# Sécurité dans les réseaux de capteurs sans fil : conception et implémentation

Wassim Drira, Chakib Bekara, Maryline Laurent


# Sécurité dans les réseaux de capteurs sans fil
# Conception et implémentation


| | | |
|---|---|---|
| **LO**giciels-**R**éseaux | **Wassim Drira**<br>**Chakib Bekara**<br>**Maryline Laurent-Maknavicius** | 08012-LOR<br>2008 |

**Security in wireless sensor networks - design and implementation**

—oOo—

***Abstract***

The wireless sensor networks applications become more diversified with the advent of miniature and efficient sensors. The ANR CAPTEURS project aims to apply wireless sensor networks to supervise the cold chain. The use of wireless communication, lack of reliable entity, the diversity of organizations producing and managing sensors, and the sake of saving sensor's energy ressources led us to design and implement a new security protocol designed essentially to authenticate exchanges between sensors. The purpose of this work is, first, the design and implementation of a security protocol and the second is testing this solution integrated with Placide, the network management protocol. The security protocol is essentially composed of two phases : a negotiation phase of pairwise keys between sensors which has been incorporated into the initialization phase of Placide, and a permanent phase during which the packets are authenticated (data encryption is optional).
**Key-words :** Security, wireless sensor network, WSN, Placide, cold chain, TinyOS, authentication, pairwise keys.

***Résumé***

Les applications des réseaux de capteurs sans fil se diversifient avec l'apparition de capteurs miniatures et performants. Le projet ANR CAPTEURS s'intéresse aux réseaux de capteurs sans fil appliqués à la supervision de la chaîne du froid. L'utilisation de la communication sans fil, l'absence d'entité de confiance, la diversité des organismes produisant et gérant les capteurs, ainsi que le souci d'économiser les ressources énergétiques des capteurs nous a conduits à concevoir et implémenter un nouveau protocole de sécurité visant essentiellement à authentifier les échanges entre capteurs. L'objet de ce travail est, en premier lieu, la conception et l'implémentation d'un protocole de sécurité et en second lieu le test de cette solution intégrée avec Placide, le protocole de gestion du réseau. Le protocole de sécurité est composé essentiellement de deux phases : une phase de négociation des clés de paires entre les capteurs qui a été intégrée dans la phase d'initialisation du protocole Placide, et la phase de régime permanent au cours de laquelle les paquets sont authentifiés (le chiffrement des données étant optionnel).
**Mots-clés :** Sécurité, réseau de capteurs sans fil, WSN, Placide, chaîne du froid, TinyOS, authentification, clés de paire.

—oOo—

| Wassim Drira | Chakib Bekara | Maryline Laurent-Maknavicius |
|---|---|---|
| Stagiaire | Doctorant | Professeur |

TELECOM & Management SudParis - Département LOR
9 rue Charles Fourier 91011 Evry cedex
{Wassim.Drira|Chakib.Bekara|Maryline.Maknavicius}@it-sudParis.eu

# Table des matières

# Table des figures

# Liste des tableaux

# Introduction

Depuis longtemps, l'être humain cherche à concrétiser son confort, son bien être, sa santé, sa sécurité, et l'innovation dans sa vie. Il a trouvé dans les capteurs une pierre pour construire ce chemin. De nos jours, les capteurs sont partout : dans les rues, dans les maisons, dans les bureaux, dans la voiture, aux frontières d'un pays, sous l'eau, dans les barrages et dans les forêts. Les avancées des technologies sans fil et de la micro-électronique ont poussé cette évolution. Actuellement, les réseaux sont composés de dizaines, de milliers, voire de millions de capteurs sans fil qui génèrent des informations (de surveillance, de commande, etc) et nécessitent des protocoles pour assurer leur organisation et sécurisation.

Dans ce cadre, nous participons au projet CAPTEURS, lancé en décembre 2005. Son but est de conçevoir et valider un protocole pour la gestion d'un réseau de capteurs sans fil pour la surveillance de la rupture de la chaîne du froid. La réalisation d'un tel projet est bénéfique à la santé des consommateurs et aux acteurs de la chaîne du froid (producteurs, transporteurs,...).

Notre contribution consiste à sécuriser le protocole *Placide* [5], conçu pour la gestion du réseau. Cette solution doit être un compromis entre simplicité et fiablité. Pour cela, nous avons intégré notre solution à la solution Placide (dans sa phase d'initialisation et en régime permanent),le protocole de gestion du réseau, en limitant le nombre de messages spécifiques pour la sécurité et en minimisant les ressources consommées (mémoire, processeur).

La sécurité dans les réseaux de capteurs sans fil (RCSF) est différente des autres solutions proposées pour les réseaux filaires ou sans fil classiques. Ceci est dû au fait que les capteurs sont des composants miniatures ayant des ressources limitées (en calcul, mémoire et énergie). Les réseaux de capteurs sans fil ne disposent généralement pas d'entités de confiance présente en permanence dans le réseau.

Ce document est organisé comme suit. Dans le premier chapitre, nous définissons ce qu'est un réseau de capteurs avec ses applications, les technologies de communica-

tions utilisées et les systèmes d'exploitation de ces composants. Une présentation de la problématique de sécurité dans ces réseaux fait l'objet du deuxième chapitre, avec un survol sur quelques solutions de sécurité proposées dans la littérature. Le troisième chapitre explique les détails du projet CAPTEURS (protocole Placide et notre solution de sécurité). Le quatrième chapitre expose les résultats d'implémentation et de tests de notre solution. Enfin, nous clôturons par une conclusion et des perspectives.

# 1 Les réseaux de capteurs

## 1.1 Introduction

L'utilisation des réseaux de capteurs est passé de l'usage exclusif par l'industrie, à différents usages. Le développement des technologies de communications sans fil et des domaines de la micro-électronique a permis de produire des composants miniature et à prix abordable. Ceci a facilité la mise en place de tels réseaux [9, 10]. De nombreuses applications sont apparues dont on peut citer essentiellement [11, 12, 13, 14] :

**Militaire :** On utilise, souvent, les réseaux de capteurs pour la surveillance des frontières, la surveillance des mouvements des ennemis lors d'une guerre. En général, on peut placer des capteurs qui détectent les personnes, les véhicules, la voix et les mouvements dans un réseau avec une caméra miniature pour capturer des images.

**Environnement :** Les réseaux de capteurs peuvent être utilisés pour surveiller la pollution de l'air, détecter les incendies dans les forêts ou même détecter les secousses sous marines (prévenir les pays sujets aux Tsunamis avant même qu'une catastrophe ne se produise). Ils peuvent aussi surveiller la pollution d'un lac à côté d'une usine chimique par exemple.

**Bâtiments :** Les réseaux de capteurs sans fil peuvent être utilisés dans les grands bâtiments ou usines pour surveiller les changements de température. Des capteurs de température sont déployés dans tous le bâtiment. De plus, les capteurs peuvent être utilisés pour surveiller les vibrations des bâtiments ou des barrages.

**Santé :** Les réseaux de capteurs sans fil peuvent être utilisés pour détecter et analyser des paramètres sur la santé d'un malade : pression artérielle, température, battements du cœur (voir figure 1.1). De cette façon, les médecins peuvent diagnostiquer un malade à distance et lui prescrive un traitement en cas d'urgence.

On désigne par le mot *capteur*, généralement dans ce document, une plateforme de test composée de : micro-contrôleur, puce radio, capteur de température, capteur d'humidité,... Dans la figure 1.2 extraite du document [2], on trouve un résumé sur l'évolution des capteurs :

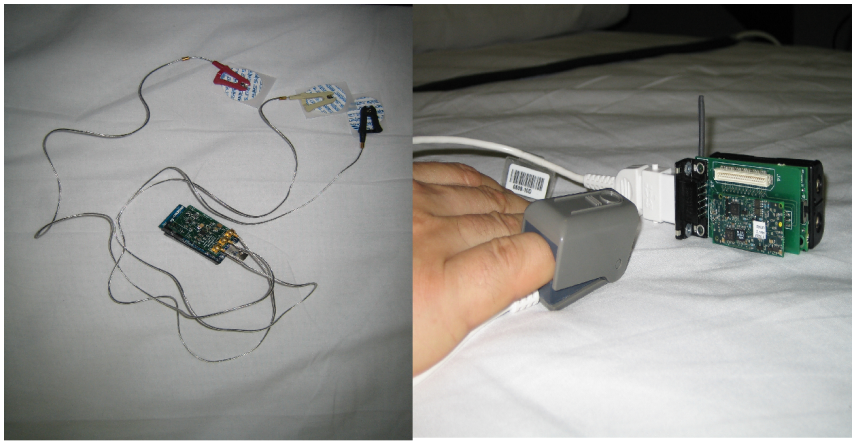


FIG. 1.1 – Capteur pour applications médicale [1]

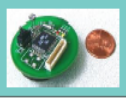
| Mote Type<br>Year | *WeC*<br>1998 | *René*<br>1999 | *René 2*<br>2000 | *Dot*<br>2000 | *Mica*<br>2001 | *Mica2Dot*<br>2002 | *Mica 2*<br>2002 | *Telos*<br>2004 |
|---|---|---|---|---|---|---|---|---|
| |  |  | |  |  |  |  |  |
| Microcontroller | | | | | | | | |
| Type | AT90LS8535 | | ATmega163 | | ATmega128 | | | TI MSP430 |
| Program memory (KB) | 8 | | 16 | | 128 | | | 60 |
| RAM (KB) | 0.5 | | 1 | | 4 | | | 2 |
| Active Power (mW) | 15 | | 15 | | 8 | | 33 | 3 |
| Sleep Power ($\mu$W) | 45 | | 45 | | 75 | | 75 | 6 |
| Wakeup Time ($\mu$s) | 1000 | | 36 | | 180 | | 180 | 6 |
| Nonvolatile storage | | | | | | | | |
| Chip | 24LC256 | | | | AT45DB041B | | | ST M24M01S |
| Connection type | $I^2C$ | | | | SPI | | | $I^2C$ |
| Size (KB) | 32 | | | | 512 | | | 128 |
| Communication | | | | | | | | |
| Radio | TR1000 | | | | TR1000 | CC1000 | | CC2420 |
| Data rate (kbps) | 10 | | | | 40 | 38.4 | | 250 |
| Modulation type | OOK | | | | ASK | FSK | | O-QPSK |
| Receive Power (mW) | 9 | | | | 12 | 29 | | 38 |
| Transmit Power at 0dBm (mW) | 36 | | | | 36 | 42 | | 35 |
| Power Consumption | | | | | | | | |
| Minimum Operation (V) | 2.7 | | 2.7 | | 2.7 | | | 1.8 |
| Total Active Power (mW) | 24 | | | | 27 | 44 | 89 | 41 |
| Programming and Sensor Interface | | | | | | | | |
| Expansion | none | 51-pin | 51-pin | none | 51-pin | 19-pin | 51-pin | 10-pin |
| Communication | IEEE 1284 (programming) and RS232 (requires additional hardware) | | | | | | | USB |
| Integrated Sensors | no | no | no | yes | no | no | no | yes |

FIG. 1.2 – Évolution de capteurs [2]

## 1.2 La technologie sans fil

Comme nous l'avons vu ci-dessus, les réseaux de capteurs ont évolué. Initialement, de simples réseaux filaires, ce sont devenus des réseaux sans fil. Cette migration a été accompagnée par la question suivante : quelle technologie de communication sans fil choisir ?

Les premières puces radio dédiées ont intégré des technologies de communication propriétaires, comme la puce CC1000 de Texas instruments [15] utilisée dans la plateforme Mica2. Ensuite, sont apparues des puces plus performantes utilisant le standard IEEE 802.15.4 [16, 17]. Les travaux de recherche actuels essaient d'intégrer les technologies Bluetooth [18, 19, 20] et/ou RFID [21, 22, 13, 14] dans les réseaux de capteurs sans fil. Les RFID peuvent représenter une technologie complémentaire de 802.15.4 et des réseaux de capteurs sans fil en général. En utilisant la plateforme EPC [23], les RCSF peuvent participer à la mise en œuvre de l'internet des objets (*internet of things*). Dans ce qui suit, on parlera de la technologie 802.15.4, tout en la comparant avec Bluetooth pour comprendre la raison qui a conduit au choix presque exclusif de la première technologie dans l'industrie des capteurs.

### La norme IEEE 802.15.4

La norme appartient à la classe des réseaux personnels sans fil à bas débit (LR-WPAN) optimisés pour les applications à bas débit (jusqu'à 250 kbit/s) avec simple ou sans exigence de qualité de service. Elle utilise de petits paquets et peut être utilisée dans des réseaux de grande densité ($\leq$ 65000 nœuds). Les puces radio, implémentant cette norme, consomme moins d'énergie que celles de Bluetooth. Ces puces radio sont de très faible coût, proche de 1$ contre 10$ pour Bluetooth. Cette norme ne définit que la couche physique et la couche MAC (Media Access Control). Les exigences matérielles pour cette norme sont très minimes : un micro-contrôleur de 8 bits, 4 MHz de fréquence, 32 ko de ROM et 8 ko de RAM.

Comme toutes les technologies sans fil, le média est partagé entre tous les nœuds. D'où la nécessité d'utiliser un algorithme pour organiser l'accès et minimiser les collisions. L'algorithme choisi est le CSMA-CA.

La norme IEEE 802.15.4 a d'excellente performances dans un environnement à faible rapport signal à bruit (SNR : signal-to-noise ratio) comparé à Bluetooth (802.15.1) et au 802.11b (voir figure 1.3).

