UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À

L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE

DE LA MAÎTRISE

EN

MATHÉMATIQUES ET INFORMATIQUE APPLIQUÉES

PAR

HAYETTE BOUMERZOUG

# GESTION DE LA SÉCURITÉ DANS LES RÉSEAUX DE CAPTEURS SANS FIL

OCTOBRE 2011

## *REMERCIEMENTS*

Je tiens à remercier, tout d'abord, mon directeur de mémoire Boucif Amar Bensaber de l'aide qu'il m'a apporté, de son sens de minutie au travail qui m'a permis de réaliser des articles scientifiques de qualité pour accomplir ce mémoire, son esprit critique qui m'a permis de remettre en question ma méthode et améliorer ma méthode de travail, ce qui a avancé et approfondi mes recherches.

Je remercie M. François Meunier, et M. Ismaïl BISKRI de l'Université du Québec à Trois-Rivières, d'avoir pris le temps d'évaluer mon mémoire, et de prendre aussi l'énergie nécessaire à cette évaluation.

Je remercie mon mari et mes frère Samir et Mohamed de leur soutien ainsi que leurs encouragements au cours de la réalisation de ce mémoire.

Je remercie mes parents de m'avoir encadré alors que je développais mon potentiel dès le plus jeune âge.

J'aimerais adresser un remerciement particulier à Daniel St-Yves et Guy Therrien pour leur aide et leur soutien technique.

Enfin, j'adresse mes plus sincères remerciements à mes deux familles Boumerzoug et Boucher, et à tous mes proches et amis, qui m'ont toujours soutenue et encouragée au cours de la réalisation de ce mémoire.
Merci à tous et à toutes.

# RÉSUMÉ

Les réseaux de capteurs sans fil ont beaucoup d'applications dans les domaines militaires et civils. Ils sont utilisés dans les activités de surveillance, de contrôle médical et de contrôle environnemental. En fait, un réseau de capteurs sans fil WSN (*Wireless Sensor Networks*) est un type spécial de réseaux ad hoc composé d'un ensemble de nœuds déployés dans une zone géographique. Chaque capteur est en mesure de recevoir et de transmettre des données de façon autonome à un nœud puits (*Sink*) qui est connecté à l'utilisateur du réseau via Internet ou par un satellite, par exemple.

Ces réseaux sont caractérisés par un déploiement à grande échelle, d'une mobilité des nœuds, d'une topologie dynamique et par des ressources minimes. En effet, chaque nœud a des contraintes extrêmes en termes d'énergie, de mémoire, de vitesse de calcul et de débit.

Nous proposons un protocole de sécurité qui offre une bonne protection en tenant compte des caractéristiques limitées des capteurs. Le protocole est basé sur une gestion des clés efficaces et résistantes avec un stockage minimal des clés.

Notre approche est fondée sur la combinaison et l'amélioration de deux méthodes déjà proposées par la communauté de recherche :

1. Une Cryptographie basée sur les Courbes Elliptiques

2. Une gestion de clés basées sur un arbre AVL.

Dans notre mémoire, nous avons décris les réseaux de capteurs sans fil et nous avons expliqué et détaillé les contraintes et les capacités limitées des nœuds. Dans le troisième chapitre, nous allons présenter les courbes elliptiques et leurs applications dans la sécurité des communications, en particulier, la cryptographie.

Après avoir présenté notre méthode, l'architecture du réseau et la méthodologie suivie, nous développerons chacune des approches proposées.

Dans le cinquième chapitre, nous allons présenter les modèles de simulation utilisés, les scénarios et la base de nos métriques d'évaluation.

Enfin, pour montrer l'efficacité de notre méthode, nous avons fait une comparaison avec deux méthodes existantes dans la littérature ce qui nous a permis de conclure l'apport positif de notre méthode pour les réseaux de capteur sans fil.

# ABSTRACT

A wireless sensor network consists of hundreds or thousands of sensor nodes that are powered by batteries and are typically deployed randomly in environments often open. These nodes are small, and therefore, have computational resources, storage memory, communication and power that are very limited. These networks are of particular interest for military, environmental, and medical applications, and for applications related to monitoring of critical infrastructures. But not all applications always have the same security constraints. Data encryption is often required for sensitive applications such as military applications and medical applications.

In many applications, the establishment of encryption keys must be done once the sensor nodes are deployed. This is the case when a large number of sensors are deployed randomly, for example, from an helicopter. Each node must establish a secret key with each of its neighbors but its neighbors are not known before the deployment. The keys exchange should take place as soon as the deployment starts.

In this paper; we propose a security protocol for sensor networks that provides good protection while taking into account the limited resources of the sensors. The protocol is based on an effective key management method with a minimum storage of keys. Our approach is based on the combination and improvement of two methods already proposed by the research community: cryptography based on elliptic curves and key management based on an AVL tree. Compared with RECC "A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks" and CECKM "High-Effect Key Management Associated With Secure Data Transmission Approaches in Sensor Networks Using a Hierarchical-based Cluster Elliptic Curve Key Agreement" two methods based on Diffie-Hellman elliptic curve cryptography method, our approach provides a positive impact on reducing energy consumption and memory

storage. It saves significant time and memory and it reduces the exchanged packets during keys installation with fewer processing operations.

# TABLE DES MATIÈRES

## 4. LA MÉTHODE PROPOSÉ



## 5. SIMULATION

# TABLE DES FIGURES

# TABLE DES TABLEAUX

# CHAPITRE 1
# PROTOCOLES DE GESTION DES CLÉS DE CRYPTAGES

## 1.1. Introduction

Un réseau de capteurs sans fil est composé de centaines voire de milliers de nœuds de capteurs qui sont alimentés par des piles et qui sont typiquement déployés de façon aléatoire dans des environnements souvent ouverts. Ces nœuds sont petits, et par conséquent, ont des ressources de calcul, de stockage, de communication et d'énergie très limitées. Ces réseaux, ont un intérêt particulier pour les applications militaires, environnementales, médicales, et les applications liées à la surveillance des infrastructures dites critiques. Ces applications ont souvent besoin d'un niveau de sécurité élevé.

Mais toutes les applications n'ont pas toujours les mêmes contraintes de sécurité. Le chiffrement des données est souvent requis pour des applications sensibles telles que les applications militaires ou les applications médicales.

Dans beaucoup d'applications, l'établissement des clés doit se faire une fois que les nœuds capteurs sont déployés. C'est le cas par exemple, lorsqu'un grand nombre de capteurs sont déployés de façon aléatoire, par exemple, à partir d'un hélicoptère. Chaque nœud doit établir une clé secrète avec chacun de ses voisins. Ses voisins ne sont pas connus avant le déploiement. Les échanges de clés doivent avoir lieu sur le terrain.

## 1.2. Les approches de gestion de clés

Divers solutions ont été présentées dans la littérature. Souvent, elles utilisent des méthodes dynamiques recommandant la mise à jour des clés pour offrir une résistance contre les attaques pour une longue période.

Dans ce paragraphe, je vais exposer les différentes approches de gestion des clés de cryptage pour les réseaux de capteurs en spécifiant leurs avantages et leurs inconvénients.

### 1.2.1. Protocoles probabilistes

Eschenauer et Gligor [1] ont proposé, en 2002, un protocole probabiliste. Ce protocole est relativement simple et il opère comme suit :

- L'administrateur du réseau génère un tableau des nombres aléatoires, indexé de 1 à L.
- Chaque nœud est configuré avec un sous-ensemble de m clés choisies aléatoirement parmi les clés précédentes.
- Lorsque deux nœuds, A et B, veulent établir une clé secrète, ils échangent les indexes de leurs clés, ensuite ils utilisent les clés qu'ils ont en commun pour générer un secret.

Dans ce protocole, il est toujours possible qu'un nœud, autre que A et B possède également les clés partagées entre A et B, alors, ce nœud pourra calculer le secret que B et A viennent d'établir. Cependant Eschenauer et Gligor ont montré que cette probabilité peut être faible si les paramètres L et m sont choisis avec précaution.

Les méthodes définies dans [2],[3], [4], [5], [6] améliorent le protocole probabiliste [1]. Elles proposent la pré-distribution des clés de façon aléatoire, ce qui implique la création d'une grande liste de clés par un administrateur du réseau, et avant le déploiement, un sous-ensemble de cette liste choisi aléatoirement est distribué à chaque nœud.

Pour établir une clé de liaison entre deux nœuds, il est possible que ces nœuds possèdent des clés communes dans leurs sous-ensembles de clés, alors ils utilisent les clés communes.

Le choix des sous-ensembles est basé sur la probabilité d'Eschenauer et Gligor [1]. Un des inconvénients de ces approches est le nombre de clés qui peut êtres est très grand.

La sécurité que le protocole probabiliste fournit, peut se dégrader rapidement avec le temps. En effet, chaque fois qu'un nœud est compromis, ces clés sont révélées.

Au fil du temps et en faisant l'hypothèse que l'attaquant compromet continuellement des nœuds, une grande partie des clés du système sera connue de l'attaquant. Celui-ci peut, alors, calculer les clés établies entre les nœuds.

### 1.2.2. Le protocole ROK

Pour résoudre le problème du protocole probabiliste C. Castelluccia et A. Francillon[3] ont proposé RoK, une extension du protocole précédent.

L'idée principale du protocole RoK est de limiter la durée de vie des clés secrètes dans le tableau et de faire évoluer les clés du système en fonction du temps.

Une solution naïve et simple serait, que le système change les clés dans le tableau à chaque période. À une période donnée T, un nœud pourrait échanger des secrets avec ses voisins et ensuite effacer les clés de configuration de sa mémoire. Par conséquent, si ce nœud est compromis plus tard, ses clés ne pourront pas être utilisées.

Pour qu'un nœud déployé à la période T + 1, RoK a attribué à chaque nœud (w * m) clés, ou w est la durée de vie moyenne d'un capteur.

Si un nœud est déployé à la période T, il sera configuré avec m clés de la période T, m clés de la période T + 1,..., et m clés de la période T + w.

Ainsi ce nœud pourra échanger un secret avec des nœuds de sa génération, mais aussi des nœuds des générations T + 1, T + 2, , T +w.

Si un nœud est déployé à la période T+1, il sera configuré avec m clés de la période T+1, m clés de la période T + 2,..., et m clés de la période T + w. Ainsi ce nœud pourra échanger un secret avec des nœuds de sa génération, mais aussi des nœuds des générations T + 2, T + 3,..., T + w.

Un des inconvénients de cette approche est qu'elle augmente le nombre de clés que chaque nœud doit stocker par un facteur w. Pour résoudre ce problème, une nouvelle construction est développée en utilisant une chaine de haschs doublement chaînée. Elle permet de réduire le coût mémoire de (w * m) à (2 *m), c'est à dire à un coût constant.

### 1.2.3. Protocoles basés sur la connaissance antérieure du déploiement

Afin que les nœuds de capteurs puissent avoir une meilleure probabilité et pour identifier l'ordre arrangé au préalable des nœuds voisins, Du et autres [2] ont proposé la connaissance antérieure du déploiement où les nœuds sont arrangés du préalable dans un ordre avant le déploiement.

Si les nœuds voisins sont connus, la pré-distribution des clés devient insignifiante; il suffit juste de produire une clé partagée entre chaque deux nœuds voisins. Ceci garantit que chaque nœud peut établir un canal bien fixé avec chacun de ses voisins après le déploiement.

### 1.2.4. Protocoles basés sur les clés aléatoires

Dans [11], E.Munivel et autres ont proposé la méthode MICRO PKI (Micro public Key Infrastructure).

Dans cette approche, seulement la station de base est authentifiée par les nœuds utilisant une clé publique. Cette clé est utilisée aussi pour décrypter quelques données envoyées par les nœuds. La station de base utilise une clé privée.

#### 1.2.4.1. Architecture du réseau

Le réseau est composé d'un ensemble de nœuds de capteur sans fil connectés l'un à l'autre.

- Chaque nœud est capable d'utiliser le chiffrage symétrique et le chiffrage asymétrique.

- Chaque nœud a la capacité de sauvegarder au moins la clé publique de la station de base et une clé de session utilisée pour chiffrer les données.

- Chaque nœud obtient la clé publique de la station de base avant le déploiement.

#### 1.2.4.2. L'installation des clés

**Établir la liaison entre la station de base et les nœuds**

Quand un nœud aura besoin d'établir une liaison garantie avec la station de base :

1- il produit une clé aléatoire, il la chiffre avec la clé publique de la station de base, puis il l'envoie à la station de base.

2-Quand la station de base reçoit le message, elle le décrypte avec sa clé privée et elle stocke la clé reçue dans une table associée à chaque nœud.

3-la station de base ré-envoie à ce nœud un message OK chiffré avec la nouvelle clé de session.

Après l'établissement de cette clé de session le nœud et la station de base commence à l'utiliser pour crypter leurs échanges jusqu'à la mise à jour de la clé suivante.

**Établir la liaison entre deux nœuds**

Après la phase d''établissement des clés entre chaque nœud et la station de base; alors si deux nœuds ont besoin d'établir un canal sécurisé entre eux, ils exécutent les étapes suivantes :

1-Un des deux nœuds envoie une demande à la station de base pour établir un lien sécurisé avec l'autre nœud. Cette demande contient l'identificateur de l'autre nœud.

2-En recevant cette demande, la station de base produit une clé aléatoire et elle envoie à chacun une copie chiffrée avec la clé correspondante.

3-En recevant la nouvelle clé, les deux capteurs commencent à l'utiliser pour crypter les échanges entre eux.

### 1.2.4.3. Mise à jour des clés

Quand un nœud arrive à une certaine période (paramètre relié à la longueur des clés et la complexité de l'algorithme utilisé) il lance une nouvelle installation; qui est réellement une mise à jour des clés.

#### 1.2.4.4. Ajout d'un nouveau nœud

Si un nouveau nœud veut joindre le réseau, l'administrateur du réseau doit charger la clé publique de la station de base dans ce nœud, après le déploiement, le nouveau nœud peut automatiquement établir une liaison avec la station de base et les nœuds voisins comme décrit auparavant.

### 1.2.5. Protocoles basés sur les arbres

#### 1.2.5.1. Une gestion de clés basées sur un arbre M-ary

Dans [8], A.S.Poornima et autres ont proposé une méthode basée sur l'arbre M-ary. Dans ce cas-ci, les capteurs d'un cluster sont organisés sous forme d'un arbre équilibré comme indiqué dans la figure 1.1. Cet arbre est maintenu par un leader, qui est désigné par 'H-nœud.

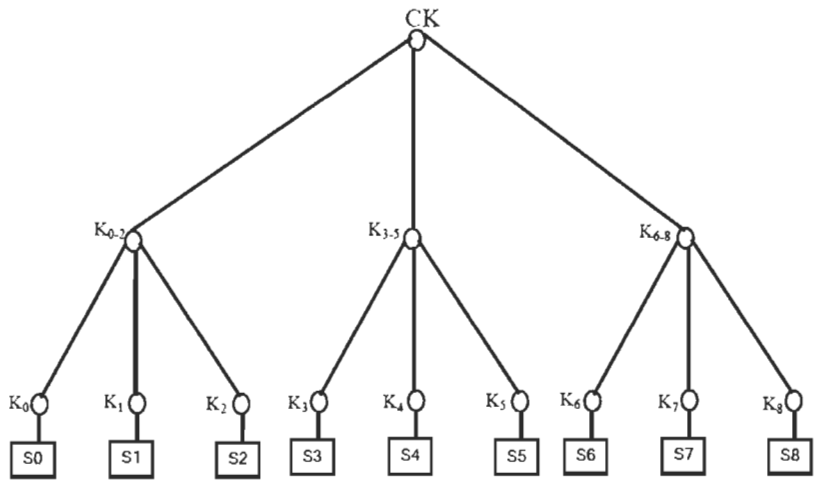



**Figure 1.1. Arbre M-ary des clés.**

Les feuilles s0, s2, …, s8 représentent les nœuds dans le cluster.

Chaque nœud partage avec le leader sa clé privée utilisée pour communiquer avec le leader.

Les nœuds k0, k1, … k8 dans l'arbre correspondent aux clés privées.

Les clés k0-2, k3-5, k6-8 représentent les clés qui sont partagées par un certain sous-ensemble de nœuds (dites des clés intermédiaires).

CK est la clé de groupe qui est partagé par tous les nœuds dans le cluster.

Chaque capteur stockera toutes les clés le long du chemin de la feuille à la racine de l'arbre.

Tous les nœuds leaders dans le réseau forment un autre arbre qui est géré par la station de base. La clé partagée par tous les leaders est la tête du cluster dite CCHK. Les leaders utilisent CCHK pour communiquer entre eux.

#### 1.2.5.1.1. Établissement des clés

**Initialement :**

- Chaque capteur est pré chargé d'une clé privée avec laquelle il échange avec le leader avant le déploiement.
- Tous les leaders sont pré chargés de toutes les clés qui sont assignées aux nœuds.

**Après le déploiement :**

- Tous les leaders émettent le message « Hello » aux nœuds normaux.

- Chaque nœud normal choisit son leader LD dont le message « Hello » a la meilleure proportion de bruit du signal.
- Après la réception de la réponse des nœuds, chaque leader supprimera les clés des nœuds qui ne sont pas dans son cluster.
- Chaque leader construit un arbre et assigne des clés dans son arbre pour chacun des nœuds qui a répondu à son message « Hello ».
- Le leader distribue toutes les clés le long du chemin de la feuille à la racine de l'arbre en chiffrant les clés grâce aux clés privées des nœuds.
- En recevant l'ensemble des clés, les nœuds peuvent communiquer avec leur leader en utilisant la clé du cluster CK aussi bien qu'avec les autres nœuds dans le cluster en utilisant les clés intermédiaires.

#### 1.2.5.1.2. Révocation de Nœud, Nœuds compromis

Aussitôt qu'un nœud est compromis, le leader changera toutes les clés le long du chemin allant de la position du nœud compromis à la racine de l'arbre.

Les clés changées sont distribuées de façon sécuritaire aux autres nœuds en utilisant les clés intermédiaires.

#### 1.2.5.1.3. Addition d'un Nouveau nœud

- Un nouveau nœud est pré-chargé d'une clé privé qu'il partage avec le leader. La station de base chiffre cette clé privée en utilisant la clé CCHK, et l'envoie à tous les leaders.
- En recevant le message, chaque leader à toute l'information concernant le nouveau nœud.
- Chaque leader envoie le message « Hello » pour ajouter le nouveau nœud de capteur.

- Le nouveau nœud choisit son leader LD dont le message « Hello » à la meilleure proportion de bruit du signal
- Le leader sélectionne une position pour le nouveau nœud dans l'arbre et l'arbre est mis à jour (c'est-à-dire, toutes les clés le long du chemin incluant la clé du cluster CK sont changées).
- Le leader distribue la nouvelle clé du cluster à tous les nœuds sauf le nouveau nœud, le message est chiffré en utilisant l'ancienne clé CH.
- Le restant des clés changées sont chiffrées en utilisant la vieille clé intermédiaire respective, puis ces clés seront distribuées à tous les autres nœuds du cluster.
- Pour distribuer toutes les clés changées au nouveau nœud, le leader les chiffre avec la clé privée du nouveau nœud.

#### 1.2.5.1.4. Rafraichissement des clés

La clé du cluster CK est changée à CK ' par les leaders, et elle est distribuée en la chiffrant avec l'ancienne clé CK.

De la même façon la station de base changera CCHK à CCHK ' et la distribue à tous les leaders solidement en chiffrant CCHK' avec CCHK.

Comme ont peut le remarquer dans cette méthode, chaque sous-ensemble de nœuds a une seule clé partagée et elle est utilisée pour la communication avec les nœuds; alors si un seul nœud est compromis, un adversaire peut écouter tous les messages entre tous les nœuds du sous-ensemble.

### 1.2.5.2. Gestion des clés dynamique utilisant un arbre AVL

Pour améliorer la sécurité des capteurs et soutenir le mécanisme dynamique dans la gestion des clés, Yi-Ying Zhang et autres [9] ont proposé une méthode de pré-distribution aléatoire des clés utilisant un arbre AVL (en raison de la caractéristique que la position de nœuds dans l'arbre AVL peut être changée dynamiquement en temps réel).

Le réseau de capteurs sans fil est hiérarchique et les nœuds sont organisés dans des clusters par les leaders LDs (le leader est celui qui a le plus d'énergie). Le leader rassemblent les messages des nœuds non leaders dans les clusters, et les font suivre à la station de base BS.

Les clés sont établies sur deux niveaux AVL [10], le premier présente les nœuds du cluster et l'arbre est maintenu par le leader; le deuxième niveau présente les leaders du réseau et l'arbre est maintenu par la station de base.

Les processus de deux niveaux d'établissement de l'arbre des clés sont semblables.

#### 1.2.5.2.1. Gestion d'initialisation des clés

Pour créer les clusters, chaque leader diffuse un message de chiffrage avec son ID et une clé aléatoire KH pour informer tout le réseau.

Quand un nœud *Nu* non leader reçoit le message, il utilise une fonction de hachage F et la clé KH pour produire la clé secrète temporaire *Ku,H* et il envoie cette clé avec son identificateur pour joindre le cluster.

Après un certain temps, chaque leader reçoit des messages d'ajout des nœuds, puis il rassemble tous les identificateurs et les clés des nœuds de son cluster et il produit l'ensemble des clés en se basant sur un arbre AVL en utilisant les données duelles (la clé

secrète temporaire et l'identificateur de chacun des nœuds) selon le sort des valeurs des clés.

Une fois que l'ensemble des clés est produit, le leader ré-envoie à chaque nœud dans le cluster un message « «ajout-réponse » contenant la position de l'identificateur du nœud dans l'arbre AVL.

Le leader utilise la position de chaque nœud *Nu* dans l'arbre AVL pour produire la clé du nœud *Nu* en utilisant la fonction d'hachage F.

Maintenant, les nœuds du cluster choisi un autre leader (celui qui a plus d'énergie), et ils envoient un message d'ajout à ce nœud.

Ainsi le CH compose le groupe et établit ensuite un TDMA (Time Division Multiple Access-la Division de Temps d'Accès Multiple) et transmet ce calendrier aux nœuds dans le groupe.

#### 1.2.5.2.2. Le modèle du Système

Avant le déploiement, chaque nœud est pré-chargé avec une clé aléatoire initiale, cette clé est partielle.

Après le déploiement chaque nœud *Nu* génère sa clé principale Ku en utilisant la clé initiale Ki et une fonction d'hachage F, puis il émet un message contenant son identificateur et sa clé Ku, et il attend la réponse de chaque voisin *Sv* avec son identificateur.

Après un temps, chaque voisin *Nv* répond avec son identificateur et sa clé Kv, puis chaque paire de nœuds voisins Nu et Nv calculent leur clé partagée.

Chaque nœud peut être authentifié par ses nœuds immédiats. S'il y a quelques nœuds malveillants, ils peuvent être distingués avec cette approche.

#### 1.2.5.2.3. La gestion dynamique des clefs

Après l'établissement des arbres AVL, la tâche des nœuds se déplace à l'étape stable (steady stage) et le réseau fonctionne de façon régulière.

Une fois que, les comportements non-normaux d'un nœud (comme des fausses clés de communication) sont détectés par un leader, il informe tous les nœuds de son groupe pour mettre à jour la vieille clé et reconfigurer l'arbre AVL, ainsi la nouvelle clé du cluster remplace l'ancienne clé.

En outre, dans le cas où la station reçoit un certain nombre de messages, elle doit informer tous les leaders de faire une mise à jour des clés.

Dans le cas où un leader CH reçoit un certain nombre de messages, il doit informer tous les nœuds dans le cluster de faire une mise à jour des clés.

L'inconvénient dans ce protocole est que chaque cluster, à un certain temps, doit changer son *leader* (choisir celui qui a le plus d'énergie) et cette opération présente une grande communication entre les nœuds dans le cluster pour établir les clés à chaque session.

### 1.2.6. Protocoles basés sur la pré-distribution d'une liste partielle de clés

La méthode SecCOSEN [10] est basée sur la pré-distribution de clés partielles et la cryptographie symétrique. Avant le déploiement, une liste des sous ensembles des clés est produite à la station de base. Chaque nœud est pré-chargé de la liste des clés. Une fois qu'un nœud a les clés qu'il utilisera avec ses voisins, il supprime le reste des clés.

Dans SecCOSEN on utilise l'appellation chaîne au lieu de cluster, ces chaînes sont composées d'un nombre fixe de capteur qui sont les membres dans la chaîne et un nœud leader qui est le leader du niveau inferieur. Dans chaque chaîne, les données agrégées par les capteurs sont acheminées le long des chaînes vers le leader de la chaîne, et ce dernier rassemble les données agrégées et les transmet à la station de base.

#### 1.2.6.1. L'architecture de protocole SecCOSEN

Les chaînes du niveau inférieur sont formées de tous les nœuds dans le réseau et chaque chaîne choisi un leader, le choi est basé sur quelques critères. Ces leaders de niveau inférieur forment à leur tour une chaîne de niveau plus haut dont un leader seul appelé le leader de niveau plus haut, ce leader est choisi (selon quelques paramètres) pour transmettre les données à la station de base.

La transmission des données collectées au niveau inférieur à la station de base passe par la chaîne du nœud se trouvant avant le leader, puis par la chaîne des leaders se trouvant avant la station de Base, comme indiqué dans la figure 1.2.

#### 1.2.6.2. Caractéristiques de SecCOSEN

- Le protocole change les leaders de niveau inférieur après un nombre optimal de tours (Distribuer la charge, Optimiser l'énergie)
- La phase de formation de la chaîne ne précède pas la phase de collecte de données sauf quand il est nécessaire de reconstruire des nouvelles chaînes.
- le mécanisme de transmission des données est semblable pour les deux niveaux (inférieur et plus haut).
- Au début de chaque tour, chaque leader dans une chaîne de niveau inférieur envoie un signe vers une fin indiquant le début de la phase de transmission des

données. Le nœud à une fin de la chaîne envoie ses données vers le nœud leader par des nœuds intermédiaires.

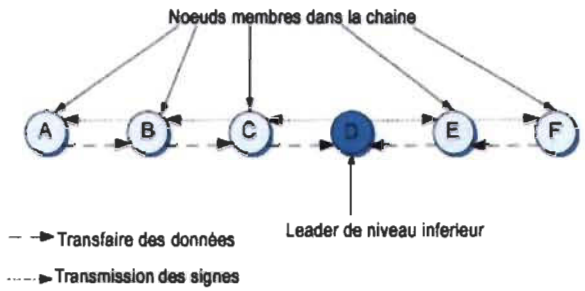



**Figure 1.2. Données et signes passants dans une chaîne.**

### 1.2.6.3. Initialisation

- Le protocole de gestion des clés SecCOSEN est basé sur la pré-distribution des clés et la cryptographie symétrique.
- Chaque nœud garde un sous ensemble de clés plutôt que tout l'ensemble des clés.
- Chaque deux nœuds correspondants utilisent toujours une nouvelle clé secrète pour le cryptage / décryptage des données à chaque tours.
- Deux nœuds voisins établissent leur clé de cryptage / décryptage par la concaténation de leurs clés.

Une cryptographie symétrique est définie comme suit :

$\mu$ : Un message simple

$\mu' = EK(\mu)$ : est un message chiffré

E-1k (μ')= μ : est le décryptage du message chiffré.

k : est la clé secrète

E : est l'algorithme de chiffrage

ID : Identifiant du nœud.

NK : la clé du réseau.

LPK : identificateur d'un sous ensemble des clés.

#### 1.2.6.4. La formation des chaînes

- Une association des clés est produite à la station de base avant le déploiement des nœuds.
- Chaque nœud doit consommer toutes ces clés *(Après la première phase de formation des chaînes, le nœud peut supprimer toutes les clés partielles supplémentaires)*
    - Note : en utilisant $n$ clés, chaque deux nœuds voisins peuvent établir jusqu'à $2n^2$ clés secrètes.
- Avant le déploiement chaque nœud est pré- chargé de l'association des clés S, de la liste des identificateurs des clés L, d'une seule clé du réseau NK, et d'un identificateur unique ID.
- Après le déploiement et la création des chaînes, tous les membres de chaque chaîne envoient leur IDS (chiffré par la clé du réseau) au leader de la chaîne.
- Après la réception de tout les $ID_S$, chaque leader local envoie tout les $ID_S$ reçus avec son propre ID à la station de base pour l'authentification.

