

Une proposition d'agrégation de MAC pour les réseaux de capteurs utilisant des fonctions de hachage universelles

Wassim Znaïdi (wassim.znaïdi@insa-lyon.fr)*

Marine Minier (marine.minier@insa-lyon.fr)*

Cédric Lauradoux (cedric.lauradoux@uclouvain.be)[†]

Résumé : In this paper, we study the problem of aggregation of message authentication codes (MACs) in Wireless Sensor Networks (WSNs). We propose to use some of the well known universal hash functions and to modify some of them to define a scheme allowing both data and MACs aggregation.

Mots Clés : sécurité, réseau de capteurs, MAC, fonctions de hachage universelles

1 Introduction

Les réseaux de capteurs sans fil sont constitués de nœuds déployés en grand nombre dans un environnement hostile en vue de collecter et de transmettre des données vers un ou plusieurs points de collecte, et ce de façon autonome. Ces réseaux ont un intérêt particulier pour plusieurs types d'applications (notamment celles du domaine militaire). Les nœuds capteurs composant le réseau possèdent généralement de faibles capacités de calcul, de mémoire et d'énergie, l'accès au médium radio étant l'élément le plus coûteux. Ainsi diminuer le nombre de bits transmis afin d'augmenter la durée de vie du réseau est un défi permanent pour ce type de réseau. Une approche répandue consiste à agréger les paquets lors de leurs acheminements vers la station de base également appelée puits (le nœud collecteur). L'agrégation des données est une tâche simple et facile (il suffit par exemple pour chaque nœud parent de faire les moyennes des valeurs reçues des nœuds fils). Cependant, si l'on souhaite garder la confidentialité des données afin que seule la station de base ait connaissance des valeurs remontées par chaque nœud, il s'agit alors d'utiliser des mécanismes de cryptographie dédiés pour atteindre un certain niveau de sécurité dans le réseau. Plusieurs méthodes ont été proposées pour atteindre ce but : [PSP03] et [CPS06] proposent l'utilisation d'arbres de Merkle, [YWZC06] et [HE03] sont fondés sur l'utilisation de MAC et de μ -TESLA dans un arbre représentant le réseau et [CMT05] propose un mécanisme fondé sur les propriétés d'homomorphismes additifs des chiffrements à flot. La confidentialité de bout en bout est vérifiée par plusieurs schémas comme [CMT05] et [AGW05]. Notons également qu'en matière de sécurité dans les réseaux de capteurs, l'usage de la cryptographie asymétrique reste souvent très coûteux en raison

* Laboratoire CITI, INSA de Lyon - 6, avenue des arts - 69621 Villeurbanne Cedex - FRANCE

† UCL / INGI / GSI - Place Saint Barbe, 2 - Louvain-la-Neuve - BELGIQUE

du nombre de calculs à effectuer et de la taille des données à transmettre. C’est pourquoi on lui préfère en général la cryptographie symétrique.

Cependant, la plupart des travaux précédents se dédouanent du problème direct de l’agrégation des MAC en surchiffant les données. Rappelons qu’un MAC ou code d’authentification de message peut être vu comme une signature bipartie en cryptographie symétrique : il s’agit de garantir d’une part l’intégrité du message envoyé et d’autre part la vérification de l’identité de l’émetteur pour la personne possédant la clé symétrique partagée. Un MAC est donc un algorithme qui prend en entrée un message m , une clé K et qui produit un condensé, un tag tag : $tag = MAC_K(m)$. L’algorithme de vérification pour le receveur consiste à vérifier si pour le message reçu m' on a bien $tag? = MAC_K(m')$, i.e. si le message reçu n’a pas été modifié en cours de route. Dans le cas d’un réseau de capteurs, il s’agit de vérifier les propriétés garanties par un MAC de bout en bout : le MAC doit être vérifié entre la station de base et chacun des nœuds envoyant un message.

