidentialité des données.

4.3. Objectifs de conception

Lors de la conception des systèmes de protection de confidentialité, nous visons à atteindre les objectifs suivants [26] :

- **Protection des données** les données partiellement ou totalement agrégées ne devraient être connus que par la station de base.
- **Robustesse** assurer la confidentialité même si tous les agrégateurs sont compromis.
- **Efficacité** la surcharge du système doit être évitée autant que possible.

5. Solution proposée

Dans cette section, nous présentons notre algorithme pour sécuriser l'agrégation des données. Nous commençons par le principe de la solution ensuite nous détaillons l'algorithme.

5.1. Principe de la solution

La conception de notre protocole FTSA se base sur cinq phases :

a. Création des parts

Chaque nœud capteur va générer un polynôme pour créer des parts.

b. Chiffrement des parts

Le principe de cryptage additif homomorphique est utilisé pour chiffrer chaque part avant de l'émettre aux agrégateurs.

c. Agrégation des parts

Chaque agrégateur partial effectue des calculs sur les données chiffrées puis les transmet à l'agrégateur central.

d. Reconstruction des parts

A ce niveau, le CHc doit reconstruire les parts en retrouvant d'abord le polynôme correspondant $P(x)$, puis récupérer $P(0)$ chiffré et l'expédier à la SB.

e. Déchiffrement de la donnée

Cette dernière phase permet à la SB de retrouver la somme des mesures captées par chaque nœud et faire sa moyenne.

5.2. Détails de la solution

Dans cette sous-section, nous expliquerons chaque phase avec du pseudo-code :

a. Au niveau de chaque nœud capteur

Algorithme3.2. Création et chiffrement des données

Données *n*, *t* : entier

a[*i*] : tableau des entiers

Résultat la donnée divisée en *n* parts et chiffrées.

// génération du polynôme $P(x)= a[0] + a[1] * x + a[2] * x^2 + \ldots + a[t-1] * x^{t-1}$

Début

j = 1 ;
tant que (*j* <= *n*) faire début
l = 1;
i = 1;
a[*i*] = Random;
P[*j*] = *a*[0] + *a*[*i*] * pow (*j*,*l*);
l++;
i++;
tant que (*l* <= (*t*-1)) faire début
a[*i*] = Random;
P[*j*] = *P*[*j*] + a[*i*] * pow (*j*,l);
i++;
l++;

Fin tant que
$c[j]$ = Enc ($P[j]$) = $P[j] + k \bmod M$;
$b[j] = j$;
$d[j] = c[j]$;
j++;
Fin tant que
Fin

b. Au niveau de l'agrégateur partial

Algorithme3.3. Agrégation des données

Données

c **:** donnée chiffrée reçu de chaque part

Résultat

C **:** Données agrégées chiffrées

Début

$C=\sum_{i=0}^{t-1} c$
Fin

c. Au niveau de l'agrégateur central

Algorithme3.4. Reconstruction des parts

Données $x[i]$, $y[i]$: les couples reçu de la part des AGG partiaux

Résultat envoyer une seule donnée chiffrée à la SB

Début

pour i=0 jusqu'à (t-1) faire début
$x[i] = b[i+1]$;
$y[i] = C[i+1]$;
Fin pour
Pour i = 0 jusqu'à (t-1) faire début
j=0 ;
$l[i]$ = 1 ;
tant que (j <= (t-1)) faire début
si ($j = i$) alors //continuer
sinon $l[i]=(x - x[j]) / (x[j] - x[i]) * l[i]$;
j++;
Fin tant que
Fin pour
$P[x]$ = 0
Pour i=0 jusqu'à (t-1) faire

$P[x] = P[x] + y[i] * l[i]$;
//On retrouve la mesure en remplaçant $P[x]$ par $P[0]$ et l'envoyer à la SB.
Fin pour
Fin

d. Au niveau de la SB

Algorithme3.4. Déchiffrement de la donnée reçue

Données

***z* :** donnée chiffrée reçu de CHc
***k* :** somme des clés de tous les capteurs

Résultat

***C* :** Donnée déchiffrée

Début

$Dec_z = z - k \bmod M$
$Moy = Dec_z / Nb_{cap}$
Fin

5.3. Illustration

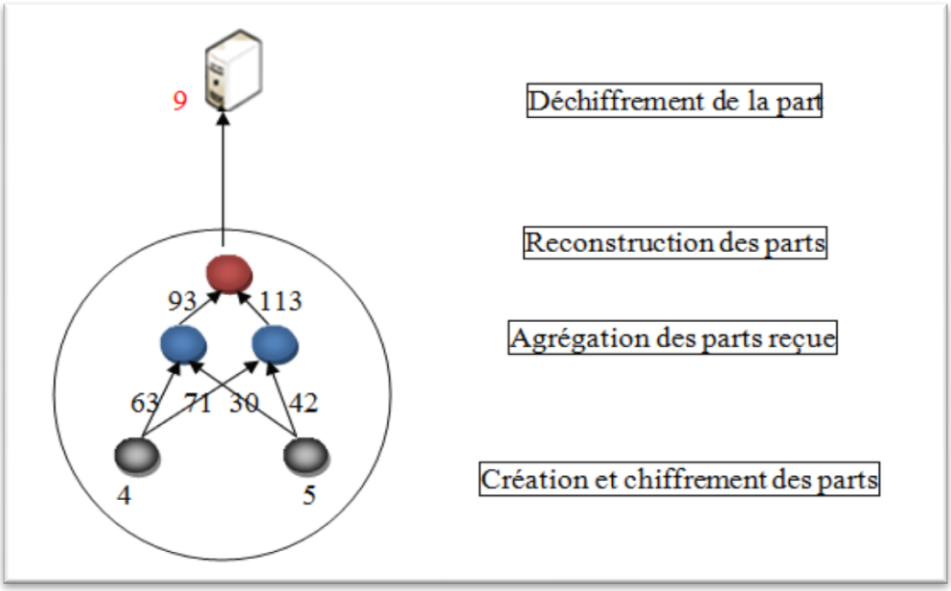


Figure 3. 3 : Illustration d'un exemple.

