Protéger les réseaux de capteurs sans fil

Claude Castelluccia et Aurélien Francillon

INRIA Grenoble-Rhône-Alpes
655, avenue de l'Europe
38334 Saint-Ismier Cedex, France
claude.castelluccia@inrialpes.fr

1 Introduction

Les progrès réalisés ces dernières décennies dans les domaines de la microélectronique, de la micromécanique, et des technologies de communication sans fil, ont permis de produire à un coût raisonnable des composants de quelques millimètres cubes de volume. De ce fait, un nouveau domaine de recherche s'est créé pour offrir des solutions économiquement intéressantes et facilement déployables à la surveillance à distance et au traitement des données dans les environnements complexes et distribués : *les réseaux de capteurs sans fil*. Les réseaux de capteurs sans fil sont constitués de noeuds déployés en grand nombre en vue de collecter et transmettre des données environnementales vers un ou plusieurs points de collecte, d'une manière autonome. Ces réseaux ont un intérêt particulier pour les applications militaires, environnementales, domotiques, médicales, et bien sûr les applications liées à la surveillance des infrastructures critiques. Ces applications ont souvent besoin d'un niveau de sécurité élevé. Or, de part de leurs caractéristiques (absence d'infrastructure, contrainte d'énergie, topologie dynamique, nombre important de capteurs, sécurité physique limitée, capacité réduite des noeuds,...), la sécurisation des réseaux de capteurs est à la source, aujourd'hui, de beaucoup de défis scientifiques et techniques.

2 Réseaux de Capteurs

Un réseau de capteur est composé de centaines ou de milliers mini-ordinateurs. Ces appareils, appelés en anglais **motes**, sont alimentés par des piles et sont typiquement déployés de façon aléatoire dans des environnements souvent ouverts. Généralement, ces capteurs font des mesures périodiques et envoient les données collectées à un dispositif plus puissant, le puit (sink) ou la station de base, qui les traite en calculant par exemple leur maximum, moyenne ou médiane.

Les noeuds qui composent un réseau de capteur sans fil sont petits et par conséquent ont des ressources de calcul, de stockage, de communication et d'énergie très limitées. L'architecture système d'un capteur sans fil est composée de quatre éléments : (i) Un sous-système de calcul composé d'un microprocesseur ou d'un microcontrôleur, (ii) un sous-système de communication composé d'une radio de courte portée (iii) un sous-système de mesure qui lie le noeud avec le monde physique, et (iv) un sous-système d'alimentation d'énergie, qui loge la pile et le convertisseur DC-DC, et qui alimente le reste du noeud.

Les applications potentielles des réseaux de capteurs sont les applications militaires et les applications de surveillance de l'environmment (monitoring). Par exemple, ils peuvent être utilisés pour la détection de feux dans des grandes zones forestières, l'observation d'environnements naturels (pollution, inondation, etc.), suivi d'écosystèmes (surveillance d'oiseaux, croissance des plantes,

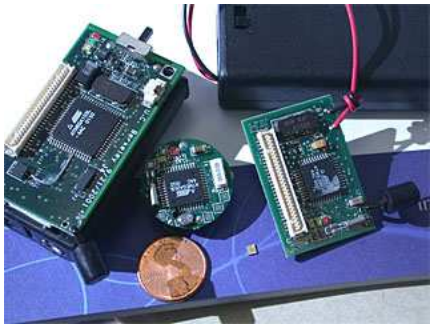

FIG. 1: Berkeley Mote

etc.), contrôle militaire (télésurveillance de champs de bataille, détection d'ennemis, etc.), analyses biomédicales et surveillance médicale (détection de cancer, rétine artificielle, taux de glucose, diabètes, etc.),. Un des intêrets des réseaux de capteurs est qu'ils peuvent être déployés dans des endroits difficiles d'accès.

Les réseaux de capteurs ont aussi beaucoup d'applications dans le domaine de la santé. Ils peuvent, par exemple, être utilisés pour surveiller à distance des patients. Dans ce cas, ils permettent non seulement d'améliorer la qualité de vie des malades, qui peuvent rester chez eux, mais aussi intervenir le plus rapidement possible si les mesures effectuées par les capteurs sont anormales. A titre d'exemple, Intel travaille sur un projet de recherche dans le but est d'assister les personnes agées. Dans ce projet, chaque objet de la maison (tasses, assiettes, chaises,...) est équipé d'un micro-capteur et enregistre les activités quotidiennes de l'occupant. Les capteurs envoient ensuite les données à un système centralisé qui permet alors à un infirmier de contrôler en permanence et à distance les activités de l'occupant.