De plus, cette norme définit trois mécanismes de sécurité au niveau MAC qui se basent sur une clé de 128 bits et l'algorithme de chiffrement par bloc AES-128 :

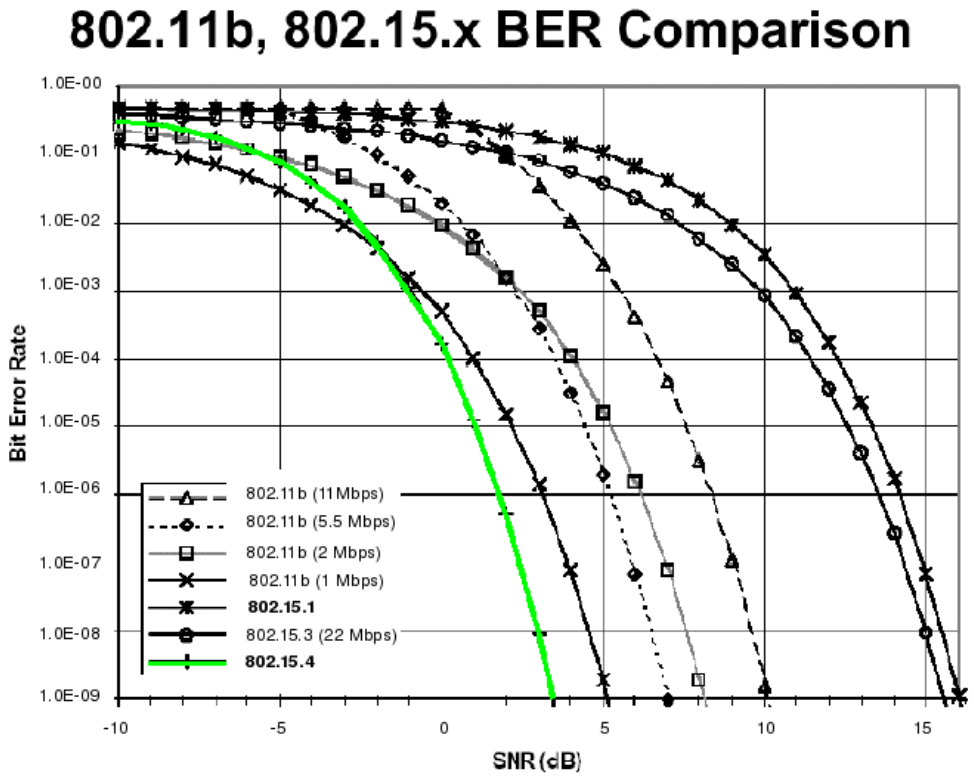



FIG. 1.3 – Comparaison entre 802.15.x et 802.11b [3]

- CTR (CounTeR mode) : un mode de chiffrement des paquets basé sur un compteur

- CBC-MAC (Cipher Bloc Chaining Message Authentication Code) : ajoute au paquet un champ de vérification d'intégrité et d'authenticité.
- CCM : combine les deux modes de sécurité CTR et CBC-MAC.

## 1.3 Les systèmes d'exploitation pour les réseaux de capteurs

### 1.3.1 TinyOS

TinyOS est un système d'exploitation open source conçu pour les réseaux de capteurs sans fil. Il est basé sur une architecture orientée composants qui favorise l'implémentation et l'innovation rapide. De plus, il génère un noyau de petite taille (quelques *ko*), comme l'exigent les contraintes de mémoire imposées par les réseaux de capteurs [4].

La première implémentation de TinyOS a été réalisée à l'université de Berkeley en 1999 [24]. La version 1.0 est sortie en septembre 2002. La version la plus récente de ce système est la 2.0.2 qui a été disponible en juillet 2007. Ce système d'exploitation est développé avec le language NesC. Le développement et la maintenance de ce système est


FIG. 1.4 – Logo TinyOS [4]

maintenant sous la responsabilité d'un consortium international, *TinyOS Alliance*.

Dans ce qui suit, on détaillera les principales caractéristiques du système d'exploitation TinyOS.

**Événementiel (Event-driven) :** TinyOS est un système d'exploitation événementiel. Son fonctionnement se base sur le traitement des événements déclenchés. Ainsi, l'activation de tâches, leur interruption ou encore la mise en veille du capteur s'effectue à l'apparition d'événements, ceux-ci ayant la plus forte priorité [10].

**Plateformes/Abstraction matérielles :** L'un des importants avantages de TinyOS est sa compatibilité avec au moins 9 plateformes matérielles, et la simplicité d'ajouter ou de modifier des plateformes. TinyOS suit une abstraction hiérarchique du matériel en trois couches [25].

**Ordonnanceur :** Les tâches en TinyOS ne sont pas préemptives, c-à-d une tâche ne peut pas interrompre l'exécution d'une autre. Mais, une interruption peut interrompre l'exécution d'une tâche. Au contraire de TinyOS 1.x, où la file de l'ordonnanceur est de taille limitée et dans laquelle une tâche peut être présente plusieurs fois, dans TinyOS 2.x, chaque tâche a sa place dans la file et ne peut y être présente qu'une seule fois [26]. L'ordonnanceur exécute les tâches une par une, et lorsque la file devient vide, il met automatiquement le microcontrôleur en veille.

**Temporisateurs :** TinyOS 2.x met à disposition une grande collection de temporisateurs de haute précision (de fréquences : 1000 KHZ, 32 KHz et 1 Hz) et de différents types (compteur, alarme, temps local) [27].

**Communication :** TinyOS n'implémente pas la pile protocolaire OSI ou TCP/IP. Le SE implémente une pile protocolaire différente. Avec le Tmote Sky [28], on utilise la technologie 802.15.4 pour les deux couches inférieures [29, 30]. Pour la couche réseau (le routage), le SE implémente les deux fonctions de routage dissémination et collecte [31].

**Gestion d'énergie :** TinyOS est conçu pour gérer au mieux la consommation énergétique : il met le nœud en veille lorsqu'il n'y a pas de tâches à exécuter, et permet ainsi la désactivation du périphérique radio ou sa mise en écoute seule et basse consommation énergitique (*low power listening*). TinyOS génère aussi un code de petite taille, ce qui diminue l'énergie utilisée pour mémoriser les données dans la RAM...

**Language *nesC* :** Nesc est le language de programmation de TinyOS et des applications. NesC est le language C avec quelques extensions pour les composants et la gestion de concurrence entre tâches et événements [32].

- **séparation entre construction et composition :** Les programmes sont construits par des composants, qui sont assemblés (*wired*) pour former le programme. Les composants définissent deux parties : une pour spécifier les noms d'interfaces et une autre pour les implémenter. De plus, dans la partie implémentation, on peut définir des tâches qui sont concurrentes entre elles. On peut contrôler ou inter-agir avec un composant à partir de ses interfaces.
- Le comportement du composant est spécifier par un ensemble d'interfaces. L'interface peut être utilisée ou produite par un composant. Les interfaces produites sont prévues pour fournir les fonctionnalités du composant à ses utilisateurs, tandis que les interfaces utilisées sont nécessaire au composant pour qu'il puisse réaliser ses fonctionnalités.
- **Les interfaces sont bidirectionnelles :** L'interface spécifie un ensemble de fonctions à implémenter par son producteur (les commandes) et un autre ensemble à implémenter par son utilisateur (les événements). Par exemple l'interface d'envoi d'un message dans *TinyOS* définit une commande *envoi* (*send*), qui est lente car le périphérique radio est plus lent que le processeur. Alors l'utilisateur de l'interface appelle la commande envoi et lorsque l'envoi est accompli, un événement spécifique (*sendDone*) est déclenché. Son traitement doit être implémenté dans le composant utilisateur de l'interface.
- Les composants sont statiquement liés les uns aux autres par l'intermédiaire de leurs interfaces.
- NesC est basé sur le modèle de concurrence d'exécution jusqu'à la fin (*run-to-completion*) pour les tâches. Les traitements des interruptions (*interrupt handlers*) interrompent l'exécution des tâches et s'interrompent entre elles. Alors, une interruption peut interrompre l'exécution d'une tâche ou le traitement d'une autre interruption.

**TOSSIM (TinyOS SIMulator) [33] :** est un un simulateur pour la plupart des applications de TinyOS. Il est maintenant compatible avec les applications qui peuvent fonctionner sur la plateforme MicaZ. Pour pouvoir utiliser la librairie TOSSIM, on doit

écrire un programme qui configure la simulation et l'exécute en utilisant C++ ou Python. Le simulateur avec la version 2.x de TinyOS est devenu plus performant : il permet de définir les coordonnées des nœuds, la qualité du lien radio (bruit, gain) et plusieurs autres techniques de déboguage et de tests (message de déboguage, accès aux variables de l'application, et injection de paquets).

### 1.3.2 Contiki

Contiki [34] est un système d'exploitation portable, open source et multitâche pour les réseaux de capteurs. Il supporte beaucoups de plateformes et il a un environnement de simulation *netsim*.

Ce système a été développé par l'équipe des systèmes embarqués dans l'institut des sciences informatique suédois.

Contiki supporte le multitâche et il implémente la pile protocolaire TCP/IP. Il ne consomme pas beaucoups de mémoire : quelques *ko* de code et quelques centaines d'octets dans la RAM.
Au contaire de TinyOS qui se base sur la notion d'événements, celui-ci se base sur le multitâche et l'édition statique des liens [35].

## 1.4 Conclusion

Dans ce chapitre, on a découvert les réseaux de capteurs sans fil, et leurs applications. La technologie de communication la plus adaptée et utilisée dans ces réseaux est la 802.15.4. On a choisit le système d'exploitation TinyOS pour implémenter notre application car il a une bonne documentation et supporté par la communauté scientifique (plus que 500 labos de recherche ont choisi TinyOS).

Dans le chapitre suivant, on abordera la sécurité dans les réseaux de capteurs et l'état de l'art des solutions proposées.

# 2 La sécurité des réseaux de capteurs

## 2.1 Introduction

Les réseaux de capteurs sans fil peuvent s'apparenter aux réseaux ad hoc sans fil [9], mais ils se différencient par plusieurs limitations, ce qui ne permet pas d'appliquer directement les solutions de sécurité existantes. Pour pouvoir développer des solutions de sécurité adaptées à ces réseaux, il faut bien comprendre leurs contraintes [36].

Dans ce chapitre, on détaillera au début les obstacles de sécurité liés aux réseaux de capteurs sans fil. Ensuite, on énumèrera les attaques possibles dans ces réseaux. Enfin, on s'intéressera à l'état de l'art des solutions de sécurité proposées dans la littérature, et qui peuvent s'adapter à la nature de notre réseau.

## 2.2 Les obstacles de sécurité liés aux réseaux de capteurs

### 2.2.1 Des ressources limitées

L'utilisation des algorithmes de sécurité nécessitent des ressources de mémoires pour la mémorisation du code et des données, des ressources en énergie et en calcul [8].

| Plateforme | micro-contrôleur | RAM | Mémoire programme | Puce radio |
|---|---|---|---|---|
| *Mica2* | ATMega128 (7,3728$MHz$) | 4 $ko$ | 128 $ko$ | CC1000 (868-916$MHz$) |
| *Tmote Sky* | MSP430 (8$MHz$) | 10 $ko$ | 48 $ko$ | CC2420 (2400-2483$MHz$) |

TAB. 2.1 – Comparaison des caractéristiques des deux plateformes Tmote Sky et Mica2 [7]

– **mémoire et espace de stockage limités :** Le capteur est un composant miniature avec un espace mémoire et de stockage limité, et avec une faible vitesse de calcul.

Dans le tableau 2.1, on a une comparaison entre les deux capteurs les plus connus [7]. Le capteur Mica2 a un processeur de fréquence 7,3728 MHz et une mémoire pour le programme de 128 ko. Sur le même capteur, on doit installer le code du système d'exploitation et de l'application. Donc le code de sécurité et les données relatives doivent être très petites.

– **Limitation en énergie :** C'est la contrainte la plus forte, puisque généralement les capteurs sont déployés à des endroits inaccessibles ou d'accès difficile, donc on ne peut pas changer les batteries ou les recharger (c'est cher à utiliser dans un capteur). Alors la ressource d'énergie embarquée avec les capteurs doit être conservée pour étendre leur vie et par la suite celle du réseau entier. L'ajout des fonctions de sécurité a un effet à prendre en considération sur la consommation des ressources, de temps processeur et par la suite sur la consommation en énergie [8].
Le tableau 2.2 présente le nombre de cycles CPU et le temps nécessaire pour chiffrer un message de longueur 29 octets par différents algorithmes de chiffrement. Le tableau 2.3 présente les mêmes paramètres mais pour la génération du MAC, tous les algorithmes opèrent en mode CBC-MAC. Ces tests sont faits sur la plateforme mica2.

| Algorithme | Temps (ms) | Cycles CPU | Energie ($\mu$j) |
|---|---|---|---|
| *SkipJack* | 2.16 | 15,925.2 | 51.84 |
| *RC5* | 1.50 | 11,059.2 | 36.00 |
| *RC6* | 10.78 | 79,478.7 | 258.72 |
| *TEA* | 2.56 | 18,874.4 | 61.44 |
| *DES* | 608.00 | 4,482,662,4 | 14,592.00 |

Tab. 2.2 – L'impact du chiffrement de 29 octets sur la consommation CPU [8]

| Algorithme | Temps (ms) | Cycles CPU | Energie ($\mu$j) |
|---|---|---|---|
| *SkipJack* | 2.99 | 22,044.6 | 71.76 |
| *RC5* | 2.08 | 15,335.4 | 49.92 |
| *RC6* | 15.84 | 116,785.2 | 380.16 |
| *TEA* | 5.07 | 37,380.1 | 121.68 |
| *DES* | 1,208.00 | 8,906,342.4 | 28,992.00 |

Tab. 2.3 – La consommation CPU lors du calcul du MAC d'un paquet de 29 octets [8]

RC5 a donné les meilleurs résultats dans les deux tests. Par exemple, dans les deux tests réalisés, il apparaît plus de sept fois plus performant qu'un RC6.
En général, la consommation en énergie ajoutée par les fonctions de sécurité est due essentiellement aux opérations de chiffrement/déchiffrement, la signature des données, la vérification de la signature, l'émission des informations de sécurité (le vecteur d'initialisation, les clés, la signature), la mémorisation des clés ...

### 2.2.2 Communication non fiable

Cette caractéristique est héritée des réseaux sans fil. Les données sont transmises dans l'air, donc chaque capteur qui se trouve dans le rayon de couverture peut écouter les messages échangés. L'application d'un bruit sur le canal peut rendre les capteurs incapables de transmettre les messages vu que le média peut apparaître comme occupé en permanence.

### 2.2.3 Les risques inattendus

Selon l'application du réseau de capteurs, les capteurs sont sans surveillance ou surveillés après une longue période. Les capteurs sont sans surveillance lorsqu'ils sont, par exemple, déployés derrière les lignes de l'ennemi. Les mises en garde au capteurs sans surveillance sont :

- **exposition aux attaques physiques :** Le capteur est généralement déployé dans un environnement ouvert aux ennemis, et peut faire face à des conditions climatiques difficiles.
- **gestion à distance :** La gestion à distance du réseau rend la détection d'une attaque physique (compromission de capteur) et la maintenance des capteurs (rechange ou recharge de batterie) impossible.
- **pas de station de base :** Le réseau de capteurs doit être conçu pour être un réseau distribué sans point de gestion central. Mais s'il y a une erreur de conception, l'organisation du réseau peut devenir difficile, inefficace et fragile.

