Table des matières

CHAPITRE III: PROPOSITION D'UN PROTOCOLE DE DETECTION DES ATTAQUES PAR REPLICATION

INTRODUCTION GENERALE

Les développements technologiques dans le domaine de la micro-électronique ont permis l'introduction d'un nouveau type de composants « les capteurs électroniques » qui sont constitués dans la majeure partie des cas d'un micro processeur, d'une mémoire vive, d'une interface radio et d'une source d'énergie. L'ensemble de ces capteurs est appelé « réseaux de capteurs sans fil ».

Vu le faible coût de ces capteurs, de la large gamme disponible sur le marché (allant du capteur de position, capteur de son, capteur météorologique jusqu'aux capteurs équipés de caméras …), de leur capacité de s'auto configurer, de se gérer sans qu'il y ait besoin d'interventions humaines, de leur facilité de déploiement ainsi que de leur tolérance aux pannes leur ont permis d'être présents dans plusieurs domaines tel que : le domaine militaire pour surveiller les mouvements des forces ennemies, ou analyser le terrain avant d'y envoyer des troupes, dans le domaine de la surveillance que ce soit dans le milieu du bâtiment pour détecter par exemple les fissures ou les altérations de la structure, aussi bien qu'en guise de système d'alarme sécuritaire ainsi que dans les domaines environnemental, domestique ou sanitaire.

La problématique principale pour ces réseaux est sans aucun doute le manque de sécurité résultant de la difficulté d'appliquer les protocoles sécuritaires ordinaires sur les réseaux de capteurs, car ces derniers sont limités par leur batterie et leur puissance de calcul. La sécurité est d'autant plus nécessaire vu que ces réseaux sont déployés dans des zones hostiles, ce qui les rends des proies faciles pour les attaquants qui peuvent provoquer différents types d'attaques, nous pouvons en cité, les attaques actives qui peuvent modifier l'état du réseau, en détruisant les capteurs ou en empêchant leur mise en veille, en supprimant ou en modifiant le flux d' informations qui circulent. Ces attaques peuvent aussi être passives en écoutant les informations qui circulent comme : l'attaque par écoute passive, le brouillage radio qui sont difficilement détectable.

Ces multiples attaques obligent les chercheurs à repenser à des solutions de détection efficaces qui assurent principalement l'authentification, l'intégrité et la confidentialité de l'information et qui respectent en plus les contraintes des capteurs en termes de mémoire et surtout de communication car l'énergie c'est la ressource qu'il faut gérer avec la plus grande

attention. Assurer la sécurité dans les réseaux de capteurs va permettre de gagner la confiance des gens en cette nouvelle technologie, permettant d'élargir encore plus le domaine d'application et d'utilisation et ainsi faire des capteurs un outil qui peut nous accompagner dans notre vie quotidienne.

Nous nous intéressons dans ce mémoire à l'une des attaques les plus dangereuses pour les réseaux de capteurs sans fil qui est « l'attaque par réplication » où un attaquant capture un nœud, utilise ses clés cryptographiques secrètes pour créer plusieurs clones de celui-ci, c'est une attaque qui peut passer inaperçue pour le reste du réseau car le nœud cloné est *une copie conforme du nœud légitime*, en plus cette attaque peut être le point d'entrée d'attaques internes insidieuses.

Plusieurs recherches se sont intéressées à cette attaque et divers protocoles ont en résulté ainsi que différents articles ont été publiés sur le sujet. Parmi les premiers protocoles et sans doute des plus célèbres, on cite les protocoles de Parno et al. [14] qui ont permis d'introduire la notion de witness et de distribution dans les protocoles de détection des attaques par réplication tout en diminuant la coût de communication et de stockage par rapport à la solution centralisée principalement. Ces protocoles, ont été le point d'inspiration et de référence d'une multitude de protocoles qui ont essayé d'améliorer les rendements et de combler les lacunes.

Nous présentons dans ce mémoire, notre modeste contribution pour la détection des attaques par réplication, c'est le protocole MCD « Mobile assisted Clone Detection scheme » qui est un protocole hybride profitant de la simplicité des réseaux WSN statiques et des avantages en termes de mobilité des réseaux WSN mobiles, d'où l'utilisation d'une méthode de détection qui combine le coté centralisé pour la révocation et celui distribué pour la détection des attaques, en utilisant en plus le principe des « honeypots à faible interaction » pour améliorer encore plus le rendement de la détection.

Nous avons essayé dans notre protocole d'alléger au maximum les suppositions souvent surréalistes et couteuses de certaines solutions, tout en obtenant des coûts inférieures à ceux des protocoles existants avec O(1) comme coût de communication par nœud qui est inférieur à celui de LSM qui est de 0(√n) et de RED O(√n) (qui est pourtant une des références parmi les protocoles de détection des attaques par réplication), et un coût de stockage de O(1) inférieur à celui du protocole LSM O(√N) et similaire à celui de RED.

Ce manuscrit est organisé en trois chapitres suivis d'une conclusion générale, à savoir :

Dans le premier chapitre : ***Concepts généraux sur les réseaux de capteurs sans fil (RCSF)***, nous essayons de donner une vue générale sur la notion de capteurs, de réseaux de capteurs sans fil, des contraintes influençant l'architecture ainsi que des divers domaines qui les utilisent.

Dans le second chapitre : ***La sécurité dans les réseaux de capteurs sans fil : L'attaque par réplication,*** nous essayons d'aborder l'importance de la sécurité dans les réseaux de capteurs en s'intéressant particulièrement à l'attaque par réplication qui est la notion principale de notre thèse, en expliquant surtout la gravité d'une telle attaque sur le réseau et on termine par cité l'état de l'art des protocoles existant de détection de cette attaque

Dans le troisième chapitre : ***Proposition d'un protocole de détection des attaques par réplication,*** nous expliquons principalement les motivations de proposer un protocole de détection des attaques par réplication MCD, ensuite on explique notre protocole de façon détaillée et on termine par donner les couts de communication et de stockage ainsi que des résultats de l'implémentation.

Nous terminons cette thèse par une ***Conclusion générale*** dans laquelle nous présentons ce qu'on a appris à travers ce projet, Nous rappelons notre contribution pour cette thèse ainsi que des perspectives envisagées pour ce travail

CHAPITRE I

CONCEPTS GÉNÉRAUX SUR LES RÉSEAUX DE CAPTEURS SANS FIL (RCSF)

Sommaire

1. INTRODUCTION

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

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

2. NŒUDS

Les nœuds sont la notion de base dans un réseau de capteurs sans fil, selon l'application et la structure choisie, un RCSF (Réseau de Capteurs Sans Fil), peut contenir différents types de nœuds comme c'est cité dans [24] :

- Un *nœud régulier* est un nœud doté d'une unité de transmission et d'une unité de traitement de données.
- Un *nœud capteur* ou *nœud source* qui est un nœud régulier équipé d'une unité d'acquisition ou de détection. L'unité d'acquisition est généralement dotée d'un capteur ou plusieurs capteurs qui obtiennent des mesures et d'un convertisseur Analogique/Numérique qui convertit l'information relevée en un signal numérique compréhensible par l'unité de traitement.
- Un *nœud actionneur* ou *robot* est un nœud régulier doté d'une unité lui permettant d'exécuter certaines tâches spécifiques comme des tâches mécaniques (se déplacer, combattre un incendie, piloter un automate, etc.)
- Un *nœud puits* est un nœud régulier doté d'un convertisseur série connecté à une seconde unité de communication (GPRS, Wi-Fi, WiMax, etc.). La seconde unité de communication fournit une retransmission transparente des données provenant de nœuds capteurs à un utilisateur final ou d'autres réseaux comme internet.
- Un *nœud passerelle* (ou gateway) est un nœud régulier permettant de relayer le trafic dans le réseau sur le même canal de communication.

3. ARCHITECTURE D'UN NŒUD CAPTEUR

L'évolution de l'architecture des capteurs est l'un des facteurs qui a permis l'essor de solutions à base de réseaux de capteurs. En effet les capteurs des générations précédentes avaient une architecture qui se limitait au capteur proprement dit (dispositif capable de mesurer une grandeur physique) et une unité d'alimentation (batterie, piles, etc...). Le détecteur d'incendies est le parfait exemple de ce type de capteur, Celui-ci n'est composé que d'un système de capture alimenté par une pile, capable de détecter la fumée et de déclencher une alarme [3], cette évolution est décrite comme suit (voir figure 1.1) [5].

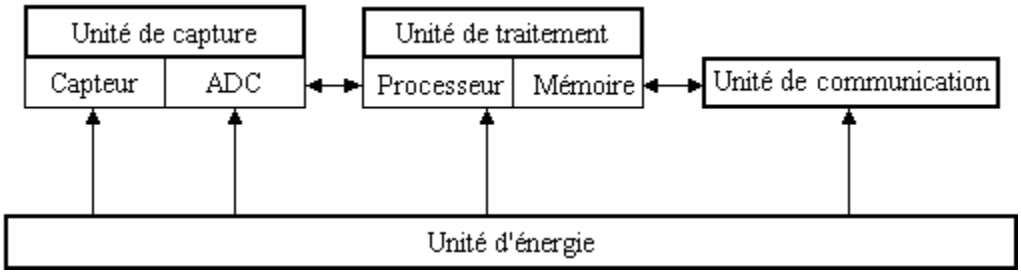

Figure 1.1 : Composants d'un nœud capteur

Unité de capture (*Sensing unit*) : elle est composée de deux sous-unités : un dispositif de capture physique qui prélève l'information de l'environnement local et un convertisseur analogique/numérique appelé ADC (*Analog to Digital Converters*) [6].

Unité de traitement (*Processing unit*) : les données captées sont communiquées au processeur où elles sont stockées dans la mémoire. Elle est généralement constituée d'un microcontrôleur dédié et de la mémoire. Les microcontrôleurs utilisés dans le cadre de réseaux de capteurs sont à faible consommation d'énergie. Leurs fréquences sont assez faibles, moins de 10 MHz pour une consommation de l'ordre de 1 mW. Une autre caractéristique est la taille de leur mémoire qui est de l'ordre de 10 Ko de RAM pour les données et de 10 Ko de ROM pour les programmes [8].

Unité de communication (*Transceiver unit*) : elle est composée d'un module radio émetteur/récepteur permettant la communication entre les différents nœuds du réseau en utilisant les ondes radio.

Unité d'énergie (*Power unit*) : c'est la batterie qui, n'est généralement ni rechargeable ni remplaçable. L'énergie limitée des capteurs représente la contrainte principale lors de conception de protocoles pour les RCSF. Les unités d'énergie peuvent être supportées par des photopiles qui permettent de convertir l'énergie lumineuse en courant électrique [6].

D'autres composants peuvent être utilisé dans certains capteurs selon les applications dans lesquelles ils sont utilisés comme « des mobilisateur » pour qu'ils puissent se déplacer ou bien de GPS (Global Position System) pour permettre de déterminer leur position géographique [9].

4. CARACTERISTIQUES PRINCIPALES D'UN CAPTEUR

Deux entités sont fondamentales dans le fonctionnement d'un capteur: l'unité d'acquisition qui est le cœur physique permettant la prise de mesure et l'unité de communication qui réalise la transmission de celle-ci vers d'autres dispositifs électroniques. Ainsi, chaque capteur possède un rayon de communication (R_c) et un rayon de sensation (R_s). La Figure 1.2 montre les zones définies par ces deux rayons pour le capteur. La zone de communication est la zone où le capteur peut communiquer avec les autres capteurs. D'autre part, la zone de sensation (ou de détection) est la zone où le capteur peut capter l'événement [36, 37, 38].

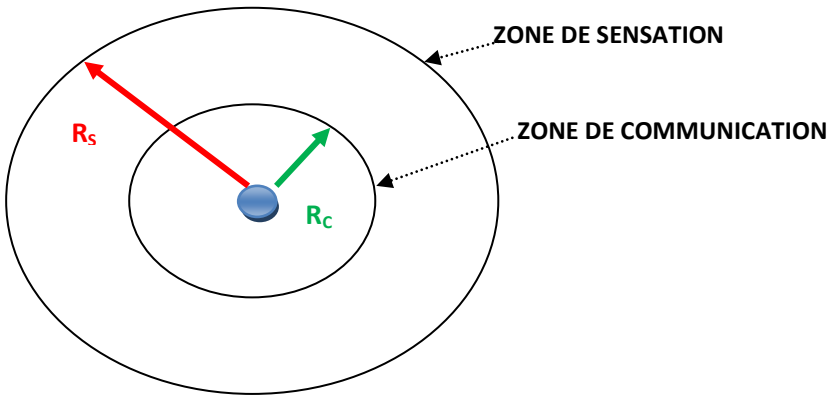

Figure 1.2 : Rayons de communication et de détection d'un capteur.

5. RESEAUX DE CAPTEURS

Un Réseau de Capteurs Sans Fil (RCSF) est un système distribué de grande échelle mettant en communication, des nœuds capteurs, à faible coût variant de quelques centaines d'éléments voir des milliers [2], répartis sur une grande surface. Les nœuds sont censés fonctionner de manière non supervisée même si de nouveaux nœuds sont ajoutés, ou de vieux nœuds disparaissaient [30] (Voir figure 1.3).

Pour le magazine Technology Review du MIT[1], les réseaux de capteurs sans fil sont l'une des dix nouvelles technologies qui bouleverseront le monde et notre manière de vivre et de travailler [4].