Toutes ces applications ont des contraintes de sécurité très différentes. Cependant, dans la plupart d'entre elles, l'intégrité et l'authenticité des données doivent être fournies pour s'assurer que des noeuds non-autorisés ne puissent pas injecter des données dans le réseau. Le chiffrement des données est souvent requis pour des applications sensibles telles que les applications militaires ou les applications médicales.

Dans ce document, nous nous intéressons essentiellement à la sécurisation du réseau (c'est à dire de l'insfrastructure). Il faut cependant noter que sécuriser le réseau est nécessaire mais pas suffisant. En effet, un attaquant peut, dans certains cas, facilement modifier l'environnement et injecter des données erronées. Par exemple, dans le cas d'un réseau utilisé pour la détection de feux dans une

forêt, un attaquant peut chauffer plusieurs capteurs avec un briquet et générer un fausse alerte. Par conséquent, le réseau doit aussi utiliser des tests de plausibilité qui permettent de vérifier que les mesures obtenues sont cohérentes. Ces tests sont généralement réalisés par le puit (sink). La sécurisation des réseaux de capteurs reste un problème difficile pour les raisons suivantes :

1. **Capacités limitées** :

   Les ressources de calcul et de mémoire des noeuds sont relativement faibles. Par exemple, les noeuds de capteur de type "mote" sont composés d'un micro-controleur 8-bits 4MHz, 40 KOctets de mémoire et une radio avec un débit d'environ 10 kbps. Cela reste vrai même pour les noeuds de moyenne gamme, comme les UCLA/ROCKWELL'S WINS, qui ont un processeur StrongARM 1100 avec une mémoire flash de 1 MO, une mémoire RAM de 128 KO et une radio de 100 Kbps. Non seulement les capacités des noeuds sont faibles, mais en plus ils opèrent sur des piles et par conséquent ont une durée de vie limitée. L'énergie limitée des capteurs est probablement la caractéristique la plus pénalisante. Le plus grand des défis dans le domaine des réseaux de capteurs reste de concevoir des protocoles, entre autre de sécurité, qui minimisent l'energie afin de maximiser la durée de vie du réseau. En d'autres mots, l'energie est sans aucun doute la ressource qui convient de gérer avec la plus grande attention.

   Les solutions de sécurité qui existent aujourd'hui ne sont pas utilisables car elles sont souvent trop coûteuses en terme de ressources. Par exemple, l'utilisation de la cryptographie à clés publiques est souvent proscrite de ce type d'environnement. De nouveaux algorithmes et protocoles de sécurité sont nécessaires.

2. **Agrégation des données** : Il a été montré dans plusieurs publications scientifiques que la transmission d'un bit est équivalent, en terme d'énergie, à l'execution d'environ 1000 instructions. Cette valeur augmente avec la portée de la radio. Plus le capteur devra transmettre loin, et par conséquent augmenter sa puissance d'emission, plus il va consommer de l'energie, et par conséquent réduire sa durée de vie. Il convient donc de réduire en compressant ou en agrégeant les données lors de leur routage.

   Les techniques d'agrégation des données, c'est à dire de traitement des données par le réseau, permettent de réduire le nombre de messages (et de bits transmis sur les liens sans-fil) et par conséquent réduire la comsommation en energie. Par exemple, si un réseau est déployé pour mesurer la température et que le puit n'est interessé que par la moyenne des températures, un noeud intermédiaire pourra additionner les valeurs reçues de ses enfants et envoyer le résultat à son père. Le puit recevra alors qu'un seul message, contenant la somme des données au lieu de n messages (ou n est le nombre de capteurs).

