République ***A***lgérienne ***D***émocratique et ***P***opulaire

Ministère de l'***E***nseignement ***S***upérieur et de La ***R***echerche ***S***cientifique

Université ***M***ouloud ***M***AMMERI de ***T***izi-Ouzou

Faculté de ***G***énie ***E***lectrique et d'***I***nformatique

Département d'***I***nformatique

Mémoire

En vue de l'obtention du diplôme de Master en informatique
Option : Réseaux, mobilité et systèmes embarqués

Thème :

Sécurité par chiffrement dans les réseaux de capteurs sans-fil.

Proposé et dirigé par :

Mme. AOUDJIT R.

Réalisé par :

NAIT ALI Aghiles

Promotion : 2012/2013

Remerciements

Je tiens à remercier ma promotrice Madame AOUDJIT R

pour ses conseils, son soutien et la confiance qu'elle m'a accordée

pendant toute cette année.

Comme je remercie aussi les membres du jury pour avoir accepté d'évaluer notre travail.

Je tiens également à remercier mes parents et amis

qui m'ont soutenu et supportée pendant toute la réalisation de ce

travail de fin d'études.

Sommaire

Chapitre II : La sécurité dans les RCSFs

Chapitre III : Description des algorithmes DES, TripleDES et AES

Chapitre IV : Implémentation et évaluation

Table des figures

Chapitre I :

Chapitre II :

Chapitre III :

Liste des tableaux :

Introduction générale

Introduction générale

Les progrès technologiques réalisés ces dernières années dans le domaine des réseaux sans fil, de la micro-fabrication et des microprocesseurs embarqués ont permis la production de nouveaux types de capteurs dotés de moyens de communication sans fil, peu onéreux et pouvant être configurés pour former des réseaux à haute autonomie et à infrastructure non prédéfinie. Ensemble, ils forment un réseau de capteurs sans fil capable de surveiller une région ou un phénomène d'intérêt, de fournir des informations utiles par la combinaison des mesures prises par les différents capteurs, de les traiter pour les communiquer ensuite via le support sans fil à un ou plusieurs points de collecte.

Ces réseaux sont présents dans les applications militaires, médicales, environnementales domotiques, etc.… Ces applications ont souvent besoin d'un niveau de sécurité élevé. C'est pourquoi, il est primordial d'établir une communication sécurisée pour la majorité des applications des RCSF. L'un des principaux atouts est de déployer de l'algorithmique performante dans les nœuds capteurs pour assurer un niveau de sécurité acceptable et qui doit prendre en compte la gestion des ressources limitées disponibles au sein des capteurs sans fil tel que la mémoire et la puissance de calcul.

Dans ce contexte se situe notre travail, dans lequel nous avons essayé d'intégrer des algorithmes cryptographiques pour faire circuler les données dans un RCSF sans interférences malicieuses et avec un niveau de sécurité approprié. Là, l'objectif consiste donc à implémenter les algorithmes cryptographiques les plus connus (DES, TripleDES et AES) et d'évaluer leur performance selon leur consommation de la mémoire et leur rapidité d'exécution.

Pour relater les travaux réalisés dans le cadre de notre projet, notre mémoire s'organisera de la façon suivante :

Le premier chapitre présente des généralités sur les réseaux de capteurs sans fil avec une description de leurs architectures et leurs caractéristiques principales ainsi que leurs domaines d'application. Nous discuterons également les principaux facteurs et les contraintes influant leurs conceptions.

Dans le deuxième chapitre, nous présentons l'aspect de sécurité dans les réseaux de capteurs sans-fil. Premièrement, nous présentons les défis qui rendent la sécurité pour ce type de réseaux assez dure. Ensuite, nous présentons les besoins de sécurité requis par les protocoles, listons après les attaques qui menacent les RCSFs et les mécanismes utilisés pour les contrer. Nous discutons à la fin les différentes approches de gestion des clés dans de tels réseaux.

Dans le troisième chapitre on donne la description des algorithmes cryptographiques DES, TripleDES et AES.

Le quatrième chapitre est consacré a la partie implémentation et évaluation de la performance des algorithmes décris dans le chapitre trois.

Chapitre I

Généralités sur les RCSFs

I) Introduction :

Les avancées croissantes des technologies de miniaturisation des systèmes électromécaniques (MEMS), et le formidable progrès des communications sans fil ont permet de produire a un cout raisonnable de minuscules capteurs consommant peu d'énergie. Ces derniers sont capables de capter des données, calculer des informations à l'aide des valeurs collectées et les communiquer à travers un réseau de capteurs (wireless sensor networks), ces réseaux sont composés d'un nombre souvent très important de dispositifs appelés nœuds, qui sont soit posés a des endroits bien précis, soit dispersés aléatoirement et communiquant entre eux en utilisant des liaisons sans fil.

Dans ce chapitre, nous présenterons les réseaux de capteurs sans fil, leurs caractéristiques, les différences qui les distinguent des réseaux ad hoc traditionnels, leurs architectures de communication, leurs applications. Nous discuterons également les principaux facteurs et contraintes qui influencent la conception des réseaux de capteurs sans fil.

I.1) Réseaux Ad hoc :

Un réseau sans-fil ad hoc (ou MANET, pour Mobile Ad hoc NETwork) est formé par un ensemble d'hôtes qui s'organisent seuls et de manière totalement décentralisée, formant ainsi un réseau autonome et dynamique ne reposant sur aucune infrastructure filaire. Ces hôtes peuvent êtres fixes ou mobiles **[3]**.

Chaque terminal (hôte) est un nœud autonome équipé d'émetteur et de récepteur sans-fil utilisant des antennes qui peuvent être omnidirectionnelles (broadcast), fortement directionnelles (point à point) ou une combinaison des deux. Ces nœuds peuvent fonctionner comme terminaux et routeurs. En d'autres termes, en plus des capacités de traitement de base d'un terminal, les nœuds mobiles peuvent également exécuter des fonctions de commutation comme routeur. Un réseau Ad hoc peut avoir des passerelles ou des interfaces qui le relie à un réseaux fixe.

I.2) Les réseaux de capteurs sans fil :

I.2.1) Qu'est ce qu'un capteur (senseur) ? :

Un capteur sans-fil est un petit dispositif électronique avec peu de ressources (énergie, mémoire,…etc) capable d'accomplir trois taches complémentaires : le relevé d'une grandeur physique environnementale (température, humidité, vibration…etc), le traitement éventuel de cette information et la communication avec d'autres capteurs via des liaisons sans fil.

Il existe une foultitude de capteurs sur le marché, ces derniers possèdent en général une architecture matérielle identique (**Figure-1**) **[1]**, composée de:

- **Une unité de capture :** elle est chargée de numériser une grandeur physique récoltée sur l'environnement ;
- **Une unité de traitement :** elle est dotée d'un processeur et d'une mémoire ;
- **Unité d'énergie :** c'est la batterie qui, n'est généralement ni rechargeable ni remplaçable. La capacité d'énergie limitée au niveau des capteurs représente la contrainte principale lors de conception de protocoles pour les réseaux de capteurs. Les unités d'énergie peuvent être supportées par des photopiles qui permettent de convertir l'énergie lumineuse en courant électrique.
- **Une unité de transmission sans fil :** effectue toutes les transmissions ou réceptions de données sur le médium sans fil, elle peut être de type radiofréquence (RF) ou de type optique;

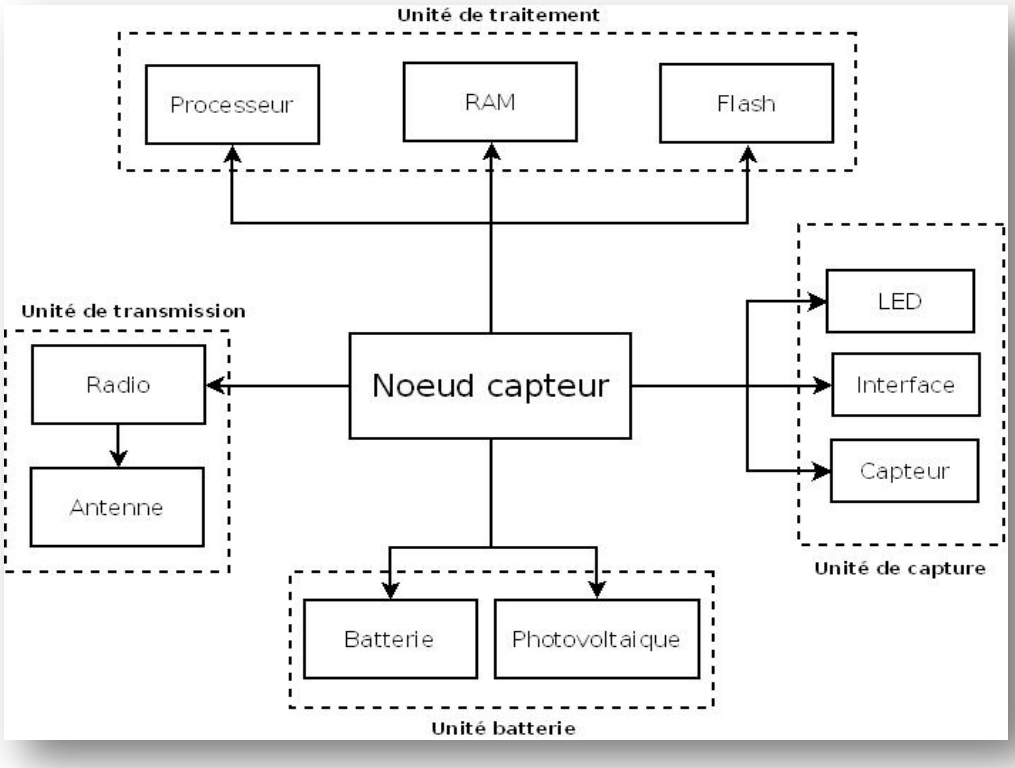


Figure-1 : Architecture d'un nœud capteur [1].

On peut également avoir d'autres unités suivant le domaine d'application. Parmi ces unités, un capteur peut contenir : un générateur d'énergie, un système de localisation GPS (Global Positioning System) et une unité assurant la mobilité.

I.2.2) Réseaux de capteurs sans fil :

Les réseaux de capteurs sans-fil sont considérés comme un type spécial des réseaux ad hoc ou l'infrastructure fixe de communication et l'administration centralisée sont absentes et les nœuds jouent, à la fois, le rôle des hôtes et des routeurs **[2]**. En général ces réseaux sont composés d'un grand nombre de nœuds déployés plus au moins aléatoirement, Chacun de ces nœuds à la capacité de collecter des données et de les transférer au nœud passerelle (dit "sink" en anglais ou puits) par l'intermédiaire d'une architecture multi-sauts. Le puits transmet ensuite ces données par Internet ou par satellite à l'ordinateur central «Gestionnaire de taches» pour les analyser et prendre des décisions.

Les réseaux de capteurs sans fil sont donc apparentés aux réseaux Ad Hoc. En effet, ces deux types de réseaux ont plusieurs points communs (Réseaux sans infrastructure, Architecture décentralisée, Autonomie, auto-configurabilité et spontanéité, Utilisation des ondes radio pour communiquer…). Cependant, les besoins et les contraintes de ces réseaux diffèrent. Les différences sont montrées dans la table suivante.

	Capteurs	**Ad Hoc**
Composants	Capteurs	Portables, PDAs, ...
Objectif	Ciblé	Générique / Communication
Contrainte clé	Energie	Débit
Communication	Broadcast	Point à point
Flot de communication	« many to one »	« any to any »
nœud central	Sink, Station de Base	Aucun
Relation entre les nœuds	Collaboration	Chaque nœud à son objectif
Taille	Très grande	Peu de nœuds
Mobilité	Très faible	forte mobilité

Tab-1 : Différences entre réseaux de capteurs et réseaux Ad Hoc. [7]

I.3) Caractéristiques des RCSF :

Les principales caractéristiques des réseaux de capteurs se résument dans ce qui suit :

I.3.1) La tolérance aux fautes :

La tolérance aux fautes est la capacité d'éviter la faille totale du système malgré la présence de panne sur un ou plusieurs capteurs du réseau, cette panne peut être causée par le manque d'énergie, un problème physique ou une interférence de l'environnement. La tolérance aux pannes est d'autant meilleure que le nombre de composants en panne est grand.

I.3.2) Scalabilité :

Le nombre de nœuds déployés pour un projet est de l'ordre d'une centaine voire des milliers de nœuds capteurs en fonction de l'application. Ce nombre peut même atteindre le million. Un nombre aussi important de nœuds engendre beaucoup de transmissions inter nodales et nécessite que le puits "sink " soit équipé de beaucoup de mémoire pour stocker les informations reçues.

I.3.3) Topologie dynamique :

Des centaines ou plusieurs centaines de nœuds sont déployés à travers un champ senseur (capteurs), ces nœuds peuvent être attachés à des objets mobiles, certain nœuds peuvent être ajoutés au réseau et d'autre peuvent tombés en panne, ce qui rend la topologie instable (dynamique).

I.3.4) Auto-organisation :

L'auto organisation s'avère très nécessaire pour ce type de réseau afin de garantir sa maintenance. Vu les différentes conséquences résultant de l'instabilité de la topologie du réseau de capteur, ce dernier devra être capable de s'auto-organiser pour continuer ses applications.

I.4) Domaines d'applications des RCSF :

Le développement des RCSFs a été suscité par des besoins militaires. En effet, les armées souhaitent être en mesure d'espionner discrètement leurs ennemis **[3]**. la diminution de taille et du cout des capteurs, ainsi que l'élargissement des gammes de capteurs disponibles (mouvement, température, vibration ...) et l'évolution des supports de communication sans fil ont élargi le champ d'application des réseaux de capteurs, Actuellement on trouve des applications pour la détection et la surveillance des désastres, le contrôle de l'environnement, l'agriculture de précision, la surveillance et la maintenance préventive des machines, la médecine et la santé, …etc.

I.4.1) Applications militaires :

Comme évoqué précédemment, les réseaux de capteurs sans fil proviennent principalement de la recherche militaire. Le déploiement rapide, le coût réduit, l'auto-organisation et la tolérance aux pannes des réseaux de capteurs sont des caractéristiques qui rendent ce type de réseaux un outil appréciable dans un tel domaine.

Comme exemple d'application dans ce domaine, on peut penser à un réseau de capteurs déployé sur un endroit stratégique ou difficile d'accès, afin de surveiller toutes les activités des forces ennemies, ou d'analyser le terrain avant d'y envoyer des troupes (détection d'agents chimiques, biologiques ou de radiations) **[5]**.

I.4.2) Application liés à la sécurité :

L'intégration des capteurs dans de grandes structures telles que les ponts ou les bâtiments aidera à détecter les fissures et les altérations dans la structure suite à un séisme ou au vieillissement de la structure.

Les différentes applications des réseaux de capteurs dans la sécurité pourraient diminuer considérablement les dépenses financières consacrées à la sécurisation des lieux et à la protection des êtres humains tout en garantissant de meilleurs résultats.

I.4.3) Application environnementales :

Le contrôle des paramètres environnementaux par les réseaux de capteurs peut donner naissance à plusieurs applications. Par exemple, le déploiement des thermo-capteurs dans une forêt peut aider à détecter un éventuel début de feu et par suite faciliter la lutte contre les feux de forêt avant leur propagation. Le déploiement des capteurs chimiques dans les milieux urbains peut aider à détecter la pollution et analyser la qualité d'air. De même leur déploiement dans les sites industriels empêche les risques industriels tels que la fuite de produits toxiques (gaz, produits chimiques, éléments radioactifs, pétrole, etc.).

Dans le domaine de l'agriculture, les capteurs peuvent être utilisés pour réagir convenablement aux changements climatiques par exemple le processus d'irrigation lors de la détection de zones sèches dans un champ agricole. **[6]**

I.4.4) Application médicales :

La surveillance des fonctions vitales d'un organisme vivant peut être facilitée par des micro-capteurs avalés ou implantés sous la peau. Des gélules sous forme de micro-capteurs ou de micro-caméras pouvant être avalées existent déjà et permettent, sans recours à la chirurgie, de transmettre des images depuis l'intérieur d'un corps humain. Une récente étude présente des capteurs pouvant fonctionner dans le corps humain pour le traitement de certaines maladies.

La surveillance de la glycémie, la surveillance des organes vitaux ou la détection précoce de certains cancers sont des applications biomédicales envisagées. Des réseaux de

capteurs permettraient également une surveillance temps réel des maternités (détection de vol de bébés) ou des patients. **[7]**

I.4.5) Application commerciales :

Il est possible d'intégrer des nœuds capteurs au processus de stockage et de livraison. Le réseau ainsi formé, pourra être utilisé pour connaitre la position, l'état et la direction d'un paquet ou d'une cargaison. Il devient alors possible pour un client qui attend la réception d'un paquet, d'avoir un avis de livraison en temps réel et de connaitre la position actuel du paquet. Pour les entreprise manufacturière, les réseaux de capteurs permettront de suivre le procédé de production a partir des matières premières jusqu'au produit final livré. **[8]**

I.5) Déploiement d'un RCSF :

Les capteurs sont au préalable déployés sur une zone à surveiller. Pour satisfaire de nouvelles contraintes ou pour pallier des pannes, un déploiement des nœuds supplémentaires, dit *itératif*, peut être requis. Différents modes de déploiement sont envisageables et dépendent essentiellement de l'application de surveillance. Une fois déployés, on suppose que les capteurs sont statiques **[11]**.

I.5.1) Phase de pré-déploiement et de déploiement :

Les nœuds capteurs peuvent être éparpillés sur le champ de captage en masse ou placés d'une manière individuelle et ceci par le biais de plusieurs moyen tel que :

- les jeter d'un avion

- utiliser une artillerie, roquette ou missile, ou

- les placer nœud par nœud d'une façon manuelle ou en utilisant des robots.

Le nombre important de nœuds utilisés dans un réseau de capteurs empêche leur déploiement suivant un plan soigneusement établi, cependant un schéma général pour le déploiement initial doit être conçu pour permettre :

- de réduire les coûts d'installation

- augmenter la flexibilité d'arrangement des nœuds

- faciliter l'auto-organisation des nœuds et leur tolérance aux pannes

I.5.2) Phase de post-déploiement :

Après la phase de déploiement, la topologie du réseau peut subir des changements dues aux :

- changement de position des nœuds
- accessibilité à cause du brouillage ou des obstacles en mouvements
- épuisement d'énergie
- mal fonctionnement des nœuds ou
- des besoins pour leur application.

En effet, Bien que les nœuds d'un réseau de capteurs puissent être déployés d'une manière statique, la panne matérielle constitue un évènement très commun à cause de l'épuisement d'énergie ou la destruction. Il est possible également d'avoir un réseau de capteur avec des nœuds mobiles qui ont une mobilité très élevée. Par conséquent, la topologie du réseau de capteur est exposée fréquemment aux changements après la phase de déploiement.

I.5.3) Phase de redéploiement des nouveaux nœuds :

Des nœuds capteurs additionnels peuvent être installés pour remplacer ceux qui sont en panne ou bien pour répondre aux besoins des tâches assignées au réseau. Cette addition entraîne la réorganisation du réseau et le changement de sa topologie.

Une bonne gestion du réseau, faisant face au facteur de changement fréquent de la topologie d'un réseau ad hoc caractérisé par une contrainte exigeante de consommation d'énergie doit passer obligatoirement par la conception des protocoles de routages spéciaux.

I.6) Architecture des réseaux de capteurs :

Un réseau de capteurs sans-fil (RCSF) est un ensemble de capteurs variant de quelques dizaines d'éléments à plusieurs milliers, déployés (aléatoirement) dans une zone géographique appelée zone de captage ou zone d'intérêt afin de surveiller un phénomène physique et récolter leurs données de manière autonome. Les nœuds capteurs utilisent une communication sans fil pour acheminer les données captées vers le nœud puits (station de base), voir figure 2. Ce dernier est considéré comme un point de collecte et peut transférer les données collectées via internet ou satellite à un ordinateur central « gestionnaire de taches » pour leur traitement. De plus, des requêtes précisant le type de données requises et le début/arrêt de captage peuvent êtres envoyées par le biais du nœud puits aux autres nœuds du réseau (un à plusieurs).

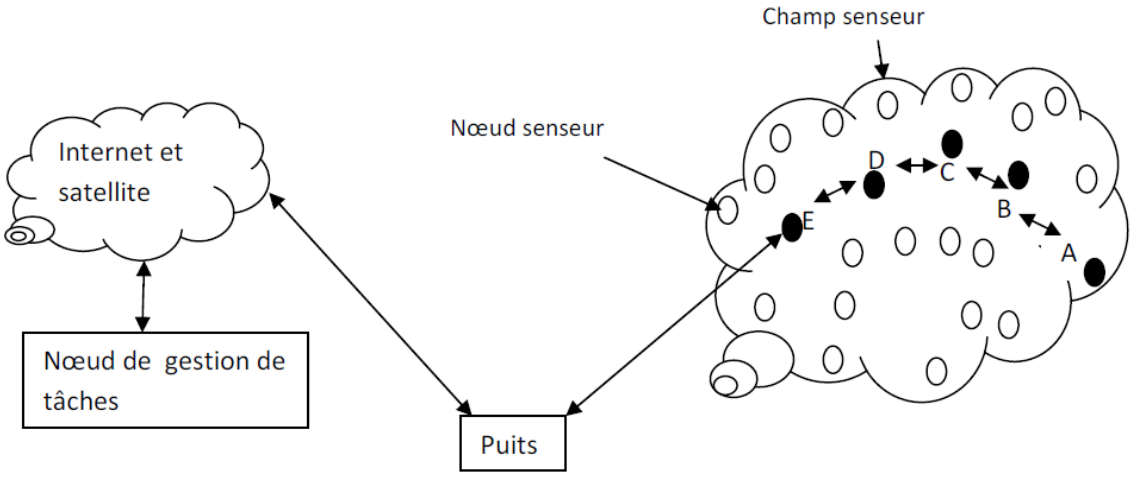


Figure-2 : Architecture d'un réseau de capteurs sans-fil.

I.6.1) Topologies du réseau :

Pour prendre en charge le transfert de données depuis les nœuds capteurs jusqu'au nœud puits, le réseau peut être configuré selon différentes topologies **[12]**.

I.6.1.1) La topologie en étoile (Star Network) :

C'est une topologie simple (représentée par la figure 3), où chaque nœud communique ses mesures directement à l'unique station de base sans avoir la possibilité d'échanger des messages avec les autres capteurs. La station de base est la seule à pouvoir recevoir ou envoyer des messages aux nœuds du réseau. Cette approche peut de manière significative simplifier la conception, car les soucis de gestion du réseau sont réduits au minimum. Néanmoins, elle présente des limites en terme : de scalabilité et de robustesse, due à sa dépendance sur un seul nœud pour contrôler et gérer le réseau.

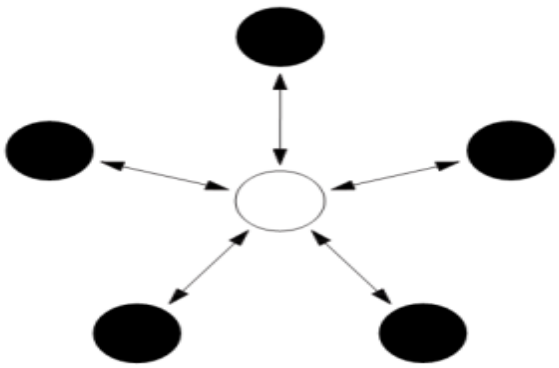


Figure-3 : La topologie en étoile.

