



UNIVERSITE MOHAMED BOUDIAF - M'SILA
FACULTE DES MATHÉMATIQUES ET
DE L'INFORMATIQUE



DEPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Réseaux

Par : Ghehiouche Amar Abdelmalek

SUJET

**Implémentation des primitives cryptographiques dans les
réseaux de capteurs sans fil**

Soutenu publiquement le : / /2016 devant le jury composé de :

Gasmia Salah

Chikouche Nouredine

Mezrag Fares

Tahri Zouhir

Université de M'sila

Université de M'sila

Université de M'sila

Université de M'sila

Président

Rapporteur

Co-Rapporteur

Examineur

Promotion : 2015 /2016

Remerciements

Je remercie ALLAH, le tout puissant, le miséricordieux, de m'avoir appris ce que j'ignorais, de m'avoir donné la santé et tout dont je necessitais pour l'accomplissement de ce mémoire

Mes remerciements s'adressent également à mon encadrant Monsieur Chikouche Noureddine, pour accepté de diriger ce travail. Son soutien, sa clairvoyance et ses compétences m'ont été d'une aide inestimable

Je remercie également mon encadrant Monsieur Mezrag Fares qui m'ont fourni les outils nécessaires à la réussite de mes études, pour m'avoir donné l'opportunité de travailler sur ce sujet.

Je tiens aussi à exprimer l'honneur qui m'est fait par les membres de jury en acceptant d'évaluer mon travail. Qu'il trouve ici ma reconnaissance et mon respect.

Nous tenons également à remercier toutes les personnes qui ont participé, à titre professionnel ou personnel à la réalisation de ce travail.

Enfin, je ne peux pas oublier de remercier tous mes amis.

Abdelmalek
Merci 

Table des matières

Introduction générale.....	1
----------------------------	---

CHAPITRE 1 - Les réseaux de capteurs sans-fil

1.1 Introduction.....	3
1.2 Environnement sans fil.....	3
1.3 Les réseaux de capteurs sans fils.....	4
1.3.1 Définition.....	4
1.3.2 Objectif de base des RCSFs.....	4
1.4 Capteur.....	5
1.4.1 Définition.....	5
1.4.2 Architecture de base d'un capteur.....	6
1.4.3 Caractéristique de capteur.....	7
1.5 Caractéristiques des réseaux de capteurs.....	7
1.6 Domaine application dans les RCSF.....	8
1.7 Contraintes de conception des RCSFs.....	9
1.7.1 La tolérance aux pannes.....	9
1.7.2 Le coût de production.....	9
1.7.3 Topologie du réseau.....	10
1.7.4 La consommation d'énergie.....	10
1.8 Architecture des RCSFs.....	11
1.8.1 Topologie plate :.....	12
1.8.2 Topologie hiérarchique :.....	12
1.9 Architecture de communication dans les réseaux de capteurs (Pile protocolaire).....	13
1.10 Couverture et Connectivité dans les RCSFs.....	15
1.10.1 Connectivité.....	15
1.10.2 Couverture.....	16
1.11 Les standards de communication pour les RCSF.....	17
1.11.1 Bluetooth.....	17
1.11.2 ZigBee.....	17
1.12 Conclusion.....	18

Table des matières

Chapitre 2 - CRYPTOGRAPHIE SYMETRIQUE

2.1 Introduction	19
2.2 Définition de la cryptographie	19
2.3 Terminologie	20
2.4 Cryptographie Symétrique	21
2.4.1 Chiffrement par bloc	23
2.4.2 Chiffrement par flux	24
2.5 Advanced Encryption Standard (A.E.S)	24
2.5.1 Chiffrement et Déchiffrement :	25
2.5.2 Avantages et limites :	30
2.5.3 Attaque :	30
2.6 Trivium	30
2.6.1 Notation	30
2.6.2 Architecteur	30
2.6.3 Génération de clé	32
2.6.4 Chiffrement	33
2.6.5 Déchiffrement	33
2.7 Conclusion	33

Chapitre 3 - CRYPTOGRAPHIE ASYMETRIQUE

3.1 Introduction	34
3.2 Cryptographie asymétrique	34
3.3 RSA	35
3.3.1 Génération des clés	35
3.3.2 Chiffrement RSA	35
3.3.3 Déchiffrement RSA	35
3.3.4 Exemple numérique pour simplifier décodage RSA	36
3.3.5 Cryptanalyse RSA	36
3.4 Elliptic Curve Cryptography (ECC)	36
3.4.1 ECC pour l'échange de clés	37
3.4.2 ECC pour chiffrement et déchiffrement	37
3.5 Elliptic Curve Integrated Encryption Scheme (ECIES)	38
3.6 McEliece	39
3.6.1 Codes de Goppa	39
3.6.2 Génération de clés	40

Table des matières

3.6.3 Chiffrement	41
3.6.4 Déchiffrement.....	41
3.7 Fonction du hachage.....	41
3.7.1 Les fonctions de hachage	41
3.7.2 Utilisation de hachage.....	42
3.8 Signature numérique.....	42
3.8.1 Conditions	43
3.8.2 Elliptic Curve Digital Signature Algorithm (ECDSA)	43
3.9 Conclusion	44

Chapitre 4 - IMPLEMENTATION ET RESULTATS EXPERIMENTAUX

4.1 Introduction.....	45
4.2 Le système d'exploitation Tinyos	45
4.2.1 Présentation	45
4.2.2 Propriétés	45
4.3 Le langage NesC	46
4.3.1 Concepts de nesC	46
4.3.2 Compiler et exécuter une application nesC.....	46
4.3.3 Bibliothèques cryptographiques	46
4.4 TOSSIM.....	47
4.5 PowerTOSSIMz	47
4.6 AvroraZ.....	48
4.7 Résultats Expérimentaux	48
4.7.1 Paramètres d'implémentation.....	48
4.7.2 Occupation de mémoire (ROM/RAM)	48
4.7.3 Consommation d'énergie	51
4.7.4 Temps de traitement	52
4.8 Discussion	53
4.9 Conclusion	54
Conclusion générale.....	55
Bibliographie.....	56
Annexe	60

Liste des figures

CHAPITRE 1 - Les réseaux de capteurs sans-fil

Figure 1.1 : Différentes catégories des réseaux sans fils	4
Figure 1.2: Exemple des capteurs.....	5
Figure 1.3: Architecture matérielle d'un capteur.	6
Figure 1.4: Quelque domaines d'application pour les RCSFs	9
Figure 1.5: Consommation d'énergie en captage, traitement et transmission	10
Figure 1.6: Schéma général d'un réseau de capteurs	11
Figure 1.7: Exemple d'une topologie plate d'un RCSF	12
Figure 1.8: Exemple d'une topologie hiérarchique d'un RCSF	13
Figure 1.9: L'architecture de communication dans les réseaux de senseurs sans fil	14
Figure 1.10: Connectivité d'un RCSF	16
Figure 1.11: (a) Couverture d'une région. (b) couverture de points	16

Chapitre 2 - CRYPTOGRAPHIE SYMETRIQUE

Figure 2.1: Principe du chiffrement.	20
Figure 2.2: Chiffrement Symétrique.	21
Figure 2.3: principe de chiffrement par bloc.....	23
Figure 2.4: Schéma de chiffrement et déchiffrement par flux	24
Figure 2.5: Schéma des différentes étapes de (AES).....	25
Figure 2.6: Pseudo-code de chiffrement	26
Figure 2.7: Fonction SubByte.....	27
Figure 2.8: S-Box inversible	27
Figure 2.9: Schéma de l'étape ShiftRow.....	28
Figure 2.10 : MixColumn	28
Figure 2.11: AddRound Key	29
Figure 2.12: Expansion de la clé	29
Figure 2.13 : illustre cœur de Trivium.	31
Figure 2.14 : Structure interne de Trivium	31

Chapitre 3 - CRYPTOGRAPHIE ASYMETRIQUE

Figure 3.1 : Chiffrement Asymétrique.	34
Figure 3.2 : Protocole de chiffrement ECIES	39
Figure 3.3: principe de hachage.	41
Figure 3.4: fonction de hachage.	42
Figure 3.5: Protocole de signature numérique ECDSA	44

Chapitre 4 - IMPLEMENTATION ET RESULTATS EXPERIMENTAUX

Figure 4.1 : Evaluation de l'occupation de mémoire de McEliece (N=64).	50
Figure 4.2 : Evaluation de l'occupation de mémoire de McEliece (N=1024)	50

INTRODUCTION GENERALE

La convergence des technologies de communication sans-fil et la micro-électronique a permis la création d'une combinaison entre les systèmes distribués et les systèmes embarqués ayant engendré les Réseaux de Capteurs Sans-fil (RCSF) ou en anglais Wireless Sensor Networks (WSN). Les RCSF sont formés par des petits outils électroniques, autonomes, contenant des capteurs. Ces réseaux sont capables de contrôler un endroit ou un événement important, de produire des informations nécessaires par l'ensemble des mesures détectées par divers capteurs et de les communiquer ensuite à travers les ondes radio à l'utilisateur.

Les RCSF sont appliqués dans différentes applications, telles que : militaires, médicales, environnementales domotiques, etc. Ces applications ont souvent besoin d'un niveau de sécurité important. Parmi les caractéristiques importantes de capteurs dans ce type de réseau, on cite la limitation de l'énergie, l'espace de mémoire et la puissance de calcul. Dans l'état de l'art de la cryptographie, on peut trouver plusieurs primitives cryptographiques développées. Les principales catégories de ces primitives sont : cryptosystèmes à clé privée, cryptosystèmes à clé public, signatures numériques et les fonctions de hachage. Pour cela, il faut choisir bien les primitives cryptographiques qui conviennent avec les caractéristiques des capteurs dans les RCSF.

Dans ce mémoire, on présente un état de l'art des principales primitives cryptographiques, et particulièrement les cryptosystèmes à clé public et cryptosystèmes à clé privée. Notre objectif est de pouvoir implémenter des différentes primitives étudiées dans un capteur pour évaluer ses performances en terme de consommation de l'énergie, occupation de l'espace de mémoire, et temps d'exécution. Nos résultats expérimentaux sont basés sur l'utilisation de deux types de simulateurs : TOSSIM [43] et Avrora [42].

Organisation du mémoire

Ce mémoire est organisé de la manière suivante :

Chapitre 1 :

Présentation des caractéristiques liées aux réseaux de capteurs sans fil, leurs domaines d'applications et leur contrainte. Ainsi l'architecteur de communication (pile protocolaire).

Chapitre 2 :

Dans ce chapitre nous commençons par citer une définition de la cryptographie et quelque conception principale. Ensuite nous présentons deux algorithmes de chiffrement symétrique (à clé privée).

Chapitre 3 :

Dans ce chapitre nous présentons des algorithmes cryptographiques asymétriques (à clé public), ainsi que la signature numérique.

Chapitre 4 :

Ce chapitre effectue une étude de simulation en utilisant l'environnement TinyOS et les simulateurs TOSSIM et Avrora. Ensuite on fait l'implémentation des différentes primitives étudiées avec l'évaluation des performances de ces primitives.

Nous finalisons ce mémoire par une conclusion et des perspectives, où nous présentons nos remarques concluantes et nos suggestions pour une recherche future.

Chapitre 1

LES RESEAUX DE CAPTEURS SANS-FIL

CHAPITRE 1

LES RESEAUX DE CAPTEURS SANS-FIL

1.1 Introduction

Grâce aux avancées technologiques, il devient aujourd'hui envisageable de produire en masse des systèmes d'une taille extrêmement réduite et embarquant des unités de calcul et de communication sans fil pour un coût réduit. Ayant ces caractéristiques, les nœuds capteurs sont capables de générer et d'échanger des données d'une manière autonome et complètement transparente pour les utilisateurs.

Les réseaux de capteurs représentent actuellement un nouveau domaine, en plein développement, émergeant des innovations des technologies de communication. Les RCSF permettent de faciliter le suivi et le contrôle à distance de l'environnement physique avec une meilleure précision. Ils peuvent aussi être déployés pour exploiter diverses applications (environnementales, militaires, médicales, etc.).

Dans la suite de ce chapitre, Nous commencerons par présentation de l'environnement sans-fil, une définition d'un capteur, son architecture. Ensuite, les domaines d'applications, contraintes de conception et Architecteur des RCSFs. Ainsi les standards de communication pour RCSF. Enfin la conclusion.

1.2 Environnement sans fil

La classification des réseaux sans fil peut s'effectuée selon le périmètre géographique ou l'infrastructure du réseau. Selon le périmètre géographique dans lequel il se situe, un réseau sans fil appartient à l'une des catégories suivantes (Figure 1.1) : réseaux personnels sans fil (WPAN), réseaux métropolitains sans fil (WMAN), réseaux locaux sans fil(WLAN) et réseaux étendus sans fil(WWAN).

La Figure 1.1 illustre les différentes classes de réseaux, les standards utilisés et leurs zones de couverture.



Figure 0.1 : Différentes catégories des réseaux sans fils [10]

1.3 Les réseaux de capteurs sans fils

1.3.1 Définition

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. Les nœuds capteurs sont des capteurs intelligents "smart sensors", capables d'accomplir trois tâches complémentaires : le relevé d'une grandeur physique, le traitement éventuel de cette information et la communication avec d'autres capteurs. L'ensemble de ces capteurs, déployés pour une application, forme un réseau de capteurs. Le but de celui-ci est de surveiller une zone géographique, et parfois d'agir sur celle-ci (il s'agit alors de réseaux de capteurs-actionneurs) [29].

1.3.2 Objectif de base des RCSFs

Les objectifs de base des réseaux de capteurs sans-fil dépendent généralement des applications, cependant les tâches suivantes sont communes à plusieurs applications : [29]

- ❖ Déterminer les valeurs de quelques paramètres suivant une situation donnée. Par exemple, dans un réseau environnemental, on peut chercher à connaître la température, la pression atmosphérique, la quantité de la lumière du soleil, et l'humidité relative dans un nombre de sites, etc.
- ❖ Détecter l'occurrence des événements dont on est intéressé et estimer les paramètres des événements détectés. Dans les réseaux de contrôle de trafic, on peut vouloir détecter le mouvement de véhicules à travers une intersection et estimer la vitesse et la direction du véhicule.
- ❖ Classifier l'objet détecté. Dans un réseau de trafic, un véhicule est-il une voiture, un bus, etc.

1.4 Capteur

1.4.1 Définition

Un capteur est un composant physique, capable d'accomplir trois tâches complémentaires : le relevé d'une grandeur physique, le traitement éventuel de cette information, et la communication avec d'autres capteurs. L'ensemble de ces capteurs déployés pour une application forme un réseau de capteurs. [31]

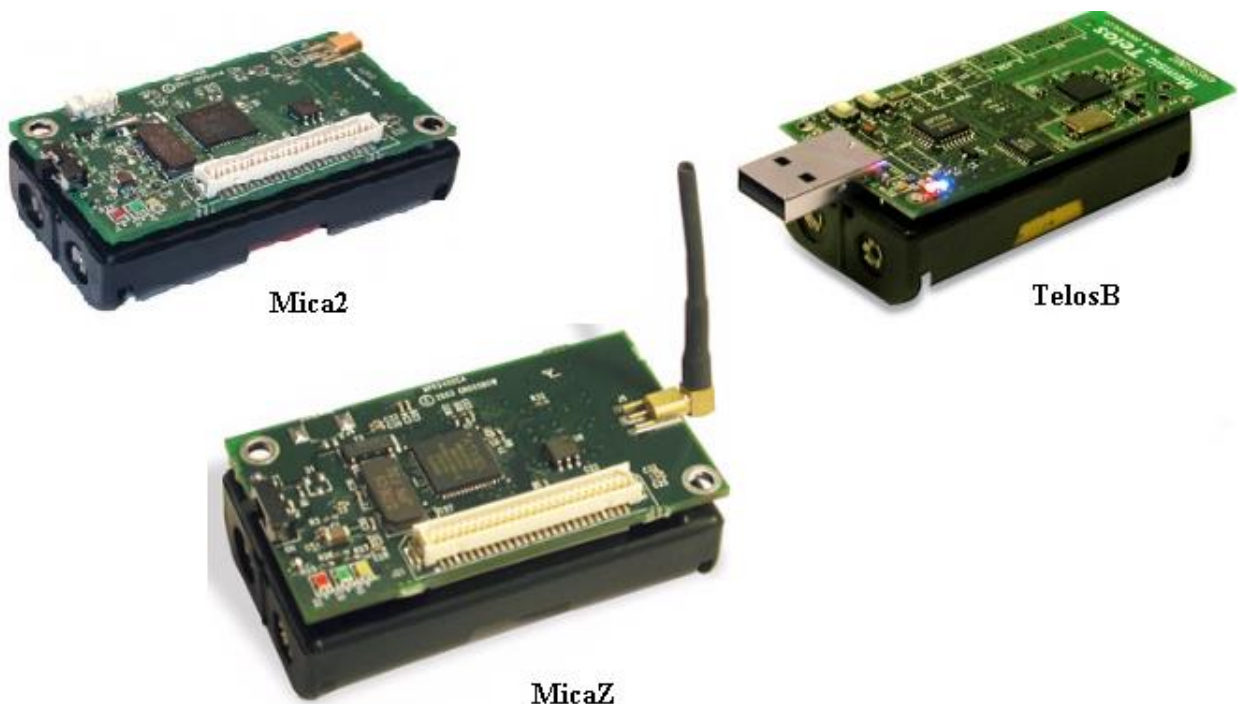


Figure 0.2: Exemple des capteurs

1.4.2 Architecture de base d'un capteur

Un capteur est composé de quatre éléments principaux :

- Unité de captage.
- Calcule (Unité de traitement (processeur), Unité de stockage(Mémoire)).
- Un élément de communication (émetteur / récepteur).
- Unité d'énergie.

Trois composants additionnels peuvent être implantés dans un capteur :

- Un système de recherche d'emplacement.
- Un générateur d'énergie.
- Une unité mobile.

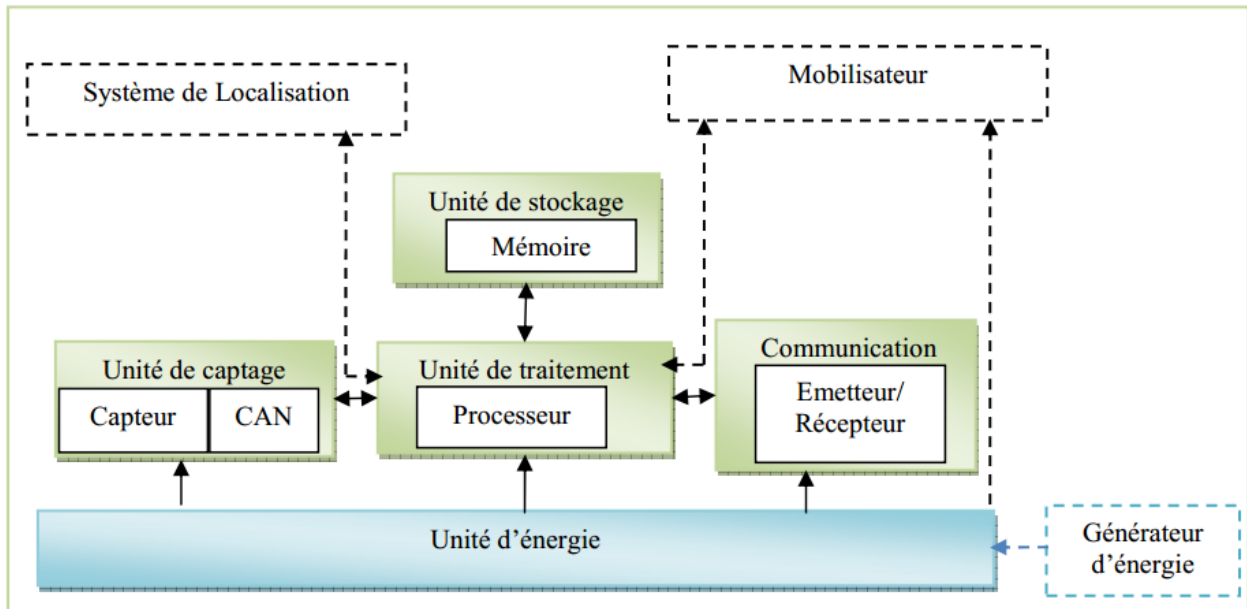


Figure 0.3: Architecture matérielle d'un capteur. [32]

➤ Unité d'énergie :

Un capteur est muni d'une source d'énergie, généralement une batterie, pour alimenter tous ses composants. Les batteries utilisées sont soit rechargeables ou non. Pour cela, l'énergie est la ressource la plus précieuse puisqu'elle influe directement sur la durée de vie des capteurs et donc d'un réseau de capteurs.