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

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

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

3 Établissement de liens sécurisés

Cette section présente certaines de nos contributions sur l'établissement de liens sécurisés dans les réseaux de capteurs. Dans beaucoup d'applications, l'établissement des clés doivent se faire une fois que les capteurs sont déployés. C'est le cas lorsque qu'un grand nombre de capteurs est déployé de façon aléatoire, par exemple à partir d'un hélicoptère. Dans ce type de scénario, chaque noeud doit établir une clé secrète avec chacun de ses voisins. Ses voisins n'étant pas connus avant le déployement, ces échanges de clés doivent avoir lieu sur le terrain. Ce problème serait trivial à résoudre si l'utilisation des protocoles utilisant la cryptographie à clé publique était possible. Les noeuds pourraient alors s'échanger leur certificat, un composant de type Diffie Hellman et obtenir une clé secrète. Malheureusement, l'utilisation d'algorithme utilisant la cryptographie à clé publique est trop coûteuse et ne peut être utilisée dans ce type d'environnement.

Pour résoudre ce problème, Eschenauer et Gligor ont proposé, en 2002, un protocole probabilistique qui est devenu le standard de-facto [5]. Ce protocole, qui est relativement simple, opère comme suit :

1. L'administrateur du réseau génère un tableau de l nombres aléatoires, indexés de 1 à l.
2. Chaque noeud est ensuite configuré avec un sous-ensemble m de clés choisies aléatoirement parmis les l clés précédentes.
3. Lorsque deux noeuds, A et B, veulent établir une clé secrete, ils s'échangent les indexes de leurs clés. Ils utilisent ensuite les clés qu'ils ont en commun pour générer un secret (par exemple en les hachant en utilisant un fonction de hachage comme SHA1).

Ce protocole est probabilistique car il est toujours possible qu'un noeud, autre que A et B ait également été configuré avec les clés que A et B possèdent en commun. Ce noeud pourra alors calculer le secret que A et B viennent d'établir. Cependant Eschenauer et Gligor ont montré que cette probabilité peut être faible si les paramètres l et m sont choisis avec précaution.

Nous avons montré que, bien que ce protocole soit efficace, la sécurité qu'il fournit se dégrade rapidement avec le temps. En effet, chaque fois qu'un noeud est compromis, ses clés sont révélées. Au fil du temps, et en faisant l'hypothèse que l'attaquant compromet continuellement des noeuds, une grande partie des clés du système sera connu de l'attaquant. L'attaquant peut, alors, calculer les clés établies entre les noeuds et lire le contenu des messages échangés ou insérer des messages érronés. Dans beaucoup d'applications, de nouveaux noeuds devront être ajoutés au fil du temps pour remplacer ceux qui ont consommé leurs piles. La probabilité que les clés de ces nouveaux noeuds soient connues par l'attaquant augmente avec le temps.

Pour résoudre ce problème nous avons proposé *RoK*, une extension du protocole précédent [3]. Cette extension améliore considérablement la sécurité du protocole initial. Plus spécifiquement, avec *RoK*, le réseau s'auto-guérit lorsque que l'attaque est temporaire (c'est à dire compromet les noeuds uniquement pendant une période de temps, puis disparait). En d'autre mots, l'état du réseau redevient sain lorsque l'attaquant disparait. Par ailleurs, l'état du réseau se stabilise lorsque l'attaquant est permanent (c'est à dire compromet des noeuds continuellement sans s'arrêter).

Les figures 2 et 3 montrent les résultats de quelques simulations (pour plus de détails sur ces simulations, le lecteur pourra lire l'article *RoK* [3]).

La figure 2 montre la proportion de liens compromis en fonction du temps (c'est à dire en fonction du nombre de noeuds compromis) pour un attaquant *permanent*. Les différentes courbes correspondent aux résultats obtenus avec le protocole standard et le protocole RoK, pour différents nombres de compromissions par période de temps. Ces résultats montrent que la sécurité du protocole standard se dégrade avec le temps et qu'au bout d'un certain nombre de périodes de temps

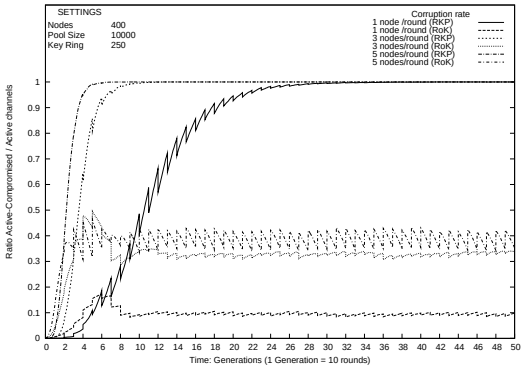

FIG. 2: Ratio de liens compromis avec attaquant permanent

(variable en fonction de l'aggressivité de l'attaquant), tous les liens du réseau sont compromis ! En comparaison, lorsque RoK est utilisé, le pourcentage de liens compromis converge vers une valeur qui varie entre 10% et 40%.