I.6.1.2) La topologie maillée (Mesh Network) :

Dans ce type de topologie, une communication multi sauts est utilisée comme mode d'acheminement de messages, où n'importe quel nœud peut envoyer à n'importe quel autre nœud dans le réseau a condition qu'il soit dans sa portée de transmission, sinon un nœud intermédiaire intervient pour envoyer le message au nœud destinataire (voir Figure 4). L'avantage de cette topologie est la possibilité du passage à l'échelle et la tolérance aux pannes, par contre la consommation d'énergie dans la communication multi sauts et les latences crées sont les inconvénients majeurs de cette topologie.

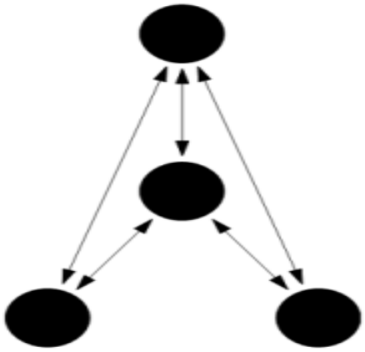


Figure-4 : La topologie maillée.

I.6.1.3) La topologie hybride (Hybrid Star – Mesh Network) :

Pour minimiser la consommation d'énergie dans les réseaux de capteurs, une topologie hybride entre celle en étoile et en mesh est conçue (voir figure 5). Elle fournit des communications réseau robustes et diverses. Il existe des nœuds qui ont la possibilité de router les messages des autres nœuds à faible puissance. Généralement, ces nœuds à capacité multi sauts ont une puissance plus élevée que les autres nœuds du réseau.

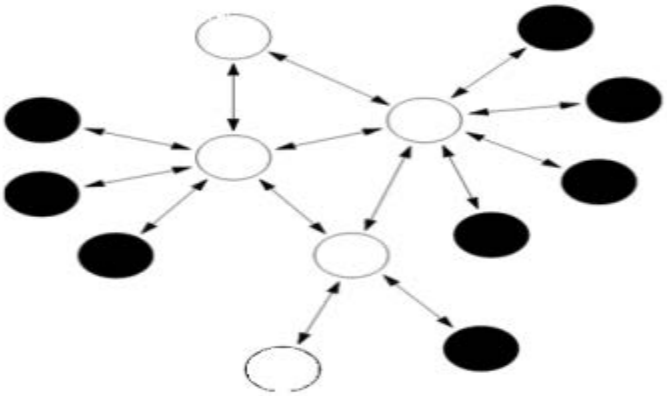


Figure-5 : La topologie hybride.

I.7) Facteurs de conception des RCSFs :

Dans ce qui suit, on présentera certains facteurs de conception de réseau de capteurs sans fil, Ces facteurs représentent la base de la conception de protocoles ou d'algorithmes pour les réseaux de capteurs.

I.7.1) La tolérance aux pannes :

Certains nœuds peuvent être défaillants ou inhibés à cause d'un épuisement d'énergie, ou subissent des dommages physiques. Et pour que les pannes n'affectent pas la tâche première du réseau, Il faut mettre en place des mécanismes qui permettent d'augmenter le taux de disponibilité d'un réseau de capteurs.

La tolérance aux fautes est la capacité de maintenir les fonctionnalités du réseau sans interruptions dues à une erreur intervenue sur un ou plusieurs capteurs **[4]**.

I.7.2) Topologie du réseau :

En raison de la forte densité des capteurs dans la zone à observer, il faut prendre des précautions de maintenance de la topologie qui change tous le temps. La conception d'un protocole d'auto-organisation qui s'adapte continuellement et rapidement aux changements, s'avère nécessaire pour assurer le bon fonctionnement du réseau.

I.7.3) Cout de production :

Le coût d'un seul nœud est très important pour justifier le coût global du réseau, et pour pouvoir concurrencer un réseau de surveillance traditionnel, il faut que le coût de fabrication de ces nouds soit tel que le coût global du réseau ne soit pas supérieur à celui d'un réseau classique afin de pouvoir justifier son intérêt.

I.7.4) L'échelle :

Le nombre de nœuds déployés pour un projet peut atteindre le million. De nouveaux schémas travaillent avec ce nombre de nœuds exploitant la nature fortement dense des réseaux de capteurs pour garantir un bon fonctionnement de ce dernier.

I.7.5) Consommation d'énergie :

La consommation d'énergie est une métrique de performance très importante dans les réseaux de capteurs sans fil, et elle doit être un objectif évident d'optimisation. Sachant que dans de nombreux scénarios, les nœuds capteurs devront compter sur un approvisionnement limité en énergie (la batterie). La recharge de la batterie est souvent trop coûteuse et parfois impossible. Il faut donc que les capteurs économisent au maximum l'énergie afin de pouvoir fonctionner.

I.7.6) Environnement :

Les nœuds capteurs doivent être conçus d'une manière à résister aux différentes et sévères conditions de l'environnement : forte chaleur, Pluit, humidité…

I.7.7) Médium de transmission :

Dans un réseau senseur multi sauts, les nœuds communicants sont chaînés par un moyen sans fil. Ces liens peuvent êtres réalisés par radio, infrarouge ou par un moyen optique.

I.7.8) Contraintes matérielles :

Ces contraintes sont liées aux capacités matérielles et physiques d'un nœud capteur, ce qui représente un handicap pour les besoins en sécurité qui nécessite en général des ressources additionnelles.

I.8) Communication dans les RCSF :

I.8.1) Modèle en couches (Pile protocolaire) :

Le rôle de ce modèle consiste à standardiser la communication entre les composants du réseau afin que différents constructeurs puissent mettre au point des produits (logiciels ou matériels) compatible. Par analogie au modèle OSI (Open Système Interconnexion) des réseaux filaires, le modèle de communication utilisé dans les réseaux de capteurs comprend cinq couches qui ont les mêmes fonctions que celles du modèle OSI. Comme la communication n'est pas le seul souci dans les réseaux de capteurs, il y a d'autres critères très importants qu'il faut en tenir compte. De ce fait, d'autres couches supplémentaires sont ajoutées pour gérer l'énergie, la mobilité des nœuds et l'ordonnancement des tâches.

La pile de protocoles utilisée par le puits (*Sink*) ainsi que par tous les nœuds capteurs est donnée dans la Figure 6. Cette pile de protocoles combine routage et gestion d'énergie et

intègre les données avec les protocoles réseau. Elle communique de manière efficace (en termes d'énergie) à travers le support sans-fil et favorise les efforts de coopération entre les nœuds-capteurs **[13]**.

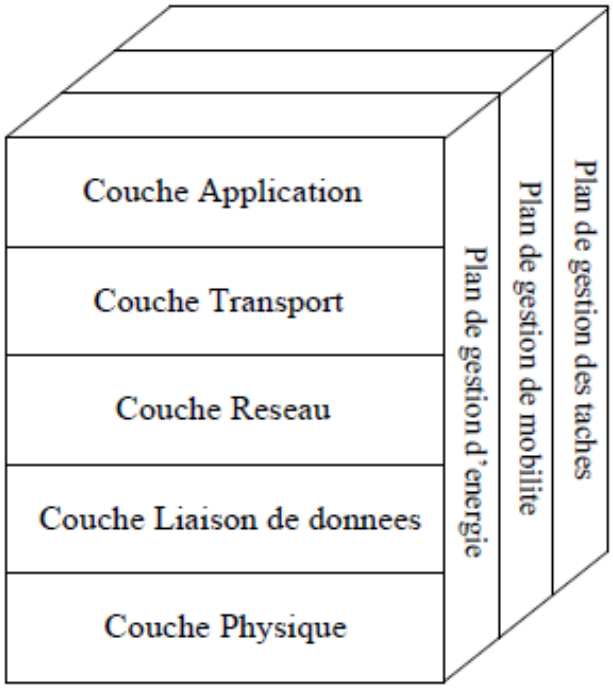


Figure-6 : La pile protocolaire des réseaux de capteurs.

Comme illustrer dans la figure 5. La pile consiste en couches application, transport, réseau, liaison de données, physique et trois plans de gestion : d'énergie, de mobilité et de tâche.

I.8.1.1) Rôles des couches :

Dans ce qui suit on donnera les rôles est les fonctions des différentes couches qui constituent la pile protocolaire.

1. **La couche physique :**

La couche physique est responsable de :

- ✓ La sélection des fréquences.
- ✓ La génération des ondes porteuses.
- ✓ La détection du signal.
- ✓ La modulation.

2. Couche liaison de données :

Cette couche est principalement responsable du multiplexage du flux de données, de la détection et le verrouillage des trames de données, du contrôle d'erreurs, du contrôle d'accès au media MAC (Medium Access Control). Elle assure de manière fiable les connexions point à point et point à multipoint selon la topologie du réseau. Cependant, les protocoles MAC conventionnels ne sont pas directement applicables dans les réseaux de capteurs. Cette couche, exige l'utilisation de protocoles qui doivent être capables : de réduire au minimum la collision avec l'émission des voisins et de minimiser les retransmissions.

3. Couche réseau :

Cette couche permet de gèrer l'adressage et le routage des données. Elle établit les routes entre les nœuds capteurs et le nœud puits et sélectionne le meilleur chemin en terme d'énergie, délai de transmission, débit, etc. Les caractéristiques spécifiques aux RCSFs exigent que leurs protocoles de routage différent de ceux des réseaux ad hoc traditionnel, notamment dans l'absence d'un adressage global, et la priorité absolu donnée à la qualité d'énergie consommée par les nœuds.

En effet, la conception de la couche réseau dans un réseau de capteurs doit être guidée par les principes suivants :

- l'efficacité en consommation d'énergie est une considération prioritaire.
- Tous les protocoles dans les réseaux de capteurs prennent en charge les données d'application qui circulent dans le réseau (Data centric).
- L'agrégation des données est utile quand elle ne cache pas l'effort collaboratif des nœuds capteurs.
- Un réseau de capteur idéal garantit un adressage basé-attributs et ses nœuds sont conscients de leur localisation.

4. Couche transport :

Cette couche constitue une interface entre la couche application et la couche réseau. Elle est responsable : du transport de données, leur découpage en paquets, contrôle de flux, conservation de l'ordre des paquets et de la gestion des éventuelles erreurs de transmission.

Afin qu'un réseau de capteurs sans-fil puisse interagir avec un autre réseau tel qu'Internet, une connexion TCP ou UDP peut être nécessaire. Cependant toutes les communications entre la station de base et les nœuds capteurs doivent utiliser un protocole de type UDP en raison des limitations en ressources des nœuds capteurs.

5. **Couche application :**

La couche application constitue l'ensemble des applications implémentées sur un réseau de capteurs. Ces applications devraient fournir des mécanismes permettant à l'utilisateur d'interagir avec le réseau de capteurs a travers différentes interfaces, et éventuellement, par l'intermédiaire d'un réseau étendue (Internet par exemple). Elle permet également de rendre transparents les mécanismes de communication dans les couches inférieurs. Tout ceci, en offrant des interfaces pour la création et la diffusion de requêtes. Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par logiciels.

I.8.1.2) Plans de gestions :

Les plans de gestions aident les nœuds capteurs à coordonner la tâche de captage et minimiser la consommation d'énergie. Ils sont donc nécessaires pour que les nœuds capteurs puissent collaborer ensemble, acheminer les données dans un réseau mobile et partager les ressources entre eux en utilisant efficacement l'énergie disponible. Ainsi, le réseau peut prolonger sa durée de vie.

1. **Plan de gestion de l'énergie :**

Le plan de gestion de l'énergie doit gérer la manière dont les nœuds senseurs utilisent leurs énergies. Par exemple : éteindre son interface de réception dés qu'il reçoit un message d'un nœud voisin afin d'éviter la réception des messages dupliqués. Du plus, quand le niveau d'énergie d'un nœud senseur est faible, il peut diffuser un message aux autres capteurs pour ne pas participer au taches de routage, et conserve le reste d'énergie aux fonctionnalités de détections.

2. **Plan de gestion de mobilité :**

Il permet la détection et l'enregistrement des mouvements des nœuds capteurs afin de maintenir des informations sur leurs localisations et d'entretenir continuellement une route vers l'utilisateur final. Dans plusieurs cas les nœuds capteurs peuvent être mobiles suite à une configuration d'une nouvelle topologie ou bien par un changement d'emplacement désiré par l'utilisateur. Le système de gestion de mobilité doit être capable de commander les nœuds pour réaliser les mouvements nécessaires.

3. **Plan de gestion de tâche :**

Lors d'une opération de capture dans une région donnée, les nœuds composant le réseau ne doivent pas obligatoirement travailler avec le même rythme. Cela dépend essentiellement de la nature du capteur, son niveau d'énergie et la région dans laquelle il a été déployé. Pour cela, le niveau de gestion des tâches assure l'équilibrage et la

distribution des tâches sur les différents nœuds du réseau afin d'assurer un travail coopératif et efficace en matière de consommation d'énergie, et par conséquent, prolonger la durée de vie du réseau

I.8.2) Les standards de communication sans-fil :

Il ya une multitude de normes comme le WiFi (la norme IEEE 802.11)**[8]** et le WiMax (la norme IEEE 802.16)**[8]** qui s'adressent au transport de données a haut débit. Mais ces normes ne sont pas adaptées pour être utilisés dans ce type de réseau, car les capteurs n'ont pas besoins d'une largeur de bande très élevé, mais plutôt d'un temps de latence faible ainsi qu'une consommation d'énergie très basse, pour une longue durée de vie sur batterie et un grand nombres de dispositifs. D'où la nécessité de concevoir d'autres normes sans-fil capables de rependre a ces exigences. Parmi les standard les plus apte à être exploiter dans les RCSF on trouve :

I.8.2.1) La technologie Bluetooth :

Elle est connue aussi sous le nom de la norme IEEE 802.15.1, elle a été lancée par Ericsson en 1994, proposant un débit théorique de 1Mbps lui permettant une transmission de la voie, des données et des images, d'une portée maximales d'une trentaine de mètres. Bluetooth est une technologie peut onéreuse grâce a sa forte d'intégration sur une puce unique de 9 mm sur 9 mm ; elle possède également l'avantage de fonctionner sur des appareilles a faibles puissances d'où une faible consommation d'énergie **[8]**. Cependant cette technologie soufre d'un problème de passage a l'échelle.

I.8.2.2) ZigBee :

Le Zigbee est un protocole de haut niveau permettant la communication de petites radios, à consommation réduite, basée sur la norme IEEE 802.15.4 pour les réseaux à dimension personnelle (WPAN). Cette technologie a pour but la communication à courte distance, à la manière du Bluetooth, mais de façon plus simple et moins chère. Ce protocole est relativement lent. Mais ses deux principaux avantages sont sa faible consommation et sa fiabilité élevée. Le petit débit qu'ils offrent (250 Kbps théorique contre 1Mps pour le Bluetooth) n'est pas vraiment un handicap pour un réseau de capteurs puisque la taille des paquets échangés n'est pas vraiment importante.

- **I-8-2-2-1) Caractéristiques de ZigBee :**

Les objectifs visés par ZigBee peuvent être résumés dans les points suivants **[4]**

- Usage sans restrictions géographiques
- Pénétration à travers les murs et plafonds
- Installation automatique/semi-automatique
- Possibilité de rajouter/retirer des dispositifs
- Coût avantageux
- Débit : 10kbps-115.2kbps
- Portée radio: 10-75m
- Jusqu'à 65k nœuds par réseau
- Jusqu'à 100 réseaux Co-localisés
- Jusqu'à 2 ans de durée de vie de batterie standards Alcaline

I.8.2.3) Infrarouge :

Permet de créer des liaisons sans fil de quelque mètres avec des débits pouvant monter à quelques mégabits par seconde, cette technologie est largement utiliser dans la domotique (télécommande), elle soufre toutefois des perturbations dues aux interférences lumineuses.

I.9) Modèles de transmission de données dans les RCSF :

La transmission de données dans les réseaux de capteurs peut se faire suivant plusieurs modèles dont on distingue trois essentielles :

- Modèle Event Driven.
- Modèle Query Driven.
- Modèle Contiuous.

A- Modèle Driven Event :

La génération et la transmission des paquets de données sont commandées par la réalisation d'un évènement. La plupart des applications *Driven Event* sont des applications intolérantes aux délais (temps réel), critiques, interactives et de non bout-en-bout. En fait, au lieu d'avoir un nœud émetteur et un autre récepteur de l'information, on trouve un nœud récepteur (le nœud de contrôle *sink*) et un groupe de nœuds capteurs, se trouvant proche de l'événement, qui sont tous des émetteurs de la même information. La réussite de ces applications pour ce modèle, repose essentiellement sur la détection de l'évènement et la rapidité des prises des réactions nécessaires pour assurer l'aspect temps réel des applications. L'inconvénient majeur de ce modèle est la redondance des données. En fait, les nœuds excités par le même événement envoient la même information au nœud de contrôle sink. Pour cela, un protocole de routage basé sur la négociation des données est recommandé.

B- Modèle Query Driven :

Le modèle Query Driven est semblable au modèle Driven Event sauf que la collecte des informations sur l'état de l'environnement est initié par des interrogations envoyées par le nœud de contrôle sink, alors que pour le modèle précédent, elle est déclenchée suite a un événement détecté. La plupart des applications Query Driven sont des applications interactives, critique, de non bout-en-bout et leur tolérance aux délais dépond de l'urgence de l'interrogation. Notons que le modèle Query Driven peut être utilisé pour contrôler et reconfigurer les nœuds. Par exemple, le sink peut envoyer des commendes au lieu des interrogations pour modifier le programme d'un nœud capteur, modifier son taux de trafique ou son rôle. Seul le nœud capteur jouant le rôle de sink est autorisé d'émettre des demandes d'interrogations ou des commandes et ce pour assurer l'ordre et l'hiérarchie de réseau de capteur.

C- Modèle Contiuous :

Dans le modèle continu, les nœuds capteurs envoient les informations d'une manière continue au nœud sink suivant un volume de trafic prédéterminé.

I.10) Différentes problématiques dans les réseaux de capteurs :

Les recherches dans le domaine des réseaux de capteurs ont révélé plusieurs problématiques, parmi ces problématiques, nous citons **[6]** :

• **Routage :** concevoir un protocole de routage performant en termes de minimisation de la consommation de l'énergie, du choix des routes optimales pour l'acheminement de l'information d'un capteur à la station de base et vice versa, de réduction du délai de délivrance des paquets...Ainsi le réseau doit passer à l'échelle sans que ses performances se dégradent.

• **Couche MAC :** la spécificité des réseaux de capteurs sans fil mobiles nécessite le développement de nouveaux protocoles MAC qui s'adaptent aux contraintes imposées par ces réseaux. Ceci dans le but d'améliorer le débit, minimiser la consommation d'énergie, optimiser le partage du médium ainsi que minimiser le délai de délivrance des paquets.

• **Qualité de service :** des protocoles au niveau de la couche MAC devraient être capables d'établir des priorités entre les flux, limiter les pertes de paquets vitaux pour la gestion du réseau, ou du moins en restreindre l'impact.

• **Cross-layer :** dans les modèles classiques, les concepteurs essaient d'optimiser les performances au niveau d'une couche indépendamment des autres couches. Par exemple, le routage et les fonctions de la couche MAC sont optimisés indépendamment les uns des autres. Cependant, une telle indépendance est communément considérée comme trop onéreuse pour les réseaux de capteurs. Par conséquent, une coopération permettant un compromis entre performances, dépendance et flexibilité doit être proposée pour optimiser les capacités du réseau en général.

• **Diffusion de l'information :** les protocoles de diffusion conçus pour les réseaux de capteurs doivent tenir compte de leurs spécificités ainsi que de leurs contraintes intrinsèques imposées. Ainsi, pour concevoir un protocole efficace, il faudrait assurer une couverture maximale des capteurs composant le réseau (taux d'accessibilité supérieur 90%), minimiser le nombre de réémetteurs et des réceptions redondantes ainsi que la consommation d'énergie.

• **Sécurité :** pour les applications qui exigent un niveau de sécurité assez élevé telles que les applications militaires, des mécanismes d'authentification, de confidentialité, et d'intégrité doivent être mis en place au sein de leur communauté. Les algorithmes de cryptographie conçus pour les réseaux de capteurs doivent tenir compte des ressources limitées que présentent ces réseaux.

I.11) Conclusion :

Les réseaux de capteurs sans-fil possèdent des caractéristiques qui les différencient des autres types de réseaux sans-fil. Ces spécificités telle que la consommation d'énergie réduite, la scalabilité ou le routage incitent le besoin de concevoir de nouveaux protocoles d'accès au support, de routage, de sécurité, de transport ou d'application qui s'adapteront aux caractéristiques des RCSFs.

Dans ce chapitre nous avons vu les concepts généraux liés aux réseaux de capteurs sans fil. Parmi ces concept nous avant identifié les facteurs de conception des RCSFs. Un point sur les protocoles de communication sans fil (Bluetooth, ZigBee, infrarouge) a été abordé. La compréhension de ces concepts est nécessaire pour les concepteurs des protocoles et application destinées aux RCSFs.

Chapitre II

La sécurité dans les RCSFs

II) Introduction :

Les différents domaines d'application des réseaux de capteurs (militaire, médicale, …) ont souvent besoins d'un niveau de sécurité élevé. Or, de part des caractéristiques des RCSFs (absence d'infrastructure, contrainte d'énergie, topologie dynamique, nombre important de capteurs, sécurité physique limitée, capacité réduite des nœuds, ...), la sécurisation des réseaux de capteurs est à la source, aujourd'hui, de beaucoup de défis scientifiques et techniques.

Dans ce chapitre, nous présentons un aperçu sur la sécurité dans les réseaux de capteurs sans fil. Premièrement, nous présentons les défis qui rendent la sécurité pour ce type de réseaux assez dure. Ensuite, nous présentons les besoins de sécurité requis par les protocoles, listons après les attaques qui menacent les RCSFs et les mécanismes utilisés pour les contrer. Nous discutons à la fin les différentes approches de gestion des clés dans de tels réseaux.

II.1) Les défis de sécurité dans les RCSFs:

La sécurisation des réseaux de capteurs est un problème difficile pour les raisons suivantes:

- **Capacités limitées :** Les ressources de calcul et de mémoire des nœuds sont relativement faibles. l'énergie est peut-être la contrainte la plus forte aux capacités d'un nœud capteur. Le plus grand des défis dans le domaine des réseaux de capteurs reste de concevoir des protocoles, entre autre de sécurité, qui minimisent l'énergie afin de maximiser la durée de vie du réseau. En d'autres mots, l'énergie est sans aucun doute la ressource qui convient de gérer avec la plus grande attention.

  Les solutions de sécurités des réseaux traditionnels sont souvent proscrites de ce type d'environnement, de nouveaux algorithmes et protocoles de sécurité sont nécessaires.

- **Agrégation des données :** Les techniques d'agrégations des données permettent de réduire la consommation énergétique en réduisant le nombre de bits transmis sur les liens sans-fil. Cela exige une attention particulière à détecter l'injection de fausses données ou la modification défectueuse de données, lors des opérations d'agrégation au niveau des nœuds intermédiaires.

  Ces techniques d'agrégations sont souvent utilisées. Elles sont cependant difficiles à mettre en œuvre lorsque les données sont chiffrées car le traitement des données devient alors très délicat.

- **Protection physique faible :** Les capteurs sont souvent déployés dans des environnements non-protégés (montagnes, forêts, champs de bataille,...). Par conséquent, ils peuvent facilement être interceptés et corrompus. De plus, à cause de leur faible coût, ils utilisent rarement des composants électroniques anti-corruption (tamper-resistant devices) **[10]**.

- **La communication sans fils multi-sauts:** la portée de la communication radio des "motes" est limitée en raison de considérations énergétiques. La communication multi-sauts est donc indispensable pour la diffusion des données dans un RCSF. Cela introduit de nombreuses failles de sécurité à deux niveaux différents: attaque de la construction et maintenance des routes, et attaque des données utiles par injection, la modification ou la suppression de paquets. En outre, la communication sans fil introduit d'autres vulnérabilités à la couche liaison en ouvrant la porte à des attaques de brouillage et de style déni de service par épuisement des batteries.