➤ Unité de captage :

La fonction principale de l'unité de captage est de capturer ou mesurer les données physiques à partir de l'objet cible. Il est composé de 2 sous-unités : le récepteur (reconnaissant la

grandeur physique à capter) et le transducteur (convertissant le signal du récepteur en signal électrique).

➤ **Calcule :**

- Unité de stockage(Mémoire) :L'unité de stockage inclut la mémoire de programme (dont les instructions sont exécutées par le processeur) et la mémoire de données (pour conserver des données fournies par l'unité de captage et d'autres données locales).
- Unité de traitements (processeur) : Il recueille des données des de l'unité de captage ou d'autres capteurs, effectue un traitement sur ces données (si nécessaire) et décide quand et où les envoyer. Il doit exécuter des programmes et des protocoles de communication différents.

➤ **Unité de communication :**

Cette unité est responsable de toutes les émissions et réceptions de données via un support de communication sans fil.

1.4.3 Caractéristique de capteur

Le processeur est à la base des calculs binaires. Les mémoires RAM et Flash servent pour le stockage, définitif ou non, des données.

Modèle	Microcontrôleur	RAM	Flash	Batterie
Mica2	ATmega128	4 KB	128 KB	2.7-3.3 V
MicaZ	8MHz ATmega128	4 KB	128 KB	2.7-3.3 V
TelosB	1MHz TI MSP430	10 KB	48 KB	1.8-3.6 V

Table 0.1 : Quelques Caractéristiques des capteurs.

1.5 Caractéristiques des réseaux de capteurs

Les principales caractéristiques des réseaux de capteurs se résument dans ce qui suit [28] :

❖ **Densité des nœuds :**

Les réseaux de capteurs se composent généralement d'un nombre très important des nœuds pour garantir une couverture totale de la zone surveillée. Ceci engendre un niveau de surveillance élevé et assure une transmission plus fiable des données sur l'état du champ de capteur.

❖ Topologie dynamique

L'instabilité de la topologie des réseaux de capteurs est le résultat des trois facteurs essentiels suivants :

- La mobilité des nœuds : les nœuds capteurs peuvent être attachés à des objets mobiles qui se déplacent librement et arbitrairement, introduisant ainsi une topologie instable du réseau.
- La défaillance des nœuds : du fait de l'autonomie énergétique limitée des nœuds, la topologie du réseau n'est pas fixée (les nœuds qui épuisent leur énergie, sont considérés comme des nœuds inexistants).
- L'ajout de nouveaux nœuds : de nouveaux nœuds peuvent facilement être rajoutés. Il suffit de placer un nouveau capteur qui soit dans la portée de communication d'au moins un autre nœud capteur du réseau déjà existant.

❖ 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.

❖ Scalabilité

Les réseaux de capteurs peuvent contenir des centaines voire des milliers de nœuds capteurs. Un nombre aussi important engendre beaucoup de transmissions intermodales et nécessite que le nœud « Sink » soit équipé d'une mémoire importante pour stocker les formations reçues.

1.6 Domaine d'application dans les RCSF

Les nœuds capteurs sont utilisés dans une large gamme (thermique, optique, vibrations,...). En effet, la taille de plus en plus réduite des micro-capteurs, le coût de plus en plus faible, ainsi que le support de communication sans fil utilisé, permettent aux réseaux de capteurs d'envahir de nouveaux domaines d'applications tels que le domaine militaire, environnemental, santé, sécurité et commercial.

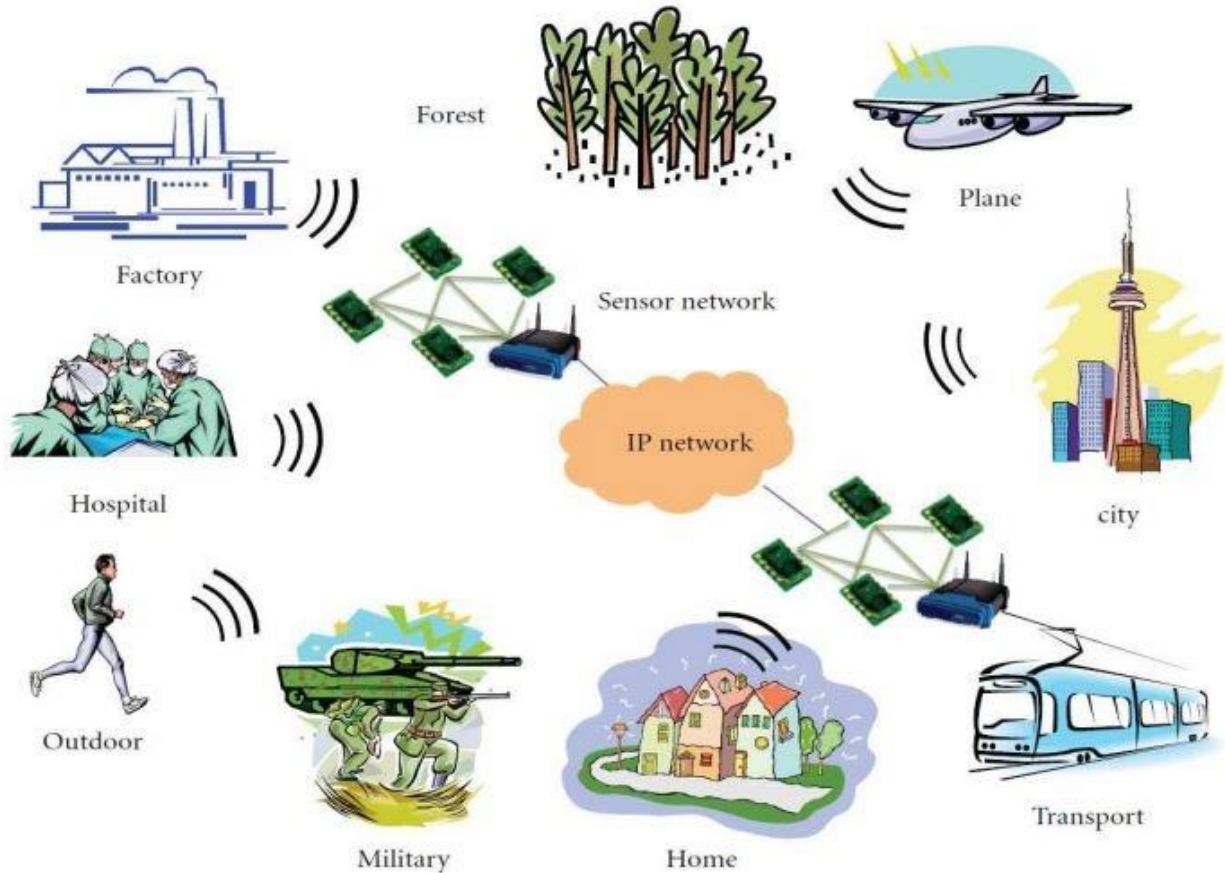


Figure 0.4: Quelques domaines d'application pour les RCSFs

1.7 Contraintes de conception des RCSFs

La conception et la réalisation des réseaux de capteurs sans fil sont influencées par plusieurs paramètres, parmi lesquels nous citons la tolérance aux pannes, le coût de production, Topologie du réseau et la consommation d'énergie. [29]

1.7.1 La tolérance aux pannes

La défaillance ou le blocage de certains nœuds dans un réseau de capteurs peut être engendrés par plusieurs causes, notamment l'épuisement d'énergie, l'endommagement physique, ou les interférences liées à l'environnement. Ces problèmes ne devraient pas affecter le reste du réseau. C'est le principe de la tolérance aux pannes.

1.7.2 Le coût de production

Le coût de production d'un seul micro-capteur est très important pour l'évaluation du coût global du réseau.

1.7.3 Topologie du réseau

Bien que les réseaux de capteurs aient évolué dans de nombreux aspects, ils continuent d'être des réseaux avec des ressources limitées en termes d'énergie, de puissance de calcul, de mémoire et de capacité de communication. Parmi ces contraintes, la consommation d'énergie est d'une importance primordiale, ce qui est démontré par le grand nombre d'algorithmes, des techniques et des protocoles qui ont été développés afin d'économiser l'énergie et par conséquent prolonger la durée de vie du réseau. La maintenance de la topologie est l'une des questions les plus importantes recherchées pour réduire la consommation d'énergie dans les RCSFs.

1.7.4 La consommation d'énergie

Les nœuds capteurs, étant des dispositifs microélectroniques, peuvent être équipés seulement d'une source d'énergie limitée. Dans certains scénarios d'application, il est impossible de réapprovisionner de l'énergie. La durée de vie d'un capteur est donc dépendante de la durée de vie de sa batterie. D'autre part, la retransmission des données, la réorganisation du réseau ainsi que le changement de sa topologie rendent la gestion et la conservation d'énergie d'une haute importance. Cette énergie est consommée par les différentes unités du capteur afin de réaliser les tâches de captage, traitement de données et communication. Cette dernière est l'opération qui consomme le plus d'énergie.

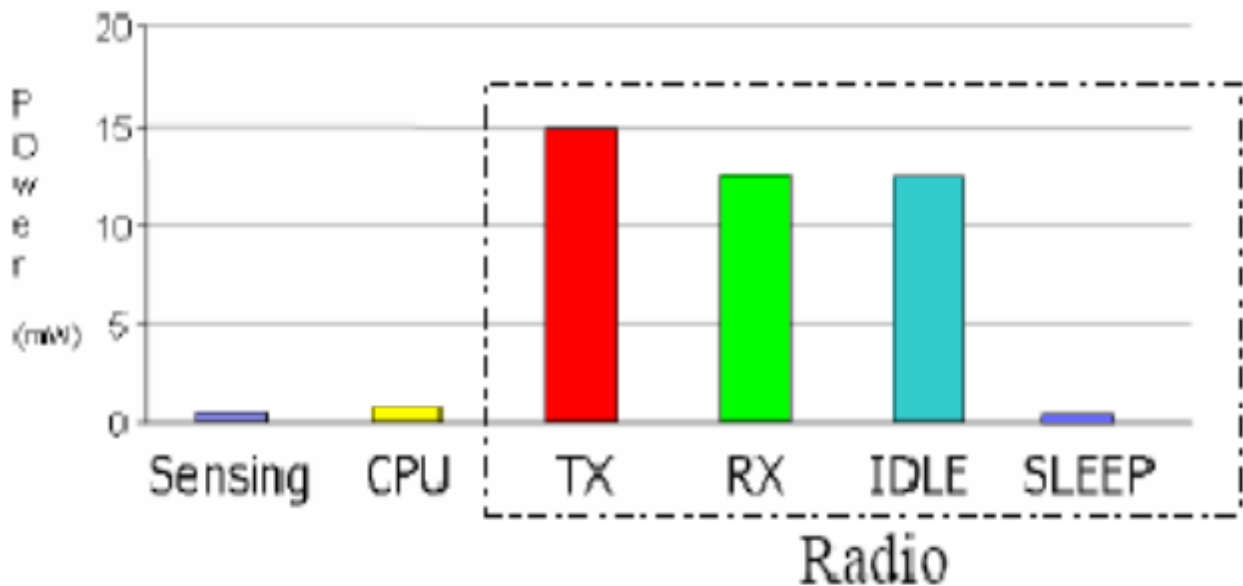


Figure 0.5: Consommation d'énergie en captage, traitement et transmission [35]

Où RX signifie énergie pour réception, TX pour transmission, IDLE en écoute de canal, SLEEP est l'état sommeil.

1.8 Architecture des RCSFs

L'architecture du réseau de capteurs est montrée dans la figure suivante (Figure 1.6). L'utilisateur accède à distance aux données capturées à travers un nœud appelé le nœud directeur de tâche "Task Manager Node". Le nœud directeur de tâche est relié à l'Internet ou au satellite à travers un nœud destinataire "puits" (sink/ station de base). Ce dernier agit en tant que passerelle pour le réseau de capteurs, c'est-à-dire qu'il relie des réseaux de capteurs à d'autres réseaux. Les nœuds capteurs sont habituellement dispersés dans une zone de capture appelée champ de captage.

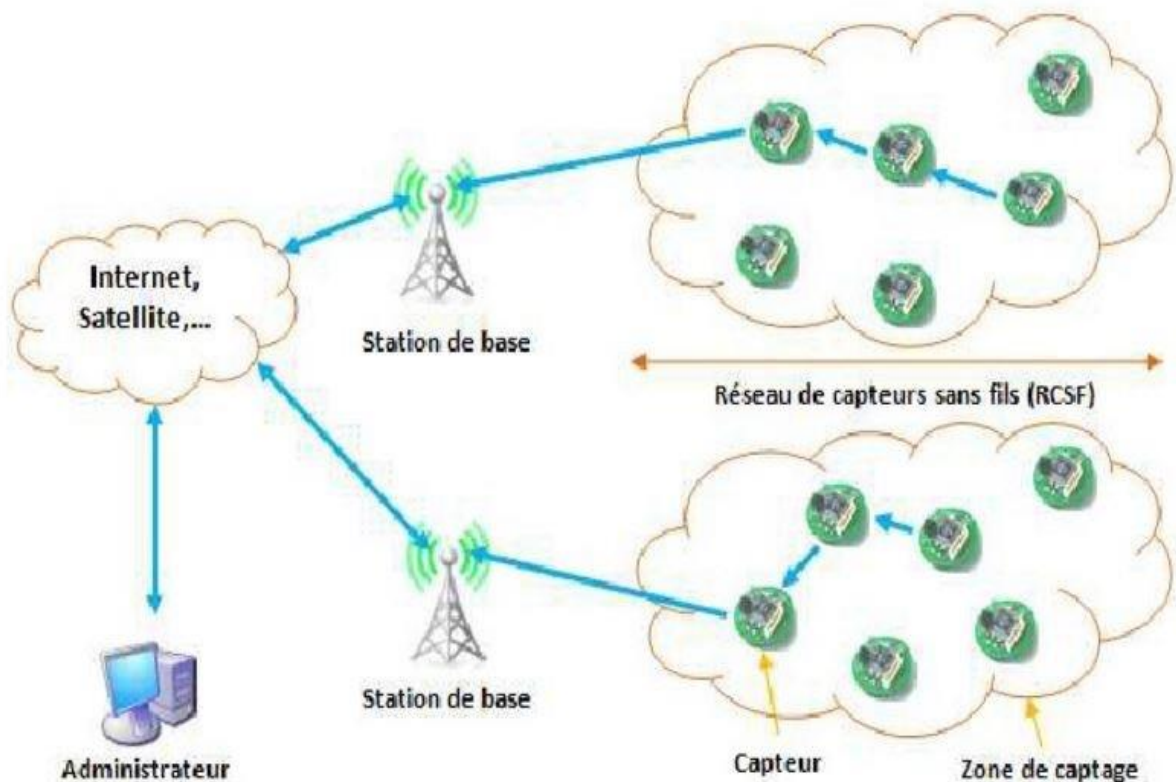


Figure 0.6: Schéma général d'un réseau de capteurs [29]

Il existe deux types d'architectures pour les réseaux de capteurs : les réseaux de capteurs plats et les réseaux de capteurs hiérarchiques.

1.8.1 Topologie plate :

Dans la topologie plate d'un RCSF, l'ensemble des nœuds capteurs ont le même rôle et sont reliés entre eux dans le but de créer des chemins qui atteignent la station de base. La Figure 1.7 montre un exemple d'une topologie plate d'un RCSF. Le nœud S désigne la station de base et les nœuds de couleur gris foncé désignent des capteurs. Les données sont routées d'un capteur à un autre afin d'arriver à la destination S. [48]

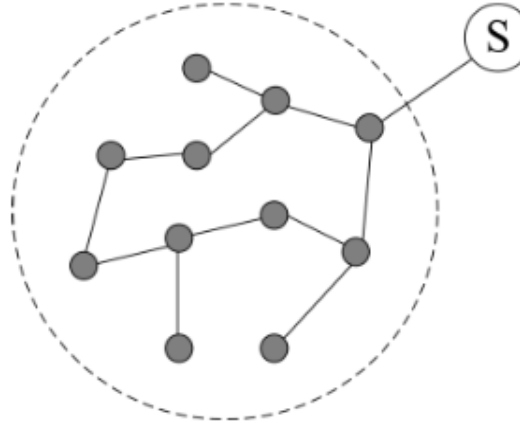


Figure 0.7: Exemple d'une topologie plate d'un RCSF

1.8.2 Topologie hiérarchique :

Dans la topologie hiérarchique d'un RCSF, tous les nœuds capteurs n'ont pas le même rôle. La topologie est organisée à partir d'un ensemble de clusters reliés entre eux grâce à des nœuds ayant des fonctionnalités de routeurs. Ces chefs de clusters ou cluster-heads sont chargés de router les informations collectées par leurs nœuds capteurs vers la station de base. La Figure 1.8 montre un exemple d'une topologie hiérarchique d'un RCSF. Le nœud S désigne la station de base, les nœuds R désignent des routeurs et les nœuds de couleur gris foncé désignent des capteurs. Les données récoltées par un nœud sont envoyées à son chef de cluster qui, à son tour, les envoient en exploitant d'une façon ascendante la hiérarchie des clusters, pour atteindre la station de base. [48]

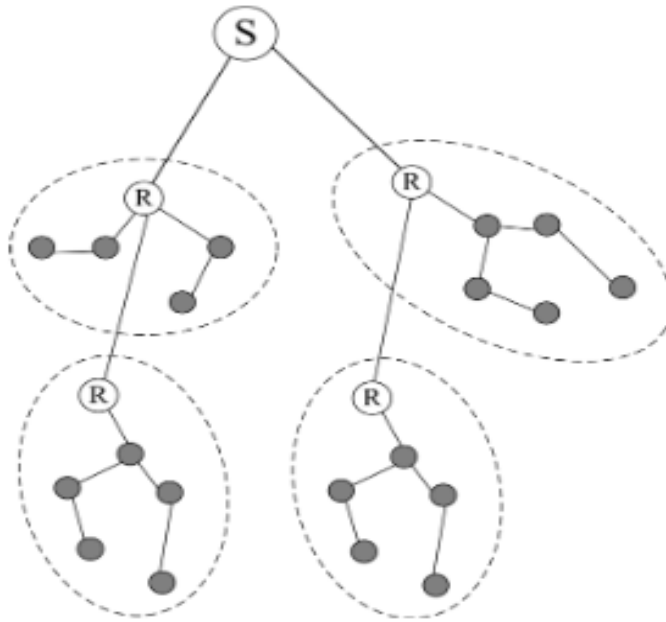


Figure 0.8: Exemple d'une topologie hiérarchique d'un RCSF

1.9 Architecture de communication dans les réseaux de capteurs (Pile protocolaire)

Le rôle de cette pile consiste à standardiser la communication entre les participants afin que différents constructeurs puissent mettre au point des produits (logiciels ou matériels) compatibles. Ce modèle comprend 5 couches qui ont les mêmes fonctions que celles du modèle OSI ainsi que 3 couches pour la gestion de la puissance, la gestion de la mobilité et la gestion des tâches.

Le but d'un système en couches est de séparer le problème en différentes parties (les couches) selon leur niveau d'abstraction. Chaque couche du modèle communique avec une couche adjacente (celle du dessus ou celle du dessous). Chaque couche utilise ainsi les services des couches inférieures et en fournit à celle de niveau supérieur.

