Résistance contre les attaques par capture dans les réseaux de capteurs

Thomas Claveirole, Marcelo Dias de Amorim, Michel Abdalla, Yannis Viniotis

Résistance contre les attaques par capture dans les réseaux de capteurs

Thomas Claveirole[1], Marcelo Dias de Amorim[1], Michel Abdalla[2] et Yannis Viniotis[3]

[1] LIP6/CNRS
Université Pierre et Marie Curie – Paris VI
Paris, France
{claveiro,amorim}@rp.lip6.fr

[2] Département d'Informatique
École Normale Supérieure
Paris, France
michel.abdalla@ens.fr

[3] Department of ECE
North Carolina State University
Raleigh, NC, USA
candice@ncsu.edu

***Résumé*— Une approche courante pour surmonter les limitations des réseaux de capteurs est d'agréger les données au niveau des nœuds intermédiaires. Garantir une sécurité de bout en bout dans ce contexte est un problème difficile, en particulier parce que les capteurs sont extrêmement vulnérables à la capture et qu'il est facile de les compromettre. Afin de sécuriser l'agrégation des données, nous proposons trois techniques qui reposent sur le routage multi-chemins. La première garantit la confidentialité des données grâce à la *cryptographie à seuil*, tandis que la seconde et la troisième assurent la disponibilité des données grâce à la *dispersion d'information*. En se basant sur des analyses qualitatives et des implémentations, nous montrons qu'en appliquant ces techniques un réseau de capteurs peut obtenir la confidentialité de ses données, leur authentification ainsi qu'une protection contre les attaques du type déni de service, même en présence de plusieurs nœuds compromis.**

I. Introduction

Les réseaux de capteurs sans-fils sont des réseaux dédiés à la surveillance de paramètres physiques grâce à des nœuds-capteurs [1]. Ils ont des applications dans de nombreux domaines, dont l'observation de la nature et de l'environnement, la sécurité des bâtiments et la domotique, la gestion du trafic routier, la surveillance médicale, ou encore les opérations militaires.

Dans beaucoup d'applications des réseaux de capteurs, les données peuvent être menacées par des évènements extérieurs qui ne devraient pas arriver au cours du fonctionnement normal du réseau. En particulier, la confidentialité et la disponibilité des données sont des fonctionnalités importantes que le réseau devrait pouvoir assurer. Garantir de telles caractéristiques est une tâche difficile, surtout quand les nœuds sont constitués d'engins électroniques peu onéreux avec des capacités matérielles limitées.[1] Le cas échéant, utiliser des protections physiques est, dans beaucoup de situations, quasiment impraticable. Capturer des nœuds est alors une possibilité intéressante pour les attaquants.

Les contraintes inhérentes aux nœuds-capteurs offrent de multiples possibilités d'attaque. Dans la mesure où le coût des communications radio est élevé en terme de consommation d'énergie, il est très important de réduire la charge des communications.[2] À cette fin, une approche intéressante est d'*agréger les données*. C'est à dire exploiter la nature distribuée du réseau et effectuer des traitements au cours de l'acheminement des données. Garantir la sécurité conjointement à des techniques d'agrégation est difficile parce qu'un nœud capturé pose un double problème. Il compromet la confidentialité des données (possibilité d'écoute) et leur disponibilité (possibilité d'attaque du type déni de service). Également, un nœud d'agrégation compromis[3] met en danger toutes les mesures qui font parti de l'agrégat dont le nœud est responsable.

Plusieurs chercheurs ont déjà étudié le problème de la sécurité de l'agrégation des données. Mykletun *et al.* [3] suggèrent l'utilisation de méthodes de chiffrement pour lesquelles des opérations arithmétiques sur le texte chiffré correspondent à des opérations sur le texte clair. Bien que cette technique assure une certaine sécurité, un nœud compromis peut toujours arrêter d'agréger et d'acheminer les données. Pire que cela, cette solution ne peut pas détecter les falsifications et les attaques par rejeu. Przydatek *et al.* [4] proposent un ensemble de techniques pour s'assurer de l'intégrité des données pour certaines fonctions d'agrégation. Bien que l'intégrité puisse être assurée avec succès, les méthodes proposées sont difficiles à implémenter et n'assurent ni la confidentialité ni la protection contre les dénis de service. Hu et Evans [5] proposent une technique qui fournit une authentification et une vérification d'intégrité, même en présence de quelques nœuds compromis. En revanche, elle est inefficace lorsque deux agrégateurs successifs sont compromis. Également, cette technique ne considère pas la confidentialité et la disponibilité. Wagner [6] étudie la sécurité inhérente de certaines fonctions d'agrégations. Mais il ne considère que l'impact qu'un capteur compromis peut avoir sur le résultat final. Son travail concerne la sécurité des fonctions d'agrégation, pas la sécurité de l'agrégation en elle-même.

Dans cet article, nous ne considérons pas l'intégrité des données comme un problème de premier plan. À la place, nous nous concentrons sur la confidentialité et la disponibilité, pour

[1] Il est important de noter que les nœuds d'un réseau de capteurs n'ont pas nécessairement des ressources limitées, mais la plupart des problèmes deviennent particulièrement ambitieux dans ce cas.

[2] Transmettre 1Ko à une distance de 100 mètres demande autant d'énergie qu'exécuter 3 millions d'instructions avec un processeur généraliste [2].

[3] C'est à dire qu'un attaquant a accès à l'état interne et aux secrets cryptographiques du nœud.

lesquels nous pensons qu'il manque des solutions efficaces. À cette fin, nous proposons, analysons et évaluons trois nouvelles techniques, nommément (a) *Agrégation Multi-chemins Secrète (*AMS*)*, (b) *Agrégation Multi-chemins Dispersée (*AMD*)*, et (c) *Agrégation Multi-chemins Dispersée avec Authentification (*AMD-A*)*. L'idée principale de ces approches est d'exploiter plusieurs chemins jusqu'à la station de base. En fait, un capteur peut séparer ses mesures en n messages distincts tels que t messages soient nécessaires pour reconstruire les mesures. En envoyant chacun des messages sur des chemins disjoints, un capteur peut s'assurer que les nœuds intermédiaires n'auront pas une connaissance complète des données. Dans un tel scénario, AMS garantit la confidentialité des données grâce à la cryptographie à seuil [7]. AMD et sa version authentifiée, AMD-A, s'intéressent à la disponibilité en dispersant l'information à travers les différents chemins [8]. Bien que reconnues dans beaucoup de domaines de recherche (par exemple le calcul parallèle, le stockage distribué, et les bases de données), étonnamment ni la cryptographie à seuil ni la dispersion d'information n'ont été appliquées dans le contexte des réseaux de capteurs ou dans celui de l'agrégation des données.

Le reste de cet article est organisé comme suit. Dans la section II, nous décrivons nos hypothèses concernant la sécurité et le réseau. Dans la section III, nous introduisons les techniques proposées. Dans la section IV, nous analysons leurs niveaux de sécurité. Dans la section V, nous fournissons une étude plus poussée des trois techniques et les comparons à d'autres approches. Enfin, en section VI, nous concluons notre article et présentons quelques questions qui restent ouvertes.

II. Formulation du problème

Par la suite, nous décrivons les problèmes, objectifs, et hypothèses que cet article considère. Cette section est composée de trois parties : (a) aspects de sécurité, (b) hypothèses sur le réseau, et (c) hypothèses sur les nœuds.