### 1.2.6.5. L'utilisation des clés partagées

- Si la station de base authentifie chaque membre dans la chaîne, elle choisit aléatoirement un nombre prédéterminé de clés et un LPK, l'identificateur du sous ensemble de clés.
- La SB envoie le LPK (chiffré par la clé du réseau) au leader du niveau inférieur. Le leader du niveau inférieur dissémine le LPK (chiffré par la clé de réseau).
- Une fois qu'un nœud a su quelles clés il utilisera avec son voisin, il supprime le reste des clés.

### 1.2.6.6. Établir les clés

Soit Nu et Nv deux nœuds voisins, pour établir les clés secrètes :

- Le nœud Nu construit une liste Lu d'ordre $O_U$ qui maintient la même cardinalité avec LPK et l'envoie (chiffré par la clé de réseau) à son voisin Nv.
- Le voisin Nv envoie une liste Lv d'ordre $O_V$ semblable à Lu.
- Les deux nœuds peuvent maintenant construire leurs clés de communication secrètes pour chaque tour.
- Dans la chaîne de niveau plus haut, les leaders de niveau inférieur peuvent construire leurs clés de session d'une façon semblable s'ils partagent quelques clés communes.

Le problème est qu'à l'arrivée d'un nouveau nœud, les nœuds voisins dans la chaîne ne peuvent pas établir des clés avec le nouveau nœud car ils ont déjà supprimé le reste des clés.

### 1.2.7. Protocoles basés sur les courbes elliptiques

Dans le but d'avoir une gestion de clé efficace et réaliser une économie de stockage significative, [13] et [14] ont présenté une Cryptographie de Courbe Elliptique.

#### 1.2.7.1. La Méthode CECKM (Cluster Elliptic Curve Cryptography Key Management)

Cette approche [14] suppose qu'on divise le réseau en plusieurs clusters à l'aide d'un algorithme approprié pour le déploiement. Cette approche consiste en 5 phases.

Soit:

X : le numéro du cluster.

N : le nombre de clusters dans le réseau.

N : le nombre de nœuds dans le cluster.

BSx: une Station de base, qui est leader du cluster X.

Mx: le nombre de nœuds dans le cluster X.

NC : le nombre de nœuds normaux dans le cluster.

NV : le nombre de nœuds normaux voisins.

Le réseau utilise la fonction $y^2=x^3+ax+b$ comme une courbe elliptique, et tous les nœuds dans le réseau (normaux ou station de base) partagent les coefficients a et b et P une valeur publique qui est un point sur la courbe elliptique.

Les étapes de cette approche sont les suivantes :

#### 1.2.7.1.1. La phase 1

Une station de base BSx est déployée comme un contrôleur de la clé de cryptage intra-cluster, qui est utilisée pour les membres du cluster Mxi, où *x* dénote l'identité du cluster et n le nombre des clusters, avec $1<x<n$, *et i* et la quantité de nœuds dans le cluster.

#### 1.2.7.1.2. La phase 2

Le système déploie une station de base RBS qui est la plus puissante et plus sécurisée comme le contrôleur des clés du réseau entier et coopère avec les autres stations de base BSx ($1<x<n$) pour produire les clés du système entier.

#### 1.2.7.1.3. La phase 3

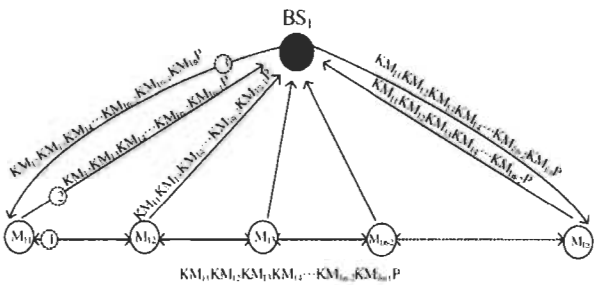



**Figure 1.3. Le calcul de la clé du cluster.**

Le processus de cette phase est détaillé dans trois étapes présentées dans la figure 1.3, ces étapes sont les suivantes:

**Étape1**

Tout d'abord, dans un cluster c, chaque nœud membre dans le cluster $Mc_i$ produit ces clés privées $KMc_i$ et $KMc_iP$ utilisant la méthode Échange de clés Diffie-Hellman ECDH (voire chapitre 4) et émet ensuite la clé $KMc_iP$ à tous les autres nœuds dans le cluster c.

Après un temps, chaque nœud reçoit toutes les clés privées des autres nœuds, il calcule la multiplication de sa clé privée $KMc_x$ et par les autres clés privées reçus.

Finalement chaque nœud à la même clé: $KMc_1KMc_2...KMc_{n-1}P$.

**Étape 2**

Chaque nœud dans le cluster $Mc_i$ extrait sa propre clé privée KMci du produit KM12KM13...KM1N-1P, et transmet $KMc_2KMc_3..KMc_{i-1}KMc_{i+1}..$ $KMc_{n-1}P$ à son leader qui est la station de base BSc.

**Étape 3**

La station de base BSc reçoit le $KMc_2KMc_3..KMc_{i-1}KMc_{i+1}..$ $KMc_{n-1}P$ du nœud $Mc_i$, puis il génère la clé privée du nœud $Mc_i$ qui est la multiplication de la clé privée de la station de base KMcn par la clé reçue, C.-à-d. le produit $KMc_2KMc_3..KMc_{i-1}KMc_{i+1}..$ $KMc_{n-1}PKMcn$.

Lorsque un nœud $Mc_i$ reçoit la clé $KMc_2KMc_3..KMc_{i-1}KMc_{i+1}..$ $KMc_{n-1}PKMcn$, il la multiplie par sa propre clé privée $KMc_i$, et il obtient la clé du cluster $KMc_1KMc_2....KMc_{n-1}KMc_nP$.

De même façon, chaque nœud peut avoir finalement la clé du cluster, cette clé est toujours égale à $\prod_{i=1}^{n} KMc_i \times P$ .Pour chaque station de base BSx cette clé est défini comme KCxP.

### 1.2.7.1.4. La phase 4

Comme il est présenté dans la figure 1.4, cette étape déroule comme le suivant :

La première station de base $BS_1$ sa clé $KC_1P$ à la deuxième station de base $BS_2$.

La station de base $BS_2$ calcule $KC_1KC_2P$, et elle envoi le résultat à la station de base suivant $BS_3$.

De même façons, chaque station de base BSx multiplie sa propre clé privée $KC_X$ avec le reçu de la station de base $BS_{X-1}$ précédente $KC_1KC_2...KC_{x-1}P$ et ensuite elle envoie le résultat à la station de base $BS_{X+1}$ suivante.

Par conséquence, la clé calculée par la station de base $BS_{N-1}$ sera $\prod_{i=1}^{N} KC_i \times P$ .

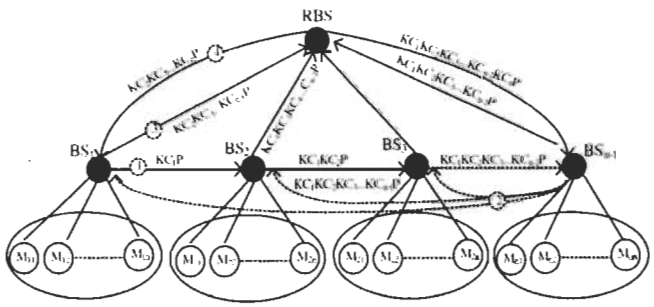



**Figure 1.4. Le calcul de la clé $\prod_{i=1}^{N} KC_i \times P$ .**

#### 1.2.7.1.5. La phase 5

La station de base $BS_{N-1}$ diffuse sa clé privée à tous les autres leaders ($BS_1$, $BS_2$, $BS_3$,…, $BS_{N-2}$).

#### 1.2.7.1.6. La phase 6

Quand une station de base $BS_x$ reçoit le produit $\prod_{i=1}^{N} KC_i \times P$ , elle extrait sa propre clé du produit, puis elle envoie le résultat $KC_1KC_2\ldots KC_{x-1}\ KC_{x+1\ldots}\ KC_{N-1}P$ à RBS la station de base choisie par le système comme un contrôleur.

#### 1.2.7.1.7. La phase 7

Finalement, Quand le RBS obtient la clé d'une station de base BSx, il multiplie sa propre clés KCN par le résultat envoyé par BSx. Puis il envoie le dernier résultat à la station de base BSx.

Quand une station de base BSx recroit le message du RBS qui contient la clé $\prod_{i=1}^{N} KC_i \times P$ { $i \neq x$, $x \in [1,N]$}, elle multiplie la clé envoyé par sa propre clé KCx.

Par conséquence, toute les stations de base peuvent avoir la même clé sera $\prod_{i=1}^{N} KC_i \times P$.

Comme on a pu le constater à travers les différentes phases et étapes, cette méthode exige énormément d'échange entre les nœuds pour établir les clés partagées.

#### 1.2.7.2. Une Cryptographie de Courbe Elliptique basée sur le Protocol de routage pour Réseaux de Capteurs Hétérogènes

La méthode [13] est basée sur le Protocol de routage pour Réseaux de Capteurs Hétérogènes.

Le réseau HSN est un réseau hiérarchique formé d'un *sink* et de deux types de capteurs. Un petit nombre de capteurs puissants servent de leaders et un grand nombre de capteurs normaux forment les clusters. Tous les leaders forment la colonne vertébrale de la communication dans le réseau. Ils accumulent les données délivrées par les nœuds normaux et ils les font suivre ensuite les données au *Sink*.

Cette méthode comprend deux approches, l'une est basée sur l'établissement centralisé des clés et l'autre sur l'établissement des clés distribué.

##### 1.2.7.2.1. L'établissement Centralisé des Clés

**Avant le déploiement**

- Chaque nœud est pré-chargé avec sa clé privée.
- Chaque leader est pré-chargé avec les clés publiques de tous les noeuds.
- La même paire de clés publique/privée ECC peut être utilisée par tous les leaders.
- La clé publique de chaque leader est aussi chargée dans chaque noeud pour authentifier des émissions des leaders.

**Après la sélection d'un cluster**

- Chaque noeud envoie à son leader un message non-chiffré « Keyrequest », qui inclut son ID et son emplacement.

- Après un certain temps, le leader devrait recevoir des messages « Keyrequest » de tous les nœuds dans son cluster et ensuite il utilise l'algorithme de l'arborescence de communication optimale (SPT) [36] pour déterminer l'arborescence dans le groupe.
- Ensuite, le leader produit les clés- partagés pour chaque noeud et ses voisins directs. Pour un nœud Nu et son voisin Nv, le leader produit une nouvelle clé (Ku,v)
- Le leader chiffre le message contenant la clé (Ku,v) en utilisant la clé publique de Nu et ensuite il envoie le message au nœud Nu.
- Le nœud Nu décrypte le message et obtient la clé partagée entre lui et son voisin le nœud Nv.
- Après que tous les noeuds obtiennent les clés partagées, ils peuvent communiquer avec leurs voisins de façon sécuritaire.

#### 1.2.7.2.2. Établissement Distribué des Clés

**Avant le déploiement**

- Chaque nœud normal est pré-chargé avec une paire de clés privée / publique ECC.
- Chaque nœud leader est pré-chargé avec la même paire de clés privée / publique ECC, et une clé spécialement pour l'ajout d'un nouveau nœud.

**Après la création des clusters**

- Chaque nœud envoie la clé publique à son leader.
- Chaque nœud Nu envoie ses informations d'emplacements à son leader, Nu calcule un Code d'Authentification de Message (MAC) sur le message en utilisant la clé privée de Nu et l'ajoute au message.

- Quand le leader reçoit le message, il peut vérifier le MAC et authentifier ensuite l'identifient de Nu, en utilisant la clé publique de Nu. Alors le leader produit un certificat pour la clé publique de Nu en utilisant sa clé privée.
    - Un certificat clé public prouve l'authenticité d'une clé publique et prouve plus loin l'identité d'un capteur à un autre capteur.
- Après la détermination de l'arborescence du cheminement dans un cluster, le leader dissémine l'arborescence et le certificat clé public correspondant à chaque nœud. Les certificats des clés publics sont signés par la clé privée du leader et peuvent être vérifiés par chaque nœud.

**L'échange des clés**

- Si deux nœuds sont le parent et l'enfant dans l'arbre de cheminement, alors ils sont des voisins l'un de l'autre et ils fonderont une clé partagée pour eux.
- Pour chaque paire de voisins, le nœud avec le plus petit ID introduit le processus d'établissement des clés.
    - Par exemple, soit les nœuds Nu et Nv voisins directs et Nu a un ID plus petit que Nv, le processus est présenté ci-dessous :
        - Nu envoie sa clé publique à Nv.
        - Nv envoie sa clé publique à Nu.
        - Nu produit la clé partagée en multipliant sa clé privée avec la clé publique de Nv.
        - Simultanément Nv produit la clé partagé.

**La révocation des clés**

Quand un nœud compromis est détecté selon un certain protocole, il est annoncé au leader du cluster, le leader diffuse un message de révocation contenant l'identité du nœud compromis. En recevant le message de révocation, chaque nœud dans le cluster vérifie

s'il communique avec le nœud compromis. Si tel est le cas, alors il révoque les clés partagées avec ce dernier.

Les messages d'échange des clés publiques entre les voisins ne sont pas chiffrés, cela n'empêche pas un adversaire d'écouter ces messages. Aussi une fois, un nœud compromis, les nœuds voisins qui ont une liaison avec lui, révoquent seulement sa clé de leur mémoire et il n y'a pas un processus de mise à jour des clés.

## 1.3. Conclusion

Les réseaux de capteurs sans fil ont beaucoup d'applications informatiques civiles ou militaires, qui nécessitent un niveau de sécurité très élevé. En effet, la sécurité est basée sur la gestion des clés, qui est difficiles à implanter à cause de la nature des réseaux de capteurs sans fil et surtout quand les capteurs sont mobiles.

Dans ce chapitre nous avons présenté différentes approches de gestion des clés de cryptage pour les réseaux de capteur sans fil. Nous avons répartie ces approches en plusieurs catégories et nous avons présenté les avantages et les inconvénients de chacune.

Nous avons pu constater que les approches basées sur le protocole probabiliste sont relativement simple, mais elles demandent une grande mémoire de stockage pour les clés et elles sont moins résistantes. L'approche basée sur l'arbre M-arry n'est pas robuste mais le nombre de messages échangé pour installer les clés est d'ordre logarithmique. Aussi dans l'approche basée sur l'arbre AVL le nombre de messages échangé pour installer les clés est d'ordre logarithmique, mais comme l'approche *Micro public Key*, elle exige un grand nombre de mises à jour. Avec l'approche SecCOSEN, il n'est pas possible d'établir des liens sécurisés à l'ajout de nouveaux nœuds dans le réseau. Enfin,

les approches basées sur les courbes elliptiques offrent un niveau de sécurité plus élevé, mais elles demandent plus de calculs et d'échanges des paquets entre les nœuds.

Alors quelle est la meilleure façon de réaliser une communication sure entre les nœuds en respectant les conditions suivantes?

- Un temps minimal de calcul.
- Un stockage minimal des clés.
- Un système de cryptographie résistant.
- Un établissement de liens sécurisés.
- Réduction du nombre de mise-à-jour des clés.

Pour le but de réaliser une bonne protection de données en respectant les conditions précédentes, nous avons proposé une méthode sure et légère qui combine des approches déjà existantes en tirant les avantages de ces approches, ces approches sont : une gestion de clés basée sur un arbre AVL [9] et Une Cryptographie de Courbe Elliptique basée sur le Protocol de routage pour Réseaux de Capteurs Hétérogènes [13].
Les avantages tirés de ces approches sont les suivant :

- Les clés basées sur une Cryptographie de Courbe Elliptique réalisent une économie de stockage significative et un niveau de sécurité plus élevé.
- Dans les réseaux de capteurs sans fil hétérogènes, l'énergie consommée est moindre que l'énergie consommée dans le réseau de capteurs homogène et le taux de livraison dans les réseaux hétérogènes est meilleur que celui dans les réseaux homogènes.
- Les gestions de clés basées sur un arbre sont plus avantageuses dans la distribution des clés, et le nombre de messages échangé est d'ordre logarithmique.
- En utilisant serveur KDC comme dans [13], pour calculer les clés basé sur les courbes elliptique, le réseau gagne plus de temps et conserve plus d'énergie.

Notre but essentiel est d'atteindre des résultats fiables à travers des simulations utilisant une approche qui combine les avantages des méthodes précédentes, et faire des scénarios proches de la réalité.

Dans le chapitre 5, nous allons décrire notre méthode qui comprend deux approches, approches basées sur la combinaison et l'amélioration de deux méthodes déjà proposées par la communauté scientifique, ces méthodes sont : 1-une gestion de clés basé sur un arbre binaire de recherche AVL, 2- une cryptographie basé sur les courbes elliptiques.

Le but de notre travaille est de trouver une méthode moins coûteuse, au niveau de la consommation d'énergie et au niveau du stockage de clés, et au même temps offrir une sécurité plus élevé.

Pour montrer l'efficacité de notre méthode, nous avons choisi deux méthodes plus récentes pour faire la comparaison, notre méthode et ces deux méthodes partagent la même configuration du réseau et le même système de cryptage (cryptographie basée sur les courbes elliptiques).

Mais avant d'entrer dans les détails de notre méthode, dans le chapitre 4, nous allons présenter les courbes elliptiques et leurs applications dans la sécurité des communications, en particulier, la cryptographie, et nous allons aussi parler de l'arbre binaire de recherche AVL.

Dans le chapitre suivant nous allons faire une présentation du réseau de capteurs sans fil, leurs applications et leurs contraintes, et les problématiques liées à ces contraintes et ces applications.

# CHAPITRE 2
# LE RÉSEAU DE CAPTEURS SANS FIL

## 2.1. Introduction

Avec l'évolution de la technologie, il est devenue important observer et contrôler des phénomènes physiques (tels que la température, la pression..) pour de nombreuses applications industrielles et scientifiques, ce besoin permette d'assurer la liaison homme – machine – environnement.

Les capteurs sont des composants qui permettent de capter et de prélever les informations et les données d'entourage et de mesurer les effets des phénomènes de toutes natures qui agissent sur l'environnement de l'homme.

Il n'y a pas si longtemps, la seule solution pour acheminer les données des capteurs jusqu'au contrôleur central était le câblage qui avait comme principaux défauts d'être coûteux et encombrant.

Mais ces derniers années, et grâce aux récents progrès des technologies sans fil, les infrastructures ne sont plus reliées par des câbles. Les supports de transmission sont généralement des ondes (radios, infra rouges, ...).

## 2.2. Description

Un réseau de capteurs sans fil est un type spécial de réseau Ad Hoc.

Les réseaux Ad Hoc sont des réseaux sans fil capables de s'organiser sans infrastructure définie préalablement.

Le réseau est composé d'un ensemble de nœuds capteurs. Les nœuds comportent un grand nombre de micro-capteurs.

Chaque capteur a la possibilité de capter et de transmettre des données de façon autonome.

Le champ de captage est une zone géographique où les nœuds sont dispersés.

Les données captées sont transférées à un nœud puits (*sink*) par un routage multi-sauts.

Le *sink* est connecté à l'utilisateur du réseau via une connexion Internet ou satellitaire.

## 2.3. Applications

Les réseaux de capteurs sans fil sont utilisés par plusieurs applications dans différents domaine dont :

- Domaine militaire: surveillance des activités des ennemies.
- Domaine civile :
  a) Sécurité des bâtiments (vieillissement, incendie, intrusion) et protection des barrages.
  b) Contrôle des machines
  c) Contrôle médical
  d) Contrôle des animaux/plants

## 2.4. Les contraintes de conception

### 2.4.1. La tolérance aux fautes

Si un ou plusieurs capteurs ont des problèmes, ces derniers ne doivent pas affecter le reste du réseau. Le réseau doit capable de fonctionner en cas de problème.

### 2.4.2. L'échelle

Les capteurs déployés peuvent atteindre un nombre très important. Le puits "*sink*" doit être équipé d'une grande capacité de mémoire pour stocker les informations reçues.

### 2.4.3. Le coût

Le prix d'un capteur ne dépasse pas 1$. Par contre le prix d'un capteur Bluetooth revient à environ 10$.

### 2.4.4. L'environnement

Les capteurs peuvent être déployés en masse dans n'importe quel endroit.

### 2.4.5. La topologie de réseau

Le déploiement d'un grand nombre de nœuds nécessite une maintenance de la topologie. Cette maintenance consiste en trois phases : déploiement, post-déploiement (les nœuds peuvent bouger, ne plus fonctionner,...), et redéploiement des nœuds additionnels

### 2.4.6. Les contraintes matérielles

1. **La taille du capteur :** C'est la principale contrainte.
2. **La consommation d'énergie :** Le réseau doit survivre le plus longtemps possible.
3. **La résistance :** Adaptation aux différents environnements.
4. **Les médias de transmission :** dans les réseaux de capteurs sans fils les supports de transmission généralement sont des ondes radio, infrarouge, bluetooth et des communications radio ZigBee.

5. **L'importance d'énergie :** Il y a des cas ou le changement de la batterie d'un capteur est impossible, alors la durée de vie de ces capteurs est reliée à la durée de vie de celle-ci.

Le disfonctionnement d'un capteur nécessite une modification de la topologie du réseau et un ré-routage des paquets.

Toutes ces tâches consomment l'énergie.

*C'est pour cette raison que les recherches se concentrent sur les méthodes pour diminuer cette consommation.*

## 2.5. La Communication

### 2.5.1. Communications multi-sauts

Le nœud peut seulement communiquer avec les voisins directs dans le réseau.

Si un nœud a besoin de communiquer avec les nœuds qui se trouvent au-delà portée radio du nœud, il doit transmettre les données à travers les nœuds intermédiaires.

### 2.5.2. Capacités de communication

La largeur de la bande de communication est étroite et changeante, la portée de cette bande ne dépasse pas quelques centaines de mètres.

Les nœuds sont toujours sous l'influence de l'impact de l'environnement (les montagnes, les constructions, les tempêtes, les obstacles de terrain).

Fréquemment, les nœuds sont débranchés, alors il y a d'échec de la communication, et cela cause la perte des données. Ce qui faut que, le logiciel et le matériel des nœuds doivent être robustes.

## 2.6. L'auto-organisation

Avant l'installation du réseau de capteurs sans fil, les nœuds n'ont pas une connaissance antérieure de leurs positions dans le réseau. Après le déploiement, et avec une collaboration, les nœuds peuvent s'ajuster, exécuter et distribuer l'algorithme de déploiement. Ils peuvent rapidement et automatiquement former le réseau, sans le besoin d'un centre de contrôle. Chaque nœud a un statut identique.

## 2.7. La sécurité

Le réseau doit fonctionner correctement et efficacement sans risque.

### 2.7.1. Authenticité de données

L'authentification des données exige du destinataire du message de vérifier l'identité de la source du message pour s'assurer de l'exactitude des données d'origine.

L'émetteur doit prouver son identité.

**But :** Assurer l'authenticité des données, ainsi que la protection contre la contrefaçon ou l'usurpation d'identité et empêcher l'injection de faux messages.

### 2.7.2. Confidentialité de données

La confidentialité des données couvre les données échangées pendant le traitement et le transfère.

Il est nécessaire que la communication entre le sink et les nœuds soit privée et que le récepteur prévu des données l'assurance que les données ne sont pas modifiées pendant la transmission.

**But :** Assurer la confidentialité de données et la protection contre l'écoute clandestine.

### 2.7.3. Intégrité des données

Exige que les données ne soient pas modifiées ou détruites.

Elle couvre les données au moment du stockage, du traitement et le transfère.

### 2.7.4. Fraîcheur de données

Exige que les messages soient courants, et ne pas reproduire les messages transmis précédemment.

## 2.8. Les Attaques dans les réseaux de capteurs sans fil

Il existe plusieurs types d'attaques dont :

### 2.8.1. Subversion d'un nœud.

La capture d'un nœud peut divulguer ses informations comprenant les clés cryptographiques, ce qui compromettrait le réseau entier.

### 2.8.2. Faux nœud

L'addition d'un nœud malveillant par un adversaire pour injecter des données malveillantes, le faux nœud peut envoyer ces données à travers les nœuds réseau, et aussi il peut recevoir les messages envoyés par les nœuds du réseau.

### 2.8.3. Défaut de fonctionnement d'un nœud

Un nœud avec un défaut de fonctionnement produira des données imprécises qui compromettraient l'intégrité du réseau, particulièrement quand ce nœud agrège les données avec un autre nœud.

### 2.8.4. Panne d'un nœud

Que se produit-il quand un nœud leader cesse de fonctionner? Les protocoles du réseau devraient être assez robustes pour atténuer les effets des pannes des nœuds en fournissant un itinéraire alternatif.

### 2.8.5. Corruption de message

Quand le contenu d'un message est modifié par un attaquant, il compromet l'intégrité du message.

### 2.8.6. Boucles de cheminement.

Un attaquant peut changer et rejouer l'information de cheminement. Les boucles de cheminement attirent ou repoussent le trafic du réseau et augmentent la latence nœud-à-nœud.

### 2.8.7. Expédition sélectif

L'expédition sélective est une manière d'influencer le trafic du réseau en faisant croire que tous les nœuds participants dans le réseau sont fiables pour expédier le message. Dans l'attaque d'expédition sélective, les nœuds malveillants laissent tomber certains messages au lieu de les expédier, et ils réduisent la latence et trompent les nœuds voisins qui sont sur un chemin plus court.

## 2.9. La cryptographie dans les réseaux de capteurs sans fil

Pour réaliser la sécurité dans n'importe quel modèle de communication, il est important de chiffrer les messages transmis aux nœuds selon un arrangement de gestion de clés convenues.

Dans les réseaux traditionnels, les méthodes employées dans la gestion des clés sont complexes. Cependant, fournir une gestion des clés sures dans des réseaux de capteurs est difficile dues à la topologie et aux limitations du réseau de capteurs sans fil et à la dynamique des ressources.

Le problème principal est la distribution des clés secrètes entre les parties communicantes pour fournir des propriétés de sécurité telle que le secret et l'authentification.

Pour amorcer la sécurité dans des réseaux de capteurs sans fil, les nœuds doivent pouvoir établir une communication de nœud-à-nœud comme suit :

- Les nœuds légitimes additionnels déployés plus tard devraient pouvoir former un raccordement assuré avec des nœuds déjà-déployés.
- Les nœuds non autorisés ne devraient pas pouvoir gagner l'entrée du réseau, par l'injection de paquet ou par usurpation d'identité en tant que nœud légitime.
- L'approche doit fonctionner sans connaissance antérieure de la topologie du réseau, c.-à-d. les nœuds ne peuvent pas savoir leurs positions et leurs voisins qu'après le déploiement.
- Le stockage des clés devrait être bas et la méthode devrait être robuste pour contrer les attaques externes.

## 2.10. Conclusion

Les réseaux de capteurs sans fil deviennent plus en plus un choix plus intéressant à cause de la facilité du déploiement et la mobilité des capteurs, la simultanéité de l'information et le coût réduit d'installation.

Mais malheureusement, les capteurs sont assez fragiles et vulnérables à diverses formes de défaillances (faible énergie, panne, attaque d'adversaire, ... etc.), ce qui rend le réseau entier vulnérable et fragile.

Le but général d'un réseau de capteurs sans fil est la collecte d'un ensemble de données environnementales (telles que la température, la pression..), afin de les acheminer vers des centres de traitement.

Pour protéger la communication, il est important d'implanter un système de chiffrage-déchiffrage des messages en utilisant des clés secrètes.