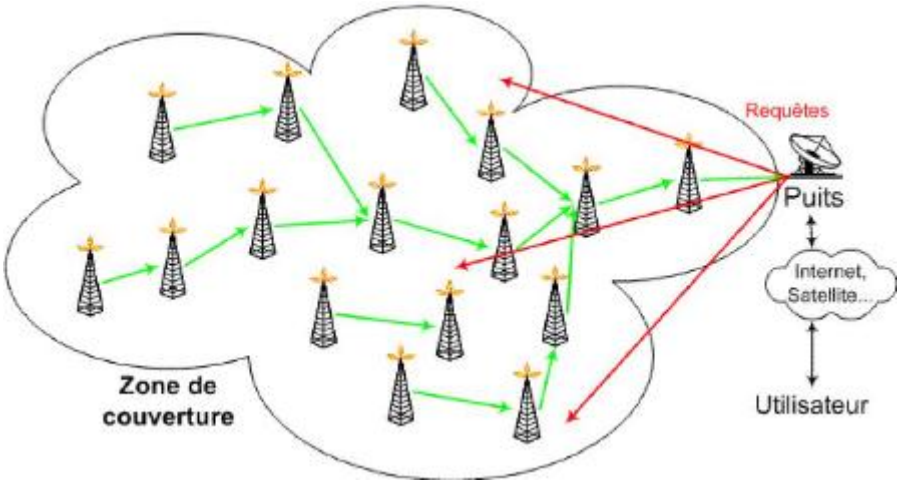

Figure 1.3 : Architecture d'un réseau de capteurs sans fil

6. ARCHITECTURE DES RESEAUX DE CAPTEURS SANS FIL

Le processus d'acheminement de l'information des capteurs à la station de base peut prendre quatre formes. Dans les architectures à plat, les capteurs peuvent communiquer directement avec la station de base en utilisant une forte puissance (figure 1.4 (a)), ou via un mode multi-sauts avec des puissances très faibles (figure 1.4 (b)), alors que dans les architectures hiérarchisées, le nœud représentant le cluster, appelé cluster-head, transmet directement les données à la station de base (figure 1.4 (c)), ou via un mode multi-saut entre les cluster-heads (figure 1.4 (d)). » [9]

[1] Massachusetts Institute of Technology

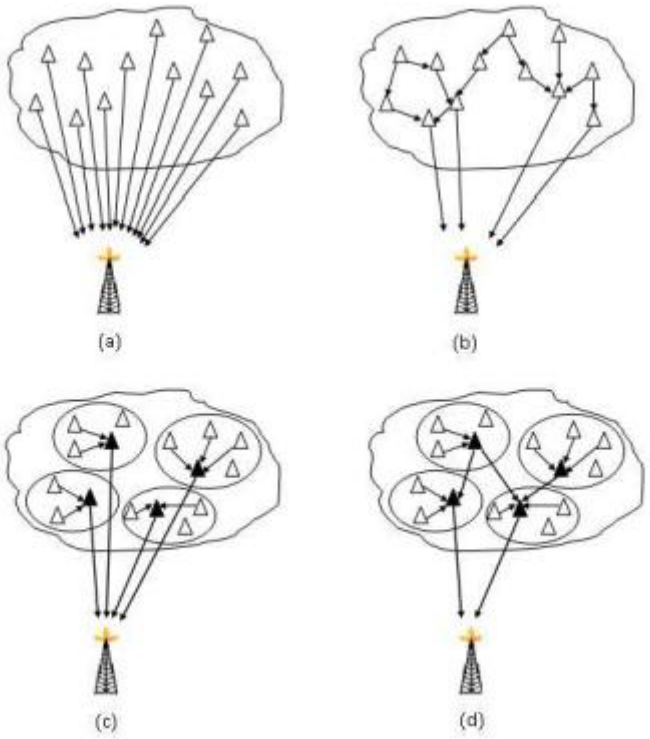

Figure 1.4 : Architecture de communication dans RCSF

7. CONTRAINTES INFLUENCANT LES RESEAUX DE CAPTEURS

Créer des protocoles nécessitant l'utilisation des réseaux de capteurs sans fil ou bien contribuant dans leur développement reste un problème difficile pour les raisons suivantes [15]:

1. Capacité limitée

Non seulement les ressources de calcul et de mémoire des nœuds sont relativement faibles, mais en plus, ils opèrent avec des piles ce qui rend leur durée de vie limitée, un protocole efficace et réaliste doit minimiser au maximum l'overhead de communication et de stockage pour ne pas pénaliser le réseau.

2. Agrégation de données

Il a été démontré dans plusieurs publications scientifiques que le fait d'agréger les données avant de les envoyées à une station de base va permettre de diminuer le nombre de messages envoyés, de réduire les puissances d'émission et ainsi économiser de l'énergie.

3. Echelle de dynamicité

Les réseaux de capteurs contiennent souvent un nombre de nœuds très important. Ces réseaux sont souvent peu stables et très dynamiques : les capteurs, qui ont épuisés leur pile, disparaissent et de nouveaux nœuds doivent être déployés pour assurer une certaine connectivité.

4. Protection physique faible

Les capteurs sont souvent déployés dans des environnements non-protégés (montagnes, forets, champs de bataille,...). Par conséquent, ils peuvent facilement être interceptés et corrompus. De plus, à cause de leur faible cout, ils utilisent rarement des composants électroniques anti-corruption (tamper-resistant devices).

8. DOMAINES D'APPLICATION DES RESEAUX DE CAPTEURS

L'utilisation des capteurs est depuis longtemps une réalité au sein de multiples domaines tel que l'industrie automobile ou aéronautique, mais l'affranchissement de la connexion filaire de part les progrès dans les technologies du sans fil permet d'étendre leur utilisation à une multitude d'autres applications [4], on en cite quelques unes :

- Ils peuvent détecter les fissures et les altérations dans les structures de ponts par exemple à la suite d'un séisme ou au vieillissement de la structure.
- Ils sont aussi utilisés dans les applications environnementales (détection d'incendies, détecter la pollution, analyser la qualité de l'air, empêcher que d'éventuels tsunami, inondations, volcan ne se produisent).
- Utilisés dans les applications commerciales, ils servent pour améliorer les processus de stockage, de livraison, pour connaitre la position, l'état et la direction d'une marchandise [10].
- ils peuvent être utilisés pour la détection de feux dans des grandes zones forestières, l'observation d'environnements naturels (pollution, inondation, etc.), suivi d'écosystèmes comme la surveillance d'oiseaux, croissance des plantes, etc….. [15].
- Le déploiement d'un réseau de capteurs de mouvement peut constituer un système d'alarme qui servira à détecter les intrusions dans une zone de surveillance [9].

9. CONCLUSION

Dans ce chapitre, nous avons essayé d'expliquer bien que brièvement la notion de capteurs, leurs caractéristiques et l'architecture des réseaux de capteurs sans fil entre autres.

Cela nous a permis de découvrir les multiples domaines d'utilisation de ces derniers vu la facilité de déploiement et du faible cout des capteurs, sans pour autant oublié les contraintes auxquelles doit faire face n'importe quel protocole travaillant sur ces capteurs surtout celle concernant « la limitation de l'énergie » qui est une ressource qu'il faut gérer avec la plus grande précaution étant donné qu'elle est en étroite liaison avec la durée de vie du protocole.

Dans le chapitre suivant, nous allons aborder le problème de la sécurité dans les réseaux de capteurs sans fil, on va s'intéresser particulièrement à l'attaque par réplication, aux répercussions qu'elle a sur le réseau et des solutions proposées dans la littérature pour détecter cette attaque.

Chapitre II

La sécurité dans les réseaux de capteurs sans fil

L'attaque par réplication

Sommaire

1. INTRODUCTION

Nous avons vu dans le chapitre précédent que les réseaux de capteurs sans fil sont des réseaux ad-hoc particuliers qui se caractérisent par leurs contraintes d'énergie et leur puissance limitée, sans oublier le fait qu'il sont déployés dans des zones hostiles, ce qui rend l'implémentation de mécanismes de sécurité efficaces, une nécessité absolue mais surtout difficile dans de tels réseaux.

La difficulté d'instaurer des mécanismes de sécurité efficaces est l'un des principaux obstacles au large déploiement des RCSF, surtout avec les besoins grandissant en matière de sécurité de nos jours ainsi que la nécessité d'avoir confiance en cette technologie pour lui ouvrir d'autres domaines d'applications.

Nous allons étudier en détails une des attaques les plus dangereuses qui est « l'attaque par réplication » et on va finir par donner une brève présentation des solutions apportées par la communauté scientifique pour détecter une telle attaque.

2. ATTAQUE PAR REPLICATION

La sécurité est un enjeu majeur dans les réseaux de capteurs sans fil et les attaques qui les touchent sont multiples d'où la nécessité de connaitre l'attaque pour comprendre comment l'attaquant agit et ainsi savoir s'en protéger.

Une des attaques les plus nuisibles sont les attaques par réplication « clones attacks » où un attaquant peut compromettre un nœud après l'avoir capturer, utiliser ses clés cryptographiques secrètes puis transférer les informations sur un nombre quelconque de nœuds pour réussir à peupler le réseau avec plusieurs clones de celui-ci, en vue d'acquérir le contrôle du réseau ou de perturber le fonctionnement normal de ce dernier [3]. Des outils facilement disponibles sont nécessaires pour cloner un nœud et un attaquant chevronné peut obtenir une copie de toute la mémoire et des données du nœud (même de ses clés cryptographiques) en seulement 1 minute après l'avoir repéré [39].

Ces attaques ont la capacité de passer à travers la couche de cryptage et d'authentification, ce qui rend leur détection vraiment difficile, vu que le nœud cloné est une copie conforme du nœud légitime du réseau, du coup, mêmes les voisins du nœud cloné ne pourront pas s'apercevoir de la supercherie.

Ce qui est encore plus grave, c'est que si l'attaquant réussi à cloner un nœud une fois, il peut le faire autant de fois que possible et ainsi en plaçant ces nœuds clonés dans des endroits stratégiques, il peut participer aux opérations du réseau, révoquer les nœuds légitimes, déconnecté le réseau en utilisant le système de vote par exemple, il est aussi possible pour un attaquant de capturer, de modifier ou de supprimer le trafic [2].

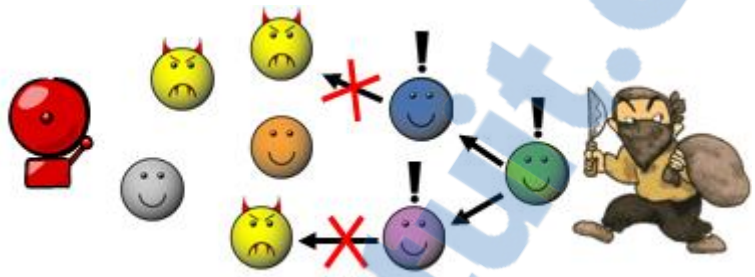

Figure 2.1 : Attaque par réplication

3. MECANISMES DE DETECTION DES ATTAQUES

Plusieurs mécanismes, basés généralement sur la notion de cryptographie, sont mis en place afin de répondre à la question de la sécurité dans les RCSF. Le mot « cryptographie » est composé des mots grecs: « crypto » signifie caché et «graphy» qui signifie écrire. C'est donc l'art de l'écriture secrète [40], les outils cryptographiques utilisés dans la sécurité des RCSF sont :

3.1. CHIFFREMENT SYMETRIQUE

Une même clé est utilisée entre deux nœuds communicants pour chiffrer et déchiffrer les données en utilisant un algorithme de chiffrement symétrique [41].

Avantages : [42]

- L'avantage principal de ce mode de chiffrement est sa rapidité.
- Pas d'opérations mathématiques complexes pour crypter ou décrypter les données.
- Pas de grandes dissipations énergétiques durant les phases de chiffrement et de déchiffrement.
- Plus adapté pour les RCSF.

Inconvénients : [41,42]

- la distribution de clés est difficile car dans un système symétrique, chaque nœud a besoin d'une clé partagée avec chaque autre nœud du réseau. Donc on aura à gérer n*(n-1)/2 clés avec : n nombre des nœuds.

3.2. CHIFFREMENT ASYMETRIQUE

Deux clés différentes sont générées : une clé publique pour le chiffrement et une clé privée maintenue pour le déchiffrement [41].

Avantages :

- impossibilité de déduire la clé privée à partir de la clé publique.
- distribution des clés moins pénibles et échange simplifié, nombre de clés générées= 2n.

Inconvénients :

- opérations mathématiques complexes gourmandes en capacité calcul et d'énergie.
- chaque capteur doit stocker les autres clés publiques des autres capteurs du réseau donc forte occupation mémoire.
- Difficilement applicable dans les RCSF (sauf pour le chiffrement asymétrique basé sur les courbes elliptiques).

3.3.CODE D'AUTHENTIFICATION DE MESSAGE MAC

Le code d'authentification de message MAC (Message Authentication Code) fait partie des fonctions de hachage (C'est le mécanisme qui assure l'intégrité de données. Cette fonction calcule une courte empreinte de taille fixe à partir d'une donnée de taille arbitraire) à clé symétrique assurant l'intégrité des données comme toute autre fonction de hachage, en plus, l'authenticité de la source de données. Comme illustré dans la figure 2.2, cette clé est utilisée pour calculer le code MAC par l'émetteur (1). Ce code est par la suite envoyé avec les données (2). [41].

Le récepteur calcule à son tour le code MAC avec cette même clé et le compare au code qu'il a reçu (3). S'ils sont bien identiques (4), alors la source est authentique et les données n'ont pas été altérées [41,43].

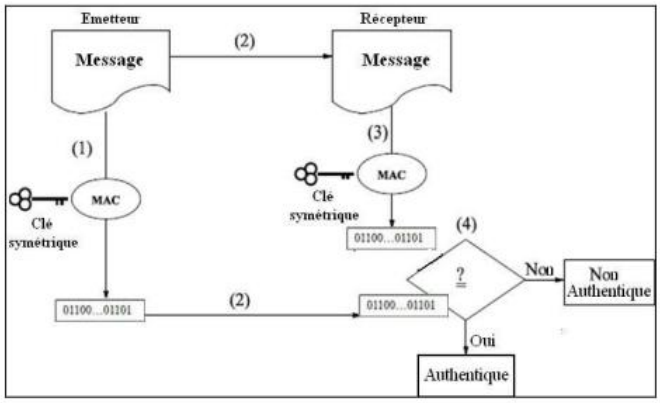

Figure 2.2 : Code d'authentification de messages MAC

3.4.HONEYPOTS

Un honeypot ou « pot de miel » est une technologie de sécurité qui fournit aux organisations un moyen d'attraper des virus, malware ou assaillants, ainsi que d'agir comme un système d'alarme qui peut découvrir les tentatives d'attaque d'un réseau [46]. En d'autres termes c'est un ordinateur ou un programme volontairement vulnérable utilisé pour être sondé, attaqué ou compromit pour attirer et piéger les pirates informatiques [26, 44, 45].

Un des points forts de ces honeypots, c'est qu'ils permettent de collecter et logguer (enregistrer les logs) les données avec de petites quantités de faux positifs et négatifs [25].

Il existe deux principaux types de honeypots : à faible interaction (passifs) et à forte interaction (actifs).

- Le pot de miel à *faible interaction* (ou passif) : utilise un mécanisme permettant d'attendre passivement les attaques afin de les détectés [45], ce sont les honeypots les plus simples, leur but est de capturer de l'information sans offrir beaucoup de privilèges à l'agresseur, et donc pas de risques potentiels pour le système puisque l'attaquant est très limité [33].

- Le pot de miel à *forte interaction* (ou actif) : utilise un mécanisme proposant le vrai service sur une machine plus ou moins sécurisée. Néanmoins, les risques sont très nombreux puisque la machine est très vulnérable. Il faut donc s'assurer que l'architecture sous-jacente soit bien sécurisée [33].

4. ETAT DE L'ART DES PROTOCOLES DE DETECTION DES ATTAQUES PAR REPLICATION

Vu la capacité de l'attaque par réplication à produire des attaques internes insidieuses, ce qui peut causer d'énormes dégâts au sein du réseau, plusieurs chercheurs se sont intéressés à cette attaque afin de pouvoir la détectée avec une probabilité élevée dès qu'elle se produise, tout en respectant les contraintes des RCSF avec qui il faudra cohabiter au mieux.

Les protocoles les plus connus dans la littérature sont cités ci-dessous :

4.1 DETECTION CENTRALISEE

Eschenauer et al. [33] ont permis de créer un protocole de détection des attaques par réplication totalement centralisé, qui consiste à ce que chaque capteur du réseau envoi à la station de base la liste de ses voisins avec leur position géographique. La station de base détecte les nœuds clonés si elle trouve des nœuds avec le même identificateur et ayant des positions différentes.