Nous cherchons donc ici un schéma comme défini dans [CC08], à savoir que, étant donné n messages (m_1, m_2, \dots, m_n) envoyés par n nœuds différents et dont le résultat agrégé est défini par une fonction f : $m = f(m_1, m_2, \dots, m_i, \dots, m_n)$, chaque nœud i ayant produit son propre “tag” (i.e. un MAC) du message m_i : $tag_i = MAC_{k_i}(m_i)$. Alors, il est possible pour la station de base ayant la connaissance de toutes les clés (k_1, \dots, k_n) et recevant $(hdr, m, tag_1, tag_2, \dots, tag_i, \dots, tag_n)$ où hdr est constitué de l’ensemble des identités des nœuds ayant répondu (i.e. $hdr = \{i, j, \dots\}$), de vérifier si oui ou non m est bien un message valide en utilisant la connaissance de $(tag_1, tag_2, \dots, tag_i, \dots, tag_n)$. Cet épineux problème n’a encore trouvé que peu de réponses et nous nous proposons donc de l’étudier. Pour cela, nous utilisons les fonctions de hachage universelles [CW79] permettant de calculer des MAC particuliers, alliées avec la méthode d’agrégation des données proposée dans [CC08].

La Section 2 de cet article présentera l’état de l’art du domaine en terme de définition des MAC et introduira quelques fonctions de hachage universelles. La Section 3 présentera notre proposition en étendant un des MAC défini précédemment. Enfin, la Section 3.3 est dédiée à l’analyse de sécurité de notre proposition dans le modèle AMAC défini dans [CC08] avant de conclure.

2 Etat de l’art et préliminaires

Peu de travaux se sont intéressés aux schémas d’agrégation des MAC (AMAC). On peut cependant citer [BHL06, KL08] dans lesquels les auteurs, après avoir donné une définition formelle des primitives employées, proposent le schéma d’agrégation de MAC suivant : chaque nœud i envoie un message m_i ainsi qu’un tag $tag_i = MAC_K(m_i)$ au nœud agrégateur qui lui-même concatène les messages et calcule le X-OR des tags reçus en plus du sien. Les auteurs montrent également la sécurité de leur schéma et mesurent l’impact de la longueur du tag sur le temps de vérification d’un simple message.

Comme signalé dans l’introduction, dans [CC08], les auteurs étudient la possibilité de construire des AMAC. Ils présentent deux modèles formels de sécurité, l’un concernant les chiffrements agrégés appelé CDA (Concealed Data Aggregation) et un concernant la sécurité du schéma défini en introduction appelé AMAC. Ils montrent ensuite les problèmes de sécurité induits directement par les définitions données.

A notre connaissance, il s’agit des deux seuls travaux établissant un lien formel entre

agrégation de MAC et réseau de capteurs. Notons également que habituellement quand on parle de MAC, il s’agit d’utiliser essentiellement soit HMAC [Pub02] soit CBC-MAC [Pub85] ou CMAC¹. Il existe dans la littérature beaucoup d’autres constructions notamment celles qui nous intéressent ici à savoir les MAC construits sur les fonctions de hachage universelles. Ces dernières ont été introduites par Carter et Wegman dans [CW79] et étudiées par la suite dans [Kra94, Sho96, Sar08].

2.1 Fonctions de hachage universelles

Pour construire un MAC avec une fonction de hachage universelle, la solution la plus simple et la plus élégante est, pour un message m de taille l découpé en t blocs de valeurs appartenant à un corps fini \mathbb{F}_p (i.e. $m = (m_1, \dots, m_t)$ avec chaque $m_i \in \mathbb{F}_p$) de calculer $MAC_{k,k'}(m) = k' + f_k(m_1, \dots, m_t)$ où k' est dans ce cas une valeur pseudo-aléatoire de \mathbb{F}_p et f désigne la fonction de hachage universelle (aussi appelée fonction de hachage avec clé). Un des premiers exemples de fonction universelle est l’évaluation d’un polynôme particulier en un point k . Dans ce cas, f est défini par $f_k(m_1, \dots, m_t) = \sum_{i=1}^t m_i \cdot k^i \pmod p$. Il est à noter que la valeur k peut être utilisée plusieurs fois tandis que k' doit être changée à chaque envoi. Notons que cette fonction est multi-linéaire et que le corps de base peut être \mathbb{F}_p ou \mathbb{F}_{2^n} . L’évaluation de polynôme a été proposée dans [Ber05] par exemple.