- **Données**

$M = 70,\ k_1 = 51,\ k_2 = 13$	$n = 2,\ t = 2,\ a_1 = 8,\ a_2 = 12$	$m_1 = 4,\ m_2 = 5$

- **Création des parts**

Vu que nous avons deux capteurs alors nous allons générer deux polynômes:

- $P_1(x)=4+8x$
- $P_2(x)=5+12x$

Nous allons créer deux parts pour chaque capteur car nous avons $n = 2$:

- $P_1(1)=12$
- $P_1(2)=20$
- $P_2(1)=17$
- $P_2(2)=29$

Les parts générées sont :

- (1,12), (2,20)
- (1,17), (2,29).

- **Chiffrement des parts**

- $c_1=12 + 51 \bmod 70=63$
- $c_2=20 + 51 \bmod 70=71$
- $c_3=17 + 13 \bmod 70=30$
- $c_4=29 + 13 \bmod 70=42$

Dans les quatre chemins nous allons envoyer ces couples :

- (1, 63), (2,71), (1, 30), (2,42)

- **Agrégation des parts reçues**

- AGG1: reçoit (1,63) et (1,30)

$c'_1=63 + 30=93$

- AGG2: reçoit (2,71) et (2,42)

$c''_2=71 + 42=113$

Les agrégateurs envoient au CHc : (1, 93) et (2, 113)

- **Reconstruction des parts**

- CHc: reçoit $(x_0, y_0) = (1 ,93)$ et $(x_1, y_1) = (2 ,113)$
  - $L0 = (x - x_1) / (x_0 - x_1) = -x + 2$
  - $L1= (x - x_0) / (x_1 - x_0) = x - 1$

Le polynôme généré: $P(x)=\sum_{i=0}^{t-1} y_i * l_i= 73 + 20 * x$

CHc calcule *P*(0) et trouve 73 donc il transmet cette valeur à la SB.

- **Déchiffrement de la donnée reçue et calcule de la moyenne**

  - Dec (*73)* = 73 - (51 + 13) mod 70 = 9
  - Moy = Dec_{73} / Nb_{cap} = 9 / 2 = 4.5

Nous remarquons que le résultat de déchiffrement c'est la somme des valeurs initiale : 5+4

6. Analyse de sécurité

L'analyse de sécurité de notre algorithme FTSA repose sur la résistance contre la compromission d'un nœud pour empêcher la lecture de la donnée par l'attaquant au niveau de :

- L'agrégateur central,
- l'agrégateur partial,
- tous les agrégateurs.

6.1. Compromission de l'agrégateur central

L'attaque de compromission de l'agrégateur central ne va rien divulguer à l'attaquant car la donnée est chiffrée. Même si l'attaquant capte les ***t*** parts il ne va pas la comprendre.

6.2. Compromission des agrégateurs partiaux

Dans ce genre d'attaque, notre but est assuré non seulement que les ***t*** parts ne sont pas réunies dans un seul agrégateur partial mais elles sont cryptées.

6.3. Compromission de l'agrégateur central et des agrégateurs partiaux

La compromission des données est un problème crucial dans les protocoles de sécurité des données agrégées car il est difficile d'assurer la confidentialité dans tous les cas. C'est ce qui nous a motivé à proposer notre protocole, qui empêche la lecture des données. Grâce à lui, même si tous les AGG sont compromis, les données originales ne pourront pas être obtenues. De ce fait, FTSA prévient ce genre d'attaques.

7. Implémentation

Nous avons implémenté notre protocole FTSA dans un réseau de capteur prototype constitué de capteurs telosb et un système d'exploitation Tinyos 2.1.0, ainsi que des outils logiciels bien spécifiques au domaine des RCSF tels que le langage NesC.

7.1. Outils utilisés

7.1.1. Capteur telosb

Il contient les différents composants matériels suivants :

- **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 [30].

- **Port USB** permettant de programmer (flasher) les capteurs, ou de faire remonter des informations et de les récupérer ensuite à l'aide d'un terminal à partir d'un ordinateur [30].

- **LED** outil visuel indispensable de vérification du fonctionnement du code intégré. Chaque capteur dispose de trois LED (rouge, vert et bleu), elles permettent de suivre le changement d'état de fonctionnement du capteur (réception, envoi, etc.) avec une programmation adéquate [30].

