ⓒfinition

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A/Mira de Béjaia
Faculté des Sciences Exactes
Département d'Informatique

MÉMOIRE DE MASTER

En Informatique

Option

Réseaux et Systèmes Distribués

Thème

Gestion de clés pour les réseaux de capteurs sans fil

Présenté par : M^{lle} BELALOUACHE Imene
M^{lle} AMRANE Samia

Soutenu le 29 Juin 2016 devant le jury composé de :

Présidente	M^{me} L. Hamza	Maître Assistante A	U. A/Mira Béjaia.
Examinatrice	M^{me} N. Battat	Maître Assistante A	U. A/Mira Béjaia.
Examinateur	M. N. El sakaane	Doctorant LMD	U. A/Mira Béjaia.
Promoteur	Dr M. Omar	Maître de Conf. A	U. A/Mira Béjaia.
Co-promoteur	M. B. Abbache	Maître Assistant A	U. A/Mira Béjaia.

Béjaia, Juin 2016.

RÉSUMÉ

Les réseaux de capteurs mobiles intervient dans beaucoup de domaines tels que l'automobile, le secteur militaire ou encore le secteur de la santé et ce grâce à la miniaturisation des objets qui permet à ses équipements (capteurs) de s'adapter à tout les milieux, mais l'utilisation de technologies aussi petites, dotées de ressources limitées, déployées aléatoirement dans la nature nécessite d'une part une sécurisation des communications/matériel et d'autre part une optimisation de ressources. Un protocole de gestion de clés cryptographiques pour WSN efficace et optimisé offrant une sécurité infaillible à toute attaque répondrait parfaitement à ces deux exigences. Un protocole répondant à toutes ces exigences n'existant pas encore, le domaine de recherche reste ouvert. Dans ce travail, nous avons proposé le protocole PPBKM. Ce mémoire contient une partie théorique qui est une taxonomie de protocoles récents de gestion de clés et le détail du protocole proposé. Il contient aussi, une partie pratique de simulation des spécificités du PPBKM et de ceux qui nous ont servi de base de comparaison. Cette simulation a abouti à des résultats analysés et expliqués. Nous avons opté pour le logiciel de calculs numériques MATLAB comme outil de simulation pour sa capacité à effectuer des expériences de calcul très rapidement.

Mots clés : Gestion de clés, simulation Matlab, WSN, capteur.

ABSTRACT

The mobile sensor networks involved in many fields such as automotive, military or the health sector and this thanks to the miniaturization of objects that allows its devices (sensors) to adapt to all environments. But the use of technologies as small, with limited resources, randomly deployed in nature requires both a secure communications/ equipment and resource optimization. A cryptographic key management protocol for efficient and optimized WSNs providing foolproof security to any attack perfectly meet these two requirements. A protocol that perfectly meets all these requirements do not yet exist, the research area remains open. In this work, we proposed the PPBKM protocol. This report contains a theoretical part which is a taxonomy of recent key management protocols and the details of the proposed protocol. It also contains a practical part simulating the specifics of PPBKM and those who served us a basis for comparison. This simulation has led to results analyzed and explained. We opted for the numerical calculations MATLAB software as a simulation tool for its ability to make computing experiences very quickly

Key words : Key management, MATLAB simulation, WSN, sensor.

Dédicaces

A mes très chers parents. Aucune dédicace n'est assez éloquente pour vous exprimer ma gratitude pour les sacrifices que vous avez consentis depuis 23ans afin de m'offrir tout ce dont j'avais besoin pour m'épanouir. L'estime et le respect que j'ai toujours eu pour vous deux sont incommensurables et la chance de vous avoir est inestimable. Vous êtes et serez toujours un modèle pour moi. Je m'excuse pour le stresse occasionné et espère que ce travail vous rendra fiers.

A mes très chers grands mères "yemma", "setsi" pour leur gentillesse, leur tendresse et pour tout ce qu'elles représentent pour moi.

A mes regrettés grands pères "vava" et "jeddi" allah irhemhoum. Même si vous n'êtes plus la, je suis sure que vous veilliez sur moi et que vous êtes fières. Que Dieu vous accorde paix et repos.

A ma tante Malika pour son soutien moral inestimable. Cette dernière année n'aurait jamais pu être pareil sans toi.

A mes deux frères Amine et Sofiane. Vous avoir est une chance et la distance n'y changera rien.

A mon fiancé. Khaled tu dois certainement être (avec mes parents) celui qui a le plus subi mon stresse cette année ce qui ne t'a pas empêché d'avoir une patience irréprochable. Merci pour tout et LUBLU.

B. Imene.

A ma très chère mère. Aimable et honorable, tu représentes pour moi le symbole de la bonté par excellence, la source de tendresse et un exemple de dévouement. Tu n'as pas cessé de m'encourager et de prier pour moi, ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études. Je te dédis ce travail en témoignage de mon profond amour, puisse Dieu, le tout puissant, te préserver et t'accorder santé, longue vie et bonheur.

A mon père. Aucune dédicace ne saurait exprimer l'estime et le respect que j'ai toujours eu pour toi. Rien en monde ne vaut les efforts fournis jour et nuit pour mon éducation et mon bien être. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation.

Mon frère, mes soeurs. Votre soutien m'a été d'un grand secours au long de ma vie professionnelle et personnelle.

A tous mes oncles, tantes, voisins. Pour qui je le sais ma réussite est très importante.

A toutes mes amies et amis. En témoignage de l'amitié qui nous uni et des souvenirs de tous les moments que nous avons passé ensemble, je vous dédie ce travail et je vous souhaite une vie pleine de santé et de bonheur.

A. Samia.

Remerciements

***C**'est avec une certaine émotion et beaucoup de sincérité que nous voudrions remercier toutes les personnes ayant soutenu et apprécié notre travail.*

***C**'est tout naturellement que nos premiers remerciements s'adressent á Dieu le tout puissant qui nous a doté de la patience et de la volonté nécessaires pour mener ce travail á bon terme.*

***N**ous tenons tout particuliérement á remercier nos trés chers parents pour leur dévouement, leurs sacrifices et leur contribution directe ou indirecte. Tout au long de notre cursus, ils nous ont toujours soutenu, encouragé et aidé. Ils ont su nous donner toutes les chances pour réussir. Qu'ils trouvent, dans la réalisation de ce travail, l'aboutissement de leurs efforts ainsi que l'expression de notre plus affectueuse gratitude.*

***N**ous remercions également notre encadreur et notre co-encadreur en l'occurrence M. OMAR Mawloud et M. ABBACHE Bournane pour leur disponibilité, leur soutien et leurs conseils qui ont été importants. Il ont beaucoup oeuvré pour la mise en valeur de notre travail et pour cela il ont toute notre reconnaissance.*

***N**ous remercions les membres du jury en l'occurrence Mme HAMZA Lamia, Mme BATTAT Nadia et M. ESSAKAN Nadim pour l'intérét qu'ils ont porté á notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.*

***N**ous voudrions également remercier nos fréres et soeurs, nos amis et toutes les personnes qui ont contribué á la bonne marche de ce travail. Ainsi que toute la promotion 2015-2016 Master2 informatique de l'Université Abderrahman mira, notamment la section ReSyD.*

***N**ous remercions enfin toutes les personnes intéressées par notre travail, en espérant qu'elles puissent trouver dans notre rapport des explications utiles pour leurs propres travaux.*

Table des matières

Table des figures

Liste des tableaux

List of Algorithms

Liste des abréviations

WSN	Wireless Sensor Network
SB	Station de Base
CH	Cluster-Head
ID	Identifiant
CH_i	Identifiant du Cluster-Head i
L_i	Identifiant du capteur simple i

Introduction générale

A une époque où la micro-électronique et communications sont les maîtres-mots de notre monde, les réseaux de capteurs représentent une convergence parfaite entre les deux domaines. Les capteurs sont des systèmes autonomes miniaturisés, permettant de détecter, en vue de le quantifier et de le représenter, un phénomène physique sous la forme d'un signal, généralement électrique [1]. Les capteurs étant organisés sous forme de réseau déployé rapidement et arbitrairement, et ayant des ressources de stockage et d'énergie limitées, deux challenge considérables se posent aux chercheurs dans ce domaine : le premier est de permettre aux capteurs, malgré les ressources limitées de récolter des données et les faire parvenir à une station de base et le second est de pouvoir leur assurer un niveau de sécurité acceptable, malgré la nécessité de la majorité des applications qui utilisent les WSN d'offrir un déploiement dans des endroits peu sûrs, tels que les champs de bataille, les lieux stratégiques (aéroports, bâtiments critiques, etc.). La cryptographie répond tout à fait aux deuxième challenge en permettant de garder secrètes les informations transmises à travers le réseaux. Accompagnée de techniques optimisant la distribution, l'établissement et la gestion des clés cryptographiques utilisées, elle permet aussi de répondre au premier challenge. Divers protocoles basés sur la cryptographie asymétrique et d'autres sur la cryptographie symétrique existent dans la littérature, mais étant données les contraintes spécifiques des WSN, concevoir des protocoles propres et conformes ayants comme mettre mot la pré-distribution de clés et l'optimisation des ressources est indispensable.

L'objectif de ce mémoire a été de pouvoir proposer un protocole de gestion de clés pour les réseaux de capteurs sans fil permettant de jouer son rôle de capture de données tout en garantissant une économie de stockage et surtout d'énergie, qui représente la ressource la plus critique dont dépend essentiellement la durée de vie d'un RCSF. Nous avons pour cela, établi une taxonomie de différents protocoles de gestion de clés proposés pour les WSN. Nous les avons classé, puis évalué leurs performances et leurs aptitudes à optimiser des ressources déjà limitées.

Les démarches précédemment cités nous ont mené à organiser le présent mémoire en trois

principaux chapitres. Après cette introduction, nous consacrons le premier chapitre aux définitions utiles suivies d'une taxonomie la plus récente et concise possible des protocoles de gestion de clés existants, avec pour chacun une critique des performances globales. Tous ces protocoles ont été classé suivant une classification que nous avons nous même construite. Nous avons ensuite défini nos critères d'évaluation et nous avons évalué chacun des protocoles suivant ces critères dans un tableau récapitulatif. Nous avons clos le chapitre par une synthèse dévoilant nos déductions sur les meilleures techniques à utiliser pour établir un système de gestion de clés conforme aux exigences des WSN. Nous proposons le protocole PPBKM et son évaluation dans le second chapitre. Nous terminons par une simulation comparant notre solution avec celles des autres protocoles appartenant à la même sous-classe de protocoles de gestion de clés (protocoles gestion dynamique distribuée à capteurs mobiles).

Chapitre 1

Taxonomie de protocoles de gestion de clés pour WSN

1.1 Introduction

La gestion de clés se fait grâce à des mécanismes efficaces et sécurisés, qui gèrent les clés cryptographiques dont un système peut avoir besoin ; ce qui rend son existence primordiale pour la sécurité de n'importe quel système de communication, en particulier les WSN de par leurs contraintes notamment d'énergie. De ce fait, un grand défi se pose qui est la conception d'un système de gestion de clés se basant sur une technique cryptographique tenant compte des contraintes des WSN. Dans ce chapitre, nous définissons quelques concepts aux quels nous faisons appel dans la suite du chapitre, puis nous avons présenté notre classification, suivie par l'étude de plusieurs protocoles existants de gestion de clés pour les WSN. Nous avons clos ce chapitre par un tableau comparatif de tous les protocoles étudiés et une synthèse de toutes les déductions faites concernant les meilleures techniques et concepts à utiliser lors de l'élaboration d'un système de gestion de clés.

1.2 Définitions

Définition 1.2.1. *La gestion des clés représente un des aspects les plus difficiles de la configuration d'un système cryptographique de sécurité. Pour qu'un tel système fonctionne et soit sécurisé, chacun des utilisateurs doit disposer d'un ensemble de clés secrètes (dans un système à clés secrètes) ou de paire de clés publiques/privées (dans un système à clés publiques). Cela implique de générer les clés et de les distribuer de manière sécurisée aux utilisateurs ou d'offrir à l'utilisateur le moyen de les générer. Il doit aussi pouvoir enregistrer et gérer ses clés publiques et privées de manière sure [4]. La figure 1.1 montre toutes ces étapes.*

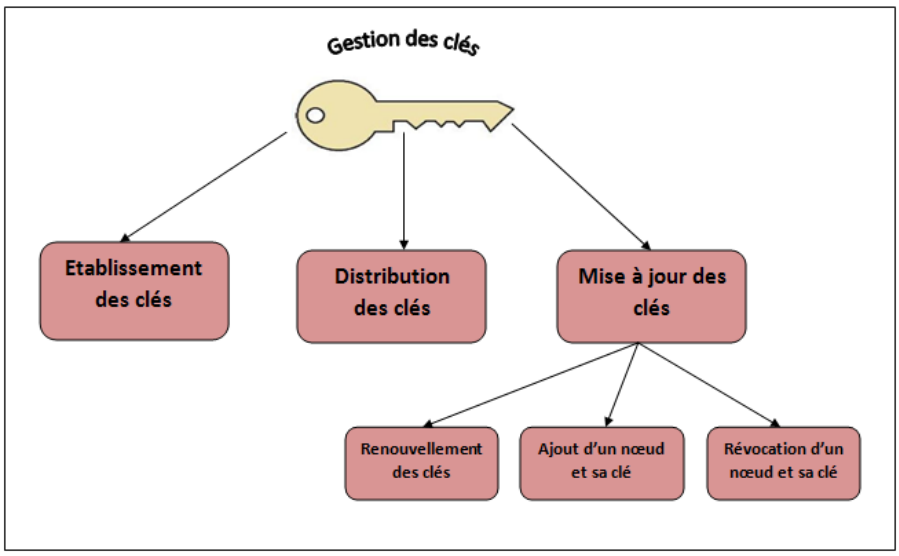

FIGURE 1.1 – Étapes de gestion de clés.

Définition 1.2.2. *Un protocole de gestion statique de clés est un ensemble de processus utilisant le principe de la pré-distribution de clés. Les clés sont fixées pour toute la durée du réseau. Toutefois, en tant que clés cryptographiques utilisées pendant une longue période, la probabilité qu'elles soient attaquées augmente de manière significative [20].*

Définition 1.2.3. *Un protocole de gestion dynamique de clés est un ensemble de processus utilisés pour effectuer les étapes de la gestion de clés citées dans la figure 1.1. Le renouvellement de clés se fait soit périodiquement ou sur demande en fonction des besoins du réseau [20].*

Définition 1.2.4. *Un protocole de gestion centralisée de clés est une sous-classe des protocoles de gestion dynamique de clés. Ils représentent un ensemble de processus qui utilisent un seul contrôleur central (SB ou tierce partie) globalement responsable de la gestion des clé [21].*

Définition 1.2.5. *Un protocole de gestion distribuée de clés est une sous-classe des protocoles de gestion dynamique de clés. Ils représentent un ensemble de processus dans les quels il n'y a pas de noeud contrôleur unique (SB, tierce partie, etc.) impliqué dans le processus de renouvellement des clés de capteurs. Au lieu de cela, la gestion des clés est assurée par plusieurs contrôleurs principaux (SB et Cluster Head, etc.),qui peuvent être soit prédéterminés ou attribués dynamiquement [21].*

Définition 1.2.6. *Un capteur mobile dans le reste de notre travail représente un capteur pouvant changer aléatoirement d'emplacement physique tout au long de la durée de vie du réseau de capteurs auquel il appartient.*

Définition 1.2.7. *Un capteur fixe dans le reste de notre travail représente un capteur immobile. Une fois déployé il ne peut plus changer d'emplacement physique jusqu'à la mort totale du réseau de capteurs auquel il appartient.*

1.3 Classification des protocoles de gestion de clés

Les protocoles de gestion de clés pour les RCSF ont été largement étudiés, donnant ainsi naissance à beaucoup de classifications. Pour notre classification illustrée sur la figure 1.2, nous nous sommes basé sur l'une d'entre elles proposées par He et al. [21] puis nous avons associé nos propres critères pour obtenir une classification plus complète.

Le premier niveau de cette classification se base sur le type même de gestion de clés (statique ou dynamique). Ce critère est très important de part son aspect très structurant et dénotant le comportement réel d'un système de gestion de clés.

Le second niveau se base sur le comportement et le rôle réel des noeuds et sur la façon qu'ont les noeuds voisins de partager des clés (centralisé ou distribué pour la gestion dynamique de clés/ probabiliste ou déterministe pour la gestion statique de clés). Ce critère nous a semblé important pour affiner la classification et bien faire la part des choses entre les détails de chaque sous-classe de protocoles.

Nous avons ensuite développé notre propre niveau qui représente le type des noeuds du réseau (fixes ou mobiles).

