RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITÉ AMAR TELIDJI - LAGHOUAT

Faculté des **S**ciences

Département de **M**athématique et **I**nformatique

MÉMOIRE

Présenté par :

MEZRAG Fares

Pour l'obtention du diplôme de MAGISTER en Informatique

Option : **I**nformatique **R**épartie et **M**obile **(IRM)**

THÈME

Sécurité du Routage Hiérarchique Basé sur les Clusters dans les Réseaux de Capteurs sans Fil

Soutenu le 01/06/2015 , devant le jury formé de :

M. M. B. YAGOUBI	Professeur	Université de Laghouat	Président
M. M.BENMOHAMMED	Professeur	Université de Constantine 2	Rapporteur
M. B. BOUDERAH	Professeur	Université de M'Sila	Examinateur
Mme. H. BOUZOUAD	Maître de conférences	Université de Laghouat	Examinatrice
M. N. LAGRAA	Maître de conférences	Université de Laghouat	Examinateur

Dédicaces

A tous ceux qui me sont chers...

ma mère,

mon père,

mes sœurs et mes frères,

mes amis et mes collègues.

Fares

Remerciements

Avant tout je remercier ALLAH, le tout puissant, de m'avoir donné le courage, la patience et la volonté pour réaliser ce travail.

Je tiens à remercier vivement mon encadreur le professeur Mohamed BEN-MOHAMMED pour sa disponibilité et ses conseils qui m'ont permis de progresser et de mener à bien ce mémoire.

J'adresse aussi mes sincères remerciements au président et aux membres de jury de m'avoir honoré en acceptant de juger ce modeste travail.

Je remercie également M. Brahim BOUDERAH, Professeur à l'université de M'Sila et Mme. Fadila MEZRAG, Docteur à l'université de M'Sila pour leur soutien tout au long de ce travail. Je suis heureux de pouvoir leur exprimer mes plus vifs remerciements et ma très sincère reconnaissance.

Un grand MERCI à ceux qui ont aidé de prés ou de loin à la réalisation de ce travail.

ملخـص

تشهد شبكات الاستشعار اللاسلكية حاليا توسع كبير و استخدام واسع في مختلف التطبيقات التي تتطلب اجراءات امنية مشددة. يتم نشر هذا النوع من الشبكات عادة في مناطق غير آمنة مما يجعلها عرضة للعديد من الهجمات التي تسمح للدخيل بالسيطرة على واحد او اكثر من اجهزة الاستشعار وذلك لعرقلة حسن سير عمل الشبكة. تبقى بروتوكولات التوجيه الهرمية في شبكات الاستشعار بدون اي اجراءات أمنية (LEACH, TEEN, PEGASIS)، بالرغم من ان الخدمات الأمنية تعتبر أمرا أساسيا لضمان نشر واسع لهذه الشبكات.

من خلال هذا البحث، قمنا باقتراح نسخة امنة للبروتوكول LEACH، تتلخص الفكرة في اضافة لهذا البروتوكول آلية امنية تأخذ بعين الاعتبار محدودية موارد اجهزة الاستشعار من جهة، و توفير الحماية ضد الهجمات التي كثيرا ما تنفذ ضد البروتوكول LEACH من جهة اخرى.

للتحقق من صحة الجانب النظري للبروتوكول المقترح، اجرينا تقييما نظريا لنموذجنا الامني فيما يخص تأمين مرحلة التثبيت و مرحلة الارسال للبروتوكول LEACH. قمنا ايضا بإجراء عدة عمليات محاكاة باستخدام المحاكي TOSSIM لتقييم أداء البروتوكول المقترح حيث اظهرت النتائج ان هذا الاخير يحقق التوازن بين الأداء ومستوى الأمن المقدم.

كلمات البحث : شبكة الاستشعار، أمن التوجيه، LEACH.

RÉSUMÉ

Les réseaux de capteurs sans fil connaissent actuellement une grande extension et une large utilisation dans différents types d'applications exigeant une grande sécurité. Ils sont typiquement déployés dans des zones hostiles, ce qui les rend vulnérables à plusieurs attaques dans lesquelles l'intrus peut prendre le contrôle d'un ou plusieurs nœuds capteurs pour perturber le bon fonctionnement du réseau. Les protocoles de routage hiérarchique dans les réseaux de capteurs sont spécifiés sans aucune mesure de sécurité (LEACH, TEEN, PEGASIS). Pourtant, les services de sécurité sont identifiés comme essentiels pour assurer un déploiement large de ces réseaux.

Durant ce travail de recherche, nous nous sommes intéressés à proposer une version sécurisée de protocole LEACH, l'idée consiste à intégrer dans ce protocole un mécanisme de sécurité qui tient compte, d'une part, des ressources limités des nœuds de capteurs et d'autre part, pour faire face aux attaques qui sont fréquemment menées contre le protocole LEACH.

Afin de valider l'aspect théorique de notre protocole, nous avons effectué une évaluation théorique de notre modèle de sécurité qui tient à sécuriser la phase d'installation et la phase de transmission du protocole LEACH, nous avons également effectué des simulations à travers le simulateur TOSSIM pour évaluer les performances de notre protocole. Les résultats obtenus montrent que notre protocole réalise un équilibre entre les performances du protocole et le niveau de sécurité offert.

Mots-clés : Réseau de Capteurs, Sécurité du routage, LEACH.

ABSTRACT

The wireless sensor networks are experiencing a large expansion and wide use in various types of applications that require high security. They are typically deployed in hostile areas, which make them vulnerable to several attacks in which the intruder can take control of one or more nodes to disrupt the proper functioning of the network. The hierarchical routing protocols in sensor networks are specified without any security measures (LEACH, TEEN, and PEGASIS). However, the security services are identified as essential to ensure widespread deployment of these networks.

In this research work, we were interested in proposing a secure version of LEACH protocol. The idea is to incorporate into the protocol a security mechanism that takes into account, on the one hand, the limited resources of sensor nodes and, on the other hand, dealing with attacks that are frequently conducted against the LEACH protocol.

To validate the theoretical aspect of our protocol, we performed a theoretical evaluation of our security model which aimes to secure the setup phase and the steady-state phase of LEACH protocol. We have also performed simulations through the simulator TOSSIM to evaluate the performance of our protocol. The results show that our protocol achieved a balance between the performance of the protocol and the level of security provided.

Keywords : Wireless sensor networks, Security of routing, LEACH.

TABLE DES MATIÈRES

TABLE DES FIGURES

LISTE DES TABLEAUX

LISTE DES ABRÉVIATIONS

AC	Authentication Confidentiality cluster based secure routing protocol for WSN
ADC	Analog to Digital Converter
APTEEN	Adaptive Periodic Threshold-sensitive Energy Efficient sensor Network protocol
CH	Cluster Head
CPU	Central Processing Unit
DD	Directed Diffusion
ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
ECDLP	Elliptic Curve Discrete Logarithm Problem
EECS	Energy Efficient Clustering Scheme
EED	End to End Delay
GAF	Geographic Adaptative Fidelity
GEAR	Geographic and Energy Aware Routing
GPS	Global Positioning System
GUI	Graphical User Interface
HEED	Hybrid, Eenergy-Efficient, Distributed clustering approach
ID	identifiant d'un nœud
JTOSSIM	Java TinyOS SIMulator
LEACH	Low-Energy Adaptive Clustering Hierarchy
LP	Lost Packet
LR	Loss Rate

MAC	Message Authentication Code
MD5	Message-Digest algorithm 5
MIPS	Million of Instructions Per Second
NesC	Network embedded system C
PEGASIS	Power-Efficient GAthering in Sensor Information System
QoS	Quality of Service
RAM	Random Access Memory
RCSF	Réseau de Capteurs Sans Fil
RSA	Rivest Shamir Adleman algorithm
SAR	Sequential Assignment Routing
SB	Station de Base
SecLEACH	Secure LEACH
SHA-1	Secure Hash Algorithm 1
SLEACH	Secure LEACH
SN	Simple Node
SP	Sent Packet
SPIN	Sensor Protocols for Information via Negotiation
SQL	Structured Query Language
TDMA	Time Division Multiple Access
TEEN	Threshold-sensitive Energy Efficient sensor Network protocol
TL-LEACH	Two-Levels Hierarchy for Low-Energy Adaptive Clustering Hierarchy
TOSSIM	TinyOS SIMulator
WSN	Wriless Sensor Network

INTRODUCTION GÉNÉRALE

Au cours des dernières années les réseaux sans fil connaissent une forte expansion, ils sont de plus en plus utilisés vu de leur facilité de déploiement et leur coût relativement faible. Toutefois l'évolution rapide de la technologie dans le domaine de la communication sans fil, a permis la manipulation de l'information à travers des unités de calcul portables qui accèdent au réseau, à travers une interface de communication sans fil permettant ainsi aux unités de calcul, une libre mobilité et ne pose aucune restriction sur la localisation des usagers. Cependant la convergence des systèmes électroniques miniaturisé et des technologies de communication sans fil a permis l'émergence des réseaux de capteurs sans fil (RCSF ou WSN : Wireless Sensor Network).

Le réseau de capteurs sans fil est une collection de nœuds capteurs (de petits appareils avec des ressources très limitées) déployés d'une manière aléatoire ou déterministe, dans une zone d'intérêt. Ces nœuds peuvent échanger des données entre eux sans utiliser une infrastructure réseau préexistante et fixe ou une administration centralisée. D'autre part, chaque nœud de réseau communique directement avec les autres nœuds qui se trouvent dans sa protée radio (rayon de communication), alors que la communication avec les nœuds distants ou hors portée radio se fait par l'intermédiaire des autres nœuds qui acheminent les données à destination. En fait, ce processus est assuré par le protocole de routage.

Le routage est une fonction fondamentale dans chaque réseau, il est basé sur une communication multi-sauts, il faut dire que les réseaux de capteurs sans fil ne sont pas exceptionnels, plusieurs protocoles de routage ont été proposés pour ces réseaux de capteurs. Ils sont généralement classés selon deux catégories [1] : la structure du réseau (Network structure) et la

stratégie de routage du protocole (Protocol operation). Les protocoles de routage basés sur la structure du réseau peuvent être classés en trois sous catégories : routage à plat (Flat network routing), routage hiérarchique (Hierarchical network routing) et routage géographique (Location based routing). Selon leurs stratégies, les protocoles de routage sont encore classés en quatre sous catégories : routage basé sur les chemins multiples (Multi-path based routing), routage basé sur les requêtes (Query based routing), routage basé sur la négociation entre les nœuds (Negociation based routing) et routage avec qualité de service (Quality of service based routing).

Le protocole de routage hiérarchique suit une approche basée sur les clusters, cette approche a montrée son efficacité par comparaison aux autres approches en termes de réduction de la consommation d'énergie, par conséquent, elle permet la prolongation de la durée de vie du réseau de capteurs sans fil. L'idée consiste à former des groupes (clusters) de nœuds capteurs et d'utiliser les CHs (Cluster-Heads) élus comme routeurs. Chaque CH collecte les données à partir de tous les nœuds capteurs qui appartiennent à leur cluster, agrège les données rassemblées et les transmet directement vers la station de base (SB). L'agrégation et le traitement des données dans le CH réduit considérablement le nombre total de messages envoyés à la station de base. LEACH (Low-Energy Adaptive Clustering Hierarchy) [2] est un protocole considéré comme étant le plus populaire des protocoles de routage hiérarchique basé sur les clusters, ce protocole repose fondamentalement sur les CHs pour l'agrégation des données et le routage.

Problématique

Généralement, les problèmes de la sécurité dans les protocoles de routage hiérarchiques dans les réseaux de capteurs, n'ont pas eu une grande attention puisque la plupart de ces protocoles ont été développés dans le but de l'acheminement efficace de l'information, cependant l'aspect sécurité a été négligé. En effet, ils sont vulnérables à différents types d'attaques qui menacent la fiabilité des données en circulation. Ces attaques peuvent être lancées de façon relativement simple, en particulier, la nature des communications sans fil facilite l'écoute, permettant ainsi une analyse facile du trafic réseau. Le manque d'une infrastructure fixe et l'hypothèse d'un environnement ouvert sans aucune surveillance humaine, permettent des attaques comme la modification, l'injection des données erronées,...etc. Comme la plupart

des protocoles de routage hiérarchique dans les réseaux de capteurs, le protocole LEACH est vulnérable à un certain nombre d'attaques de sécurité [3], telles que l'écoute et l'analyse du trafic échangé, l'altération des données et la négligence des messages. Les attaques impliquant des CHs, sont les plus préjudiciables. Si un nœud malicieux réussi à devenir un CH, il peut lancer des attaques comme selective forwarding pour perturber le fonctionnement du réseau. Par ailleurs, un nœud malicieux peut choisir de ne pas attaquer le CH, et essayer également d'injecter des données erronées dans le réseau. Ainsi le protocole LEACH a besoin d'assurer la confidentialité, l'intégrité, la fraîcheur et l'authentification du nœud originaire du message transmis.

Dans ce travail, nous nous intéressons aux problèmes de sécurité des protocoles du routage hiérarchique dans les réseaux de capteurs, en particulier au protocole LEACH, aux vulnérabilités et aux attaques contre ce protocole. Pour cela, nous proposons un nouveau protocole [4] dérivé du protocole LEACH qui offre une bonne protection en tenant compte des caractéristiques limitées des capteurs. L'objectif est donc, d'assurer les services de sécurité les plus importants. Ainsi le protocole proposé peut être robuste face aux attaques qui sont fréquemment menées contre le protocole LEACH et il empêche le nœud malicieux d'être choisi comme CH. Le protocole proposé est fondé sur la combinaison de deux approches : (*i*) la cryptographie basée sur les Courbes Elliptiques pour échanger une clé symétrique ; (*ii*) la cryptographie basée sur la clé symétrique pour faire du cryptage des données et pour calculer un MAC.

Organisation du mémoire

Ce mémoire est organisé de la manière suivante :

Introduction générale : présente le contexte et la problématique visée par ce mémoire.

Chapitre 1 : est une introduction aux réseaux de capteurs sans fil qui sont considérés comme un type particulier de réseaux Ad-hoc.

Chapitre 2 : Ce chapitre se focalise sur le routage dans les réseaux de capteurs, nous y présentons les différentes considérations de conception des protocoles de routage dans ces réseaux, ainsi qu'une classification des différents protocoles de routage proposés dans la littérature. Ensuite, nous décrivons quelques protocoles de routage hiérarchique, l'un de ces protocoles est le protocole LEACH que nous expliquerons en détail dans ce chapitre.

Chapitre 3 : Ce chapitre aborde la problématique de la sécurité dans les réseaux de capteurs, en expliquant les différentes vulnérabilités d'un réseau de capteur, les différents types d'attaques qui peuvent viser ce réseau ainsi que les objectifs de base de sécurité existants et proposés pour faire face à ces menaces. Ensuite, nous présentons les différentes primitives cryptographiques utilisées dans les réseaux de capteurs, les attaques contre le protocole LEACH et les principales solutions proposées pour sécuriser ce protocole.

Chapitre 4 : Dans ce chapitre nous exposons en détails le principe général de notre protocole qui considéré comme une solution de sécurité pour le protocole LEACH.

Chapitre 5 : Ce chapitre effectue une étude de simulation en utilisant l'environnement TinyOS et le simulateur TOSSIM, c'est pour démontrer l'aspect pratique et de mieux analyser les performances de notre protocole.

Conclusion générale : Elle résume notre contribution et définie nos perspectives de recherches pour nos travaux futures.

CHAPITRE 1

GÉNÉRALITÉS SUR LES RÉSEAUX DE CAPTEURS SANS FIL

1.1 Introduction

Les réseaux de capteurs sans fil sont une nouvelle génération des réseaux informatiques, ils sont apparus grâce aux progrès réalisés dans le domaine des technologies de communication sans fil et la miniaturisation du matériel informatique. Ces réseaux facilitent le suivi, la surveillance à distance et le traitement des données dans des endroits ouverts et hostiles. Ils peuvent avoir diverses applications (environnementales, militaires, médicales,... etc.). Un réseau de capteurs est généralement formé d'entités appelées nœuds capteurs qui sont déployées en grand nombre, chaque nœud est composé principalement d'un capteur, d'une unité de traitement, d'un module de communication et d'une batterie. Ces nœuds communiquent entre eux selon une topologie (fixe ou mobile) du réseau, afin d'acheminer les informations vers un point de collecte appelé *puits* ou *station de base*, situé en dehors de la zone de déploiement.

Dans ce chapitre, nous essayons de donner des généralités sur les réseaux de capteurs sans fil, nous tentons d'exposer l'architecture et les technologies des capteurs. Une présentation des réseaux de capteurs et une description de leurs domaines d'applications, leurs contraintes de conception.

1.2 Le nœud capteur

Un nœud capteur est un petit dispositif électronique qui possède généralement des ressources très limitées. Il est capable d'acquérir des données sur son environnement (température, luminosité, pression, etc.), les traiter et les communiquer.

1.2.1 L'architecture matérielle d'un nœud capteur

L'architecture générale présentée dans la littérature est illustrée dans la Figure1.1), cette architecture comprend quatre unités essentielles : l'unité de capture, l'unité de traitement, l'unité de communication et l'unité de contrôle d'énergie[5]. Selon son domaine d'application, elle peut contenir également des modules supplémentaires, à savoir : un système de localisation (GPS : Global Position System), un système générateur d'énergie (cellule solaire) et un mobilisateur permettant de faire déplacer le nœud capteur.

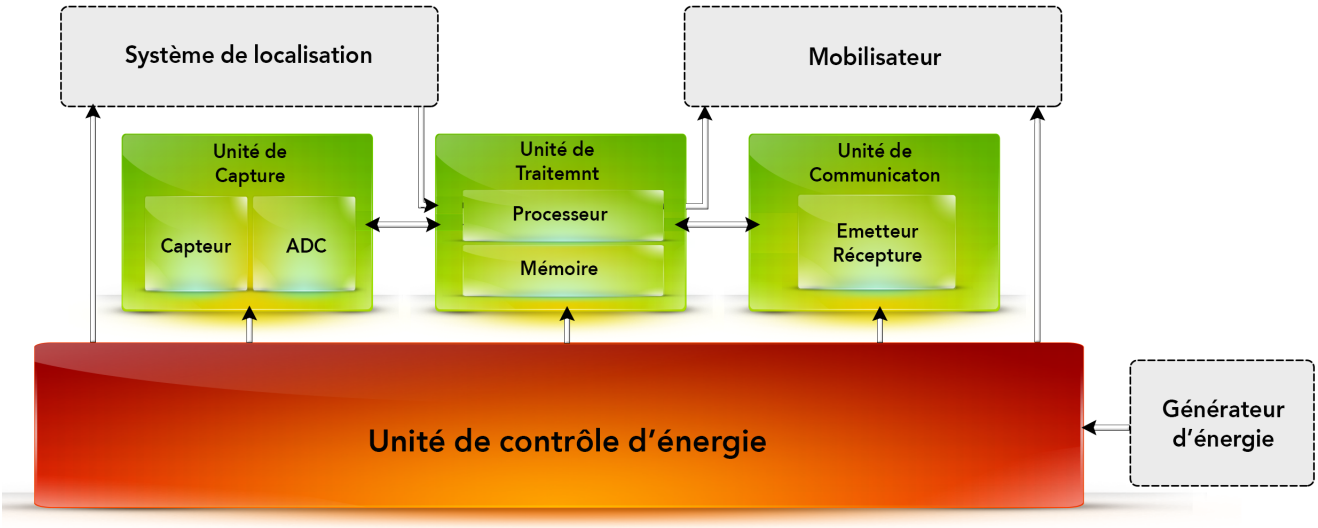


FIGURE 1.1 – Architecture physique d'un nœud capteur [5]

L'unité de capture : Elle est généralement composée de deux sous-unités, le capteur et le convertisseur analogique/numérique appelé ADC (Analog to Digital Converter). Le capteur obtient des mesures sur les paramètres environnementaux et les transforme en signaux analogiques, ADC convertit ces signaux analogiques en signaux numériques et ensuite les transmet à l'unité de traitement.

L'unité de traitement : Elle contient un processeur (microcontrôleur) et une mémoire, cette unité possède deux interfaces : une interface avec l'unité de capture et une autre avec l'unité de communication. Elle obtient les informations en provenance de l'unité de capture et les stocke en mémoire ou les envoie à l'unité de communication, elle est chargée aussi d'exécuter les protocoles de communications qui permettent de mener à bien la collaboration entre les nœuds.

L'unité de communication : Cette unité est responsable d'effectuer toutes les émissions et les réceptions des données sur un médium sans fil.

L'unité de contrôle d'énergie : L'unité de contrôle d'énergie constitue l'un des composants les plus importants dans un nœud capteur[5], elle est responsable de la gestion de l'énergie et de l'alimentation de tous les composants du nœud capteur.

Chaque nœud capteur possède un rayon de communication (Rc) et un rayon de sensation (Rs). La Figure1.2 montre les zones définies par ces deux rayons pour le nœud A, la zone de communication, où le nœud A peut communiquer avec les autres nœuds (B et C) et la zone de sensation, où le nœud A peut capter l'événement.

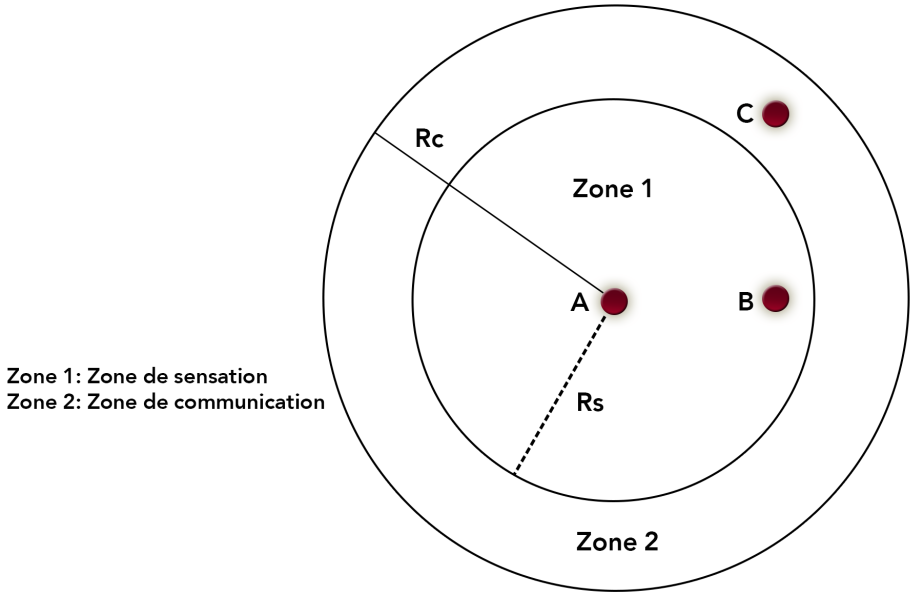


FIGURE 1.2 – Rayons de communication et de sensation d'un capteur

1.2.2 Les technologies des capteurs

Les capteurs se déclinent en une multitude de modèles suivant le type d'application, ils diffèrent suivant la taille, la capacité de calcul et la taille de la mémoire. Il existe dans le monde plusieurs fabricants de capteurs, nous citons Crossbow, Shockfish SA, et Moteiv. La Figure 1.3 montre quelques modèles de capteurs sans fil.

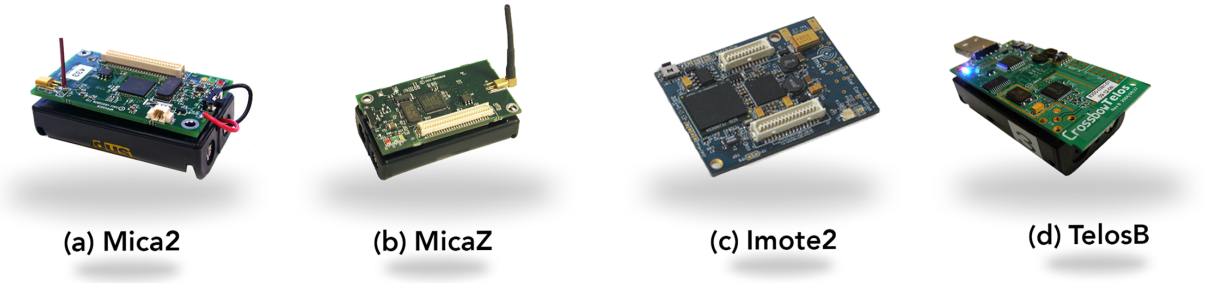


FIGURE 1.3 – Quelques modèles de capteurs sans fil.

La technologie prévalent dans le développement de nœuds capteurs est basée sur le système modulaire, en effet, les nœuds capteurs sont en fait des cartes intégrées qui regroupent l'unité de communication et l'unité de traitement, tandis que l'unité de capture est conçue comme une carte distincte qui peut être raccordée à l'unité principale. Cela permet bien sûr de pouvoir réutiliser les mêmes unités pour différentes applications. Le tableau 1.1 récapitule les caractéristiques de quelques capteurs [6, 7].

Un nœud Mica2 peut être combiné avec une carte MTS310 qui comprenant un capteur de température, un capteur lumière, un capteur de son et un capteur de champ magnétique. De même, on peut combiner le nœud Mica2 avec une carte MTS420 pour le doter d'un capteur d'humidité et d'un capteur de pression barométrique, on peut même prévoir un GPS pour le positionnement géographique, on peut aussi ajouter un capteur de température et un capteur d'humidité en combinant le même nœud avec une carte d'acquisition MDA300. La Figure 1.4 montre quelques cartes intégrées pouvant être combinées avec un nœud.

Modèle	**Micro contrôleur**	**RAM**	**Flash (pour des données)**	**EEprom (configuration)**	**U. communication (type radio)**	**U. capture**	**batterie**
MICA2 (Crossbow)	ATmega 128L	4 KB	128 KB	4 KB	Chipcon CC1000 (38kbps)	Connecteur pour carte de capteurs externe	2.7 - 3.3V
MICAZ (Crossbow)	ATmega 128L	4 KB	128 KB	4 KB	Chipcon CC2420 (250kbps)	Connecteur pour carte de capteurs externe	2.7 - 3.3V
Imote2 (Crossbow)	Intel PXA271	256 KB	32 MB	32 MB	Chipcon CC2420 (250kbps)	Connecteur pour carte de capteurs externe	3.2 - 4.5V
TelosB (Crossbow)	Texas Instruments MSP430	10 KB	48 KB	16 KB	Chipcon CC2420 (250kbps)	Connecteur pour carte de capteurs externe	1.8 - 3.6 V
TinyNode (Shockfish SA)	Texas Instruments MSP430	10 KB	48 KB	16 KB	Semtech XE 1205 (153kbps)	Connecteur pour carte de capteurs externe	2.1 - 3.6V
TmoteSky (Moteiv)	Texas Instruments MSP430	10 KB	48 KB	128 KB	Chipcon CC2420	Connecteur pour carte de capteurs externe	2.1 - 3.6V

Tableau 1.1 – Caractéristiques de quelques nœuds capteurs existants [6, 7]

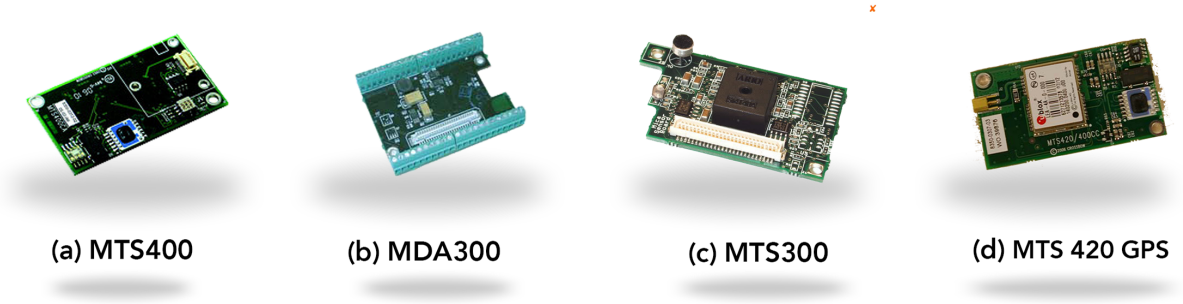


FIGURE 1.4 – Quelques cartes de capture

1.3 Le réseau de capteurs sans fil

Un réseau de capteurs sans fil (RCSF), "Wireless Sensor Network (WSN)" est considéré comme un type particulier des réseaux Ad-hoc où l'infrastructure fixe de communication et l'administration centralisée sont absentes et les nœuds jouent, à la fois, le rôle des hôtes et des routeurs. Il est composé d'un grand nombre de nœuds capteurs distribués sur une zone donnée soit d'une manière aléatoire (avion, missile) ou déterministe (manuelle, robot). Les nœuds capteurs communiquent entre eux par des liens radio pour le partage d'information et le traitement coopératif. Dans ce type de réseau, les nœuds capteurs échangent des informations comme par exemple sur l'environnement pour construire une vue globale de la région contrôlée. Les données collectées par ces nœuds sont acheminées directement ou via les autres nœuds capteurs de proche en proche à un « point de collecte », appelé station de base (Puits ou SINK). Cette dernière peut être connectée à l'utilisateur du réseau via une connexion Internet ou par satellite. La Figure suivante montre un exemple d'un réseau de capteurs.

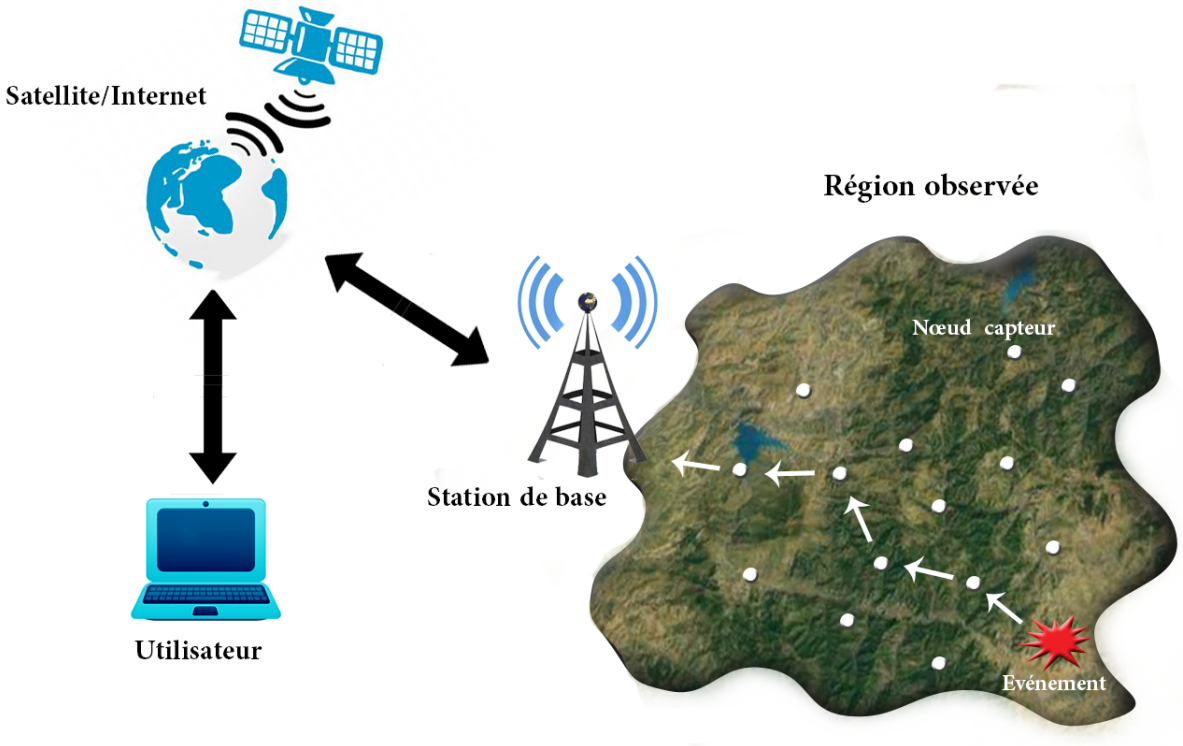


FIGURE 1.5 – Architecture d'un réseau de capteurs sans fil [5]

1.3.1 Les domaines d'applications des réseaux de capteurs

La miniaturisation des micro-capteurs, l'élargissement de la gamme des types de capteurs disponibles (thermique, optique,...etc.), le faible coût ainsi que le support de communication sans fil utilisé ont permis aux réseaux de capteurs de couvrir de nombreuses applications. Les réseaux de capteurs peuvent être classés dans deux catégories selon leur application [8] : un réseau de capteurs intérieur (Indoor WSN) et un réseau de capteurs extérieur (Outdoor WSN). Le réseau de capteur intérieur peut être installé dans les bâtiments, les maisons, les hôpitaux, les usines,...etc, par contre le réseau de capteurs extérieur peut être installé pour le champ de bataille, la marine, le sol et la surveillance atmosphérique, détection d'incendie de forêt, détection d'inondation, études de pollution,...etc.

Dans ce qui suit, nous présentons des exemples des domaines d'applications des réseaux de capteurs sans fil :

Le domaine militaire : Comme dans le cas de plusieurs technologies, le domaine militaire a été le locomotive de la recherche pour les réseaux de capteurs. Ces derniers peuvent être déployés dans un lieu stratégique ou difficile à accéder (hostile) 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 attaques chimiques, biologiques et nucléaires.

Le domaine de la surveillance et le contrôle industriel : L'application des réseaux de capteurs dans ce domaine peut diminuer considérablement les dépenses financières consacrées à la sécurisation des lieux et à la protection des êtres humains. Ainsi les fissures et les altérations dans de grandes structures telles que les ponts ou les bâtiments causées par un séisme ou au vieillissement, peuvent être détectées par des capteurs intégrés dans le matériau de construction comme le béton. La détection de fuites d'eau et la protection des barrages permettent d'éviter des dégâts et peuvent être accomplie en y introduisant des capteurs[9].