Cette méthode présente plusieurs inconvénients tel que : la présence d'un seul point de vulnérabilité ainsi que le cout élevé de communication du aux très nombreux messages qui circulent dans le réseau, ce qui la rend inutilisable.

4.2 DETECTION LOCALE

Chan et al. [34] ont créé un protocole utilisant un mécanisme de vote entre les voisins d'un nœud pour décider de sa légitimité.

Hélas, bien que cette méthode résout d'une certaine manière le problème de l'unique point de vulnérabilités du à l'utilisation d'une station de base, elle ne permet cependant pas de détecter les attaques situées à deux sauts du nœud concerné.

4.3 DETECTION DISTRIBUEE

Les protocoles distribués ont été créés afin de palier aux inconvénients des méthodes centralisées et locales en faisant collaborer un ensemble de nœuds pour éviter d'une part le point central d'échec et d'améliorer le rendement des protocoles de détection des attaques par réplications au sein du réseau, parmi les protocoles utilisant ce mécanisme, on cite :

4.3.1. Diffusion nœud-réseau

Le premier protocole distribué présenté par Parno et al. [14], utilise le principe du « broadcast », qui consiste à ce que chaque nœud diffuse par inondation une déclaration signée de sa position géographique à travers tous le réseau et sauvegarde les déclarations de ses voisins immédiats. Si un nœud reçoit une déclaration signée d'un nœud qui est en conflit avec celle de l'un de ses voisins, il diffuse dans tout le réseau une preuve de révocation contenant les déclarations en conflit, ce protocole assure une probabilité de détection d'attaques répliquées de 100% mais avec un cout de communication très élevé qui est de $O(n^2)$.

4.3.2. Multicast Déterministe

Le second protocole MD (Multicast Deterministe) présenté par Parno et al. [14], c'est le premier protocole qui a utilisé la notion de « witness » qui a inspiré plusieurs protocoles par la suite. Il s'appuie sur une fonction publique F qui, pour chaque nœud ayant une identité α, produit un ensemble de nœuds témoins F(α). Chaque nœud α effectue une diffusion locale de sa position géographique signée et chaque voisin la retransmet avec une probabilité p à un sous-ensemble de témoins du nœud qui sont choisis aléatoirement parmi les F(α) témoins. Les témoins sont choisis en fonction des identifiants des nœuds. Si un attaquant duplique un nœud, les témoins recevront deux positions géographiques différentes avec une même identité, donc, révocation des nœuds répliqués [27].

5. Randomized Multicast

Dans ce protocole RM (Randomized Multicast) de Parno et al. [14], chaque nœud envoi sa signature et son emplacement géographique à ses voisins et chaque voisin envoi avec une probabilité p la copie de la signature numérique du nœud à un ensemble de nœuds choisis de manière aléatoire et transmet le message au nœud le plus proche du point géographique choisit au hasard.

Ce protocole utilise la notion du : « Paradox des Anniversaires » qui assure qu'un nœud cloné et son clone vont entrer en collision avec une forte probabilité c'est-à-dire qu'un même témoin a une forte probabilité de recevoir deux déclarations de positions différentes avec la même signature.

6. Line Selected Multicast

Ce protocole LSM (Line Selected Multicast) de Parno et al. [14] est similaire au protocole RM mais il introduit une amélioration remarquable en termes de probabilité de détection. Dans LSM, quand un nœud annonce son emplacement, chaque voisin vérifie d'abord localement la signature de la demande, puis, avec une probabilité p, le transmet à $g \geq$ nœuds de destination aléatoirement sélectionnés. De plus, chaque nœud qui achemine ce message doit vérifier la signature, pour stocker le message, et vérifier la cohérence avec les demandes de localisation déjà reçus dans la même course du protocole de détection. La réplication est détectée par le nœud (le cas échéant) à l'intersection de deux chemins générés par deux nœuds différents portant le même numéro d'identifiant et en provenance de deux nœuds différents.

7. RED

Randomized Efficient Distributed (RED) a été proposé par Mauro Conti et al. [23], c'est un des protocoles les plus réputés pour la détection des attaques par réplication. Il se déroule en deux étapes : pour la première étape, une valeur aléatoire est partagée entre tous les nœuds du réseau, pour la deuxième étape chaque nœud signe et diffuse localement sa déclaration de position. Pour chaque nœud, chacun de ses voisins envoie la déclaration reçue à g >= 1 nœuds témoins pseudo-aléatoirement choisis, chaque témoin vérifie, s'il a déjà reçu une déclaration comportant le même identifiant. Si ce n'est pas le cas, le témoin sauvegarde la déclaration reçue, sinon, il vérifie si elles sont émises à partir du même emplacement géographique. Si le test est négatif, le témoin lance une procédure de révocation vis-à-vis du nœud déclarant [48].

8. CONCLUSION

Les protocoles de détection des attaques par réplication sont multiples, mais il n'existe pas encore *de standard* dans le domaine, ce qui motive les chercheurs pour étudier encore ce type d'attaques et essayer d'apporter « ce petit plus » qui fera du protocole proposé « le meilleur » par rapport aux autres protocoles existants, que ce soit en améliorant la probabilité de détection ou le coût de communication, aussi bien que le stockage.

Dans le chapitre suivant, nous allons essayer de présenter notre modeste contribution pour la détection des attaques par réplication.

Chapitre III

Proposition d'un protocole de détection des attaques par réplication

Sommaire

1. INTRODUCTION

La sécurité est importante pour de nombreuses applications de réseaux de capteurs sans fil WSN (Wireless Sensor Networks). Le fait que ces réseaux soient déployés dans des environnements hostiles, où l'adversaire peut physiquement capturer certains de ces nœuds ainsi que les ressources matérielles limitées de ces capteurs telles que (énergie limitée, faible puissance de calcul, utilisation des ondes radio, mémoire limitée, …) rendent ces réseaux sujet à plusieurs attaques.

Comme c'est cité dans le chapitre précédent, les attaques par réplication arrivent au sommet des attaques les plus insidieuses et des plus nuisibles pour le réseau RCSF, parce qu'elles passent non seulement inaperçues pour les autres nœuds, étant donné que le nœud cloné est une copie conforme du nœud légitime du réseau mais aussi parce qu'elles sont le point d'entrée d'une multitude d'attaques internes comme : initier une attaque wormhole[2], créer une attaque de type blackhole[3], injecter de fausse donnée où agréger les données de manière à influencer le résultat final etc.

Vu le danger que peut causer une telle attaque, une multitude de protocoles de détection d'attaques par réplication, ont été proposés dans la littérature, les plus connus sont : la détection centralisée de (Eschenauer et al. [33]), la détection locale de (Chan et al. [34]), sans oublier les protocoles distribués de (Parno et al. [14]) comme : la diffusion nœud-réseau qui est la méthode la plus simple mais aussi la plus naïve, ensuite, les premières solutions vraiment élaborées sont : RM (Randomized Multicast protocol) et LSM (Line Selected Multicast) qui permettent une détection distribuée et non déterministe.

Ces protocoles, ont été le point d'inspiration de plusieurs solutions que ce soit dans les réseaux WSN statiques, mobiles ou bien hybrides, avec des détections actives ou passives, en utilisant des hypothèses et des mécanismes différents les uns des autres, pour essayer de minimiser l'overhead et le taux de communication, afin que la proposition soit réaliste et utilisable dans les réseaux de capteurs sans fil.

Dans ce chapitre, nous présentons notre protocole de détection des attaques par réplication dans les réseaux de capteurs à architecture plate. Nous commencerons d'abord par présenter

[2] Avec au moins deux nœuds malicieux dans le réseau [13], l'attaquant peut tromper les autres nœuds sur les distances lors du routage et ainsi l'attaquant pourra récupérer les informations qui circulent

[3] Avec un nœud malicieux dans le réseau, il va modifier les tables de routages pour obligé ses voisins à faire passer l'information par lui, toute information qui va passer ne sera jamais retransmise.

les motivations de cette proposition, ensuite nous présenterons les détails de notre protocole, l'analyse de sécurité ainsi que l'implémentation en utilisant les capteurs telosb.

2. MOTIVATIONS

Les contraintes de conception des réseaux de capteurs sans fil telles que : la limitation de la mémoire (un capteur telosb possède 10 kilo octets de RAM) ainsi que la limitation de l'énergie qui est probablement une des caractéristique les plus pénalisantes, sont sans doute les plus grands défis pour concevoir des protocoles de sécurité efficaces et qui minimisent l'énergie afin de maximiser la durée de vie du réseau. En d'autres mots, l'énergie est sans aucun doute la ressource qu'il faut gérer avec la plus grande attention [15]. Une étude dans [16] a révélé que chaque bit transmis dans les réseaux de capteurs consomme environ autant d'énergie que l'exécution de 800 à 1000 instructions. Ainsi, la communication est plus coûteuse que le calcul dans les réseaux de capteurs.

Dans LSM, chaque ligne de segment contient $O(\sqrt{n})$ nœuds et chaque nœud enregistre $O(\sqrt{n})$ déclarations de localisation, ce besoin énorme en mémoire est impraticable dans des réseaux avec des milliers de nœuds .

La majeure partie des solutions existantes nécessitent la connaissance de la localisation des capteurs, ce qui exige l'implémentation d'algorithmes de localisation géographiques coûteux en mémoire ou bien l'utilisation de récepteurs GPS qui sont coûteux pour les capteurs, comme c'est le cas pour LSM et RM[14], LMA [21], RED [23] entre autres.

Il existe aussi, d'autres protocoles qui utilisent la diffusion, ce qui produit des coûts de communication très élevés comme c'est le cas dans le protocole « diffusion nœud – réseau » qui a un coût de communication de $O(n^2)$ et dans le protocole « multicast déterministe » [14] qui est de $O(g \ln g^* \sqrt{n} / d)$, ces coûts élevés vont avoir une répercussion négative sur les nœuds et sur le réseau, surtout dans le cas où l'architecture est constituée de centaines voir de milliers de capteurs. D'autres solutions utilisent aussi des suppositions trop lourdes et très contraignantes selon les besoins de leur protocole.

Le but de notre travail, c'est de proposer un protocole réaliste et utilisable, dans les petits comme dans les grands réseaux, qui concilie détection avec respect des différentes contraintes des WSNs. Les choix que nous avons faits lors de la conception ont été influencés par ces problématiques, en essayant de minimiser le trio (énergie, mémoire et coût) et de maximiser la probabilité de détection des attaques et en allégeant au maximum les suppositions.

En s'inspirant du protocole RAHIM de (Labraoui [7]) et en considérant la limitation de la consommation d'énergie et le montant élevé des frais généraux de communication des procédures de détection basées sur des nœuds statiques ainsi que les progrès réalisés dans la technologie des micro-robots, nous proposons un nouveau protocole nommé MCD «Mobile assisted Clone Detection scheme) » pour la détection des attaques par réplication, qui est un protocole hybride permettant d'utiliser les capteurs *statiques* ainsi que les capteurs *mobiles* dans un même réseau pour profiter de la simplicité des uns et les avantages en terme de déplacement des autres, ce protocole est considéré comme hybride parce qu'il utilise deux modèles de délivrances de données qui sont le mode périodique utilisé par les capteurs dans un état de fonctionnement normal du réseau et le mode événementiel qui se produit lors de la détection d'un événement bien particulier. Ce protocole est considéré aussi comme hybride parce qu'il permet de combiner deux approches différentes qui sont l'approche centralisée et l'approche distribuée et cela, en utilisant une détection *distribuée* de la réplication des nœuds tout en faisant appel à une décision de révocation *centralisée* de la part de la station de base, avec l'introduction en plus, d'une notion très peu utilisée dans les réseaux de capteurs qui est l'utilisation des *honeypots (pots de miel)* pour améliorer encore plus le rendement de la détection.

Notre protocole permet la détection des attaques par réplication, les autres attaques que peuvent connaitre les RCSF ne sont pas pris en considération dans notre proposition.

Dans le cas idéal où le système de détection n'est pas attaqué, notre protocole a un taux très faible de communication O (1) par nœud et un taux faible de mémoire qui est de O (1) ce qui est vraiment satisfaisant par rapport aux autres protocoles existants tout en ayant un taux de détection très élevé de l'ordre de 100%.

3. MODELE DU SYSTEME

Cette section illustre notre modèle du système qui inclut les modèles du réseau, des honeypots et de l'adversaire.

3.1. Modèle du réseau

Dans notre étude, nous considérons un réseau hétérogène avec une topologie plate constituée de : - Une station de base (SB) de confiance que l'attaquant ne peut pas répliquer.

- Un ensemble de nœuds statiques, déployés de manière aléatoire dans le réseau. chaque nœud à un identificateur (id) unique assigné par la station de base à l'initialisation.

Nous supposons que les nœuds capteurs ne connaissent pas leurs positions géographiques car le fait d'imposer aux nœuds la connaissance de leur localisation est une contrainte supplémentaire pour le réseau WSN.

- Le réseau est divisé en une topologie virtuelle rectiligne contenant des petites zones ayant la forme d'un carré, en respectant le principe que chaque capteur à l'intérieur d'une zone doit pouvoir écouter les évènements qui se produisent dans sa zone, chaque zone partage une clé en commun qui lui permet de communiquer à l'intérieur de la zone. Dans chaque zone, on choisit parmi ces derniers deux nœuds relais de façon aléatoire pour les besoins de notre protocole, ils ont pour rôle d'agréger et de retransmettre les mesures provenant des nœuds de la zone, afin que celles-ci parviennent à un utilisateur final [24].

- Comme dans tous les protocoles qui utilisent la notion de temps, nous supposons une certaine synchronisation entre les capteurs et la patrouille.

- Nous supposons que tous les nœuds peuvent atteindre directement la station de base comme ça été supposé dans le protocole LEACH (Handy, et al. [46]). Néanmoins, pour minimiser l'overhead de communication dans le réseau, on restreint cette supposition à la patrouille et aux deux nœuds relais qu'on nomme « WITNESS DE NIVEAU 1 » et « WITNESS DE NIVEAU 2 ». les honeypots communiquent éventuellement exclusivement et de façon directe avec la station de base.

- Une patrouille est assurée par un robot configuré avec les informations suivantes [28]:

  1. le chemin trajectoire, qui contient les paramètres de la courbe spécifique, on choisit comme trajectoire pour notre protocole une courbe remplissante qui passe par tous les points de la zone où sont déployés les nœuds du réseau sans jamais se couper (exemple : courbe de Peano [17]).
  2. La vitesse de patrouille (Vp), qui est la vitesse constante que le patrouilleur utilise lors de son déplacement.
  3. La période d'itération (T_{patr}), qui représente le temps que le patrouilleur prend pour traverser le chemin du point de départ au point final.

4. La durée de patrouille (D_{pt}), qui est la durée de tous les rounds que la patrouille est programmée à faire. On suppose aussi que la patrouille est plus puissante en termes de stockage, énergie et communication que les autres capteurs du réseau [18].