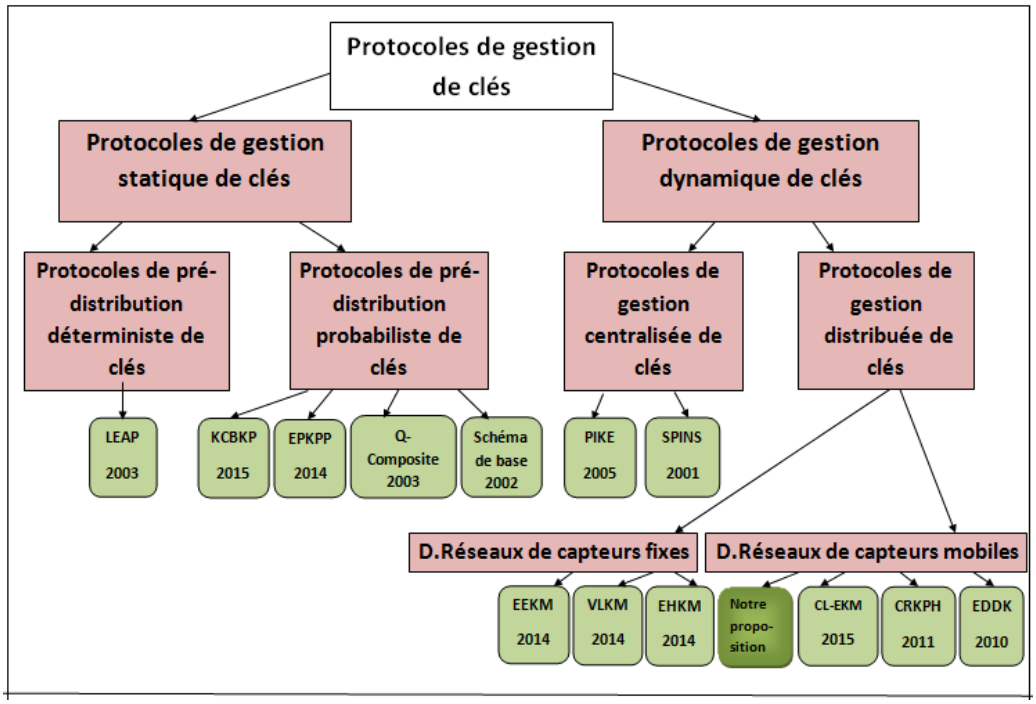

FIGURE 1.2 – Classification de protocoles de gestion de clés.

1.4 Protocoles de gestion statique de clés

Les protocoles de gestion statique de clés ont fait leur succès à leur apparition, mais ne sont pas considérés assez complets pour assurer une gestion de clés car une fois les phases de pré-distribution de clés et d'établissement de clés passées, le renouvellement de clés ne se fait à aucun moment.

1.4.1 Protocoles de pré-distribution probabiliste de clés

Des chaînes de clés sont aléatoirement choisies parmi un ensemble de clés et distribuées sur les capteurs du réseau. Offrant ainsi une connectivité dépendante d'une probabilité forte ou faible (suivant le protocole).

1.4.1.1 Protocole efficace de pré-distribution de clés pour les WSN (EPKPP)

Le protocole de pré-distribution proposé par Kun et Li [14] est basé sur la théorie aléatoire et la théorie des probabilités avec l'utilisation de fonctions de hachage. Dans La phase de pré-distribution un pool de clés est généré avec chaque clé son identifiant. Chaque noeud, est

doté d'une partie des clés de ce pool et les stocke dans sa propre mémoire. Afin de garantir une communication sécurisée entre deux noeuds la découverte de clés partagées sera effectuée ; Pour cela, les capteurs découvrent leurs voisins avec lesquels ils partagent au moins une clé commune en diffusant les identifiants des clés (leur partie du pool).

La particularité de ce système réside dans le fait que les capteurs sont dotés d'un niveau de sécurité différent, et avec la propriété de calcul à sens unique de la fonction de hachage, les capteurs compromis ne révèlent pas les informations des clés des capteurs d'un niveau au dessus du leur. Le coût de stockage lui est assez important étant donné le nombre de clés devant être enregistrées par chaque noeud. Le coût de communication est aussi élevé pendant la phase de découverte des voisins et le coût de calcul est relativement élevé à cause de l'application de la fonction de hachage a sens unique qui engendre beaucoup de calculs.

1.4.1.2 Protocole de pré-distribution basé sur les chaînes de clés (KCBKPP)

Hazra et al. [10] ont proposés un système de pré-distribution de clés basé sur des chaînes de clés hachées, tel qu'avant déploiement un centre de génération de clés utilise une fonction de hachage pour établir une clé partagée secrète avec d'autres noeuds en se basant sur la génération de chaînes de clés linéaires. Cela se déroule comme suit : C chaînes de clés sont générées tel qu'une chaîne de clés Ch_i est formée en fusionnant deux chaînes de hachage de longueur égale. Supposons qu'on veuille avoir la chaîne de clés Ch_1 et qu'on aie deux chaînes de hachage nommées Ch_{1a} et Ch_{1b} tel que :

$Ch_{1a} => h^1(s_{1a})h^2(s_{1a})...h^l(s_{1a})$ et

$Ch_{1b} => h^l(s_{1b})h^{l-1}(s_{1b})...h^1(s_{1b})$

ou h est une fonction de hachage, s_{1a} et s_{1b} deux valeurs aléatoires de départ, la puissance du $h$représente le nombre de fois ou la fonction de hachage h est appliquée. Donc, la chaîne de clés Ch_1 est produite par la concaténation des deux chaînes de hachage : $Ch_1 = ((Ch_{1a})^i||(Ch_{1b})^i)$. Comme la longueur de la chaîne de hachage est l alors la longueur de la chaîne de clés est aussi égale à l noté $(Ch_1, Ch_2, ..., Ch_l)$. Ensuite, des C chaînes de clés générées, un ensemble de porte-clés différents de taille m sont calculés puis chacun est affecté à un noeud du réseau. Les noeuds diffusent les identifiants des clés de leur porte clé la liste de les clés communes entre chaque deux noeuds leur serviront de clés de paires.

Si un adversaire compromet deux valeurs de clés d'une chaîne, il aura deux valeurs dans chaque chaîne de hachage qui a servi à la génération de la chaîne de clés en question. Il pourra donc calculer toutes les autres valeurs de la chaîne de hachage et donc facilement calculer toutes les clés secrètes de la chaîne de clés ce qui augmente le risque de compromettre tout le réseau. Le coût de stockage de ce protocole est très élevé étant donné le nombre de clés qu'un noeud doit stocker mais le coût de calcul est relativement bas. Une fois les clés de paires établies, il

n'y a plus de communications autres que le transfert de données entre 2 capteurs donc le coût de communication est plutôt bon dans ce protocole.

1.4.1.3 Protocole aléatoire de pré-distribution de clés (Schéma De Base)

Eschenauer et Gligor [8] ont proposé un schéma de gestion de clés basé sur la probabilité de partager une clé entre les noeuds d'un graphe aléatoire. Ce schéma fournit des techniques pour la pré- distribution de clés, la découverte de la clés partagées, l'établissement de chemins de clés, et la révocation de clés. Le principe de ce protocole est de distribuer aléatoirement un certain nombre de clés, issues d'un ensemble fini, à chaque noeud du réseau avant son déploiement. Deux noeuds quelconques seront en mesure de s'échanger des messages sécurisés s'ils possèdent une clé commune. Un grand ensemble S de clés est généré. Pour chaque noeud, m clés sont choisies au hasard de l'ensemble S ($S = (kid_1, key_1), (kid_2, key_2), ...$). Ces m clés sont stockées dans la mémoire du noeud et forment le trousseau de clés du noeud. Le nombre de clés |S| de l'ensemble est choisi de telle manière que deux sous-ensembles aléatoires de S de taille m auront une certaine probabilité p d'avoir au moins une clé en commun, par exemple pour une probabilité $p = 0.5$ on a besoin d'un sous ensemble de taille $m = 75$ clés de l'ensemble S de taille $|S| = 10,000$ clés. Les noeuds découvrent leurs voisins et plus particulièrement ceux avec qui ils sont en mesure de communiquer de façon sécurisée car ils possèdent une clé identique dans leur sous-ensemble de clés respectif. Chaque noeud va diffuser la liste des identités des clés de son sous-ensemble. Une des clés partagées devient la clé de session du lien entre les deux noeuds. La révocation d'un noeud compromis se fait par l'élimination de son sous-ensemble de clés. Pour cela, un noeud contrôleur (qui a une grande connectivité et peut être mobile) annonce un message simple de révocation contenant une liste des identificateurs des clés pour que ces clés soient retirées des sous-ensembles de clés des autres noeuds.

Le coût de stockage est très élevé étant donné le nombre de clé à stocker par chaque noeud sans autant rendre le protocole plus sécurisé car une seule des clés communes servira de clé partagée et donc sa compromission repose sur la probabilité de trouver la clé choisie comme clé partagée entre toutes les clés communes.

1.4.1.4 Protocole de pré-distribution aléatoire de clés (Q-Composite)

Chan et Perrig [6] ont proposé le protocole Q-Composite qui est identique à celui de Eschenaur et Gligor [8] sauf qu'au lieu d'exiger le partage d'une clé commune pour sécuriser un lien, une paire de noeud doit partager q clés avec q > 1 pour établir un lien sécurisé. La nouvelle clé utilisée pour la communication entre ces deux noeuds est le haché de toutes les clés partagées, par exemple pour deux noeuds quelconques qui partagent q' clés ($q' \prec q$) la clé utilisée pour la communication est $K = hash(k_1||k_2||...||kq')$.

Plus le nombre de clés partagées augmente plus la résilience contre la capture du noeud augmente. Autrement dit, lorsque le nombre exigé de clés partagées augmente, il devient plus difficile à un attaquant avec un ensemble donné de clés de casser un lien. Cependant, pour préserver une probabilité donnée p que deux noeuds partageant des clés suffisantes pour établir un lien sécurisé, il est nécessaire de réduire la taille de l'ensemble de clés S, ce qui permet à un attaquant de compromettre un plus grand échantillon de S en compromettant peu de noeuds.

1.4.2 Protocoles de pré-distribution déterministe de clés

Ils assurent le fait que chaque noeud soit capable d'établir une clé avec chacun de ses voisins à partir d'une seule clé (ou autre paramètre) pré-distribuée. Offrant ainsi une connectivité parfaite de 100%.

1.4.2.1 Mécanismes efficaces de sécurité pour réseaux de capteurs grande échelle distribués (LEAP)

le protocole LEAP proposé par Sencun, Zhu et al. [17] est un protocole déterministe de gestion de clés pour les réseaux de capteurs sans-fils. LEAP prend en charge l'établissement de quatre types de clés pour chaque noeud : clé individuelle, clé de paire, clé de groupe et clé globale.

LEAP est basé sur une clé initiale transitoire KIN chargée dans chacun des noeuds du réseau. Le protocole LEAP suppose que pour compromettre un noeud, l'adversaire nécessite un temps minimal $Tmin$. Il exploite ce temps (de confiance) pour permettre à deux noeuds voisins d'établir d'une manière sécurisée une clé symétrique de session à partir de la clé initiale transitoire KIN. Le contrôleur (SB) génère une clé initiale KIN et charge chaque noeud avec cette clé avant son déploiement. Chaque noeud u dérive une clé principale (Maser Key) $Ku = fKIN(u)$ avec f une fonction pseudo-aléatoire. Immédiatement après son déploiement, le noeud u essaye de découvrir ses voisins en diffusant un message HELLO qui contient son identifiant. Aussi, il initie un timer. Le noeud u attend un ACK de chacun de ses voisins v. l'ACK est authentifié en utilisant la clé principale Kv qui est dérivée comme suit $Kv = fKIN(v)$. Comme le noeud u possède la clé KIN, il pourra aussi dériver Kv, ainsi il pourra vérifier l'authenticité des ACK reçus. Ensuite Le noeud u calcule sa clé de paire Kuv avec v comme suit $Kuv = fKv(u)$. Le noeud v peut de même calculer Kuv de la même manière.Lorsque le timer expire après $Tmin$, le noeud u efface KIN et toutes les clés principales Kv de ses voisins [17].

Un adversaire écoutant tout le trafic ne pourra à aucun moment intercepter la clé KIN, étant donné qu'elle n'est envoyée à aucun moment. Un adversaire compromettant un noeud

après *Tmin*, obtient seulement les clés du noeud compromis et ne pourra à aucun cas obtenir celles d'autres noeuds. Quand un noeud compromis est détecté, ses voisins suppriment simplement les clés de paires avec ce dernier. Malgré les mesures de sécurité citée, LEAP reste un protocole statique qui ne contient pas de renouvellement de clés, ce qui le rend extrêmement in-sécurisé. Si malgré tout plusieurs noeuds sont compromis, tous le réseau sera in-sécurisé. Le coût de communication est plutôt important pendant la phase de découverte des voisins car tous les HELLO et AKC sont transmis à ce moment la, puis faiblit ensuite car chaque noeud calcule à son niveau les clé de paires et donc ne génère aucune communication mais engendre un coût de calcule plutôt élevé.

1.5 Protocoles de gestion dynamique et distribuée de clés pour capteurs fixes

L'aspect dynamique assure un rafraîchissement à temps régulier des clés ou à la demande du réseau ce qui accroît la sécurité. L'aspect distribué offre une meilleure évolutivité et une révocation/distribution de clés plus rapide. Les capteurs dans cette sous-classe sont totalement immobile et ce jusqu'à la fin de vie du réseau de capteurs.

1.5.1 Protocole de gestion de clés efficace pour les réseaux de capteurs dynamiques (VLKM)

Vaid et Katiyar ont proposé le protocole VLKM qui est un protocole de gestion de clés pour réseaux de capteurs sans fil fixes, qui utilise le principe des emplacements virtuels pour générer les clés utilisées par le RCSF sur le quel on l'applique [20].

Le protocole VLKM opère sur les WSN à clustering composés de capteurs simples, de Cluster-Heads (CH) et d'une SB. Il procède à la gestion de 2 types de clés dans le réseau : une clé de capteur et une clé de cluster.

Ce protocole procède comme suit [20] : Chaque capteur est pré-chargé d'un emplacement virtuel initial, d'un ongle de mouvement, d'une vitesse de mouvement et de limites physique ou il est libre de se déplacer virtuellement. Chaque réseau admet une origine virtuelle ou les clusters cartographient leurs emplacements virtuels. Par exemple, si l'emplacement virtuel d'un cluster est (5,2) et l'origine virtuelle du réseau est (3,2) alors l'emplacement virtuel de ce cluster après la cartographie est (3+5,2+2). Les capteurs, cartographient leurs emplacements virtuels suivant l'emplacement virtuel de leur CH. Chaque capteur/CH dispose d'un emplacement virtuel initial, un ongle virtuel de mouvement et une direction virtuelle de mouvement que suivent tous ces capteurs lors de leur déplacement virtuel. Le CH applique une fonction de hachage sur

son emplacement virtuel actuel (après avoir cartographié son emplacement initial sur l'origine virtuelle du réseau) ainsi que son identifiant pour produire la clé de cluster. Un capteur applique une fonction de hachage sur son emplacement virtuel (après avoir cartographié son emplacement initial sur l'emplacement virtuel de son CH) ainsi que son identifiant pour produire la clé capteur. Dans un cluster à chaque fois qu'un nouveau capteur est élu CH tous les capteurs mettent à jours leurs emplacements virtuels en se cartographiant sur le nouveau CH et ainsi les clés de capteurs changent. Comme le CH a changé donc la clé de cluster change aussi. Dans un réseau dès que tous les capteurs sont élus CH l'origine virtuelle du réseau est mise à jour et les CHs mettent à jours leurs emplacement virtuels sur cette nouvelle origine et ainsi les clés de clusters et de capteurs changent.

Le VLKM réduit de manière significative le nombre de transmissions nécessaires à la régénération de clés mais augmente le coût de stockage car un noeud doit enregistrer les vitesses de mouvement, les ongles de mouvement, etc. Ce protocole assure aussi qu'une clé compromise à un certain tour n'affectera pas les autres dans le future ni dans le passé car si l'emplacement virtuel actuel est compromis beaucoup d'autres paramètres qui ne sont jamais communiqués (ongles de mouvement, vitesse de mouvement...) sont pris en compte pour générer le nouvel emplacement donc ça devient impossible pour un attaquant de prévoir le prochain emplacement virtuel. Même si tous les paramètres de la génération de clés sont compromis il n'y aura qu'un noeud ou au maximum un cluster qui sera compromis car par définition du VLKM les paramètres de génération de clés sont différents entre chaque deux clusters et entre chaque deux noeuds.

1.5.2 Protocole de gestion de clés efficace et hybride (EHKM)

Zhang et Ji ont protposé le protocole EHKM qui est un protocole de gestion de clés hybride pour les réseaux de capteurs sans fil fixes hétérogènes. Pour l'établissement des clés, ce protocole est basé sur le principe de Deffie-Hellman [2] et de la cryptographie à courbe elliptique (ECC) [23].

Le protocole EHKM opère sur un réseau de capteur hétérogène composé d'une SB, de Cluster-Heads (CH) et de capteurs simples. Il procède à la gestion de quatre types de clés : paires de clés pub/priv, clés de paires (capteur et CH), clés de sessions (entre deux capteurs) et les clés de cluster.