Le domaine environnemental : L'utilisation des réseaux de capteurs dans le domaine des applications environnementales pourrait apporter une surveillance des changements environnementaux et de déterminer les valeurs de paramètres (température, pression atmosphérique, lumière) à un endroit donné, comme par exemple le déploiement des nœuds capteurs dans la nature, peut aider à détecter des événements tels que des feux de forêts, des tempêtes ou des

inondations, ceci permet une intervention beaucoup plus rapide et efficace. Le déploiement des nœuds capteurs dans les endroits urbains peut aider à détecter la pollution et analyser la qualité d'air. De même leur déploiement dans les sites industriels empêche les risques industriels tels que la fuite de produits toxiques (gaz, produits chimiques, éléments radioactifs, pétrole,...etc.).

Le domaine médical : Les réseaux de capteurs peuvent être utilisés dans le domaine médical, dans le but d'assurer une surveillance permanente des organes vitaux de l'être humain et cela grâce à des micro-capteurs qui pourront être avalés ou implantés sous la peau (surveillance de la glycémie, détection de cancers,...etc). Ils peuvent aussi faciliter le diagnostic de quelques maladies en effectuant des mesures physiologiques telles que : (la tension artérielle, battements du cœur) à l'aide des capteurs ayant chacun une tâche bien particulière. D'autre part, ces réseaux peuvent détecter des comportements anormaux (chute d'un lit, choc, cri,...etc) chez les personnes handicapées ou âgées vu leurs contraintes de déplacement fréquent.

Les applications domestiques : Grâce au développement technologique, des nœuds capteurs intelligents peuvent être intégrés dans les appareils électroménagers comme des aspirateurs, des fours micro-ondes, des réfrigérateurs, et des magnétoscopes. Ces nœuds capteurs peuvent interagir entre eux ainsi qu'avec les réseaux externes, via Internet ou à travers les satellites pour permettre à l'utilisateur de contrôler plus aisément ces appareils d'une façon distante[8].

1.3.2 Les facteurs de conception des réseaux de capteurs

Un ensemble de facteurs permet de déterminer la conception des réseaux de capteurs. Ces facteurs influent sur l'architecture des réseaux de capteurs et le choix des protocoles à implémenter [5, 10].

La tolérance aux pannes : Quelques nœuds de capteurs peuvent être bloqués ou tomber en panne, en raison de manque d'énergie ou de destruction physique, la panne d'un nœud capteur ne doit pas affecter le fonctionnement global de son réseau. La tolérance aux pannes est donc la capacité de maintenir les fonctionnalités du réseau sans interruption provoquée par la panne d'un nœud capteur.

Le passage à l'échelle : Le nombre de nœuds capteurs déployés dans un réseau pour la surveillance d'un phénomène peut être à l'ordre des centaines voire même des milliers, suivant certaines applications, ce nombre peut atteindre encore quelques millions de nœuds capteurs. Afin de garantir le bon fonctionnement du réseau, les nouveaux schémas de déploiement doivent être capables de travailler avec ce nombre élevé de capteurs. Par ailleurs, ils doivent utiliser la propriété de haute densité dans les réseaux de capteurs ; et donc pouvoir déployer un grand nombre de nœuds dans une petite surface (plusieurs centaines de capteurs dans une région de taille inférieure à 10 mètres de diamètre)[5].

Le coût de production : Le coût de production d'un seul capteur est très important pour l'évaluation du coût global du réseau, si ce dernier est plus cher alors le coût du réseau de capteurs sans fil n'est pas justifié. Par conséquent, la réduction du coût de production d'un nœud capteur est un objectif important pour la faisabilité de la solution des réseaux de capteurs sans fil[5].

Les contraintes matérielles : Toutes les unités du nœud capteur peuvent exiger leur intégration dans un boîtier de taille minimale n'excédant pas généralement quelques centimètres cubes.

La topologie du réseau : La topologie du réseau de capteurs instable est le résultat des trois facteurs essentiels suivants :

Mobilité des nœuds : les nœuds capteurs peuvent être attachés à des objets mobiles qui se déplacent librement, introduisant ainsi une topologie instable du réseau.

Défaillance des nœuds : du fait de l'autonomie énergétique limitée des nœuds, la topologie du réseau n'est pas fixée (les nœuds « morts » sont, d'un point de vue logique, simplement supprimés).

Ajout de nouveaux nœuds : de nouveaux nœuds peuvent facilement être rajoutés, il suffit de placer un nouveau capteur et qu'il soit dans la portée de communication d'au moins un autre nœud capteur du réseau déjà existant.

Le support de transmission : Dans un réseau de capteurs, la communication à multi-sauts entre les nœuds est réalisée avec des liens sans fil, à l'aide de média radio-fréquence, optique ou infrarouge, cependant, le type radio-fréquence est préféré au média optique ou

encore à l'infrarouge, cela est due à sa facilité d'installation et au faible coût qu'il induit.[5].

La consommation d'énergie : Les nœuds capteurs sont des composants micro-électroniques, ils peuvent être seulement équipés que par des sources limitées d'énergie. De plus, dans certaines applications, ces nœuds ne peuvent pas être dotés de mécanismes de rechargement d'énergie, par conséquent, la durée de vie d'un nœud capteur dépend fortement de la durée de vie de la batterie associée.

La consommation d'énergie peut être divisée en trois parties : capture, traitement et communication, elles sont consommées par l'unité de capture, l'unité de traitement et l'unité de communication respectivement. Un exemple de consommation d'énergie d'un capteur de MicaZ est montré sur la Figure 1.6. nous pouvons voir que, parmi ces trois unités, l'unité de communication est celle qui consomme le plus d'énergie.

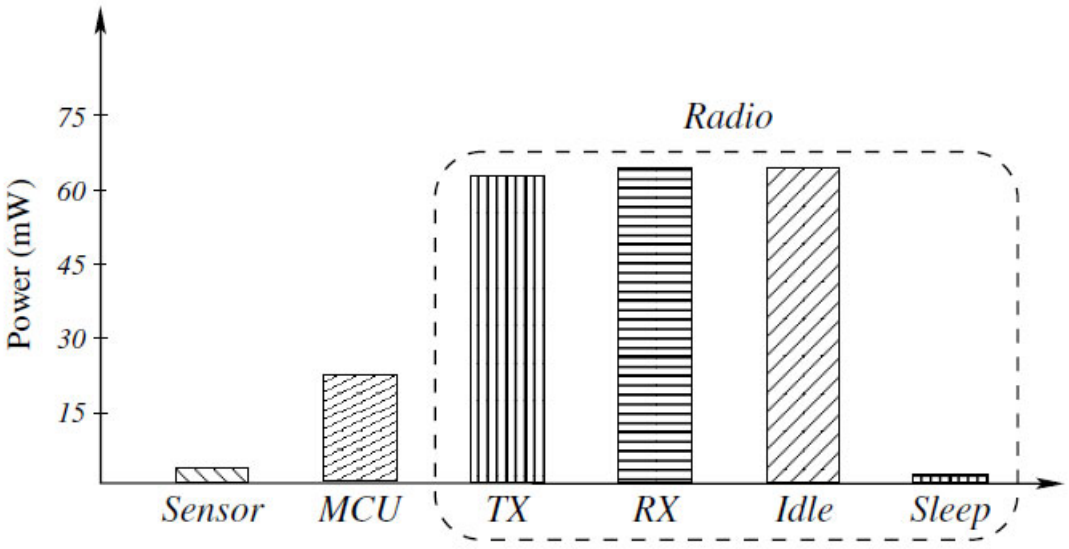


FIGURE 1.6 – Consommation d'énergie d'un capteur MicaZ[11]

L'unité de communication peut se présenter en quatre modes : émission(TX), réception(RX), inactif(Idle) et sommeil (Sleep) ; La différence importante de la consommation d'énergie entre le mode sommeil et le mode inactif, favorise les protocoles de communication qui laissent le nœud dans le mode de sommeil le plus longtemps possible.

1.4 Conclusion

Dans ce chapitre, nous avons essayé de donner quelques généralités sur les réseaux de capteurs sans fils que nous avons considéré comme un type particulier de réseaux Ad-hoc. Un réseau de capteurs sans fil est composé de plusieurs nœuds capteurs déployés dans une région d'observation afin de surveiller un phénomène particulier. Toutefois, la taille réduite des nœuds capteurs, leur coût réduit et leur mode de fonctionnement (thermique, chimique, cinétique, optique, etc.) permettent aux réseaux de capteurs d'envahir plusieurs domaines d'applications.

Dans le prochain chapitre, nous mettrons l'accent sur le routage dans les réseaux de capteurs.

CHAPITRE 2

LE ROUTAGE DANS LES RÉSEAUX DE CAPTEURS SANS FIL

2.1 Introduction

Le routage est un processus par lequel des chemins sont sélectionnés dans un réseau pour transférer des informations d'une source vers une destination, il est à noter que ce processus est très important dans tous les réseaux, puisque, sans routage, la communication sur n'importe quel réseau sera impossible.

Dans les réseaux de capteurs sans fil, le routage constitue un service important qui permet l'acheminement des données captées à partir des nœuds capteurs vers la station de base. Toutefois, le problème qui se pose dans le contexte des RCSF, est de trouver une méthode d'acheminement pour un grand nombre de nœuds existants, dans un environnement caractérisé par de modestes capacités de calcul, des réserves d'énergie et des capacités mémoire limitées. Cependant, l'utilisation des protocoles de routage conçus pour les réseaux Ad-Hoc traditionnels dans les réseaux de capteurs sans fil est inappropriée, en raison des caractéristiques par lesquelles se distinguent les deux types de réseaux, d'où la nécessité de développer de nouveaux protocoles de routage spécifiques aux réseaux de capteurs.

Dans ce chapitre, nous présentons les principales considérations qui influent sur la conception des protocoles de routage dans les réseaux de capteurs. Nous exposons aussi les différentes classes des protocoles de routage développés pour ce type de réseau. Ensuite, nous essayons de donner les principaux protocoles de routage hiérarchique de la littérature, l'un de ces protocoles est le protocole LEACH que nous expliquerons en détail.

2.2 Les considérations de conception des protocoles de routage dans les réseaux de capteurs

La conception d'un protocole de routage pour les réseaux de capteurs, exige la prise en compte de plusieurs considérations qui sont d'une grande importance, puisque elles servent de directives guidant les concepteurs pour qu'une communication efficace puisse être assurée. Comme les réseaux capteurs impliquent des taches spécifiques et des contraintes sur les ressources, leur efficacité et survivabilité dépendent considérablement de la qualité des protocoles utilisés. Dans ce qui suit, nous essayons d'exposer un certain nombre de considérations qui sont indispensables pour la conception des protocoles de routage dans les réseaux de capteurs.

2.2.1 La mobilité

La plupart des architectures des réseaux de capteurs supposent que les nœuds sont fixes. Cependant, la mobilité des nœuds capteurs et la station de base est parfois nécessaire dans plusieurs applications [1]. Le routage des messages à partir, ou vers des nœuds mobiles est plus difficile puisque la stabilité des chemins et l'imprévisibilité des positions des nœuds ajoute une difficulté importante et nécessite une consommation d'énergie supplémentaire. Ceci s'explique par le fait que la mobilité des nœuds augmente le nombre de paquets perdus, et ainsi le nombre de retransmissions.

2.2.2 La connectivité

Dans les réseaux de capteurs, la forte densité des nœuds empêche les nœuds qui forment le réseau d'être totalement isolés les uns des autres, par conséquent, on s'attend à ce que des nœuds de capteurs soient fortement connectés. Cependant, la topologie du réseau peut être variable et les nœuds en panne peuvent être supprimés.[1].

2.2.3 La couverture

Dans les réseaux de capteurs, chaque nœud obtient une vue de l'environnement. cette action est limitée à la fois dans la portée et dans la précision, elle peut juste couvrir une zone limitée de l'environnement. En conséquence, la couverture d'une région est également un paramètre de conception important dans les réseaux de capteurs [1].

2.2.4 Le déploiement des nœuds

La méthode de déploiement des nœuds d'un réseau de capteurs dépend de l'application et affecte la performance du protocole de routage. Cette opération peut être déterministe ou aléatoire (auto-organisée). Dans le premier cas, les nœuds capteurs sont placés manuellement et les données sont routées à travers des chemins prédéterminés. Mais, vu l'hostilité de la zone à surveiller ou l'immensité du réseau, une grande majorité des applications dans les réseaux de capteurs utilisent un déploiement aléatoire d'un grand nombre de capteurs[12], ces derniers ne sont pas uniformément répartis sur le champ de capture, ce qui implique que certaines régions du champ de déploiement puissent bénéficier d'une meilleure connectivité par rapport à d'autres et les phénomènes captés dans ces régions peuvent êtres routés plus facilement.

2.2.5 La consommation d'énergie

L'énergie du nœud capteur est utilisée pour alimenter les autres unités, telle que l'unité de capture, l'unité de traitement et l'unité de communication. La miniaturisation des nœuds capteurs impose l'utilisation de ressources d'énergie limitées telles que les batteries. La durée de vie d'un nœud dépend de la durée de vie de sa batterie, car celle-ci est généralement irremplaçable et dépourvue d'une unité de rechargement, de ce fait, la durée de vie du réseau de capteurs en entier dépend fortement de celle des nœuds capteurs. Des études concernant les réseaux de capteurs ont montré que la phase de transmission est la plus consommatrice en énergie. En effet, l'exemple donné dans la référence [13] montre que la transmission de 1Kb de données sur une distance de 100 mètres consomme une énergie de 3 Joules. Tandis qu'un processeur avec une modeste capacité de calcul de 100 MIPS(Million of Instructions Per Second) peut exécuter 300 millions d'instructions avec la même quantité d'énergie. Cet exemple illustre qu'il serait plus judicieux pour un capteur, au moment du routage, de faire un traitement local et de ne transmettre que les informations utiles, plutôt que d'envoyer

toutes les données captées dans leur état brut. Dans les réseaux de capteurs, l'efficacité en consommation d'énergie du nœud capteur représente alors un facteur de performance significatif, qui influe directement sur la durée de vie du réseau en entier. Pour cela, les concepteurs peuvent au moment du développement des protocoles, négliger les autres métriques de performance tels que les délais de transmission des données captées, au détriment de cette considération[14].

2.2.6 Le modèle de livraison des données

le modèle de livraison des données vers la station de base dépend de l'application du réseau de capteurs, donc, il peut être continu (time-driven), orienté événement (event-driven), orienté requête (query-driven) et hybride. Dans le modèle continu, chaque nœud capteur envoie les données périodiquement. Les modèles orienté événement et orienté requête génèrent un trafic de données après l'occurrence d'un certain événement (par exemple, détection des feux de forêt), ou lors d'une réponse à une requête générée et envoyée par la station de base. Le modèle hybride est une combinaison des trois autres modèles. La conception du protocole de routage est fortement influencée par le modèle de livraison des données concernant la consommation d'énergie [1].

2.2.7 Les capacités des nœuds

Les nœuds capteurs ont des capacités de calculs, de stockage et de communication limitées. Ces nœuds jouent non seulement un rôle crucial dans l'acquisition et le traitement des données captées, mais ils jouent aussi un rôle de routeur, et participent aux communications entre les autres nœuds capteurs et la station de base. Ceci peut nécessiter une importante complexité des algorithmes utilisés. Cependant, la faible capacité de calcul et de stockage des nœuds capteurs empêche l'utilisation d'algorithmes très complexes, de ce fait les concepteurs de protocoles de routage pour les réseaux de capteurs doivent donc prendre en compte ces limitations. Les protocoles proposés doivent englober des opérations simples et peu exigeantes en capacité de calcul et de stockage.

2.2.8 L'agrégation des données

Dans les réseaux de capteurs, les données produites par les nœuds capteurs sont très reliées, ce qui implique l'existence de redondances de données. Les paquets similaires des

différents nœuds peuvent être agrégés pour réduire le nombre de transmissions. Autrement dit, l'agrégation des données est la combinaison des données provenant de différentes sources selon une certaine fonction d'agrégation (par exemple : moyenne, suppression, minimum, maximum,...,etc.). Cette technique a été utilisée pour atteindre l'efficacité énergétique et l'optimisation de transfert de données dans un certain nombre de protocoles de routage.

2.2.9 L'hétérogénéité des nœuds

Dans de nombreuses études, tous les nœuds de capteurs ont été supposés homogènes cela veut dire ayants une capacité égale en termes de calcul, de communication et de ressource énergétique. Ces nœuds capteurs peuvent rapidement épuiser leurs ressources énergétiques, et dégrader ainsi les performances du réseau. Puisque ils accomplissent plusieurs tâches à la fois, telles que, la capture, le traitement et le routage des données. L'existence d'un ensemble hétérogène de capteurs soulève de nombreuses questions techniques liées au routage des données, citons quelques applications qui pourraient exiger plusieurs types de capteurs, conçu pour la surveillance de température, de pression, de l'humidité de l'environnement, la détection des mouvements ou la prise des images pour surveiller des objets mobiles. Même la capture et la livraison des données peuvent être produites par ces capteurs à différents taux [1].

2.2.10 La tolérance aux pannes

La tolérance aux pannes est donc la capacité de maintenir les fonctionnalités du réseau sans interruptions malgré la défaillance d'un ou plusieurs de ses nœuds. Les protocoles de routage conçus pour les réseaux de capteurs doivent assurer le bon fonctionnement du réseau, même si quelques nœuds tombent en panne. A cet effet, ils doivent procéder à la formation de nouvelles liaisons et router les données collectées à la station de base.

2.2.11 Le passage à l'échelle

Le nombre de nœuds de capteurs déployés dans une région de capture peut être à l'ordre des centaines ou des milliers ou bien plus. N'importe quel schéma de routage doit pouvoir travailler avec ce grand nombre de nœuds capteurs. En outre, les protocoles de routage devraient exploiter la nature fortement dense des réseaux de capteurs [1].

2.2.12 Le support de transmission

Dans un réseau de capteurs, les nœuds sont reliés par une architecture sans fil, pour permettre au réseau d'accomplir la totalité de ses tâches. Le média de transmission peut être un support optique ou des fréquences radio. Pour ce type du réseau, les unités de transmission intégrées au niveau des nœuds doivent être de petite taille et a faible consommation d'énergie[14].

2.2.13 La qualité de service

Certaines applications dans les réseaux de capteurs exigent que les données devraient être livrées dans un temps spécifique, sinon elles sont inutiles, par exemple, dans le domaine de la sécurité, la détection d'intrusion doit être acheminée au plus bref délai vers la station de base. Les protocoles de routage basés sur la qualité de service dans les réseaux de capteurs essayent de répondre à quelques exigences de qualité de service (délais de transmission ou niveau de fiabilité) et cela pendant le routage des données et doivent faire l'équilibre entre la qualité de service de routage et la consommation d'énergie.

2.3 La classification des protocoles de routage dans les réseaux de capteurs sans fil

Les protocoles de routage dans les réseaux de capteurs peuvent être classés en deux catégories : la structure du réseau (Network structure) et la stratégie de routage du protocole (Protocol operation). Les protocoles de routage basés sur la structure du réseau peuvent à leur tour être classés en trois sous catégories : routage plat (Flat network routing), routage hiérarchiques (Hierarchical network routing) et routage géographique (Location based routing). Selon leurs stratégies, les protocoles de routage sont encore classés en quatre sous catégories : routage basé sur les chemins multiples (Multi-path based routing), routage basé sur les requêtes (Query based routing), routage basé sur la négociation entre les nœuds (Negociation based routing) et routage avec qualité de service (Quality of service "QoS" based routing)[1].

La Figure 2.1 montre une taxonomie des protocoles de routage dans les réseaux de capteurs

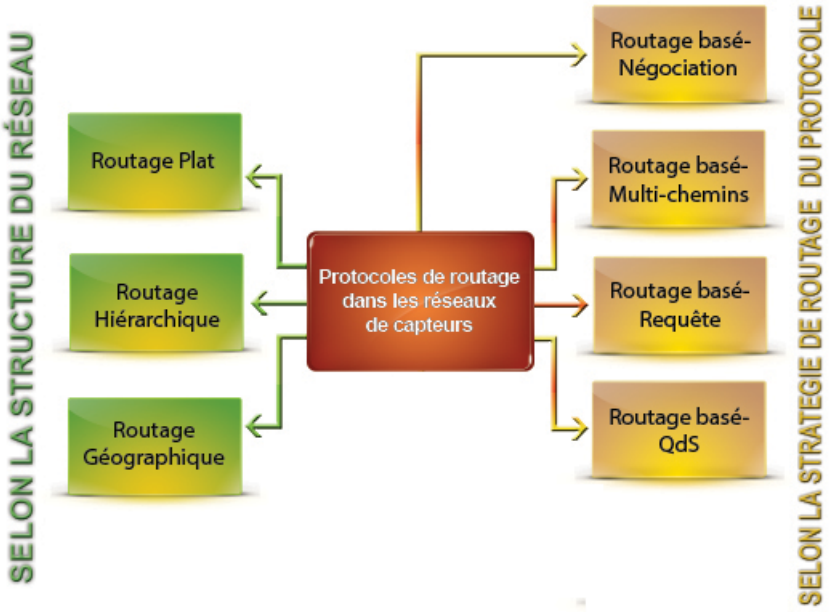


FIGURE 2.1 – Classification des protocoles de routage dans les réseaux de capteurs [1]

2.3.1 Classification selon la structure du réseau

Dans cette section, nous présentons les différentes techniques de routage basé sur la structure du réseau.

Le routage plat : Ce type de routage considère que tous les nœuds ont le même rôle et ils collaborent entre eux pour accomplir la tache globale du réseau. Vu le nombre important des nœuds de capteurs, il est difficile d'assigner un identifiant global à chaque nœud.

Pour pailler à ce problème, les protocoles de routage de ce type, sont basés sur le principe centrés données "data-centric[15], où la station de base envoie des requêtes à certaines régions du réseau et elle attend le retour des données provenant des capteurs situés dans ces régions. Puisque les données sont demandées à travers des requêtes, la désignation des attributs est nécessaire pour indiquer aux noeuds capteurs les caractéristiques des données sollicitées par la SB. Il faut tenir également à utiliser l'agrégation des données pour éviter le gaspillage d'énergie dû aux redondances de données [16].

Le routage hiérarchique : Il permet d'une façon considérable à renforcer le passage à l'échelle, l'efficacité énergétique et l'augmentation de la durée de vie du réseau [17]. Deux approches sont dérivées de ce type de routage :

(*i*) *l'approche basée sur les clusters* (cluster-based approach), l'idée consiste à regrouper les nœuds en groupes dits (Clusters) où chaque cluster est représenté par un nœud appelé

(CH : Cluster Head), ce dernier permet l'agrégation des données afin de diminuer le nombre de messages transmis à la station de base, par conséquent, cette approche réduit la consommation d'énergie [18]. On peut distinguer deux types de sélection du CH : le premier est physique, où le choix du représentant est basé sur les capacités physiques du nœud capteur (plus de mémoire, plus de capacité de stockage,...,etc.) ; dans ce cas, on parle d'un réseau de capteurs hétérogènes. Dans le deuxième type, les nœuds capteurs utilisés sont physiquement identiques, et le choix du CH se base sur plusieurs critères, à savoir : l'identifiant, la distance à la station de base, l'énergie résiduelle,...etc.

(*ii*) *l'approche basée sur la chaîne* (chaine-based approach), l'idée consiste à former une chaîne des nœuds de capteurs où chaque nœud ne communique qu'avec les deux nœuds reliés directement à lui par cette chaîne. En plus, il n y a qu'un seul nœud (leader) qui rassemble les données détectées par l'ensemble des nœuds du réseau en utilisant la chaîne formée, puis il transmet le résultat à la station de base après avoir effectué quelques traitements de données (par exemple : l'agrégation des données).

Le routage géographique : Ce genre de routage est utilisé dans les applications qui nécessitent à déterminer les coordonnées géographiques des différents nœuds capteurs. En fait, dans ce type de routage, le calcul de routes se base sur la position déterminée pendant que le nœud capteur peut obtenir sa position grâce à la communication par satellite, cela est assuré lorsque le nœud est équipé d'un récepteur GPS (Global Positioning System) de faible énergie[19]. Le nœud capteur peut être également localisé par l'évaluation de la position au sein du réseau.

2.3.2 Classification selon la stratégie de routage du protocole

Suivant la stratégie de routage, les protocoles peuvent être distingués en quatre sous catégories à savoir : le routage basé sur les chemins multiples, le routage basé sur les requêtes, le routage basé sur la négociation entre les nœuds et le routage avec qualité de service.

Routage basé sur les chemins multiples : Les protocoles de cette sous catégorie créent plusieurs chemins au lieu d'un chemin unique entre une source et une destination. Ces chemins alternatifs sont presque indépendants, autrement dit, ils ne partagent qu'un nombre réduit (voir nul) des nœuds de capteurs. Ceci permet d'avoir des chemins de secours lorsque le chemin principal est défaillant et par conséquent, d'assurer une bonne fiabilité du routage.

Cependant, ces chemins alternatifs sont crées aux dépens d'une consommation d'énergie et d'un trafic de contrôle supplémentaires.

Routage basé sur les requêtes **:** Dans ce type de routage, les nœuds propagent de voisin en voisin une requête de données, émise par la station de base. Cependant, les nœuds qui possèdent les données requises doivent les envoyer à travers le chemin inverse de la requête. Généralement, ces requêtes sont décrites dans un langage naturel ou des langages d'interrogation spécifiques (par exemple SQL : Structured Query Language).

Routage basé sur la négociation entre les nœuds **:** Cette approche permet d'éliminer la transmission des données redondantes sur la base de la négociation, et par la suite, il permet la réduction de la consommation d'énergie. L'idée principale est qu'un nœud capteur doit supprimer l'information double et d'empêcher l'envoi des données redondantes au prochain capteur ou à la station de base, en échangeant des messages de négociation, appelés métadonnées avant même la transmission effective des données [20, 21].

Routage avec qualité de service **:** Le principe des protocoles de ce type du routage, se base sur le fait que le réseau doit être capable de satisfaire certaines métriques de qualité de service, comme le délai de transmission ou le niveau de fiabilité, tout en tenant compte de la consommation d'énergie. Ce genre de protocoles est utilisé dans les applications qui ont des exigences en temps-réel (par exemple : les applications de surveillance dans les centrales nucléaires)

2.3.3 Classification des principaux protocoles de routage dans les réseaux de capteurs

Le tableau 2.1 présente des principaux protocoles de routage dans les réseaux de capteurs sans fil.

protocoles	**Structure du réseau**				**stratégie de routage**			
	plat	**hiérarchique(Cluster)**	**hiérarchique(chaîne)**	**géographique**	**chemins multiples**	**les requêtes**	**la négociation entre les nœuds**	**la qualité de service**
LEACH [2]		X						
TEEN [22]		X						
APTEEN [23]		X						
PEGASIS [24]			X					
PEGASIS hiérarchique [25]		X	X					
DD(Direct Diffusion)[20]	X				X	X	X	
SPIN[21]	X				X	X	X	
GAF[19]				X				
GEAR[26]				X				
SAR[27]				X		X	X	X

Tableau 2.1 – Classification des principaux protocoles de routage dans les réseaux de capteurs

2.4 Présentation de quelques protocoles de routage hiérarchique

Dans cette section, nous allons essayer d'exposer les principaux protocoles de routage hiérarchique de la littérature.

2.4.1 LEACH

LEACH (Low-Energy Adaptive Clustering Hierarchy)[2] est un protocole de routage hiérarchique basé sur les clusters, il est destiné aux RCSF. Son principal avantage est de minimiser la consommation d'énergie des éléments du réseau, pour cela, il utilise la rotation aléatoire de la position de CH pour distribuer équitablement la charge d'énergie entre les différents nœuds du réseau. L'idée de ce protocole, consiste à organiser les nœuds du réseau en groupes dits (clusters) en se basant sur la force reçue du signal, où chaque cluster est représenté par un nœud appelé (CH : Cluster-Head) comme le montre la Figure 2.2. Une fois que le CH obtient toutes les données à partir des nœuds appartenant à son cluster, il se charge d'agréger ces données puis de les transmettre directement à la SB. Ceci permet d'économiser de l'énergie puisque les transmissions des données vers la SB seront effectuées uniquement par les CHs plutôt que par tous les nœuds capteurs. LEACH est totalement distribué et n'a besoin d'aucune connaissance globale du réseau.

Le protocole LEACH se déroule en plusieurs itérations (rounds) tandis que ces dernières sont des intervalles de durée égale. Chaque itération se compose de deux phases : *phase d'installation* (Setup phase) et *phase de transmission* (Steady-state phase). L'algorithme du protocole LEACH est présenté dans l'Algorithme 1

Phase d'installation;

1. $CH \rightarrow broadcast : id_{CH}||adv$

2. $SN \rightarrow CH : id_{SN}||id_{CH}||join\_req$

3. $CH \rightarrow SN : id_{CH}||id_{SN}||Timeslot(SN)$

Phase de transmission;

4. $SN \rightarrow CH : id_{SN}||id_{CH}||info$

5. $CH \rightarrow SB : id_{CH}||id_{SB}||aggr\_info$

Algorithme 1: Le protocole LEACH [56]

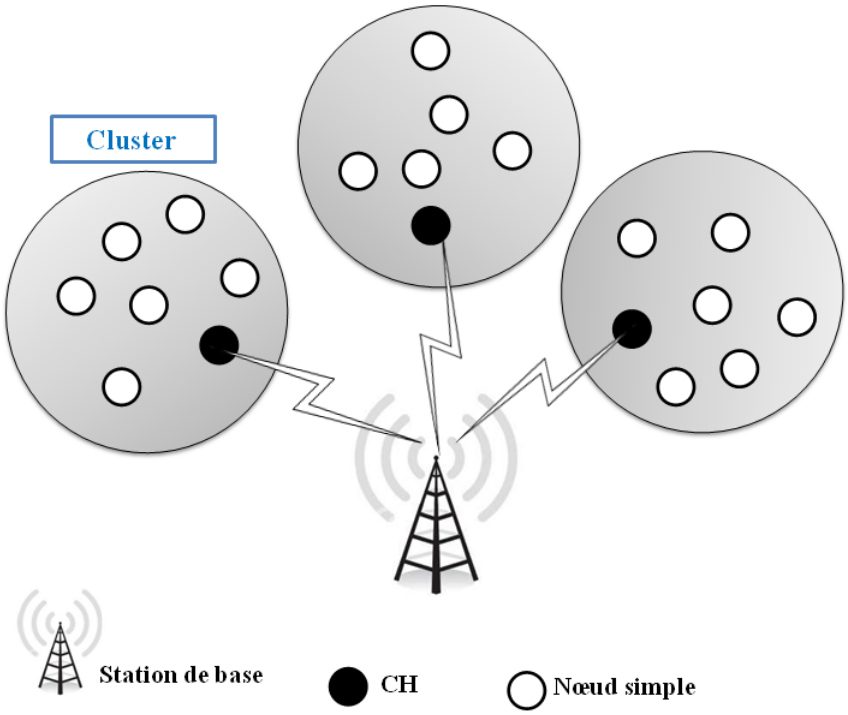


FIGURE 2.2 – Architecture en clusters dans le protocole LEACH

Initialement, le pourcentage désiré des CHs devrait être entre 5% et 15% du nombre total des nœuds du réseau. Si ce pourcentage n'est pas respecté, cela provoque une grande dissipation d'énergie. En effet, si le nombre des CHs dépasse 15%, on aura un nombre important de CHs qui se consacrent à des tâches très coûteuses en termes d'énergie. Par contre, si le nombre de CHs est très petit (inférieur à 5%), ces derniers vont gérer des clusters de grandes tailles, et cela mènera à un épuisement rapide de leurs batteries. Notons que ce pourcentage est fixé et il est inchangé durant toutes les itérations.

Phase d'installation (Setup phase)

Cette phase commence par l'annonce d'une nouvelle itération par la SB. Par la suite, le processus d'élection est déclenché pour choisir des futurs CHs. A cet effet, chaque nœud n prend une valeur aléatoire x,$(0 < x < 1)$, si x est inferieur à une valeur seuil $T(n)$, le noeud n se désigne CH, sinon il devient membre du cluster. Par ailleurs, la valeur $T(n)$ est calculée comme suit :

$$T(n) = \begin{cases} \frac{P}{1-P*(r \bmod P^{-1})} & \text{si} \quad n \in G \\ 0 & \text{sinon} \end{cases}$$

Où :

- ❖ P est le pourcentage désiré de CHs dans le réseau.
- ❖ r est le numéro de l'itération courante.
- ❖ G est l'ensemble des nœuds n'ayant pas été sélectionnés comme des CHs, durant les P^{-1} dernières itérations.

Une fois un simple nœud devient CH, alors il diffuse un message de notification *adv* (*Advertisement message*) à l'ensemble des nœuds pour les inviter à se joindre à leurs clusters (étape 1 -Algorithme 1). Par la suite, les nœuds non-CH collectent les messages de notification et décident de leurs appartenances à un cluster. Tandis que, le choix du CH qui les représente est basé sur la force du signal reçu de l'*adv*, autrement dit, le CH ayant le signal d'*adv* le plus fort est choisi. Après cela, ils envoient à leur tour un message de type *Join_req* au CH choisi, pour l'informer de cette décision (étape 2 -Algorithme 1).

A l'intérieur de chaque cluster, le CH établi un planning TDMA et assigne à chaque nœud membre une période temporelle *Timeslot* durant laquelle il lui est permis de transmettre ses données à son CH (étape 3 -Algorithme 1), de cette façon, le nœud membre peut ainsi mettre en veille son système de communication en attendant son tour, ce qui permet une économie d'énergie.

Phase de transmission (Steady-state phase)

Pendant cette phase, chaque nœud membre transmet ses données captées à son CH pendant son propre Timeslot (étape 4 -Algorithme 1). Cependant, le CH collecte les données provenant de tous les membres de son cluster, les agrège et les transmet à la SB (étape 5 -Algorithme 1).