- **Echelle et dynamicité :** Les réseaux de capteurs contiennent souvent un nombre de nœuds très important. Ces réseaux sont souvent peu stables et très dynamiques : les capteurs, qui ont consommé leur pile, disparaissent et des nouveaux nœuds doivent êtres déployés pour assurer une certaine connectivité.

II.2) Exigences en sécurité :

Comme évoqué précédemment les réseaux de capteurs sans fil nécessitent dans de nombreuses applications des solutions qui assurent la sécurité des informations circulant sur le réseau. La sécurité des informations transitant sur les réseaux de type capteurs sans fil doivent répondre à plusieurs pré-requis :

- ***Confidentialité des données :*** la confidentialité des données est la question la plus importante dans la sécurité des réseaux, elle consiste à Préserver le secret des données transmises. l'approche standard pour sécuriser le transfère de données est de crypter les données avec une clé secrète connue par l'émetteur et le récepteur.

- ***L'authentification :*** L'authentification des capteurs est nécessaire pour s'assurer que l'identité déclarée par un capteur est bien celle du capteur déclarant. En l'absence d'un mécanisme permettant d'authentifier clairement un nœud du réseau, de nombreuses attaques peuvent se mettre en place.

- ***Intégrité des données :*** un nœud intrus (adversaire) peut modifier les données transférées. Par exemple un nœud malveillant peut ajouter quelques fragments ou manœuvrer les données dans un paquet. Ce nouveau paquet peut alors être envoyé au

récepteur original. La perte ou les dommages de données peut même se produire sans présence de nœud malveillant du a l'environnement dur de communication. Ainsi, l'intégrité des données s'assure qu'aucune donnée reçue n'a été changée en transit.

- ***Le contrôle d'accès :*** est la capacité des nœuds du réseau (ou bien d'une unité centrale comme la station de base) à accorder l'accès approprié aux ressources (connectivité, données,...) en fonction d'informations sûres. Autrement dit, c'est la capacité d'empêcher des éléments externes d'accéder au réseau, et cela en attribuant aux participants légitimes des droits d'accès afin de discerner les messages provenant des sources internes du réseau de ceux externes.

- ***La non-répudiation :*** assure la source d'un paquet. Empêche l'émetteur de nier avoir transmis un message et Le destinataire prouve qu'une source déterminée vient d'émettre le message en question

- ***Disponibilité du réseau :*** Le réseau doit pouvoir être disponible à tout instant, c'est-à dire que l'envoi d'information ne doit pas être interrompu, de même que la circulation de l'information ne doit pas être stoppée. Dans le cas d'un réseau de capteurs réactif, il faut qu'un capteur qui détecte un événement puisse transmettre à tout instant cette information vers la base du réseau de capteurs pour l'en informer

- ***Fraicheur des données :*** Par fraîcheur des données, nous entendons savoir si la donnée est récente ou non. Cela signifie qu'il faut s'assurer que la donnée transmise corresponde à un état présent. La fraîcheur des données garantit ainsi que ces données ne reflètent pas un état passé qui n'a plus cours. Sans mécanisme de sécurité vérifiant que les données transmises sont récentes, un attaquant pourrait capturer des informations circulant sur le réseau à une date T, puis les retransmettre à une date T+1 pour tromper le réseau et faire circuler de fausses informations. On peut prendre pour exemple un réseau de capteurs censé détecter les incendies, qui détecterait une première fois un incendie réel. L'attaquant enregistrerait les informations envoyées lors de cet événement. Il pourrait alors plus tard renvoyer ces mêmes données pour déclencher une fausse alerte **[9]**.

- ***Temps synchronisé :*** De nombreuses solutions de sécurité nécessitent des capteurs synchronisés pour qu'elles soient effectives. Il faut ainsi s'assurer que les capteurs du réseau ou des sous-réseaux du réseau ont une horloge commune afin par exemple d'éviter des attaques de type rejeu de paquets.**[9]**

II.3) Les attaques contre les RCSFs :

Une variété d'attaque contre les réseaux de capteur sans fil est rapportée dans la littérature spécialisée. Chacune avec ses objectifs propres. Par exemple, certaines attaques visent à affecter l'intégrité des messages qui transitent dans le réseau, tandis que d'autres visent à réduire la disponibilité du réseau ou de ses composants. Les attaques se produisent souvent par l'insertion d'éléments intrus dans le réseau. Il existe aussi des attaques contre l'environnement extérieur au réseau, les quelles provoquent des altérations ou des interférences sur les signaux transmis.

Une classification des attaques consiste à distinguer les attaques passives des attaques actives. On parlera d'attaque active si un attaquant modifie l'état du réseau, et d'attaque passive dans le cas où il ne cherchera qu'à l'écouter. Nous présentons dans la suite les attaques (active ou passive) les plus connues dans les RCSFs.

- ***Ecoute du réseau (eavesdropping) :*** Du fait que les transmissions se font en diffusion par les ondes radio, aucun contrôle d'accès au réseau n'est possible, ce qui est d'autant plus vrai que le réseau peut être déployé dans un environnement ouvert accessible à tout le monde. Il est donc très facile d'intercepter des données échangées sur un réseau de capteurs et d'accéder à leur contenu si aucun service de confidentialité n'est prévu.

- ***Analyse du trafic :*** L'analyse du trafic est une attaque qui met en jeu des mécanismes d'écoute passive et de surveillance du réseau. L'attaquant en analysant uniquement les chemins empruntés par les paquets sur le réseau pourra récupérer des informations précieuses sur les vulnérabilités de ce réseau **[9]**.

- ***Attaque physique (tampering) :*** Comme les RCSFs sont très souvent déployés dans des zones sans aucune protection, ils sont très exposés aux attaques physiques qui peuvent être considérées sous différents points de vue. L'un est lié au matériel qui n'est pas qualifié d'inviolable. Dans ces conditions, une attaque aura pour but de récupérer du matériel cryptographique comme les clés utilisées pour le chiffrement. Un autre objectif serait de reprogrammer le capteur pour perturber le réseau et l'application en provoquant volontairement un comportement anormal du nœud. La seconde attaque physique consisterait simplement à supprimer le capteur du réseau en le détruisant (on retombe sur la question de l'inviolabilité) ou en le subtilisant **[14]**.

- ***Brouillage radio (jamming) :*** Une attaque bien connue sur la communication sans-fil, est celle qui consiste à perturber le canal radio en envoyons des informations inutiles sur la bande de fréquences utilisées. Ce brouillage peut être temporaire intermittent ou permanant.

- ***Flooding :*** Dans ce type d'attaque, un attaquant utilise un ou plusieurs nœuds malicieux ou un dispositif particulier avec dans certains cas une puissance d'émission forte, pour envoyer régulièrement des messages sur le réseau afin de le saturer. On est en présence d'une attaque active qui est de même type que les attaques de type déni de service dans les réseaux classiques.

- ***Hello Flooding :*** Les protocoles de découverte dans les réseaux ad-hoc utilisent des messages de type "HELLO" pour découvrir ses nœuds voisins et pour s'insérer dans un réseau. Dans une attaque dite de HELLO Flooding, un attaquant utilise ce mécanisme pour consommer l'énergie des capteurs et empêcher leurs messages d'être routés.

- ***Réplication des données :*** Si les paquets envoyés sur le réseau peuvent être lus et enregistrés par un attaquant, il peut renvoyer ces mêmes paquets à une date ultérieure pour tromper le réseau. Pour illustrer cette attaque, on peut prendre pour exemple un réseau de capteurs qui a pour objectif de détecter un incendie : si un premier incendie est détecté et qu'un capteur envoie un paquet pour en informer la base, l'attaquant pourra enregistrer ce paquet, même s'il est chiffré et qu'il ne peut le déchiffrer, puis l'émettre à une autre date en se faisant passer pour ce capteur et faire croire à un nouvel incendie.

  Cette attaque est réalisable si le paquet ne contient pas d'information concernant la date de l'envoi ou si cette date est accessible et facilement modifiable par un attaquant.

- ***Injection de message :*** L'attaquant va chercher par divers moyens (utilisation de nœuds malicieux, envoi de paquets sur la même fréquence radio, réplication de données, etc. . .) à injecter des messages dans le réseau. Cette injection de messages peut avoir pour effet de perturber le réseau, le saturer ou le tromper en envoyant de fausses informations.

- ***Altération des messages :*** Un nœud malicieux va récupérer un message et l'altérer, en lui ajoutant des fausses informations (sur le destinataire, l'émetteur, l'information en elle-même), en le modifiant ou bien en détruisant des paquets pour rendre incompréhensible le message

- ***Attaque du trou noir (black hole attack) :*** L'attaque du trou noir consiste tout d'abord à insérer un nœud malicieux par divers moyens dans le réseau. Ce nœud va modifier les tables de routage pour obliger le maximum de nœuds voisins à faire transiter leurs informations par lui. Ensuite, toutes les informations qui passent par ce nœud ne seront jamais retransmises. La figure 10 représente un trou noir mis en place par un nœud malicieux X qui a modifie le routage pour que les clusters 1, 2, 3 et 4 fassent passer l'information par lui. pour communiquer entre cluster dans ce cas de figure, le trou

noir X ne retransmettra aucune information, empêchant toute communication entre les différents clusters.

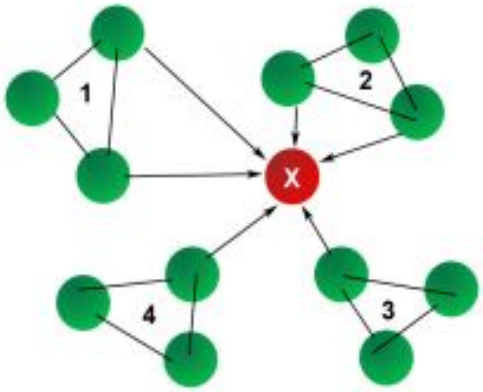


Figure-7 : exemple d'un trou noir dans un réseau clustérisé. [9]

- ***Attaque du trou gris*** *(grey hole attack)* ***:*** L'attaque de trou gris est une variante améliorée de l'attaque de trou noir **[9]**. le trou gris relaye certaine information. Par exemple le trou gris va relayer toute les informations concernant le routage, sauf pour les informations critique. Ce type d'attaque est ainsi plus difficile à détecter que l'attaque du trou noir, le capteur malicieux tant qu'il se comporte de manière normale ne peut être détecté.

- ***Attaque du trou de ver (worm hole attack)****:* L'attaque du trou de ver nécessite l'insertion dans le réseau de capteurs d'au moins deux nœuds malicieux. Ces deux nœuds sont reliés entre eux par une connexion puissante qui peut être filaire ou radio, ce qui permet de tromper les autres nœuds sur les distances les séparant **[9]**.

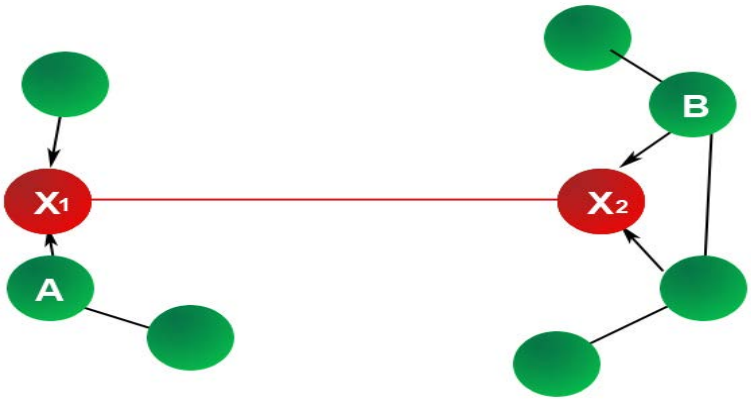


Figure-8 : Attaque du trou de ver (wormhole attack).

- ***Attaque du trou de la base (sinkhole attack)* :** Dans une attaque *sinkhole*, le nœud malicieux essaye d'attirer vers lui le plus de chemins possible permettant le contrôle sur la plupart des données circulant dans le réseau. Pour ce faire, l'attaquant doit apparaitre aux autres comme étant très attractif, en présentant des routes optimales.

- ***Attaque par chantage (Blackmail attack):*** Dans ce type d'attaque, un noud malicieux fait annoncer qu'un autre nœud légitime est malicieux pour éliminer ce dernier du réseau. Si le nœud malicieux arrive à attaquer un nombre important de nœud, il Pourra perturber le fonctionnement du réseau.

- ***Attaque sybile (Sybil attack) :*** Une attaque de type « Sybil attack » consiste à ce qu'un capteur malicieux se fasse passer pour plusieurs capteurs en utilisant l'identité d'autres capteurs légitimes du réseau. L'attaque sybile va alors pouvoir tenter de mettre en péril les mécanismes comme l'agrégation des données, la sécurité, le routage, l'allocation de ressource.

- ***Attaque spécifique au type de capteur :*** Dans cette attaque, un attaquant modifie de manière physique le comportement du capteur. Il peut par exemple allumer une flamme devant un capteur thermique ou bien allumer une lampe devant un capteur de luminosité pour tromper un capteur. A cet effet, ce dernier va remonter l'information continuellement jusqu'à l'épuisement son énergie ou enregistrer de fausses informations sur le réseau.

II.4) Modèle de l'attaquant :

En général, plus l'attaquant dispose de ressources, plus la défense est coûteuse. La connaissance de la capacité de l'attaquant permet de définir au mieux la défense. La conception de réseaux de capteurs doit prendre en considération les menaces les plus fréquentes en énumérant les capacités des attaquants (leur nombre, leur coordination, leur capacité technique et leur intérêt d'influence). La définition des capacités techniques des attaquants est importante pour connaitre la nature de leur menace, par exemple un attaquant peut seulement recevoir la transmission de données, mais il peut aussi se présenter comme un capteur légal du réseau, et avoir accès à la totalité des services du réseau.

Selon **[14]**, un réseau de capteurs peut être attaqué par deux types d'attaquants : un attaquant puissant et un attaquant réaliste.

II.4.1) Attaquant puissant :

L'adversaire est considéré comme présent avant et après le déploiement des nœuds. Il peut surveiller toutes les communications, n'importe où, et à tout instant.

II.4.2) Attaquant réaliste :

L'attaquant est capable de surveiller un pourcentage fixe des canaux de communication lors du déploiement du réseau.

II.5) Mécanismes de sécurité :

Pour contrer les attaques qui menacent les réseaux de capteurs sans fil, plusieurs équipes de recherche tentent de trouver des solutions appropriées. Ces solutions doivent bien sûr prendre en compte les spécificités des réseaux de capteurs sans fil. Il faut donc trouver des solutions simples qui permettent de sécuriser le réseau tout en consommant le moins d'énergie possible et adapter ces solutions à une puissance de calcul faible.

Plusieurs mécanismes, basés généralement sur la notion de cryptographie, sont mis en place afin de répondre à la question de sécurité dans les RCSFs.

II.5.1) La cryptographie :

II.5.1.1) Définition de la cryptographie :

La cryptographie est l'étude des techniques mathématiques qui permettent d'assurer certains services de sécurité. Elle est définie comme étant une science permettant de convertir des informations «en clair» en informations cryptées (codées), c'est-à-dire non compréhensible, et puis, a partir de ces informations, de restituer les informations originales **[16]**.

Il s'agit d'une science mathématique comportant deux branches : la cryptographie et la cryptanalyse **[3]**.

- **Cryptographie :** La cryptographie est l'étude des méthodes donnant la possibilité d'envoyer des données de manière confidentielle sur un support donné.
- **Cryptanalyse :** Opposée a la cryptographie, elle a pour but de retrouver le texte en clair a partir de textes chiffres en déterminant les failles des algorithmes utilises.

Un algorithme cryptographique est une fonction mathématique utilisée pour le chiffrement et le déchiffrement **[17]**.

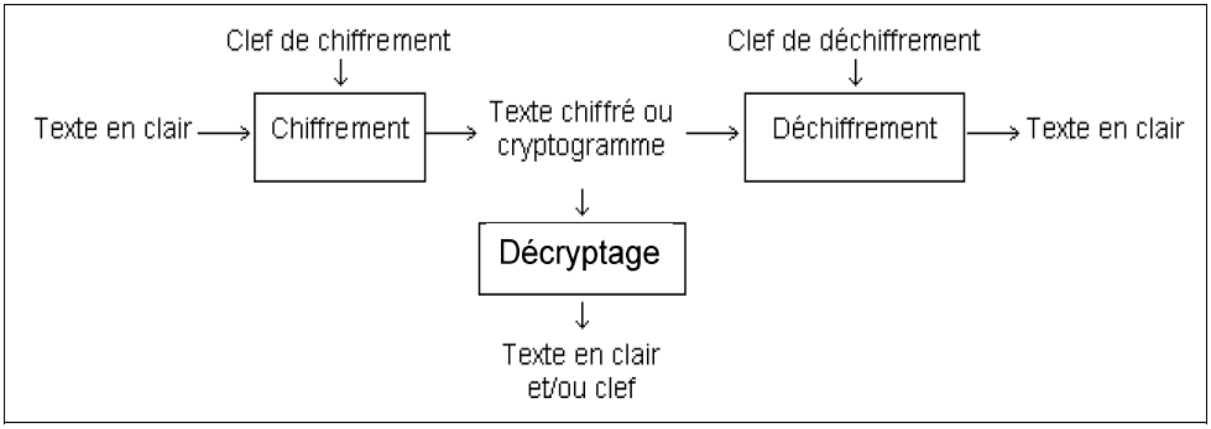


Figure-9 : Protocole de chiffrement.

Il existe deux grandes familles d'algorithmes cryptographiques à base de clefs : les algorithmes à *clef secrète* ou **algorithmes symétriques**, et les algorithmes à *clef publique* ou **algorithmes asymétriques [17]**. Avant de les aborder, il est commode de définir la notion de clé qui sera utilisée tout au long de cette partie.

Une clé: Paramètre constitué d'une séquence de symboles et utilisé, avec un algorithme cryptographique, pour transformer, valider, authentifier, chiffrer ou déchiffrer des données **[2]**.

II.5.1.2) Chiffrement symétrique ou à clef secrète :

Le principe de chiffrement symétrique se base sur le partage d'une même clé *K* de chiffrement entre deux entités communicantes (voir la figure 8).

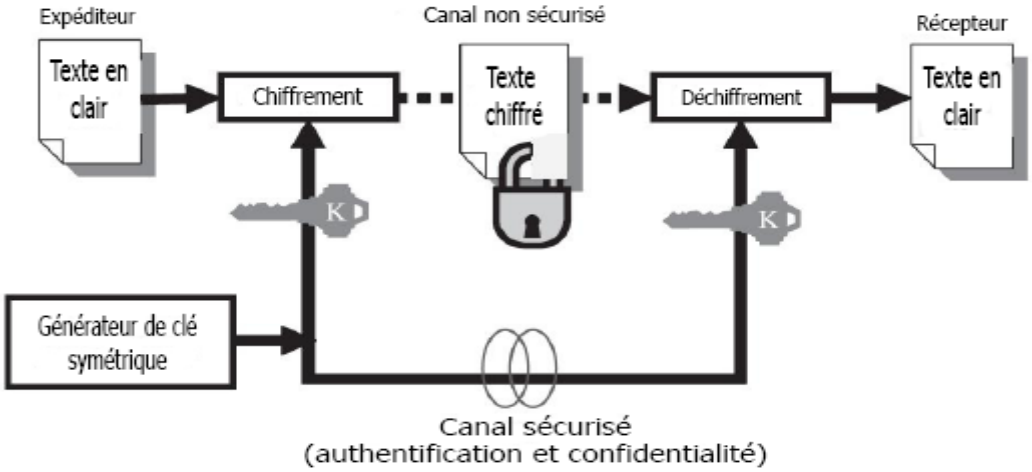


Figure-10 : Chiffrement symétrique [18].

Les algorithmes de cryptographie symétrique se décomposent en deux sous ensembles, les algorithmes de chiffrement en chaine (en continue) et les algorithmes de chiffrement par blocs.

- **Chiffrement en chaine (en continu) :**

  Ce système chiffre les données bit par bit quelle que soit la longueur du message à coder sans besoin de les découper et/ou attendre la réception entière des données.

  La technique utilisée dans cette classe consiste à chiffrer le message à transmettre en effectuant un XOR avec la clé de chiffrement.

  Soit $\boldsymbol{M}$ le message à chiffrer, $\boldsymbol{K}$ la clé de chiffrement, et $\oplus$ l'opération booléenne XOR, le chiffrement correspondant est :

  $$\boldsymbol{M} \oplus \boldsymbol{K} = \boldsymbol{MK}$$ ou $\boldsymbol{MK}$ est le message chiffré.

  Le déchiffrement se fera alors par :

  $$\boldsymbol{MK} \oplus \boldsymbol{K} = \boldsymbol{M} \oplus \boldsymbol{K} \oplus \boldsymbol{K} = \boldsymbol{M}$$

- **Chiffrement par bloc :**

  Le chiffrement par blocs consiste à fractionner un message $\boldsymbol{M}$ en blocs de n bits. Ces blocs seront ensuite chiffrés par un algorithme de chiffrement $\boldsymbol{F}$ et une clé $\boldsymbol{k}$ extraite d'une clé maître $\boldsymbol{K}$.

  Soient $\boldsymbol{M}$ le message à chiffrer et $\boldsymbol{K}$ la clé de chiffrement dont sont extraites les clés $\boldsymbol{k_i}$ et $\boldsymbol{F}$ la fonction de chiffrement. $\boldsymbol{M}$ sera découpé en $\boldsymbol{r}$ blocs de $\boldsymbol{n}$ bits. Pour chaque bloc $\boldsymbol{b_x}$ de $\boldsymbol{M}$, le chiffrement se fera de la manière suivante :

  $$\boldsymbol{C_1 = F(k_1 , b_x)}$$

  $\boldsymbol{F}$ est ensuite itérée avec une nouvelle clé extraite de la clé maître $\boldsymbol{K}$ pour garantir la sécurité de l'algorithme de chiffrement, ainsi :

  $$\boldsymbol{C_2 = F(k_2 , C_1)}$$
  $$\boldsymbol{C_y = F(k_y , C_{y-1})}$$

  Le déchiffrement se fait avec une fonction $\boldsymbol{G}$, inverse de la fonction $\boldsymbol{F}$ et les différentes clé $\boldsymbol{k_i}$ partagées extraites de la clé commune $\boldsymbol{K}$, de la manière suivante :

$$C_{y-1} = G(k_y , C_y) = G(k_y , F(k_y , C_{y-1}))$$

$$b_x = G(k_1 , C_1)$$

La cryptographie symétrique par blocs est la technique de cryptographie symétrique la plus répandue, utilisée entre autre par les algorithmes de cryptographies comme DES, AES ou Blow-fish **[9]**.

II.5.1.3) Le chiffrement asymétrique :

Le chiffrement asymétrique appelé aussi chiffrement à clé publique utilise une *paire* de clés pour le cryptage : une clé publique qui chiffre des données et une clé privée ou secrète correspondante pour le déchiffrement. La clé privée doit rester secrète alors que la clé publique doit être diffusée. Un exemple est illustré dans la figure 11.