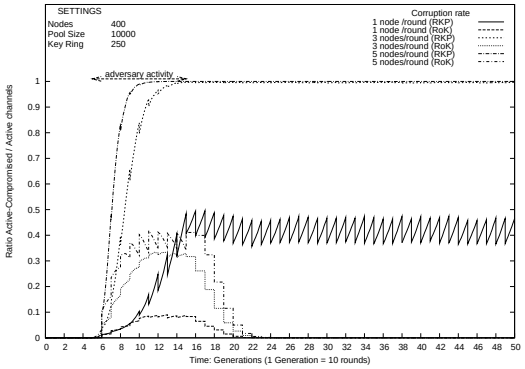

FIG. 3: Ratio de liens compromis avec attaquant temporaire.

La figure 3 montre la proportion de liens compromis en fonction du temps pour un attaquant *temporaire*. Dans nos simulations, l'attaquant est présent entre les périodes 5 et 15, puis disparaît. Les différentes courbes correspondent aux résultats obtenus avec le protocole standard et le protocole *RoK*, pour différents nombres de compromisions par période de temps. Ces résultats montrent que la sécurité du protocole standard se dégrade avec le temps et que, comme précédemment, au bout d'un certain nombre de périodes de temps, tous les liens du réseau sont compromis. En revanche, avec RoK, le pourcentage de liens compromis augmente entre 10% et 40%, mais ensuite converge vers 0 lorsque l'attaquant disparait. Ces résultats illustrent bien la propriété d'auto-guérisson du protocole RoK.

L'idée principale du protocole *RoK* est de limiter la durée de vie des clés secrètes contenues dans le tableau et de faire évoluer les clés du système en fonction du temps. Ce choix est motivé par l'observation que la durée de vie d'un capteur est limitée et que par conséquent ses clés n'ont pas besoin d'être permanentes.

Une solution naive serait que le système change les clés de son tableau à chaque période. A une période donnée T, un noeud pourrait échanger des secrets avec ses voisins et ensuite effacer les clés de configuration de sa mémoire. Par conséquent, si ce noeud est compromis plus tard, ses clés ne pourront pas être utilisées. En d'autres mots, l'attaquant serait condammé à être très rapide et compromettre les noeuds dès leur déployement. Cette solution, bien qu'efficace en terme de sécurité, a un gros inconvénient : un noeud déployé à la période $T+1$, ne peut pas configurer de secret avec les noeuds déployés pendant les périodes précédentes ! Elle a donc peu d'intêret en pratique.

RoK résout ce problème en attribuant à chaque noeud $w * m$ clés, ou w est la durée de vie moyenne d'un capteur. Si un noeud est déployé à la période T, il sera configuré avec m clés de la période T, m clés de la période $T+1$,..., et m clés de la période $T+w$. Ainsi ce noeud pourra échanger un secret avec des noeuds de sa génération, mais aussi des noeuds des génération $T+1$, $T+2$,...,$T+w$. Un des inconvénients de cette approche est qu'elle augmente le nombre de clés que chaque noeud doit stocker par un facteur de w. Pour résoudre ce problème, nous avons développé une nouvelle construction qui utilise une chaine de hashs doublement chainée. Elle permet de réduire le coût mémoire de $w * m$ à $2 * m$, c'est à dire un coût constant. L'article [3] présente le protocole RoK en détail. Il analyse également sa sécurité analytiquement et par simulation.

4 Sécurisation des données agrégées

Les réseaux de capteurs sont des réseaux ad-hoc composés de dispositifs minuscules qui ont des capacités de calcul, mémoire et énergie très limitées. Comme la transmission des données est l'opération la plus coûteuse en terme d'énergie, il est essentiel, pour optimiser la durée de vie du réseau, de réduire le nombre de bits envoyés et relayés par les noeuds intermédiaires.