Après un certain temps prédéterminé, une nouvelle itération est lancée en revenant à la première phase. Ce processus est répété jusqu'à ce que tous les nœuds du réseau soient élus en tant que CH, une seule fois et cela durant toutes les itérations. Dans ce cas, le numéro de l'itération est réinitialisé à 0.

2.4.2 TL-LEACH

TL-LEACH (Two-Levels Hierarchy for Low-Energy Adaptive Clustering Hierarchy) [28] Est une extension proposée de l'algorithme LEACH, il utilise deux niveaux de CHs (primaire et secondaire) au lieu d'un seul niveau. Dans ce protocole, le CH primaire dans chaque cluster communique avec les CHs secondaires, et ces derniers communiquent à leur tour

avec les nœuds capteurs dans leur sous groupe (Sub cluster). L'agrégation des données peut également être effectuée comme dans LEACH. En plus, la communication au sein d'un Cluster est toujours programmée en utilisant un planning TDMA. La phase d'installation pour TL-LEACH consistera en formant les clusters et de sélectionner les CHs primaires et secondaires, en utilisant le même mécanisme que LEACH.

La transmission des données du nœud source vers la SB est réalisée dans deux étapes :

- ❖ Les CHs secondaires collectent les données à partir de tous les nœuds capteurs appartenant à leurs clusters respectifs. L'agrégation des données peut être effectuée à ce niveau.
- ❖ Les CHs primaires collectent les données à partir de tous les CHs secondaires appartenant à leurs clusters respectifs. L'agrégation des données peut également être mise en œuvre au niveau de CH primaire.

La structure à deux niveaux de TL-LEACH réduit la quantité de nœuds qui doivent être transmis à la SB, réduisant la consommation d'énergie totale.

2.4.3 TEEN et APTEEN

Deux protocoles de routage hiérarchique appelés TEEN (*Threshold-sensitive Energy Efficient sensor Network protocol*) et APTEEN (*Adaptive Periodic* TEEN) sont proposé dans les références [22, 23].

Ces protocoles ont été proposés pour les applications critiques en termes de temps. Dans TEEN, les nœuds captent de manière continue, mais la transmission de données n'est pas faite fréquemment. Il s'agit d'un protocole conçu pour être sensible aux changements soudains de certains attributs captés dans le réseau de capteurs (par exemple, la température). La majorité du comportement de TEEN est semblable au protocole LEACH. Cependant, quelques différences existent. Après la formation des clusters, chaque CH transmet deux seuils à ses membres au lieu de transmettre un planning TDMA, ces deux seuils, notés H_T (*hard threshold*) et S_T (*soft threshold*) pour l'attribut détecté.

H_T : détermine la valeur minimum au delà de laquelle les membres sont susceptibles de transmettre leurs rapports des données.

S_T : spécifie le changement minimal obligeant le nœud à transmettre un nouveau rapport des données.

Lorsque la valeur captée a dépassée H_T, le nœud doit envoyer le rapport des données au CH. Il ne retransmet un nouveau rapport que si la différence entre la valeur courante et la valeur précédente dépasse S_T. TEEN permet de construire un comportement réactif, ce qui permet de minimiser le nombre de messages et d'économiser de l'énergie. Cependant, l'inconvénient principal de ce protocole est que, si les seuils H_T et S_T ne sont pas atteints, les nœuds ne communiqueront jamais, et aucune donnée ne sera transmise à la SB.

Pour surmonter à ces limitations, les auteurs ont proposé une extension de TEEN appelée APTEEN (*Adaptive Periodic* TEEN) qui permet de trouver un compromis entre les protocoles hiérarchiques proactifs (comme LEACH) et les protocoles hiérarchiques réactifs (comme TEEN). Dans APTEEN le CH émit à ses membres les paramètres suivants :

- ❖ Les Attributs : représentent la tâche demandée au capteur.
- ❖ Les deux seuils : H_T et S_T.
- ❖ Un compteur de temps TC (*Count Time*) : représente la durée maximum du temps entre deux transmissions successives d'un nœud.
- ❖ Un planning TDMA : permettant d'assigner à chaque nœud un intervalle fini de temps appelé slot.

APTEEN permet aux nœuds de capteurs d'effectuer le même mécanisme de seuil du protocole TEEN, et dans le cas où le nœud ne transmet pas de données pendant une période dépassant de temps TC, il devrait effectuer un transfert de données vers CH pendant son slot TDMA d'émission.

2.4.4 PEGASIS et PEGASIS hiérarchique

Dans la référence [24], Lindsey et Raghavendra ont proposé une version améliorée de LEACH appelée PEGASIS. L'idée de base de ce protocole est de former une chaîne entre les nœuds de sorte que chaque nœud ne communique qu'avec les deux nœuds reliés directement à lui par cette chaine, ce qui permet de minimiser la dissipation d'énergie dépensée par les nœuds de capteurs. A chaque itération (*round*), il y a un seul nœud (*leader*) de la chaîne est sélectionné pour assurer la transmission vers la SB. Les données collectées sont transmises d'un nœud à un autre qui les agrège jusqu'à ce qu'elles arrivent à un nœud (*leader*) celui la va les transmettre à la SB.

La Figure 2.4 montre un exemple de construction de chaîne, le nœud *C0* émit ses données au nœud *C1*, qui agrège ses propres données avec celles du nœud *C0*, puis les communique

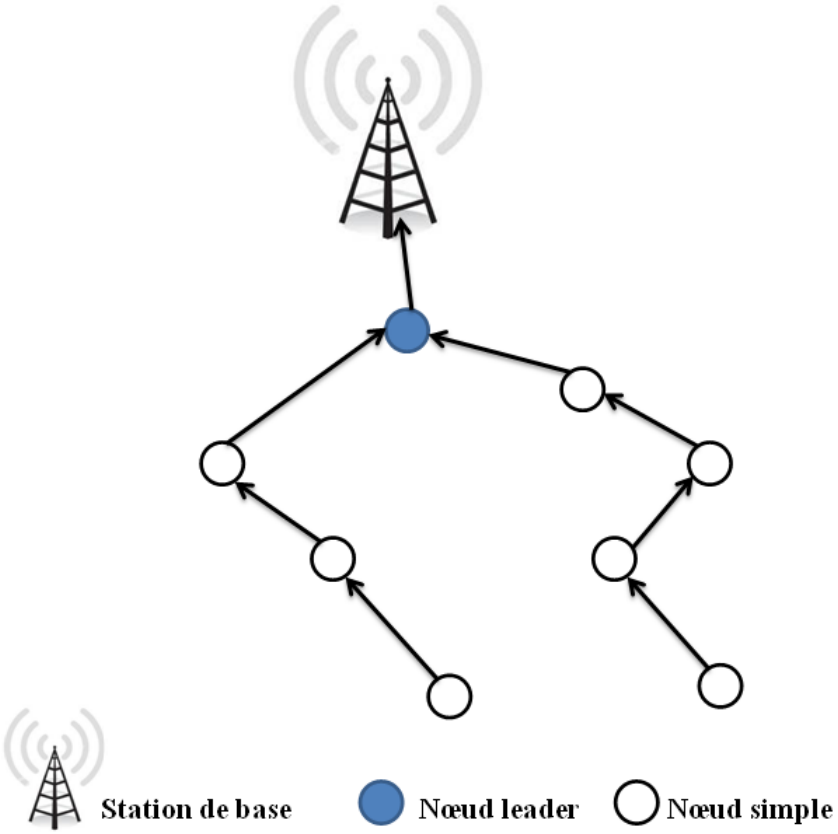


FIGURE 2.3 – Routage hiérarchique basé sur la chaine

au nœud leader *C2*. Le nœud *C2* passe la main au nœud *C4* qui transmet ses données au nœud *C3*. Ce dernier agrège éventuellement ses données avec celles du nœud *C4* et les envoie à son tour au nœud *C2*, qui est en attente pour recevoir des données de ses voisins, puis les agrège avec ses propres données et transmet un message unique à la station de base.

Une comparaison avec LEACH montre que PEGASIS permet de mieux prolonger la durée de vie du réseau que LEACH. Un tel gain est atteint grâce à l'élimination de la surcharge (overhead) causée par la formation dynamique de clusters dans LEACH, la réduction du nombre de nœuds transmettant les données à la station de base. Cependant PEGASIS présente un retard excessif des nœuds éloignés dans la chaîne.

Ainsi PEGASIS ne garantit pas la livraison des données à la station de base à chaque itération puisque un nœud peut tomber en panne lors de son rôle de leader.

PEGASIS hiérarchique [25] est une extension améliorée du protocole PEGASIS dont le but est de réduire les délais de transmission des données vers la station de base. Dans le protocole PEGASIS hiérarchique, l'organisation des nœuds appartenant au même cluster sous

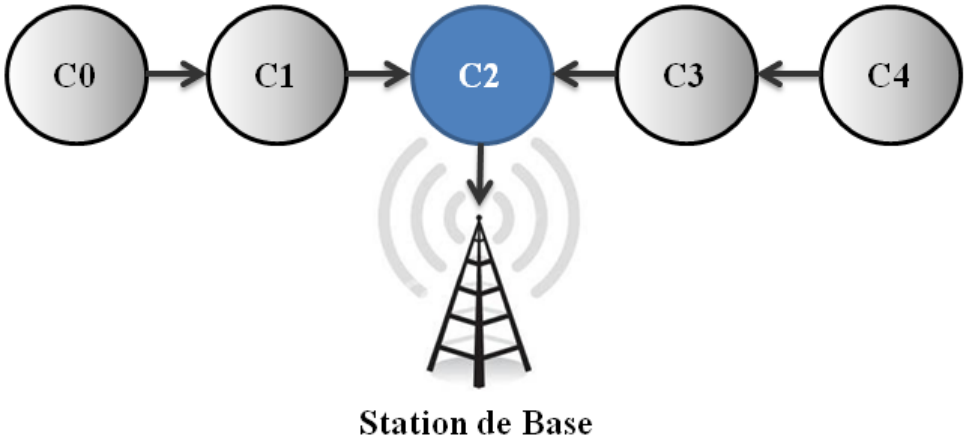


FIGURE 2.4 – le chainage dans PEGASIS [24]

forme d'une chaîne permet d'améliorer et de réguler la dissipation d'énergie, ce qui permet de réduire la charge sur le CH. En effet, les nœuds communiquent uniquement avec leurs voisins et non pas directement avec CH, ce qui économise d'avantage l'énergie. L'agrégation de données au niveau de chaque nœud dans la chaîne réduit la quantité de données échangées entre les nœuds et leur CH, la Figure2.5 montre comment les nœuds seront organisés à l'intérieur des clusters.

Dans PEGASIS Hiérarchique, chaque nœud transmet ses données à son proche voisin, ce dernier agrège les données reçues avec les siennes et les transmet à son voisin jusqu'à atteindre le CH qui les transmet directement à la SB.

2.4.5 EECS

Le protocole EECS (Energy Efficient Clustering Scheme)[29] est un algorithme de regroupement dans lequel les candidats Cluster-Head concurrent pour devenir CH durant un tour donné (itération). Cette concurrence inclut des candidats diffusant leurs énergies résiduelles à des candidats voisins. Si un nœud donné ne trouve pas un autre nœud ayant une énergie résiduelle plus grande, il devient automatiquement CH. La formation des clusters est différente de celle de LEACH car ce dernier forme des clusters basé sur la distance minimale entre les nœuds et le CH. Le protocole EECS met le point sur le fait que les clusters plus distants de la station de base nécessite plus d'énergie pour la transmission que ceux qui sont plus proches, améliorant ainsi la distribution de l'énergie tout au long de l'ensemble du réseau et entraînant une meilleure utilisation des ressources et donc une longue durée de vie du réseau.

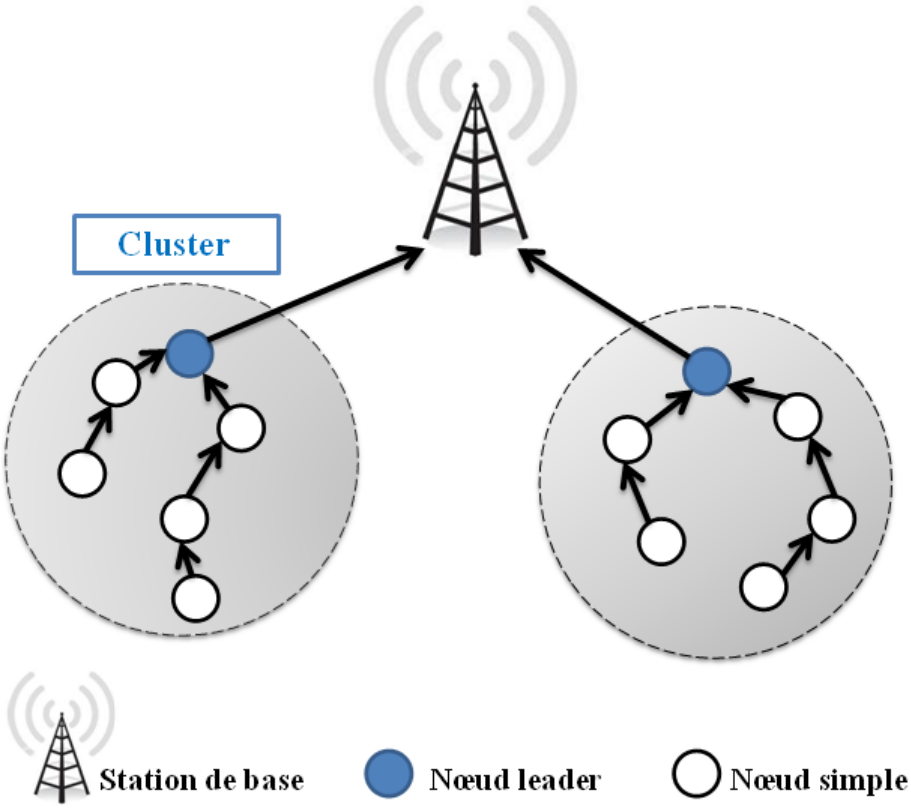


FIGURE 2.5 – Routage hiérarchique basé sur les clusters et les chaînes

EECS est un schéma de regroupement comme LEACH où le réseau est divisé en un ensemble de cluster chacun ayant un seul CH. La communication entre le CH de chaque cluster et la SB est directe (single-hop). Dans la phase de déploiement du réseau, la SB diffuse un message de type "HELLO" à tous les nœuds ayant un certain niveau de puissance. En se basant sur la puissance du signal reçu, chaque nœud peut calculer la distance approximative jusqu'à la station de base aidant ainsi les nœuds à choisir le niveau de puissance approprié pour communiquer avec la station de base. De plus, cette distance est utilisée pour équilibrer la charge entre les Custers-Head. Durant la phase des élections du CH, ceux qui sont bien distribués sont élus avec un certain contrôle de surcharge. Dans la phase de la formation des clusters, une nouvelle fonction de pondération est introduite pour former des clusters à charges équilibrées.

2.4.6 HEED

HEED (Hybrid, Eenergy-Efficient, Distributed clustering approach)[30] est un protocole de routage hiérarchique basé sur les clusters pour les réseaux de capteurs sans fil, en met-

tant l'accent sur le regroupement efficace par une sélection appropriée des clusters-Head en fonction de la distance physique entre les nœuds. Les principaux objectifs de ce protocole sont [30] :

- ❖ Distribuer la consommation d'énergie pour prolonger la durée de vie du réseau ;
- ❖ Minimiser l'énergie pendant la phase de sélection des cluster- heads ;
- ❖ Minimiser Overhead.

L'aspect le plus important du HEED est la méthode de sélection des cluster-heads qui sont déterminés en fonction de deux paramètres importants [30] :

- ❖ Le premier paramètre est l'énergie résiduelle du nœud. Ainsi, un nœud avec une grande énergie résiduelle a plus de chance de s'élire cluster-head. Ce paramètre est couramment utilisé dans de nombreux autres schémas de clustering.
- ❖ Le second paramètre est le coût de communication intra-cluster. Il est utilisé par les nœuds afin de déterminer le cluster à y joindre. Ceci est particulièrement utile si un nœud donné se situe dans la plage de plus d'un CH. Le protocole HEED utilise ce paramètre pour résoudre les conflits, c'est à dire, quand un nœud se trouve à la portée de plusieurs cluster-Head en même temps.

2.5 Conclusion

Le routage dans les RCSF est défini comme étant un processus de base et essentiel pour le bon fonctionnement de ces réseaux. Il consiste à déterminer un acheminement efficace d'informations vers la SB, en prenant compte la consommation d'énergie. Donc, il s'agit d'un sujet qui a fait l'objet de nombreuses recherches.

Par contre, pendant de nombreuses années, la problématique de la sécurité des protocoles de routage n'a pas eu une grande attention, puisque la plupart de ces protocoles ont été développés pour fournir des routes fiables durant l'échange des données, cependant, l'aspect de sécurité à été négligé. Le chapitre suivant sera consacré à l'étude de la sécurité dans les réseaux de capteurs sans fil.

CHAPITRE 3

LA SÉCURITÉ DANS LES RÉSEAUX DE CAPTEURS SANS FIL

3.1 Introduction

L'utilisation des réseaux de capteurs sans fil est de plus en plus répandue et les applications de cette technologie sont multiples (militaires, environnementales, médicales,..., etc). Cela est dû au faible coût de production et la facilité de déploiement de tels réseaux. Par contre, la conception de ces applications suppose que tous les nœuds du réseau sont coopératifs et dignes de confiance. Cependant, ceci n'est pas le cas dans les déploiements du monde réel, où les RCSF sont vulnérables à différents types d'attaques qui peuvent perturber le fonctionnement du réseau et menacer la fiabilité des données échangées. Donc, il est nécessaire de sécuriser ce type de réseaux, tout en prenant en considération les ressources limités d'un nœud capteur, et cela dans le but de fournir la protection des communications entre les nœuds dans un environnement potentiellement hostile.

Dans ce chapitre, nous présentons les contraintes de la sécurité des RCSF. Nous exposons également les objectifs de la sécurité, la classification des attaques et les attaques qui sont menées contre le routage dans ces réseaux. Ensuite, nous essayons de présenter les différentes primitives cryptographiques utilisées dans les réseaux de capteurs. Puis, nous décrivons les attaques auxquelles le protocole LEACH est vulnérable et les principales solutions proposées pour sécuriser ce protocole.

3.2 Les contraintes de la sécurité dans les RCSF

Les RCSF ont de nombreuses contraintes qui les rendent vulnérables d'une part aux attaques malicieuses dans les environnements ouverts et hostiles, et d'autre part il est difficile d'appliquer directement les approches de sécurité actuelles sur ces réseaux. Mais, afin d'élaborer des mécanismes de sécurité efficaces tout en empruntant des idées à partir des techniques de sécurité existantes, il est toutefois nécessaire de connaître et de comprendre ces contraintes [31].

3.2.1 La contrainte des ressources

Toutes les approches de sécurité exigent une certaine quantité de ressources pour leur mise en œuvre, y compris la mémoire de données, l'espace du code et l'énergie pour alimenter le nœud capteur. Toutefois, ces ressources sont très limitées dans ces nœuds capteurs sans fil. Le tableau 3.1 présente des limitations physiques pour quelques nœuds capteurs.

	MicaZ (2004)	**TelosB (2004)**
CPU	16 MHz	8 MHz
memoire flash	128 Ko	48 Ko
RAM	4 Ko	10 Ko

Tableau 3.1 – Limitations physiques des nœuds capteurs [6, 7]

Limitation de la mémoire et de l'espace de stockage

Le nœud capteur est doté d'une mémoire très limitée (voir le tableau 3.1). Ceci signifie qu'un mécanisme complexe de sécurité pourrait avoir un nombre d'instructions trop grand et donc réserver trop de mémoire, et ne laisser que très peu de mémoire ou presque pas pour d'autres opérations pour le nœud capteur. Ainsi, la taille du code de sécurité doit être la plus petite possible et le nombre de clés stockées doit être également petit.

Contrainte de la puissance d'énergie

L'énergie est sans aucun doute la ressource qui doit être gérer avec une grande attention. Nous supposons une fois que les nœuds sont déployés dans un environnement, ils ne peuvent plus être facilement remplacés ou rechargés. Par conséquent, l'énergie initiale de chaque

nœud capteur doit être conservée pour prolonger sa durée de vie et par la suite, prolonger la durée de vie du réseau. D'un autre côté, l'impact énergétique du code de sécurité ajouté dans les nœuds capteurs doit être pris en considération.

3.2.2 Manque de fiabilité de communication

La sécurité du réseau dépend du mécanisme de sécurité définit qui à son tour dépend de la communication. Dans la phase de conception des mécanismes de sécurité, la perte de paquets et la latence doivent être pris en considération. Si le taux d'erreurs du canal est élevé, alors un traitement d'erreur doit être effectué afin que les paquets critiques de sécurité tels que les clés cryptographiques ne soient pas endommagés. Un canal sans fil est un moyen de communication ouvert accessible par toute personne qui se trouve dans la portée du signal. Cependant, ce moyen est à son tour un obstacle pour la sécurité, rendant facile la production des attaques sur le réseau de capteurs.

3.2.3 Fonctionnement sans surveillance

Les nœuds capteurs peuvent être placés dans des environnements sans aucune surveillance pendant une longue période de temps. Ceci peut produire des faiblesses de sécurité pour le réseau, notamment si les nœuds sont déployés dans des environnements hostiles. Dans cette section, nous essayons de présenter les principales réserves pour les nœuds capteurs sans surveillance [32]

Exposition aux attaques physiques

Vu la nature ouverte de l'environnement de déploiement du réseau, les nœuds sont exposés aux attaques physiques. Et donc l'attaquant peut avoir le contrôle total sur des nœuds du réseau. Par contre, l'attaquant peut supprimer le nœud capteur, cela se fait par la destruction du nœud. Par exemple, les nœuds de capteurs dans l'océan peuvent être mangés par les poissons ou emportés pendant les orages.

Gestion à distance

Étant donné l'environnement ouvert de déploiement des nœuds et le manque de la surveillance humaine, il est important de gérer à distance les nœuds capteurs après leur déploiement. Par exemple, dans un scénario militaire, dans lequel les nœuds de capteurs sont placés

derrière les lignes des ennemies pour des missions de reconnaissance, aucun accès direct ne sera possible après le déploiement.

Aucun point central de gestion

Les réseaux de capteurs peuvent s'organiser automatiquement pour former un réseau distribué et sans point de gestion central. Cela fournit un réseau de communication efficace et dynamique, pour envoyer des informations aux serveurs dans le monde extérieur. La communication entre les nœuds de capteurs nécessite d'intégrer des fonctionnalités de sécurité contre les attaques possibles.

3.3 Les objectifs de sécurité

Les principaux objectifs ou services de la sécurité sont :

3.3.1 L'authentification de l'origine

Elle garantit que le récepteur du message doit être capable de vérifier la validité d'identité de l'émetteur. Autrement dit, le récepteur s'assure que les données utilisées proviennent de la source. Cela garantit l'exactitude des données d'origine [33].

3.3.2 La confidentialité

La confidentialité reste un point important dans la communication sans fil des RCSF. Donc elle consiste à assurer que les messages d'un nœud ne sont rendus accessible ou révélés qu'au destinataire approprié. En d'autres termes, la confidentialité préserve le secret des messages échangés.

3.3.3 L'intégrité

Le rôle de l'intégrité est de garantir que les messages ne subissent aucune altération pendant leur acheminement dans le réseau de manière volontaire ou accidentelle. Pour cela, le récepteur peux s'assurer que le message reçu est le même que le message envoyé par l'émetteur.

3.3.4 La fraîcheur

Même si l'authentification, l'intégrité et la confidentialité de données sont assurées, la fraîcheur de chaque message doit être également assurée. En effet, la fraicheur des messages implique que les messages sont récents et actuels. Cela signifie qu'ils ne sont pas une réinjection des précédents échanges interceptés par un attaquant.

3.3.5 La disponibilité

La disponibilité permet de garantir que les services d'un réseau de capteurs devraient être toujours disponibles même en présence des attaques internes ou externes.

3.4 Classification des attaques

Les attaques contre les réseaux de capteurs peuvent être classées selon les catégories suivantes :[34, 35]

3.4.1 Attaque externe $\mathcal{VS}$ Attaque interne

Selon l'appartenance d'un nœud, les attaques sont classées en deux catégories : les attaques externes et les attaques internes. Les attaques externes se produisent par des nœuds qui ne sont pas déployés à l'intérieur du réseau et que ne sont pas autorisés à participer dans le réseau, alors que les attaques internes se produisent par des nœuds internes malveillants. cette dernière catégorie est le type de menace le plus sévère qui peut perturber le bon fonctionnement du réseau, et difficile à détecter.

3.4.2 Attaque passive $\mathcal{VS}$ Attaque active

L'objectif de l'attaque passive est d'obtenir des informations sans aucune modification sur l'échange. Habituellement, l'attaquant se limite à l'écoute du trafic échangé. Il recueille un grand volume de données et procède à une analyse de données pour extraire les informations secrètes, ou bien la connaissance des nœuds importants dans le réseau (Cluster-Head). Ces informations extraites peuvent ensuite servir l'attaquant à des fins malveillantes. Dans une attaque active, l'attaquant tente d'exploiter les failles de sécurité du réseau pour lancer des

attaques diverses dans le but de modifier les données ou perturber le bon fonctionnement du réseau.

3.4.3 Attaques physiques $\mathcal{VS}$ Attaque à distance

Dans une attaque physique un adversaire accède physiquement au nœud de capteur qui devrait être lésé par la falsification ou la destruction du matériel de capteur. En revanche, une attaque à distance est mise en œuvre à partir d'une distance, par exemple, en émettant un signal à haute énergie pour interrompre la communication.

3.4.4 Attaque Laptop-class $\mathcal{VS}$ Attaque Mote-class

L'attaque mote-class se produit par un noeud de capteur (mote en anglais). Autrement dit, le dispositif d'attaque est du même type de matériel que les nœuds de capteurs qui devraient être attaqués. En revanche, dans l'attaque laptop-class, l'adversaire utilise un dispositif qui est supérieur aux nœuds de capteurs qui devraient être attaqués en termes de puissance de calcul et puissance de transmission.

3.5 Attaques contre le routage dans les RCSF

Les réseaux de capteurs sont particulièrement vulnérables à des attaques variées. De nombreux protocoles de routage dans ce type de réseau étaient très simples et n'ont pas été conçus avec la sécurité comme un objectif. Ainsi l'adversaire peut lancer diverses attaques dans le réseau. Principalement, les protocoles de la couche réseau souffrent de nombreuses attaques [3], nous présentons dans la suite les plus connues :

3.5.1 Spoofed, altered and replayed routing information

C'est une attaque contre les protocoles de routage. Elle permet à l'attaquant de viser les informations échangées entre les nœuds de capteurs [3]. Un adversaire pourra injecter des précédents échanges interceptés par celui-ci (attaque par rejeu), ou de fausses données dans le réseau pour confondre les nouds capteurs. Un nœud malicieux peut également modifier les données reçus avant de les transmettre vers la destination finale.

3.5.2 Selective forwarding

Dans l'attaque selective forwarding, l'attaquant peut insérer ou compromettre des nœuds capteurs dans le réseau afin que ces nœuds malicieux refusent de transmettre certains paquets provenant des nœuds voisins. Le choix des paquets est fondé sur certains critères (contenu des paquets, adresse source de l'émetteur) ou d'une façon aléatoire [36]. Dans la Figure 3.1, le nœud malicieux 5 transmet tous les paquets sauf ceux qu'il reçoit du nœud 4, fondée sur l'adresse d'origine.

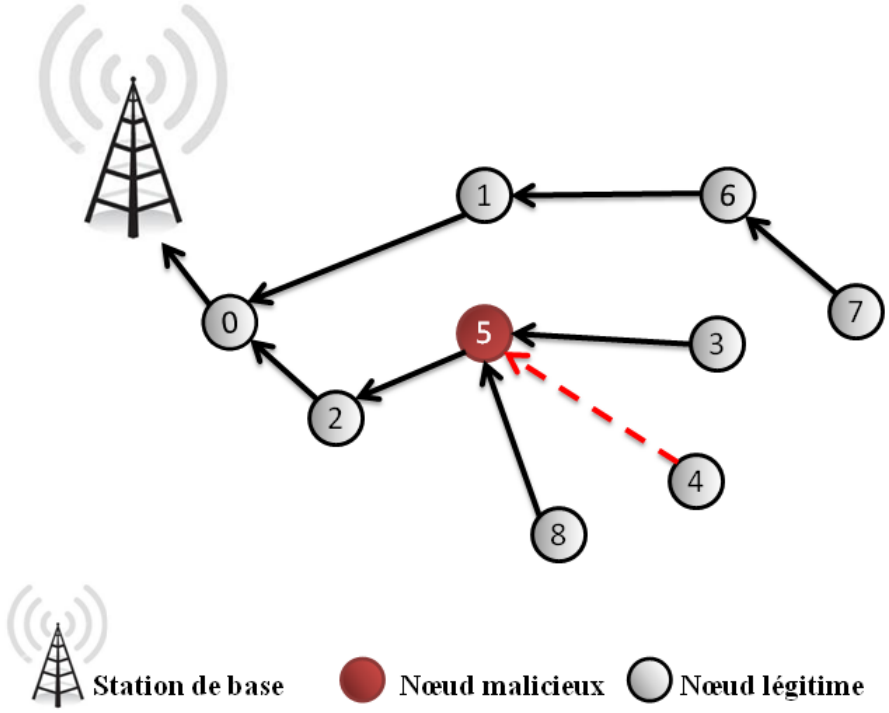


FIGURE 3.1 – Exemple d'une attaque selective forwarding

3.5.3 Sinkhole

Dans cette attaque, le nœud malicieux tente d'attirer vers lui tout le trafic permettant de contrôler la plus part des données circulant dans le réseau et dans le but d'empêcher la SB d'obtenir des données complètes et correctes, parce que la SB est le point important qui recueille le maximum de données sur l'intégralité du réseau[37].Ceci est possible lorsque le nœud malicieux propose aux autres nœuds des fausses informations de routage, souvent des routes optimales pour atteindre la SB. Ainsi les nœuds voisins vont choisir le noeud malicieux

comme prochain saut. Toutes les données qui transitent de ces nœuds vers la SB peuvent être récupérées par l'attaquant (voir la Figure 3.2).

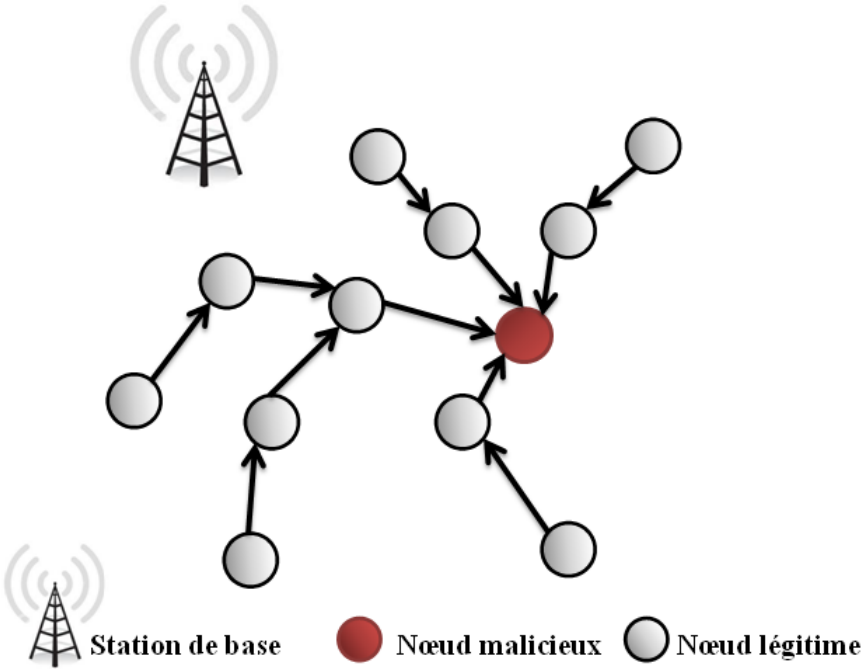


FIGURE 3.2 – Exemple d'une attaque sinkhole

3.5.4 Sybil

Il s'agit d'une attaque[38, 39] où un nœud malicieux peut se faire passer pour plusieurs nœuds, en recueillant plusieurs identités dans le réseau. Soit par la fabrication ou le vol de l'identité des nœuds légitimes. Cette attaque peut dégrader l'efficacité de plusieurs fonctionnalités comme la distribution de données, en visant à changer l'intégrité des données et les mécanismes de routage. En outre, un nœud malicieux dans cette attaque peut gagner un avantage important pour une élection de nœud Cluster-Head par exemple.

dans la Figure 3.3 le nœud malicieux recueille les identités des nœuds A, B, C et D

3.5.5 Wormhole

Cette attaque [3, 40] consiste à créer un lien (un tunnel) de faible latence entre deux nœuds malicieux dans le réseau, ce lien est utilisé par l'adversaire pour l'injection, la modification et la retransmission des données.

La Figure 3.4 illustre un exemple de l'attaque wormhole.