Dans le chapitre suivant nous allons présenter la cryptographie basée sur les courbes elliptique, puis nous allons parler des arbres binaires de recherche AVL, qui sera utilisé dans notre approche de gestion de clés.

# CHAPITRE 3
# LA CRYPTOGRAPHIE BASÉE SUR LES COURBES ELLIPTIQUES ET LES ARBRES AVL

## 3.1. Introduction

Dans le deuxième chapitre nous avons présenté différentes méthodes de distribution de clés de cryptage, et nous avons montré que pour réaliser la sécurité dans les réseaux de capteurs sans fil, il est important de chiffrer les messages transmis aux nœuds selon une méthode de gestion de clés convenues.

On a constaté que les systèmes de cryptage basé sur les courbes elliptique offre un niveau de sécurité plus élevé et au même temps un stockage de clés plus significatif.

Et, on a remarqué que les gestions de clés basées sur un arbre sont plus avantageuses dans la distribution des clés, et le nombre de messages échangés est d'ordre logarithmique.

Pour le but de mieux comprendre notre méthode, et avant d'entrer dans les détails de notre méthode, nous avons présenté et expliqué les deux piliers de notre méthode, qui sont l'arbre AVL et la cryptographie basé sur les courbes elliptique.

## 3.2. La cryptographie

L'origine du mot cryptographie provient du grec kryptós (cache) et gráfein (écrire) [28]. On peut définir la cryptographie comme l'ensemble des techniques permettant de protéger une communication et assurer que l'information contenue dans un message ne soit révélée qu'au seul destinataire de ce message.

La cryptographie permet de correspondre de manière sécuritaire en utilisant des canaux non sécurisés. Pour cela, une fonction de chiffrement est appliquée au message à

transmettre, le résultat de ce chiffrement est appelé texte chiffré. Ce dernier pourra être transmis à l'autre entité qui possède le moyen de déchiffrement du texte chiffré afin d'obtenir un texte clair. Si le texte chiffré tombe entre les mains de l'adversaire, aucune information ne pourra être dévoilée. Pour cela, les fonctions de chiffrement et de déchiffrement doivent demeurer secrètes.

Pour des raisons pratiques, ces fonctions sont décomposées en algorithmes paramétrés par une clé. Le fonctionnement de ces algorithmes est public et seule la valeur de la clé est tenue secrète.

La cryptographie consiste à mettre au point des systèmes cryptographiques afin d'assurer la confidentialité, l'authentification et l'intégrité.

La cryptographie est divisée en deux sous-groupes :

- La cryptographie symétrique.
- La cryptographie asymétrique.

## 3.3. La cryptographie symétrique

Un sous-groupe repose sur le fait que deux entités partagent un même secret, ce qui leur permettra une communication sécuritaire.

## 3.4. La cryptographie asymétrique

Elle permet à n'importe quel individu de chiffrer un message et seulement un unique individu pourra le déchiffrer.

Le concept de la cryptographie asymétrique, est aussi appelé cryptographie à clé publique. Dans un tel système, la clé de chiffrement peut être largement diffusée et la clé de déchiffrement est secrète.

## 3.5. La cryptographie basée sur les courbes elliptiques ECC

En cryptographie, les courbes elliptiques sont des objets mathématiques, elles peuvent être utilisées pour des opérations asymétriques comme des échanges de clés sur un canal non-sécurisé ou un chiffrement asymétrique.

Les clés employées pour un chiffrement par courbe elliptique sont plus courtes qu'avec un système fondé sur le problème de la factorisation comme La cryptographie à clé publique du RSA [37]. Cela signifie des calculs plus rapides et une consommation d'énergie plus faible ainsi que des économies de mémoire et de bande passante. De plus, l'ECC procure un niveau de sécurité équivalent ou supérieur aux autres méthodes. Un autre attrait de l'ECC est qu'un opérateur bilinéaire peut être défini entre les groupes. Les opérateurs bilinéaires se sont, récemment, vus appliquer de nombreuses façons en cryptographie, par exemple pour le chiffrement basé sur l'identité.

La résistance d'un système fondé sur les courbes elliptiques repose sur le problème du logarithme discret dans le groupe correspondant à la courbe elliptique. Les développements théoriques sur les courbes étant relativement récents, la cryptographie avec courbe elliptique n'est pas très connue et souffre d'un grand nombre de brevets qui empêchent son développement. [32]

Dernièrement, ECC a attiré beaucoup d'attention comme une solution de sécurité pour les réseaux de capteurs sans fil en raison de la petite taille de la clé ECC, par exemple une clé ECC de 160 bits offre une sécurité comparable à une clé RSA de 1024 bits.

Dans [26], les auteurs ont pu démontrer que tous les protocoles cryptographiques classiques existants peuvent être adaptés sur les courbes elliptiques.

Le tableau 3.1 présente une comparaison de la taille des clés entre RSA et ECC en assurant le même niveau de sécurité.

| Niveau-Sécurité | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Taille-Clé-ECC | 112 | 160 | 224 | 256 | 384 | 512 |
| Taille-Clé-RSA | 512 | 1024 | 2048 | 3072 | 7680 | 15360 |

**Tableau 3.1. Comparaison de la taille des clés entre RSA et ECC.**

## 3.6. Les Courbes Elliptiques

Dans ce qui suit, nous présentons une brève introduction de la théorie des courbes elliptiques.

### 3.6.1. Rappel sur les groupes

Soit (G,+) un groupe fini de cardinal N et dont l'unité pour l'addition est 0.

Soit x un élément de G, on définit la suite v0=0, vn+1=vn+x.

Alors il existe k>0 avec vk=0.

La plus petite valeur pour laquelle vk=0 s'appelle l'ordre de x.

L'ordre de x divise N (théorème de Lagrange).

### 3.6.2. Corps de Galois

Pour un entier premier p et n un entier supérieur ou égal à 1, on appelle GF(pn) le corps de Galois à pn éléments.

Un élément de cet ensemble est représenté comme un n-uplet de p éléments de

Z/pZ: e=(e1,…,en).

Cet ensemble est muni d'une addition et d'une multiplication. En fait, c'est un corps: tout élément non nul est inversible.

L'addition est simple: c'est l'addition composantes à composantes dans Z/pZ.

La multiplication est plus compliquée, mais plus simple qu'une multiplication modulo un nombre de la taille de pn.

Pour p et n fixé, GF(pn) est unique à un isomorphisme près.

### 3.6.3. Définition des courbes elliptique

Soit : K un corps fini.

Le plan projectif P2(K) est l'ensemble des points P (a, b, c) $\neq$ (0, 0, 0)$\in K^3$.

Une courbe elliptique est une cubique irréductible non singulière de P2(K), qui possède un point O l'élément neutre.

Une courbe elliptique est une courbe birationnellement équivalente aux points de l'équation de Weierstrass :

$$Y^2+ a_1xy+ a_3y= x^3+a_2x^2+ a_4x + a_6 \qquad (1)$$

Les courbes elliptiques ECC comptent sur la difficulté du problème de logarithme discret, c'est-à-dire, étant donné les points P et Q, il est difficile de trouver un nombre K tel que Q = KP.

### 3.6.4. Courbes elliptiques sur GF(2n)

Soit les paramètres a et b dans GF(2n) avec b non nul ( GF corps de Galoi) . La courbe elliptique E associée est l'ensemble des points (x,y) dans (GF(2n))2 tels que $y^2+xy=x^3+ax+b$ auquel on adjoint un point spécial O appelé point à l'infini.

La définition est différente pour p>2.

### 3.6.5. Courbes elliptiques dans R2

Une courbe elliptique dans $R^2$ prend la forme suivante :

$$y^2=x^3+ax+b. \qquad (2)$$

La figure 3.1 présente le graphe d'une courbe elliptique dans R2.

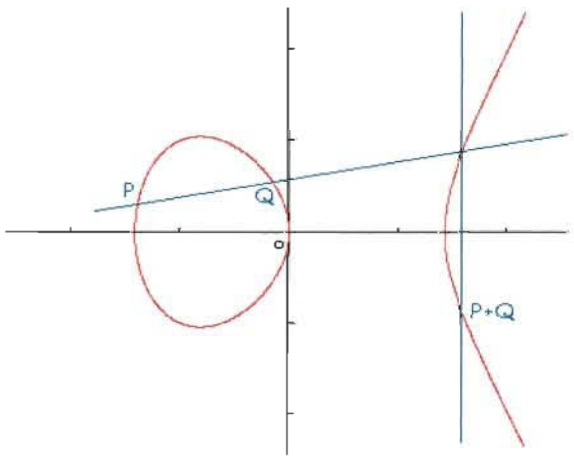



**Figure 3.1. Une courbe elliptique et opération sur cette courbe.**

### 3.6.6. L'addition de deux points

Soit les deux points P ($p_1$, $p_2$) et Q ($q_1$, $q_2$), $p_1 \neq q_1$.

La somme de deux points P et Q d'une courbe elliptique (comme dans la figure 3.I) est ($p_1$, $p_2$) régie les propriétés suivantes :

($p_1$, $p_2$)+ ($q_1$, $q_2$) = ($r_1$, r2)

$$R = (r_1, r_2)\begin{cases} r_1 = -a_2 + \lambda^2 - a_1\lambda - q_1 \\ r_2 = -(\lambda r_1 + \gamma) - a_1 r_1 - a_3 \end{cases}$$

Avec

$$\lambda = \begin{cases} \dfrac{q_2 - p_2}{q_1 - p_1} \; si \; q_1 \neq p_1 \\ \dfrac{3p_1^2 + 2a_2p_2 + a_4 - a_1p_2}{2p_2 + a_1p_1 + a_3} \; si \; q_1 = p_1 \end{cases} \qquad et \; \gamma = p_2 - \lambda p_1$$

Si $q_1$=$p_1$ et si $q_2$= $p_2$-$a_1p_1$-$a_3$, alors R= O, l'élément neutre pour l'opération addition.

On remarque que l'opération "+" possède toutes les propriétés de l'addition des nombres, on peut faire tous les calculs de type addition, soustraction et division.

### 3.6.7. La multiplication

La multiplication scalaire d'un point P et un entier k est la suivante :

$$K * P = P + P + \ldots + P \text{ (K fois).} \qquad (3)$$

### 3.6.8. Choix d'une courbe elliptique

Soit E une courbe elliptique définie par l'équation :

$$y^2+xy=x3+ax+b. \qquad (4)$$

Si on choisit aléatoirement a et b, pour choisir un point sur E, il faut résoudre une équation du troisième degré sur GF(2n). C'est un problème difficile.

En fait, on choisit un point quelconque x0, y0 et on choisit a. b a alors une solution unique que l'on peut calculer facilement.

Si on doit choisir plusieurs points aléatoirement sur la courbe, on calculera N(x0,y0) avec n très grand nombre aléatoire. En fait N>>2n. [33]

### 3.6.9. Échange de clés Diffie-Hellman

Soient Alice et Bob deux correspondants, Alice et Bob se mettent d'accord (publiquement) sur une courbe elliptique E(a,b,p), c'est-à-dire qu'ils choisissent une courbe elliptique $(\mathbf{y^2 = x^3 + ax + b \bmod p})$

Ils se mettent aussi d'accord (toujours publiquement) sur un point P situé sur la courbe.

Secrètement, Alice choisit un entier dA, et Bob un entier dB.

Alice envoie à Bob le point dAP, et Bob envoie à Alice dBP.

Chacun de leur côté, ils sont capables de calculer dA(dBP) = (dAdB)P qui est un point de la courbe, et qui constitue leur clé secrète commune.

Si Eve a espionné leurs échanges, elle connaît E(a,b,p),P,dAP,dBP.

Pour pouvoir calculer dAdBP, il faut pouvoir calculer dA connaissant P et dAP.

C'est ce que l'on appelle résoudre le logarithme discret sur une courbe elliptique.

## 3.7. Les arbres AVL

L'Arbre AVL est un arbre de recherche binaire équilibré. Pour chaque nœud de l'arbre, la différence de hauteur de ses sous-arbres est au maximum égale à 1 [34].

L'intérêt d'un arbre AVL se situe dans la recherche. En effet, puisque l'arbre est équilibré, on évite d'avoir des situations où la recherche est spécialement longue.

Pour insérer un nœud dans un arbre AVL, l'opération prend à la plupart une rotation (rotation simple ou double) pour rééquilibrer l'arbre.

Pour supprimer un nœud, l'opération prend à la plupart le log(n) des rotations pour rééquilibrer l'arbre.

Pour chercher dans un arbre équilibré, il faut conserver l'équilibre lors des ajouts et/ou des suppressions.

La figure 3.2 montre à gauche un arbre non équilibré et à droite le même arbre après un équilibrage.

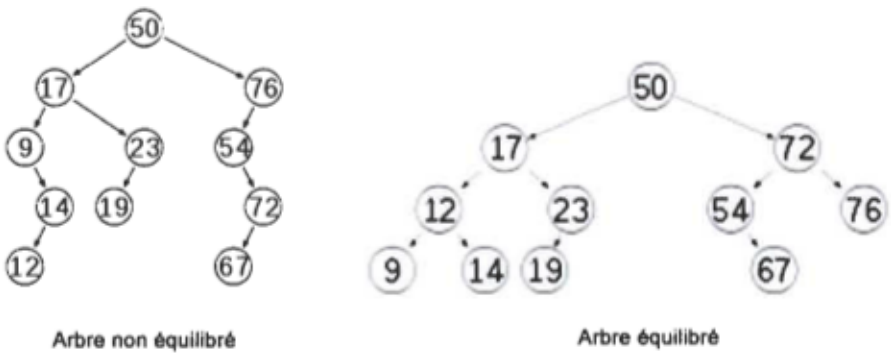



**Figure 3.2. Équilibrage d'un arbre non-AVL.**

### 3.7.1. Pourquoi les arbres AVL?

Si les procédures d'insertion et de suppression permettaient de garder un arbre équilibré, la hauteur de l'arbre serait minimisée et le temps de calcul serait également réduit.

Les Arbres Binaires de recherche permettent d'effectuer des opérations d'insertion, de suppression et de recherche en un temps proportionnel à la hauteur de l'arbre qui, lui-même, si on fait attention, est presque proportionnel à log(n).

Dans [25], l'auteur a démontré que l'utilisation d'une opération globale réduit le nombre de mises à jour des clés.

Depuis, pour chaque changement d'adhésion, les clés qui sont le long du chemin du membre affecté à la racine doivent être changées, si plusieurs demandes sont traitées ensemble, certains des chemins peuvent partiellement se chevaucher, les clés appartenant à la partie du chevauchement doivent être mise à jour seulement une fois dans chaque opération globale.

Quand tous les membres dans un sous-arbre exigent le ré-verrouillage dans une opération globale, les clés le long du chemin de la racine de ce sous-arbre à la racine de l'arbre des clés doivent être changées.

### 3.7.2. Équilibrage d'un arbre binaire AVL

Suite à un ajout d'un nœud, on remonte du nouveau nœud jusqu'à la racine de l'arbre en calculant la différence de profondeur des sous-arbres de chacun des nœuds rencontrés. Si cette différence est égale à deux ou moins deux, on rééquilibre avec la bonne rotation.

### 3.7.3. Conclusion

Les courbes elliptiques peuvent être vues comme un sous-ensemble d'un corps fini. Les calculs dans ce sous ensemble étant plus complexes que sur le corps fini lui-même, un meilleur niveau de sécurité est plus rapidement atteint (en terme de taille de clé).

La cryptographie utilisant les courbes elliptiques permet de réduire, notamment, la taille des clés et des données échangées, ainsi que la puissance de calcul nécessaire.

Un arbre AVL, est un arbre binaire de recherche qui offre une meilleure garantie sur le temps d'insertion, de suppression et de recherche dans les cas défavorables. Ceci permet l'utiliser dans des applications en temps réel.

L'utilisation de l'approche de gestion de clé basée sur un arbre AVL nous permet de réduire énormément la quantité de clés stockées ainsi limite les envois des clés à ceux qui en ont besoin, soit environ 2 log(n) clés.

Dans le chapitre suivant, nous décrivons notre méthode de gestion de la sécurité dans les réseaux de capteurs sans fil, l'architecture du réseau et la méthodologie suivie, et la gestion des clés procurées.

# CHAPITRE 4

# LA MÉTHODE PROPOSÉE

## 4.1. Introduction

Comme nous avons vue dans le deuxième chapitre, dans [12], la méthode CECKM a implanté les clés basées sur les courbes elliptiques, ce qui assure un niveau de sécurité très élevé mais l'inconvénient est que le nombre de messages échangés est énorme ainsi le nombre de clés stockées, et dans cette méthode chaque nœud dois générer sa paire de clés ECC, et cette opération demande beaucoup de calculs, et par conséquence plus d'énergie consommée.

Pour éviter le calcule des clés ECC, dans [13], la méthode utilise un serveur KDC, et les clés ECC sont pré-chargées dans les nœuds leaders, et pour réduire le nombre de clés stockés, chaque nœud ne stocke que les clés utilisés pour la communication avec ces voisins, alors pour cela il faut calculer l'arbre de routage avant l'installation des clés, et cela demande un grand nombre d'échange de paquets, et toutes ces échanges ne sont pas protégés, ce qui met le réseau en péril.

Une autre méthode qui nous a attiré l'attention est Gestion des clés dynamique utilisant un arbre AVL [9], les gestions basées sur des arbres binaires échangent moins de paquets pour installer les clés, mais dans cette approche les clés n'étaient pas robustes.

Notre but est de tirer et d'exploiter et de combiner les avantages de ces méthodes, en ajoutant des améliorations et des modifications qui permettent une gestion de clés efficace et moi couteuse.

Ces avantages sont les suivants :

- Des clés ECC qui sont légères et résistantes.
- Une gestion basée sur un arbre binaire AVL, qui permet un nombre de messages échangé d'ordre logarithmique.
- Un réseau de capteurs sans fil hétérogène, qui consomme moins d'énergie, et assure un taux de livraison plus élevé.

Dans ce chapitre nous allons enter dans les détails de notre méthode, mais d'abord nous allons décrire en bref les deux différentes architectures des réseaux de capteur sans fil, et nous allons aussi expliquer en bref la gestion des clés.

## 4.2. Les réseaux de capteurs homogènes et hétérogènes

Dans les réseaux de capteurs homogènes, tous les nœuds sont identiques en termes d'énergie et de complexité du matériel.

Il est évident que les nœuds leaders des clusters sont surchargés avec des taches supplémentaires et nécessaires, comme l'agrégation des données et la coordination du protocole de routage, par conséquent, leur durée de vie expire avant celle de ses autres nœuds.

La première solution est de faire tourner le rôle du leader du cluster périodiquement sur tous les nœuds du cluster tel que proposé dans l'approche Leach (Low Energy Adaptive Clustering Hierarchy) [24]. Mais les ressources des nœuds sont limités, ils n'ont pas de matériel assez puissant et également le réseau homogène peut souffre des mauvaises performances.

Pour améliorer les performances du réseau, il est mieux de former un réseau avec un petit nombre de nœuds de haute capacité et une transmission à longue portée et d'un grand nombre de capteurs de basse capacité [6].

P. Samundiswary et autres [18], ont implémenté un réseau de capteurs sans fil hétérogènes et ils ont comparé avec d'autres réseaux de capteurs sans fil homogène. Ces réseaux de capteurs hétérogènes ont moins de sauts, pour atteindre le nœud *Sink* et ils impliquent moins de multi-sauts que les réseaux homogènes.

Les résultats démontrent que l'énergie consommée en utilisant le réseau de capteurs hétérogènes est réduite de 92,5% de moins que le réseau de capteurs homogène.

Aussi, également, le taux de livraison dans les réseaux hétérogènes est meilleur que celui dans les réseaux homogènes.

## 4.3. Gestion des clés

La gestion des clés est une procédure pour stocker et distribuer des clés cryptographiques d'une façon sûre; la génération et de la distribution des clés aux destinataires autorisés doit être aussi assurée et sécurisée.

Après le déploiement, les capteurs ont besoin d'établir des clés cryptographiques avec leurs voisins pour assurer des services de sécurité.

La gestion de clé doit fournir un établissement de clés entre tous les nœuds, et elle doit fonctionner même si la topologie du réseau n'est pas prédéfinie. Et, les nœuds non autorisés ne peuvent pas effectuer des communications avec les nœuds du réseau.

La sécurité du réseau est un défi sérieux; la gestion statique des clés n'est pas appropriée particulièrement après une longue période. Alors, nous proposons une gestion basée sur le système d'arbre AVL, parce que dans le cas de changement d'un nœud (dans l'arbre AVL), il peut causer un changement d'un sous-arbre, et les clés seront aussi changées en même temps.

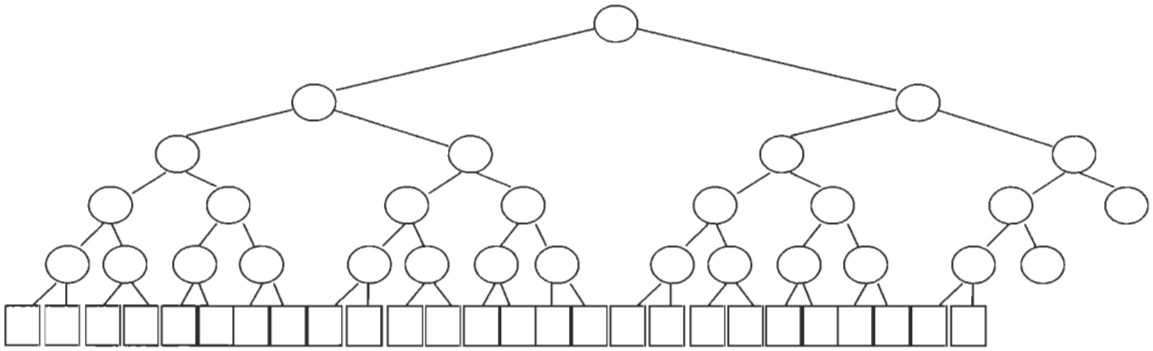


**Figure 4.1. Arbre AVL de clés.**

Dans la figure 4.1, les capteurs sont représentés par des carrés, et les clés sont représentées par des cercles.

Dans l'arbre AVL, comme la figure 4-1 le démontre, chaque à H clés, où H dénote la hauteur de l'arbre AVL, chaque feuille doit posséder toutes les clés correspondantes aux nœuds de l'arbre AVL. Les feuilles ne connaissent pas les clés dont elles n'ont pas besoin.

Le facteur d'équilibrage est -1, c.-à-d. la différence entre la hauteur d'un sous-arbre droit et un sous-arbre gauche égale à -1.

Lorsque des messages doivent être envoyés à tous les nœuds, on utilise la clé racine de l'arbre AVL car cette clé est connue de tous.

Si le message s'adresse d'un contrôleur à une feuille, la clé qui se trouve dans le niveau H et la position horizontale de la feuille peuvent être utilisées.

Si le message s'adresse d'une feuille à une feuille, la première clé commune et la plus proche aux feuilles, et les positions horizontales des feuilles peuvent être utilisée.

### 4.3.1. L'algorithme de clé commune

L'algorithme de clé commune est utilisé pour trouver la clé partagée entre deux nœuds, à condition qu'elle soit la première clé commune plus proche des feuilles, l'algorithme est le suivant :

- Soit Pu est la position de la feuille U, et Pv est la position de la feuille V, où cette position est la position horizontale de gauche à droite.
- Soit C un entier égale à zéro, C est utilisé comme un compteur.
- Soit D un entier, oû D est la valeur absolue de la différence entre Pu et Pv.
- Tant-que D<>0 faire.
    - Incrémente C.
    - Divise D par 2 et remplace D par la partie entière du résultat.
- Soit Niv un entier égale à la différence entre la hauteur de l'arbre et C
- Niv est la position (le niveau) de la clé partagée.

## 4.4. La méthode proposée

Nous avons choisi un réseau hétérogène [29] qui contient un petit nombre de capteurs de haute gamme qui agissent comme leaders. Ces capteurs sont puissants et équipés de matériel résistants [12]. Le réseau contient ainsi un grand nombre de nœuds capteurs de bas de gamme qui forment des groupes autour des leaders.

Les capteurs qui sont de haute gamme reçoivent les messages des nœuds de bas gamme dans le cluster, et les font suivre à la station de base. Dans ce qui suit, on appellera un nœud bas de gamme un nœud normal.

La méthode de gestion de clés choisie est basée sur la cryptographie des courbes elliptique, ou un serveur est utilisé pour produire les clés publiques ECC.

Avant le déploiement, chaque leader LD est pré-chargé par la même clé du réseau CR et une clé unique CLR est utilisée pour la communication entre les leaders, et chaque nœud normal ND est pré-chargé seulement avec la clé unique du réseau CR.

Après la sélection des groupes, chaque leader construit une liste de membres (liste_Membre ) se qui présente la liste des nœuds dans le cluster. En fonction du nombre de membres, il construit un arbre AVL formé des clés publiques, puis il remplace la clé du réseau par la clé racine dans l'arbre AVL et il la diffuse. Le leader envoie à chaque nœud dans le cluster un message (chiffré avec la nouvelle clé) contenant la position du nœud dans la liste des membres, puis un message contenant l'ensemble des clés qui se trouvent le long du chemin du nœud à la racine de l'arbre AVL.

La figure 4.2, présente la structure de la gestion des clés basée sur l'arbre AVL.

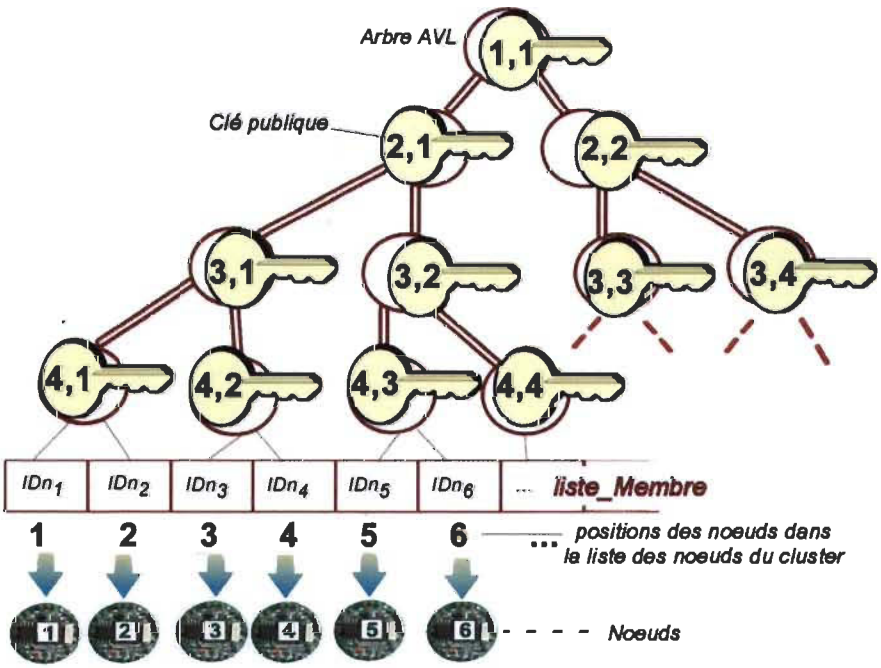



**Figure 4.2. La structure de l'arbre AVL des clés ECC et la liste des nœuds.**

Si deux nœuds veulent établir une liaison, ils s'échangent leur position dans la liste de membres, et il calcule la clé secrète qui est composée de leur position et de la clé commune qui a le niveau le plus loin de la racine dans l'arbre AVL.

Si un nœud veut établir une communication avec son leader, il utilise une clé composée de la clé racine avec sa position dans la liste des membres.

Pour les leaders, puisqu'ils sont équipés de matériels résistants, ils utilisent toujours la même clé pré-chargée avant le déploiement.

La figure 4.3 montre des exemples de calcule des clés secrètes pour établir les liaisons entre les différents nœuds.