Une approche répandue consiste à agréger les données lors de leur acheminement vers le puit (sink)- le noeud qui collecte et traite l'ensemble des données.

Prenons l'exemple d'un réseau qui est déployé pour mesurer la température moyenne dans une zone géographique donnée. Ce réseau est configuré comme un arbre ou les feuilles et les noeuds intermédiaires sont les capteurs et la racine est le puit. Chaque capteur envoie périodiquement ses données vers le puit. Chaque message est relayé noeud par noeud du capteur vers le puit. Par conséquent, si le réseau est constitué de n noeuds, le puit recevra à chaque période de mesure n messages !

Pour réduire le nombre de messages et de bits transmits, chaque noeud intermédiaire peut additionner les données (températures) reçues de ces fils, ajouter la valeur de sa mesure, et envoyer le résultat à son pére. Le puit recevra alors un seul message qui contiendra la somme des messages au lieu de n messages (voir Figure 4). Il pourra alors diviser cette somme par n et obtenir la moyenne de la température. Si les valeurs mesurées par chaque capteur sont codées sur m bits, le puit recevra $log_2(n) + m$ bits au lieu de $n + m$. Le gain en bande passante est alors de $(log_2(n) + m)/(n + m)$.

L'agrégation des données est relativement triviale, mais devient problématique lorsque l'on veut y ajouter de la sécurité et plus particulièrement de la confidentialité (chiffrement). Dans certaines applications, il est essentiel de s'assurer que les informations qui sont transmises sur le réseau ne puissent être interceptées et lues par des personnes non-autorisées. Elles doivent donc être chiffrées. Mais le chiffrement et l'agrégation ne vont pas très bien ensemble.

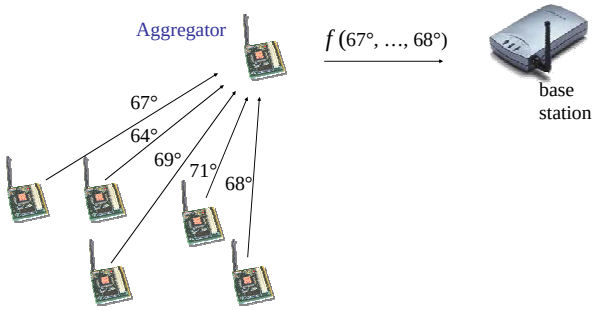

FIG. 4: Agrégation dans réseau de capteurs

Il y a bien entendu deux solutions simples (mais naives) à ce problème. La première consiste à configurer tous les noeuds du réseau avec une clé de groupe. Chaque noeud chiffre alors ses données avec cette clé en utilisant un algorithme de chiffrement symétrique standard, comme AES ou RC5, et envoie le résultat à son père. Ce noeud déchiffre alors tous les messages reçus de ces enfants, les ajoute, ajoute sa mesure, chiffre le résultat avec la clé de groupe et envoie le message chiffré à son pére. Les messages sont ainsi relayés et agrégés de noeud en noeud jusqu'au puit. Le puit peut alors déchiffrer le message et retrouver la somme des données. Cette solution a plusieurs inconvénients : (1) Elle est peu sure car il suffit de compromettre un seul noeud pour découvrir la clé de groupe et déchiffrer l'ensemble des messages. (2) Elle est peu efficace car chaque noeud doit déchiffrer plusieurs messages et en chiffrer un.

Une deuxième solution consiste à utiliser, au lieu d'une clé de groupe, une clé différente lien par lien. En d'autres mots, dans cette solution, chaque noeud établit une clé secrète avec chacun de ses voisins. Il déchiffre alors les données reçus de ses fils, les agrège, puis chiffre le résultat avec la clé qu'elle possède avec son père. Cette solution est meilleure, en terme de sécurité, que la précédente car la compromision d'un noeud ne révèle que les clés utilisées par ce noeud et non une clé globale. Cependant elle a les inconvénients suivants : (1) La compromission d'un noeud près du puit permet d'obtenir un agrégat qui est significatif car chaque noeud a accès aux données agrégées envoyées par ses fils. (2) Elle nécessite l'établissement de clé entre chaque noeud voisin, ce qui n'est pas trivial. (3) Comme la solution précédente, elle est relativement coûteuse.