A. Menaces et objectifs de sécurité

L'objectif de cet article est de fournir des techniques d'agrégation qui résistent à la capture des nœuds. C'est à dire qu'un nœud compromis ne devrait pas avoir seul le pouvoir d'écouter des données, de falsifier des messages, ou d'empêcher les autres nœuds d'accéder aux données. Lorsqu'aucun nœud n'est compromis, cet article suppose que des techniques de niveau liaison peuvent assurer la protection contre ces attaques [9], [10].

Même un unique agrégateur capturé représente une menace sérieuse pour la sécurité d'un réseau de capteurs. En conséquence, il faut concevoir des techniques qui assurent une sécurité raisonnable en présence d'agrégateurs compromis. Idéalement, il faudrait que la sécurité du réseau se dégrade progressivement au fur et à mesure que des nœuds sont compromis. Dans cet article, sécurité signifie résistance contre les attaques suivantes : écoute passive, falsification de données, injection de paquets, et déni de service. Les autres attaques ne sont pas le propos de cet article.

Écoute passive. Une écoute passive se produit lorsqu'un attaquant capture un nœud et étudie le trafic qui le traverse sans en altérer le fonctionnement. Puisqu'un nœud d'agrégation traite des données venant de plusieurs nœuds dans le réseau, les informations qu'il envoie ne concernent pas uniquement ce nœud, mais plutôt un groupe de nœud.

Falsification de données et injection de paquets. Un nœud compromis peut modifier les paquets qui le traversent. Il peut aussi envoyer de faux messages. Puisqu'un message qui agrège des données embarque des informations qui concernent un ensemble de capteurs, il est plus intéressant pour un attaquant de falsifier de tels messages plutôt que la simple mesure d'un capteur. Un attaquant qui contrôle la signification des messages qu'il envoie peut avoir un impact lourd sur le résultat final calculé par la station de base (le puit).

Un attaquant qui ne contrôle pas la signification des messages falsifiés (par exemple, si le message est chiffré avec une clef inconnue de l'attaquant) peut tout de même faire des dégâts. Il peut envoyer du bruit sans signification (des déchets) et rendre le réseau inutilisable — cela peut être vu comme une forme de déni de service. Également, un type particulier d'injection de paquet est l'attaque par rejeu, au sein de laquelle un nœud mal-intentionné écoute des messages dans le but de les ré-envoyer plus tard.

Déni de service. Un nœud compromis peut arrêter d'agréger et d'acheminer les données. Ce faisant, il empêche la station de base de récupérer des informations au sujet de plusieurs nœuds dans le réseau. Si l'attaquant continue d'échanger des messages de routage en dépit de son comportement malhonnête, ce problème peut se révéler difficile à résoudre. De cette façon, un agrégateur mal-intentionné peut rendre le réseau inutilisable. Des attaques plus évoluées peuvent aussi ne jeter que certains messages d'une façon aléatoire. Il est également difficile de détecter quand un attaquant envoie des déchets. Finalement, il est important de réaliser que de telles attaques ne nécessitent pas forcément un coût élevé ou des compétences techniques avancées. Par exemple, une attaque primitive consisterait simplement à détruire le capteur physiquement.

B. Hypothèses sur le réseau

Nous supposons que chaque capteur dispose de plusieurs chemins vers la station de base et possède la capacité de chiffrer ses liens. Un nœud peut alors séparer un flux en plusieurs sous-flux et diriger chacun d'entre eux de manière sécurisée vers la station de base. Grâce au chiffrement, un nœud ne peut pas écouter un sous-flux à moins qu'il ne fasse parti du chemin pour ce sous-flux.

Pour obtenir plusieurs chemins vers la station de base, des solutions sont d'utiliser un protocole de routage multi-chemins ou de disperser géographiquement plusieurs stations de bases reliées entre elles par des liens rapides et sécurisés. Ganesan *et al.* [11] étudient l'établissement de plusieurs chemins dans un réseau de capteurs. Dulman *et al.* [12] explorent les relations existantes entre la fiabilité d'un réseau et la quantité de trafic

sur les chemins. Il faut noter que les techniques décrites par cet article requièrent des chemins multiples disjoints. Des chemins non-disjoints peuvent être utilisés, mais alors il n'est plus possible de garantir une sécurité optimale.

Nous supposons également que le protocole de routage sous-jacent est sûr. En particulier, il faut faire attention aux usurpations d'identité et aux « attaques Sybil » [13]. Grossièrement, cela signifie qu'un nœud ne devrait pas être capable d'usurper l'identité d'un autre nœud ou de prétendre avoir des identités multiples. Cela ne devrait néanmoins pas être un problème si les clefs de chiffrement des liens sont uniques pour chaque lien.

C. Hypothèses sur les capacités de calcul, de mémoire et de stockage des nœuds

Nous supposons que les nœuds ont des capacités de calcul, de mémoire et de stockage très limitées. Cela rend beaucoup de protocoles et d'algorithmes cryptographiques difficile à pratiquer, sinon impossible. Les techniques proposées dans cet article ont été conçues pour s'adapter à de telles contraintes.

Nous avons implémenté ces techniques en utilisant des capteurs Crossbow MICAz [14]. Ils utilisent un micro-contrôleur Atmel ATmega128L (CPU 8 bit à 8 MHz) avec 4 Ko de RAM et 128 Ko de mémoire flash pour stocker le code et les données pré-calculées. Leur énergie est fournie par deux piles AA ($\sim$ 3V). Ils communiquent à l'aide d'un trans-receveur radio IEEE 802.15.4 à 2,4 GHz.

III. Résistance à la capture : techniques proposées

Préliminaires. Dans cette section nous présentons trois techniques pour sécuriser l'agrégation dans les réseaux de capteurs : *Agrégation Multi-chemins Secrète* (AMS), *Agrégation Multi-chemins Dispersée* (AMD), et *Agrégation Multi-chemins Dispersée avec Authentification* (AMD-A). Chacune de ces techniques a ses caractéristiques propres. AMS offre une forte confidentialité mais c'est aux prix d'une surcharge de communication. AMD est optimal vis à vis des communications mais offre un niveau de confidentialité légèrement moins fort. AMD-A ajoute l'authentification à AMD, mais avec un léger surcoût en communication. L'utilisation d'une technique plutôt qu'une autre dépend du scénario de l'application. Toutes ces propriétés sont analysées et quantifiées dans les sections IV et V.

Bases. Les trois techniques proposées utilisent le même principe de base : un capteur découpe ses mesures en plusieurs *parts* et envoie ces parts à travers différents chemins. Chaque part est acheminée jusqu'à la station de base. Durant son transport, une part peut être manipulée par les agrégateurs. Une fois que la station de base a récupérée suffisamment de parts, elle peut reconstruire un ensemble précis de mesures. Toutefois, une unique part n'est pas intelligible pour un nœud intermédiaire. La figure 1 présente ceci.

Tolérance aux pertes. La manière dont les parts sont construites dépends de la technique utilisée (par exemple, AMS ne code qu'une seule mesure par part tandis que AMD et AMD-A codent plusieurs mesures par part). Les nombres de parts transmises et de parts nécessaires pour la reconstruction ne sont pas nécessairement égaux. Cela signifie que le système peut tolérer que des parts soient perdues durant leur acheminement.