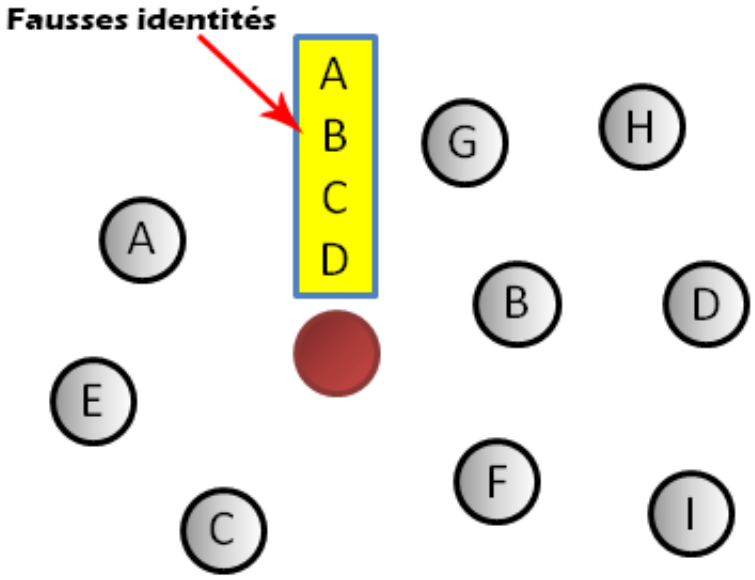


FIGURE 3.3 – Exemple d'une attaque sybil

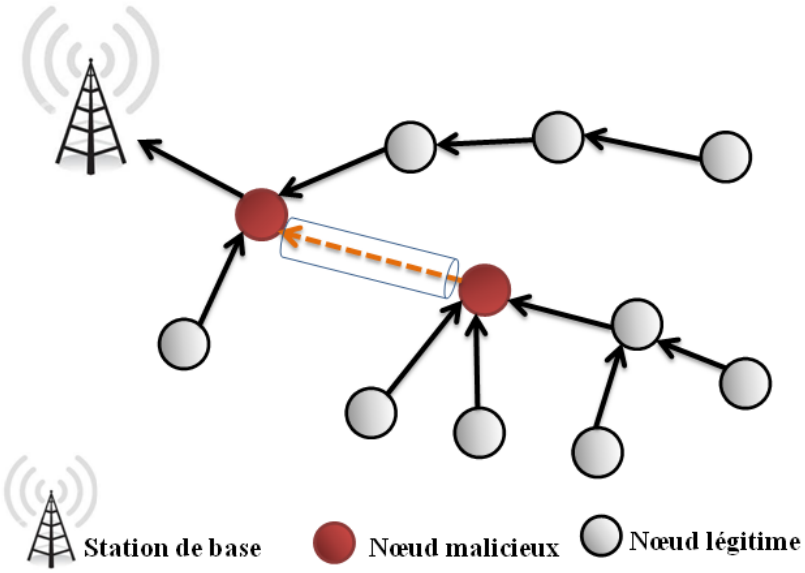


FIGURE 3.4 – Exemple d'une attaque wormehole

3.5.6 Hello flood

De nombreux protocoles de routage utilisent des paquets Hello pour la découverte du voisinage [3]. Dans une attaque dite de Hello flood, un nœud malicieux essaye de diffuser de tels paquets en utilisant un puissant signal, et cela dans le but de consommer l'énergie des nœuds et empêcher leurs messages d'être échangés, comme l'illustre la Figure 3.5

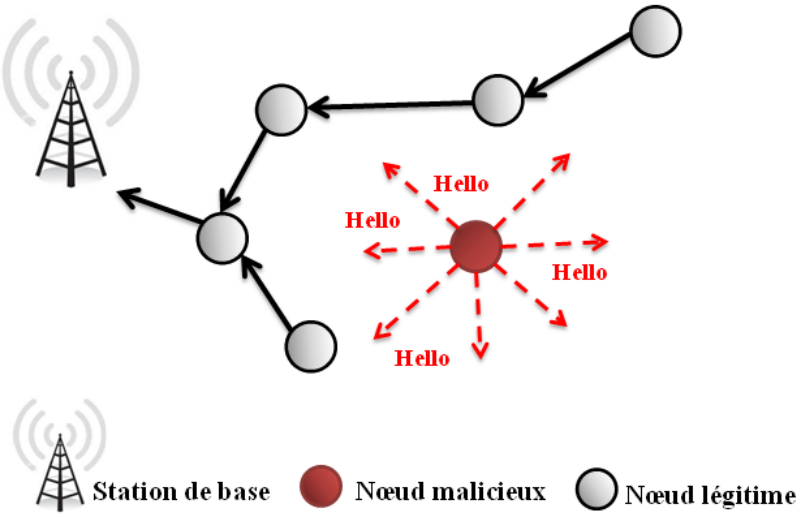


FIGURE 3.5 – Exemple d'une attaque Hello flood

3.6 Primitives cryptographiques utilisées dans les RCSF

Nous présentons par la suite, les différentes primitives cryptographiques qui sont utilisées dans les réseaux de capteurs sans fil.

3.6.1 La cryptographie

L'origine du mot cryptographie provient du grec "Kryptos-Graphein" qui signifie "cacher-écrire". C'est donc l'art de l'écriture en langage codé [41]. La cryptographie est définie comme étant une science permettant de protéger une communication et d'assurer que l' information contenue dans un message n'est révélée qu'au destinataire de ce message. Elle permet de transformer un message dit "texte clair" en un message dit "texte crypté". Par conséquent, le dernier message devient incompréhensible, c'est ce qu'on appelle le cryptage. L'opération inverse est appelée le décryptage, elle permet de restituer le texte clair à partir du texte crypté.

Le cryptage et le décryptage des messages sont effectués par des algorithmes cryptographiques. Ces algorithmes reposent généralement sur des problèmes mathématiques complexes et difficiles à résoudre, tels que la factorisation des nombres premiers, les logarithmes discrets,...,etc. Les algorithmes cryptographiques modernes nécessitent une clé pour le cryptage et une clé pour le décryptage.

On distingue deux familles de cryptographie : La cryptographie symétrique et la cryptographie asymétrique.

La cryptographie symétrique

Dans la cryptographie symétrique (ou la cryptographie à clé secrète), la clé de cryptage et la clé de décryptage sont identiques, elle est dite clé symétrique ou clé secrète utilisée par l'émetteur pour crypter le message et par le récepteur pour le décrypter. La Figure 3.6 illustre le principe de la cryptographie symétrique.

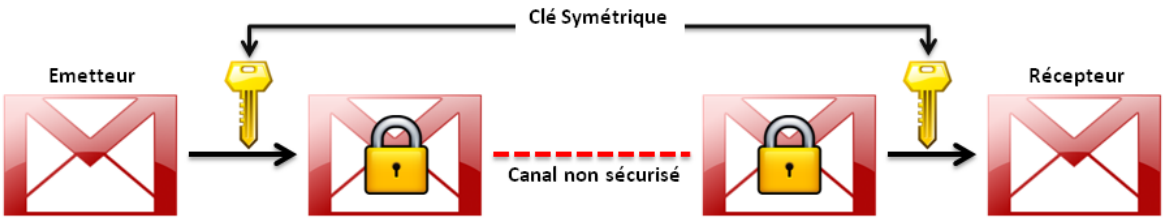


FIGURE 3.6 – La cryptographie symétrique

La cryptographie asymétrique

Dans la cryptographique asymétrique (ou la cryptographie à clé publique), la clé de cryptage et la clé de décryptage sont différentes. Une des clés appelée clé publique (qui est diffusée) utilisée généralement pour crypter le message. Tandis que l'autre clé appelée clé privée (gardée secrète), permet de décrypter le message crypté . La Figure 3.7 montre le principe de la cryptographie asymétrique.

Symétrique $\mathcal{VS}$ Asymétrique

La cryptographie symétrique réduit considérablement la consommation d'énergie des nœuds capteurs, le temps de calcul et l'espace de stockage réservé pour accueillir les clés. Cependant, l'échange de clés dans tel système est plus difficile et compliqué, donc le problème

FIGURE 3.7 – La cryptographie asymétrique

majeur avec la cryptographie symétrique est de pouvoir trouver une méthode qui achève l'établissement de clé entre les nouds. Tandis que, la cryptographie asymétrique assure de faciliter la gestion de clés. De même que, elle exige un espace mémoire assez grand et de haute capacité de calcul, ce qui la rend inappropriée pour les réseaux de capteurs. Cependant, les références [42, 43, 44, 45, 46] montrent qu'il est possible d'appliquer la solution asymétrique (ECC, *Elliptic Curve Cryptography*) aux réseaux de capteurs. Puisque la cryptographie utilisant les courbes elliptiques nécessite des tailles de clés inférieures à RSA et par conséquent, elle réduit la consommation d'énergie, le temps de calcul.

La cryptographie basée sur les courbes elliptiques ECC

ECC (Elliptic Curve Cryptography) est considérée comme une approche de cryptographie asymétrique opère sur des points appartenant à une courbe elliptique. Cette dernière ont été proposée indépendamment par Victor Miller [47] et Neal Koblitz [48] en 1985 pour être utilisées dans la cryptographie. ECC peut être utilisée pour des opérations asymétriques comme des échanges de clés sur un canal non sécurisé ou un cryptage asymétrique. Elle a attirée beaucoup d'attention comme un moyen de sécurité pour les réseaux de capteurs en raison de la petite taille de la clé ECC en comparaison avec la clé RSA, par exemple une clé ECC de 160 bits offre une sécurité équivalente à une clé RSA de 1024 bits [49]. Ce qui permet de réduire le temps de calcul, d'économiser l'énergie et la mémoire. Le tableau 3.2 inspiré de la référence [49] présente une comparaison de la taille des clés entre RSA et ECC en assurant le même niveau de sécurité.

Les concepts de base de la cryptographie de courbe elliptique ECC **:** Soient F_q un corps fini à q éléments et $\overline{F}_q$ la clôture algébrique de F_q, c'est-à-dire que tout polynôme de degré supérieur ou égal à 1, à coefficients dans F_q, admet au moins une racine dans $\overline{F}_q$. Une

Taille de clé RSA (bit)	Taille de clé ECC (bit)
1024	160
2048	224
3072	256
7680	384
15360	512

Tableau 3.2 – Comparaison de la taille de clé pour ECC et RSA [49]

courbe elliptique E est l'ensemble des couples ou points $(x, y) \in \overline{F}_q X \overline{F}_q$ vérifiant l'Équation (3.1)

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \quad \text{avec} \quad a_i \in F_q$$

Équation (3.1) : Équation générale de courbes elliptiques [48].

L'Équation (3.1) est connue sous le nom de l'équation de *Weierstrass* sur le corps F_q. Pour leur usage en cryptographie, a_1, a_2 et a_3 doivent avoir une valeur nulle. Avec les cryptographes a_4 est devenu a et a_6 est devenu b, d'où l'Équation (3.2) assortie d'une condition portant sur les coefficients.

$$y^2 = x^3 + ax + b \quad \text{avec} \quad 4a^3 + 27b^2 \neq 0$$

Équation (3.2) : Equation d'une courbe elliptique utilisée dans la cryptographie [50].

La courbe elliptique utilisée dans la cryptographie est constituée d'un ensemble de points (x, y) avec un point particulier nommé le point à l'infini $\mathcal{O}$, $E = \{(x, y) \in \overline{F}_q X \overline{F}_q\} \cup \{\mathcal{O}\}$. Le point $\mathcal{O}$ va servir comme l'élément d'identité ou d'élément neutre de (E). Notons que souvent le corps F_q est choisi d'une façon telle que $q = p^m$, p étant un nombre premier ($p = 2$, typiquement) et m étant la taille de la clé de chiffrement ($m = 160$, typiquement).

La Figure 3.8 montre l'addition de deux points sur une courbe elliptique d'équation $y^2 = x^3 - 3x + 5$. Pour additionner les deux points P et Q , nous traçons la droite qui passe par ces deux points. La droite coupe la courbe en un point $-R$. Le symétrique du point $-R$ en ce qui concerne l'axe des abscisses donne le point R, qui est le résultat de l'addition de points P et Q. Dans le cas où $P = Q$, la droite qui passe par P et Q est bien la tangente à P (Voir la Figure 3.9). Une autre opération importante est la multiplication scalaire. Par exemple, pour calculer $2.P$ nous faisons l'addition $P + P$. Soit k un entier positif, le calcul

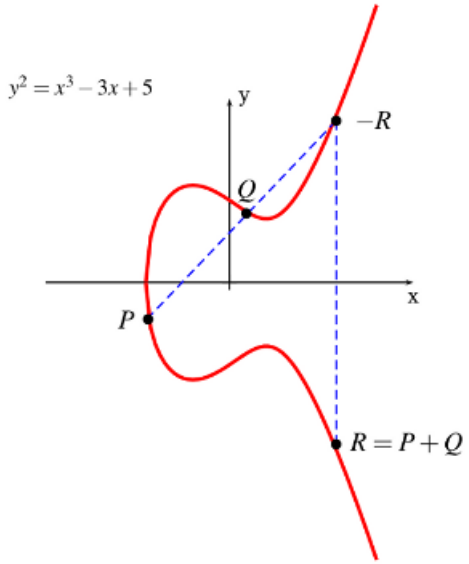


FIGURE 3.8 – Addition de deux points sur une courbe elliptique.

de $k.P$ est égale à $\underbrace{P + ... + P}_{k}$. La constante k est considérée comme une clé privée et le point obtenu sera considéré comme la clé publique après avoir vérifié qu'il appartient à la courbe. Pour plus de détails sur les opérations d'addition et de multiplication d'ECC avec des exemples, veuillez-vous référer à Koblitz et al [50].

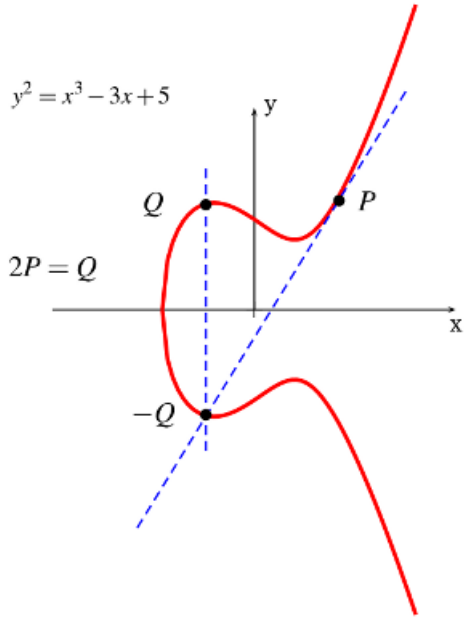


FIGURE 3.9 – Doublement de point

La sécurité ECC est liée à la difficulté pour résoudre le problème de logarithme discret ECDLP (*Elliptic Curve Discrete Logarithm Problem*), c'est-à-dire, étant donné les points P et Q, il est difficile de trouver un nombre K tel que $Q = k.P$.

ECDH

C'est une méthode d'échange de clé basé sur les courbes elliptiques permettant à deux entités d'établir une clé secrète commune qui peut être utilisée pour la cryptographie à clé symétrique. Supposons que E est la courbe elliptique sur un corps fini F_q et Q est un point sur la courbe. Alice choisit secrètement un entier k_A (La clé privée d'Alice) et calcule le point $k_A.Q$ (La clé publique d'Alice) qui sera envoyé à Bob. A son tour, Bob choisit secrètement k_B (La clé privée de Bob) et calcule le point $k_B.Q$ (La clé publique de Bob) qui sera envoyé à Alice. Alice multiplie le point reçu par k_A et Bob le multiplie par k_B. Le secret partagé calculé des deux côtés est le point $k_A.k_B.Q$. Un adversaire connaissant Q, $k_A.Q$ et $k_B.Q$ mais pas k_A ni k_B doit résoudre le problème du logarithme discret pour calculer le secret partagé.

3.6.2 La fonction de hachage

Elle permet de générer une chaîne de taille inférieure et généralement fixe à partir d'une chaîne de longueur quelconque. Par conséquent, la chaîne résultante est appelée empreinte (digest en anglais). d'un autre coté, une fonction de hachage est une fonction à sens unique, autrement dit qu'il est facile à calculer l'empreinte d'une chaîne donnée, mais il est impossible de déduire à la chaîne initiale à partir d'une empreinte donnée.

Cette fonction est utilisée pour la vérification de l'intégrité des messages transmis (voir la Figure 3.10). L'émetteur utilise la fonction de hachage pour créer une empreinte du message à transmettre, puis il transmet le message et l'empreinte vers le récepteur. À la réception du message, le récepteur calcule l'empreinte du message reçu et il la compare à l'empreinte initiale. Si les deux empreintes correspondent, c'est que le message n'a pu être altéré.

Les algorithmes SHA-1 (*Secure Hash Algorithm 1*, 160 bits) [51] et MD5 (*Message-Digest algorithm 5*, 128 bits) [52](p. 5) sont les fonctions de hachage les plus utilisées. Cependant, MD5 n'est plus considéré comme sûr car il a été constaté qu'il pouvait y avoir « une collision » (deux messages différents ayant deux empreintes identiques) à partir de deux messages au contenu aléatoire [53, 54]. Ces deux messages ont généré la même empreinte en utilisant MD5 ce qui n'est pas acceptable puisque l'empreinte est supposée être unique.

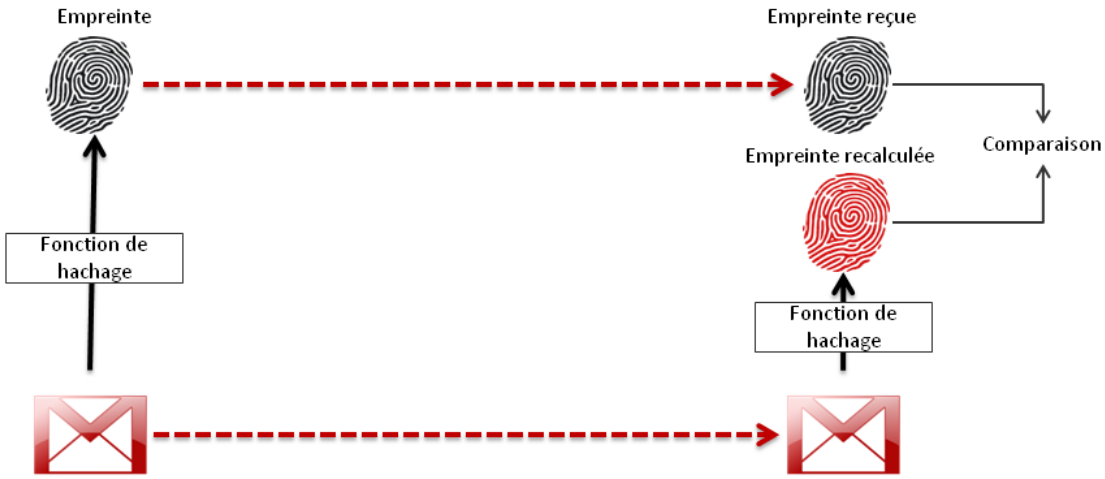


FIGURE 3.10 – La fonction de hachage

3.6.3 Codes d'authentification de message (MAC)

Un code d'authentification de message (MAC : Message Authentication Code) est le résultat d'une fonction de hachage à sens unique dépendant d'une clé symétrique. Ce code accompagne des données afin d'assurer l'intégrité de ces dernières, en permettant de vérifier qu'elles n'ont subi aucune modification après la transmission, en plus, l'authentification de la source de données.

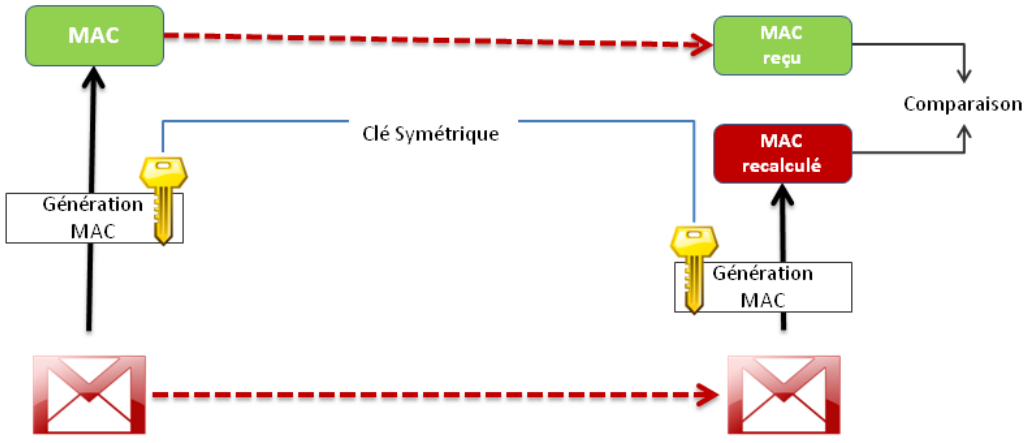


FIGURE 3.11 – Codes d'authentification de message - MAC

3.7 Les attaques contre le protocole LEACH

Dans un réseau de capteurs utilisant le protocole de routage LEACH, chaque nœud membre doit correctement envoyer les données captées à son CH, une fois que le CH obtient

toutes les données à partir des nœuds appartenant à son cluster, il doit agréger ces données puis de les transmet à la SB. Toute fois une faille dans ce processus aurait un effet sur le bon fonctionnement de ce protocole et par la suite sur le fonctionnement du réseau. Or, le protocole LEACH ne fournit aucune spécification de sécurité à prendre en compte ce qui rend ce protocole vulnérable à plusieurs types d'attaques [3, 55], comme l'attaque d'écoute, l'attaque par rejeu, l'altération des données, l'attaque Hello flood, l'attaque selective forwarding et l'attaque sybil. Par ailleurs, les attaques impliquant des CHs, sont les plus dommageables. Si un nœud malicieux réussi à devenir un CH, il peut lancer des attaques comme celle selective forwarding pour perturber le fonctionnement du réseau. Notons qu'un nœud malicieux peut choisir de ne pas attaquer le CH, et essayer également d'altérer ou d'injecter des fausses données dans le réseau.

3.8 Les principales solutions proposées pour sécuriser LEACH

Dans cette section, nous présenterons un ensemble des principales solutions proposées dans la littérature. Pour cela, nous utiliserons les notations suivantes :

- id_x : Identité de nœud x
- SB : Station de base
- CH :Cluster-Head qui joue le rôle de tête du cluster
- SN : Nœud de capteur simple
- K_x : Clé symétrique partagée entre l'entité x et la station de base
- K_j : Clé globale
- N_x : Nonce partagé entre l'entité x et la station de base
- $MAC(K, M)$: Code d'authentification du message M avec la clé symétrique K
- $E(K, M)$: Le message M est crypté en utilisant la clé K
- $Sign_K(M)$: Signature numérique d'un message M en utilisant la clé privée K

3.8.1 SLEACH

Selon les auteurs de [56], ce protocole est la première version sécurisée du LEACH. Il utilise la cryptographie symétrique pour assurer une protection contre les attaques. Le protocole SLEACH se déroule en trois phases : la phase avant le déploiement, la phase d'installation

et la phase de transmission. l'algorithme du protocole(les deux phases : l'installation et la transmission) est décrit dans l'Algorithme 2.

La phase avant le déploiement

Dans cette phase, La SB calcule une séquence $S = k_0, k_1, k_2, \ldots, k_{n-1}, k_n$, pour ce faire, elle génère K_0 et calcule la suite k_i (voir la Figure 3.12) à l'aide une fonction de hachage à sens unique F tel que :

$$F(k_i) = k_{i+1}, 0 \leq i < n$$

La SB stocke toutes les clés de S à son niveau, mais elle partage la dernière clé K_n avec le

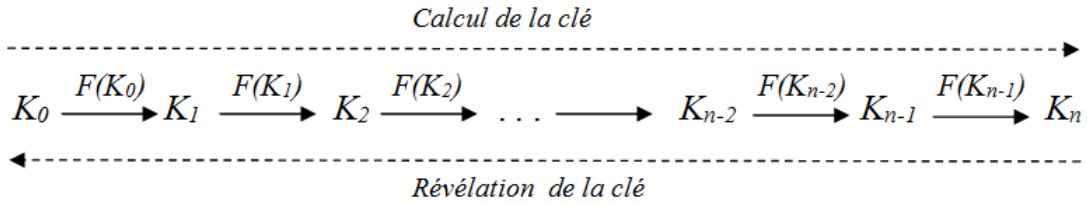


FIGURE 3.12 – la séquence de clés S

reste du réseau. Chaque nœud SN est préchargé avec deux clés :(i) K_{SN}, une clé symétrique (pairwaise key), partagée entre le noeud SN et la SB ;(ii) K_n, une clé globale qui est partagée entre chacun des noeuds du réseau et la SB. Pour des raisons concernant la fraîcheur des messages, chaque nœud SN partage également un nonce N_{SN} avec la SB.

Phase d'installation;

1.1. $CH \rightarrow broadcast : id_{CH}, MAC(K_{CH}, id_{CH}||N_{id_{CH}}||adv)$

$SN : stocker(id_{CH})$

SB : **si** $MAC(K_{CH}, id_{CH}||N_{id_{CH}}||adv)$ *est valide* **alors**
| $ajouter(id_{CH}, \mathcal{V})$

fin

1.2. $SB \rightarrow broadcast : \mathcal{V}, MAC(K_j, \mathcal{V})$

1.3. $SB \rightarrow broadcast : K_j$

SN : **si** $(F(K_j) = K_{j+1})$ *et* $(id_{CH} \in \mathcal{V})$ **alors**
| id_{CH} est authentifié

fin

2. $SN \rightarrow CH : id_{SN}||id_{CH}||join\_req$

3. $CH \rightarrow SN : id_{CH}||id_{SN}||Timeslot(SN)$

Phase de transmission;

4. $SN \rightarrow CH : id_{SN}||id_{CH}||info, MAC(K_{SN}, id_{SN}||N_{id_{SN}})$

5.1. $CH \rightarrow SB : id_{CH}||id_{SB}||aggr\_info, MAC(K_{CH}, id_{CH}||N_{id_{CH}}||aggr\_info)$

5.2. $CH \rightarrow SB : id_{CH}||id_{SB}, (..., id_{SNi}, MAC(K_{SNi}, id_{SNi}||N_{id_{SNi}}), ...),$

$MAC(K_{CH}, id_{CH}||N_{id_{CH}})$

6. $SB \rightarrow CH$: ids des nœuds malicieux.

Algorithme 2: Le protocole SLEACH

La phase d'installation

Une fois un simple nœud devient CH, il diffuse alors un message de notification (étape 1.1 - Algorithme 2) qui contient son identifiant et une valeur MAC produite, en utilisant la clé K_{CH}. Les nœuds non-CH rassemblent tous les messages de notification diffusés, en même temps, la SB reçoit chacun de ces messages et vérifie leur authenticité. Pour tout CH valide, la SB ajoute son id_{CH} à une liste des CHs valides appelée $\mathcal{V}$ (étape 1.1 - Algorithme 2), ensuite elle identifie la dernière clé K_j dans S qui n'a pas été révélée (notant que toute clé K_i , tel que $i > j$, a été révélée, tandis que toute clé K_i , tel que $i \leq j$, n'a pas été révélée). Après cela, la SB diffuse la liste $\mathcal{V}$ vers tous les nœuds du réseau (étape 1.2 - Algorithme 2) en utilisant $\mu TESLA$[57](protocole de broadcast authentifié pour les réseaux de capteurs) et puis, elle diffuse la clé K_j, cette dernière est révélée après un certain temps (étape 1.3 - Algorithme 2), autrement dit, après que tous les nœuds dans le réseau ont reçu le message

de broadcast précédent.

Après avoir reçu à la fois le broadcast et la clé correspondante, les nœuds non-CH vérifient l'intégrité de cette clé, pour cela, ils doivent sauvegarder la clé la plus récente ayant été authentifiée K_{j+1}. Lorsque le nœud reçoit la révélation de la clé K_j, il vérifie par la suite la relation $F(K_j) = K_{j+1}$, puis , ils vérifient l'authentification du broadcast de la SB pour savoir la liste des CHs légitimes pendant l'itération (round) courante. Les nœuds non-CH choisissent un CH de cette liste (le CH ayant le signal le plus fort sera choisi) puis, ils envoient l'information qui contient leur décision d'appartenance au CH choisi (étape 2 - Algorithme 2). Chaque CH crée un planning TDMA et envoie à chaque nœud membre un slot du temps *Timeslot* durant lequel il peut transmettre ses données (étape 3 - Algorithme 2).

La phase de transmission

Pendant cette phase, les nœuds membres(non-CH) envoient à leur CH l'information contenant la valeur captée (étape 4 - Algorithme 2). Pour authentifier l'origine de ces informations, chaque nœud membre ajoute une valeur MAC produite, en utilisant la clé K_{SN}. Chaque CH collecte alors les informations provenant de tous les nœuds membres de son cluster et les transmet, après avoir les agréger vers la SB (étape 5.1 - Algorithme 2). Par la suite, il retransmet tous les MACs reçus de ses nœuds membres vers la SB (étape 5.2 - Algorithme 2), du moment que le CH est incapable de les vérifier.

La SB vérifie donc à la fois le MAC calculé par le CH ainsi que ceux de ses membres. Dans le cas où il y a des MACs non valides, la SB refuse le résultat agrégé par le CH et de ce fait les nœuds propriétaires des MACs non valides seront vus comme des nœuds malicieux, ce qui permet à la station de base de révéler par la suite les identités de ces nœuds malicieux à leur CH (étape 6 - Algorithme 2), afin que ce dernier supprime les messages reçus à partir de ces nœuds.

Analyse et Discussion

Le protocole SLEACH est une version sécurisée du LEACH utilisant la cryptographie symétrique, il empêche les attaques du type : Altération/insertion des faux messages, selective forwarding, HELLO flood et l'attaque par rejeu. Il empêche aussi qu'un nœud malicieux de devenir CH ou d'envoyer des données falsifiées au CH. Le protocole SLEACH assure l'authentification partielle (SB-CH et SB-SN), l'intégrité et la fraîcheur des messages. De

plus, il n'est pas couteux en termes de mémoire de stockage (2 clés stockées seulement dans chaque nœud). Cependant, il n'assure ni la confidentialité ni l'authentification (CH-SN). L'intrus peut diffuser un faux planning Timeslot qui peut entrainer par la suite une perturbation dans la communication entre les nœuds membres du cluster et leur CH, et cela durant la phase de transmission. De plus, la référence à la SB dans ce protocole pour vérifier la validité des nœuds (CH et SN) ,permet de générer un nombre important de messages qui limitent le passage à l'échelle du réseau.

3.8.2 SecLEACH

Dans la référence [58] les auteurs ont proposé une version améliorée de SLEACH appelée SecLEACH. Ce protocole utilise également la cryptographie symétrique et un schéma aléatoire de pré-distribution de clés pour sécuriser le protocole LEACH. Le déroulement du protocole SecLEACH se fait en trois phases : La phase avant le déploiement, la phase d'installation et la phase de transmission. Les deux phases (l'installation et la transmission) sont présentées dan l'Algorithme 3, ce dernier forme l'algorithme du protocole SecLEACH.

La phase avant le déploiement

Dans cette phase, un grand ensemble P *(Pool)* de clés est généré et chaque clé est associée a un identifiant $P = ((kid1,key1),(kid2,key2),\dots)$. Pour chaque nœud, m clés sont choisies aléatoirement à partir de l'ensemble P. Ces m clés sont stockées dans la mémoire du nœud et forment le porte-clés du nœud (key ring). Le nombre de clés $|P|$ est choisi de telle manière que deux sous-ensembles aléatoires de P de taille m auront une certaine probabilité p d'avoir au moins une clé en commun, par exemple pour une probabilité $p = 0.5$ seulement 75 clés sont stockées dans les nœuds avec $|P| = 10{,}000$ clés [59].

En outre, et comme nous l'avons déjà cité dans la section 3.8.1, chaque nœud est préchargé par une clé globale (la dernière clé d'une sequence de clés S générée par la SB), et une clé *Pairwise* partagée avec la SB.

Phase d'installation;

1.1. $CH \rightarrow broadcast : id_{CH}, MAC(K_{CH}, id_{CH}||nonce||adv)$

$SN : stocker(id_{CH})$

SB : **si** $MAC(K_{CH}, id_{CH}||nonce||adv)$ *est valide* **alors**
| $ajouter(id_{CH}, \mathcal{V})$

fin

1.2. $SB \rightarrow broadcast : V, MAC(K_j, \mathcal{V})$

1.3. $SB \rightarrow broadcast : K_j$

SN : **si** $(F(K_j) = K_{j+1})$ *et* $(id_{CH} \in \mathcal{V})$ **alors**
| id_{CH} est authentifié

fin

SN : choisir r tel que $r \in (\mathcal{R}_{CH} \cap \mathcal{R}_{SN})$

2. $SN \rightarrow CH : id_{SN}||id_{CH}||r||join\_req, MAC(K_r, id_{SN}||id_{CH}||r||nonce)$

3. $CH \rightarrow SN : id_{CH}||id_{SN}||Timeslot(SN)$

Phase de transmission;

4. $SN \rightarrow CH : E(K_r, id_{SN}||id_{CH}||info), MAC(K_r, id_{SN}||id_{CH}||info||nonce + j)$

5. $CH \rightarrow SB : E(K_r, id_{CH}||id_{SB}||aggr\_info), MAC(K_{CH}, id_{CH}||nonce||aggr\_info)$

Algorithme 3: Le protocole SecLEACH

La phase d'installation

Dans cette phase, les étapes 1.1, 1.2 et 1.3 (Algorithme 3) du protocole SecLEACH sont similaires à celle du protocole SLEACH où les étapes : 1.1, 1.2 et 1.3 sont détaillées dans la section 3.8.1. Mais dans l'étape 1.3, les auteurs ont ajoutés une sous étape où le CH diffuse en clair la liste des identifiants $\mathcal{R}_{CH}$ de son porte-clés à ses membres, pour découvrir des clés communes entre le CH et chaque nœud membre. Notons que si l'intrus découvre cette liste, il n'aura pas l'opportunité de découvrir les clés de la liste et par la suite il n'aura pas la clé partagée utilisée pour le cryptage.