## 2.3 Les attaques dans les réseaux de capteurs

Les attaques de sécurité possibles dans les réseaux de capteurs sont [37] :

#### Écoute passive du réseau

L'attaquant, qui dispose d'un équipement puissant (vitesse de calcul, espace de stockage, ressource en énergie, etc), collecte les informations échangées dans le réseaux de capteurs si elles ne sont pas chiffrées.

#### Compromission du nœud

En compromettant un nœud, l'attaquant peut récupérer les informations incluses : programme, clés cryptographiques. [38] a démontré que la compromission d'un capteur de type mica2 peut se faire en moins d'une minute.

### Injection de nœuds malveillants

L'attaquant peut ajouter dans le réseau des nœuds malveillants pour injecter des données falsifiées dans le réseau. D'habitude ces nœuds seront plus robuste pour attirer les messages des autres nœuds, car dans les réseaux de capteurs sans fil, le choix du meilleur chemin par les algorithmes de routage se base sur plusieurs paramètres ; parmi lesquelles l'énergie et la puissance de calcul des nœuds.

### Le mauvais fonctionnement d'un nœud

Le mauvais fonctionnement d'un nœud peut générer des données inexactes qui peuvent mettre en péril l'intégrité du réseau, et surtout quand ce nœud joue un rôle important dans l'aggrégation des données. Par exemple un chef de la grappe (*cluster*).

### La panne d'un nœud

La panne d'un nœud peut affecter le fonctionnement du réseau, surtout quand ce nœud est un chef de grappe. Alors le protocole de routage doit être robuste pour annuler l'effet d'une telle panne en construisant des chemins de routage alternatifs.

### La corruption de message

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

### L'analyse du trafic

Cette analyse peut se faire même si les messages sont chiffrés. En analysant les modes de communications et les activités des nœuds, l'attaquant peut déduire des informations sur l'organisation du réseau, par exemple sur les chefs de grappes.

### Les boucles de routage

Ce type d'attaque vise les informations échangées entre les nœuds. Le re-jeu ou la modification des informations de routage par un attaquant génère de faux messages d'erreurs. Les boucles de routage attirent ou repoussent le trafic dans le réseau.

### Transmission sélective

Supposons que tous les nœuds participent à la propagation des messages dans le réseau. Dans cette attaque, le nœud malveillant supprime quelques messages au lieu de les transmettre. L'efficacité de cette attaque dépend de deux facteurs. Le premier est la place du nœud malveillant, plus il est proche de la station de base plus il reçoit de messages. Deuxièmement est le pourcentage des messages qu'il supprime.

### Trou noir (sinkhole)

Le nœud malveillant se place à un endroit stratégique (proche de la station de base par exemple) et supprime tous les messages qu'il doit retransmettre. Cette attaque est grave lorsqu'il n'y a qu'une seule station de base dans le réseau.

### Usurpation d'identités (sybil attacks)

Dans ce type d'attaques, le nœud malveillant prend un grand nombre d'identités qui peuvent être volés ou imaginaires. Cette attaque peut être utilisée contre les protocoles de routage.

### Réplication de nœud (clonage)[39]

L'attaquant récupère un nœud. Il en extrait les secrets et il les transfère à des nœuds génériques. Enfin, il déploie ces nœuds. Alors il peut injecter de fausses données et supprimer des données légitimes.

### Trou de ver (wormhole)

L'intrus capture un message et, en utilisant un canal de faible latence, le retransmet vers un lieu distant dans le réseau. Le canal ainsi créé fait transiter un message à un endroit du RCSF auquel il ne devrait normalement pas arriver, ou sinon avec une plus grande latence. Cette attaque a une influence notable sur le routage dans le réseau [9].

### Attaque par inondation avec le message HELLO

Le nœud malveillant diffuse un message Hello dans le réseau avec une grande énergie d'émission, tout en prétendant qu'il provient de la station de base. Les receveurs de ce message essayeront de transmettre tous leurs messages à travers le nœud malveillant, car ils pensent qu'il appartient au plus proche chemin vers la station de base. Ce qui gaspillera l'énergie des nœuds en transmettant des messages inutilement.

### Les attaques de déni de service par interférence [9]

Le nœud malveillant inonde avec du bruit les fréquences radio utilisées par le réseaux de manère à empêcher les transmissions et/ou les réceptions de messages. Ce type d'attaques peut affecter tout ou partie du réseau selon la portée radio de l'intrus. Dans ce cas-là, l'intention est de provoquer un déni de service.

## 2.4 Les solutions de sécurité pour les réseaux de capteurs

Les exigences de sécurité, dans les réseaux de capteurs, dépendent de la nature de l'application. Les applications militaires sont très exigentes en sécurité, voire non tolérante à ce niveau ; car le réseau est un épée à double tranches, et peut devenir une arme ennemie. Mais dans des applications de surveillance de l'environnement, par exemple, la sécurité n'est pas très exigente. L'application d'un simple mécanisme demeure suffisant pour protéger le réseau des attaques primaires.

L'application des solutions de sécurité doit être accompagnée d'un mécanisme de gestion des clés. Notre réseau, comme on le verra dans le chapitre 3, suppose la non présence d'une station de base ; donc on s'interessera plus aux solutions adaptées à cette contrainte. On verra, d'abord, les mécanismes de gestion des clés et ensuite les deux plus importantes solutions de sécurité proposées dans la littérature.

### 2.4.1 Les protocoles de distribution des clés

Les protocoles de distribution de clés traditionnels ne sont pas adaptés aux réseaux de capteurs sans fil, qui ont des ressources limitées et aussi pour le manque d'informations sur la topologie du réseau et sur les voisins. Plusieurs adaptations ont été proposées dans la littérature à ces protocoles. Les trois classes essentielles sont :

1. **Clé globale :** est une clé symétrique partagée par tous les nœuds, ce qui permet de minimiser l'espace de stockage utilisé par la sécurité et de résister aux attaques de déni de service, car le calcul de MACs est rapide. Mais cette solution ne tolère pas la compromission de nœuds. Un nœud compromis par un attaquant signifie la destruction totale du mécanisme de sécurité.
2. **Clé de paire :** est une clé unique partagée entre deux nœuds. Cette solution résiste aux compromissions de nœuds et assure l'authentification entre les paires (*node-to-node authentication*). Mais cette solution consomme excessivement de mémoire. Si on a N nœuds dans le réseau, chaque nœud doit stocker (N-1) clés ou on doit utiliser un protocole de pré-distribution de clés spécifique comme celui de Blom [40].
   Blom a proposé une méthode de pré-distribution de clés qui permet à chaque paire de nœuds d'établir une clé secrète partagée. Le protocole fonctionne comme suit :
   D'abord on choisit une matrice de G de taille $(\lambda+1)\times$N appartenant au groupe fini de GF(q) (*Galoi Field*) (q est un nombre premier et essai grand), et N est le nombre de nœuds dans le réseau. G est une matrice publique pouvant être connue par les attaquants et contient au moins $(\lambda + 1)$ colonnes linéairement indépendantes. Ensuite, on construit une matrice D symétrique aléatoire et privée de taille $(\lambda+1)\times(\lambda+1)$,

et on calcule la matrice A = $(D.G)^T$ de taille N×($\lambda$+1), avec $(D.G)^T$ qui est la transposée de (D.G). Alors, on peut avoir :

$$(A.G)^T = G^T.A^T = G^T.D.G = G^T.D^T.G = (D.G)^T.G = A.G$$

Alors A.G est une matrice symétrique. Chaque nœud du réseau prend une k$^e$ ligne aléatoire de la matrice A et la k$^e$ colonne de G. Si les nœuds i et j veulent communiquer, on suppose que le nœud i stocke la i$^e$ ligne de A, et le nœud j stocke la j$^e$ ligne de A. Comme G est publique, i envoie la i$^e$ colonne de A à j, et j envoie la j$^e$ colonne de A à i. Les deux nœuds peuvent calculer les deux éléments de la matrice symétrique (A.G) $k_{ij}$ et $k_{ji}$, la clé finale peut se baser sur $k_{ij} = k_{ji}$. La figure 2.1 illustre la génération des valeurs $k_{ij}$ et $k_{ji}$. La sécurité de [40] a été prouvée si on n'a pas plus que $\lambda$ nœuds compromis.

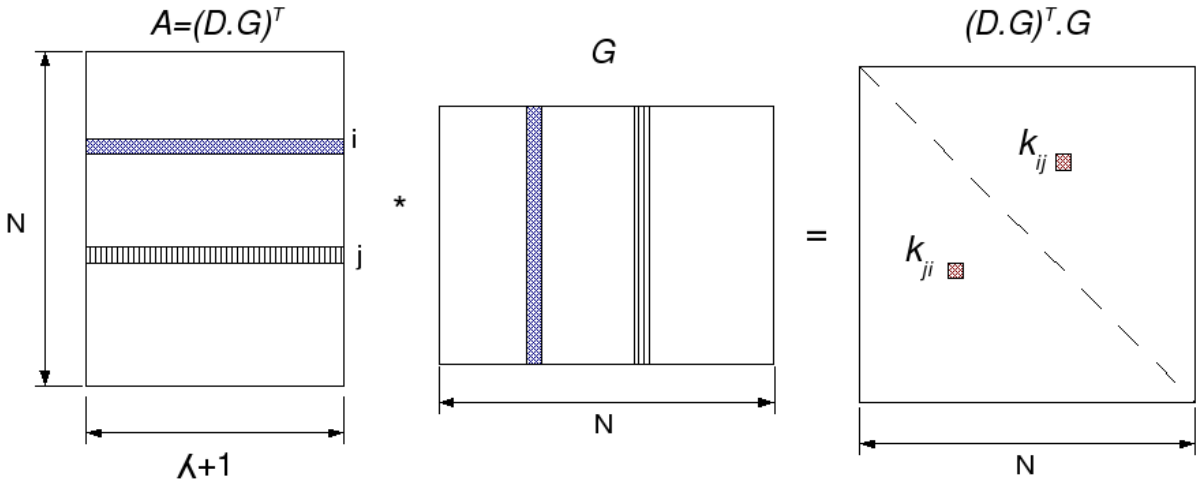



FIG. 2.1 – La génération des valeurs $k_{ij}$ et $k_{ji}$ dans la méthode de Blom

3. **Clé publique :** Ce mécanisme de gestion de clés est le plus utilisé dans les réseaux traditionnels mais il est difficilement envisageable dans les réseaux de capteurs sans fil, car il est gourmand en terme de ressources et temps de calcul, et ce, malgré les travaux réalisées dans la littérature [41, 42, 43, 44].

### 2.4.2 Les protocoles de sécurité

Plusieurs propositions ont vu le jour, mais jusqu'à nos jours il n'y a pas une solution qui fais le compromis attendu par les utilisateurs entre sécurité, consommation en énergie et temps de calcul. C'est pourquoi ce domaine est encore en exploration par les chercheurs. Dans ce qui suit, on verra de plus près les deux solutions *TinySec* et *SNEP*.

#### 2.4.2.1 TinySec

La solution de sécurité qui a été proposée dans TinyOS 1.x est TinySec [45]. Elle se présente comme une couche de sécurité au niveau liaison de données. TinySec utilise une clé de groupe commune, initialement chargée dans tous les capteurs du réseau avant leur déploiement. En fait, les capteurs sont divisés en des groupes, et chaque groupe partage une clé. Seulement les capteurs appartenant au même groupe peuvent communiquer. Dans les applications, on peut activer ou désactiver la sécurité, et en activant la sécurité, on peut soit utiliser l'authentification de paquets uniquement, soit utiliser le chiffrement et l'authentification des paquets. TinySec implémente les algorithmes de chiffrement *RC5* et *Skipjack* et utilise CBC-MAC comme algorithme de génération de MAC (qui utilise l'algorithme choisi par l'utilisateur RC5 ou Skipjack).

TinySec peut garantir la confidentialité des paquets et leur authentification (Il ne permet pas de s'assurer de l'identité de l'émetteur mais il ne permet qu'aux messages des nœuds qui possèdent la clé de groupe d'être acceptés)

**Les avantages de TinySec sont multiples :**

- TinySec est une solution intégrée avec la version 1.x de TinyOS
- Son utilisation est facile et transparente à l'utilisateur
- Il permet d'activer ou de désactiver la sécurité pendant l'exécution du programme

**Les inconvénients de TinySec sont les suivants :**

- TinySec n'a pas été porté à la version 2.x de TinyOS
- Zéro tolérance aux compromissions : il suffit qu'un seul nœud soit compromis pour que la clé de groupe soit divulguée, et qu'un attaquant puisse usurper l'identité de tout nœud du réseau, et puisse écouter toutes les communications dans le réseau [9].
- Si les capteurs sont gérés par plusieurs organisations (voir 3.3.1), il appartiennent à plusieurs groupes. Or TinySec ne permet que la communication entre les capteurs appartenant au même groupe. En activant la sécurité et en déployant les capteurs dans une seule zone, ils ne vont pas pouvoir communiquer puisqu'ils ont des clés différentes.
- TinySec suppose tous les équipements identiques (partageant la même clé). TinySec ne fait pas la différence entre un capteur et un équipement externe (ex. PDA).
- Ne protège pas contre les attaques de jamming, re-jeu de paquets et déni de service.

#### 2.4.2.2 SNEP : Sensor Network Encryption Protocol

SNEP [46] est un protocole de sécurité pour les réseaux de capteurs qui utilise l'algorithme de chiffrement CBC-DES (Cipher Block Chaining Data Encryption Standard) et offre beaucoup d'avantages.

Le message envoyé de A à B avec ce protocole a la forme suivante :

$$A \rightarrow B : D < K_c, C >, MAC(K_{mac}, C|D < K_c, C >)$$

Avec :

$K_c$ **et** $K_{mac}$ **:** sont respectivement la clé utilisée pour le chiffrement et la clé pour la génération du MAC (Message Authentication Code) et elles sont dérivées d'une clé principale $K$

**C :** Compteur

$D < K_c, C >$ **:** les données sont chiffrées en fonction de $K_c$ et de C

**| :** Concatéation

$MAC(K_{mac}, C|D < K_c, C >$ **:** Le champ du digest calculé en fonction de $K_{mac}$ et de $C$ sur les données chiffrées.