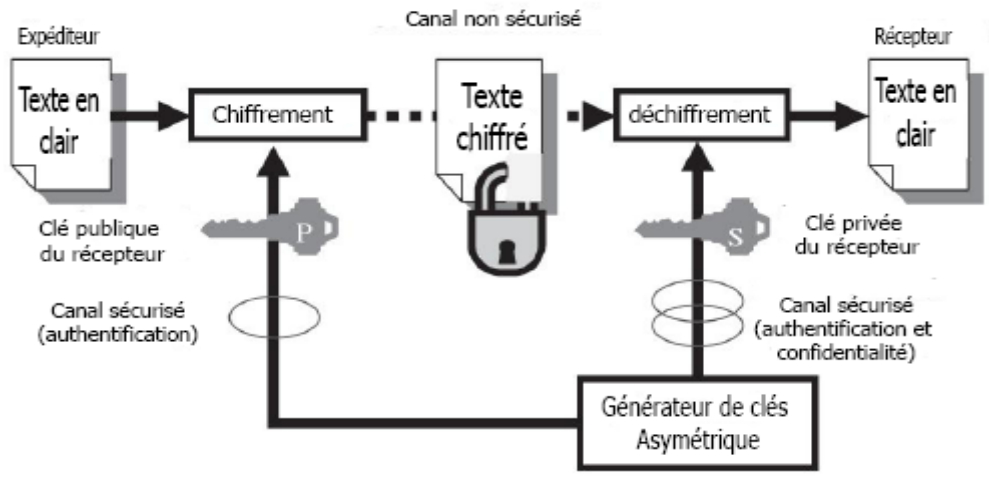


Figure-11 : Chiffrement asymétrique [18].

Formellement le chiffrement et le déchiffrement de données d'un message entre deux nœuds ***A*** et ***B*** correspondent au mécanisme suivant : soit K_P la clé publique et K_S la clé privée de ***A***, ***F*** la fonction de chiffrement et ***G*** la fonction de déchiffrement. ***A*** diffuse sa clé publique dans le réseau. Soit ***M*** le message que souhaite transmettre ***B*** à ***A***, alors :

$Mk = F(K_P , M)$ où ***Mk*** est le message chiffré,
$M \neq G(K_P , Mk)$

B envoie le message ***Mk*** à ***A*** qui va ensuite pouvoir le déchiffrer avec sa clé privée :

$$M = G(K_S , Mk)$$

II.5.1.4) Symétrique vs Asymétrique :

Chacune de ces deux formes de cryptographie possède ses propres avantages et inconvénients. En général, les algorithmes de chiffrement symétrique sont très rapides. D'une part, ils ont une complexité moins élevée qu'un algorithme de chiffrement à clé publique et d'autre part, ces algorithmes sont souvent très bien adaptés à l'architecture du processeur utilisé. Ils sont donc très bien appropriés au chiffrement de grande quantité de données. Les plus connus sont les algorithmes DES (*Data Encryption Standard*) et AES (*Advanced Encryption Standard*). En revanche, la gestion des clés des crypto-systèmes symétriques est assez problématique.

Ainsi les algorithmes de chiffrement asymétrique résolvent le problème de distribution des clés que l'on peut rencontrer dans le chiffrement symétrique. L'atout principal de chiffrement à clé publique réside dans la facilité de gestion du parc des clés des utilisateurs. En effet, l'augmentation du nombre d'utilisateurs ne complexe pas le protocole. Néanmoins, les techniques asymétriques souffrent de leur grande lenteur. Bien sûr, il ne faut pas une heure pour chiffrer un message mais c'est bien 100 à 1000 fois plus longues que certaines techniques symétriques.

Pour le cas des réseaux de capteurs, qui sont des réseaux contraints, la cryptographie symétrique est souvent privilégiée par rapport à la cryptographie à clé publique pour les deux raisons suivantes **[18]** :

- la vitesse de traitement des données est plus importante dans le cas d'un chiffrement symétrique, ce qui conduit à une consommation énergétique et un temps de calcul beaucoup moins important.
- la taille des clés et des paquets à chiffrer ou signer est nettement inférieur pour un même niveau de sécurité dans le cas de la cryptographie symétrique.

Ces deux justifications sont cruciales pour un environnement comme un réseau de capteurs. C'est pour cette raison que la majorité de solutions de sécurité pour les RCSFs se base sur la cryptographie symétrique.

II.5.2) Fonctions de hachage :

Lors d'échanges de messages cryptés, il est important de pouvoir s'assurer que le message n'a pas été altéré ou modifié par un tiers pendant l'envoi. Les fonctions de hachage permettent alors de s'assurer de l'intégrité du message. Le principe est qu'un message clair de longueur quelconque doit être transformé en un message de longueur fixe inferieure à celle de départ. Le message réduit portera le nom de "hache" ou de "condense". L'intérêt est d'utiliser

ce condense comme empreinte digitale du message original afin que ce dernier soit identifie de manière univoque.

Une fonction de hachage cryptographique doit répondre aux trois critères suivants :

- Il est facile de calculer **h(m)**, c'est-à-dire, de calculer l'empreinte a partir du contenu du message.
- Il est difficile de trouver le contenu du message a partir de l'empreinte, c'est-à-dire quand la valeur hachée H = h(M) pour un message M est donnée, il est difficile de retrouver le message M.
- il est difficile de trouver deux messages distincts M et M' tel que h(M) = h(M'). c'est-à-dire, il est difficile de trouver deux messages aléatoires qui donnent la même empreinte et cela mène à la résistance aux collisions.

Les algorithmes de hachage les plus utilises actuellement sont :
- MD5 (MD signifiant Message Digest) créant une empreinte digitale de 128 bits.
- SHA (Secure Hach Algorithm, pouvant être traduit par algorithme de hachage sécurisé) créant des empreintes d'une longueur de 160 bits.

II.5.3) La signature :

Les fonctions de hachage permettent de s'assurer de l'intégrité d'un message mais un autre problème se pose : comment être certain que l'identité de l'expéditeur n'a pas été usurpée pour vous envoyer un message ? Ou que l'expéditeur ne va pas nier vous l'avoir envoyé ?

C'est le rôle de la signature numérique, celle-ci fournissant donc les services d'intégrité des données, d'authentification de l'origine des données et de non-répudiation. Elle repose sur les clés asymétriques. L'expéditeur signe le message entier ou le condensé avec sa clé privée en produisant une signature digitale. Ce dernier est par la suite envoyé avec les données. Si elle peut être déchiffrée par le récepteur mais en utilisant la clé publique de l'émetteur et si son résultat est identique aux données reçues alors la signature est valide, c'est-à-dire que les données proviennent bien de leur émetteur légitime qui ne pourra pas nier l'émission de ses données dans le futur.

II.5.4) Code d'authentification de message ou MAC :

Le concept est relativement semblable aux fonctions de hachage. Il s'agit ici aussi d'algorithmes qui créent un petit bloc authentificateur de taille fixe. La grande différence est que ce bloc authentificateur ne se base plus uniquement sur le message, mais également sur une clé secrète connue seulement par les deux entités échangeant le message. Un MAC est donc un algorithme qui prend en entrée un message M, une clé K et qui produit un condensé ou ce qu'on appelle un ***tag*** : ***tag*** $= MAC_K(M)$. L'algorithme de vérification pour le receveur

consiste à vérifier si pour le message M' reçu on a bien ***tag? = $MAC_K(M')$***. Ainsi, si la relation est vérifiée, le receveur a bien la preuve que le message n'a pas été modifié au cours de la transmission et que l'émetteur du message est bien celui qui connaît également K.

II.6) La gestion des clés dans les RCSFs :

L'établissement d'une communication sécurisée pour la majorité des applications des Réseaux de Capteurs Sans-fil (RCSFs) revêt un caractère primordial. Le problème est comment établir des clés cryptographiques entre les nœuds capteurs, afin d'assurer la sécurisation des communications. Il serait inutile d'intégrer des algorithmes cryptographiques dans un système si la gestion des clés correspondante n'est pas satisfaisante. Dans les RCSFs, cette gestion est cruciale pour pouvoir offrir aux communications inter nœuds un niveau de sécurisation appréciable.

II.6.1) La fonction de gestion de clés dans les RCSF:

II.6.1.1) Définition :

La gestion des clés est l'un des aspects les plus difficiles de la configuration d'un système cryptographique de sécurité. Pour qu'un tel système fonctionne est soit sécurisé, chacun des utilisateurs doit disposer d'un ensemble de clés secrètes (dans un système à clés secrètes) ou de paire de clés publiques/privés (dans un système à clés publiques). Cela implique de générer les clés et de les distribuer de manière sécurisée aux utilisateurs ou d'offrir à l'utilisateur le moyen de les générer. Il doit aussi pouvoir enregistrer et gérer ses clés publiques et privées de manière sure. Dans les systèmes à clés publiques, la gestion des clés comprend la capacité à vérifier et à gérer les clés publiques des autres utilisateurs qui sont signées sous formes de certificats numériques. **[4]**

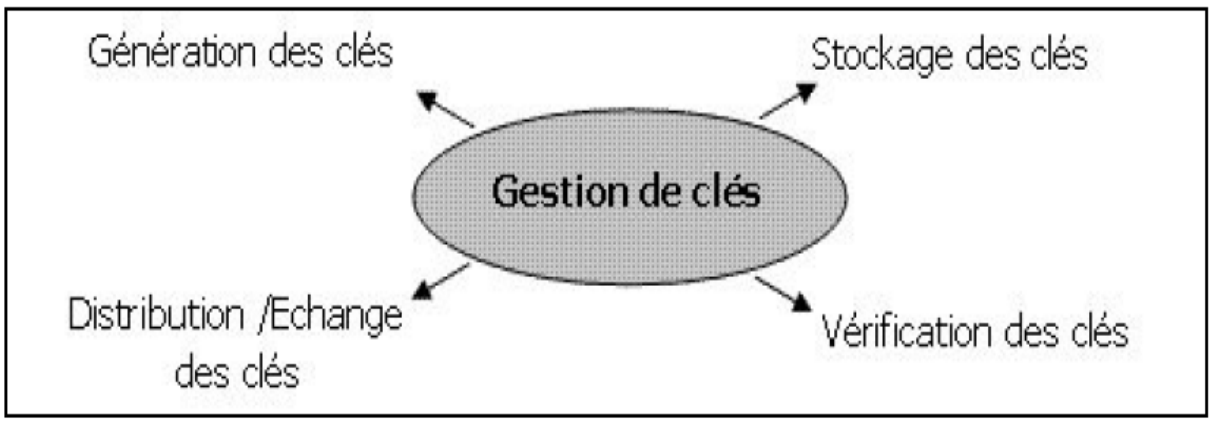


Figure-12 : Fonction de la gestion des clés.

II.6.1.2) Pourquoi la gestion de clés dans les RCSF ? :

Après leur déploiement, les capteurs ont besoin d'établir des clés cryptographiques avec leurs voisins pour:

- ✓ Assurer l'échange sécurisé des données.
- ✓ Supporter l'ajout et la suppression des nœuds.
- ✓ Fonctionner dans des environnements non défini à priori.
- ✓ Un nœud non autorisé ne peut pas effectuer des communications avec les nœuds du réseau.

II.6.1.3) Contraintes de conception :

La figure suivante résume les contraintes découlant des propriétés des RCSFs, à prendre en compte dans la conception d'une solution de gestion de clés pour les RCSFs **[4]**.

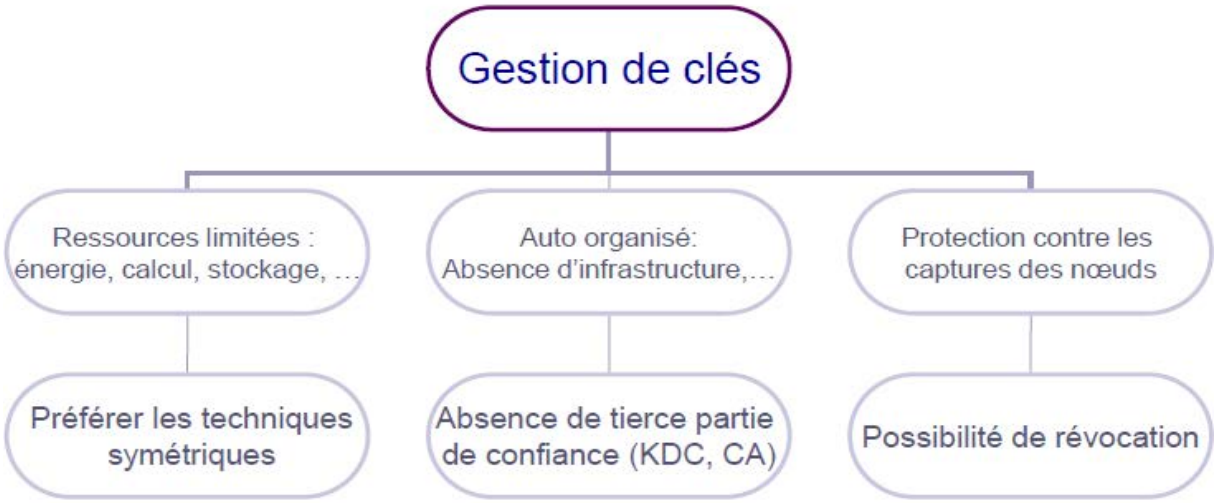


Figure-13 : Contraintes de conception de solutions de gestion de clés.

II.6.2) Les approches de gestion de clés :

Comme nous l'avons déjà expliqué (dans la section II.5.1.4), il est préférable dans les réseaux de capteurs d'utilisé la cryptographie symétrique. Ainsi un nouveau problème émerge dans ce cas et concerne la gestion des clés partagées dans le réseau d'une part entre deux nœuds voisins et entre chacun des nœuds et la station de base.

Une solution commune pour ce problème consiste à utiliser une méthode de pré-distribution dans laquelle les clés sont chargées dans les nœuds avant le déploiement. La

figure 14 illustre une taxonomie des solutions de gestion de clés basée sur la pré-distribution. Dans cette taxonomie, les protocoles sont classés dans plusieurs catégories selon la technique utilisée pour la construction des clés partagées ainsi que la topologie du réseau utilisée. On peut distinguer 4 catégories **[18]** :

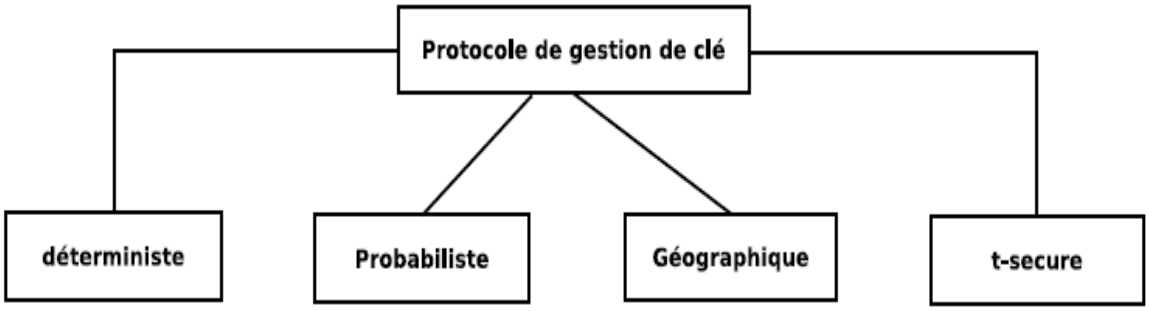


Figure-14 : Taxonomie des protocoles de gestion de clés.

- Les protocoles déterministes (qui utilisent une clé maîtresse pour dériver les clés entre paires de capteurs) ;
- Les protocoles probabilistes (qui supposent que deux nœuds voisins vont partager une clé avec une certaine probabilité) ;
- Les protocoles géographiques (qui utilisent des informations de positions géographiques pour construire les clés partagées) ;
- Les protocoles t-secure (qui résistent à la compromission de t nœuds dans le réseau).

Dans ce qui suit, nous présentons quelques protocoles de gestion de clés basés sur la pré-distribution, ainsi que des solutions introductives dites naïves.

II.6.3) Solutions Introductives :

- **Solution 1 : Une seule clé partagée par le réseau :**

La solution la plus simple consiste à utiliser une clé unique partagée par tous les nœuds du réseau. Les avantages de cette solution sont :

- Gestion simple des clés, car il suffit de pré-charger les nœuds, avant le déploiement, par une seule clé.

- Toutes les communications peuvent être chiffrées simplement en utilisant un minimum de mémoire (stockage d'une seule clé).

Par contre cette méthode a la vulnérabilité suivante :

- Elle ne présente aucune résilience contre la compromission d'un nœud, parce que si un attaquant compromet un nœud du réseau, et étant donné que tous les nœuds du réseau communiquent entre eux en utilisant la même clé, dans ce cas, la sécurité de tout le réseau est menacée.

- **Solution 2 : clé partagée par paire de nœuds:**

Dans cette solution, chaque nœud est pré-chargé avec N - 1 clés secrètes, chacune de ces clés est connue seulement par ce nœud et un des N - 1 autres nœuds (N étant le nombre de nœuds dans le réseau). Cette solution présente l'avantage suivant :

- La résilience est parfaite car la compromission d'un nœud n'affecte pas la sécurité des autres nœuds.

Par contre cette solution n'est pas appropriée aux RCSF car :
- Elle exige une capacité mémoire importante pour stocker les N -1 clés (N peut être grand).
- L'ajout de nouveaux nœuds est difficile parce que les nœuds existants ne possèdent pas les clés de ces nouveaux nœuds.

- **Solution 3 : Clé partagée par groupe de nœuds :**

Dans ce cas de figure, chaque groupe ou cluster partage une clé en commun qui lui permet de communiquer à l'intérieur du groupe. Les nœuds maîtres, ou cluster head, communiquent entre eux avec, soit une clé commune à tous les clusters heads, soit une clé commune par paire de cluster head. Cette solution est une solution hybride des deux premières techniques (solutions). Elle permet de limiter le nombre d'encryptages dans les communications. Cependant elle a pour défaut de reporter l'essentiel du travail d'encryptage sur les clusters heads. Pour rester efficace, il faut donc s'assurer de changer régulièrement de nœud maître dans un groupe pour ne pas consommer toute l'énergie du cluster head.

- **Solution 4 : Clé individuelle :**

Dans cette solution chaque nœud possède une clé personnelle pour crypter son information. Cette clé n'est connue que de la base. L’avantage de cette solution est :

- Un message envoyé par un nœud circulera de manière cachée sur le réseau jusqu'à atteindre la base.

Cependant elle présente l'inconvénient suivent :

- Elle ne permet pas de sécuriser les informations transmises entre nœuds car ceux-ci ne possèdent pas de clé de chiffrement pour communiquer de manière sécurisée entre eux.

II.6.4) Protocoles basés sur la pré-distribution de clés :

II.6.4.1) Protocoles de gestion de clés probabilistes :

Eschenauer et Gligor ont proposé un schéma de gestion de clé basé sur la probabilité de partager une clé entre les nœuds du réseau. Ce protocole est relativement simple et il opère comme suit **[19]** :

- L'administrateur du réseau génère un tableau des nombres aléatoires, indexé de 1 à L.
- Chaque nœud est pré-chargé, avant le déploiement, avec un sous ensemble de m clés prélevées aléatoirement à partir d'un grand ensemble de clés.
- 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.

Bien que ce schéma à l'avantage d'être simple et complètement distribué, il a deux principaux inconvénients, qui sont :

- Plus la probabilité de partager une clé entre des nœuds voisins augmente plus l'espace mémoire, pour contenir un grand sous ensemble de clés, est important.

- La sécurité fournit par ce protocole, 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.

II.6.4.2) Protocoles de gestion de clés déterministes :

Contrairement aux schémas de gestion de clés probabilistes, les protocoles de gestion de clés déterministes assurent que chaque nœud est capable d'établir une clé par-paire avec ses voisins **[20]**. Pour garantir le déterminisme, les protocoles, tels que LEAP et OTMK, utilisent une clé commune, transitoire et pré-chargée dans tous les nœuds avant leurs déploiements. Cette clé est utilisée en vue de générer des clés par-paires entre chacun des deux nœuds voisins. Pour sécuriser les nœuds contre les attaques de captures, la clé transitoire est effacée du nœud après la génération des clés par-paires.

Ces protocoles ont été les premiers décrits dans la littérature et la plupart d'entre eux présentent deux inconvénients majeurs à savoir :

- Ils ne résistent pas à la compromission de nœuds et si un nœud est compromis avant qu'il supprime la clé transitoire, l'attaquant peut construire impunément toutes les clés qu'il souhaite avec ses voisins en se faisant passer pour un nœud légitime.

- Ils nécessitent souvent beaucoup de communications supplémentaires dans le réseau, Ce qui influencera la durée de vie de ce dernier.

II.7) Conclusion :

Nous avons abordé dans ce chapitre l'aspect de sécurité dans les RCSF et les contraintes de ces derniers qui rendent impossible l'application des méthodes classiques de sécurité dans de tels réseaux, ainsi que les différentes approches existantes pour la gestion des clés. Dans le chapitre qui suit nous étudierons les algorithmes de chiffrement par bloc AES, DES et TripleDES.

Chapitre III

Description des algorithmes cryptographiques DES, TripleDES et AES

III) Introduction :

Dans les RCSFs, les algorithmes de chiffrement symétriques par bloc sont souvent les plus utilisés pour garantir la confidentialité des données. D'une manière générale, ces algorithmes utilisent des opérations arithmétiques, des substitutions et des permutations, et la plupart sont des réseaux de ***Feistel***.

Dans la suite de cette partie, les algorithmes de chiffrement symétriques (le DES TripleDES et l'AES) sont décrits.

III.1) Algorithmes utilisant des substitutions et des permutations :

Ces algorithmes utilisent des tables de substitution et des tables de permutation. C'est à dire, une table de substitution est une table d'entiers, le nombre d'entrée constitue l'adresse de la table, la sortie est le contenu de la table. Une table de permutation est une table qui définit l'emplacement des bits après transformation.

Les algorithmes de chiffrement par blocs tels que l'AES et le DES, sont basés sur le principe de réseau de substitution-permutation. Mais les détails et les mécanismes de génération des sous clefs de ces algorithmes sont totalement différents.

III.2) L'algorithme DES :

L'algorithme DES, Data Encryption Standard, a été créé dans les laboratoires de la firme IBM Corp. Il est devenu le standard du NIST en 1976 et a été adopté par le gouvernement en 1977 **[21]**. C'est un algorithme de chiffrement symétrique (chiffrement par bloc) basé sur le schéma de *Feistel* et utilisant des clés de 56 bits.

La clé du DES est en fait constituée de 64 bits, dont les 56 bits sont générés aléatoirement et utilisés dans l'algorithme. Les huit autres bits peuvent être utilisés pour la détection d'erreurs (dans une transmission par exemple). Chacun des huit bits est utilisé comme bit de parité des sept groupes de 8 bits.

III.2.1) Le fonctionnement de DES :

L'algorithme DES transforme un bloc de 64 bits en un autre bloc de 64 bits. Il est constitué de **16** étapes avec **16** sous-clés de 48 bits générées (une clé par étape). Au début et à la fin de cet algorithme, il est appliqué respectivement une permutation IP sur 64 bits et son inverse IP^{-1}. Les différentes opérations effectuées par cet algorithme sont présentées dans la figure suivante.