- Un ensemble de capteurs de leurre qui utilisent le principe des « honeypots passifs » qui n'ont pas de matériel cryptographique, donc, ne peuvent pas communiquer avec les autres capteurs, leur seule communication va s'adresser à la SB qui est la seule à connaitre leur existence en utilisant une clé bien spécifique.

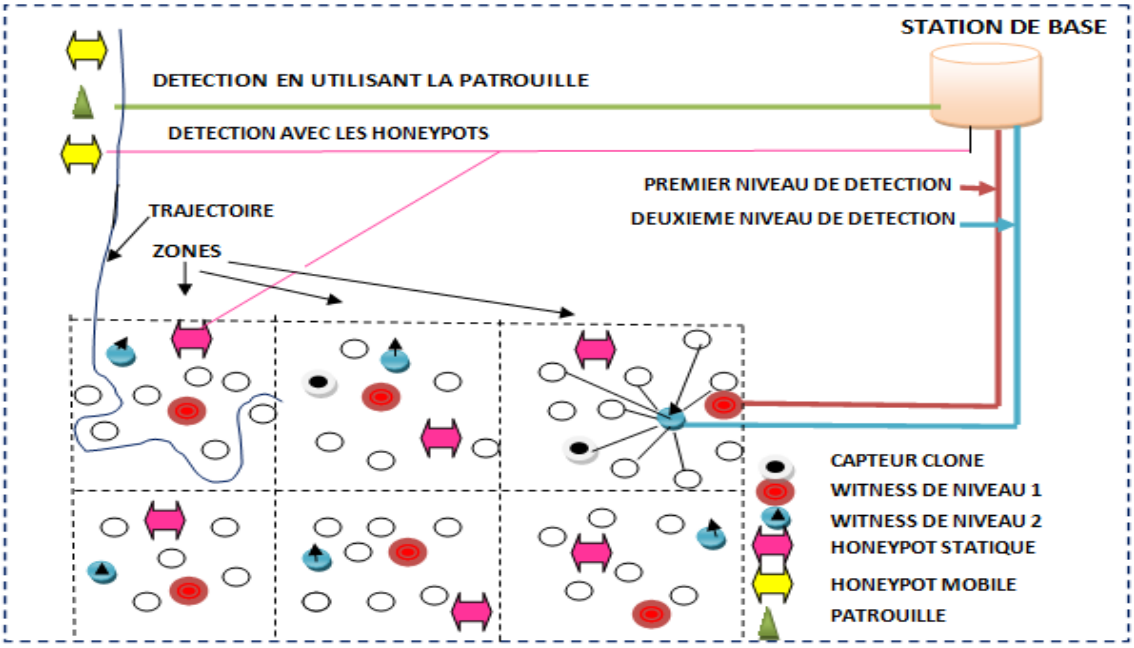

Figure 3.1 : Modèle du réseau.

3.2. Modèle de l'attaquant

Nous supposons que l'attaquant n'est pas capable d'allouer de nouvelles identités [19], il peut éventuellement cloner un ou plusieurs nœuds du réseau et les placés dans différents endroits du réseau selon ses besoins, il peut être mobile ou statique. Nous utilisons aussi la même supposition que les protocoles précédents [14], [21], c'est-à-dire que les nœuds contrôlés par l'adversaire continuent de suivre le protocole de détection des attaques par réplication, étant donné que l'adversaire veut toujours passer inaperçu pour les autres nœuds du réseau.

Nous supposons aussi, que pour cloner un nœud il faut le séparer du réseau pendant un certain temps et une fois la réplication effectuée, le réintégrer au réseau [31], [32], [14].

3.3. Notation

Dans notre travail, nous utilisons les notations suivantes illustrées dans le tableau3.1.

Notation	Description
SB	station de base
Id_i	identificateur du capteur i
rand	valeur aléatoire attribuée par la SB aux différents capteurs
NS	nœuds source (les nœuds stationnaires)
NR	nœuds relais
m	nombre de zones
n	nombre de NS
P	capteur patrouille

Tableau 3.1: Notations.

4. PROTOCOLE PROPOSE : MCD

Dans cette section, nous présentons notre algorithme pour détecter les attaques par réplication dans un réseau de capteurs sans fil, nommé MCD. Nous commencerons par donner une vue d'ensemble sur l'algorithme ensuite nous le détaillons**.**

4.1. IDEE PRINCIPALE

La conception de notre protocole est basée sur une architecture hybride (héterogène), similaire à celle utilisée dans [20]. Nous essayons de tirer partie de tous les avantages existants dans les systèmes statiques et mobiles en introduisant en plus la notion de « capteurs de leurre » pour améliorer les résultats.

Pour détecter les répliques, nous utilisons dans notre protocole quatre scénarios différents qui sont :

- Détection des attaques par réplication au sein de l'ensemble des capteurs statiques qui constituent notre réseau en utilisant un robot patrouille.
- Détection en utilisant les honeypots qui sont un piège pour détecter les nœuds malveillants et contrôler le trafic [22].
- Utilisation de deux niveaux de détection hiérarchiques pour détecter les attaques par réplication au niveau de la patrouille (P) et/ ou du capteur « WITNESS DE NIVEAU 1 » (inspirée de [7]) qui sont « WITNESS DE NIVEAU 1 » et « WITNESS DE NIVEAU 2».

- Utilisation des « WITNESS DE NIVEAU 2» de toutes les zones, pour la détection des répliques dans le réseau, comme méthode de détection des répliques en substitution dans le cas seulement où la patrouille sera répliquée et révoquée par la suite.

4.2. DETAILS DU PROTOCOLE

Notre protocole de détection d'attaques par réplication, est constitué de trois étapes principales qui nous permettent de détecter les réplications au sein du réseau avec une patrouille qui n'est pas clonée. Cependant, si la compromission de la patrouille et/ou du nœud WITNESS DE NIVEAU 1 a été détectée, le protocole exécute trois étapes supplémentaires pour remédier à cette attaque.

4.3. ETAPES PRINCIPALES :

1) Initialisation : cette étape inclue :

- La phase d'initialisation avant le déploiement des nœuds capteurs, dans laquelle la SB assigne chaque capteur i par un identifiant unique Id_i et une clé qui est un nombre aléatoire « rand », chacun des capteurs des différentes zones est pré chargé avec le T_{patr}.
- Dans chaque zone du réseau, et à chaque round, deux nœuds capteurs sont choisis WITNESS DE NIVEAU 1 et WITNESS DE NIVEAU 2 de manière *aléatoire* parmi les capteurs ordinaires de la zone afin de préserver le coté *indéterministe* du protocole, ces WITNESS ont la particularité de communiquer directement avec la SB

- A chaque round : la patrouille (P) diffuse des requêtes tout le long de la trajectoire

$$P \rightarrow *: id_{patr}, , n° \; round \; || \; MAC_{P}^{SB}(id, n°round) \qquad (1)$$

- chaque capteur qui écoute la patrouille, note ce temps T_1, s'il écoute toujours la patrouille après ce temps mais avec le même numéro de round : il ignore la requête et fait le calcul suivant :

$$T_{patr\ (suivant)} = T_{patr} + T_1 + \varepsilon \qquad (2)$$

(Avec ε : 0.. 60 secondes)

Comme la patrouille suit une trajectoire remplissante qui passe par tous les points du réseau, donc lors de l'ajout d'un nouveau capteur légitime dans le réseau cela ne va en

rien affecter T_{patr} entre deux rounds, parce que ce nœud va obligatoirement être inclus dans sa trajectoire initiale sans apporter de changements à cette dernière. Le même cas de figure est utilisé lorsqu'un capteur quitte le réseau, donc le temps T_{patr} de la configuration reste valable et correcte quel que soit les changements qui surviennent dans le réseau.

Le facteur ε est ajouté à l'équation, pour prévoir les aléas des capteurs et pour prendre en considération les cas ou le capteur ne répond pas assez vite à la demande de la patrouille.

Afin que les capteurs ne soient pas tout le temps éveillés et pour minimiser la consommation d'énergie des capteurs, comme chaque capteur connait le temps de la prochaine visite de la patrouille, nous considèrons pour notre protocole que les capteurs ne se réveillent que lorsque le $T_{patr(suivant)}$ approche (genre de compte à rebours qui est enclenché avec $T_{patr(suivant)}$ comme paramètre).

2) Détection des répliques en utilisant la patrouille

Un robot patrouille (P) est programmé pour faire la tournée de tous les nœuds pour détecter les répliques dans les nœuds statiques du réseau, il visite les zones du réseau « *une à une* » selon une trajectoire qui suit une *courbe remplissante,* passant par tous les points de la surface de la zone donnée sans jamais se couper, donc, on ne peut pas rencontrer le même nœud légitime plus d'une fois sauf existence de répliques et cela suivant un principe tout simple qui est « le recensement des nœuds ». La patrouille diffuse en continu sur toute la trajectoire la « *requête* » de la formule (1).

Cette détection suit le principe suivant:

- Pour le premier nœud visité, quand il va écouter la requête de la patrouille pour la première fois, il va lui communiquer son (id_1, $rand_1$), puisque c'est le premier, la patrouille va seulement sauvegarder ces informations.
- A partir du deuxième nœud et à chaque fois que la patrouille va recevoir l'identifiant et la clé, avant de l'enregistrer, elle va comparer avec les précédents enregistrements, s'il n'y a pas de similitude, alors, elle enregistre l'identifiant et la clé et poursuit sa trajectoire, si elle rencontre le même identifiant une nouvelle fois alors, elle envoi une alerte à la SB, cette dernière attend pendant une durée (d) une alerte mettant en cause la légitimité de la patrouille envoyée par les niveaux de détection (avec d très petit, car dans le cas d'une véritable réplication, il ne faudrait pas laisser beaucoup de temps à

l'attaquant, sinon il va générer plusieurs répliques), si le WITNESS DE NIVEAU 1 et/ou WITNESS DE NIVEAU 2 n'envoient aucune alerte, cela signifie, que la patrouille est légitime et la SB révoque les répliques signalées par la patrouille.

- La patrouille garde une trace de l'identifiant du capteur concerné par la réplication dans une liste noir dans le cas ou il n'y'à pas qu'une mais plusieurs répliques du nœud dans le réseau au même round.
- Les données enregistrées par la patrouille au cours d'un round, seront effacées avant de commencer un nouveau round.

3) Détection des répliques en utilisant les honeypots

Les pots de miel, ne sont pas une solution que l'on place pour résoudre un problème mais un outil à exploiter. Leur but c'est de détecter et/ou retarder l'attaquant pour qu'il n'attaque pas le vrai système [25]. Nous les utilisons dans notre réseau pour protéger d'une part le robot patrouille, du coup nous allons donner à quelques honeypots la même configuration interne qu'une patrouille pour donner l'illusion que c'est une véritable patrouille et d'autre part, nous allons placer un autre ensemble de capteurs de leurre parmi les capteurs stationnaires (NS) pour piéger les attaquants et détecter ainsi d'éventuelles attaques.

Les honeypots permettent une détection active des attaques avec un taux faible de faux positifs (déclarer qu'il y'à une attaque alors que réellement c'en est pas une) et de faux négatifs (déclarer qu'il n'y'à pas d'attaque alors que réellement c'en est une) [25].

Toute connexion qui s'adresse à un honeypot est suspecte, car les nœuds du réseau n'ont pas à communiquer avec lui du au fait qu'il n'a pas de matériel cryptographique à communiquer.
Un honeypot ne connait pas les autres nœuds et ne participe pas aux opérations du réseau, il n'est utilisé que pour donner un leurre pour les attaquants.

- Pour piéger « the smart attacker » (l'attaquant intelligent qui profite de toutes les informations sur le protocole de détection mis en œuvre, nous utilisons la notion de « *capteurs de leurre patrouille* » qui ont les mêmes propriétés que cette dernière telle que : la même trajectoire, la même vitesse, car un tel attaquant va vouloir attaquer la patrouille pour avoir le contrôle du réseau), le rôle du honeypot dans ce cas là est de soutenir la patrouille e*n augmentant ses chances de ne pas être répliquée :* un attaquant qui va vouloir cloner une patrouille, va savoir que tous les nœuds du réseau sont statiques et que seuls les capteurs mobiles peuvent être considérés comme « patrouille », il va essayer de capturer cette dernière pour la clonée sans savoir qu'il peut tomber sur le leurre, donc le honeypot va permettre de retarder sinon dans le meilleur des cas éviter

qu'elle ne soit clonée étant donné que s'est la base du protocole de détection et que si elle est attaquée ça aura des répercussions graves sur le protocole de détection.

- Pour piéger un « oblivious attacker » [23] (un attaquant inconscient qui à chaque étape de la séquence d'attaque, choisit un nœud à cloner de façon aléatoire parmi ceux qui n'ont pas encore été compromis), on utilise la notion de « *capteurs de leurres statiques* ».

Les honeypots permettent aussi de :

- *Générer des alertes à chaque fois d'un capteur s'adresse à lui :* comme nous utilisons des honeypots à faible interactions donc ces derniers vont émettre des alertes à la SB, ils ne peuvent pas révoquer les attaquants, donc à chaque message entrant, le honeypot envoi une alerte à la SB.

- *Enregistrer les signatures des attaques dans sa base de donnée :* car un honeypot détecte *toutes les attaques* et pas seulement les attaques par réplication, pour chaque attaque rencontrée, il garde sa signature.

4.4 ETAPES SUPPLEMENTAIRES:

En s'inspirant de [7], nous introduisons le principe de la détection hiérarchique sur plusieurs niveaux.

4) DETECTION DE PREMIER NIVEAU

Notre premier niveau de détection est l'utilisation de WITNESS DE NIVEAU 1. Ils sont utilisés principalement pour détecter une éventuelle attaque sur la patrouille P. Un capteur relais est avant tout un capteur du réseau qui a fait le calcul comme tous les autres capteurs statiques de chaque zone du $T_{patr(suivant)}$ (formule (2)) pour savoir quand sera la prochaine visite de P, vu que le robot P est programmé pour faire le même trajet durant le même intervalle de temps, ce calcul de temps reste donc inchangé.

Si, lors du prochain round T_{patr} sera différent du $T_{patr(suivant)}$ calculé, il va en déduire que P a été attaquée, nous avons les cas de figures suivants :

- Pour tous les WITNESS DE NIVEAU 1 de chaque zone, si ($T_{patr} \neq T_{patr(suivant)}$) alors envoyer alerte à la SB contenant un indice de confiance négatif pour signifier que P est répliquée.

$$WITNESS\ DE\ NIVEAU\ 1_i \rightarrow SB: \{id_i, rand_i, (-1) \,||\, MAC(id_i, rand_i)\} \quad (3)$$

- Si le WITNESS DE NIVEAU 1 (de la zone où P a envoyé l'alerte), apprend par écoute passive que la patrouille a envoyé une alerte à la SB pour révocation de capteurs et qu'il n'a constaté aucune anomalie en ce qui concerne son comportement (T_{patr}= $T_{patr(suivant)}$), alors, il n'envoi aucune alerte à la SB.