Le protocole procède comme suit [23] : La SB est pré-chargée avec les clés publiques de tous les CH et sa propre paire de clés. Chaque CH et est pré-chargé avec la clé publique de tous les capteurs de son cluster, sa propre paire de clé, la clé publique de SB ainsi qu'une fonction de hachage. Un capteur se voit attribué un identifiant et est préchargé avec sa clé privée, la clé publique de son CH et une fonction de hachage. Une clé de paire différente est établie

entre chaque capteur et son CH. Grâce à un nombre aléatoire que le CH génère et diffuse sur tous les capteurs de son cluster deux capteurs peuvent calculer une clé de session Kuv (avec u et v capteurs) ainsi la clé de cluster $K0$. Quand un capteur est capturé, Le CH diffuse un message de révocation contenant l'identifiant du noeud compromis. A la réception de ce message, les capteurs vérifient s'ils communiquent avec le noeud compromis. Si oui alors toutes les clés partagées seront révoquées. Les clés partagées doivent être mises à jour périodiquement. Après une tranche de temps T, les CH génèrent un nouveau nombre aléatoire. Pour économiser de l'énergie la mise à jour de clé se fait de deux façons différentes soit qu'il y aie un noeud compromis ou non. Si le réseau est sain, les clés et la procédure de génération de clé restent les mêmes avec un nouveau nombre aléatoire r' suivant la figure 1.3. Sinon la clé de cluster $K0$ est considérée exposée donc inutilisable et est remplacer par la clé de paire et le nouveau nombre aléatoire r' suivant la figure 1.4

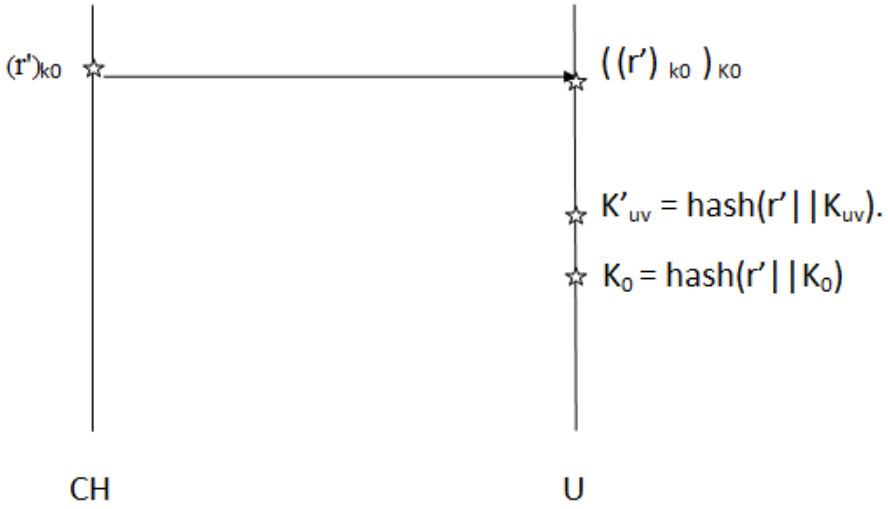

FIGURE 1.3 – Mise à jour EHKM si le réseau est sain.

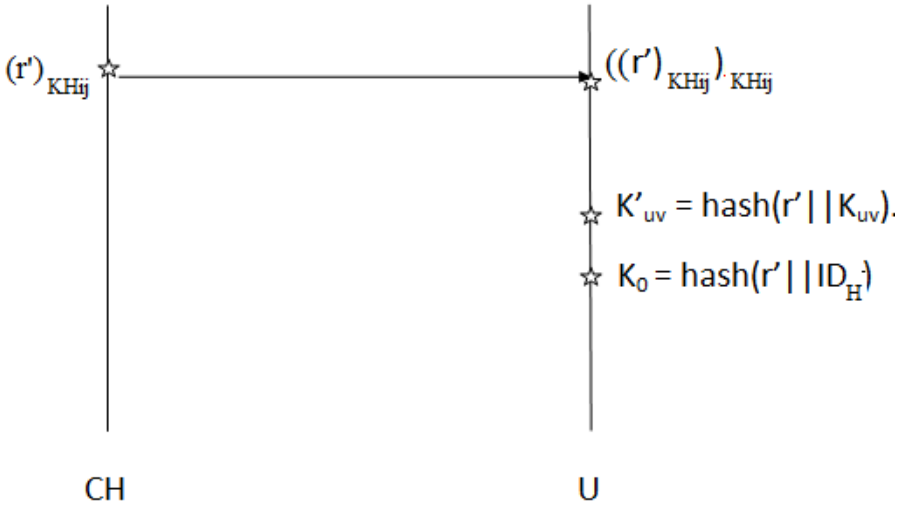

FIGURE 1.4 – Mise à jour EHKM si un noeud est compromis.

Supposons qu'un noeud S veut rejoindre le réseau. S est pré-chargé de sa clé privée, de la clé publique de son CH et d'une fonction de hachage. selon l'emplacement de S, la SB peut savoir à quel groupe il appartient et informe le CH du groupe, le CH informe les noeuds qui doivent établir une clé de session avec S.

1.5.3 Protocole de gestion de clés efficace en énergie (EEKM)

Suganthi et Sumathy [19] ont proposé le protocole EEKM qui est un protocole de gestion de clés pour WSN fixes utilisant des fonctions polynomiales non gourmandes en énergie pour la gestion de toutes les clés.

Dans tout ce qui suit, Ka est la clé individuelle du noeud a, Kia est la clé initiale du noeud a, IDa est l'identifiant du noeud a, Kab la clé de paire entre les deux noeuds a et b, Fid est l'identifiant de la fonction utilisée parmi l'ensemble de fonctions polynomiales, Rn est un nombre aléatoire communiqué par la SB, Kg la clé de groupe et Gk le générateur de clé de groupe présent dans chaque noeud [19].

Le EEKM opère sur un réseau de capteurs homogène composé de capteurs simples et d'une SB. Il procède à la gestion de 4 types de clés : les clés initiales, chaque noeud en possède une qui sera utilisée pour générer une clé individuelle, les clés individuelles sont propre à chaque noeud, elle le relie avec la SB et sert à crypter les communications avec elle et ce grâce à $Ka = Rf(Kia + IDa)$, les clés de paires, cette clé est utilisée pour s'assurer que le message vers le noeud voisin destination ne sera pas dévoilé à d'autres noeuds voisins et ce grâce à $Kab = Fid(Rn, IDa)$ et la clé de groupe, tel qu'un générateur de clés de groupe se charge de la générer localement dans chaque noeud. Elle est partagée par tous les noeuds du réseau et est

utile pour crypter un message de sécurité provenant de la SB et ce grâce à $Kg = Fid*(Rn, Gk)$. Chaque noeud est pré-chargé d'une fonction pseudo aléatoire Rf et de son identifiant.

Le protocole procède comme suit [19] : La SB communique un nombre aléatoire Rn et l'identifiant d'une fonction polynomial Fid à tous les noeuds avant leur déploiement puis un timer est chargé pour assurer le rafraîchissement quand il atteindra son seuil.

Génération des clés de paires : la fonction polynomiale prend le nombre aléatoire et l'identifiant du noeud qui amorce la communication comme entrée pour calculer la clé de paire ; donc après que l'initiateur aie calculé la clé de paire il envoie son identifiant à l'autre partie de la paire qui aura donc tous les paramètres pour la calculer aussi.

Génération des clés de groupes : un générateur de clé de groupe est présent dans chaque noeud. Il s'occupe de calculer la clé de groupe commune à tous les noeuds.

Les clés de paires et la clé de groupe peuvent être mise à jour dans le cas de noeuds compromis ou dans le cas ou le timer est atteint, la mise à jour de la clé de groupe se fait grâce aux mêmes formules de calcule précédentes mais en changeant l'identifiant de la fonction polynomiale par un autre et ce afin de rafraîchir la clé et augmenter la sécurité.

Dans le EEKM l'utilisation des fonctions polynomiales qui ne sont pas gourmandes en énergie ce qui réduit la consommation d'énergie. Le coût de transmission est optimisé par le fait qu'aucune clé n'est transmise à la place des paramètres moins coûteux tels que les Fid, les Rn et les identifiants sont transmis. Par contre ce protocole oblige un noeud à stocker les paires de clés, le Rn, la fonction pseudo-aléatoire, la clé de groupe, un jeu de fonctions polynomiales pour le changement de clé ce qui est très coûteux en terme de coût de stockage mais renforce la sécurité car étant donné qu'il y à un jeu de fonctions polynomiales et qu'à chaque tour le noeud en choisi une aléatoirement pour calculer les clés, les adversaires seront incapables de savoir laquelle est utilisée à un moment donné.

1.6 Protocoles de gestion dynamique distribuée de clés pour capteurs mobiles

Comme la sous-classe précédente, l'aspect dynamique assure un rafraîchissement à temps régulier des clés ou à la demande du réseau ce qui accroît la sécurité. L'aspect distribué offre une meilleure évolutivité et une révocation/distribution de clés plus rapide. Les capteurs dans cette sous-classe sont apte à changer aléatoirement d'emplacement physique et ce à plusieurs reprise jusqu'à la fin de vie du réseau de capteurs.

1.6.1 Protocoles de gestion de clés déterministe efficace en énergie (EDDK)

Le protocole EDDK proposé par Xing, Jingsha et Qian [22] est un protocole de gestion de clés avec certificat pouvant supporter la mobilité des capteurs dans un réseaux de capteurs sans fil.

Le protocole EDDK opère sur un réseau de capteurs a clustering composé de capteurs statiques simples, de capteurs mobiles simples, de CHs, et d'une SB et d'un autorité de certification TA. Il procède à la gestion de deux types de clés : les clés de cluster et les clés de paires.

La paires de clés publique/privée de l'TA : avant le déploiement du réseau, la partie publique de cette paires de clés publique/privée est chargée dans chaque noeud ce qui va servir pour les opérations de certification.

La clé initiale : spécifique à chaque noeud, elle y est préchargée avant déploiement avec une fonction pseudo aléatoire et servira au calcule de la clé individuelle qui elle servira au calcule des clés de paires.

Clés de paires : entre deux noeuds différents. Un noeud peut dynamiquement générer cette clé grâce à sa clé individuelle.

Clés de clusters : partagée entre les noeuds d'un même cluster, une clé de cluster permet de sécuriser les communications dans le cluster telles que la diffusion de l'information de changement d'état d'un noeud (il devient CH, noeud sortant, etc.).

Le protocole procède comme suit [22] : Chaque noeud est préchargé d'une fonction pseudo-aléatoire et d'une clé initiale qui serviront après déploiement à calculer une clé individuelle. Les noeuds diffusent un message JOIN pour déterminer les relations avec leurs voisins et établissent de ce fait des clés de paires, suite à quoi une table des voisins contenant les identifiants des voisins et les clés de paires est établie au niveau de chaque noeud. Un numéro de séquence incrémenté à chaque envoie ayant une limite définie accompagne chaque message. Quand cette limite est atteinte la partie des clés de paires de la table des voisins est mise à jour avec la diffusion d'un message CLE-PAIRE-MSG et les clés de clusters sont mises à jour avec la diffusion d'un message CLUSTER-KEY-MSG. Quand un noeud est compromis ses voisins suppriment de leurs tables des voisins respectives les clés de paires les reliant à ce dernier. La clé de cluster au quel le noeud compromis était relié est mise à jour. Lors de l'ajout d'un noeud il diffuse un REPLAY-MSG qui servira à un noeud déjà existant dans le réseau de pouvoir calculer la clé de paire partagée avec lui. Comme ce protocole supporte la mobilité, l'ajout d'un noeud mobile est fait suivant les mêmes procédure que l'ajout d'un noeud fixe. Les noeuds mobiles étant considérés comme les plus susceptibles d'être malicieux une limite à la table des voisin à été définie à fin de limiter les

noeuds mobiles pouvant y être rajouter et ainsi limiter les risques de compromission du réseau. Un time out est chargé au début du déploiement et une fois atteint toutes les clés sont mises à jour malgré l'absence de noeuds compromis. Ce protocole offre la possibilité d'utiliser ECC (elliptic curve cryptography) pour réduire le coût de calcule étant donné qu'ECC offre des clés très courtes.

La sécurité au niveau du protocole EDDK se base sur la sécurité des multiples clés qu'il gère : pour compromettre le réseau il faudrait avoir accès aux clés initiales, individuelles, de paire, de cluster et à la clé privée de l'autorité de certification ce qui réduit de beaucoup les risques. La limitation de la taille des tables de voisins réduit aussi les risques dus à l'ajout de noeuds mobiles. La présence du time-out rafraîchi le réseau. Quant au numéro de séquence accompagnant chaque message il sécurise le réseau contre les attaques de rejoue.

Ce protocole offrant la possibilité d'utiliser ECC comme moyen de cryptage, ainsi le coût de calcule peut être réduit. Le coût de communication est aussi réduit par le fait qu'un noeud aie besoin d'envoyer un seul message JOIN au début puis calcule tout à son niveau sans avoir besoin de réponse des autres. Le coût de stockage en revanche est plutôt élevé car un noeud doit stocker sa clé initiale, sa clé individuelle, sa table des voisins (identifiants+clés de paires), la clé de cluster et la clé privée de l'autorité de certification.

1.6.2 Protocoles de gestion centralisée de clés pour les WSN (CRKPH)

Le CRKPH proposé par Mu et Li [13] est un protocole de gestion de clés supportant la mobilité des noeuds. Il passe par l'établissement de liens sures entre les capteurs et leurs CH respectifs puis il s'aider de ces derniers pour établir des liens sécurisés entre eux mêmes. CRKPH opère sur un réseau de capteurs à clustering composé de capteurs simples, de Clusters-Heads (CH) et d'une SB. Il procède à la gestion des clés suivantes : les clés de base pré-chargées dans chaque noeud, les clé dérivées qui sont calculées à partir de clés de base, ainsi les clés de cluster [13].

Le CRKPH procède comme suit [13] : Pour la pré-distribution de clés, un pool de clés de base est généré. Chaque capteur simple et chaque CH est préchargé de K et C clés de base (respectivement) qu'il utilisera pour établir des liens avec les capteurs voisins. La dérivation de nouvelles clés sera effectuée comme suit : la SB définit des niveaux (un niveau représente le périmètre autour d'un noeud tel que chaque niveau est de 40m) puis envoi à chaque CH la liste des niveaux avec la distance par rapport à la SB de chaque niveau, les CHs à leur tour envoient cette liste aux noeuds du cluster, ensuite les noeuds en utilisant ses clés de base ainsi que les niveaux calculent les clés dérivées correspondantes, avec pour chaque noeud, K clés dérivées par niveau. Pour l'établissement d'un lien sécurisé avec le CH : chaque noeud envoi à son CH la

liste des identifiants des clés de base ainsi que les identifiants des niveaux, Le CH répond pour chacun d'eux par une liste contenant les clés communes des $sous_pools$ de clés de chaque noeud. Pour l'établissement de liens sécurisés entre les noeuds chaque capteur A envoie son identifiant ID_A et l'identifiant du voisin ID_B à son CH qui génère un lien sécurisé entre A et B suivant deux cas :

- Si A et B appartiennent au même cluster alors le CH commun envoie la clé partagée au deux.
- Si B n'appartient pas au cluster de A alors le CH_a génère une clé partagée l'envoie à A et passe par le CH_b pour l'envoyer à B.

Quand un noeud mobile quitte son CH, il lui envoie un message LEAVE-REQ qui avertit les autres noeuds pour qu'ils rompent le lien entre eux et le noeud sortant, Le CH lui transmet avant son départ une clé qui lui servira a s'authentifier dans le cas ou il revient.

Quand un noeud rejoint un cluster si c'est un cluster qu'il a déjà quitté le CH de celui-ci identifie le noeud arrivant grâce à la clé qu'il lui a remise avant son départ le fait authentifier par la SB puis la phase d'établissement de liens précédemment citée s'exécute. Si au contraire le cluster qu'il veut rejoindre ne le connaît pas, Le CH de ce dernier transmet l'identifiant de ce noeud arrivant à la SB qui génère elle même un lien partagé entre l'arrivant et le CH en question puis la phase d'établissement de liens entre ce noeud arrivant et les autre s'exécute.

Le protocole CRKPH à un coût de stockage relativement élevé étant donné le nombre de clé qu'un noeud doit stocker (K, $K * niveau$ clés dérivées,...). Le coût de calcule est plutôt élevé aussi étant donné le fait que les clés dérivées sont calculée sur la base de cryptage et de fonction de hachage et ce au niveau de chaque noeud. La sécurité est assurée par le fait qu'il y aie un nombre de clés important et que la probabilité de compromettre une clé est faible.

1.6.3 Protocole de gestion efficace de clés pour les réseaux de capteurs dynamiques (CL-EKM)

Le protocole CL-EKM proposé par Seung-Hyun, Jongho et al. [18] est un protocole de gestion de clés sans certificats.

CL-EKM opère sur un réseau de capteurs à clustering composé d'un certain nombre de capteurs, de Clusters-Heads (CH) et d'une SB. Il procède à la gestion de cinq types de clés : les paires de clés publique/privée, les clés individuelles, les clés de paire maîtresses, et les clés de paire cryptographiques, ainsi que les clés de clusters.

Paires de Clés publique/privée : avant le déploiement du réseau, les paires de clés publiques/privées sont générées par un centre de génération de clés et installées dans chaque

noeud. Ces paires de clés sont utilisées après déploiement pour générer les clés de paires maîtresses.

Clés de paires(maîtresses et cryptographiques) : entre deux noeuds voisins, cette clé permet à un capteur de rallier un cluster en toute sécurité. Un noeud peut dynamiquement générer cette clé grâce à sa paire de clés publique/privée.

Clés individuelles : comme son nom l'indique cette clé est individuelle. Chaque noeud en possède une qu'il partage uniquement avec la SB et qu'il utilise pour l'échange de données sensibles ou de messages d'alerte.

Clés de clusters : partagées entre les noeuds d'un même cluster. Une clé de cluster permet de sécuriser les communications dans le cluster telles que la diffusion de l'information de changement d'état d'un noeud (il devient cluster-Head, noeud sortant, etc.).

Le protocole procède comme suit [18] : Configuration du système : La SB enregistre les noeuds en leur affectant des identifiants uniques, puis les paires de clés sont générées (principe de ECC [9])par le centre de génération de clés et installées dans chaque capteur.

