Université de Montréal

Détection de la retransmission sélective sur les réseaux de capteurs

Par

Edgard Haddad

Département d'informatique et de recherche opérationnelle

Faculté des Arts et des Sciences

Mémoire présenté à la Faculté des Arts et des Sciences
en vue de l'obtention du grade de Maître es sciences (M.Sc.)
en informatique

Avril 2011

© Edgard Haddad 2011

Université de Montréal

Faculté des Arts et des Sciences

Ce mémoire intitulé :

Détection de la retransmission sélective sur les réseaux de capteurs

Présenté par :

Edgard Haddad

a été évalué par un jury composé des personnes suivantes :

Pierre McKenzie, président-rapporteur

Louis Salvail, directeur de recherche

Gilles Brassard, membre du jury

RÉSUMÉ

L'attaque de retransmission sélective est une menace sérieuse dans les réseaux de capteurs sans fil (*WSN*), en particulier dans les systèmes de surveillance. Les nœuds peuvent supprimer de manière malicieuse certains paquets de données sensibles, ce qui risque de détruire la valeur des données assemblées dans le réseau et de diminuer la disponibilité des services des capteurs. Nous présentons un système de sécurité léger basé sur l'envoi de faux rapports pour identifier les attaques de retransmission sélective après avoir montré les inconvénients des systèmes existants. Le grand avantage de notre approche est que la station de base attend une séquence de faux paquets à un moment précis sans avoir communiqué avec les nœuds du réseau. Par conséquent, elle sera capable de détecter une perte de paquets.

L'analyse théorique montre que le système proposé peut identifier ce type d'attaque et peut alors améliorer la robustesse du réseau dans des conditions d'un bon compromis entre la fiabilité de la sécurité et le coût de transmission.

Notre système peut atteindre un taux de réussite élevé d'identification face à un grand nombre de nœuds malicieux, tandis que le coût de transmission peut être contrôlé dans des limites raisonnables.

Mots-clefs : Sécurité des réseaux sans fil, attaque de retransmission sélective, protocoles de routage

ABSTRACT

The selective forwarding attack is a serious threat in wireless sensor networks (*WSN*), especially in surveillance systems. Nodes can maliciously delete some sensitive data packets, which could destroy the value of the data assembled in the network and reduce its sensors availability. After describing the drawbacks of the existing systems in this thesis, we will present a lightweight security system based on sending fake reports used to identify selective forwarding attacks. The great advantage in our approach is that the base station expects a number of packets at a specific time. Therefore, it will be able to detect missing or delayed packets.

Theoretical analysis shows that the proposed system can identify this type of attack, which will improve the robustness of the network under conditions of a good tradeoff between the security, reliability and communication overhead.

Our system can achieve a high ratio of identification when facing a large number of malicious nodes, while the communication overhead can be controlled within reasonable bounds.

Keywords: Selective forwarding attack, *WSN* security, *WSN* routing protocols

TABLES DES MATIÈRES

LISTE DES FIGURES

Je dédie ce mémoire à ma famille.

REMERCIEMENTS

Je tiens à remercier toutes les personnes qui ont participé de près et de loin à la bonne réalisation de ce travail, en particulier Monsieur Louis Salvail, professeur à l'Université de Montréal et directeur de mémoire pour son soutien, ses nombreux conseils et sa patience tout au long de cette maîtrise.

CHAPITRE 1. Introduction

Les réseaux de capteurs sans fil (*WSN*) [26, 42, 48,49] naissent d'une évolution significative de la technologie de fabrication du matériel, combinée avec le développement des algorithmes puissants permettant ainsi de former des réseaux composés de nombreux petits capteurs à faible coût tout en utilisant les communications sans fil. Ce nouveau genre de réseaux a attiré l'attention de l'industrie de l'informatique et des départements de recherche universitaires en raison de la popularité de nombreuses applications qu'elles peuvent effectuer, telles que la surveillance de l'environnement, les soins médicaux, la gestion des appareils électroménagers, la surveillance des champs de bataille et les scénarios de sécurité intérieure.
Un *WSN* [26, 42, 48,49] est donc un vaste réseau de nœuds de capteurs de ressources limitées et de fonctions prédéfinies, tel que la détection des changements (par exemple : température, humidité, trafic) et leur traitement. Le faible coût de ces capteurs et leur petite taille fournissent un moyen de déploiement pratique dans une variété de zones [43] (par exemple: champs de bataille, autoroute) dans le but d'exécuter des tâches militaires (par exemple: surveillance des tanks) ou civiles (par exemple : surveillance du trafic). Par contre, ces capteurs souffrent d'un manque de puissance d'énergie et de stockage de données, ce qui rend considérablement difficile la mise en place de sécurité.

Dans ce premier chapitre, nous vous présentons les différents composants du réseau de capteurs sans fil, les défis et les exigences de sécurité, les attaques possibles, le routage des données, l'établissement des clefs cryptographiques et l'authentification des données.

1.1. Architecture du nœud de capteurs sans fil

L'architecture typique des nœuds de capteurs est représentée dans la figure 1.1 :

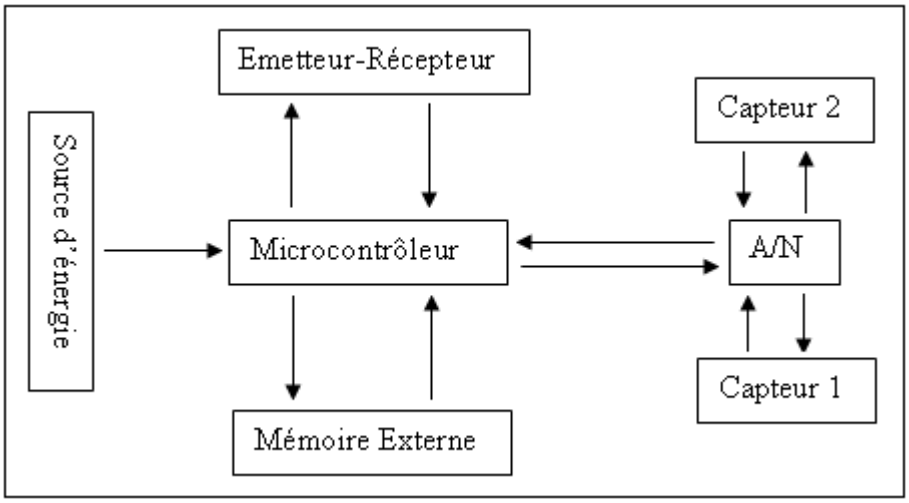

Figure 1.1. Architecture des nœuds de capteurs sans fil.

Les composantes essentielles d'un nœud de capteurs sans fil sont un microcontrôleur, une mémoire externe, une source d'énergie, un émetteur-récepteur et un ou plusieurs capteurs. Le contrôleur effectue des tâches, gère les données et contrôle la fonctionnalité d'autres composantes dans le nœud capteur. L'émetteur-récepteur utilise librement les bandes ISM (industriel, scientifique et médical) de fréquences 173, 433, 868 et 915 MHz et 2,4 GHz, qui ne sont pas soumises à des réglementations nationales, pour émettre et recevoir les messages. Le type de mémoire externe utilisée est la mémoire *Flash* [24,44] en raison de son coût de fabrication et sa capacité de stockage. Deux catégories de mémoires sont implémentées dans la mémoire externe ; 1) La mémoire utilisateur sert à stocker des données des applications.2) La mémoire programme utilisée pour la programmation de l'appareil.

Le nœud de capteurs consomme de l'énergie pour la détection, la communication et le traitement des données. Cependant, la consommation d'énergie pour les tâches de communication est la plus importante. Le coût énergétique de la transmission de 1 Ko sur une distance de 100 m est approximativement le même que celui utilisé pour l'exécution de 3 millions d'instructions de calcul selon l'étude dans [25]. La source principale d'énergie dans le nœud de capteurs est l'utilisation des batteries ou des condensateurs (rechargeables ou non-rechargeables). L'énergie solaire, la différence de température et les vibrations peuvent être des sources additionnelles d'énergie.

Les capteurs sont des dispositifs matériels qui produisent un signal identifiant un changement d'un état physique comme le changement de température ou de pression. Ils mesurent les données

physiques du paramètre à surveiller. Le signal analogique continu produit par les capteurs est numérisé par un convertisseur analogique-numérique (A/N) [45] et envoyé au contrôleur pour un traitement ultérieur.

Un capteur effectif doit donc être de petite taille, autonome, doit consommer très peu d'énergie, doit être adapté à l'environnement et fonctionner à haute densité volumétrique et sans surveillance. Comme les nœuds de capteurs sans fil sont généralement des appareils électroniques très petits, ils ne peuvent pas être équipés d'une source d'alimentation de plus de 0,5 à 2 ampères-heures et 1.2 à 3.7 volts.

Figure 1.2. Capteur omnidirectionnel.

Figure 1.3. Capteur unidirectionnel.

L'ensemble des travaux théoriques effectués sur les réseaux de capteurs sans fil utilise des capteurs omnidirectionnels (figure 1.2) capables d'envoyer et recevoir les signaux dans toutes les directions et qui n'ont aucune notion de direction impliquée dans leurs mesures, à l'instar des capteurs unidirectionnels (figure 1.3) comme ceux retrouvé dans les caméras.

Chaque nœud de capteurs possède une zone de couverture pour laquelle il peut générer des rapports (ou messages) de façon fiable et précise sur l'événement particulier qu'il observe. Dépendamment de l'application, la densité spatiale des nœuds de capteurs dans certains cas peut être assez élevée (20 nœuds par mètre cube).

1.2. Architecture du réseau de capteurs sans fil

L'architecture typique des réseaux capteurs sans fil est illustrée dans la figure 1.4 :

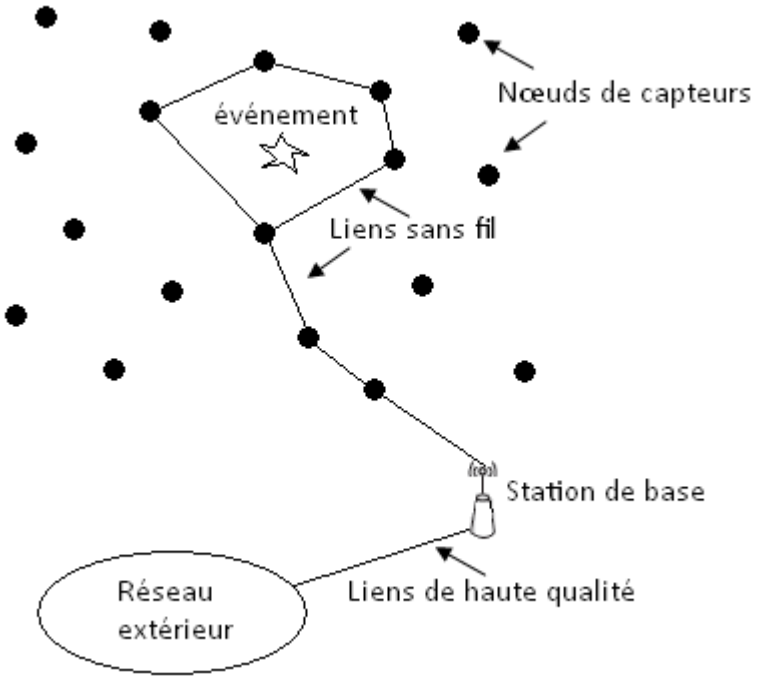

Figure 1.4. Architecture du réseau de capteurs sans fil.

Un réseau de capteurs sans fil [26, 42, 48,49] est de nature statique quand la station de base et les nœuds de capteurs sont fixes et immobiles. Les nœuds de capteurs sont fixes la plupart du temps, alors que les nœuds mobiles peuvent être déployés selon les besoins de l'application. Une ou plusieurs stations de base (BS) sont déployées avec le réseau. Une station de base peut être statique ou mobile. Les nœuds de capteurs continuent à surveiller la zone du réseau après avoir été déployé. Quand un événement se produit, tel que le passage des tanks de l'ennemi dans la zone de surveillance, l'un des capteurs entourant cette zone génère un rapport de détection afin de le transmettre à la station de base à partir des liaisons multiples sans fil, d'où vient la nomination *multi-hop* pour ces réseaux. Ces liaisons multiples (*multi-hop*) sont l'ensemble de capteurs, appelés nœuds intermédiaires, acheminant le message entre la source et la destination. Une collaboration peut être réalisée si plusieurs nœuds détectent le même événement. Dans ce cas, l'un d'eux génère un rapport final après avoir collaboré avec les autres nœuds et l'envoie à la station de

base. Cette dernière peut traiter le rapport et le retransmettre ensuite, soit par des liens de haute qualité avec ou sans fil, au monde extérieur pour un traitement ultérieur.
L'autorité du réseau (les administrateurs) peut envoyer des commandes ou des requêtes à la station de base, qui propage ces commandes ou ces requêtes dans le réseau. Ainsi, une station de base, qui possède généralement plus de ressources que les capteurs, agit comme une passerelle entre les réseaux de capteurs et le monde extérieur.

1.3. Défis de sécurité dans les WSN

Les *WSN* ont de nombreuses caractéristiques qui les rendent très vulnérables à des attaques malicieuses [28,50] dans les différents environnements. Par exemple, les attaques de déni de service, l'injection et la modification des données, la retransmission sélective, etc. [28,46] peuvent être dommageable pour le *WSN*, ce qui rend difficile d'appliquer directement les approches de sécurité existantes sur des réseaux de capteurs sans fil. Par conséquent, afin d'élaborer des mécanismes de sécurité utiles tout en empruntant des idées de techniques de sécurité actuelles, il est nécessaire de comprendre les caractéristiques suivantes :

1- La contrainte des ressources.
2- La fiabilité des communications.
3- Le fonctionnement autonome.

1.3.1. La contrainte des ressources [51]

Toutes les approches de sécurité nécessitent 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 capteur. Toutefois, ces ressources sont très limitées dans ces petits appareils de capteurs sans fil en raison de la prise en considération des coûts de fabrication :

• la mémoire limitée et l'espace de stockage

Un capteur est un petit appareil avec seulement une petite quantité de mémoire et d'espace de stockage pour le code. Afin de construire un mécanisme de sécurité efficace, il est nécessaire de limiter la taille du code de l'algorithme de sécurité. En moyenne, la plupart des nœuds de capteurs sans fil possèdent un microcontrôleur de 8-16 bits, avec seulement 10-64K de mémoire programme et 512K-4Mo capacité de stockage flash [24,44]. Avec une telle contrainte, le logiciel du capteur doit également être assez

petit. L'espace total du code *TinyOS*, l'un des systèmes d'exploitation populaires pour les capteurs sans fil, est d'environ 4K [25]. Par conséquent, le code relié à la sécurité doit également être de petite taille.

• Contrainte de la puissance [47]

L'énergie utilisée est le plus grand obstacle à la capacité des capteurs sans fil. Nous supposons que, une fois que les nœuds de capteurs sans fil sont déployés dans un réseau, ils ne peuvent plus être facilement remplacés ou rechargés. Par conséquent, l'énergie initiale des capteurs doit être conservée pour prolonger leur durée de vie individuelle et la durée de vie du réseau de capteurs en général. Lors de l'implémentation des protocoles de sécurité dans les capteurs, l'impact énergétique du code de sécurité ajouté doit être considéré.

Lors de l'ajout de sécurité à un nœud de capteur, nous nous intéressons à l'impact de la sécurité sur la durée de vie d'un capteur (par exemple, la durée de vie de sa batterie). La puissance supplémentaire consommée par les nœuds de capteurs en raison de la sécurité est liée au traitement requis pour les fonctions de sécurité (par exemple, le chiffrement, le déchiffrement, la signature de données, la vérification des signatures), l'énergie nécessaire pour transmettre les données relatives à la sécurité (par exemple, les vecteurs d'initialisation nécessaires pour le chiffrement / déchiffrement) et l'énergie nécessaire pour sauvegarder les paramètres de sécurité d'une manière sûre (par exemple, le stockage de clefs de chiffrement).

Lors de la conception de la plupart des protocoles de sécurité dans les *WSN*, le chiffrement (cryptage) à clef privée (par exemple, *DES* [52], *RC5* [53]) est préférable au chiffrement à clef publique (par exemple, *Diffie-Hellman* [54], *RSA* [55]). Ce dernier est possible, mais consomme beaucoup plus de ressources. Un *WSN* formé de milliers de capteurs nécessite des protocoles de sécurité simples et flexibles. Par contre, construire ce genre de protocoles n'est pas une tâche facile. Un protocole de sécurité plus robuste utilise plus de ressources au niveau des capteurs, ce qui peut conduire à une dégradation de performance au niveau des applications. Dans la plupart des cas, un compromis entre la sécurité et la performance doit être fait. Evidement, les faibles protocoles de sécurité peuvent être plus faciles à attaquer par les adversaires.

1.3.2. La fiabilité des communications

Un canal sans fil est un moyen de communication ouvert qui peut être consulté par toute personne dans la portée du signal. Cependant, cette ouverture démontre un grand avantage, car elle réduit le coût de l'infrastructure. Mais avec une interface radio configurée à la même bande de fréquence, les adversaires peuvent surveiller ou participer à la communication. Ceci pose plusieurs problèmes qui sont expliqués ci-dessous:

• Manque de fiabilité de transfert

Contrairement aux réseaux de capteurs filaires (connectés avec des fils), les canaux sans fil ne sont pas aussi fiables. Ils sont sensibles aux interférences, aux erreurs du canal, à la congestion (requêtes supérieures à la capacité du traitement dans les capteurs) et aux différents objets se déplaçant dans la portée du signal. Ces conditions peuvent être permanentes ou temporaires et peuvent endommager ou laisser tomber des paquets (un rapport demande un ou plusieurs paquets) sur le réseau sans fil. Si un protocole sans fil ne prévoit pas la gestion des erreurs, il peut conduire à une communication incohérente et à la perte des paquets critiques de sécurité (par exemple, une clef de chiffrement), produisant ainsi des nœuds de capteurs incapables de communiquer de façon sécurisée.

• Conflits/Collisions

Même si nous supposons que le canal est fiable, la communication peut encore ne pas être fiable à cause des collisions de paquets dans le canal sans fil. Cela est dû à la nature d'émission dans les réseaux de capteurs sans fil. Si deux capteurs voisins (l'un à la portée du signal de l'autre), tentent d'émettre un paquet chacun en même temps, une collision entre les deux paquets peut se produire et le transfert lui-même peut échouer. Dans un réseau de capteurs de grande densité, cela peut être un problème majeur. Plus de détails sur l'effet de la communication sans fil peuvent être trouvés dans [26].

• Temps de latence

La synchronisation entre les nœuds de capteurs sera une tâche difficile et affectera la déclaration des événements et la distribution de clefs cryptographiques [27]. Ceci est d'autant plus vrai lorsque le routage à partir des liaisons multiples (aussi appelé *multi-hop*), la congestion du réseau et le traitement des données au niveau des nœuds de capteurs augmentent le délai normal de transmission dans le réseau.

1.3.3. Le fonctionnement autonome

Un des principaux avantages des réseaux de capteurs est la possibilité de placer les nœuds de capteurs dans un environnement sans aucune surveillance. Ceci peut produire des faiblesses de sécurité pour le réseau si les nœuds de capteurs sont déposés dans des environnements difficiles ou d'une manière non garantie. L'absence de la protection physique peut permettre à plusieurs types d'attaques.

- L'exposition à l'environnement / Attaques physiques

Les nœuds de capteurs sans fil peuvent être déployés dans un environnement ouvert à des agressions physiques. Par exemple, les nœuds de capteurs dans l'océan peuvent être mangés par les poissons ou emportés pendant les orages. Étant donné que ces nœuds sont déployés dans des environnements ouverts, ils peuvent également être attaqués ou volés par des adversaires.

- Gestion à distance

Un des avantages des réseaux de capteurs sans fil est leur capacité d'être gérés à distance. Ceci permet aux nœuds de capteurs d'être placés dans des environnements dangereux ou inaccessibles. Par contre la gestion à distance exige la présence des mécanismes de sécurité pour protéger le réseau et les informations transmises au centre de contrôle. La sécurité est également nécessaire pour protéger les serveurs du centre de contrôle étant donné que le réseau peut être utilisé par des adversaires afin d'accéder aux systèmes du serveur principal.

- Infrastructure non-fixe

Les réseaux de capteurs peuvent s'organiser automatiquement pour former un réseau distribué. Cela fournit un réseau de communication robuste et dynamique pour envoyer des informations aux serveurs dans le monde extérieur. Par contre, si un réseau est

mal conçu, l'organisation de ce réseau devient difficile, inefficace et fragile. La communication entre les nœuds de capteurs nécessite d'intégrer des fonctionnalités de sécurité contre les attaques possibles.

1.4. Exigences de sécurité

Les conditions difficiles des environnements de déploiement et l'existence des menaces nécessitent un traitement de problèmes de sécurité durant la conception des protocoles des réseaux de capteurs sans fil [42, 51]. En général, les services de sécurité suivants doivent être fournis:

• *La confidentialité* est un service de sécurité de base pour maintenir la confidentialité des données importantes transmises entre les nœuds de capteurs. D'habitude, les informations critiques d'un paquet sont chiffrées avant que le paquet ne soit transmis par le nœud expéditeur. Par la suite, ces informations sont déchiffrées au niveau du nœud récepteur.
Sans les clefs de déchiffrement correspondantes, les attaquants (les individus qui lancent les attaques) sont incapables d'accéder à l'information critique. Dépendamment des besoins des applications, une seule partie du paquet des données est chiffrée, ou bien tout le paquet est chiffré.

• *L'authenticité* est essentielle pour la vérification de l'identité des nœuds pendant les communications. Chaque nœud doit vérifier si le message reçu provient d'un nœud authentique du réseau. Sans authentification, les attaquants peuvent facilement frauder des identités afin de diffuser de fausses informations dans le réseau de capteurs sans fil. Habituellement, un code d'authentification de message (*MAC*) peut être utilisé pour authentifier l'origine d'un message. Ceci nécessite cependant un partage préalable des clefs secrètes.