Une solution de *bout-en-bout* serait préférable car la compromision d'un noeud ne fournirait aucune information sur l'aggrégat ou les données envoyées par les autres noeuds du système. L'idéal serait d'avoir une solution ou chaque noeud chiffrerait ses données avec une clé qu'il partagerait avec le puit et avec laquelle les noeuds intermédiaires manipuleraient des données chiffrées sans

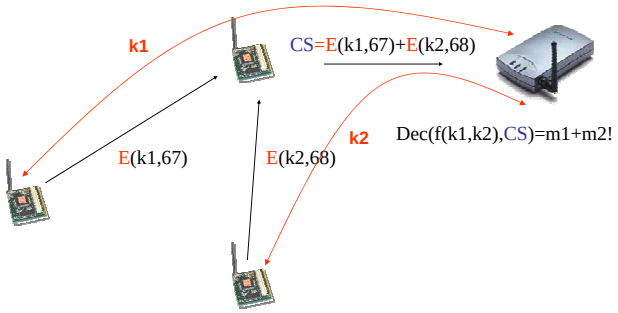

FIG. 5: Agrégation Sécurisée

jamais accéder aux données en clair (voir Figure 5). Pour arriver à ce resultat, nous avons besoin d'un fonction de chiffrement homomorphique par l'addition, c'est une fonction de chiffrement $enc()$ qui posséde la propriété suivante :

$$enc(k_1, m_1) \otimes enc(k_2, m_2) = Enc(f(k_1, k_2), m_1 + m_2)$$

En d'autres mots, la somme des données en clair $m_1 + m_2$, peut être obtenues en déchiffrant avec une clé qui dépend de k_1 et k_2, le résultat de $\otimes$, une fonction à définir, sur les messages chiffrés m_1 et m_2.

Avec une telle fonction de chiffrement, il suffirait alors que chaque noeud exécute la fonction $\otimes$ en utilisant comme paramètres d'entrée, les valeurs chiffrées de ses fils. Le puit pourrait alors obtenir la somme des données en déchiffrant le message qu'il reçoit avec une clé qui dépendrait de l'ensemble des clés qu'il partage avec les noeuds.

Nous avons proposé une fonction de chiffrement qui possède cette propriété et qui, en plus, est très performante et bien adaptée aux réseaux de capteurs. Nous avons montré qu'en remplaçant dans un chiffrement par flot (stream-cipher) le ou-exclusif *xor* par une addition modulaire (+) nous obtenons une algorithme de chiffrement qui est homomorphique par l'addition et dont le sécurité est prouvable. Cette fonction est décrite dans le tableau 4.

Nous faisons l'hypothèse que $0 \leq m < M$. Etant donné que l'addition est une opération commutative, la fonction décrite précédemment est homomorphique par l'addition. En effet, si $c_1 = Enc(m_1, k_1, M)$ et $c_2 = Enc(m_2, k_2, M)$ alors $c_1 + c_2 = Enc(m_1 + m_2, k_1 + k_2, M)$. Si n messages chiffrés sont ajoutés, alors M doit être plus grand que $\sum_{i=1}^{n} m_i$. En effet, si M est plus petit que $\sum_{i=1}^{n} m_i$, le résultat du déchiffrement donnerait une valeur m' inférieure à M et plusieurs

Additively Homomorphic Encryption Scheme

Encryption :

1. Represent message m as integer $m \in [0, M-1]$ where M is large integer.
2. Let k be a randomly generated keystream, where $k \in [0, M-1]$
3. Compute $c = Enc(m, k, M) = m + k \pmod{M}$

Decryption :

1. $Dec(c, k, M) = c - k \pmod{M}$

Addition of Ciphertexts :

1. Let $c_1 = Enc(m_1, k_1, M)$ and $c_2 = Enc(m_2, k_2, M)$
2. For $k = k_1 + k_2$, $Dec(c_1 + c_2, k, M) = m_1 + m_2$

valeurs de $m_1 + m_2$ seraient alors possibles. En pratique, si $p = max(m_i)$ alors M doit être égale à $M = 2^{\lceil log_2(p*n) \rceil}$.