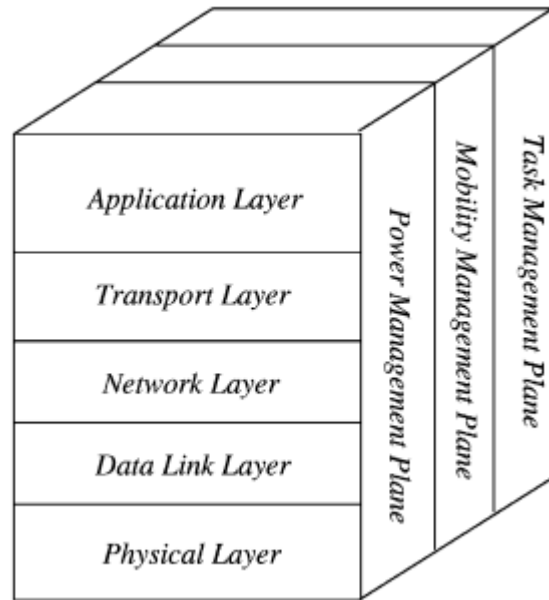


Figure 0.9: L'architecture de communication dans les réseaux de senseurs sans fil [33]

Ce modèle comprend : [29]

❖ **La couche physique**

Cette couche doit assurer des techniques d'émission, de réception et de modulation de données.

❖ **La couche liaison**

Elle assure la liaison point à point et point à multipoint dans un réseau de communication. Elle est composée de la couche de contrôle de liaison logique (LLC pour Logical Link Control) qui fournit une interface entre la couche liaison et la couche réseau en encapsulant les segments de messages de la couche réseau avec des informations d'entête additionnelles, et la couche de contrôle d'accès au médium (MAC pour Medium Access Control) qui contrôle la radio.

Comme l'environnement des réseaux de capteurs est bruyant et les nœuds peuvent être mobiles, la couche de liaison de données doit garantir une faible consommation d'énergie et minimiser les collisions entre les données diffusées par les nœuds voisins.

❖ **La couche réseau**

Cette couche permet de gérer l'adressage et le routage des données, c'est-à-dire leur acheminement via le réseau.

❖ **La couche transport**

Cette couche est chargée du transport des données, de leur découpage en paquets, du contrôle de flux, de la conservation de l'ordre des paquets et de la gestion des éventuelles erreurs de transmission.

❖ **La couche application**

Cette couche assure l'interface avec les applications. Il s'agit donc du niveau le plus proche des utilisateurs, géré directement par les logiciels.

❖ **Le niveau de gestion d'énergie**

Les fonctions intégrées à ce niveau consistent à gérer l'énergie consommée par les capteurs.

❖ **Le niveau de gestion de mobilité**

Ce niveau détecte et enregistre tous les mouvements des nœuds capteurs, de manière à leur permettre de garder continuellement une route vers l'utilisateur final, et maintenir une image récente sur les nœuds voisins. Cette image est nécessaire pour pouvoir équilibrer l'exécution des tâches et la consommation d'énergie.

❖ **Le niveau de gestion des tâches**

Lors d'une opération de capture dans une région donnée, les nœuds composant le réseau ne doivent pas obligatoirement travailler avec le même rythme. 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.

1.10 Couverture et Connectivité dans les RCSFs

Dans un RCSF, chaque nœud perçoit une vision limitée et purement locale de son environnement, relative uniquement à sa zone de perception. Cette dernière doit être mise en relation avec la zone de communication de ce capteur, afin de déterminer la densité optimale de capteurs à déployer. Et il doit économiser leur énergie tout en observant correctement leur environnement.

1.10.1 Connectivité

Un réseau de capteurs sans fil est dit connecté si et seulement s'il existe au moins une route entre chaque paire de nœuds mobiles. [34]

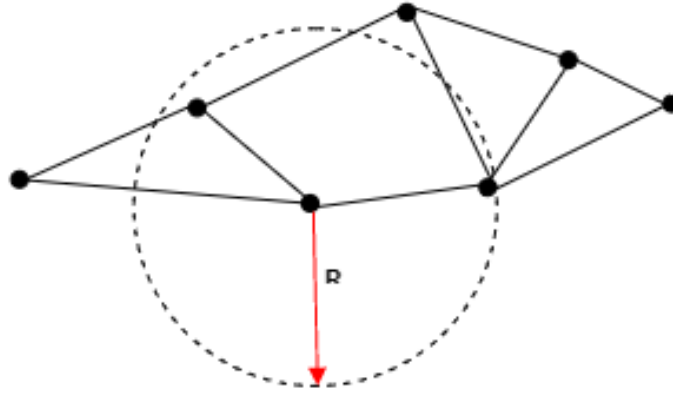


Figure 0.10: Connectivité d'un RCSF [34]

Pour une communication directe, deux capteurs (a) de rayon de transmission R_a et (b) de rayon R_b peuvent se communiquer directement si et seulement si : [34]

$$d(a, b) \leq R_b \text{ et } d(b, a) \leq R_a$$

1.10.2 Couverture

C'est la surface totale se trouvant en dessous de la marge ou de la portée de capture des données au moins d'un nœud.

1.10.2.1 Couverture d'un point

On dit qu'un capteur S_i couvre un point q si et seulement si la distance $d(q, s_i) \leq r_i$.

1.10.2.2 Couverture d'une zone

On dit qu'un capteur couvre une zone A si et seulement si pour chaque point q dans A , la distance $d(q, s_i) \leq r_i$.

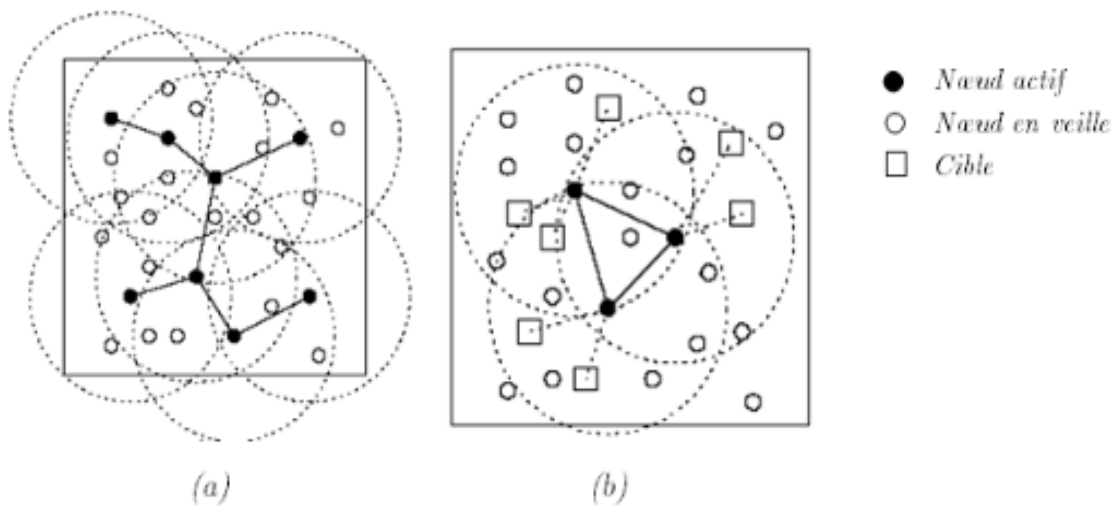


Figure 0.11: (a) Couverture d'une région. (b) couverture de points [34]

1.11 Les standards de communication pour les RCSF

Il y a plusieurs normes sans fil comme le WiFi (le standard IEEE 802.11 [36, 37]) et le WiMax (le standard IEEE 802.16 [38, 39]) qui s'adressent au transport des données à haut débit. Certains dispositifs comme les capteurs n'ont pas besoin d'une largeur de bande très élevée, mais plutôt d'un temps de latence faible ainsi qu'une consommation d'énergie très basse, pour une durée de vie sur batterie et un grand nombre de dispositifs, d'où la nécessité de concevoir d'autres normes sans fil capables de répondre à ces exigences. Parmi les standards les plus aptes à être exploités dans les RCSFs se retrouvent les standards Bluetooth et ZigBee présentés ci-après [41].

1.11.1 Bluetooth

Bluetooth [40] est un standard de communication sans fil qui fournit une configuration spéciale de pico-réseau maître/esclave avec le maximum de huit unités actives. Il supporte les connexions spontanées entre les périphériques sans leur exigeant des connaissances détaillées sur l'autre. Bluetooth permet des transferts de données entre les unités sur des distances, nominalement, jusqu'à 10 mètres. Les gros débits de données de 1 Mbps sont partagés entre tous les participants d'un pico-réseau. Bluetooth fonctionne à 2.4 GHz ISM (Industrie Scientifique Médical) dans le spectre sans licence (2.400 à 2.484 GHz) et utilise FHSS (Frequency Hopping Spread Spectrum) pour minimiser les interférences. Le succès de la technologie Bluetooth l'a amené à être utilisée dans le cadre des LAN (Local Area Network) sans fil. Le Bluetooth est orienté vers une faible consommation d'énergie et vise le marché des consommateurs avec une disponibilité dans le monde entier et à bas prix. Cependant, la faible consommation d'énergie ne répond pas encore à des exigences des capteurs. En outre, la technologie Bluetooth souffre d'un problème de passage à l'échelle.

1.11.2 ZigBee

Une des applications émergentes de RCSF est le suivi et le contrôle sans fil. ZigBee [24] [30] est une telle application qui utilise des capteurs en réseau à faible puissance et à faible débit de données. Il a été développé par la ZigBee Alliance, une association de l'industrie des sociétés de semi-conducteurs et des entreprises d'équipements de réseau tels qu'Embler, Honeywell, Mitsubishi Electric, Motorola, Samsung et Philips. Il est à noter que le terme ZigBee se réfère à la communication silencieuse entre abeilles où l'abeille danse dans un motif en zig-zag afin de dire aux autres l'emplacement, la distance et la direction de certains aliments nouvellement

trouvés. La communication dans les RCSFs ressemble un peu au principe de ZigBee en ce sens qu'ils doivent être simples et efficace. Les caractéristiques clés de ZigBee comprennent ce qui suit :

- Bandes de fréquences : 868 MHz, 915 MHz et 2.4 GHz.
- Taux de transfert jusqu'à 250 Kbps.
- La portée de transmission de signal de 10 à 100 m, en fonction des capteurs utilisés.
- Cryptage AES des données.
- Plusieurs applications ZigBee peuvent travailler les unes avec les autres.
- Faible consommation d'énergie.

1.12 Conclusion

Les réseaux de capteurs sans fil se propagent dans plusieurs domaines d'application. Ils sont devenus indispensables pour les mesures de température, humidité, vibration, etc.

Flexibilité, tolérance aux pannes, hautes capacités de captage, coût réduit, installation rapide sont les caractéristiques qui ont permis aux réseaux de capteurs d'avoir des nouveaux domaines de nos vies futures.

Chapitre 2

CRYPTOGRAPHIE

SYMETRIQUE

CHAPITRE 2

CRYPTOGRAPHIE SYMETRIQUE

2.1 Introduction

La cryptographie a été inventée en même temps que l'écriture. Dans l'ancienne Egypte par exemple, les scribes utilisaient des symboles hiéroglyphiques qu'eux seuls pouvaient déchiffrer. Du temps des romains, la cryptographie était basée sur une rotation de l'alphabet. Cette méthode est connue sous le nom de chiffrement de César. Ce chiffrement fut cassé par les mathématiciens arabes du temps d'Al Kindi (801-873). Pour chiffrer et déchiffrer avec ce type de cryptographie, il faut connaître la clé secrète. Ce procédé cryptographique est appelé cryptographie à clé secrète ou encore la cryptographie symétrique. Les crypto systèmes les plus connus dans la cryptographie symétrique sont Rijndael de AES (J. Daemen et V. Rijmen, 2002), IDEA (X. Lai et J. Massey, 1990) et 3DES (W. Tuchman, 2005). Ce type de cryptographie peut présenter évidemment un danger car il faudrait échanger les clés. En 1976, une cryptographie différente a été inventée par W. Diffie et M. Hellman [1]. Ce fut le début de la cryptographie moderne, connu sous le nom de cryptographie à clé publique ou encore cryptographie asymétrique. La cryptographie est une technologie essentielle dans la sécurité des systèmes informatique.

Dans ce chapitre nous commençons par citer une définition de la cryptographie et quelque conception principales Ensuite nous présentons l'algorithme de chiffrement par bloc AES ainsi l'algorithme de chiffrement par flux Trivium enfin une petite conclusion

2.2 Définition de la cryptographie

La cryptographie est la science qui utilise les mathématiques pour chiffrer et déchiffrer des données. La cryptographie vous permet de stocker des informations sensibles ou de les transmettre à travers des réseaux non sûrs (comme Internet) de telle sorte qu'elles ne puissent être lues par personne à l'exception du destinataire convenu. Alors que la cryptographie est la science de la sécurisation des données, la cryptanalyse est la science de l'analyse et du cassage des communications sécurisées. La cryptanalyse classique mêle une intéressante combinaison de

raisonnement analytique, d'application d'outils mathématiques, de découverte de redondances, de patience, de détermination, et de chance. Les cryptanalystes sont aussi appelés attaquants [2].

La sécurité des données cryptées repose entièrement sur deux éléments : l'invulnérabilité de l'algorithme de cryptographie et la confidentialité de la clé.

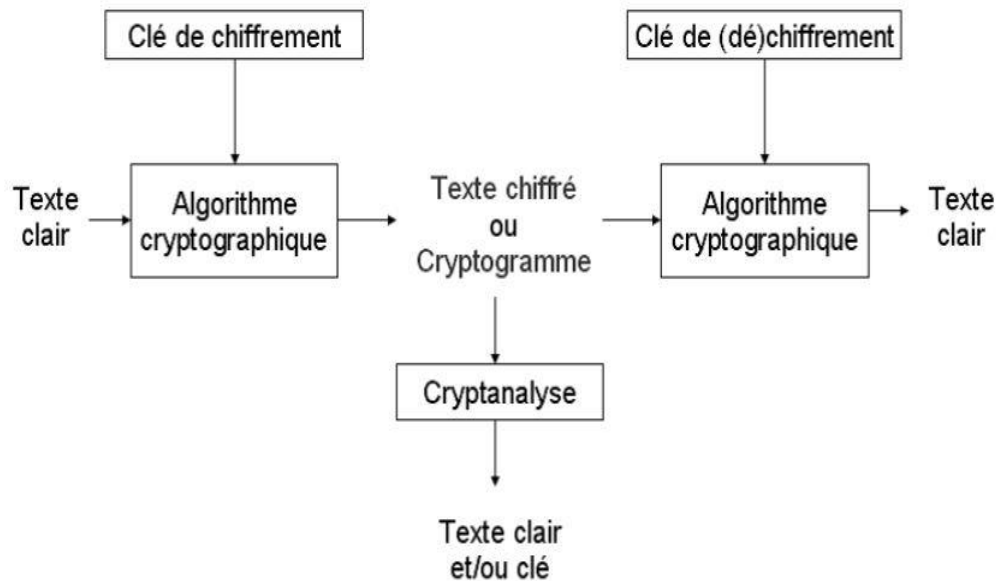


Figure 0.1: Principe du chiffrement.

2.3 Terminologie

- **Cryptosystème** : Il est défini comme l'ensemble des clés possibles (espace de clés), des textes clairs et chiffrés possibles associés à un algorithme donné.
- **Chiffrement** : Le chiffrement consiste à transformer une donnée (texte, message, ...) afin de la rendre incompréhensible par une personne autre que celui qui a message et celui qui en est le destinataire. La fonction permettant de retrouver le texte clair à partir du texte chiffré porte le nom de déchiffrement.
- **Texte chiffré** : Appelé également cryptogramme, le texte chiffré est le résultat de l'application d'un chiffrement à un texte clair.
- **Clé** : Il s'agit du paramètre impliqué et autorisant des opérations de chiffrement et/ou déchiffrement. Dans le cas d'un algorithme symétrique, la clé est identique lors des deux opérations. Dans le cas d'algorithmes asymétriques, elle diffère pour les deux opérations.
- **Entité (agent)** : Quelqu'un ou quelque chose qui envoie, reçoit ou modifie de l'information. Elle peut être une personne physique ou morale, un ordinateur, etc.

- **Expéditeur** : Entité qui envoie légitimement de l'information dans une transmission à deux parties.
- **Récepteur** : Entité destinée à recevoir l'information dans une transmission à deux parties.
- **Adversaire** : Entité qui n'est pas l'expéditeur ni le récepteur et qui tente de déjouer la sécurité d'une transmission à deux parties.
- **En cryptographie**, la propriété de base est que $M = D(E(M))$ où
 - M représente le texte clair.
 - $E(x)$ est la fonction de chiffrement.
 - $D(x)$ est la fonction de déchiffrement.

2.4 Cryptographie Symétrique

Cryptographie symétrique également appelée cryptage de clé secrète, une seule clé suffit pour le cryptage et le décryptage. On suppose que M est le message à chiffrer et k est la clé secrète partagée par L'émetteur (Alice) et le destinataire (Bob) du message. Alice utilise la fonction de cryptage $E()$ et la clé k pour générer un message chiffré $E(M, k) = M'$. Après la réception du message chiffré, Bob utilise une fonction de décryptage $D()$ et la même clé k pour déchiffrer le message $D(M', k) = M$.

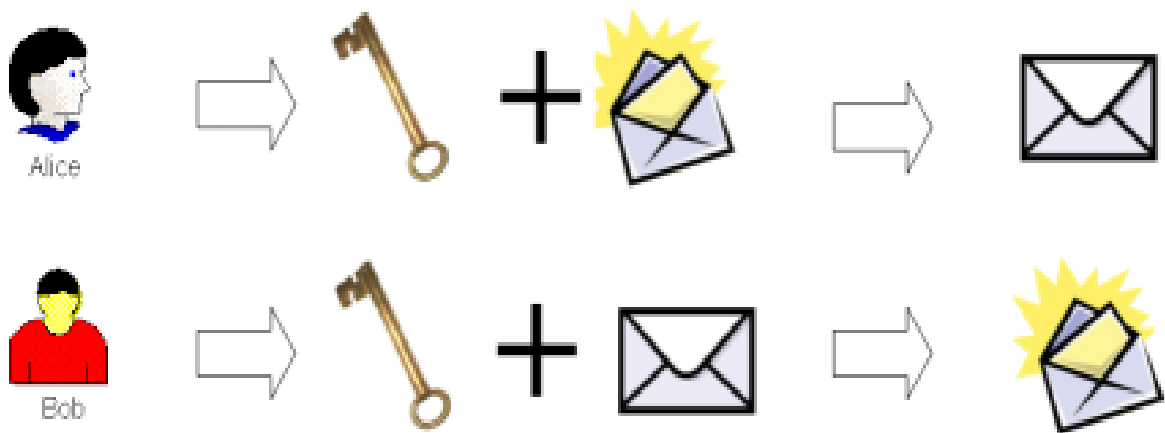


Figure 0.2: Chiffrement Symétrique.

Les algorithmes de chiffrement symétrique sont très rapides. Ils sont donc très bien appropriés au chiffrement de grande quantité de données. Mais la gestion des clés possède des problèmes

Pour que 2 nœuds puissent communiquer entre eux, il faut qu'ils utilisent exactement la même clé, et la distribution de clés est devenue un défi inévitable. Il existe quatre types de distribution:

➤ **Clé globale :**

C'est la clé partagée par l'ensemble des nœuds du réseau. C'est une clé qui est utilisée par la station de base pour chiffrer des messages diffusés à tout le groupe. C'est la solution la plus facile à mettre en place, et elle offre un minimum de protection contre l'espionnage de données. Si l'attaque réussit à compromettre un seul nœud, tout le mécanisme de sécurité va échouer.