• *L'intégrité* devrait être fournie pour garantir que les messages transmis ne soient pas modifiés par des adversaires. Les attaquants peuvent introduire des interférences sur quelques morceaux d'un paquet transmis afin de changer sa destination. Un nœud malicieux peut également modifier des données importantes des paquets qu'il reçoit avant de les transmettre. Un *MAC* peut protéger

les paquets contre toutes les modifications faites par un adversaire n'ayant pas la clef secrète pour produire le code d'authentification de message.

• *La disponibilité* est une composante importante dans les réseaux de capteurs sans fil pour fournir des services quand ils sont requis. Cependant, les adversaires peuvent lancer des attaques de déni de service (Dos) [29] ciblant la performance du réseau ou même détruisant l'ensemble du réseau. Ceci est réalisé en introduisant du bruit sur les mêmes bandes radio utilisées par le réseau, ou en épuisant les ressources des nœuds grâce à d'autres attaques.

Les quatre prochains sous chapitres représentent un survol théorique pour définir les termes utilisés dans le reste du mémoire. Plusieurs adversaires profitent des failles de sécurité présentes dans les réseaux de capteurs sans fil pour lancer des attaques contre ces réseaux. Ces attaques peuvent être efficaces et dangereuses dans plusieurs scenarios si des mesures de sécurité adéquates ne sont pas prises. Les prochains sous chapitres représentent aussi les protocoles de routage ainsi que les protocoles d'établissement des clefs cryptographiques pour mieux comprendre leur fonctionnement et leur mode d'utilisation. Enfin, les méthodes d'authentification des données sont comprises dans les approches proposées afin de confirmer l'identité de l'expéditeur des données dans le réseau.

1.5. Attaques

Les attaques [28] contre les réseaux de capteurs peuvent être classées selon leur nature active ou passive, ou bien selon leur origine interne ou externe:

- ***Passive ou active***

Selon le mode de fonctionnement, les attaques peuvent être passives ou actives. Dans une attaque passive, l'objectif de l'attaquant est d'obtenir des informations sans être détecté. Habituellement, l'attaquant reste passif pour écouter le trafic [56]. S'il connaît les protocoles de communication, l'attaquant peut les suivre comme s'il était un nœud de capteurs. En participant au réseau d'une manière passive, l'attaquant recueille un grand volume de données et procède à une analyse de données dans le but d'extraire les informations secrètes. Ces informations extraites peuvent

ensuite servir l'attaquant à des fins malveillantes. En général, l'attaque passive est très difficile à détecter car l'attaquant ne laisse pas beaucoup de traces. Dans une attaque active, l'attaquant exploite les failles de sécurité du protocole utilisé dans le réseau pour lancer des attaques diverses [28]. L'impact des attaques actives est beaucoup plus grave que celui des attaques passives. Par contre, des anomalies peuvent permettre d'identifier les attaques actives parce que l'adversaire est activement impliqué dans les communications du réseau.

- ***Externe ou interne***

En général, un réseau de capteurs sans fil est déployé et géré par une seule autorité. Tous les nœuds du réseau peuvent être considérés comme des entités honnêtes et coopératives, alors que les attaquants sont exclus du réseau et n'ont pas le droit d'accès au réseau. Les attaquants externes peuvent lancer des attaques sans avoir accès au réseau. Parfois, les adversaires ne sont pas intéressés au contenu des données qui circulent dans le réseau. Ils peuvent tout simplement introduire du bruit sur les mêmes bandes radio (*Jamming*) pour perturber les communications entre les nœuds [29]. Si un attaquant utilise des équipements avec des ressources illimitées, il peut toujours bloquer le canal sans fil afin de couper les communications normales et entraîner une perte de paquets. L'impact de la plupart des attaques externes est limité au déni de service.

Si un attaquant obtient l'autorisation d'accéder au réseau, il devient un attaquant interne. Dans ce cas, l'attaquant peut causer des dommages beaucoup plus graves, car il est considéré comme une entité légitime. D'habitude, pour qu'un attaquant puisse lancer des attaques internes, il doit compromettre un nœud légitime et le remplacer par un nœud malicieux capable de contourner le mécanisme de contrôle d'accès au réseau. Prendre contrôle d'un nœud de capteurs (*Node Compromise Attack*) est l'une des attaques les plus préjudiciables à un réseau de capteurs [28]. Parce que les réseaux de capteurs sont habituellement déployés dans un environnement hostile sans surveillance continue, un adversaire peut capturer un nœud de capteurs afin d'extraire tous ses secrets utilisés dans les protocoles de sécurité.

Les secrets exposés donnent à l'adversaire une plus grande capacité à lancer des attaques internes. Ces attaques internes ciblant les protocoles de routage peuvent perturber un réseau de capteurs en utilisant les techniques suivantes [28] :

A. Injection, modification et retransmission des données (Spoofed, altered, or replayed routing information)

L'attaque la plus évidente contre les protocoles de routage est de viser les informations échangées entre les nœuds de capteurs.

Les paquets reçus par un nœud compromis peuvent être retransmis plus tard (*Relay Attack* [3,28]) ou dans une autre zone du réseau afin de créer des incohérences dans le système. Des faux paquets peuvent être injectés (*Spoofing Attack* [3,28]) dans le réseau pour confondre les nœuds de capteurs. Un nœud malicieux peut également modifier les paquets reçus avant de les transmettre vers la destination finale (*Altered and Replayed Packets Attack* [3,28]).

B. Sinkhole

L'objectif de l'adversaire dans cette attaque [28,57] est d'attirer le trafic de données dans une zone de capteurs et de le faire passer par un nœud compromis. En respectant l'algorithme de routage, un nœud compromis fait semblant que la transmission optimale des données passe à travers lui pour attirer l'attention de ses voisins. Cette technique aide l'adversaire à lancer d'autres attaques internes.

C. Sybil

Il s'agit d'une attaque [30, 58] où un nœud malicieux présente plus d'une identité dans le réseau. Cette technique peut dégrader l'efficacité de plusieurs fonctionnalités comme la distribution de stockage de données, la répartition des ressources entre les nœuds, la maintenance de la topologie du réseau et le routage des paquets. Comme l'attaque *Sinkhole*, l'attaque *Sybil* [30,58] peut être combinée avec d'autres attaques pour perturber les systèmes de routages.

D. Wormhole

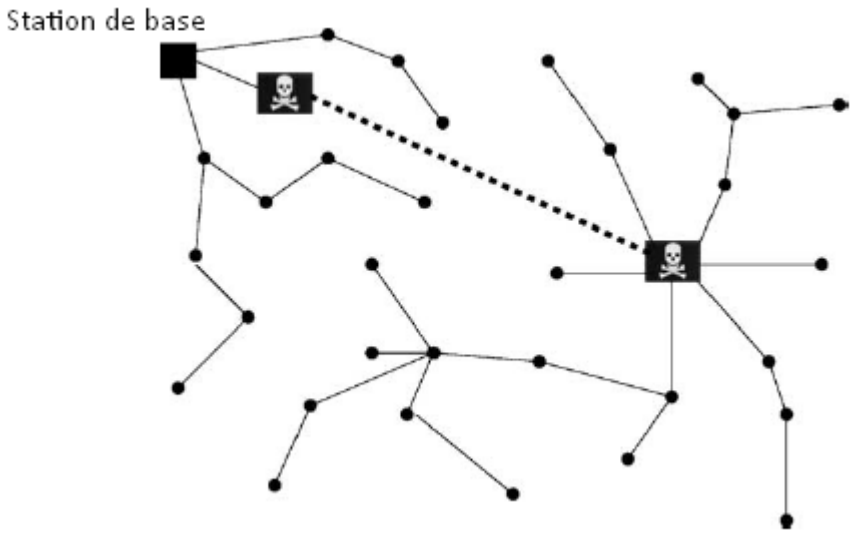

Figure 1.5. L'attaque *Wormhole*.

Une attaque *Wormhole* [28, 57] crée un lien de faible latence entre deux portions du réseau, utilisé par l'adversaire pour l'injection, la modification et la retransmission des données. Une latence dans un réseau de capteurs sans fil est le temps requis par un paquet pour passer d'un capteur à l'autre. Une faible latence signifie un lien de haute efficacité de transmission entre les deux capteurs du réseau. Ce lien peut être un nœud commun entre deux zones différentes du réseau qui sera responsable de transmettre les paquets d'un endroit à l'autre, ou bien ce lien peut être constitué de deux nœuds situés chacun dans la même zone, et capables de communiquer entre eux. L'attaque *Wormhole*, qui est souvent reliée à la technique *Sinkhole,* est réalisée quand un nœud malicieux écoute les paquets passant à travers lui et envoie une copie à un autre nœud malicieux avant de les transmettre vers leur destination finale. Cela peut entraîner une perturbation dans les protocoles de routage et une destruction des ressources des capteurs lorsque la station de base reçoit le même paquet plusieurs fois

.

E. Hello Flood

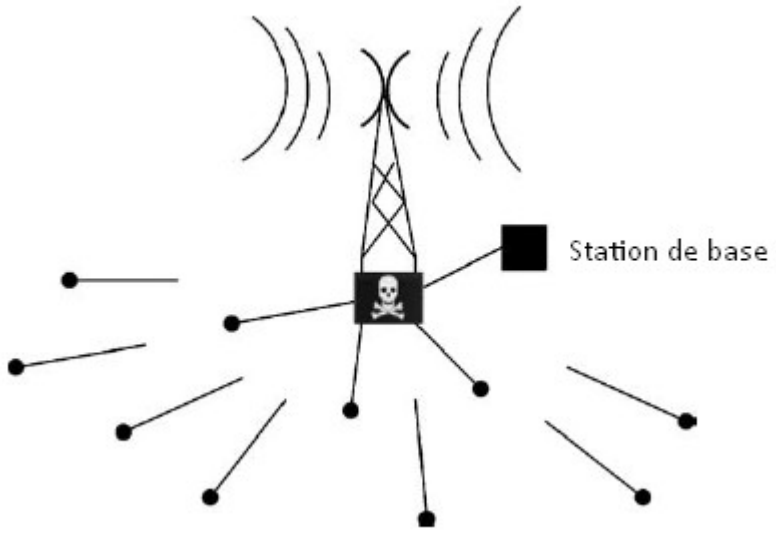

Figure 1.6. L'attaque *Hello Flood* [28].

Plusieurs protocoles de routage nécessitent que les nœuds envoient des paquets, appelés paquets *Hello* [28], pour s'annoncer eux-mêmes à leurs voisins lors du déploiement. Un nœud qui reçoit un tel paquet va supposer qu'il est dans la portée du signal de l'expéditeur, et par la suite déduira que l'expéditeur est son voisin. Cette supposition peut être fausse : Dans la figure 1.6 un adversaire possédant un haut potentiel de ressources (énergie et signal de transmission) est capable d'envoyer des paquets *Hello* à tous les capteurs afin de les convaincre qu'il est un nœud voisin avec le plus court chemin vers la station de base. Quand les capteurs vont envoyer des paquets vers la station de base, ils vont l'envoyer à celui de l'adversaire.

F. Acknowledgment spoofing

Plusieurs protocoles de routage utilisés dans les réseaux de capteurs sans fil nécessitent l'utilisation des paquets *acknowledgment* [3,28]. Ces paquets sont normalement envoyés par le nœud destinataire au nœud expéditeur pour confirmer la réception d'un message donné. Un nœud malicieux peut envoyer des paquets *acknowledgment* à ses voisins expéditeurs afin de leur fournir de fausses informations. Par exemple, faire croire à l'expéditeur que le destinataire est encore fonctionnel tandis qu'il est hors service est une fausse information qui peut causer une perte de paquets et une perte d'énergie de la part de l'expéditeur.

G. Retransmission sélective (Selective forwarding)

Dans l'attaque de retransmission sélective [28], le sujet de ce mémoire, les nœuds malicieux refusent de transmettre certains messages provenant de leurs voisins en s'assurant qu'ils ne seront pas reproduits plus loin.

Une forme simple de cette attaque interne peut être représentée par l'exemple suivant : un nœud malicieux se comporte comme un trou noir en refusant de transmettre tous les paquets qu'il reçoit. Toutefois, un tel attaquant prend le risque que les nœuds voisins le perçoivent comme ayant échoué à retransmettre les paquets et décident de chercher un autre chemin. Une forme plus subtile de cette attaque est lorsqu'un adversaire transmet les paquets d'une manière sélective. C'est-à-dire qu'un adversaire intéressé à supprimer ou modifier les paquets provenant d'un nœud spécifique laisse passer d'une manière fiable le reste du trafic, ce qui limite les soupçons de ses méfaits. Les attaques de retransmission sélective sont généralement plus efficaces lorsque l'attaquant est explicitement inclus sur le chemin d'un flux de données. Toutefois, il est concevable qu'un adversaire écoutant le flux passant par des nœuds voisins puisse être capable d'émuler la retransmission sélective en causant une collision sur chaque paquet transmis.

Un adversaire lançant une attaque de retransmission sélective va probablement suivre le même chemin que le flux de données. Il peut donc efficacement s'installer sur le chemin du flux de données ciblées en se servant d'autres types d'attaques telles que *Sinkhole*.

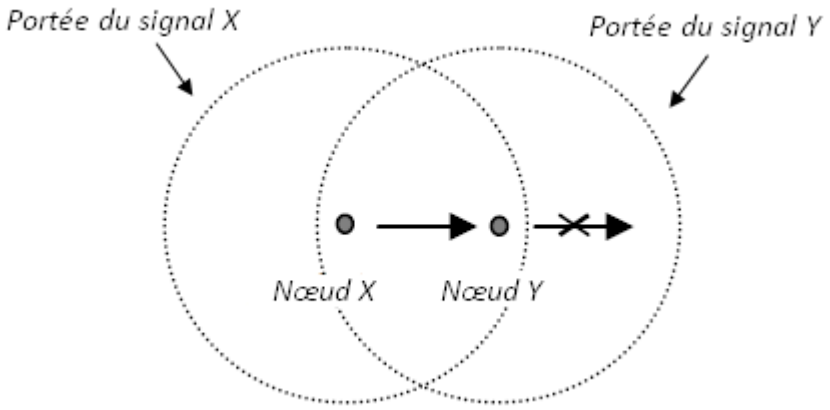

Figure 1.7. L'attaque de *Retransmission Sélective*.

On prend l'exemple de la figure 1.7, en chemin vers la destination (la station de base), le nœud *X* envoie ou transmet un paquet à son nœud voisin *Y*. L'attaque de retransmission sélective peut s'appliquer à partir du nœud malicieux *Y* de plusieurs façons :
Scénario 1 : *Y* détruit le message.
Scénario 2 : *Y* enregistre le message dans son cache et l'envoie plus tard causant un délai de transmission.
Scénario 3 : *Y* retransmet le paquet à un autre nœud malicieux *Z* grâce à une connexion directe de haute qualité.

Dans les trois scénarios, le résultat peut être une perte du message sans détection ou un retard dans la transmission de données. Nous rappelons que dans une attaque de retransmission sélective, une portion des messages est ciblée et non pas tous les messages, ce qui rend la détection de cette attaque difficile. Nous aborderons les travaux existants concernant l'identification de cette attaque et proposerons une nouvelle approche capable d'ajouter plus d'efficacité au niveau des mesures de sécurité dans le chapitre 3 de ce mémoire.

1.6. Routage et transport des données

La transmission et le routage des données sont un service essentiel pour permettre la communication entre les réseaux de capteurs. Malheureusement, les protocoles de routage actuels souffrent encore de nombreuses failles de sécurité [28] et font face à plusieurs attaques efficaces. La plus dangereuse attaque existe quand un adversaire peut compromettre un nœud du réseau. Les protocoles de routage les plus populaires sont les suivants :

1- Directed diffusion [3].
2- PSFQ [2].
3- SEEM [12].

- ***Directed Diffusion*** [3]

Directed Diffusion [3] est un protocole de routage où la transmission de données des nœuds source vers la station de base se fait après que la station de base ait transmis son intérêt pour un

service donné. Les attributs de données qui sont définis dans le schéma de nommage (description de la tâche) représentent les intérêts et les données à transmettre. Chaque application utilise son propre schéma de nommage en fonction de ses besoins.

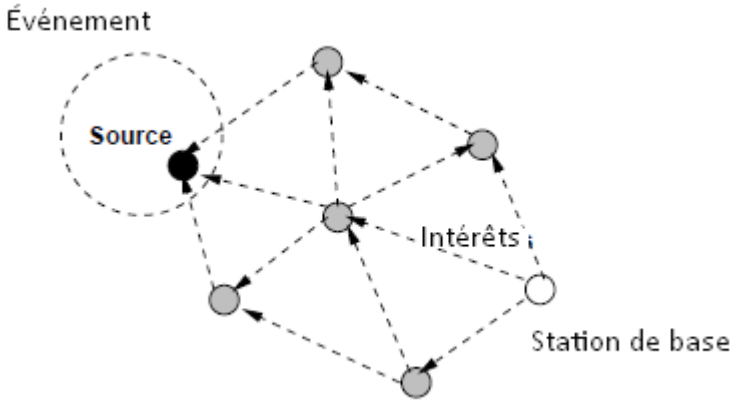

Figure 1.8. Propagation des intérêts [3].

Dans la figure 1.8, la station de base envoie périodiquement son intérêt à tous ses voisins d'abord avec un faible débit (par exemple : 1 événement par seconde). Chaque nœud maintient l'intérêt dans son propre cache. Chaque fois qu'un nœud reçoit un intérêt, il le compare avec ceux présents dans le cache. Si c'est un nouvel intérêt, il va être ajouté au cache, sinon il sera rejeté.

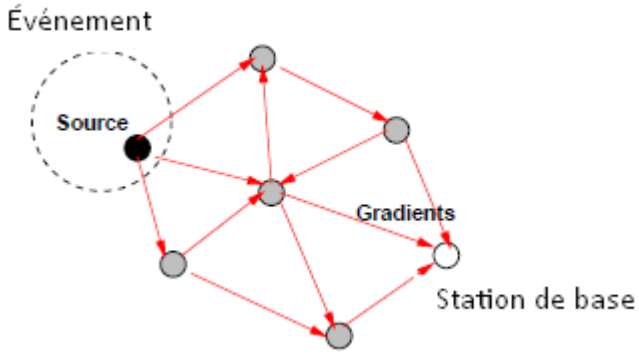

Figure 1.9. Diffusion des gradients.

Les gradients sont ensuite mis en place dans le réseau. Un gradient définit le débit de données et la nouvelle direction d'envoi des données. Les nœuds locaux interagissent alors les uns avec les autres (figure 1.9). Quand un nœud détecte un événement dans son entourage, il le compare avec

les intérêts déjà enregistrés. Ensuite, Le nœud accumule et enregistre les données qui correspondent aux intérêts présents dans le cache.

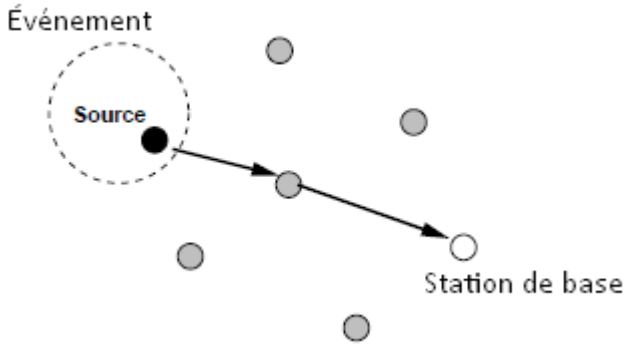

Figure 1.10. Livraison des données à travers le chemin optimal.

Finalement, toutes les données dupliquées sont supprimées et de nouvelles données plus précises sont renvoyées à la station de base par le chemin optimal (moindre délai de transmission et consommation d'énergie).
Chaque nœud agit comme la source ou la destination. L'objectif principal de cette méthode est la conception d'un modèle stable avec un maintient de la communication et l'utilisation de l'énergie plus efficacement, même quand des faiblesses sont présentes dans les nœuds de capteurs.

- ***PSFQ*** [2] " *Pump Slowly, Fetch Quickly* "

Ce protocole de transport est utilisé uniquement pour la diffusion des données dans le but de reprogrammer un nœud de capteur. Pour qu'un nœud soit programmé, il doit avoir tous les morceaux de données (paquets) qui vont être utilisés pour reprogrammer ce dernier. Il est donc extrêmement important que chaque nœud reçoive tous les morceaux. Par exemple, un *WSN* qui surveille la température de l'eau et qui est programmé pour envoyer un rapport quand la température augmente de 5°C peut être reprogrammé pour envoyer le rapport une fois que la température ne varie que de 3°C. Le protocole PSFQ a été créé pour répondre à ce besoin.

PSFQ comporte deux parties

1. *Pump Slowly*
2. *Fetch Quickly*

La partie *Pump Slowly* de ce protocole envoie les paquets lentement. Chaque paquet reçu par un nœud est ensuite diffusé à tous ses voisins jusqu'à ce que tous les nœuds dans le réseau reçoivent cette partie de données envoyées. La partie *Fetch Quickly* de ce protocole est utilisée si un paquet de données est perdu entre deux nœuds voisins afin de récupérer le paquet manquant.

SEEM [12] " *Secure and energy efficient multipath routing protocol* "

Il s'agit d'un protocole de routage. Il assure la sécurité contre les attaques *Sinkhole* (attirer le trafic de données dans une zone de capteurs et de le faire passer par un nœud compromis) tout en équilibrant les niveaux d'énergie de chaque nœud dans un réseau. Le protocole *SEEM* est très efficace car il prend l'avantage de la prédominance de la station de base dans le réseau. Dans *SEEM*, le choix et l'entretien du chemin des paquets sont effectués par la station de base. De plus, la station de base prend en considération l'énergie disponible dans chacun des nœuds de capteurs afin de choisir le meilleur chemin, parmi plusieurs chemins possibles, pour l'envoi des paquets.