La fonction f paramétrée par la valeur $k \in \mathcal{K}$ est une fonction de hachage universelle et on dit qu’elle est ϵ -presque universelle si sa probabilité de collision pour une distribution aléatoire de la valeur k sur l’ensemble \mathcal{K} (i.e. $Pr_k(f_k(m) = f_k(m')), \forall k \in_R \mathcal{K}$) est inférieure à ϵ . On dit également que f est ϵ -presque XOR universelle (ϵ -AXU) si la probabilité différentielle pour une distribution uniforme de la valeur k sur \mathcal{K} est bornée par ϵ , i.e. $\forall(m, m', a), Pr_k(f_k(m) - f_k(m') = a) \leq \epsilon$.

Beaucoup de schémas vérifiant ces propriétés ont été proposés, on peut notamment citer le travail de V. Shoup dans [Sho96]. Ce dernier propose de répartir en trois grandes classes les fonctions de hachage universelles vérifiant ces propriétés : la première est celle composée des évaluations de polynômes sur des corps premiers ou finis dont nous venons de voir un exemple, la deuxième est constituée des divisions de polynômes sur \mathbb{F}_2 décrite par Krawczyk dans [Kra94] sous le nom de CRC cryptographique ou de “LFSR (Linear Feedback Shift Register) hash”, la troisième est constituée des divisions de polynômes sur \mathbb{F}_{2^k} qu’étudie Shoup dans son article. Notons également qu’à Crypto’08 [HP08], B. Preneel et H. Handschuh, ont montré qu’il existait des attaques de type diviser et conquérir sur les MAC utilisant comme fonction de hachage universelle des évaluations de polynômes sur des corps premiers et finis.

2.2 Trois constructions particulières

2.2.1 CRC cryptographique

Comme décrit dans [Sho96], le premier schéma proposé par Krawczyk dans [Kra94] est fondé sur l’opération de division modulaire par un polynôme irréductible sur le corps \mathbb{F}_2 . Il s’agit en fait d’une variante cryptographique des codes de redondance cyclique (CRC), standards des mécanismes de détection d’erreurs dans les réseaux. Plus précisément, chaque message M est vu comme sa représentation équivalente polynomiale $M(x)$ sur

¹ voir par exemple : <http://csrc.nist.gov/groups/ST/toolkit/BCM/index.html>

le corps \mathbb{F}_2 , les coefficients correspondants aux bits de M . Ainsi, pour chaque polynôme irréductible $q(x)$ de degré n dans \mathbb{F}_2 , la fonction de hachage universelle associée est $h_q(M) = M(x) \cdot x^n \pmod{q(x)}$. Notons qu'il est nécessaire dans ce schéma de multiplier $M(x)$ par x^n afin d'assurer la sécurité du schéma pour la notion de ϵ -AXU contrairement au schéma initialement proposé par M. Rabin dans [Rab81].

La famille (m, n) de fonctions de hachage h_q est donc l'ensemble des polynômes irréductibles de degré n et des messages de taille m . Notons que cette famille de fonctions de hachage est \oplus -linéaire (i.e. $h_q(M_1) \oplus h_q(M_2) = h_q(M_1 \oplus M_2)$), ϵ -presque universelle (avec $\epsilon \leq \frac{n+m}{2^n-1}$) et ϵ -presque XOR universelle (avec $\epsilon \leq \frac{n+m}{2^n-1}$).

Signalons aussi que l'implémentation matérielle et logicielle de tels mécanismes est très efficace car la division modulaire par des polynômes dans \mathbb{F}_2 peut être implantée en utilisant un simple LFSR. Notons également que l'extension proposée et prouvée sûre par Shoup concerne le cas où le corps de base est \mathbb{F}_{2^k} . Dans ce cas, la valeur correspondante du ϵ est d'environ $\frac{m}{2^{kn}}$.