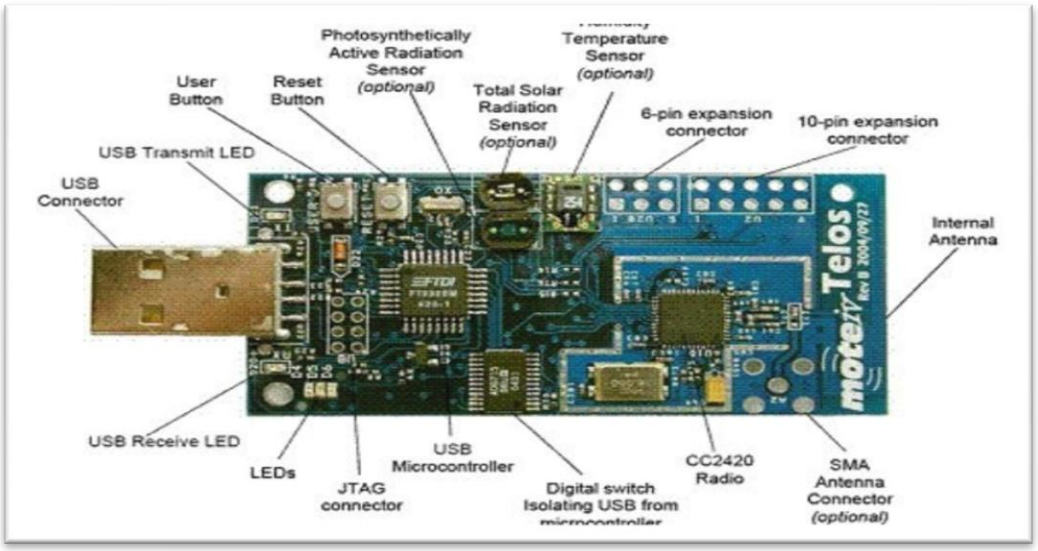


Figure 3. 4 : Capteur telosb Recto [4].

7.1.2. TinyOs

TinyOS est un système d'exploitation open-source conçu pour des réseaux de capteurs sans fil développé et maintenu par l'université de Berkeley et de nombreux contributeurs. C'est le plus répandu des OS pour les réseaux de capteurs sans fil en plus, il est utilisé dans les plus grands projets de recherches. Il permet de respecter les contraintes de mémoire et de puissance des nœuds en utilisant une programmation par composants, qui permet de réduire le code nécessaire à son fonctionnement. L'avantage de ce système est de permettre une programmation simple mais puissante des nœuds

tout en gardant la portabilité du code pour les nombreuses plateformes supportées [29]. TinyOS est en grande partie écrit en C mais on peut très facilement créer des applications personnalisées en langages C, NesC, et Java [4].

Dans notre projet, nous avons utilisé Tinyos-2.1.0.

7.1.3. NesC

NesC est un langage de programmation dérivé du langage C, fait pour minimiser l'utilisation de mémoire et de puissance de calcul par les capteurs, qui très souvent disposent de ressources très limitées (batterie de faible puissance, mémoire réduite...).

7.2. Mise en place de la plateforme

La mise en place de la plateforme nécessite deux étapes: l'installation logicielle et l'installation matérielle.

a. **Installation logicielle** tout d'abord, on a commencé par l'installation de TinyOs 2.1.0 dans Ubuntu ce qui nous a pris beaucoup de temps, ensuite on a essayé de se familiariser avec les capteurs telosb et le langage NesC.

b. **Installation matérielle** dans notre cas, nous avons utilisé au minimum 6 capteurs dont chacun à un rôle bien particulier, les détails sont présentés ci-dessous. Grace à la commande ***make telosb docs*** nous avons générer une représentation graphique de chaque programme.

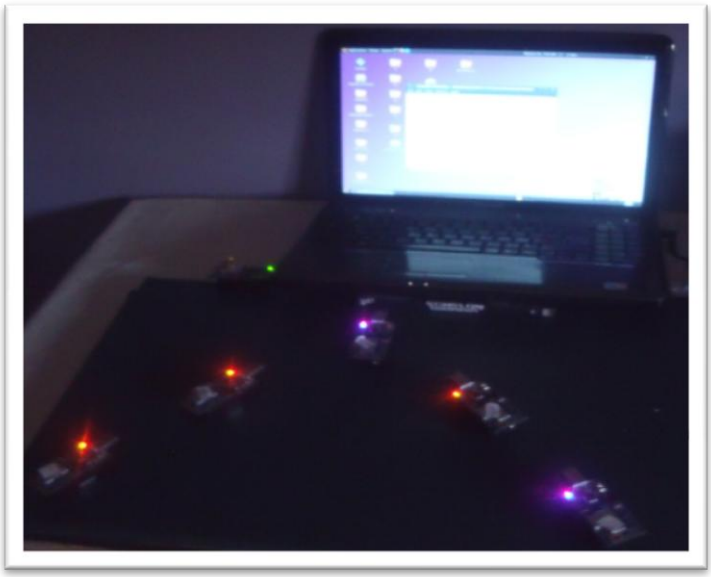


Figure 3. 5 : Environnements de travail.

7.3. Scénario de notre application

Notre application contient six capteurs :

- Deux d'entre eux captent la température de l'environnement, créent des parts, chiffrent ces derniers et les envoyer.
- Deux autres capteurs, jouent le rôle des agrégateurs partiaux, leur rôle est de faire la somme des parts chiffrées reçues.
- Le cinquième, est un agrégateur central qui fait la reconstruction des températures captées,
- Le sixième, représente la station de base qui fait le déchiffrement de la température et calcule sa moyenne.

7.3.1. Représentation graphique des composants

a. Application Capteurs

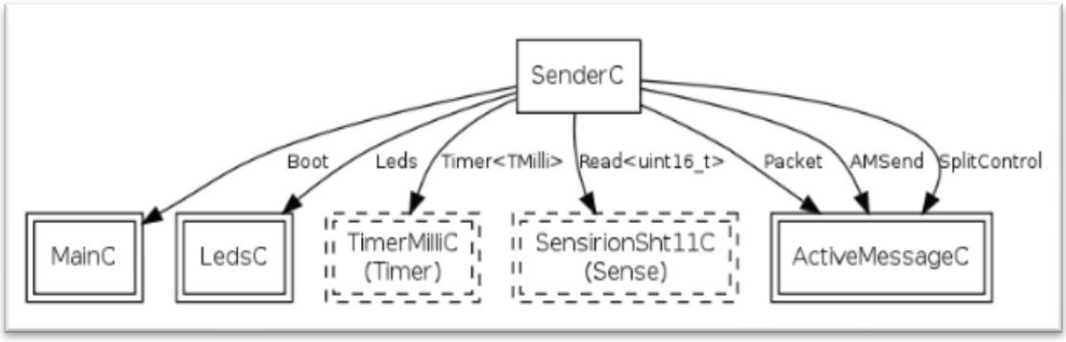


Figure 3. 6 : Représentation graphique du composant SenderC.