SEEM consiste en trois phases : la construction de la topologie, la transmission des données et l'entretien des chemins. La construction de la topologie est la phase de mise en place de la topologie du réseau. La station de base envoie des paquets *ND* (*Network Discovery*) à tous les nœuds de capteurs du réseau. Chaque capteur renvoie à la station de base un paquet contenant son adresse et les adresses de ses voisins. Par la suite, la station de base analyse les paquets et construit la topologie du réseau en associant à chaque nœud une variable *W* (*Weight*). Cette variable *W* correspond à l'énergie totale de chaque nœud divisée par l'énergie nécessaire pour l'envoi ou le reçu d'un paquet.

Pendant la phase de transmission des données, la station de base envoie ses intérêts dans des paquets *DE* (*Data Enquiry*) à tous les nœuds du réseau. Chaque nœud de capteurs satisfaisant ces

intérêts renvoie un paquet *DER (Data Enquiry Reply)* à la station de base. La station de base calcule le chemin qui consomme le minimum d'énergie, et le renvoie au nœud source dans un paquet *RR* (*Route Reply*) à travers le même chemin. Par la suite, le nœud source renvoie à la station de base un paquet *ACK* (*Acknowledge*) confirmant sa réception du paquet *RR*. De même, le nœud source introduit dans le paquet *ACK* le nombre *n* de paquets de données à envoyer. Quand la station de base reçoit le paquet *ACK*, elle attend *n* paquets provenant du nœud source.
Pendant la phase d'entretien des chemins, la station de base calcule périodiquement l'énergie disponible sur les chemins d'envoi des données en diminuant la valeur de *W* associée à chaque nœud de capteurs. Pour équilibrer la consommation d'énergie du réseau, quand la station de base trouve un autre chemin avec une quantité d'énergie disponible plus élevée, elle l'envoie au nœud source.
Par conséquence, *SEEM* prolonge la durée de vie du réseau d'environ 35% par rapport à *Direct diffusion* [3]. La sélection du chemin par la station de base dans *SEEM* est une fonction utile.

Ces approches de routage ou de transports contiennent encore de nombreuses failles de sécurité auxquelles nous tenterons de répondre. Après avoir compris le fonctionnement du transport de l'information dans le réseau de capteurs sans fil, nous devons mentionner les méthodes utilisées pour introduire la sécurité dans les réseaux.

1.7. Etablissement des clefs cryptographiques

L'établissement des clefs cryptographiques lors du déploiement d'un réseau de capteurs sans fil est l'une des premières nécessités pour se prémunir contre plusieurs types d'attaques. Depuis quelques années, les chercheurs ont proposé plusieurs variétés de protocoles pour ce problème. Étant donné que les capteurs souffrent toujours de contraintes de ressources, le chiffrement à clef publique n'est pas encore pratique. Les techniques d'établissement des clefs doivent être extensibles à des réseaux formés par des centaines et des milliers de capteurs sans fil. De plus, le protocole de communication des réseaux de capteurs sans fil est différent de celui des réseaux traditionnels. Parfois un nœud de capteurs a besoin d'établir une clef avec son nœud voisin ou avec un autre nœud du réseau.

La solution la plus simple pour l'établissement des clefs est une clef pour tout le réseau. Malheureusement, le compromis d'un seul capteur du réseau est suffisant pour dévoiler la clef secrète du réseau et ainsi le déchiffrement de tout le trafic. L'idée d'une clef partagée par tout le réseau pourra servir à établir des clefs de sessions entre les nœuds voisins avant qu'elle soit effacée de la mémoire, mais ces clefs de sessions ne permettront plus d'ajouter de nouveaux capteurs au réseau, ce qui cause un problème d'extensibilité de ce dernier. Une autre approche possible serait de préconfigurer le réseau avec une clef symétrique partagée uniquement entre chaque paire de nœuds. Par contre, dans un réseau de *n* capteurs, chaque nœud doit pouvoir sauvegarder dans sa mémoire $n-1$ clefs, d'où le besoin d'établir $n \times (n-1) \div 2$ clefs pour tout le réseau, ce qui n'est pas trop pratique. Cette approche souffre aussi d'un problème d'extensibilité.

Le chiffrement à clef publique pourra être une autre option dans le futur. Il n'est pas encore supporté d'une manière efficace par les réseaux actuels. Récemment, quelques tentatives d'utiliser le chiffrement à clef publique dans les réseaux de capteurs sans fil [35, 36, 37] existent dans la théorie. Ces articles montrent la possibilité d'effectuer des opérations limitées sur les plateformes existantes comme les capteurs Micaz [38]. La cryptographie sur les courbes elliptiques (*ECC*) [33] est le meilleur choix parmi les différentes approches du chiffrement à clef publique en raison de la petite taille des clefs et la vitesse de calcul. Avec ce genre de chiffrement, chaque nœud serait capable d'établir une clef de session avec n'importe quel nœud dans le réseau et, par la suite, offrir un meilleur niveau de sécurité contre les différentes attaques.

Avec l'absence d'une solution définitive et afin d'éviter l'utilisation du chiffrement à clef publique, les chercheurs ont développé plusieurs protocoles de distribution de clefs symétriques qui sont devenus populaires en raison de leur efficacité :

1- Protocoles de pré-distribution aléatoire des clefs cryptographiques [34, 13,14].
2- SPINS [7].
3- LEAP [22].
4- Fonctions symétriques polynomiales à deux variables [6].
5- Location-binding node ID [8].

- ***Protocoles de pré-distribution aléatoire des clefs cryptographiques*** [34, 13,14] Cette pré-distribution aléatoire existe quand une séquence de clefs symétriques sera choisie aléatoirement parmi un grand nombre de clefs. Chaque séquence de clefs symétriques sera sauvegardée dans un nœud. Chaque paire de capteurs voulant communiquer vont chercher, chacun dans sa propre séquence, une clef commune. S'ils en trouvent une, ils vont l'utiliser pour chiffrer les messages. Cela n'est pas toujours le cas, mais avec une probabilité d'établissement de clefs suffisante, les nœuds peuvent partager des clefs symétriques avec plusieurs nœuds afin d'obtenir un réseau connecté. L'avantage de ce protocole est que la station de base ne sera pas un point unique de défaillance (point dont le reste du réseau est dépendant et dont une panne entraîne l'arrêt complet du système) et que chaque nœud aura besoin de sauvegarder une séquence de clefs symétriques et non pas $n-1$ clefs pour un réseau formé de *n* capteurs. Par contre, la communication entre une paire de nœuds de capteurs n'est pas garantie, et si un adversaire réussit à compromettre un nombre de capteurs suffisant, il sera capable de découvrir toutes les clefs du réseau.

- ***SPINS*** [7] " *Security Protocols for Sensor Networks* " profite de la station de base pour la distribution des clefs. Cette approche propose d'avoir une seule clef partagée entre chaque nœud et la station de base. Si deux nœuds veulent communiquer entre eux, ils doivent établir leurs clefs de sessions à travers la station de base.
Cette méthode n'exige pas beaucoup de mémoire pour sauvegarder une clef mais rend la station de base un point défaillance et augmente le coût de transmission.

- ***LEAP*** [22] " *Localized Encryption and Authentication Protocol* "est un chiffrement localisé et protocole d'authentification. C'est un protocole de gestion des clefs pour les réseaux de capteurs conçu pour appuyer le traitement du réseau interne tout en limitant l'impact de la sécurité d'un nœud compromis sur son voisinage. La conception du protocole est motivée par le fait que différents types de messages échangés entre les nœuds de capteurs ont des exigences de sécurité différentes, et qu'un mécanisme d'établissement de clefs unique ne convient pas pour répondre à ces exigences de sécurité différentes.
Le protocole *LEAP* soutient la création de quatre types de clefs pour chaque nœud de capteurs :

- Une clef personnelle partagée avec la station de base

- Une clef partagée par paires(*Pairwise*) avec un autre nœud capteur
- Une clef *cluster* partagée avec plusieurs nœuds voisins
- Une clef de groupe qui est partagée avec tous les nœuds du réseau.

Le protocole utilisé pour établir et mettre à jour ces clefs est efficace pour ce qui a trait à la communication et à l'énergie, et minimise l'implication de la station de base. Le protocole *LEAP* comprend également un protocole efficace pour l'authentification de la diffusion locale basée sur l'utilisation de porte-clefs à sens unique. L'analyse de performance montre que *LEAP* est très efficace dans le calcul, la communication et le stockage. L'analyse de sécurité du *LEAP*, selon des modèles d'attaques différents, démontre que *LEAP* est très efficace dans la défense contre nombreuses attaques sophistiquées telles que l'attaque *HELLO Flood*, attaque *Sybil* et l'attaque *Wormhole*. Par contre si un nœud est compromis par l'adversaire il peut toujours lancer l'attaque de retransmission sélective sans être détecté.

- *Fonctions symétriques polynomiales à deux variables* [6]: Le Polynôme symétrique à deux variables $f(u, v)$ est utilisé pour établir une clef *Pairwise* entre deux nœuds de capteurs dans un réseau. Avant le déploiement, le serveur génère un polynôme symétrique à deux variables de degré k, $f(u,v) = \sum_{i,j=0}^{k} a_{ij} u^i v^j$ sur un corps fini Fq, où q est un nombre premier qui est assez grand pour accueillir une clef cryptographique. Un polynôme $f(u, v)$ est dit symétrique si $f(u, v) = f(v, u)$.

Le serveur de clef sauvegarde dans chaque nœud ce polynôme. Après le déploiement, chaque nœud régénère un polynôme $g(x)$ en utilisant son identifiant de nœud: $g(x) = f(ID, x)$, et efface $f(u, v)$ de sa mémoire. Supposons que le nœud A possède $g_a(x) = f(a, x)$, et le nœud B possède $g_b(x) = f(b, x)$. Maintenant supposons que le nœud A veut envoyer un paquet au nœud B.

Le nœud A calcule la clef $k1 = g_a(b)$ pour générer un *MAC (message authentification code)* pour le paquet, puis envoie le paquet avec le *MAC* et son *ID* au nœud B. Après avoir reçu le paquet, nœud B calcule $k2 = g_b(a) = g_a(b) = k1$, comme la clef de déchiffrement et de vérification du *MAC*. Si la vérification passe, le nœud B estime que le paquet arrive à partir du nœud authentifié A.

Lors du déploiement, chaque nœud essaie de trouver les nœuds d'acheminement des paquets qui

peuvent être loin. Le nœud a juste besoin d'enregistrer les identifiants de ses voisins dans sa mémoire pour générer une clef *pairwise* plus tard. Les informations d'identification peuvent être échangées à l'aide des protocoles de routage de messages existants tels que le *TinyOS beaconing* [25] ou le *Directed Diffusion* [3]. De plus, peu de mémoire et de frais supplémentaires de communication seront dépensés pour ces opérations. Après que le polynôme à deux variables $f(u, v)$ soit effacé de la mémoire du nœud, il sera difficile pour les adversaires de le régénérer.

- *Location-binding node ID* [8] : Cette technique contient deux étapes. Tout d'abord, avant le déploiement, le serveur génère un polynôme symétrique à deux variables de degré *d*, $f(u,v) = \sum_{i,j=0}^{d} a_{ij} u^i v^j$ sur un corps fini *Fq*, où q est un nombre premier qui est assez grand pour accueillir une clef de chiffrement. Ensuite, le serveur de clefs installe un polynôme dans chaque nœud. Dès que les nœuds sont déployés dans le champ cible, ils commencent à acquérir leur position géographique grâce à des algorithmes de positionnement sécurisés tels que [9]. Chaque nœud lie son identité avec sa position géographique: $ID = x \,||\, y$, où $||$ dénote la concaténation et (x, y) représente les coordonnées géographiques. Enfin, le nœud régénère un polynôme pour l'établissement des clefs *pairwise*: $g(v) = f(ID, v)$, et efface $f(u, v)$ de son cache.

Dans le futur, nous espérons voir un nombre supplémentaire de recherches sur l'amélioration des approches de pré-distribution aléatoire des clefs en fournissant plus de résilience au niveau du compromis des nœuds de capteurs, ainsi que le développement du matériel pour pouvoir soutenir le chiffrement à clef publique.

1.8. Authentification des données

Comme dans les réseaux traditionnels, la plupart des applications dans les réseaux de capteurs sans fil exigent aussi une protection contre l'injection et la modification des paquets. Cette tâche cryptographique s'appelle authentification des données. Les protocoles d'authentification les plus populaires sont les suivants:

1- *OHC* [4,5]
2- *µTESLA* [7]

- *OHC* [4, 5] " *One-Way Hash Chains* "

C'est une séquence de chiffres générée par une fonction publique à sens unique *F*. Pour générer le porte-clefs à sens unique, il faut choisir au hasard la dernière clef K_n de la chaîne, appliquer la fonction *F* plusieurs fois afin de calculer toutes les autres clefs du porte-clefs : $K_i = F(K_{i+1})$.
Dans cette approche, chaque nœud génère et maintient une chaîne de hachage à sens unique $<K0, K1, ..., Kn>$, où $K_i = F(K_{i+1})$. Chaque nœud peut envoyer *K0* à ses nœuds voisins lors de la phase de découverte des voisins. La clef *pairwise* produite par un polynôme à deux variables chiffre le message *K0*. Lorsque un nœud U veut envoyer un message au nœud V, il envoie : $\{data, K_i, MAC\ K_i\ (data)\}$, où $i \geq 1$ et il incrémente *i* de 1 après que le message soit envoyé. Pour authentifier son premier message, V doit juste vérifier si $K_0 = F(K_1)$. Si le message est authentifié avec succès, V sauvegarde K_1 dans sa mémoire pour authentifier le prochain message à venir. Pour authentifier le deuxième message, V a besoin de vérifier si $K_1 = F(K_2)$. Si le message est authentifié avec succès, V sauvegarde K_2 dans sa mémoire pour authentifier le prochain message à venir. Si la mémoire des capteurs est trop limitée, V peut authentifier le deuxième message sans avoir enregistré K_1, $K_0 = F(F(K_2))$. Par contre, les capteurs vont consommer plus d'énergie pour faire ce calcul.
$K_0 = F(F....F(K_i))$, où K_i est le dernier numéro *OHC* reçu par le nœud V, et *i* est le nombre d'appels itératifs de *F*(). Si la vérification échoue, le nœud V peut demander que le nœud U renouvelle ses numéros *OHC*.

- *µTESLA* [7] " *Micro TESLA* "

Cette approche est utilisée pour que la station de base, ou n'importe quel nœud, puisse émettre des informations authentifiées à n'importe quel nœud du réseau.
Le protocole *µTESLA* exige que la station de base soit synchronisée avec les nœuds et admet pour chaque nœud une marge d'erreur maximale de synchronisation. Pour envoyer un paquet authentifié, la station de base calcule simplement un *MAC* sur le paquet avec une clef secrète qui reste confidentielle temporairement. Quand un nœud reçoit un paquet, il peut vérifier que la clef correspondante au *MAC* reçu n'ait pas encore été divulguée par la station de base (en se basant sur

la synchronisation du temps, sa marge d'erreur maximale de synchronisation et le calendrier de la divulgation des clefs). Étant donné que le nœud récepteur sait que la clef du *MAC* n'est connue que par la station de base, alors il est sûr qu'aucun adversaire n'est capable de modifier le paquet pendant sa transmission. Le nœud enregistre le paquet dans son cache. Au moment de la divulgation des clefs, la station de base émet les clefs de vérification à tous les récepteurs. Quand un nœud reçoit la clef de vérification, il peut facilement vérifier son exactitude. Si la clef est correcte, le nœud peut maintenant l'utiliser pour authentifier les paquets stockés dans sa mémoire. Chaque clef *MAC* est un élément d'un porte-clefs, généré par une fonction publique à sens unique *F*. Pour générer un porte-clefs à sens unique (*OHC*), l'expéditeur choisit la dernière clef de la chaîne *Kn* au hasard, et applique *F* à plusieurs reprises afin de calculer toutes les autres clefs: $K_i = F(K_{i+1})$. Chaque nœud peut facilement effectuer la synchronisation du temps et récupérer une clef d'authentification du porte-clefs d'une manière sécurisée et authentifiée, en utilisant le protocole *SNEP* [7] " *Secure Network Encryption Protocol* ".

Après avoir revu d'une façon superficielle les éléments compris dans un réseau *WSN*, ce qui le construit, ce qui le maintient en place, ce qui le fait fonctionner et les possibles attaques à son égard, nous pouvons analyser les différentes méthodes proposées pour lutter contre l'attaque de retransmission sélective et enfin proposer notre hypothèse de recherche.

CHAPITRE 2. Approches existantes contre la retransmission sélective

L'attaque de retransmission sélective est l'une des attaques les plus dangereuses et les plus difficiles à détecter. Pour éviter cette attaque, C. Karlof et D. Wagner ont proposé dans [28] l'envoi des messages par tous les chemins possibles (*Multipath routing* [28]) vers la station de base (figure 2.1). Par contre, si tous les nœuds autour de la station de base sont compromis, les messages ne pourront pas arriver à la station de base. De plus, cette approche consomme plus d'énergie que l'envoi des messages par un seul ou un petit nombre de chemins.

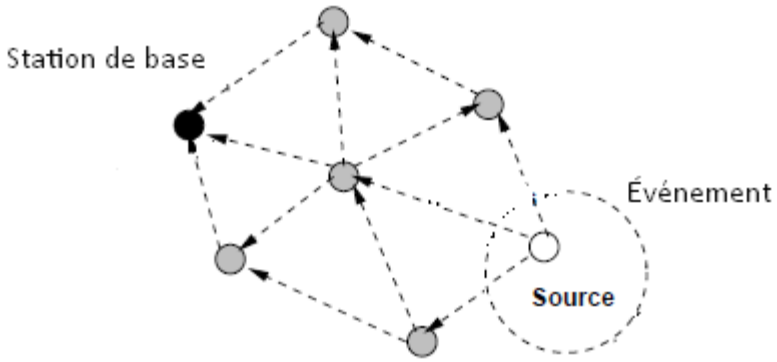

Figure 2.1. Routage à chemins multiples.

Pour cette raison, plusieurs propositions subséquentes ont développé des algorithmes pour se prémunir contre cette menace [28]. Dans ce chapitre, nous allons essayer de résumer les différentes approches existantes afin d'évaluer leur efficacité et leur degré de protection. Par contre, leurs hypothèses et leurs exigences ne sont pas toujours les mêmes, ce qui complique leur comparaison.

Cette première étude, le résultat du travail de Bo Yu et Bin Xiao [1] (2006-2007), propose l'envoi des paquets *Acknowledgment* vers la source afin de confirmer la réception des messages par le destinataire. Cette approche qui, fût introduite en 2006, sera améliorée en 2007 comme nous allons voir plus tard. On commence par la première proposition.

2.1. Detecting Selective Forwarding Attacks in Wireless Sensor Networks [1]

Celle-ci peut fonctionner sur les protocoles de routage et transport *Directed Diffusion* [3] et *PSFQ* [2] présentés dans le premier chapitre. Une clef unique secrète est partagée entre la station de base et chaque nœud de capteurs avant le déploiement du réseau. La distribution des clefs cryptographiques entre les nœuds de capteurs est offerte par le protocole *Fonctions symétriques polynomiales à deux variables* [6]. Chaque nœud doit être équipé d'un système d'acquisition de la position géographique *GPS*. L'authentification des messages entre les nœuds de capteurs sans fil est réalisée par le protocole d'authentification *OHC (One-Way Hash Chain–Based Authentication* [4, 5]).

Comme décrit plus loin, chaque approche utilise des termes particuliers pour s'exprimer, et parfois des nouvelles nominations sont créées. Dans cette première approche, la *fiabilité d'alarme* mesure le rapport entre le nombre de paquets malicieux rejetés détectés et le nombre total de paquets perdus détectés y compris ceux qui sont perdus suite à des collisions. Ainsi que le *Taux non détecté* mesure le rapport entre le nombre non détecté de paquets malicieusement rejetés et le nombre total de paquets malicieusement rejetés. Finalement, le *Chargement de communication* mesure le rapport de chargement total de communication dans un système qui intègre cette méthode de détection avec un système sans protection. Cette dernière mesure est commune aux évaluations de toutes les approches.

Plusieurs suppositions doivent prendre place avant l'évaluation de l'efficacité de chaque approche. Ici, les auteurs considèrent que la station de base et les nœuds de capteurs sont statiques (*immobiles*) et que la station de base ne peut pas être compromise par un adversaire. Avant le déploiement, le serveur charge chaque nœud avec une clef unique secrète et une *fonction symétrique polynomiale à deux variables f (u, v)* [6] qui sera utilisée pour la distribution des clefs cryptographiques entre les nœuds de capteurs. De plus, des protocoles de transport et de routage tels que *Directed Diffusion* et *PSFQ* sont installés dans les capteurs. Au cours de la phase de déploiement, chaque capteur peut acquérir sa position géographique et synchroniser son heure avec la station de base. Comme discuté dans le premier chapitre, la synchronisation dans le réseau

est essentielle pour l'utilisation du protocole d'authentification *OHC (One-Way Hash Chain–Based Authentication* [4, 5]).
Les auteurs supposent aussi que l'adversaire ne parvient pas à compromettre un nœud au cours de la phase de déploiement. Cela veut dire que pendant la distribution des clefs, aucune attaque ne sera lancée contre le réseau. Par contre, après que la phase de déploiement soit terminée, des nœuds de capteurs pourront être compromis par un adversaire dans le but de lancer une attaque de retransmission sélective.
Finalement, la station de base et le nœud source peuvent utiliser des algorithmes plus compliqués IDS (*Intrusion Detection System*) pour prendre des décisions après la détection de l'attaque.