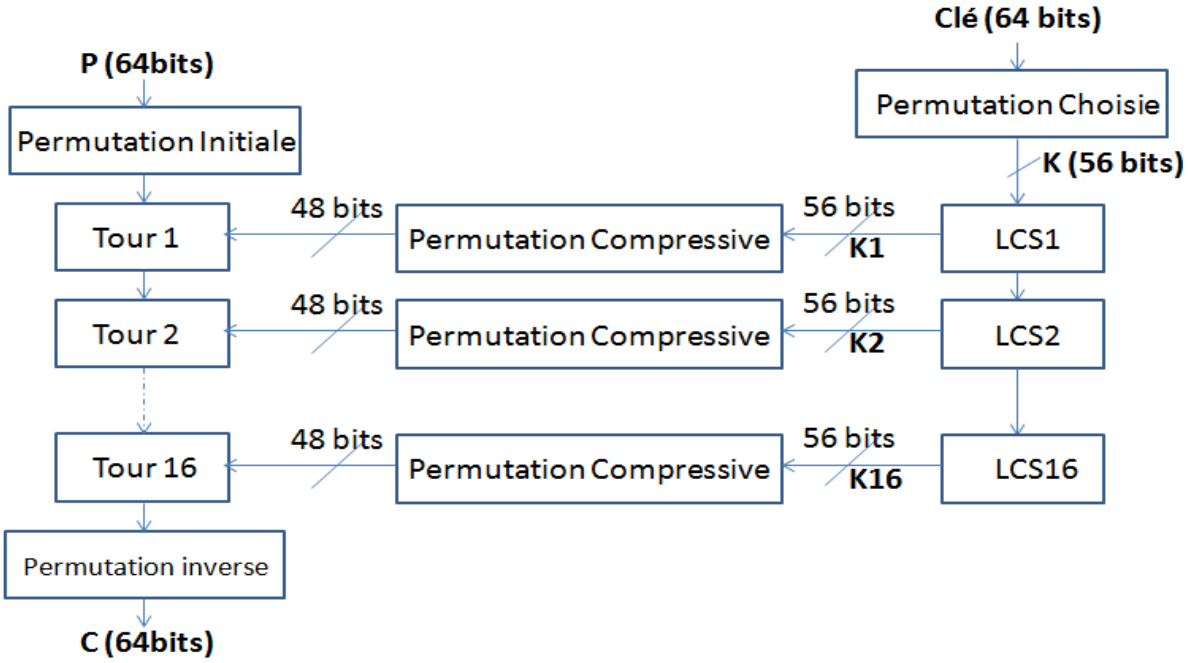


Figure-15 : Le principe de fonctionnement du DES. [22]

III.2.1.1) La permutation initiale :

La permutation initiale consiste à réordonner les 64 bits du bloc de texte en clair. Ce nouvel ordre est décri dans la table donnée ci-dessous (à lire de gauche à droite et de haut en bas). Elle produit un bloc de même taille.

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Tab-2 : Permutation initiale (PI) de l'algorithme DES.

La permutation initiale déplace le 58eme bit jusqu'a la première position, le bit 50 jusqu'à la second position,… etc.

III.2.1.2) La permutation choisie :

La permutation choisie consiste à réduire la taille de la clé en ignorant un bit sur huit, les bits restants (56 bits) seront réordonnées selon l'ordre décrie dans la table suivante.

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	11	3	60	52	44	36
63	55	47	39	31	23	15	7	62	54	46	38	30	22
14	6	61	53	45	37	29	21	13	5	28	20	12	4

Tab-3 : Permutation choisie (PC) du DES.

III.2.1.3) Décalage circulaire à gauche de la clé (LCS) :

La clé de 56 bits est divisée en deux parties de 28 bits chacune, en suite chaque partie est décalée à gauche et indépendamment de l'autre de un ou de plusieurs bits en fonction du tour actuel.

Le nombre de décalage a effectué en fonction du tour est donné par la table suivante :

Ronde	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Nombre de décalages	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tab-4 : Nombre de décalage de la clef pour le cryptage avec le DES.

III.2.1.4) la permutation compressive :

La permutation compressive consiste à réduire la taille de la clé afin d'extraire la clé du tour de 48 bits. Ces 48 bits sont également réordonnée comme indiqué dans la table suivante.

14	17	11	24	1	5	3	28	15	6	21	10
13	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	57	45
44	49	39	56	34	53	46	42	50	36	29	32

Tab-5 : permutation compressive du DES.

III.2.1.5) les tours du DES :

Le bloc de 64 bits résultant de la permutation initiale est scindé en deux blocs de 32 bits notés L_i et R_i. Ces deux bloc vont ensuite traverser les 16 rounds (tours du DES). Chaque round est composé de plusieurs étapes résumées dans la figure suivante.

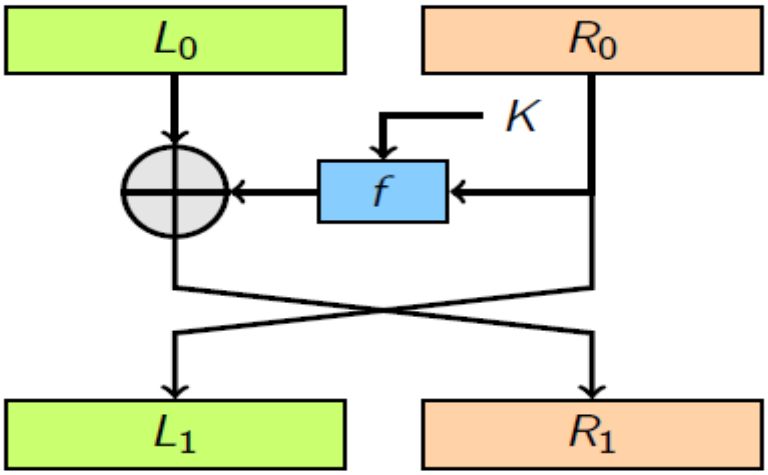


Figure-16 : Schéma d'un tour du DES (schéma de Feistel) [23].

Pour chaque cycle de ronde, les bits de la clef sont décalés et 48 bits sont sélectionnés parmi 56, ensuite :

- La fonction ***f*** est appliquée à la partie droite (noté R) des données.
- La sortie de '***f***' (32 bits) est combinée avec la moitié de gauche (noté L) par XOR.
- Le résultat devient la nouvelle moitié de droite.
- L'antienne moitié de droite devient la nouvelle moitié de gauche. Donc, ce qui suit un réseau de ***Feistel*** défini par la relation suivante :

$$\begin{cases} Li = Ri\text{-}1 \\ Ri = Li\text{-}1 \oplus f(Ri\text{-}1, Ki) \end{cases}$$

Tel que la fonction ***f*** est (voir figure 17):

$$f(Ri\text{-}1, Ki) = P(S(E(Ri\text{-}1) \oplus Ki)).$$

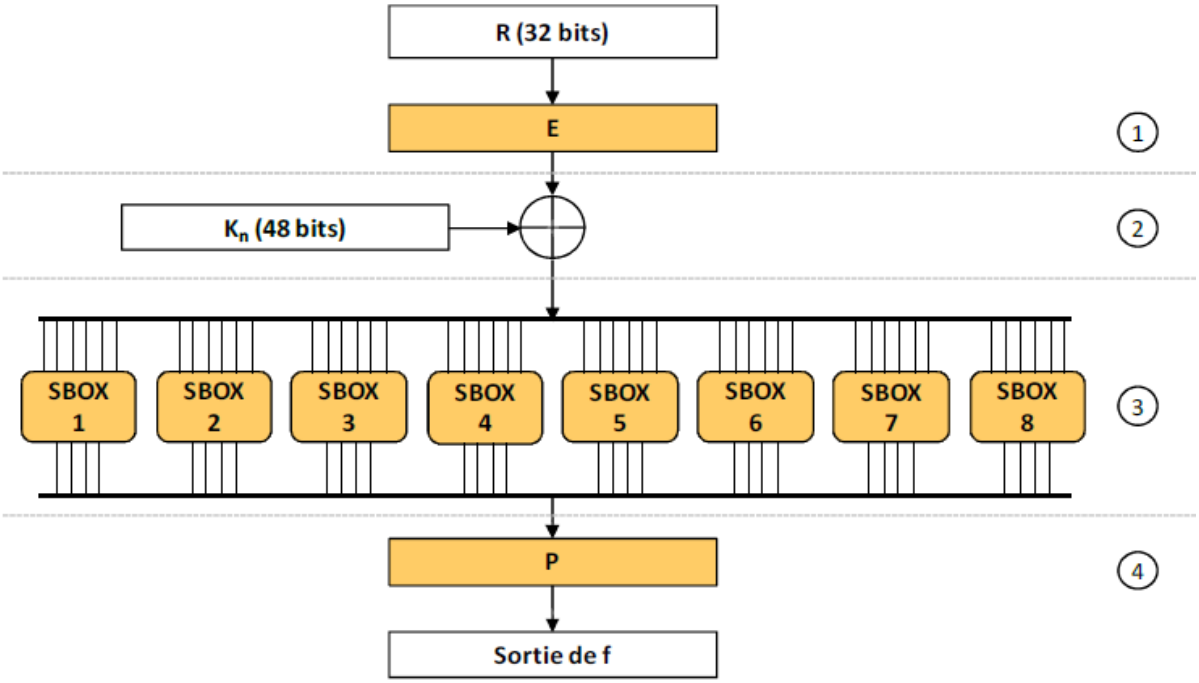


Figure-17: La fonction *f* du DES [24].

Cette fonction *f* se compose de 4 transformations qui sont :

1. **Permutation Expansive :**

La permutation expansive permet de produire en sortie un ensemble de 48 bits en fonction des 32 bits reçu en entrés. Cette extension est accompagnée d'une permutation selon la table donnée ci-dessous.

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Tab-6 : La table E du DES.

2. OU exclusif (XOR) avec la clef :

Les 48 bits résultant de l'opération présidente sont combinés avec la clé du tour actuel (de 48 bits) grâce à un XOR.

3. Substitution par table-S :

Le résultat de 48 bits est soumis à une opération de substitution à l'aide de huit tables S. Il est découpé en 8 morceaux de 6 bits qui sont injectés dans les huit tables S. Une table S est une matrice d'entiers de 4 bits, composée de 4 lignes et 16 colonnes. Cette table est indexée par les 6 bits d'entrée de la manière suivante:

- ✓ Le bit 1 et le bit 6 indexent la ligne
- ✓ Les bits 2, 3, 4, 5 indexent la colonne

Par exemple : Si le mot d'entrée est (en représentation binaire) *b0b1b2b3b4b5*, alors *b0b5* donne la ligne, *b1b2b3b4* donne la colonne, et alors un mot de 4 bits de la matrice est sélectionné.

Voici les différentes tables S (S-Box) de l'algorithme DES :

S_1

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Tab-7 : S-Box 1 du DES.

S_2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
2	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
3	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

Tab-8 : S-Box 2 du DES.

S_3

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
1	13	7	0	9	3	4	5	10	2	8	5	14	12	11	15	1
2	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
3	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

Tab-9 : S-Box 3 du DES.

S_4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
2	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

Tab-10 : S-Box 4 du DES.

S_5

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
2	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
3	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

Tab-11 : S-Box 5 du DES.

S_6

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
3	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

Tab-11 : S-Box 6 du DES.

S_7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
2	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
3	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

Tab-12 : S-Box 7 du DES.

S_8

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
2	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
3	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Tab-13 : S-Box 8 du DES.

Cette phase de substitution donne 8 blocs de 4 bits qui sont concaténés pour former un seul bloc de 32 bits, ce bloc passe à la *permutation-P*.

4. La permutation P :

Les 32 bits de sortie de la substitution par table S subissent une permutation avant d'être combinés avec les 32 bits de la partie gauche. Elle est définit par la table suivante :

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

Tab-14 : Table de la permutation P du DES.

Finalement, le résultat de la permutation ***P*** est combiné avec la moitié gauche du bloc initial de 64 bits, et une nouvelle ronde commence.

III.2.1.6) La permutation inverse :

La permutation inverse est la dernière opération avant l'obtention du bloc chiffré. Elle est l'inverse de la première permutation (permutation initiale) effectuée au début de l'opération de chiffrement. La table suivante indique comment effectuer cette permutation.

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

Tab-15 : Table de la permutation inverse (IP^{-1}) du DES.

III.2.2) Déchiffrement du DES :

Pour le déchiffrement, la seule différence est que les sous clefs doivent être utilisées dans l'ordre inverse. Si les sous clefs de chiffrement sont K1, …, K16 ; alors les sous clefs de déchiffrement sont K16, …, K1.

Si on a que la clef, l'algorithme qui engendre les sous clefs de déchiffrement est également circulaire et le nombre de décalage *(vers la droite*) à effectuer devient (tab-9) :

Ronde	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Nombre de décalage	0	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Tab-16 : Nombre de décalage de la clef pour le décryptage avec le DES.

III.3) Triple DES :

Le TripleDES (aussi appelé 3DES) est en fait l'algorithme DES appliqué trois fois sur les données. Cette utilisation de trois chiffrements DES a été développée par Walter Tuchman (chef du projet DES chez IBM) **[25]**, il existe en effet d'autres manières d'employer trois fois DES mais elles ne sont pas forcément sûres. La version de Tuchman utilise un chiffrement, suivi d'un déchiffrement pour se conclure à nouveau par un chiffrement. Le Triple DES est utilisé avec deux ou trois clés différentes.

III.3.1) le chiffrement avec TripleDES :

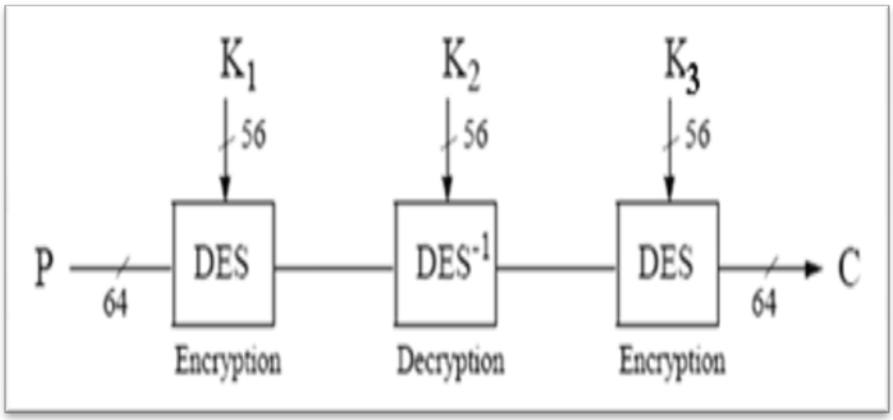


Figure-18 : chiffrement TripleDES.

III.3.2) le déchiffrement avec TripleDES :

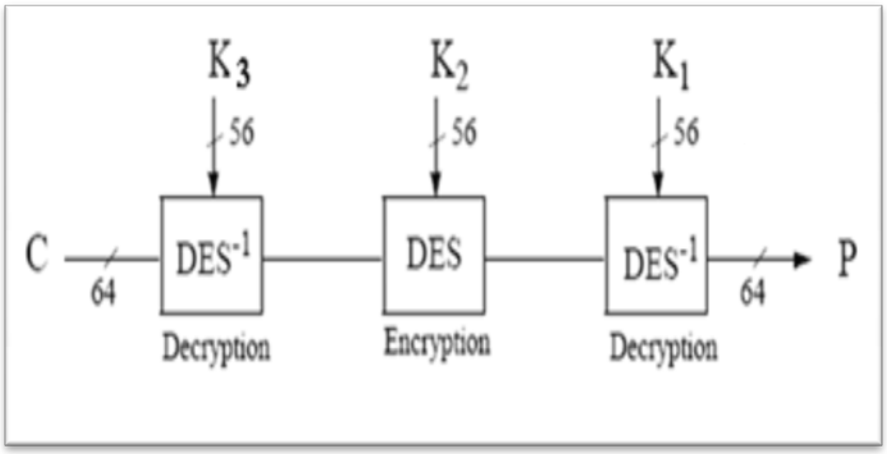


Figure-19 : déchiffrement TripleDES.

III.4) Présentation générale de l'algorithme AES :

L'AES est le sigle d'Advanced Encryption Standard, en français « standard de chiffrement avancé », développé par deux chercheurs belges - Vincent Rijmen et Joan Daemen. Cet algorithme a été officiellement approuvé comme standard le 6 décembre 2001 **[26]**. L'AES est un algorithme de chiffrement par bloc, il permet de chiffrer un texte en clair (Plaintext) constitué de 128 bits de données à l'aide d'une clef secrète constituée de 128, 192 ou 256 bits. L'algorithme se compose principalement d'un module "ronde" qui sera itéré 10, 12 ou 14 fois suivant la taille de la clef secrète utilisée. Une ronde d'AES est constituée de 4 opérations : *SubBytes*, *ShiftRows*, *MixColumns* et *Add Round Key*. L'opération *MixColumns* n'est pas effectuée lors de la dernière ronde. Nous considérerons par la suite que la taille de la clef est 128 bits.

III.4.1) Le chiffrement « Cipher » :

Le principe du chiffrement de cet algorithme est présenté sur la figure suivante (Figure-19).

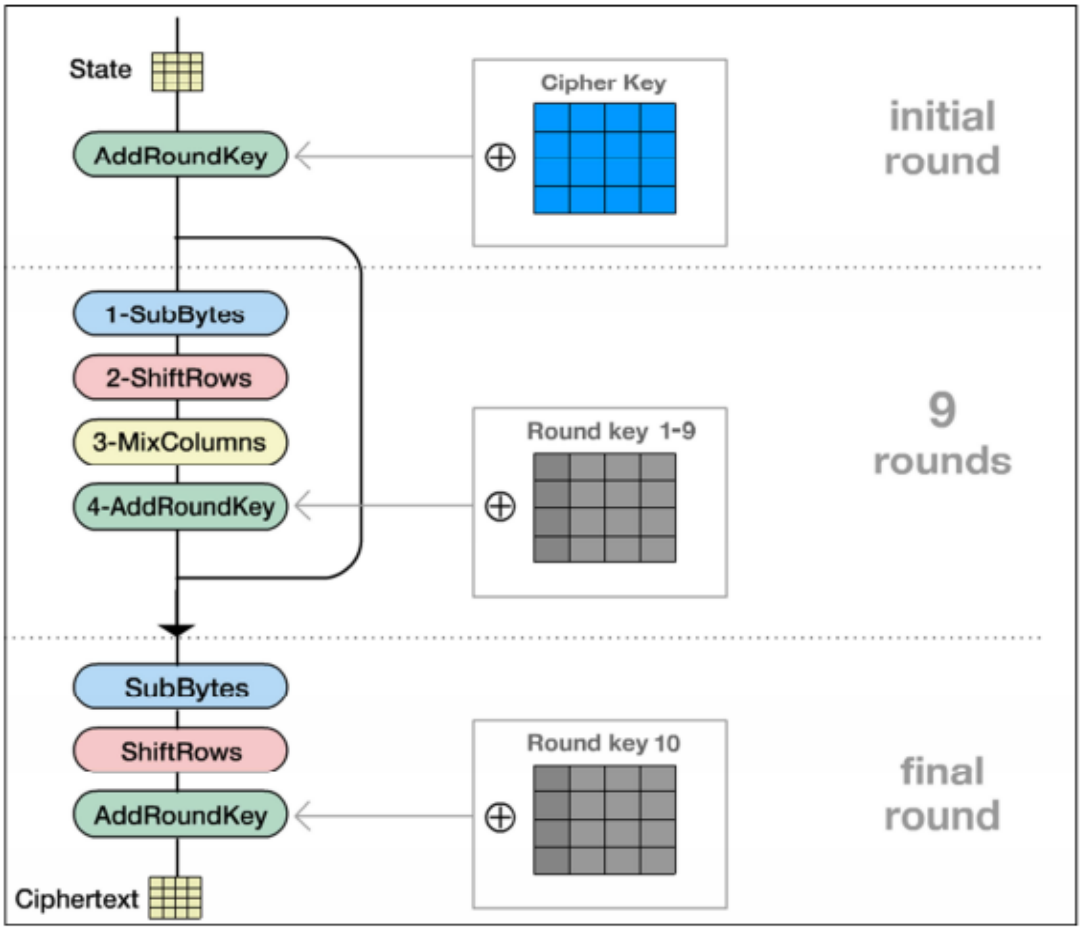


Figure-20 : processus de chiffrement [27].

Les différentes opérations agissent sur des blocs de données de 128 bits organisés sous forme de matrice d'octet 4 * 4.

a) L'opération SubBytes() :

C'est une opération non linéaire qui s'applique à chacun des octets de la matrice. Cette opération consiste à remplacer chaque octet par une autre valeur (figure-20). La substitution se fait en utilisant une table appelée S-Box (tab-10). Chaque octet S_{lc} (lc pour ligne/colonne) est représenté par deux nombres de 4 bits x et y. Les octets (notés sous forme hexadécimale) contenus dans la table sont les octets de remplacement S'_{lc}. L'intersection de la ligne x avec la colonne y représente la valeur de remplacement de l'octet xy (exemple : si l'octet de départ vaut B8, x = B et y = 8, l'octet de remplacement vaut 6C). Ils sont définis sur la base mathématique des champs finis (ou champs de Galois). Les valeurs de la S-Box sont construites par deux fonctions : la première consiste à prendre l'inverse de l'octet dans GF(2^8) sachant que l'octet xy = 00 est son propre inverse, et la seconde consiste à lui appliquer une transformation affine dans GF(2).

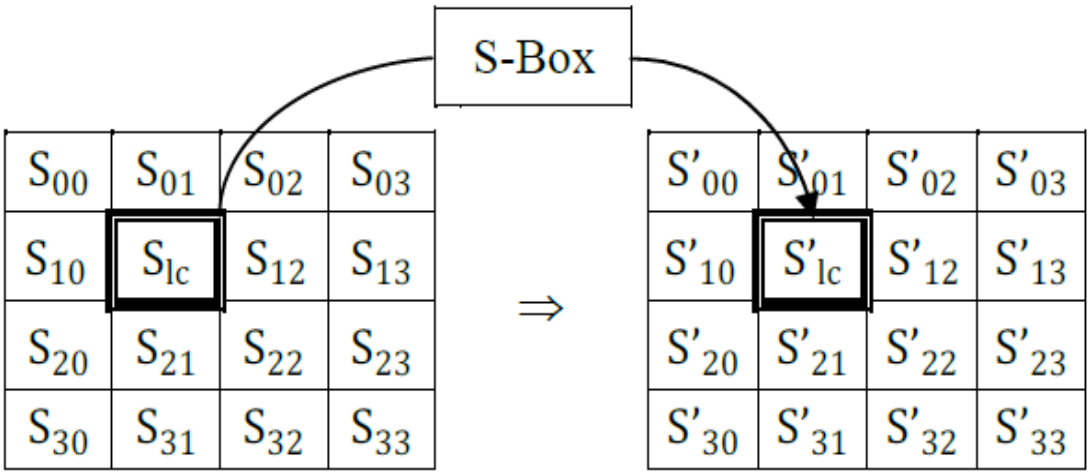


Figure-21: Principe de l'opération SubBytes().

La « S-box » utilisée dans cette fonction peut être présentée sous la forme d'un tableau de 16 lignes et 16 colonnes de valeurs hexadécimales.

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Tab-17: S-box de L'AES.

b) L'opération ShiftRows() :

L'opération ShiftRows est un décalage circulaire des lignes de données de la matrice. La première ligne ne subit pas de décalage, la seconde, la troisième et la quatrième se décalent de façon circulaire vers la gauche respectivement de un, deux et trois octet.