Formation de clusters : Après déploiement, chaque CH détecte ses voisins en envoyant des messages de balises reçus qui indiquent la distance. Il procède à leur authentification en envoyant une liste de ces noeuds à la SB, qui elle est dotée d'un système de détection d'intrusions et de noeuds compromis et qui est tout à fait apte à procéder à des authentifications. La SB effectue ensuite une validation en avertissant le CH concerné du statut de chaque noeud. Après cette étape le CH diffuse sa clé de cluster sur toute la liste.

Génération des clés de paires : une fois que les clusters sont formés, si deux noeuds voisins voulant communiquer, ils établissent d'abord une clé de paire maîtresse de laquelle sera dérivée la clé de paire cryptographique.

Mouvement des noeuds : un noeud peut soit quitter un cluster soit le rejoindre. le départ peut être soit proactif (décider volontairement de quitter) soit réactif (le CH décide de l'exclure s'il ne répond plus ou qu'il est suspect). Un noeud peut aussi rejoindre un cluster qu'il a déjà quitté ou alors rejoindre un tout nouveau cluster. L'ajout de noeuds se fait quand un noeud rejoint le réseau et qu'il envoie une demande au CH voulu, qui envoie un message informant la SB de l'identité du nouveau noeud, si l'authentification est favorable le CH re-génère une clé de cluster et la diffuse pour tous, même le nouveau venu.La clé de cluster est modifiée à chaque changement occurrent dans le cluster pour assurer la sécurité au passé et au présent.

Révocation des clés : dés qu'un noeud rejoint ou quitte un cluster la SB procède à l'analyse de la validité de la liste de noeuds reçue par le CH concerné par le départ ou le ralliement et décide de révoquer les clés des noeuds entrant non valides, sortants ou encore celles des noeuds qui ont disparus pendant un temps moyen (pas de réponse aux communications).

Ce protocole semble être très résistant au clonage, il assure aussi une sécurité au présent et au passé avec la re-génération des clés de clusters dés qu'un noeud rejoint ou quitte le cluster. Étant sans certificat ce protocole réduit le coût de communication due à l'authentification et le cryptage à base de certificat.Il est tout de même extrêmement coûteux en terme d'énergie à cause des multiples calculs et messages effectués à chaque étape du protocole, notamment l'étape de mouvement des noeuds avec renouvellement des clés. Il est aussi coûteux en terme de stockage à cause des multiples clés qu'un noeud doit stocker notamment les clés de paire (maîtresses/cryptographiques) qui le relient à chacun de ses voisins.

1.7 Protocoles de gestion dynamique et centralisée de clés

Comme l'autre catégorie déjà citée, l'aspect dynamique assure un rafraîchissement à temps régulier des clés ou à la demande du réseau ce qui accroît la sécurité. L'aspect centralisé lui, leur offre une gestion plus simple car un seul noeud contrôleur existe mais accroît considérablement le coût de communication qui est le plus vorace en terme d'énergie, c'est pour cela que les protocoles de cette catégorie ne sont pas très nombreux et qu'ils ont été moins détaillés que les autres protocoles des autres catégories dans cet état de l'art.

1.7.1 Protocole de gestion de clé basé sur une tierce partie de confiance (PIKE)

Le protocole PIKE proposé par Chan et Perrig [5] est l'un des protocoles d'établissement de clés qui consiste à utiliser un noeud comme intermédiaire de confiance pour l'établissement de clés. A chaque noeud est associé un ID de la forme (x,y). Un noeud (x, y) partage une clé avec tout noued ayant le même x ou le même y, par exemple le noeud (9 ;4) partage une clé différente avec chaque noeud (9 ; y) et chaque noeud (x ; 4). L'une des exécutions de PIKE se fait avec la SB comme unique intermédiaire.

Dans le cadre du protocole Pike, de nouvelles clés seront établies entre les noeuds voisins. Ces clés peuvent être utilisées d'une manière identique aux clés de paires originales afin de faciliter la mise en place d'autres clés. Par exemple, supposons que le noeud $(A1; A2)$ a mis en place une clé de paire avec le voisin $(B1; B2)$. Maintenant $(A1; A2)$ veut établir une clé avec un autre voisin $(B1; B3)$. Les noeuds $(B1; B2)$ et $(B1; B3)$ doivent partager une clé, car ils se trouvent sur la même ligne et $(A1; A2)$ peut utiliser $(B1; B2)$ comme intermédiaire pour effectuer l'établissement de clés avec $(B1; B3)$ grâce à la clé partagée.

Le protocole PIKE étant basé sur une tierce partie de confiance la sécurité peut être bonne mais le coût de communication est très élevé et le coût de stockage aussi car un noeud doit

stocker les clés ayant le même x ou le même y que lui alors qu'il ne les utilisera pas forcément durant toute la durée de vie du réseau.

1.7.2 Protocole de sécurité pour WSN (SPINS)

Perrig, Szewczyk et al. [15], ont proposé la méthode SPINS basée sur la SB. Dans cette dernière la SB est considérée comme tierce partie de confiance avec laquelle chaque noeud partage une clé secrète. Pour que deux noeuds puissent communiquer entre eux ils contactent la SB qui leur transmet une clé symétrique en utilisant la clé secrète partagée avec chacun d'eux. Si un noeud est compromis la SB rompt le lien avec lui et contacte les noeuds relié à lui pour qu'il fassent de même.

comme le protocole PIKE, le protocole SPINS offre une résistance au attaque et une connectivité parfaite mais le coût de communication est très élevé.

1.8 Comparaison des protocoles de gestions de clés étudiés

Dans cette section, nous avons définis dés critères d'évaluation d'efficacité et de performances auxquels tout protocole de gestion de clés est soumis. Nous avons ensuite comparé suivant les critères précédemment cités, tous les protocoles de gestion de clés étudiés dans les sections précédentes de ce chapitre. Nous terminons la section et le chapitre par une synthèse résumant tout le chapitre.

1.8.1 Critères d'évaluation d'efficacité et de performances pris en compte

Dans ce qui suit, nous avons défini ces critères puis nous avons calculé leurs valeurs par rapport à un noeud simple quelconque du réseau. Voir le tableau 1.1 :

Les critères d'évaluation d'efficacité et de performances [12] pris en compte sont :

- **Coût de communication** est le nombre de messages (paquets) envoyés et reçus par un capteur.
- **Coût de stockage** est le nombre d'unités de mémoire utilisées pour stocker les clés nécessaires.
- **Coût de calcul** est le nombre de hachages et de chiffrements que fait un noeud dans toutes les étapes du protocole de gestion de clés pour le réseau auquel ce noeud appartient.
- **Connectivité** est la probabilité que deux noeuds voulant communiquer puissent établir un lien et une clé de lien.

1.8.2 Comparaison des protocoles étudiés, suivant les critères prédéfinis

Nous avons dans cette section une définitions de quelques notations, utilisées dans le tableau comparatif des protocoles de gestion de clés 1.1. Nous avons :

- C. com : coût de communications pour chaque capteur simple.
- C. stock : coût de stockage pour chaque capteur simple.
- C.cal : coût de calcul pour chaque capteur simple.
- Connec la connectivité qu'offre le protocole.
- d : le nombre de voisins du noeud pris en compte.
- m' : nombre de clé dans la chaîne de clés du noeud pris en compte.
- Pi : la probabilité que le noeud pris en compte puisse établir une clé de paire avec un autre noeud. Elle est différente d'un protocole à un autre.

- ID : l'identifiant du noeud pris en compte.
- n : le nombre de noeuds dans le réseau.
- Rn : nombre aléatoire.
- Rf : fonction polynomiale.
- Fh : fonction de hachage
- d' : nombre de voisins avec qui le noeud pris en compte veut communiquer.
- k : nombre de clés de base distribuées sur le noeud pris en compte.
- niv : nombre de niveaux pour CRKPH.
- l : nombre de niveaux pour EPKPP.
- hach : nombre de hachage.
- chiff : nombre de chiffrement.
- NbCH : nombre de CH à portée.

Dans le tableau :

Protocole	**C. com**	**C. stock**	**C. cal**	Connec
LEAP	(2*d)+1	(3*d)+2+m'	d+1	100%
KCBKPP	2*(d*Pi)	m'	NA	Pi
EPKPP	2*d	2*m'	(l*k)hach	Pi
Q-Composite	d+1	2*m'	NA	Pi
Schéma de base	d+1	m'*2	NA	Pi
PIKE	2*(Pi*d)	ID+Pi*d	2hach	Pi
SPINS	n*(3/2)	5+μTESLA	NA	100%
EEKM	2*d	3+d+Rn+ID+Rf	3chiff	100%
VLKM	NA	params+2	1hach	100%
EHKM	d+3	d+3	2chiff+4hach	100%
CL-EKM	$3 * d + 6$	$6 + 2 * d$	2hach+4chiff	100%
CRKPH	2*d+6	k+(k*niv)+1	(k*niv)hach+4chiff	Pi
EDDK	6*d+2	5+ taille table voisins	5*d+1	100%
Proposition	**5+NbCH**	**4**	**4chiff**	**100%**

TABLE 1.1 – Tableau comparatif des protocoles de gestion de clés étudiés

1.8.3 Synthèse des protocoles étudiés dans la taxonomie

La gestion de clés dans les WSN est jusqu'à aujourd'hui l'un des domaines les plus complexes car il doit tenir compte les contraintes des ressources des capteurs et de la sécurité des données transmises. Dans ce contexte, nous avons étudiés quelques protocoles de gestion de clés

existants dans la littérature, et nous avons tenté de les classifier suivant les travaux de He et al. [21], de Zhang et al. [12] et d'apports personnels tels que la mobilité des noeuds. A partir de notre classification et des protocoles étudiés nous avons déduit que la pré-distribution et le renouvellement de clés s'adaptaient le mieux aux contraintes de sécurité et de ressources des WSN. La cryptographie symétrique aussi, semble à première vue adaptée aux contraintes se ressources des capteurs mais elle est aussi faible en terme de sécurité. Alors que la cryptographie asymétrique offre des clés très robustes mais exige des ressources élevées. C'est pour cela que la cryptographie hybridée entre des clés asymétrique et d'autres symétrique (suivant le besoin) semble être le moyen le plus adapté pour générer des clés cryptographiques dans les WSN. En outre, la mobilité des capteurs connaît actuellement un engouement tel de nombreuses applications s'y sont développées, mais le domaine reste plutot novateur et donc ouvert aux recherches. Il serait donc plus intéressant de développer une proposition pour des capteurs mobiles.

1.9 Conclusion

La gestion des clés est l'un des domaine les plus critiques dans la sécurité des WSN, d'où l'existence des différents et nombreux travaux effectués afin d'avoir un schéma performant qui assure un niveau maximum de sécurité en optimisant les ressources utilisées. Ce schéma n'étant pas encore atteint nous avons entrepris d'apporter notre modeste contribution à ce domaine de la sécurité des WSN tout en tenant compte des déduction faites dans l'étude de la littérature dans ce chapitre. Cette contribution fera l'objet du prochain chapitre.

Chapitre 2

Protocole de Gestion de Clés Basé sur les paires de clés Publiques/Privées (PPBKM)

2.1 Introduction

L'une des plus grandes contraintes des WSN et d'établir des communications sécurisées. Pour ce, nous devons intégrer des clés cryptographiques et donc un système de gestion de clés efficace et robuste. Comme nous l'avons conclu dans le chapitre précédent, la pré-distribution et le renouvellement de clés ainsi que la cryptographie hybridée entre des clés asymétriques et symétriques s'imposent comme les outils les mieux adaptés aux exigences de sécurité et de ressources des WSN. Dans ce chapitre, nous proposons un schéma de gestion de clés basé sur une cryptographie hybridée nommé PPBKM (Public/Private-Based key Management), et utilisant la pré-distribution de clés avec renouvellement à la demande. Les performances de notre protocole seront comparées à celles des protocoles de gestion de clés à noeuds mobiles étudiés dans le chapitre précédent.

2.2 Modèle de réseau du PPBKM

Dans ce travail nous avons considéré un réseau de capteurs sans fil hétérogène supportant la mobilité tel que le montre la figure 2.1. Ce réseau est constitué d'une SB qui recueille des données provenant des capteurs et gère le réseau, les CHs sont certifiés par la SB ; ils sont considérés ayant des capacités de traitement élevées et sont définis avant le déploiement du réseau. Le réseau est aussi constitué d'un certain nombre de capteurs de capacités faibles. Les CHs sont considérés fixes, tandis que les capteurs peuvent être mobiles : ils peuvent quitter leur cluster, rejoindre un autre cluster et aussi retrouver dans un cluster précédemment quitté.

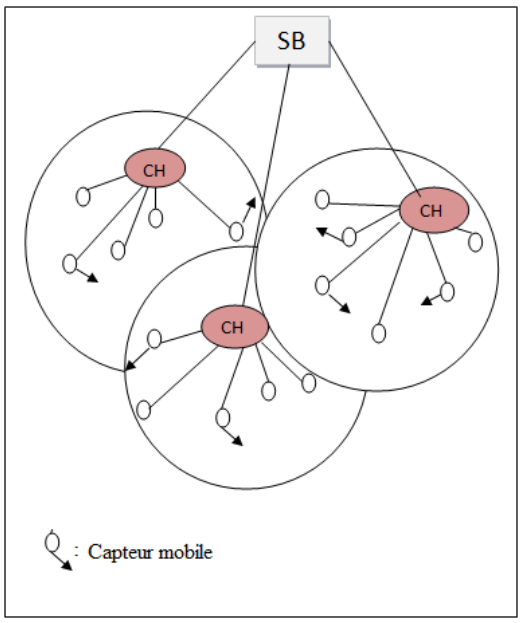

FIGURE 2.1 – Réseau hétérogène de capteurs sans fil et mobiles.

2.3 Hypothèses prises en compte par le PPBKM

Le PPBKM admet les hypothèses suivantes :

- Le déploiement est fait dans un environnement hostile de façon aléatoire.
- Les CHs sont considérés fixes (encres) et les capteurs peuvent être mobiles.
- Un centre de génération de clés (KGC) hébergé par la SB génère en partie les paires de clés publiques/privées ce qui permet d'éviter la certification.
- La SB est dotée d'un système de détection d'intrusions, de noeuds compromis et d'adversaires qui vérifie constamment la validité des noeuds du réseau.
- Les capteurs ne communiquent pas entre eux mais communiquent tous avec leurs CHs respectifs grâce aux clés de paires et aux clés de cluster.
- Chaque capteur se voit assigné un identifiant unique L_i et chaque CH se voit assigné un identifiant unique CH_i.
- Les CHs sont tous à portée de la SB.
- Les capteurs d'un cluster sont à portée à un saut de leur CH.
- Le PPBKM a été inspiré du CL-EKM (configuration du système).

2.4 Modèles d'attaque et exigences de sécurité pris en compte par PPBKM

Le PPBKM pallie aux attaques suivantes :

- **Résistance aux noeuds compromis** : le fait qu'un noeud soit compromis ne doit pas affecter la sécurité des informations stockées dans les autres noeuds ou révéler les clés des autres noeuds légitimes.
- **Résistance aux clonage** : un adversaire peut mener une attaque de clonage, en extrayant les clés d'un noeud qu'il vient de compromettre et en les plaçant dans des noeuds clones qu'il essaye d'injecter dans le réseau.
- **Sécurité au passé et au présent** : Le système doit assurer la transmission du secret pour empêcher un noeud d'utiliser une ancienne clé pour continuer à décrypter les nouveaux messages. Il doit aussi assurer le secret au passé pour empêcher un noeud avec la nouvelle clé de revenir en arrière dans le temps pour déchiffrer les messages précédemment échangés cryptés avec des clés antérieures.

2.5 Vue générale de PPBKM

Dans ce travail, nous avons proposé un protocole hybride (clés asymétriques et clés symétriques) de gestion de clés basé sur des paires de clés publiques/privées (PPBKM) pour les WSN à noeuds mobiles. Ce protocole procède à la gestion de deux types de clés : la paire de clés publique/privée et la clé de cluster. Dans la suite de ce chapitre, nous avons mentionné les types de clés utilisées, et les cinq étapes par les quelles passe la gestion de ces clés : configuration du réseau, formation de clusters, mise à jours des clés, mouvement des noeuds et leur révocation.

2.5.1 Types de clés

Le PPBKM procède à la gestion des clés suivantes :

- **Paire de clés publique/privée sans certificat** : avant qu'un noeud ne soit déployé, le KGC hébergé par la SB additionné au noeud lui même, procède à la génération d'une unique paire de clés publique/privée et l'installe dans le noeud (principe de ECC [9]). Cette paire de clés servira à crypter les communications entre un capteur et son CH ou alors entre un CH et la SB pour pouvoir transmettre les données.
- **Clé de cluster** : tous les noeuds d'un cluster partagent une clé de cluster. Cette clé sert à diffuser d'une manière sécurisée les messages d'avertissement dans le cluster quand aucun noeud est détecté compromis si ce n'est pas le cas les paires de clés sont utilisées

Symbole	**signification**
L_i	identifiant unique pour un capteur simple i
CH_j	identifiant unique pour un cluster-head j
$LastCH_{L_a}$	identité de l'ancien CH du noeud L_a
NCH_{L_a}	identité dunouveau CH du noeud L_a
q	nombre premier
E/F_q	courbe elliptique
F_q	champs de valeurs définis pour les variables de la courbe elliptique
P	point de la courbe elliptique
K	paramètre secret choisi par le KGC
Ω	paramètres du système
R_{Li}	clé publique partielle unique pour chaque noeud
D_{Li}	clé privée partielle unique pour chaque noeud
$P_{k_{L_i}}$	clé publique unique pour chaque noeud
$S_{k_{L_i}}$	clé privée unique pour chaque noeud L_i
G_{CH_j}	clé de cluster
$\Im$	liste de membres légitimes du réseau constituée par la SB
$\Im_j$	liste de membres du cluster j constituée par CH_j
$\Re$	liste de membres révoqués du réseau constituée par la SB

TABLE 2.1 – Notations liées aux protocole PPBKM

ou alors les messages de maintenance du réseau (messages balises). Le CH d'un cluster est l'unique noeud qui met jour la clé de cluster et ce quand un noeud rejoint ou quitte le cluster ou alors quand un noeud est détecté compromis.