Les articles scientifiques [1,4] présentent notre solution en détail. Il présente également des résultats de performance et la preuve de sécurité de cette nouvelle construction. La solution proposée utilise uniquement des additions modulaires et est, par conséquent, très bien adaptée aux réseaux de capteurs. Elle constitue, aujourd'hui, la seule solution pratique à ce problème.

Un inconvénient de cette solution est que le puit doit connaitre, pour calculer correctement la clé de dechiffrement, les identités de tous les noeuds qui ont participé à l'agrégat. En d'autres mots, les capteurs qui ont participé doivent envoyer leurs identités. La transmission de ces identifiants peut avoir un coût non négligeable lorsque le réseau est peu stable et que le nombre de capteurs qui participent est très variable. Nous avons développé une optimisation qui permet de réduire le nombre de bits utilisés par l'émission des identifiants [2]. Cette optimisation peut réduire le coût de transmission des identifiants par un facteur pouvant aller jusqu'à 15.

5 Virus et Vers Informatiques pour les réseaux de capteurs

Les réseaux de capteurs se développant et faisant partie des infrastructures critiques, il est tout à fait naturel de penser à la menace que posent les virus et vers sur ces nouveaux réseaux.

Dans l'Internet, un attaquant peut compromettre des machines en exploitant des vulnérabilités, résultant souvent d'un dépassement d'un buffer en mémoire, permettant d'écrire dans la pile. Un capteur étant un mini-ordinateur, possédant un CPU, de la mémoire, des Entrées/Sorties, à priori, tout peut nous faire penser que ces types d'attaque y sont transposables.

Cependant, les capteurs ont plusieurs caractéristiques qui rendent leur compromision à distance par un virus très délicat :

- Les mémoires "programmes" et "données" sont souvent physiquement séparées (mémoire FLASH pour le programme, et mémoire SRAM pour les données et la pile). Il est alors souvent impossible d'éxécuter du code qui serait inséré dans la pile, comme c'est souvent le cas dans les attaques qui exploitent un dépassement de buffer pour ecraser la pile (Stack-based buffer overflow).

- Le code application est souvent protégé en écriture. Un attaquant ne peut pas modifier les programmes présents en mémoire.
- La taille des paquets que peut recevoir un capteur est très limitée (typiquement 28 octets), ce qui rend l'injection de code "utile" difficile.

Les techniques qu'utilisent les vers pour compromettre une machine sur Internet, ne peuvent donc pas être utilisées directement sur les capteurs. Nous avons cependant montré, en concevant un des premiers virus/vers pour capteurs de type Micaz/TinyOS, que la conception de virus, bien que difficile, n'est pas impossible.

Nous avons utilisé, pour arriver à notre objectif, deux propriétés que possèdent souvent les réseaux de capteurs :

- un réseau de capteurs est très souvent homogène, c'est à dire composé de dispositifs similaires, configurés avec les mêmes composants. La configuration mémoire de tous les capteurs est donc souvent identique. En compromettant un noeud, un attaquant peut facilement identifier le code présent en mémoire sur l'ensemble des noeuds du réseau.
- Chaque capteur doit souvent être reconfigurable à distance après déploiement, au cas ou un bug doit être corrigé ou un autre programme doit être chargé en mémoire. Cette reconfiguration est souvent réalisée par un logiciel (par exemple Deluge sous TinyOS), prélablement installé sur le capteur, qui copie le nouveau programme de la mémoire RAM externe vers la mémoire exécutable.

Le virus que nous avons conçu opère comme suit :