Ce protocole assure :

- La confidentialité grâce au chiffrement de données en utilisant la fonction de chiffrement CBC-DES.
- **La sécurité sémantique :** en utilisant un compteur qui est partagé entre les deux nœuds (source et destination) et qui sera incrémenté après chaque message. Il n'est pas transmis avec le message. Il est utilisé aussi pour générer le vecteur d'initialisation nécessaire à l'algorithme de cryptage. Alors deux données identiques n'auront pas le même résultat après chiffrement grâce au compteur $C$.
- **L'authentification des données :** Si le MAC est vérifié alors le récepteur peut s'assurer que le message provient de la bonne source.
- **Protection contre le re-jeu de paquets :** L'utilisation du compteur pour générer la MAC permet de ne pas re-jouer les anciens messages. Notons bien si le compteur n'était pas présent lors du calcul de la MAC, l'attaquant peut facilement re-jouer les paquets.
- Le compteur permet de garantir la fraîcheur des messages échangés entre deux nœuds et les messages auront un certain ordre.

## 2.5 Conclusion

Le besoin en teme de sécurité dans les réseaux de capteurs sans fil a été démotré dans ce chapitre. On a vu deux solutions proposées dans la littérature, TinySec et SNEP. Dans le chapitre suivant, on verra en détails l'objet du projet CAPTEURS, ses acteurs, et ses particularités.

# 3 Le projet CAPTEURS

## 3.1 Introduction

Le projet CAPTEURS est un projet d'une durée de trois années supporté par l'ANR (Agence Nationale de Recherche). Il a débuté en décembre 2005 et se termine fin 2008. Le consortium est composé de partenaires industriels et académiques : TELECOM & Management SudParis, CRITT Transport et Logistique, HLP Technologies, ENSEEIHT, INVENTEL/THOMSON.

Le projet CAPTEURS propose l'utilisation des réseaux de capteurs pour la supervision et le contrôle de la chaîne du froid.

La chaîne du froid est l'ensemble des opérations logistiques et domestiques (transport, manutention, stockage) visant à maintenir un ou des produits (généralement alimentaires) à une température basse pour assurer le maintien de sa salubrité.
En cas de rupture de la chaîne du froid, les aliments dégèlent, puis regèlent. Leur consommation peut alors entraîner une intoxication alimentaire dont les effets varient selon la fragilité de l'individu : la diarrhée, la fièvre et même la mort [47].

Pour supervisionner la chaîne, on fixe un capteur à chaque palette, et il doit communiquer avec les autres capteurs pour journaliser l'état de la température dans tout le réseau.

Les variations de la température sont plus critiques pendant les phases de transport et de charge/décharge. On se limitera dans ce projet à ces phases là.
Pour des raisons de tolérances aux pannes, les alarmes de variation de la températures signalées par un capteur doivent être propagée dans tous le réseau. Alors chaque capteur se trouvant dans le camion peut signaler la présence d'une alarme éventuelle. De cette façon, si le capteur, qui a détecté la variation, est tombé en panne, on peut retrouver l'alarme dans un autre capteur. A la fin du transport, un opérateur peut interroger n'importe quel capteur pour savoir l'état des biens lors du transport.

Pour garantir une longévité des capteurs et, par suite, du réseau entier, on doit les mettre le plus longtemps possible en état de veille. Mais ceci rend la communication entre eux au sein du réseau impossible (on ne peut pas garantir qu'ils seront tous éveillés en même temps). Le but du protocole PLACIDE est de garantir cette approche. On détaillera ce protocole dans la section suivante.

Selon la marchandise chargée dans les palettes, les capteurs peuvent être initilisés avec les seuils de température minimale et maximale idéaux. On peut y enregitrer aussi la date limite, le type de marchandise...

Tout au long du transport, les capteurs doivent enregistrer :
- la température associée à la date et l'heure, et on peut l'interroger à tout moment pour donner un rapport sur l'historique des mesures.
- les alarmes générées avec la température, la date et l'heure
- les alarmes reçues des autres capteurs appartenant au même réseau

## 3.2 Placide

Placide [5] est une solution solution protocolaire pour la mise en œuvre optimisée d'un réseau de capteurs sans fil sans infrastructure dans le domaine de la surveillance de la chaîne du froid. Les concepteurs de ce protocole sont partis du constat que la transmission d'un octet est équivalente approximativement à 11000 cycles de calcul [48], c'est pourquoi ils ont pensé à désactiver le composant radio pendant 95% de la vie d'un capteur. La solution exclut l'usage d'une station de base.

Les capteurs sont rangés dans des palettes, qui sont chargées dans le camion. Par un dispositif extérieur (PDA par exemple), on envoie un message spécial qui déclenchera la phase d'initialisation du réseau. Une fois le réseau formé, les échanges entre les capteurs dans le régime permanent permettent de propager les alarmes dans tout le réseau de telle sorte que chaque capteur a connaissance de l'état du transport et du réseau (perte ou ajout de nœuds, ..). Dans ce qui suit, on décrira brièvement le réseau dans le régime permanent et dans la phase d'initialisation.

### 3.2.1 Régime permanent

Après la phase d'initialisation, les capteurs forment un réseau semblable à une liste doublement chaînée (figure 3.1). Ils ont des temps de réveil partagé en deux périodes : une période de réception des messages (de synchronisation, d'alarmes ) du nœud précédent et une autre pour l'émission des messages vers le nœud suivant. Ces périodes sont alignées

de telle sorte que si le nœud précédent est en période d'émission, le nœud suivant est en réception, et dans le cycle suivant, le nœud successeur devient prédécesseur et vice-versa. Un nœud intermédiaire dans le réseau suit la séquence sommeil-réception-émission. Le premier nœud et le dernier suivent une séquence légèrement différente. Par exemple, le premier suit la séquence émission-sommeil-réception.

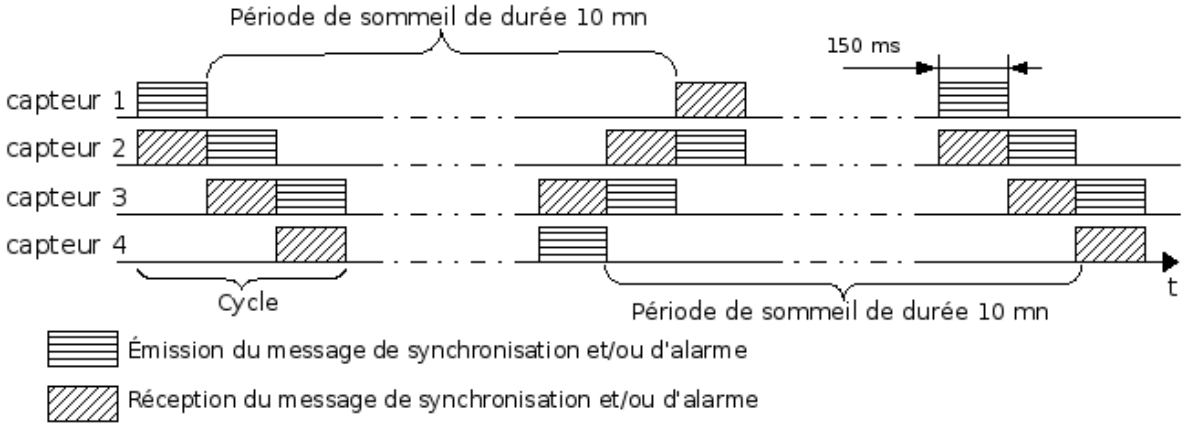



FIG. 3.1 – Sommeil/réveil des capteurs en régime permanent [5]

### 3.2.2 Phase d'initialisation

Cette phase commence lorsque les nœuds reçoivent un message spécial du périphérique externe. Alors chaque capteur arme un temporisateur. Le premier capteur, dont le temporisateur expire (C1 dans figure 3.2 -a-), prend l'identifiant 1 et diffuse un message de synchronisation SYNC. En recevant ce message les autres capteurs arment un nouveau temporisateur pour répondre à ce message. Le capteur dont le temporisateur expire le premier (C2 dans figure 3.2 -b-), répond avec un acquittement ACK au nœud source du message SYNC (ici C1) et envoie ensuite un message SYNC. Et ainsi de suite, jusqu'au dernier capteur qui répond au SYNC reçu par un ACK et envoie un SYNC mais il ne reçoit pas de réponse. Alors, il déduit qu'il est le dernier. Mais il ne s'endort pas pendant sa `première` période de sommeil, il reste en mode écoute pour détecter une éventuelle présence d'une autre chaîne (figure 4.3). S'il la détecte, il procède à une phase de fusion entre les deux chaînes [5].

Comme on peut remarquer, la sécurisation de la phase d'initialisation et des échanges en régime permanent est nécessaire pour protéger le réseau et l'intégrité des échanges.

## 3.3 Sécurisation du réseau

Dans ce qui suit, nous décrivons en détail les spécifications de la solution de sécurité que nous proposons dans le projet CAPTEURS [49]. Vu les contraintes matérielles (énergie

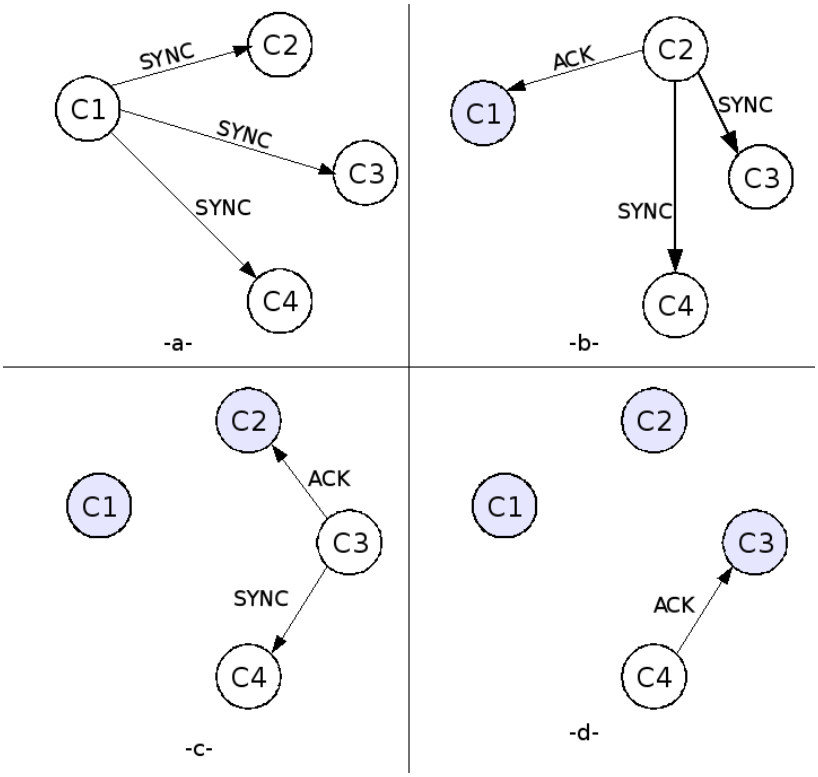



FIG. 3.2 – Phase d'initialisation d'un réseau composé de quatre nœuds [5]

et mémoire) que présentent les capteurs (voir 2.2.1), ainsi que la nature de notre réseau et le nombre important d'acteurs constituant et interagissant avec la chaîne du froid, la solution présentée diffère des solutions classiques de la littérature (voir 2.4.2) qui, souvent, possédent une entité de confiance, utilisent des clés globales partagées ou des clès de paire avec usage excessif de ressources (mémoire, calcul et échange de messages). La tâche principale de ce projet consiste à sécuriser du mieux possible le protocole PLACIDE de supervision de la chaîne du froid , et ceci, en veillant à minimiser le coût en énergie et en espace mémoire.

Dans le projet CAPTEURS, notre réseau de capteurs se différencie des réseaux de capteurs habituellement considérés dans les solutions de sécurité de la littérature sur les aspects suivants :

- Les capteurs sont mobiles et se déplacent de réseau en réseau (de site en site), et le modèle de déplacement est un modèle de déplacement par groupe.
- Les stations de base (SB) ne sont pas utilisées dans notre réseau, et aucune entité de confiance n'existe dans le réseau. Dans les solutions de sécurité classiques, un réseau de capteurs dispose d'une SB qui est une entité de confiance dans le réseau.
- Les capteurs sont gérés par différentes organisations (et fabriqués par différents fabricants).

Ces différences, ajoutées aux contraintes posées par les capteurs, ne nous permettent pas de proposer une solution de sécurité qui résiste contre tous les types d'attaques qu'un réseau de capteurs peut subir. De plus, la solution sera incluse dans le protocole Placide (Placide seul il utilise, avec le code de débogage, plus de : 43840 octets de mémoire programme et 1964 octets de RAM). En effet, une solution complète (qui utilise, par exemple, le mécanisme de cryptographie à clé publique) sera financièrement coûteuse pour les acteurs de la chaîne du froid, car d'un côté, soit il sera nécessaire d'utiliser des palettes équipées avec des capteurs évolués et donc coûteux, soit il faudra utiliser des capteurs bon marché, et dans ce cas, la durée de vie d'un capteur (palette) ne dépassera pas quelques semaines, alors qu'une durée de vie d'une palette classique est de 4 à 5 ans.