➤ **Clé partagée par paire de nœuds :**

Chaque nœud partage une clé dédiée avec son voisin. Si un nœud possède n voisins, il aura $(n-1)$ clés à stocker pour pouvoir communiquer avec ses voisins. Chacune de ces clés est connue seulement par ce nœud et un des $(n-1)$ autres nœuds. L'attaque d'un nœud et la récupération de ses informations secrètes n'affectent pas la sécurité des autres nœuds.

Cette technique exige une capacité mémoire importante pour stocker les $(n-1)$ clés (n peut être grand). Elle est très couteuse en temps de calcul car chaque nœud qui reçoit un message doit le décrypter et l'encrypter avant de le renvoyer.

➤ **Clé individuelle :**

Chaque nœud détient une clé qui est partagée par lui-même et la station de base, mais cette distribution sécurise seulement la communication entre les nœuds et la station de base. Les nœuds ne peuvent pas communiquer entre eux, car ils ne possèdent pas de clé partagée.

➤ **Clé partagée par groupes de nœuds :**

Une clé est partagée entre un groupe de nœuds, par exemple, dans un cluster, le cluster-head est chargé de la génération et de la distribution de clé, et la communication intérieure du cluster est sécurisée avec ces clés partagées.

La cryptographie symétrique n'offre pas la signature numérique, sans laquelle les nœuds ne peuvent pas authentifier les messages reçus, et sont vulnérables aux attaques de vol d'identité.

Les chiffrements symétriques se subdivisent en deux familles :

2.4.1 Chiffrement par bloc

Un algorithme de chiffrement par bloc transforme un message (claire) de taille fixée en un message (chiffré) de même taille sous l'action d'une clé secrète. Le déchiffrement est effectué en utilisant la transformation inverse et la même clé. La taille des blocs de message est le plus souvent de 64 ou 128 bits. Les algorithmes de chiffrement par bloc les plus connus et utilisés sont le DES, SAFER et l'AES.

L'idée générale du chiffrement par blocs est la suivante :

- 1 Remplacer les caractères par un code binaire.
- 2 Découper cette chaîne en blocs de longueur donnée.
- 3 Chiffrer un bloc en l'"additionnant" bit par bit à une clef.
- 4 Déplacer certains bits du bloc.
- 5 Recommencer éventuellement un certain nombre de fois l'opération 3.
- 6 Passer au bloc suivant et retourner au point 3 jusqu'à ce que tout le message soit chiffré.

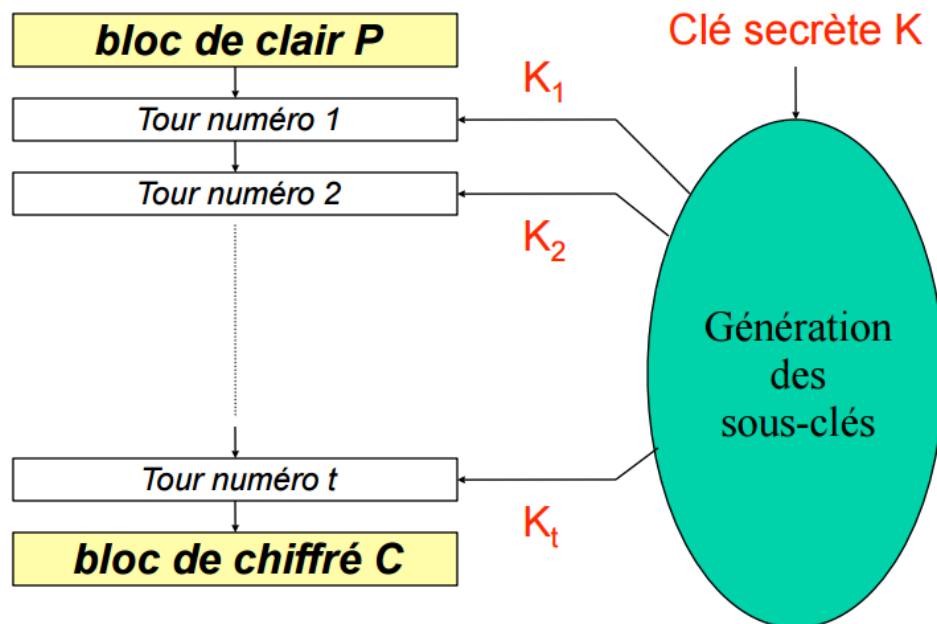


Figure 0.3: principe de chiffrement par bloc

Les catégories du chiffrement dans le mode par blocs sont :

- **Chiffrement par substitution** : Les substitutions consistent à remplacer des symboles ou des groupes de symboles par d'autres symboles ou groupes de symboles dans le but de créer de la confusion.
- **Chiffrement par transposition** : Les transpositions consistent à mélanger les symboles ou les groupes de symboles d'un message clair suivant des règles prédéfinies pour créer de la diffusion. Ces Règles sont déterminées par la clé de chiffrement. Une suite de transpositions forme une permutation.
- **Chiffrement par produit** : C'est la combinaison des deux. Le chiffrement par substitution ou par transposition ne fournit pas un haut niveau de sécurité, mais en combinant ces deux transformations, on peut obtenir un chiffrement plus robuste.

2.4.2 Chiffrement par flux

L'opération de chiffrement s'opère sur chaque élément du texte clair (caractère, bits). On chiffre un bit/caractère à la fois. La structure d'un chiffrement par flot repose sur un générateur de clés qui produit une séquence de clés k_1, k_2, \dots, k_i .

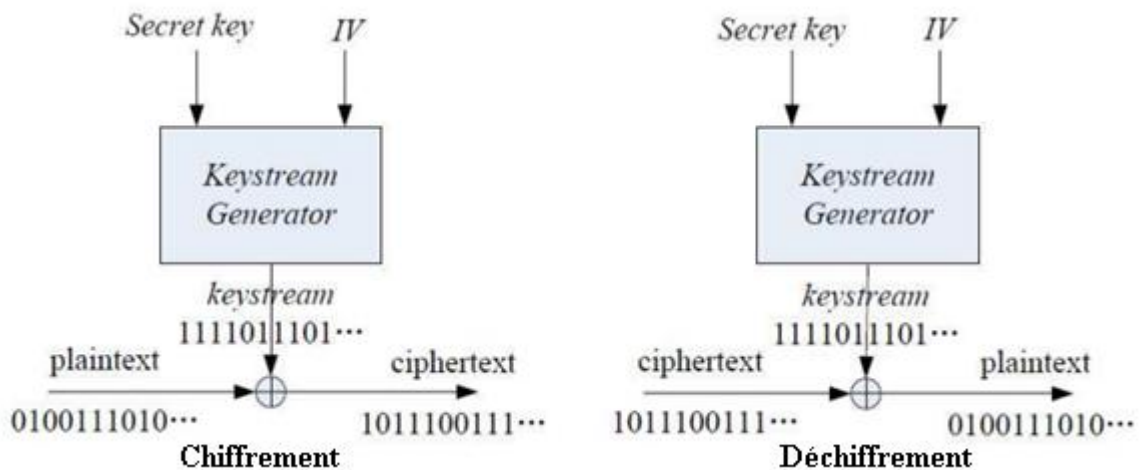


Figure 0.4: Schéma de chiffement et déchiffement par flux [5]

En termes de propagation d'erreur, une erreur dans C_i n'affecte qu'un bit de M_i . La perte ou l'ajout d'un bit de C_i affecte tous les bits suivants de M après déchiffement.

2.5 Advanced Encryption Standard (A.E.S)

Le véritable nom de l'AES est le Rijndael, nom résultant de la contraction des noms de ses inventeurs : Rijmen et Deamen. L'algorithme de chiffement par blocs AES a été proposé comme

algorithme de chiffrement dans le standard IEEE 802.15.4 pour sécuriser au niveau de la couche MAC les données transitant sur des réseaux de capteurs sans fil [6].

AES possède les propriétés suivantes :

- Plusieurs longueurs de clef et de bloc sont possibles : 128, 192, ou 256 bits.
- Le nombre de cycles (ou "rondes") varie en fonction de la longueur des blocs et des clés (de 10 à 14).
- La structure générale ne comprend qu'une série de transformations / permutations / sélections.
- Il est facilement adaptable à des processeurs de 8 ou de 64 bits.
- Le parallélisme peut être implémenté.

2.5.1 Chiffrement et Déchiffrement :

Pour le chiffrement et le déchiffrement, l'algorithme AES utilise une fonction ronde qui est composée de quatre différentes transformations, SubByte, ShiftRow, MixColumn et AddRoundKey. L'ordonnancement des étapes est illustré à la Figure 2.5.

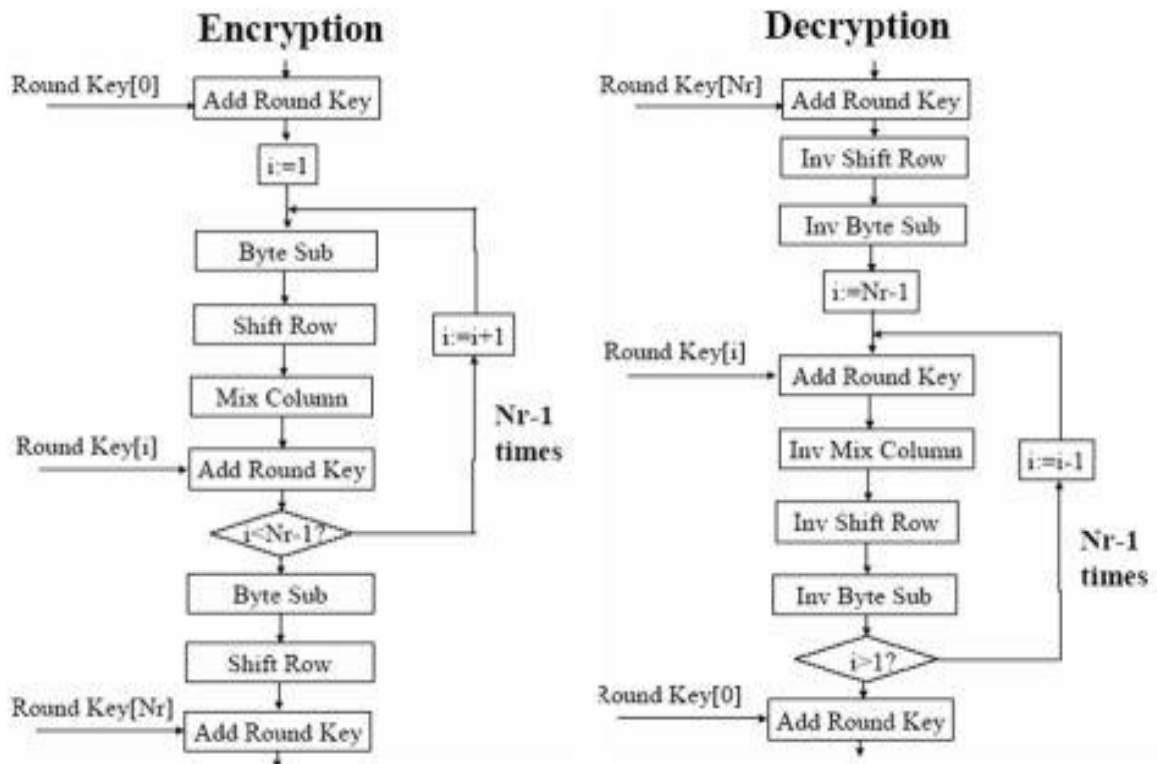


Figure 0.5: Schéma des différentes étapes de (AES)

Avec :

Nb : la longueur des blocs dans le chiffre de Rijindael est $32 \cdot N_b$ (nombre de mots). $N_b = 4$.

N_k : Une clé consiste en N_k mots de 32 bits, avec $N_k = 4, 6$ ou 8 .

N_r : nombre de tournées (rondes).

$$N_r = \begin{cases} 10 & \text{si } N_k=4 \text{ (128 bits)} \\ 12 & \text{si } N_k=6 \text{ (192 bits)} \\ 14 & \text{si } N_k=8 \text{ (256 bits)} \end{cases}$$

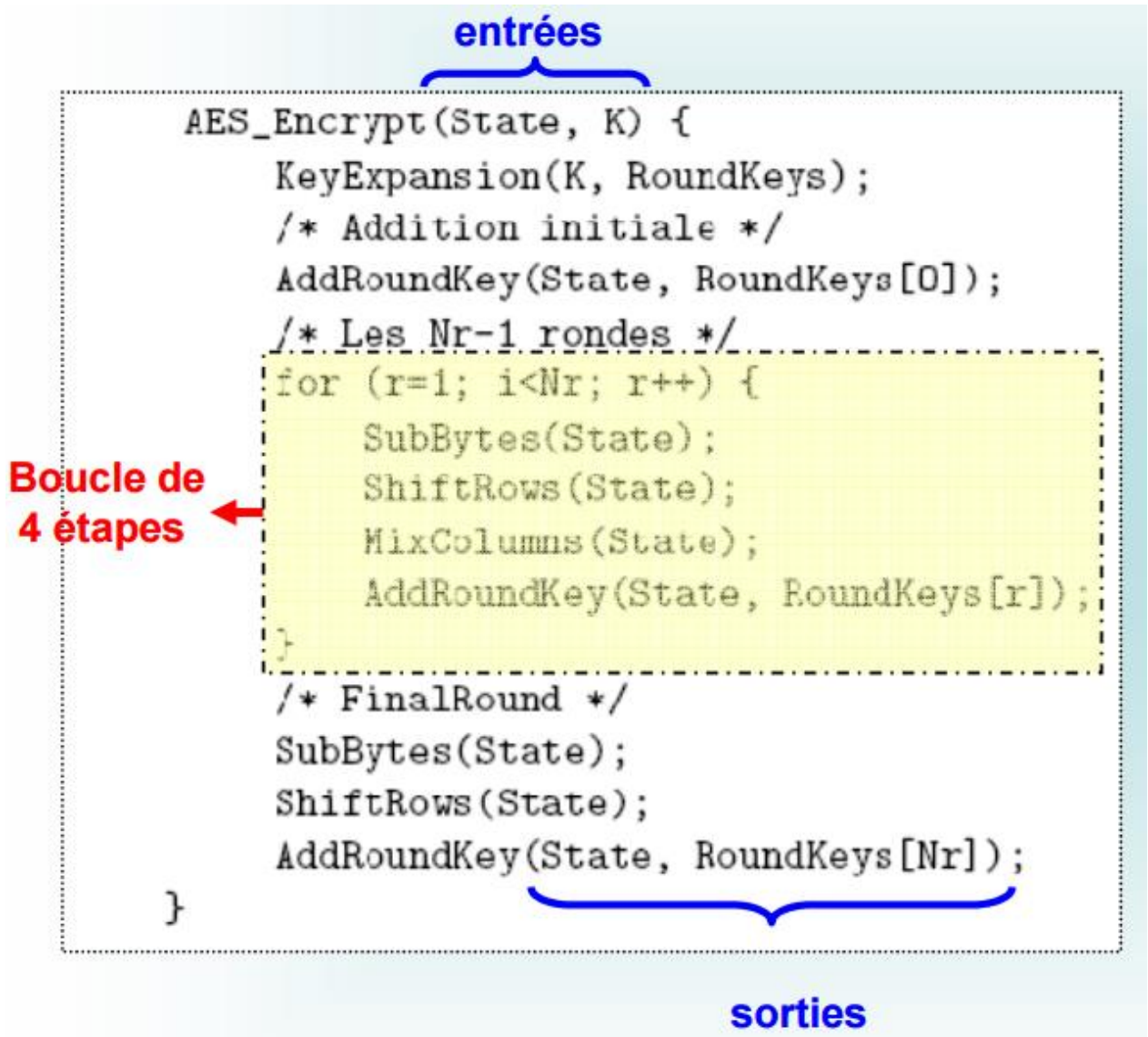


Figure 0.6: Pseudo-code de chiffrement

2.5.1.1 SubByte

Est une fonction non-linéaire opérant indépendamment sur chaque bloc à partir d'une table dite S-Box inversible.

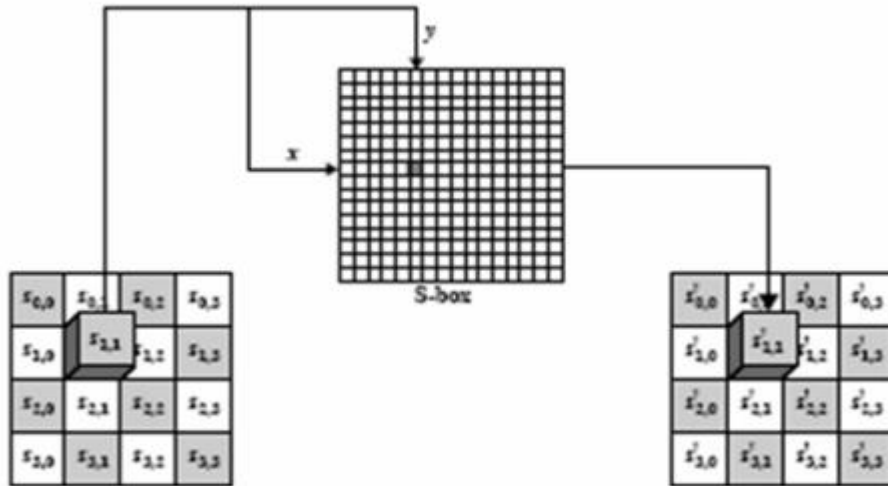


Figure 0.7: Fonction SubByte

		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

Figure 0.8: S-Box inversible

2.5.1.2 ShiftRow

Est une fonction opérant des décalages (typiquement elle prend l'entrée en 4 morceaux de 4 octets et opère des décalages vers la gauche de 0, 1, 2 et 3 octets pour les morceaux 1, 2, 3 et 4 respectivement).

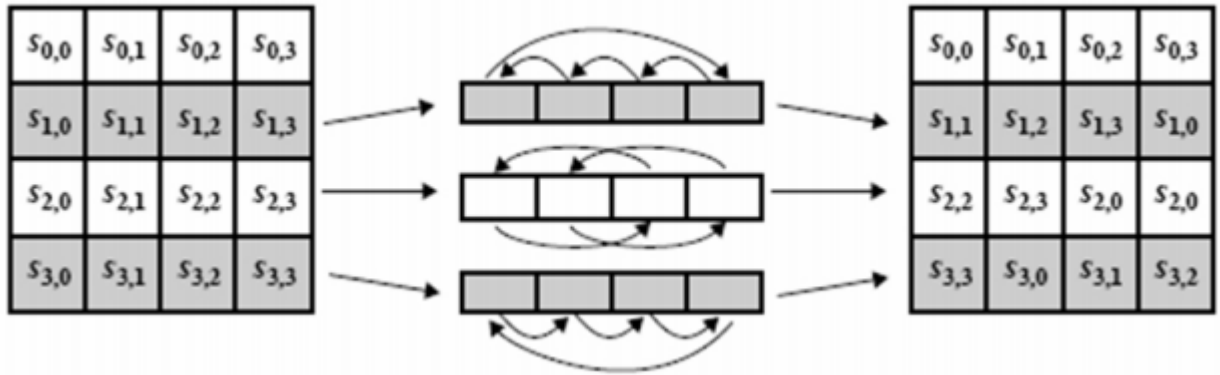


Figure 0.9: Schéma de l'étape ShiftRow

Selon la taille des blocs de message (Nb), les décalages ne seront pas toujours identiques.

- La ligne 0 n'est jamais décalée.
- La ligne 1 est décalée de C1.
- La ligne 2 est décalée de C2.
- La ligne 3 est décalée de C3.