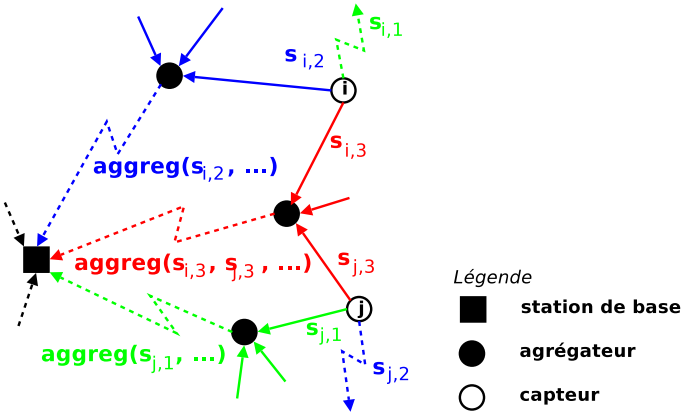


Fig. 1. *Principes de base des techniques proposées.* Les nœuds i et j découpent respectivement leurs mesures dans les parts $s_{i,1}, s_{i,2}, s_{i,3}$ et $s_{j,1}, s_{j,2}, s_{j,3}$. Les parts sont ensuite envoyées et agrégées sur différents chemins.

Implications sur la sécurité. Les propriétés précédemment mentionnées ont deux implications intéressantes du point de vue de la sécurité. D'abord, un attaquant doit compromettre plusieurs nœuds pour être capable de reconstituer les mesures. Cela assure la confidentialité. Ensuite, les nœuds malintentionnés qui arrêtent d'acheminer les parts n'ont qu'un impact limité sur le système, parce qu'un autre sous-ensemble de parts peut toujours être utilisé pour reconstruire les mesures. Cela assure la protection contre les dénis de service.

Homomorphisme. Un point clef de ces techniques est leurs propriétés homomorphiques. C'est à dire qu'un agrégateur a la possibilité d'effectuer des calculs sur les parts malgré leur signification inconnue. Supposons que les nœuds i et j mesurent les valeurs r_i et r_j. Un nœud d'agrégation peut additionner deux parts provenant de i et j, cette somme correspondant à une nouvelle part qui code la valeur $r_i + r_j$. Cela est vrai pour un certain nombre de fonctions d'agrégation, comme par exemple la somme, la moyenne, la variance, et le comptage [3], [6].

A. Technique 1 : Agrégation Multi-chemins Secrète (AMS)

AMS utilise la cryptographie à seuil pour créer les parts, ce qui est une approche courante lorsqu'il faut assurer une sécurité malgré des possibilités de capture.

Création des parts. Supposons qu'un nœud i dispose de p chemins distincts vers la station de base, parmi lesquels on tolère que $t-1$ d'entre eux soit compromis (*c.-à-d.* qu'un nœud doit disposer d'au moins t parts pour reconstruire les mesures). Lorsqu'il mesure une valeur r_i, un nœud-capteur tire aléatoirement un polynôme $P_i(x)$ de degré $t-1$ tel que $P_i(0) = r_i$. Un tel polynôme peut se construire en tirant

aléatoirement ses coefficients d'ordre plus grand que zéro : $a_{i,k}$, $\forall k \in [1, t-1]$ en utilisant ensuite $P_i(x) = r_i + a_{i,1}x + a_{i,2}x^2 + \ldots + a_{i,t-1}x^{t-1}$. C'est une opération simple et très abordable. Chacune des p parts est ensuite constituée par les valeurs $P_i(q)$ ($1 \leq q \leq p$). Le nœud i envoie ensuite un message contenant $P_i(q)$ sur chaque chemin q.

Reconstruction. Afin de récupérer r_i, il faut d'abord retrouver P_i grâce à une interpolation polynomiale et ensuite calculer $r_i = P_i(0)$. Cette opération nécessite au moins t parts distinctes. Il y a une infinité de polynômes de degré $t-1$ qui passent à travers $t-1$ points donnés. Dès lors, $t-1$ nœuds compromis ne peuvent déduire quoi que ce soit au sujet de P_i et r_i. Également, la station de base peut tolérer jusqu'à $p-t$ nœuds muets et être toujours capable de reconstituer r_i. En conséquence, cette technique fournit la confidentialité et une robustesse contre les dénis de service, même en présence de quelques nœuds mal-intentionnés.

Agrégation de données. Supposons qu'un nœud d'agrégation le long d'un chemin q doive fusionner les mesures des nœuds i et j, soit $r_i = P_i(0)$ et $r_j = P_j(0)$. Étant sur le chemin q, les seules données qu'il reçoit sont $P_i(q)$ et $P_j(q)$. Il retransmet $P_i(q) + P_j(q) = (P_i + P_j)(q)$. La même opération est effectuée sur les autres parts de ces nœuds sur les différents chemins. En recevant t échantillons, la station de base peu alors reconstruire $P_i + P_j$ et ensuite $(P_i + P_j)(0) = r_i + r_j$. Ce résultat est aussi valable pour la multiplication et la division par un scalaire.

Discussion. Grâce aux propriétés inhérentes de la cryptographie à seuil, AMS offre une confidentialité très forte. Un attaquant qui n'as pas récupéré au moins t parts ne peut rien deviner au sujet des mesures des capteurs. La confidentialité assurée par AMS est obtenue au coût d'une surcharge dans les transmissions radio et donc dans la consommation d'énergie. Lorsqu'un évènement est mesuré, p messages doivent être envoyés, chacun d'entre eux étant de la même taille que la mesure d'origine.[4] C'est la raison principale pour laquelle nous proposons les deux autres techniques (AMD et AMD-A).

B. Technique 2 : Agrégation Multi-chemins Dispersée (AMD)

La dispersion d'information est une technique couramment utilisée pour introduire de la redondance et se protéger du problème des généraux Byzantins. Comme pour la cryptographie à seuil, il s'agit d'une technique qui produit p parts à partir d'une donnée particulière, de telle sorte que t d'entre elles soient suffisantes pour reconstruire la donnée. Contrairement à la cryptographie à seuil, un bloc de données de taille t est découpé en au moins t pièces de tailles unitaires.[5]

Création des parts. Chaque capteur est pré-équipé avec la même matrice $\mathbf{A} = [a_{i,q}]$ de taille $t \times p$. $\mathbf{A}$ doit être choisie de manière à ce que chaque combinaison de t colonnes forme un matrice de taille $t \times t$ inversible. Lorsqu'il mesure des évènements, un capteur i accumule ses mesures dans un vecteur de taille t, $\mathbf{R}_i = \left[\begin{array}{cccc} r_{i,1} & r_{i,2} & \ldots & r_{i,t} \end{array}\right]$. Cela forme un bloc de mesures. Une fois que le tampon est plein, le nœud est prêt à calculer p parts différentes de taille unitaire à envoyer le long des chemins. Ces parts sont les différents éléments de $\mathbf{M} = \mathbf{R}_i.\mathbf{A}$:

$$\left[\begin{array}{cccc} m_{i,1} & m_{i,2} & \ldots & m_{i,p} \end{array}\right] = \left[\begin{array}{cccc} r_{i,1} & r_{i,2} & \ldots & r_{i,t} \end{array}\right] \cdot \left[\begin{array}{ccc} a_{1,1} & \cdots & a_{1,p} \\ \vdots & \ddots & \vdots \\ a_{t,1} & \cdots & a_{t,p} \end{array}\right]. \quad (1)$$

C'est à dire :

$$m_{i,q} = r_{i,1}a_{1,q} + r_{i,2}a_{2,q} + \ldots + r_{i,t}a_{t,q}. \quad (2)$$

Reconstruction. Lorsque la station de base reçoit t parts, elle peut reconstruire les données. Lorsqu'elle reçoit $\mathbf{M}_i = \left[\begin{array}{cccc} m_{i,q_1} & m_{i,q_2} & \ldots & m_{i,q_t} \end{array}\right]$, les mesures sont reconstruites en résolvant :

$$\left\{\begin{array}{ccc} r_{i,1}a_{1,q_1} + r_{i,2}a_{2,q_1} + \ldots + r_{i,t}a_{t,q_1} = m_{i,q_1} \\ r_{i,1}a_{1,q_2} + r_{i,2}a_{2,q_2} + \ldots + r_{i,t}a_{t,q_2} = m_{i,q_2} \\ \vdots \quad \vdots \quad \vdots \\ r_{i,1}a_{1,q_t} + r_{i,2}a_{2,q_t} + \ldots + r_{i,t}a_{t,q_t} = m_{i,q_t} \end{array}\right. \quad (3)$$

Cela peut être fait avec une simple élimination de Gauss ou bien en inversant la matrice constituée des différentes colonnes $q_1, \ldots, q_t$ de $\mathbf{A}$. Si la matrice $\mathbf{A}$ est tirée aléatoirement, il n'existe pas de méthode connue pour reconstruire même partiellement les données originelles depuis $t-1$ échantillons, même si l'on peut déduire des corrélations entre les différents $r_{i,q}$.