Dans toutes les approches existantes, il est important de savoir quel élément du réseau a la responsabilité de détecter la perte des paquets causée par l'attaque de retransmission sélective. Cet élément sera responsable d'agir contre toute menace. Ici, chaque nœud intermédiaire transmettant les paquets est en charge de la détection des nœuds malicieux. Nous rappelons que *le polynôme symétrique à deux variables* [6] est utilisé pour l'établissement des clefs entre deux nœuds voisins et que *OHC (One-Way Hash Chain–Based Authentication* [4, 5]) est utilisé pour authentifier les nœuds expéditeurs.
Dans tous les réseaux de capteurs sans fil, les informations sont transmises dans des paquets, normalement appelés paquet de *Rapport* ou *packet report*. Dépendamment des applications et des protocoles, d'autres types de paquets peuvent être ajouté pour servir des besoins spécifiques. Dans cette approche, Yu et Xiao ont introduit 2 paquets additionnels au paquet de *Rapport*. Les formats proposés de chaque type des trois paquets sont représentés à la figure 2.2 avec leur taille (octets) correspondante.

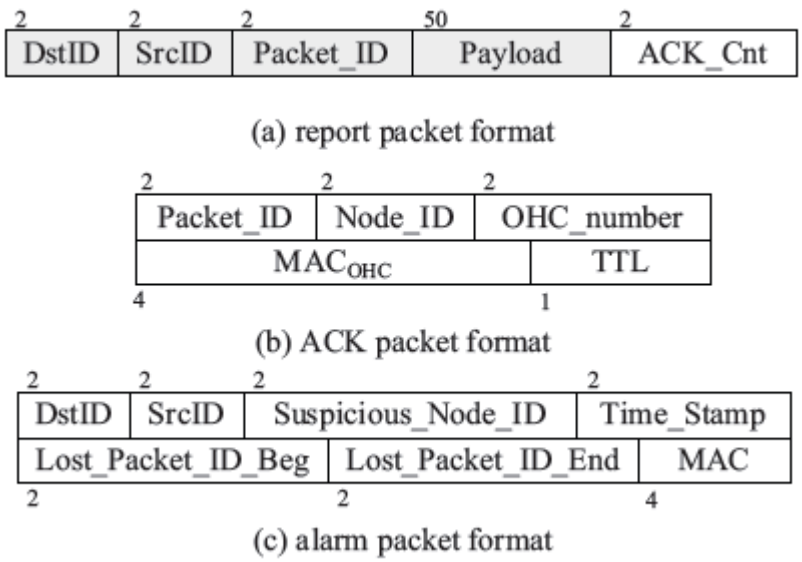

Figure 2.2. Types de paquets[1].

a) Le paquet *Rapport* est de taille 58 octets et contient 5 champs. L'identité de la destination (*DstID*), l'identité de la source (*SrcID*), l'identité du paquet (*Packet_ID*) et le compteur des *ACKs* (*ACK*_Cnt) sont de 2 octets chacun. Ainsi que le reste des 58 octets est réservé pour le rapport ou le message (*Payload*) à transmettre. Ce paquet est celui qui contient le rapport de détection qui sera acheminé vers la station de base.

b) Le paquet *ACK* est de taille 11 octets et contient 5 champs. L'identité du rapport précédent confirmée par cet *ACK* (*Packet_ID*), l'expéditeur actuel du paquet *ACK* (*Node_ID*) et la séquence de chiffres du protocole d'authentification (*OHC_number*) sont de 2 octets chacun. L'authentification du nœud expéditeur (*MAC_ohc*) prend 4 octets d'espace tandis qu'un octet est réservé pour le numéro *TTL*. Ce dernier est un numéro prédéfini qui diminue lors de la transmission du paquet *ACK* nœud par nœud, quand il arrive à 0, le paquet *ACK* sera détruit. Ce paquet est une confirmation, envoyée par le destinataire à l'expéditeur, de réception du paquet *Rapport* correspondant.

c) le paquet d'alarme, de taille 16 octets, contient 7 champs. L'identité du nœud destinataire (*DstID*), l'identité de l'expéditeur actuel du paquet d'alarme (*SrcID*), l'identité du nœud soupçonné d'être malicieux (*Suspicious_Node_ID*), l'heure du système dans le nœud capteur actuel (*Time_Stamp*) et l'identité d'ensemble des paquets perdus (*Lost_packet_ID_Beg,*

Lost_packet_ID_End) consomment 2 octets d'espace. Le reste des 4 octets est réservé à l'authentification de l'expéditeur *MAC*. Ce paquet est envoyé par un nœud soupçonnant la présence d'une attaque de retransmission sélective.

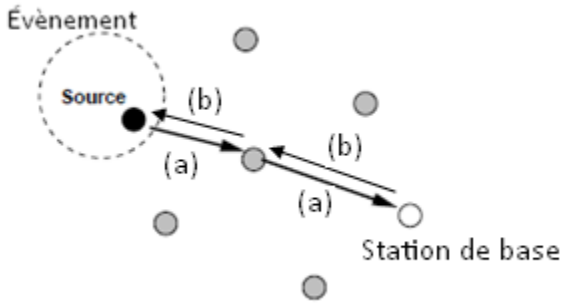

Figure 2.3. L'envoi des paquets *Rapport* et *ACK.*

Dans les conditions normales, le nœud source envoie des paquets *Rapport*, représentés par (a) à la figure 2.3, vers la station de base à travers les nœuds intermédiaires. La destination et les nœuds intermédiaires qui vont être désignés comme *ACK_node* seront responsables de l'envoi des paquets *ACK,* représentés par (b) à la figure 2.3, dans le sens inverse. Un nœud est considéré comme *ACK_node* quand il reçoit un paquet *Rapport* avec un *ACK_Cnt* égal à zéro.

On suppose que l'adversaire a réussi à compromettre plusieurs nœuds du réseau de capteurs après la phase de déploiement. Une fois compromis, les nœuds malicieux vont commencer l'attaque de retransmission sélective.

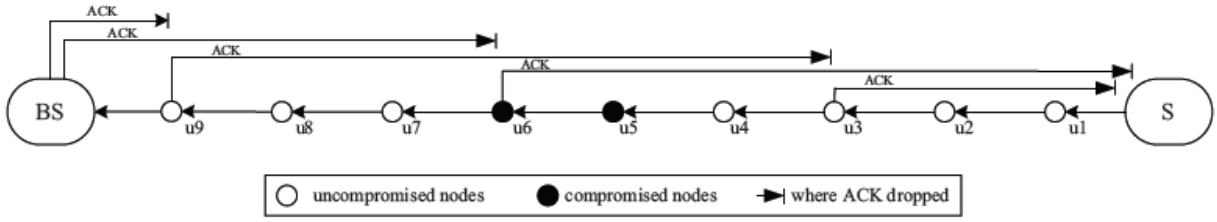

Figure 2.4. L'envoi des paquets *ACK* [1].

Dans la figure 2.4 ci-dessus, le nœud source (*expéditeur paquet du Rapport*) est représenté par la lettre S et la station de base est représentée par BS (*Base Station*). Après la détection d'un événement par S, les paquets *Rapport* sont envoyés de S vers BS liaison par liaison à travers les

nœuds intermédiaires U_1, U_2, U_3, U_4, U_5, U_6, U_7, U_8, U_9. Par contre, les paquets *ACK* sont envoyés par les nœuds U_3, U_6, U_9 (si *ACK_Span* = 2), appelés *ACK_node*, dans le sens inverse. Ces *ACK_node* sont responsables aussi de la destruction du paquet *ACK* après l'avoir vérifié. *TTL* représente le nombre d'*ACK_node* que le paquet *ACK* doit traverser en sens inverse de la propagation des paquets *Rapport* avant qu'il soit détruit. Nous supposons que les nœuds compromis par l'adversaire soient U_5 et U_6. Voici le mode de détection :

- Détection d'un événement: le rapport sera transmis vers la station de base liaison par liaison.
- Lorsque chaque nœud intermédiaire reçoit le paquet *Rapport*:
    - Il fait une copie du *Rapport* dans son cache.
    - Il diminue le compteur *ACK_ Cnt* par un ou réinitialise le compteur *ACK_ Cnt* à sa valeur initiale *ACK_ Span* si le compteur est déjà égal à 0
    - Il transmet le paquet du *Rapport* au nœud suivant vers la station de base.
- Si le nœud trouve *ACK_Cnt* est égal à 0:
    - Il génère un paquet *ACK*, où le *TTL* du paquet *ACK* est fixé initialement à *ACK_TTL*, qui est un indicateur prédéfini représentant le nombre de liaisons que le paquet *ACK* doit traverser avant qu'il soit détruit.
    - Il envoie le paquet *ACK* sur le même chemin vers le nœud source, mais dans le sens opposé.
- Suivant de la retransmission, par le nœud intermédiaire, du paquet *Rapport* à son voisin vers la station de base, il attend les paquets *ACK*.
    - Si moins que *t ACK* paquets sont obtenus pendant un maximum de période *Tack* :
        - Le nœud soupçonne que le *Rapport* précédent ait été rejeté par un nœud malicieux.
        - Le paquet alarme sera transmis à travers plusieurs chemins (*multipath*) vers le nœud source.

Selon l'évaluation de Yu et Xiao, ce système réalise 80% de fiabilité d'alarme avec un taux de collision de paquets d'au plus 15%. Cela veut dire que si 15% des paquets sont détruits à cause de

la nature de la transmission du canal et non pas à cause de l'adversaire, 20 % des paquets alarmes déclenchés après la perte des paquets seront de fausses alarmes. De plus, le *Taux non détecté* et le *coût de transmission* (*communication overhead*) sont proportionnels au taux de collision de paquets.

Le coût de transmission d'un *Rapport* sur un chemin formé de *n* nœuds de capteurs est représenté par l'expression (1) ;

$$CT = n/t + n = n\,(1 + 1/t) \qquad (1) \qquad \text{avec } t = ACK\_TTL$$

et le taux d'efficacité du schéma par l'expression (2) ;

$$\tau = CT / n = 1 + 1/t \qquad (2)$$

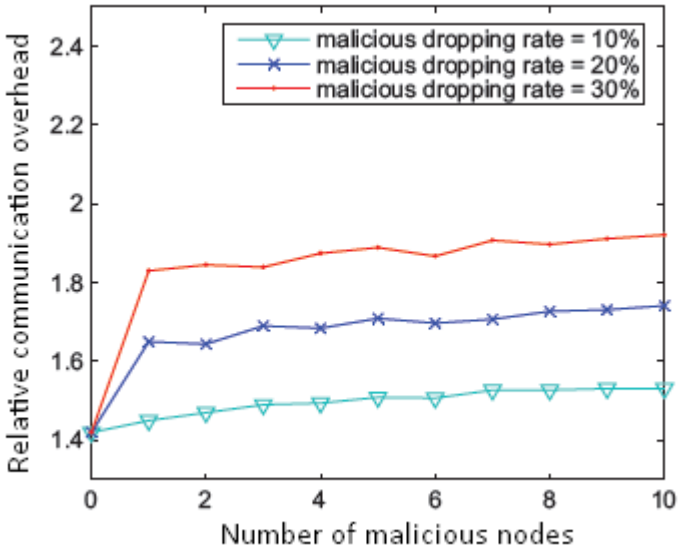

Figure 2.5. Évaluation du coût de transmission [1].

Les résultats de l'évaluation dans la figure 2.5 montrent aussi que lorsque le réseau ne contient pas de nœuds malicieux, le *coût de transmission* (*communication overhead*) n'est d'environ que de 1,4 fois le *coût normal* (avant d'utiliser cette approche). Toutefois, nous trouvons qu'avec l'augmentation des nœuds malicieux sur le même chemin, et avec 20% de paquets perdus à cause de l'attaque de retransmission sélective, le *coût de transmission* (*communication overhead*) sera d'environ 1,65 fois le *coût normal*.

Pour les applications qui demandent l'envoi des rapports à basse fréquence, comme les réseaux de détection du feu dans une forêt, le *coût de transmission* est efficace car les paquets *ACK* sont proportionnels au nombre de paquets *Rapport*.
Yu et Xiao ont donc proposé une simple approche capable de détecter rapidement la perte malicieuse des paquets. Le nœud source ou les nœuds intermédiaires sont capables de détecter les attaques de retransmission sélective. Cela peut être considéré comme un avantage car si le nœud source détecte la perte de paquets, il sera capable de retransmettre le paquet *Rapport* une seconde fois. Par contre, si tous les chemins contiennent des nœuds malicieux, la station de base sera incapable de détecter l'intrusion.

Les paramètres *ACK_Span* et *ACK_TTL* fournissent un compromis entre la capacité de détection et les frais additionnels de la communication. Ils augmentent la sécurité quand un certain nombre de nœuds sur le trajet sont compromis, mais en même temps augmentent le *coût de transmission* (*communication overhead*) et le temps de latence.

On peut dire que cette approche manque d'efficacité lorsque les ressources limitées. Les nœuds de capteurs dépensent beaucoup d'énergie à la détection des nœuds malicieux contre l'attaque de retransmission sélective seulement, tandis que les ressources doivent être réparties pour se prémunir contre toutes les attaques possibles. De plus, si cette approche détecte que certains paquets ont été abandonnés, elle consacre du temps à identifier les nœuds de capteurs possiblement malicieux et trouver d'autres voies afin de renvoyer les paquets perdus jusqu'à la station de base. De même, lors de la détection d'une attaque, les paquets alarmes sont envoyés par tous les chemins possibles (*multipath routing* [28]).
Finalement, si les nœuds expéditeurs des paquets *ACK* (*ACK_node*) sont compromis, un adversaire peut forger des faux paquets *ACK* sans qu'il ne soit détecté. Il suffit que l'adversaire sache *ACK_Span* pour savoir quels sont les nœuds intermédiaires *ACK_node* à cibler. Des améliorations au niveau de ce dernier point ont été faites dans la deuxième version de cette approche représentée à la section suivante.

2.2. CHEMAS: Identify suspect nodes in selective forwarding attacks[8]:

C'est une version améliorée de la première approche *Detecting Selective Forwarding Attacks in Wireless Sensor Networks [1]*. Dans cette section, nous allons démontrer les changements faites par Yu et al dans *CHEMAS* [8] et leur impact sur l'efficacité du protocole proposé.
Comme la version de 2006, cette approche améliorée peut fonctionner sur les protocoles de routage et transport *Directed Diffusion* [3] et *PSFQ* [2] présentés dans le premier chapitre. Une clef unique secrète est partagée entre la station de base avant le déploiement du réseau. La distribution des clefs cryptographiques entre les nœuds de capteurs est offerte par le protocole *Fonctions symétriques polynomiales à deux variables* [6]. Chaque nœud doit être équipé d'un système d'acquisition de la position géographique *GPS*. La station de base et les nœuds de capteurs sont statiques (*immobiles*). La seule différence est que l'authentification des messages entre les nœuds de capteurs sans fil est réalisée par le protocole d'authentification *µTESLA* [7].

Au niveau des définitions, Yu et Xiao ont ajouté les termes *segment* et *Checkpoint*. Les *Checkpoints* designent l'*ACK_node* (U_3, *U6, U9* de la figure 2.4) qui représentent les nœuds intermédiaires responsables de l'envoi des paquets *ACK* dans le sens inverse de transmission des paquets *Rapport*. Un *segment* représente la liaison des nœuds intermédiaires entre deux *Checkpoints* consécutifs (par exemple : la liaison entre U_3 et U_4).
Le but de cette nouvelle version est de changer la nature fixe du choix des *Checkpoints* (*ACK_node*) et de les remplacer par un choix aléatoire. Comme mentionné précédemment dans la version de 2006 [1], l'adversaire est capable de savoir quels nœuds sont des *ACK_node* s'il connait l'*ACK_Span*. Cette information lui permet de savoir quels nœuds il peut compromettre dans le réseau sans qu'il soit détecté lorsqu'il lance une attaque de retransmission sélective. Yu et Xiao et Gao [8] (2007) proposent que le nœud source génère aléatoirement un numéro appelé *Checpoint_seed* responsable de déterminer la liste de *Checkpoints* pour chaque paquet *Rapport* avant de l'envoyer. Ainsi, les nœuds intermédiaires vont partager le rôle de l'envoi des paquets *ACK* dans le sens inverse de l'envoi des paquets *Rapport*. Par la suite, l'adversaire doit compromettre tous les nœuds sur le chemin de transmission du paquet *Rapport* pour détruire les paquets *Rapport* sans être détecté.

Finalement, nous pouvons considérer que l'impact de cette amélioration au niveau du coût de transmission est négligeable tandis que les caractéristiques de détection ont améliorées la protection du réseau contre l'attaque de retransmission sélective.

2.3. CADE: Cumulative Acknowledgement based Detection of Selective Forwarding Attacks in Wireless Sensor Networks [15]

En 2008, une nouvelle approche *CADE* [15] est publiée par K.Young. Ce nouveau papier a pris en considération les deux premières approches faites par Yu et al [1,8] pour proposer une solution un peu différente. Nous avons vu dans les approches précédentes que des paquets *ACK* sont générés par des nœuds intermédiaires et envoyés vers la source du paquet *Rapport*. Ici c'est l'inverse, les paquets *ACK* sont envoyés à la station de base qui sera responsable de détecter l'attaque de retransmission sélective et non pas les nœuds intermédiaires ou le nœud source. *CADE* [15] consiste en 3 phases : la construction de la topologie et le choix du trajet, la transmission des données, et le processus de détection.

Cette approche propose d'utiliser le protocole de routage *SEEM* [12] présenté dans le premier chapitre. Une clef unique secrète est partagée entre la station de base et chaque nœud de capteurs avant le déploiement du réseau. La distribution des clefs et l'authentification des messages entre les nœuds de capteurs sans fil peuvent être réalisées à l'aide du protocole *SPINS* [7] ou d'un *protocole de pré-distribution aléatoire des clefs cryptographiques* [13,14].

Dans cette étude, l'identité d'un nœud est représentée par *l'adresse* d'un nœud. De plus, un *Numéro de séquence* est un nombre utilisé pour identifier les paquets et δ désigne le délai de transmission maximale entre deux nœuds voisins (Temps de latence).

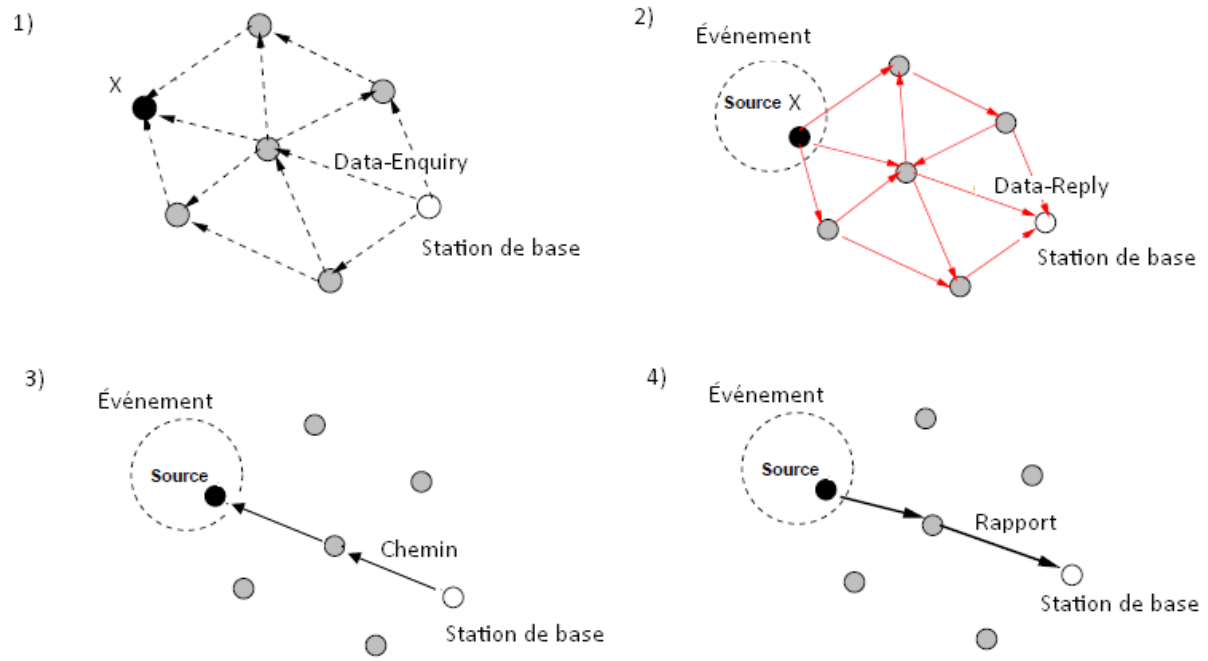

Figure 2.6. L'envoi des paquets *Data-Enquiry*, *Data-Reply*, *Chemin et Rapport* [3].

Au niveau de la communication, *Data-Enquiry,* est un paquet envoyé par la station de base qui contient la description de l'événement à détecter intéressant la station de base (figure 2.6). Il remplace l'expression *Intérêt* du protocole de routage *Direct Diffusion* [3]. Tandis qu'un paquet *Data-Reply* est un paquet envoyé par un nœud responsable d'aviser la station de base que le nœud source est prêt à lui envoyer un message. Par la suite la station de base envoie le chemin que le paquet *Rapport* doit suivre.

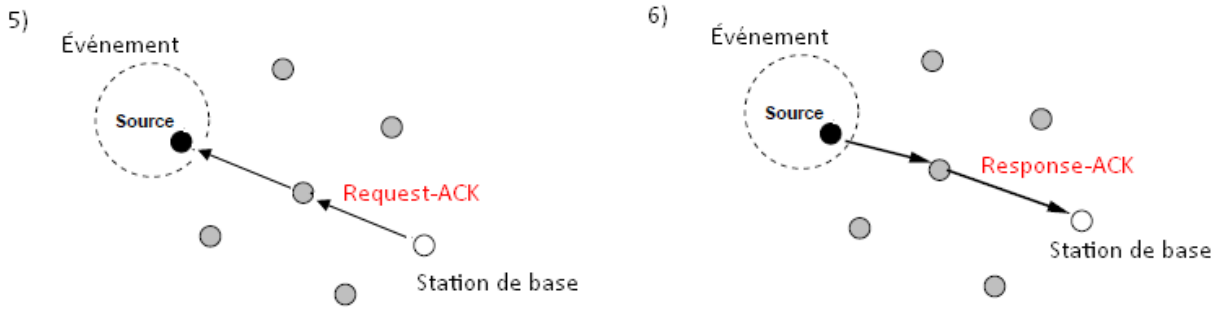

Figure 2.7. L'envoi des parquets *Request-Ack* et *Response-Ack.*

Si la station de base reçoit un paquet *Data-Reply* sans recevoir le paquet *Rapport* dans un délai défini, elle envoie un paquet appelé *Request-ACK* pour investiguer la perte du message (figure 2.7). Finalement, Les nœuds qui reçoivent le paquet *Request-ACK* doivent envoyer un paquet *Réponse* à la station de base.