L'application SenderC installée au niveau des capteurs est constituée de 3 fichiers : fichier d'en-tête, fichier de configuration et le fichier Module.

• Fichier d'en-tête que nous avons nommé « Message.h » qui contient la structure du message à envoyé.

```
#ifndef MESSAGE_H
#define MESSAGE_H
typedef nx_struct Temperature_Msg{
        nx_uint16_t nodeid;
        nx_uint16_t part [3] [3];
} Temperature_Msg;
enum {AM_TEMPERATURE_MSG = 6};
#endif /* MESSAGE_H */
```

• Fichier de configuration du capteur nommé « SenderAppC »

```
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_MS
G];
  App.RadioControl -> ActiveMessageC;
}
```

• Module « SenderC »

```
module SenderC {
  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(){ /*….*/}
event void RadioControl.startDone(error_t err){/*....*/}
event void RadioControl.stopDone(error_t err){}
event void Timer.fired(){/*….*/}
event void Read.readDone(error_t result, uint16_t
```

```
val){/*….*/}
                    }
```

b. Application Agrégateur

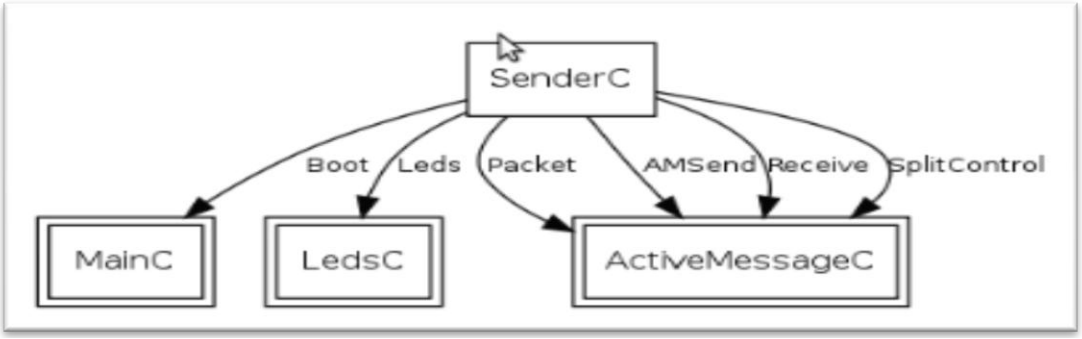


Figure 3. 7 : Représentation graphique du programme agrégateur.

L'application SenderC installée au niveau des agrégateurs est constituée de 3 fichiers : fichier d'en-tête, fichier de configuration et le fichier Module

• Fichier d'en-tête que nous avons nommé « Message.h » qui contient la structure du message à envoyé.

```
#ifndef MESSAGE_H
#define MESSAGE_H
typedef nx_struct Temperature_Msg{
        nx_uint16_t nodeid;
        nx_uint16_t part [3] [3];
} Temperature_Msg;

typedef nx_struct TemperatureChiffre_Msg{
        nx_uint16_t nodeid;
        nx_uint16_t agg [3];
        nx_uint16_t temp;
} TemperatureChiffre_Msg;
enum {AM_TEMPERATURE_MSG = 6};
enum {AM_TEMPERATURECHIFFRE_MSG = 6};
#endif /* MESSAGE_H */
```

• Configuration SenderAppC

```
configuration SenderAppC{}
implementation{
  components MainC, LedsC;
  components SenderC as App;
  components ActiveMessageC;

  App.Boot -> MainC.Boot;
  App.Leds ->LedsC.Leds;
  App.Packet -> ActiveMessageC;
  App.AMSend -> ActiveMessageC.AMSend[AM_TEMPERATURECHIFFRE_MSG];
  App.Receive -> ActiveMessageC.Receive[AM_TEMPERATURECHIFFRE_MSG];
  App.RadioControl -> ActiveMessageC;
}
```

• Module SenderC

```
module SenderC{

 uses{
   interface Boot;
   interface Leds;
   interface Packet;
   interface AMSend;
   interface Receive as Receive;
   interface SplitControl as RadioControl;
  }
}

implementation{
event void Boot.booted(){/*….*/}
event void RadioControl.startDone(error_t err){/*….*/}
event void RadioControl.stopDone (error_t err){}
event message_t* Receive.receive (message_t* msg, void* payload,
uint8_t len){/*….*/}
event void AMSend.sendDone (message_t* msg, error_t err){/*….*/}
```

c. Application Station de base

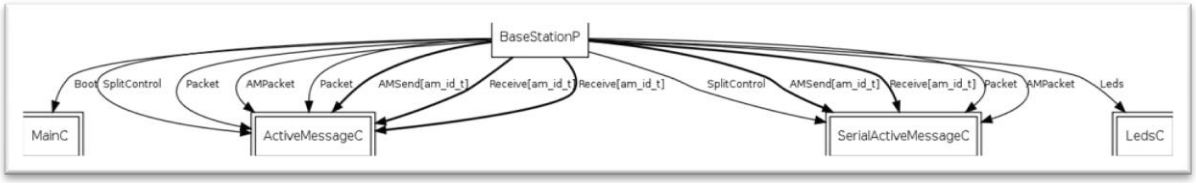


Figure 3. 8 : Représentation graphique du programme station de base

L'application BaseStationP installée au niveau des capteurs est constitué de 3 fichiers : fichier d'en-tête, fichier de configuration et le fichier Module.