### 3.3.1 Composants de la chaîne du froid

La figure 3.3 schématise les composants de la chaîne du froid considérés ainsi que les interactions dans la chaîne. On distingue trois niveaux de composants :

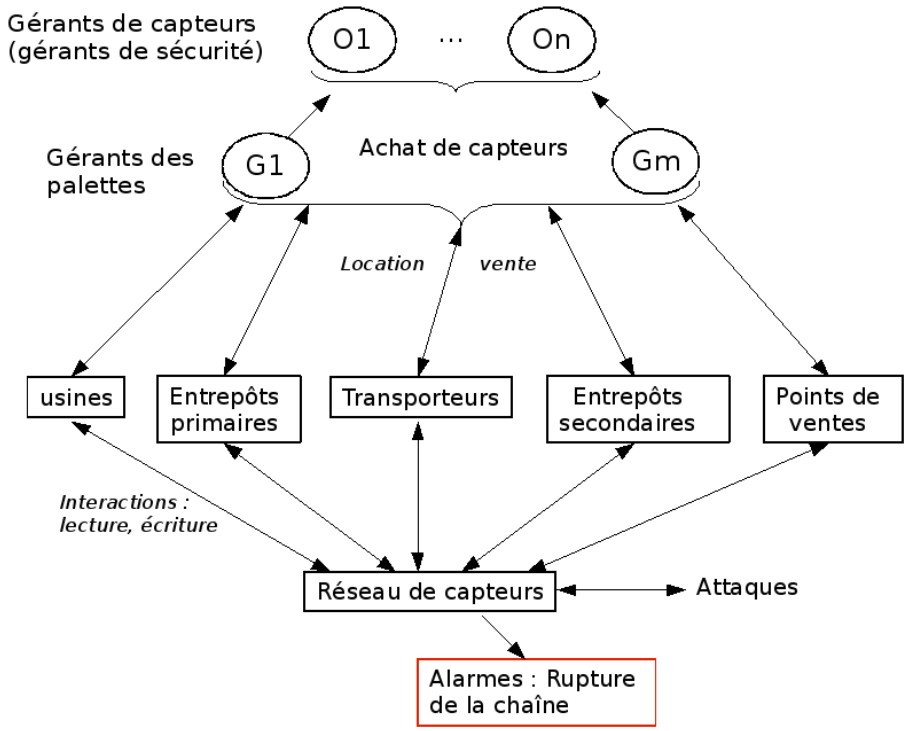



FIG. 3.3 – Composants de la chaîne du froid

- Les gérants de capteurs, qui sont des organisations autonomes responsables de la fabrication des capteurs.
- Les gérants de palettes qui sont responsables de la fabrication des palettes.
- L'intégrateur qui est responsable de l'intégration des capteurs dans les palettes.
- Les acteurs de la chaîne du froid, qui sont les acteurs constituant la chaîne, c'est-à-dire l'usine de fabrication des produits et les points de distribution des produits, en

passant par les transporteurs et les entrepôts.

On peut noter à ce stade que la configuration de la sécurité (c-à-d clés cryptographiques) dans les capteurs peut être effectuée par le gérant de capteurs ou l'intégrateur. Le rôle de l'intégrateur peut être assuré par le gérant de palettes, le gérant de capteurs ou bien par une société indépendante.
Les interactions avec la chaîne du froid représentent toutes les communications externes venant d'équipements de lecture/écriture, ex : PDA. Ces communications représentent des requêtes émises par des équipements vers les capteurs, pour s'informer (lecture) de l'état de la chaîne (s'il y a eu rupture ou pas), ou pour charger les capteurs (écriture) avec de nouveaux paramètres (ex : nouveau seuil de température pour le déclenchement d'alarmes).

Notre chaîne du froid peut être contrôlée à tout moment, et au niveau de n'importe quel acteur de la chaîne.

### 3.3.2 Services de sécurité offerts par notre solution

Comme cité précédemment, la nature de notre réseau de capteurs, ainsi que les contraintes en énergie et espace mémoire des capteurs, ne nous permettent pas de proposer une solution de sécurité complète, englobant tous les services de sécurité.

Une solution de sécurité utilisant la cryptographie à clé publique serait la mieux adaptée, mais nécessiterait des coûts en mémoire, calcul et communication trop élevées pour une intégration au sein de capteurs. Par conséquent, nous avons opté, dans notre solution, pour l'utilisation de la cryptographie symétrique qui offre des coûts réduits en calcul et en stockage, et ceci, malgré les problèmes de gestion de clés (création, distribution, renouvellement et révocation) introduits par la cryptographie symétrique.

Nous optons pour l'utilisation de clés de paire uniques entre chaque couple de capteurs générées après une phase d'initialisation et à partir de clés chargées préalablement dans les capteurs. Ce choix influence bien sûr le panel de services de sécurité possibles à mettre en œuvre, ainsi que le modèle de l'attaquant qui dispose d'un choix plus ou moins grand d'interactions avec le système de protection.

**Les ervices de sécurité proposés sont :**

- **Authentification :** permet de vérifier l'identité de l'émetteur d'un paquet, et son autorisation à participer au réseau. Il permet aussi de vérifier que le paquet envoyé a bien été reçu par un nœud autorisé du réseau. Il n'y a pas d'authentification de bout en bout, c-à-d chaque capteur peut vérifier l'authenticité de la source du

message $u$, sans pouvoir authentifier la vrai source ($v$) du paquet si $u$ était un nœud intermédiaire.

- **Intégrité des données :** permet de s'assurer que les données reçues n'ont pas été modifiées par des nœuds malveillants. Il n'y a pas d'intégrité de bout en bout.
- **Confidentialité :** un besoin optionnel dans le projet, garantissant seulement que seuls les nœuds successeurs ont accès au contenu des messages. Il n'y a pas de confidentialité de bout en bout.
- **Fraîcheur des messages :** permet de guarantir que les données échangées dans le réseaux sont fraîches et ne sont pas re-jouées par un attaquant.
- En se basant sur ces services, on peut aussi garantir une protection partielle de l'agrégation, en garantissant que les données agrégées proviennent de nœuds autorisés du réseau, sans pouvoir s'assurer que ces données sont correctes, ni de s'assurer de l'identité réelle des nœuds émetteurs.
- Protection contre des attaquants externes seulement, ne possédant pas de clés de groupe. Cette protection consiste à empêcher un équipement externe (ex : PDA) non autorisé d'interagir avec le réseau de capteurs en lecture/écriture. Aucune garantie n'est apportée quant à la protection contre des attaques internes provenant de nœuds malveillants, ou contre un attaquant externe qui a pu récupérer les clés secrètes en compromettant un nœud interne du réseau.

Notez que si jamais un attaquant parvient à compromettre un nœud interne du réseau, et à récupérer les clés secrètes qu'il contient, il pourra alors injecter ses propres nœuds dans le réseau, et perturber le bon fonctionnement du système. Enfin, notre solution ne traite pas la révocation des nœuds compromis ou malveillants, ni leur détection.

### 3.3.3 Modèle de l'attaquant

Notre solution de sécurité a été mise au point pour faire uniquement face à des attaquants présentant les caractéristiques suivantes :

- On considère des attaquants non avertis, qui n'ont pas de connaissances approfondies sur les systèmes d'exploitation embarqués tels que TinyOS, le langage assembleur et le langage C, ainsi que les outils de déboguage.
- On considère des attaquants qui ne sont pas en mesure de compromettre les capteurs, et ainsi ne sont pas en mesure d'extraire les clés secrètes que renferment les capteurs.
- Un attaquant peut entrer en contact (envoyer des paquets) avec n'importe quel nœud du réseau.
- Un attaquant est en mesure de placer ses propres capteurs dans n'importe quelle partie du réseau.

### 3.3.4 Spécifications de la solution de sécurité

Les capteurs sont gérés par plusieurs organisations autonomes, plusieurs gérants de palettes autonomes, plusieurs intégrateurs si bien que des palettes peuvent être équipées de capteurs issus de différentes organisations. Les organismes gérants la sécurité des capteurs doivent configurer ces capteurs avec les clés de chiffrement adéquates pour assurer la sécurité des communications entre les capteurs de la même organisation, entre capteurs de différentes organisations, ainsi que la sécurité des communications entre les capteurs et les équipements externes tels que les PDAs. De ce fait nous distinguons trois relations de sécurité :

**sécurité intra-organisation :** pour assurer la sécurité entre les capteurs de la même organisation, chaque capteur doit mémoriser le numéro de son organisation et la clé secrète correspondant à cette organisation.

**sécurité inter-organisations :** pour assurer la sécurité entre les capteurs de différentes organisations, une organisation doit partager une clé avec chaque autre organisation. Cette clé servira à sécuriser la communication entre des capteurs appartenant à deux organisations différentes.

**sécurité entre capteur et équipement externe :** pour assurer la sécurité entre un capteur et un équipement externe (PDA par exemple), cet équipement doit mémoiser autant de clés secrète qu'il y en a d'organisations. Ces clés sont calculées en fonction de ses droits et de son identifiant (voir 3.3.4.2).

L'initialisation de la couche sécurité doit s'intégrer avec Placide pour exploiter ses messages d'initialisation. Ce qui permettra de minimiser le temps d'initialisation et évitera toute redondance de messages.

Dans notre solution, chaque capteur générera deux clés de paires à partir d'une clé de base, une partagée avec son prédécesseur et une autre avec son successeur. Mais les deux capteurs particuliers dans la chaînes sont : le premier n'a qu'une seule clé de paire partagée avec son successeur, et respectivement le dernier capteur avec son prédécesseur.

#### 3.3.4.1 Notations et hypothèses

Nous supposons dans ce qui suit que nous avons $n$ organisations qui gèrent la sécurité. Donc chaque capteur doit mémoriser la clé de son organisation et $n-1$ autres clés d'inter-organisations lui permettant de communiquer avec les capteurs des autres organisations.

##### 3.3.4.1.1 Notations

- $k_i$ La clé secrète partagée par les capteurs de l'organisation de numéro i

- $k_{ij}$ La clé secrète qui assure la sécurisation des communications entre les capteurs de l'organisation i et ceux de l'organisation j
- $k^p_{uv}$ La clé de paire partagée entre les capteurs $u$ et $v$
- $P^i_{ID_A}$ La clé partagée entre l'équipement externe d'identifiant $ID_A$ et les capteurs de l'organisation i. Cette clé est enregistrée dans l'équipement $ID_A$, mais elle est calculée par le capteur quand il en a besoin.
- $MAC_k(D)$ Application de la fonction MAC avec la clé k sur les données D. Le résultat est un digest.
- $|$ Concaténation
- $N_u$ Un nombre aléatoire choisi par l'équipement (capteur ou PDA) $u$
- $DR_A$ Les droits du périphérique externe A (lecture, écriture, initialisation)
- $H(D)$ Application de la fonction de hachage $H$ aux données D
- $E_P(D)$ Chiffrement des données D avec la clé P
- $n$ le nombre d'organisations

#### 3.3.4.2 Sécurité avec l'équipement externe

L'opérateur a un PDA $A$ d'identifiant $ID_A$. S'il veut communiquer avec un capteur $u$ de l'organisation $i$, il doit d'abord activer le périphérique radio du capteur. Ensuite la communication débute par l'envoi d'un message HELLO en diffusion au capteur $u$ comme suit :

$$A \rightarrow * : ID_A, DR_A, N_A, \{MAC_{P^j_{ID_A}}(D)\}_{j=1..n}$$

Avec : $D = ID_A \mid DR_A \mid N_A$

En recevant ce message, le capteur $u$ de l'organisation $i$ ,calcule la clé de l'équipement $A$ : $P^i_{ID_A} = H(k_i \mid ID_A \mid DR_A)$. Ensuite, il calcule le MAC du message reçu. Enfin, il la vérifie.

- Si le MAC est différent de celui reçu, il ignore le message.
- Sinon il envoie le message suivant à $A$ :

$$u \rightarrow A : ok, i, N_u, MAC_{P^i_{ID_A}}(ok \mid N_u \mid N_A)$$

Alors $A$ mémorise l'organisation de $u$ et utilise la clé correspondante pour vérifier le MAC du message.

- Si la vérification échoue, alors le capteur est considéré comme malveillant
- Sinon A mémorise $ID_u$, le numéro d'organisation i, et les deux nombres aléatoires $N_u$ et $N_A$. Et il envoie sa requête à $u$ :

$$A \rightarrow u : REQ, MAC_{P^i_{ID_A}}(REQ \mid N_u \mid N_A)$$

Le capteur $u$ vérifie le MAC du message.

- Si la vérification échoue alors le message est considéré comme rejoué ou modifié ou bien le PDA $A$ est considéré malveillant

– Sinon $u$ mémorise $ID_A$, $DR_A$, et les deux nombres aléatoires $N_u$ et $N_A$. Et il vérifie les droits de $A$ vis-àvis de la requête :
  – Si $DR_A$ satisfait la requête, il lui répond :

$$u \rightarrow A : REP, MAC_{P^i_{ID_A}}(REP \mid N_u \mid N_A)$$

  – Sinon il ne l'exécute pas et n'envoie rien.

#### 3.3.4.3 Sécurité entre les capteurs

Comme nous l'avons vu dans 3.2.2, le protocole Placide nécessite une phase d'initialisation pour l'organisation du réseau. La fonction de sécurité nécessite aussi une phase d'initialisation pour partager les clés de paires (voir 3.3.4). On détaille dans ce qui suit la phase d'établissement des clés de paires, puis la sécurité en régime permanent du réseau.

##### 3.3.4.3.1 Phase d'initialisation

**Début d'initialisation** La phase d'initialisation de Placide commence par la réception d'un message d'initialisation du périphérique externe :

– Le périphérique externe envoie une requête de réinitialisation comme décrit dans 3.3.4.2. Après la propagation de la requête dans le réseau, les capteurs commencent une phase de réinitialisation ($REQ = init, N_{init}$ avec $N_{init}$ un nombre aléatoire pour garantir la fraîcheur des messages au cours de l'initialisation).
– Si les capteurs sont réveillés, en utilisant le périphérique externe A, l'opérateur envoie une requête d'initialisation spéciale en diffusion de la forme :

$$A \rightarrow * : init, ID_A, DR_A, N_{init}, \{MAC_{P^j_{ID_A}}(D)\}_{j=1..n}$$

  Avec : $D = init \mid ID_A \mid DR_A \mid N_{init}$, et $init$ est la requête d'initialisation utilisé par Placide.
  Alors comme dans le cas précédent :
  – Les capteurs génèrent la clé correspondante de A
  – Ils vérifient le MAC corespondante
  – Ils vérifient les droits de A
  – Si tout est vérifié, ils commencent l'initialisation

**Initialisation** La structure du paquet d'initialisation du capteur $u$ de l'organisation $i$ devient :

$$u \rightarrow * : SYNC, i, MAC_{k_i}(D), \{MAC_{k_{ij}}(D)\}_{j=1..n, i \neq j}$$

Avec : $D = SYNC \mid N_{init} \mid i \mid ID_u \mid ID_{destination}$ et $ID_{destination}$ est l'adresse de diffusion. Les capteurs recevant ce message vérifient $N_{init}$ puis le MAC en fonction de leur organisation et celle de la source du message $i$. Si ce n'est pas vérifié le message est ignoré. Sinon le SYNC est accepté et sera traité par Placide.
Les messages de réponse $ACK$ sont authentifiés avec une clé de paire calculée par le capteur $v$ :

$$k_{uv}^{p} = H(k \mid mac \mid ID_u \mid ID_v \mid N_{init})$$

Avec :

- si $u$ et $v$ appartiennent à la même organisation : $k = k_i$ et $mac = MAC_{k_i}(D)$ (reçu avec le message $SYNC$)
- si $u$ et $v$ appartiennent à des organisations $i$ et $j$ différentes : $k = k_{ij}$ et $mac = MAC_{k_{ij}}(D)$ (reçu avec le message $SYNC$)