Ici, les auteurs considèrent que la station de base ne peut pas être compromise par un adversaire. Avant le déploiement, chaque nœud partage une clef secrète avec la station de base appelée *Master Key* et utilise *SPINS* [7] ou un *protocole de pré-distribution aléatoire des clefs cryptographiques* [13,14] pour la distribution des clefs cryptographiques entre les nœuds de capteurs. De plus, le protocole de transport et de routage *SEEM* [12] est installé dans les capteurs pour la transmission des paquets. Au cours de la phase de déploiement, chaque capteur envoie la liste de ses voisins à la station de base. Différemment du protocole *µTESLA* [7], la synchronisation du temps dans le réseau n'est pas obligatoire pour se servir du protocole d'authentification *SPINS* [7]. La liste de voisins sera utilisée par la station de base afin de designer le chemin pour chaque nœud voulant lui envoyer un message.

Les auteurs supposent aussi que l'adversaire ne parvient pas à compromettre un nœud au cours de la phase de déploiement. Cela veut dire que pendant la distribution des clefs, aucune attaque ne sera lancée contre le réseau. Par contre, après la phase de déploiement, des nœuds de capteurs peuvent être compromis par un adversaire dans le but de lancer une attaque de retransmission sélective.

Ici, l'élément qui sera responsable d'agir contre les menaces causées par l'attaque de retransmission sélective est la station de base. Comme dans tous les réseaux de capteurs sans fil, les informations sont transmises dans des paquets. Dans *CADE* [15], K.Young a introduit plusieurs genres de paquets qui sont utilisés dans la détection de l'attaque de retransmission sélective ;

a) *Data-Enquiry* est un paquet contenant la description de l'événement à détecter intéressant la station de base. Ce paquet est envoyé par la station de base à tout le réseau ou parfois à une partie seulement.

b) *Data-Reply* est un paquet envoyé par un nœud par tous les chemins possibles (*multipath*). Il est responsable de notifier la station de base que le nœud source est prêt à lui envoyer un message. Ce paquet contient son *adresse* (identité) et le numéro de séquence du message à envoyer.

c) Un paquet *Chemin (Route)* est un paquet envoyé par la station de base contenant le chemin que le nœud doit utiliser pour envoyer le paquet *Rapport*.

d) Un paquet *Rapport* est un paquet envoyé par le nœud source vers la station de base sur le chemin désigné par la dernière à l'aide du paquet *Chemin*. Il contient son adresse, un numéro de séquence et les données.

e) Un paquet *Request-ACK* est envoyé par la station de base pour investiguer la perte d'un paquet *Rapport*. Il contient le numéro de séquence du paquet *Rapport* perdu, un nombre aléatoire *R* et le nombre de liaisons entre chaque nœud sur le chemin et le nœud source qui a envoyé le paquet *Rapport* perdu.

f)

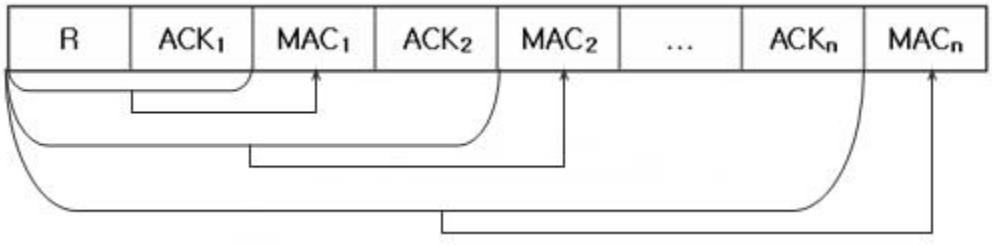

Figure 2.8. Response-ACK [15].

On représente à la figure 2.8 ci-dessus les paquets *Response-ACK* qui sont formés d'un nombre aléatoire *R* provenant du paquet *Request-ACK*, d'un *ACK* et d'un *MAC*. Le champ *ACK* est une variable de 2 bits qui peut prendre la valeur *Data-ACK* si le nœud a déjà vu le paquet de données ou bien *DATA-NAK* si ce n'est pas le cas.
Si le nœud n'a même pas vu le paquet *Chemin* envoyé par la station de base, la valeur d'*ACK* sera *Route-NAK*. Ce paquet *Response-ACK* est transmis à travers le même chemin que le paquet *Rapport*. Un *MAC* est calculé avec la clef partagée entre la station de base et chaque nœud.

L'idée est simple. Chaque nœud voulant envoyer un paquet *Rapport* à la station de base notifie la dernière avec un paquet *Data-Reply* contenant le numéro de séquence du paquet *Rapport* (figure 2.6). Quand la station de base reçoit le paquet *Data-Reply,* elle envoie au nœud source le chemin

que le paquet *Rapport* doit suivre pour atteindre la station de base et commence un chronomètre pour attendre le paquet *Rapport*. Si la station de base ne reçoit pas le paquet *Rapport*, elle envoie un paquet *Request-ACK* à tous les nœuds sur le chemin correspondant pour demander s'ils ont reçu le paquet *Rapport*. Les nœuds répondent par oui ou non à travers un paquet *Response-ACK*. La station de base analyse les réponses des nœuds et en déduit ceux qui sont malicieux.

Comme mentionné précédemment, *CADE* [15] est composé de trois phases. Dans la première phase, la construction de la topologie et le choix du trajet ou tout simplement la phase de déploiement, K.Young utilise le protocole *SEEM* [12]. Selon *SEEM* [12], la station de base transmet un message appelé *ND (Neighbor Discovery)* à tous les nœuds de capteurs du réseau. Chaque nœud qui reçoit un *ND*, enregistre dans sa liste de voisins l'adresse de l'expéditeur de ce *ND*. Si le numéro de séquence du *ND* n'est pas déjà reçu, le nœud le retransmet. Après que la liste de voisins soit remplie, chaque nœud génère plusieurs copies de cette liste et les envoie à la station de base par tous les chemins possibles. Quand la station de base reçoit toutes les listes de voisins, elle peut créer la topologie du réseau pour envoyer les paquets *Chemin* (*Route*) plus tard.

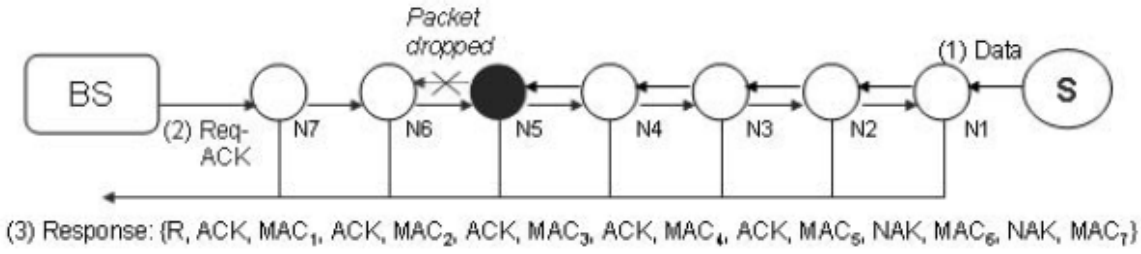

Figure 2.9. Mode de détection de l'attaque de retransmission sélective [15].

Dans la deuxième phase du *CADE* [15], appelée phase de transmission de données, la station de base envoie le paquet *Data-Enquiry* à l'ensemble du réseau. Comme dans la figure 2.9, les nœuds avec les données correspondantes avisent (avec un paquet *Data-Reply* contenant le numéro de séquence du paquet *Rapport)* la station de base qu'ils sont prêts à envoyer des paquets *Rapport*. La station de base sélectionne un chemin pour chaque nœud qui a envoyé un paquet *Data-Reply* et l'envoie à l'aide du paquet *Chemin*. Au moment où ce paquet *Chemin* est envoyé, la station de base définit à chaque nœud un *chronomètre* qui sera utilisé dans le processus de détection. Finalement, le nœud source envoie le paquet des données ou *Rapport* à la station de base en utilisant le chemin reçu de ce dernier.

La troisième et la dernière phase de *CADE* [15] est présentée par l'algorithme de détection suivant :

- Si le *chronomètre* de la station de base qui a été fixé à la phase de transmission de données expire avant qu'un paquet de données arrive :
  - La station de base envoie un message *Request-ACK* vers le nœud source du paquet perdu sur le même chemin utilisé pendant la transmission de données.
  - Chaque nœud qui reçoit le paquet *Request-ACK*:
    - Enregistre le *nombre de liasons* entre le nœud source et lui-même déterminé par la station de base.
    - Définit un chronomètre avec $2\delta \times$ *nombre de liasons* secondes
    - Transmet le paquet *Request-ACK* au nœud suivant sur le chemin vers le nœud source.
    - Si le *chronomètre* du nœud expire avant qu'il reçoive un paquet *Response-ACK* provenant du nœud source :
      - Il génère un paquet *Response-ACK* lui-même et l'envoie vers la station de base.
    - Si le nœud reçoit un paquet *Response-ACK* avant que son *chronomètre* expire:
      - Il arrête le chronomètre
      - Ajoute un *ACK* et un *MAC* au paquet *Response-ACK*.
      - Envoie le paquet *Response-ACK* vers la station de base à travers le nœud suivant sur le chemin de transmission.

o Après la réception d'un paquet *Response-ACK,* la station de base gère un processus de décision.

o Si la station de base ne reçoit aucun paquet *Response-ACK* de tous les nœuds du chemin et avant que son chronomètre n'expire, elle conclut que son nœud voisin sur le chemin de transmission est malicieux.

Dans l'évaluation de *CADE* [15], K.Young compare son travail avec d'autres systèmes existants, tels que *CHEMAS* [8] et *Multipath routing* [28]. L'auteur affirme que *CADE* [15] est capable d'identifier les nœuds malicieux, ce que le protocole *Multipath routing* [28] est incapable de faire. De plus, il affirme que *CADE* [15] a un coût de transmission moindre que les deux autres car il utilise le *Multipath routing* [28] seulement pour l'envoi des petits paquets *Data-Reply*, et il génère des paquets *ACK* juste quand un paquet *Rapport* est perdu. De même, cette approche n'a pas besoin de synchroniser le temps entre les nœuds et la station de base. D'après K,Young, le protocole *CADE* [15] a besoin de moins de mémoire que dans *CHEMAS* [8] vu qu'il remplace le protocole d'authentification *OHC one-way chain* [4,5] par *SPINS* [7].
Finalement, pour les applications qui demandent l'envoi des rapports à haute fréquence, comme les réseaux de détection de trafic sur les autoroutes, le *coût de transmission* n'est pas du tout efficace car les paquets *Data-Reply* sont proportionnels aux paquets *Rapport*.

Le protocole *CADE* [15] est une approche qui permet la détection des nœuds malicieux déclenchant une attaque de retransmission sélective. Chaque nœud doit envoyer à la station de base un paquet de notification (*Data-Reply*) avant d'envoyer un paquet *Rapport*. Si la station de base reçoit le paquet de notification sans le rapport, elle soupçonne une attaque malicieuse.
Cela n'est sûrement pas pratique pour les applications qui demandent l'envoi des rapports à haute fréquence ; même pour les applications à basse fréquence chaque nœud doit envoyer chaque fois deux paquets à la station de base, ce qui double le coût de transmission et de calcul dans les conditions normales. Il est vrai que cette approche identifie les nœuds malicieux ; par contre elle manque de précision car elle peut soupçonner deux nœuds sans savoir lequel est le coupable.
L'avantage de *CADE* [15] est qu'il est capable de détecter l'attaque *Sinkhole* et ne nécessite pas la synchronisation du réseau. À l'instar de *CHEMAS* [8] qui utilise le *Multipath routing* [28] pour l'envoi des paquets alarmes, *CADE* [15] lui aussi envoie les paquets *Data-Reply* de la même façon.
Finalement, le point faible dans cette approche est qu'elle n'est pas capable de détecter l'attaque de retransmission sélective lorsque l'adversaire réussit à compromettre les nœuds autour de la station de base car il pourra détruire les paquets *Data-Reply* sans détection.

2.4. Detecting Selective forwarding attacks in wireless system networks using Two-hops Neighbor Knowledge [17]:

Dans les deux prochaines approches, le mode de détection de l'attaque de retransmission sélective est différent de celles des trois précédentes. Ici, les auteurs profitent de la densité des nœuds de capteurs dans le réseau de capteurs sans fil pour détecter l'attaque de retransmission sélective. Les nœuds voisins du nœud malicieux sont responsables de la détection de l'attaque sans l'utilisation des paquets de confirmation *ACK*. Nous commençons par l'approche publiée en 2008 par Hai et Huh [17].

Cette approche propose d'utiliser un protocole de gestion de clefs comme celui de *LEAP* [22] présenté dans le premier chapitre. Nous rappelons que *LEAP* [22] offre 4 genres de clefs : une clef personnelle partagée avec la station de base, une clef partagée par paires (*Pairwise*) avec un autre nœud capteur, une clef *Cluster* partagée avec plusieurs nœuds voisins et une clef de groupe qui est partagée avec tous les nœuds du réseau. Ce genre de clefs, surtout la clef *Cluster*, permettent à un nœud quelconque d'écouter les messages retransmis par ses voisins afin de surveiller leurs comportements. Au niveau des protocoles de routage, les auteurs n'exigent pas l'utilisation d'un protocole spécifique.

Dans cette approche, un module de détection est installé et activé dans tous les capteurs du réseau. Ce module de détection est une application responsable de détecter passivement l'attaque de retransmission sélective lancée par un nœud voisin. Les nœuds de capteurs dont le module de détection sont installés dans leur mémoire et activé, s'appellent *Monitor nodes*.

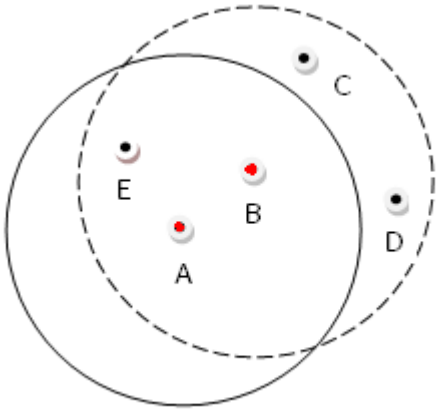

Figure 2.10. Voisinage du capteur A.

Une Liste de voisinage *one-hop* est une liste enregistrée dans chaque nœud de capteurs identifiant tous ses voisins, tandis que *Two-hops* est la liste identifiant tous les voisins de ses voisins. Par exemple dans la figure 2.10 ci-dessus, la liste de voisinage directe ou *One-hop* du nœud A contient les nœuds B et E. Par contre, la liste de voisinage indirecte ou *Two-hops* du nœud A à partir du voisin B contient les nœuds C et D.
Finalement, chaque nœud associe un compteur à chacun de ses voisins, ce compteur sera responsable de déterminer une activité anormale d'un nœud voisin. Une fois que ce compteur dépasse le seuil permis, le nœud efface le voisin correspondant au compteur de sa liste de voisin *one-hop*.

D'abord, les auteurs supposent que n'importe quel nœud sur le chemin de transmission peut être malicieux, rendu tel par un adversaire, y compris le nœud source. Comme dans toutes les approches précédentes, aucune attaque n'est lancée contre le réseau de capteurs pendant la phase de déploiement. De plus, les auteurs supposent que la nature du réseau est statique, ce qui signifie que la station de base et les nœuds sont fixes et les voisins d'un nœud de capteurs ne changeront pas dans le futur. Finalement, l'efficacité de cette approche dépend de la densité des nœuds dans la zone de déploiement. Plus de nœuds voisins sont présents, plus il y a de chances de détection de l'attaque.

Pendant la phase de déploiement du réseau de capteurs sans fil, le module de détection enregistre la liste de voisinage *Two-Hops* dans le cache de chaque nœud. Si nous prenons la figure 2.10 ci-

dessus, on remarque que les nœuds C et D ne sont pas les voisins du nœud A. Cela veut dire qu'il n'existe pas de communication directe entre A et C ou D. Pendant la phase de déploiement, chaque nœud construit sa liste *one-hop*, et l'envoie à ses voisins. Par exemple B envoie l'information que ses voisins sont A, C, D et E. Quand A reçoit cette liste du nœud B, il enregistre dans sa liste de voisinage *Two-hops* les nœuds C et D pour s'en servir plus tard dans le module de détection.

Une fois la phase de déploiement terminée, chaque nœud du réseau possède deux listes dans sa mémoire (*one-hop* et *Two-hops*) et le module de détection. Selon la topologie du réseau, les nœuds de capteurs voisins partagent une clé *Cluster* pour chiffrer les messages entre eux.

Par exemple si E envoie un paquet à B, le nœud A peut l'écouter et le déchiffrer avec la clé *Cluster* partagée entre les trois. Cette approche profite de cette option offerte par le protocole *LEAP* [22] pour vérifier sélective à partir du module de détection s'il y a un voisin qui lance une attaque de retransmission. Pour se rendre compte de la perte de paquets à cause d'une collision, deux facteurs α et β sont fournis pour améliorer la précision de détection: $0 < \alpha < 0.5 < \beta < 1$. Nous observerons plus tard comment ces valeurs vont déterminer si un nœud est malicieux ou non.

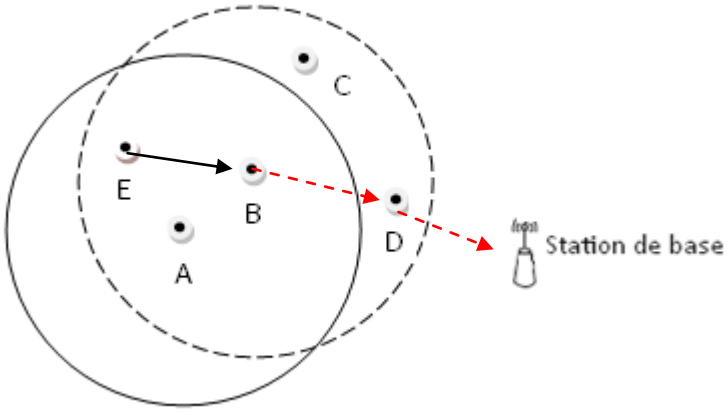

Figure 2.11. L'envoi du paquet *Rapport* vers la station de base.

L'idée derrière cette approche est de profiter de la densité des nœuds de capteurs pour surveiller le comportement de chaque nœud. Dans cette approche, une condition nécessaire pour commencer la phase de détection dans un nœud est que le paquet qu'il va écouter soit transmis entre deux nœuds qui sont ses voisins (existant sur la liste *one-hop*). Si on prend l'exemple de la figure 2.11, on va supposer que le nœud E envoie un paquet au nœud B. Si A n'est pas un voisin direct de B, il reste passif. Mais vu que le nœud A est le voisin du nœud E, et que le paquet est chiffré avec la clé

Cluster, A va écouter le paquet. Par la suite, le nœud A va vérifier la destination du paquet, qui dans cet exemple est B, avant qu'il soit retransmis vers la station de base (destination finale). Si B est un voisin direct du nœud A, ce dernier va enregistrer le paquet et surveiller le comportement du nœud B. Ici, trois scénarios peuvent se produire ; le nœud B retransmet le paquet à la bonne destination vers la station de base, le nœud B laisse tomber le message et ne le retransmet plus, le nœud B retransmet le message mais dans la mauvaise direction. Étant donné que le nœud A peut aussi écouter les paquets retransmis par le nœud B, il peut comparer la destination du paquet avec sa liste de voisinage *Two-hops*. Par la suite il peut savoir si le nœud B a retransmis le paquet vers la station de base à travers le bon chemin qui est le nœud D dans notre exemple. Dans le premier scenario, si le nœud B retransmet le message vers la bonne destination, le nœud A efface le paquet de sa mémoire. Dans le deuxième scenario, si le nœud B laisse tomber le message, le nœud A envoie un paquet alarme au nœud source E avec un facteur α ($0<\alpha<0.5$). Par la suite, les nœuds E et A vont ajouter α au compteur associé à leur voisin B (Un nœud source ne peut pas être un *Monitor Node*). Si ce compteur dépasse une valeur X déterminée par l'application, le nœud B sera considéré comme malicieux et il ne recevra plus de paquets à transmettre car il sera exclu de la liste de voisinage. Même chose pour le troisième scenario si le nœud B décide de transmettre le paquet au nœud C au lieu de D, mais ici la valeur du facteur β sera plus grande ($0.5 <\beta <1$) car dans le deuxième scenario, la perte du paquet peut être causée par collision et non pas par une activité malicieuse. Voici le module de détection fourni par les auteurs :

Module de détection dans le *Monitor Node* :

Répéter
<Ecouter les paquets dans l'entourage>
Vérifier <Les adresses du paquet>
Si l'identité du nœud source et destination se trouvent dans la liste des voisins 1-hop :
    Enregistrer le paquet avec un *chronomètre* τ
        Si (le paquet est retransmis vers une destination présente sur la liste de voisinage *two-hops*)
            Supprimer le paquet enregistré
        Sinon envoyer un paquet d'alerte avec un facteur β

Si le temps τ est expiré

Envoyer un paquet d'alerte avec le facteur α et supprimer les paquets enregistrés

Jusqu'à ce qu'il n'y ait plus de transmission

Parfois, plusieurs nœuds témoins « *Monitor Node* » observent l'activité malicieuse en même temps. Au lieu d'envoyer plusieurs paquets alarme au nœud source, les auteurs proposent que si un nœud témoin « *Monitor Node* » remarque une transmission de paquets alarmes avant qu'il envoie le sien au nœud source, il arrête l'envoi de paquet pour ne pas causer une collision et pour sauver de l'énergie. Sinon, le *Monitor Node* envoie le paquet alarme au nœud source.
Dans l'analyse théorique de cette approche, la probabilité de détection d'une attaque de retransmission sélective dépend de trois facteurs :

- Nombre de voisins du nœud malicieux *Monitor Nodes*.
- Probabilité de détection manquée P_c d'un *Monitor Node*
- Le seuil malicieux X

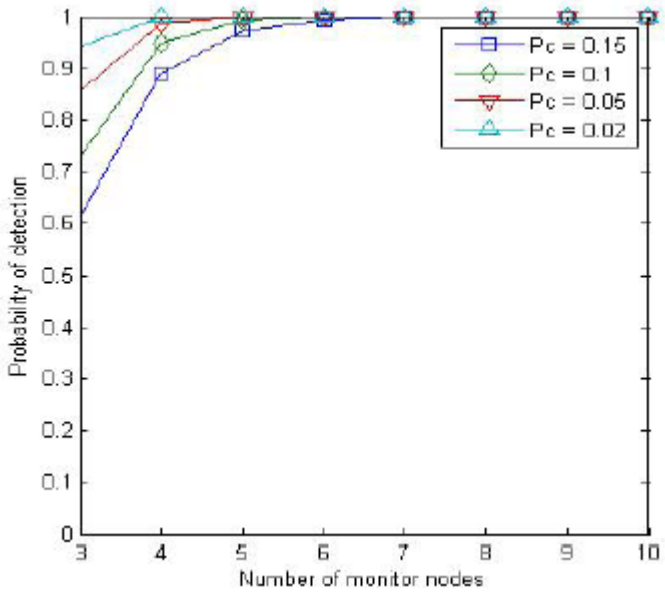

Figure 2.12. Probabilité de détection P_D de l'attaque de retransmission sélective [17].