• Fichier d'en-tête que nous avons nommé « Message.h » qui contient la structure du message à envoyé.

```
#ifndef MESSAGE_H
#define MESSAGE_H
typedef nx_struct Temperature_Msg{
    nx_uint16_t nodeid;
    nx_uint16_t part1 [3] [3];
    nx_uint16_t agg;
} Temperature_Msg;

typedef nx_struct TemperatureChiffre_Msg{
    nx_uint16_t nodeid;
    nx_uint16_t agg [3];
    nx_uint16_t temp;
} TemperatureChiffre_Msg;
enum {AM_TEMPERATURECHIFFRE_MSG = 6};
#endif /* MESSAGE_H */
```

• Fichier de configuration du capteur nommé « BaseStationC »

```
configuration BaseStationC {
}
implementation {
  components MainC, BaseStationP, LedsC;
  components ActiveMessageC as Radio,      SerialActiveMessageC
as Serial;
```

```
    MainC.Boot <- BaseStationP;
    BaseStationP.RadioControl -> Radio;
    BaseStationP.SerialControl -> Serial;
    BaseStationP.UartSend -> Serial;
    BaseStationP.UartReceive -> Serial.Receive;
    BaseStationP.UartPacket -> Serial;
    BaseStationP.UartAMPacket -> Serial;
    BaseStationP.RadioSend -> Radio;
    BaseStationP.RadioReceive ->    Radio.Receive;
    BaseStationP.RadioSnoop -> Radio.Snoop;
    BaseStationP.RadioPacket -> Radio;
    BaseStationP.RadioAMPacket -> Radio;
    BaseStationP.Packet -> Radio;
    BaseStationP.Leds -> LedsC;
  }
```

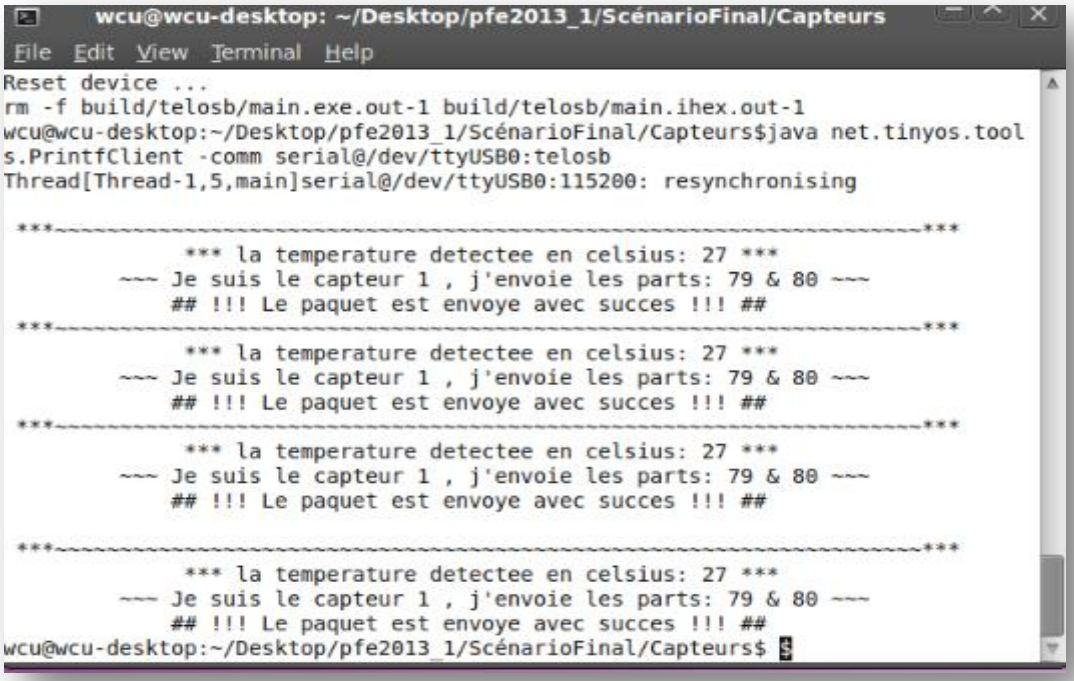
• Module « BaseStationP »

```
#include "AM.h"
#include "Serial.h"
#include "printf.h"
#include "Message.h"
module BaseStationP @safe() {
  uses {
    interface Boot;
    interface SplitControl as SerialControl;
    interface SplitControl as RadioControl;
     interface AMSend as UartSend[am_id_t id];
    interface Receive as UartReceive[am_id_t id];
    interface Packet as UartPacket;
    interface AMPacket as UartAMPacket;
    interface Packet;
    interface AMSend as RadioSend[am_id_t id];
```