	C1	C2	C3
Nb=4	1	2	3
Nb=6	1	2	3
Nb=8	1	3	4

Table 0.1: Décalage selon la taille des bloc de messages

2.5.1.3 MixColumn

Cette fonction traite chaque colonne par produit matriciel comme montre la figure 2.10

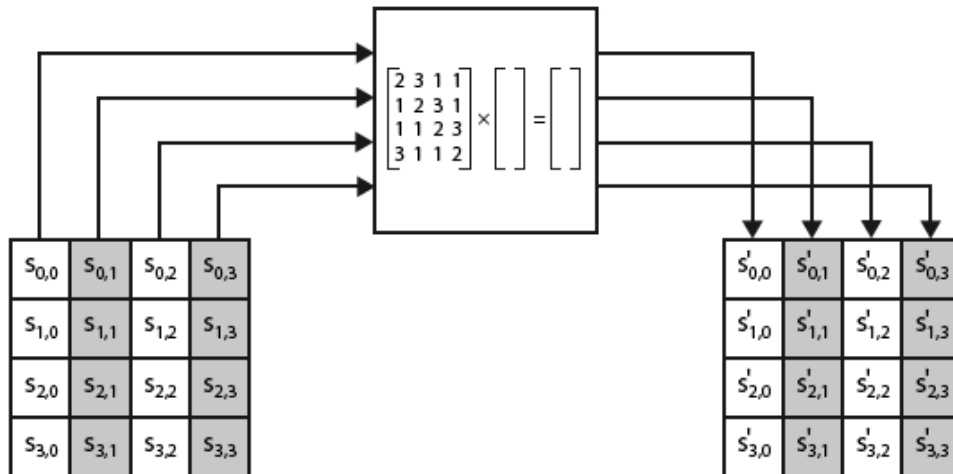


Figure 0.10 : MixColumn

2.5.1.4 AddRoundKey

C'est un simple \oplus des clés. Il s'agit d'additionner des sous-clés aux sous-blocs correspondants

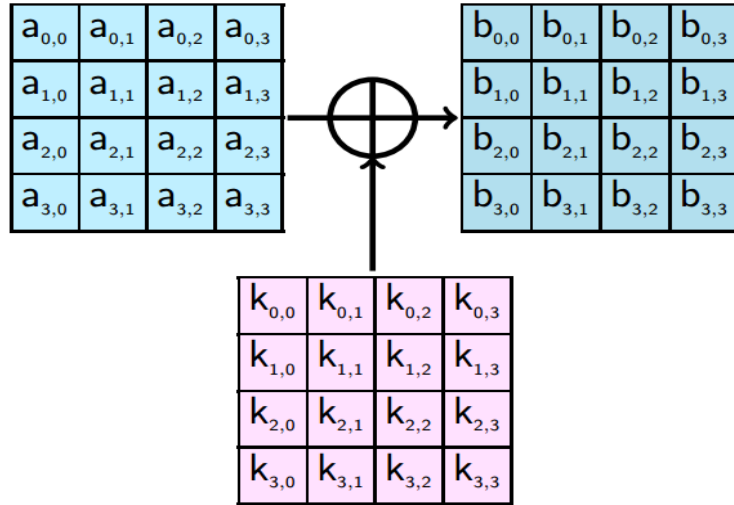


Figure 0.11: AddRound Key

2.5.1.5 Expansion de la clé

Le calcul de l'expansion de la clé se fait de deux manières distinctes selon le sous-bloc de la clé concerné.

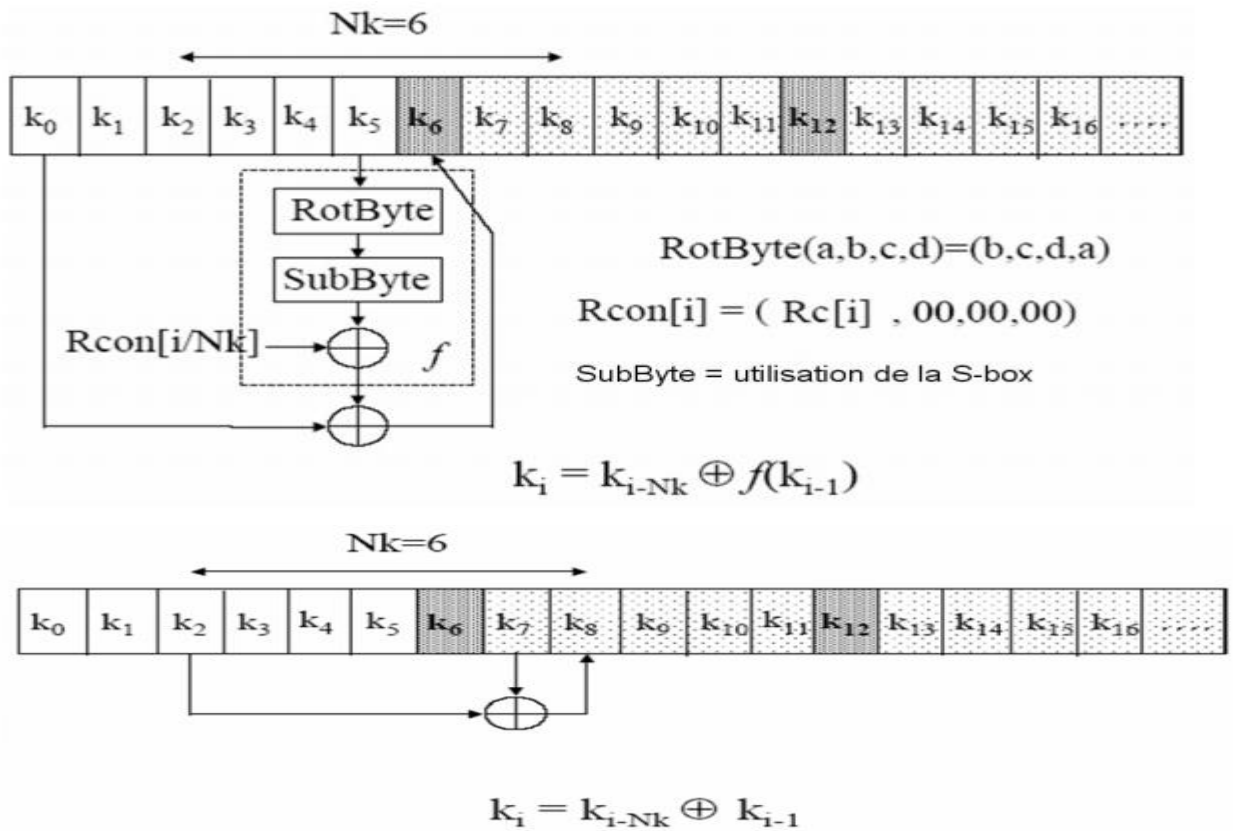


Figure 0.12: Expansion de la clé

2.5.2 Avantages et limites :

Les principaux avantages sont :

- Des performances très élevées.
- il ne comprend pas d'opérations arithmétiques : ce sont uniquement des décalages et des XOR.
- il ne possède pas de clés faibles.

Il possède pourtant quelques inconvénients et limites :

- le code et les tables sont différents pour le chiffrement et déchiffrement.
- dans une réalisation matérielle, il y a peu de réutilisation des circuits de chiffrement pour effectuer le déchiffrement.

2.5.3 Attaque :

En 2009, un article [4] met en évidence des problèmes de sécurité sur des versions allégées de l'AES-256. Bien que non applicables à l'algorithme standard (sur 128 bits).

2.6 Trivium

Trivium est relativement nouveau chiffrement de flux qui utilise une clé privée de 80 bits, et une valeur initiale (IV) de 80 bit. Il est basé sur une combinaison de trois registres à décalage. [11]

2.6.1 Notation

(S_m, \dots, S_n) l'état interne de $(m-n+1)$ bits.

S_i l'état.

Z_t bit générer de KeyStream dans le temps t .

$+$ XOR

\cdot AND

2.6.2 Architecteur

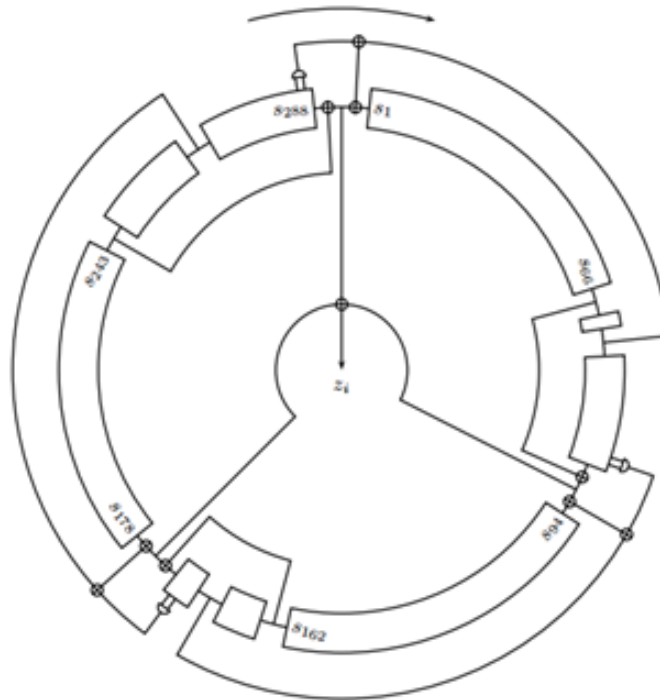


Figure 0.13 : illustre cœur de Trivium.[7]

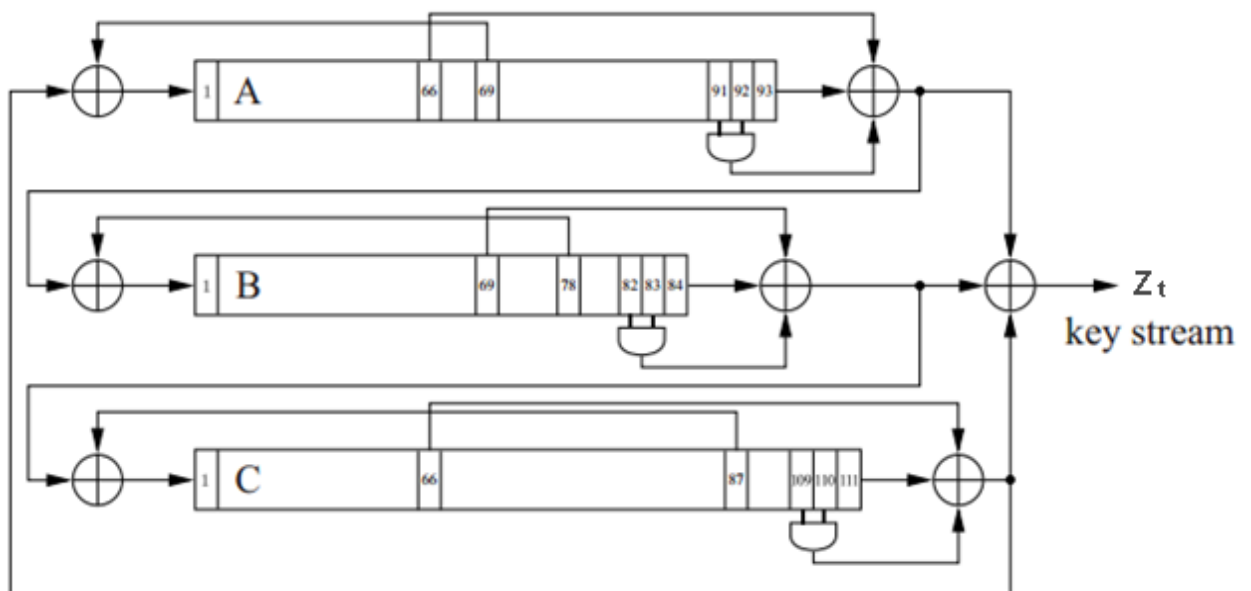


Figure 0.14 : Structure interne de Trivium [7]

La sortie de chaque registre est reliée à l'entrée d'un autre registre. Le chiffrement constitué d'un registre circulaire avec une longueur totale de $93 + 84 + 111 = 288$.

L'entrée de chaque registre est calculée comme XOR de deux bits :

- Le bit de sortie d'un autre registre selon la Figure 2.14. Par exemple, la sortie du registre A est une partie de l'entrée du registre B.
- Un bit de registre à un endroit précis est renvoyé à l'entrée. Les positions sont donnée dans le tableau 2.2 Par exemple, le bit 68 du registre A est renvoyée à son entrée.

La sortie de chaque registre est calculée comme XOR de trois bits :

- Le bit le plus à droite du registre.
- Un bit de registre à un emplacement spécifique est alimenté vers l'avant à la sortie. Les positions sont indiquées dans le Tableau 2.2 Par exemple, le bit 66 du registre A est amené à sa sortie.
- La sortie d'une fonction AND logique dont l'entrée est de deux bits de registre spécifiques. les positions des portes AND sont indiquées dans le Tableau 2.2

	longueur de registre	Bit vers l'entrée	Bit vers l'avant	Les entrées de (AND)
A	93	69	66	91,92
B	84	78	69	82,83
C	111	87	66	109,110

Table 0.2 : spécification du trivium

2.6.3 Génération de clé

L'état interne est initialisé utilisant la clé et IV. Donc l'Etat mise à jour répétitions et utilisé pour générer une clé de flux du bit.

$$(S_1, S_2, \dots, S_{93}) \leftarrow (K_1, \dots, K_{80}, 0, \dots, 0)$$

$$(S_{94}, S_{95}, \dots, S_{177}) \leftarrow (IV_1, \dots, IV_{80}, 0, \dots, 0)$$

$$(S_{178}, S_{279}, \dots, S_{288}) \leftarrow (0, \dots, 0, 1, 1, 1)$$

la performance de l'algorithme se produit pendant $4 * 288 = 1152$ fois, ce qui assure la dépendance de chaque bit de l'état initial de chaque bit de la clé et chaque vecteur binaire d'initialisation.

```
for  $i = 1$  to  $4 \cdot 288$  do  
     $t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$   
     $t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$   
     $t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$   
     $(s_1, s_2, \dots, s_{93}) \leftarrow (t_3, s_1, \dots, s_{92})$   
     $(s_{94}, s_{95}, \dots, s_{177}) \leftarrow (t_1, s_{94}, \dots, s_{176})$   
     $(s_{178}, s_{279}, \dots, s_{288}) \leftarrow (t_2, s_{178}, \dots, s_{287})$   
end for
```

2.6.4 Chiffrement

Pour chiffrer les messages doivent mener des opérations XOR sur le message et le flux de clé. En commençant par le bit de sortie du cycle de 1153.

$$y_i = x_i + z_i \bmod 2$$

2.6.5 Déchiffrement

Le décryptage est effectué d'une manière similaire, l'opération XOR est effectuée sur la séquence clé et le texte chiffré.

$$x_i = y_i + z_i \bmod 2$$

2.7 Conclusion

Au niveau de ce chapitre, nous avons présenté en détail deux algorithmes de cryptographie symétrique pour les deux catégories (par bloc, par flux). Il existe d'autre type de la cryptographie, c'est l'asymétrique qui nous allons le voir dans le chapitre suivant.

Chapitre 3

CRYPTOGRAPHIE

ASYMETRIQUE

CHAPITRE 3

CRYPTOGRAPHIE ASYMETRIQUE

3.1 Introduction

En 1976, une cryptographie a été inventée par W. Diffie et M. Hellman [1]. Ce fut le début de la cryptographie moderne, connu sous le nom de cryptographie à clé publique ou encore cryptographie asymétrique.

Dans ce chapitre nous présentons l'ensemble d'algorithmes Cryptographie Asymétrique ainsi fonction de hachage Ensuite la signature numérique enfin une petite conclusion.

3.2 Cryptographie asymétrique

Les problèmes de distribution des clés sont résolus par la cryptographie asymétrique (cryptographie à clé publique).

La cryptographie asymétrique utilisant une paire de clés pour le cryptage : une clé publique qui crypte des données et une clé privée ou secrète correspondante pour le décryptage. Vous pouvez ainsi publier votre clé publique tout en conservant votre clé privée secrète. Tout utilisateur possédant une copie de votre clé publique peut ensuite crypter des informations que vous êtes le seul à pouvoir lire (Seule la personne disposant de la clé privée correspondante peut les décrypter). Même les personnes que vous ne connaissez pas personnellement peuvent utiliser votre clé publique.

D'un point de vue informatique, il est impossible de deviner la clé privée à partir de la clé publique.

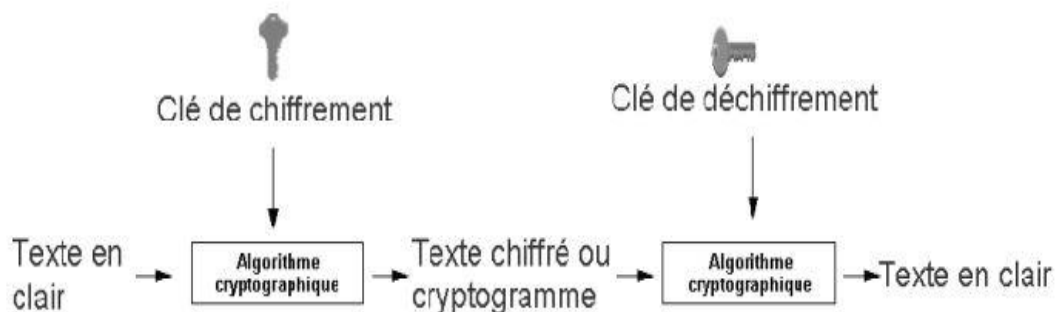


Figure 0.1 : Chiffrement Asymétrique.

3.3 RSA

Le principe de la cryptographie à clé publique proposé en 1976 et en 1977 présenté le premier cryptosystème effectif RSA (du nom de ses auteurs Rivest, Shamir et Adleman) [12]. RSA est basé sur le problème de la factorisation des grands nombres entiers, RSA est encore aujourd'hui la primitive la plus utilisée en protocoles de communication.

3.3.1 Génération des clés

Le module public N (ou module RSA), est défini comme le produit de deux grands nombres premiers tirés aléatoirement p et q .

$$N = p * q.$$

L'indicateur d'Euler :

$$\Phi(N) = (p - 1) (q - 1).$$

La clé publique est (n, e) :

$$e \text{ est premier, } 1 \leq e \leq \Phi(N), (e, \Phi(n))=1.$$

La clé privée est (n, d) :

$$1 \leq d \leq \Phi(N), e.d \equiv 1 \pmod{\Phi(N)}.$$

Exemple

- $P = 5$ et $q = 7$
- $N = p * q = 35$
- $\Phi(35) = (5-1)*(7-1) = 24$

On cherche e et d tels que :

- e est premier et $e.d \equiv 1 \pmod{24}$
 - Soit $e=5$ (par exemple, et on a bien $(e, \Phi(n))=1$)
 - On détermine que $d=33$ (inverse modulaire de e sur $Z\Phi(n)$)
- \Rightarrow Clé publique $(35,5)$ et la clé privée $(35,33)$

3.3.2 Chiffrement RSA

Le message ' m ' (type entier) qui doit être chiffrée en utilisant la clé publique ' e ' à donner le message crypté comme ' c ' où ' c ' est calculé comme

$$c = m^e \pmod{N}.$$

3.3.3 Déchiffrement RSA

Le message ' m ' décryptée se trouve à l'aide de la clé privé ' d ', est calculée comme :

$$m = c^d \pmod{N}$$

3.3.4 Exemple numérique pour simplifier décodage RSA

- $p = 47, q = 71, n = p \cdot q = 3337, (p - 1)(q - 1) = 3220, e = 79, d = 1019$
- On vérifie que $e \cdot d = 80501 = 25 \times 3220 + 1 = 1 \pmod{3220}$
- Prenons un message préalablement transformé en un nombre :
 $m = 6882326879666683$
- On découpe m : $m_1 = 688 \ m_2 = 232 \ m_3 = 687 \ m_4 = 966 \ m_5 = 668 \ m_6 = 3$
- Le premier bloc est chiffré par le nombre $c_1 = 68879 \pmod{3337} = 1570$; de même :
 $c_2 = 2756 \ c_3 = 2091 \ c_4 = 2276 \ c_5 = 2423 \ c_6 = 158$.
- Le déchiffrement du message se fait en calculant :
 $c_1 = 1570^{1019} \pmod{3337} = 688$, etc...

3.3.5 Cryptanalyse RSA

Il existe plusieurs approches à attaquer RSA parmi ceux-ci [13]:

1. recherche par force brute de la clé (impossible étant donné la taille des données)
2. attaques mathématiques
3. attaques de synchronisation (sur le fonctionnement du déchiffrement).