Alors le message $ACK$ sécurisé est de la forme :

$$v \rightarrow u : ACK, MAC_{k_{uv}^{p}}(ACK, N_{init}, ID_v, ID_u)$$

**3.3.4.3.2 Régime permanent** En régime permanent le $N_{init}$ sera incrémenté à chaque nouveau cycle (voir figure 3.1). Tous les messages envoyés par Placide ($mess$) sont authentifiés :

$$u \rightarrow v : mess, MAC_{k_{uv}^{p}}(mess, N_{init}, ID_u, ID_v)$$

**Ajout d'un capteur dans le réseau** Un nœud $u$ (de l'organisation $i$), qui a détecté la présence d'une chaîne et veut l'intégrer, attend le dernier capteur dans la chaîne $v$ (de l'organisation $j$) et procède à un échange de messages pour pouvoir s'intégrer :

$$u \rightarrow v : Request, i, MAC_{k_i}(D_1), \{MAC_{k_{ij}}(D_1)\}_{j=1..n, i \neq j}$$

$$v \rightarrow u : SYNC, j, N_{init}, MAC_k(D_2)$$

$$u \rightarrow v : ACK, MAC_{k_{uv}^{p}}(ACK, N_{init}, ID_u, ID_v)$$

Avec :

- $Request$ message de demande d'intégration
- $D_1 = Request \mid ID_u \mid ID_v$
- $D_2 = SYNC \mid N_{init} \mid mac \mid ID_u \mid ID_v$
- si $u$ et $v$ appartiennent à la même organisation : $k = k_i$ et $mac = MAC_{k_i}(D_1)$ (reçu avec le message $Request$)
- si $u$ et $v$ appartiennent à des organisations $i$ et $j$ différentes : $k = k_{ij}$ et $mac = MAC_{k_{ij}}(D_1)$ (reçu avec le message $Request$)

**Fusion de deux chaînes** Le dernier capteur de la chaîne nouvellement formée reste en écoute pendant sa première période de sommeil pour détecter l'éventuelle présence d'une autre chaîne (voir 3.2.2). S'il détecte cette présence, il doit attendre le dernier capteur. Ensuite il procède à une intégration dans la chaîne comme s'il était seul.

## 3.4 Conclusion

Dans ce chapitre nous avons vu les particularités du projet CAPTEURS et les solutions que nous avons proposées. Ces solutions sont en cours d'implémentation sur le système d'exploitation TinyOS 2.x. L'implémentation et les tests feront l'objet du chapitre suivant.

# 4 Implémentation et tests

## 4.1 Introduction

Le protocole Placide est en cours d'implémentation par l'entreprise HLP Technologies. Nous n'avons eu en notre possession le livrable de cette application qu'au début du mois de juin 2008 pour tester la solution de sécurité, mais nous avons changé de stratégie de validation en préférant intégrer la solution de sécurité dans Placide, comme l'explique la section 4.3.1. Ce livrable n'implémente pas toutes les fonctions de Placide.
Ce chapitre est organisé comme suit : tout d'abord nous décrivons la plateforme de test, ensuite, nous expliquons les choix qu'on a fait lors de l'implémentation de l'application. Puis nous verrons les résultats des tests réalisés. Enfin nous clôturons avec une conclusion.

## 4.2 Plateforme de test

### 4.2.1 Choix du système d'exploitation

Nous avons choisi le système d'exploitation TinyOS 2.x (voir 1.3.1) car ce système est le plus supporté par la communauté des chercheurs, utilisé par plus de 500 laboratoires autour du monde. Une documentation riche et un forum très actif sont les principaux atouts de TinyOS. Ce système est spécifique aux réseaux de capteurs sans fil, au contraire de Contiki qui dérive de linux. De plus, la grande majorité des articles utilisent ce système dans les tests.

### 4.2.2 Choix du matériel

Nous avons choisi la plateforme *tmote sky* (voir tableau 2.1) pour développer et tester la solution. Cette plateforme est de la famille *telos b* (voir figure 1.2).

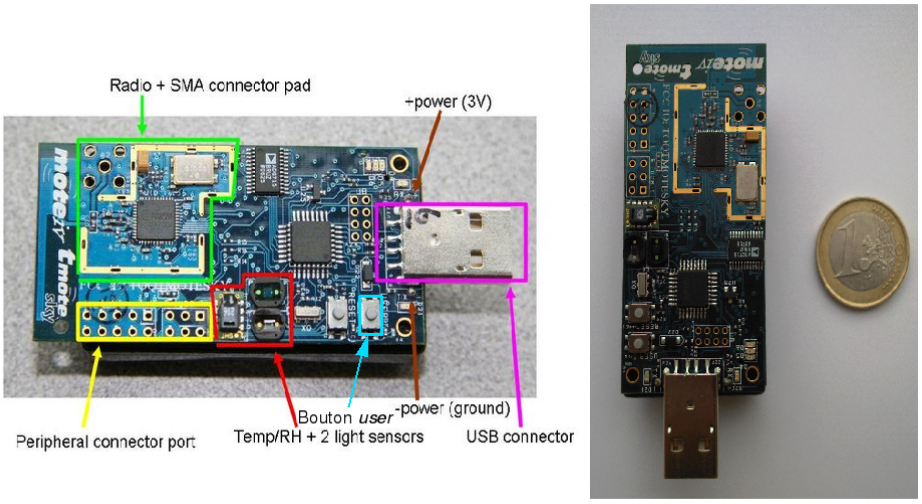



FIG. 4.1 – Capteur *tmote sky*

*Tmote sky* utilise le protocole de communication IEEE802.15.4, et a un processeur performant de 8 MHz et une mémoire flash de 1Mo. Il n'est pas gourmand en consommation d'énergie, vu qu'il utilise la puce CC2420 et le microcontrôleur MSP430, comme décrit dans la figure 4.2 de [6].

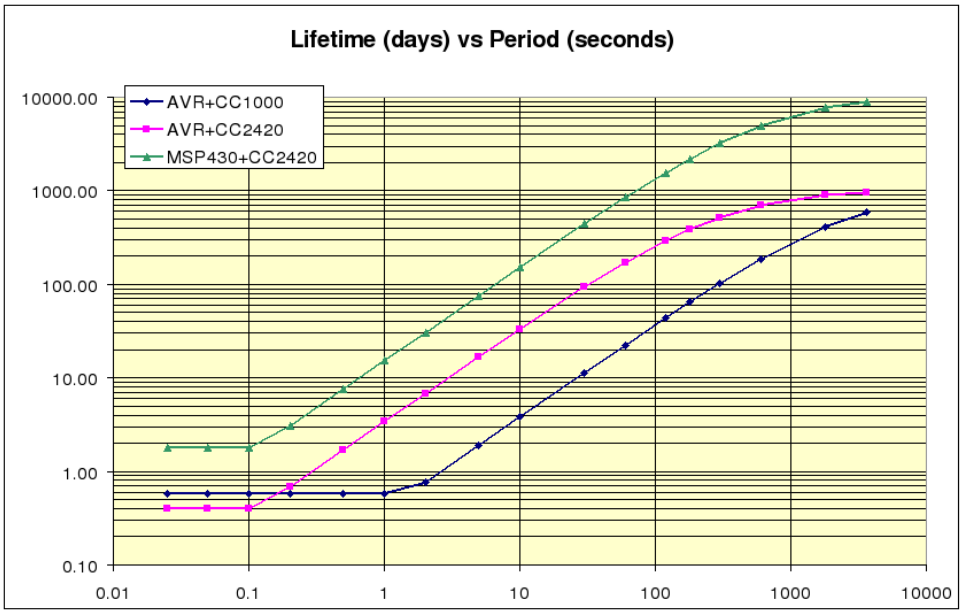



FIG. 4.2 – Comparaison entre la durée de vie des puces en fonction des périodes de sommeils [6]

### 4.2.3 Plateforme de test

L'implémentation de Placide est dépendante de la plateforme *tmote sky*. Comme nous avons intégré l'implémentation de la sécurité dans celle de Placide, donc on ne peut

exécuter l'application que sur ces capteurs. Nous avons dans le laboratoire quatre capteurs, j'utilise un capteur comme périphérique externe A et les trois autres comme des capteurs.

Le capteur A est alimenté par des piles et les trois autres sont connectés au PC, via le port *usb*, pour qu'il affiche les messages de déboguages et les mesures de temps (voir Annexe D). Le message d'initialisation (voir 3.3.4.3.1) est envoyé par A en appuyant sur le bouton *user* du capteur (voir figure 4.1). Le nombre $N_{init}$ est généré aléatoirement.

## 4.3 Implémentation

### 4.3.1 Difficultés

L'une des majeures difficultés rencontrées lors de l'implémentation de la solution de sécurité est l'intégration du protocole de sécurité dans Placide, et ce pour avoir le maximum d'efficacité, sécurité, performance et le minimum de redondance (messages, données) et espace mémoire. Ce travail de réflexion s'est déroulé en trois étapes.
D'abord nous avons conçu et implémenté une solution de sécurité totalement indépendante de Placide. Cette solution a une phase d'initialisation de durée 27 secondes pour 33 capteurs et qui consomme toute seule 58312 octets de mémoire programme et 2429 octets de RAM. Mais nous avons constaté que mettre les deux solutions Placide et sécurité dans un même capteur n'était pas faisable vu les contraintes de ressources mémoire imposées par les capteurs et l'incompatibilité des deux solutions dans leur fonctionnement.
Ensuite, nous avons eu l'idée d'implémenter les deux solutions dans deux couches indépendantes. La couche sécurité génère des interfaces pour la couche Placide pour envoyer/recevoir des messages à encapsuler dans un message de sécurité. A part la complexité d'une telle solution qui a nécessité l'écriture de squelettes de code, pour interfacer les deux couches, sur plus de 300 lignes, l'initialisation de Placide ne peut pas s'achever car cette phase est basée sur des temporisateurs et la couche sécurité engendre un retard de traitement dans l'envoi et la réception.
Enfin, on s'est mis d'accord de caler les deux solutions, et le résultat de ce travail est celui décrit dans 3.3. Notre solution utilise les messages d'initialisation de Placide pour s'initialiser. Puisque Placide est très dépendant du temps, le retard généré par le calcul de MAC et sa vérification peut perturber le fonctionnement du protocole. C'est pourquoi dans l'implémentation, nous avons essayé d'intervenir avant que Placide arme ses temporisateurs.

### 4.3.2 Choix d'implémentation

Notre protocole de sécurité se base essentiellement sur l'authentification des messages. En se basant sur le tableau 2.3 extrait de [8] et sur nos tests (tableau 4.1), j'ai choisi l'al-

gorithme CBC-MAC basé sur RC5 pour la génération des MACs. En fait, le tableau 2.3 a démontré la rapidité de CBC-MAC RC5 par rapport à Skip Jack et RC6 et nos tests ont démontré sa rapidité par rapport à SHA-1. Le calcul de MAC d'un message de 20 octets en utilisant CBC-MAC RC5 sur un tmote sky ne consomme que 4,88 ms. Nous utilisons aussi l'algorithme RC5 pour le chiffrement, parce qu'il enregistre de meilleurs résultats (voir tableau 2.2) comparativement à d'autres algorithmes, et pour utiliser le minimum de code pour la sécurité, puisqu'il est déjà utilisé (chargé en mémoire) par CBC-MAC.

| Algorithme | Temps (ms) | Cycles CPU |
|---|---|---|
| *RC5* | 4.88 | 390 400 |
| *SHA-1* | 20.50 | 1 640 000 |

TAB. 4.1 – Comparaison entre CBC-MAC RC5 et SHA-1 : génération de MAC de 4 octets sur un message de 20 octets

Le protocole Placide est très dépendant du temps car il a des périodes de réveils et de sommeils précises. Pendant la période de réveil, nous devons générer le message, l'authentifier, l'envoyer et vérifier le MAC, et ceci pour les deux messages $SYNC$ et $ACK$. C'est pourquoi le choix des algorithmes de sécurité a été basé essentiellement sur la vitesse de traitement et le nombre de cycles processeur.

Dans l'implémentation nous avons supposé que nous avons deux organisations. J'ai adapté les algorithmes RC5 et CBC-MAC, utilisés par tinySec, à la deuxième version de TinyOS.

### 4.3.3 Implémentation

Les ressources du capteur *tmote sky* sont très limitées (tableau 2.1), et pour tester l'application, on doit avoir aussi un espace pour le code de débogage. La consommation de la mémoire par notre solution (sans code de débogage) est décrite dans le tableau 4.2.

| Mémoire | Placide | Placide+sécurité | Sécurité | TinySec |
|---|---|---|---|---|
| *programme (octets)* | 37 220 | 41 504 | 4 284 | 7 146 |
| *RAM (octets)* | 1 432 | 2 092 | 660 | 728 |

TAB. 4.2 – Consommation mémoire par notre application

Les valeurs de mesures de la mémoire consommée sont données par le compilateur de *necC*. Les valeurs de mémoire consommé par la sécurité est le résultat d'une opération de

soustracion entre la mémoire consommée par *Pacide+sécurité* et *Placide*. Les valeurs de TinySec sont prises de [45].
La solution de sécurité, que nous avons proposée, consomme moins de ressources mémoire que TinySec, malgré que notre solution propose un mécanisme de gestion des clés en plus des fonctions de génération de MACs et chiffrement. Notre solution consomme aussi de la mémoire pour mémoriser les clés cryptographiques et leurs contexte d'exécution (les tables nécessaires par chaque clé pour exécuter l'algorithme de génération de MAC ou de chiffrement).

## 4.4 Résultats des tests

Le test se déroule comme suit (voir figure 4.3) : l'équipement A envoie le message d'initialisation, une fois reçu, les capteurs génèrent la clé de A et vérifient l'authenticité du message et cela nécessite une moyenne de temps de 17,5 ms (si le message est sécurisé). Ensuite, les capteurs débutent la phase d'initialisation de Placide, c'est ce temps qui est présenté dans le tableau 4.3.