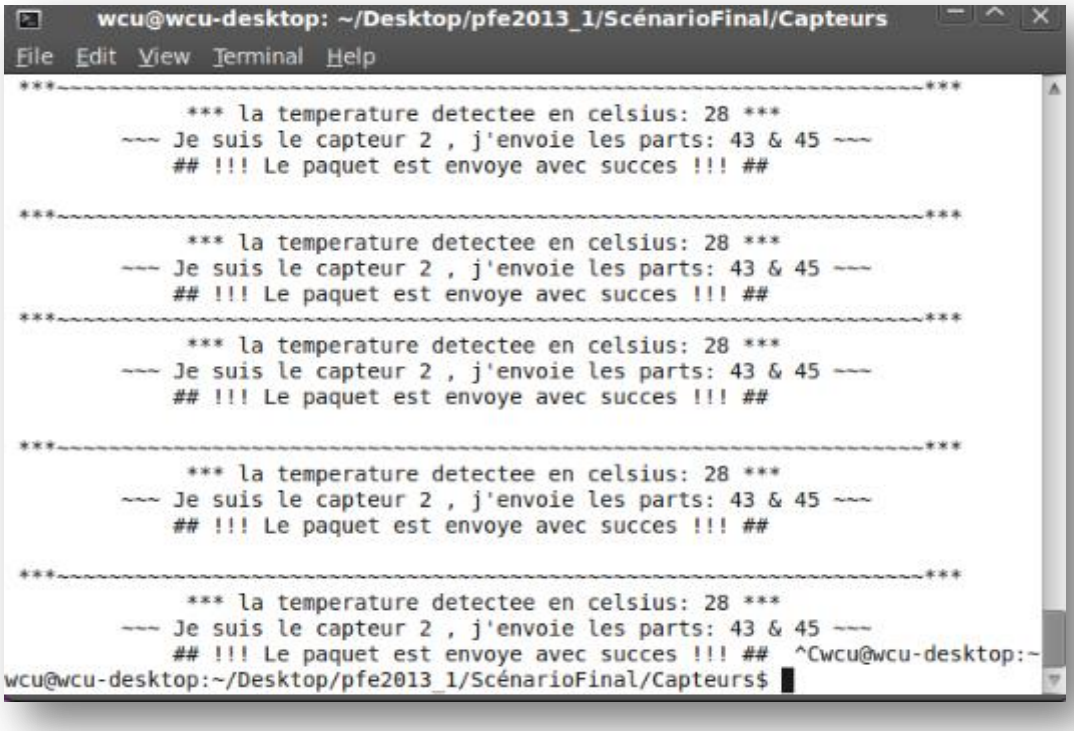
```
    interface Receive as RadioReceive[am_id_t id];
    interface Receive as RadioSnoop[am_id_t id];
    interface Packet as RadioPacket;
    interface AMPacket as RadioAMPacket;
    interface Leds;
 }
}
implementation
{
event void Boot.booted() {/*….*/}
 event void RadioControl.startDone(error_t error) {/*….*/}
event void SerialControl.startDone(error_t error) {/*….*/}
event void SerialControl.stopDone(error_t error) {}
  event void RadioControl.stopDone(error_t error) {}
  message_t* ONE receive(message_t* ONE msg, void* payload, uint8_t len);
  event message_t *RadioSnoop.receive[am_id_t id](message_t *msg, void *payload,
uint8_t len)                {/*….*/ }
 event message_t *RadioReceive.receive[am_id_t id](message_t *msg, void *payload,
uint8_t len) {/*….*/ }
  message_t* receive(message_t *msg, void *payload, uint8_t len) {/*….*/
  event void UartSend.sendDone[am_id_t id](message_t* msg, error_t error) {/*…..*/}
  event message_t *UartReceive.receive[am_id_t id](message_t *msg, void *payload,
uint8_t len) {/*….*/ }
  event void RadioSend.sendDone[am_id_t id](message_t* msg, error_t error) {/*….*/ }
}
```