2.5.2 Configuration du réseau

Avant le déploiement du réseau, la SB et le KGC génèrent les paramètres du système. Le KGC et les noeuds eux-mêmes génèrent les paires de clés et les installent dans chaque noeud puis la SB enregistre le noeud dans $\Im$. Les notations utilisées sont définies dans le tableau 2.1.

1. **Génération des paramètres du système** : le KGC choisit un paramètre de sécu-

rité $K \in Z^+$ et en sort les paramètres système suivant $\Omega = \{F_q, E/F_q, G_q, P, F_h\}$ et x (paramètre secret). La SB publie Ω pour tous les noeuds et garde x secret.

2. **Génération des paires de clés** : dans ce qui suit nous décrivons la génération d'une paire de clés pour un noeud L_i(même procédure pour les CH_i), Tout L_i choisit une valeur $x_{L_i} \in Z_q^*$ propre à lui et calcule $P_{L_i} = x_{L_i}.P$ qu'il met à disposition du KGC. Le KGC choisit $r_{Li} \in Z_q^*$ puis procède au calcule d'une paire de clés partielle (R_{L_i}, D_{L_i}), et l'enregistre dans le noeud L_i, tel que :
   $R_{L_i} = r_{L_i}.P$
   $D_{L_i} = r_{L_i} + x.F_h(L_i, R_{Li}, P_{L_i}) mod q$
   De son coté L_i calcule une paire de clés (P_{kLi}, S_{kL_i}) sur la base de la paire de clés partielle calculée par le KGC, tel que :
   $P_{k_{L_i}} = (P_{L_i}, R_{L_i})$
   $S_{k_{L_i}} = (D_{L_i}, x_{L_i})$
3. **Enregistrement des noeuds** : la SB génère une liste de membres $\Im$ où elle enregistre les noeuds. Cette liste contient les identifiants, les clés publiques et les CHs auxquels appartiennent tous les noeuds. Elle initialise une liste de révocation $\Re$ qui contiendra les noeuds révoqués en cours de vie du réseau. La SB installe dans chaque CH, un certificat lui conférant le privilège d'être CH et de former son cluster sur cette base.

Identifiant du noeud	**Clé publique du noeud**	**CH du noeud**
L_1	$P_{k_{L_1}}$	CH_4
L_2	$P_{k_{L_2}}$	CH_1
L_3	$P_{k_{L_3}}$	CH_3
.	.	.
.	.	.
.	.	.
.	.	.
L_n	$P_{k_{L_n}}$	CH_1

TABLE 2.2 – Liste de membres $\Im$ tenue par la SB

2.5.3 Formation des clusters

Une fois les noeuds déployés, chaque CH récupère les identifiants des capteurs simples qui l'entourent (voisins) grâce à des messages de balises qu'il diffuse, puis procède à leur authentification en passant par la SB. Si cette dernière s'avère positive, le CH partage une clé de cluster

commune avec tous les noeuds voisins. Pour simplifier les explications, dans ce qui suit, nous allons nous concentrer sur la découverte du noeud L_i par CH_j et l'échange de la clé de cluster entre eux.

1. **Découverte des noeuds** : CH_j diffuse un message m_j servant à découvrir ses voisins et à leur prouver qu'il est CH, tel que $m_j = \left\{P_{k_{CH_j}}, CH_j, (CH_j, privilge)S_{k_{SB}}\right\}$. Le noeud L_i recevant plusieurs messages, choisit celui dont le signal est le plus puissant comme CH, vérifie son rôle de cluster-head et son identité. Si elle s'avère positive il lui envoie un accusé de réception $r_i = \left\{L_i, P_{k_{L_i}}\right\}$. Suit à quoi CH_j constitue une liste $\Im_j$.

Algorithm 1 Découverte des noeuds

```
1: PROCEDURE Découverte_noeuds
2: for (k=1 ; k ≤ Nb_CH ; k++) do
3:    CH_j exécute diffuser(P_{k_{CH_j}}, CH_j, (CH_j, privilge)S_{k_SB}) ;
4:    for (i=1 ; i ≤Nb_L_{CH_j} ; i++) do
5:       if L_i reçoit de CH_j then
6:          L_i exécute CalculDist(L_i,CH_j) ;
7:          if la distance est minimale par rapport aux autres CHs then
8:             L_i exécute VerifLegitime(CH_j) ;
9:             L_i exécute Envoyer(ACK,CH_j) ;
10:            CH_j exécute Stocker(L_i,ℑ_j) ;
11:         end if
12:      end if
13:   end for
14: end for
15: END PROCEDURE
```

2. **Validation des voisins découverts** : après avoir découvert tous ses voisins CH_j envoie à la SB m_2 tel que $m_2 = \left\{CH_j, (CH_j, \Im_j)S_{k_{CH_j}}\right\}$. A la réception de m_2, la SB vérifie la légitimité du CH et la validité des noeuds de la liste $\Im_j$. Si le CH est légitime et que tous les noeuds sont valides, la SB répond par un acquittement au CH_j. Sinon elle forme une liste $\Re_j$ et la renvoie à CH_j. Après cette validation, le CH_j stocke $\Im_j$ nettoyée des noeuds invalidés par la SB.

Algorithm 2 Validation des noeuds voisins

PROCEDURE Validation_noeuds_voisins
for (j=1 ; j $\leq Nb\_CH_j$; j++) **do**
  CH_j exécute envoyer(CH_j, $(CH_j, \Im_j)S_{k_{CH_j}}$, SB) ;
  $i = 1$;
  while i$\leq Nb_capteurs_CH_j$ **and** $\Im_j$ valide) **do**
    SB exécute VerifValidité(L_i,$\Im_j$) ;
    $i++$;
  end while
  if $\Im_j$ non valide **then**
    SB exécute Former($\Re_j$) ;
    SB exécute Envoyer($\Re_j$,CH_j) ;
    CH_j exécute Stocker($\Im_j - \Re_j$) ;
  else
    SB exécute Envoyer(ACK,CH_j) ;
    CH_j exécute Stocker($\Im_j$) ;
  end if
end for
END PROCEDURE

3. **Génération de la clé de cluster** : après validation de $\Im_j$ le CH_j génère une clé de cluster G_{CH_j} puis constitue $m_3 = \left\{CH_j, (CH_j)_{Sk_{CH_j}}, (G_{CH_j})P_{k_{L_1}}, (G_{CH_j})P_{k_{L_2}}.....,(G_{CH_j})P_{k_{L_n}}\right\}$ avec $L_1, L_2....L_n$ les noeuds de $\Im_j$ à qui CH_j envoie m_3. Chaque noeuds de $\Im_j$ recevant ce message, vérifie d'abord l'identité du CH avec la première partie du message, puis déchiffre la seule partie qu'il peut déchiffrer : celle chiffrée avec sa clé publique. Suite à quoi, chaque noeud aura la clé de cluster qu'il stockera. Notons que si un noeud est détecté par la SB comme étant compromis, le CH_j n'aura qu'a chiffrer une clés erronée $((GErronnee_{CH_j})P_{k_{Lcompromis}})$pour dérouter le noeud compromis et ne pas devoir construire de liens ni de communications avec lui.

Algorithm 3 Génération de la clé de cluster

PROCEDURE Génération_clé_cluster
for (j=1 ; j ≤ Nb_CH ; j++) **do**
  CH_j exécute Generer(G_{CH_j}) ;
  CH_j exécute Diffuser($CH_J, (CH_j)_{Sk_{CH_j}}, (G_{CH_j})P_{k_{L_1}}, .., (G_{CH_j})P_{k_{L_{CH_j}}}$) ;
  for (i=1 ; j ≤$Nb\_L_{CH_j}$; i++) **do**
    if L_i reçoit de CH_j **then**
      L_i exécute VerifIdent(CH_j) ;
      if identité vérifiée **then**
        L_i exécute Déchiffrer($(G_{CH_j})P_{k_{L_i}}$) ;
        L_i exécute Stocker((G_{CH_j}) ;
      end if
    end if
  end for
end for
END PROCEDURE

2.5.4 Mouvement des noeuds

La phase de mouvement des noeuds peut avoir plusieurs cas tel qu'un noeud peut quitter un cluster : volontairement (Proactif) ou non (réactif), ou alors rejoindre un cluster qu'il a déjà quitté auparavant avant ou auquel il n'a jamais eu à faire.

2.5.4.1 Quitter un cluster

Un capteur peur quitter son cluster en raison de défaillance, changement d'emplacement physique ou encore l'échec de la communication avec le CH. On distingue deux cas quand un capteur quitte un cluster : cas proactif, et cas réactif.

- **cas proactif :** Quand un noeud L_p décide de quitter un cluster activement, par exemple en remarquant l'affaiblissement de la force du signal, du à l'éloignement progressif de son CH). Il prend la date de son départ et son identifiant, il les chiffre avec sa clé privée $(Date, L_p)_{Sk_{L_p}}$ puis envoie au CH le message $(Date, L_p, (Date, L_p)_{Sk_{L_p}})$. Quand le CH reçoit le message, il sauvegarde la date et l'identifiant du noeud qui lui serviront à s'identifier s'il revient vers le CH. Le CH avertit ensuite la SB pour qu'elle change le status du noeud partant dans la table $\Im$. La SB répond par un ACK puis le CH supprime L_p de $\Im_j$

Algorithm 4 Quitter un cluster volontairement

PROCEDURE Quitter_proactif
p=1 ;
while L_p veut quitter son cluster **do**
  L_i exécute envoyer($(Date, L_p, (Date, L_p)_{Sk_{L_p}}), CH_{L_p}$) ;
  CH_{L_p} exécute Stocker($L_p, Date$) ;
  CH_{L_p} exécute AvertirSB(L_p, SB) ;
  SB exécute Supprimer(L_p,$\Im$) && Envoyer(ACK,CH_{Lp}) ;
  CH_{L_p} exécute Supprimer($L_p, \Im_j$)
  CH_{L_p} exécute MiseAJour($G_{CH_{L_p}}$) ;
  CH_{L_p} exécute Envoyer($G'_{CH_{L_p}}, L_p$) ;
  p++ ;
end while
END PROCEDURE

- **Quitter un cluster (cas réactif) :** Quand un noeud L_r disparaît d'un cluster sans prévenir pour cause de déchargement de batterie ou tout autre problème technique, il sera détecté par son CH grâce aux messages qu'un CH et ses noeuds échangent constamment, pour toute la durée de vie du réseau. Le CH dans ce cas là, avertit la SB qui procède à la phase de révocation. Ensuite, il met à jour la clé de cluster $G_{CH_{L_r}}$.

Algorithm 5 Quitter un cluster involontairement

PROCEDURE Quitter_réactif
j=1 ;
while CH_j détecte un noued disparu L_r **do**
  CH_{L_r} exécute AvertirSB(L_r) ;
  SB exécute Révocation(L_r) && Envoyer(ACK,CH_{Lr}) ;
  CH_{L_r} exécute MiseAJour($G_{CH_{L_r}}$) ;
  CH_{L_r} exécute Envoyer($G'_{CH_{L_r}}, L_r$) ;
  j++ ;
end while
END PROCEDURE

2.5.4.2 Rejoindre un cluster

quand un noeud se déplace, il peut se retrouver dans deux cas : rejoindre un cluster qu'il a déjà quitté, ou encore rejoindre un nouveau cluster.

- **Rejoindre un cluster déjà quitté :** dans le cas où un noeud L_a revient dans son ancien cluster, il envoi au CH le message $(Date, L_p)_{Sk_{L_p}}$. A la réception de ce message, le CH le déchiffre en utilisant la clé publique du noeud, et vérifie si les données contenues dans le message reçu sont correspondantes a celles qu'il a sauvegardé quand le noeud

a quitté son cluster. Si c'est le cas alors l'identification est faite, suite à quoi, le CH avertit ensuite la SB qui tente à authentifier et valider le noeud (système de détection d'intrusions et de neouds compromis). Si la validation est effectuée, renvoie un ACK au CH concerné et le noeud rejoint, sinon la demande d'ajout est ignorée.

Algorithm 6 Rejoindre un ancien cluster

1: **PROCEDURE** Rejoindre_ancien_cluster
2: a=1 ;
3: **while** L_a revient dans un ancien cluster **do**
4: L_a exécute envoyer($(Date, L_a)_{Sk_{L_a}}$),$LastCH_{L_a}$) ;
5: $LastCH_{L_a}$ exécute AvertirSB(L_a) ;
6: SB exécute VerifValid(L_a) ;
7: **if** (L_a) valide **then**
8: SB exécute envoyer(ACK,$LastCH_{L_a}$) ;
9: L_a rejoint $LastCH_{L_a}$;
10: $LastCH_{L_a}$ exécute MiseAJour($G_{LastCH_{L_a}}$) ;
11: $LastCH_{L_a}$ exécute Envoyer($G'_{LastCH_{L_a}}, L_a$) ;
12: **else**
13: SB exécute Révocation(L_a) ;
14: $LastCH_{L_a}$ ignore la demande ;
15: **end if**
16: a++ ;
17: **end while**
18: **END PROCEDURE**

– **Rejoindre un nouveau cluster :** Dans le cas où un noeud L_{nv} va dans un nouveau cluster, il envoie au CH le message $(L_{nv}, P_{k_{L_a}})$. A la réception de ce message, le CH avertit la SB qui tente d'authentifier et valider le noeud. Si la validation est faite, elle renvoie un ACK au CH concerné et le noeud rejoint le cluster, sinon la demande d'ajout est ignorée. Le CH procède ensuite à la mise à jour de la clé de cluster.

Algorithm 7 Rejoindre un nouveau cluster

PROCEDURE Rejoindre_nouveau_cluster
nv=1 ;
while L_{nv} va dans un nouveau cluster **do**
  L_{nv} exécute envoyer($(L_{nv}, P_{k_{L_{nv}}})$,$NCH_{L_{nv}}$) ;
  $NCH_{L_{nv}}$ exécute AvertirSB(L_{nv}) ;
  SB exécute VerifValid(L_{nv}) ;
  if (L_{nv}) valide **then**
    SB exécute envoyer(ACK,$NCH_{L_{nv}}$) ;
    L_{nv} rejoint $NCH_{L_{nv}}$;
    $NCH_{L_{nv}}$ exécute MiseAJour($G_{NCH_{L_{nv}}}$) ;
    $NCH_{L_{nv}}$ exécute Envoyer($G'_{NCH_{L_{nv}}}, L_{nv}$) ;
  else
    SB exécute Révocation(L_{nv}) ;
    $NCH_{L_{nv}}$ ignore la demande ;
  end if
  nv++ ;
end while
END PROCEDURE

2.5.5 Mise à jour des clés

Un noeud compromis peut être détecté et supprimé (système de détection d'intrusions) et sa compromission n'affecte que sa propre paire de clé publique/privée et la clé de cluster. Si la clé de paire est compromise n'impactera aucun autre noeud alors que si la clé de cluster est compromise elle affectera tout les noeuds du cluster. C'est pour quoi dans ce qui suit , nous allons mettre en exergue le renouvellement de la clé de cluster G_{CH_j}, pour un cluster spécifique dont le CH est CH_j ; mais la procédure est similaire pour tous les CH. Pour garantir la sécurité au présent et au passé, le renouvellement de la clé de cluster se fait dans deux cas :

- Quand un noeud est compromis.
- A chaque changement occurrent dans le cluster (capteur rejoint ou quitte le cluster)

La procédure de renouvellement est la même dans les deux cas. Le CH_j étant le seul à pouvoir mettre a jours cette clé, il re-génère une nouvelle clé G'_{CH_j} et la diffuse suivant le même principe que le message m_3 précédemment cité.

Algorithm 8 Mise à jour de clés

1: **for** (j=1 ; j $\leq$ Nb_CH ; j++) **do**
2: **if** noeud compromis **or** changement dans le cluster **then**
3: CH_j génère une nouvelle clé
4: retourner G'_{CH_j}
5: **end if**
6: **end for**

2.5.6 Révocation des noeuds

Par hypothèse, la SB est capable de détecter les noeuds compromis. Pour ce, elle est supposée dotée d'un système de détection d'intrusions, de noeuds malicieux ou d'adversaires [24],[16]. De ce fait, nous ne prenons pas en compte comment la SB effectue la détection de noeuds compromis mais nous supposons que la mise à jour de l'état du cluster transmise par chaque CH, à chaque changement occurrent dans le cluster, est l'une des techniques utilisées. Pour simplifier l'explication du processus de révocation nous considérons la détection d'un noeud compromis L_c appartenant à un cluster j dont le CH est CH_j. Si un noeud simple L_c est compromis, la SB cherche dans la table $\Im$ le CH du noued L_c et l'avertit en envoyant $m_4 = (noeud\_comp, L_c)P_{k_{CH_j}}$. Le CH_j supprime la clé publique du noeud L_c stockée et lance la procédure de renouvellement de la clé de cluster. Si à l'inverse, la SB détecte un CH compromis CH_c, elle avertit les noeuds du cluster du CH_c en envoyant le message $m_5 = ((Head\_comp, CH_j)P_{k_{L_1}}, (Head\_comp, CH_j)P_{k_{L_2}}, ..., (Head\_comp, CH_j)P_{k_{L_n}}$. Les L_i concernés par le message m_5 suppriment la clé publique du CH_c et la clé de cluster stockées et cherchent un nouveau CH.