Agrégation des données. Cette technique possède des propriétés homomorphiques similaires à celles de la cryptographie à seuil. Soient les messages $m_{i,q}$ et $m_{j,q}$ envoyés par les nœuds i et j sur le chemin q. Un nœud d'agrégation calcule $m_{i,q} + m_{j,q}$.

Puisque l'on a :

$$m_{i,q} + m_{j,q} = \sum_{k=1}^{t} (r_{i,k} + r_{j,k})a_{k,q}, \quad (4)$$

alors, après réception d'au moins t messages de la sorte, la station de base est en mesure de reconstruire tous les $r_{i,k} + r_{j,k}$ d'une manière similaire à l'équation 3 :

$$\left\{\begin{array}{ccc} (r_{i,1} + r_{j,1})a_{1,q_1} + \ldots + (r_{i,t} + r_{j,t})a_{t,q_1} = m_{i,q_1} + m_{j,q_1} \\ (r_{i,1} + r_{j,1})a_{1,q_2} + \ldots + (r_{i,t} + r_{j,t})a_{t,q_2} = m_{i,q_2} + m_{j,q_2} \\ \vdots \quad \vdots \quad \vdots \\ (r_{i,1} + r_{j,1})a_{1,q_t} + \ldots + (r_{i,t} + r_{j,t})a_{t,q_t} = m_{i,q_t} + m_{j,q_t} \end{array}\right. \quad (5)$$

Discussion. Cette technique est efficace vis à vis de la taille, c'est à dire que reconstruire t mesures ne nécessite que t

[4] C'est un résultat bien connu de la cryptographie à seuil qui se démontre facilement grâce à la théorie de l'information.

[5] Afin d'éviter les confusions : une taille de 1 ne signifie pas 1 bit, mais un bloc de bit de taille unitaire. Cette taille dépend de la taille d'un agrégat.

parts dont la taille totale est identique. Il est toutefois possible d'utiliser plus de parts ($p > t$) pour permettre de se protéger contre les dénis de service.

Bien que cette méthode soit plus efficace que AMS en terme de communications, il faut garder à l'esprit qu'elle offre une confidentialité légèrement moins importante. Capturer un nœud permet à l'attaquant de récupérer une certaine quantité d'information sur les mesures, même s'il n'est pas capable de faire une reconstruction partielle. Cela fournit néanmoins une confidentialité suffisante pour un réseau de capteurs. Il est donc possible d'utiliser cette méthode pour assurer une confidentialité « souple » ainsi qu'une protection contre les dénis de service. Il faut noter également qu'elle ne nécessite aucune opération lourde ; seulement la station de base doit résoudre un système d'équations.

*C. Technique 3 : Agrégation Dispersée Multi-chemins avec Authentification (*AMD-A*)*

AMS et AMD tels que présentés précédemment n'assurent ni la protection contre les attaques par rejeu ni l'authenticité des données. Un attaquant mal-intentionné peut écouter les messages d'un capteur puis les dupliquer plus tard pour en tirer un avantage. Il peut aussi envoyer des messages aléatoires sans signification mais rester non-détecté. Bien sûr, la station de base peut détecter de telles attaques en effectuant plusieurs reconstructions avec des ensembles de parts différents et en remarquant que les résultats sont différents. Mais elle ne sera pas capable de détecter laquelle des reconstructions est correcte.

Authentification. Il existe des techniques qui permettent de rajouter l'authentification à la cryptographie à seuil mais elles sont actuellement impraticables dans les réseaux de capteurs — parce qu'elles nécessitent la publication de beaucoup de valeurs supplémentaires (appelées des engagements). Pour cette raison, nous nous focalisons sur une solution d'authentification pour AMD. Nous proposons de remplacer la dernière mesure de $\mathbf{R}_i$ avec un élément qui inclut des informations de séquence et qui dépend d'un secret partagé entre i et la station de base. Soit $\mathbf{R}_i = \left[\begin{array}{cccc} r_{i,1} & \dots & r_{i,t-1} & h(k_i, s) \end{array} \right]$, avec $h(\cdot)$ une fonction de hachage sûre modélisée comme un oracle aléatoire [15], k_i une clef secrète entre i et la station de base, et s un numéro de séquence.

Reconstruction. Après la reconstruction de $\sum \mathbf{R}_i$, la station base a seulement besoin de vérifier que le dernier élément reconstitué est égal à $\sum h(k_i, s)$. Si ce n'est pas le cas, alors un nœud d'agrégation est en train de mentir et il faut utiliser un autre sous-ensemble de parts pour reconstruire $\sum \mathbf{R}_i$. Au sens strict, ce n'est pas vraiment une vérification d'intégrité, parce que les valeurs d'authentification $h(k_i, s)$ ne dépendent pas des informations des mesures $r_{i,k}$. En conséquence, un attaquant qui a compromis t nœuds peut être capable de reconstruire $h(k_i, s)$ et de falsifier les données sans être perçu. Mais il n'est pas possible d'utiliser des informations d'authentification venant de $r_{i,k}$ parce que la station de base ne connaît pas chaque $r_{i,k}$ indépendamment : elle ne reconstruit que $\sum r_{i,k}$. On peut remarquer que l'utilisation de la valeur s comme seul test d'intégrité pourrait être suffisante en pratique. Nous préférons néanmoins utiliser une clef secrète k_i et une fonction de hachage pour rendre les valeurs d'authentification moins prévisibles. Cela complique la tâche de l'attaquant pour falsifier les données et permet donc d'offrir un meilleur niveau de sécurité en pratique.

Cette technique offre un gros avantage par rapport au petit coût généré. Seul un champ du vecteur est utilisé pour des informations de contrôle, soit $\frac{1}{t}$ de l'information totale contenue dans le vecteur. Lorsque t est grand, on minimise donc le surcoût de cette technique par rapport à AMD. Au besoin, on peut générer plus de parts qu'il y a de chemins disponible (pour augmenter t), et envoyer plusieurs parts sur chaque chemin.

D. Résumé et discussion

AMS découpe chaque mesure en un nombre donné de parts et envoie ensuite une part par chemin. AMD accumule plusieurs mesures dans une mémoire interne avant de la disséminer dans plusieurs parts. Elle envoie une part par chemin. AMD-A accumule plusieurs mesures dans une mémoire interne, puis ajoute une valeur d'authentification à cette mémoire avant de la disséminer dans plusieurs parts. Éventuellement, cela génère plus de parts que le nombre de chemin disponibles.