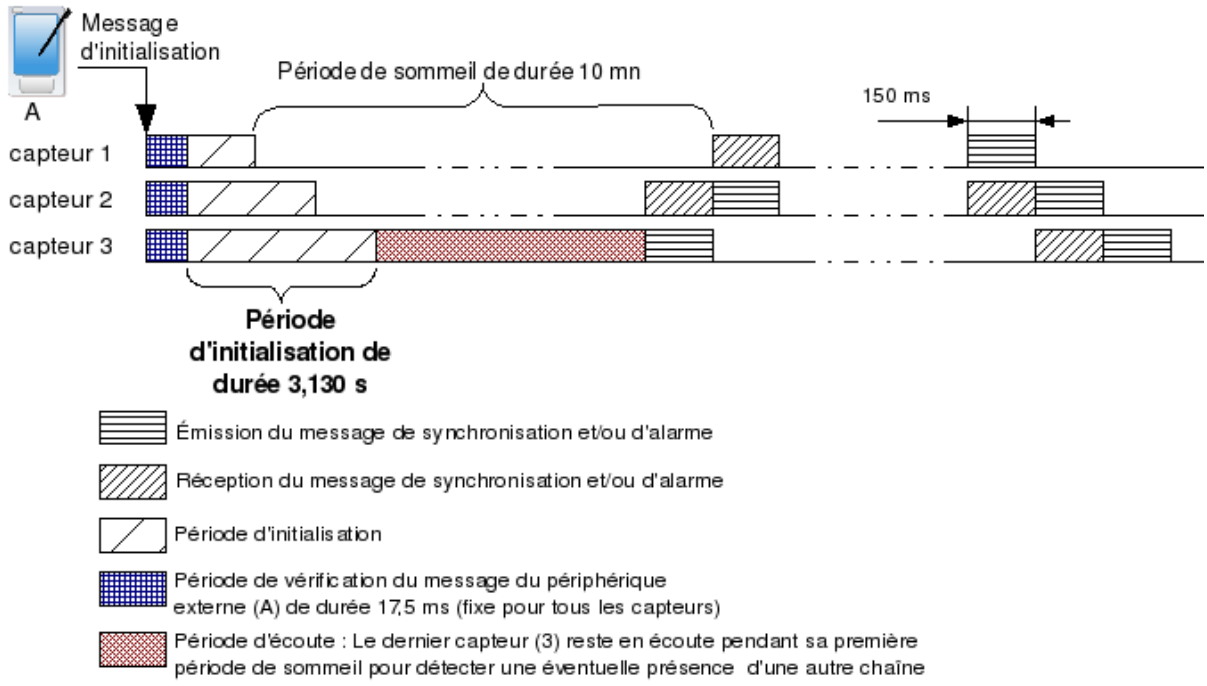



FIG. 4.3 – Scénario d'initialisation d'un réseau à trois capteurs

Nous avons mesuré le temps d'initialisation de Placide seul et ensuite de l'application complète (sécurité et Placide) et ce, pour former un réseau constitué d'un seul capteur, deux capteurs et trois capteurs. Le résultat des moyennes d'une série de tests est présenté

dans le tableau 4.3.

| Nombre de capteurs | 1 | 2 | 3 |
|---|---|---|---|
| *Placide* | 1,948 | 2,504 | 2,595 |
| *Placide+sécurité* | 2,680 | 3,021 | 3,130 |
| *Différence* | 0,732 | 0,517 | 0,535 |

TAB. 4.3 – Comparaison du temps (en seconde) nécessaire pour l'initialisation de Placide et de la sécurité intégrée dans Placide

La ligne de différence dans le tableau 4.3 entre le temps nécessaire par Placide pour s'initialiser et la solution de sécurité avec Placide montre que le temps ajouté par la sécurité pour s'initialiser ne présente pas plus de $\frac{1}{5}$ du temps nécessaire par toute la solution (Placide+sécurité) dans un réseau de 2 ou 3 nœuds. Ce qui est intéressant, car pendant cette période il y a établissement des clés de paires.

Les messages échangés entre les capteurs dans les périodes de réveils sont authentifiés, ce qui engendre un temps de retard de 4.88 $ms$ dû au calcul du MAC.

Les autres fonctions de Placide : la fusion de chaînes, la perte de nœud(s), ne sont pas encore implémentés. Il ne nous sera donc pas possible de faire des tests que lorsque ce code sera disponible.

## 4.5 Discussions

Notre solution de sécurité fournit les services de sécurité demandés dans le projet CAPTEURS (voir 3.3.2). L'une des faiblesses de notre solution est le messages d'initialisation. Mais ce message ne permet à son émetteur que le droit d'initialiser le réseau (commencer la phase d'initialisation de Placide). Il est rare car une fois le réseau est initié, il peut s'adapter avec les nouvelles topologies dynamiquement (ajout/suppression de capteurs). De plus, si un attaquant veut rejouer ce message, il doit réveiller tous les capteurs qui sont généralement en état de sommeil, ce qui nécessite une intervention physique.

Lors de la formation de la chaîne (initialisation de Placide), les capteurs prennent des identifiants ascendants, c'est à dire le premier capteur de la chaine prend l'identifiant **2**, le second prend l'identifiant **3**, et ainsi de suite. Ceci donne une fraîcheur aux messages échangés dans la phase d'initilisation. Alors un attaquant ne peut pas rejouer le message envoyé par exemple par le premier capteur car les autres capteurs attendent un identifiant supérieur car *un* est déjà endormi. Le $N_{init}$ est une prévention contre le rejeu de paquets d'une autre phase d'initialisation avec un $N'_{init}$.

Notre solution remplit les services de sécurité (voir 3.3.2) exigés dans le projet CAPTEURS.

Notre réseau peut résister à l'attaque d'injection de nœuds malveillants, puisqu'ils doivent s'authentifier lors de l'initialisation ou l'intégraton dans une chaîne par la clé d'intra-organisation ou d'inter-organisation.
Une synchronisation entre les nœuds, grâce au message SYNC de Placide, se passe à chaque période de réveil entre le capteur et son successeur et prédécesseur, s'ils existent, dans la chaîne pour résister au décalage des horloges de capteurs ou le mauvais fonctionnement d'un capteur. La panne d'un capteur ou son mauvais fonctionnement peut être considéré comme une perte de nœud. Nous avons prévu ce cas, et nous avons proposé une solution pour établir de nouvelles clés de paires entre un capteur et son nouveau successeur.
L'usurpation d'identité n'est pas possible dans notre réseau, car le capteur doit toujours authentifier son message avec sa clé et la bonne valeur de $N_{init}$.

## 4.6 Conclusion

Nous avons vu dans ce chapitre les difficultés que nous avons eues pour avoir une solution performante et intégrée dans le protocole Placide, les résultats de l'implémentation et des tests réalisés sur la solution de Placide et de sécurité. Les tests sont satisfaisants en terme de temps ajouté par les fonctions de sécurité (authentification et échange de clés) et de consommations mémoire. Notre solution a enregistré de meilleurs résultats que celle de TinySec en terme de consommation mémoire malgré que notre solution prévoit un mécanisme de gestion des clés. Malheureusement on n'a pas pu tester toutes les fonctions de sécurité car l'implémentation de Placide n'a pas été encore terminée.
Le chiffrement des messages est optionnel, et peut être réalisé avec une clé dérivée de la clé de paire.

# Conclusion

Les réseaux de capteurs prennent une place de plus en plus importante dans notre vie du fait du nombre et de la diversité des applications. La sécurité de ces réseaux est une demande exigée mais à différents niveaux selon l'application.

Le projet CAPTEURS, nous a permis d'implémenter et de tester une solution spécifique à la surveillance de la chaîne du froid. Cette solution a été valorisée par l'acceptation de notre article [49] dans la conférence internationale EUNICE'2008 (`http://conferences.telecom-bretagne.eu/eunice2008`). Nous comptons tester notre solution (Placide+sécurité) au mois de septembre 2008 dans des conditions réelles (transport de marchandises dans un camion), puisque ce projet est à usage industriel.

Des extensions à ce projet peuvent être envisagées pour offrir des services supplémentaires comme la localisation des produits et la surveillance de leur date limite. Cette solution peut être adaptée à d'autre applications ne demandant pas des traitement en temps réel telles que la surveillance de la pollution de l'air ou la surveillance des processus de production dans les usines. Par exemple, pour exporter des produits en Europe, ces produits doivent respecter la norme ISO 9000. Les réseaux de capteurs sont très adaptés à ce genre d'applications (suivi de la production).

# Remerciements

Nous tenons à remercier l'ANR (Agence Nationale de la Recherche) pour son support financier et les partenaires du projet pour leur collaboration pendant le projet CAPTEURS, à savoir ENSEEIHT, CRITT Transport et Logistique, et HLP Technologies.

# Bibliographie


[1] John A. Stankovic. Dust to doctors : Wsn for assisted living. Microsoft Research Faculty Summit 2007, July 2007.

[2] Joseph Polastre, Robert Szewczyk, Cory Sharp, and David Culler. The mote revolution : Low power wireless sensor network devices. *HOT CHIPS 16*, 2004.

[3] Zigbee and 802.15.4 for personal area and sensor networks. CS 117, Winter 2004, March 2004.

[4] "site web de tinyos", 2003. `http://www.tinyos.net`.

[5] Rahim Kacimi, Riadh Dhaou, and André-Luc Beylot. Placide : An ad hoc wireless sensor network for cold chain monitoring. *International Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NET 2008), Karlskrona, Sweden, 18/02/08-20/02/08.*

[6] Joe Polastre. Designing low power wireless systems telos / tmote sky. Presentation. Moteiv Corporation.

[7] M Healy, T Newe, and E Lewis. Efficiently securing data on a wireless sensor network. *Journal of Physics : Conference Series 76*, 2007. Sensors and their Applications XIV (SENSORS07).

[8] Germano Guimarães, Eduardo Souto, Djamel Sadok, and Judith Kelner. Evaluation of security mechanisms in wireless sensor networks. *IEEE Proceedings of the 2005 Systems Communications (ICW'05)*, 2005.

[9] Hakima Chouachi and Maryline Laurent-Maknavicius. *La sécurité dans les réseaux sans fil mobiles 3*, volume 3, chapter Sécurité dans les réseaux de capteurs sans fil, page 291. April 2007.

[10] Wikipedia. "tinyos". site web, 2008. `http://fr.wikipedia.org/wiki/TinyOS`.

[11] Vijay Garg. *WIRELESS COMMUNICATIONS & NETWORKING*, chapter Wireless Personal Area Networks : Low Rate and High Rate, pages 675 – 711. The Morgan Kaufmann Series in Networking,. Hardbound, June 2007.

[12] Saeyoung Jeong. "rfid + wsn application ". Blog, 2008. `http://tooth2.blogspot.com/2008/01/rfid-wsn-application.html`.

[13] Aaron Ricadela. "sensors everywhere ". site web, 2005. `http://www.informationweek.com/news/management/showArticle.jhtml?articleID=57702816&pgno=1`.

[14] Laurie Sullivan. "rfid lets nasa monitor hazardous materials ". site web, 2005. `http://www.informationweek.com/news/mobility/RFID/showArticle.jhtml?articleID=57300617`.

[15] Texas Instruments. *CC1000 Single Chip Very Low Power RF Transceiver.*

[16] *Part 15.4 : Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*, ieee computer society edition, 2006. IEEE Std 802.15.4.

[17] Texas Instruments. *CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver*.

[18] Jens Eliasson, Magnus Lundberg, and Per Lindgren. Time synchronous bluetooth sensor networks. *IEEE CCNC 2006*, pages 336 – 340, Jan. 2006.

[19] Mads Bondo, Dydensborg Philippe Bonnet, and Martin Leopold. Bluetooth and sensor networks : A reality check. *ACM SenSys'03*, pages 103 – 113, Nov. 2003.

[20] Aleksandar Rodzevski, Jonas Forsberg, and Ivan Kruzela. Wireless sensor network with bluetooth. *Intelligent Systems Group School of Technology and Society University of Malmö, Sweden*, 2003.

[21] Tomás Sánchez López and Daeyoung Kim. Wireless sensor networks and rfid integration for context aware services. Technical report, Information and Communications University 119 Yuseong-gu, 305-714, Daejeon, South Korea, 2007.

[22] Laura J. Celentano. *RFID-Assisted Wireless Sensor Networks for Cardiac Telehealthcare*. PhD thesis, Rochester Institute of Technology Kate Gleason College of Engineering, November 2007.

[23] Jongwoo Sung and Tomas Sanchez Lopezand Daeyoung Kim. The epc sensor network for rfid and wsn integration infrastructure. *Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops(PerComW'07)*, 2007.

[24] Wikipedia. "tinyos". site web, 2008. `http://en.wikipedia.org/wiki/TinyOS`.

[25] Vlado Handziski, Joseph Polastre, Jan-Hinrich Hauer, Adam Wolisz, David Culler, and David Gay. Hardware abstraction architecture. Best Current Practice 2.x, 2006. TEP2.

[26] Philip Levis and Cory Sharp. Schedulers and tasks. Documentary 2.x, 2006. TEP106.

[27] Cory Sharp, Martin Turon, and David Gay. Timers. Documentary 2.x, 2006. TEP102.

[28] Features ieee 802.15.4 module.

[29] David Moss, Jonathan Hui, Philip Levis, and Jung Il Choi. Cc2420 radio stack. Documentary 2.x, 6 2007. TEP102.

[30] Philip Levis. message_t. Documentary 2.x, 2005. TEP111.

[31] Gilman Tolle. Remote programming dissemination collection network management. Présentation, 2005. `http://www.cs.berkeley.edu/~prabal/teaching/cs294-11-f05/slides/day1b.ppt`.

[32] David Gay, Philip Levis, David Culler, and Eric Brewer. *nesC 1.2 Language Reference Manual*, August 2005.

[33] "TinyOS wiki". Tossim. site web. `http://docs.tinyos.net/index.php/TOSSIM`.

[34] Contiki. web site. http ://www.sics.se/contiki.

[35] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki – a lightweight and flexible operating system for tiny networked sensors. IEEE EmNetS-I, November 2004. Swedish Institute of Computer Science.

[36] John Paul Walters, Zhengqiang Liang, Weisong Shi, and Vipin Chaudhary. *Security in Distributed, Grid, and Pervasive Computing*, chapter Wireless Sensor Network Security : A Survey. Auerbach Publications, 2006.

[37] Tanveer Zia and Albert Zomaya. Security issues in wireless sensor networks. *IEEE Systems and Networks Communications, 2006. ICSNC '06*, 2006.

[38] Carl Hartung, James Balasalle, and Richard Han. Node compromise in sensor networks : The need for secure systems. Technical report, Department of Computer Science University of Colorado at Boulder, January 2005.

[39] Virgil Gligor, Bryan Parno, and Adrian Perrig. Distributed detection of node replication attacks in sensor networks. *University of Maryland & Carnegie Mellon University*, 2008.