Pour garantir une bonne sécurité, il faut respecter certaines règles telles que :

- Ne jamais utiliser de valeur N trop petite.
- N'utiliser que des clés fortes ($p-1$ et $q-1$ ont un grand facteur premier).
- Ne pas chiffrer de blocs trop courts.
- Ne pas utiliser de N communs à plusieurs clés,

3.4 Elliptic Curve Cryptography (ECC)

Pour utiliser les courbes elliptiques en cryptographie, il faut trouver un problème difficile (tel que la factorisation d'un produit en ses facteurs premiers dans le cas du RSA). Considérons l'équation $Q = kP$ où $Q, P \in E_p(a, b)$ et $k < p$.

Il est facile de calculer Q connaissant k et P , mais il est difficile de déterminer k si on connaît Q et P . Il s'agit du problème du logarithme discret pour les courbes elliptiques : $\log P(Q)$. [8] [9]

3.4.1 ECC pour l'échange de clés

Dans le groupe associé à une courbe elliptique, le problème du logarithme discret est considéré comme difficile. On peut définir l'échange de clés Diffie-Hellman basé sur les courbes elliptiques.

Alice et Bob se mettent d'accord (publiquement) sur une courbe elliptique $E(a,b,p)$, c'est-à-dire qu'ils choisissent une courbe elliptique $y^2 = x^3 + ax + b \pmod p$. Ils se mettent aussi d'accord (toujours publiquement) sur un point P situé sur la courbe.

On définit $nP = P + P + \dots P$ (n fois) où l'opération $+$ correspond à la somme de 2 points définie par le symétrique du troisième point d'intersection de la droite définie par les 2 points originaux avec la courbe elliptique. Dans le cas où les deux points à ajouter sont identiques, on considère que la droite qui les joint est la tangente à la courbe elliptique passant par l'un d'entre eux. Un tel point est rationnel.

Secrètement, Alice choisit un entier d_A , et Bob un entier d_B . Alice envoie à Bob le point d_AP , et Bob envoie à Alice d_BP . Chacun de leur côté, ils sont capables de calculer $d_A(d_BP) = d_B(d_AP) = d_{ADB}P$ qui est un point de la courbe, et constitue leur clef secrète commune.

Si Eve a espionné leurs échanges, elle connaît $E(a,b,p)$, P , d_AP , d_BP . Pour pouvoir calculer $d_{ADB}P$, il faut pouvoir calculer d_A connaissant P et d_AP . C'est ce que l'on appelle résoudre le logarithme discret sur une courbe elliptique. Or, actuellement, si les nombres sont suffisamment grands, on ne connaît pas de méthode efficace pour résoudre ce problème en un temps raisonnable.

Exemple (Schaefer, Santa Clara University) :

- Soient $p = 211$, $E_p(0, -4) (\Rightarrow y^2 = x^3 - 4)$ et $G = (2, 2)$. On calcule que $240G = O$ et donc $n = 240$.
- A choisit $n_A = 121$, ce qui lui donne $P_A = 121(2, 2) = (115, 48)$.
- B choisit $n_B = 203$, ce qui lui donne $P_B = 203(2, 2) = (130, 203)$.
- La clé secrète K générée est $121(130, 203) = 203(115, 48) = (161, 69)$.

3.4.2 ECC pour chiffrement et déchiffrement

Pour chiffrer le message, A déterminé aléatoirement un nombre entier positif k et produit C_m comme un couple de points tel que :

$$C_m = \{kG, P_m + kP_B\}$$

Pour déchiffrer, B devra multiplier le premier point par sa clé privée, et soustraire le résultat au second point reçu :

$$P_m + kP_B - nB(kG) = P_m + k(nBG) - nB(kG) = P_m$$

3.5 Elliptic Curve Integrated Encryption Scheme (ECIES)

Le protocole ECIES [19] est un système de chiffrement à clé publique basé sur ECC.

Supposons qu'Alice désire envoyer un message M à Bob d'une manière sécurisée, ils doivent d'abord disposer de toutes les informations suivantes :

- KDF (Fonction Dérivation Key) : une cryptographique fonction de hachage peut générer les clés de toute longueur.
- MAC (Message Authentication Code) : Code transmis avec les données dans le but d'assurer l'intégrité de ces dernières.
- SYM : Algorithme de chiffrement symétrique.
- $E(F_p)$: La courbe elliptique utilisée avec le point de générateur G dont $\text{ord}_p(G) = n$.
- K_B : La clé publique de Bob $K_B = k_B.G$ où $k_B \in [1, n-1]$ est sa clé privée.

Pour chiffrer le message M , Alice doit effectuer des opérations suivantes :

1. Choisir un nombre entier $k \in [1, n - 1]$ et calculer $R = k.G$.
2. Calculer $Z = k.K_B$.
3. Générer les clés $(k_1, k_2) = \text{KDF}(\text{abscisse}(Z), R)$.
4. Chiffrer le message $C = \text{SYM}(k_1, M)$.
5. Générer le code MAC $t = \text{MAC}(k_2, C)$.
6. Envoyer (R, C, t) à Bob.

Pour déchiffrer le message (R, C, t) , Bob doit effectuer des calculs ci-dessous :

1. Rejeter le message si $R \notin E(F_p)$.
2. Calculer $Z = k_B.R = k_B.k.G = k.K_B$.
3. Générer les clés $(k_1, k_2) = \text{KDF}(\text{abscisse}(Z), R)$.
4. Générer le code MAC $t' = \text{MAC}(k_2, C)$.
5. Rejeter le message si $t' \neq t$.
6. Déchiffrer le message $M = \text{SYM}^{-1}(k_1, C)$.

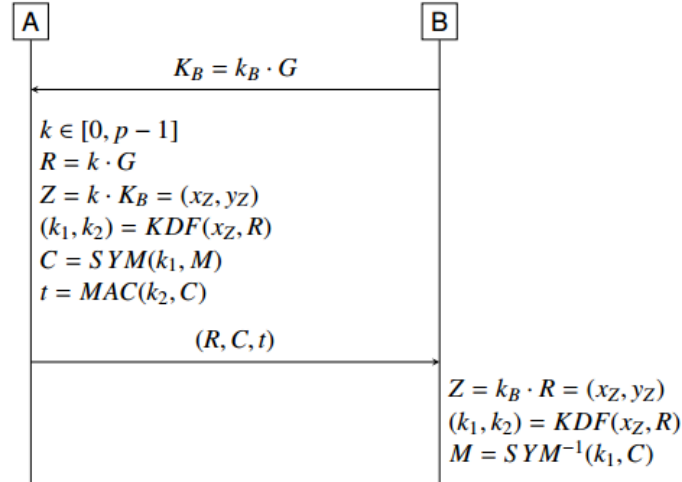


Figure 0.2 : Protocole de chiffrement ECIES [19]

3.6 McEliece

Le premier cryptosystème basée sur les codes correcteurs est introduit par Robert McEliece en 1978 [16]. Il utilise la théorie des codes correcteurs d'erreurs à des fins cryptographiques, et plus précisément pour un le chiffrement asymétrique. Le principe du cryptosystème qu'il décrit consiste à faire envoyer par Alice un message contenant un grand nombre d'erreurs, erreurs que seul Bob sait détecter et corriger.

Soit $n = 2^m$, x le message, G la matrice génératrice sous forme systématique d'un code, $x' = xG$ est de la forme (x/y) où y est $(n - k)$ -bits. Ce sont ces $n - k$ bits (*redondance*) qui permettent de corriger des erreurs. Il faut m bits pour corriger une erreur, soit, pour corriger t erreurs, mt bits au plus. Comme on a $2^m - k$ redondances, on peut corriger t_0 erreurs, avec $t_0 \geq (2^m - k)/m$. Par exemple, avec $m = 10$ (donc $n = 1024$) et $k = 512$, on peut corriger 52 erreurs au moins.

3.6.1 Codes de Goppa

Les codes de Goppa [17] sont des codes linéaires sur un corps fini F_p . Cependant, leur construction passe par l'utilisation d'une extension F_{p^m} . Un code de Goppa $\Gamma(L, g)$ est défini par son polynôme de Goppa g de degré t à coefficients dans F_{p^m} et son support $L \subset F_{p^m}$ de n éléments. Si on note $\alpha_0, \dots, \alpha_{n-1}$ les éléments de son support, sa matrice de parité est alors obtenue à partir de la matrice suivante :

$$H = \begin{pmatrix} \frac{1}{g(\alpha_0)} & \cdots & \frac{1}{g(\alpha_{n-1})} \\ \vdots & \vdots & \vdots \\ \frac{\alpha_0^{t-1}}{g(\alpha_0)} & \cdots & \frac{\alpha_{n-1}^{t-1}}{g(\alpha_{n-1})} \end{pmatrix} \quad (2)$$

Chaque élément de cette matrice est ensuite décomposé en m éléments de F_p , placés en colonnes, en utilisant une projection de F_p^m dans F_p^m . On passe ainsi d'une matrice de taille $t \times n$ sur F_p^m à une nouvelle matrice de parité H de taille $mt \times n$ sur F_p . Les éléments du code $\Gamma(L, g)$ seront donc tous les éléments c de F_p^m tels que :

$$H \times c^T = 0.$$

C'est donc un code de longueur n et dimension $k \geq n - mt$. De plus, un tel code a une distance minimale au moins égale à $t + 1$. En effet, toute sous-matrice carrée $t \times t$ de H est inversible car H s'écrit comme le produit d'une matrice de Vandermonde et d'une matrice diagonale inversible :

$$H = \begin{pmatrix} 1 & \dots & 1 \\ \vdots & \vdots & \vdots \\ \alpha_0^{t-1} & \dots & \alpha_{n-1}^{t-1} \end{pmatrix} \times \begin{pmatrix} \frac{1}{g(\alpha_0)} & & 0 \\ & \ddots & \\ 0 & & \frac{1}{g(\alpha_{n-1})} \end{pmatrix} \quad (3)$$

Il n'existe donc pas de mot de code de poids inférieur ou égal à t . Les codes de Goppa sont donc de la forme $[n, k \geq n - mt, d \geq t + 1]$ sur F_p , avec comme seule contrainte de longueur $n \leq 2^m$.

3.6.2 Génération de clés

Entrer : t, m (entiers)

Sortie : $K_{\text{pub}}, K_{\text{priv}}$

- $n \leftarrow 2^m, k \leftarrow n - m \cdot t$
- $C \leftarrow$ aléatoire code linéaire C capable de corriger t erreurs, dimension (k, n) .
- $G \leftarrow$ matrice génératrice pour le code C de dimension (k, n) .
- $S \leftarrow$ une matrice binaire inversible de dimension (k, k) .
- $P \leftarrow$ une matrice de permutation de dimension (n, n) .
- $G' \leftarrow$ une matrice $S \times G \times P$ de dimension (k, n) .
- return clé publique (G', t) ; clé privée (S, G, P) .

3.6.3 Chiffrement

Bob veut alors envoyer le message x à Alice. Pour cela, il calcule le produit de x par G' auquel il ajoute un vecteur d'erreur de poids convenable. Il envoie alors à Alice le vecteur calculé que l'on appellera y . Mathématiquement cela s'écrit

$$y = x.G' + e \quad \text{où } w(e) \leq t \quad (4).$$

3.6.4 Déchiffrement

Alice reçoit y , calcule $u = y.P^{-1}$ (5), u est alors un mot du code de Goppa de matrice génératrice SG contenant t erreurs, déterminer u' en corrigeant les erreurs de u . Calculer

$$x = u'.S^{-1} \quad (6).$$

3.7 Fonction du hachage

Une fonction de hachage est une fonction prenant en entrée un élément de taille variable et renvoyant en sortie un élément de taille fixe et prédéterminée. [18]

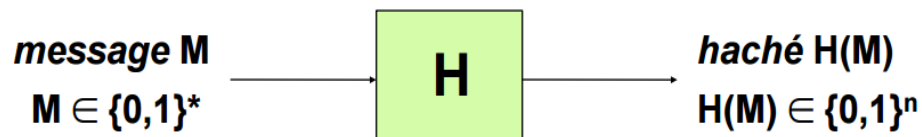


Figure 0.3: principe de hachage.

3.7.1 Les fonctions de hachage

Une fonction de hachage H de taille de sortie n est un algorithme faisant correspondre à un message M de taille arbitraire un élément $H(M)$ de taille n bits appelé haché. En pratique, n est de l'ordre de plusieurs centaines de bits, typiquement 128, 160, 256 ou 512 bits. [18]

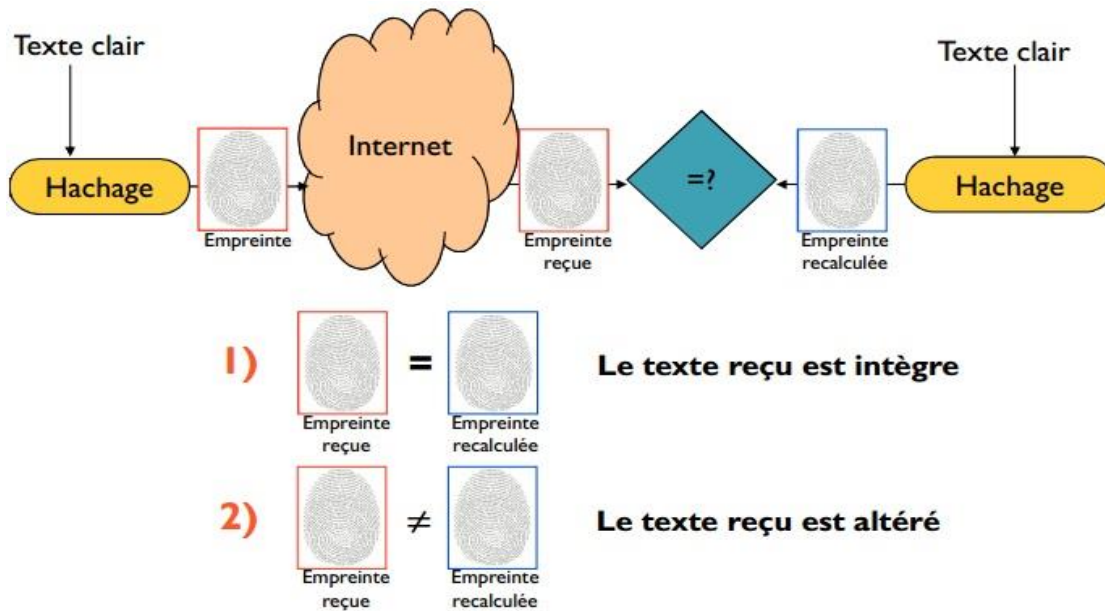


Figure 0.4: fonction de hachage.

Voici quelque fonction de hachage :

- MD4 et MD5 (Message Digest) furent développées par Ron Rivest. MD5 produit des hachés de 128 bits en travaillant les données originales par blocs de 512 bits. [20][21]
- SHA-1 (Secure Hash Algorithm 1), basé sur MD4. Il fonctionne également à partir de blocs de 512 bits de données et produit par contre des condensés de 160 bits en sortie. Il nécessite donc plus de ressources que MD5. [22]
- SHA-2 (Secure Hash Algorithm 2) a été publié récemment et est destiné à remplacer SHA-1. Les différences principales résident dans les tailles de hachés possibles : 256, 384 ou 512 bits. Il sera bientôt la nouvelle référence en termes de fonction de hachage. [23]

3.7.2 Utilisation de hachage

- Intégrité de fichier.
- Stockage de mots de passe.
- Intégrité de communications.
- Signature numérique.

3.8 Signature numérique

La signature numérique (parfois appelée digitale/électronique) est un mécanisme permettant de garantir l'intégrité d'un document électronique et d'en authentifier l'auteur, par analogie avec la

Signature manuscrite d'un document papier. Et aussi la signature digitale apporte la non-répudiation à l'origine.

Une signature digitale est produite par un algorithme de génération de signatures digitales et est vérifiée par un algorithme de vérification de signatures digitales.

3.8.1 Conditions

Les conditions suivantes doivent être réunies :

- Authentique : L'identité du signataire doit pouvoir être retrouvée de manière certaine.
- Infalsifiable : La signature ne peut pas être falsifiée. Quelqu'un ne peut se faire passer pour un autre.
- Non réutilisable : La signature n'est pas réutilisable. Elle fait partie du document signé et ne peut être déplacée sur un autre document.
- Inaltérable : Un document signé est inaltérable. Une fois qu'il est signé, on ne peut plus le modifier.
- Irrévocable : La personne qui a signé ne peut le nier.

3.8.2 Elliptic Curve Digital Signature Algorithm (ECDSA)

Le protocole ECDSA [19] est proposé par Johnson et al. Dans [15], il est une variante de DSA qui utilise les techniques de cryptographie sur les courbes elliptiques. Nous supposons qu'Alice et Bob utilisent la même courbe elliptique $E(F_p)$ pour sécuriser la communication entre eux. Nous supposons que la clé publique d'Alice est $K_A = k_A.G$ où k_A est sa clé privée et G est le point de générateur de l'ordre n .

Pour signer un message M , Alice doit suivre les opérations suivantes :

1. Choisir un nombre aléatoire $k \in [1, n - 1]$.
2. Calculer $R = k.G$.
3. Calculer $r \equiv X_R \pmod{n}$.
Si $r = 0$, retourner à l'étape 1.
4. Calculer $s \equiv k^{-1} (H(M) + k_A r) \pmod{n}$ où H est une fonction de hachage.
Si $s = 0$, retourner à l'étape 1.
5. Envoyer (r, s) à Bob.

Après avoir reçu le message signé, Bob vérifie la signature du message :

1. Vérifier si $K_A \neq \infty$ (point à l'infini) et $K_A \in E(F_p)$.
2. Vérifier si $n.K_A = \infty$ car $n.K_A = n.k_A.G$ et $\text{ord}_p(G) = n$.

3. Vérifier si $(r, s) \in [1, n - 1]$.
4. Calculer $R = (H(M)s^{-1} \bmod n) G + (rs^{-1} \bmod n) K_A$ (voir la formule 1).
5. Vérifier si $r \equiv \text{abscisse}(R) \pmod{n}$.

$$\begin{aligned}
 R &= (H(M)s^{-1})G + (rs^{-1}) K_A \pmod{n} \\
 &= (H(M)s^{-1})G + (rs^{-1}) K_A G \pmod{n} \\
 &= s^{-1}G(H(M) + r K_A) \pmod{n} \quad (1) \\
 &= k(H(M) + K_A r) - 1G(H(M) + r K_A) \pmod{n} \\
 &= kG
 \end{aligned}$$

Le déroulement du protocole est montré dans la figure 3.9.

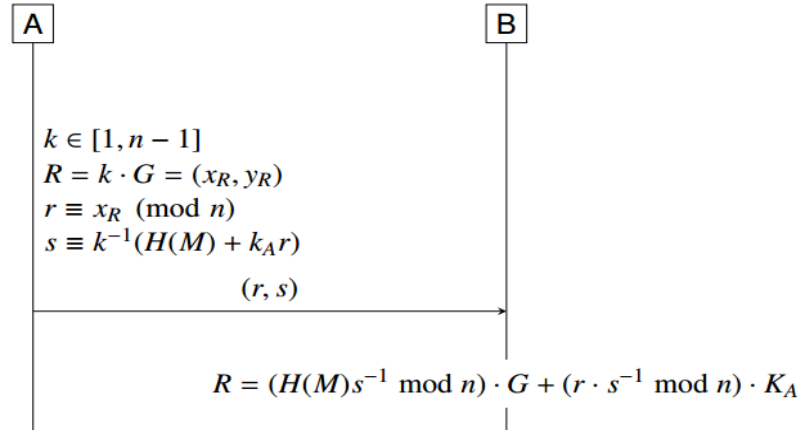


Figure 0.5: Protocole de signature numérique ECDSA [19]

3.9 Conclusion

Le rôle de la cryptologie a évolué au fil des siècles. Auparavant, elle n'avait pour rôle que de protéger un texte écrit. Actuellement, la cryptologie s'étend dans différents domaines tels que la téléphonie, le stockage des données, et les communications avec les satellites. La cryptographie joue un rôle important pour la sécurisation des réseaux de capteurs sans fil. Dans le chapitre suivant, on fait une implémentation de différentes primitives cryptographiques étudié dans le capteur des RCSF.