Chacune de ces techniques possède des avantages et des inconvénients. Certains d'entre eux sont globaux pour toutes les techniques, tandis que d'autres sont spécifiques. D'abord, toutes les techniques offrent une résistance contre les échecs non-intentionnels et les attaques de dénis de service. Ensuite, toutes ces techniques cachent (une certaine quantité) des données aux nœuds d'agrégation, de telle sorte qu'il ne soit pas possible à quelques nœuds compromis de reconstruire les mesures, au moins complètement.

AMS procure une confidentialité totale, *i.e.* strictement aucune information ne peut être obtenue des parts, au sens de la théorie de l'information, à moins que t d'entre elles soient réunies. Dans ce cas, la sécurité s'effondre complètement. Cette confidentialité forte est toutefois obtenue au prix de la duplication de chaque mesure pour chaque chemin. D'un autre coté, AMD et AMD-A sont plus efficaces, mais chaque part collectée donne un peu plus d'information sur les mesures. Malgré tout, il n'existe pas de technique pour reconstruire, même partiellement, les mesures depuis $t - 1$ parts.

IV. ANALYSE DE LA SÉCURITÉ

Rappelons que des mécanismes appropriés dans les couches inférieures peuvent protéger un réseau en cas d'absence de nœuds compromis [16]. La protection contre l'écoute peut se faire avec un chiffrement des liaisons. La falsification des données et l'injection de paquets sont également inefficaces lorsqu'elles sont confrontées à une authentification et un chiffrement de niveau liaison. Et il existe des techniques de niveau physique et des protocoles de routage qui permettent de contourner les dénis de service. Mais aucune de ces techniques n'est efficace contre les nœuds compromis.

Dans notre cas, l'analyse de la sécurité doit donc se faire comparativement au nombre de nœuds compromis. Chaque technique dissémine les données mesurées le long de plusieurs chemins. La plupart du temps, capturer un unique nœud est insuffisant pour réussir une attaque : il y a un seuil qui détermine quel est le nombre minimum de nœuds qu'un attaquant doit capturer pour réussir ses attaques. Trois questions fondamentales sont :

1) Dans le pire des cas, combien de nœuds un attaquant doit-il capturer pour écouter avec succès et ainsi casser la confidentialité ? Quels sont les nœuds qu'il faut attaquer ?
2) Quel est le nombre minimal de nœuds qu'un attaquant a besoin de capturer pour injecter de fausses données dans le réseau ? Quels nœuds faut-il choisir ?
3) Combien de nœuds faut-il capturer dans le pire des cas pour réussir une attaque de déni de service ?

Il est important de souligner qu'un attaquant ne possède pas nécessairement le choix des nœuds qu'il va attaquer. En pratique, si n nœuds doivent être capturés pour réussir une attaque, peut-être que l'attaquant n'a pas accès à ces n nœuds. Également, si l'attaquant n'a pas une connaissance complète de la topologie du réseau, il peut lui être difficile de détecter les nœuds intéressant à capturer. Il peut donc être nécessaire qu'un attaquant ait besoin de compromettre plus de nœuds que le seuil théorique de chaque technique.

Dans l'analyse suivante, nous considérons sans perte de généralité qu'une part seulement est envoyée par chemin.

A. *Écoute passive*

Dans cette section nous étudions la résistance des techniques proposées aux captures lorsqu'il faut faire face à des écoutes passives (*c.f.* section II-A).

Il apparaît clairement en regardant les techniques proposées qu'au moins t nœuds doivent être compromis pour être capable de reconstruire les données. Néanmoins il subsiste quelques subtilités. Tout d'abord, si chaque part de AMS ne fournit aucune information jusqu'à ce que t d'entre elles soient réunies, les choses sont plus compliquées pour AMD et AMD-A. Ensuite, rien ne garantit qu'en choisissant t nœuds sur des chemins distincts un attaquant puisse reconstruire tout de même les données. Voici une explication de ces phénomènes.

1) Fuites d'information de AMD *et* AMD-A *:* Selon la théorie de l'information, comme la dispersion d'information est efficace vis à vis de la taille et comme chacune des p parts joue un rôle complètement symétrique, chaque part contient exactement $\frac{1}{t}$ de l'information contenue dans les mesures. En conséquence chaque part renseigne une certaine quantité d'information lorsqu'elle est récupérée par un attaquant. Il n'existe pas de méthode connue, néanmoins, pour reconstruire partiellement les mesures depuis un sous-ensemble de moins de t parts.

2) Possibilités de reconstruction des données à partir de t chemins compromis : Pour que l'opération de reconstruction fonctionne correctement, les parts pour chaque agrégat doivent contenir des contributions venant des mêmes nœuds. Par contre, les parts se propagent selon des chemins différents, et au sein de ces chemins, les nœuds d'agrégation reçoivent des contributions de nœuds probablement différents. Cela rend les attaques d'écoute difficiles à implémenter.

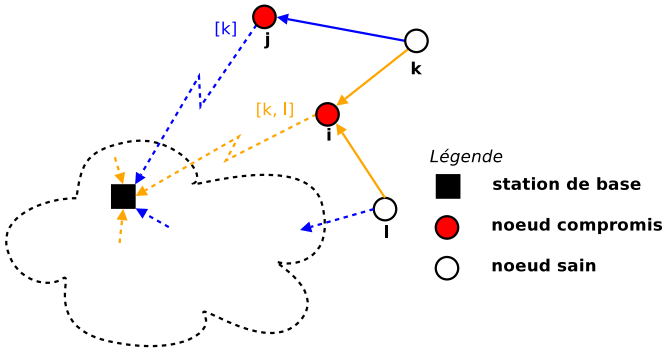


Fig. 2. Exemple de cas dans lequel compromettre t noeuds ne permet pas de reconstruire des données. Ici $t = 2$ et les noeuds i et j sont compromis. Il n'est cependant pas possible de reconstruire les mesures des noeuds k et l.

La figure 2 présente un exemple. Imaginons que $t = 2$ et qu'un attaquant a réussi à compromettre deux nœuds i et j sur deux chemins différents. Si les parts collectées par i contiennent des contributions de k et l et que les parts collectées par j ne contiennent que des contributions du nœud k, alors aucune reconstruction ne sera possible. Pourtant, l'attaquant a bien compromis deux chemins distincts. Si les parts collectées par j avaient contenu des contributions des deux nœuds k et l alors les reconstructions auraient été possibles. Notons que la station de base se situe à l'extrémité de tous les chemins, et qu'elle reçoit donc pour chaque part les contributions de tous les noeuds. Elle ne souffre donc pas de ce problème lors de ses reconstructions.

B. *Falsification de données et injection de paquets*

Dans cette section nous analysons la résistance des techniques proposées aux captures face à des attaques de falsification de données et d'injection de paquets (*c.f.* section II-A).

Il est clair qu'un attaquant qui a capturé moins de t nœuds d'agrégation ne pourra pas contrôler effectivement la signification des données qu'il injectera dans le réseau. Même si l'attaquant arrive à compromettre t nœuds, rien ne lui garantit que la station de base va utiliser exactement les informations de ces t nœuds pour lancer ses reconstructions (elle peut aussi utiliser des parts venant de chemins non-compromis). On peut aussi imaginer un scénario où l'attaquant compromet un ou plusieurs nœuds sur chaque chemin. Même dans ce cas, il est possible que l'attaquant ne puisse pas contrôler la signification de ses injections, pour les raisons décrites en IV-A.2.

Enfin, comme AMD-A fournit un test d'authenticité, un attaquant qui n'est pas capable de reconstruire les mesures (et donc les valeurs d'authentification pour chaque séquence) sera vraisemblablement incapable de tromper la station de base avec de fausses données. Ceci parce que (a) l'attaquant n'a pas connaissance des valeurs d'authentification attendues, et

(b) il va lui être difficile d'injecter des parts qui vont modifier les mesures sans modifier la valeur d'authentification lors des reconstructions.