L'analyse pratique de cette approche (figure 2.12) montre l'augmentation du nombre des *Monitor Node* qui augmente la probabilité de détection P_D dans un réseau de capteurs sans fil. Par contre, cette augmentation accroît aussi la probabilité de collision, ce qui peut entraîner des pertes de

paquets d'alarme. Avec moins de 10% de nœuds malicieux présents dans le réseau et entre 2% et 5 % de pourcentage de collision, cette approche est capable de détecter 90 % des nœuds malicieux.

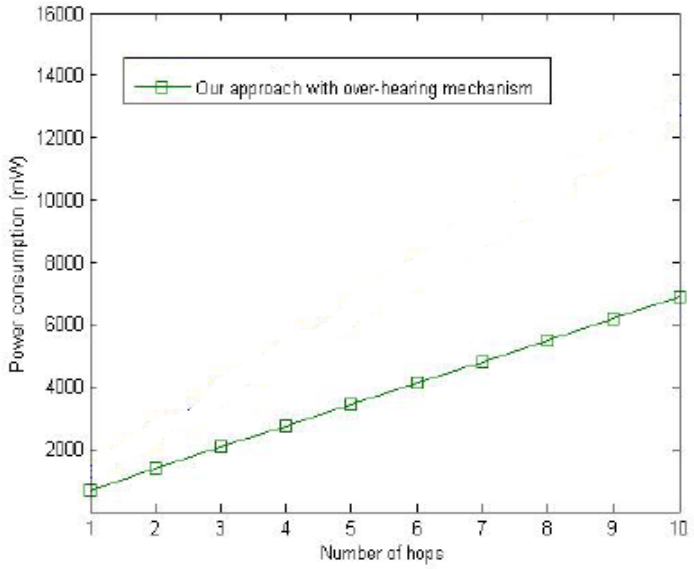

Figure 2.13. Évaluation du coût de transmission [17].

Finalement, au niveau de la consommation d'énergie, Hai et Huh analysent la consommation des *Monitor nodes* (en mW) après une transmission entre un nœud A et un nœud B séparés par *n* liaisons (*hop*) dans la figure 2.13. En utilisant des modèles de consommation d'énergie [39,40], les auteurs supposent que leur approche augmente légèrement la consommation des ressources.

À partir de cette approche, Hai et Huh ont présenté un nouveau mécanisme de détection de l'attaque de retransmission sélective. Afin de profiter de la densité des nœuds dans un environnement, chaque nœud est responsable de la surveillance de ses voisins. D'abord, cela permet au nœud source de détecter les attaques sélectives de transmission et il peut ainsi renvoyer le paquet *Rapport* sur le même chemin si le nœud soupçonné n'est pas encore considéré malicieux, ou bien sur un autre chemin s'il l'est déjà. Par la suite, le module de détection n'exige pas la synchronisation de temps entre les nœuds de capteurs ou avec la station de base pour identifier les nœuds malicieux présents sur le chemin de transmission. De même, la consommation d'énergie réduite de la part des *Monitor Nodes* est un aspect important dans la sécurité des réseaux de capteurs sans fil. L'analyse pratique montre que la durée de vie de cette approche est plus longue que celle de l'approche proposée précédemment *CHEMAS* [8].

Tout de même, nous trouvons que dans cette approche proposée en 2008 la topologie du réseau est statique, ce qui signifie qu'un nouveau nœud ne peut être ajouté à la liste des voisins une fois la phase de déploiement terminée. De plus, d'après les auteurs, si plus de 25 % des nœuds sont compromis, cette approche devient inefficace pour la détection des nœuds malicieux. Finalement, cette approche profite de la dense présence des *Monitor nodes* pour détecter les nœuds malicieux, mais ce n'est pas toujours le cas. Enfin, la faiblesse de ce système est que n'importe quel un nœud compromis peut utiliser les paquets alarme pour accuser un nœud légitime d'être malicieux.

2.5. A Resilient Packet-Forwarding Scheme against Maliciously Packet-Dropping Nodes in Sensor Networks [19]

Dans cette approche, le mode de détection de l'attaque de retransmission sélective est semblable à l'approche précédente. L'envoi des paquets *Rapport* se fait à travers un seul chemin dans les conditions normales. Les nœuds de capteurs du réseau qui sont voisins du nœud malicieux sont responsables de détecter l'attaque de retransmission sélective.

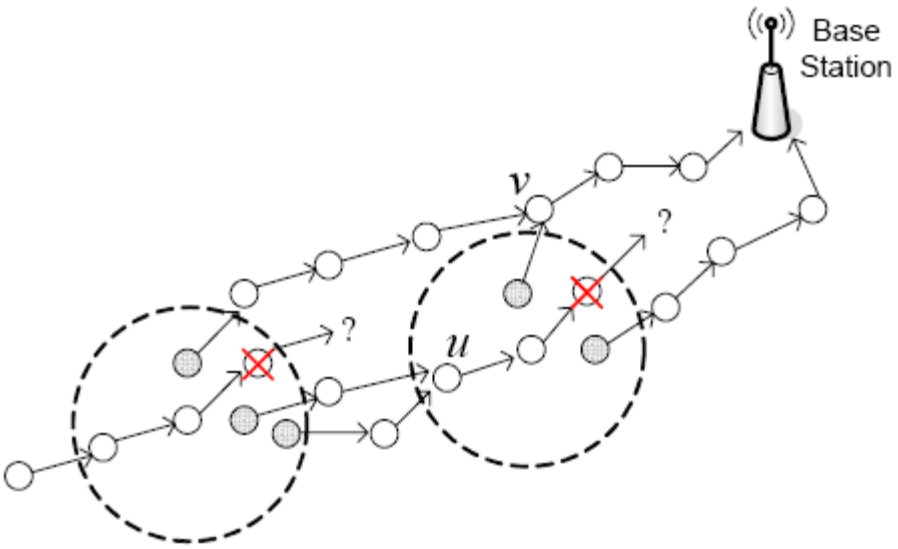

Figure 2.14. L'envoi des parquets *Rapport* vers la station de base [19].

Ici, la différence est la réaction du nœud source et voisin. Dans l'approche précédente, lorsqu'un nœud voisin (*Monitor nodes*) soupçonne une activité malicieuse, il envoie un paquet alarme au nœud source qui sera responsable de renvoyer le paquet *Rapport* de nouveau à la station de base

sur le même chemin, ou un chemin différent. Dans ce dernier modèle (figure 2.14), si les *Monitor nodes* (appelés *Sub watch*) observent un comportement malicieux, ils envoient eux-mêmes le paquet *Rapport* sur tous les chemins possibles (*multipath*) vers la destination finale sans informer le nœud source (appelé *Primary watch*). Afin d'utiliser la clef *Cluster*, Lee et Choi utilisent dans cette approche le même protocole de gestion de clefs que celui de l'approche précédente *LEAP* [22], et *TinyOS Beaconing* [25] comme protocole de routage.

Selon l'analyse pratique, s'il n'existe pas de nœuds malicieux sur le chemin, le coût additionnel en ressource est presque négligeable. Par contre, pour un réseau de haut degré *d* (moyenne de capteurs voisins à un nœud), les exigences supplémentaires de mémoire pour chaque nœud est *d* (8 + *ld*) octets avec *l* la taille de l'*ID* d'un nœud.

Finalement dans cette approche comme celle qui précède, le nœud source et ses voisins ont la capacité de détecter une attaque de retransmission sélective possible et la synchronisation entre les nœuds n'est pas nécessaire pour activer le mode de détection contre l'attaque. De plus, si un rejet d'un paquet est détecté, le paquet sera transmis à l'aide de routage *multipath* (plusieurs chemins possibles), mais la station de base ne peut pas identifier le nœud malicieux.

Ceci entraine que le nœud malicieux ne peut pas être exclu du réseau puisque aucune alerte n'est envoyée à la station de base ou au nœud source. Ainsi, tous les paquets passant par un nœud malicieux, seront renvoyés par ses voisins à travers tous les chemins possibles, ce qui cause une perte d'énergie au niveau du réseau. De même, les ressources de mémoire de chaque nœud sont proportionnelles à la densité des nœuds dans le réseau pour enregistrer la liste des voisins et leurs clefs correspondantes.

2.6. Selective Forwarding Attack Detection using Watermark in WSNs [20]

Afin de détecter l'attaque de retransmission sélective, une dernière approche est proposée en 2009 par Deng et al. Selon une séquence de bits définie par la station de base, chaque nœud source ajoute à chaque paquet le symbole « 1 » ou « » (espace vide) avant de l'envoyer à la station de base. Cette technique appelée *Watermark*, permet à la station de base de détecter une perte de

paquet en comparant les symboles des paquets reçus, avec la séquence de bits. Si la perte des paquets dépasse le taux de perte permis, la station de base commence le mode de détection du nœud malicieux, et diminue sa valeur de confiance une fois trouvé.

Dans le but de choisir un chemin fiable pour retransmettre les paquets, il est important de distinguer entre les nœuds honnêtes et les nœuds malicieux. Pour cela, une valeur de confiance (*Trust value*), représentée par un numéro, est attribuée à chaque nœud du réseau. Chaque nœud connaît la valeur de confiance de ses voisins à travers la station de base. Lorsque le nœud source choisit un chemin de transmission, il choisit les nœuds dont les valeurs de confiance (*Trust value)* sont les plus élevées pour que les paquets *Rapport* atteignent la station de base en toute sécurité. Pendant la phase de déploiement, tous les nœuds possèdent la même valeur de confiance. Si un nœud est soupçonné malicieux, la station de base diminue sa valeur de confiance puis l'augmente, mais lentement.

Au début de la phase de déploiement, la valeur de confiance *tvi* prend la valeur *a*. Si un nœud *i* est soupçonné malicieux, la valeur de confiance diminue de moitié :

- Méthode de diminution multiplicative: $tv_i = 1/2\ tv_i$

Quand la valeur d'un nœud *i* diminue, ca veux pas dire que le nœud *i* ne sera plus utilisé. La valeur de confiance tv_i augmente linéairement avec le temps :

- Méthode d'augmentation additive: $tv_i = tv_i + k$ $\qquad (tvi + k < a)$

La valeur de *k* n'est pas fixe et elle peut changer dynamiquement.

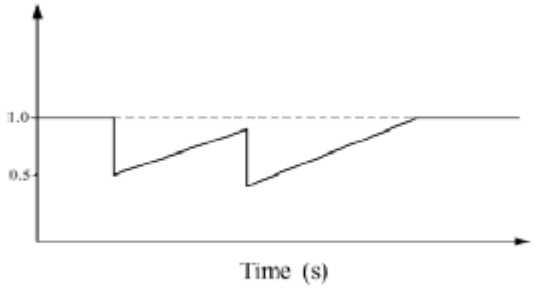

Figure 2.15. Valeur de confiance tv_i [20].

Par exemple dans la figure 2.15, en supposant que la valeur de confiance est 1, si un nœud est soupçonné malicieux, sa valeur de confiance est diminuée jusqu'a 0.5 et augmentée lentement. À

l'aide de la valeur de confiance, cette méthode empêche un adversaire à continuer son attaque, et si un nœud est soupçonné malicieux par erreur (à cause d'une collision de paquets), il sera réutilisé après que sa valeur de confiance tv_i augmente pour atteindre sa valeur initiale a.
Au niveau des protocoles de routage, cette approche utilise une méthode de transmission géographique [41] pour transmettre les données qui exige que chaque nœud de capteurs retransmet le paquet à son voisin qui est le plus proche de la destination. Par contre, dans cette approche, la sélection du chemin se fait non seulement à partir du vecteur distance d'un nœud à la station de base dv_i, mais selon la valeur de d'équilibre $bv_i = dv_i / tv_i$ d'un nœud i (tv_i représente la valeur de confiance du nœud i).
L'algorithme de sélection du chemin est représenté ci-dessous :

Entrée: nœud source X, destination finale Y
Sortie: le chemin de routage optimal
Etapes à suivre:

Pour tous les nœuds du réseau, initialiser la position des nœuds:

1) Si X et Y sont des voisins

2) Le chemin de transmission contient X et Y

3) Sinon, si X et Y ne sont pas voisins

4) X calcule la valeur d'équilibre de tous ses voisins

5) $bv_i = dv_i / tv_i$

6) X choisit le voisin i avec la moindre valeur d'équilibre pour lui transmettre le paquet

7) Si i est la station de base

8) Stop

9) Si i n'est pas la station de base

10) X=i

11) Allez à 4)

Avec cet algorithme, un chemin de transmission de paquet peut être choisi. x est le nœud source qui envoie les paquets *Rapport* à la station de base y. Avant que le choix du chemin commence,

tous les nœuds reçoivent le vecteur distance de leurs nœuds voisins à la station de base dv_i et le sauvegarde dans la mémoire. Par la suite, le nœud source reçoit les valeurs de confiance tv_i de la station de base et calcule la valeur d'équilibre $bv_i = dv_i / tv_i$ de tous ses voisins. En se basant sur la valeur d'équilibre minimale, le nœud source choisit le voisin correspondant pour envoyer le paquet *Rapport.* Dans la figure 2.16 ci-dessous, le nœud source est prêt à envoyer des paquets *Rapport* à la station de base, les nœuds *a,b* et *x* sont les voisins du nœud source. Le nœud source calcule la valeur d'équilibre de ses trois voisins, si on suppose que la valeur d'équilibre du voisin *a* est 3, *x* est 2 et *b* est 4, alors le nœud source va envoyer les paquets au nœud voisin *x*.

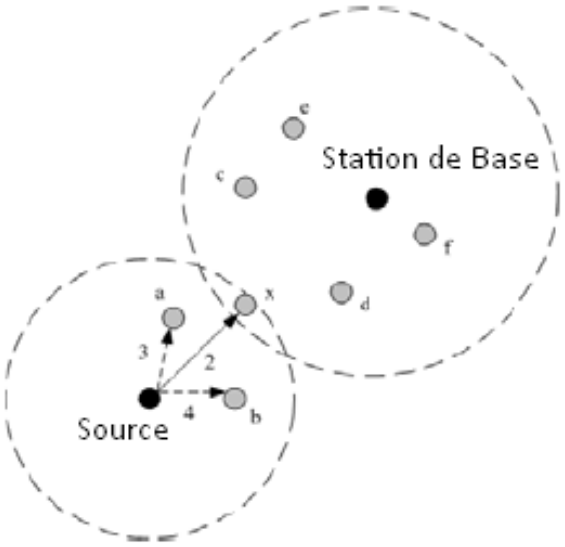

Figure 2.16. L'envoi des parquets *Rapport* à la station de base [20].

Tout d'abord, la station de base génère une séquence de *K* bits comme un *Watermark* w_n original. Par la suite, les paquets prêts à être envoyé par le nœud source sont divisés en *K* groupes, et chaque paquet de ce groupe possède un numéro de séquence transmis selon ce numéro, et une valeur de un bit (0 ou vide) qui sera extraite par la station de base. Quand les paquets arrivent à la station de base, elle les arrange selon leurs numéros de séquence et extrait la valeur du bit de chaque paquet pour reconstruire un *Watermark* w_n'. Par la suite, la station de base compare le *Watermark* w_n original au *Watermark* w_n'. Si la perte dépasse un taux défini, le processus de détection commence. Les algorithmes d'ajout et d'extraction des *Watermark* w_n' aux paquets sont représentés ci-dessous :

- Algorithme d'intégration du Watermark par le nœud source:

Entrée: Les données d_n qui sont prêtes à envoyer.
Sortie: Les données d_n' avec le Watermark.
Etapes à suivre:

1) int i = 0
2) pour (i <n; i + +)
3) si (w_i = = "0")
4) $d_i^{'} = d_i$
5) Sinon, si (w_i = ="1")
6) $d_i^{'} = d_i +$ " "

- Algorithme d'extraction du Watermark par la station de base:

Entrée: Les données $d_n^{'}$ arrivent à la station de base
Sortie: La Watermark $w_n^{'}$
Etapes à suivre:

1) Pour (i = 0; i <m; i + +)
2) si (le dernier bit de la $d_i^{'}$ est " ")
3) $w_n^{'} = w_n^{'} +$ "1"
4) Sinon
5) $w_n^{'} = w_n^{'} +$ "0"

Si la valeur du dernier bit du paquet à transmettre est '1', le nœud source ajoute le symbole vide '' au dernier bit du paquet. Par contre, si la valeur du dernier bit de données est «0 », le nœud source n'ajoute rien au paquet. La station de base reconstruit le *Watermark* $w_n^{'}$ en remplaçant le

vide par «1 ». Une fois le *Watermark* $w_n^{'}$ reconstruit, la station de base compare la valeur initiale avec la valeur reçue (figure 2.17).

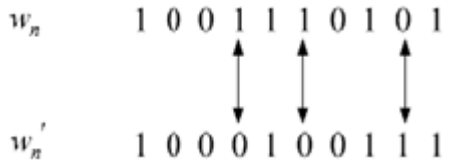

Figure 2.17. *Watermark* avec K = 10.

Si la station de base reçoit un taux de paquets perdus (calculé par la technique du *Watermark)* plus grand que la perte normale des paquets, le mode de détection commence. La station de base transmet un avis et un nouveau *Watermark* w_n à tous les nœuds du chemin. La détection débutera sur le chemin entre la station de base et le nœud source. Chaque nœud sera responsable de vérifier lui même la séquence *Watermark* $w_n^{'}$. Quand un nœud reçoit les paquets, il extrait la séquence *Watermark* $w_n^{'}$ des paquets, au lieu de les retransmettre immédiatement. Le nœud compare le *Watermark* $w_n^{'}$ extrait avec le *Watermark* w_n original généré par cette méthode proposée dans le système de détection des attaques. Si la détection des paquets perdus est plus grande que la valeur normale, le nœud précédent est considéré comme le nœud malicieux qui lance l'attaque de retransmission sélective. La valeur de confiance du nœud soupçonné sera diminuée et le nœud source choisira un autre chemin sécurisé pour transmettre des paquets. S'il y a plus que deux nœuds malicieux sur le chemin, le nœud malicieux le plus proche de la station de base sera détecté. L'autre sera détecté dans la prochaine séquence de détection.

L'influence du taux normal de perte de paquets est donnée dans la figure 2.18:

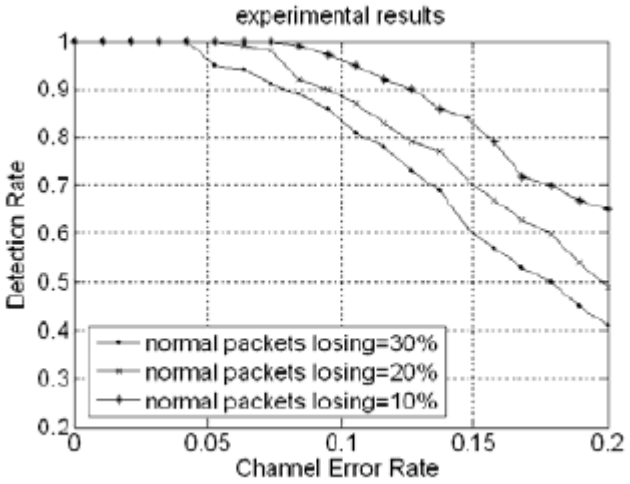

Figure 2.18. Probabilité de détection de l'attaque de retransmission sélective [20].

Si la perte des paquets est le résultat des collisions causées par le canal, une augmentation du taux de paquets perdus va diminuer le taux de détection. Selon les résultats de l'expérience, lorsque la perte normale des paquets est supérieure à 30%, le taux de détection est inférieur à 40%.

Le coût énergétique de la détection:

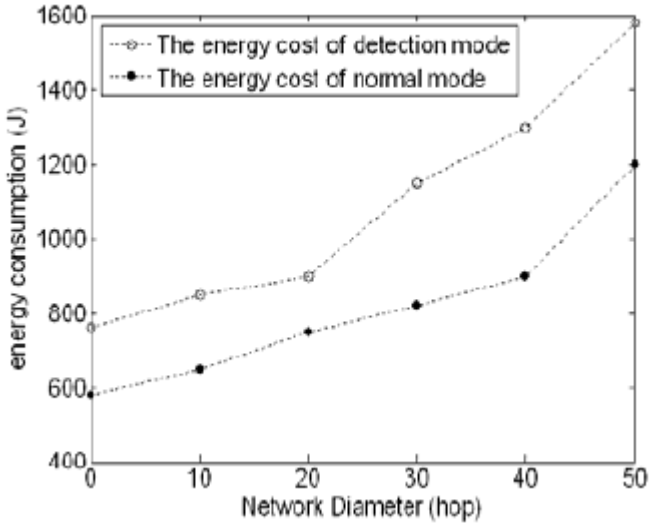

Figure 2.19. Comparaison du coût énergétique entre le mode normal et le mode de détection [20].