Chapitre 4

IMPLEMENTATION

ET RESULTATS

EXPERIMENTAUX

CHAPITRE 4

IMPLEMENTATION ET RESULTATS

EXPERIMENTAUX

4.1 Introduction

Dans ce chapitre nous présentons l'environnement de simulation utilisé. Ensuite on évalue la performance de différentes primitives cryptographiques étudiées dans un capteur en termes de consommation de l'énergie, occupation de l'espace de mémoire, et le temps d'exécution. Nos résultats expérimentaux sont basés sur l'utilisation de deux types des simulateurs TOSSIM [43] et AvroraZ[42] et en utilisant le capteur MicaZ.

4.2 Le système d'exploitation Tinyos

4.2.1 Présentation

TinyOS [44] est un système d'exploitation intégré, modulaire, destiné aux réseaux de capteurs miniatures. Cette plate-forme logicielle ouverte et une série d'outils développés par l'Université de Berkeley est enrichie par une multitude d'utilisateurs. TinyOS respecte une architecture basée sur une association de composants, réduisant la taille du code nécessaire à sa mise en place. Cela s'inscrit dans le respect des contraintes de mémoires qu'observent les réseaux de capteurs.

La librairie TinyOS comprend les protocoles réseaux, les services de distribution, les drivers pour capteurs et les outils d'acquisition de données. TinyOS est en grande partie écrit en C mais on peut très facilement créer des applications personnalisées en langages C, NesC, et Java.[25]

4.2.2 Propriétés

TinyOS a été créé pour répondre aux caractéristiques des réseaux de capteurs, telles que :

- Une basse consommation d'énergie.
- Une taille de mémoire réduite.
- Des opérations robustes (puissance de calcul).
- Une faible Bande passante.

4.3 Le langage NesC

NesC est un langage de programmation dérivé du langage C, fait pour minimiser l'utilisation de mémoire et de puissance de calcul par les capteurs, qui très souvent disposent de ressources très limitées (batterie de faible puissance et non changeable, mémoire réduite...) [26].

4.3.1 Concepts de nesC

4.3.1.1 Composant

L'unité de code de base de nesC est le composant "component", Il existe deux types de composants :

- a. Module : composant implémenté avec du code
- b. Configuration : composants reliés ensemble pour former un autre composant

4.3.1.2 Interface

Une interface définit les interactions entre deux composants. Elles spécifient un ensemble de fonctions à implémenter par les composants fournisseurs de l'interface (commands), et un ensemble à implémenter par les composants utilisateurs de l'interface (events).

4.3.2 Compiler et exécuter une application nesC

Le compilateur traite les fichiers nesC en les convertissant en un fichier C. Ce fichier contiendra l'application et les composants de l'OS utilisés par l'application. Ensuite un compilateur spécifique à la plateforme cible compile ce fichier C. Il deviendra alors un seul exécutable. Le chargeur installe le code sur le "mote" (Micaz, TelosB, etc.) [25].

Extension des fichiers nesC: .nc

- Clock.nc : soit une interface (ou une configuration)
- ClockC.nc : une configuration
- ClockM.nc : un module

4.3.3 Bibliothèques cryptographiques

Pour évaluation de primitive Cryptographique avec le simulateur TOSSIM on a utilisé des bibliothèques basées sur le langage NesC. Ces bibliothèques sont :

- TinyPKC : il effectue la cryptographie asymétrique pour TinyOS 2.x basé sur l'originale mise en œuvre CyaSSL [45]. Il offre un soutien pour les opérations de chiffrement/déchiffrement du cryptosystème RSA.

- TinyECC [46] : Il fournit un schéma de signature numérique (ECDSA), un protocole d'échange de clé (ECDH), et un système de chiffrement à clé publique (ECIES).
- Trivium [47] : c'est une bibliothèque qui offre les fonctions suivantes: chiffrement / déchiffrement du cryptosystème AES, et le chiffrement / déchiffrement du cryptosystème Trivium pour 8-bit et 16-bit microcontrôleur.

Concernant le cryptosystème McEliece, on développe l'algorithme de chiffrement avec le langage NesC.







Dans les différentes primitives implémentées, on utilise les clés (publique et privée) avec des valeurs statiques pour éviter l'étape de génération de clé aléatoire.

4.4 TOSSIM

Le simulateur TOSSIM [43] afin de simuler le comportement des capteurs, un outil très puissant a été développé et proposé sous le nom de TOSSIM. Ce dernier est souvent utilisé avec une interface graphique JTOSSIM pour une meilleure compréhension et visualisation de l'état du réseau. L'utilisation de ces deux logiciels est immédiate dès lors que TinyOS est opérationnel. L'annexe 1 présente détaillé l'installation et utilisation de JTOSSIM.

4.5 PowerTOSSIMz

Permet de simuler la consommation dans chaque périphérique au niveau de chaque nœud du réseau. Pour récupérer l'énergie consommée par les nœuds du réseau, il doit être suivi les étapes suivantes :[27]

-  Accédez à l'application à simuler.
-  Créez deux sous-dossiers "Topologies" et "Simulations" dans le dossier principal. Qui contiennent successivement les fichiers de topologies utilisées pendant la simulation et le fichier de trace généré par la simulation.
-  Compilez l'application (**make micaz sim**).
-  Créez le script de simulation "MySimulation.py"
-  Exécutez le script "MySimulation.py", ce dernier va générer un fichier "Energy.txt" dans le sous-dossier "Simulations" (\$ **python MySimulation.py**)
-  Exécutez "postprocessZ.py" sur la trace de simulation "Energy.txt".
(**python postprocessZ.py Energy.txt > Result.txt**)

- ✚ Le résultat enregistre l'énergie totale utilisée par chaque composant sur chaque nœud.

4.6 AvroraZ

Avrora [42] est un simulateur pour les réseaux de capteurs. Pour obtenir une simulation fiable de chaque nœud, il simule toutes les instructions qui s'exécutent sur ce nœud. Avrora est écrit en Java et chaque nœud est implémenté par un thread Java.

Pour simuler avec AvroraZ suivre les étapes suivant :

- ✚ `cd $TOSROOT/apps/Blink`
- ✚ `make micaz`
- ✚ `mv build/micaz/main.exe Blink.elf`
- ✚ `java -jar avroraZ.jar-platform=micaz -simulation=sensor-network -seconds=3 -monitors = energy Blink.elf`

4.7 Résultats Expérimentaux

4.7.1 Paramètres d'implémentation

On rappelle que le type de capteur choisi est de type MicaZ qui comporte les caractéristiques essentielles suivantes: la taille de ROM Flash est 128 KOctet et la taille de RAM est 4KOctet. Pour plus détailler, voir la Table 1.1 (chapitre 1).

Les tailles des clés utilisées pour chaque cryptosystème est comme le suivant:

- Cryptosystème AES avec une clé de taille 128 bits,
- Cryptosystème Trivium avec une table d'état interne de 288 bits, avec une clé de taille 80 bits
- Cryptosystème ECIES et ECDSA avec une clé de taille 160 bits,
- Cryptosystème McEliece avec les paramètres suivantes ($N=64$, $K=34$, $T=5$), qui signifie la taille de la matrice publique $64 \times 34 = 2176$ bits.

4.7.2 Occupation de mémoire (ROM/RAM)

la table 4.1 représente les résultats obtenus après l'exécution des algorithmes cryptographiques symétrique dans les deux cas, chiffrement et déchiffrement, par rapport l'occupation de mémoire (ROM et RAM) par le simulateur TOSSIM. Par exemple, dans le cas AES l'espace occupée de chiffrement est 3912 et 4798 pour déchiffrement.

	Chiffrement		Déchiffrement	
	ROM	RAM	ROM	RAM
AES	3912	1037	4798	1805
Trivium8	8478	115	8420	114
Trivium16	6076	87	6018	86

Table 0.1 Occupation de mémoire pour la cryptographie symétrique (en Octet)

la table 4.2 représente les résultats obtenus après l'exécution des algorithmes cryptographiques asymétriques.

	Chiffrement		Déchiffrement	
	ROM	RAM	ROM	RAM
McEliece	29046	2437	-----	-----
ECIES	25766	2396	29046	2437
RSA	44936	2187	-----	-----

Table 0.2 : Occupation de mémoire pour la cryptographie asymétrique (en Octet)

la table 4.3 représente les résultats obtenus de l'exécution d'ECDSA pour la signature et leur résultat de vérification.

	Signature		Vérification	
	ROM	RAM	ROM	RAM
ECDSA	25422	2050	27120	2050

Table 0.3: occupation de mémoire pour la signature (en Octet).

La Figure 4.1 nous montre le résultat d'évaluation de la mémoire pour le chiffrement d'un texte par le cryptosystème McEliece. Les paramètres utilisés sont mentionné dans la section 7.1 (N=64, K=34, T=5).

4.7.3 Consommation d'énergie

la table 4.4 représente les résultats obtenus après l'exécution des algorithmes cryptographique symétrique par rapport consommation d'énergie. Par exemple Dans le cas Trivium8, l'énergie consommée pour le chiffrement est 25.5371 millijoule et 25.5369 milli joule pour le déchiffrement.

Unité de calcul en milli joule	PowerTOSSIMz		AVRORA	
	Chiffrement	Déchiffrement	Chiffrement	Déchiffrement
AES	0	0	25.4297	25.4590
Trivium8	0	0	25.5371	25.5369
Trivium16	0	0	25.4389	25.3163

Table 0.4: Consommation d'énergie pour la cryptographie symétrique

la table 4.5 représente les résultats obtenus après l'exécution des algorithmes cryptographiques symétriques par rapport la consommation d'énergie.

Unité de calcul en milli joule	PowerTOSSIMz		AVRORA	
	Chiffrement	Déchiffrement	Chiffrement	Déchiffrement
McEliece	0	-----	25.4925	-----
ECIES	293	290	43.89935	43.89973
RSA	294	-----	43.89022	-----

Table 0.5: Consommation d'énergie pour la cryptographie Asymétrique

la table 4.6 représente les résultats obtenus de l'exécution de la signature et leur résultat de vérification avec l'algorithme ECDSA.

Unité de calcul en milli joule	PowerTOSSIMz		AVRORA	
	Signature	Vérification	Signature	Vérification
ECDSA	293	291	43.7353	43.8958

Table 0.6: Consommation d'énergie dans ECDSA

4.7.4 Temps de traitement

On utilise le simulateur Avrora pour évaluer le temps de traitement des différentes phases (chiffrement, déchiffrement, signature et vérification) des algorithmes étudiés. La table 4.7 représente les résultats obtenus après l'exécution des algorithmes cryptographiques symétriques dans les deux cas chiffrement et déchiffrement par rapport au temps de simulation. Par exemple, dans le cas de Trivium16, le temps de chiffrement est 0.00315 ms et 0.00297 pour le déchiffrement.

	Chiffrement (ms)	Déchiffrement (ms)
AES	0.00301	0.00302
Trivium8	0.00347	0.00326
Trivium16	0.00315	0.00297

Table 0.7: Temps de simulation de cryptographie symétrique.

La table 4.8 représente les résultats obtenus après l'exécution des algorithmes cryptographiques asymétriques dans les deux cas, le chiffrement et le déchiffrement, par rapport au temps de simulation.

	Chiffrement (ms)	Déchiffrement (ms)
McEliece	0.00845	-----
ECIES	0.00271	0.00281
RSA	0.00315	-----

Table 0.8: Temps de simulation de cryptographie Asymétrique.

La table 4.9 représente les résultats obtenus de l'exécution ECDSA pour la signature et leur résultat de vérification.

	Signature (ms)	Vérification (ms)
ECDSA	0.00314	0.00318

Table 0.9 : Temps de simulation pour singateur

4.8 Discussion

Parmi les caractéristiques importantes des réseaux de capteurs sans fil, on cite la limitation des ressources informatiques des nœuds de capteurs. Pour cela, on a évalué chaque primitive cryptographique en terme de : l'espace de mémoire exigé dans la RAM et la ROM, le temps de traitement pour chaque phase (chiffrement, déchiffrement, signature, et vérification) dans les primitives étudiés, et la consommation d'énergie.

Occupation de la mémoire ROM Flash, d'après les tableaux 4.1, 4.2, et 4.3, AES exige 3912 octets pour le chiffrement et 4798 octets pour le déchiffrement. AES est le meilleur par rapport Trivium 8 et 16. Dans le coté des cryptosystèmes à clé publique, le premier est ECIES qui exige un espace 25766 KOctets , et le moins efficace est RSA où exige 44936 octets.

La mémoire RAM, Trivium16 exige seulement 87 octets pour la phase de chiffrement. Dans l'autre côté ECIES exige 2396 octets et RSA exige 2187 octets. Si on prise en considération que la mémoire volatile dans les nœuds de capteur de MicaZ de taille 4 KOctets, donc les résultats obtenus sont convient avec les capacités des capteurs.

La consommation d'énergie, d'après les tableaux (4.4, 4.5, et 4.6) et on prend les résultats de Avrora comme référence, le meilleur résultat de chiffrement est 25.4297 milli joule avec l'AES et aussi le cryptosystème McEliece avec 25.4925 milli Joule.

Le temps de traitement, d'après les tabeaux 4.7, 4.8, et 4.9, le meilleur algorithme de chiffrement est ECIES avec 0.00271 ms.

Pour choisir le meilleur cryptosystème, il faut prend en considération les contraintes suivantes : la sécurité, la performance, et le domaine d'application. Le cryptosystème AES est un cryptosystème de type symétrique (à clé privée), qui se pose le problème d'échange de clé entre les nœuds de capteur. Ce problème n'est pas posé dans les cryptosystèmes à clé publique. D'après nos résultats expérimentaux, le cryptosystème RSA est un cryptosystème très lourd par

rapport ECC (ECIES, ECDSA) et McEliece. Le problème qui se pose dans McEliece est la taille importante de la clé publique, pour le niveau de sécurité 60, il exige un espace de 536576 bits (67 KOctet), il ne convient pas avec les capacités des capteurs (voir la Figure 4.2). Donc le meilleur choix actuellement est le cryptosystème ECC. On peut implémenter ECC dans des nœuds de capteur de type MicaZ et aussi des autres types comme TelosB et Imote.

4.9 Conclusion

Nous avons présenté dans ce chapitre une implémentation et l'évaluation des différentes primitives cryptographies sur le capteur MicaZ, cette implémentation est réalisable en réel. Les résultats obtenus basés sur l'utilisation de systèmes asymétriques et symétriques.

CONCLUSION GENERALE

Depuis quelques années, les avancées technologiques en termes de miniaturisation des machines et des supports de communication y afférant ont rendu envisageable le déploiement et l'exploitation de milliers de capteurs. D'ailleurs, les réseaux de capteurs ont été identifiés comme l'une des technologies clefs de l'avenir et ce en raison de l'incroyable potentiel applicatif qu'elle renferme. Cependant, en raison de la jeunesse de cette technologie, le domaine de réseaux de capteurs soulève d'importantes problématiques de recherche en termes de la sécurisation de ces réseaux.

Dans ce mémoire, nous avons présenté les caractéristiques essentielles des réseaux de capteurs sans fil, ainsi que les besoins et les défis de la sécurité dans ces derniers. Ensuite, Nous avons présenté des primitives cryptographiques de type symétrique, asymétrique, et signature numérique.

On a développé l'algorithme de chiffrement du cryptosystèmes McEliece en NesC et on a utilisé des bibliothèques cryptographiques (TinyECC, TinyPKC, etc.) pour implémenter les autres cryptosystèmes. On a implémenté les différents cryptosystèmes étudiés dans le système embarqué des réseaux de capteurs sans fil, le TinyOS. Basant sur cette implémentation et avec l'utilisation de deux simulateurs TOSSIM et Avrora, on a fait une étude comparative entre ces primitives en termes de (1) l'occupation de la mémoire ROM Flash et la mémoire RAM, (2) consommation de l'énergie de nœud de capteur, et (3) le temps de traitement.

Notre travail réalisé est une étape initiative pour proposer une nouvelle approche de sécurisation des protocoles de communication dans les RCSF en utilisant les primitives cryptographiques à bas coût, celui-ci une perspective de notre travail.

BIBLIOGRAPHIE

- [1] W. Diffie, E. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, 1976, pp. 644–654.
- [2] A. PACHA, N. HADJ-SAID, La Cryptographie et ses principaux systèmes de références, Université des Sciences et de la Technologie d'Oran, Vol 12 n°01, Année 2002.
- [3] R. Dumont, Cryptographie et Sécurité informatique, Université de Liège, [En ligne] <http://www.montefiore.ulg.ac.be/~dumont/> , consulté le : 05/01/2016.
- [4] A. Biryukov, O. Dunkelman, N. Keller, D. Khovratovich and A. Shamir, Key Recovery Attacks of Practical Complexity on AES Variants With Up To 10 Rounds, Cryptology ePrint Archive, Report 2009/374, 2009.
- [5] Y. Tian, G. Chen, J. Li, On the Design of Trivium, Shanghai Jiaotong University, China.
- [6] I. Mallouli, Implémentation d'une bibliothèque de fonctions cryptographiques optimisées pour les réseaux de capteurs sans fil, Ingénieur, L'Ecole Nationale d'Ingénieurs de Sfax, 2011.
- [7] C. De Cannière, B. Preneel, Trivium Specifications, Katholieke Universiteit Leuven Belgium, 2002.
- [8] S. Ballet, A. Bonecaze, Courbes Elliptiques Application à la Cryptographie, Ecole Polytech de Marseille, <http://alexis.bonnecaze.perso.luminy.univ-amu.fr/CryptoAvancee.pdf>, consulté le : 25/2/2016
- [9] H. Michael, Courbes elliptiques et cryptographie, Marusia Rebolledo, 2006, <http://math.univ-bpclermont.fr/~rebolledo/page-fichiers/projetMichael.pdf>, consulté le : 25/2/2016
- [10] Z. KHERBACHE, A. LARIBI, Étude de la Qualité de Service (QoS) dans les réseaux WIFI, Master, Université Tlemcen, 2011.
- [11] C. De Cannière, B. Preneel, "Trivium – A Stream Cipher Construction Inspired by Block Cipher Design Principles", Katholieke Universiteit Leuven Belgium, 2002.
- [12] D. SOW, Courbes elliptiques, Cryptographie à clés publiques et Protocoles d'échange de clés, Doctorale, Dakar, 2013.