Même lorsqu'il n'y a pas d'authentification, la station de base peut détecter des données falsifiées en testant plusieurs reconstructions mais avec des ensembles de parts différents et en constatant des résultats différents. Dans ce cas, néanmoins, elle ne pourra pas détecter si l'un des sous-ensembles utilisé est valide.

Un attaquant qui ne contrôle pas le sens des données falsifiées peut au mieux essayer d'effectuer une sorte de déni de service. C'est à dire qu'il peut essayer de falsifier suffisamment de messages pour rendre les reconstructions impossibles. Dans ce cas, les paramètres de sécurité se comportent comme dans la section suivante.

C. Déni de service

Dans cette section nous analysons la résistance des techniques proposées face à des attaques de dénis de service (*c.f.* section II-A).

Il y a deux types d'attaques par déni de service : celles où les attaquants cessent d'émettre des données (appelons les DdS sans données) et celles où ils émettent des données aléatoires sans signification (appelons les DdS avec données-déchets). Les DdS avec données-déchets sont plus difficiles à gérer. En l'absence d'authentification, un attaquant a juste besoin de compromettre un seul chemin et d'envoyer ses données malicieuses dessus. Dans ce cas la station de base aura plusieurs valeurs de reconstruction possibles mais ne pourra pas savoir lesquelles sont valides. En présence d'authentification, les DdS avec données-déchets ont les même propriétés que les DdS sans données — les reconstructions invalides seront rejetées comme si la part n'était jamais arrivée.

Les DdS sans données et les DdS avec données-déchets en présence d'authentification ont besoin d'empêcher la station de base de collecter t parts valides. En conséquence, un attaquant a besoin de compromettre au moins $p-t+1$ chemins distincts, c'est à dire en pire cas $p-t+1$ nœuds. Si l'attaquant ne connaît pas la topologie mise en place par le routage, il n'a pas d'autre choix que de compromettre les nœuds aléatoirement. En conséquence, il lui faudra probablement capturer plus de $p-t+1$ nœuds.

Soient respectivement t_e et t_d les nombres minimaux de nœuds requis pour écouter les communications et pour réussir un déni de service. On sait des sections précédentes que $t_e = t$ et $t_d = p-t+1$. Remarquez que plus t_e est grand, plus t_d sera petit, et réciproquement. Il est possible de faire un compromis en choisissant $t \approx \frac{p+1}{2}$. N'importe quelle valeur plus grande permettra de mieux résister aux écoutes passives tandis que n'importe quelle valeur plus petite permettra de mieux résister aux dénis de service.

La table I récapitule et donne les bornes inférieures sur le nombre de nœuds capturés requis par un attaquant pour réussir les différentes attaques décrites précédemment.

TABLE I

BORNES INFÉRIEURES SUR LE NOMBRE DE NŒUDS CAPTURÉS REQUIS POUR RÉUSSIR DIFFÉRENTES ATTAQUES.

	Limite	Remarque
Écoute passive	t	Les parts compromises doivent avoir les même nœuds contributeurs.
Falsification⋆	t	Les parts compromises doivent avoir les même nœuds contributeurs.
Déni de service	$p-t+1$	1 pour les DdS avec données-déchets pour AMS ou AMD

⋆ Seules les falsifications dans lesquelles l'attaquant contrôle la signification des données falsifiées sont considérées.

TABLE II

FONCTIONNALITÉS DES TECHNIQUES D'AGRÉGATION SÉCURISÉES.

Technique	Protection contre… écoute	falsification	dénis de service
arbre simple	non	non	non
Mykletun et al. [3]	oui	faible⋆	non
Hu and Evans [5]	non	oui	non
AMS	oui	faible⋆	faible⋆⋆
AMD	oui	faible⋆	faible⋆⋆
AMD-A	oui	oui	oui

⋆ Un attaquant peut altérer des données mais pas contrôler la signification des données modifiées.
⋆⋆ L'attaque peut réussir si l'attaquant envoie des « données-déchets »

V. AUTRES ANALYSES

Dans cette section nous commençons par présenter d'autres approches et les comparons avec nos techniques du point de vue du coût supplémentaire en communications et de la résistance aux attaques. Ensuite, nous présentons les détails de notre implémentation et quelques résultats de simulation.

A. Comparaison avec d'autre approches

L'approche non sécurisée classique au sujet de l'agrégation est d'avoir un arbre unique qui joint tous les nœuds du réseau. Chaque nœud interne de l'arbre agrège les données de ses fils avant de retransmettre vers son parent. Avec un seul message par nœud, c'est la technique la plus efficace pour les communications, malgré son absence de sécurité. Dans la suite, nous appellerons cette méthode « arbre simple ». Sans coût supplémentaire, il est possible d'utiliser des techniques de chiffrement spéciales qui fournissent une confidentialité mais permettent d'effectuer une agrégation [3]. On peut aussi ajouter différents mécanismes d'authentification, mais cela génère des messages plus gros [5]. La table II récapitule les fonctionnalités de toutes ces méthodes. Comme on peut le voir, les techniques d'agrégation multi-chemins sont celles qui fournissent le plus de protections contre la capture.

AMD et AMD-A génèrent un délai entre le moment où les mesures sont effectuées et le moment où elles sont reportées à la station de base. Cela est dû au fait que les capteurs doivent accumuler temporairement leurs mesures dans un tampon interne. Par exemple, un capteur qui utilise AMD et mesure des données toutes les m minutes devra envoyer ses messages

avec un intervalle de $t \times m$ minutes. Le premier message de la séquence devra attendre que d'autres mesures remplissent le tampon de taille t avant que l'information ne soit dispersée et envoyée vers la station de base.

B. Coût supplémentaire des communications

Une part fait strictement la même taille qu'une mesure d'un capteur; un message de l'une de nos techniques n'est pas plus grand qu'un message généré par la technique de l'arbre simple.

Nous définissons donc le surcoût en communication comme le nombre de messages supplémentaires envoyés par mesure vis à vis de l'approche de l'arbre simple.

1) AMS *:* Un capteur envoie p messages à chaque fois qu'il effectue une mesure. Avec un arbre simple, il n'en enverrait qu'un seul. En conséquence, le surcoût de AMS est de $p-1$ messages par mesure.

2) AMD *:* Un capteur doté de AMD envoie p messages chaque fois qu'il fait t mesures. Avec un arbre simple, il n'en enverrait que t. En conséquence, le surcoût de AMD est de $\frac{p-t}{t}$ messages par mesure. On remarque que ce surcoût est nul lorsque $p = t$. Cela correspond à la situation dans laquelle toutes les parts sont nécessaires pour reconstruire les mesures. C'est une conséquence de l'efficacité de AMD vis à vis de la taille : lorsqu'il n'y a pas de redondance, il n'y a pas de surcoût. Mais cela signifie aussi qu'il n'y a pas de protection contre les dénis de service. Plus généralement, le surcoût imputé à AMD sera le seul fait de la redondance des données. En choisissant la quantité de redondance, c'est à dire le rapport $\alpha = \frac{p}{t}$, on peut déterminer complètement le surcoût de AMD.

3) AMD-A *:* Un capteur doté de AMD-A envoie p messages chaque fois qu'il fait $t-1$ mesures. Avec un arbre simple, il n'en enverrait que $t-1$. En conséquence, le surcoût de AMD-A est de $\frac{p-t+1}{t-1}$ messages par mesure. La valeur minimale est obtenue lorsque p est minimal, c'est à dire $p = t$. Comme pour AMD, cela correspond à la situation où toutes les parts sont nécessaires pour la reconstruction.