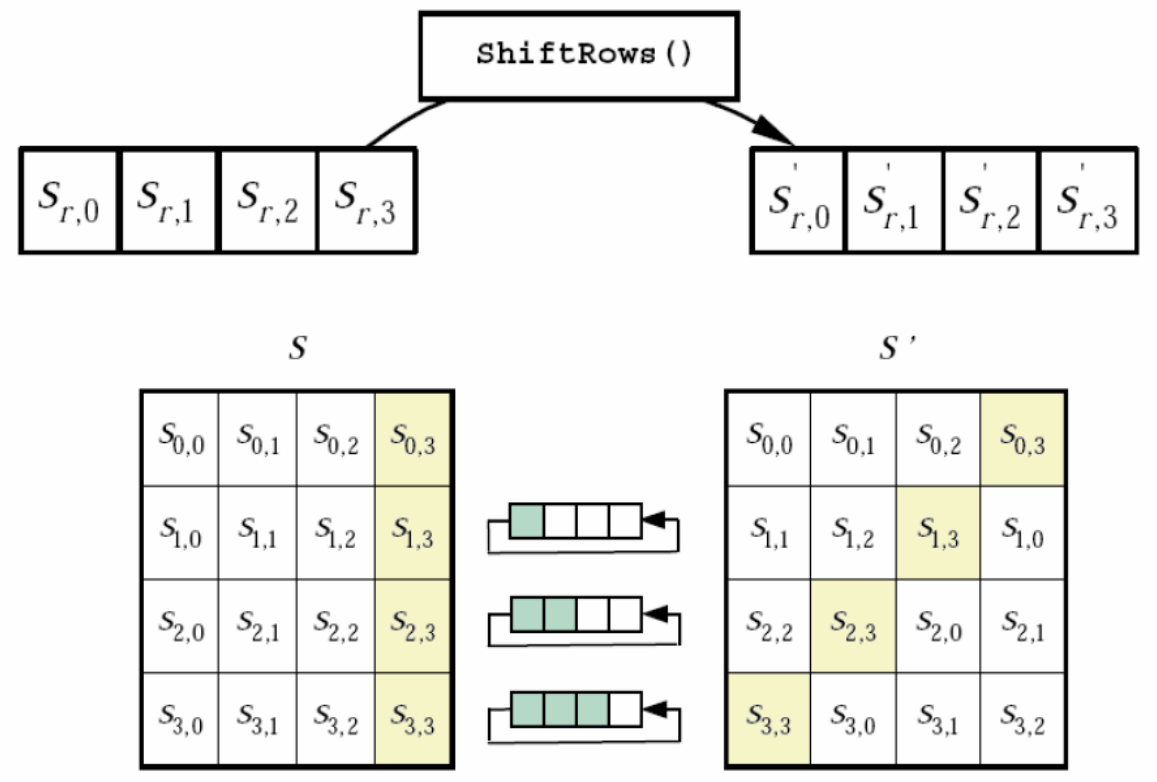


Figure-22 : Principe de l'opération ShiftRows().

c) **L'opération MixColumns() :**

C'est une transformation linéaire : un produit matriciel utilisant les 4 octets d'une colonne. Les colonnes sont traitées comme des polynômes de 4 termes (dans GF(2^8)) et multipliées modulo $x^4 + 1$ avec les polynômes fixes donnés dans la figure suivante :

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c})$$

Figure-23 : Principe de l'opération *MixColumns()*.

La transformation appliquée à un état colonne après colonne.

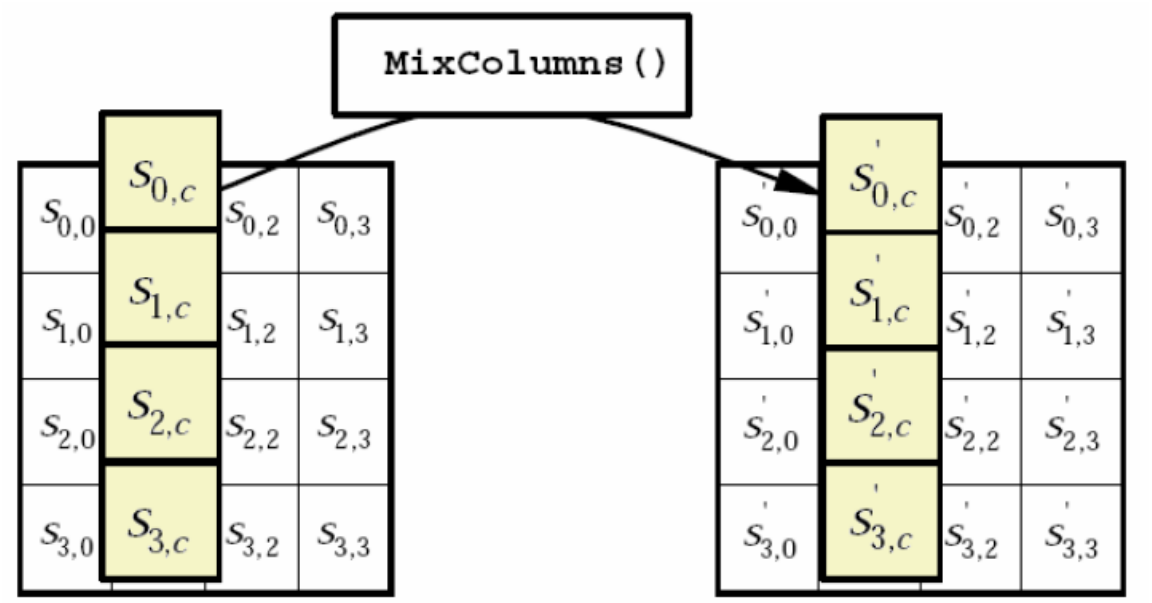


Figure-24 : Principe de l'opération MixColumns().

d) L'opération AddRoundKey :

L'opération *AddRoundKey* est une opération de ou exclusif (XOR) entre la matrice de données et la clef de ronde (Figure-24). Les dix clefs de ronde (une pour chaque ronde) sont calculées à partir de la clef secrète.

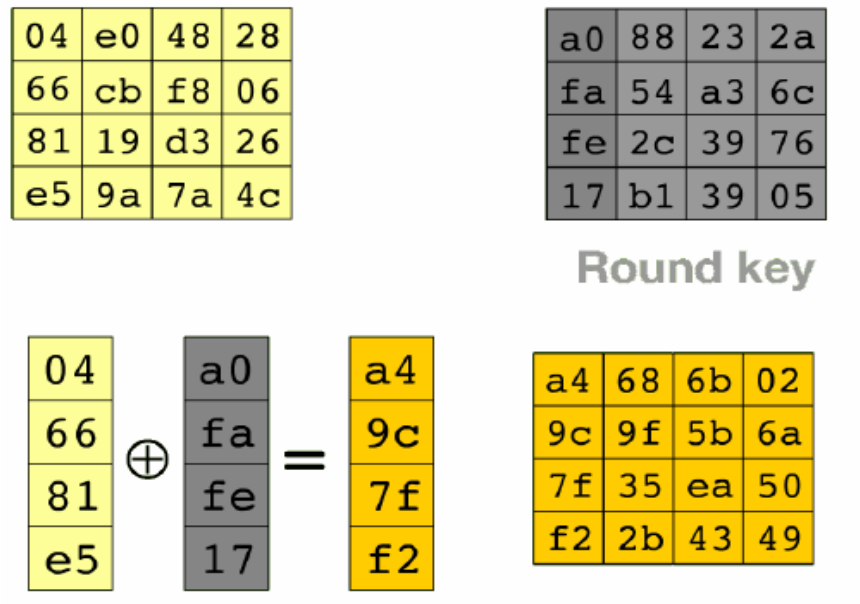


Figure-25. le fonctionnement de AddRoundKey().

Pour obtenir les 10 clefs de rondes, une opération de génération ou d'expansion de clef est réalisée. Ainsi une clef de ronde différente est utilisée à chaque ronde.

III.4.1.1) Diversification de la clef *« Key Expansion »* :

Considérons les deux matrices suivantes, Clef Secrète et Rcon :

K_{00}	K_{01}	K_{02}	K_{03}
K_{10}	K_{11}	K_{12}	K_{13}
K_{20}	K_{21}	K_{22}	K_{23}
K_{30}	K_{31}	K_{32}	K_{33}

Clef secrète

01	02	04	08	10	20	40	80	1B	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

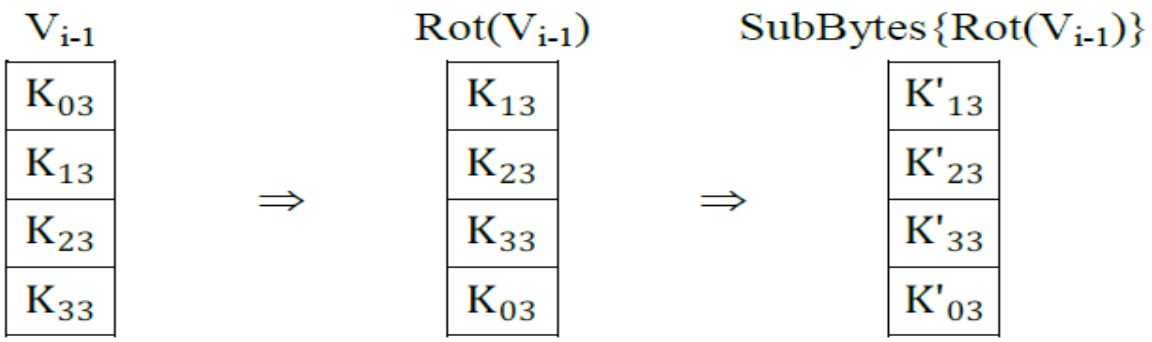
Rcon

La première clef de ronde dérive de la clef secrète, puis la seconde de la première clef et ainsi de suite. En prenant une représentation matricielle de la clef, chaque clef est engendrée colonne par colonne. Le terme vecteur est employé pour parler d'une colonne.

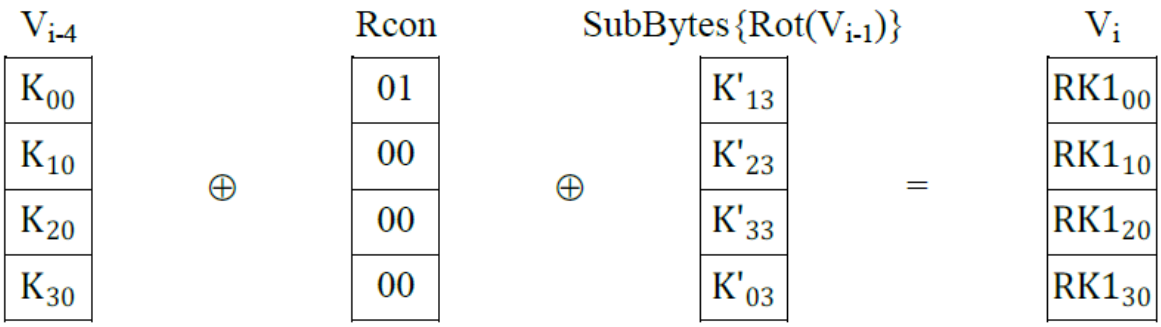
V_{i-4}			V_{i-1}	V_i
K_{00}	K_{01}	K_{02}	K_{03}	?
K_{10}	K_{11}	K_{12}	K_{13}	?
K_{20}	K_{21}	K_{22}	K_{23}	?
K_{30}	K_{31}	K_{32}	K_{33}	?

Clef secrète

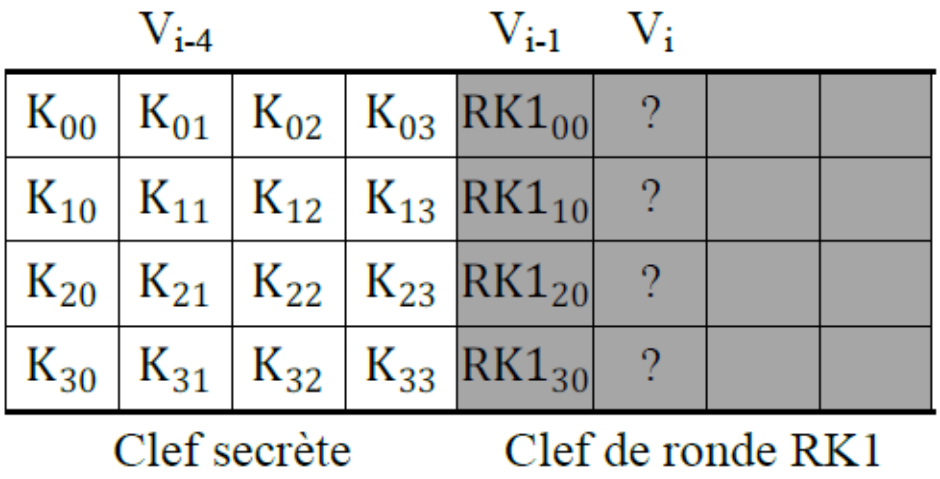
Le vecteur Vi-1 subit une opération dite de rotation qui consiste en un simple décalage des quatre octets du vecteur vers le haut. Ensuite, on applique à ce résultat l'opération SubBytes() :



Le vecteur obtenu (SubBytes{Rot(Vi-1)}) doit encore être additionné modulo 2 avec le vecteur Vi-4, ainsi qu'avec le premier vecteur de la matrice Rcon :



La clef de ronde s'écrit donc sous la forme suivante :



Le deuxième vecteur de la clef de ronde s'obtient par l'opération suivante :

$$Vi = Vi\text{-}4 \oplus Vi\text{-}1.$$

Sur le même principe le troisième et le quatrième vecteur de la clef de ronde s'obtiennent en faisant Vi = Vi-4 ⊕Vi-1 (avec Vi désignant le vecteur à déterminer).

			V_{i-4}			V_{i-1}	V_i
K_{00}	K_{01}	K_{02}	K_{03}	$RK1_{00}$	$RK1_{01}$	$RK1_{02}$	$RK1_{03}$
K_{10}	K_{11}	K_{12}	K_{13}	$RK1_{10}$	$RK1_{11}$	$RK1_{12}$	$RK1_{13}$
K_{20}	K_{21}	K_{22}	K_{23}	$RK1_{20}$	$RK1_{21}$	$RK1_{22}$	$RK1_{23}$
K_{30}	K_{31}	K_{32}	K_{33}	$RK1_{30}$	$RK1_{31}$	$RK1_{32}$	$RK1_{33}$
Clef secrète				Clef de ronde RK1			

Pour déterminer la clef de ronde RK2, on utilise la même procédure en remplaçant la clef secrète par RK1 et en utilisant les données de la deuxième colonne de *Rcon* et ainsi de suite pour les autres clefs de rondes. De fait cette procédure est effectuée au total 10 fois pour générer les 10 clefs de rondes requises pour le chiffrement.

III.4.2) Déchiffrement « Inverse Cipher » :

Le déchiffrement consiste à appliquer les opérations inverses, dans l'ordre inverse et avec des clés de rondes également dans l'ordre inverse. La procédure de génération des clefs de rondes est identique en chiffrement et déchiffrement.

a) L'opération *«InvShiftRows() »* :

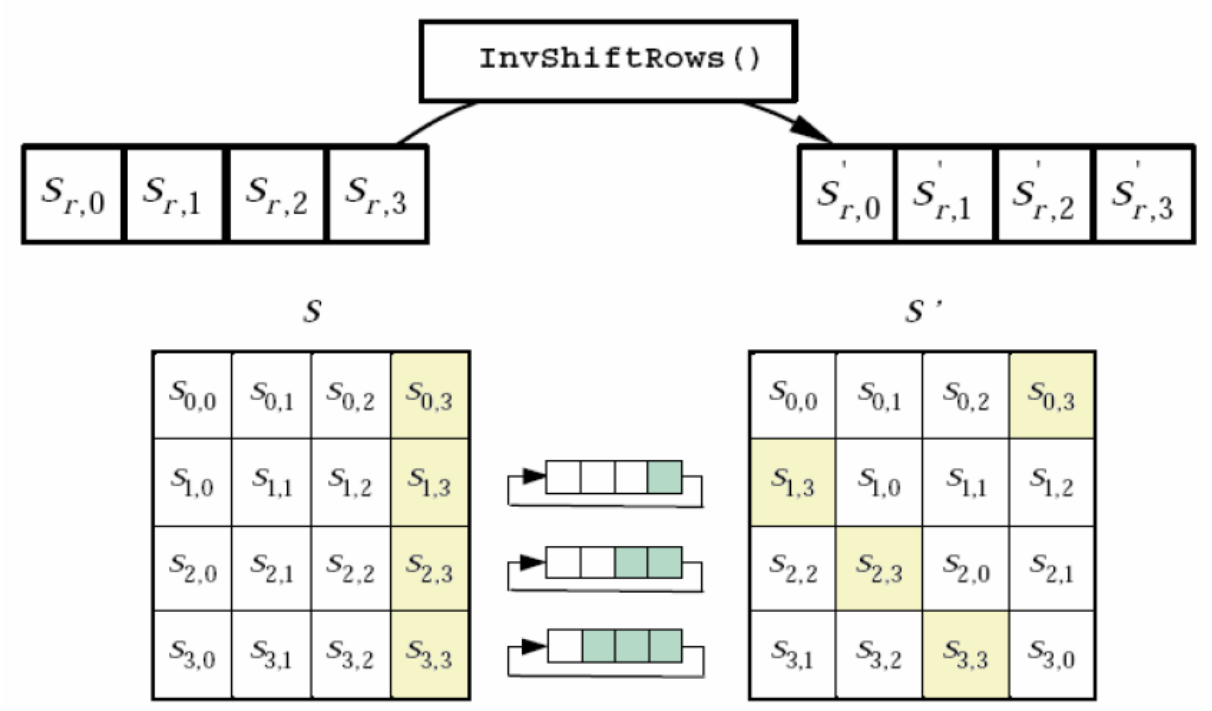


Figure 26 : Principe de l'opération InvShiftRows().

b) L'opération «*InvSubBytes()* » :

InvSubBytes () est l'inverse de la transformation SubBytes(), dans laquelle l'inverse SBOX est appliqué à chaque octet de l'État.

x \ y	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Tab-18 : inverse S-box de l'AES.

Pour si,j = {ed}

s'i,j = InvSubBytes(si,j) = {53}

c) L'opération *«InvMixColumns() » :*

InvMixColumns() est l'inverse de la transformation MixColumns().

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 0e & 0b & 0d & 09 \\ 09 & 0e & 0b & 0d \\ 0d & 09 & 0e & 0b \\ 0b & 0d & 09 & 0e \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix}$$

$$s'_{0,c} = (\{0e\} \bullet s_{0,c}) \oplus (\{0b\} \bullet s_{1,c}) \oplus (\{0d\} \bullet s_{2,c}) \oplus (\{09\} \bullet s_{3,c})$$

$$s'_{1,c} = (\{09\} \bullet s_{0,c}) \oplus (\{0e\} \bullet s_{1,c}) \oplus (\{0b\} \bullet s_{2,c}) \oplus (\{0d\} \bullet s_{3,c})$$

$$s'_{2,c} = (\{0d\} \bullet s_{0,c}) \oplus (\{09\} \bullet s_{1,c}) \oplus (\{0e\} \bullet s_{2,c}) \oplus (\{0b\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{0b\} \bullet s_{0,c}) \oplus (\{0d\} \bullet s_{1,c}) \oplus (\{09\} \bullet s_{2,c}) \oplus (\{0e\} \bullet s_{3,c})$$

Figure-27 : Principe de l'opération *InvMixColumns().*

III.5) conclusion :

La plupart des algorithmes de chiffrement par bloc utilisent des techniques cryptographiques telles que la substitution par tables S, la permutation, l'opération XOR, les décalages et des opérations arithmétiques (addition et multiplication modulo). Mais, les détails diffèrent d'un algorithme à un autre, ainsi que les mécanismes de génération des sous clefs. De plus, le degré de résistance contre les méthodes de cryptanalyse change d'un algorithme à un autre.

Dans ce chapitre nous avons décrit les algorithmes DES, TripleDES et AES. Le chapitre suivant sera consacré à la partie implémentation et évaluation de la performance de ces trois algorithmes.

Chapitre VI

Implémentation et évaluation

IV) Introduction :

Après avoir détaillé dans le chapitre précédent les algorithmes cryptographiques DES, TripleDES et AES, on abordera dans ce chapitre la phase d'implémentation.

Nous commençons, dans la première partie, par préciser l'environnement et les technologies utilisés dans le développement. Ensuite, on s'intéresse à présenter quelques parties de code des principaux modules développés. Et nous terminons par l'évaluation de la performance des algorithmes implémentés.

IV.1) Environnement de travail :

IV.1.1) Le système d'exploitation TinyOS :

TinyOS est un système d'exploitation open source pour les réseaux de capteurs sans-fil. Sa conception a été entièrement réalisée en langage ***NesC***, langage orienté composant syntaxiquement proche du C que nous allons détailler plus loin.

La bibliothèque de composants de TinyOS est particulièrement complète puisqu'on y retrouve des protocoles réseaux, des pilotes de capteurs et des outils d'acquisition de données.

Un programme s'exécutant sur TinyOS est constitué d'une sélection de composants systèmes et de composants développés spécifiquement pour l'application à laquelle il sera destiné (mesure de température, du taux d'humidité…).

Figure-28 : Logo de TinyOs. [28]

TinyOS s'appuie sur un fonctionnement évènementiel, c'est-à-dire qu'il ne devient actif qu'à l'apparition de certains évènements, par exemple l'arrivée d'un message radio. Le reste du temps, le capteur se trouve en état de veille, garantissant une durée de vie maximale connaissant les faibles ressources énergétiques des capteurs.

IV.1.2) Le langage de programmation NesC :

Le système d'exploitation TinyOS s'appuie sur le langage NesC. Celui-ci propose une architecture basée sur des composants, permettant de réduire considérablement la taille mémoire du système et de ses applications. Chaque composant correspond à un élément matériel (LEDs, timer, ADC …) et peut être réutilisé dans différentes applications. Ces applications sont des ensembles de composants associés dans un but précis. Les composants peuvent être des concepts abstraits ou bien des interfaces logicielles aux entrées sorties matérielles de la cible étudiée (carte ou dispositif électronique).

NesC permet de déclarer deux types de fichiers: les modules et les configurations **[29]**.

Le fichier configuration est la définition du ou des composants qui seront utilisés par l'application déployée sur le capteur.

Les modules constituent les briques élémentaires de code et implémentent une ou plusieurs interfaces.

Les interfaces sont des fichiers décrivant les commandes et évènements proposés par le composant qui les implémente.

IV.1.3) TOSSIM :

TOSSIM **[30]** est le simulateur de TinyOs. Il permet de simuler le comportement d'un capteur (envoie/réception de messages via les ondes radios, traitement de l'information, ...) au sein d'un réseau de capteurs. Cependant, 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.

IV.1.4) Le langage java :

Java est un langage de programmation à usage général, évolué et orienté objet dont la syntaxe est proche du C, apparu en fin 1995 début 1996 et développé par SUN Microsystems. Il permet d'exécuter des programmes au travers d'une machine virtuelle (JVM).

IV.1.5) L'IDE NetBeans :

NetBeans est un environnement de développement intégré (IDE) pour java, placé en open source par SUN 2000 **[31]**. En plus de java NetBeans peut supporter d'autre langages comme C, C++, PHP, XML,… etc. Il est conçu en java il permet d'écrire, compiler et d'exécuter des programme. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projet multi-langage,…etc).

NetBeans est gratuit est disponible sous (Windows, Linux, Solaris,…), son installation nécessite l'installation de la JDK (Java Developpement Kit). Dans notre travail on a utilisé la version NetBeans 6.9.

IV.2) Implémentation des algorithmes DES, TripleDES et AES dans le système d'exploitation TinyOS-1.x :

Les algorithmes DES, TripleDES et AES présentés dans le chapitre président sont implémentés de façon à optimiser le temps de calcul et la consommation de la mémoire car la puissance de calcul et la capacité mémoire sont des ressources critique dans le cas des systèmes embarques.

Dans cette partie nous allons présenter la démarche qu'on a suivie pour l'implémentation de ces algorithmes tout en donnant des exemples de simulation.

1- DES

Notre application pour DES se compose de deux parties (à embarquer) écrites en NesC, une partie émettrice qui contient le programme de chiffrement et une partie réceptrice qui contient le programme de déchiffrement.

1.1- Partie émettrice :

Le fonctionnement de cette partie est relativement simple, le capteur doit chiffrer un message de 16 octets, envoyer le message chiffré au nœud récepteur.

Le code est divisé en sept fichiers distincts qui sont: DES.nc, DESM.nc, ChiffreDES.nc, ChiffrementM.nc, Struct_DES_Msg.h, fichier_H.h et le fichier makefile. Nous allons détailler uniquement le fichier DES.nc et les fichiers d'entête (les fichiers .h):

- ***DES.nc* :** est le fichier de configuration qui implémente les différents composants utilisés par l'application. Les composants déclarés dans ce fichier sont :

  - **Main :** est le premier composant exécuté par tinyOS.
  - **GenericComm :** qui implémente les interfaces SendMsg et ReceiveMsg permettant d'envoyer et recevoir des messages respectivement.
  - **DESM :** ce composant implémente l'interface *chiffreDES* qui permet de chiffrer et de déchiffrer un bloc de 64 bits.
  - **chiffrementM :** utilise les interfaces *chiffreDES, SendMsg* et implémente l'interface StdControl obligatoirement (imposé par tinyOS) pour son initialisation, son lancement et son arrêt.