7.4. Quelques executions

Capteur 1



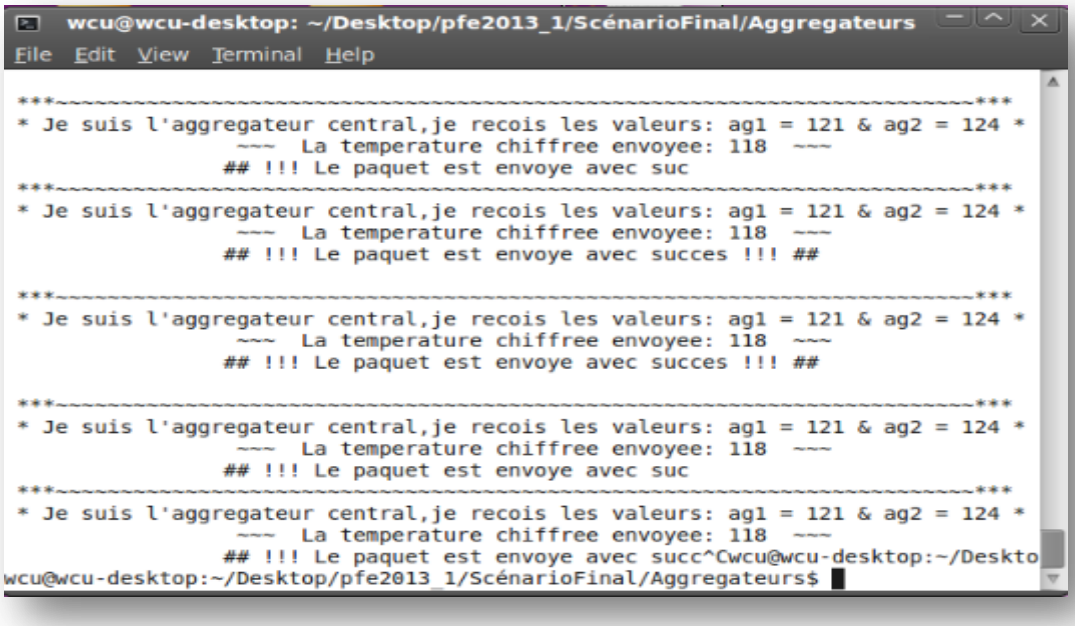
Capteur 2



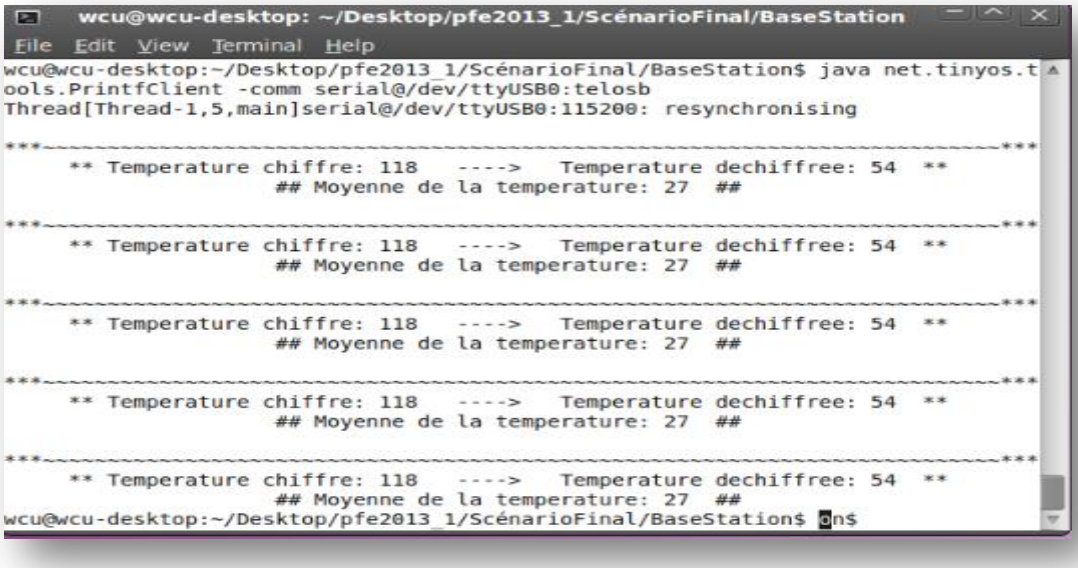
Agrégateur 1

Agrégateur 2

Agrégateur central



Station de base



7.5. 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.

Notre application contient *au minimum* 6 capteurs dont le capteur station de base est relié au centre de contrôle qui contient un programme prédéfini dans tinyos.

7.5.1. Centre de contrôle

Exécute une application développée en Java, nous avons 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 nous avons 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 les températures détectées par chaque capteur, de plus il effectue le déchiffrement à l'aide de la formule du cryptage additif homomorphique et il fait la moyenne.

7.5.2. Exemple d'exécution

Figure 3. 9 : Notre interface java.

8- Conclusion

Nous avons présenté dans ce chapitre notre protocole FTSA (Fault-Tolerant Secure data Aggregation in wireless sensor networks) pour tolérer la perte de messages occasionnées par le canal radio et renforcer la confidentialité des données, ceci en combinant deux algorithmes : l'algorithme AMS [24] et le chiffrement homomorphique [22]. De ce fait notre protocole assure une confidentialité de bout en bout, même si une partie ou tous les agrégateurs ont été compromis.

Conclusion générale

Les réseaux de capteurs sans fil (WSNs) sont souvent qualifiés de technologie émergente qui va bouleverser notre quotidien. Ces composants électromécaniques d'une taille très réduite et qui communiquent via un réseau sans fil omniprésent, ouvrent largement les horizons des applications construites jusqu'à maintenant.

Les applications des réseaux de capteurs sans fil sont nombreuses. Elles comprennent différents domaines : médicale, agricole, militaire,… Pour que ces réseaux puissent mener à bien leurs missions ils doivent assurer un certain niveau de sécurité qui diffère selon l'application déployée. Beaucoup de chercheurs consacrent une grande importance à l'étude des menaces et des failles sécuritaires qui peuvent paralyser les réseaux de capteurs sans fil. Par la suite, ils développent des mécanismes qui permettent aux utilisateurs d'atteindre les besoins de sécurité requis.

Nous avons essayé à travers ce mémoire de faire le tour de ce nouveau phénomène qui touche les réseaux de capteurs sans fil. Nous avons commencé par présenter les généralités qui entourent le domaine des WSNs, puis nous avons exposé les différents défis que doit surmonter la conception des protocoles de communication dans ces réseaux. Nous nous sommes intéressés aux problèmes liés à la sécurité, et en particulier à la sécurisation des données agrégées dans les RCSF. Les techniques d'agrégation des données, permettent de réduire le nombre de messages et par conséquent réduire la consommation en énergie.

Dans notre projet nous avons présenté un algorithme nommé FTSA qui traite le problème de sécurité des données agrégées. Ceci est dans le but de tolérer les pertes de messages, renforcer la confidentialité et de remonter l'information au centre de contrôle en toute sécurité.

L'implémentation de notre protocole nous a permis d'appréhender un nouveau environnement des systèmes embarqués Tinyos, d'apprendre un nouveau langage de programmation NesC. Il faut tout de même noter que la prise en main de ce système d'exploitation n'est pas une mince affaire et que l'implémentation réelle sur des capteurs n'est pas évidente à cause des collisions et du canal radio bien souvent non fiable.

Comme perspective, il serait intéressant de simuler notre protocole sur un réseau de capteurs plus grand afin d'évaluer ses performances selon certaines métriques d'évaluation choisies. Cette simulation permettra d'étudier le comportement du protocole et d'en tirer des conclusions afin de l'améliorer.

Bibliographie

[1] HADDOU BENDERBAL Hichem, KOULOUGHLI Imen, « Etude et proposition d'un modèle pour l'économie d'énergie dans les réseaux de capteurs sans fil: application médicale sportive », Mémoire de fin d'études pour l'obtention du diplôme de Master en Informatique, Université de Tlemcen, Septembre 2012.