- Si par contre le WITNESS DE NIVEAU 1 de cette zone en apprenant par écoute passive que la patrouille a envoyer une alerte à la SB pour révocation de capteurs et qu'il a constaté que P est clonée (selon la notion de T_{patr}), alors il envoie à son tour un message d'alerte à la SB pour signaler la réplication de la patrouille, et demander sa révocation, selon la formule (3)

5) DETECTION DE DEUXIEME NIVEAU

Consiste en l'utilisation de la notion de WITNESS DE NIVEAU 2, ce deuxième niveau de détection est utilisé, pour détecter une attaque sur la patrouille et/ou sur les WITNESS DE NIVEAU 1.

Si les nœuds statiques de la zone que vient de quitter la patrouille constatent que la patrouille est répliquée selon T_{patr} et que WITNESS DE NIVEAU 1 de leur zone n'a pas envoyé de message pour alerter la SB de la réplication de la patrouille soit parce que ce WITNESS DE NIVEAU 1 est cloné aussi, soit parce que la réplication n'a pas été détecté par ce dernier.

Chaque capteur ayant détecté la réplication, envoi une alerte au WITNESS DE NIVEAU 2, contenant un indice de confiance négatif pour signifier que P est clonée.

$$i \rightarrow WITNESS\ DE\ NIVEAU\ 2 : \{id_i, rand_i, (-1) \,||\, MAC(id_i, rand_i)\} \quad (4)$$

Dans le cas où le nombre de messages reçus de la part des capteurs de la zone est supérieur à n/2m (avec m : le nombre de zones dans le réseau) alors : WITNESS DE NIVEAU 2 va faire l'agrégation de toutes les alertes envoyées par les capteurs de sa zone et cela en faisant la somme de tous les indices de confiances envoyés, en prenant la supposition que dans une zone, il y'à bien plus de capteurs légitimes que de capteurs clonés, donc :

- si la valeur obtenue est positive alors, WITNESS DE NIVEAU 2 constate que la patrouille est légitime, donc ne fait rien

- sinon, si la valeur obtenue est négative, WITNESS DE NIVEAU 2 envoi une alerte à la SB, avec un indice de confiance négatif signifiant que P est répliquée et demandant sa révocation

$$WITNESS\ DE\ NIVEAU\ 2 \rightarrow SB: \{id_i, rand_i, (-1)\ ||MAC(id_i, rand_i)\} \qquad (5)$$

4.5 REVOCATION DE LA PATROUILLE

Dans le cas où la SB décide de la révocation de la patrouille et pour ne pas bloquer ou bien retarder le processus de détection des attaques par réplication ce qui donnera un avantage majeur aux attaquants pour prendre le contrôle du réseau, nous utilisons dans notre protocole le WITNESS DE NIVEAU 2 comme mécanisme de substitution pour la détection des attaques en attendant la réhabilitation d'une nouvelle patrouille.

- Si révocation de P alors
- La SB va diffuser une alerte à tous les capteurs de toutes les zones, leur signalant que P a été révoquée.

$$SB \rightarrow WITNESS\ DE\ NIVEAU\ 2_i : \quad révoc\ (P), id_i \qquad (6)$$

- Tous les capteurs de chaque zone, vont envoyer leur id au WITNESS DE NIVEAU 2 de leur zone
- Chaque WITNESS DE NIVEAU 2, va envoyer à la SB un message contenant la concaténation de tous les id_i de sa zone.
- La SB va faire la révocation des répliques à chaque fois qu'un identifiant est cité plus d'une fois.

5. ANALYSE DE SECURITE

L'analyse de notre protocole repose sur :

1. *Détection de la réplication de la patrouille sans révocation*
2. *Détection de la réplication de la patrouille avec révocation*
   2.1. Patrouille répliquée, WITNESS DE NIVEAU 1 légitime ayant détecté la réplication
   2.2. Patrouille répliquée, WITNESS DE NIVEAU 1 légitime n'ayant pas détecté la réplication
   2.3. Patrouille répliquée, WITNESS DE NIVEAU 1 répliqué
3. *Détection de la réplication du premier niveau avec révocation*
4. *Réplication sur le deuxième niveau de détection*

5. *Détection des répliques en utilisant le WITNESS DE NIVEAU 2 :* avantages et inconvénients
6. *Résistance contre l'attaque des honeypots :* une telle attaque peut elle avoir lieu et qu'elles en sont les conséquences ?

5.1. DETECTION DE LA REPLICATION DE LA PATROUILLE SANS REVOCATION

Ce cas se produit, lorsque la patrouille répliquée ne fait aucune révocation de répliques, durant tout un round pour ne pas être détectée par les deux niveaux de détection d'attaques par réplication.

Dans notre protocole, nous proposons une solution concernant ce cas de figure, en considérant le fait que pour terminer tout un round, la patrouille clonée a du passer par un certain nombre de WITNESS DE NIVEAU 1 pour ne pas dire tous (nous prenons en compte, les cas où elle peut se positionner à n'importe quel endroit de la trajectoire sans la suivre complètement et le cas ou elle est répliquée au milieu ou bien à la fin de la trajectoire).
Nous considèrons aussi que le nombre de WITNESS DE NIVEAU 1 clonés est inférieur à celui des WITNESS DE NIVEAU 1 légitimes, car pour cloner seulement un seul WITNESS DE NIVEAU 1, l'attaquant aura plusieurs contraintes : une faible probabilité pour qu'il puisse trouver le WITNESS DE NIVEAU 1 rapidement parmi tous les capteurs de la zone surtout qu'il est choisit de façon aléatoire, nous ajoutons à cela le temps conséquent que nécessite le fait de cloner un capteur et d'exploiter les informations au sein du réseau, et tous cela durant le même round.

Donc, tous les WITNESS DE NIVEAU 1 de toutes les zones ayants détectés la réplication de la patrouille (selon $T_{patr)}$ vont envoyer une alerte à la SB (formule (3))
Dans le cas, où la patrouille est répliquée à la fin d'un round et en prenant la supposition qu'un relais cloné ne voudra pas révoquer une patrouille clonée pour complicité ce qui est assez logique, donc :
Si au moins un seul WITNESS DE NIVEAU 1 envoi une alerte à la SB, pour réplication de P et demandant sa révocation ***et* si** SB ne reçoit aucune alerte du WITNESS DE NIVEAU 2 de la zone associée à WITNESS DE NIVEAU 1 ***alors*** révocation de P

5.2. DETECTION DE LA REPLICATION DE LA PATROUILLE APRES REVOCATION

Ce cas se produit, lorsque la patrouille répliquée envoi une alerte de révocation de répliques, dans ce cas là, nous avons deux cas de figures selon que le WITNESS DE NIVEAU 1 est répliqué ou légitime :

5.2.1. Patrouille répliquée, WITNESS DE NIVEAU 1 légitime ayant détecté la réplication

Nous appliquons dans ce cas là, le premier niveau de détection des attaques par réplication qui consiste à ce que WITNESS DE NIVEAU 1 de la zone concernée par la révocation d'un de ses capteurs, va envoyer une alerte à la SB pour signaler la réplication de la patrouille.

5.2.2. Patrouille répliquée, WITNESS DE NIVEAU 1 légitime n'ayant pas détecté la réplication

La réplication de la patrouille s'est faite dans la zone du WITNESS DE NIVEAU 1 mais après avoir passée ce dernier, du coup, il n'a pas détecté la réplication de la patrouille étant donner qu'elle n'a été répliquée qu'après le relais, dans notre protocole, nous utilisons un deuxième niveau de détection qui est WITNESS DE NIVEAU 2, mais qui ne suffira peut être pas dans le cas, ou le nombre de capteurs ayants détectés la réplication est inférieur à n/2m.

Le WITNESS DE NIVEAU 2 va envoyer l'alerte de la formule (7) à la SB

La solution proposée ici, c'est d'appliquer les deux niveaux de détection WITNESS DE NIVEAU 1 et/ou WITNESS DE NIVEAU 2 mais dans la zone suivante.

5.2.3. Patrouille répliquée, WITNESS DE NIVEAU 1 répliqué

C'est le deuxième niveau de détection WITNESS DE NIVEAU 2 qui va envoyer une alerte concernant la réplication de la patrouille, formule (6), mais par contre la réplication du WITNESS DE NIVEAU 1 va pouvoir passer inaperçue étant donné, que les capteurs ne peuvent pas savoir si le WITNESS DE NIVEAU 1 est légitime et qu'il n'a pas détecté la réplication ou bien qu'il est répliqué et qu'il n'a pas voulu dénoncer la patrouille par conséquent la réplication de WITNESS DE NIVEAU 1 va être détectée au round suivant

5.4 DETECTION DE LA REPLICATION DU PREMIER NIVEAU DE DETECTION

Nous avons dans ce cas là, deux cas de figures suivant que la patrouille est répliquée ou non, nous citons dans ce qui suit seulement le cas où la patrouille est légitime vu que le cas où elle est répliquée a déjà été cité auparavant

- **Patrouille légitime, WITNESS DE NIVEAU 1 répliqué**

En considérant, le WITNESS DE NIVEAU 1 comme avant tout un capteur normal qui doit communiquer son identifiant et sa clé à la SB durant sa tournée même s'il est répliqué et cela

selon l'hypothèse qui stipule que les capteurs clonés doivent suivre le protocole de détection, donc, quand il lui communique ses informations, elle va détectée lors de sa trajectoire (sa ou ses) répliques donc, elle va envoyer à la SB pour révocation, nous avons deux cas de figures :

- La SB ne reçoit aucune objection de la part des deux niveaux de détection alors elle révoque les répliques (WITNESS DE NIVEAU 1_i et ses répliques) et au prochain round de la patrouille, de nouveaux WITNESS DE NIVEAU 1 pour toutes les zones sont choisis.
- La SB reçoit une alerte du premier niveau mettant en cause la légitimité de la patrouille de la part de WITNESS DE NIVEAU 1,formule (3), les capteurs de cette zone, vont savoir par écoute passive, que WITNESS DE NIVEAU 1 a envoyé une alerte et pourtant la patrouille n'a aucune anomalie, donc ils vont envoyé des alertes à WITNESS DE NIVEAU 2, formule (5),

  Une alerte de ce type va non seulement « innocenter » la patrouille, mais aussi déclarer la réplication du WITNESS DE NIVEAU 1

5.5. REPLICATION SUR LE DEUXIEME NIVEAU DE DETECTION

Le WITNESS DE NIVEAU 2 va essayer de générer des alertes pour ainsi avoir le contrôle du réseau, étant donné que son alerte est prioritaire sur toutes les autres cela va lui permettre de révoquer une patrouille légitime, ou de garder une patrouille répliquée.

Mais, étant donné que l'alerte est basée sur le vote majoritaire des capteurs source de la zone, donc il ne pourra pas créer une alerte valide [7].

5.6. DETECTION DES REPLIQUES EN UTILISANT WITNESS DE NIVEAU 2

L'utilisation de robot patrouille offre des avantages évidents en termes de coût de communication mais surtout plusieurs contraintes, car non seulement, le robot patrouille n'est pas résistant contre les attaques « tamper proof », il va être une cible idéale pour les attaquants, surtout qu'il est le point de détection central des réplications, et s'il est cloné c'est tous le réseau qui risque de s'effondrer faute d'absence de système sécuritaire de détection des attaques.

L'utilisation des honeypots n'est pas anodine, elle permet de protéger et de retarder les attaques contre la patrouille.

Nous ajoutons à cela, l'utilisation du WITNESS DE NIVEAU 2 comme système de détection dans le cas de la révocation de la patrouille pour garder un système de détection « *raisonnable* » afin de protéger le réseau en attendant la réhabilitation de la patrouille.

L'un des inconvénients majeurs de l'utilisation du WITNESS DE NIVEAU 2, c'est qu'il va augmenter l'overhead de communication au sein du réseau, mais c'est le prix à payer pour empêcher l'attaquant de prendre le contrôle du réseau.

5.7. RESISTANCE CONTRE L'ATTAQUE DES HONEYPOTS

« A l'instar des machines usuelles, les honeypots sont des systèmes conçus pour être scannés, attaqués et compromis » [25]. Le fait d'utiliser le principe des honeypots à faible interactions dans notre protocole en termes de « *capteurs de leurre* » a pour but de récolter un maximum d'informations tout en offrant un minimum de privilèges aux attaquants. Ils permettent de limiter les risques au maximum, ils ne font que simuler ces services et ne les possèdent pas réellement. Ils ne peuvent donc pas être exploités [26].

Etant donné qu'un capteur de leurre ne possède pas de matériel cryptographique, le fait de s'attaquer à lui n'a aucun impact négatif sur le réseau, étant donné qu'il ne connait rien du réseau ni des informations qu'il traite et cela va seulement permettre de retarder l'attaquant qui veut s'attaquer à la patrouille et de signaler cet attaquant à le SB pour qu'il soi révoqué.

Concernant le pourcentage de capteurs de leurres qu'il faut installer pour protéger une éventuelle attaque contre la patrouille, on remarque évidemment que plus leur nombre est grand meilleur est le résultat, mais il faut prendre en compte le coût, la charge du réseau, il ne faudrait surtout pas alourdir le protocole de détection.

6. COUT DE COMMUNICATION ET DE STOCKAGE

L'une des motivations principales derrière l'idée de notre protocole, c'est de créer un protocole qui permet d'atteindre un *taux de détection élevé* des attaques par réplication au sein d'un réseau hybride, tout en essayant de réduire au maximum l'overhead de communication et la quantité de mémoire utilisée.

Nous avons plusieurs scénarios possibles dans notre protocole, suivant les cas d'attaques par réplication dans le réseau.

6.1. COUT DE COMMUNICATION

Le cout de communication est exprimé par le nombre de messages qui circulent dans tous le réseau, ce coût est exprimé en O ().

- **Scénario 1 :** Détection en utilisant la patrouille

Dans ce cas là, chaque capteur envoi un seul message à la patrouille, donc, le cout de communication pour un seul capteur est de O(1).

Pour tout le round, chaque capteur doit envoyer un message à la patrouille sans oublier les messages d'alertes que doit envoyer la patrouille à la SB pour déclarer les éventuelles réplications détectées, on note ces messages (x)

On aura donc, un cout de communication total pour tous le réseau de : O (n)+ x avec : x<<n

Donc le coût de communication total est de O(n) qui est une bonne estimation par rapport aux autres protocoles comme le protocole LSM de O(n√n) et inférieur à celui de RED qui est de (dpg* n√n).

- **Scénario 2 :** Détection en utilisant le WITNESS DE NIVEAU 1

Dans ce cas là, si la patrouille envoi une révocation de n'importe qu'elle zone aussi bien de la 1ère que de la mèmezone, le coût de communication va dépendre du nombre de zones visitées ajouter à cela le message du WITNESS DE NIVEAU 1 envoyé à la SB, l'estimation du coût dans ce cas là est :

(Nombre de zones visitées* nombre de capteurs d'une zone) + Message d'alerte de WITNESS DE NIVEAU 1+ messages d'alertes de la patrouille

Dans le meilleur des cas ou le WITNESS DE NIVEAU 1 a détecté la réplication dans la 1ère zone, le coût est de : O(s)+1+1= O(s) avec : s= n/m : nombre de capteurs dans une zone et s<<n

Dans le pire des cas ou le WITNESS DE NIVEAU 1 a détecté la réplication dans la mème zone, le coût est de : m*O(s)+1+x= O(n).