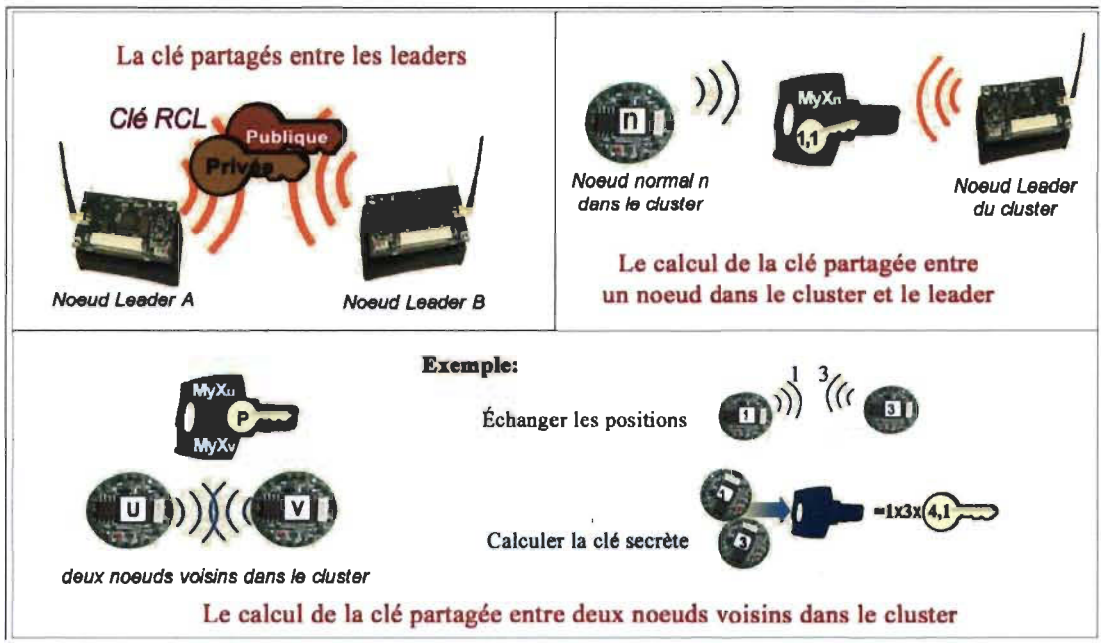



**Figure 4.3. Les clés de chiffrement utilisées par les nœuds.**

Quand un nouveau nœud demande son ajout dans le cluster, le leader le met dans la liste des membres et il rafraîchit les clés correspondantes à sa position dans la liste de l'arbre AVL, puis il renvoie à ce nœud sa position dans la liste des membres et son ensemble de clés. Aussi, il envoie un message de rafraîchissement de clés à tous les nœuds dans le cluster.

Si un nœud quitte le cluster, ou une fois que le leader détecte un nœud compromis, alors il fait la mise à jour des clés publiques qui se trouvent le long du chemin de ce nœud à la racine de l'arbre AVL et reconfigure l'arbre AVL.

## 4.5. Le scénario

Dans les pages suivantes, nous présentons les deux approches de notre méthode, les deux approches utilisent un serveur KDC pour générer les clés ECC. Ces approches sont appelées AVL_KDC et AVL_Leaders, dans AVL_KDC de control qui distribue les clés est la station de base, et dans AVL_Leaders l'unité de control et le leader du cluster.

Le but d'avoir ces deux approches est réaliser deux scénarios qui peuvent se trouver dans la réalité, et aussi on veut savoir après les expériences et les testes de la simulation, laquelle de ces approches est plus avantageuse selon les données initiales et les besoins.

### 4.5.1. La structure du réseau

Comme dans [13], c'est un réseau hétérogène qui contient un petit nombre de nœuds capteurs de haute gamme qui agissent comme leaders. Ces capteurs sont puissants et équipés de matériel résistants [12]. Le réseau contient aussi un grand nombre de nœuds capteurs de bas de gamme qui forment des groupes autour des leaders, comme c'est présenté dans la figure 4.4.

Les capteurs de haute gamme reçoivent les messages des nœuds de bas gamme dans le cluster, et les font suivre à la station de base.

Dans la suite du document, on appellera les nœuds de bas de gamme, des nœuds normaux.

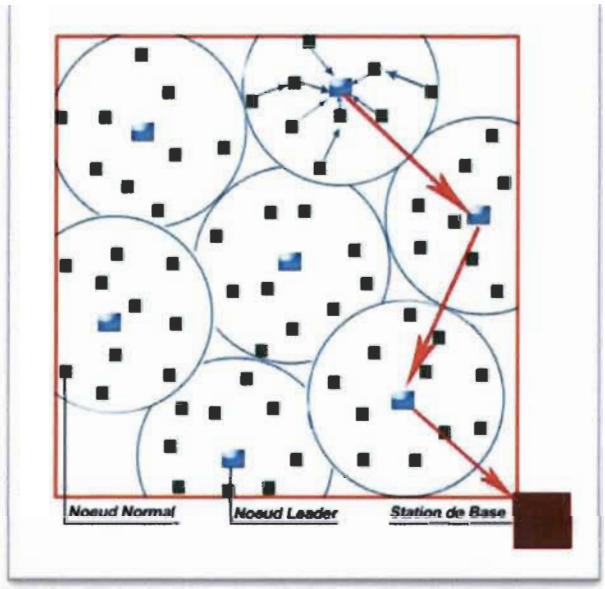



**Figure 4.4. L'architecture du réseau.**

Voici quelques définitions de sigle utilisés dans la suite du document.

LD : Leader.

P : Clé publique.

ND : Nœud de capteur normal.

CRL : Une paire de clés publique/privée, utilisée par les leaders pour communiquer entre eux.

CR: Une paire de clés publique/privée, appelée clé du réseau, utilisée pour la communication entre les capteurs normaux et les leaders.

CC : La clé du cluster.

Liste_Membre : La liste contenant les identifications des nœuds du réseau.

Ensemble_clés : Ensemble des clés publiques qui se trouvent le long du chemin allant du nœud à la racine de l'arbre AVL.

MyX : la position du nœud dans la liste_Membre.

IDu : Identificateur unique envoyé à chaque nœud.

### 4.5.2. Génération des Clés

Dans notre méthode de gestion des clés, la station de base est équipée d'un serveur KDC, qui est utilisé pour produire les clés publiques ECC et les paires de clés CRL et CR.

### 4.5.3. L'établissement des clés distribués AVL-Leaders

Dans l'établissement des clés distribués AVL-leaders, l'unité de control qui maintient l'arbre AVL est le leader du cluster, le leader du cluster prend la charge de gérer et de stocker les clés ECC dans l'arbre AVL, il associe chaque nœud dans le cluster avec sa position dans l'arbre AVL, et il prend la charge aussi de la distribution des clés ECC .

#### 4.5.3.1. L'installation des Clés

##### 4.5.3.1.1. Avant le déploiement

- Chaque leader est pré-chargé par la même paire de clés publique/privée du réseau CR et une clé unique CRL utilisée pour la communication entre les leaders dans le réseau.
- Chaque leader LD est pré-chargé avec un ensemble de clés publiques.
- Chaque Nœud ND est pré-chargé de la même clé CR, et un identificateur unique $ID_U$. Ces informations sont utilisées pour l'authentification des émissions et l'identification des messages échangés entre les capteurs.

##### 4.5.3.1.2. Formation du Groupe

Après le déploiement (comme dans [13]) :

- Chaque leader LD émet un message « join » crypté avec la clé du réseau CR, pour créer son cluster.
- Quand un nœud ND reçoit le premier message « join », il le décrypte avec la clé CR, puis il renvoie au LD un message « joined » crypté avec CR utilisant son ID.
- Après un certain temps, quand LD reçoit un message « joined » d'un nœud ND, il le décrypte avec la clé CR.
- Ensuite, chaque LD produit une « Liste_Membre » contenant les ID des membres dans le cluster.

##### 4.5.3.1.3. Après la sélection d'un groupe

- En fonction du nombre de membres dans le cluster, le leader LD construit l'arbre AVL formé des clés publiques (qui sont déjà pré-chargées).

- Le leader LD envoie à chaque nœud sa position MyX, qui est sa position réelle dans la « List_Membre », ce message est crypté avec la clé du réseau publique CR.
- Le leader LD décrypte le message avec la clé du réseau Privée CR.
- Le leader LD envoie d'abord la clé du cluster CC qui se trouve dans la racine de l'arbre AVL à tous les nœuds du cluster, ce message est crypté avec la clé CR/privée.
- Chaque nœud ND décrypte le message avec la clé du réseau CR/publique, puis il supprime la paire de clés CR publique/privée.
- Le leader LD envoie à chaque nœud dans le cluster un message «Ensemble_clés » (ce sont les clés qui se trouve le long du chemin du nœud à la racine de l'arbre AVL).

#### 4.5.3.2. Échange de clés

L'opération d'échange de clés entre chaque paire de nœuds $ND_u$, $ND_v$ dans le cluster se fait comme suit :

- Soit $X_u$=My$X_u$ et $X_v$=My$X_v$ les positions des noeuds $ND_u$ et $ND_v$ respectivement dans la « Liste_Membre ».
- Le nœud $ND_u$ envoie au nœud $ND_v$ la valeur $X_u$ et $ND_v$ envoie à $ND_u$ la valeur $X_v$.

- Les nœuds $ND_u$ et $ND_v$ cherchent le point $P_{uv}$ qui est la clé publique commune, qui se trouve au niveau le plus haut dans l'arbre AVL ( la plus proche des feuilles).
- Les nœuds $ND_u$ et $ND_v$ calculent $X_u(X_vP_{uv}) = X_v(X_uP_{uv})$ qui constitue leur clé secrète commune.

#### 4.5.3.3. Ajoût d'un nouveau nœud

Avant le déploiement, chaque nouveau nœud est pré-chargé par son identification ID.

Après le déploiement d'un nouveau nœud, il demande son ajout dans le cluster, et il fournit son ID.

Le leader LD met l'ID du nouveau nœud dans la liste des nœuds «Liste_Membre », et il rafraîchit les clés correspondant à sa position dans l'arbre AVL, puis il renvoie à ce nœud sa position dans la liste-Membre et son ensemble de clés. Aussi, il envoie un message de rafraîchissement des clés à tous les nœuds du cluster.

#### 4.5.3.4. Rafraichissement des clés

Comme dans [9], l'opération de rafraîchissement des clés est la suivante :

Si une opération d'ajout d'un nouveau nœud est réussite, le leader envoie un message de rafraîchissement des clés à tous les membres du groupe.

#### 4.5.3.5. Mise à jour des clés

Une fois que les comportements anormaux d'un nœud sont détectés par le leader, ou une fois qu'un nœud quitte le cluster, le leader fait la mise à jour des clés publiques et reconfigure l'arbre AVL. Dans cette mise à jour les clés publiques qui se trouvent le long du chemin du nœud (qui a quitté le cluster) à la racine de l'arbre AVL sont changées (Comme dans [7]).

### 4.5.4. L'établissement des clés centralisées AVL-KDC

Dans cette approche, l'unité de contrôle est la station de base, qui est équipée d'un serveur KDC, dans ce cas la station de base génère les clés ECC, elle gère l'arbre AVL

qui stocke toutes les clés du réseau, elle associe tous les nœuds normaux dans le réseau avec leur position dans l'arbre AVL.

#### 4.5.4.1. Avant le déploiement

- En fonction du nombre de nœuds normaux, le serveur construit l'arbre AVL formé des clés publiques et la « Liste_Membre ».
- Chaque leader LD est pré-chargé par la même clé CRL et la même clé CR qui est en réalité la clé qui se trouve dans la racine de l'arbre AVL.
- Chaque Nœud ND est pré-chargé de «Ensemble_clés » qui présente les clés qui se trouvent le long du chemin du nœud à la racine de l'arbre AVL et un identificateur unique $ID_U$, et sa position MyX. Ces informations sont utilisées pour l'authentification des émissions et l'identification du noeud.

#### 4.5.4.2. Après la sélection d'un groupe

- Chaque Nœud ND envoi à son leader LD un message contenant sa position MyX.
- Le leader LD associe la position MyX du Nœud avec son identificateur ID dans sa propre liste « Liste_Membre ».
- Le leader associe MyX la position du Nœud avec son identificateur ID dans sa «Liste_Membre ». Ensuite, le leader et le nœud calculent leur clé secrète MyX× CR.

Pour chaque paire de nœuds $ND_u$, $ND_v$ dans le cluster :

- Soit $X_u$=$MyX_u$ et $X_v$=$MyX_v$ les positions de $ND_u$ et $ND_v$ respectivement dans la liste « Liste_Membre ».

- Les nœuds $ND_u$ et $ND_v$ cherchent le point $P_{uv}$ qui est la clé publique commune, qui se trouve au niveau le plus haut dans l'arbre AVL.

- Le nœud $ND_u$ envoie au noeud $ND_v$ la valeur $X_u$ et $ND_v$ envoie à $ND_u$ la valeur $X_v$.
- $ND_u$ et $ND_v$ calculent $X_u(X_vP_{uv}) = X_v(X_uP_{uv})$ qui constitue leur clé secrète commune.

#### 4.5.4.3. Ajoût d'un nouveau nœud

Avant d'ajouter un nouveau nœud dans le réseau, le serveur KDC met l'ID du nouveau nœud dans la liste des nœuds, et il rafraichit les clés correspondantes à sa position dans l'arbre AVL.

Le nouveau nœud $ND_n$ est pré-chargé de son «Ensemble_clés », son identificateur $ID_n$, et de sa position $MyX_n$.

Le serveur envoie à tous les leaders l'identificateur $ID_n$ du nouveau nœud et un message de rafraichissement des clés.

Quand le nouveau nœud demande son ajout dans un cluster, il envoie au leader du cluster son identificateur $ID_n$ et sa position $MyX_n$.

Le leader LD met l'$ID_n$ et la position $Myx_n$ du nouveau nœud dans sa « Liste_Membre », puis il envoie un message de rafrîichissement des clés à tous les nœuds du cluster.

#### 4.5.4.4. Mise à jour des clés

Une fois les comportements anormaux d'un nœud sont détectés par un leader, ou une fois qu'un nœud quitte le réseau, le serveur fait la mise à jour des clés publiques qui se trouvent le long du chemin du nœud (qui a quitté le cluster) à la racine de l'arbre AVL et il reconfigure l'arbre AVL. Ensuite, il envoie un message de mises à jour à tous les leaders, et à son tour, chaque leader diffuse le message à tous les nœuds dans son cluster.

### 4.5.5. Établissement d'une communication

#### 4.5.5.1. La communication entre les nœuds

Si deux nœuds veulent établir une liaison, ils échangent leurs positions dans la liste « Liste_Membre », comme indiqué précédemment, puis pour crypter et décrypter les messages échangés entre eux, ils utilisent la clé composée de la multiplication de la clé partagée dans l'arbre AVL avec leurs positions dans la liste « Liste_Membre ».

#### 4.5.5.2. La communication entre un nœud et son leader

Si un nœud $ND_u$ veut établir une communication avec son leader, ils cryptent et décryptent les messages échangés utilisant une clé composée de la multiplication de la clé racine dans l'arbre AVL avec $MyX_u$ la position du nœud $ND_u$ dans la liste « Liste_Membre ».

#### 4.5.5.3. La communication entre les leaders

Comme nous avons indiqué précédemment, tous les leaders sont équipés de matériels résistants, et avant le déploiement du réseau, chaque leader est pré-chargé par la même paire de clés CRL. Alors ils utilisent toujours la clé CRL publique pour crypter les messages et la clé CLR privée pour décrypter les messages échangés.

## 4.6. Conclusions

Quand on étudie une méthode de gestion des clés, il est nécessaire de prendre en compte la notion de robustesse de la méthode, de la consommation d'énergie et de la taille de la mémoire utilisé.

- La robustesse est liée à la nature des clés de cryptage et la façon de distribution des clés.
- La consommation d'énergie est lié au nombre de messages échangés durant l'installation des clés, et le coût des opérations de calcul des clés secrètes partagés.
- La taille de mémoire utilisée est liée à la taille de la clé et le nombre de clés stockées.

La méthode que nous avons proposée vise à assurer une communication sûre, et réduire le nombre de transmissions entre les nœuds durant l'installation des clés.

L'utilisation d'un système basée sur les courbes elliptiques assure robustesse avec une utilisation de mémoire réduite.

En utilisant l'arbre AVL, la quantité de messages transmis entre l'unité de contrôle et les nœuds est réduite. Et en utilisant cette technique, les nœuds voisins connaissent les clés partagés, alors ils peuvent calculer leurs clés secrètes sans le besoin d'échanger ces clés partagées, et par conséquence, cela réduit le nombre d'échanges.

# CHAPITRE 5
# LA SIMULATION

## 5.1. Introduction

En général, la simulation peut fournir une façon d'étudier des choix de conception de système dans un environnement contrôlé, elle nous offre l'exploration des configurations de système qui sont presque impossibles à construire physiquement et elle nous permet l'observation des interactions qui sont difficiles dans le réel.

## 5.2. Environnement de simulation

On a choisi le système Tinyos pour réaliser notre simulation. Tinyos est un système d'exploitation *open-source* fonctionnant sous Unix. Il est adapté aux capteurs. Il supporte de nombreuses plates-formes et il fournit des concepts très importants pour réaliser les simulations.

Un simulateur Tinyos se caractérise par les conditions suivantes :

### 5.2.1. Adaptabilité

Le simulateur doit être en mesure de manipuler les grands réseaux formés de milliers de nœuds dans une large portée.

### 5.2.2. Perfection

Le simulateur doit couvrir le plus possible les interactions de système, capturant précisément le comportement du réseau dans une large portée.

### 5.2.3. Fidélité

Le simulateur doit révéler les interactions imprévues, non seulement ceux qu'un développeur suspecte.

Le simulateur doit remplir le fossé entre l'algorithme et la mise en œuvre, permettant aux développeurs de tester et vérifier le code qui fonctionnera sur le matériel réel.

Le simulateur de Tinyos est le TOSSIM, il permet de simuler le comportement d'un capteur (envoie/réception de messages via les ondes radios, traitement de l'information, ...) dans le cadre d'un réseau de capteurs sans fil.

TOSSIM fonctionne sur un PC, cela permet aux utilisateurs de déboguer, tester et analyser des algorithmes dans un environnement contrôlé et répétable.

L'objectif principal de TOSSIM est de fournir une simulation d'application Tinyos fidèle. Pour cette raison, il simule l'ensemble des applications Tinyos plutôt que de simuler le monde réel.

Comme tout simulateur, TOSSIM n'est pas toujours la solution de simulation parfaite, il est utilisée pour comprendre les causes des comportements observés dans le monde réel, mais il ne saisit pas tout, alors il ne doit pas être utilisé pour des évaluations absolues.

TOSSIM ne modélise pas le temps d'exécution du CPU, aussi il ne peut pas fournir facilement les informations précises pour calculer la consommation d'énergie du processeur; alors cette tâche sera effectuer manuellement à partir du nombre de paquets échangés durant l'installation des clés.

TOSSIM est une bibliothèque, pour cela il faut écrire un programme qui configure la simulation et l'exécute. TOSSIM prend en charge deux interfaces de programmation:

Python et C + +, le langage Python permet d'interagir avec une simulation en cours, de façon dynamique.

Le modèle de la radio de TOSSIM par défaut est une cellule unique sans erreur, tous les nœuds écoutent parfaitement.

TOSSIM utilise la plate-forme MicaZ[21], c'est un appareil avec une faible puissance, il se caractérise par :

Microcontrôleur ATMEGA 128.

Émetteur récepteur conforme de 802.15.4/ZigBee.

Mémoire RAM 4KOctets.

Mémoire ROM 128KOctets.

Mémoire Externe 512KOctets.

Programmation NESC.

## 5.3. Le but de la simulation

Nous nous intéressons dans notre simulation à la taille de la mémoire utilisée par un nœud car c'est une contrainte principale nous nous intéressons aussi à la consommation d'énergie par un nœud, vu que ce paramètre est relié à la durée de vie d'un nœud. En effet, moins les nœuds dépensent l'énergie plus la durée de vie du nœud est longue. De plus, nous portons une attention sur intéresse le nombre de paquets échangés entre les nœuds pour établir les clés, plus la communication est grande plus la possibilité qu'un adversaire puisse modifier une clé pendant son parcours, est grande.

## 5.4. Les Paramètres

Pour nos simulations, nous avons utilisé un réseau qui est formé de 600 à 1000 nœuds de type MicaZ. Parmi ces nœuds, 20 nœuds sont des nœuds leaders. Les nœuds sont répartis uniformément et aléatoirement dans une zone de 1000 × 1000 m. La portée de transmission d'un capteur est de 60 m, la taille d'un paquet est de 32 Octets, et le taux d'erreur de transmission est zéro.

Le nombre de nœud normaux est :

$Nbr_{NoeudsN} = Nbr_{Noeuds}$ -20.

Le nombre de nœuds normaux dans chaque cluster est le nombre de nœuds normaux divisé par 20.

$Nbr_{Noeuds\ Cluster} = Nbr_{NoeudsN}/20.$

Le nombre moyen de voisins est calculé comme dans [13] par la formule suivante :

$Nbr_{NoeudsN} \times \pi \times 602 / (1000 \times 1000).$

## 5.5. Les Métriques

Pour évaluer la performance de notre méthode, nous nous intéressons aux métriques suivantes:

- Le nombre total des paquets échangés dans le réseau durant l'installation des clés.
- La consommation d'énergie moyenne par un nœud normal et un nœud leader
- Le nombre de clés ECC stockées dans un nœud normal et un nœud leader.

La consommation d'énergie ne comprend que les énergies utilisées pour établir les clés de cryptage, mais ne comprend pas les énergies dépensées pour les autres communications de données, ainsi ne comprend pas l'énergie pour générer les clés dans la méthode CECKM [28].

## 5.6. Évaluation des paramètres de la méthode Routing-Driven courbe elliptique.

Comme nous avons vue dans le deuxième chapitre, Mohsen Guizani et des al. [13], ont proposé une méthode basée sur le protocole du routage GPSR [35] avant l'installation les clés ECC.

Ils ont implémenté les deux façons suivantes pour réaliser leur approche, l'une centralisée RECC-CENT et l'autre distribuer RECC-DIST.

Dans les tableaux 5.1 et 5.2 nous avons calculé le nombre de clés stockées par chaque leader et chaque nœud normal, dans toutes les deux approches de la méthode [13].

Soit:

M : le nombre de leaders dans le réseau

N : le nombre de nœuds normaux dans le réseau

NC : le nombre de nœuds normaux dans le cluster

NV : le nombre de nœuds normaux voisins

Les tableaux 5.1 et 5.2 présentent le nombre de clés ECC stockées dans chaque type de nœuds.

| Nœud | RECC-CENT | RECC-DIST |
|---|---|---|
| Leader | N+2 | 3+NC |
| Normal | 2+NV | 3+NV |

**Tableau 5.1 : Le nombre de clés ECC stockées dans la méthode Routing-Driven courbe elliptique.**

Une clé ECC sous python à une taille de 40 bytes et la taille d'un paquet sous TOSSIM est 28 bytes, alors pour transmettre une clé publique ECC, il faut diviser le message en 2 paquets. Une évaluation du nombre de paquets transmis et reçus dans le réseau, est présentée dans le tableau 5.2 :

| Nœud | Message | ECC-Cent | ECC-DIST |
|---|---|---|---|
| Leader | Envoyé | 2(NC x NV) | NC (3+NV) |
| | Reçu | NC | 3NC |
| Normal | Envoyé | 3+NV | 3+2NV |
| | Reçu | 4NV | 3(NV+1) |

**Tableau 5.2 : Le nombre de clés ECC stockées dans la méthode Routing-Driven courbe elliptique.**

## 5.7. Conclusion

Dans ce chapitre nous avons présenté le simulateur utilisé dans notre recherche, et les avantages qu'il nous apporte, comme la modélisation de notre application dans un environnement virtuel.

Nous avons donné les détails des paramètres de notre réseau et les métriques pour évaluer la performance de notre méthode.

Dans le chapitre suivant nous allons présenter les différents résultats de notre simulation, et la comparaison entre notre méthode et les méthodes CECKM et les deux approches de la méthode RECC en détail.

# CHAPITRE 6
# RÉSULTATS

## 6.1. Introduction

Dans ce chapitre, nous présentons les résultats obtenus avec les simulations effectuées.

Au vue des données obtenus à partir du système de sortie de débogage de Tossim, nous avons classé ces données dans des tableaux selon les types des nœuds et cela nous a permis de calculer la consommation moyenne d'énergie par les nœuds et tracer les courbes relatives à nos métriques, en fonction du nombre total de nœuds dans le réseau.

## 6.2. Les paramètres de simulation

Les paramètres dans les simulations effectuées varient relativement en fonction de la taille du réseau. Ils sont décrits dans le tableau 6.1 avec tous les détails:

| Taille du Réseau (Nombre de nœuds total) | 600 | 700 | 800 | 900 | 1000 |
|---|---|---|---|---|---|
| **Nombre de nœuds leaders** | 20 | 20 | 20 | 20 | 20 |
| **Nombre de nœuds normaux** | 580 | 680 | 780 | 880 | 980 |
| **Nombre de nœuds dans un cluster** | 29 | 34 | 39 | 44 | 49 |
| **Nombre de voisins** | 7 | 8 | 9 | 10 | 11 |

**Tableau 6.1. Les paramètres de la simulation.**

Les paramètres du contexte de la simulation sont présentés dans le tableau 6.2:

| | |
|---|---|
| Le modèle de radio | CC2420 |
| Fréquence de radio | 2400 à 2483.5 |
| Portée | 60m |
| Taille du paquet | 28 Octets |
| Type du paquet | AM |
| Puissance de réception | 19,7mA |
| Puissance de transmission | 17,4mA |
| Dimension du réseau | 1000 ×1000m |
| Modèle de Topologie | Aléatoire statique |

**Tableau 6.2. Les paramètres du contexte de la simulation.**

## 6.3. Les Métriques

Dans les opérations radio, la tension nécessaire est 3V [22]. À partir de cette information, nous avons calculé l'énergie dépensée comme ce qui suit:

L'énergie consommée pour recevoir un paquet est ERx=19,7mA x 3V = 59,1mW. La consommation d'énergie pour émettre un paquet est ETx =17,4mA x 3V=52,2mW.

Les équations 1 à 5 montrent les méthodes utilisées pour calculer nos métriques:

- L'énergie consommée dans le réseau est :

$$\text{(Nb Paquets Reçus} \times \text{ERx)} + \text{(Nb Paquets envoyés} \times \text{ETx)} \quad (1)$$

- La taille de la mémoire utilisée par un nœud :
  - Nœud normal : $T_N = (Nbr_{ClésStockées} \times 40 \text{ Bytes})$. (2)
  - Nœud leader: $T_L = \sum_{i=0}^{H} 2^i \times 40 \text{ bytes}$ . (3)
- La consommation moyenne d'énergie par un nœud :
  - Nœud normal:

$$E_N = \left( \frac{\sum_{i=1}^{NbrNoeudN} \left( Nbr_{Packet_{Received}} \times 59{,}1 \right) + \left( Nbr_{Packet_{Sent}} \times 52{,}2 \right)}{NbrNoeudN} \right) \text{mW} \quad (5)$$

  - Nœud leader:

$$E_L = \left( \frac{\sum_{i=1}^{20} \left( Nbr_{Packet_{Received}} \times 59{,}1 \right) + \left( Nbr_{Packet_{Sent}} \times 52{,}2 \right)}{20} \right) \text{mW} \quad (6)$$

H est la hauteur de l'arbre AVL selon le nombre de nœuds dans le cluster.