- [13] C. Grenier, Techniques de cryptanalyse de RSA, 2009.
- [14] Signature numérique, [En ligne] http://pybookmarks.readthedocs.io/en/latest/development/security/signature_numerique. Consulté le : 15/4/2016
- [15] D. Johnson, A. Menezes, and S. Vanstone. The elliptic curve digital signature algorithm (ECDSA). International Journal of Information Security, 1, pp. 36–63, 2001.
- [16] 1978. Robert J. McEliece. "A public-key cryptosystem based on algebraic coding theory." Jet Propulsion Laboratory DSN Progress Report 42–44, 114–116.
- [17] P.-L. Cayrel, Construction et optimisation de cryptosystèmes basés sur les codes correcteurs d’erreurs, Doctorat, Université de Limoges, 2008.
- [18] T. Peyrin, Analyse de fonctions de hachage cryptographiques, Doctorat, Paris, 2008.
- [19] Y. SHOU, Cryptographie sur les courbes elliptiques et tolérance aux pannes dans les réseaux de capteurs, Doctorat, l’Université de Franche-Comté, 2014
- [20] MD4 Message-Digest Algorithm, [En ligne] <http://www.ietf.org/rfc/rfc1320.txt> consulté le : 06/02/2016
- [21] MD5 message-digest Algorithm. [En ligne] <http://www.ietf.org/rfc/rfc1321.txt> consulté le : 06/02/2016
- [22] National Institute of Standards and Technology. FIPS 180-1: Secure Hash Standard, April 1995. [En ligne] <http://csrc.nist.gov/groups/ST/hash/index.html> consulté le : 07/02/2016
- [23] National Institute of Standards and Technology. FIPS 180-2: Secure Hash Standard, August 2002. [En ligne] <http://csrc.nist.gov/groups/ST/hash/index.html> consulté le : 07/02/2016
- [42] ZigBee Specification. Zigbee standards organization. Document 053474r17, Jan 17 (2008).
- [25] C. Yacine, « Réseaux de Capteurs Sans Fils », support-SIT60, Vol. 103, pp. 14-17, 2008.
- [26] D. Gay, P. Levis, R. von Behren, M. Welsh, E. Brewer, and D. Culler, “The nesC language: A holistic approach to networked embedded systems,” in SIGPLAN Conference on Programming Language Design and Implementation (PLDI’03), 2003
- [27] F. MEZRAG, Sécurité du Routage Hiérarchique Basé sur les Clusters dans les Réseaux de Capteurs sans Fil, Magister, Université Amar Telidji - Laghouat, 2015.

- [28] S.Bouguer, étude et simulation comparative entre les réseaux de capteurs sans fils traditionnels et les réseaux de capteurs véhiculaires, Ingénieur, Université Tlemcen, 2012.
- [29] N. Bounegta, N. Aici, Approche Décentralisé pour la sécurité d'un Réseau de Capteurs Sans Fil (RCSF), Ingénieur, Université de Bechar, 2010
- [30] Adams and T. Jon. An introduction to IEEE STD 802.15. 4. in Aerospace Conference, pp. 8. IEEE (2006).
- [31] T. Watteyne, Proposition et validation formelle d'un protocole MAC temps réel pour réseaux de capteurs linéaires sans fils, Laboratoire CITI 2004/2005.
- [32] Y. Younes, Minimisation d'énergie dans un réseau de capteurs, magister, université mouloud Mammeri de Tizi-Ouzou ,2012
- [33] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci. "Wireless sensor networks: a survey". Computer Networks 38, Elsevier Science, pp. 393–422, 2002.
- [34] S. Maarouf, S. Ouadah, Implémentation et évaluation des schémas de routage sur une plateforme réelle de réseaux de capteurs sans fil, Master, Université Tlemcen, 2014.
- [35] M.Messai, Sécurité dans les Réseaux de Capteurs Sans-Fil, Magistère, Université Abderrahmane Mira de Bejaia, 2008.
- [36] Crow, P. Brian, Widjaja, Indra, Kim, LG, Sakai, and T. Prescott. IEEE 802.11 wireless local area networks. Communications Magazine, IEEE 35(9), pp.116–126 (1997).
- [37] IEEE 802.11, [En ligne] <http://www.ieee802.org/11/>, Consulté le: 15/04/2016
- [38] Eklund, Carl, Marks, B. Roger, Stanwood, L. Kenneth, Wang, and Stanley. IEEE standard 802.16: A technical overview of the WirelessMAN/sup TM/air interface for broadband wireless access. Communications Magazine, IEEE 40(6), 98–107 (2002).
- [39] IEEE 802.16, [En ligne] <http://www.ieee802.org/16/>, Consulté le : 15/04/2016
- [40] SIG Bluetooth. Inc., specification of the Bluetooth system: Core, (2001).
- [41] M.HADJILA, protocoles de routage économes en énergie pour les réseaux de capteurs sans fil, Doctorat, Université De Tlemcen, 2014.

- [42] Ben L. Titzer, al, “Aurora: Scalable Sensor Network Simulation with Precise” Timing, 2005
- [43] Simulateur TOSSIM, [En ligne] <http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM> consulté le : 06/05/2016.
- [44] Tinyos Documentation Wiki [En ligne], http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Documentation_Wiki. Consulté le : 29/04/2016
- [45] CyaSSL [En ligne], <http://www.yassl.com/yaSSL/Products-cyssl.html> consulté le:10/05/2016
- [46] Tiny ecc page officielle [En ligne], <http://discovery.csc.ncsu.edu/software/TinyECC/>. consulté le : 11/05/2016.
- [47] Trivium site officielle [En ligne], <http://www.ecrypt.eu.org/stream/triviumpf.html> consulté le : 12/05/2016
- [48] I.MANSOUR, Contribution à la sécurité des communications des réseaux de capteurs sans fil, Doctorat, Université BLAISE PASCAL, 2013.

Annexe 1 : JTOSSIM

1.1 Installation

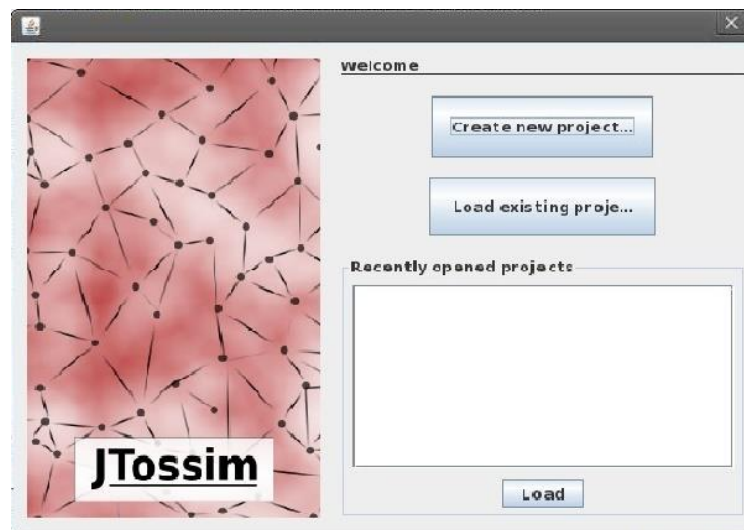
- Téléchargez le fichier **jtossim-0.1.1.zip**
- décompressez ce fichier dans **/opt/tinyos-2.x**.

Pour vérifier que est correctement configuré essayez d'exécuter **make micaz sim** dans votre répertoire de projet

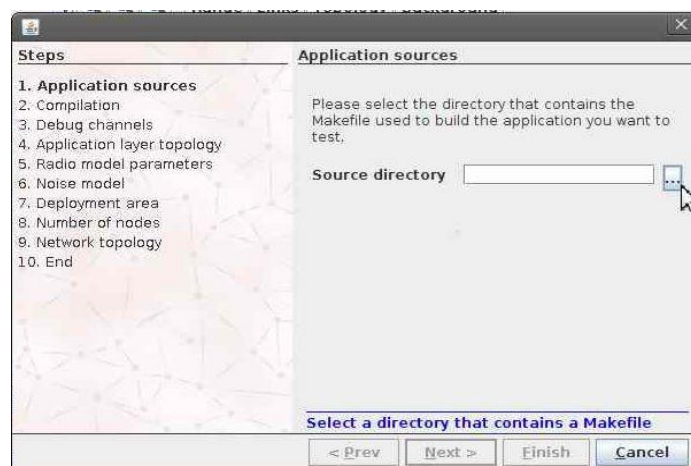
1.2 Lancement de JTOSSIM

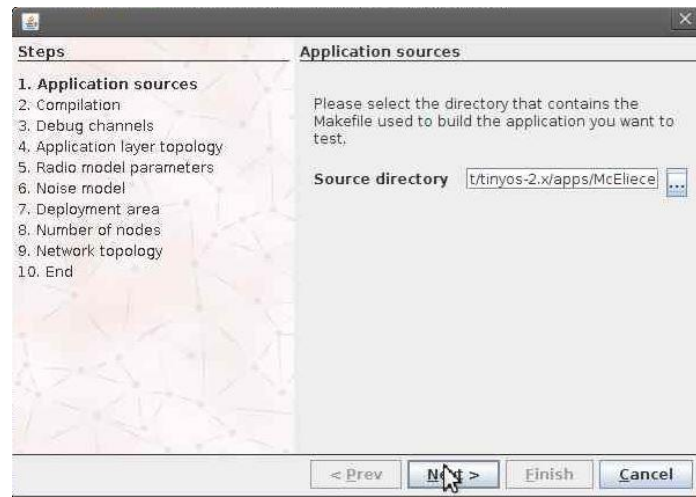
Pour lancer Jtossim, il est nécessaire de passe par les étapes suivantes :

- + Ouvrez le terminal.
- + Allez dans le répertoire jtossim-0.1.1 en utilisant la commande **cd**
- + Tapez **java -jar JtosSim.jar**
- + Cliquez sur le bouton **Create new project**.

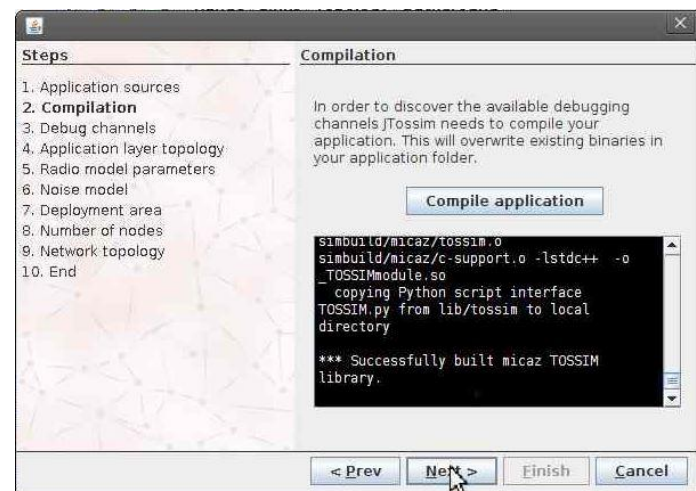
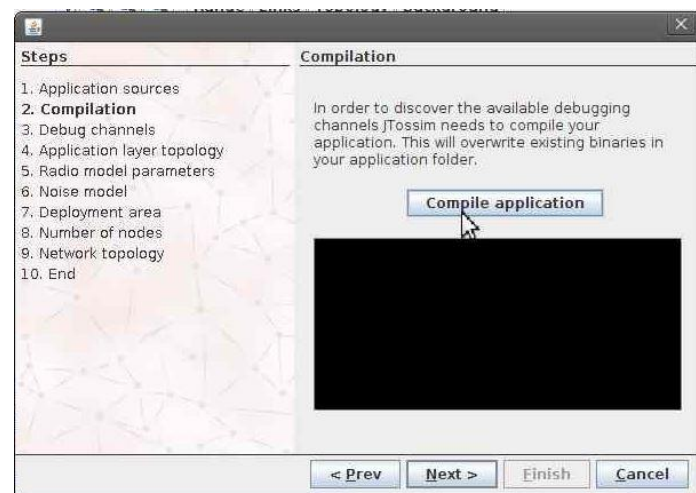


- + Choisissez le dossier qui contient l'application à compiler. Ensuite, cliquez sur **Next**.

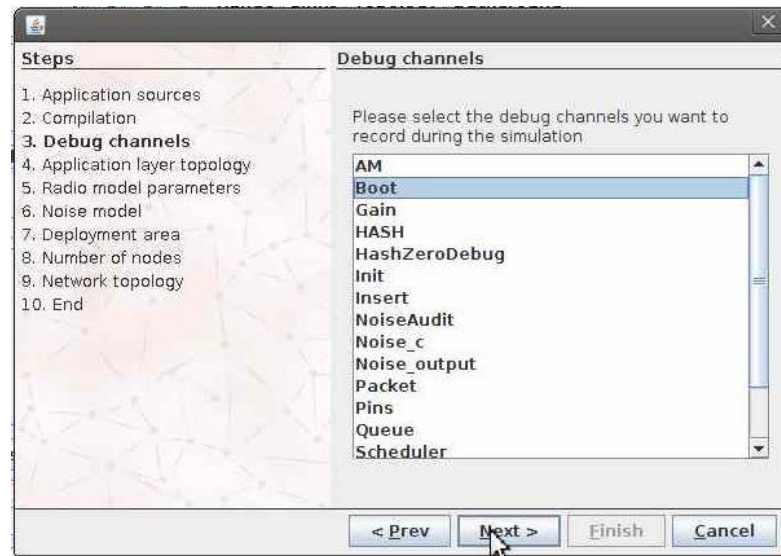




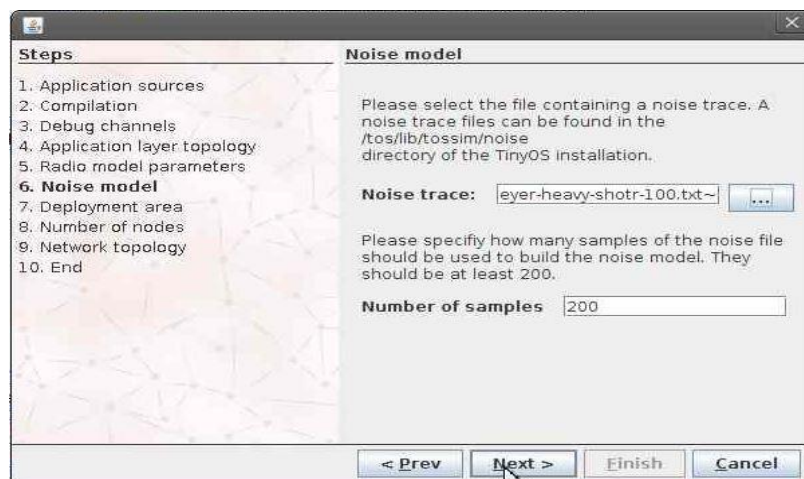
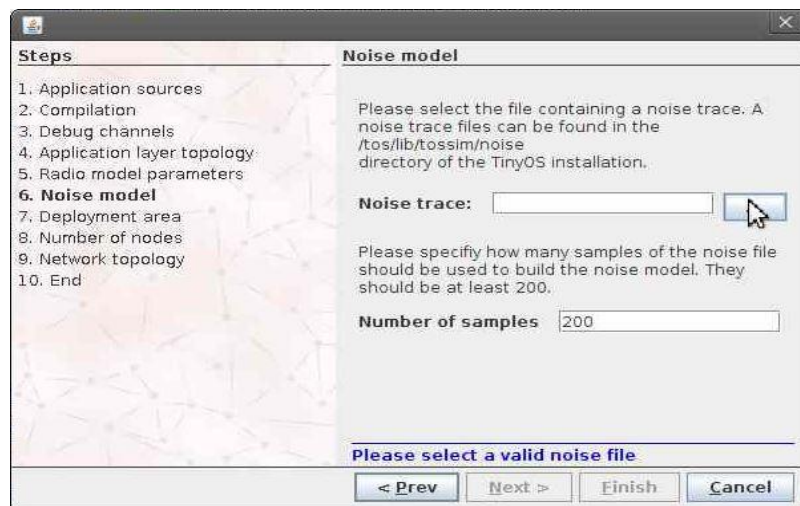
- + Cliquez sur le bouton **Compile application**, Une fois que la compilation est terminée sans erreurs, vous pouvez cliquer sur **Next**.



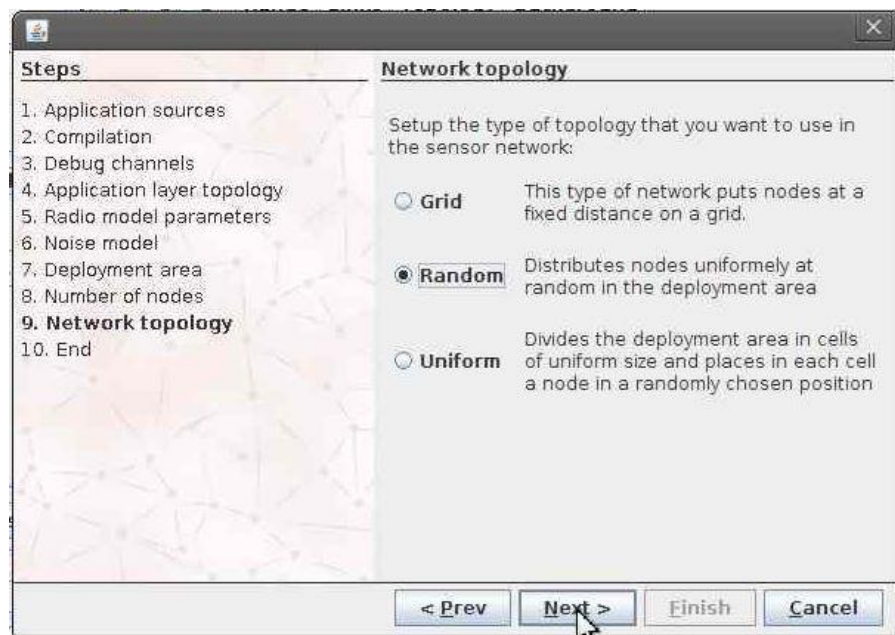
- ✚ Choisissez par la suite le canal de sortie (debug channel) qui est utilisé par votre application.



- ✚ Dans cette étape, vous pouvez designer le modèle de bruit à utiliser dans la simulation.



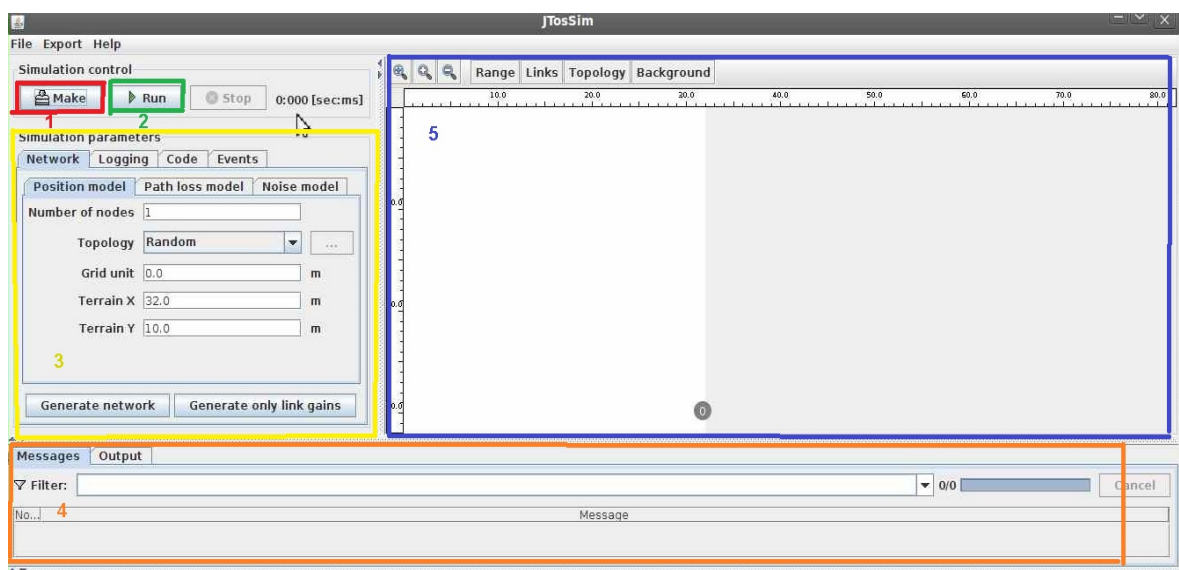
- ✚ Dans la prochaine étape, vous devez déterminer la taille de la zone où les nœuds seront d'employés, cliqué sur **Next**.
- ✚ choisir le nombre de nœuds que vous devez déployer, cliquer sur **Next**.
- ✚ Dans la prochaine étape, vous avez besoin de choisir le type de la topologie désiré. ensuite cliquer sur **Next**.



- ✚ Enfin cliquer sur **Finish**.

1.3 Simulation

La figure suivante illustre l'interface principale de JTossim.



- 1 Make, pour la compilation de projet.
- 2 Run, pour lancer la simulation et voir les messages de cette simulation.
- 3 Permet de modifier les paramètres de simulation.
- 4 Permet de visualiser la simulation sous forme de messages.
- 5 Permet de visualiser l'emplacement des capteurs.