- **Scénario 3 :** Détection en utilisant le WITNESS DE NIVEAU 2.

Le coût de communication du protocole est estimé par :

(Nombre de zones visitées* nombre de capteurs d'une zone) + alerte de WITNESS DE NIVEAU 2+ messages d'alertes de la patrouille à la SB.

Sachant que l'alerte du WITNESS DE NIVEAU 2 est générée par les capteurs de la zone ayants envoyer une alerte ($s_{dect} \leq s$), donc le coût d'une telle alerte est de o ($s_{detc}\sqrt{s}$).

Au meilleur des cas, le coût va être de : O(s)+ O ($s_{detc}\sqrt{s}$)+ 1= O($s\sqrt{s}$)

Au pire des cas (dernière zone) : O(n) + O ($s_{detc}\sqrt{s}$) + x= O($n\sqrt{s}$)

- **Scénario 3bis :** Détection en utilisant WITNESS DE NIVEAU 1 et WITNESS DE NIVEAU 2.

Le coût de communication va être : coût de communication de WITNESS DE NIVEAU 2+1
Qui est égale à : O (n√s).

- **Scénario 4 :** Détection en utilisant WITNESS DE NIVEAU 2 comme système de détection.

Le coût de communication est de : m* O(s√s) + m= O(n√s).

6.2. COUT DE STOCKAGE

- **Scénario 1 :** Dans ce cas là, chaque capteur va mémoriser seulement le $T_{patr(suivant)}$ ainsi que sa clé, ce qui va générer un cout de stockage de O (1), par contre, la patrouille va enregistrer les données de tous les autres capteurs (mis à part les répliques), d'où la patrouille va avoir un coût de stockage de O (n), ce qui est négligeable, étant donné que la patrouille est supposée être plus puissante en termes de stockage, énergie et communication que les autres capteurs du réseau [18].

- **Scénario 2 :** l'utilisation du premier niveau de détection WITNESS DE NIVEAU 1, ne va pas augmenter le coût de stockage des capteurs qui va rester égale à O (1).

- **Scénario 3 :** le deuxième niveau de détection WITNESS DE NIVEAU 2, ne va pas augmenter le coût du stockage, du au fait que le WITNESS DE NIVEAU 2 n'a pas à enregistrer les données des autres capteurs mais seulement faire l'addition des indices de confiances envoyés dans les messages d'alerte.

- **Scénario 3bis :** l'utilisation des deux niveaux de détection WITNESS DE NIVEAU 1 et WITNESS DE NIVEAU 2, garde toujours le cout de stockage à O (1).

- **Scénario 4 :** l'utilisation du deuxième niveau de détection comme mécanisme de détection principal, va engendrer un cout de communication supplémentaire pour la SB, mais cela est « négligeable » étant donné que cette dernière a une mémoire plus grande que les capteurs du réseau, ajouté à cela, le fait que les capteurs n'ont plus à mémoriser le $T_{patr(suivant)}$.

Le tableau 3.1 illustre la comparaison en termes de taux de stockage et de communication entre notre protocole MCD et les autres solutions existantes.

PROTOCOLES	COMMUNICATION		STOCKAGE
Diffusion nœud – réseau [14]	$O(n^2)$		$O(d)$
Multicast Déterministe [14]	$O(g \ln g * n\sqrt{n})$		$O(g \ln g)$
Randomized Multicast [14]	$O(n^2)$		$O(\sqrt{n})$
Line Selected Multicast [14]	$O(n\sqrt{n})$		$O(\sqrt{n})$
LMA [21]	$O(dp*n\sqrt{n}) + O(s*n)$		$O(sp_s)$
RED [23]	$O(dpg* n\sqrt{n})$		$O(dpg)$
NOTRE PROTOCOLE MCD	***Scénario 1***	$O(n)$	$O(1)$
	Scénario 2	$O(n)$	$O(1)$
	Scénario 3	$O(n\sqrt{s})$	$O(1)$
	Scénario 4	$O(n\sqrt{s})$	$O(1)$

Table 3.2 : Analyse de performance asymptotique.

7. RESULTATS DE L'IMPLEMENTATION

Au lieu d'utiliser un simulateur tel que Tossim par exemple et compte tenu du fait que dans tinyos 2.x il n'existe pas d'interface graphique tel que tinyviz dans tinyos 1.x, ce qui aurait facilité la tâche de simulation, Nous avons utilisé dans notre cas une implémentation en utilisant les capteurs afin de visualiser le fonctionnement de notre protocole de manière plus réaliste.

L'objectif principal de cette implémentation est de modéliser notre protocole « MCD » afin de déterminer son efficacité, ses points faibles ainsi que les perspectives envisagées.

7.1. CHOIX TECHNIQUES

L'implémentation de notre protocole a nécessité l'utilisation de différents outils matériels qui sont les capteurs telosb ainsi que des outils logiciels bien spécifiques au domaine des réseaux de capteurs sans fil, tels que NesC comme langage et TinyOs comme système d'exploitation, un aperçu de ces outils est décrit dans ce qui suit :

7.1.1. Capteur telosb : (voir Figure 3.2)

La plate-forme TelosB a été élaborée et publiée à la communauté scientifique par l'université Berkeley. Cette plate-forme offre une faible consommation d'énergie, elle est compatible avec la distribution open-source de TinyOs. Ce type de nœud peut être utilisé dans les applications suivantes :

- Plate-forme à faible puissance pour le développement de la recherche.
- Expérimentation des réseaux de capteurs sans fil.
-

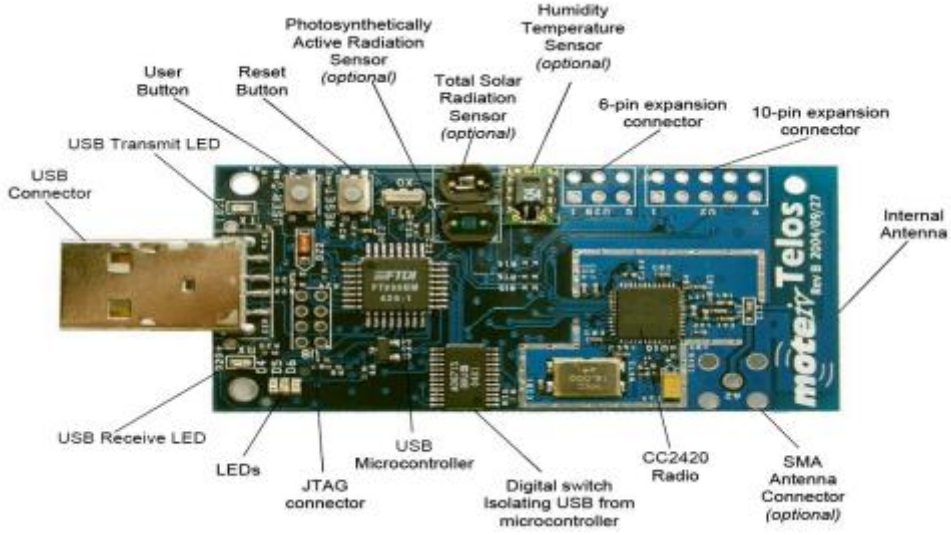

Figure 3.2 : Capteur telosb

7.1.2. TinyOS

Selon [4] TinyOS est un système d'exploitation pour les réseaux de capteurs sans fil open-source, conçu par des chercheurs de Berkeley, il respecte au maximum les contraintes des capteurs et présente une multitude d'avantages tel que :

- Il s'appui sur un fonctionnement événementiel c'est à dire qu'il ne devient actif qu'à l'apparition de certains évènements, par exemple l'arrivée d'un message radio. Le reste du temps, le capteur se trouve en état de veille, garantissant une durée de vie maximale .

- Il utilise une programmation orientée composant, ce qui permet de réduire la taille du code nécessaire à sa mise en place et respecte ainsi les contraintes émises par les capteurs en termes d'économie de mémoire et d'énergie.

- Un composant correspond à un élément matériel (LEDs, timer, ...) et peut être réutilisé dans différentes applications.
- Il dispose d'une bibliothèque de composants particulièrement complète qui peut être utilisée pour programmer.

7.1.3. NesC

C'est un langage orienté composant, syntaxiquement proche du C conçu pour incarner les concepts de structure et le modèle d'exécution de TinyOS.

7.2. MISE EN PLACE DE LA PLATEFORME

La mise en place de la plateforme s'articule autour de deux parties : l'installation logicielle et l'installation matérielle.

7.2.1. Installation logicielle : cette étape nous a permis de nous familiariser avec les capteurs telosb et le langage NesC ainsi que l'installation du système d'exploitation TinyOS 2.X sous Ubuntu, ce qui n'a pas été une tâche facile.

7.2.2. Installation matérielle : consiste en la réalisation de la plateforme implémentant notre protocole (voir Figure 3.3), Nous utilisons pour cela quatre capteurs telosb chacun représentant un rôle bien spécifique comme décrit ci-dessous :

Figure 3.3 : Plateforme d'exécution du protocole

a. Un capteur est flashé avec le programme « Patrouille ». La Figure 3.4 illustre la documentation de ce programme.

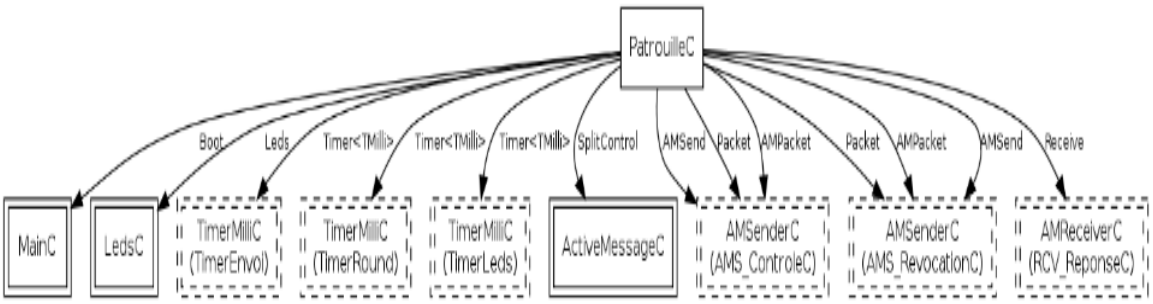

Figure 3.4. Représentation graphique du programme « Patrouille »

Ce programme utilise les interfaces:

- Boot : permet d'initialiser tous les composants au démarrage, elle est fournie par la configuration MainC qui est le cœur de l'application
- Leds : utilisée pour la manipulation des leds, fournie par LedsC
- Timer<TMilli> : c'est une interface de synchronisation qui permet de gérer le timer d'émission, de round et d'allumage des leds
- SplitControl : utilisée pour l'émission radio fournie par la configuration
- ActivemessageC : permet l'accès à la liaison sans fil et l'encapsulation de messages qui pourront être ensuite envoyés via la liaison sans fil.
- AMSend : pour l'envoi du paquet
- Packet : pour accéder aux données du message
- AMPacket : fournie l'adresse locale et la fonctionnalité d'accès au paquet
- Receive : pour la réception des messages

On utilise aussi, une table dans laquelle sont stockés les identifiants et les numéros de round des capteurs à chaque fois que ces derniers envoient des messages à la patrouille

b. Les trois autres capteurs sont utilisés pour représenter les capteurs stationnaires du réseau, on leur attribut les identifiants suivants : pour le premier capteur : id=1, pour le second capteur id=2, pour le troisième capteur id=3 et le quatrième capteur va représenter le clone du capteur 1 par exemple, on lui attribut l'identifiant id=1.
La représentation graphique du programme des nœuds est donnée par la figure 3.5

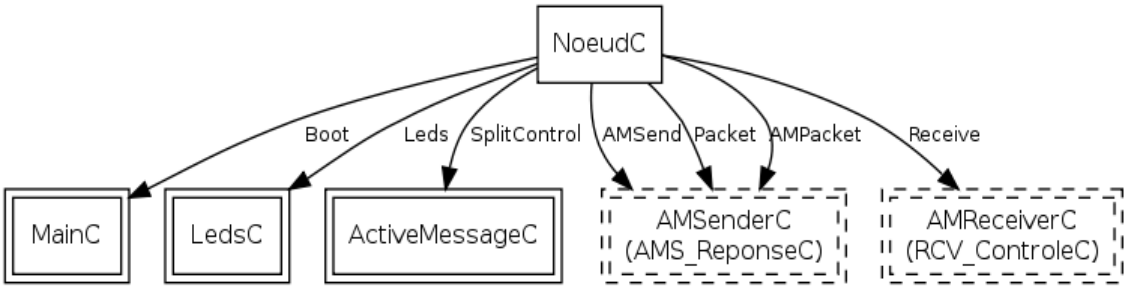

Figure 3.5. Représentation graphique du programme des nœuds

7.3. QUELQUES EXECUTIONS :

Nous utilisons pour l'exécution de notre protocole la plateforme de la figure 3.3, tout en faisant appel d'une part à la bibliothèque « *printf* » de TinyOs pour afficher le déroulement du processus d'exécution et d'autre part à un mécanisme de gestion de Leds tel que décrit dans le tableau 3. 3.