Le tableau 6.3 présente la hauteur de l'arbre AVL en fonction du nombre de nœuds dans le cluster (dans le réseau pour l'approche AVL-KDC).

| Nombre de Nœuds | [8-15] | [16-31] | [31-63] | [64-127] | [128-255] | [256-511] | [512-1023] |
|---|---|---|---|---|---|---|---|
| H : Hauteur d'AVL | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Nombre de clés | 15 | 31 | 63 | 127 | 255 | 511 | 1023 |

**Tableau 6.3. La hauteur de l'arbre AVL en fonction du nombre de nœuds.**

## 6.4. Installation des clés

### 6.4.1 Comparaison des paquets échangés

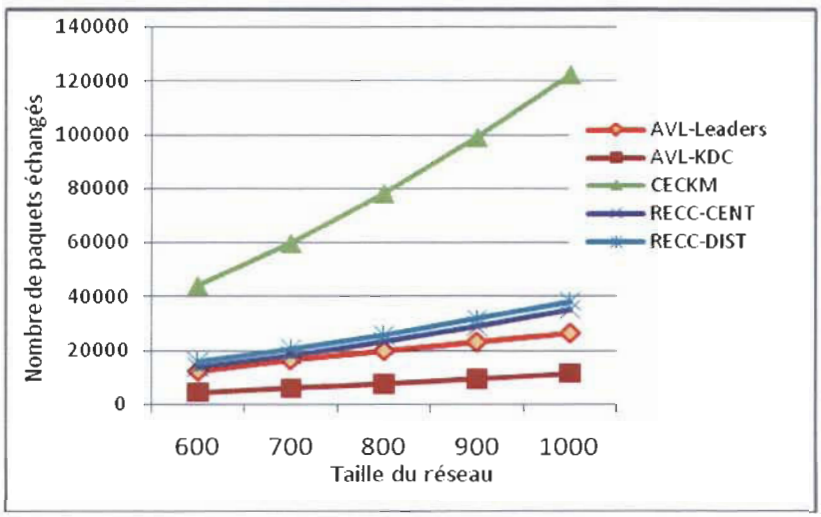



**Figure 6.1. Nombre de paquets échangés lors de l'installation des clés.**

La figure 6.1 présente le nombre de paquets échangés durant l'installation des clés de cryptage pour chaque méthode.

Les courbes montrent que notre approche AVL-KDC échange moins de paquets. Cependant la méthode CCKCM utilise un nombre de paquets très grand, 4 fois plus que notre méthode AVL-Leaders et 10 fois plus qu'AVL-KDC. Toutefois les méthodes RECC-DIST et RECC-CENT utilisent un nombre de paquets plus que notre approche mais beaucoup moins que l'approche CECKM.

Notre méthode présente une communication plus petite dû au fait qu'elle échangent moins de paquets. Elles nécessitent une transmission plus courte que les autres méthodes, Ceci garantie un temps d'installation des clés plus court, donc plus sécuritaire, puisque, pour un intrus, si la communication est de longue durée, il a plus de chance de capter et modifier une clé publique pendant l'échange.

### 6.4.2 Comparaison de l'énergie dépensée

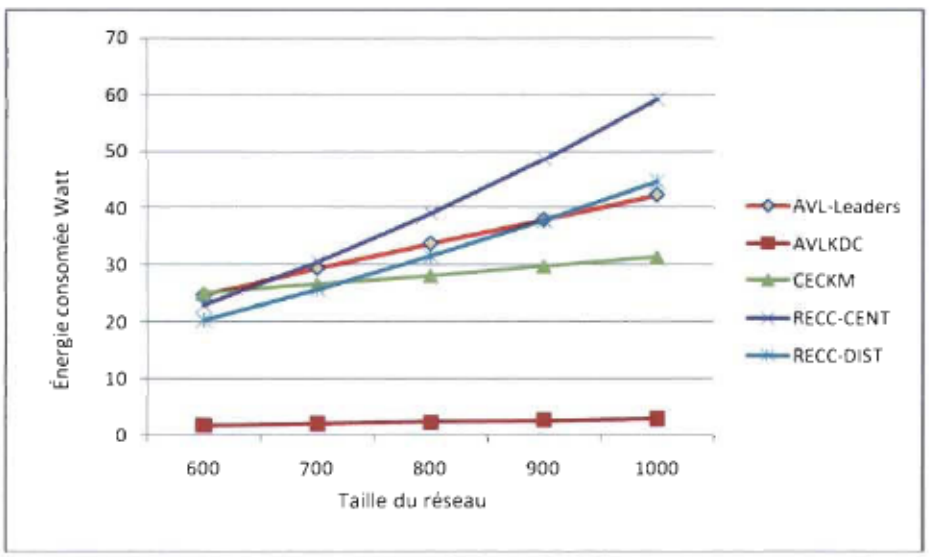



**Figure 6.2. La consommation d'énergie par un leader.**

La figure 6.2 montre la consommation d'énergie par un leader dans le réseau durant l'installation des clés en fonction de la taille du réseau.

On remarque qu'il est bien clair que dans la méthode RECC-CENT, le leader consomme plus d'énergie qu'un leader dans les deux approches de notre méthode (AVL-KDC, AVL-Leaders) et on observe aussi qu'il y a un grand écart entre AVL-KDC et les autres approches. Cette grande différence est dû au fait qu'un leader dans AVL-KDC transmet moins de paquets à tous les nœuds de son cluster, et dans l'approche AVL-Leaders, un leader doit transmettre les clés publiques à tous les nœuds de son cluster, mais le nombre de paquets transmis est moindre que dans la méthode RECC-DIST à partir du point 900, car dans l'approche RECC-DIST, le nombre de transmissions de messages est relié à la taille du cluster et du réseau et dans l'approche AVL-Leader, ce nombre est relié à la hauteur de l'arbre AVL qui reste plus stable, même si la taille du cluster augmente.

On remarque aussi que dans l'approche CECKM, un leader consomme moins d'énergie que dans AVL-Leaders, mais on a compté seulement l'énergie de la communication et on n'a pas pris en compte l'énergie de calcule des clés ECC, et dans l'approche CECKM, un leader à une échelle de calcul des clés ECC très grande par rapport à notre approche.

Cela montre que pour les nœuds leaders, notre approche AVL-KDC à une meilleure efficacité énergétique que les autres approches. Ce rendement assure aux nœuds leaders une durée de vie plus longue.

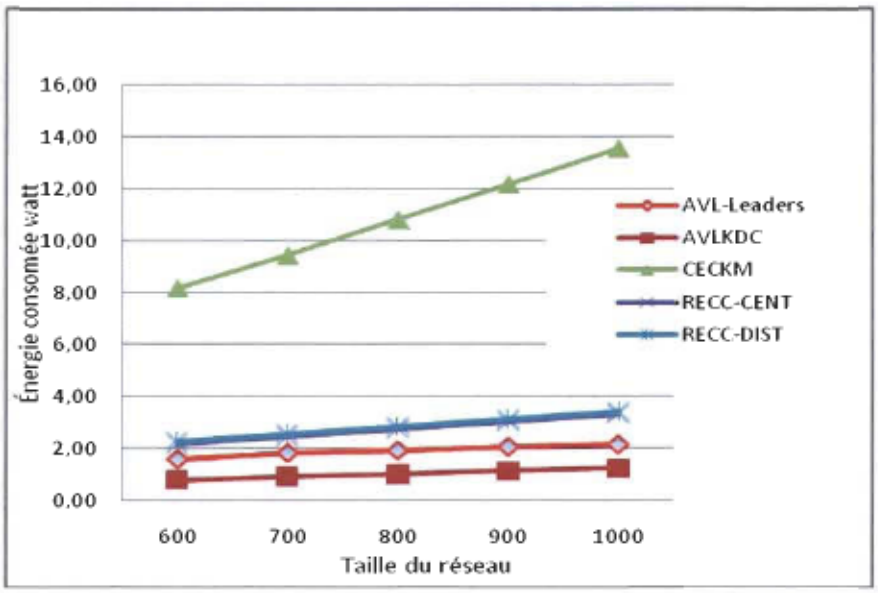



**Figure 6.3. La consommation d'énergie par un nœud normal.**

La figure 6.3 montre la consommation d'énergie par un nœud normal en fonction de la taille du réseau.

Les courbes montrent qu'un nœud dans la méthode CECKM dépense une quantité énorme d'énergie, comparativement avec les autres approches; et on observe aussi que les nœuds normaux dans les deux approches RECC consomment plus d'énergie que dans notre méthode, cette différence est du au fait que dans notre méthode, les nœuds normaux échangent moins de paquets que dans les autres méthodes.

On conclut donc, que nos deux approches garantissent à un nœud normal une vie plus longue comparativement aux autres méthodes.

### 6.4.3 Comparaisons du nombre de clés stockées

Les nœuds de capteurs sans fil se caractérisent par une taille de mémoire très limitée, par exemple, les nœuds MicaZ ont une mémoire RAM 4kOctets et une mémoire flash de taille 512 Kbits, c'est une contrainte très importante et chaque clé ECC à une taille de 40 Octets.

Pour cela, nous nous intéressons au nombre de clés ECC stockées dans chaque type de nœud, car la taille d'espace mémoire de stockage exploité est fortement reliée au nombre de clés stockées.

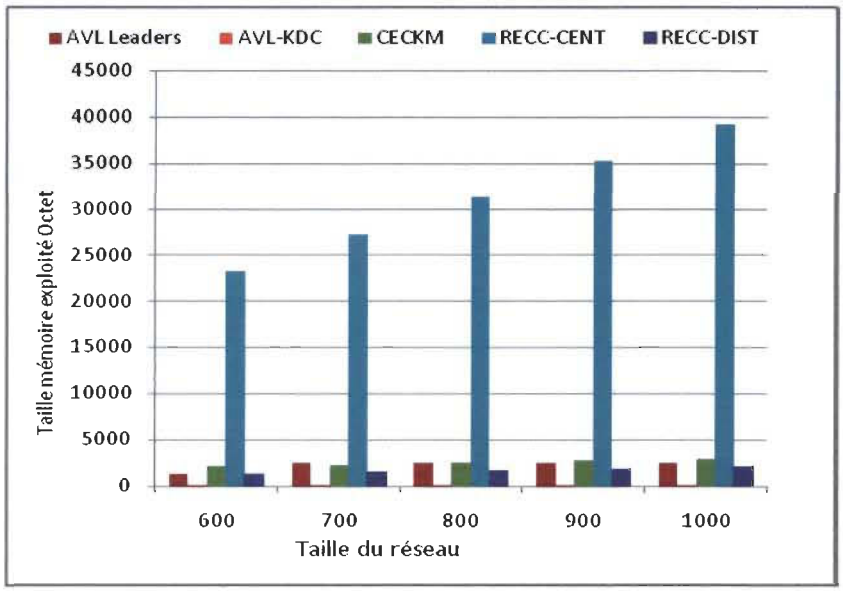



**Figure 6.4 : L'utilisation de la mémoire par un nœud leader**

La figure 6-4 montre la taille de la mémoire exploitée pour stocker les clés par un nœud leader en fonction de la taille du réseau.

Il est bien clair que dans l'approche RECC-CENT, pour stocker les clés, un leader utilise un espace mémoire très grand (plus que 23280 Octets), parce que le leader est pré-chargé avec toutes les clés publiques de tous les nœuds du réseau. Les approches AVL-Leaders et CECKM et RECC-DIST utilisent moins d'espace mémoire. L'espace utilisé varie entre 1280 et 2560 octets. L'approche AVL-KDC exploite un espace mémoire très limité, du fait que le leader n'est pas chargé avec les clés des nœuds dans le cluster. C'est le serveur KDC qui est le responsable de l'arbre AVL, ce qui offre au leader une économie de mémoire de stockage.

On constate que dans notre approche un leader effectue une économie de stockage comparée aux autres approches.

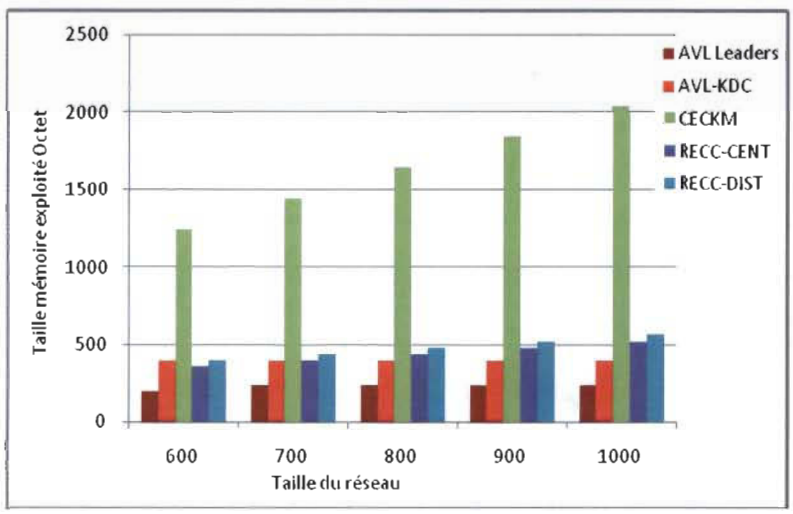



**Figure 6.5 : utilisation de la mémoire par un nœud normal.**

La figure 6.5 montre la taille de la mémoire exploitée pour stocker les clés par un nœud normal en fonction de la taille du réseau.

On peut voir que dans CECKM, pour stocker les clés, un nœud normal utilise le plus d'espace mémoire (plus que 1KOctets), et un nœud dans l'approche AVL-Leaders utilise le moins d'espace mémoire (moins que 250 octets). Dans la méthode RECC, le nombre de clés stockées est relié au nombre de voisins et à la taille du réseau et c'est la raison pour laquelle on remarque qu'à partir d'une taille de 700 nœuds et plus, les approches RECC utilisent plus d'espace mémoire que dans l'approche AVL-KDC.

L'approche AVL-Leaders proposée offre des économies de stockage comparé aux autres approches et notre approche AVL-KDC économise plus d'espace que les deux approches RECC dans les réseaux de grande taille. Du fait que le nombre des clés stockées est relié à la hauteur de l'arbre AVL comme il est présenté dans la figure 6.6, cette hauteur est plus petite dans l'approche AVL-Leaders que dans l'approche AVL-KDC. Cependant, dans la méthode RECC, le nombre des clés stockées est relié au nombre de voisins et dans la méthode CECKM, le nombre des clés stockées est relié au nombre de nœuds du cluster.

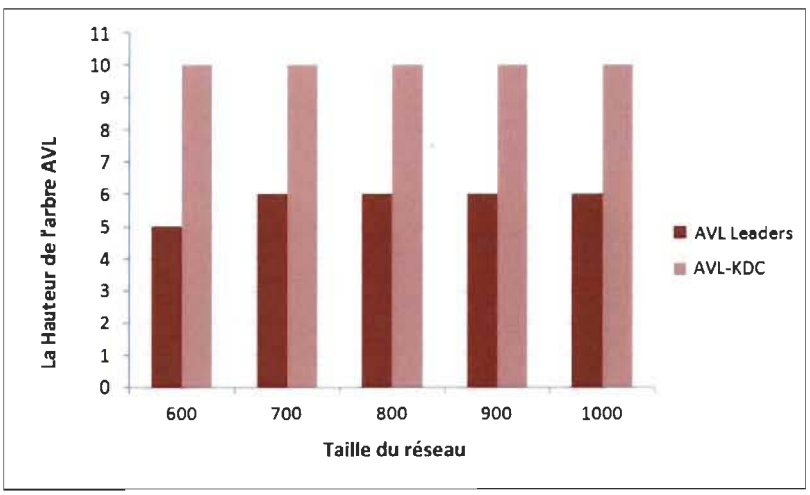



**Figure 6.6. La hauteur de l'arbre AVL**

La figure 6.6 montre la hauteur de l'arbre AVL utilisé en fonction de la taille du réseau et l'approche utilisée.

Il est clair que la hauteur de l'arbre AVL dans l'approche AVL-Leaders est plus petite que la taille de l'arbre AVL dans l'approche AVL-KDC, parce que cette hauteur est reliée au nombre des identificateurs des nœuds dans la liste « Liste_Membre », et cette hauteur est presque proportionnel à $\log(Nbr_{NouedCluster})$ dans l'approche AVL-KDC et à $\log(Nbr_{NouedN})$ dans l'approche AVL-Leaders.

### 6.4.4 Comparaison de l'espace mémoire RAM exploité

Une multiplication de deux clés ECC prend 282 octets de mémoire [19] et dans [20] selon la méthode Row-Wise, la taille de mémoire utilisée pour faire une multiplication d'un entier avec un entier de taille n, les mots prennent $n^2+3n$ octets de mémoire.

#### 6.4.4.1 Calcul d'une clé secrète entre deux nœuds normaux

Pour calculer la clé secrète entre deux nœuds normaux avec la méthode CECKM et la méthode RECC-DIST, les nœuds calculent la multiplication de leurs clés échangées. Une telle opération demande 282 Octets de RAM [30], mais avec notre méthode, les nœuds calculent la multiplication de leurs positions avec la clé partagée dans l'arbre AVL. Ces opérations nécessitent un maximum de 43 Octets de RAM [16].

Dans l'approche RECC-CENT, le leader calcule les clés secrètes de chaque nœud. Si un nœud à 7 voisins, il utilise 7*282 octets soit 1974 Octets de RAM, qui est une taille très importante. Si on utilise un nœud TELOSB [20] qui a une RAM de 10 KOctets (TELOSB est le plus puissant), la mémoire nécessaire pour calculer les clés secrets est au

minimum 19% de la taille total de la RAM. Mais, notre méthode utilise seulement 0,4% de la taille de la RAM.

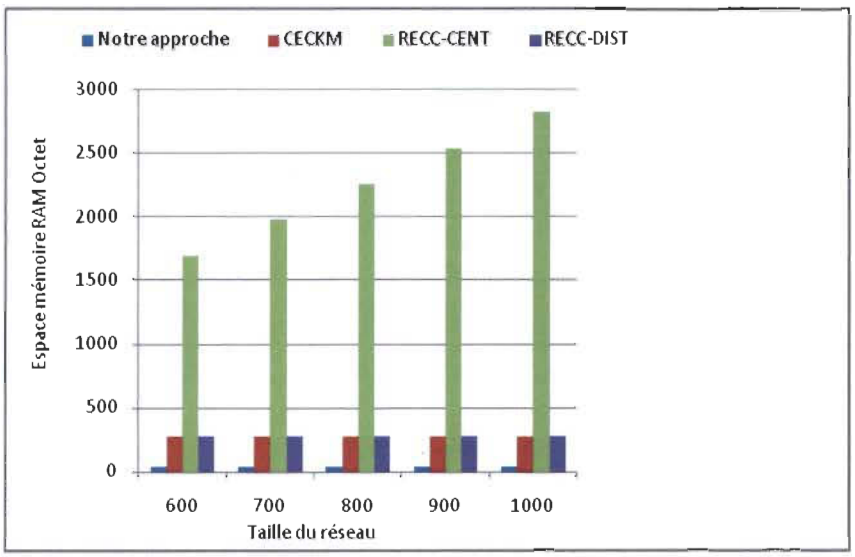



**Figure 6.7. Utilisation de la mémoire RAM pour calculer une clé partagé entre deux nœuds normaux.**

La figure 6.7 présente la taille de la mémoire utilisée pour calculer une clé partagée entre deux nœuds normaux en fonction de la taille du réseau.

#### 6.4.4.2 Calcul d'une clé secrète entre le leader et un nœud normal

Avec notre méthode, pour calculer la clé secrète entre un leader et un nœud normal du cluster, la clé secrète est égale à la multiplication de la position du nœud dans la

« liste_Membre » avec la clé racine de l'arbre AVL. Cette opération demande au maximum 18 octets de RAM, alors que dans la méthode CECKM un nœud normal doit faire l'opération de multiplication de toutes les clés publiques de tous les autres nœuds dans le cluster. Pour un nœud MicaZ, avec une RAM de 4 KOctets, il est impossible de faire une telle opération, car avec 4 octets, on peut supporter au maximum 14 multiplications de clés. Ainsi, pour un leader, si on utilise TELOSB qui a une RAM de 10Koctets, il ne peut pas calculer la clé secrète si la taille du cluster dépasse 32 nœuds.

La figure 6.8 présente la taille de la mémoire utilisée pour calculer une clé partagée entre un nœud normal et un leader en fonction de la taille du réseau.

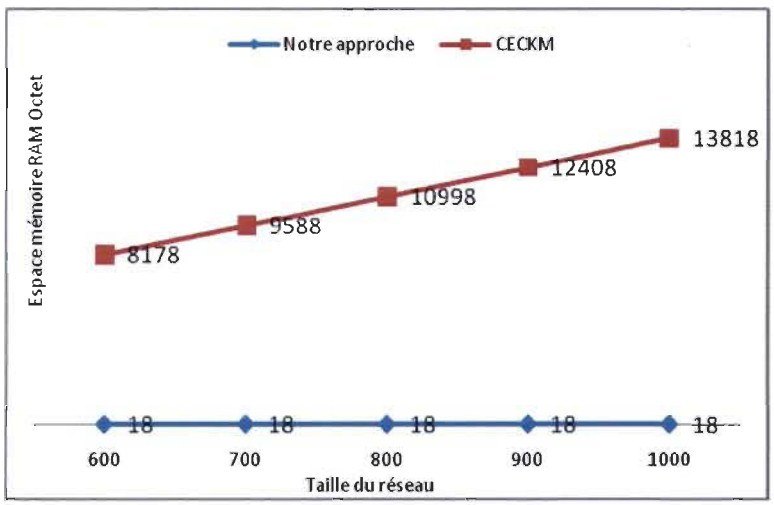



**Figure 6.8. Utilisation de la mémoire RAM pour calculer une clé partagé entre un nœud et un leader**

Notre méthode prouve que quelque soit la taille du réseau, soit pour un nœud normal ou un nœud leader, il est toujours possible de calculer les clés secrets et cette opération est légère et ne cause aucun problème contrairement aux autres méthodes.

## 6.5. Mise à jour des clés

Dans notre méthode, quand un nœud normal quitte le réseau, les clés publiques qui se trouvent le long du chemin du nœud vers la racine de l'arbre AVL seront changées et propagées aux nœuds dans le cluster.

Dans la méthode RECC, quand un nœud normal quitte le réseau, le leader du cluster dissémine un message de révocation des clés contenant l'identité du nœud compromis. En recevant le message de révocation, un nœud normal vérifie s'il communique avec le nœud compromis, s'il en est, le nœud normal révoque la clé partagée entre eux.

Dans l'approche RECC-DIST, il n'y a pas de méthode pour la mise à jour, alors, si un nœud est compromis, l'adversaire peut calculer les clés secrètes de tous les voisins car le nœud compromis stocke toutes les clés des ces voisin. D'un autre point de vue, comme cette méthode est basée sur l'arbre du routage, quand un nœud normal quitte le réseau, il cause une modification dans l'arbre de cheminement. Cette dernière exige la rediffusion de l'arborescence de cheminement et le recalcul des clés secrètes. La complexité de cette tache est reliée à l'importance de la position du nœud dans le réseau.

Dans la méthode CECKM, il n y a pas une méthode pour la mise à jour, mais on suppose qu'il faut faire une nouvelle installation des clés au niveau du cluster, car le nœud compromet toutes les clés publiques de tous les nœuds dans le cluster.

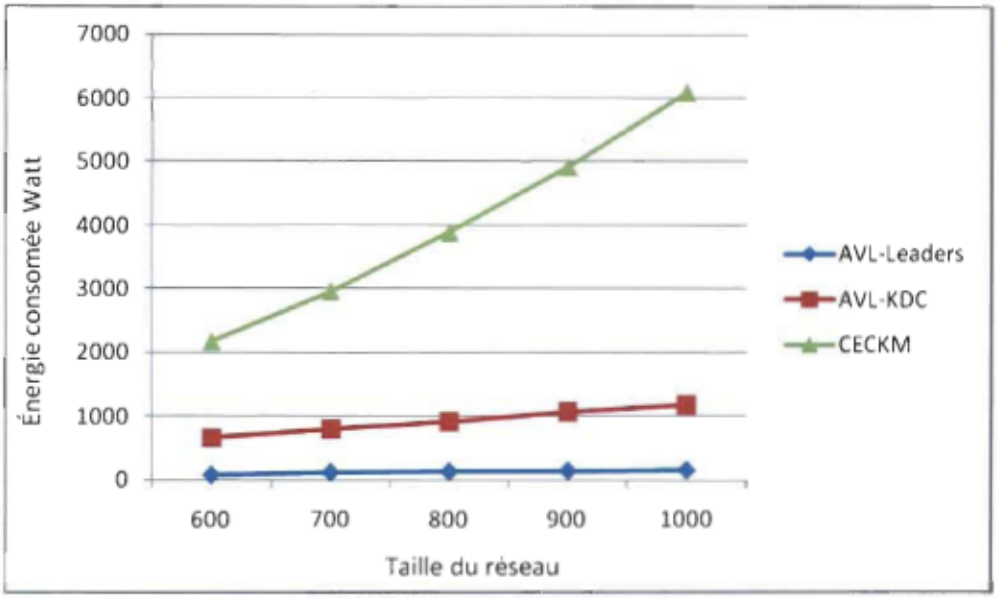



**Figure 6.9. La consommation d'énergie lors de la mise à jour des clés.**

La figure 6.9 présente la consommation d'énergie durant la mise à jour des clés de cryptage pour chaque méthode.

Les courbes montrent que notre approche AVL-Leaders échange moins de paquets. Cependant la méthode CECKM utilise un nombre de paquets très grand, soit 6 fois plus qu'AVL-KDC et plus 10 fois plus qu'AVL-Leaders.

Notre méthode présente une communication plus petite dû au fait qu'elle échangent moins de paquets. Elles demandent une transmission plus courte que la méthode CECKM. Ceci garantie un temps de mise à jour des clés plus court, par conséquent, moins d'énergie consommée.

## 6.6. L'ajout d'un nouveau nœud

Dans la méthode RECC-CENT, les nœuds leaders ne sont pas pré-chargés par des clés supplémentaires pour autoriser l'ajout d'un nouveau nœud, et dans RECC-DIST, un nœud leader n'a le moyen d'ajouter qu'un seul nouveau nœud; mais la procédure d'ajouter un nouveau nœud n'existe pas, aussi il n'y a pas de rafraîchissement des clés. S'il y a lieu d'ajouter un nouveau nœud dans le réseau, cela causera une modification dans l'arbre de routage et le recalcul des clés secrètes.

Aussi, dans la méthode CECKM, il n'y a pas une méthode pour ajouter un nouveau nœud. Mais on suppose que cela demande un échange de clés publiques entre le nouveau nœud et les autres nœuds du cluster et chaque nœud doit recalculer la clé partagée avec le leader et cela demande plus de communication et plus d'énergie.

Dans notre méthode à l'ajout d'un nouveau nœud, le leader envoie un message de rafraîchissement des clés à tous les nœuds du cluster.

La figure 6.10 présente le nombre de paquets échangés durant l'ajout d'un nouveau nœud pour chaque méthode.

Les courbes montrent que notre approche AVL-KDC échange moins de paquets. Cependant la méthode CECKM utilise un nombre de paquets très grand. Toutefois notre approche AVL-Leaders échange un nombre de paquets plus grand que AVL-KDC mais beaucoup moins que l'approche CECKM.

Cela montre encore que notre méthode présente moins d'échanges pour ajouter un nouveau nœud dans le réseau, et elles demandent une transmission plus courte que la méthode CECKM.