Le nœud membre (non-CH), choisit un CH de la liste $\mathcal{V}$, le CH ayant le signal le plus fort sera choisi. Lorsque le nœud membre reçoit la liste $\mathcal{R}_{CH}$, il la compare par la suite avec sa liste $\mathcal{R}_{SN}$ afin de trouver un identifiant commun r et par conséquent, d'établir une clé commune K_r entre le CH et son nœud membre (Voir la Figure 3.13), puis, il envoie l'information qui contient sa décision d'appartenance au CH choisi. Cet information est protégée par un MAC généré en utilisant la clé K_r, le MAC comprend un nonce ainsi que l'identifiant r pour

la protection contre la attaque par rejeu. (étape 2 - Algorithme 3).

De ce fait, chaque CH crée un planning TDMA et envoie à chaque nœud membre un slot du temps *Timeslot* durant lequel il peut transmettre ses données (étape 3 - Algorithme 3). Cette transmission est authentifiée de la même manière que l'étape 1.1 de Algorithme 3. Pour rendre la présentation claire du protocole, les auteurs ne reproduisent pas la version authentifiée entière dans l'étape 3.

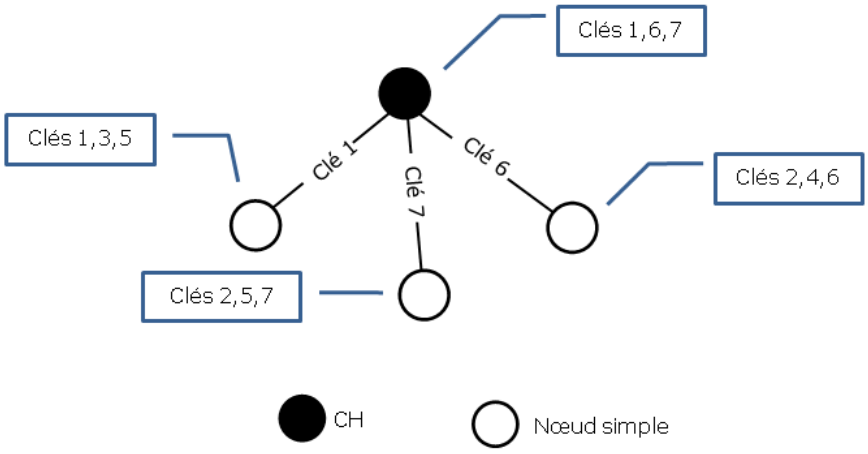


FIGURE 3.13 – Exemple de pre-distribution aléatoire de clés

La phase de transmission

Pour protéger la communication entre les nœuds membres et ses CH, le message d'un nœud membre doit être protégé par un MAC généré en utilisant la clé K_r. Afin d'assurer la protection contre la attaque par rejeu, le MAC de chaque nœud membre comprend un nonce+j, où j est le numéro d'itération courante (étape 4 - Algorithme 3). Chaque CH envoie l'information agrégée $aggr\_info$ vers la SB, il la protège ($aggr\_info$) par un MAC généré en utilisant la clé partagée avec la SB (étape 5 - Algorithme 3).

Analyse et Discussion

Le protocole SecLEACH est une version sécurisée de LEACH qui assure la confidentialité, l'intégrité, la fraîcheur et l'authentification du nœud originaire du message transmis. Ce protocole garantit l'authentification des deux messages *adv* et *join_req*, ce qui permet aux nœuds légitimes seulement de participer pendant la formation de clusters et d'empêcher les

nœuds non autorisés de devenir CH. SecLEACH possède la capacité de résister à plusieurs attaques comme : selective forwarding, sybil et hello flood. Cependant, il pré-charge les nœuds avec des portes-clés (m clés) qui est plus coûteux en termes d'espace mémoire. Un autre inconvénient, la connectivité dans ce protocole est variante et dépend de la taille du porte-clé pré-chargé dans les nœuds capteurs. Autrement dit, si un nœud ne partage pas une clé avec Cluster-Head alors il ne peut pas participer avec lui, par conséquent, ce nœud sera isolé (nœud orphelin).

3.8.3 AC

Les auteurs de la référence [60] ont proposé une version sécurisée de LEACH appelé AC (*Authentication Confidentiality cluster based secure routing protocol for wireless sensor network*). Ce protocole utilise la cryptographie asymétrique pour renfoncer la sécurité de LEACH. Pour le bon fonctionnement du protocole proposé, les auteurs ont spécifié quelques hypothèses.

- Chaque nœud est pré-chargé avec une paire de clé privé-publique (K_{SN}^{-}, K_{SN}^{+}) et la clé publique de la station de base K_{SB}^{+}.
- La phase de sélection des CH et de formation des clusters est similaire à celle du protocole LEACH.

Le schéma de base

$CH \longrightarrow broadcast : ID_{CH}$

Chaque CH diffuse son identifiant à l'ensemble des nœuds qui appartient à son cluster.

$SN \longrightarrow CH : ID_{SN}||K_{SN}^{+}||Sign_{K_{SN}^{-}}(K_{SN}^{+})$

Lorsque le noeud membre SN reçoit le message, il envoie par la suite son identifiant et sa clé publique vers son CH.

$CH : VERIFY(Sign_{K_{SN}^{-}}(K_{SN}^{+}))$

Le CH vérifie la validité de la signature de SN en utilisant la clé publique de SN.

$SN \longrightarrow CH : info$

Le nœud membre SN envoie les données captées (info) vers son CH.

$CH \longrightarrow SB : ID_{CH}||K_{CH}^{+}||Sign_{K_{CH}^{-}}(K_{CH}^{+})$

Le CH envoie son identifiant et sa clé publique vers la station de base(SB).

$SB : VERIFY(Sign_{K_{CH}^{-}}(K_{CH}^{+}))$

$SB \longrightarrow CH : H(ID_{CH})$

La SB vérifie la validité de la signature de CH en utilisant la clé publique de CH. Si la signature est valide, elle calcule une valeur hachée en utilisant une fonction de hachage à sens unique H. elle envoie par la suite, cette information au CH.

$CH \longrightarrow SB : E_{K_{SB}^{+}}(aggr\_info)||Sign_{K_{CH}^{-}}(H(ID_{CH}))$

Le Cluster Head (CH) envoie les données agrégées et la signature vers la SB. La signature est sous forme de valeur hachée, cryptée par la clé privée de CH.

$SB : VERIFY(Sign_{K_{CH}^{-}}(H(ID_{CH})))$

Lorsque la SB reçoit les données et la signature, elle vérifie la validité de la dernière en utilisant la clé publique de CH. Si la signature est valide, la SB accepte les données agrégées sinon elle les refuse.

Analyse et Discussion

Le protocole AC est considéré comme une extension sécurisée du LEACH, qui est basé sur la cryptographie asymétrique. Il permet de garantir les trois services de sécurité : l'authentification, l'intégrité et la confidentialité partielle (CH-SB). Il n'est pas couteux en termes d'espace mémoire (3 clés stockées seulement dans chaque nœud). Ce protocole élimine le référentiel à la station de base et utilise des clés asymétriques permettant de passer à l'échelle plus facilement. Cependant, ce protocole présente l'inconvénient que les auteurs se concentrent principalement sur comment sécuriser la phase de transmission des données, alors que la phase d'installation reste toujours sans sécurisation, cela explique que les nœuds malicieux peuvent participer pendant la formation des clusters et devenir CH. Un autre point négatif de ce protocole, c'est que l'opération du cryptage par la clé publique, la génération et la vérification des signatures numériques sont coûteuses en termes de temps de calcul et de consommation d'énergie.

3.9 Comparaison

Dans le Tableau 3.3, nous comparons les protocoles cités dans la section précédente selon les critères : les objectifs de sécurité assurés (authentification, confidentialité, intégrité et fraîcheur), les attaques contre le protocole LEACH, le système cryptographie utilisé, la connectivité, la mémoire de stockage, le passage à l'échelle et la consommation d'énergie. Notons que le stockage en mémoire évalué dans le tableau prend en considération seulement la taille des clés stockées dans les nœuds et non pas la taille du code des algorithmes et des primitives cryptographiques.

Critères \ **Protocoles**		**SLEACH**	**SecLEACH**	**AC**
objectifs de sécurité	Authentification	X	X	X
	Confidentialité		X	X
	Intégrité	X	X	X
	Fraîcheur	X	X	
Attaques	En écoute		X	X
	Modification	X	X	X
	Insertion	X	X	X
	Par rejeu	X	X	
	Selective forwarding	X	X	
	Hello flood	X	X	
	Sybil		X	
	Système cryptographie	symétrique	symétrique	asymétrique
	Connectivité	100%	probabiliste	100%
	Mémoire(nombre de clés stockées)	2 clés	m cles	3 clés
	Passage à l'échelle	moyen	moyen	bien
	Consommation d'énergie	moyenne	moyenne	forte

Tableau 3.3 – Comparaison entre les protocoles : SLEACH, SecLEACH et AC

3.10 Conclusion

Les réseaux de capteurs offrent une grande flexibilité de déploiement et sont de plus en plus utilisés, ce type de réseaux représente également un environnement ouvert et hostile, où les nœuds capteurs sont exposés à des menaces importantes. Ainsi que cet environnement apporte plusieurs contraintes de sécurité. Il est clair que les mécanismes de sécurité utilisés dans les réseaux traditionnels ne peuvent pas être directement appliqués aux réseaux de capteurs, vu de la différence architecturale qui existe entre ces deux types de réseaux. En conséquence, les RCSF exigent donc le développement des mécanismes de sécurité convenables à leur nature et tiennent compte de leurs caractéristiques différentes et de leurs vulnérabilités.

CHAPITRE 4

CONTRIBUTION

4.1 Introduction

Le protocole LEACH [2] est considéré comme étant le plus populaire des protocoles de routage hiérarchique basé sur les clusters qui reposent fondamentalement sur leurs CHs pour l'agrégation des données et le routage [61]. Comme la plupart des protocoles pour les réseaux de capteurs, LEACH est vulnérable à un certain nombre d'attaques de sécurité [3, 55], telles que l'écoute et l'analyse du trafic échangé, l'altération des données, le rejeu des anciens messages, l'attaque selective forwarding, l'attaque sybil et l'attaque hello flood. Les attaques impliquant des CHs, sont les plus préjudiciables. Si un nœud malicieux réussi à devenir un CH, il peut lancer des attaques comme selective forwarding pour perturber le fonctionnement du réseau. Notons qu'un nœud malicieux peut choisir de ne pas attaquer le CH, et essayer également d'injecter des données erronées dans le réseau.

En profitant des inconvénients et des avantages des protocoles SLEACH, SecLEACH et AC, nous proposons dans ce chapitre une version sécurisée [4] du LEACH qui garantit les objectifs de sécurité les plus importants et tient compte des ressources limités des noeuds capteurs. Ainsi elle résiste aux différentes attaques auxquelles LEACH est vulnérable.

Ce chapitre, a pour objet de présenter notre protocole sécurisé dérivé du protocole LEACH. Nous commençons d'abord par donner les objectifs de conception et les hypothèses prises en compte afin de le développer. Puis, nous détaillons les grandes phases d'exécution

de notre protocole. Enfin, nous élaborons l'analyse de sécurité ainsi qu'une comparaison entre ce protocole et les autres protocoles présentés dans le chapitre précédent.

4.2 Spécification générale

Pour le bon fonctionnement du protocole proposé [4], nous devons spécifier quelques hypothèses.

H1. Les nœuds capteurs sont homogènes en termes de capacité de traitement, de communication, d'énergie et de stockage. Cependant la station de base possède des ressources illimitées, elle est digne de confiance et responsable de la configuration des nœuds avant le déploiement.

H2. Un attaquant peut être passif ou actif durant le fonctionnement du réseau.

H3. La station de base et les nœuds de capteurs ne sont pas mobiles.

Le Tableau 4.1 résume les notations utilisées dans notre travail.

Notation	**Description**
SB	Station de Base
CH_i	Cluster-Head qui joue le rôle de tête du cluster
SN_j	Nœud de capteur simple
ID_i	Identité de nœud i
K_{Gp}	Clé globale
K_y^x	Clé symétrique partagée entre les deux entités x et y
K_x^+	Clé publique de l'entité x
K_x^-	Clé privée de l'entité x
$A\|\|B$	Concaténation de l'information A avec l'information B
nonce	nonce généré par l'entité
$MAC(K, M)$	Code d'authentification du message M avec la clé symétrique K
$E(K, M)$	Fonction symétrique de chiffrement du message M utilisant la clé K

Tableau 4.1 – Notation

4.3 Objectifs de conception

Dans les hypothèses mentionnées ci-dessus et tout en prenant en considération les contraintes de limitation des ressources énergétiques et physiques, nous avons proposé un protocole sécurisé basé sur LEACH qui puisse atteindre les objectifs suivants :

- Intégrer dans le protocole LEACH, un mécanisme de sécurité qui tient compte des ressources limitées des nœuds capteurs.
- Faire face aux attaques qui sont fréquemment menées contre le protocole LEACH.
- Assurer les service de sécurité : l'authentification, la confidentialité, l'intégrité et la fraîcheur des messages.
- Notre modèle de sécurité tient à sécuriser la phase d'installation et la phase de transmission du protocole LEACH.
- Empêcher les nœuds malicieux de devenir CH.

4.4 Protocole proposé

Dans cette section, nous commençons par donner une vue générale sur notre protocole ensuite nous le détaillons.

4.4.1 Vue d'ensemble du protocole proposé

Le protocole proposé est une version sécurisée du LEACH. Il repose sur des clés symétriques (*Pairwise Keys*) suivantes : *(i)* Une clé K_{CHi}^{SB} partagée entre la SB et le CHi ; *(ii)* Une clé K_{SNj}^{CHi} partagée entre le nœud SNj et le CHi qui forment le même cluster. Ces clés sont utilisées uniquement pour sécuriser les communications pendant la phase de transmission, soit entre la station de base et le cluster-Head (pour la clé K_{CHi}^{SB}), soit entre le cluster-Head et le nœud membre (pour la clé K_{SNj}^{CHi}). Comme exemple, la distribution des clés entre les nœuds qui forment notre réseau est illustrée dans la Figure 4.1. En fait, les clés sont placées dynamiquement au cours de la phase d'installation. Pour des raisons de sécurité, elles sont renouvelées à chaque itération (round). L'établissement de ces clés dans le protocole proposé dépend uniquement du mécanisme d'échange de clé ECDH.

La mise en œuvre de notre protocole se fait en trois phases. La première est une phase d'initialisation dans laquelle chaque nœud simple SN est pré-chargé avec une clé K_{G0} et

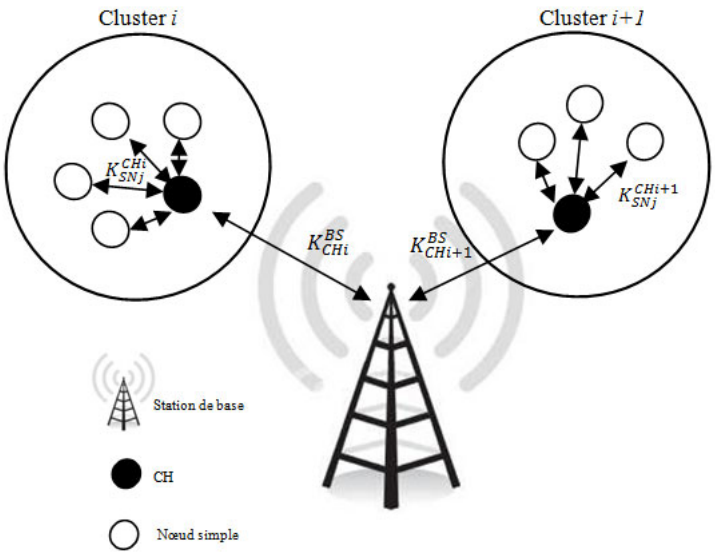


FIGURE 4.1 – Différents liens de communication à sécuriser

une paire de clés privée-publique (K_{SN}^{-}, K_{SN}^{+}). La clé K_{G0} est considérée alors comme une clé globale lors de l'itération 0. Autrement dit, cette clé sera utilisée au cours de la phase d'installation pendant l'itération 0. Elle est partagée entre chacun des nœuds déployés sur le champ et la SB. Généralement, son usage sera soit pour chiffrer des messages (Par exemple, le message de diffusion transmis par la SB pour annoncer une nouvelle itération) ou bien pour calculer un MAC. Comme mesure de sécurité, cette clé est renouvelée après la formation des clusters durant la prochaine phase. Dans la deuxième phase (phase d'installation), les CHs sont élus et les clusters sont formés, où chaque CH diffuse une annonce aux nœuds voisins, les invitant d'être membres de son cluster de manière sécurisée. Ainsi, les clés (*Pairwise Keys*) sont générées pendant cette phase. La dernière phase est la phase de transmission dans laquelle les données collectées par les nœuds simples, sont transmises aux CHs qui vont à leur tour les transmettre à la SB. Seulement, les deux phases : installation et transmission qui sont répétées à chaque itération. Dans notre protocole, les messages échangés sont cryptés, et par conséquent, la confidentialité des messages est assurée. Il convient de noter que le nœud émetteur ajoute un MAC et un *nonce* au message à envoyer pour garantir son authentification, l'intégrité et la fraîcheur de message.

4.4.2 Description des étapes du protocole proposé

Dans cette section, nous allons présenter en détail notre protocole sécurisé. Comme nous l'avons mentionné précédemment, notre protocole se déroule en trois phases à savoir :

Phase d'initialisation (avant le déploiement)

Pendant cette phase :

- la SB calcule un ensemble de clés $S = K_{G0}, K_{G1}, ..., K_{Gp}, ..., K_{Gn}$. Pour ce faire, elle génère la clé k_{Gn} et calcule la clé suivante K_{Gp} à l'aide d'une fonction de hachage F tel que :

$$\boxed{F(k_{p+1}) = k_p, 0 \leq p < n}$$

- La SB génère des paires ECC de clés (K^-_{SNj}, K^+_{SNj}) pour chaque nœud de capteur SNj dans le réseau, puis elle génère sa paire ECC de clés (K^-_{SB}, K^+_{SB}).
- Chaque nœud SNj est pré-chargé par une paire ECC de clé (K^-_{SNj}, K^+_{SNj}).
- La station de base sélectionne la clé K_{G0} de l'ensemble S, puis la pré-charge dans tous les nœuds de capteurs. Après déploiement, chaque nœud dans le réseau peut employer cette clé pour chiffrer et déchiffrer des messages échangés ou bien de calculer le MAC. Notant que la clé K_{Gp} peut être considérée comme une clé globale où elle sera utilisée pendant la phase d'installation de l'itération p.

Phase d'installation

Cette phase commence par l'annonce d'une nouvelle itération par la SB. Cette dernière chiffre son identifiant, un nonce et une valeur-seuil T en utilisant la clé K_{Gp}, elle génère un MAC et diffuse ces informations et sa clé publique dans le réseau (1). Il est à noter que dans LEACH, cette valeur-seuil représente une probabilité donnée qui est utilisée par les nœuds de capteurs pour devenir CH. En fait, si un nœud de capteur génère une valeur inférieure à T, il se comporte en tant que CH[2].

$$\begin{aligned} SB \longrightarrow broadcast : & E(K_{Gp}, ID_{SB}||nonce||T) \qquad (1) \\ & ||K^+_{SB}||MAC(K_{Gp}, ID_{SB}||nonce||T) \end{aligned}$$

Seulement les nœuds légitimes qui possèdent la clé K_{Gp}, peuvent déchiffrer le message (1) et vérifier la validité du MAC. Une fois un simple nœud devient CH, alors il chiffre son identifiant ID_{CHi}, génère un MAC et envoie ces informations à la SB. Lorsque la SB reçoit le message (2) et si le MAC est valide, le CH_i et la SB établissent la clé K^{SB}_{CHi} en utilisant le mécanisme d'échange de clé ECDH.

$$CH_i \longrightarrow SB : E(K_{Gp}, ID_{CHi}||nonce) \qquad (2)$$
$$||ID_{SB}||K^{+}_{CHi}||MAC(K_{Gp}, ID_{CHi}||nonce)$$

Après cela, le CH_i chiffre un message de notification *adv* et génère un MAC en utilisant la clé K_{Gp}. Ensuite, il diffuse ces informations et sa clé K^{+}_{CHi} à l'ensemble des nœuds.

$$CH_i \longrightarrow broadcast : E(K_{Gp}, ID_{CHi}||adv||nonce) \qquad (3)$$
$$||K^{+}_{CHi}||MAC(K_{Gp}, ID_{CHi}||adv||nonce)$$

Le nœud SN_j reçoit le message de notification et seul le nœud légitime qui peut déchiffrer (3) et décide de son appartenance à un cluster. Après cela, il envoie un message chiffré $Join\_req$ et sa clé publique K^{+}_{SNj} au CH_i choisi, d'une part, pour l'informer de cette décision et d'autre part, pour établir la clé K^{CHi}_{SNj} en utilisant le mécanisme ECDH.

$$SN_j \longrightarrow CH_i : E(K_{Gp}, ID_{SNj}||Join\_req||nonce) \qquad (4)$$
$$||ID_{CHi}||K^{+}_{SNj}||MAC(K_{Gp}, ID_{SNj}||Join\_req||nonce)$$

Lorsque le CH_i reçoit le message (4) et si le MAC est valide, le nœud membre SN_j et le CH_i établissent la clé K^{CHi}_{SNj} de la même manière comme la clé K^{SB}_{CHi}. Après la formation des clusters, la SB envoie à chaque CH_i la clé globale de la prochaine itération.

$$SB \longrightarrow CH_i : E(K^{SB}_{CHi}, ID_{SB}||K_{Gp+1}||nonce) \qquad (5)$$
$$||ID_{CHi}||MAC(K^{SB}_{CHi}, ID_{SB}||K_{Gp+1}||nonce)$$

Chaque CH crée un planning TDMA et envoie à chaque nœud membre un slot du temps $Timeslot$ durant lequel il peut transmettre ses données. Ainsi, le CH_i transmet la clé globale de la prochaine itération. Toutefois, les slots de temps et la clé globale de la prochaine itération, sont cryptés par la clé K^{CHi}_{SNj}.

$$CHi \longrightarrow SNj : E(K^{CHi}_{SNj}, ID_{CHi}||K_{Gp+1}||(Timeslot(SN_j))||nonce) \qquad (6)$$
$$||ID_{SNj}||MAC(K^{CHi}_{SNj}, ID_{CHi}||K_{Gp+1}||nonce)$$

Phase de transmission

Pendant cette phase, un nœud membre SN_j envoie à son CH_i l'information contenant la valeur captée, d'une façon chiffrée.

$$SN_j \longrightarrow CHi : E(K_{SNj}^{CHi}, ID_{SNj}||Info||nonce) \qquad (7)$$
$$||ID_{CHi}||MAC(K_{SNj}^{CHi}, ID_{SNj}||nonce)$$

le CH_i collecte les informations provenant de tous les membres légitimes de son cluster et les transmet, après l'agrégation à la SB.

$$CHi \longrightarrow SB : E(K_{CHi}^{SB}, ID_{SB}||Info||nonce) \qquad (8)$$
$$||ID_{CHi}||MAC(K_{CHi}^{SB}, ID_{CHi}||nonce)$$

la SB déchiffre le message(8) et elle vérifie la validité du MAC, dans le cas favorable, elle accepte les données agrégées sinon elle les refuse.

4.5 Analyse théorique de la sécurité

Le protocole proposé ci-dessus assure la confidentialité, car tous les messages échangés sont cryptés. Ainsi il peut garantir la fraîcheur, l'intégrité et l'authentification du nœud originaire du message transmis en ajoutant un *nonce* et un MAC au message envoyé. Notre protocole permet également de faire face aux attaques qui sont fréquemment menées contre le protocole LEACH, parmi ces attaques :

4.5.1 Attaque d'écoute (Eavesdropping)

Ce type d'attaque vise la confidentialité des messages. Elle permet à l'adversaire d'écouter facilement les transmissions pour récupérer par la suite le contenu des messages circulant dans le réseau. Notre protocole évite cette attaque par l'utilisation du cryptage à clé symétrique avec un renouvellement de clés d'une manière périodique, et cela dans le but de renfoncer la sécurité.

4.5.2 Modification/insertion des données

Cette attaque vise l'intégrité des données. L'adversaire peut altérer les messages transmis par le nœud membre dans le but de falsifier le résultat d'agrégation, par conséquent, le CH

accepte les données altérées par l'adversaire et les agrège. Ainsi, le résultat final est forcément erroné. Le protocole proposé permet de lutter contre cette attaque car le message est protégé par un MAC.

4.5.3 L'attaque par rejeu

Ce type d'attaque vise la fraîcheur des messages. Dans le protocole proposé, les messages précédents interceptés par un intrus, ne peuvent être réinjectés puisque chaque message transmis comporte une valeur aléatoire appelée *nonce*. Un message comportant un *nonce* différent du *nonce* attendu sera refusé. En outre, l'intrus ne peut pas modifier la valeur du *nonce* car le message est crypté et d'autre part, il est protégé par un MAC.

4.5.4 Hello flood

Dans LEACH, le nœud membre rejoint le cluster selon la force de signal du CH. Un nœud malicieux peut être facilement lancé l'attaque HELLO Flood où il diffuse un signal plus fort pour créer un grand nombre de nœuds ajoutés à son cluster. Puis ce nœud malicieux peut être utilisé dans d'autre attaque telles que selective forwarding, altération des message,...,etc. Dans le protocole proposé, nous garantissons la confidentialité, l'authentification et l'intégrité des deux messages *adv* et *join_req*, ce qui permet aux nœuds légitimes seulement de participer pendant la formation des clusters.

4.5.5 Selective forwarding

Cette attaque se produit souvent, dans LEACH, en combinant avec l'attaque HELLO flooding[55]. Autrement dit, si un nœud malicieux devient CH, il peut causer par la suite l'attaque selective forwarding. Notre protocole a la capacité d'empêcher le nœud malicieux de devenir CH. En plus, les clés sont renouvelées proactivement pour éviter la compromission des clés. Ce qui réduit la production de cette attaque.

4.5.6 L'attaque Sybil

Dans ce type d'attaque, un nœud malicieux peut présenter des identités multiples aux autres nœuds du réseau, afin de prendre l'avantage sur les nœuds légitimes. Cette attaque peut dégrader l'efficacité de plusieurs fonctionnalités comme la distribution de données. Les

techniques d'authentification et de cryptage peuvent empêcher un étranger de lancer une attaque Sybil sur le réseau de capteurs. Notre protocole crypte et authentifie tous les message échangés pendant la formation des cluster et la transmission de données. En plus il utilise le mécanisme ECDH qui se base sur la cryptographie à clé publique pour établir des clé (*pairwise keys*) afin d'offre l'authentification nœud-à-nœud.

4.6 Le nombre de clés stockées

Dans le protocole proposé, chaque nœud a besoin de stocker initialement trois clés seulement avant le déploiement (sa paire de clés privée-publique et la clé globale K_{G0}). Après le déploiement, chaque nœud CH_i a besoin de stocker sa paire de clés privée-publique, d clés *pairwise key* partagées entre le CH_i et ses nœuds membres, la clé globale et la clé de type *pairwise* partagée entre le CH_i et la SB. La valeur d désigne le nombre de nœuds normaux dans le c1uster. L'ensemble de clés stockées par le nœud $SN_j(non - CH)$ consiste à : sa paire de clés privée-publique, la clé globale et la clé de type pairwise partagée entre le SN_j et le CH_i. La complexité en mémoire de notre protocole est acceptable pour les capteurs de nos jours (comme MicaZ et TelosB).

Type de nœud	Nombre de clés stockées	Type de clés
CH_i	$d+4$	K^{+}_{CHi}, K^{-}_{CHi}, $d * K^{CHi}_{SNj}$, K^{SB}_{CHi}, K_{Gp}
$SN_j(Non - CH)$	4	K^{+}_{SNj}, K^{-}_{SNj}, K^{CHi}_{SNj}, K_{Gp}

Tableau 4.2 – Nombre de clés stockées

4.7 La connectivité

La connectivité d'un réseau, en termes de clé, est le fait de garantir à ses nœuds d'avoir plusieurs chemins sécurisés pour envoyer ses données. Le protocole proposé assure la connectivité totale (100%), cela à cause de l'utilisation (après le déploiement) du mécanisme d'échange de clé basé sur les courbes elliptiques ECDH permettant à deux nœuds d'établir une clé secrète commune (*Pairwise Key*). Par conséquent, n'importe quel nœud légitime peut établir un lien sécurisé avec son voisin.

4.8 Le passage à l'échelle

Notre protocole présenté ci-dessus est scalable pour supporter les différentes tailles de réseaux, du fait de notre protocole utilise des clés asymétriques permettant de passer à l'échelle plus facilement. Le protocole proposé est également flexible contre l'augmentation du réseau du fait qu'il permet aux nouveaux nœuds de rejoindre le réseau, pour ce faire, il suffit seulement de stocker deux clés asymétriques et une clé symétrique (une clé globale) dans un nouveau nœud avant son déploiement.

4.9 Comparaison

Dans le Tableau 4.3 , nous présentons une comparaison entre les protocoles sécurisés présentés dans le chapitre précédent et notre protocole en se basant bien sûr sur plusieurs critères.

Critères \ Protocoles		**SLEACH**	**SecLEACH**	**AC**	**Notre protocole**
objectifs de sécurité	Authentification	X	X	X	X
	Confidentialité		X	X	X
	Intégrité	X	X	X	X
	Fraîcheur	X	X		X
Attaques	En écoute		X	X	X
	Modification	X	X	X	X
	Insertion	X	X	X	X
	Par rejeu	X	X		X
	Selective forwarding	X	X		X
	Hello flood	X	X		X
	Sybil		X		X
	Système cryptographie	symétrique	symétrique	asymétrique	hybride
	Connectivité	100%	probabiliste	100%	100%
	Nombre de clés stockées	2 clés	m cles	3 clés	4 clés pour SN,$(d+4)$ clés pour CH
	Passage à l'échelle	moyen	moyen	bien	bien

Tableau 4.3 – Comparaison entre notre protocole et les protocoles : SLEACH, SecLEACH et AC

4.10 Conclusion

Dans ce chapitre, nous nous sommes focalisés sur la sécurité du protocole de routage hiérarchique basé sur les clusters pour les réseaux de capteurs, et nous avons proposé une version sécurisée du LEACH. Il semble que le protocole proposé améliore la sécurité du protocole LEACH en assurant les objectifs de sécurité les plus importants. Ainsi il peut être robuste face aux attaques qui sont menées contre le protocole LEACH. Le prochain chapitre sera consacré à la présentation et l'interprétation des tests et résultats obtenus par simulation pour démontrer l'aspect pratique et de mieux analyser les performances de notre protocole.

CHAPITRE 5

SIMULATION ET ÉVALUATION DES PERFORMANCES

5.1 Introduction

Dans ce chapitre, nous effectuons un ensemble de simulations par le simulateur TOSSIM pour évaluer les performances de notre approche par rapport à celle du protocole original LEACH. Nous évaluons les performances du protocole LEACH et notre protocole en termes de consommation d'énergie, de perte de données et de délai de bout en bout. Il est à noter que dans notre travail, le protocole LEACH et notre protocole sont utilisés pour la collecte de la température : Chaque nœud membre mesure la température et la transmet à son CH. Ce dernier calcule la moyenne de la température au sein de son cluster et envoie le résultat calculé à la SB.

Dans ce même chapitre, nous présentons en premier lieu l'environnement de simulation utilisé, par la suite, nous exposons un aperçu sur l'implémentation de deux protocoles (LEACH et notre protocole), puis, nous présentons les métriques de performances mesurées, les paramètres de simulation, ainsi que l'interprétation des résultats obtenus à l'issue de simulation.

5.2 Outils logiciels utilisés

Nous avons implémenté notre approche dans l'environnement TinyOS, ce qui nous permettra dans le futur de le porter facilement sur de vrais capteurs. Dans cette section, nous présentons les différents outils utilisés pour la mise en œuvre et l'évaluation de notre approche.

5.2.1 TinyOS

TinyOS [63] est un système d'exploitation open-source spécialement conçu pour les réseaux de capteurs sans fil. Il a été développé par l'université de Berkeley [64] de Californie (États-Unis). Sa programmation a été entièrement réalisée en NesC [65], c'est un dérivé du langage C. Du point de vue architectural, TinyOS est orienté événement, autrement dit, le capteur ne devient donc actif qu'à l'apparition de certains évènements (par exemple : l'arrivée d'un message radio), le reste du temps, il se trouve en état de veille. Ce processus permet de mieux économiser les ressources énergétiques des capteurs. TinyOS a été créé pour répondre aux caractéristiques et nécessités des réseaux de capteurs telles que la taille de mémoire réduite et la basse consommation d'énergie.

5.2.2 Le langage NesC

NesC [65] (*Network embeded system C*) est une extension du langage C ; il supporte alors la syntaxe du langage C et il est compilé vers le langage C avant sa compilation en binaire. Grâce à NesC, il est possible de créer une application par un assemblage de composants nécessaires à l'application. Autrement dit un composant (*Component*) est un élément de base pour former une application NesC correspond à un élément matériel (LEDs, timer, ADC ...) et peut être réutilisé dans différentes applications. Il existe deux types de composants : modules et configurations :

- Module : Il permet d'implémenter (*provides*)/ d'utiliser (*uses*) une ou plusieurs interfaces. La Figure 5.1 illustre un module nommé AppM qui fournit l'interface Primitive, et il utilise les interfaces Boot et Leds .
- Configuration : Elle permet de relier (*wired*) des composants existants pour former un nouveau composant (voir la Figure 5.2).

```
module AppM
{
      provides {
            interface Primitive;
      }
      uses {
           interface Boot;
           interface Leds;
      }
 }

implementation
{
 /* C code
  .......
  ...
 */
}
```

FIGURE 5.1 – Exemple d'un fichier module

```
configuration AppC
 {
 }
implementation
 {

  components MainC,  AppM,  LedsC;

   AppM.Boot -> MainC.Boot;
   AppM.Leds -> LedsC.Leds;

 }
```

FIGURE 5.2 – Exemple d'un fichier configuration

Une interface définit les interactions entre composants . On distingue deux sortes d'interfaces : les interfaces fournies et les interfaces utilisées. Les interfaces fournies sont prévues pour représenter la fonctionnalité que le composant fournit à l'utilisateur, les interfaces utilisées représentent la fonctionnalité dont le composant a besoin pour réaliser son travail.