	PATROUILLE	NOEUDS
BOOT	Led1	Led1
ENVOI	Led2	Led2
RECEPTION	Led1+ Led2	Led1+ Led2
DETECTION DE REPLIQUES	Led0	/

Tableau 3.3. Gestion des Leds

1. **Scénario 1 : Implémentation du protocole sans la présence de clones**

Dans ce cas de figure, nous avons attribué aux capteurs les identifiants suivant :

- Patrouille : id=0
- Nœuds : id=1, id=2, id=3
- Nous obtenons, l'affichage de la figure 3.6

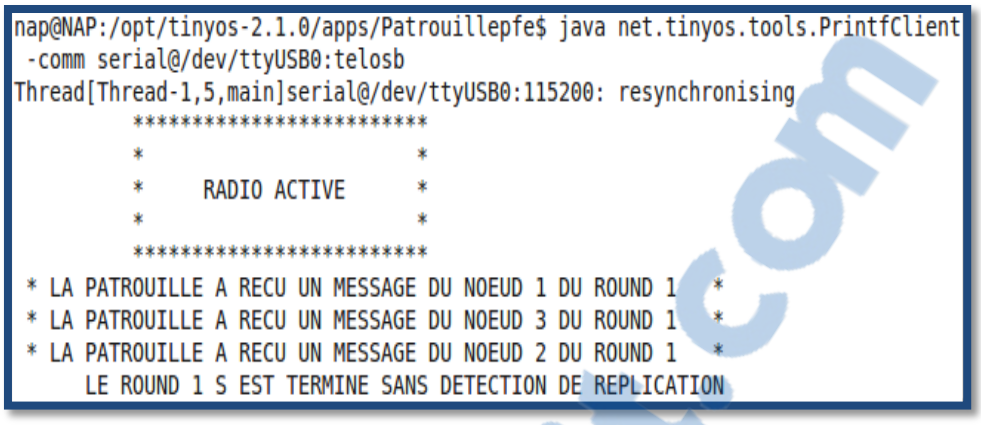

Figure 3.6: Implémentation du protocole sans la présence de clones

2. Scénario 2 : Implémentation du protocole avec la présence d'un clone

Dans ce cas de figure, nous avons attribué aux capteurs les identifiants suivant :

- Patrouille : id=0
- Nœuds : id=1, id=2 et pour représenter le capteur clone, nous utilisons un autre capteur auquel nous attribuons le même identifiant qu'un des capteurs déjà présents dans le réseau, par exemple : id=1
- Nous obtenons le résultat décrit dans la figure 3.7

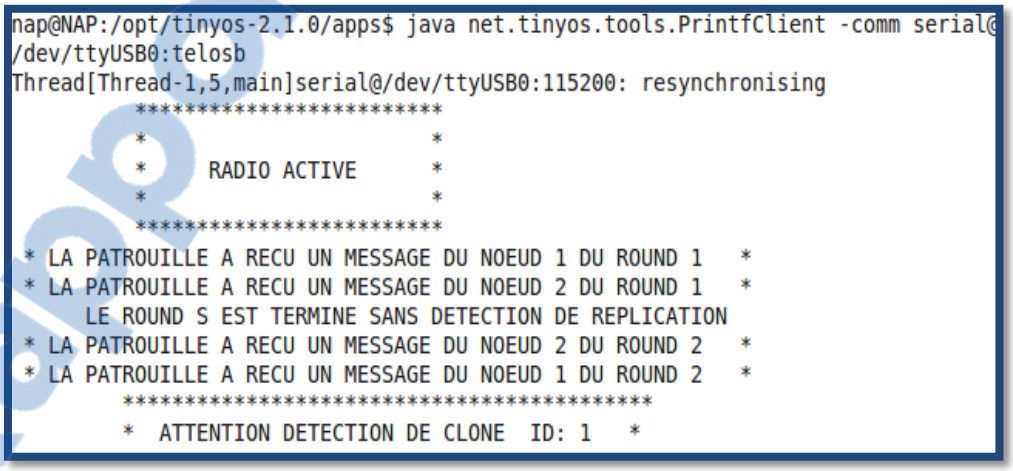

Figure 3.6. Implémentation du protocole avec la présence d'un clone

8. CONCLUSION

Nous avons présenté dans ce chapitre notre protocole de détection des attaques par réplication « MCD » qui apporte une nouveauté par rapport aux autres protocoles de détection existants qui est le coté hybride, l'utilisation de niveaux de détection hiérarchiques ainsi que l'introduction de la notion de Honeypots afin d'améliorer le rendement de la détection.

Notre protocole permet d'avoir un taux de détection avoisinant les 100% avec un coût de communication inférieur à celui des autres protocoles de détection des attaques par réplication et un coût mémoire égale voir même inférieur à ces derniers.

CONCLUSION GENERALE

Les réseaux de capteurs sans fil ont un large potentiel et constituent un sujet de recherche innovant ainsi qu'un outil convoité par plusieurs domaines.

C'est sans aucun doute, une technologie qui va nous accompagner pour les prochaines années et ainsi faire partie de notre vie quotidienne. Cependant, il y a encore beaucoup de problèmes qui doivent être abordés pour un fonctionnement efficace de ces réseaux dans des applications réelles. Parmi les problèmes fondamentaux et importants dans ces réseaux de capteurs nous citons la problématique de la sécurité qui est une nécessité absolue à laquelle des solutions adéquates doivent être proposées.

- **Synthèse :**

Le travail consigné dans ce mémoire a été le fruit d'une étude menée dans le contexte des réseaux de capteurs sans fil, ce qui nous a permis de découvrir les propriétés de ces derniers, de leurs contraintes et des domaines variés qui les utilisent. Nous nous sommes intéressé principalement à la problématique de la sécurité en étudiant une des attaques des plus insidieuses et des plus dangereuses pour les RCSF, qui est *l'attaque par réplication* étant donné qu'elle peut être le point d'entrée d'une multitude d'attaques internes qui peuvent causer des dégâts importants au sein du réseau et menacer ainsi l'authenticité et l'intégrité des données qui y circulent.

L'étude de l'attaque par réplication, nous a donné la possibilité de connaitre les solutions de détection proposées par la communauté scientifique et ainsi percevoir les avantages et les inconvénients de chaque protocole, ce qui nous a beaucoup aidé à l'élaboration de notre proposition de détection que nous avons nommé MCD : Mobile assisted Clone Detection scheme, qui est un nouveau protocole utilisant à la fois les capteurs statiques et mobiles, ainsi qu'une méthode de détection faisant collaborer la mobilité avec l'utilisation de deux niveaux de détection hiérarchiques, pour s'adapter au maximum aux contraintes des RCSF et satisfaire au mieux le besoin en sécurité.

Finalement, pour tester les performances de notre protocole MCD, nous avons développé une plateforme constituée de capteurs telosb, ce qui nous a permis de tester le fonctionnement de notre programme et démontrer son efficacité.

- **Perspectives :**

La conception d'un protocole pour sécuriser un réseau de capteurs est une tâche très difficile quelque soit le type d'attaque, car aucune attaque n'est anodine.

Un tel protocole devra combiner les contraintes propres aux systèmes distribués et aux systèmes embarqués et améliorer les résultats en termes de communication, de stockage et de probabilité par rapport aux protocoles déjà existants.

Vu le nombre d'attaques croissant, qui menacent les réseaux de capteurs sans fil et compte tenu du fait, que les capteurs constituent un axe de recherche très fertile, les perspectives ouvertes par ces réseaux sont donc nombreuses et variées.

Dans ce mémoire, nous avons utilisé des résultats basés sur l'implémentation sur des capteurs telosb, une des perspectives envisagées pour ce travail, c'est d'utiliser la simulation pour avoir une vue générale sur le comportement du protocole à grande échelle, avec la présence d'un grand nombre de clones et ainsi pouvoir utiliser les deux niveaux de détection hiérarchique et prendre en considération les capteurs de leurres afin de pouvoir calculer les métriques d'évaluation du protocole.

ANNEXE : Code source des programmes utilisés

Dans ce mémoire, nous avons travaillé avec Tinyos 2.x qui est un système d'exploitation pour les réseaux de capteurs sans fil open-source et en utilisant le langage de programmation NesC.

Comme il existe peu de documentation en français sur ces deux nouveaux concepts et quasi l'inexistence de codes sources dans le langage NesC disponibles sur le web, nous mettons à votre disposition le code source des programmes qu'on a utilisé dans ce projet.

I. Programme de la patrouille :

1. PatrouilleC.nc :

```
#include <Timer.h>
#include "Radio.h"
#include "printf.h"
/* ------------------------------------------------
 * Définition des Interfaces
  * Boot           Démarrage
 * Leds            Gestion des leds
 * Timer           Gestion d'un timer
 * AMControl       Contrôle interface radio
 * Interfaces emission et réception trames
 * ------------------------------------------------*/
module PatrouilleC {
  uses interface Boot;
  uses interface Leds;
  uses interface Timer<TMilli> as TimerEnvoi;
  uses interface Timer<TMilli> as TimerRound;

  /* Modules d'émission radio                */
  uses interface SplitControl as AMControl;

  /* Modules d'émission msg Controle     */
  uses interface Packet as Packet_CTLNOEUD;              /* Accès aux données du message */
  uses interface AMPacket as AMPacket_CTLNOEUD;          /* idem                         */
  uses interface AMSend as AMSend_CTLNOEUD;              /* Envoi du paquet              */

  /* Modules d'émission msg Revocation */
  uses interface Packet as Packet_REVNOEUD;
  uses interface AMPacket as AMPacket_REVNOEUD;
  uses interface AMSend as AMSend_REVNOEUD;

  /* Modules de réception Réponse         */
  uses interface Receive as Receive_REPNOEUD;
}

implementation {
/* ------------------------------------------------*
 *      Définition des variables              *
 * ------------------------------------------------*/
  uint16_t V_round = 0;                  /* compteur round                  */
  uint16_t id, round;
  bool V_result = FALSE;
  bool busy = FALSE;                     /* Booléen si radio occupée        */
```

```
  message_t pktctl, pktrep, pktrev;          /* Trames                          */
  struct TableNoeud tabNoeud[50];            /* Table des noeuds                */
  int cptTable = 0;                          /* Compteur associé à la table     */
  int i = 0;
  int j=0;
  int compteur = 0;
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
 *     Traitement cyclique des messages reçus              *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ */
  T_Reponse* trameREP[LEN_TABLE_CYCLIQUE];
  uint8_t cptRin = 0;
  uint8_t cptRout = 0;
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        *
 *                        BOOT: Démarrage de la radio                                                  *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        */
  event void Boot.booted() {
    call AMControl.start();
    call Leds.set(LEDS_LED1);
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        *
 *                          Phase démarrage radio OK.                                                  *
 *                                 Si OK : Lancement du Timer Envoi + Round                            *
 *                                  KO : Relance radio                                                 *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        */
  event void AMControl.startDone(error_t err) {
    if (err == SUCCESS) {
          V_round = 1;
      printf("        ************************* \n");
      printf("        *     RADIO ACTIVE         * \n");
      printf("        ************************* \n");
      call TimerEnvoi.startPeriodic(TIMER_ENVOI);
      call TimerRound.startPeriodic(TIMER_ROUND);
    }
    else {
          call AMControl.start();
    }
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        *
 *                                 Extinction de la radio.                                             *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        */
  event void AMControl.stopDone(error_t err) {
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        *
 *                               TIMER : Round terminé                                                 *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        */
  event void TimerRound.fired() {
    if (V_result == TRUE) {
    printf("     LE ROUND S EST TERMINE AVEC DETECTION DE REPLICATION SUR LE
CAPTEUR ID %u \n",id);
    } else {
    printf("     LE ROUND S EST TERMINE SANS DETECTION DE REPLICATION \n");
    }
    V_result = FALSE;
    V_round++;
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        */
 event void TimerEnvoi.fired() {
    if (!busy) {  T_Controle* Envoi = (T_Controle*)(call Packet_CTLNOEUD.getPayload(&pktctl,
sizeof(T_Controle)));
          Envoi->dIDPatrouille = TOS_NODE_ID;
          Envoi->dRound        = V_round;
          if (call AMSend_CTLNOEUD.send(AM_BROADCAST_ADDR, &pktctl, sizeof(T_Controle)) ==
SUCCESS) {
```

```
            atomic { busy = TRUE; }
          }
     }   }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          *
 *                          Contrôle si la trame Controle est bien envoyée                   *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          */
  event void AMSend_CTLNOEUD.sendDone(message_t* msg, error_t error) {
    if (&pktctl == msg) {
          atomic {busy = FALSE;}
      call Leds.set(LEDS_LED2);
    }
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          *
 *                      Traitement Réception d'une trame Reponse                             *
 *                 Possibilité de plusieurs réponses -> mise en place tâche                  *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          */
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          *
 *                 TASK                  Mise en table réponse                               *
 *                                   Si déja présent même valeur round -> réplique           *
 *                                                                  -> envoi révocation      *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          */
  task void TacheReponse() {
    /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
     *                   Vérification si donnée en attente             *
     *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
    if (cptRin == cptRout) return;
    /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
     * Contrôle si Données en table                                    *
     *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
    call Leds.set(LEDS_LED1 | LEDS_LED2);
    compteur=99;
    for (i=0;i<cptTable;i++) {
          if (trameREP[cptRout]->dIDNoeud == tabNoeud[i].tNoeud) {
            /* Je mémorise l'index i dans compteur */
            compteur=i;
          }
    }
    /* Cas où le noeud n'est pas connu                    */
    if (compteur == 99) {
          tabNoeud[cptTable].tNoeud=trameREP[cptRout]->dIDNoeud;
          tabNoeud[cptTable].tRound=trameREP[cptRout]->dRound;
          cptTable++;
    } else {
          /* Cas où le noeud est connu                         *
           * Cas 1 : round reçu égal au round mémorisé */
          if (trameREP[cptRout]->dRound == tabNoeud[compteur].tRound) {
            call Leds.set(LEDS_LED0);
        printf("        ******************************************* \n");
        printf("        *  ATTENTION DETECTION DE CLONE  ID: %u    *\n",id);
        printf("        ******************************************* \n");
        V_result = TRUE;
          } else {
          /* Cas 2 : round reçu différent du round mémorisé */
            tabNoeud[compteur].tRound=trameREP[cptRout]->dRound;
            printf("round different pas de probleme \n");
          }
    }
    cptRout = (cptRout + 1) % LEN_TABLE_CYCLIQUE;
    post TacheReponse();
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          *
 *                      EVENT          Réception de la trame Reponse                         *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~          */
```

```
  event message_t* Receive_REPNOEUD.receive(message_t* msg, void* Payload, uint8_t len) {
    if (len == sizeof(T_Reponse)) {
        T_Reponse* Reception = (T_Reponse*)Payload;
        /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
         *                  Traitement Tache               *
         *          Avec incrémentation compteur IN module table   *
         *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
     id = Reception->dIDNoeud;
     round = Reception->dRound;
     printf(" * LA PATROUILLE A RECU UN MESSAGE DU NOEUD %u DU ROUND %u
*\n",id,round);
        atomic {
          trameREP[cptRin] = Reception;
          cptRin = (cptRin + 1) % LEN_TABLE_CYCLIQUE;
        }
        post TacheReponse();
    }
    return msg;
  }
/* Fin implementation - ne pas supprimer */
}
```

2. PatrouilleAppC.nc :

```
configuration PatrouilleAppC {
}
implementation {
  components MainC;                               /* Programme principal - obligatoire      */
  components LedsC;                               /* Gestion leds                           */
  components PatrouilleC as App;                  /* Mon Programme                          */
  components new TimerMilliC() as TimerRound;/* Gestion Timer round                     */
  components new TimerMilliC() as TimerEnvoi; /* Gestion Timer émission contrôle        */
  components ActiveMessageC;
  components SerialActiveMessageC;

/* Gestion émission des données */
  components new AMSenderC(D_CTLNOEUD)  as AMS_ControleC;
  components new AMSenderC(D_REVNOEUD)  as AMS_RevocationC;

  /* Gestion réception des données        */
  components new AMReceiverC(D_REPNOEUD) as RCV_ReponseC;
 /* Association des Composants aux interfaces */

  App.Boot -> MainC.Boot;
  App.Leds -> LedsC;
  App.TimerEnvoi -> TimerEnvoi;
  App.TimerRound -> TimerRound;
  App.AMControl -> ActiveMessageC;
  App.SerialControl -> SerialActiveMessageC;

 /* TRAME Controle                */
  App.AMSend_CTLNOEUD -> SerialActiveMessageC.AMSend[AM_TEMPERATURECHIFFRE_MSG];
  App.Packet_CTLNOEUD -> SerialActiveMessageC;
  App.AMPacket_CTLNOEUD -> SerialActiveMessageC;

  /* TRAME Reception Id Noeud          */
  App.Receive_REPNOEUD -> ActiveMessageC.Receive[AM_TEMPERATURECHIFFRE_MSG];
}
```