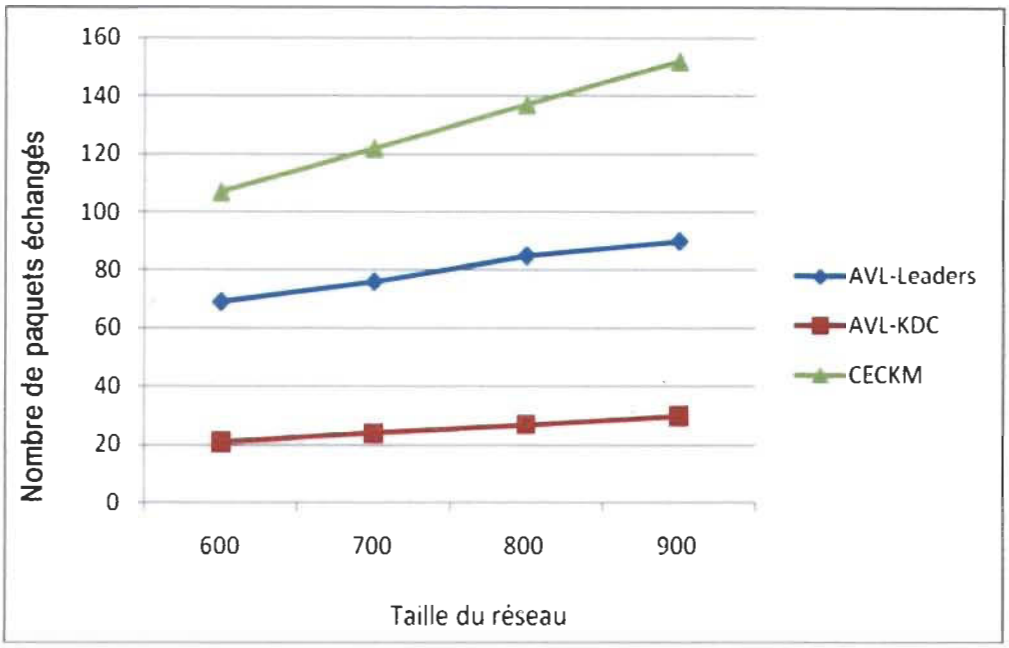



**Figure 6.10. Nombre de paquets échangés lors de l'installation des clés.**

## 6.7. Analyse de sécurité

Dans les deux approches RECC, après le déploiement, les nœuds dans le réseau évaluent leur position dans le réseau et calculent la table de routage. Toutes ces applications demandent beaucoup de communications entre les nœuds et plus de temps avant d'installer les clés de cryptage, et cela met le réseau en péril, car un adversaire peut s'introduire dans le réseau comme un nœud normal.

Aussi, dans la méthode CECKM, les clés échangées entre les nœuds et les leaders ne sont pas chiffrées, ce qui permet à l'adversaire de capter une clé et la modifier.

Par contre, dans notre méthode, avant le déploiement, tous les nœuds sont pré-chargés avec une clé du réseau, et après la création des clusters, chaque cluster à sa propre clé, et toutes les communications à partir du déploiement sont chiffrées. Cela ne donne aucune chance à un adversaire pour changer les clés.

Dans la méthode RECC-DIST, quand un nœud est compromis, les nœuds révoquent seulement la clé partagée. L'adversaire peut obtenir les clés publiques des voisins et calculer les clés secrètes. Dans ce cas, il peut jouer le rôle des autres nœuds qui ont partagé les clés secrètes calculées par l'adversaire.

Dans notre approche, chaque nœud stocke un ensemble de clés publiques, et si un nœud est compromis, alors toutes les clés partagées avec les autres nœuds seront changées, et dans cette façon l'adversaire ne pourra pas échanger avec les autres nœuds.

Notre méthode assure et garantie une sécurité plus importante qui couvre tout le réseau, une fois que, les nœuds sont déployés.

## 6.8. Conclusion

Dans ce chapitre, nous avons présenté les résultats obtenus avec les simulations effectuées. Le système d'exploitation utilisé est Tynios , et la plate-forme est Tossim.

Nous avons écrit un programme en Python, le système de sortie de débogage de l'application nous a permis d'évaluer les paramètres suivantes : le nombre de paquets envoyés et reçus, la taille de chaque paquet et l'énergie consommée.

Nous avons constaté que notre approche génère moins de paquets ce qui fait que le réseau consomme moins d'énergie, et nous avons observé que notre approche est plus légère car elle utilise moins d'espace mémoire que les autres méthodes.

Dans le chapitre suivant, nous allons présenter la conclusion de la recherche que nous avons amorcée dans le cadre de ce travail.

# CHAPITRE 7
# CONCLUSION

Dans ce mémoire, nous avons proposé un protocole de sécurité les réseaux de capteurs qui offre une bonne protection, en prenant en compte les caractéristiques limitées des capteurs qui influencent directement la performance globale du réseau, en particulier la sécurité et la durée de vie.

Le protocole de gestion des clés est conçu pour les réseaux de capteurs sans fil hétérogènes avec un déploiement à grande échelle.

L'approche nécessite un serveur qui génère les clés de courbes elliptiques ECC, et un arbre AVL pour stocker les clés du réseau.

Les clés utilisées sont des clés de 160 bits, ces clés assurent le même niveau de sécurité des clés RSA de 1024 bits.

La gestion de clés se fait soit au niveau global du réseau, et dans ce cas, le serveur construit l'arbre AVL et stocke toutes les clés du réseau, ou au niveau régional où chaque leader prend la charge de l'arbre AVL et stocke les clés de son cluster.

Chaque nœud est associé à une feuille de l'arbre, et il stocke seulement les clés se trouvant le long du chemin du nœud à la racine de l'arbre AVL. Cela permet une réduction significative du stockage et réduit énormément la communication et le calcul, ce qui offre une meilleure sécurité.

Nous avons aussi développé une simulation qui réalise l'implémentation de notre méthode; et afin de démontrer l'efficacité de cette méthode nous avons fait quelques tests et nous les avons comparés avec RECC "A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks" et CECKM

"High-Effect Key Management Associated With Secure Data Transmission Approaches in Sensor Networks Using a Hierarchical-based Cluster Elliptic Curve Key Agreement", deux méthodes basées sur la méthode de Cryptographie de Courbe Elliptique Diffie-Hellman.

Les analyses des résultats de notre simulation montrent que:

- Les deux approches de notre méthode génèrent moins de paquets et utilisent moins de mémoire que les autres méthodes. Ce résultat fait en sorte que les nœuds consomment moins d'énergie. Cela garantie à un nœud une vie plus longue, donc au réseau une longue vie.
- Notre proposition est plus légère, elle utilise moins de clés partagées et économise l'espace de stockage des clés.
- Notre méthode assure et garantie une sécurité plus importante qui couvre le réseau une fois que les nœuds sont déployés.

Pour conclure, nous pourrons dire que notre approche permet un gain significatif de la mémoire de stockage et une réduction énorme des paquets échangés lors de l'installation des clés avec un moins de calculs, tout en garantissant une meilleure sécurité.

# PERSEPECTIVE

Malgré que l'apprentissage du simulateur Tossim m'a pris un grand temps, je le recommande aux futurs développeurs de simulations sur les réseaux de capteurs sans fil, car il est possible ajouter facilement des modèles, et il nous permet de changer les paramètres de configuration du réseau, ce qui le rend plus intéressants à utiliser.

Le présent travail peut être enrichi par l'ajout de la fonctionnalité de «La signature numérique».

La signature numérique est un mécanisme cryptographique asymétrique qui permet de s'assurer de l'authenticité ainsi que de l'intégrité d'un message, en plus de permettre la non-répudiation.

Cet aspect est très important pour la sécurité dans les réseaux de capteurs sans fil, sa mise en place devrait se faire durant le déploiement, la création des groupes et l'installation des clés, car dans tout ce temps les nœuds dans le réseau échangent les messages utilisant des clés publiques pour crypter les messages.

L'idée est donc, d'implémenter une méthode qui permet le calcule d'une signature d'un message à l'aide des clés ECC utilisées dans notre méthode.

Comme notre protocole est conçu dans le but de réduire la consommation de l'énergie, alléger la taille du stockage, minimiser le calcul et tout en maximisant les performances de la sécurité; la méthode de la signature doit être gérée du même comportement pour assurer au réseau de capteur sans fil une longue vie avec une meilleur sécurité.

Dans notre méthode proposée, la façon de supprimer ou d'ajouter un nœud dans le réseau était simple, on n'a pas inclue le facteur de la mobilité du réseau, et on n'a pas étudié les

effets de la vitesse de cette mobilité sur le nombre d'échanges des messages et par conséquence la consommation d'énergie.

Alors la méthode de mise à jour clé peut être encore améliorée, avec une étude plus poussées sur des techniques d'encryptions et de mise à jour clé, ces techniques doivent réduire la quantité de clés transmises lors du départ d'un nœud, tout en garantissant une meilleure sécurité.

## BIBLIOGRAPHIE


[1] Laurent Eschenauer, Virgil D. Gligor, " A Key-Management Scheme for Distributed Sensor Networks", CCS '02 Proceedings of the 9th ACM conference on Computer and communications security, ACM New York, NY, USA , 2002.

[2] W. Du, J. Deng, Y. S. Han, S. Chen et P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge", 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004) Atlanta, GA, USA, ISBN 0-7803-8355-9, November 2004 .

[3] C.Castelluccia and Angelo Spognardi, " RoK: A robust key pre-distribution protocol for multi-stage Wireless Sensor Networks", IEEE Securecomm, Nice, France, September 2007.

[4] Donggang Liu, Peng Ning, "Establishing Pairwise Keys in Distributed Sensor Networks", Journal of ACM, October 27–31, 2003, Washington, DC, USA.

[5] R. Di Pietro, L. V. Mancini et A. Mei, "Efficient and Resilient Key Discovery Based on Pseudo-Random Key Pre-Deployment, " in 18th International Parallel and Distributed Processing Symposium (IPDPS'04), p. 217, 2004.

[6] P.Samundiswary, Padma priyadarshini et P. Dananjayan, " Performance Evaluation of Heterogeneous Sensor Networks", International Conference on Future Computer and Communication, ISBN 978-0-7695-3591-3, ICFCC, 2009.

[7] Quazi Ehsanul Kabir Mamun, et Sita Ramakrishnan, " SecCOSEn – A Key Management Scheme for Securing Chain Oriented Sensor Networks ", Communication Networks and Services Research Conference (CNSR 2008), ISBN 978-0-7695-3135-9, May 2008.

[8] A.S.Poornima, B.B.Amberker, "Tree-based Key Management Scheme for Heterogeneous Sensor Networks", 16th IEEE International Conference, ICON 2008, ISBN 978-1-4244-3805-1, NewDelhi, 2008.

[9] Yi-Ying Zhang, Wen-Cheng Yang, Kee-Bum Kim, Myong-Soon Park, "An AVL Tree-Based Dynamic Key Management in Hierarchical Wireless Sensor Network ", International Conference on Intelligent Information Hiding and Multimedia Signal Processing, ISBN 978-0-7695-3278-3, August 2008.

[10] AVL Tree in : Dictionnaires et Encyclopédies sur Academic, http://fr.academic.ru/dic.nsf/frwiki/124260.

[11] E.Munivel, Dr G M Ajit, " Efficient Public Key Infrastructure Implementation in Wireless Sensor Networks ", International conference on Wireless Communication and Sensor Computing, ISBN 978-1-4244-5136-4, February 2010.

[12] O.Arazil, H.Qi, D.Rosel, "A Public Key Cryptographic Method for Denial of Service Mitigation in Wireless Sensor Networks ", 4th Annual IEEE Communications Society, Conference on Sensor, Mesh and Ad Hoc Communications and Networks, ISBN 1-4244-1268-4 , San Diego, CA , August 2007 .

[13] Xiaojiang Du, Mohsen Guizani, Yang Xiao, and Hsiao-Hwa Chen, "A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks", IEEE International Conference on Communications, ICC 2007, ISBN 1-4244-0353-7 , Glasgow, August 2007 .

[14] Hua-Yi Lin, " High-Effect Key Management Associated With Secure Data Transmission Approaches in Sensor Networks Using a Hierarchical-based Cluster Elliptic Curve Key Agreement", ncm, pp.308-314, Fifth International Joint Conference on INC, IMS and IDC, ISBN 978-0-7695-3769-6 , Seoul, Korea, 2009.

[15] Marc Joye, " Introduction élémentaire à la théorie des courbes elliptiques, UCL Crypto Group Technical Report Series ", book in : http://www.dice.ucl.ac.be/crypto/, GC-1995/1, pages (17) (73) (29-52).

[16] A.Boukerche, H.A.B.F.Oliver,E.F.Nakamura and A.A.F.Loureiro.A Voronoi , " Vehuculer ad-hoc networks-A new callenge for localization" ,Elsevier Computer Communications, 2008.

[17] Implementing AVL Trees in: http://www.lri.fr/~fiorenzi/Teaching/Cours_Ing2000/arbravl_7.pdf.

[18] M. J. Handy, M. Haase, D. Timmermann, " Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection", Fourth IEEE Conference on Mobile and Wireless Communications Networks, Stockholm, erschienen in Proceedings, S. 368-372, ISBN 0-7803-7606-4, World Scientific Publishing Co. Pte. Ltd., September 2002.

[19] Arvinderpa.S.W, Nils Gura, Hans Eberl, Vipul Gupta, Sheuuling Chang Shantz, " Energy analyses of public key cryptography for wireless sensor networks ", Third IEEE International Conference on Pervasive Computing and Communications, (PerCom'05), ISBN 0-7695-2299-8m, Kauai Island, Hawaii, 2005, page (5).

[20] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, Sheueling Chang Shantz, " Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs ", in the book " Cryptographic Hardware and Embedded Systems - CHES 2004, 6th International Workshop Cambridge, MA, USA, August 11-13, 2004 ", ISBN 978-3-540-22666-6, Chaptre(4), page (8), Springer Berlin / Heidelberg, July 08, 2004.

[21] TELOSB mote platform in: http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html.

[22] Garci , a, E.M., Bermu , dez, A., Casado, R. , " Range-free localization for air-dropped WSNs by filtering node estimation improvements ", 6th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE , E-ISBN : 978-1-4244-7741-8 , November 2010.

[23] Mikhaylov, Konstantin Tervonen," Wireless Pervasive Computing (ISWPC)", 5th IEEE International Symposium, E-ISBN : 978-1-4244-6857-7, June 2010.

[24] Yong-Min, LiuXin-Hua Jiang, "A Protocol Model for Wireless Sensor Network", Proceedings of the 2009 International Conference on Networks Security, Wireless Communications and Trusted Computing, ISBN: 978-0-7695-3610-1, 2009.

[25] S. Duquesne, " Cryptographie sur les courbes elliptiques ", avril 2005, dans : http://www.lirmm.fr/~ejc2005/EJC_PDF/notes.pdf

[26] Claude Castelluccia et Aurélien Francillon, "Protéger les réseaux de capteurs sans fil ", 2008, dans : http://actes.sstic.org/SSTIC08/Proteger_Reseaux_Capteurs_Sans_Fil/SSTIC08-article-Castelluccia_Francillon-Proteger_Reseaux_Capteurs_Sans_Fil.pdf.

[27] Chung Kei Wong, M.Gouda, S. Lam, "Secure Group Communication Using key Graphs", IEEE/ACM Transactions on Networking, ISSN: 1063-6692, Feb 2000.

[28] Christophe Giraud, "Attaques de crypto systèmes embarques et contre-mesures associées", THESE présentée et soutenue publiquement le 26 octobre 2007, A l'Ecole Normale Supérieure, Université de Versailles Saint-Quentin, Paris, dans : http://www.prism.uvsq.fr/fileadmin/CRYPTO/TheseCG-new.pdf

[29] Geng Hao, N. V. Vinodchandran, Byrav Ramamurthy, "A Balanced Key Tree Approach for Dynamic Secure Group Communication", Proceedings 14th International Conference on Computer Communications and Networks, ISBN: 0-7803-9428-3, 2005.

[30] N.Bulusu, J.Heideman,D.Estrin et T.Tran, "Self-configuring localisation system :Design and experimental evaluation", Journal: ACM Transactions on Embedded Computing Systems (TECS), Volume 3 Issue 1, February 2004.

[31] Haowen Chan, Adrian Perrig , et Dawn Song, " Random Key Predistribution Schemes for Sensor Networks", SP '03 Proceedings of the 2003 IEEE Symposium on Security and Privacy, ISBN:0-7695-1940-7, 2003.

[32] Cryptographie sur les courbes elliptiques dans : http://fr.wikipedia.org/wiki/Cryptographie_sur_les_courbes_elliptiques.

[33] Les courbes elliptiques dans : public.enst-bretagne.fr/~saouter/**Elliptique**.ppt.

[34] Les arbres AVL dans : http://www.seg.etsmtl.ca/FHenri/inf145/Suppléments/arbres%20AVL.htm.

[35] Brad Karp , H. T. Kung, " GPSR: Greedy Perimeter Stateless Routing for Wireless Networks", Proceeding MobiCom '00 Proceedings of the 6th annual international conference on Mobile computing and networking, ISBN:1-58113-197-6, 2000.

[36] R. Cristescu et B. Beferull-Lozano, "Lossy network correlated data gathering with high-resolution coding," in Proc. IEEE IPSN 2005.

[37] P. Szczechowiak, L. B. Oliveira, M. Scott, M. Collier, et R.Dahab, "NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks", European conference on Wireless Sensor Networks, LNCS, vol. 4913. pp. 305-320, Feb.2008.

[38] Jean-Paul Quelen," Petit théorème de Fermat et codage RSA ", dans : http://jpq.pagesperso-orange.fr/divers/rsa/rsa.pdf

# ANNEXE 1: PROJET DE SIMULATION

Le projet utilisé pour les simulations se trouve dans le répertoire PROJET sur le DVD.

Le DVD du projet représente la machine virtuelle utilisé pour réaliser le projet, le système d'exploitation est Ubuntos2.1, les simulations se trouvent dans le répertoire « System\opt ».

Il est possible ouvrir cette machine virtuelle avec logiciel VMwareplayer.

# ANNEXE 2: COMMUNICATION

Hayette Boumerzoug, Boucif Amar Bensaber«Gestion de la sécurité dans les réseaux de capteurs sans fil. », le 79e congrès de l'Acfas, Cherbrouk, Mai 2011.

# ANNEXE 3: PUBLICATION

Hayette Boumerzoug, Boucif Aamar Bensaber, Ismaîl Biskri, « A key management method based on an AVL tree and ECC cryptography for wireless sensor networks », The 14th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, November 04 2011.

# Gestion de la sécurité dans les réseaux de capteurs sans fil

**Boumerzoug Hayette, Boucif Amar Bensaber**

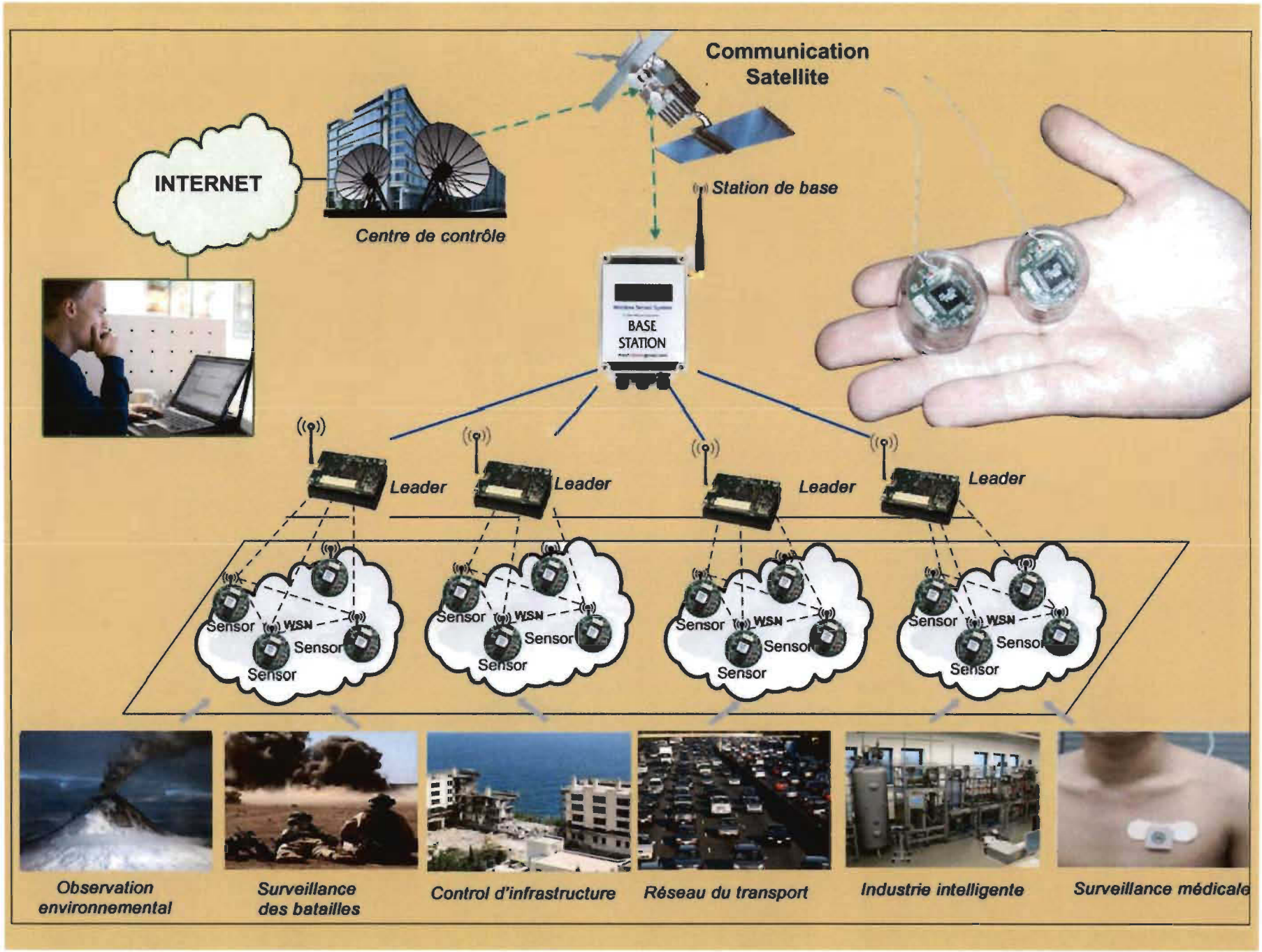
Département de Mathématiques et Informatique,UQTR

# I. INTRODUCTION

- Les réseaux de capteur sans fil sont composés de centaines ou de milliers de petits nœuds de capteurs déployés dans une zone géographique.
- Les nœuds ont des ressources de calcul, de stockage, de communication et d'énergie très limitées.
- Ces réseaux ont beaucoup d'applications dans les domaines militaire et civil, ces applications ont souvent besoin d'un niveau de sécurité élevé.

## Objectif

- Notre travail consiste à définir un protocole de sécurité qui permet d'assurer une bonne protection des données en prenant en compte les caractéristiques limitées des capteurs.


Communication Satellite
INTERNET
Centre de contrôle
Station de base
BASE STATION
Leader
Leader
Leader
Leader
Sensor
WSN
Sensor
Sensor
Sensor
WSN
Sensor
Sensor
Sensor
WSN
Sensor
Sensor
Sensor
WSN
Sensor
Sensor
Observation environnemental
Surveillance des batailles
Control d'infrastructure
Réseau du transport
Industrie intelligente
Surveillance médicale

# II. MÉTHODE

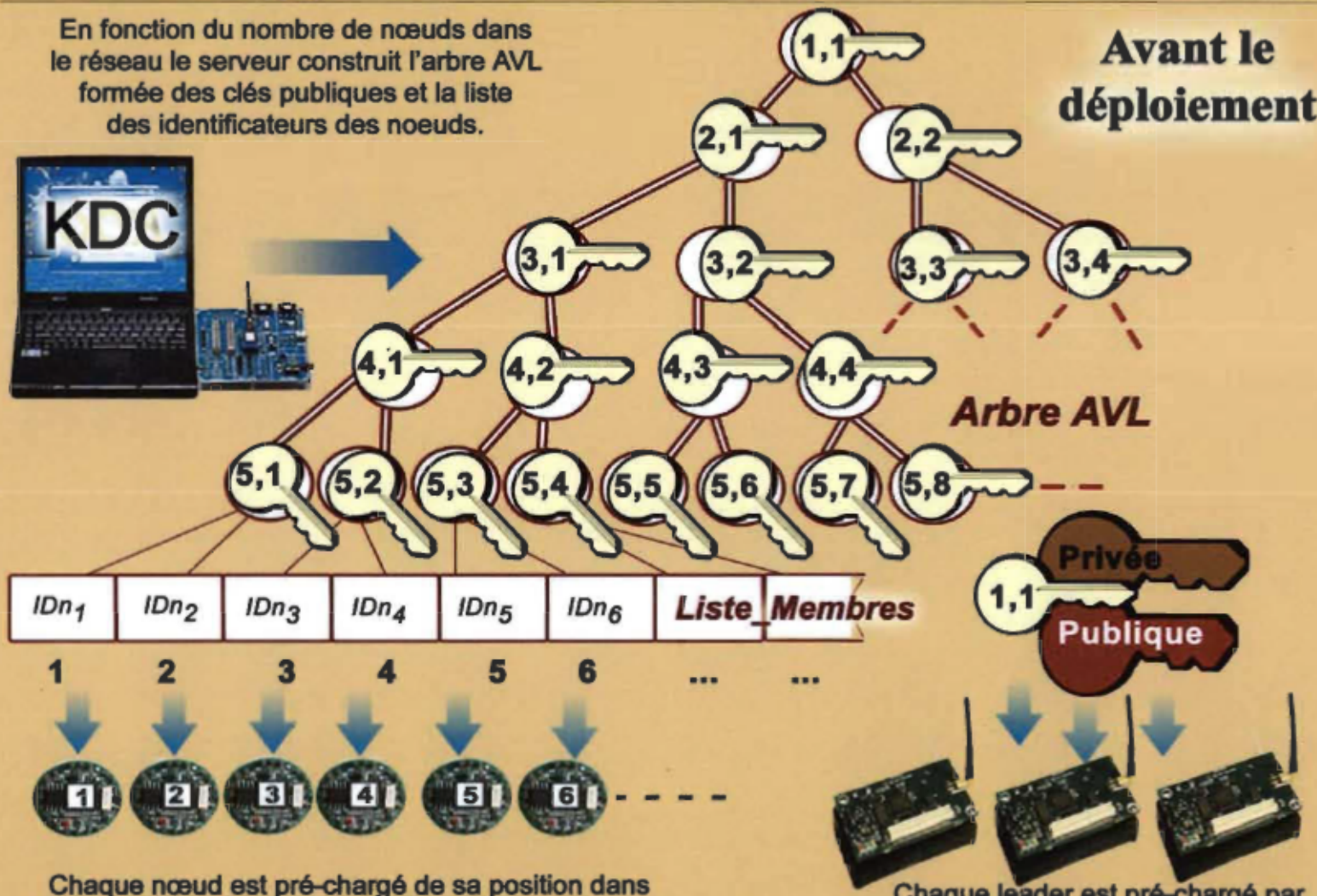
- Notre méthode comprend deux approches **AVL-Leaders** et **AVL-KDC**.
- Les deux approches sont basées sur la combinaison et l'amélioration de deux méthodes déjà proposées par la communauté scientifique :

1-Une Cryptographie basée sur les Courbes Elliptique.
2- Une gestion des clés basée sur un arbre binaire de recherche AVL.

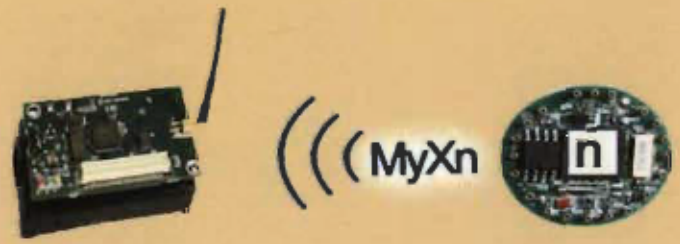
***Note:*** Une clé ECC est un point situé sur la courbe elliptique.

# II.1. L'installation des clés

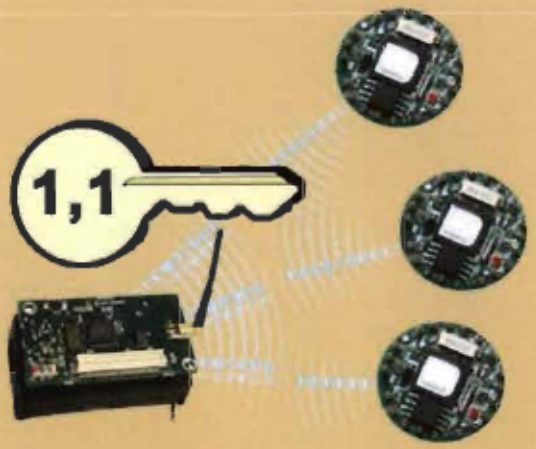
## II.1.a. L'approche AVL-KDC



# Création des Clusters




Durant la création des clusters, les nœuds et les leaders cryptent et décryptent les messages échangés avec la clé racine de l'arbre AVL.