Pour réduire le coût supplémentaire des communications, il faut choisir de grandes valeurs pour t. On peut prendre $p = \alpha t$ pour un α donné tel que $\alpha \geq 1$. On s'assure ainsi qu'au moins $t(\alpha-1)$s nœuds peuvent être compromis sans que l'on soit menacé par un déni de service. Le surcoût devient alors $\frac{\alpha-1+1/t}{1-1/t}$. Cela signifie que plus t est grand, plus le surcoût se rapproche de $\alpha - 1$ (surcoût de AMD).

Récapitulatif. AMS coûte le plus cher en terme de communications, tandis que AMD et AMD-A peuvent être nettement plus efficaces à ce sujet, en fonction de leurs paramètres. Il faut faire un compromis entre la quantité de redondance souhaitée et l'efficacité en terme de communication. AMD est efficace vis à vis de la taille, ce qui signifie que le surcoût en communication est seulement dû à la redondance des données. AMD-A possède presque cette propriété. D'un autre coté, en dépit de son coût en communication, AMS est tout de même intéressant. Il offre un confidentialité très forte et peut être utilisé dans des réseaux de capteurs sans contraintes d'énergie.

C. Implémentation

Dans cette section nous détaillons nos implémentations des trois techniques d'agrégation proposées ainsi que quelques résultats pratiques. Ces implémentations doivent être perçues comme une preuve de concept de la faisabilité des techniques proposées, pas comme une solution complète clefs en main.

1) Présentation: Des implémentations particulières de AMS, AMD et AMD-A ont été développées pour des capteurs MICAz Crossbow (voir section II-C) avec le système TinyOS. Toutes les opérations sont effectuées sur un corps fini $GF(p)$ qu'il est possible de paramétrer et nous avons utilisé les routines de calcul multi-précision de TinyECC [17], qui sont elles-même basées sur RSAREF [18]. Le code source de ces implémentations est disponible à l'adresse `http://www-rp.lip6.fr/~claveiro/secure-aggreg/`.

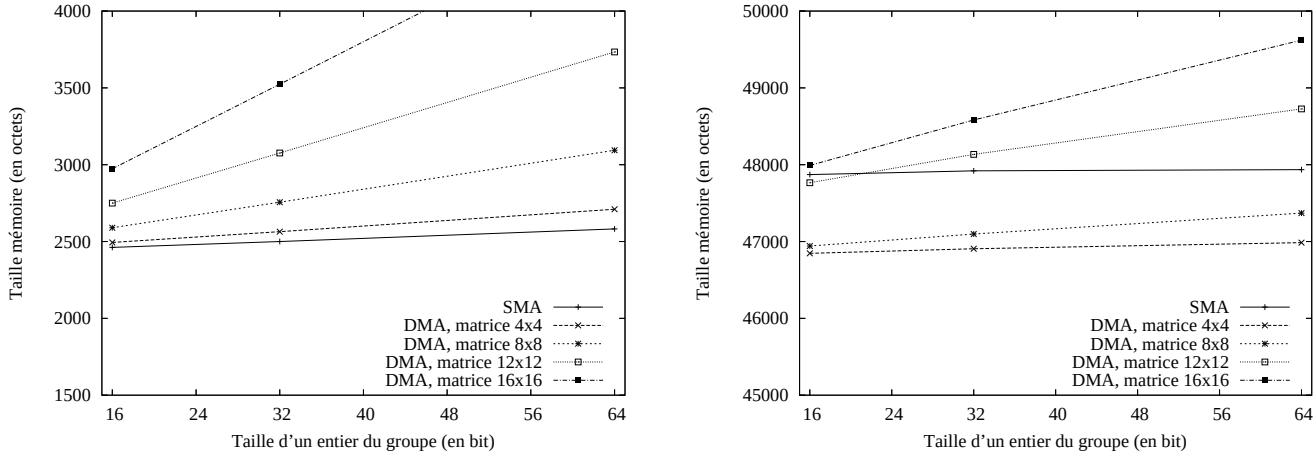
Pour des raisons de simplicité et pour éviter d'introduire un biais lié à la couche de routage, nous avons utilisé un routage multi-chemins statique. Nous utilisons la couche de liaison par défaut de TinyOS. Celle-ci ne satisfait pas les hypothèses de sécurité présentées au début de cet article, mais grâce à la modularité de TinyOS, il serait tout à fait possible d'écrire ses propres couches de routage et de liaison sécurisées sans modifier le code de nos implémentations.

Une fois compilé, beaucoup de paramètres ont un rôle sur l'occupation mémoire. Considérons le groupe $GF(p)$ ou la taille de la matrice $\mathbf{A}$ pour la dispersion d'information. La figure 3 présente les empreintes mémoire pour quelques paramètres typiques. L'empreinte de AMS est assez bonne quel que soit le groupe utilisé (environ la moitié de la RAM d'un MICAz, par exemple). Les empreintes de AMD et AMD-A dépendent beaucoup de la taille de la matrice $\mathbf{A}$ pour la dispersion d'information. Elle détermine le nombre maximum de parts p et le seuil t des techniques AMD et AMD-A. Pour des paramètres p et t donnés il faut une matrice $t \times p$. Certaines grosses valeurs, comme une matrice 16×16 avec des entiers de 64 bit ne tiennent pas dans un capteur MICAz. Mais la taille nécessaire pour d'autres valeurs plus réalistes est très acceptable vis à vis de l'occupation mémoire.

Le temps requis pour que les nœuds effectuent certaines opérations comme la création des parts ou l'agrégation des données est négligeable et n'a jamais attiré notre attention durant les tests.

Le processus d'agrégation marche comme suit. Les nœuds mesurent certaines données à des intervalles réguliers et envoient les parts vers la station de base assorties d'un numéro de séquence. Un nœud agrégateur n'agrège que les parts étiquetées par le même numéro de séquence. Tant que des parts arrivent, le nœud les agrège jusqu'à l'arrivée d'une nouvelle part avec un numéro de séquence plus élevé, ou bien l'expiration d'un temporisateur.

2) Expérimentations: Nous avons effectué à la fois des expériences réelles et des simulations à partir des implémentations précédemment décrites. Les expériences ont été faites à petite échelle (six nœuds et une topologie à trois chemins) pour tester la pratique de ces techniques. Pour stresser un peu plus les implémentations, nous avons



(a) Empreintes mémoire RAM

(b) Empreintes mémoire ROM

Fig. 3. *Empreintes mémoire des implémentations.* Pour des raisons de lisibilité, les empreintes de AMD-A ne sont pas représentées. Leurs consommations en RAM sont identiques à celle de AMD. Leurs occupations en ROM sont légèrement plus élevées, parce qu'il faut du code supplémentaire pour calculer les valeurs d'authentification.

effectué des simulations en utilisant TOSSIM. TOSSIM est un simulateur de réseau de capteurs qui compile directement du code TinyOS et simule la pile réseau de TinyOS bit par bit. Cela a l'avantage de parfaitement modéliser le comportement de l'implémentation.