5.2.3 TOSSIM

TOSSIM [66](*TinyOS SIMulator*) est un simulateur conçu pour simuler les réseaux de capteurs en utilisant le système d'exploitation TinyOS. L'objectif principal de TOSSIM est de fournir une simulation (d'application TinyOS programmée par NesC) très proche de ce qui se passe dans les RCSF dans le monde réel. Le type de nœud que TOSSIM simule est MicaZ, ce dernier se caractérise par : Microcontrôleur ATmega 128L, RAM 4 Ko, ROM 128 Ko, Radio CC2420/802.15.4, programmation NesC. Dans notre travail, nous avons utilisé la version récente de ce simulateur (TOSSIM 2).

5.2.4 Modèle de bruit

La simulation du bruit dans un RCSF est très importante, car elle permet de reproduire le comportement de la communication entre les nœuds, et de mieux tester les protocoles de routage.

TOSSIM2 utilise une méthode de modélisation du bruit proposé dans [67]. Le paquet est envoyé vers le destinataire avec une certaine force du signal , quand ce paquet atteint le destinataire, cette force est moins élevée. Dans la Figure 5.3, nous considérons deux exemples de bruit, la trace de gauche représente le modèle *Meyer heavy* de Stanford, il est considéré comme étant un modèle bruité, inspiré de la réalité. La trace de droite représente un modèle moins bruité, fait par *Casino Lab* de Colorado Schools of Mines. La force du signal est représentée par la ligne rouge, et le rapport entre elle et le bruit est appelé SNR (Signal to Noise Ratio).

5.2.5 PowerTOSSIMz

TOSSIM ne peut pas fournir des informations concernant le changement de l'état énergétique pour les nœuds du réseau. De ce fait, une extension a été indépendamment développé

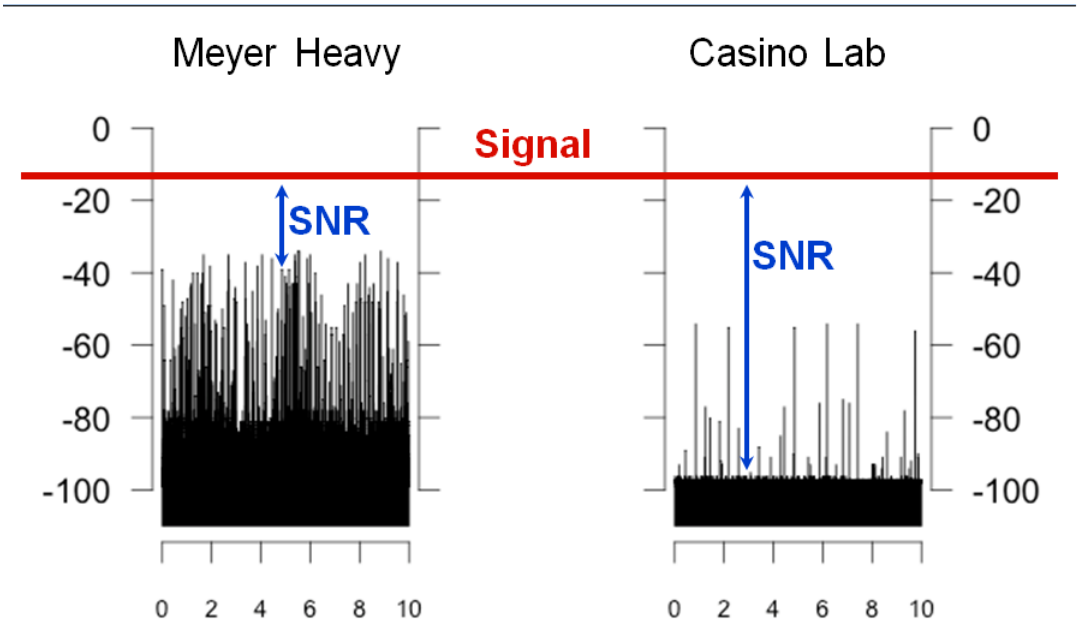


FIGURE 5.3 – Niveau de bruit [69]

pour TOSSIM, nommée PowerTOSSIMz [68] qui permet d'estimer la consommation d'énergie des capteurs (voir Annexe A). Il est à noter que cette extension modélise le comportement de nœuds de type MicaZ.

5.2.6 JTOSSIM

JTOSSIM [70] est une application Java. Elle est conçu pour être une interface utilisateur graphique (GUI) pour TOSSIM, le simulateur TinyOS. Cet outil permet aux utilisateurs de définir des paramètres de simulation comme les paramètres de la radio et de la topologie du réseau (voir Annexe B) et assure une visualisation différente des résultats des simulations. L'interface JTOSSIM est illustrée par la Figure 5.4.

5.3 Implémentations, simulation et évaluation de performances

Afin d'évaluer les performances de notre protocole, nous avons procédé à le comparer au protocole de routage original : LEACH. Pour cela :

1. Nous avons implémenté le protocole LEACH.
2. Nous avons implémenté notre protocole.

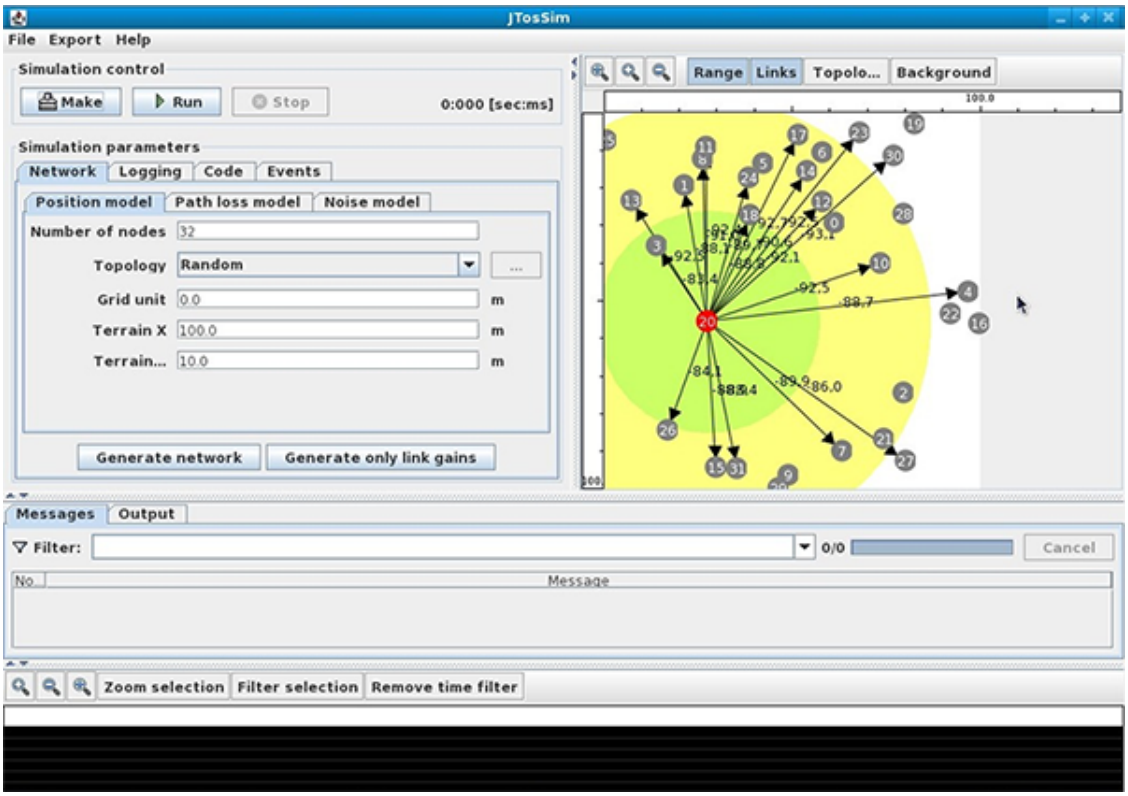


FIGURE 5.4 – L'interface JTOSSIM [70]

3. Nous avons effectué des simulations avec les mêmes paramètres et métriques pour les deux protocoles.

5.3.1 Implémentation du protocole LEACH

Dans le protocole LEACH, les structures de messages diffèrent selon le rôle du nœud (SB, CH ou nœud membre MBR).

La station de base (SB) :

- Source ID : L'identifiant de la SB qui correspond à 0.
- Destination ID : L'identifiant de destination.
- Round : L'itération courante.
- Probability : La probabilité que chaque nœud devienne CH.
- Depth : La puissance du signal d'un CH dans le réseau.

Le Custer-Head (CH) :

- Source ID : L'identifiant de CH.

Champs	**Taille (Bytes)**
Source ID	2
Destination ID	2
Round	1
Probability	4
Depth	1
TOTAL	10

Tableau 5.1 – La structure de message pour SB

Champs	**Taille (Bytes)**
Source ID	2
Destination ID	2
Data_Agre	1
Slot_Att	2
Freq	2
TOTAL	9

Tableau 5.2 – La structure de message pour CH

- Destination ID : L'identifiant du membre qui appartiendra à ce CH.
- Data_Agre : La donnée agrégée à envoyer à la SB.
- Slot_Att : Le slot attribué à chaque membre.
- Freq : La fréquence avec laquelle un membre envoie sa donnée.

Le nœud membre (MBR) :

Champs	**Taille (Bytes)**
Source ID	2
Destination ID	2
Temp	1
TOTAL	5

Tableau 5.3 – La structure de message pour MBR

– Source ID : L'identifiant de membre.

– Destination ID : L'identifiant du CH auquel appartiendra ce membre.

– Temp : La température captée.

Les principales interfaces que nous implémentons dans le protocole LEACH, sont énumérées dans le tableau ci-dessous

Interface	**Description**
LEACH_ReceiveMsg	Permet au nœud voisin de recevoir la probabilité envoyée par la SB
ANNONCE_ReceiveMsg	Permet d'annoncer de l'élection d'un CH.
ORGANISATION_ReceiveMsg	Permet de former des clusters
SLOT_ReceiveMsg	Permet aux MBR de recevoir les slots envoyés par le CH
Temperature_ReceiveMsg	Permet au CH de recevoir les données mesurées par de MBR
AGGREGATION_ReceiveMsg	Permet à la SB de recevoir les données agrégées par le CH
ReqRelayTimer.fired()	Permet de renvoyer la probabilité vers les nœuds non voisins à la SB
RoundTimer.fired()	Permet de déclencher d'une nouvelle itération (round).

Tableau 5.4 – Description des principales interfaces implémentées dans LEACH

5.3.2 Implémentation de notre protocole

En plus des champs utilisés dans les structures des messages mis en place pour LEACH, notre protocole a besoin des champs suivants :

La station de base (SB) :

Champs	**Taille (Bytes)**
Nonce	1
MAC	16
NextGlobalKey	16
PublicKey	40
TOTAL	10 + 73 = 83

Tableau 5.5 – La structure de message pour SB

- Nonce : pour assurer la fraîcheur de données .
- MAC : pour assurer l'authentification des sources de données et l'intégrité de données .
- NextGlobalKey : la clé globale de la prochaine itération (round).
- PublicKey : la clé publique de la SB.

Le Cluster-Head (CH) :

Champs	**Taille (Bytes)**
Nonce	1
MAC	16
NextGlobalKey	16
PublicKey	40
TOTAL	9 + 73 = 82

Tableau 5.6 – La structure de message pour CH

- Nonce : pour garantir la fraîcheur de données .
- MAC : pour garantir l'intégrité et l'authentification du nœud originaire du message transmis.
- NextGlobalKey : la clé globale de la prochaine itération (round).
- PublicKey : la clé publique du CH.

Le noeud membre (MBR) :

Champs	**Taille (Bytes)**
Nonce	1
MAC	16
PublicKey	40
TOTAL	5 + 57 = 62

Tableau 5.7 – La structure de message pour MBR

- Nonce : pour assurer la fraîcheur de données .

– MAC : pour garantir l'intégrité et l'authentification du nœud originaire du message transmis.
– PublicKey : la clé publique du MBR.

En plus des interfaces implémentées dans le protocole LEACH, la Figure 5.8 présente les principales fonctions/interfaces implémentées dans notre protocole.

Fonction/Interfaces	**Description**
Pre_Distribution_GlobalKey	Distribue la clé globale.
Pre_Distribution_PrivateKey	Permet à chaque nœud d'être pré-chargé par sa clé privée.
Pre_Distribution_PublicKey	Permet à chaque nœud d'être pré-chargé par sa clé publique.
ECDH.key_agree	Calcule la clé secrète partagée (SB-CH , CH-MBR)
Get_MAC	Calcule la valeur de MAC
MAC_Egal	Vérifie l'égalité entre le MAC envoyé et celui calculé
AES.Encrypt	Est utilisé pour chiffrer des données
AES.Decrypt	Est utilisé pour déchiffrer des données

Tableau 5.8 – Description des principales fonctions /interfaces de sécurité qui sont implémentées dans notre protocole.

❖ Pour calculer la valeur de MAC, nous avons utilisé l'algorithme SHA-1. Ce choix revient au SHA-1 qui est considéré comme sûr (il ne permet à deux messages différents de générer la même empreinte). Nous avons utilisé dans notre proposition l'algorithme asymétriques basés sur les courbes elliptiques (ECDH) détaillé dans la section 3.6.1. L'implémentation de cet algorithme est basée sur la version 2.0 de TinyECC [44], qui est une librairie configurable pour les opérations d'ECC dans les RCSF.

❖ Les principaux modules utilisés dans notre application sont les suivants :
– **CRoutingM** : Contient le code de notre application, en mettant en œuvre des interfaces (vues précédemment).
– **ECDHC** : Contient le mécanisme d'échange de clé basé sur les courbes elliptiques
– **SHA1M** : Est utilisé pour calculer la valeur de MAC
– **AESM** : Sert à chiffrer/déchiffrer une donnée

5.3.3 Métriques évaluées

Pour évaluer les performances de deux protocoles (LEACH et notre solution), nous avons utilisé les métriques suivantes :

La consommation d'énergie

L'efficacité en consommation d'énergie représente une métrique de performance significative, qui influence directement sur la durée de vie du réseau en entier. Pour cela, nous analysons l'impact de mécanisme de sécurité intégré dans notre protocole sur l'énergie consommée par rapport au protocole LEACH. Pour se faire, nous prenons comme critère, l'énergie moyenne consommée par chaque nœud du réseau.

La perte de paquets

Cette métrique représente le rapport entre le nombre de messages perdus par l'ensemble des nœuds et le nombre de messages envoyés dans tout le réseau. Pour la mesurer, nous calculons la moyenne des taux de perte de paquets de températures entre les membres et leurs CH, et de paquets d'agrégation de ces températures entre les CH et la station de base.

Le délai de bout en bout (EED :End to End Delay)

Le choix de cette métrique, comme étant une mesure de performance, revient à sa nécessité dans certaines applications en temps-réel où on est obligé d'obtenir l'information le plutôt possible, afin de prendre les mesures nécessaires. Certains auteurs utilisent comme critère le temps moyen pour qu'un paquet soit acheminé d'une source jusqu'à la station de base. Cependant, les CH dans le protocole LEACH et notre protocole ne permettent pas d'utiliser ce critère. En effet les températures captées par les membres ne sont pas envoyées à la SB, mais, elles sont agrégées par les CH. Rappelons que ces derniers attendent que tous les membres achèvent l'opération de capture avant de passer à la phase d'agrégation. Donc, le critère que nous utilisons est l'EED moyen de tous les paquets transitant dans le réseau.

$$EED = \frac{\sum Temps\ de\ livraison\ d'un\ paquet}{Nombre\ de\ paquets\ reçus}$$

5.3.4 Paramètres de simulation

Nous avons effectué nos simulations selon différentes topologies de réseau, en utilisant le simulateur TOSSIM, alors nous avons considéré quatre réseaux avec des densités de 50, 100, 150 et 200 nœuds. Il est à noter que ces nœuds sont homogènes et déployés d'une manière aléatoire. La durée de simulation étant identiques pour chaque réseau, il est de l'ordre de 500 secondes. Le nombre de CH choisi est fixé à 10% par rapport au nombre de nœuds de capteurs à l'intérieur du réseau. Le résultat que nous allons présenter est la moyenne de cinq résultats de simulation pour chaque réseau.

Le tableau 5.9 résume les paramètres utilisés dans nos simulations

Paramètre	**Valeur**
Nombre de nœuds du réseau	200 nœuds
Durée de la simulation	500 secondes
Dimension du réseau (m^2)	100 x 100m
Modèle de topologie	Aléatoire statique
Modèle de bruit	Meyer heavy
Type de capteur	MicaZ
Chip radio	CC2420
L'énergie initiale	21600 J

Tableau 5.9 – Paramètres du contexte de la simulation

5.3.5 Résultats et interprétations

Dans cette section, nous allons présenter et analyser les résultats obtenus suivant les métriques discutées précédemment et les comparons pour les deux protocoles (LEACH et notre protocole).

La consommation d'énergie

Pour avoir les tests en consommation d'énergie, nous avons utilisé le fichier trace généré par l'extension PowerTOSSIMz. Les cas de figures suivants peuvent se présenter :

A. Consommation d'énergie des CH et des membres sur un échantillon de 50 nœuds

Dans ce test, nous avons mesuré le taux de consommation d'énergie des CH par rapport aux nœuds membres (MBR) pour LEACH et notre protocole.

	LEACH	**Notre protocole**
Consommation énergétique des MBR (Joules)	29.522	29.922
Consommation énergétique des CH (Joules)	34.946	35.969
Énergie additionnelle des CH par rapport aux nœuds MBR	15.52%	16.81%

Tableau 5.10 – Consommation d'énergie des CH par rapport aux membres

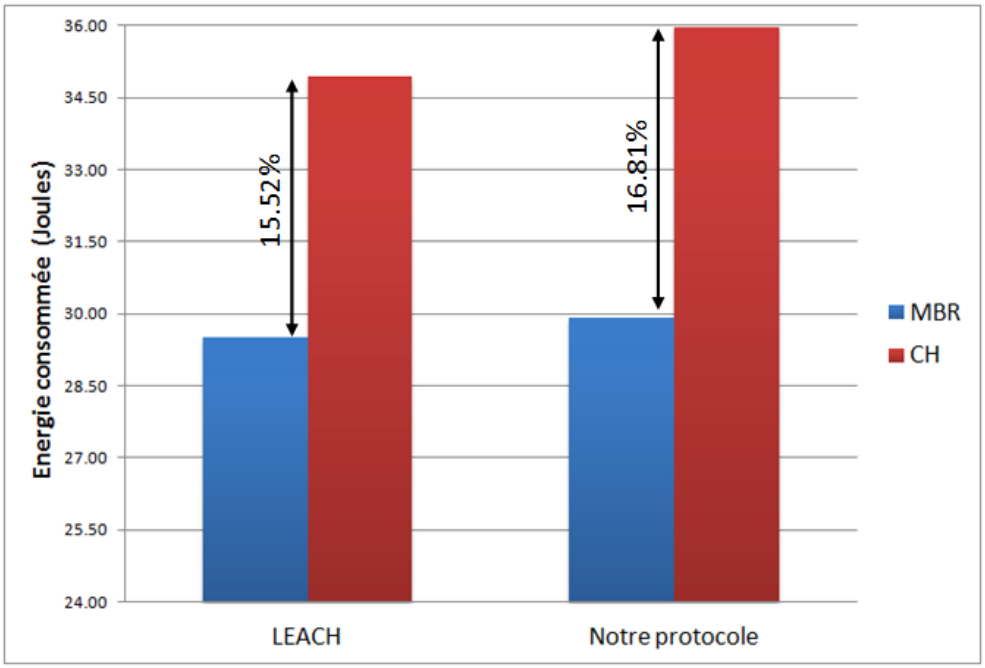


FIGURE 5.5 – Energie moyenne consommée par les CH et les membres

La Figure 5.5 illustre l'énergie moyenne consommée par les CH et les nœuds membres. On remarque que la moyenne de consommation d'énergie des CH dans le protocole LEACH est relativement plus élevée par rapport aux nœuds membres avec un taux de 15,52%.

Cette augmentation dans la consommation d'énergie est expliqué que les CH sont en activité permanente puisqu'ils communiquent soit avec l'ensemble des nœuds appartenant à leurs clusters soit ils échangent des données avec la SB. Dans notre protocole, le CH effectue en plus de ces tâches, le chiffrement des données, le calcul des MAC et l'établissement des clés

secrètes communes. Dès lors, ce taux atteint 16,81% dans notre protocole. Par ailleurs, notre protocole maintient l'énergie additionnelle pour les CH par rapport aux nœuds membres.

B. Consommation d'énergie par nœud sur un échantillon de 50 nœuds

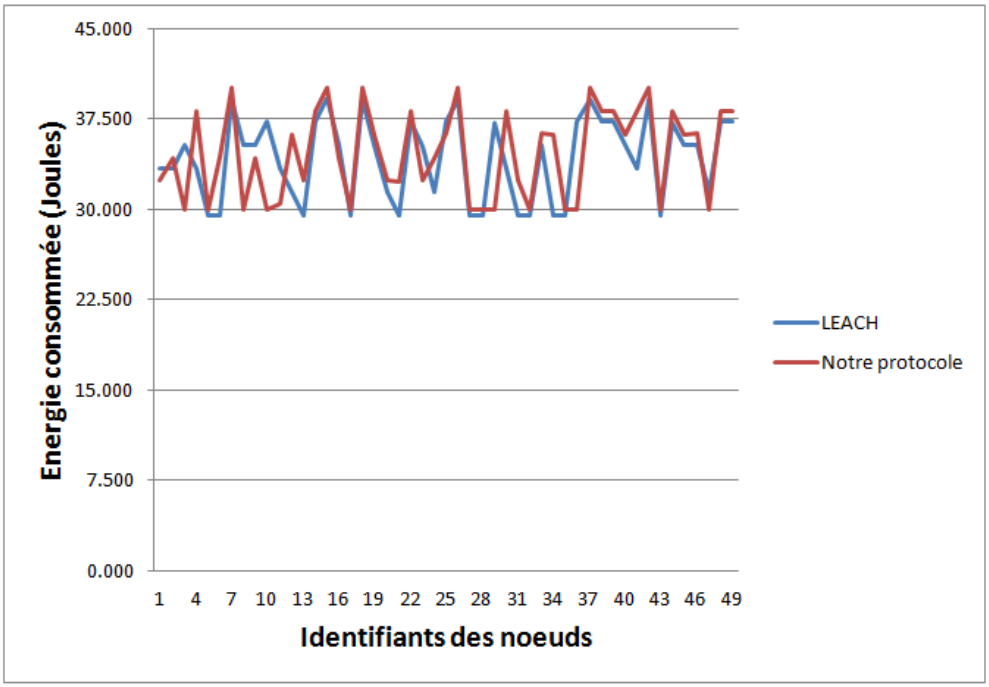


FIGURE 5.6 – Énergie consommée par nœud

Comme l'illustre la Figure 5.6, les sommets représentent l'énergie consommée par des nœuds qui ont été élus CH durant la simulation. Nous pouvons bien constater que les nœuds dans notre protocole consomment plus d'énergie que ceux du protocole LEACH, d'un taux approximativement égal à 2.20%, à cause de l'utilisation des opérations cryptographiques.

C. Consommation d'énergie suivant un réseau de différente taille

Nombre de nœuds	50	100	150	200
Moyenne de consommation d'énergie dans LEACH (Joules)	33.833	33.385	33.496	33.397
Moyenne de consommation d'énergie dans notre protocole (Joules)	34.594	34.150	34.261	34.176

Tableau 5.11 – Moyenne de consommation d'énergie dans LEACH et dans notre protocole

La Figure 5.7 illustre l'énergie moyenne consommée suivant un réseau de différente taille. D'après cette figure, nous pouvons remarquer que l'ecart entre notre protocole et le protocole

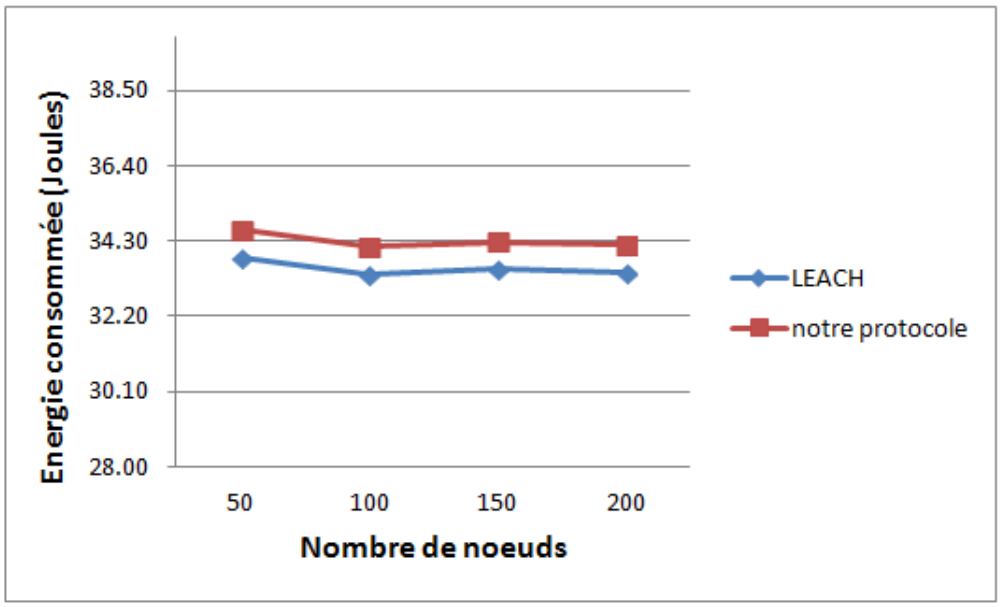


FIGURE 5.7 – Energie moyenne consommée dans le réseau

LEACH est faible, avec un taux de 2.37% d'énergie dissipée pour notre protocole par rapport au protocole LEACH. Ceci vient du fait que, dans notre protocole les nœuds de capteur font un calcul supplémentaire pour la sécurité.

La perte de paquets

Pour tester le taux de pertes de paquets, il est nécessaire de calculer le rapport des paquets perdus et des paquets envoyés. Voici le tableau des résultats de ce test :

	50 nœuds	**100 nœuds**	**150 nœuds**	**200 nœuds**
LEACH	13/168	21/263	36/418	47/525
Notre protocole	15/177	23/259	41/440	52/543

Tableau 5.12 – Nombre de paquets perdus sur le nombre de paquets envoyés

Nous définissons le taux de perte par la formule suivante :

$$LR = \frac{LP}{SP} * 100\%$$

Où :

- LR : "Loss Rate" taux de perte(%).
- LP : "Lost Packet" nombre de paquets perdus.
- SP : "Sent Packet" nombre de paquets envoyés.

D'après la formule précédente, on a ainsi, les taux de pertes de paquets (Tableau 5.13).

	50 nœuds	**100 nœuds**	**150 nœuds**	**200 nœuds**
LEACH	7.74%	7.98%	8.61%	8.95%
Notre protocole	8.47%	8.88%	9.32%	9.58%

Tableau 5.13 – Taux de pertes de paquets

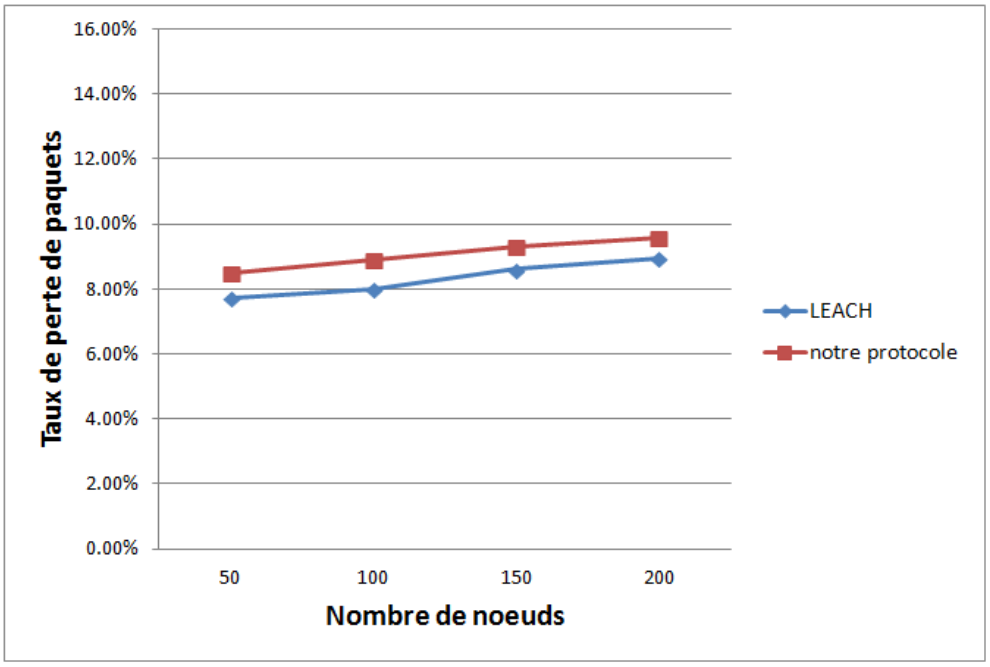


FIGURE 5.8 – Taux de pertes de paquets selon la taille du réseau

La Figure 5.8 montre l'effet du nombre de nœuds sur le taux de perte LR.

Nous remarquons la présence des taux de perte de paquets tolérables pour les deux protocoles. Ils varient entre 7.74% et 9.58% selon le nombre de nœuds déployés.

Une fois le nombre de nœuds augmente, le taux de perte augmente. Cela est dû à la présence de bruit dans les communications. De plus, nous constatons que notre protocole maintient un taux de perte satisfaisant et légèrement élevé que le protocole LEACH. Ceci peut être expliqué par le mécanisme de l'authentification implémenté dans notre protocole, qui élimine tout paquet dont la source n'est pas authentifiée.

Le délai de bout en bout

Pour évaluer le délai de bout en bout (EED : End to End Delay), nous avons eu recours au fichier trace généré par l'outil JTOSSIM. En effet, ce fichier nous a permis de récupérer les temps des émissions et des réceptions de paquets de température(le Tableau 5.14)

Nombre de nœuds	**50**	**100**	**150**	**200**
EED moyen dans LEACH (millisecondes)	6.579	6.501	6.415	6.560
EED moyen dans notre protocole (millisecondes)	8.364	8.454	8.461	8.482

Tableau 5.14 – EED moyen dans LEACH et dans notre protocole

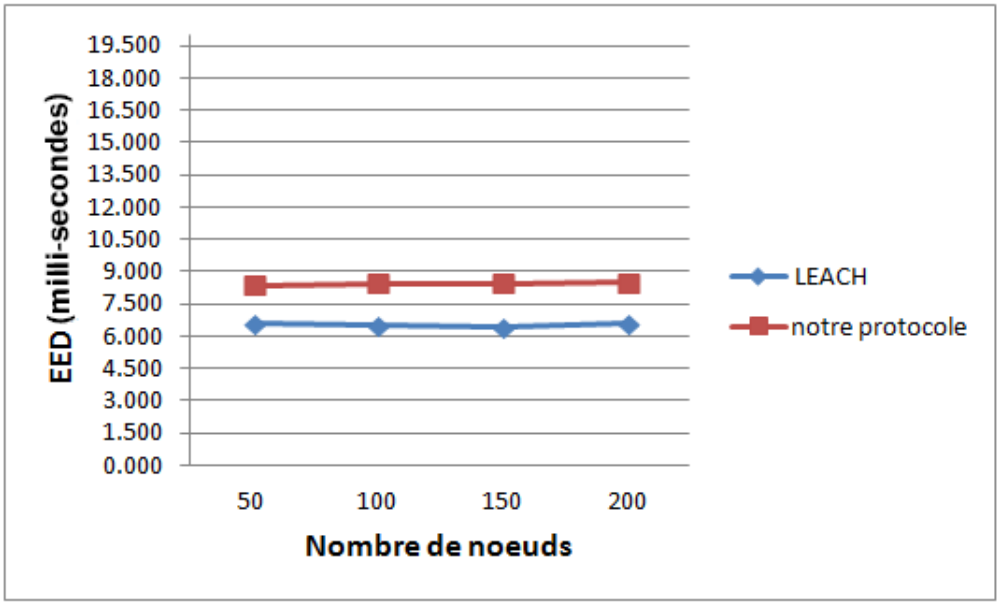


FIGURE 5.9 – Moyenne des EED selon la taille de réseau

D'après la Figure 5.9, nous remarquons que les EED sont indépendants du nombre de nœuds déployés dans le réseau. En effet, l'échange d'information (la température mesurée) se fait à saut unique.

L'information issue des nœuds membres passe directement vers les CH correspondants. Ces derniers envoient l'information agrégée directement vers la SB. Bien que la taille du réseau augmente, cela n'influe pas sur le délai bout en bout. L'écart constaté entre les deux protocoles est également faible. Ceci signifie que le chiffrement des messages le calcul des MAC ne prend pas beaucoup de temps.

5.4 conclusion

Dans ce chapitre, nous avons évalué les performances de deux protocoles (le protocole LEACH et notre proposition) à travers des simulations effectuées sur : la consommation d'énergie, la perte de paquets de données échangés et le délai de bout en bout . Ces deux protocoles ont été implémentés par le langage NesC et simulés dans TOSSIM (le simulateur de TinyOS). Nos résultats ont montré que la charge générée par notre protocole est presque faible par rapport au protocole LEACH. En plus, notre protocole permet de stabiliser la consommation d'energie et le délai de bout en bout, même si le nombre de nœuds augmente dans le réseau.