[2] 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, Tlemcen, 2009.

[3] DHIB Eya, « Routage avec QoS temps réel dans les réseaux de capteurs », RAPPORT DE PROJET DE FIN D'ÉTUDES, École supérieure des communications, Tunis, 2007.

[4] CHALLALYacine, «Réseaux de Capteurs Sans Fils», support-SIT60, Vol. 103, pp. 14, 17, Novembre 2008,

[5] Claude Castelluccia, « La Sécurité des Capteurs et Réseaux de Capteurs », PLANETE, INRIA, Juin 2008.

[6] XUE Yong, AGUILARAndres [et al.], *«* Agrégation de données dans les réseaux de capteurs *»*, Projet SR04, Rapport final, Vol. 21, pp. *3-9, 13-14, 16*, 2010.

[7] BOUNEGTA, « Réseaux de capteurs sans fil », Université de Bechar , Ingénieur d'état en informatique, 2010.

[8] SAHRAOUI Belkheyr., « La Géo--localisation dans les Réseaux de Capteurs sans Fil », Mémoire de fin d'études pour l'obtention du diplôme Ingénieur d'Etat en Informatique, Juillet 2011.

[9] LABRAOUI Nabila, Mourad Guerroui [et al.], « Data Aggregation Security Challenge in Wireless Sensor Networks », A Survey, Published by licenseunder the OCP Science imprint, Vol. 324, pp. 304-305, 311, October 2010.

[10] Roosta T, Shieh S and Sastry S, «Taxonomy of security attacks in sensor networks», 1st IEEE International Conference on System Integration and Reliability Improvements, Washington, 2006.

[11] Alzaid H, Foo E and Gonzalez J M, «secure data aggregation in wireless sensor networks», A Survey, 6th Australasian Information Security Conference (AISC), Australia, L. Brankovic and M.Miller, editors, Vol. 81, pp. 93–105, 2008.

[12] LABRAOUI Nabila, «La sécurité dans les réseaux de capteurs sans fil», Thèse pour l'obtention du diplôme de doctorat, 2012.

[13] MERAD Omar, « la sécurité des données agrégées dans les réseaux de capteurs sans fil », mémoire de fin d'études pour l'obtention du diplôme de Master en Informatique, 2010.

[14] John Paul Walters [et al.], «Wireless Sensor Network Security»,A Survey, Department of Computer ScienceWayne State University, 2006.

[15] Hani Alzaid, Ernest Foo et Juan Gonzalez Nieto, «Secure Data Aggregation in Wireless Sensor Network», A Survey, vol. 13, pp. 3, 2011.

[16] SalheddineKabou, BelgourariAbdessamed, «Etat de l'art sur les réseaux de capteurs sans fil», Mémoire de fin d'études pour l'obtention de Licence en Informatique, Université de Bechar, Algérie, 2010.

[17] N.Labraoui, M.Gueroui,T.Zia, «Data Aggregation Security Challenges in Wireless Sensor Networks», in the proceeding soft he 9th International Symposium on Programming and Systems (ISPS 2009), May25-27, 2009, Algiers, Algeria.

[18] Claude Castelluccia et Aurélien Francillon, « Protéger les réseaux de capteurs sans fil », Actes du symposium SSTIC08, vol. 11, pp. 8, 2005.

[19] B. Przydatek, D. Song, and A. Perrig, «SIA : Secure information aggregation in sensor networks», In Proceedings of the 1st international conference on Embedded networked sensor systems, Los Angeles, CA, Nov 5–7, 2003.

[20] H. Chan, A. Perrig, and D. Song, «Secure hierarchical in-network aggregation in sensor networks», In Proceedings of the 13th ACM conference on Computer and communications security (CCS '06), pp. 278–287, 2006.

[21] Du, W., Deng, J., Han, Y. S. and P. Varshney, «A witness-based approach for data fusion assurance in wireless sensor networks», In IEEE Global Communications Conference (GLOBECOM), Vol. 3, pp. 1435–1439, 2003.

[22] Castelluccia C, Mykletun E and Tsudik G, «Efficient Aggregation of Encrypted Data Wireless Sensor Network »,Proceeding ACM/IEEE Mobiquitous, San Diego, 2005.

[23] tebaa [et al.], «homomorphic encryption applied to the cloud computing security», Proceedings of the World Congress on Engineering, London, U.K, vol. 4 , pp.2, 2012.

[24] Claveirole T [et al.], «Securing wireless sensor networks against aggregator compromises», IEEE Communications Magazine, 2008.

[25] Hur J H [et al.], «Trust-based aggregation in wireless sensor networks», 2005.

[26] Taiming [et al.], «Confidentiality Protection for Distributed Sensor Data Aggregation», Department of Computer Science, Iowa State University, vol. 9, pp. 3, 2008.

[27] Gursel A [et al.], «Robust Trust Mechanisms for Monitoring Aggregator Nodes in Sensor Networks», International Workshop on Agent Technology for Sensor Networks, Estoril, 2008.

[28] Thomas Claveirole1 [et al.], «Résistance contre les attaques par capture dans les réseauxde capteurs», vol. 10, pp. 1-5, 2007.

[29] Nicolas ESTEVES, « Capture de Mouvement par Réseau de Capteurs Sans Fil », rapportEsteves, Juillet 2007, Vol. 41, pp. 13, 15.

[30] Gérard CHALHOUB, « Les réseaux de capteurs sans fil », Complexe scientifique des Cézeaux , Clermont Université, Vol. 6, pp. 4