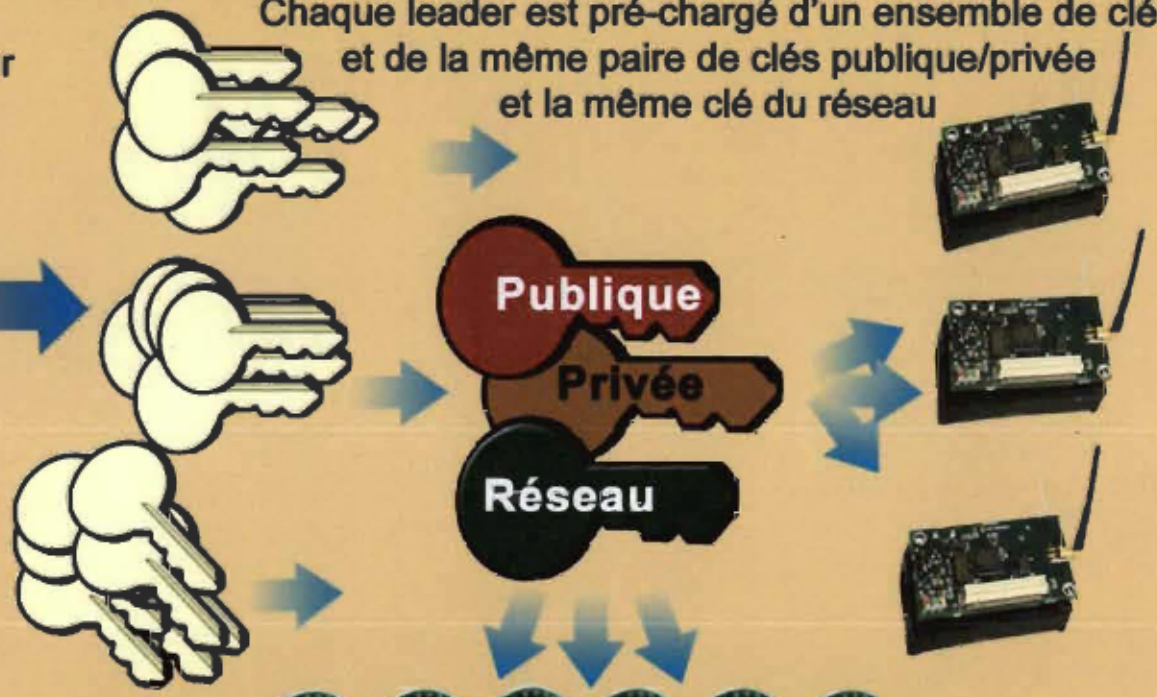



Chaque noeud dans le cluster envoie à son leader sa position dans la liste Liste_Membres.

# II.1.b. L'approche AVL-Leaders

Un serveur KDC est utilisé pour produire les clés d'ECC.


Chaque leader est pré-chargé d'un ensemble de clés et de la même paire de clés publique/privée et la même clé du réseau

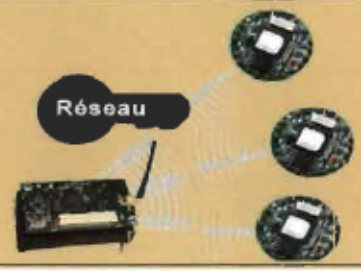



**Avant le déploiement**

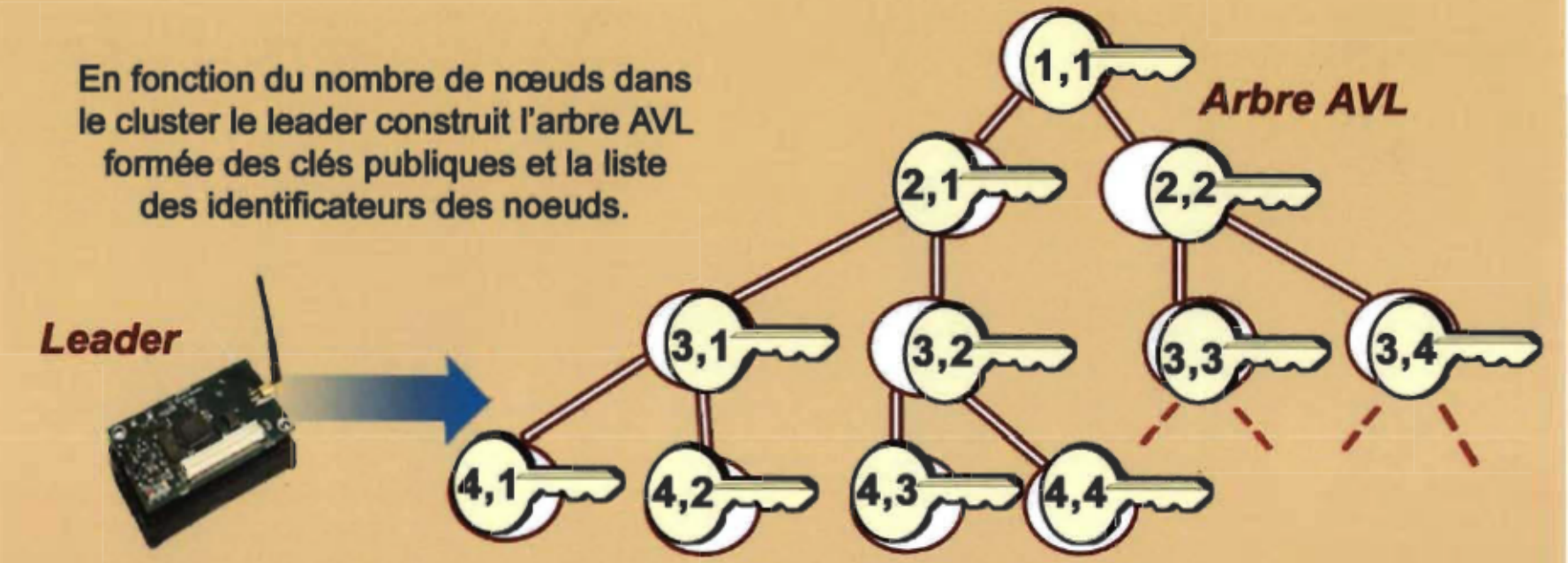

Chaque nœud de capteur est pré-chargé de la même clé du réseau.

## Création des Clusters

Durant la création des clusters, les nœuds et les leaders cryptent et décryptent les messages échangés avec la clé du réseau.



En fonction du nombre de nœuds dans le cluster le leader construit l'arbre AVL formée des clés publiques et la liste des identificateurs des noeuds.




Le leader envoie à chaque nœud dans le cluster sa position dans la liste Liste_Membres et les clés se trouvant le long du chemin du nœud à la racine de l'arbre AVL.

**Après la creation des clusters**

# II.2. Établir une communication

Les clés de chiffrement utilisées pour protéger les messages échangés sont les suivantes :

## Entre les leaders

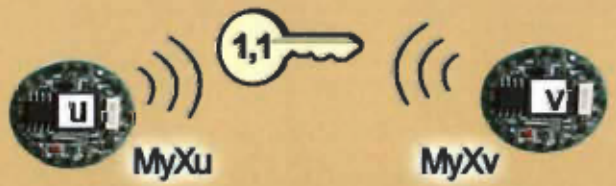

Les messages échangés entre les leaders sont cryptés avec la clé publique et décryptés avec la clé privée.

## Entre un noeud et son leader


La clé secrète est la multiplication de la clé racine de l'arbre AVL et sa position dans la liste Liste_membres.

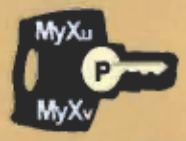
## Entre deux noeuds voisins

Soit Nu et Nv deux neouds voisins dans le cluster



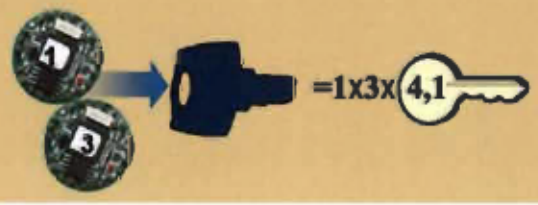

Nu et Nv échangent leurs positions dans la liste Liste_Membres utilisant la clé racine de l'arbre AVL

Soit P la clé commune, qui se trouve au niveau le plus haut dans l'arbre AVL.




Nu et Nv calclule la clé secrète =Xu.Xv.P


**Exemple:**

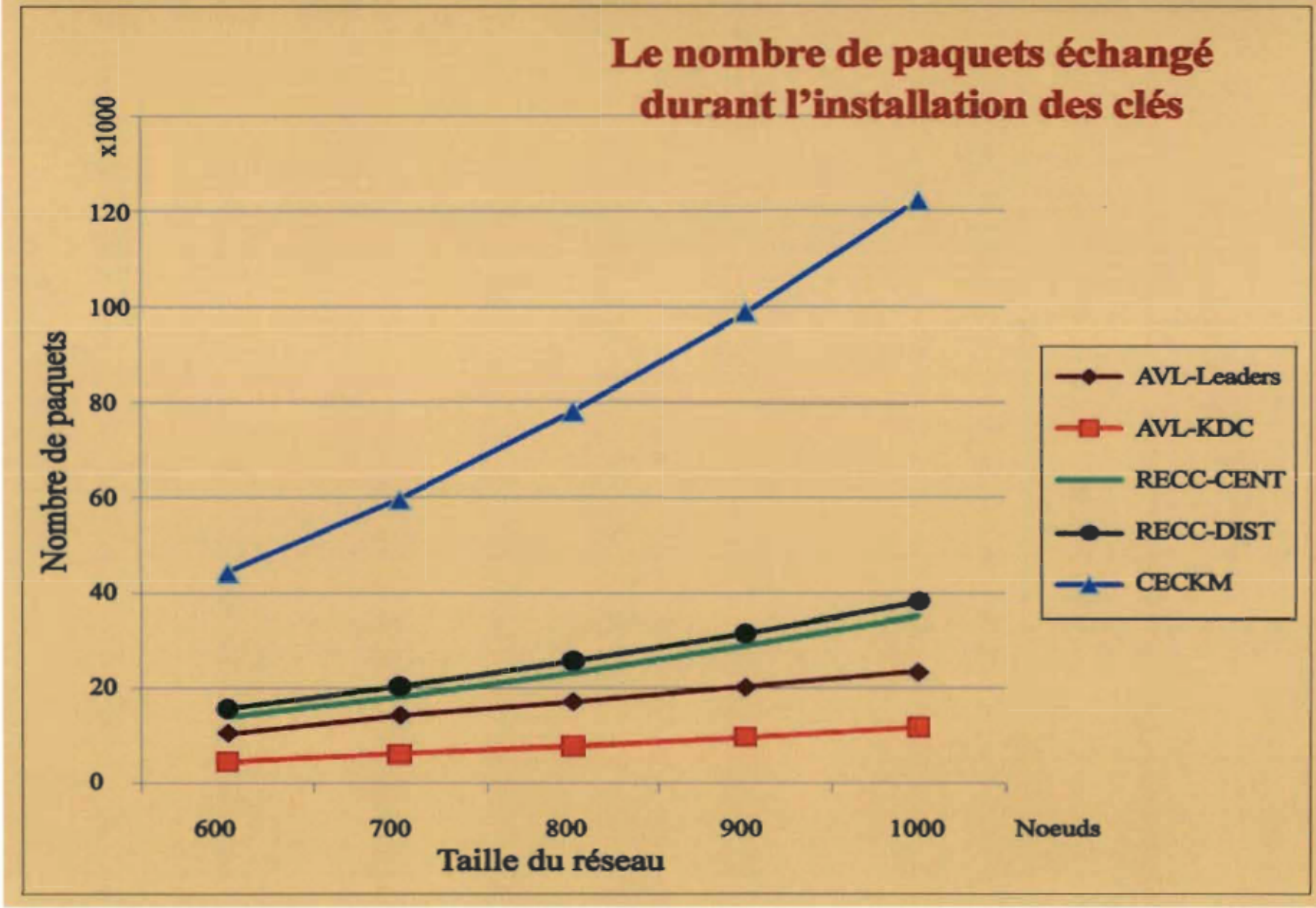


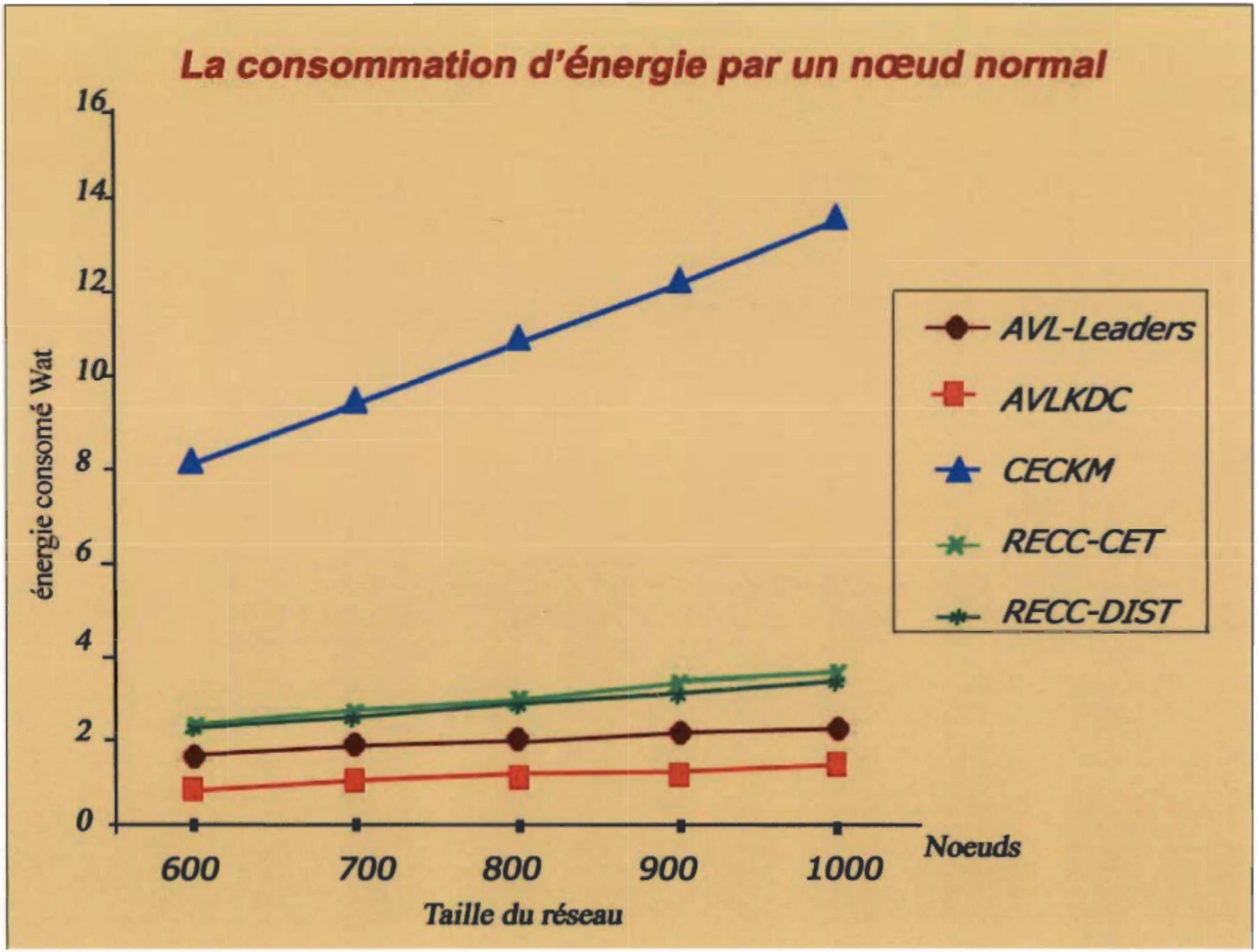
# III. LA SIMULATION

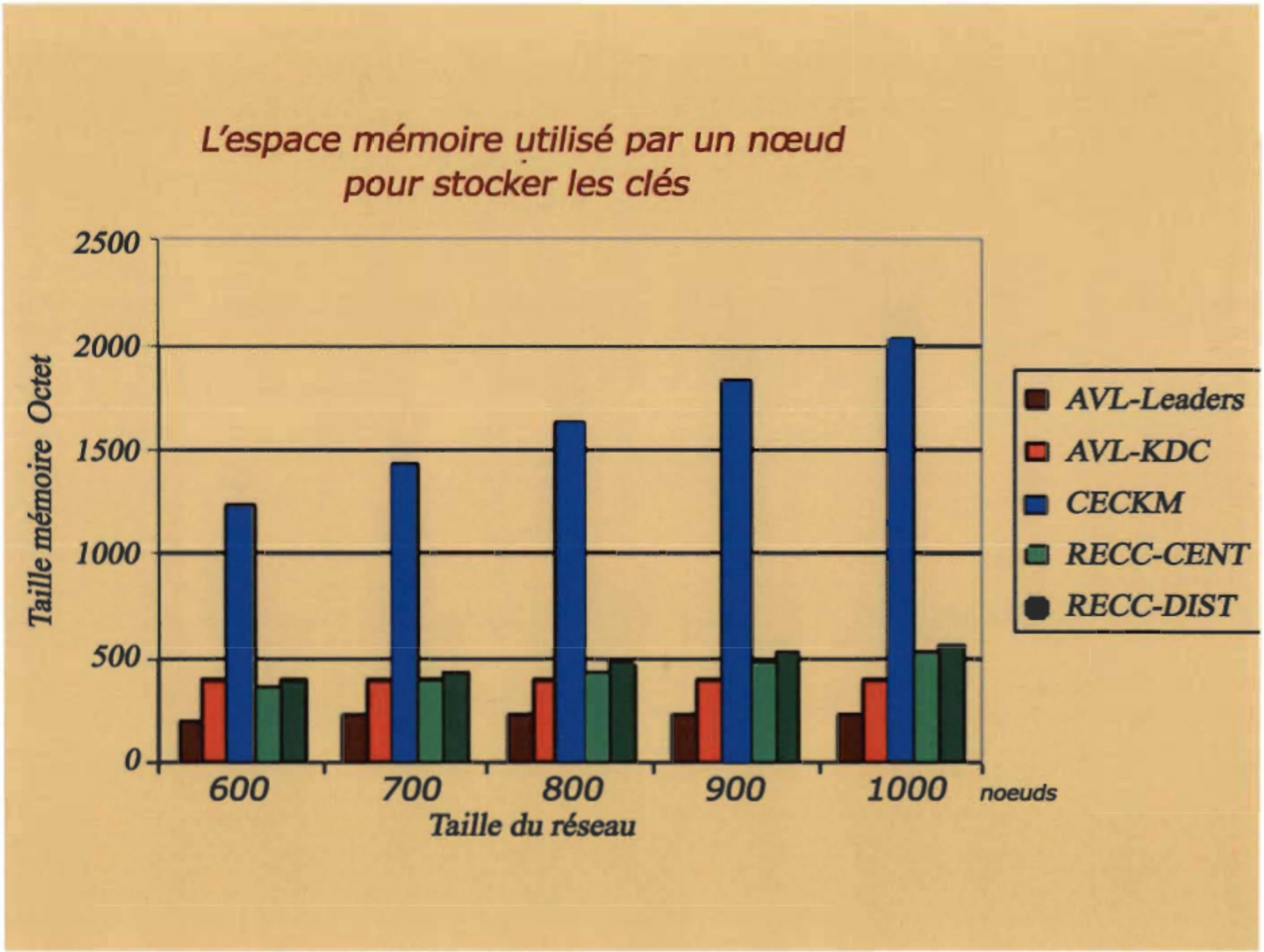
Pour démontrer l'efficacité de notre méthode, nous avons comparé son rendement par rapport aux deux méthodes existantes RECC et CECKM, et nous présentons son impact positif sur la réduction de la consommation d'énergie et l'exploitation de la mémoire.

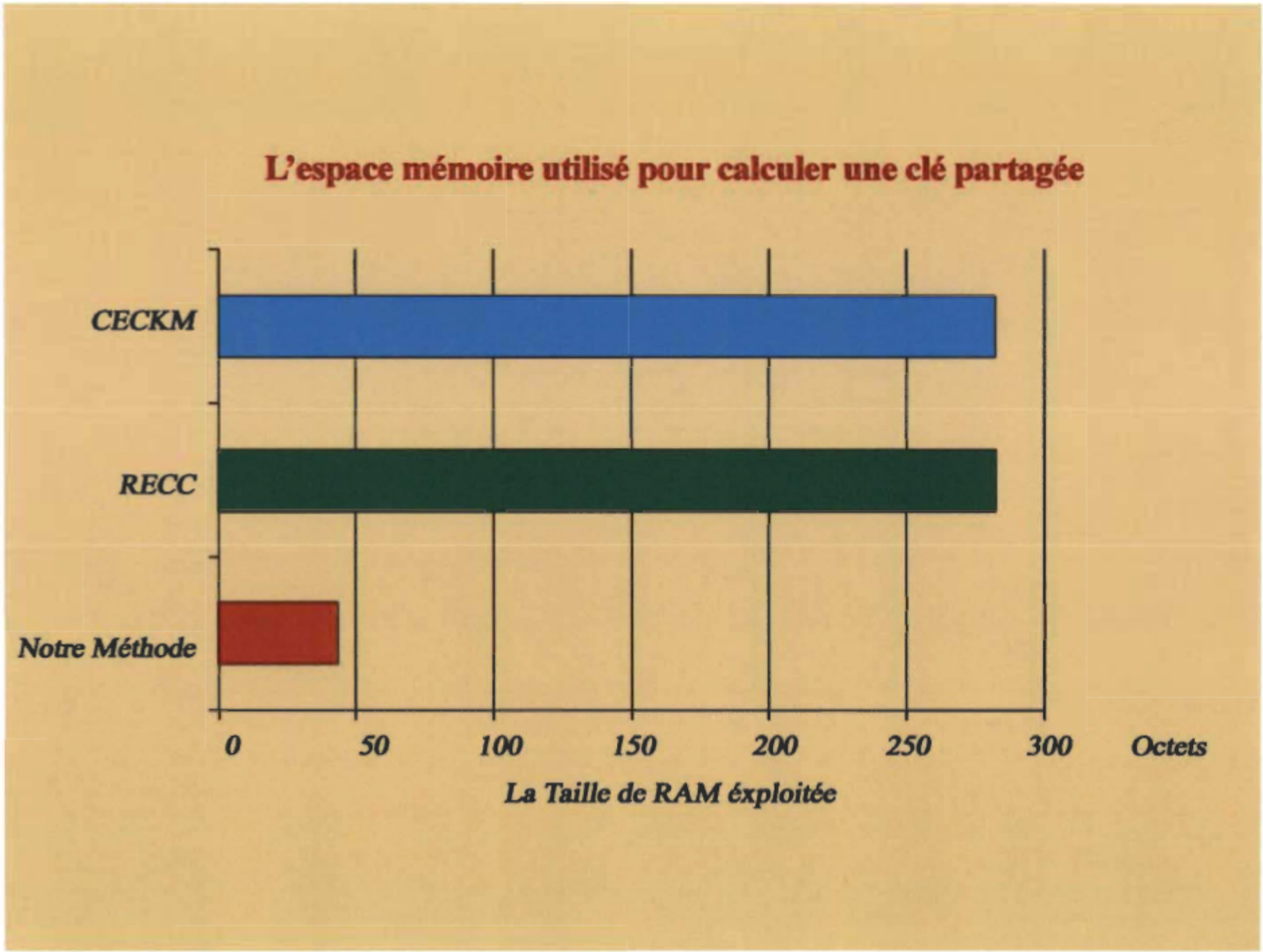
Le réseau est formé de 600 à 1000 nœuds parmi eux 20 leaders, les différents nœuds sont répartis uniformément au hasard dans une zone de 1000m × 1000m.

# IV. RÉSULTATS




La consommation d'énergie par un nœud normal
16
14
12
10
8
6
4
2
0
énergie consomé Wat
600
700
800
900
1000
Noeuds
Taille du réseau
AVL-Leaders
AVLKDC
CECKM
RECC-CET
RECC-DIST


L'espace mémoire utilisé par un nœud
pour stocker les clés
2500
2000
1500
1000
500
0
Taille mémoire Octet
600
700
800
900
1000
noeuds
Taille du réseau
AVL-Leaders
AVL-KDC
CECKM
RECC-CENT
RECC-DIST


L'espace mémoire utilisé pour calculer une clé partagée
CECKM
RECC
Notre Méthode
0
50
100
150
200
250
300
Octets
La Taille de RAM éxploitée

# V. ANALYSE

- Les courbes montrent que les deux approches de notre méthode génèrent moins de paquets et utilisent moins de mémoire que les autres méthodes. Ce qui fait que les nœuds consomment moins d'énergie. Cela garanti à un nœud une vie plus longue.
- On observe que notre proposition est plus légère, elle utilise moins de clés partagées et économise l'espace de stockage des clés.

# Analyse de sécurité

- Dans la méthode CECKM, les clés échangées entre les nœuds et les leaders ne sont pas chiffrées, ce qui permet à un adversaire de capter une clé et la modifier.
- La méthode RECC exige le calcul de l'arbre du routage avant l'installation des clés, cela demandent beaucoup de communications non chiffrées, qui met en péril le réseau .
- Par contre, notre méthode assure et garantie une sécurité plus importante qui couvre le réseau une fois que les nœuds sont déployés.

# VI. CONCLUSION

- Nous avons présenté un protocole de gestion des clés efficace pour les réseaux de capteur sans fil hétérogènes avec un déploiement à grande échelle.
- Notre proposition nécessite un serveur qui génère les clés ECC, et un arbre AVL pour stocker les clés du réseau.
- Chaque nœud est associé à une feuille de l'arbre, et il stocke seulement les clés se trouvant le long du chemin du nœud à la racine de l'arbre AVL. Cela permet une réduction significative du stockage et réduit énormément la communication et le calcul en offrant en même temps une meilleure sécurité.

## Remerciements

▪À mon directeur Boucif Amar Bensaber pour son suivi et son encadrement, au professeur Alain Goupil pour son aide et aux deux chargés de projets Guy Therrien et Daniel Saint Yves pour leurs soutiens techniques.

## Références

1. S. Duquesne, Ecole Jeunes Cherceurs, Cryptographie sur les courbes elliptiques, 5 avril 2005, http://www.lirmm.fr/~eic2005/EJC PDF/notes.pdf
2. http://www.seg.etsmtl.ca/FHenri/inf145/Suppléments/arbres%20AVL.html
3. Xiaojiang Du, Mohsen Guizani, Yang Xiao, and Hsiao-Hwa Chen, A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks, IEEE 2009.
4. Hua-Yi Lin, High-Effect Key Management Associated With Secure Data Transmission Approaches in Sensor Networks Using a Hierarchical-based Cluster Elliptic Curve Key Agreement, Department of Information Management, China University of Technology, Taiwan, R.O.C.