Algorithm 9 Révocation des noeuds

1: La SB vérifie les noeuds ;
2: **if** L_c détecté **then**
3: SB exécute Envoyez($(noeud\_comp, L_c)P_{k_{CH_j}}, CH_j$) ;
4: CH_j exécute Supprimer($P_{k_{L_c}}$) ;
5: CH_j exécute MisAJour(G_{CH_j}) ;
6: **end if**
7: **if** CH_c détecté **then**
8: **for** (i=1 ; i $\leq Nb\_L_{i_{CH_c}}$; i++) **do**
9: SB exécute envoyez($(Head\_comp, CH_j)P_{k_{L_i}}, L_i$) ;
10: L_i exécute Supprimer($P_{k_{CH_c}}$) ;
11: L_i exécute RechercherCH() ;
12: **end for**
13: **end if**

2.6 Analyse de sécurité

Dans la suite de cette section, nous analysons la sécurité de notre solution.

2.6.1 Résistance aux noeuds compromis

Nous supposons qu'un adversaire qui capture le noeud L_i appartenant au cluster CH_j peut extraire toutes les clés stockées dans le noeud : paire de clés, clé publique du CH_j et la clé de cluster G_{CH_j}. Cependant étant donné que les paires de clés sont indépendantes les unes des autres et uniques, la capture d'un noeud ne dévoilera que la paire de clés du noeud capturé et à aucun moment n'aura pas d'impact sur les autres paires de clés des autres noeuds. Donc les communications resteront sécurisées dans tout le reste du réseau et à la prochaine action de détection d'intrusions le noeud capturé sera révoqué. Quant à la clé de cluster qui sera aussi dévoilée par la capture d'un noeud, elle est unique et différente d'un cluster à un autre et donc au pire des cas, un seul cluster sera compromis, mais à la prochaine action de détection d'intrusion cette clé sera renouvelée.

2.6.2 Résistance au clonage

Nous supposons qu'un adversaire qui compromet un noeud pourra avoir toutes les clés stockées dans ce dernier. Ces clés sont ensuite enregistrées dans plusieurs autres capteurs clones, que l'adversaire essayera d'injecter dans le réseau. Cependant étant donné que dans notre protocole, les CHs font valider leurs listes de noeuds par la SB avant d'établir des liens avec eux et que la SB est capable détecter les noeuds clones. Elle procédera à leurs révocations puis avertira les CHs concernés et ces noeuds ne seront plus acceptés par aucun autre CH.

2.6.3 Sécurité au passé et au présent

Dans PPBKM, les messages échangés entre les CHs et les capteurs simples sont cryptés avec les paires de clés ou alors avec les clés de clusters. Pour assurer cette sécurité au passé, PPBKM assure la mise à jour et la révocation de clés et ce, à chaque fois qu'un noeud rejoint un cluster ou alors qu'il est détecté compromis. Pour assurer la sécurité au présent, PPBKM assure aussi la fonction de mise à jour à chaque nouveau noeud arrivant dans un cluster. En effet quand un noeud est révoqué, toutes ses clés sont invalidées et la clé de cluster enregistrée dans le noeud est mise à jour. Le CH envoie la nouvelle clé de cluster à tous les noeuds valides en la cryptant avec la clé publique de chaque noeud ne donnant ainsi aucune possibilité au

noeud compromis de récupérer la nouvelle clé de cluster étant donné qu'il ne connaît aucune clé privée d'aucun autre capteur que lui même.

2.7 Conclusion

Dans ce chapitre nous avons proposé un protocole de gestion de clés pour les réseaux de capteurs sans fil nommé PPBKM puis nous avons détaillé son fonctionnement dans cinq phases principales. Nous avons développé des pseudo-algorithmes pour faciliter le suivi des procédures de chaque phase. Nous avons conclu par une validation du modèle d'attaques et des exigences de sécurité pris en compte par PPBKM. Les conclusions établissent que notre protocole répond à toutes les exigences de sécurité définies. A première vue, il paraît aussi être économique en terme de stockage et d'énergie comparativement aux protocoles appartenant à la même sous-classe (CL-EKM, CRKPH). Cependant, nous procéderons dans le chapitre suivant à une simulation de ces trois protocoles, afin de les évaluer en terme de stockage, d'énergie due aux communications et d'énergie totale (due aux communications et aux calculs).

Chapitre 3

Simulation et évaluation de performances

3.1 Introduction

Nous avons eu recours au langage Matlab afin d'évaluer l'efficacité et les performances de notre protocole de gestion de clés pour WSN (PPBKM). Pour ce, nous avons considéré le passage à l'échelle, le coût de stockage, l'énergie due au coût de communication et pour finir l'énergie totale. Autant de métriques importantes à prendre en compte pour proposer une solution adaptée aux WSN. Notre simulation se fait dans un environnement spécifique et sur l'étape admettant le plus de communications pour les capteurs simples : l'étape de mouvement des noeuds avec clustering et renouvellement des clés. L'énergie dans notre simulation a été calculée sur la base de la loi d'Einzelmann [11]. Nous avons comparé les résultat de notre simulation avec deux protocoles que nous avons aussi entrepris de simuler : CL-EKM [18] et CRKPH [13]. Dans ce chapitre nous avons commencé par une introduction puis nous avons présenté brièvement le simulateur MATLAB. Nous avons ensuite définit l'environnement de simulation avec tous les paramètres de simulation utilisés. Nous avons fait un rappel des critères et des métriques que nous avons pris en compte lors de la simulation. Nous avons aussi rappelé le principe général des protocoles qui nous ont servi de base de comparaison. Nous avons ensuite présenté les résultats obtenus et nous les avons analysés pour finir par un conclusion résumant tout le chapitre.

3.2 Présentation du simulateur MATLAB

MATLAB (MATrix LABoratory) est un logiciel pour effectuer des calculs numériques. Il a été conçu initialement pour faciliter le traitement des matrices mais il est maintenant utilisé dans tous les domaines des sciences qui nécessite de faire des calculs [3]. De manière générale,

MATLAB est utilisé pour faire des expériences de calcul très rapidement. C'est pourquoi il nous apparaît comme un outil adéquat pour notre simulation étant donné les résultats d'évaluation d'efficacité recherchées.

3.3 Environnement de simulation

La simulation est faite sur un réseau de capteurs de 200 capteurs simples et 50 Cluster-Heads, dispersés aléatoirement sur une surface carrée de $1000 * 1000m^2$. Nous supposons que :

- le nombre de CHs (50) représente 25% du nombre de capteurs simple (200).
- Tous les capteurs simples sont aptes à changer de position durant toute La période de simulation.
- Toutes les 2 unités de temps un mouvement se fait.
- Le rayon de communication est de 100m et est le même pour tous les capteurs.
- La valeur de l'énergie de départ est la même pour tous les capteurs.
- les calculs effectués ne se font que sur les capteurs simples considérés dotés de ressources limitées et non sur les CHs considérés dotés de ressources suffisantes

Les paramètres de notre simulation sont résumés dans le tableau 3.1.

Paramètres	**valeurs**
Surface	1000*1000 m^2
Nombre capteurs simples	200
Nombre de Cluster-Heads (CH)	50
Rayons de chaque noeud	150
temps de simulation	60s
Taille moyenne d'une clé cryptographique	40 bits
Taille moyenne d'un message transmis	2000 bits
Énergie consommée pour un cryptage symétrique [7]	9 μJ
Énergie consommée pour un cryptage Asymétrique [7]	19 mJ
Énergie consommée pour une fonction de hachage[7]	$4 * 10^{-4}$ J
Eelec, Efs [11]	50, 10

TABLE 3.1 – Paramètres de simulation.

La figure 3.1 représente le déploiement de notre réseau. les cercles rouges constituent les CHs et les astérisques bleus les capteurs simples.

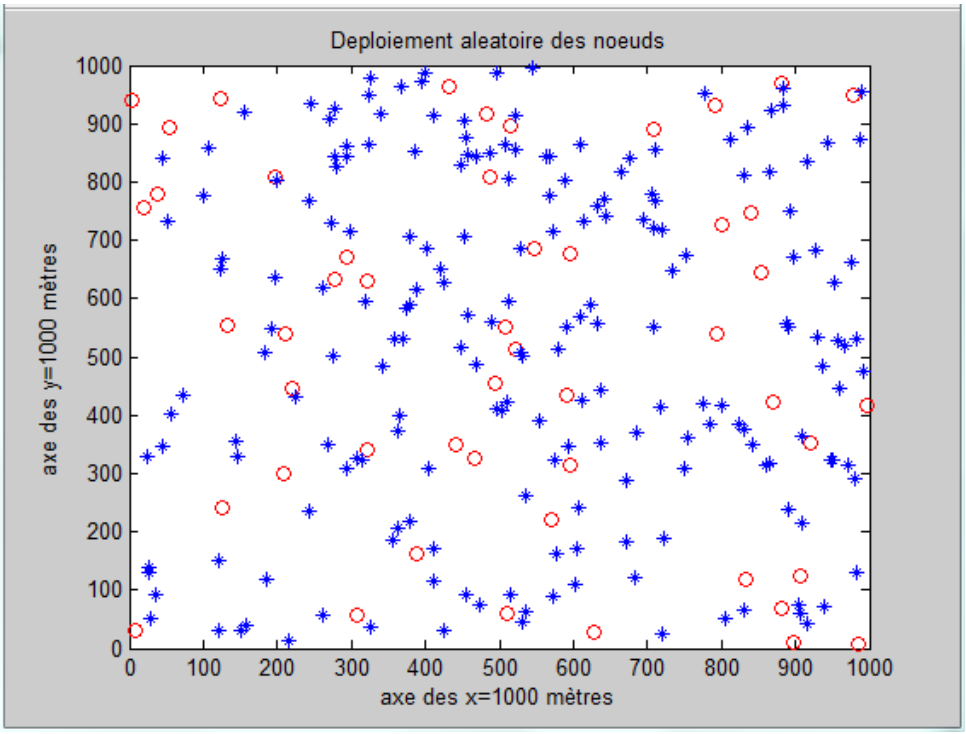

FIGURE 3.1 – Déploiement aléatoire des noeuds du réseau.

Quand à la figure 3.2, elle représente le partitionnement du réseau en clusters, avec le rayon de communication de chaque CH représenté avec un cercle vert.

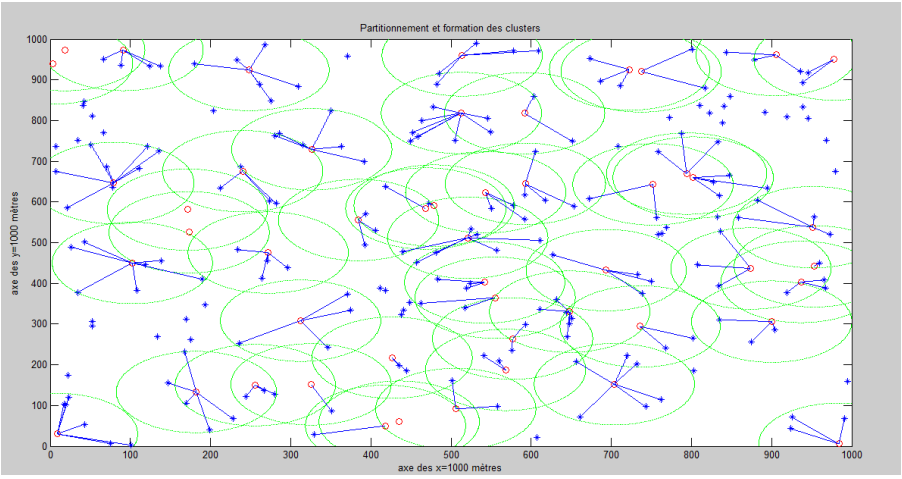

FIGURE 3.2 – Partitionnement du réseau en clusters.

3.4 Rappel des critères d'évaluation et des protocoles de comparaison

Dans ce qui suit nous faisons un bref rappel des critères d'évaluation d'efficacité (partiellement définis dans le chapitre1). Nous définissons les métriques de simulation utilisées. Nous faisons aussi un rappel des deux protocoles (CL-EKM [18] et CRKPH [13]) avec lesquels a été comparé notre protocole PPBKM.

3.4.1 Critères d'évaluation d'efficacité

Définition 3.4.1. *Coût de stockage est le nombre d'unités de mémoire utilisées pour stocker les identifiants et les clés nécessaires à tous les noeuds du réseau à un temps donné.*

Définition 3.4.2. *L'énergie due au coût de communication est la somme des énergies consommées par chaque noeud pour envoyer et recevoir un certain nombre de messages (paquet), elle est donc calculée pour tous le réseau à un temps donné.*

Définition 3.4.3. *L'énergie due au coût de calcul est la somme des énergies consommées par tous les noeuds du réseaux pour effectuer des calculs particuliers (cryptage, hachage, etc) à un temps donné.*

Définition 3.4.4. *L'énergie totale est la somme des énergies consommées par tous les noeuds du réseaux lors des calculs et des communications*

3.4.2 Métriques de simulation

Définition 3.4.5. *Le passage a l'échelle représente le fait de faire varier le nombre de noeuds du réseau et d'évaluer les critères précédemment cités et cela au premier déploiement du réseau.*

Définition 3.4.6. *Le partitionnement représente le fait de changer les rayons de communication des noeuds en évaluant les critères précédemment cités. Ceci nous permet ainsi de tester entre autre la disponibilité qu'offrent les protocoles.*

Définition 3.4.7. *La fréquence de mobilité représente le fait de d'évaluer les critères précédemment cités à chaque fréquence T tout en sachant que chaque fréquence est caractérisée par un mouvement des noeuds.*

3.4.3 Rappel sur les protocoles pris en compte pour la simulation

Nous avons pour notre simulation pris en compte 2 protocoles ayant les mêmes spécificités et portant sur le même modèle de réseau que notre proposition. Ces deux protocoles sont le CL-EKM [18] et le CRKPH [13].

- Le protocole CL-EKM est un protocole de gestion de clés sans certificats. Il est conçu pour sécuriser la communication dans les réseaux de capteurs sans fil mobiles [18]. Composé d'une SB fiable, de CHs et de capteurs simples, le CL-EKM opère sur un réseau de capteurs à clustering composé de capteurs simples de CHs et d'une SB. Il procède à la gestion de 5 types de clés : les paires de clés publique/privée pour l'établissement de clés de paires sécurisées, les clés individuelles pour les communications avec la SB, les clés maîtresses de paires et les clés cryptographiques de paires pour la sécurisation des communications et les clés de clusters pour la diffusion de l'information de changement d'état d'un noeud (il devient CH, noeud sortant, etc.). Le CL-EKM passe par 5 étapes : la configuration du système ou tous les paramètres ECC sont définis et où les paires de clés sont générées et stockées dans les noeuds. La formation de clusters où les CH forment leurs clusters en faisant authentifier les noeuds par la SB. La génération des clés de paires où des liens sécurisés sont établis entre des capteurs voisins. La révocation des clés où les clés des noeuds détectés invalides par la SB sont révoquées. Pour finir, le mouvement des noeuds qui est l'étape la plus coûteuse en énergie et qui fera l'objet de notre simulation, où les noeuds peuvent quitter ou rejoindre un cluster en se déplaçant physiquement d'un cluster à un autre. Le CL-EKM est très résistant au clonage, il assure aussi une sécurité au présent et au passé avec la re-génération des clés de clusters dés qu'un noeud rejoint ou quitte le cluster.Il est tout de même extrêmement coûteux en terme d'énergie à cause des multiples calculs et messages effectués à chaque étape du protocole, notamment l'étape de mouvement des noeuds avec renouvellement des clés. Il est aussi coûteux en terme de stockage à cause des multiples clés qu'un noeud doit stocker notamment les clés de paires (maîtresses/cryptographiques) qui le relient à chacun de ses voisins.
- Le CRKPH est un protocole de gestion de clés qui opère sur un réseau à clustering composé d'une SB fiable, un certain nombre de CHs et des capteurs simples. Il prend en charge la mobilité des noeuds.il procède d'abord à l'établissement de liens sécurisés entre les noeuds et leurs CHs, puis à l'aide du CH, les noeuds voisins, établissent des liens sécurisés entre eux et ce en se basant sur le principe d'un pool de clés de base dont des groupes différents de K clés sont aléatoirement choisis et chargé dans les noeuds avant déploiement. Après déploiement, un certain nombre de niveaux est défini (un niveau représente le périmètre autour d'un noeud tel que chaque niveau est de 40m). Pour chaque niveau un capteur calcule k clés dérivées correspondantes au K clés de base qu'il

a stocké. La clé de paire utilisée par le noeud A avec le noeud B sera, si elle existe une clé commune entre le sous-pool de clés dérivées contenus dans le noeuds A et celui généré suivant le niveau du noeud B par rapport à A et qui est contenu dans le noeud B.