2.2.2 LFSR hashing

Dans le même article, Krawczyk introduit une deuxième construction fondée cette fois-ci sur des matrices aléatoires. Plus précisément, soit A une matrice booléenne de Toeplitz de taille $n \times m$ (i.e. chaque diagonale inférieure est fixée, i.e. si $k - i = l - j$ pour n'importe quel indice alors $A_{i,j} = A_{k,l}$) et soit un message M de taille m . la fonction de hachage universelle $h_A(M)$ est alors la multiplication binaire de la matrice A par le vecteur colonne composé des bits du message M : $h_A(M) = A \cdot M$.

Une méthode simple permettant de construire ce type de matrice est l'utilisation d'un LFSR : soit $q(x)$ un polynôme irréductible de degré n sur \mathbb{F}_2 . Soit s_0, s_1, \dots la séquence de sortie des bits générés par le LFSR défini par $q(x)$ et l'état initial du LFSR $s = (s_0, s_1, \dots, s_{n-1})$. Pour chaque polynôme $q(x)$ et pour chaque séquence initiale non nulle, on associe la fonction de hachage $h_{q,s}(M)$ définie par la combinaison linéaire $\bigoplus_{j=0}^{m-1} M_j \cdot (s_j, s_{j+1}, \dots, s_{j+n-1})$, où M_j est le j -ième bit de M . En d'autres termes, à chaque tour d'horloge, le LFSR change d'état interne en tenant compte de chaque bit du message. Cette famille de fonction de hash est également \oplus -linéaire, ϵ -presque universelle (avec $\epsilon \leq \frac{n+m}{2^n-1}$) et ϵ -presque XOR universelle (avec $\epsilon \leq \frac{m}{2^n-1}$).

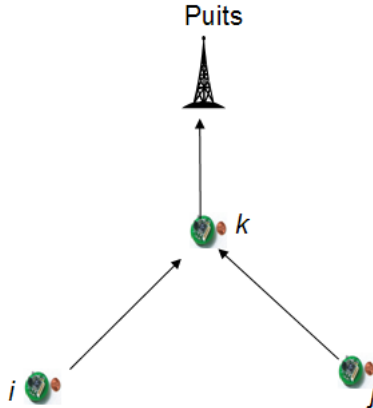
2.2.3 Fonction de hachage universelle multi-linéaire

Dans [Sar08], P. Sarkar propose l'évaluation suivante : Soit un corps \mathbb{F}_p et une extension de ce corps \mathbb{F}_{p^n} avec $n \geq 1$. Soit ϕ une transformation linéaire de \mathbb{F}_{p^n} dans lui-même telle que le polynôme minimal de ϕ dans $\mathbb{F}_p[x]$ soit de degré n et irréductible sur \mathbb{F}_p . Le message à chiffrer M est découpé en $l \leq n$ éléments (M_1, \dots, M_l) de \mathbb{F}_p . La famille de fonctions de hachage universelles associée est donc

$$G_K(M) = M \cdot (K, \phi(K), \dots, \phi^{l-1}(K)) = M_1 K + M_2 \phi(K) + \dots + M_l \phi^{l-1}(K)$$

où K est un élément de \mathbb{F}_{p^n} .

La famille G_K est donc une combinaison linéaire de (M_1, \dots, M_l) et de $(K, \phi(K), \dots, \phi^{l-1}(K))$, elle est donc multi-linéaire (i.e. linéaire en chacune de ses composantes d'entrée), ϵ -presque universelle avec $\epsilon \leq 1/q^n$ et également ϵ -presque XOR universelle avec la même valeur du ϵ .

FIG. 1: Cas d'étude à trois nœuds : i ; j et k .