3. Radio.h :

```
#ifndef RADIO_H
#define RADIO_H
enum {
```

```
  D_CTLNOEUD = 10,                              /* Trame contrôle noeud */
  D_REPNOEUD = 11,                              /* Trame réponse noeud */
  D_REVNOEUD = 12,                              /* Trame révocation noeud */
  TIMER_ROUND = 30000,                   /* Round (en millisecondes)  = 5 minutes */
  TIMER_ENVOI = 6000,                 /* Timer emission trame contrôle (en milisecondes) */
  LEN_TABLE_CYCLIQUE = 50 /* Cycle table In-Out */
};
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*
 *                  Tableau                        *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~*/
typedef struct TableNoeud {
  uint16_t tNoeud;
  uint16_t tRound;
} TableNoeud;
 /*--------------------------------------*
  *    Définition de la structure      *
  *--------------------------------------*/
typedef nx_struct T_Controle {
  nx_uint16_t  dIDPatrouille;
  nx_uint16_t  dRound;
} T_Controle;
/*--------------------------------------*/
typedef nx_struct T_Reponse {
  nx_uint16_t  dIDNoeud;
  nx_uint16_t  dRound;
} T_Reponse;
#endif
```

4. Makefile:

```
COMPONENT=PatrouilleAppC
CFLAGS += -I$(TOSDIR)/lib/printf
CFLAGS += -DCC2420_NO_ACKNOWLEDGEMENTS
CFLAGS += -DCC2420_NO_ADDRESS_RECOGNITION
CFLAGS += -DTASKLET_IS_TASK
include $(MAKERULES)
```

II. Programme des capteurs :

1. NoeudC.nc :

```
#include <Timer.h>
#include "Radio.h"
/* -------------------------------------------
 * Définition des Interfaces
 * Boot             Démarrage
 * Leds             Gestion des leds
 * AMControl    Contrôle interface radio
 * Interfaces emission et réception trames
 * -------------------------------------------*/
module NoeudC {
  uses interface Boot;
  uses interface Leds;

  /* Modules d'émission radio                */
  uses interface SplitControl as AMControl;

  /* Modules d'émission msg Reponse      */
  uses interface Packet as Packet_REPNOEUD;    /* Accès aux données du message */
  uses interface AMPacket as AMPacket_REPNOEUD; /* idem                        */
  uses interface AMSend as AMSend_REPNOEUD;         /* Envoi du paquet                  */

  /* Modules de réception Controle          */
```

```
  uses interface Receive as Receive_CTLNOEUD;
}
implementation {
/* ---------------------------------------------
 * Définition des variables
 * ---------------------------------------------*/
  uint16_t V_round = 0;               /* compteur round              */
  uint16_t tmpRound;                  /* Variable temporaire         */
  uint16_t tmpPatrouille;             /* idem                        */
  bool busy = FALSE;                  /* Booléen si radio occupée    */
  message_t pktctl, pktrep;           /* Trames                      */

/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   *
 * BOOT: Démarrage de la radio                                        *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   */
  event void Boot.booted() {
    call AMControl.start();
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   *
 *                    Phase démarrage radio OK.                                *
 *                      Si OK : Init valeur round à 0                          *
 *                          KO : Relance radio                                 *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   */
  event void AMControl.startDone(error_t err) {
    if (err == SUCCESS) {
          V_round = 0;
    }
    else {
          call AMControl.start();
    }
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   *
 *                          Extinction de la radio.                            *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   */
  event void AMControl.stopDone(error_t err) {
  }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   *
 *                          Reception d'une trame Controle                     *
 *                            1) Vérification si déjà reçue (valeur V_round)   *
 *                     2) Si différent,  envoi réponse + mémorisation round    *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~   */
  event message_t* Receive_CTLNOEUD.receive(message_t* msg, void* Payload, uint8_t len) {
   call Leds.led1On();  /* Allume LED 1 */
   call Leds.led0Off(); /* Eteint les autres */
   call Leds.led2Off();
   if (len == sizeof(T_Controle)) {
          T_Controle* Reception = (T_Controle*)Payload;
          /* Mémorisation des données dans le cas où réception autre trame */
          tmpRound = Reception->dRound;
          tmpPatrouille = Reception->dIDPatrouille;
          if (tmpRound == V_round) {
        /* Cas où trame déjà reçue : Allume LED 2 */
             call Leds.led2On();
          } else {
             if (!busy) {
                T_Reponse* Envoi = (T_Reponse*)(call Packet_REPNOEUD.getPayload(&pktrep,
sizeof(T_Reponse)));
                call Leds.led0On(); /* Allume LED 0 */
                atomic { busy = TRUE; }
                Envoi->dIDNoeud = TOS_NODE_ID;
                Envoi->dRound = tmpRound;
                if (call AMSend_REPNOEUD.send(tmpPatrouille, &pktrep, sizeof(T_Reponse)) == SUCCESS)
{
```

```
                } else {
                      /* Si problème pas de relance */
                  busy = FALSE;
                }
              }
          }
        }
        return msg;
      }
/* ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        *
 *                               Contrôle si la trame Reponse est bien envoyée                 *
 * ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~        */
  event void AMSend_REPNOEUD.sendDone(message_t* msg, error_t error) {
    if (&pktrep == msg) {
        atomic {
          busy = FALSE;
          /* Incrémentation valeur round */
          V_round = tmpRound;
        }
    }
  }
/* Fin implementation - ne pas supprimer */
}
```

2. NoeudAppC.nc :

```
#include <Timer.h>
#include "Radio.h"
configuration NoeudAppC {
}
implementation {
  components MainC;                    /* Programme principal - obligatoire       */
  components LedsC;                    /* Gestion leds                            */
  components NoeudC as App;            /* Mon Programme                           */
  components ActiveMessageC;           /* Composant pour Radio                    */

  /* Gestion émission des données         */
  components new AMSenderC(D_REPNOEUD)   as AMS_ReponseC;

  /* Gestion réception des données         */
  components new AMReceiverC(D_CTLNOEUD) as RCV_ControleC;

 /* Association des Composants aux interfaces */
  App.Boot -> MainC.Boot;
  App.Leds -> LedsC;
  App.AMControl -> ActiveMessageC;

  /* TRAME ID Noeud              */
  App.AMSend_REPNOEUD -> AMS_ReponseC;
  App.Packet_REPNOEUD -> AMS_ReponseC;
  App.AMPacket_REPNOEUD -> AMS_ReponseC;

  /* TRAME Reception Controle */
  App.Receive_CTLNOEUD -> RCV_ControleC;}
```

3. Radio.h :

```
#ifndef RADIO_H
#define RADIO_H
enum {
  D_CTLNOEUD = 10, /* Trame contrôle noeud */
  D_REPNOEUD = 11  /* Trame réponse noeud */
};
/*--------------------------------------*
```

```
 * Définition de la structure          *
 * Trame envoi Controle                *
 *-------------------------------------*/
 typedef nx_struct T_Controle {
   nx_uint16_t  dIDPatrouille;
   nx_uint16_t  dRound;
} T_Controle;
/*-------------------------------------*/
typedef nx_struct T_Reponse {
   nx_uint16_t  dIDNoeud;
   nx_uint16_t  dRound;
} T_Reponse;

#endif
```

4. Makefile :

```
COMPONENT=NoeudAppC
CFLAGS += -DCC2420_NO_ACKNOWLEDGEMENTS
CFLAGS += -DCC2420_NO_ADDRESS_RECOGNITION
CFLAGS += -DTASKLET_IS_TASK

include $(MAKERULES)
```

BIBLIOGRAPHIE

[1] A. Dhib, Routage avec QOS temps réel dans les réseaux de capteurs, thèse de Master, 2006-2007.

[2] K.B. Kredo and B.P. Mohapatra, Medium access control in wireless sensor networks, *Computer network 51(4)*, pp 961-994, 2007.

[3] D. Martins, Sécurité dans les réseaux de capteurs sans fil stéganographie et réseaux de confiance, Thèse de doctorat, 2010.

[4] C. LABORDE and F. DELGADO, Réseaux de capteurs sans-fil : Configuration et test de connectivité, Projet Tutoré, Mai 2010.

[5] S. Akyildiz, W. Su, Y. Sankarasubramaniam and E. Cayirci, A survey on sensor networks, *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102-114, Août 2002.

[6] S. MOAD, Optimisation de la consommation d'énergie dans les réseaux de capteurs sans fil, Rapport de stage, 2007-2008.

[7] N. LABRAOUI, La sécurité dans les réseaux sans fil ad hoc: Agrégation de données et Localisation, thèse de doctorat, 2012.

[8] K. Holger and A. Willig, Protocols and architectures for wireless sensor networks, John Wiley and Sons, 2005.

[9] M. LEHSAINI, Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique, Thèse de Doctorat, 2009.

[10] T. Vassileios, G. Alyfantis, T. Hasiotis, O. Sekkas and S. Hadjiefthymiades, Commercial wireless sensor networks: Technical and business issues. In 2nd International Conference on Wireless on Demand Network Systems and Service (WONS 2005), January 2005.

[11] S. KABOU and A. Belgourari, Etat de l'art sur les réseaux de capteurs sans fil, thèse de licence, 2009-2010.

[12] M. Ilyas and I. Mahgoub, Handbook of sensor networks Compact wireless and wired Sensing Systems, ISBN 08493196864. CRC PRESS LLS, USA, 2005.

[13] Y.C. Hu, A.Perrig, and David B. Johnson. Wormhole attacks in wireless networks. IEEE Journal on Selected Areas in Communications, 24(2):370–380, 2006.

[14] B. Parno, A. Perrig, and V. D. Gligor, Distributed detection of node replication attacks in sensor networks. In Proceedings of 2005 IEEE Symposium on Security and Privacy (S&P'05), 2005.

[15] C. Castelluccia et A.Francillon, Protéger les réseaux de capteurs sans fil, SSTIC2008, Juin 2008

[16] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler and K. Pister, System architecture directions for networked sensors, In Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 93-104, 2000.

[17] J. Lajoie, les fractales, Accromath vol1 été-automne 2006.

[18] Y. Lou, Y.Zhang and S.Liu , Single Hop Detection of Node Clone Attacks in Mobile Wireless Sensor Networks, Procedia Engineering, 29,pp.2798 – 2803, 2012.

[19] J. Newsome and D. Song, GEM: Graph embedding for routing and data-centric storage in sensor networks without geographic information. In ACM Conference on Embedded Networked Sensor Systems (SenSvs), 2003.

[20] L.M. Wang and Y. Shi, Patrol Detection for Replica Attacks on Wireless Sensor Networks, Sensors, 11, 2496-2504, 2011.

[21] Bo. Zhu, V.G.K. Addada, S. Setia, S. Jajodia, S. Roy, Efficient distributed detection of node replication attacks in sensor networks, IEEE transaction on MOBILE COMPUTING, 9 ET 7, 2010.

[22] M. Buvaneswari and Ms.T. Subha, IHONEYCOL: A COLLABORATIVE TECHNIQUE FOR MITIGATION OF DDoS ATTACK, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Special Issue 1, January 2013.

[23] M. Conti, R.Di. Pietro, L.V. Mancini and A. Mei, A Randomized Efficient and Distributed Protocol for the Detection of Node Replication Attacks in Wireless Sensor Networks, ACM MobiHoc, pp 80-89, 2007.

[24] C.T. KONE, Conception de l'architecture d'un réseau de capteurs sans fil de grande dimension, thèse de doctorat, 2011.

[25] F. Pouget, Système Distribué Capteurs de «pots de miel»: la discrimination et analyser corrélative des Processus d'Attaques, Thèse de doctorat, 2006

[26] H. Boussaid, M. Mkacher, E. Msallem and H. Tbourbi, Collecte de malwares avec un Honeypot, SECURIDAY 2009 SECURINETS Atelier, 2009.

[27] B.A. SALEM, Sécurisation des Réseaux Ad hoc : Systèmes de Confiance et de Détection de Répliques, thèse de doctorat, 2011.

[28] C. Gui and P.Mohapatra, Virtual Patrol: A New Power Conservation Design for Surveillance Using Sensor Networks. *Information Processing in Sensor Networks*, 2005.

[29] K.R. Meredith, a Survey and Comparison of Target Tracking and Detection Algorithms for Sensor Network, University of Binghamton

[30] V. Manjula and Dr.C. Chellappan, Replication attack mitigations for static and mobile WSN, International Journal of Network Security & Its Applications (IJNSA), Vol.3, No.2, March 2011.

[31] M. Conti, R.Di. Pietro, A. Mei, Emergent Properties: Detection of the Node-capture Attack in Mobile Wireless Sensor Networks, WiSec'08, 2008.

[32] A. Perrig, J. Stankovic, and D. Wagner, Security in wireless sensor networks. Comm. of ACM, 47(6):53–57, 2004.

[33] L. Eschenauer and V. Gligor, A key-management scheme for distributed sensor networks. In Proceedings of the ACM Conference on Computer and Communication Security (CCS), Nov. 2002.

[34] H. Chan, A. Perrig, and D. Song, Random key predistribution schemes for sensor networks, In Proceedings of IEEE Symposium on Security and Privacy, May 2003.

[35] A. Gallais, F. Ingelrest, J. Carle and D. Simplot-Ryl, Maintien de la couverture de surface dans les réseaux de capteurs avec une couche physique non idéale, CFIP, Colloque Francophone sur l'Ingénierie des Protocoles. 2006.

[36] A. Gallais, Ordonnancement d'activité dans les réseaux de capteurs : l'exemple de la couverture de surface, Rapport de thèse, 2007.

[37] A. Selatna, Implémentation d'une application orientée surveillance pour les réseaux de capteurs, thèse de master, 2012.

[38] H-C. LE, Optimisation d'accès au médium et stockage de données distribuées dans les réseaux de capteurs, Thèse de doctorat, 2008

[39] H. Bettahar and Y. Challal, Introduction à la sécurité informatique, Supports de cours : Systèmes Intelligents pour le Transport, 2008.

[40] S. ATHMANI, Protocole de sécurité Pour les Réseaux de capteurs Sans Fil, thèse de magister, 2010.

[41] M.L. Messai, Sécurité dans les Réseaux de Capteurs Sans-fil, Thèse de magister, Université de Sétif, 2008.

[42] E. Yoneki and J. Bacon. A Survey of Wireless Sensor Network Technologies: Research Trends and Middleware's Role. Technical Report UCAM-CL-TR646, 2005.

[43] L. Spitzner, Honeypots: Tracking Hackers, 1st edition. Addison-Wesley Professiona, 2002.

[44] F. Pouget, Système Distribué de Capteurs Pots de Miel: Discrimination et Analyse Corrélative des Processus d'Attaques, thèse de doctorat, 2005.

[45] Y. Alosefer and O. Rana, Honeyware: a web-based low interaction client honeypot, Ateliers STIC, 2010.

[46] M. Handy, M. Haase and D. Timmerman, Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection, IEEE Mobile and Wireless Communications Networks, 2002.

LISTE DES FIGURES

LISTE DES TABLEAUX