Les relations entre ces différents composants sont montrées dans la figure suivante :

```
Main.StdControl -> DESM;
Main.StdControl -> chiffrementM;
Main.StdControl -> GenericComm.Control;
chiffrementM.ChifferDES ->DESM.ChifferDES;
chiffrementM.SendMsg ->GenericComm.SendMsg[AM_DES_MSG];
```

Figure-29 : Mappage des composants utilisés par l'application.

- **Struct_DES_Msg.h :** Ce fichier contient la structure du message à envoyé.

- **fichier_H.h :** Ce fichier contient la clé de chiffrement, un tableau d'octets représentant le texte en clair et un autre tableau d'octets destiné pour contenir le message chiffré.

La compilation de cette partie (émettrice) pour les capteurs micaz est montrée dans la figure suivante.

```
Administrateur@esdi-62dd0b5a7f /opt/tinyos-1.x/apps/DES_emetteur
$ make micaz
mkdir -p build/micaz
    compiling DES to a micaz binary
ncc -o build/micaz/main.exe -Os -finline-limit=100000 -Wall -Wshadow -DDEF_TOS_A
M_GROUP=0x7d -Wnesc-all -target=micaz -fnesc-cfile=build/micaz/app.c -board=mica
sb -I%T/lib/Deluge -DIDENT_PROGRAM_NAME=\"DES\" -DIDENT_USER_ID=\"Administrateur
\" -DIDENT_HOSTNAME=\"esdi-62dd0b5a7f\" -DIDENT_USER_HASH=0xc85a5dceL -DIDENT_UN
IX_TIME=0x52597894L -DIDENT_UID_HASH=0x5b9932e7L -I/opt/tinyos-1.x/tos/lib/CC242
0Radio DES.nc -lm
C:/tinyos/cygwin/opt/tinyos-1.x/tos/platform/micaz/HPLCC2420InterruptM.nc:161: w
arning: `CCATimer.start' called asynchronously from `CCA.startWait'
C:/tinyos/cygwin/opt/tinyos-1.x/tos/lib/CC2420Radio/CC2420RadioM.nc:116: warning
: `Send.sendDone' called asynchronously from `sendFailed'
fichier_H.h:4: warning: integer constant is too large for "long" type
    compiled DES to build/micaz/main.exe
        14378 bytes in ROM
         1384 bytes in RAM
avr-objcopy --output-target=srec build/micaz/main.exe build/micaz/main.srec
avr-objcopy --output-target=ihex build/micaz/main.exe build/micaz/main.ihex
    writing TOS image

Administrateur@esdi-62dd0b5a7f /opt/tinyos-1.x/apps/DES_emetteur
```

Figure-30 : Compilation de la partie émettrice de l'algorithme DES.

1.2- Partie réceptrice :

Le fonctionnement de cette partie est pratiquement l'opposé de la partie émettrice, le capteur doit recevoir le message chiffré venant du nœud émetteur, déchiffrer le message avec la même clé utilisé pour le chiffrement.

Le code est divisé en sept fichiers qui portent les mêmes noms que ceux de la partie émettrice. Les relations entre les composants qui constituent l'application sont montées dans la figure suivante :

```
Main.StdControl -> DESM;
Main.StdControl -> chiffrementM;
Main.StdControl -> GenericComm.Control;
chiffrementM.ChifferDES ->DESM.ChifferDES;
chiffrementM.ReceiveMsg ->GenericComm.ReceiveMsg[AM_DES_MSG];
```

Figure-31 : Mappage des composants utilisés par l'application.

Une fois le capteur (récepteur) soit prêt à recevoir des messages, l'évènement ReceiveMsg.receive () associée à la réception d'un message se déclenche. Voici un extrait du code de la fonction ReceiveMsg.receive ().

```
event TOS_MsgPtr ReceiveMsg.receive(TOS_MsgPtr pmsg) {

        dbg(DBG_USR2, " ____________________ Récepteur ____________________\n");

        struct DES_Msg *ptr;
        ptr = (struct DES_Msg *)(pmsg->data);

        // déchiffrement
        j=0;
        while(j<sizeof(ptr->tab)){

            for(i=j;i<j+8;i++){
                bloc=(bloc << 8) |(ptr->tab[i] & 0xff);
            }

            bloc=call ChifferDES.dechiffrer(bloc, clef1);

            t=7;
            for(i=j;i<j+8;i++){
                clair[i]=(bloc >> t*8) & 0xFF;
                t--;
            }
            j=j+8;   // passé au bloc suivant
        }
        return pmsg;
}
```

Figure-32 : L'événement associé a la réception d'un message.

La compilation de cette partie (réceptrice) pour les capteurs micaz est montrée dans la figure suivante.

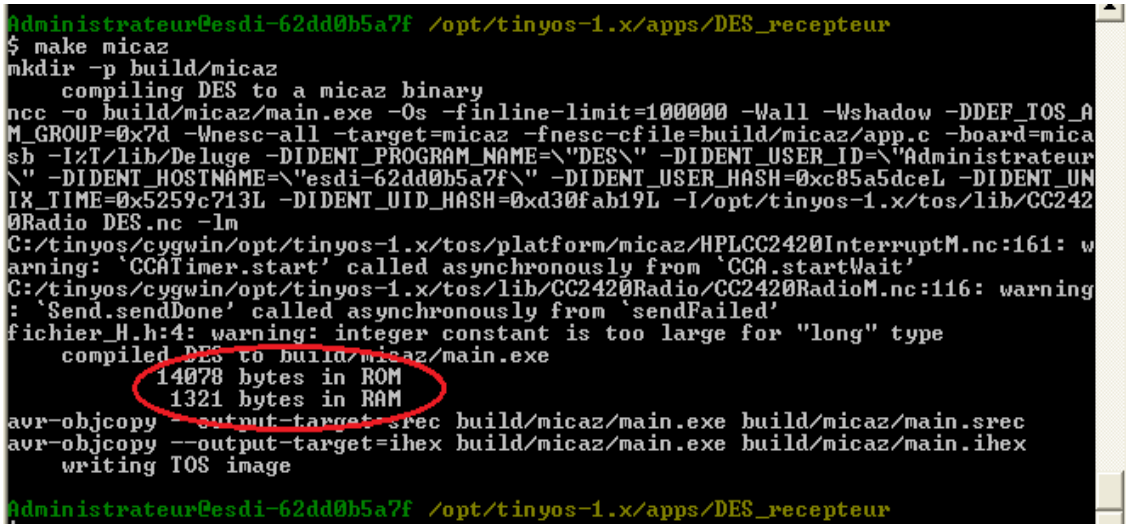


Figure-33 : Compilation de la partie réceptrice de l'algorithme DES.

1.3- Simulation :

Pour tester notre application, on a rassemblé les deux parties précédentes (émettrice et réceptrice) dans le but de simuler le fonctionnement de ce crypto-système. Dans notre simulation, nous avons utilisé deux nœuds de type MicaZ. Un nœud « 1 » qui crypte un texte en clair « *Plain* » et envoie le message chiffré correspondant « *Chiffer* », et un nœud « 0 » qui reçoit et décrypte le message envoyé par le nœud «1» voir la figure suivante.

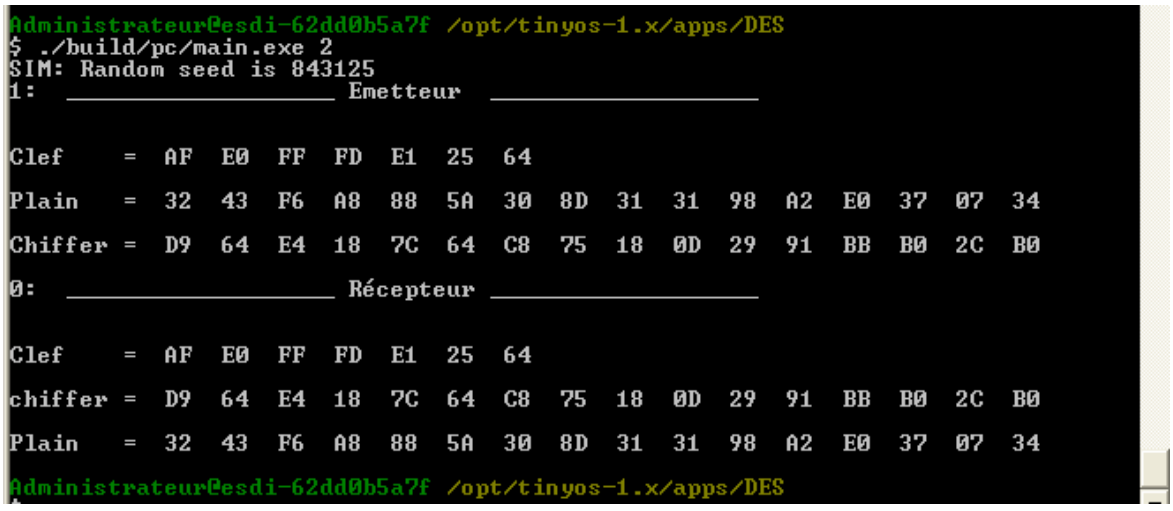


Figure-34 : exemple de simulation de l'algorithme DES.

2- TripleDES :

Notre application pour TripleDES se compose de deux parties (émettrice et réceptrice) dont le code est pratiquement identiques a celui du DES, les seules différences rapportées sont :

- Le chiffrement d'un bloc avec TripleDES est effectué en trois fois (chiffrement DES - > déchiffrement DES - > chiffrement DES).

- Le déchiffrement d'un bloc avec TripleDES est aussi effectué en trois fois (déchiffrement DES - > chiffrement DES - > déchiffrement DES).

Comme on l'avait mentionné dans chapitre précédent, le TripleDES est utilisé avec deux ou trois clés. Dans notre travail nous avons traité les deux cas :

2.1.1- TripleDES avec trois clés :

❖ **Partie émettrice:**

La compilation de cette partie (émettrice) pour les capteurs micaz est montrée dans la figure suivante.

```
Administrateur@esdi-62dd0b5a7f /opt/tinyos-1.x/apps/3DES_168_emetteur
$ make micaz
mkdir -p build/micaz
    compiling T_DES_168 to a micaz binary
ncc -o build/micaz/main.exe -Os -finline-limit=100000 -Wall -Wshadow -DDEF_TOS_A
M_GROUP=0x7d -Wnesc-all -target=micaz -fnesc-cfile=build/micaz/app.c -board=mica
sb -I%T/lib/Deluge -DIDENT_PROGRAM_NAME=\"T_DES_168\" -DIDENT_USER_ID=\"Administ
rateur\" -DIDENT_HOSTNAME=\"esdi-62dd0b5a7f\" -DIDENT_USER_HASH=0xc85a5dceL -DID
ENT_UNIX_TIME=0x525979f2L -DIDENT_UID_HASH=0x7641b55eL -I/opt/tinyos-1.x/tos/lib
/CC2420Radio T_DES_168.nc -lm
C:/tinyos/cygwin/opt/tinyos-1.x/tos/platform/micaz/HPLCC2420InterruptM.nc:161: w
arning: `CCATimer.start' called asynchronously from `CCA.startWait'
C:/tinyos/cygwin/opt/tinyos-1.x/tos/lib/CC2420Radio/CC2420RadioM.nc:116: warning
: `Send.sendDone' called asynchronously from `sendFailed'
fichier_H.h:4: warning: integer constant is too large for "long" type
fichier_H.h:6: warning: integer constant is too large for "long" type
fichier_H.h:8: warning: integer constant is too large for "long" type
    compiled T_DES_168 to build/micaz/main.exe
          21118 bytes in ROM
           1658 bytes in RAM
avr-objcopy --output-target=srec build/micaz/main.exe build/micaz/main.srec
avr-objcopy --output-target=ihex build/micaz/main.exe build/micaz/main.ihex
    writing TOS image

Administrateur@esdi-62dd0b5a7f /opt/tinyos-1.x/apps/3DES_168_emetteur
```

Figure-35 : Compilation de la partie émettrice de l'algorithme TripleDES-168.

❖ **Partie réceptrice :**

La compilation de cette partie (réceptrice) pour les capteurs micaz est montrée dans la figure suivante.

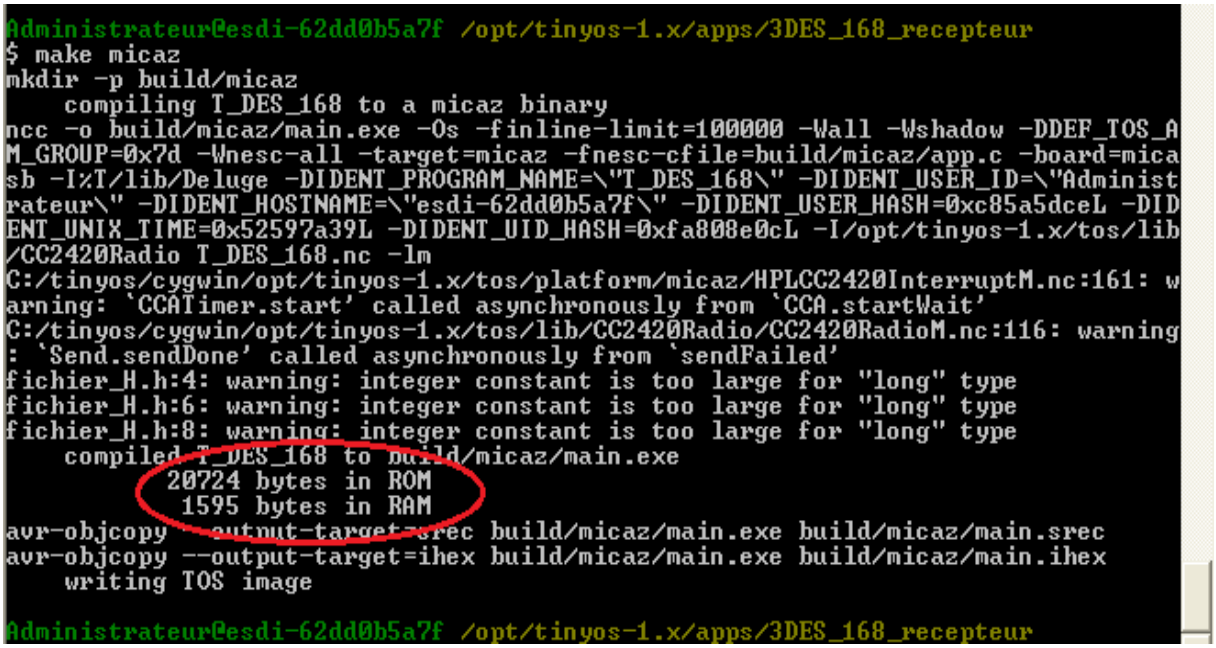


Figure-36 : Compilation de la partie réceptrice de l'algorithme TripleDES-168.

❖ **Simulation :**

Dans cette simulation nous avons utilisé deux nœuds. Un nœud « 1 » qui crypte un message « *Plain* » avec l'algorithme *TripleDES_168* et il le diffuse, et un nœud « 0 » qui reçoit le message chiffré venant du nœud « 1 » et il le déchiffre.

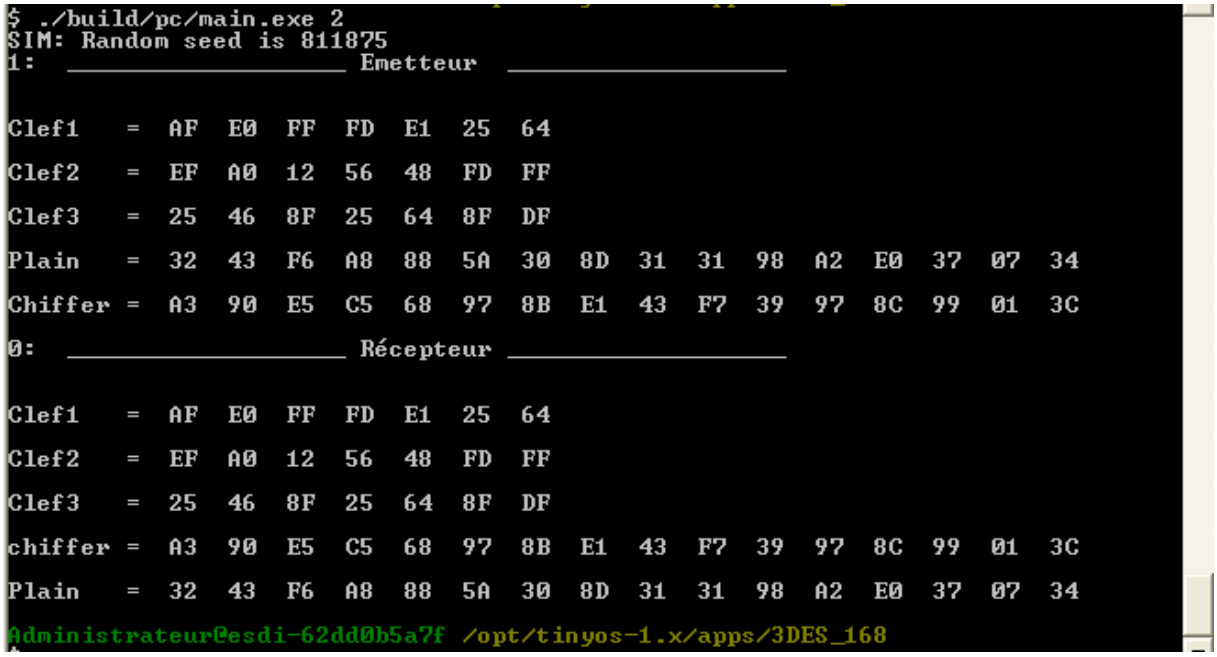


Figure-37 : exemple de simulation de l'algorithme TripleDES-168.

2.1.2- TripleDES avec deux clés :

❖ **Partie émettrice:**

La compilation de cette partie pour les capteurs micaz est montrée dans la figure suivante.

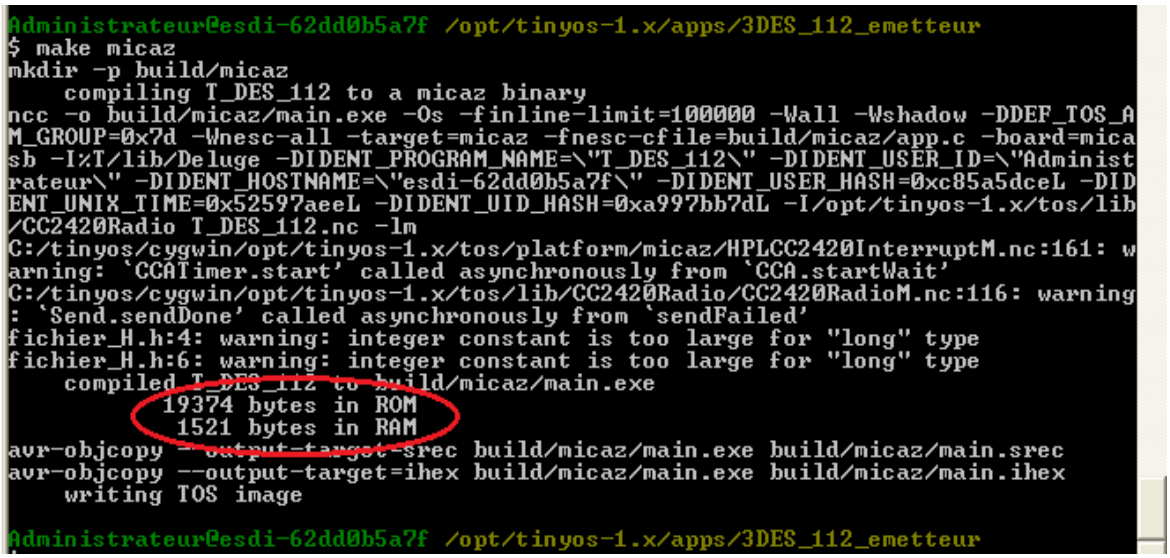


Figure-38 : Compilation de la partie émettrice de l'algorithme TripleDES-112.

❖ **Partie réceptrice:**

La compilation de cette partie pour les plateformes micaz est montrée dans la figure suivante.

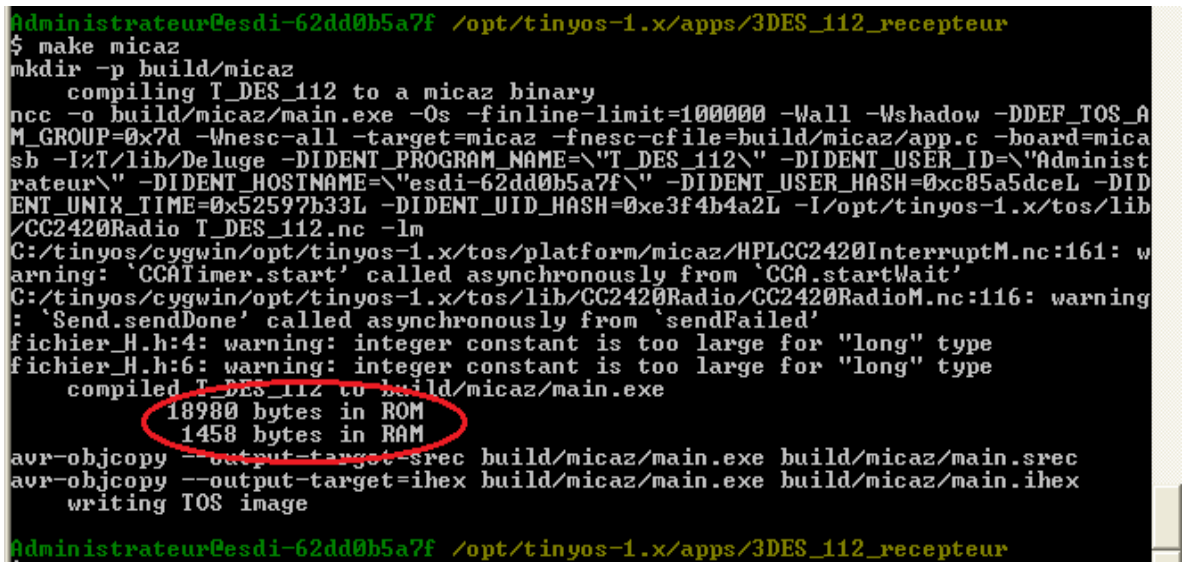


Figure-39 : Compilation de la partie réceptrice de l'algorithme TripleDES-112.

❖ **Simulation :**

Dans cette simulation, le nœud « 1 » crypte le message « *Plain* » avec l'algorithme *TripleDES_112* et il l'envoi au nœud « 0 ». Ce dernier reçoit le message chiffré « *chiffer* » venant du nœud émetteur (nœud 1) et il le décrypte avec la même clé utilisé pour le chiffrement.