La figure 2.19 nous montre que si le mécanisme de détection n'est pas démarré par la station de base, le coût d'énergie est de 400 Joules pour les nœuds source, incluant l'intégration des messages *Watermarks* dans chaque groupe de paquets.

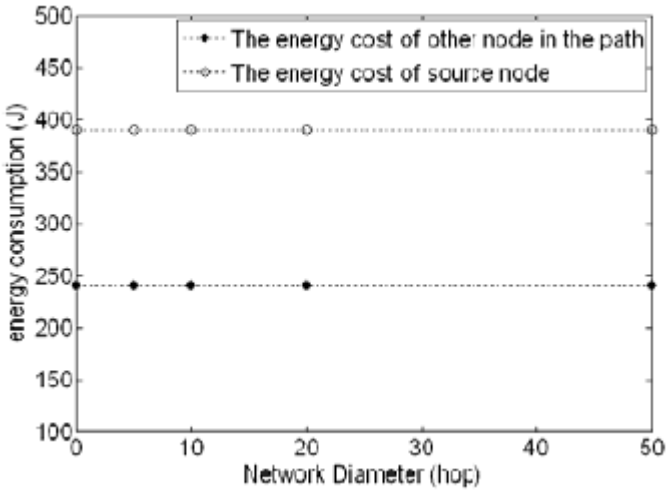

Figure 2.20. Comparaison du coût énergétique entre le nœud source et un nœud intermédiaire [20].

Si la station de base détecte une attaque de retransmission sélective sur le chemin, elle commence le mécanisme de détection. Etant donné que chaque nœud doit extraire le *Watermark* et comparer les paquets perdus, le coût de l'énergie va augmenter. Le coût en mode de détection est 400 Joules de plus que l'état normal, le double (figure 2.20).

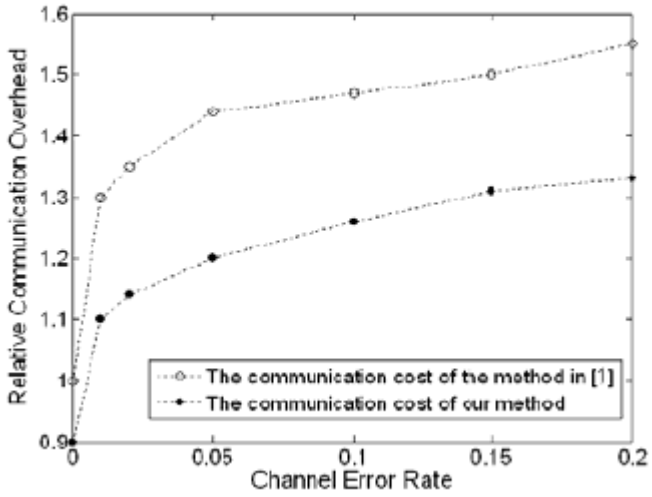

Figure 2.21. Évaluation du coût de transmission [20].

Par rapport à la première approche proposée par Yu et Xiao (2006), les auteurs ont trouvés à partir de leur analyse pratique (figure 2.21) que cette approche reste plus efficace en termes de coût de transmission au niveau des nœuds de capteurs sans fil.

En conclusion, ce système augmente légèrement le coût de transmission quand la perte de paquets causée par les collisions ne dépasse pas le taux normal du réseau. Par contre, s'il y a plus de deux nœuds malicieux dans le chemin, il faut lancer la détection plusieurs fois pour trouver tous les nœuds malicieux. Ainsi, afin d'identifier les nœuds malicieux, l'extraction des *Watermark*s permettra un retard de transmission des paquets *Rapport*. De même, le protocole de transmission géographique utilisée est un algorithme de routage qui consomme beaucoup d'énergie.
Finalement cette approche manque d'efficacité et contient des failles de sécurité vu que tous les nœuds utilisent la même séquence *Watermark*. De plus, un nœud compromis peut affirmer qu'un nœud légitime est malicieux et peut facilement rejeter les paquets provenant de la station de base sans être détecté.

Suite à l'analyse des méthodes actuelles de détection de l'attaque de retransmission sélective, nous sommes prêts à présenter notre hypothèse de recherche. Nous incorporerons les avantages de chacune des méthodes présentées dans ce chapitre et éviterons leurs failles afin de proposer une

nouvelle méthode impliquant la création de faux rapports et permettant une identification plus efficace et moins coûteuse de l'attaque de retransmission sélective.

CHAPITRE 3. Détection de la retransmission sélective sur les réseaux de capteurs

L'envoi des paquets sous forme de plusieurs copies à travers plusieurs chemins possibles assure une livraison de paquets plus sûre que l'envoi d'une copie à travers un seul chemin. Elle présente par contre deux inconvénients:

- Consommation supplémentaire d'énergie
- Manque d'identification des nœuds malicieux.

L'envoi des paquets à travers un seul chemin est plus efficace pour la consommation de l'énergie, mais il est moins sûr que les paquets vont atteindre leur destination à cause des collisions, de la qualité de radio dans certains environnements, ou bien d'une attaque de retransmission sélective. Ici, nous introduisons un nouveau système simple et souple capable de détecter l'attaque de retransmission sélective en utilisant des faux rapports de données.

Tout d'abord, notre approche est différente de toutes les autres approches précédentes. Ici, même si tous les nœuds entourant la station de base sont des nœuds malicieux, la station de base peut détecter l'attaque de retransmission sélective. Le problème dans la détection de l'attaque de retransmission sélective est que la station de base ne sait pas si elle va recevoir des paquets et quand elle les recevra. Dans l'approche *CADE* [15], les auteurs ont proposé que les nœuds source envoient des paquets de notification avant l'envoi des paquets *Rapport*, ce qui permet à la station de base de savoir le nombre de paquets *Rapport* que les nœuds source vont envoyer. Cette méthode proposée par *CADE* [15] exige l'envoi de deux paquets (notification de la part du nœud source et le paquet du chemin envoyé par la station de base) avant l'envoi du paquet *Rapport* ce qui n'est pas pratique au niveau du coût de transmission. En plus, s'il existe des nœuds malicieux entourant la station de base, l'attaque sera presque impossible à détecter. Dans notre approche, nous proposons l'utilisation des faux paquets *Rapport*. Les nœuds source envoient des faux paquets *Rapport* parmi les paquets *Rapport* normaux (*NReport*) à la station de base. Les motifs derrière cette idée sont les suivants : Premièrement, tous les paquets sont chiffrés avec une clef partagée entre chaque nœud et la station de base (une clef différente pour chaque nœud), ce qui

signifie qu'un nœud intermédiaire malicieux ne peut pas distinguer entre un faux et un vrai paquet *Rapport* et par la suite s'il lance l'attaque de retransmission sélective, il détruira avec une bonne probabilité des faux paquets *Rapport*. Deuxièmement, la station de base est supposée avoir beaucoup plus d'énergie et de puissance de calcul et de mémoire que les nœuds de capteurs présents dans le réseau. Nous pouvons profiter de cette ressource pour lui permettre de prédire le nombre de faux paquets *Rapport* provenant de chaque capteur selon la méthode que nous proposons. Troisièmement, le coût énergétique additionnel est contrôlable vu qu'il est dépendant du nombre des faux paquets *Rapport* envoyés.

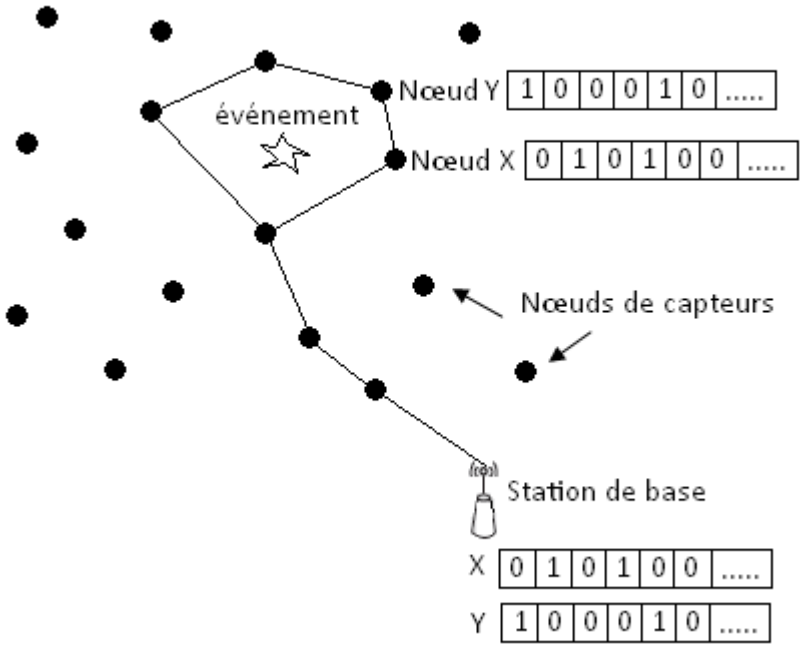

Figure 3.1. L'envoi des parquets *NReport* et *FReport*.

Les faux paquets de *Rapport*, appelés *FReports*, sont envoyés selon une séquence de bits (figure 3.1) générée pseudo-aléatoirement à partir de l'algorithme *LSFR* [23], connue par les nœuds source et la station de base. Quand la valeur du bit est '1', le nœud envoie un faux paquet de *Rapport* (*FReport*). Quand la valeur du bit est '0', le nœud envoie un vrai paquet de *Rapport* (*NReport*) s'il y a eu un événement, sinon le nœud n'envoie rien. Étant donné que les nœuds intermédiaires malicieux ne peuvent pas distinguer entre les faux et les vrais paquets *Rapport*,

l'attaque de retransmission sélective peut aussi bien affecter les faux paquets *FReports* attendus par la station de base. Cette dernière compare les *FReports* arrivant avec la séquence de bits pseudo-aléatoires et vérifie si la perte des *FReports* dépasse le taux de perte permis. Les *FReports* sont envoyés plusieurs fois car un seul *FReport* risque de ne pas atteindre la station de base à cause du problème de collisions des paquets.

À cause des limitations de ressources dans les capteurs sans fil, nous proposons la génération des séquences pseudo-aléatoires à partir de l'algorithme *LSFR* [23] (*Linear Feedback Shift Register*). Il s'agit d'un registre qui peut être chargé pour définir sa valeur, ou décalé pour générer une sortie d'un seul bit et pour changer la valeur du registre. Une fois que le registre est chargé avec une séquence de bits initiale (clef), il suffit de copier chaque bit à la case voisine de sa droite pour exécuter un *LFSR Fibonacci*. Le bit d'origine le plus à droite est considéré comme la sortie. L'entrée du côté gauche est la parité (Ex : OR, NOR, XOR, XNOR, etc.) d'un sous-ensemble de cases de bits spécifiques du registre (voir Figure 3.2 ci-dessous).

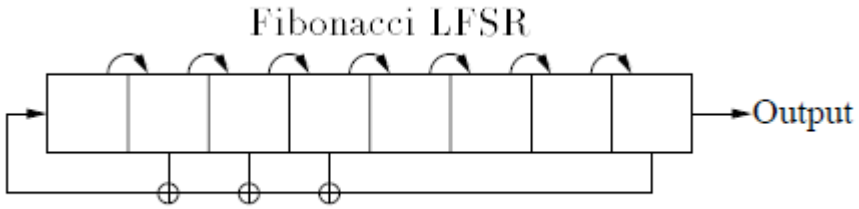

Figure 3.2. Générateur pseudo-aléatoire LFSR.

Si les cases sont bien choisies, le déplacement (à partir de toute valeur non nulle) répété permet de faire défiler toutes les valeurs possibles non nulles du registre. Le nombre maximal des valeurs possibles non nulles est égale à 2^n–1, où n est le nombre d'éléments dans le registre *LSFR*. Peterson et Weldon [23] ont compilé des tables de *LSFR* de longueur maximale à laquelle les concepteurs peuvent se référer.

Les *LSFR* [23] sont des simples générateurs de séquences pseudo-aléatoires qui demandent un coût énergétique bas ce qui est souhaitable dans notre scenario. Lorsque les sorties des cases du registre sont chargées avec une valeur de départ et lorsque le *LSFR* est cadencé dans les nœuds de capteurs et la station de base, il va générer des séquences pseudo-aléatoires contenant

exactement $2^{(n-1)}$ de '1' et $2^{(n-1)} - 1$ de '0'. Ces séquences pseudo-aléatoires seront utilisées pour l'envoi des *FReport* par les nœuds de capteurs sans fil pendant la phase de surveillance.

Exemple: Selon le tableau des valeurs maximales des compteurs *LFSR* [23], avec une séquence de longueur L=137 bits nous pouvons générer une autre séquence de longueur L'=$2^{137} - 1$ bits pseudo-aléatoires, contenant 2^{136} de '1' bits et $2^{136} - 1$ de '0' bits (≃ 50% de '1' et 50% de '0'), avant d'avoir une répétition. Le bit d'entrée dans cet exemple sera le résultat de la fonction logique d'équivalence (XNOR) à deux entrées, 116 et 137.

XNOR

Entrée		Sortie
A	B	A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

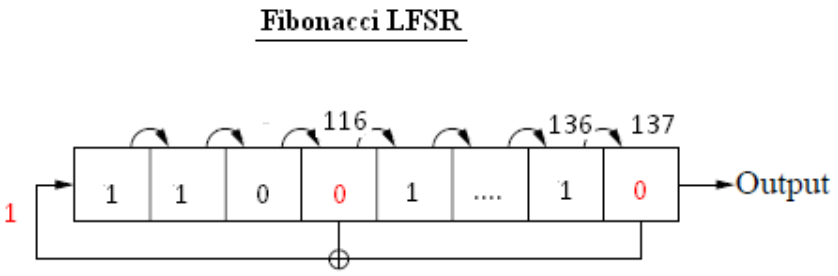

Figure 3.3. Générateur pseudo-aléatoire *LFSR* à partir de la fonction logique (XNOR) à deux entrées.

Dans l'exemple de la figure 3.3 ci-dessus, nous avons choisi une fonction logique à deux entrées (116 et 137) dans le but de minimiser le coût énergétique provenant des différentes opérations. Toutefois, d'autres fonctions logiques (Ex : OR, NOR, XOR, XNOR, etc.) avec plus que deux entrées peuvent être utilisées.

Pour réaliser notre système, une clef unique secrète est partagée entre la station de base et chaque nœud (chaque nœud possède une clef différente) pour chiffrer les vrais paquets *Rapport* appelés *NReports* et les faux paquets *Rapport* appelés *FReports* et pour authentifier les capteurs expéditeurs. Ce chiffrement empêche les nœuds intermédiaires de distinguer entre les deux types de paquets. De plus, le temps doit être synchronisé entre la station de base et les nœuds de capteurs pour vérifier le temps de transmission des paquets sur un chemin donné de la part de la station de base. Nous pouvons trouver ces options dans la plupart des protocoles de routage et gestion de clefs comme *SEEM* [12] et *LEAP* [22].

Comme dans l'approche *CADE* [15], et pendant la phase de déploiement supposée sécuritaire, la station de base construit la topologie du réseau à l'aide de l'outil *SEEM* [12]. Elle transmet un message appelé *ND (Neighbor Discovery)* à tous les nœuds de capteurs du réseau. Chaque nœud qui reçoit un *ND*, enregistre dans sa liste de voisins l'adresse de l'expéditeur de ce *ND*. Si le numéro de séquence du *ND* est nouveau, le nœud le retransmet. Une fois la liste de voisins remplie, chaque nœud génère plusieurs copies de cette liste et les envoie à la station de base par tous les chemins possibles. Quand la station de base reçoit toutes les listes de voisins de tous les nœuds du réseau, elle déduit les meilleurs chemins possibles pour l'envoi des paquets et associe chaque nœud à une liste de chemins de transmission.
Maintenant que la topologie est connue par la station de base, elle commence à accueillir les *Tpacket*. Un *TPacket* est un paquet de test envoyé par chaque nœud déployé à la station de base pour calculer le temps nécessaire pour que les paquets correspondant au nœud atteignent la station de base d'une part, et pour fixer la date d'envoi du premier *FReport* d'autre part. Puisque le temps entre la station de base et les nœuds du réseau est synchronisé, la station de base enregistre le temps requis pour tous les paquets provenant d'un nœud expéditeur spécifique vers la station de base. À l'aide de cette méthode, la station de base peut savoir combien de temps associé à un paquet provenant d'un nœud X, est nécessaire pour atteindre la station de base à travers un chemin C. Ceci sera utilisé pour la détection des différentes attaques comprenant l'attaque de retransmission sélective, *Wormhole*, modification et retransmission des données. Avant que la phase de déploiement soit terminée, la station de base partage une séquence aléatoire de L bits (valeur de départ pour le *LSFR* [23]) avec chaque nœud de capteurs (chaque nœud une séquence initiale différente) et attend le premier paquet *FReport* provenant de chaque nœud du réseau tel qu'il est défini dans les *TPackets* précédents.

Pendant la phase de surveillance, Nous supposons que la station de base ne peut pas être compromise. Tous les nœuds intermédiaires sont possiblement compromis une fois la phase de déploiement terminée. Le temps du réseau est synchronisé et tous les paquets portent leur temps d'émission, ce qui permet à la station de base de mettre à jour le temps nécessaire pour que les paquets correspondant au nœud source atteignent la station de base. Chaque paquet contient l'information sur son chemin traversé quand il atteint la station de base.

Notre système peut s'accommoder de toutes les approches de routage à chemin unique. Étant donné que chaque nœud de capteurs est capable de générer des séquences de bits pseudo-aléatoires selon la méthode *LFSR* [23] (*Linear Feedback Shift Register*) en même temps que la station de base, qui s'attend à recevoir des *FReport* à un temps précis provenant d'un nœud X selon l'exemple dans la figure 3.4 ci-dessous :

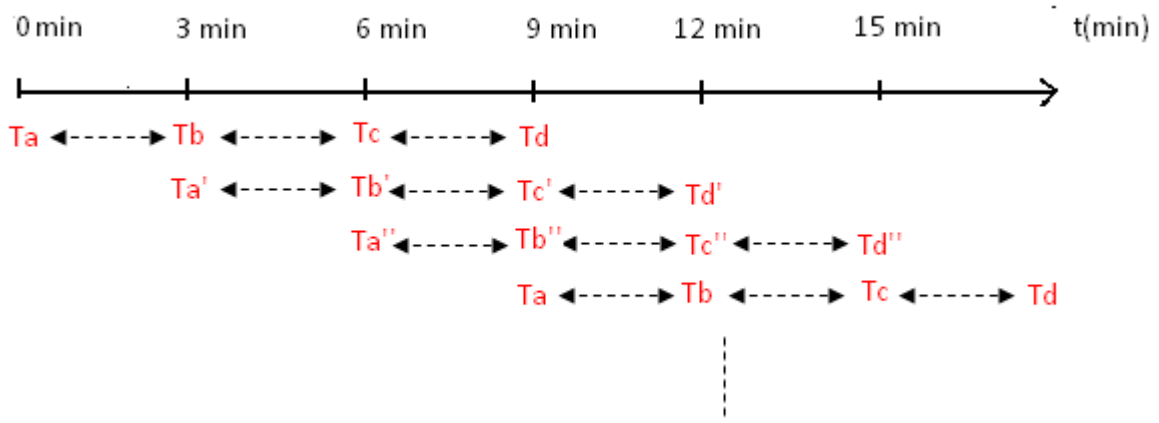

Figure 3.4. Périodes du mode de détection.

Tn: Temps réel.

N_N: Nombre de *NReports* envoyés par le nœud à la station de base entre Ta et Tb.

N_F : Nombre de *FReports* envoyés à la station de base entre Tc et Td.

P : Valeur fixe de la période entre Ta, Tb, Tc et Td

Θ: Valeur prédéterminée identifiant le pourcentage du *surchargement de communication* désiré

Trois processus (Ta → Td, Ta'→Td', Ta''→Td'') sont exécutés en parallèle avec un décalage d'une période P prédéfinie (Ex : 3 min). Quand le processus se termine, il recommence de nouveau. Chaque processus est constitué de trois parties (Ta → Td : Ta → Tb, Tb → Tc, Tc → Td) expliqués ci-dessous :

<u>Entre Ta et Tb (même démarches entre Ta' et Tb', Ta'' et Tb''):</u>

- Pendant une période P prédéfinie (Ex : 3 min), chaque capteur va enregistrer le nombre N_N de vrais rapports *NReports* envoyés vers la station de base :

$N_N = 0$;

Répéter

Si un nœud envoie un vrai paquet Rapport à la station de base ;

N_N ++ ;

Jusqu'à (T_n-T_a = P)

$N_F = N_N * \Theta$;

Une fois que le nombre N_N de vrais rapports *NReports* envoyés vers la station de base est enregistré entre Ta et Tb, le nombre N_F de faux rapports *FReports* sera facilement déduit.

Entre Tb et Tc (même démarches entre Tb' et Tc', Tb'' et Tc''):

- Chaque capteur utilise les paquets *FReports* pour envoyer à la station de base le nombre N_N de vrais rapports *NReports* envoyés entre Ta et Tb.
- Chaque capteur génère en parallèle avec la station de base une séquence pseudo-aléatoire de bits dont le nombre de bits '1' est égal à la variable N_F tout en formant près de 25% (Θ) de la taille totale de la séquence *Tab3*.

AND

Entrées		Sortie
a	**b**	**L**
0	0	0
0	1	0
1	0	0
1	1	1

Figure 3.5. Table de vérité AND.