Sarkar remarque également que ϕ peut être implantée facilement car elle peut être vue comme un LFSR sur \mathbb{F}_p . L'auteur étudie ensuite les valeurs de p permettant une implantation rapide et efficace en hardware et software. Les exemples donnés sont $q = 2$, $n = 128$; $q = 2^8 + 1$, $n = 16$; $q = 2^{16} + 1$, $n = 8$; $q = 2^{32} + 15$, $n = 4$. Il donne également des exemples d'extensions de corps sur \mathbb{F}_2 que nous ne détaillerons pas ici.

3 Proposition d'agrégation de MAC utilisant les fonctions de hachage universelles

Nous allons à présent nous intéresser aux applications des différentes fonctions proposées dans la section précédente dans le cas d'un réseau de capteurs. Notre cas d'étude sera simple pour faciliter la compréhension et sera celui décrit dans la figure 1 où deux nœuds fils i et j dépendent du parent k qui a le rôle d'un nœud agrégateur. Ce dernier dépend directement du puits mais ce schéma peut être élargi et généralisé sans difficulté.

3.1 Agrégation des xors : adaptation des approches de Krawczyk's pour les réseaux de capteurs

La première construction décrite par Krawczyk peut directement être appliquée à l'agrégation de MAC dans le cas de l'utilisation d'un x-or. Cette approche peut être directement combinée à l'agrégation des données proposée dans [CMT05] toujours dans le cas du x-or.

Supposons donc un réseau de capteurs composé de N nœuds i . Chaque nœud reçoit durant une phase d'initialisation une clé de chiffrement K_{Ei} partagée entre le nœud et la station de base, une clé d'authentification K_{Ai} partagée entre le nœud et la station de base et un polynôme commun à tous les nœuds et à la station de base $q(x)$. Imaginons un réseau simple comme décrit à la figure 1 où les trois nœuds i , j et k décrivent un arbre dépendant directement du puits.

Lorsque le nœud i souhaite envoyer (à la demande ou à intervalle régulier) un mes-

sage m_i , il chiffre ce message à l'aide d'une suite pseudo-aléatoire générée à partir d'un algorithme de chiffrement à flot E (par exemple RC4 ou SNOW v2 [EJ02]). Pour utiliser un tel algorithme, le nœud et la station de base ont besoin de la clé commune K_{E_i} et d'une valeur commune d'initialisation IV_i qui doit absolument être changée à chaque chiffrement. Ainsi, le nœud chiffre $C_i = m_i \oplus r_i$ où r_i est la suite chiffrante produite par $E(K_{E_i}, IV_i)$. Il faut également comme mentionné dans [CMT05] que le nœud transmette à la station de base son identité i et la valeur unique IV_i qui tient le rôle de compteur pour également éviter les attaques par rejeu. Nous proposons donc tout d'abord que IV_i soit la seule valeur transmise en écrivant $IV_i = i || CTR_i$, c'est à dire comme un concaténé de l'identité du nœud et d'un compteur CTR_i incrémenté de 1 à chaque nouvel envoi (notons cependant que cette valeur doit être retransmise à chaque fois en cas de non-transmission d'un message particulier).

Le nœud i produit également le MAC du message m_i en utilisant la construction de Krawczyk présentée précédemment : il calcule $tag_i = (m_i(x) \cdot x^n \bmod q(x)) \oplus r'_i$ où r'_i est une suite pseudo-aléatoire générée à partir de E initialisé à l'aide de K_{A_i} et IV_i . Ainsi, le nœud i transmet à son nœud parent k : $\{hdr, data, tag\}$ avec $hdr = IV_i$, $data = C_i$, $tag = tag_i$.

Imaginons que le nœud j dépendant également du nœud k transmette également un message m_j , dans ce cas, j envoie à k $\{IV_j, C_j, tag_j\}$. Le nœud k transmet alors à la station de base (en considérant que lui-même doit envoyer la valeur m_k) :

$$\{IV_i, IV_j, IV_k, C_i \oplus C_j \oplus C_k, tag_i \oplus tag_j \oplus tag_k\}.$$

La station de base déchiffre $C_i \oplus C_j \oplus C_k \oplus r_i \oplus r_j \oplus r_k = m_i \oplus m_j \oplus m_k = M$ à l'aide de sa connaissance des différentes clés de chiffrements et des différentes valeurs d' IV . Elle vérifie ensuite $tag_i \oplus tag_j \oplus tag_k \oplus r'_i \oplus r'_j \oplus r'_k = m_i \cdot x^n \oplus m_j \cdot x^n \oplus m_k \cdot x^n \bmod q(x) = (m_i \oplus m_j \oplus m_k) \cdot x^n \bmod q(x) = M \cdot x^n \bmod q(x)$.