[40] R Blom. An optimal class of symmetric key generation systems. *Proc. of the EUROCRYPT 84 workshop on Advances in cryptology : theory and application of cryptographic techniques, Paris, France*, pages p.335–338, December 1985.

[41] Kui Ren, Wenjing Lou, Kai Zeng, and Patrick J. Moran. On broadcast authentication in wireless sensor networks. *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS*, 11 2007.

[42] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. *Research SUN*, 12 2004.

[43] Benjamin Arazi, Itamar Elhanani, Ortal Orazi, and Hairong Qi. Revisiting public-key cryptography for wireless sensor networks. *EMBEDDED COMPUTING*, 11 2005.

[44] Rodrigo Roman and Cristina Alcaraz. Applicability of public key infrastructures in wireless sensor networks. Presentation, June 2007. University of Malaga, Spain.

[45] Chris Karlof, Naveen Sastry, and David Wagner. Tinysec : A link layer security architecture for wireless sensor networks. *ACM SenSys '04*, November 2004.

[46] Adrian Perrig, Robert Szewczyk, Victor Wen, David Culler, and J. D. Tygar. Spins : Security protocols for sensor networks. *Mobile Computing and networking*, 2001.

[47] Wikipedia. "chaîne du froid". site web, 2008. `http://fr.wikipedia.org/wiki/Chaine_du_froid`.

[48] David Culler and Shankar Sastry. Wireless oep breakout session, Mai 2001.

[49] Maryline Laurent-Maknavicius, Chakib Bekara, and Wassim Drira. Light and simple security solution for cold chain supervision. In *EUNICE 2008 copyright IFIP 2008*. A. Gravey, Y. Kermarrec, X. Lagrange (Eds.), September 2008.

# Glossaire

| | |
|---|---|
| AES | Advanced Encryption Standard, 10 |
| ANR | Agence Nationale de Recherche, 24 |
| BER | Bit Error Rate, 10 |
| CBC-DES | Cipher Block Chaining Data Encryption Standard, 22 |
| CBC-MAC | Cipher Bloc Chaining Message Authentication Code, 11 |
| CCM | Ctr Cbc-Mac, 11 |
| CSMA-CA | Carrier Sense Multiple Access with Collision Avoidance, 10 |
| CTR | CounTeR mode, 11 |
| EPC | Electronic Product Code, 8 |
| IEEE | Institute of Electrical and Electronics Engineers, 10 |
| ISO | International Organization for Standardization, 43 |
| LR-WPAN | Low Rate Wireless Personal Area Network, 8 |
| MAC | Media Access Control, 8 |
| MAC | Message Authentication Code, 16 |
| PDA | Personal Digital Assistant, 25 |
| RAM | Random Access Memory, 12, 28 |
| RCSF | Réseaux de Capteurs Sans Fil, 5 |
| RFID | Radio-Frequency IDentification, 8 |
| SE | Système d'Exploitation, 12 |
| SNEP | Sensor Network Encryption Protocol, 22 |
| SNR | signal-to-noise ratio, 10 |
| TOSSIM | TinyOS SIMulator, 13 |

# A Structure du paquet de la norme 802.15.4

La structure principale du paquet est celle définie dans le fichier tos/types/message.h est ($uint8_t$ est un octet non signé) :

```
typedef nx_struct message_t {
  uint8_t header[sizeof(message_header_t)];
  uint8_t data[28];
  uint8_t footer[sizeof(message_footer_t)];
  uint8_t metadata[sizeof(message_metadata_t)];
} message_t;
```

La queue du message est vide et la structure de l'en-tête est (tos/chips/cc2420/IEEE802154.h) :

```
typedef nx_struct cc2420_header_t {
uint8_t length;
uint16_t fcf;
uint8_t dsn;
uint16_t destpan;
uint16_t dest;
uint16_t src;

/** Si on a définit la fonctionnalité optionnelle d'interopérabilité avec d'autres SE */

#ifdef CC2420_IFRAME_TYPE
nxle_uint8_t network;
#endif

uint8_t type;
} cc2420_header_t;
```

# B Pile protocolaire de la puce CC2420

La pile protocolaire de la puce CC2420 (la puce radio du capteur *tmote sky*) est décrite dans [29]. La figure B.1 décrit l'organisation des différents couches de cette pile.

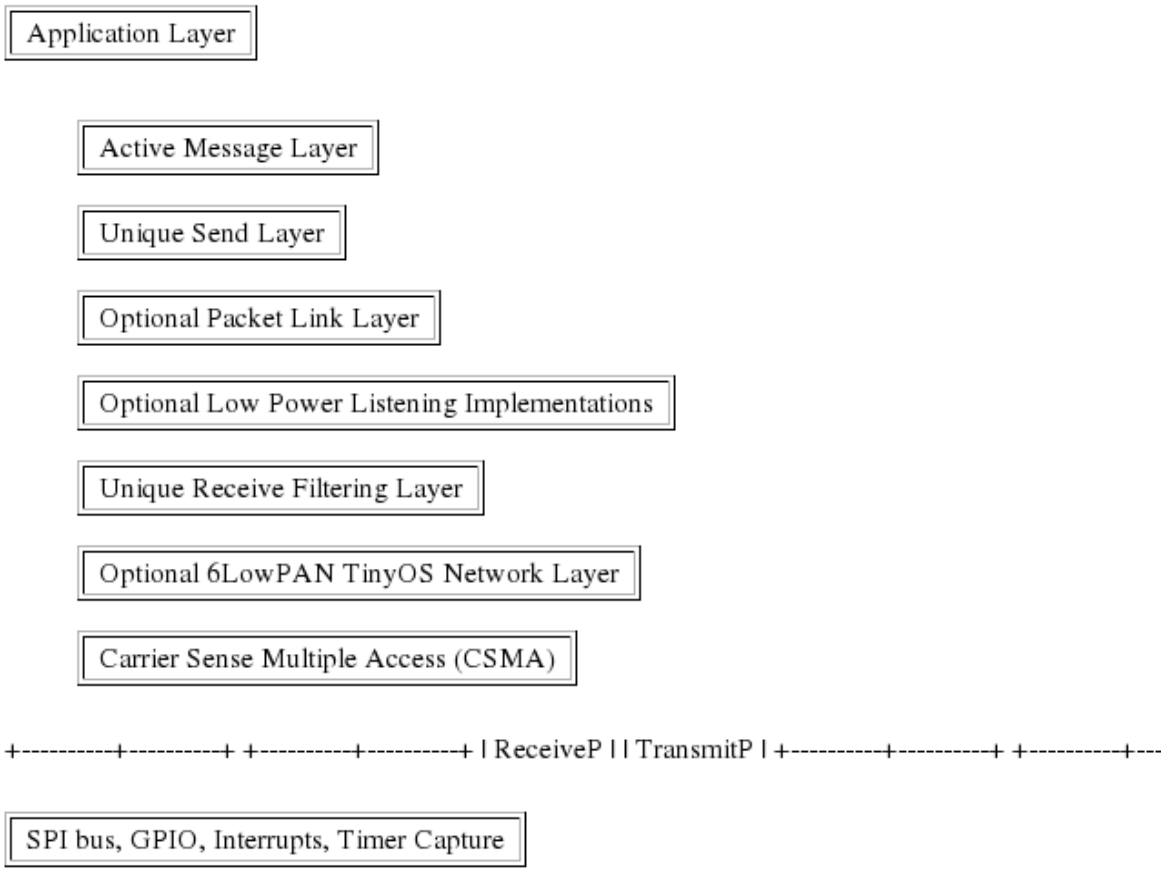



FIG. B.1 – Les couches de la pile protocolaire de la puce CC2420

- **Active Message Layer :** C'est la couche supérieure dans la pile et elle est responsable du remplissage de l'en-tête du message et produit des informations sur les paquets pour la couche application.
- **UniqueSend :** Cette couche génère le nombre de séquence DSN (Data Sequence

Number) qui est unique et un champ de l'en-tête du message. Cet octet sera incrémenté à chaque message sortant, et sert au récepteur pour détecter les paquets dupliqués.

- **PacketLink :** Cette couche est responsable de la retransmission automatique des paquets.
- **Low Power Listening :** Cette couche implémente des composants qui servent à l'écoute avec faible consommation énergétique du média radio pour recevoir ou émettre quand ceci est nécessaire.
- **UniqueReceive :** Cette couche maintient un historique des derniers DSN reçus pour filtrer les paquets dupliqués.
- **TinyosNetworkC :** Cette couche permet à la pile radio de TinyOS 2.x de communiquer avec les autres systèmes non TinyOS.
- **CSMA :** elle est responsable de la définition du champ FCF (du paquet de 802.15.4) dans les messages sortants et produit le temps d'attente *backoff* quand le canal est occupé.
- **TransmitP/ReceiveP :** sont responsables de la communication directe avec le périphérique radio.

# C Les messages de Placide

*SYNC*

```
typedef nx_struct sync_msg {
  uint16_t node_id_max;      // Last node identificator
  uint16_t number_node;      // Number of nodes
  uint32_t sending_time;     // the local time when the message is prepared for sending
  uint32_t wake_time;        // Time before wake up
  uint16_t dummyOffset;      // the real local time when the message is sent
} sync_msg_t;
```

*ACK*

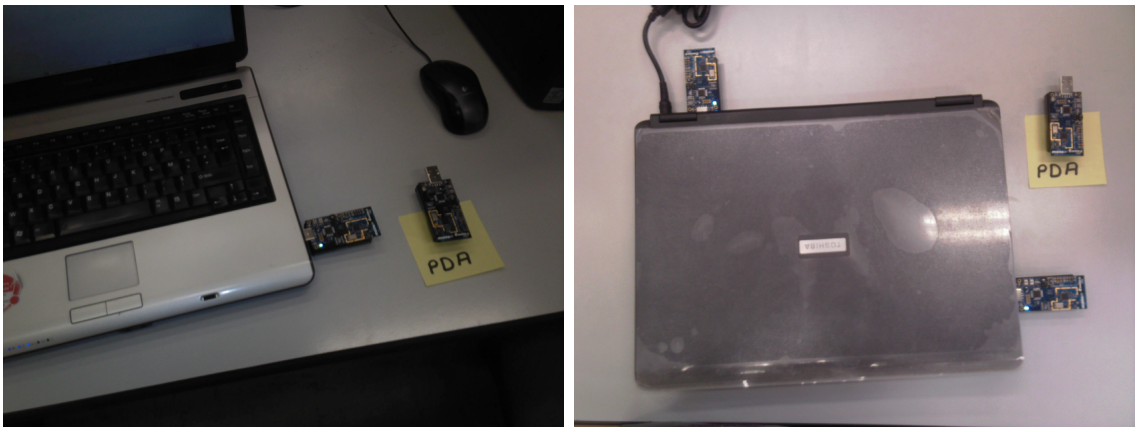
```
typedef nx_struct ack_msg {
  bool ack;
} ack_msg_t;
```

*init*

```
typedef nx_struct base_msg {
  bool data;
  bool db;
  bool erase;
} base_msg_t;
```

# D Tests

Notre plateforme de test (voir figure D.1) se compose d'un capteur contenant le programme PDA et d'autres capteurs que nous les connectons au PC pour afficher les messages de débogage.




(a) (b)

FIG. D.1 – Plateforme de test

Pour lancer la phase d'initialisation, on doit appuyer sur le bouton *user* du PDA (voir figure D.2), le message d'initialisation sera envoyé aux capteurs, qui lancent la phase d'initialisation, après la vérification du message reçu.

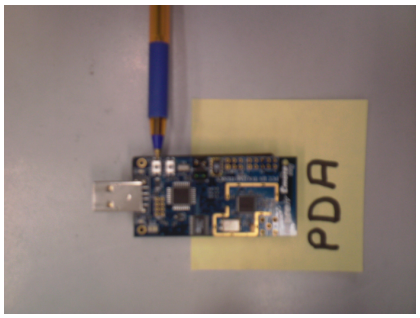
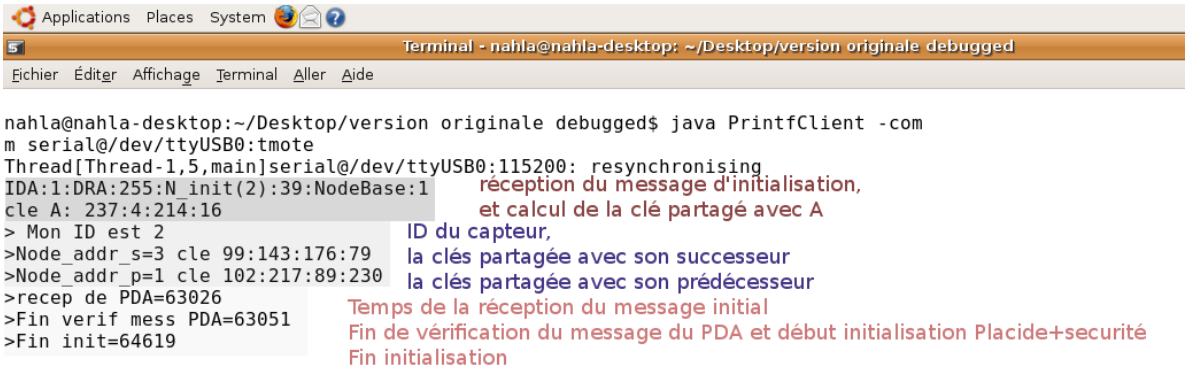


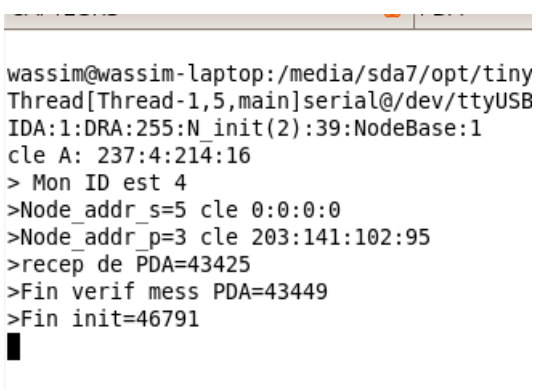

FIG. D.2 – Bouton *user*

Les captures d'écran dans la figure D.3 montrent les traces après la fin de la phase d'initialisation.




(a) capteur d'idetifiant 2


(b) capteur d'idetifiant 3




(c) capteur d'idetifiant 4

FIG. D.3 – Captures d'écran de la trace des capteurs lors de la formation d'un réseau à trois nœuds