Nous avons mesuré le nombre de messages émis par nos implémentations pour différents paramètres et différentes techniques. Nous avons utilisé cinq topologies aléatoires de quarante nœuds avec un temps de simulation d'une heure. Ces topologies utilisent des arbres aux nœuds disjoints et enracinés à la station de base pour le routage multi-chemins. Deux topologies ont quatre chemins, les autres en ayant respectivement trois, six, et huit. Notez que le nombre de chemins n'influence pas le nombre de messages mesuré, qui n'est modifié que par les paramètres p et t. Quand il y a plus de parts que de chemins disponibles, certains chemins transportent plusieurs parts. Quand il y a plus de chemins que de parts, certains chemins ne sont pas utilisés. La figure 4 représente le nombre total de messages envoyés sur les cinq topologies en fonction du temps de simulation. Nous comparons ce nombre de messages avec la technique de l'arbre simple précédemment décrite.

Il est possible d'observer quelques propriétés prévisibles de nos techniques. Comme analysé précédemment, le coût de AMS est le plus élevé, et ne dépend que de p. En conséquence, AMS nécessite grossièrement $p-1$ messages supplémentaires comparé à la technique de l'arbre simple. On peut également constater que le coût des techniques basées sur la dispersion d'information dépend de $\alpha = \frac{p}{t}$. Il n'est pas surprenant que AMD-A et AMD exhibent des performances similaires, avec AMD étant un petit peu plus efficace. Les coûts en communication sont néanmoins un peu plus élevés que leurs prévisions théoriques calculés en section V-B. Par exemple, AMD-A avec $p = 12$ et $t = 8$ possède un surcoût de 1 au lieu de la prédiction de 0.7. Également, AMD-A avec $p = t = 12$ exhibe une petite surcharge de 0.25 à la place de 0.1. Cela est dû à des considérations pratiques qui obligent les agrégateurs à rater certaines des possibilités d'agrégation.

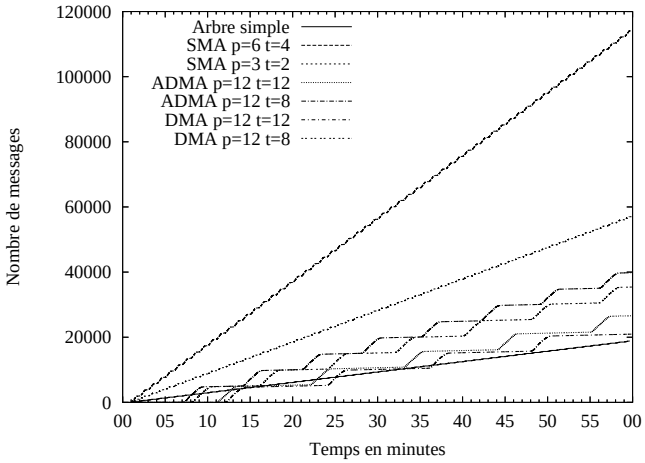


Fig. 4. Nombre de messages émis en fonction de la technique utilisée.

D. Coût vs. sécurité

La sécurité implique nécessairement un coût vis à vis d'une certaine métrique. Certaines techniques génèrent une surcharge de communication, d'autre de consommation CPU, etc. Ce qu'il est important de définir dans ce contexte est une solution qui apporte le niveau de sécurité requis pour un coût acceptable. Dans cette optique, nos propositions sont vraiment prometteuses. En effet, en utilisant les techniques proposées, un réseau peut tolérer plusieurs captures sans mettre en péril la confidentialité, l'authenticité et la disponibilité. Donc les coûts de AMS, AMD et AMD-A sont tout à fait justifiés. D'autant plus qu'il est possible de configurer ces coûts en ajustant les différents paramètres de ces techniques. En fonction des

ressources allouées à la sécurité, on peut échanger de la sécurité contre l'efficacité des communications.

VI. Conclusion et travaux futurs

Dans cet article nous proposons trois techniques pour sécuriser l'agrégation des données en utilisant un routage multi-chemins. Elles sont basées sur la cryptographie à seuil et la dispersion d'information. Dans les techniques proposées les capteurs découpent leurs mesures au sein de différentes parts et les distribuent sur plusieurs chemins. À la réception d'un nombre minimal de parts, la station de base peut reconstruire une valeur agrégée.

En fonction de la technique utilisée et de ses paramètres, il est possible d'obtenir différents degrés de résistance aux attaques de déni de service, aux écoutes passives, et aux falsifications de données. En utilisant l'agrégation multi-chemins secrète, on peut garantir qu'un sous-ensemble de nœuds compromis ne transmet aucune information au sujet des mesures. Cela possède un certain surcoût. En utilisant l'agrégation multi-chemins dispersée, on obtient un surcoût optimal, mais avec un niveau de confidentialité plus faible. En fonction de l'application ou du scénario, une approche peut présenter plus ou moins d'avantages sur l'autre.

À notre connaissance, les trois techniques proposées sont les premières qui étudient la résistance à la capture conjointement à l'agrégation de données en utilisant plusieurs chemins. Nous projetons de fournir des études plus poussées concernant la modélisation du comportement des paramètres de sécurité lorsque le réseau est soumis à des captures de nœuds aléatoires. Il est aussi probablement possible de généraliser et d'appliquer ces techniques dans d'autre contextes que les réseaux de capteurs.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks : a survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, Mar. 2002.

[2] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Commun. ACM*, vol. 43, no. 5, pp. 51–58, 2000.

[3] E. Mykletun, G. Tsudik, and C. Castelluccia, "Efficient aggregation of encrypted data in wireless sensor networks," in *MobiQuitous 2005 : Conference on Mobile and Ubiquitous Systems : Networking and Services*, 2005, pp. 109– 117.

[4] B. Przydatek, D. Song, and A. Perrig, "SIA : secure information aggregation in sensor networks," in *SenSys '03 : Proceedings of the 1st international conference on Embedded networked sensor systems*, 2003, pp. 255–265.

[5] L. Hu and D. Evans, "Secure aggregation for wireless networks," in *SAINT-W '03 : Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops)*, 2003, p. 384.

[6] D. Wagner, "Resilient aggregation in sensor networks," in *SASN '04 : Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, 2004, pp. 78–87.

[7] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[8] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *J. ACM*, vol. 36, no. 2, pp. 335–348, 1989.

[9] C. Karlof, N. Sastry, and D. Wagner, "TinySec : A link layer security architecture for wireless sensor networks," in *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)*, 2004.

[10] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key predistribution scheme for wireless sensor networks," in *CCS '03 : Proceedings of the 10th ACM conference on Computer and communications security*, 2003, pp. 42–51.

[11] D. Ganesan, R. Govindan, S. Shenker, and D. Estrin, "Highly-resilient, energy-efficient multipath routing in wireless sensor networks," in *MobiHoc '01 : Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, 2001, pp. 251–254.

[12] S. Dulman, T. Nieberg, J. Wu, and P. Havinga, "Trade-off between traffic overhead and reliability in multipath routing for wireless sensor networks," 2003.

[13] C. Karlof and D. Wagner, "Secure routing in wireless sensor networks : Attacks and countermeasures," *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, vol. 1, no. 2–3, pp. 293–315, 2003.

[14] MICAz *data sheet*, Crossbow. [Online]. Available : http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf

[15] M. Bellare and P. Rogaway, "Random oracles are practical : A paradigm for designing efficient protocols," in *ACM Conference on Computer and Communications Security*, 1993, pp. 62–73.

[16] A. Perrig, J. Stankovic, and D. Wagner, "Security in wireless sensor networks," *Commun. ACM*, vol. 47, no. 6, pp. 53–57, 2004.

[17] D. P. Ning and A. Liu, "TinyECC : Elliptic curve cryptography for sensor networks," 2005. [Online]. Available : http://discovery.csc.ncsu.edu/software/TinyECC/

[18] RSA Laboratories, "RSAREF : A free cryptographic toolkit," 1994. [Online]. Available : http://www.csm.ornl.gov/~dunigan/rsaref.txt