Ainsi, la station de base peut vérifier la valeur des tags agrégés en fonction de la valeur somme reçue.

Exemple de tailles de valeurs à envisager. En ce qui concerne la taille des clés, elles doivent toutes faire au minimum 128 bits. Le polynôme $q(x)$ peut être un polynôme primitif de longueur 64 bits. Les IV peuvent être de taille 48 bits (24 bits par exemple servant à fixer l'identité du nœud et 24 bits réservé au compteur²). Les messages à considérer doivent également être définis par une longueur maximale fixée à 64 bits (cette longueur ne peut être inférieure à celle du polynôme). Dans ce cas, la taille des messages est fonction du nombre de nœuds N ayant transmis un message : $48N + 64 + 64$ (on peut également envisager des tailles de messages et de polynôme de taille 96 bits, dans ce cas la taille des messages devient $48N + 96 + 96$), la borne de sécurité de la fonction universelle étant dans ce cas égale à 2^{-56} (resp. 2^{-87}). L'overhead créé sur le réseau dépend clairement de la taille du header et donc du nombre de nœuds renvoyant un message. Afin de réduire cette valeur, on peut imaginer le mécanisme suivant à la demande : lorsque la station de base souhaite recevoir des valeurs, elle envoie une valeur unique de 24 bits notée IV . Les nœuds qui la reçoivent utilisent la méthode précédemment décrite pour chiffrer et authentifier leurs données simplement dans ce cas, la valeur d'initialisation étant alors constituée de leur

² Notons que pour éviter les travers d'une attaque de type Wep, une fois que toutes les valeurs possibles du compteur ont été parcourues, il faut renouveler les clés de chiffrement.

identité concaténée avec la valeur envoyée par la station de base *IV*. Dans ce cas, afin de limiter la valeur de l'overhead induit, on peut imaginer que le header déterminant l'identité des nœuds ayant répondu soit constitué d'un filtre de Bloom de taille m chiffré comme proposé dans [ACH06]. La clé de chiffrement K du filtre est partagée par tous les nœuds du réseau et chaque nœud, dans ce cas, chiffre k fois son identité mettant à 1 le bit correspondant à la valeur du chiffré obtenu. Cette méthode semble relativement performante même si la probabilité d'obtenir un faux positif est d'environ $0,6185^{m/n}$ où n est le nombre d'éléments à insérer et en considérant que k vaut $0,7\frac{m}{n}$. Ainsi, pour un réseau d'environ 200 nœuds, un filtre de Bloom de taille 2048 bits avec un k valant 7 a une probabilité de faux positifs de moins de 1%. On peut également pour diminuer la taille du filtre de Bloom envisager de compter non pas sur des bits mais par exemple sur des mots de 4 bits ou des octets. Il faut également signaler que dans ce cas, chaque nœud fait k hachages supplémentaires et que la station de base doit tester l'identité de tous les nœuds dans le filtre de Bloom ce qui correspond à kN calculs de hash supplémentaires.

Nous venons de proposer ici une utilisation directe de la fonction proposée par Krawczyk dans le cas où on cherche à obtenir le x-or des messages et non la somme. Nous avons pris comme exemple la première construction proposée, notons également que le même raisonnement peut être appliqué si on souhaite utiliser la deuxième construction proposée par Krawczyk. Dans ce cas, afin d'utiliser la \oplus -linéarité de la fonction de hachage universelle, il faut que chaque nœud soit initialisé avec la même matrice A , les autres paramètres restant les mêmes que les précédents. Notons également que comme dans le cas précédent une taille de MAC de 64 ou 96 bits laisse une marge de sécurité raisonnable.