CONCLUSION GÉNÉRALE

Le routage hiérarchique utilisant l'approche du clustering, est considéré comme un processus qui a montré son efficacité en termes de réduction de la consommation d'énergie afin d'augmenter la durée de vie du réseau. Ce type de routage consiste à regrouper les nœuds du réseau en clusters (groupes) où chaque cluster est représenté par un nœud CH (Cluster-Head). Les CH se chargent de l'agrégation et la transmission de données vers la SB, alors que, les autres nœuds membres se chargent de la transmission de données locale vers leurs Cluster-Head. Dans les RCSF. La sécurisation du routage hiérarchique basé sur les clusters reste un problème majeur. Nous devons non seulement éviter de nombreuses attaques causées par les nœuds malicieux, mais aussi assurer que la dégradation de la performance causée par le mécanisme de sécurité soit limitée.

Dans ce mémoire, nous nous sommes intéressés à l'étude des problèmes de la sécurité de routage hiérarchique, basé sur les clusters dans les réseaux de capteurs de point de vue théorique. Cette étude a révélé un nombre de difficultés liées à l'absence d'infrastructure centralisée, la contrainte d'énergie, les ressources limitées,...etc. Notre travail se focalise sur la sécurité du protocole de routage hiérarchique basé sur les clusters(LEACH), dans le but de proposer une solution de sécurité efficace qui permette de préserver les performances globales du réseau. Pour cela, nous avons étudié les différentes attaques qui ciblent ce type de protocole. Nous avons trouvé que LEACH n'a pas une stratégie claire pour séparer les nœuds légitimes de ceux malicieux. Un nœud légitime ou malicieux peut se joindre sans distinction au réseau. En plus, si un nœud malicieux réussi à devenir un CH, il aura un avantage dans le réseau du fait qu'il peut écouter, modifier le trafic qui transite par lui ou lancer des attaques

comme selective forwarding et l'attaque par rejeu. Notons qu'un nœud malicieux peut tenter également d'injecter des données erronées dans le réseau. Ceci présente un danger pour la confidentialité, l'intégrité, la fraîcheur et l'authentification du noeud originaire du message transmis.

Notre contribution consiste à proposer un nouveau protocole hiérarchique sécurisé. Ce protocole est considéré comme une version sécurisée du protocole LEACH. Il semble que notre protocole proposé augmente le niveau de sécurité pour le protocole LEACH en assurant les objectifs de sécurité : l'authentification, la confidentialité, l'intégrité et la fraîcheur des messages transmis. Ainsi il permet de renforcer la robustesse face aux attaques qui sont fréquemment menées contre le protocole LEACH. De plus, notre proposition empêche un nœud malicieux de devenir CH ou d'envoyer des données falsifiées au CH.

Après la prise en main de l'environnement de simulation (malgré qu'elle nous a pris un grand temps), nous avons implémenté une version sécurisée du protocole LEACH. Par la suite, nous avons effectué nos simulations selon des différentes topologies et densités de réseau. Plus précisément, nous avons analysé les performances de notre proposition et le protocole LEACH en termes de consommation d'énergie, de délai de bout en bout et de perte de paquets. Les résultats obtenus sont comparables pour les deux protocoles.

Les résultats de la simulation montrent clairement que notre proposition réalise un compromis entre l'efficacité en terme de sécurité et les performances globales du réseau. Donc, notre approche est bien adaptée au contexte RCSF.

Le travail que nous avons effectué dans ce mémoire nous ouvre de nombreuses perspectives de recherche. Nous structurons nos réflexions comme suit :

- Nous prenons en considération le cas où les nœuds légitimes sont compromis par la capture physique des capteurs, autrement dit l'attaquant qui capture un nœud, peut lire sa mémoire et extraire du code et des clés cryptographiques et par conséquent peut substituer à ce nœud un « dispositif remplaçant » plus puissant comme un ordinateur portable par exemple. Ce dernier permet de simuler l'architecture du nœud qu'il remplace et de le transformer en un nœud d'attaque en tant que « nœud compromis ». Il 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. Pour résoudre ce problème, nous proposons d'ajouter pour notre protocole un mécanisme qui permet d'identifier le fait qu'un nœud est compromis.

- Nous proposons d'implémenter les protocoles SLEACH, SecLEACH et AC sous TinyOS, afin de simuler ces protocoles et les comparer avec notre proposition.

Annexes

ANNEXE A

POWERTOSSIMZ

Pour récupérer l'énergie consommée par les nœuds du réseau, il doit être suivi les étapes suivantes :

❖ Accédez à l'application à simuler. Dans cet exemple, nous utilisons l'application "Blink".

```
$ cd /opt/tinyos-2.x/apps/Blink
```

❖ Créez deux sous-dossiers "Topologies" et "Simulations" dans le dossier principal "Blink". Il est à noter que le dossier "Topologies" doit contenir les fichiers de topologies utilisées pendant la simulation. Tandis que le dossier "Simulations" contient le fichier de trace généré par la simulation.

❖ Compilez l'application

```
$ make micaz sim
```

❖ Créez le script de simulation "MySimulation.py", dans le dossier "Blink". Ce script contient le code suivant :

```
import sys
from random import *
from TOSSIM import *

t = Tossim([]) #pour créer un objet Tossim.
r = t.radio();
```

```
##########
#TOPOLOGY
##########

f = open("Topologies/topology.txt", "r") #ouvre le fichier
#topology en mode lecture.
lines = f.readlines()
for line in lines:
  s = line.split()
  if (len(s) > 0):
    if (s[0] == "gain"):
        print " ", s[1], " ", s[2], " ", s[3];
        r.add(int(s[1]), int(s[2]), float(s[3]));
        numNodes=int(s[1]);
print "numnodes ",int(s[1]);
#cette partie fait la lecture de chaque ligne
#du fichier topology.

######################
#NOISE TRACE & BOOTING
######################
noise = open("Noise/meyer-heavy.txt", "r")
lines = noise.readlines()
for line in lines:
  str = line.strip()
  if (str != ""):
    val = int(str)
    for i in range(0, numNodes+1):
      t.getNode(i).addNoiseTraceReading(val)
```

```
for i in range(0, numNodes+1):
  print "Creating noise model for ",i;
  t.getNode(i).createNoiseModel();
#cette partie fait la lecture de chaque ligne
#du fichier meyer-heavy

for i in range(0, numNodes+1):
  bootTime=i * 2351217 + 23542399;
  t.getNode(i).bootAtTime(bootTime);
  print "Boot Time for Node ",i, ": ",bootTime;
#initialisation de chaque noeud.

#########
#CHANNELS
#########
bla = open("Simulations/Energy.txt", "w");
t.addChannel("TESTLEDS", sys.stdout);

##########
#EXEC LOOP
##########
t.runNextEvent();
time=t.time();
    # 100000000000 = 100 seconds
        #c est le temps de simulation
while (time + 100000000000 > t.time()):
  t.runNextEvent();
#pour lancer l application dans chaque noeud
```

❖ Exécutez le script ”MySimulation.py”, ce dernier va générer un fichier ”Energy.txt” dans le sous-dossier ”Simulations”

```
$ python MySimulation.py
```

❖ Accédez au sous-dossier ”Simulations”

```
$ cd Simulations
```

❖ Exécutez ”postprocessZ.py” sur la trace de simulation ”Energy.txt”.

```
$ python postprocessZ.py Energy.txt > Result.txt
```

Le résultat enregistre l'énergie totale utilisée par chaque composant sur chaque nœud. Il est sous la forme suivante :

```
Mote 0, cpu total : 719.503906
Mote 0, radio total : 1235.255862
Mote 0, adc total : 0.000000
Mote 0, leds total : 571.570576
Mote 0, sensor total : 0.000000
Mote 0, eeprom total : 0.000000
Mote 0, cpu_cycle total : 0.000000
Mote 0, Total energy : 2526.330344

Mote 1, cpu total : 635.394462
Mote 1, radio total : 1090.990102
Mote 1, adc total : 0.000000
Mote 1, leds total : 504.416514
Mote 1, sensor total : 0.000000
Mote 1, eeprom total : 0.000000
Mote 1, cpu_cycle total : 0.000000
Mote 1, Total energy : 2230.801078
```

Pour ne pas perdre ce résultat, ce dernier est enregistré dans le fichier ”Result.txt”

ANNEXE B

JTOSSIM

B.1 Prérequis

JTossim requiert Java 1.6 ou une version ultérieure, il nécessite aussi que TinyOS 2.x et TOSSIM 2 sont bien installés.

B.2 Installation

- Allez sur le site : http ://sourceforge.net/projects/jtossim/
- Téléchargez le fichier **jtossim-0.1.1.zip**
- décompressez ce fichier dans **/opt/tinyos-2.x**. Vous obtenez un répertoire **/opt/tinyos-2.x/jtossim-0.1.1**

B.3 Lancement de JTOSSIM

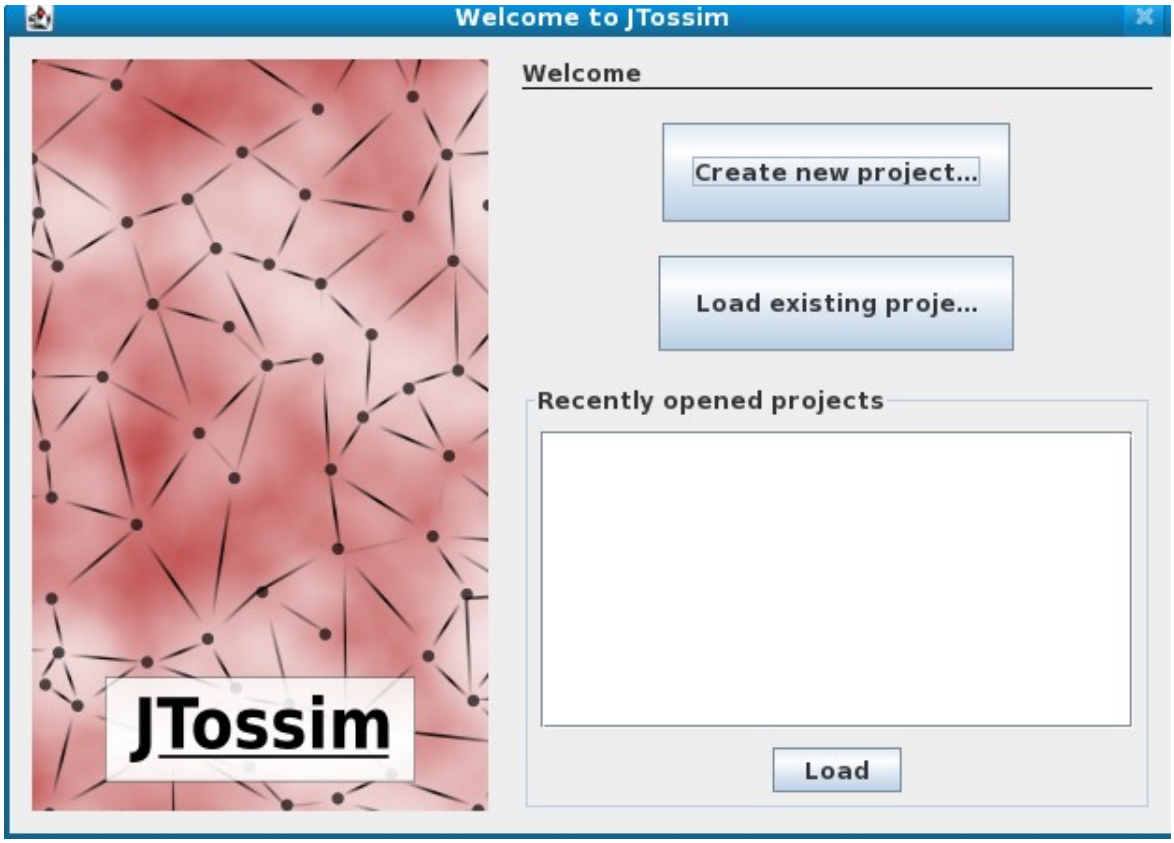
Pour pouvoir lancer Jtossim, il est nécessaire de passe par les étapes suivantes :

- Ouvrez le terminal de Linux
- Allez dans le répertoire jtossim-0.1.1 en utilisant la commande **cd**
- Pour lancer JTossim, tapez **java -jar JtosSim.jar**

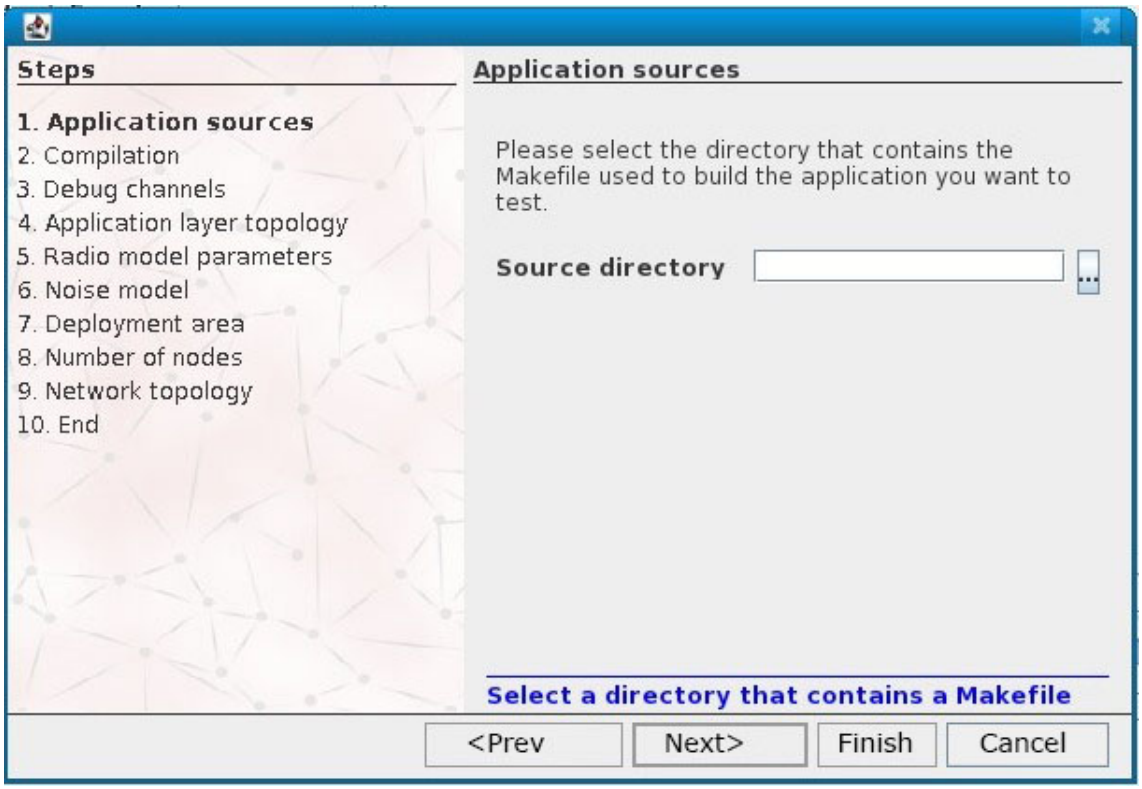
B.4 Étapes de configuration

A travers cet outil, l'utilisateur peut configurer facilement son réseau (topologie, modèle de bruit,..., etc.). La configuration se fait sur plusieurs étapes :

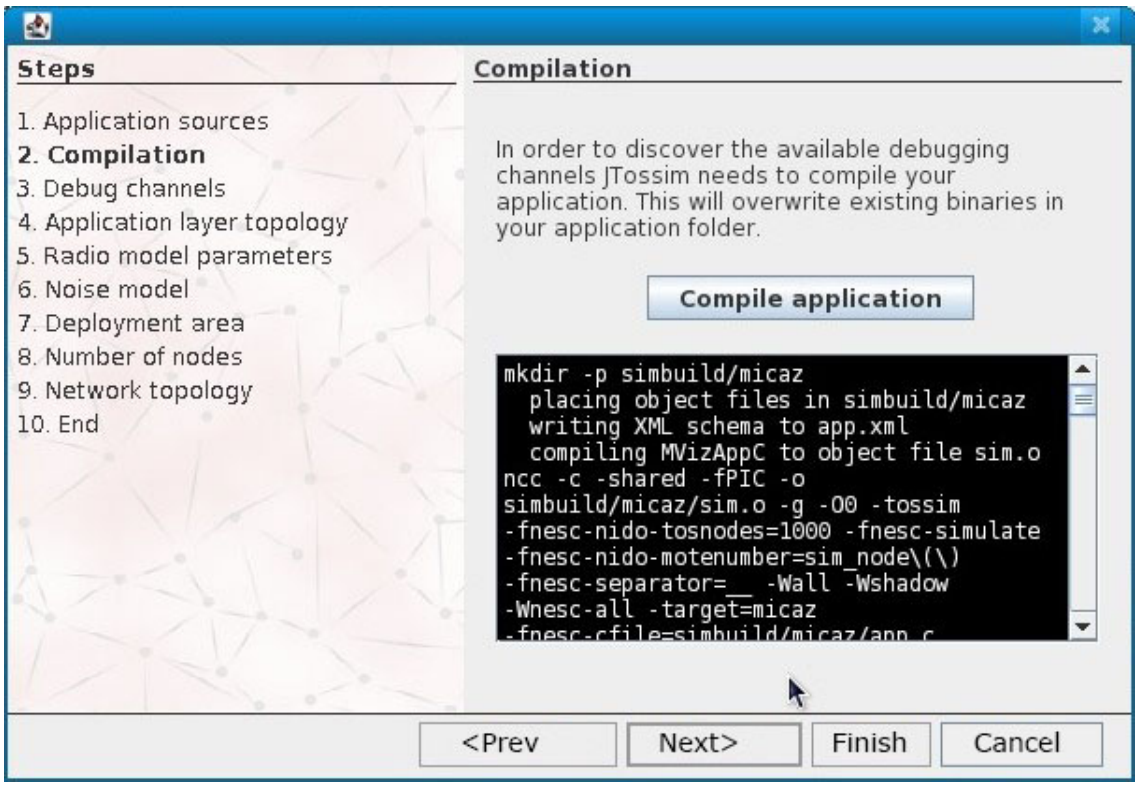
❖ Cliquez sur le bouton **Create new project**.



❖ Choisissez le dossier qui contient l'application à compiler. Ensuite, cliquez sur **Next**.

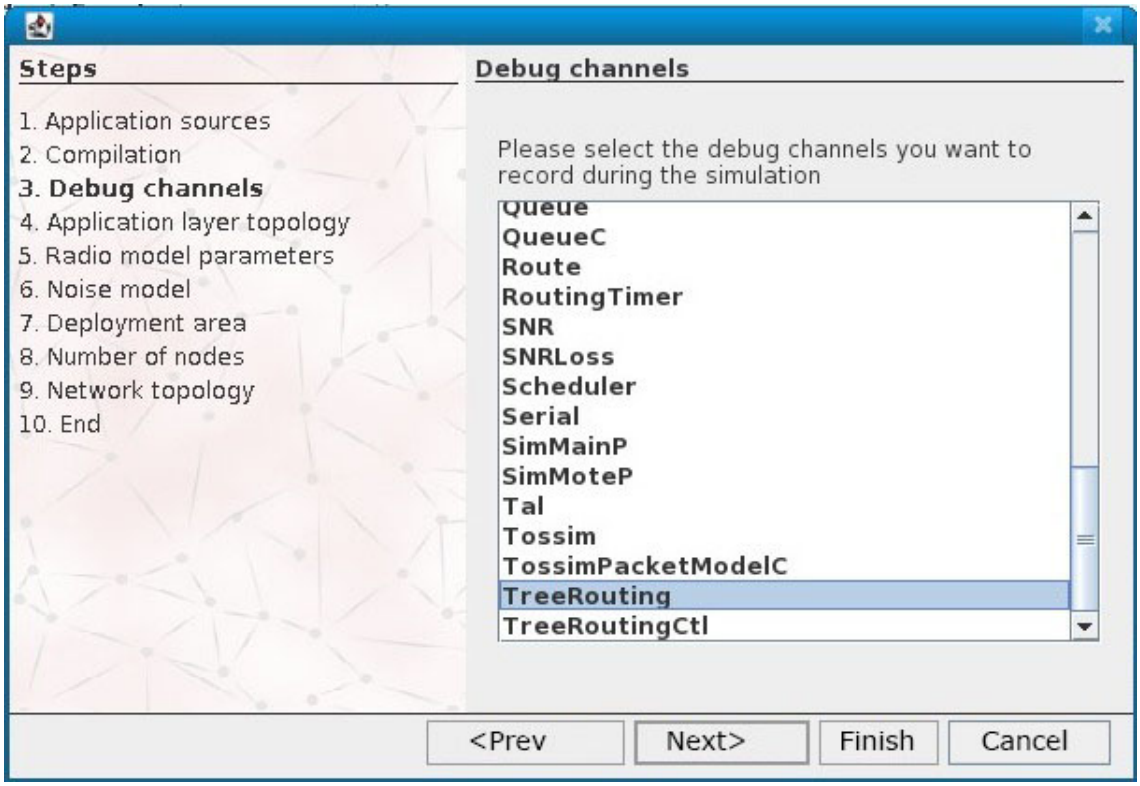


❖ Cliquez sur le bouton **Compile application**. Jtossim va exécuter la commande **make micaz sim** pour l'application sélectionnée. Une fois que la compilation est terminée sans erreurs, vous pouvez cliquer sur **Next**.



❖ Choisissez par la suite le canal de sortie (debug channel) qui est utilisé par votre application. Il est à savoir qu'un canal de sortie est une chaîne de caractères qui définit les

messages de sortie, par exemple, si on veut afficher le message "Bonjour Tout Le Monde". On doit rajouter la ligne de code suivante dans l'application : **dbg("MonApp", "Bonjour Tout Le Monde. \n)";** Après la compilation, on va choisir le canal de sortie **MonApp**.

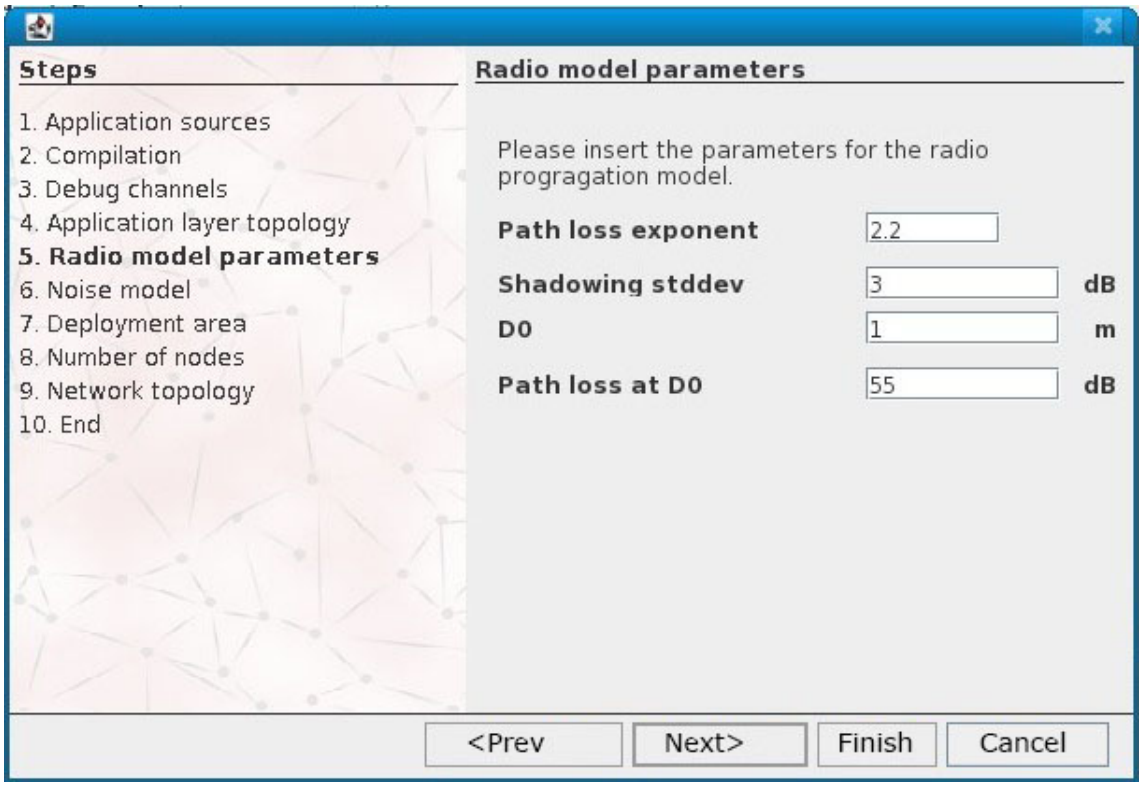


❖ Dans cette étape, vous devez spécifier les paramètres de la radio afin de calculer le gain de liaison entre chaque paire de nœuds voisins. Ces paramètres sont les mêmes utilisés dans TOSSIM. Le calcul de gain est déterminé selon la formule suivante [71] :

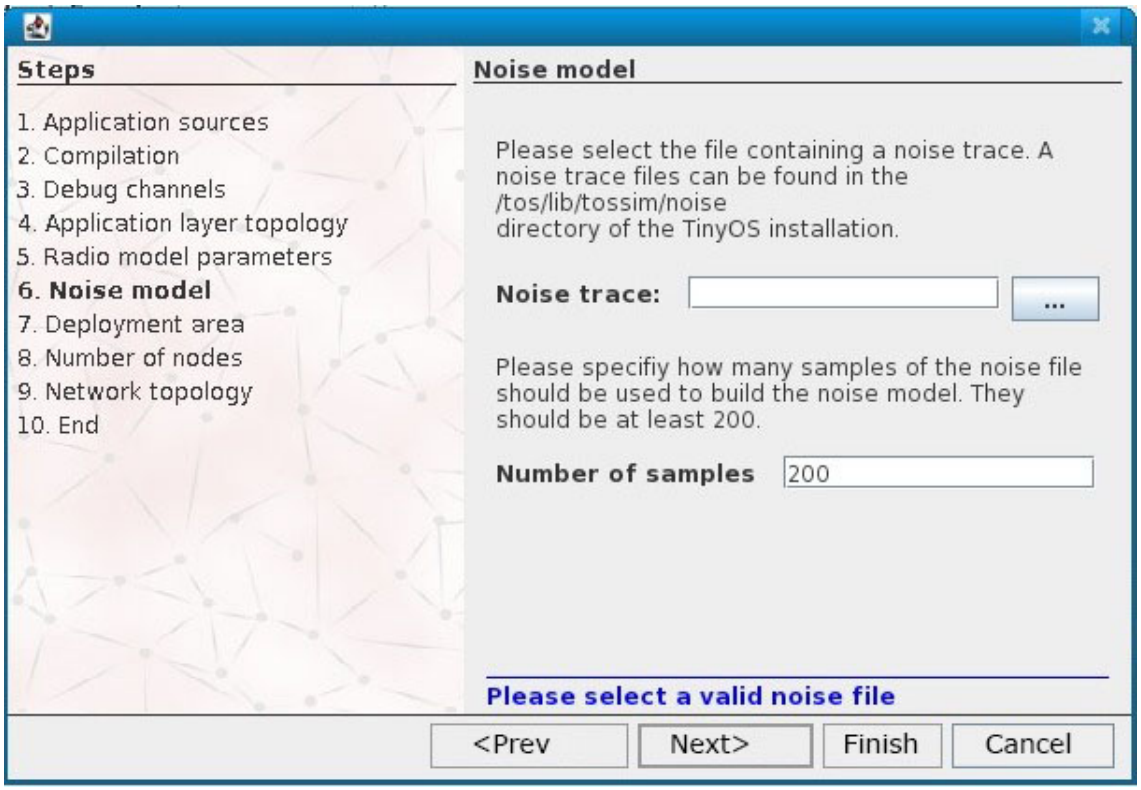
$$GAIN(A,B) = -PLD0 - 10 * PLE * math.log_{10}(\frac{dist}{D0}) + random.gauss(0, SSD)$$

Où :

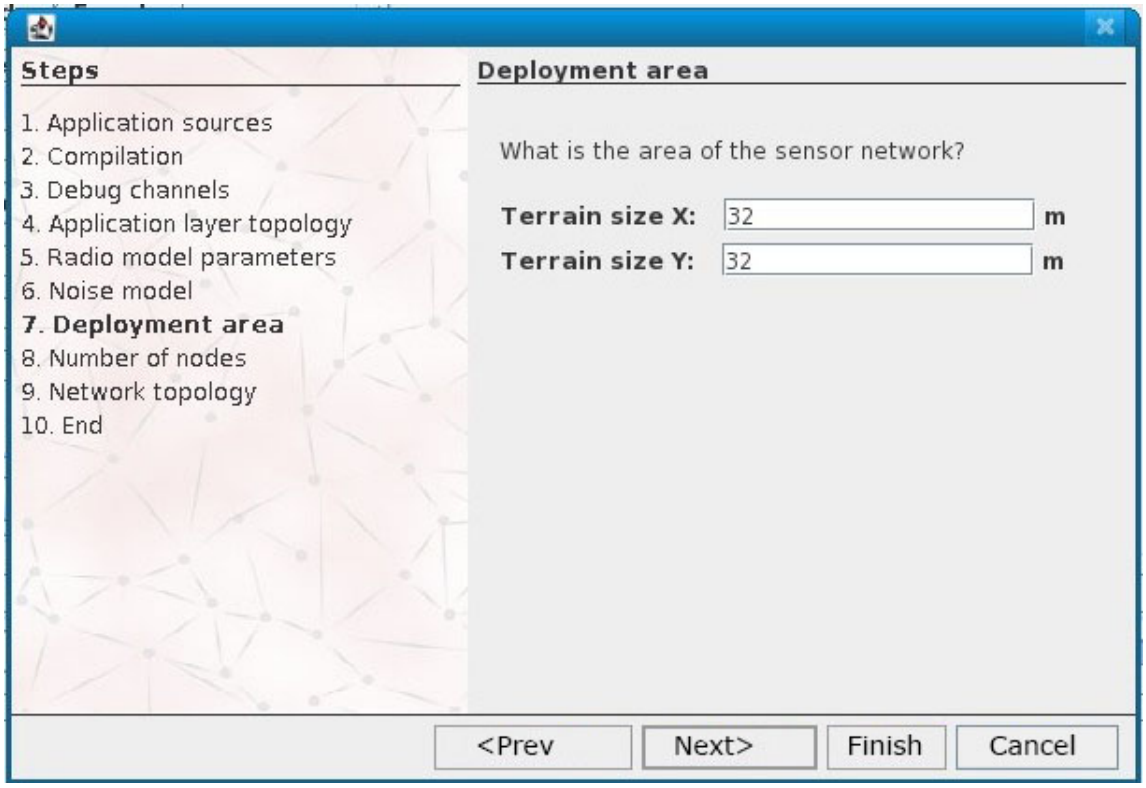
- PLE (Path loss exponent) : représente le taux à lequel le signal se dégrade.
- SSD (Shadowing stddev) : écart-type du bruit
- D0 : la distance de référence (mètres), D0 détermine également la distance minimale autorisée entre deux noeuds.
- PLD0 (Path Loss at D0) : décroissance de puissance pour le D0 (DB).
- Dist : distance entre A et B (mètre).



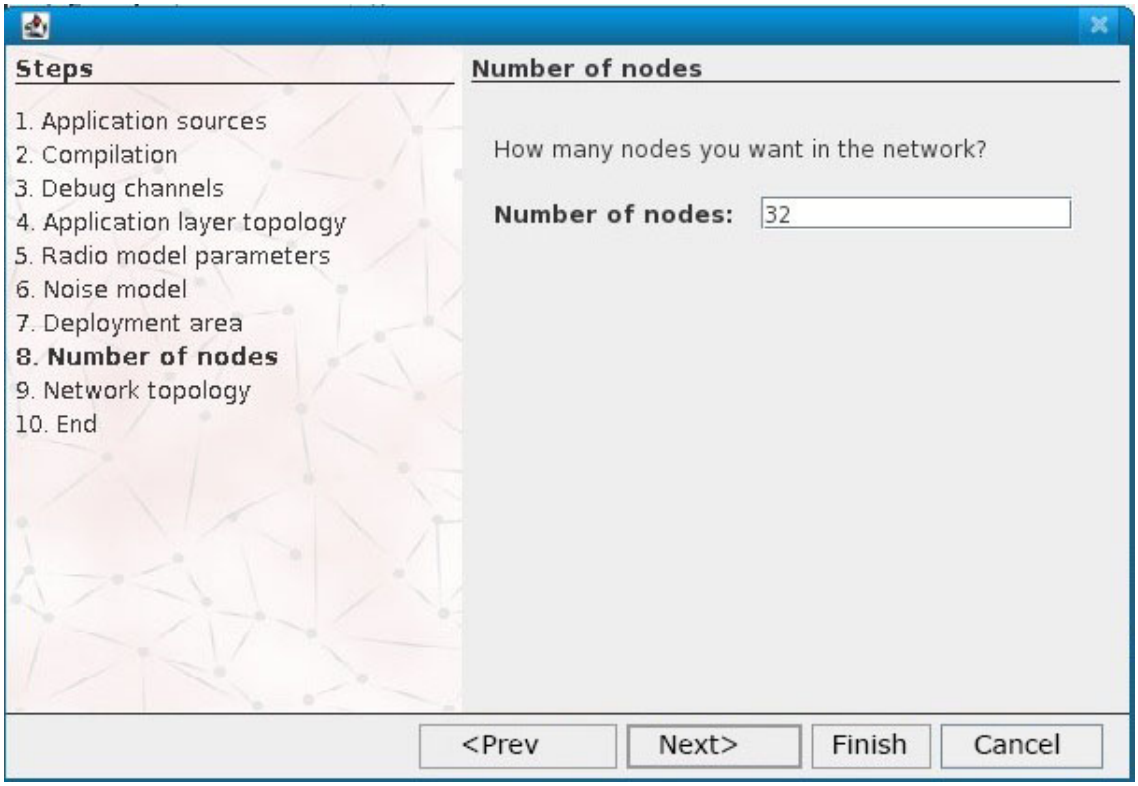
❖ Dans cette etape, vous pouvez designer le modèle de bruit à utiliser dans la simulation.