Étant donné que les générateurs de séquences pseudo-aléatoires *LFSR* [23] génèrent des séquences contenant approximativement 50% de '1' et 50% de '0', nous utilisons deux générateurs de séquences *Tab1* et *Tab2* pour alimenter les deux entrées de la fonction logique AND, afin générer une séquence pseudo-aléatoire finale *Tab3* dont le nombre de bits '1' est égal à la variable N_F tout en formant près de 25% de la taille totale de la séquence *Tab3*.

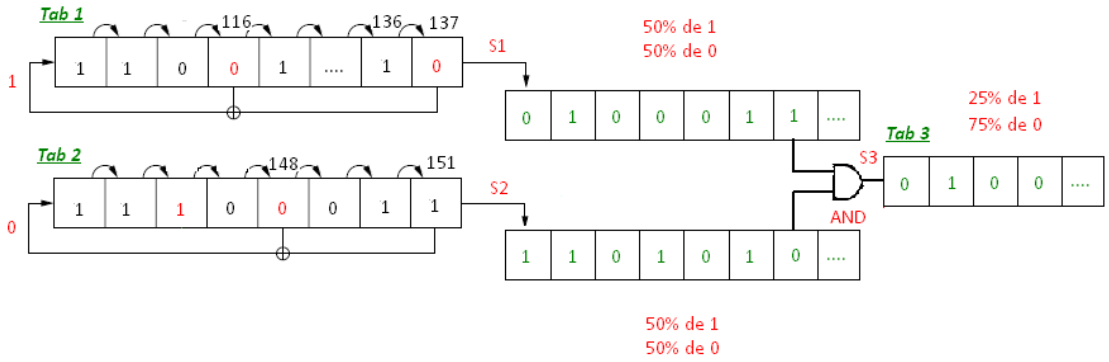

Figure 3.6. Génération de la séquence pseudo-aléatoire.

Dans la figure 3.6 ci-dessus, nous avons choisi une fonction logique à deux entrées (116,137) pour le premier générateur pseudo-aléatoire *Tab1* de longueur 137 et (148,151) pour le deuxième générateur *Tab2* de longueur 151 dans but de minimiser le coût énergétique provenant des opérations de calcul. Ces deux générateurs vont sortir *S1* et *S2* pour former deux séquences pseudo-aléatoires contenant 50% de '1' bits chaque. Ces deux séquences représenteront les deux entrées de la fonction AND. Chaque sortie *S3* de la fonction AND sera comparée à la variable N_F. Une fois le nombre de '1' bits générés $N_{(1)}$ est égal à N_F, la séquence finale *Tab3* sera utilisée pour l'envoi des *FReports*. Ce processus se répète à chaque période P mais avec un N_F différent.

Répéter

Tab1 [n] AND Tab1 [m];

$>>_1$ Tab1; // Décalage des bits à droite

Tab2 [K] AND Tab2 [L];

$>>_1$ Tab2; // Décalage des bits à droite

$S_3 = S_1$ AND S_2;
Tab3 [0] = S_3;

Si (S_3 = '1');
$N_{(1)}$ ++;

Jusqu'à $N_{(1)} \geq N_F$

Entre Tc et Td (même démarches entre Tc' et Td', Tc'' et Tc''):

- Le nœud fixe un chronomètre de valeur T=P/ Nombre total de bits de *Tab3*.
- A chaque période T et selon la séquence pseudo-aléatoire *Tab3,* un nœud envoie un paquet N_F de *FReport* vers la station de base si la valeur du bit est égale à '1'. Les vrais paquets de *Rapport* sont envoyés seulement quand la valeur du bit est égale à '0'.

Répéter à chaque période T

Si Tab3 [] = '1' ;

Envoi d'un FReport à la station de base ;

Si Tab3 [] = '0' ;

Si un Nreport existe dans le cache

Envoi d'un NReport à la station de base ;

Sinon

Pas d'action;

Jusqu'à fin du Tab3 []

Si un nœud malicieux est prêt à lancer une attaque de retransmission sélective, il ne peut pas distinguer entre les vrais rapports *NReport* et les faux rapports *FReport* envoyés par un nœud X à la station de base, et donc il prend donc le risque d'être détecté s'il rejette les paquets. Sachant que la collision peut provoquer des pertes de paquets et que le pourcentage d'erreur est le même pour les paquets *Nreport* et *FReport*, nous estimons que si plus qu'un pourcentage Θ des *FReport* provenant d'un nœud X n'atteignent pas la station de base à l'heure prévue, la station de base va déduire une activité malicieuse sur le chemin. En analysant tous les *FReports* entrants de tous les nœuds du réseau, la station de base sera capable de déduire quel est le nœud malicieux. La probabilité de détection est représentée par : $P_D = N_F / (N_N + N_F) = \Theta / 1 + \Theta$.

Étant donné le coût de transmission total prédéfini avant le déploiement du réseau, il peut être fixé par exemple à 1.25 * la consommation normale d'énergie, c'est-à-dire parmi 125 paquets de données envoyés entre Tc et Td (ex : 3 min), il en y a 100 qui sont des paquets de données normaux *NReport* et 25 qui sont des faux paquets *Report*. Ceci cause un coût de transmission

additionnel de 25% par rapport au coût de transmission normale, ce qui est très raisonnable. Par contre, un adversaire sera capable d'éliminer 80% du nombre total des paquets sans détection. N'oublions pas que l'augmentation du nombre de *FReports* envoyés selon un pourcentage prédéfini augmente les collisions des paquets ainsi que la perte de données. Si nous comparons les résultats de cette approche avec l'approche proposée par Yu et Xiao [1,8], on trouve que le coût de transmission de notre système est 25% plus petit que l'autre.
Notre approche est beaucoup plus efficace si elle est utilisée dans des applications qui demandent l'envoi des rapports à haute fréquence, étant donné que l'envoi des paquets *FReport* est proportionnel aux paquets *NReport*, ce qui augmente les chances de détection des menaces.

En conclusion, il s'agit d'une méthode avec un coût de transmission contrôlable. Ce schéma simple est compatible avec n'importe quel protocole de routage, même celui à chemins multiples « *multipath* ». Le grand avantage de la méthode d'utilisation de faux rapports, qui n'existe dans aucune des approches précédentes, est que même si les nœuds entourant la station de base sont tous compromis, la station de base détectera l'attaque. De même, notre système est capable de détecter si un nœud malicieux transmet des paquets sur un autre chemin sur le réseau pour provoquer des retards de données. De plus, l'approche est également efficace contre la modification ou la retransmission retardée des paquets.
Par contre, la détection peut être retardée pendant que la station de base analyse les faux paquets *FReports* entrants. De plus, l'approche proposée demande une synchronisation de temps entre la station de base et les nœuds.
Finalement, la méthode est fiable et offre beaucoup d'avantages pour les applications qui demandent l'envoi de rapports à haute fréquence comme par exemple l'analyse du trafic, mais moins efficace pour les applications qui demandent l'envoi des rapports à basse fréquence comme les réseaux de détection du feu dans une forêt.

CHAPITRE 4. Tableau de comparaison des approches

	Multipath [28]	Detecting selective forwarding attaque [1]	CHEMAS [2]	Two-hops Neighbor Knowledge [3]	CADE [4]	Watermark [5]	Neighbor Watch System [6]	Faux Rapports
Identification du nœud malicieux	NON	OUI	OUI	OUI	OUI	OUI	NON	OUI
Synchronisation du temps	Pas Obligatoire	Obligatoire	Obligatoire	Pas Obligatoire	Pas Obligatoire	Pas Obligatoire	Pas Obligatoire	Obligatoire
Fonctions OHC	Pas Obligatoire	Obligatoire	Obligatoire	Pas Obligatoire	Pas Obligatoire	Pas Obligatoire	Pas Obligatoire	Pas Obligatoire
Duplication des paquets	Données	Aucun	Aucun	Aucun	Data-Reply	Aucun	Après une perte d'un paquet de données	Aucun
Génération des Paquets *ACK*	Aucun	Toujours	Toujours	Aucun	Après une perte d'un paquet de données	Aucun	Aucun	FReport
Topologie statique	NON	OUI	OUI	OUI	OUI	OUI	OUI	OUI

La comparaison de notre système avec les systèmes précédents est donnée dans le tableau ci-dessus. Comme nous l'avons mentionné plus tôt, les approches *multi-path* et *NWS* ne sont pas capables d'identifier les nœuds malicieux. De même, plusieurs approches comme *multi-path, watermark, Detecting selective forwarding attaque* [1] et *CHEMAS* [21] ne détectent pas la retransmission retardée des paquets.

Dans notre système la station de base a besoin de synchroniser son temps avec le reste du réseau, par contre la notion de porte-clefs à sens unique pour l'authentification de réception n'existe pas,

ni la duplication des paquets. Finalement la topologie de toutes les approches proposées est statique à l'exception de l'approche *Multipah* [28] qui permet aux réseaux de capteurs sans fil de fonctionner dans des topologies mobiles.

CHAPITRE 5. Conclusion

La contrainte des ressources dans les nœuds de capteurs reste un problème majeur devant la sécurité des réseaux de capteurs sans fil. Plus de recherches au niveau de l'amélioration de la capacité de ces petits capteurs et au niveau du chiffrement à clef publique doivent prendre place. On a introduit dans ce papier les réseaux de capteurs sans fil, leur utilité, leur mode de fonctionnement, ainsi que les différentes attaques qui peuvent leur causer des dommages. La sécurité dans ces réseaux est une tâche très importante, capable de dégrader énormément l'efficacité et la fiabilité des applications si elle n'est pas prise en considération. Dans ce contexte, nous avons proposé une approche simple capable de détecter l'une des plus dangereuses menaces contre les réseaux de capteurs sans fil, l'attaque de retransmission sélective, à partir des faux messages. Plusieurs approches existent pour lutter contre cette attaque, mais aucune ne présente une solution sûre et efficace capable de protéger le réseau avec un coût de communication raisonnable. Dans ce mémoire, nous avons présenté un nouvel algorithme simple qui peut être appliqué sur la plupart des applications de réseaux de capteurs sans fil tout en consommant moins d'énergie que les approches existantes tout en augmentant le niveau de sécurité.

CHAPITRE 6. Bibliographie

[1] B. Yu and B. Xiao, "*Detecting selective forwarding attacks in wireless sensor networks*". In Proc. 20th International Parallel and Distributed Processing Symposium IPDPS 2006 (SSN2006), pp.1-8, 2006.

[2] C. Y. Wan, A. T. Campbell and L. Krishnamurthy, "*PSFQ: A reliable transport protocol for wireless sensor networks*". In Proc. of the first ACM International Workshop on Wireless Sensor Networks and Applications, pp. 1-11, 2002.

[3] C. Intanagonwiwat, R. Govindan and D. Estrin, "*Directed diffusion: A scalable and robust communication paradigm for sensor networks*". In Proc. of ACM MobiCom, pp. 56-67, 2000.

[4] L. Lamport, "*Constructing digital signatures from one-way function*". In technical report SRI-CSL-98, SRI International, pp. 1-7, 1979.

[5] J. Deng, R. Han and S. Mishra, "*Defending against Path based DoS Attacks in Wireless Sensor Networks*". In Proc. The 3rd ACM on the Security of Ad Hoc and Sensor Networks (SASN 2005), pp. 89-96, 2005.

[6] S. Zhu, S. Setia, S. Jajodia and N. Peng, "*An interleaved hop-by-hop authentication scheme for filtering of injected false data in sensor network"*. In Proc. of IEEE Symposium on Security and Privacy, pp. 259-271, 2004.

[7] A. Perrig, R. Szewczyk, V. Wen, D. Culler and D. Tygar, "*SPINS: security protocols for sensor networks*". In Proc. of ACM MobiCom, pp. 189–199, 2001

[8] B. Xiao, B. Yu, and C. Gao, "*CHEMAS: Identify suspect nodes in selective forwarding attacks*". Journal of Parallel and Distributed Computing archive, Vol. 67, Issue: 11, pp. 1218-1230, 2007.

[9] S. Capkun and J. Hubaux, "*Secure positioning of wireless devices with application to sensor networks*". In Proc. of IEEE InfoCom, pp. 1917–1928, 2005.

[10] Y. Wang, "*A survey of security issues in wireless sensor networks*", IEEE Communication Surveys, Vol. 8, Issue: 2, pp. 2-23, 2006.

[11] D. Djenouri and L. Khelladi and A. Badache. N, "*A survey of security issues in mobile ad hoc and sensor networks*". Communications Surveys & Tutorials, IEEE, Vol. 7, Issue :4, pp. 2- 28, 2006.

[12] N. Nasser and Y. Chen, "*SEEM: Secure and energy efficient multipath routing protocol for wireless sensor networks*". Computer Communications, Vol. 30, Issue: 11-12, pp. 2401-2412 , 2007.

[13] W. Du, and J. Deng, "*A pairwise key pre-distribution scheme for wireless sensor networks*". In Proc. of 10th Computer and Communications Security, pp. 42-51, 2003.

[14] L. Eschenaur, and V. Gligor, "*A Key-management scheme for distributed sensor networks*". In Proc. of the 9th ACM conference on Computer and Communications Security, pp. 41-47, 2002.

[15] K.Young, "*CADE: Cumulative acknowledgement based detection of selective forwarding attacks in wireless sensor networks*". In Proc of the Third 2008 International Conference on Convergence and Hybrid Information Technology, pp.416-422, 2008.

[16] K. Ioannis et al, "*Toward intrusion detection in sensor networks*", 13th European Wireless Conference, Paris, pp.1-7, 2007.

[17] T. H. Hai and E.-N. Huh, "*Detecting selective forwarding attacks in wireless sensor networks using two-hops neighbor knowledge*". In Proc. of the 2008 Seventh IEEE International Symposium on Network Computing and Applications, pp. 325-331, 2008.

[18] N. Abu-Ghazaleh, K. Kang, and K. Liu, "*Towards resilient geographic forwarding in wireless sensor networks*", In Proc. of the 1st ACM International Workshop on Quality of Service & Security in wireless and mobile networks, pp: 71–78, 2005.

[19] S.-B. Lee and Y.-H. Choi, "*A resilient packet forwarding scheme against maliciously packet dropping nodes in sensor networks*". SASN '06: In Proc. of the Fourth ACM Workshop on Security of ad hoc and sensor networks, ACM Press, pp. 59–70, 2006.

[20] H. Deng, X. Sun, B. Wang and Y. Cao, "*Selective forwarding attack detection using watermark in WSNs*". Computing, Communication, Control, and Management, 2009. CCCM 2009. ISECS International Colloquium, pp.109-113, 2009.

[21] S. Marti, T.J. Giuli, K. Lai, and M. Baker, Mitigating, "*Routing Misbehavior in Mobile Ad Hoc Networks*". ACM/IEEE International Conference on Mobile Computing and Networking, pp. 255-265, 2000.

[22] S. Zhu, S. Setia, and S. Jajodia, "*LEAP: Efficient Security Mechanisms for Large-Scale distributed Sensor Networks*". In Proc of The 10th ACM Conference on Computer and Communications Security (CCS '03), pp. 62-72, 2003.

[23] P.Alfke, "*Efficient shift registers, LFSR counters and long pseudo-random sequence generators*". In Proc. Xilinx Application Note, pp.1-6, 1996.

[24] M. Gaurav, D. Peter, G. Deepak, and S. Prashant, "*Capsule: an energy-optimized object storage system for memory-constrained sensor devices*". In Proc. of The 4th International Conference on Embedded Networked Sensor Systems Boulder, Colorado, USA: ACM, pp. 195-208, 2006.

[25] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. Pister, "*System architecture directions for networked sensors*". In Architectural Support for Programming Languages and Operating Systems, pp. 93–104, 2000.

[26] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "*A survey on sensor networks*". IEEE Communications Magazine, Vol. 40, Issue: 8, pp. 102–114, Août 2002.

[27] J. A. Stankovic, T. E. Abdelzaher, C. Lu, L. Sha, and J. C. Hou, "*Real-time communication and coordination in embedded sensor networks*", In Proc. of the IEEE, Vol. 91, pp. 1002-1022, 2003.

[28] C. Karlof and D. Wagner, "*Secure routing in wireless sensor networks: attacks and countermeasures*", In Proc. 1st IEEE Int'l. Wksp. Sensor Network Protocols and Applications (SNPA'03), pp. 293-315, 2003.

[29] A. Wood and J. Stankovic, "*Denial of service in sensor networks*", IEEE Computer Mag., Vol. 35, Issue: 10, pp. 54–62, 2002.

[30] J. R. Douceur, "*The sybil attack*". In 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), pp.1-6, 2002.

[31] W. Yong, G. Attebury, and B. Ramamurthy, "*A survey of security issues in wireless sensor networks"*. Communications Surveys & Tutorials, IEEE, Vol. 8, pp. 2-23, 2006.

[32] L. Eschenauer and V. Gligor, "*A key-management scheme for distributed sensor networks*". In Proc. of the 9th ACM Conference on Computer and Communication Security (Washington, D.C., Nov.). ACM Press, New York, pp. 41–47, 2002.

[33] D. J. Malan, M. Welsh, and M. D. Smith, "*A public-key infrastructure for key distribution in tinyos based on elliptic curve cryptography*". In Proc. IEEE SECON, Santa Clara, CA, pp. 71–80, 2004.

[34] D. Liu and P. Ning, "*Establishing pairwise keys in distributed sensor networks*". In Proc. of 10th ACM Conference on Computer and d Communications Security (CCS'03), pp. 52–61, 2003.

[35] N. Gura, A. Patel, and A.Wander, "*Comparing elliptic curve cryptography and RSA on 8-bit CPUs*". In Proc. of the 2004 Workshop on Cryptographic Hardware and Embedded systems (CHES), pp.2-15, 2004.

[36] D. Malan, M. Welsh, and M. Smith, "*A public-key infrastructure for key distribution in TinyOs based on elliptic curve cryptography*". In Proc. of IEEE Conference on Sensor and Ad Hoc Communications and Networks (SECON), pp. 71–80, 2004.

[37] H. Wang and Q. Li. Efficient, "*Implementation of public key cryptosystems on mote sensors*". In Proc. of International Conference on Information and Communication Security (ICICS), pp. 519–528, 2006.

[38] MICAz: Wireless measurement system. URL: www.xbow.com/Products/Product_pdf_11files/Wireless_pdf/MICAz_Datasheet.pdf. Consulté le 4 Mars 2011.

[39] T.H. Hai, E.N. Huh, "*Optimal selection and activation of intrusion detection agents for wireless sensor networks*". In Proc. of International Conference on Future Generation Communication and Networking, pp. 350-355, 2007.

[40] K. Holger, W. Adreas W, "*Energy consumption of sensor nodes*". Protocols and Architecture for Wireless Sensor Networks, John Wiley & Son Press, Chapter 2.2, 2005.

[41] N. Abu-Ghazaleh, K. Kang and K. Liu, "*Towards resilient geographic forwarding in wireless sensor networks*". In Proc. of the 1st ACM international workshop on Quality of service & security in wireless and mobile networks, pp: 71–78, 2005.

[42] Ian F. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, "*Wireless sensor networks: A survey*". Computer Networks Elsevier Journal, Vol. 38, Issue: 4, pp. 393-422, 2002.

[43] K.Römer, F. Mattern, "*The design space of wireless sensor networks"*, IEEE Wireless Communications, pp. 1-6, 2004.

[44] A. Haque Chowdhur, K. Ki-Hyung, "*A survey of flash memory design and implementation of database in flash*", IEEE Wireless Communications, pp. 1256-1259, 2008.

[45] Ikalogic, URL : http://ikalogic.com/tut_adc.php/ Consulté le 4 Mars 2011.

[46] J. Sen, "*A survey on wireless sensor network s*ecurity". International Journal of Communication Networks and Information Security (IJCNIS), pp. 55-78, 2009.

[47] S. Aslam, F. Farooq, S. Sarwar, "*Power consumption in wireless sensor networks*". In Proc. of the 7th International Conference on Frontiers of Information Technology, pp. 1-9, 2009.

[48] J. M. Kahn, R. H. Katz and K. S. J. Pister, "*Next century challenges: mobile networking for smart dust*". In Proc. ACM Int'l.Conf. Mobile Computing and Networking (MobiCom'99), pp. 217–78, 1999.

[49] G. J. Pottie and W. J. Kaiser, "*Wireless integrated network sensors*". Commun. ACM, Vol. 43, Issue: 5, pp. 51–58, 2000.

[50] B. Przydatek, D. Song and A. Perrig, "*A. SIA: Secure information aggregation in sensor networks*". In Proc. of the 1st ACM International Conference on Embedded Networked Sensor Systems (SenSys 2003) (Los Angeles, Nov. 5–7). ACM Press, pp. 255–265, 2003.

[51] D.W. Carman, P.S. Krus and B.J. Matt, "*Constraints and approaches for distributed sensor network security*". Technical Report 00-010, NAI Labs, Network Associates Inc., Glenwood, MD, pp.1-136, 2000.

[52] FIPS PUB 46-2, "*Data encryption standard (DES)*". 1993.

[53] IETF RFC 2040, "*The RC5, RC5-CBC, RC5-CBC-PAD, and RC5-CTS algorithms*". 1996.

[54] W. Diffie and M. E. Hellman, "*New directions in cryptography*". IEEE Trans. Info. Theory, Vol. IT-22, Issue: 6, pp. 644–54, 1976.

[55] R. L. Rivest, A. Shamir, and L. Adleman, "*A method foroObtaining digital signatures and public-key cryptosystems*". Commun. ACM, Vol. 21, Issue: 2, pp. 120–26, 1978.

[56] P. Li, Y. Lin, and W. Zeng, "*Search on security in sensor networks*". Journal of Software, Vol. 17, pp. 2577- 2588, 2006.

[57] Y. Hu, A. Perrig and D. B. Johnson, "*Packet leashes: a defense against wormhole attacks in wireless networks*". In INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 3, pp. 1976–1986, 2003.

[58] J. Newsome, E. Shi, D. Song and A. Perrig, "*The sybil attack in sensor networks: analysis & defenses*". In Proc. of the third international symposium on Information processing in sensor networks, pp. 259–268, 2004.