– une vulnerabilité dans le programme est exploitée en envoyant un paquet, formaté de façon adéquate, qui écrit, par un dépassement de buffer, dans la pile. Ce dépassement de buffer est utilisé pour exécuter par plusieurs groupes d'instructions qui vont copier un octet du paquet vers une zone mémoire inutilisée. Plus spécifiquement, le premier groupe d'instructions configure les registres (en utilisant les données qui sont dans la pile et qui ont été écrasées par le paquet lors du dépassement de la pile), qui vont permettre au deuxième groupe d'instructions de copier l'octet qui se trouve dans le paquet vers la position en mémoire qui aura été choisie. Le paquet doit être convenablement formaté afin de contenir les adresses des intructions à exécuter et les valeurs des registres à configurer.
– Le paquet précédent permet de copier un octet envoyé sur la mémoire donnée du capteur. En envoyant plusieurs paquets de ce type, nous pouvons créer en mémoire une fausse pile.
Cette fausse pile sera écrite dans une zone mémoire n'etant pas utilisée par le code (étant située au delà des zones *data* et *bss* et en dessous de la valeur maximum de la pile, cette zone n'est alors pas effacé lors du reboot). Cette fausse pile contient des données qui permettent de configurer les registres utilisés par les instructions appelées lors de l'étape suivante, ainsi que le code malveillant que l'on veut insérer dans le capteur.
– Lorsque la fausse pile est insérée en mémoire, il suffit alors d'envoyer un dernier paquet qui, en exploitant la même vulnérabilité, va : (1) exécuter un groupe d'instructions qui redirige le pointeur de pile (stack pointeur) vers la fausse pile, (2) lancer un groupe d'instructions qui configure les registres pour le groupe d'instructions suivant, qui (3) copie le code malicieux en mémoire exécutable.
– Un dernier paquet peut alors lancer l'exécution du code malicieux.

Il faut noter qu'après chaque étape, le capteur est rebooté en retournant à l'adresse exécutable 0. Le code malicieux peut lui-même lancer la même attaque sur les voisins du capteur compromis et transformer le virus en vers. Les détails de ce travail sont décrits dans un article qui est en cours de soumission [7].

6 Conclusions

La conception de réseaux de capteurs autonomes, reliés par des liens sans fil, est un domaine de recherche très actif. Ces capteurs sont, par exemple, utilisés dans des applications militaires, domotiques, ou de surveillance de l'environnement. Ces applications ont souvent besoin d'un niveau de sécurité élevé car ils fournissent des services essentiels, voire vitaux. Une équipe de recherche vient de montrer que certains équipements médicaux (défibrillateurs dans ce cas) ne contiennent aucun mécanismes de sécurité et peuvent donc être facilement abusés [6]. Une personne mal intentionné peut dérégler ou tout simplement arrêter le défibrillateur d'une victime s'il possède l'équipement de communication nécessaire. Ce type d'attaque peut, bien evidemment, être fatale pour la victime. Il devient donc urgent de se pencher sur les problèmes de sécurité et de protection de la vie privée qu'ils engendrent avant qu'ils envahissent nos vies et environnements.

Cet article a présenté quelques problèmes de sécurité et proposé quelques solutions. Nous avons décrit quelques protocoles de sécurité et des algorithmes de chiffrement qui minimisent le temps de calcul et l'énergie consommée. Nous avons également montrée que des vers/virus peuvent se propager dans ce type d environnement. Si ces réseaux se développent, il faudra développer des parades contre ces attaques.

La liste des problèmes mentionnés dans cette article est loin d'être exhaustive. Par exemple, dans beaucoup d'applications, il est nécessaire que le puit connaisse la localisation géographique des noeuds du réseau. Les protocoles de localisation qui existent ne guarantissent pas une localisation correcte en présence d'un attaquant actif. La sécurisation des protocoles de localisation est un sujet de recherche très difficile qui mérite reflexion. D'autres sujets importants sont la détection d'intrusion, la sécurisation des systèmes d'exploitation, la sécurisation des protocoles de mise à jour de codes, etc.

Références

1. C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor net works. In *MobiQuitous'05*, pages 1–9, July 2005.
2. Claude Castelluccia. Securing very dynamic groups and data aggregation in wireless sensor networks. In *IEEE MASS*, Pisa, Italy, October 2007.
3. C.Castelluccia and Angelo Spognardi. RoK : A robust key pre-distribution protocol for multi-stage wireless sensor networks. In *IEEE Securecomm*, Nice, France, September 2007.
4. Aldar Chan and Claude Castelluccia. On the security of concealed data aggregation. In *European Symposium On Research in Computer Security (ESORICS 2007)*, Dresden, Germany, September 2007.
5. Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proceedings of the 9th ACM conference on Computer and communica tions security, CCS, Washington, DC, USA*, November 2002.
6. Daniel Halperin et al. Pacemakers and implantable cardiac defibrillators : Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy*, May 2008.
7. Aurelien Francillon and Claude Castelluccia. How to corrupt a WSN. In *Submission*, April 2008.