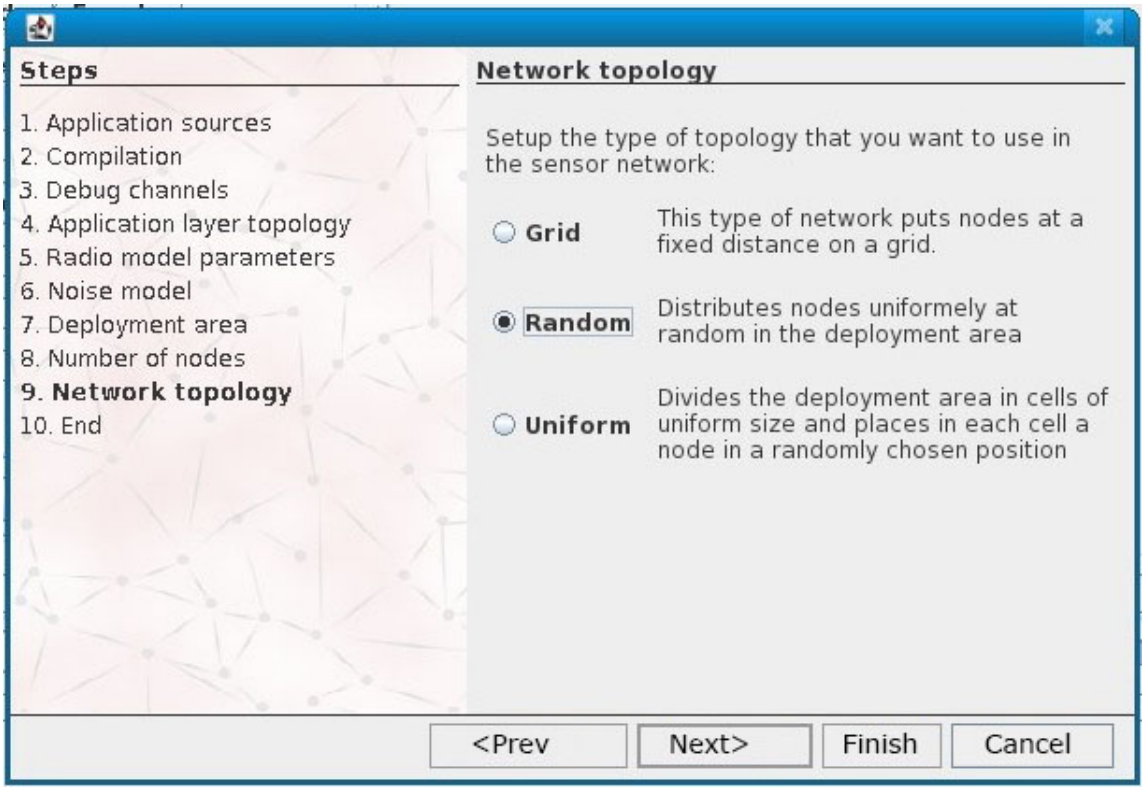
❖ Dans la prochaine étape, vous devez déterminer la taille de la zone où les nœuds seront déployés.



❖ Vous aurez juste à choisir le nombre de nœuds que vous devez déployer.



❖ Dans la prochaine étape, vous avez besoin de choisir le type de la topologie désiré.



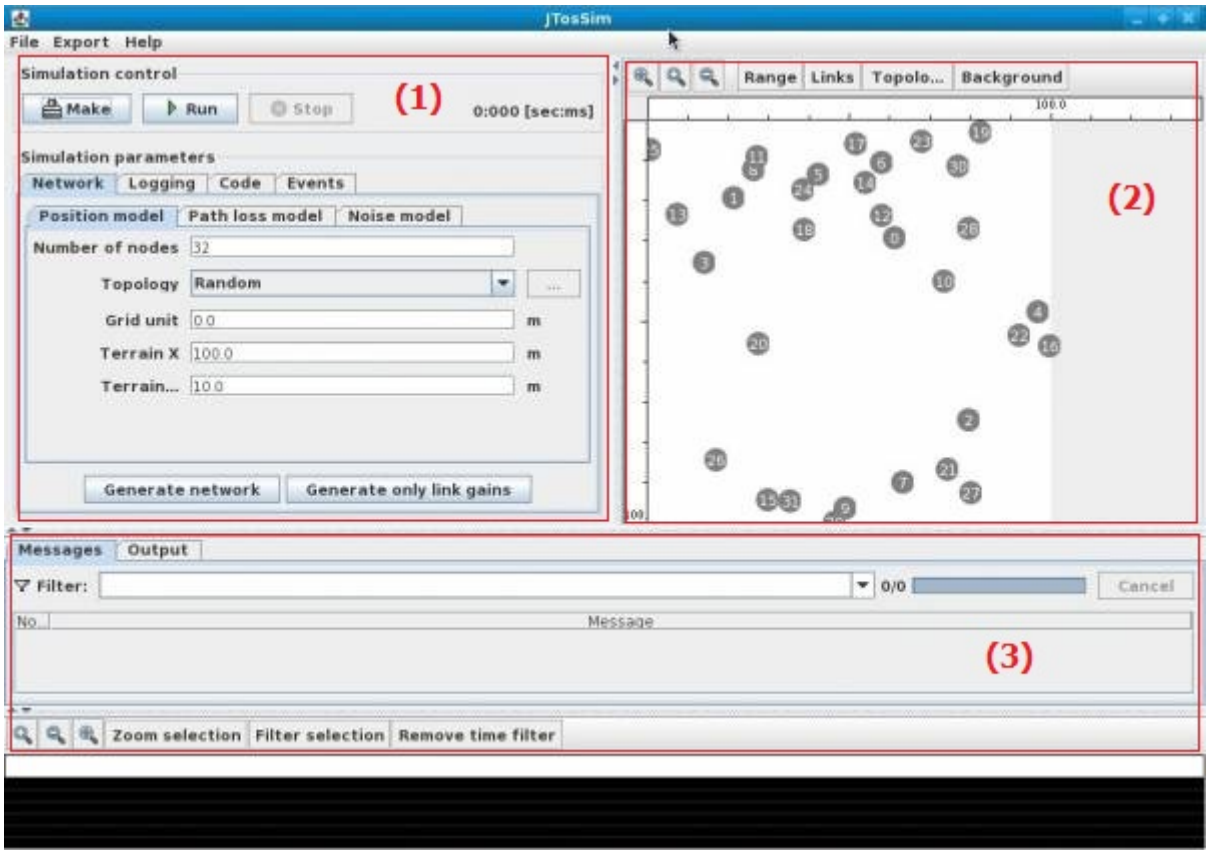
B.5 Simulation

Maintenant, nous sommes dans la fenêtre principale de JTossim. La fenêtre est divisée en trois sections principale :

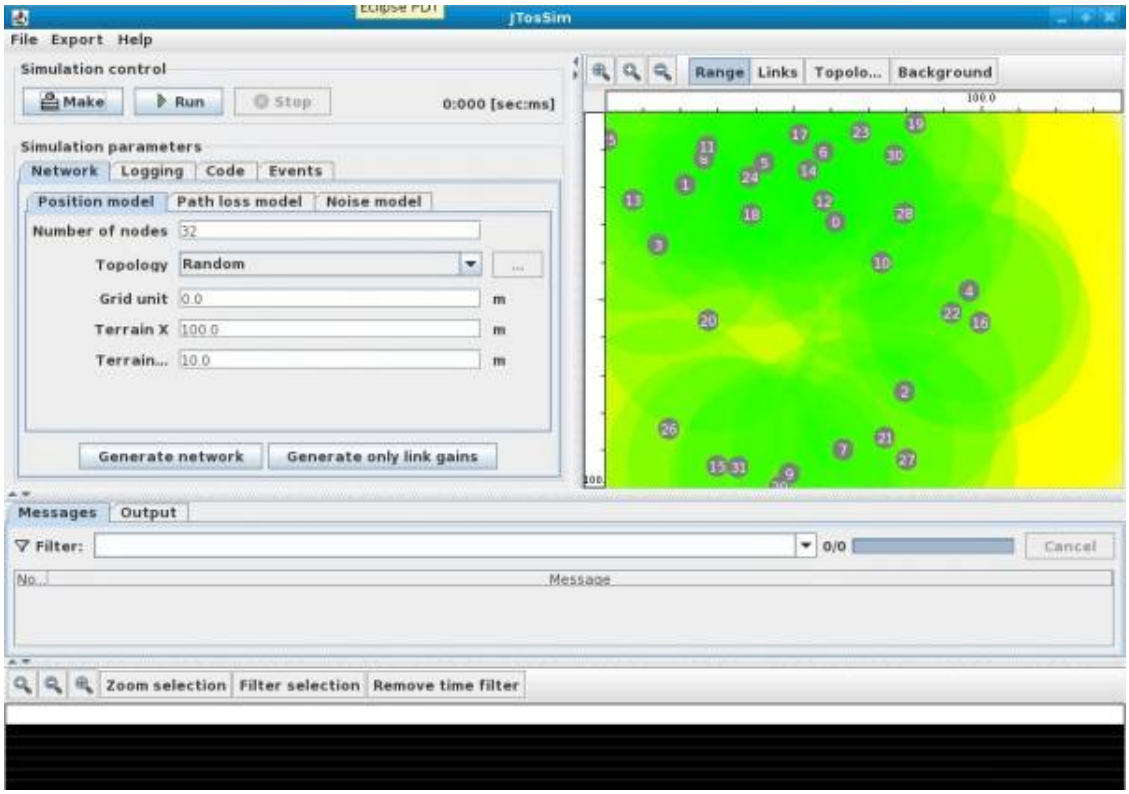
- **La section (1)** : Permet de modifier les paramètres de simulation et de contrôler l'exécution de la simulation.

- **La section (2)** : Permet de voir le déploiement du réseau. Elle permet également d'utiliser les boutons de zoom pour changer l'affichage.

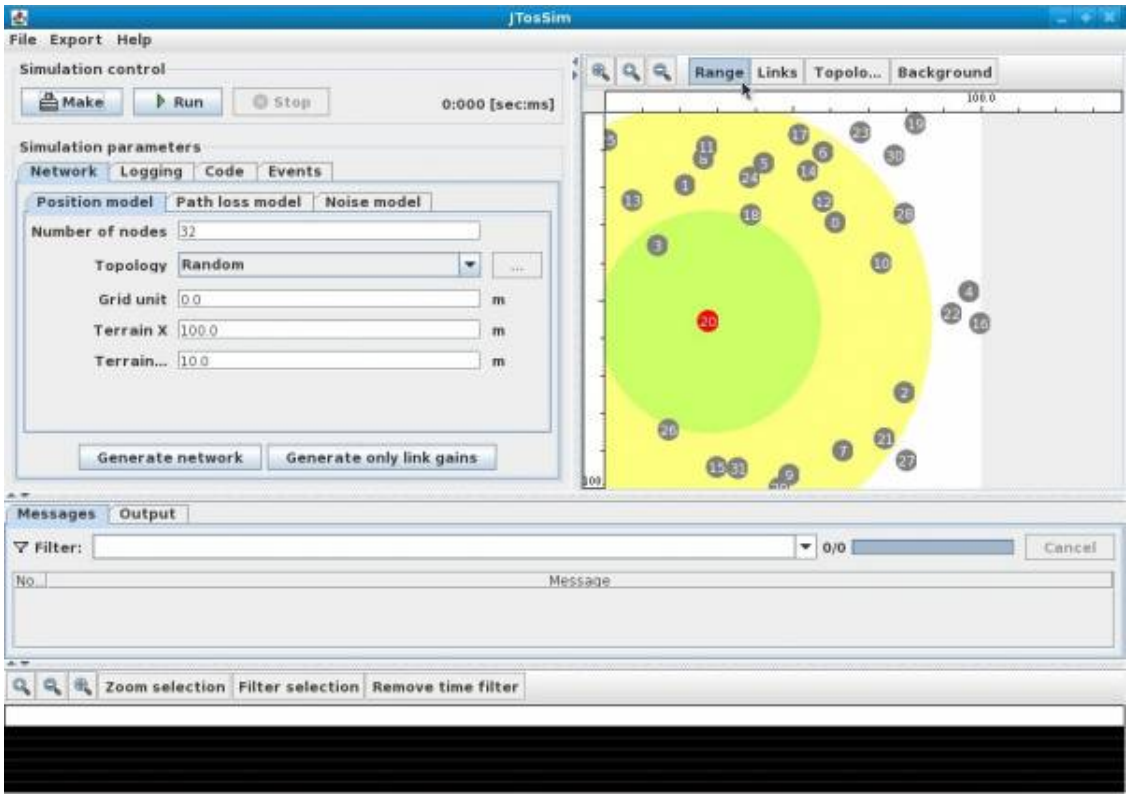
- **La section (3)** : Permet de visualiser la simulation sous forme de messages.



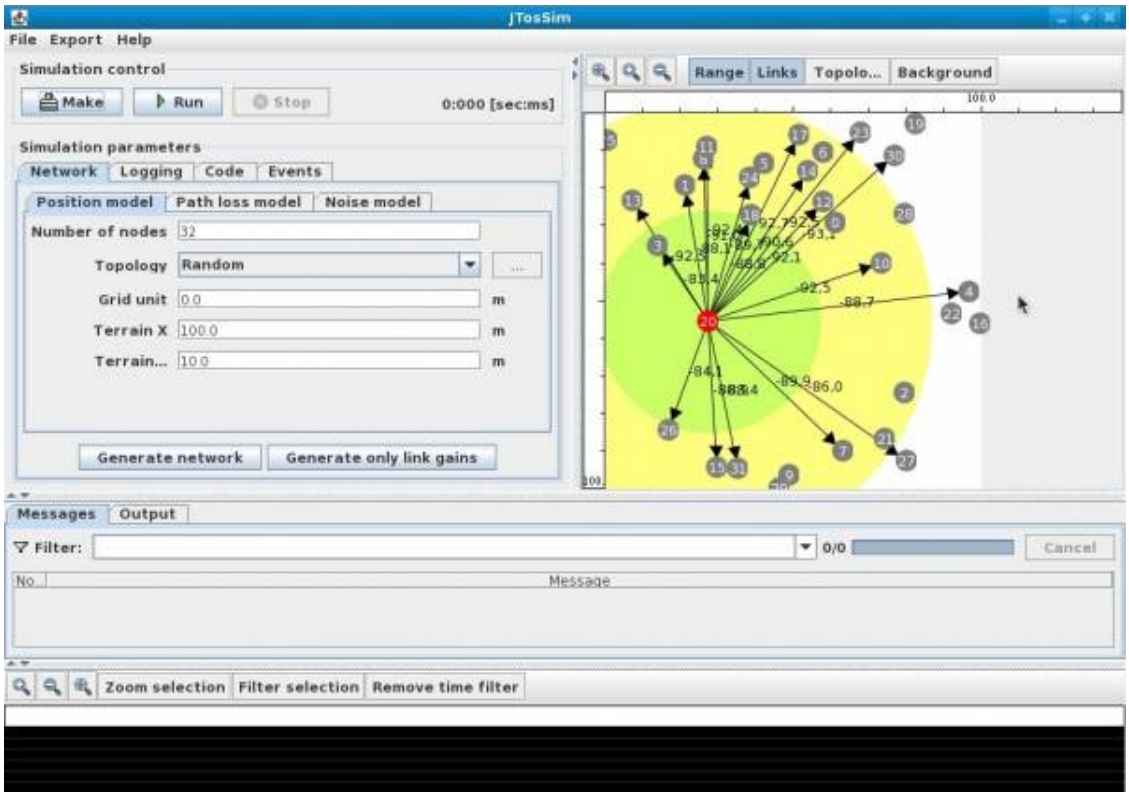
❖ Si vous cliquez sur le bouton ”Range” dans la section (2), vous verrez la portée radio de chaque nœud déployé.



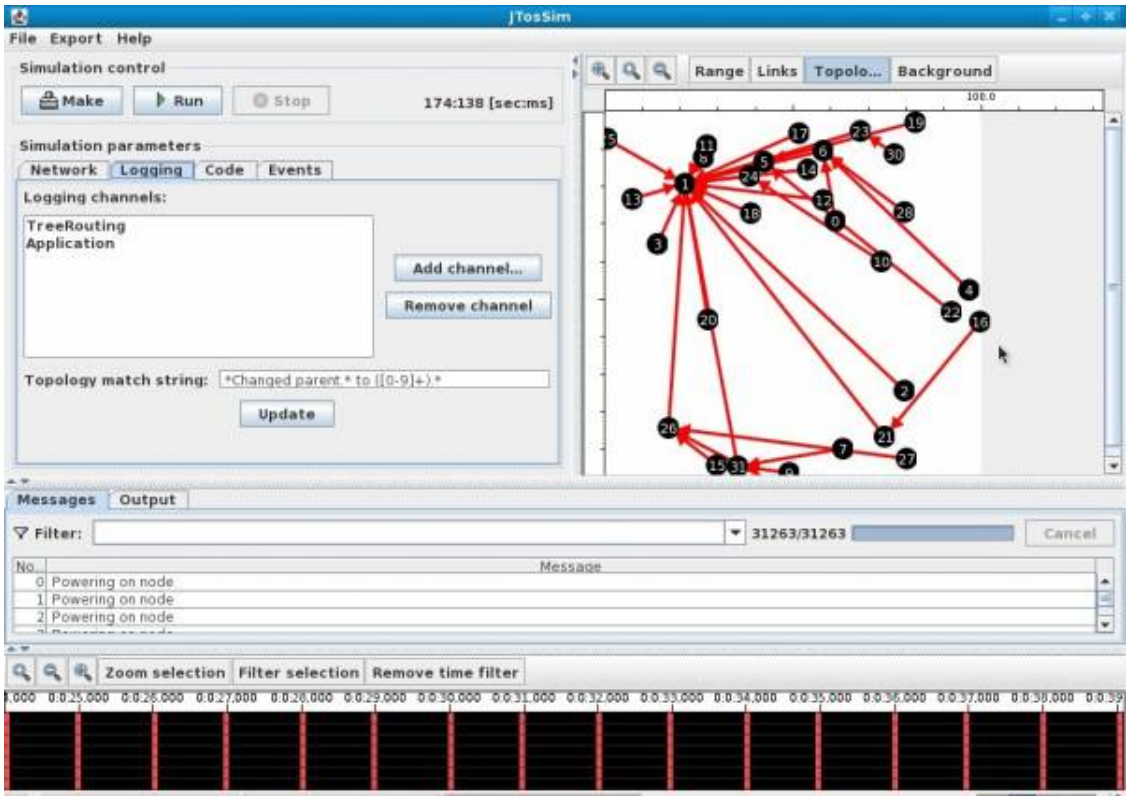
❖ En cliquant sur un nœud, la portée radio du nœud sélectionné s’affiche uniquement.



Notons que la couleur ”vert” dans la portée radio, représente la zone où la plupart des paquets sont reçus, tandis que la couleur ”jaune” représente la zone dans laquelle les paquets peuvent être reçus. Pour voir le gain pour les autres nœuds, vous pouvez cliquer sur le bouton ”Links” dans la section (2).



❖ Lorsque vous cliquez sur le bouton ”Topology” dans la section (2), la topologie logique du réseau, va apparaitre (en couleur rouge) au cours de la simulation.



BIBLIOGRAPHIE

[1] J. N. Al-Karaki and A. E. Kamal, "Routing techniques in wireless sensor networks : A survey". IEEE Wireless Communication, Vol. 11, No. 6, December 2004.

[2] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks". Proceedings of the IEEE Hawaii International Conference on System Sciences,2000.

[3] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks :attacks and countermeasures".in IEEE SNPA,pp. 113-127, May 2003.

[4] F. Mezrag et M. Benmohammed, "Sécurité du Routage Hiérarchique Basé sur les Clusters dans les Réseaux de Capteurs sans Fil". Proceedings of the first International Symposium on Informatics and its Applications (ISIA2014), M'Sila, 2014.

[5] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.l. Cayirci, "A survey on sensor networks". IEEE Communications Magazine, pp. 102-114, August 2002.

[6] C. Duran-Faundez, "Transmission d'images sur les réseaux de capteurs sans fil sous la contrainte de l'énergie". Thèse de doctorat, Université Henri Poincaré, Centre de recherche en automatique de Nancy, Juin 2009.

[7] Z. Wassim, "Quelques propositions de solutions pour la sécurité des réseaux de capteurs sans fil". Thèse de doctorat, Institut national des sciences appliquées de Lyon, Octobre 2010.

[8] K. Kifayat, M. Merabti, Qi. Shi and D. Llewellyn-Jones. "Security in Wireless Sensor Networks". Handbook of Information and Communication Security , pp. 513-552, Springer 2010.

[9] K. Beydoun, "Conception d'un protocole de routage hiérarchique pour les réseaux de capteurs". Thèse de Doctorat, Université de Franche-Comté, 16 décembre 2009.

[10] R. Kacimi, "Techniques de conservation d'énergie pour les réseaux de capteurs sans fil". Thèse de Doctorat, Université de Toulouse. 28 Septembre 2009.

[11] Ian F. Akyildiz, Mehmet Can Vuran. "Wireless Sensor Networks". John Wiley & Sons Ltd, 2010.

[12] A. Makhoul, " Réseaux de capteurs : localisation, couverture et fusion de données". Thèse de doctorat, Université de Franche-Comté, 2008.

[13] G.J.Pottie and W.J.Kaiser, "Wireless Integrated Network Sensors". ACM, vol. 43, no. 5, pp. 51-58, May 2000.

[14] L. Khelladi et N. Badache, "Les réseaux de capteurs : état de l'art". Rapport de Recherche, Faculté électronique et informatique Bab Ezzouar-Algérie, Février 2004.

[15] B. Krishnamachari, D. Estrin and S. Wicker, "Modelling Data-Centric Routing in Wireless Sensor Networks". University of Southern California, Computer Engineering Technical Report CENG 02-14, 2002.

[16] L. Khelladi and N. Badache, "Improving Directed Diffusion With Power-Aware Topology Control For Adaptation to High Density", LOCALGOS'08 workshop, in conjunction with The 4th IEEE/ACM International Conference on Distributed Computing In Sensor SystemS (DCOSS 2008), Greece, 2008.

[17] M. Aissani, "Optimisation du routage dans les réseaux de capteurs pour les applications temps-réel", Thèse de doctorat, USTHB, 13 mars 2011.

[18] L.T. Nguyen, X. Defago, R. Beuran and Y. Shinoda, "An Energy-Efficient Routing Scheme for Mobile Wireless Sensor Networks" Proc. of the 5th IEEE Int'l Symposium on Wireless Communication Systems, pp. 568-572, Reykjavik, Iceland, October 21-24, 2008.

[19] Y. Xu, J. Heidemann and D. Estrin, "Geography-informed Energy Conservation for Ad-hoc Routing", In Proceedings of the 7th annual international conference on Mobile computing and networking, pp. 70-84, ACM, 2001.

[20] C. Intanagonwiwat, R. Govindan and D. Estrin, "Directed diffusion : A scalable and robust communication paradigm for sensor networks", Proceedings of the 6th annual international conference on Mobile computing and networking, pp. 56-67, ACM, 2000.

[21] W. Heizelman, J. Kulik and H. Balakrishnan, "Adaptive protocols for information dissemination in wireless sensor networks", Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, pp. 174-185, 1999.

[22] A. Manjeshwar and D.P. Agrawal, " TEEN : A routing protocol for enhanced efficiency in wireless sensor networks", Proceedings of the International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing,IEEE Computer Society, 2001.

[23] A. Manjeshwar and D.P. Agarwal, "APTEEN : A hybrid protocol for efficient routing and comprehensive information retrieval in wireless sensor networks", the 16th International Parallel and Distributed Processing Symposium, IEEE Computer Society, 2002.

[24] S. Lindsey and C.S. Raghavendra, "PEGASIS : Power efficient gathering in sensor information systems", Proceedings of the IEEE Aerospace Conference, pp. 1125-1130, 2002.

[25] K. Akkaya and M. Younis, "A survey on routing protocols for wireless sensor networks", Ad hoc Networks, pp. 325-349, 2005.

[26] Y. Yu, R. Govindan and D. Estrin, "Geographical and energy-aware routing : A recursive data dissemination protocol for wireless sensor networks", Technical Report, University of California, Los Angeles, May 2001.

[27] K. Sohrabi J. Gao, V. Ailawadhi and G.J. Pottie, "Protocols for self-organization of a wireless sensor network", IEEE Personal Communications, pp. 16–27, October 2000.

[28] V. Loscrì, G. Morabito, and S. Marano, "A two-levels hierarchy for low-energy adaptative clustering hierarchy (tl-leach)", In Proc. VTC2005, pp. 1809–1813, Dallas (USA), September 2005.

[29] M.Ye, C.Li, G.Chen and J.Wu, "EECS : An Energy Efficient Clustering Scheme in Wireless Sensor Networks", 24th IEEE International Performance Computing and Communications Conference (IPCCC05), pp. 535-540, April 2005.

[30] O. Younis and S. Fahmy, "HEED : A Hybrid Energy-Efficient Distributed Clustering Approach for Ad Hoc Sensor Networks", IEEE Transactions on Mobile Computing, vol. 3, no. 4, pp. 366-379, 2004.

[31] D.W. Carman, P.S. Kruus, and B.J. Matt, "Constraints and approaches for distributed sensor network security", Technical Report, 2000.

[32] J. Walters, Z. Liang, W. Shi and V. Chaudhary, "Wireless Sensor Network Security : A Survey", CRC Press, 2006.

[33] J. Sen, "A survey on wireless sensor network security", International Journal of Communication Networks and Information Security (IJCNIS), vol. 1, No. 2, pp. 55–78, August 2009.

[34] D. Djenouri, L. Khelladi and N. Badache, "A survey of security issues in mobile ad hoc and sensor networks", IEEE Communications Surveys and Tutorials ,Vol. 7 , pp. 2-28, 2005.

[35] A. Kellner, O. Alfandi, and D. Hogrefe, "A Survey on Measures for Secure Routing in Wireless Sensor Networks", International Journal of Sensor Networks and Data Communications Vol. 1, pp. 1-17, 2012

[36] K.Benahmed, "Surveillance Distribuée pour la sécurité d'un réseau de capteur sans fil", Thèse de Doctorat, Université d'Oran, 2011.

[37] D. Martins and H. Guyennet, "Wireless Sensor Network Attacks and Security Mechanisms : A Short Survey", Proceedings of the 13th International Conference on Network-Based Information Systems, pp.313-320, September 14-16, 2010.

[38] J. R. Douceur, "The sybil attack", In 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), 2002.

[39] J. Newsome, E. Shi, D. Song and A. Perrig, "The sybil attack in sensor networks : analysis and defenses", Proceedings of the 3rd international symposium on Information processing in sensor networks, California, USA, 2004.

[40] Y. Hu, A. Perrig and D. B. Johnson, "Packet leashes : a defense against wormhole attacks in wireless networks", in IEEE Annual Conference on Computer Communications (INFOCOM), pp. 1976-1986, 2003.

[41] G. ZAÏBI. "Sécurisation par dynamiques chaotiques des réseaux locaux sans fil au niveau de la couche MAC", Thèse de doctorat, Université de Toulouse 2, Décembre 2012.

[42] K. Piotrowski, P. Langendoerfer, and S. Peter, "How public key cryptography influences wireless sensor node lifetime", in SASN '06 : Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks, pages 169-L76, New York, NY, USA, 2006.

[43] A. Wander, N. Gura, H. Eberle, V. Gupta, and S. Chang, "Energy analysis for public-key cryptography for wireless sensor networks", Proc. IEEE PerCom, 2005.

[44] TinyECC version 2.0 (02/03/2011), "A Configurable Library for Elliptic Curve Cryptography in Wireless Sensor Networks". [En ligne]. Disponible sur : http ://discovery.csc.ncsu.edu/software/TinyECC/. [Consulté le : 07-Novembre-2013].

[45] D. J. Malan, M. Welsh, and M. D. Smith, "A Public-Key Infrastructure for Key Distribution in TinyOS based on Elliptic Curve Cryptography", in 1st IEEE IEEE International Conference on Sensor and Ad Hoc Communications and Networks, Santa Clara, CA, October 2004.

[46] H. Wang, B. Sheng and Q. Li, "Elliptic curve cryptography-based access control in sensor networks", International Journal of Security and Networks (IJSN), Special Issue on Security Issues on Sensor Networks, pp. 127-137, 2006.

[47] V. Miller, " Use of Elliptic Curves in Cryptography ", CRYPTO '85 Proceedings, vol. 218, H. Williams, pp. 417-426, 1986.

[48] N. Koblitz, "Elliptic curve cryptosystems", mathematics of computation, vol. 48, pp. 203–209, 1987.

[49] M. Lu, "Study on Secret Key Management Project of WSN Based on ECC". Journal of networks, Vol. 7, NO. 4, pp. 652-659, April 2012.

[50] N. Koblitz, A. Menezes and S. Vanstone, " The State of Elliptic Curve Cryptography", Designs, Codes and Cryptography, vol. 19, pp. 173-193, 2000.

[51] D. Eastlake and P. Jones, "US Secure Hash Algorithm 1 (SHA1)". [En ligne]. Disponible sur : http ://www.hjp.at/doc/rfc/rfc3174.html. [Consulté le : 27-juillet-2013].

[52] R. Rivest, "The MD5 Message-Digest Algorithm". [En ligne]. Disponible sur : https ://-tools.ietf.org/html/rfc1321. [Consulté le : 27-juillet-2013].

[53] O. Mikle, " Practical attacks on digital signatures using MD5 message digest", Cryprology ePrint Archive, report 2004/356. 2004.

[54] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions", in Advances in Cryptology – EUROCRYPT 2005, vol. 3494, pp. 17–36. Springer, Heidelberg, 2005.

[55] K. Zhang, C. Wang, and C. Wang, "A secure routing protocol for cluster-based wireless sensor networks using group key management", In Proc. 4th IEEE International conference on Wireless Communications, Networking and Mobile Computing (WiCOM'08), pp. 1–5, October 2008.

[56] A. C. Ferreira, M. A. Vilaça, L. B. Oliveira, E. Habib,H. C. Wong, and A. A. Loureiro, "On the security of cluster-based communication protocols for wireless sensor networks", In Proc. 4th IEEE International Conference on Networking (ICN'05), volume 3420 of Lecture Notes in Computer Science, pages 449–458, 2005.

[57] A. Perrig, R. Szewczyk, V. Wen, D. Cullar and J. D. Tygar, "SPINS : Security protocols for sensor networks", Proceedings of the 7th annual international conference on Mobile computing and networking, pp. 189-199, ACM, 2001.

[58] L. B. Oliveira, A. Ferreira, M. A. Vilaça, H. C. Wong,M. Bern, R. Dahab and A. A. F. Loureiro, "SecLEACH-on the security of clustered sensor networks", Signal Processing, vol. 87, no. 12, pp. 2882–2895, December 2007.

[59] L. Eschenauer and V. Gligor, "A Key Management Scheme for Distributed Sensor Networks", Proceeding of the 9th ACM conference on Computer and Communications Security, November 2002.

[60] R. Srinath, Reddy, A. Vasudev and R. Srinvasan, "AC : Cluster Based Secure Routing Protocol for WSN. " IEEE, Third International Conference in Networking and Services (ICNS'07), 2007.

[61] Suresha and Nalini.N, "Evaluation of Performance of Ciphers for Routing Protocols in Distributed Sensor Networks", International Journal of Data & Network Security Volume 1, No.2, October 2012.

[62] H. Chan, A. Perrig, and D. Song, "Random Key Predistribution Schemes for Sensor Networks", In Proc. of the IEEE Security and Privacy Symposim , May 2003.

[63] TinyOS Alliance, " TinyOS". [En ligne]. Disponible sur : http ://www.tinyos.net/. [Consulté le : 20-Décembre-2013].

[64] "Université de Berkeley". [En ligne]. Disponible sur : http ://www.berkeley.edu/. [Consulté le : 20-Décembre-2013].

[65] UC Berkeley, "nesC : A Programming Language for Deeply Networked Systems". [En ligne]. Disponible sur : http ://nescc.sourceforge.net/. [Consulté le : 20-Décembre-2013].

[66] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM : Accurate and Scalable Simulation of Entire TinyOS Applications", the 1st international conference on Embedded networked sensor systems, Los Angeles, California, USA, pp. 126–137, 2003.

[67] L. HyungJune, A. Cerpa and P. Levis, "Improving Wireless Simulation Through Noise Modeling". In IPSN07 : international conference on Information Processing in Sensor Networks. 2007.

[68] "PowerTOSSIM-Z : Realistic Energy Model in TOSSIM for the MicaZ Mote". [En ligne]. Disponible sur : https ://www.cs.tcd.ie/∼carbajrs/powertossimz/. [Consulté le : 20-Décembre-2013].

[69] S. Dawson-Haggerty, O. Gnawali, D. Gay, P. Levis, R. Musaloiu-e, K. Klues, and J. Regehr. "TinyOS 2.1", San Francisco, IPSN 2009.

[70] EPFL, "JTOSSIM". [En ligne]. Disponible sur : http ://arni.epfl.ch/software/. [Consulté le : 20-Décembre-2013].

[71] M. Zuniga, "Building a Network Topology for TOSSIM". [En ligne]. Disponible sur :http//www.tinyos.net/tinyos-2.x/doc/html/tutorial/usc-topologies.html. [Consulté le : 10-Janvier-2014].