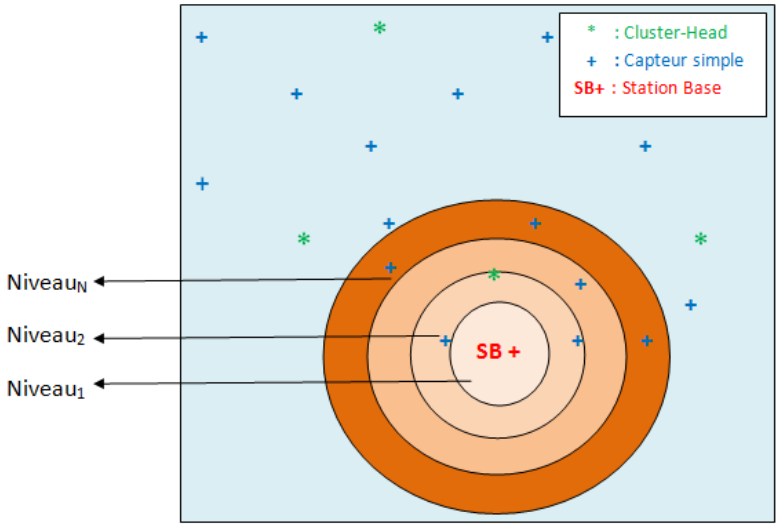

FIGURE 3.3 – Niveaux de dérivation du CRKPH.

3.5 Résultats obtenus lors de la simulation

Dans cette section nous avons considéré chaque paramètre de simulation pris en compte avec tous les critères d'évaluation pris en compte. Nous avons mis en exergue les figures résultantes de notre simulation puis nous avons analysé les résultats obtenus.

3.5.1 Résultats suivant le passage a l'échelle

Le passage à l'échelle comme défini précédemment dans ce chapitre, représente le fait de faire varier le nombre de noeud simples du réseau. Pour ce paramètre nous avons fixé le nombre de CHs à 50 le rayon de communication à 100m et la surface à $1000 * 1000m^2$.

3.5.1.1 Coût de stockage

La figure 3.4 représente le coût de stockage (bits) du PPBKM et des deux autres protocoles en fonction du nombre de noeuds dans le réseau.

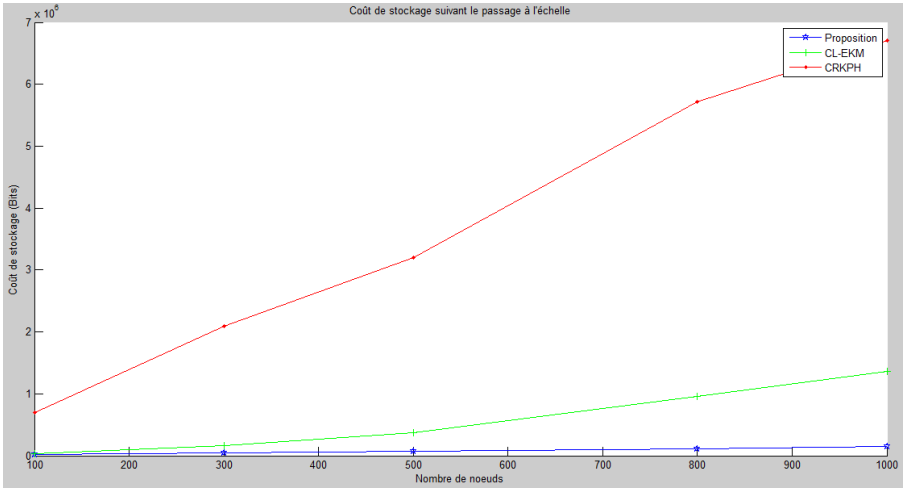

FIGURE 3.4 – Graphe du coût de stockage en fonction du nombre de capteurs.

Comme le montre la figure 3.4, pour les 2 protocoles CL-EKM et notamment pour le CRKPH plus le nombre de noeud du réseau augmente plus le coût de stockage du réseau augmente et ce de façon conséquente comparativement à notre protocole qui montre une croissance négligeable. La raison de ce résultat est le fait que notre protocole exploite les paire de clés publiques/privées pour le cryptage des communications alors que les autres font appel à un grand nombre de clés supplémentaires notamment les clés de paires pour le CL-EKM et les clés de bases et dérivée pour le CRKPH qu'il stock depuis le 1er déploiement. Notre protocole reste donc adapté même avec le passage à l'échelle.

3.5.1.2 Énergie consommée due aux communications

La figure 3.5 représente l'énergie consommée due au communications, du PPBKM et des deux autres protocoles en fonction du nombre de noeuds dans le réseau.

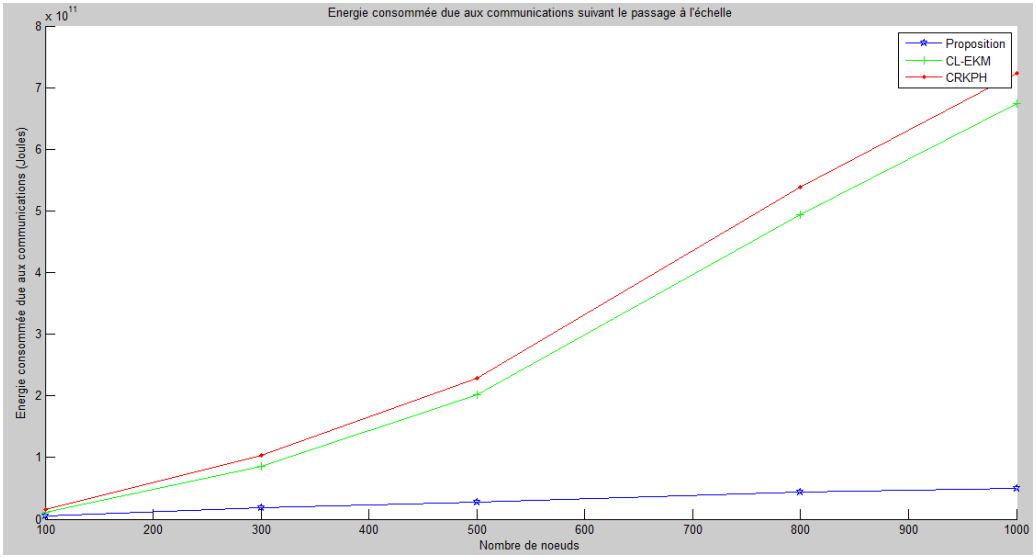

FIGURE 3.5 – Graphe de l'énergie consommée due aux communications en fonction du nombre de capteurs.

Comme le montre la figure 3.5, pour les 2 protocoles CL-EKM et CRKPH plus le nombre de noeuds du réseau augmente plus l'énergie consommée pour les communications du réseau augmente et ce de façon conséquente comparativement à notre protocole qui connaît une croissance légère. La raison de ce résultat est le fait que notre protocole pour l'étape simulée (mouvement des noeuds, clustering et renouvellement de clés), ne nécessite que très peu de communications pour permettre à un noeud de quitter ou de rejoindre un cluster, alors que le CL-EKM et encore plus le CRKPH effectue énormément de communications pour cette étape.

3.5.1.3 Énergie totale due aux calculs et aux communications

La figure 3.6 représente l'énergie consommée due au communications et aux calculs, du PPBKM et des deux autres protocoles en fonction du nombre de noeuds dans le réseau.

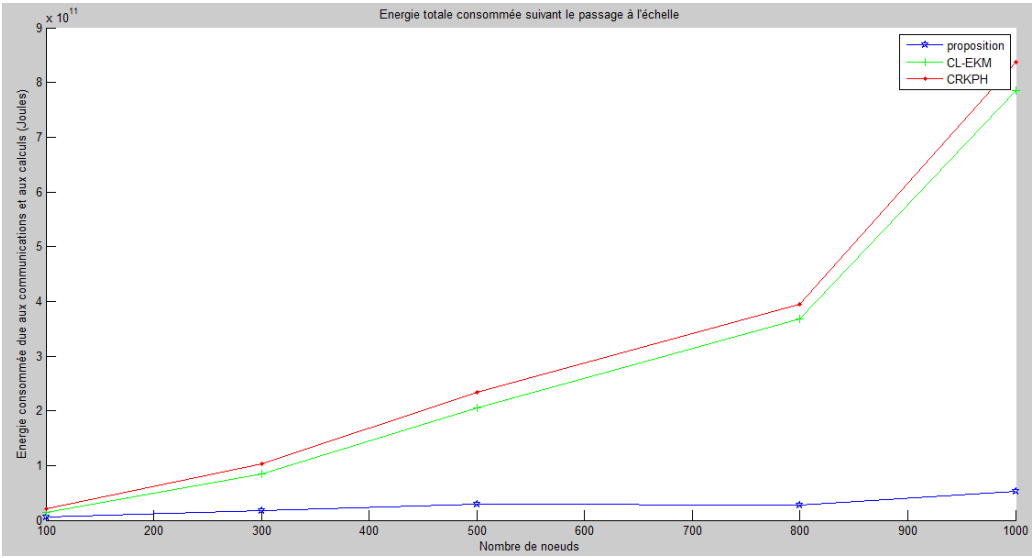

FIGURE 3.6 – Graphe de l'énergie consommée due aux calculs et aux communications en fonction du nombre de capteurs.

Notre simulation faite sur le coût de calcul en fonction du nombre de noeuds du réseau a montré que notre protocole était moins économique que les 2 autres, à cause du fait que la cryptographie asymétrique soit plus coûteuse que la cryptographie symétrique, mais nous constatons sur la figure 3.6 que le CL-EKM et le CRKPH malgré leur coût de calcul moins élevé, sont beaucoup plus coûteux en terme d'énergie totale et ce à cause de l'a différence considérable en terme d'énergie due aux communications entre notre protocole et les 2 autres. Une différence qui accroît avec l'augmentation du nombre de noeuds car chacun des noeuds rajoutés fera ses propres communications pour rejoindre un cluster et établir des liens avec lui.

3.5.2 Résultats suivant le partitionnement

Le partitionnement comme nous l'avons défini précédemment, est le fait de changer le rayon de communication des noeuds pour voir l'impacte sur les ressources des noeuds du réseau.

3.5.2.1 Coût de stockage

La figure 3.7 représente le coût de stockage (bits), du PPBKM et des deux autres protocoles en fonction du rayon de communication des noeuds du réseau.

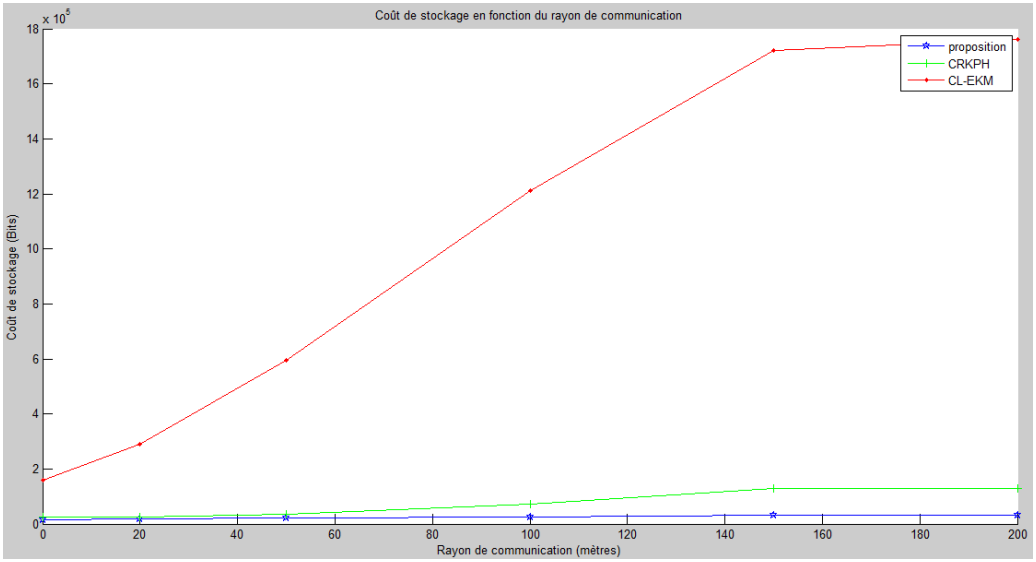

FIGURE 3.7 – Graphe du coût de stockage en fonction du rayon de communication.

Comme le montre la figure 3.7, pour le protocole CL-EKM plus le rayon augmente moins il y a des noeuds isolés et chaque noeud est susceptible d'avoir plusieurs voisins et donc plus de clés à stocker. Pour le CRKPH plus le rayon augmente moins il y a de noeuds isolés ou chacun stocke 20 clés de base et 20*niv clés dérivées qui sommées donne une croissance élevée. Alors que pour notre protocole l'augmentation du rayon réduit le nombre de noeuds isolés mais chaque noeud comporte un nombre réduit de clés (4clés) donc l'augmentation du cout de stockage est faible comparativement aux deux autres.

3.5.2.2 Énergie due aux communications

La figure 3.8 représente l'énergie consommée due au communications, du PPBKM et des deux autres protocoles en fonction du rayon de communication des noeuds du réseau.

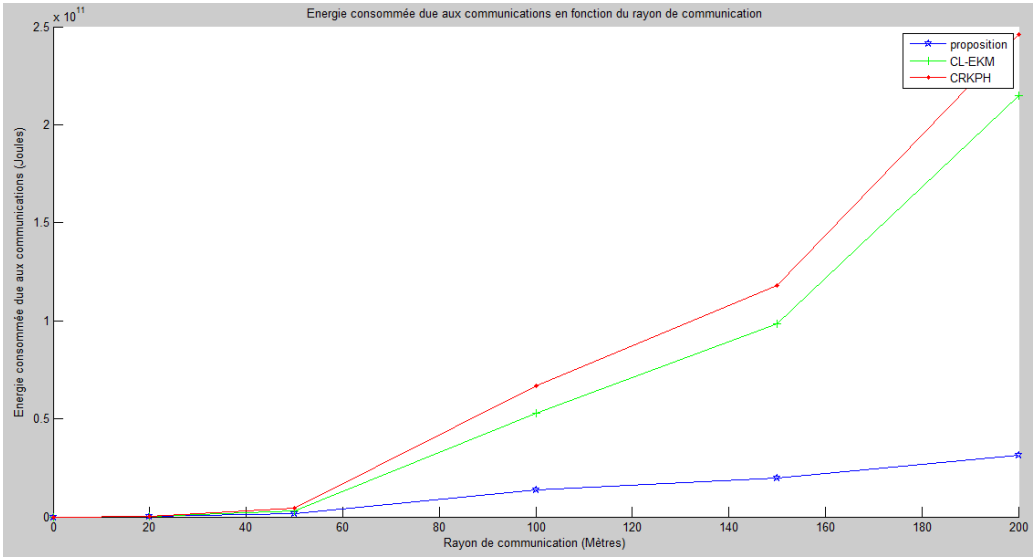

FIGURE 3.8 – Graphe de l'énergie consommée due aux communications en fonction rayon de communication.

Comme le montre la figure 3.8, jusqu'à un rayon de communication de 50m les 3 protocoles sont aussi faibles les uns que les autres en terme d'énergie due aux communications. En effet ce rayon de 50m relativement à la surface de $1000 * 1000m^2$ donnera une quantité considérable de noeuds isolés n'ayant donc pas la capacité de communiquer avec un CH réduisant ainsi l'énergie due au communications du réseau. Cependant dès que le rayon dépasse 50m l'énergie de CL-EKM et de CRKPH augmentent vite à cause de l'énergie que dépense tous les noeuds pour établir des clés de paires et des clés de clusters alors que celle de notre protocole observe une croissance lente liée uniquement à l'établissement de la clé de cluster.

3.5.2.3 Énergie totale due aux calculs et aux communications

La figure 3.9 représente l'énergie consommée due au communications et aux calculs, du PPBKM et des deux autres protocoles en fonction du rayon de communication des noeuds du réseau.

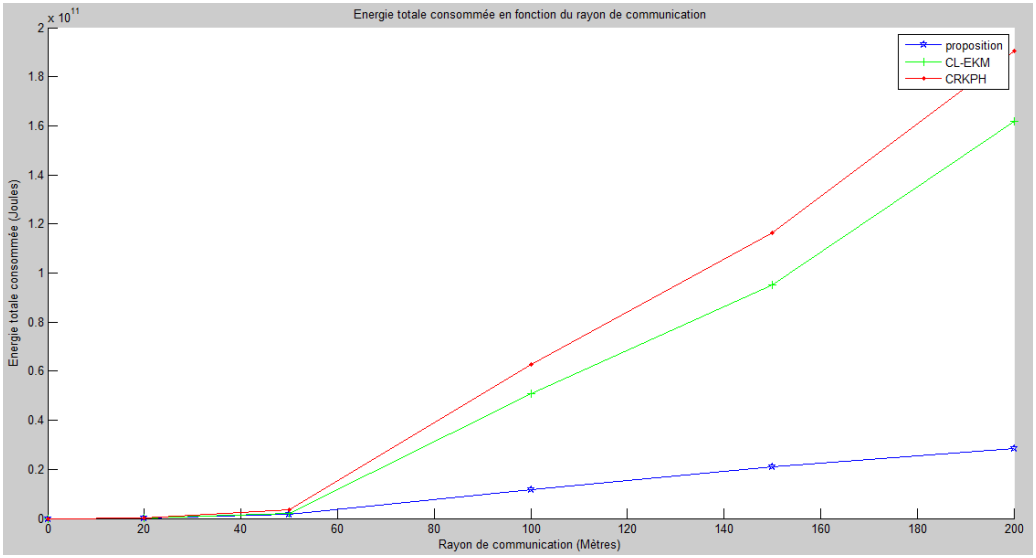

FIGURE 3.9 – Graphe de l'énergie consommée due aux calculs et aux communications en fonction du rayon de communication.