# A keys management method based on an AVL tree and ECC cryptography for wireless sensor networks

Hayette Boumerzoug
Laboratoire de Mathématiques et Informatique appliquées LAMIA
Department of Mathematics and Computer Science
University of Quebec at Trois-Rivières, Canada
3351 Bd des Forges, C.P 500
Hayette.Boumerzoug@uqtr.ca

Boucif Amar Ben Saber
Laboratoire de Mathématiques et Informatique appliquées LAMIA
Department of Mathematics and Computer Science
University of Quebec at Trois-Rivières, Canada
3351 Bd des Forges, C.P 500
Boucif.Amar.Bensaber@uqtr.ca

Ismail Biskri
Laboratoire de Mathématiques et Informatique appliquées LAMIA
Department of Mathematics and Computer Science
University of Quebec at Trois-Rivières, Canada
3351 Bd des Forges, C.P 500
Ismail.Biskri@uqtr.ca

## ABSTRACT

In this paper; we propose a security protocol for sensor networks that provides good protection while taking into account the limited resources of the sensors. The protocol is based on an effective key management method with a minimum storage of keys. Our approach is based on the combination and improvement of two methods already proposed by the research community: cryptography based on elliptic curves and key management based on an AVL tree. Compared with RECC "A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks" and CECKM "High-Effect Key Management Associated With Secure Data Transmission Approaches in Sensor Networks Using a Hierarchical-based Cluster Elliptic Curve Key Agreement" two methods based on Diffie-Hellman elliptic curve cryptography method, our approach provides a positive impact on reducing energy consumption and memory storage. It saves significant time and memory and it reduces the exchanged packets during keys installation with fewer processing operations.

## General Terms

Security

## Keywords

Keys management, sensor networks, elliptic curve cryptography, AVL tree.

## 1. INTRODUCTION

A wireless sensor network consists of hundreds or thousands of sensor nodes that are powered by batteries and are typically deployed randomly in environments often open. These nodes are small, and therefore, have computational resources, storage memory, communication and power very limited. These networks are of particular interest for military, environmental, and medical applications, and for applications related to monitoring of critical infrastructures. But not all applications always have the same security constraints. Data encryption is often required for sensitive applications such as military applications and medical applications.

In many applications, the establishment of encryption keys must be done once the sensor nodes are deployed. This is the case when a large number of sensors are deployed randomly, for example, from a helicopter. Each node must establish a secret key with each of its neighbors but its neighbors are not known before the deployment. The keys exchange should take place as soon as the deployment starts.

## 2. STATE OF THE ART

Various solutions have been presented in the literature. Often, they use dynamic methods recommending the updating of the keys to provide resistance against attacks for an extended period.

In 2002, Eschenauer and Gligor proposed a probabilistic protocol [1], in this protocol the network administrator creates a table of L random keys, and each node is pre-loaded with a subset of M keys randomly chosen from the created keys before. When two nodes, A and B want to establish a secret key, they exchange their keys' indexes, and then they use their common keys to generate a secret. In this protocol, it is always possible that a node other than A and B knows the shared keys between A and B, and then he can compute the secret that A and B have just established. However, Eschenauer and Gligor showed that this probability may be low if the parameters L and M are chosen carefully.

Methods [2, 3, 4, 5, 6] improve the probabilistic protocol [1]. They provide a random pre-distribution of keys that involves creating a large list of keys by a network administrator, and before deployment, a subset of this list randomly chosen is distributed to each node. It is possible that two nodes have common keys in their subsets of keys, so they use common keys to establish a link key between them. The choice of subsets is based on the probability of Eschenauer and Gligor [1]. One drawback of these approaches is that the number of keys is very large.

COSEN method [7] is based on a partial key pre-distribution and a symmetric cryptography. Before deployment, a partial list of keys is generated at the base station. Each node is preloaded with a partial list of keys. Once a node knows which keys it will use with its neighbors, it removes the remaining keys. The problem is that it is impossible to add a node or to update the keys with this method.

In [8], the authors proposed a method based on m-ary tree. Each subset of nodes has a single shared key used for communication between nodes. If a node is compromised, an adversary can listen to all messages between all nodes in the subset.

In [9], the proposed method is based on random pre-distribution of keys based on the AVL tree [10]. After deployment, each node generates its master key, and then sends it to its direct neighbors with its identity. After, each pair of nodes produce their shared key. This method generates a lot of exchange and also in each period, the cluster has to change its leader (he chose the node that has more energy).

In [11], the proposed method is called "Micro Public Key Infrastructure". Only the base station is authenticated by the nodes, using the public key. First, each node produces a random key, and then sends it to the base station, so that it is used as the shared secret key. Afterwards, if two nodes need to establish a secure channel, one of them, sends a request to the base station, and then, the base station produces a random key and sends it to each of them. Periodically, each node starts an update, which is actually a new installation related to the key length and to the complexity of the algorithm used, so that the network consumes much energy.

[12] and [13] present a cryptography based on elliptic curves. Thus, in the method [12], a certification authority calculates and delivers to each node a pair of private / public keys, and if two nodes want to establish a connection, they produce an ephemeral common key for each session requiring many calculations. The method [13] is based on the routing protocol for heterogeneous sensor networks. The main idea is that a node can only communicate with few neighbors. Before installing the keys, the protocol calculates the routing tree by using a service to evaluate the locations of the nodes, and this requires a lot of exchange of messages between nodes. Of course, all these communications are not protected, allowing an attacker to capture and change these messages. Also, the protocol does not have the option of adding a new node, or indeed, do an update of the keys.

# 3. CRYPTOGRAPHY BASED ON ELLIPTIC CURVES (EEC)

In recent years, ECC has attracted much attention as the security solution for wireless networks, due to the fact that ECC keys are smaller compared to the RSA keys, for example, 160 bits ECC key offers the same level of protection as of 1024 bits RSA key [14].

An elliptic curve is a very simple object but which has properties quite surprising. They have in most cases the form ($y2 = x3 + ax + b$) [15]. ECC is a key point P, located on the curve, P = (x, y).

There are mainly two kinds of cryptosystems [15]. Asymmetric crypto systems where a key known to everyone is used to encode the message but it cannot deduce the key to decrypt the message. In the Symmetric crypto-systems, the two correspondents agree on a shared secret key that only they know. In our approach we will use the symmetric crypto-systems.

# 4. THE PROPOSED METHOD

It is based on a secure control unit [16] (leader node or KDC server) that stores the ECC keys in an AVL tree. Each node is associated with a leaf of the tree, and all keys located along the path from the leaf to the root of the AVL tree belong to that node.

The AVL tree [17] is a balanced binary search tree. For each node of the tree, the height difference of its sub trees is at most equal to 1. The binary search trees used to make operations such as insertion, deletion and search in a time proportional to the height of the tree, which is nearly proportional to log(n).

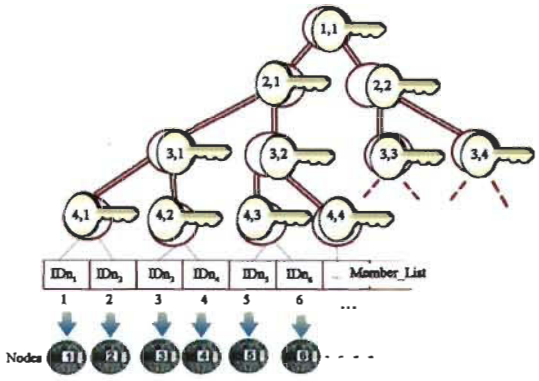



**Figure 1: The AVL tree of keys**

## 4.1 The network structure

It is a heterogeneous network [18] which contains a small number of high-quality sensor nodes that act as leaders. These sensors are equipped with powerful and durable materials [13]. The network thus contains a large number of low-quality sensor nodes that form groups around leaders. The high-quality sensors receive messages from low -quality nodes in the cluster, and then followed them to the base station.

Here are some definitions of acronyms used in the rest of the document:

KDC: keys Distributor Center.

LD: Leader.

P: Public Key.

ND: Normal sensor node.

CRL: A pair of public / private keys, used by leaders to communicate.

CR: A pair of public / private keys, called a network key used for communication between normal nodes and leaders.

CC: Key of a cluster.

Member_List: List of nodes IDs in the network.

Set_PublicKeys: All public keys those are located along the path from the node to the root of the AVL tree.

MyX: Position of the node in the Member_List.

UID: Unique identifier that is send to each node.

## 4.2 Keys generation

With our key management method, the base station is equipped with a KDC server that produces the ECC public keys and the (CRL,CR) keys pair.

## 4.3 Keys establishment at the base station: AVL-KDC

In the AVL-KDC method, the keys are generated at the base station.

### *4.3.1 Before deployment*

Depending on the number of regular nodes, the server constructs the AVL tree formed by the public keys and the "Member_List".

Each leader LD is pre-loaded with the same key CRL and the same key of network CR which is the key at the root of the AVL tree. Thus, each node ND is pre-loaded with his "Set_PublicKeys", its unique identifier UID, and its position MyX in the "Member_List". These informations are used for identification and authentication when exchanging.

#### *4.3.2 Creating clusters*

After deployment (as in [7]), each leader sends a message "join" encrypted with the network key CR. When a node ND receives the first message "join", it decrypts it with the key CR, and then it returns to the Leader LD a message "joined" containing its ID encrypted with network key CR. With all the IDs of nodes in the cluster, the leader LD produces the "Member_List" list.

#### *4.3.3 Calculation of shared keys*

If a node NDU wants to establish a communication with its leader, it sends him a message containing its position MyXU. The leader associates the position MyXU with its node ID in its "Member_List" list. Then the leader and the node compute their secret key (MyXU × CR).

If two neighboring nodes in the cluster (NDU,NDV) want to establish a connection, they exchange their positions (MyXU, MyXV), and then they determine the PUV point that is the common public key, which is at the highest level in the AVL tree (closest to the leaves). Then, nodes NDU and NDV calculate MyXU(MyXVPUV)=MyXV(MyXUPUV) to be their common secret key.

As leaders are equipped with durable materials, so for their exchange, they will always use the same key CRL pre-loaded before deployment.

### 4.4 Keys establishment at level leaders: AVL-Leaders

#### *4.4.1 Before deployment*

Before deployment each leader is pre-loaded by the same pair of public / private network key RC, a unique key CRL used for communication between the leaders, and a set of public keys. Also, each node is normally pre-loaded with only the same pair of network key RC.

#### *4.4.2 Creating clusters*

After selecting the groups as in AVL-KDC approach presented in the previous section, and depending on the number of nodes in the cluster, each leader built an AVL tree formed by public keys. Then, he replaced the network key CR by the root key CC, and sends it to all nodes in the cluster. After that, he sends to each node in the cluster NDU a message (encrypted with the new key CC) containing his position MyXU in the "Member_List", and finally he sends him another message containing a set of keys called Set_PublicKeys.

#### *4.4.3 Calculation of shared keys*

If two nodes want to establish a connection, they use the same steps as presented in the AVL-KDC approach. If a node NDU wants to establish a communication with his leader, he uses a key composed by multiplying the root key with its position MyXU.

### 4.5 Adding a new node

In the AVL-KDC approach before adding a new node in the network, the KDC server adds the ID of this new node in the nodes list, and he refreshes the keys corresponding to its position in the AVL tree. The new node NDN is pre-loaded with the keys "Set_PublicKeys", its identifier IDN, and its position MyXN. The server sends to all the leaders the new node identifier IDN and a refresh keys message.

When the new node NDN requests its adding to the cluster, it provides its identifier IDN and its position MyXN. The leader LD adds the parameters IDN and MyXN of the new node in its list "Member_List", and then it sends a message to refresh the keys to all nodes in the cluster.

In the AVL-Leaders approach, when a new node NDN requests its adding to the cluster, it provides its identifier IDN and the leader adds it in the list "Member_List", and it refreshes the keys corresponding to its position in the AVL tree. Then, it returns to this node its position MyXN in the list "Member_List" and its set of keys "Set_PublicKeys". Finally, the leader sends a message to refresh the keys to all nodes in the cluster.

### 4.6 Updated Keys

In the KDC-AVL approach, once an abnormal behavior of a node is detected by a leader, or once a node leaves the network, the server initiates the update of the public keys that are along the path of the node (which has left the cluster) to the root of the AVL tree and reconfigures the AVL tree. Then, it sends a message of update to all the leaders, and finally, each leader broadcasts the message to all nodes in its cluster.

In the AVL-Leaders approach, the Leader initiates the update of the public keys that are along the path of the node that has left up to the root of the AVL tree and reconfigures the AVL tree. Finally, it broadcasts the message to all nodes in the cluster.

## 5. SIMULATION

In our simulations, we used a network that consists of 600 to 1000 nodes MicaZ type. Among these nodes, 20 nodes are leaders' nodes. The nodes are distributed uniformly and randomly in an area of 1000 × 1000 m. The transmission range of a sensor is 60 m, the size of a packet is 32 bytes, and the error rate of transmission is zero.

The average number of neighbors is calculated as in [13] by the following formula:

$$Nbr_{Nodes N} \times \pi \times 602 / (1000 \times 1000) \qquad (1)$$

### 5.1 Comparison of packets exchanged

Figure 2 shows the number of packets exchanged during the installation of the encryption keys for each method

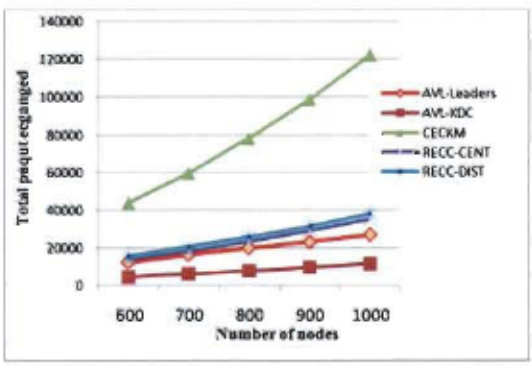



**Figure 2: The number of packets exchanged during the installation of the encryption keys**

The curves show that both approaches of our method generate few packets. They require a transmission which is shorter than the other methods, because they send few keys compared to others. This guarantees shorter keys installation time, and therefore more security. Indeed, for an intruder, if the communication is lengthy, it is more likely to capture and edit a public key during the exchange.

## 5.2 Comparison of the energy

Figure 3 shows the energy consumption per normal node depending on the size of the network.

The curves show that a normal node in the CECKM method spends a huge amount of energy compared with other approaches, and we can also observe that in the RECC approaches, the normal nodes consume more energy than in our method; this difference is because in our method, the normal nodes exchange fewer packets than other methods.

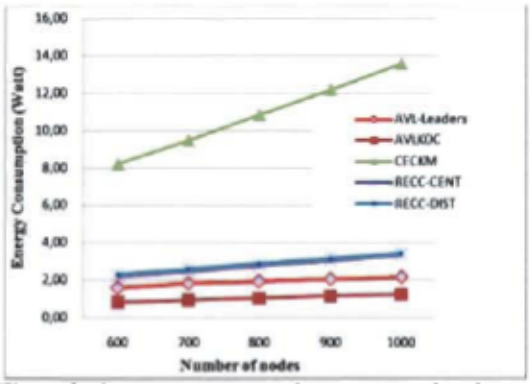



**Figure 3: the energy consumption per normal node.**

So we conclude that our two approaches ensure to a normal node a longer live compared to other methods.

## 5.3 Comparison of the number of stored keys

Figure 4 shows the size of memory used by a leader to store the keys according to the network size.

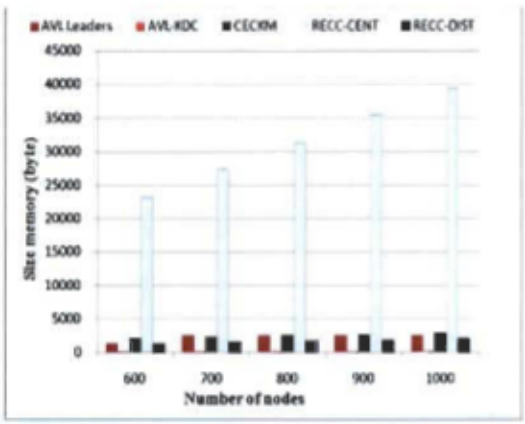



**Figure 4: The size of memory used to store the keys by a leader node**

It is clear that with RECC-CENT approach, to store keys, a leader uses a large memory space (more than 23280 bytes), because the leader is pre-loaded with all the public keys of all nodes in the network. AVL-Leaders, CECKM and RECC-DIST approaches use less memory space. The space used varies between 1280 and 2560 bytes. AVL-KDC operates a very limited memory space, because the leader is not loaded with the keys of the nodes in the cluster. The server KDC is the responsible for the AVL tree, which offers to the leaders an economy of storage memory.

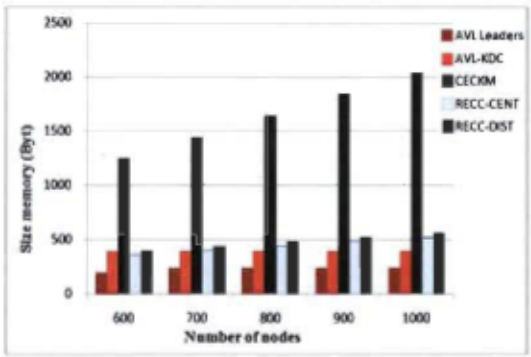



**Figure 5: the use of memory space.**

Figure 5 shows the size of memory used to store the keys of a normal node according to network size.

We can see that with CECKM method, to store keys, a normal node uses more memory (more than 1Kbytes), and a node in the AVL-Leaders method uses less memory (less than 250 bytes). With the RECC method, the number of stored keys is related to the number of neighbors and to the size of the network. This is why; we can see that from 700 nodes, RECC approaches use more memory than AVL-KDC approach.

The AVL-Leaders approach proposed offers storage savings compared to other approaches and our approach AVL-KDC saves more space than the two RECC approaches in large networks. Due to the fact that the number of stored keys is connected to the height of the AVL tree, the height in AVL-Leaders is smaller than in AVL_KDC. In the RECC method, the number of stored keys is related to the number of neighbors and in the CECKM method, the number of stored keys is related to the number of nodes in the cluster.

## 5.4 Comparison the memory space use

To calculate the secret key between two normal nodes with the CECKM method and the RECC-dist method, nodes compute the multiplication of their exchanged keys. This operation requires 282 bytes of RAM [19], but with our method, the nodes calculate multiplication of their positions with the shared key in the AVL tree, these operations require a maximum of 43 bytes of RAM [20].

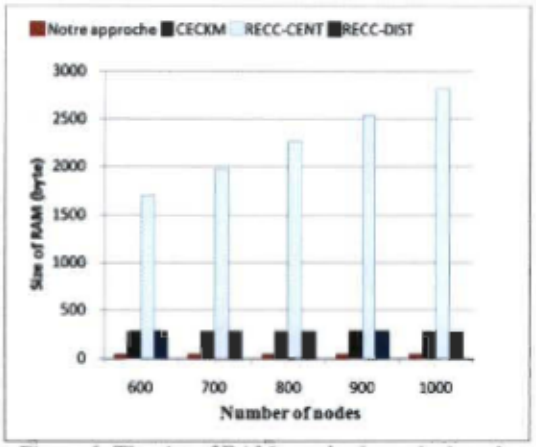



**Figure 6: The size of RAM required to calculate the secret key between two normal nodes.**

In the RECC-CENT approach, the leader computes the secret keys of each node. If a node has 7 neighbors, it uses 7 * 282 bytes or 1974 bytes of RAM, which is too much. If using a TELOSB node [21] that has a 10 Kbytes of RAM (TELOSB is the powerful sensor) the memory required to calculate the secret key is at least 19% of the RAM. However, our method uses only 0.4% of the RAM.

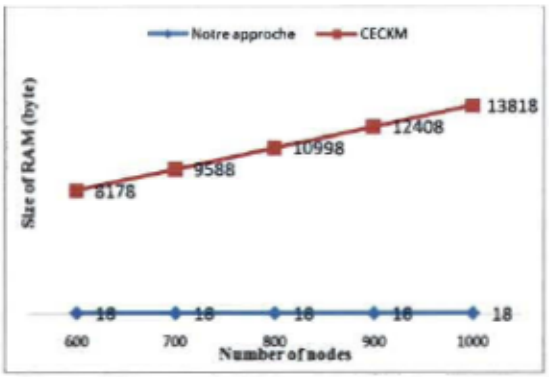



**Figure 7: The size of RAM required to calculate the secret key between a node and his leader**

With our method, to calculate the secret key between a leader and a normal node in the cluster, the secret key is equal to the multiplication of the position of the node in the "Member_List" list with the root key of the AVL tree. This operation requires a maximum of 18 bytes of the RAM, whereas in the CECKM method, a node must compute the multiplication operation of all public keys of all other nodes in the cluster. For a MicaZ node with 4 Kbytes of RAM, it is impossible to do such an operation, because with 4 bytes, we can hold a maximum of 14 keys multiplications. Thus, for a leader, if we use that TELOSB 10 Kbytes of RAM, it cannot compute the secret key if the cluster size exceeds 32 nodes.

Our method proves that whatever the size of the network, either for a normal node or a leader node, it is always possible to compute the secret keys and this transaction is a lightweight operation and causes no problem unlike the other methods.

## 5.5 Security Analysis

In both RECC approaches, after the deployment, the nodes evaluate their positions in the network and calculate the routing table. All these operations require a lot of communication between nodes and more time [22] before installing the encryption keys and it puts the network at risk, because an intruder can enter during this time in the network as a normal node.

Also in the CECKM method, keys exchanged between nodes and the leaders are not encrypted, allowing the intruder to capture a key and change it. In our method, prior to the deployment, all nodes are preloaded with a network key, and after the creation of the clusters, each cluster has its own key, and then, all communications are encrypted when deployment is started. This gives no chance for an intruder to change the keys.

In the RECC-DIST method, when a node is compromised, the nodes only revoke the shared key. The intruder can obtain public keys from neighbors and calculate the secret keys. In this case, it can play the role of other nodes that have shared secret keys with him.

In our approach, each node stores a set of public keys, and if a node is compromised, then all shared keys with other nodes will be changed, and the intruder cannot interact with the other nodes.

Our method ensures and guarantees greater security, which covers the entire network, once the nodes are deployed.

# 6. CONCLUSION

In this paper, we presented an efficient key management protocol for wireless sensor networks with a heterogeneous large-scale deployment. It is based on cryptography using elliptic curves (ECC) and the AVL tree. The approach requires a KDC server that generates the ECC keys, and an AVL tree for storing the keys. Each node is associated with a leaf of the tree, and it stores only the keys located along the path to the root node of the AVL tree. Our approach saves significant memory and reduces the number of packets exchanged during installation keys with fewer computing operations, while guaranteeing a better security.

# 7. ACKNOWLEDGMENTS

This work was completed with the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

# 8. REFERENCES

[1] Laurent Eschenauer, Virgil D. Gligor, " A Key-Management Scheme for Distributed Sensor Networks", CCS '02 Proceedings of the 9th ACM conference on Computer and communications security, ACM New York, NY, USA , 2002.

[2] W. Du, J. Deng, Y. S. Han, S. Chen, and P. K. Varshney, "A key management scheme for wireless sensor networks using deployment knowledge", 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004) Atlanta, GA, USA, ISBN 0-7803-8355-9, November 2004 .

[3] C.Castelluccia and Angelo Spognardi, " RoK: A robust key pre-distribution protocol for multi-stage Wireless Sensor Networks", IEEE Securecomm, Nice, France, September 2007.

[4] Donggang Liu, Peng Ning, "Establishing Pairwise Keys in Distributed Sensor Networks", Journal of ACM, October 27–31, 2003, Washington, DC, USA.

[5] R. Di Pietro, L. V. Mancini, and A. Mei, "Efficient and Resilient Key Discovery Based on Pseudo-Random Key Pre-Deployment, " in 18th International Parallel and Distributed Processing Symposium (IPDPS'04), p. 217, 2004.

[6] P.Samundiswary, Padma priyadarshini and P. Dananjayan, " Performance Evaluation of Heterogeneous Sensor Networks", International Conference on Future Computer and Communication, ISBN 978-0-7695-3591-3, ICFCC, 2009.

[7] Quazi Ehsanul Kabir Mamun and Sita Ramakrishnan, " SecCOSEn – A Key Management Scheme for Securing Chain Oriented Sensor Networks ", Communication Networks and Services Research Conference (CNSR 2008), ISBN 978-0-7695-3135-9, May 2008.

[8] A.S.Poornima, B.B.Amberker, "Tree-based Key Management Scheme for Heterogeneous Sensor Networks", 16th IEEE International Conference, ICON 2008, ISBN 978-1-4244-3805-1, NewDelhi, 2008.

[9] Yi-Ying Zhang, Wen-Cheng Yang, Kee-Bum Kim, Myong-Soon Park, "An AVL Tree-Based Dynamic Key Management in Hierarchical Wireless Sensor Network ", International Conference on Intelligent Information Hiding and Multimedia Signal Processing, ISBN 978-0-7695-3278-3, August 2008.

[10] AVL Tree in : Dictionnaires et Encyclopédies sur Academic, http://fr.academic.ru/dic.nsf/frwiki/124260.

[11] E.Munivel, Dr G M Ajit, "Efficient Public Key Infrastructure Implementation in Wireless Sensor Networks ", International conference on Wireless Communication and Sensor Computing, ISBN 978-1-4244-5136-4, February 2010.

[12] O.Arazil, H.Qi, D.Rosel, "A Public Key Cryptographic Method for Denial of Service Mitigation in Wireless Sensor Networks ", 4th Annual IEEE Communications Society, Conference on Sensor, Mesh and Ad Hoc Communications and Networks, ISBN 1-4244-1268-4 , San Diego, CA, August 2007.

[13] Xiaojiang Du, Mohsen Guizani, Yang Xiao, and Hsiao-Hwa Chen, "A Routing-Driven Elliptic Curve Cryptography Based Key Management Scheme for Heterogeneous Sensor Networks", IEEE International Conference on Communications, ICC 2007, ISBN 1-4244-0353-7 , Glasgow, August 2007.

[14] Hua-Yi Lin, " High-Effect Key Management Associated With Secure Data Transmission Approaches in Sensor Networks Using a Hierarchical-based Cluster Elliptic Curve Key Agreement", ncm, pp.308-314, Fifth International Joint Conference on INC, IMS and IDC, ISBN 978-0-7695-3769-6, Seoul, Korea, 2009.

[15] Marc Joye, " Introduction élémentaire à la théorie des courbes elliptiques, UCL Crypto Group Technical Report Series ", book in : http://www.dice.ucl.ac.be/crypto/, GC-1995/1, pages (17) (73) (29-52).

[16] A.Boukerche et al, " Vehicular ad-hoc networks: A new challenge for localization-Based System", Elsevier Computer Communications, 2007.

[17] Implementing AVL Trees in: http://www.lri.fr/~fiorenzi/Teaching/Cours_Ing2000/arbravl_7.pdf.

[18] M. J. Handy, M. Haase, D. Timmermann, " Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection", Fourth IEEE Conference on Mobile and Wireless Communications Networks, Stockholm, erschienen in Proceedings, S. 368-372, ISBN 0-7803-7606-4, World Scientific Publishing Co. Pte. Ltd., September 2002.

[19] Arvinderpa.S.W, Nils Gura,Hans Eberl, Vipul Gupta, sheuuling Chang shantz, " Energy analyses of public key cryptography for wireless sensor networks ", Third IEEE International Conference on Pervasive Computing and Communications, (PerCom'05), ISBN 0-7695-2299-8m, Kauai Island, Hawaii, 2005, page (5).

[20] Nils Gura, Arun Patel, Arvinderpal Wander, Hans Eberle, Sheueling Chang Shantz, " Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs ", in the book " Cryptographic Hardware and Embedded Systems - CHES 2004, 6th International Workshop Cambridge, MA, USA, August 11-13, 2004 ", ISBN 978-3-540-22666-6, Chaptre(4), page (8), Springer Berlin / Heidelberg, July 08, 2004.

[21] TELOSB mote platform in: http://www.memsic.com/products/wireless-sensor-networks/wireless-modules.html.

[22] Garci , a, E.M., Bermu , dez, A., Casado, R. , "Range-free localization for air-dropped WSNs by filtering node estimation improvements", 6th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), 2010 IEEE , E-ISBN : 978-1-4244-7741-8, November 2010.

## Contact person for correspondence:

Boucif Amar Bensaber, Ph.D
Laboratoire de Mathématiques et Informatique appliquées LAMIA.
Department of Mathematics and Computer Science
University of Quebec at Trois-Rivières, Canada
3351 Bd des Forges, C.P 500
Trois-Rivières, G9A 5H7 - Qc – Canada.
Email: Boucif.Amar.Bensaber@uqtr.ca