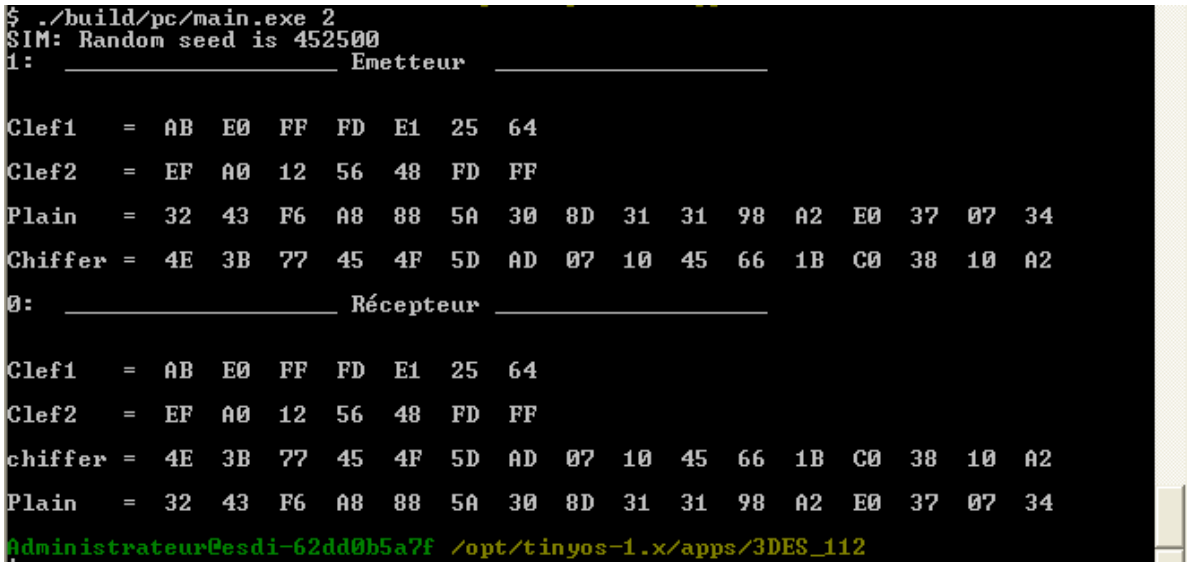


Figure-40 : exemple de simulation de l'algorithme TripleDES-112.

3- AES :

Notre application pour AES se compose de deux parties (à embarquer) écrites en NesC, une partie émettrice qui contient le programme de chiffrement et une partie réceptrice qui contient le programme de déchiffrement.

3.1- Partie émettrice :

Le fonctionnement de cette partie est relativement simple, le capteur doit chiffrer un message de 16 octets, envoyer le message chiffré au nœud récepteur.

Le code est divisé en sept fichiers distincts qui sont: AES.nc, AESM.nc, ChifferM.nc, ChifferAES.nc, Struct_AES_Msg.h, fichier_H.h et le fichier makefile. Nous allons détailler uniquement le fichier AES.nc ainsi que les deux fichiers d'entête (les fichiers .h) :

❖ ***AES.nc* :** est le fichier de configuration qui implémente les différents composants utilisés par l'application. Les composants déclarés dans ce fichier sont :

- **Main :** est le premier composant exécuté par tinyOS.
- **GenericComm :** qui implémente les interfaces SendMsg et ReceiveMsg permettant d'envoyer et recevoir des messages respectivement.

- **AESM :** ce composant implémente l'interface *chiffreAES* qui permet de chiffrer et de déchiffrer un bloc de 128 bits.
- **ChifferM :** utilise les interfaces *chiffreAES, SendMsg* et implémente l'interface StdControl obligatoirement (imposé par tinyOS) pour son initialisation, son lancement et son arrêt.

Les relations entre ces différents composants sont montrées dans la figure suivante :

```
Main.StdControl -> AESM.StdControl;
Main.StdControl -> ChifferM .StdControl;
Main.StdControl -> GenericComm.Control;
ChifferM.ChifferAES ->AESM.ChifferAES;
ChifferM.SendMsg ->GenericComm.SendMsg[AM_AES_MSG];
```

Figure-41 : Mappage des composants utilisés par l'application.

- **Struct_AES_Msg.h :** Ce fichier contient la structure du message à envoyé.

- **fichier_H.h :** ce fichier contient la clé de chiffrement, un tableau d'octets représentant le texte en clair et un autre tableau d'octets destiné pour contenir le message chiffré.

La compilation de cette partie (émettrice) pour les capteurs micaz est montrée dans la figure suivante.

```
Administrateur@esdi-62dd0b5a7f /opt/tinyos-1.x/apps/AES_emetteur
$ make micaz
mkdir -p build/micaz
    compiling AES to a micaz binary
ncc -o build/micaz/main.exe -Os -finline-limit=100000 -Wall -Wshadow -DDEF_TOS_A
M_GROUP=0x7d -Wnesc-all -target=micaz -fnesc-cfile=build/micaz/app.c -board=mica
sb -I%T/lib/Deluge -DIDENT_PROGRAM_NAME=\"AES\" -DIDENT_USER_ID=\"Administrateur
\" -DIDENT_HOSTNAME=\"esdi-62dd0b5a7f\" -DIDENT_USER_HASH=0xc85a5dceL -DIDENT_UN
IX_TIME=0x52475477L -DIDENT_UID_HASH=0x27407f5fL -I/opt/tinyos-1.x/tos/lib/CC242
0Radio AES.nc -lm
C:/tinyos/cygwin/opt/tinyos-1.x/tos/platform/micaz/HPLCC2420InterruptM.nc:161: w
arning: `CCATimer.start' called asynchronously from `CCA.startWait'
C:/tinyos/cygwin/opt/tinyos-1.x/tos/lib/CC2420Radio/CC2420RadioM.nc:116: warning
: `Send.sendDone' called asynchronously from `sendFailed'
    compiled AES to build/micaz/main.exe
        11592 bytes in ROM
          925 bytes in RAM
avr-objcopy --output-target=srec build/micaz/main.exe build/micaz/main.srec
avr-objcopy --output-target=ihex build/micaz/main.exe build/micaz/main.ihex
    writing TOS image

Administrateur@esdi-62dd0b5a7f /opt/tinyos-1.x/apps/AES_emetteur
```

Figure-42 : Compilation de la partie émettrice de l'algorithme AES.

3.2- Partie réceptrice :

Le fonctionnement de cette partie est pratiquement l'opposé de la partie émettrice, le capteur doit recevoir le message chiffré venant du nœud émetteur, déchiffrer le message avec la même clé utilisé pour le chiffrement.

Le code est divisé en sept fichiers qui portent les mêmes noms que ceux de la partie émettrice. Les relations entre les composants qui constituent l'application sont montées dans la figure suivante :

```
Main.StdControl -> AESM.StdControl;
Main.StdControl -> ChifferM .StdControl;
Main.StdControl -> GenericComm.Control;
ChifferM.ChifferAES ->AESM.ChifferAES;
ChifferM.ReceiveMsg ->GenericComm.ReceiveMsg[AM_AES_MSG];
```

Figure-43 : Mappage des composants utilisés par l'application.

Une fois le capteur (récepteur) soit prêt à recevoir des messages, l'évènement ReceiveMsg.receive () associée à la réception d'un message se déclenche. Voici un extrait du code de la fonction ReceiveMsg.receive ().

```
event TOS_MsgPtr ReceiveMsg.receive(TOS_MsgPtr pmsg) {
        dbg(DBG_USR2, " _____ Récepteur _____\n");

        struct AES_Msg *ptr;

        ptr = (struct AES_Msg *)(pmsg->data);

        // déchiffrement
        i=0;
        while(i<sizeof(ptr->tab[i])){

            for(j=0;j<16;j++){
                res[j] = ptr->tab[i+j];
            }

            call ChifferAES.dechiffrer(res, clef, res2);

            for(j=0;j<sizeof(Plaintxt);j++){
                Plaintxt[i+j]=res2[j];
            }

            i=i+16;
        }

        // affichage du texte clair

        return pmsg;
}
```

Figure-44 : L'événement associé a la réception d'un message.

La compilation de cette partie (réceptrice) pour les capteurs micaz est montrée dans la figure suivante.

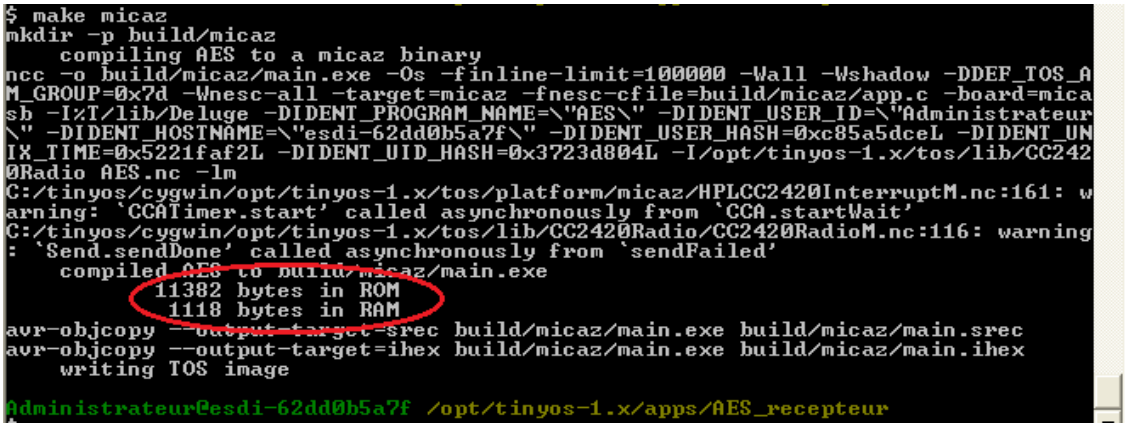


Figure-45 : Compilation de la partie réceptrice de l'algorithme AES.

3.3- Simulation :

Dans cette simulation, le nœud « 1 » crypte le message « *Plain* » avec l'algorithme AES et il l'envoi au nœud « 0 ». Ce dernier reçoit le message chiffré « *chiffer* » venant du nœud émetteur (nœud « 1 ») et il le décrypte avec la même clé utilisé pour le chiffrement.

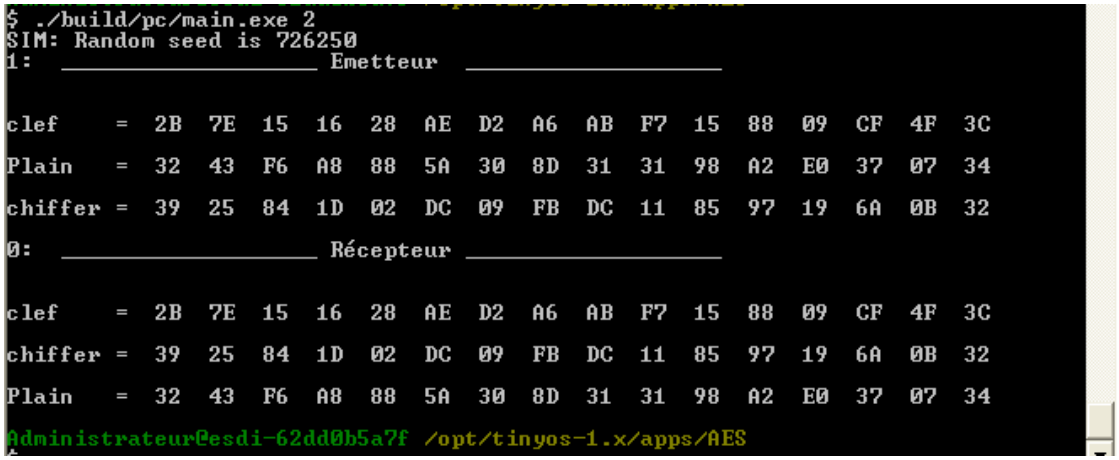


Figure-46 : exemple de simulation de l'algorithme AES.

IV.3) Paramètres d'évaluation :

En raison de la limitation des ressources de réseaux de capteurs, il est essentiel de mesurer l'utilisation de la mémoire, le temps de chiffrement et de déchiffrement de la sécurité mis en œuvre. Dans cette section, nous décrivons les paramètres d'évaluation tels que ROM,

RAM, et la rapidité d'exécution, des différents programmes implémentés. Ces paramètres sont utilisés dans notre évaluation plus tard.

- ✓ ***ROM* :** ou mémoire de code, est la mémoire flash qui est utilisée par le programme quand il est chargé dans le microcontrôleur. Pour la mesure de la performance des composants cryptographiques nous les extraits dans un programme minimal qui ne contient que les "principaux" éléments qui sont nécessaires pour initialiser le nœud. Nous sommes intéressés par la mesure de la ROM utilisée par l'implémentation du composant cryptographique seulement et non pas par l'ensemble du programme qui comprend les "principaux" éléments avec le code pour initialiser le microcontrôleur et appeler les fonctions cryptographiques. Pour ce faire, nous compilons le code «principal» (programme minimal) pour obtenir la taille apportée par le composant principal et soustraire ce nombre de la taille du programme contenant le composant cryptographique testé.

- ✓ ***RAM* :** ou mémoire de données se réfèrent à la volatilité. Cette ressource est extrêmement limitée sur la plupart des systèmes embarqués. Cette mémoire contient des variables globales, l'initialisation des paramètres et interfaces, etc. Comme nous le faisons pour la mémoire ROM, afin d'obtenir la mémoire RAM requise pour le composant cryptographique éprouvé, nous soustrayons la taille de la RAM du programme principal à partir de la taille de la RAM de l'ensemble de programme.

- ✓ ***Temps d'exécution* :** est le temps nécessaire par le composant cryptographique pour être exécuté sur le nœud capteur. Plusieurs interfaces tinyOS permettent de calculer ce temps comme l'interface LocalTimer et SysTimer. Dans notre cas on ne peut pas utiliser ces interfaces pour calculer ce temps parce qu'on ne se dépose pas de capteurs et que le simulateur TOSSIM ne modélise pas le temps d'exécution du CPU. Pour remédier à ce problème on a implémenté les algorithmes avec le langage java, uniquement dans le but de comparer leur temps d'exécutions pour voire quel est le plus rapide d'entre eux. Donc ce dernier paramètre est plutôt la rapidité d'exécution et non pas le temps d'exécution.

IV.4) Evaluation de performance des différents programmes implémentés :

Dans cette partie nous allons évaluer les performances des différents programmes implémentés (pour le chiffrement et pour le de déchiffrement) selon les trois paramètres décris précédemment (RAM, ROM, Rapidité).

Les résultats de performance RAM et ROM ont été mesurés pour les plateformes MICAz à l'aide du système TinyOS-1.x. Tandis que la rapidité d'exécution a été mesurée en java sur PC.

- **Consommation de la RAM**

La table suivante montre l'utilisation de la RAM des programmes de chiffrement :

Algorithme	RAM (en octet)
DES	987
TripleDES-168	1261
TripleDES-112	1124
AES	528

Tab-19 : Consommation de la RAM des programmes de chiffrement.

La table suivante montre l'utilisation de la RAM des programmes de déchiffrement :

Algorithme	RAM (en octet)
DES	987
TripleDES-168	1261
TripleDES-112	1124
AES	784

Tab-20 : Consommation de la RAM des programmes de déchiffrement.

- **Consommation de la ROM**

La table suivante montre l'utilisation de la ROM des programmes de chiffrement :

algorithme	ROM (en octet)
DES	4464
TripleDES-168	11204
TripleDES-112	9460
AES	1678

Tab-21 : Consommation de la ROM des programmes de chiffrement.

La table suivante montre l'utilisation de la ROM des programmes de déchiffrement :

algorithme	ROM (en octet)
DES	4626
TripleDES-168	11272
TripleDES-112	9528
AES	1930

Tab-22 : Consommation de la ROM des programmes de déchiffrement.

- **Temps d'exécution:**

Dans ce qui suit on chiffre et on déchiffre des fichiers de différentes tailles avec les trois algorithmes implémentés (en JAVA) dans le bute de comparer leurs temps d'exécution pour voir quel est le plus rapide d'entre eux. Le calcule du temps d'exécution est fait grâce a la fonction java (méthode) *currentTimeMillis()* qui donne le temps actuel du système en milliseconde.

La figure 47 montre le temps de chiffrement de ces fichiers, tendis que la figure 48 montre le temps correspondant au déchiffrement des fichiers chiffrés précédemment:

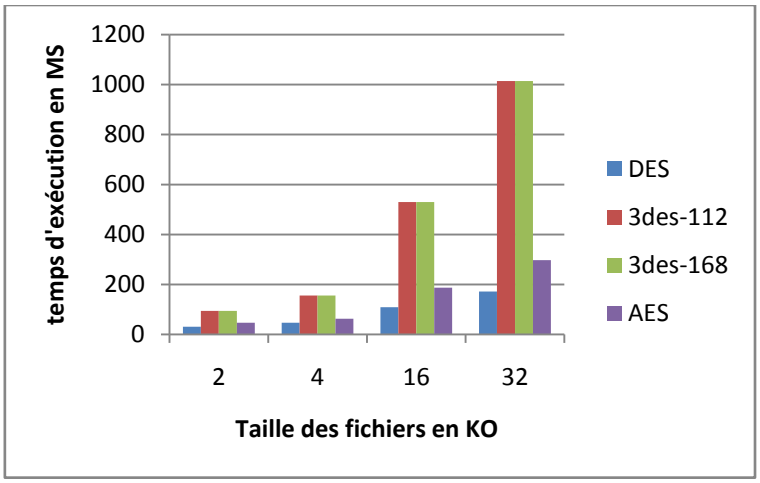


Figure-47 : Temps de chiffrement.

La figure suivante montres le temps de déchiffrement des fichiers chiffré auparavant :

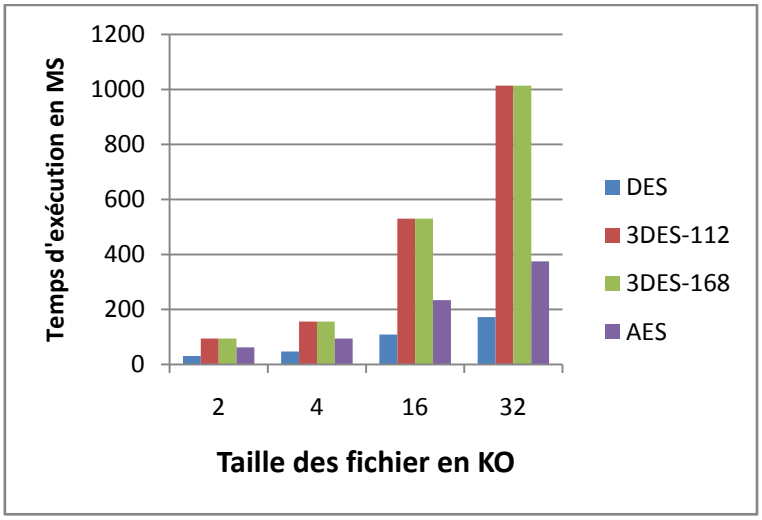


Figure-48 : Temps de déchiffrement.

IV.5) Conclusion :

Dans ce chapitre, nous avons présenté les détails relatifs à l'implémentation de nos applications. Nous avons mesuré les performances de côté consommation de la mémoire et rapidité d'exécution de nos applications. En plus nous avons présenté les architectures de nos systèmes, ainsi que les résultats de simulation.

Conclusion générale

Conclusion générale

Dans ce mémoire, nous avons mis en avant les caractéristiques essentielles des réseaux de capteurs sans fil, ainsi que les besoins et les défis de la sécurité dans ces derniers. Nous avons étudié les mécanismes cryptographiques qui permettent d'offrir le service de sécurité de base pour n'importe quel système basé sur la communication. Nous avons vu aussi que dans ce type de réseaux la cryptographie symétrique est souvent privilégiée par rapport à la cryptographie à clés publique et que les clés sont pré-chargées dans les capteurs avant leur déploiement.

Dans notre projet, nous avons implémenté les algorithmes cryptographiques DES, TripleDES et AES avec deux langages différents, premièrement avec le langage NesC sous la plateforme de programmation adéquate qui est TinyOs. Et deuxièmement avec le langage JAVA. Nous avons mesuré les performances de ces algorithmes selon leurs consommations de la mémoire (RAM et ROM) et leurs rapidités d'exécutions tel que : les résultats de performance RAM et ROM ont été mesurés pour les plateformes MICAz à l'aide du système TinyOS-1.x, tandis que la rapidité d'exécution a été mesurée en java sur PC.

Une perspective pour notre application est d'essayer d'implémenter ces algorithmes sur des capteurs réels et de mesurer le temps d'exécution de chacun d'eux sur ces petits dispositifs.

Références bibliographiques:

[1] : ZNAIDI Wassim : Quelques propositions de solutions pour la sécurité des réseaux de capteurs sans fil. Thèse doctorat, INSA lyon 2010.

[2] : www.securiteinfo.com.

[3] : Messai Mohamed Lamine : Sécurité dans les Réseaux de Capteurs Sans-fil. Mémoire magistère, Bejaia 2007/2008.

[4] : YACINE CHALLAL : Réseaux de Capteurs Sans Fils, cours, 2008.

[5] : Boubiche Djallel Eddine: protocole de routage pour les RCSF. Mémoire magister 2008.

[6] : LEHSAINI Mohamed : Diffusion et couverture basées sur le clustering dans les réseaux de capteurs : application à la domotique. Thèse, université A.B Tlemcen 2009.

[7] : Chérif DIALLO : Techniques d'amélioration du routage et de la formation des clusters multi-sauts dans les réseaux de capteurs sans fil. Thèse, L'Université Pierre et Marie Curie (UPMC) 2010.

[8] : samir athmani : Protocol de sécurité pour les réseaux de capteurs sans fil. Mémoire magister, université Hadj Lakhder batna 2010.

[9] : David Martins : Sécurité dans les réseaux de capteurs sans fil Stéganographie et réseaux de confiance. Thèse, université de France-comité 2010.

[10] : Claude Castelluccia et Aurélien Francillon : Protéger les réseaux de capteurs sans fil. Article, INRIA.

[11] : L. KHELLADI & N. BADACHE : les réseaux de capteurs : rapport de recherche, Université de Bab Ezzouar 2004.

[12] : Manel KHELIFI : Optimisation de la consommation de l'énergie et maximisation de la durée de vie des réseaux capteurs sans fil. Mémoire de Magister, université de Bejaïa 2007/2008.

[13] : Rahim KACIMI : Techniques de conservation d'énergie pour les réseaux de capteurs sans fil. Institut National Polytechnique de Toulouse 2009.

[14] : LABRAOUI Nabila : La sécurité dans les réseaux sans-fil AD HOC. Thèse, Université de Tlemcen 2012.

[15] : David MARTINS et Hervé GUYENNET : état de l'art sur la Sécurité dans les réseaux de capteurs sans fil. LIFC, Équipe SDR 2008 france.

[16] : www.securiteinfo.com.

[17] : Omar Cheikhrouhou et Maryline Maknavicius : « Sécurité des réseaux Mesh », Institut National des Télécommunications, Département LOR , France 2006.

[18] : ZNAIDI Wassim : Quelques propositions de solutions pour la sécurité des réseaux de capteurs sans fil. Thèse doctorat, INSA lyon 2010.

[19] : BOUMERZOUG Hayette: Gestion de la sécurité dans les réseaux de capteurs sans-fil. Mémoire magister, université du Québec à Trois-Rivières 2011.

[20] : Noureddine LASLA : La gestion de clés dans les réseaux de capteurs sans-fil, mémoire magistère, Institut National de formation en Informatique (I.N.I) alger, 2007.

[21] : Emonet jean-Bruno : Algorithmes de chiffrement. Article, centre de ressources informatiques « CRI », juin 2005.

[22] : DAOUI M : cours sécurité réseaux, UMMTO, 2011.

[23] : Emmanuel Bresson : Cryptographie chiffrement symétrique. Cours M1 option cryptographie, université paris XII France.

[24] : Marion DOULCIER : Test intégré de circuits cryptographiques. Thèse, université Montpellier II 2008, France.

[25] : http://www.root-me.org/fr/Documentation/Cryptologie/Triple-DES

[26] : Marion DOULCIER, Marie-Lise FLOTTES, Bruno ROUZEYRE : L'auto-test d'un coeur de chiffrement AES. Article, Université Montpellier II / CNRS 2008.

[27] :http://www.arduer.com/samsung-solid-state-drive-con-hardwareencryption/15747/

[28] : http://fr.wikipedia.org/wiki/TinyOS.

[30]: Laurent MOUNIER, EL Mehdi Damou : Simulation d'un réseau de capteurs avec TinyOS. Article, France 2008.

[31] : Techno-science.net