Nous pouvons constaté clairement de la figure 3.9 qu'elle se rapproche beaucoup de la figure 3.8 et ce à cause du coût de calcul négligeable comparativement à l'énergie due aux communications qui est en valeurs de 10^{11}.

jusqu'à un rayon de communication de 50m les 3 protocoles sont aussi faibles les uns que les autres en terme d'énergie due aux communications. En effet ce rayon de 50m relativement à la surface de $1000 * 1000m^2$ donnera une quantité considérable de noeuds isolés n'ayant dont pas la capacité de communiquer avec un CH réduisant ainsi l'énergie due au communications du réseau. Cependant des que le rayon dépasse 50m l'énergie de CL-EKM et CRKPH croie vite à cause de l'énergie que dépense tous les noeuds pour établir des clés de paires et des clés de clusters alors que celle de notre protocole observe une croissance lente liée uniquement à l'établissement de la clé de cluster.

3.5.3 Résultats suivant la fréquence de mobilité

La fréquence de mobilité représente la fréquence à laquelle les noeuds effectuent un mouvement. Pour notre simulation nous avons considéré une fréquence de mobilité de 2s, sur un temps de simulation total de 60s nous donnant ainsi des valeurs pour les critères de d'évaluation défini précédemment, avec un nombre total de 30 mouvements.

3.5.3.1 Coût de stockage

La figure 3.10 représente le coût de stockage (bits), du PPBKM et des deux autres protocoles en fonction de la fréquence de mobilité des noeuds dans le réseau.

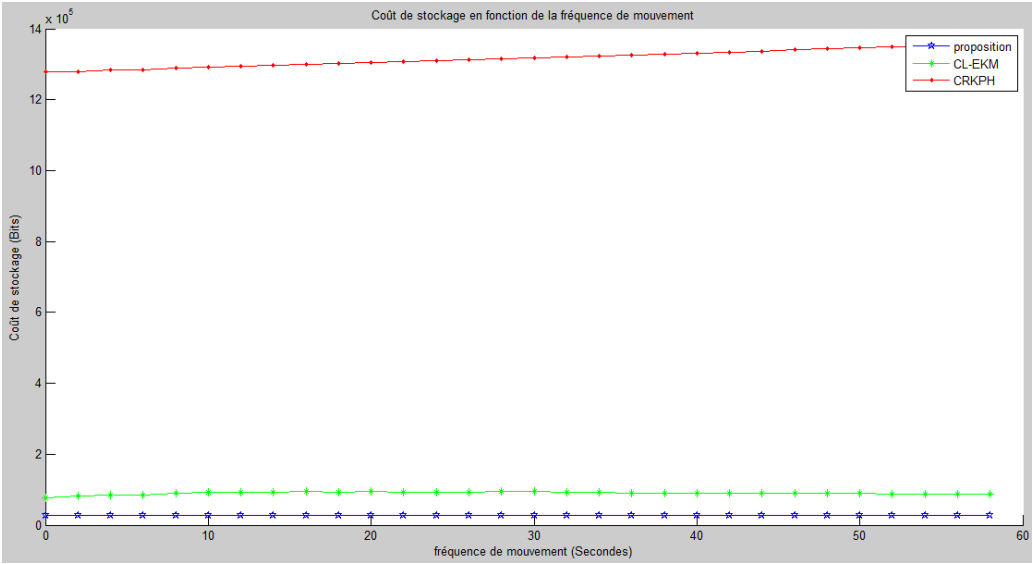

FIGURE 3.10 – Graphe du coût de stockage en fonction de la fréquence de mouvement.

Nous constatons sur la figure 3.10, le coût de stockage du CRKPH est élevé dés le 1er déploiement et augmente au fur et à mesure des mouvements des noeuds à cause de la clé que stocke chaque neoud à chaque fois qu'il veut quitter un CH. Pour le CL-EKM et notre protocole, les valeurs de stockage dés le départ sont plutôt faibles ; mais l'augmentation se fait plus consistante pour le CL-EKM au fur et à mesure des mouvements à cause des clés de paires qu'un noeud peut avoir à chaque nouveau mouvement. Alors que pour notre protocole aucune clé de paire n'est établie et donc rajoutée, donc le nombre de clés de départ reste le même si le noeud se déplace et qu'il n'est pas isolé. S'il est isolé, il perd encore 2 clés (cluster et publique du CH) et donc le stockage de départ baisse. Notre protocole est donc tout à fait adapté pour un mouvement aléatoire des noeuds.

3.5.3.2 Énergie consommée due aux communications

La figure 3.12 représente l'énergie consommée due au communications, du PPBKM et des deux autres protocoles en fonction de la fréquence de mobilité des noeuds dans le réseau.

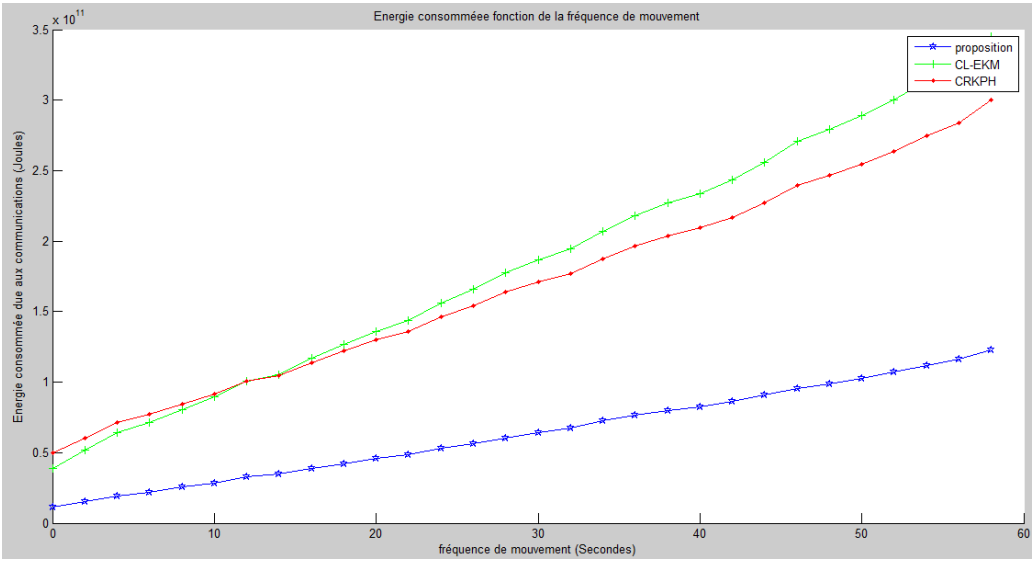

FIGURE 3.11 – Graphe de l'énergie consommée due aux communications en fonction de la fréquence de mouvement.

Comme le montre la figure 3.11, pour les 2 protocoles CL-EKM et CRKPH plus il y a de mouvements plus l'énergie consommée pour les communications du réseau augmente et ce de façon conséquente comparativement à notre protocole qui connaît une croissance moins élevée. La raison de ce résultat est le fait que notre protocole pour l'étape simulée (mouvement des noeuds, clustering et renouvellement de clés), ne nécessite que très peu de communications pour permettre à un noeud de quitter ou de rejoindre un cluster, alors que le CL-EKM et encore plus le CRKPH effectue énormément de communications pour cette étape. Aussi, à chaque mouvement chaque noeud du CL-EKM et CRKPH en plus des communications pour le renouvellement des clés de cluster, effectue des communications pour établir les clés de paires avec leurs nouveaux voisins.

3.5.3.3 Énergie totale consommée due aux communications et aux calculs

La figure 3.12 représente l'énergie consommée due au communications, du PPBKM et des deux autres protocoles en fonction de la fréquence de mobilité des noeuds dans le réseau.

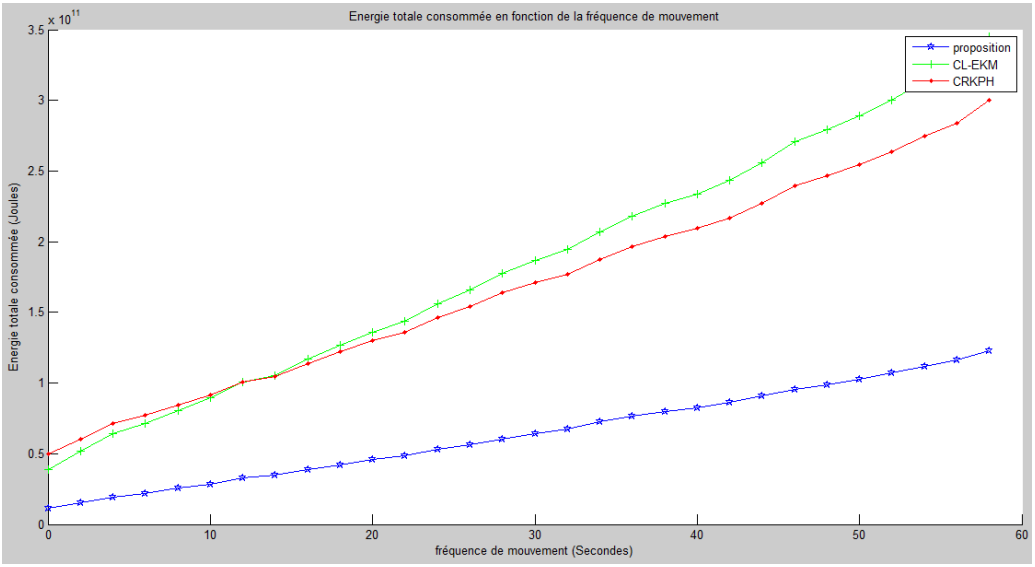

FIGURE 3.12 – Graphe de l'énergie consommée due aux calculs et aux communications en fonction de la fréquence de mouvement.

Nous pouvons constaté clairement de la figure 3.12 qu'elle se rapproche beaucoup de la figure 3.11 et ce à cause du coût de calcul négligeable comparativement à l'énergie due aux communications qui est en valeurs de 10^{11}. La croissance plus rapide de l'énergie totale dans le CL-EKM et le CRKPH est due à l'énergie que dépense tous les noeuds pour ré-établir à chaque mouvements des clés de paires et des clés de clusters alors que celle de notre protocole observe une croissance plus lente liée uniquement au ré-établissement de la clé de cluster.

3.6 Conclusion

Après avoir fait une simulation complète de notre proposition et des deux protocoles appartenant à la classe de protocoles de gestion de clés à gestion dynamique et distribuée à capteurs mobile de notre classification des protocoles de gestion de clés, nous avons confirmé mathématiquement l'aspect économique en terme de stockage et d'énergie de notre proposition pour l'étape la plus coûteuse dans chacun des 3 protocoles : l'étape de mouvement des noeuds avec clustering et renouvellement des clés. Nous pouvons déduire des résultats de cette simulation la réelle tentative de notre protocole de se rapprocher d'une solution la mieux adaptée aux ressources limitées des WSN.

Conclusion générale et perspectives

Grâce à la miniaturisation des composants et à l'allongement de la durée de vie des batteries, les WSN constituent un domaine de recherche passionnant qui peut avoir un apport important dans diverses disciplines telles que les sciences, la sécurité militaire et la communication. Cependant les WSN possèdent des contraintes avérées, dont les solutions représentent une véritable prouesse que les chercheurs du domaine doivent accomplir, nous citerons les plus critiques telles que les ressources limitées des capteurs ou encore la sécurité des données et du capteur lui même. La solution à ces exigences serait de développer un protocole de gestion de clés spécifique pour les WSN, qui optimiserait la consommation d'énergie lors de la gestion d'un nombre minimum de clés cryptographiques aptes à totalement sécuriser les communications et le réseau de capteurs.

Dans le travail consigné dans ce mémoire, nous avons étudié, proposé une classification et critiqué plusieurs schémas de gestion de clés proposés dans la littérature, pour arriver à une synthèse des meilleures techniques adaptées aux WSN. Sur la base de notre synthèse, notre contribution consiste en une proposition d'un protocole de gestion de clés pour les WSN mobiles. Dans cette proposition nous avons optimisé l'usage des ressources notamment l'énergie qui a un fort impact sur la durée de vie du réseau, tout en garantissant un niveau de sécurité correcte (résistance aux noeuds compromis, au clonage et la sécurité au présent et au passé).

Notre solution montre à travers les résultats de simulation et d'évaluation qu'elle a optimisé l'énergie totale consommée et le coût de stockage (nombre de clés) tout en garantissant la sécurité. Cependant, la sécurité de se protocole reste dans le cadre du modèle d'attaque pris en compte et n'est donc pas infaillible à tout les types d'attaques répertoriés. La simulation a aussi montré que même si sur l'énergie totale consommée l'impact de l'énergie due au calculs est imperceptible, notre solution s'avère tout de même un peu plus coûteuse en terme de calculs que les autres solutions (CL-EKM, CRKPH). Il serait donc plausible, comme perspectives de notre travail de tenter d'améliorer cet aspect tout en préservant le niveau de sécurité existant

ou même d'étendre le modèle d'attaques à plusieurs autres attaques existantes pour garantir plus de sécurité.

Bibliographie

[1] Définition capteurs. http ://www.aquaportail.com/definition-8834-capteurs.html. Accessed : 2016-06-20.

[2] Diffie-hellman key exchange (exponential key exchange). http ://searchsecurity.techtarget.com/definition/Diffie-Hellman-key-exchange. Accessed : 2016-06-27.

[3] Présentation de matlab. "http ://www.samuelboudet.com/fr/matlab", 07/06/2016.

[4] S. Athmani. Protocole de securité pour les reseaux de capteurs sans fil. Master's thesis, Université Hadj Lakhder, Batna, 2010.

[5] H. Chan and Perrig. Pike : peer intermediaries for key establishment in sensor networks. In *Proceedings of the 24th annual joint conference of the IEEE computer and communications societies*, volume 1, pages 524–535, 2005.

[6] H. Chan and A. Perrig. Random key predistribution schemes for sensor networks. In *Proceedings of the 2003 IEEE symposium on security and privacy*, volume 1, pages 197–213, Miami, 2003.

[7] G. de Meulenaer, F. Gosset, F. Standaert, and O. Pereira. On the energy cost of communication and cryptography in wireless sensor networks. In *2008 IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*, volume 1, pages 580–585, 2008.

[8] B. Eschenauer, L ANDS Gligor. A key management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on computer and communication security*, volume 1, pages 41–47, New York, 2002.

[9] N. Gura, A. Patel, A. Wander, and al. Comparing elliptic curve cryptography and rsa on 8-bit cpus. In *in Proc. 6th Int. Workshop Cryptograph. Hardw. Embedded Syst.*, volume 1, pages 119–132, 2004.

[10] P. Hazra, D. Giri, and A. Das. Key chain-based key predistribution protocols for securing wireless sensor networks. In *Proceedings in Mathematics Statistics*, volume 139, pages 135–154, Haldi, India, 2015.

[11] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless micro sensor networks. In *Proceedings of the 33rd IEEE Hawaii International Conference on System Sciences HICSS*, volume 1, 2000.

[12] Z. Junqi and V. Vijay. Wireless sensor network key management survey and taxonomy. *Journal of Network and Computer Applications*, 33 :63–75, 2010.

[13] K. Mu and L. Li. Centralized key management scheme in wireless sensor networks. *Wireless Pers Commun*, 60 :463–474, 2011.

[14] K. Mu and L. Li. An efficient pairwise key predistribution scheme for wireless sensor networks. *Journal of networks*, 9 :277–282, 2014.

[15] A. Perrig, R. Szewczyk, V. Wen, D. Cullar, and J. Tygar. Spins : security protocols for sensor networks. In *Proceedings of the 7th annual ACM/IEEE international conference on mobile computing and networking*, volume 1, pages 189–99, Massachusetts, 2001.

[16] M. Rassam, M. Maarof, and A. Zainal. A survey of intrusion detection schemes in wireless sensor networks. *Amer. J. Appl. Sci.*, 9 :1636–1652, 2012.

[17] Z. Sencun, S. Sanjeev, and J. Sushil. Leap : Efficient security mechanisms for large-scale distributed sensor networks. In *the 10th ACM conference on Computer and comunications security*, volume 1, pages 62–72, New York, 2003.

[18] S. Seung-Hyun, W. Jongho, S. Sultana, and E. Bertino. Effective key management in dynamic wireless sensor networks. *IEEE, Transactions on information forensics and security*, 10 :371–383, 2015.

[19] N. Suganthi and V. Sumathy. Energy efficient key management scheme for wireless sensor networks. *Int J Comput Commun*, 9 :71–78, 2014.

[20] R. Vaid and V. Katiyar. Vlkm :virtual location-based key management scheme for wireless sensor networks. In *International Conference on Parallel*, volume 1, pages 53–61, Ambala, India, 2014.

[21] H. Xiaobing, N. Michael, and d. M. Hermann. Dynamic key management in wireless sensor networks : A survey. *Journal of Network and Computer Applications*, 36 :611–622, 2013.

[22] Z. Xing, H. Jingsha, and W. Qian. Eddk : Energy-efficient distributed deterministic key management for wireless sensor networks. *Journal on Wireless Communications and Networking*, 1 :11–18, 2011.

[23] Y. Zhang and P. Ji. An efficient and hybrid key management for heterogeneous wireless sensor networks. In *Control and Decision Conference*, volume 1, pages 1881–1885, Shanghai, China, 2014.

[24] W. Zhu and J. e. a. Zhou. Detecting node replication attacks in mobile sensor networks : Theory and approaches. *Secur. Commun. Netw.*, 5 :496–507, 2012.