3.2 Agrégation sur l'ensemble \mathbb{F}_p

Nous nous proposons dans cette section d'étendre le MAC initialement proposé par Krawczyk à \mathbb{F}_p avec p premier afin de rendre le MAC proposé non plus seulement \oplus -linéaire mais $+$ -linéaire comme celui proposé par Sarkar.

3.2.1 Extension des constructions de Krawczyk à \mathbb{F}_p

Nous introduisons tout d'abord les notations suivantes : on note \mathbb{F}_p^l l'espace vectoriel de taille l et \mathbb{F}_p^n celui de taille n . Le message M dont on souhaite calculer le MAC s'écrit alors $M = (M_0, \dots, M_{l-1})$ où chaque M_i appartient à \mathbb{F}_p de même, la sortie de la fonction de hash sera considérée comme un vecteur de \mathbb{F}_p^n .

Le schéma de Krawczyk devient donc dans notre cas : Soit $M = (M_0, \dots, M_{l-1})$ dans \mathbb{F}_p^l . On réécrit M comme un polynôme à coefficients dans \mathbb{F}_p : $M(x) = M_0x^l + M_1x^{l-1} + \dots + M_{l-1}$ puis on choisit un polynôme $q(x)$ irréductible à coefficients dans \mathbb{F}_p de degré n avec comme coefficient de plus haut degré 1. La fonction de hash devient alors $h_q(M) = M(x) \cdot x^n \pmod{q(x)}$ vue comme un vecteur de taille n d'éléments de \mathbb{F}_p . Ensuite, il reste à calculer le tag : $tag = h_q(M) + r$ où r est un vecteur de nombres aléatoires pris dans \mathbb{F}_p^n .

Il reste donc à calculer $Pr_h(h_q(M) - h_q(M') = c(x))$ pour prouver la validité de ce schéma. On utilise pour cela directement les résultats de [Sho96] et [Kra94] pour dire que si $h_q(M) - h_q(M') = c(x)$ cela signifie par linéarité que $h_q(M - M') = c(x)$ et donc que $q(x)$ divise $(M(x) - M'(x)) \cdot x^n - c(x)$. Ce polynôme est un polynôme de degré maximal $l + n$. $q(x)$ étant un polynôme irréductible de degré n , le nombre de facteurs de degré n du polynôme précédent est donc $\frac{n+l}{n}$. Il y a donc au plus $\frac{n+l}{n}$ fonctions h_q dans l'ensemble

des fonctions possibles qui transforment $M + M'$ en $c(x)$ tandis que le nombre de fonctions possibles est l'ensemble des polynôme irréductibles de \mathbb{F}_p^n soit $\frac{p^n - 1}{n}$. On en déduit donc directement que $Pr_n(h_q(M) - h_q(M') = c(x)) \leq \frac{n+1}{p^n - 1}$ donc que la famille de fonctions proposées est ϵ -presque XOR universelle, ϵ -presque universelle avec la même valeur du ϵ et qu'elle est multi-linéaire.

On a donc généralisé l'approche proposée par Krawczyk à un corps premier \mathbb{F}_p tout en gardant les propriétés de linéarité souhaitées. Notons que le deuxième schéma proposé dans l'article initial peut se généraliser de la même manière. Notons que dans ce cas, cette dernière construction est très proche de celle proposée par Sarkar mis à part la définition de l'espace d'arrivée vue dans notre construction comme un espace vectoriel et vue dans le cas de Sarkar comme une extension de corps. Intéressons nous à présent à la manière d'utiliser cette nouvelle fonction dans le cas d'un réseau de capteurs.

3.2.2 Application à un réseau de capteurs

La question qu'il est à présent légitime de se poser est comment utiliser cette construction dans un réseau de capteurs. Afin de pouvoir utiliser au mieux les potentialités de la fonction définie précédemment, on suppose ici que les messages (par exemple les relevés de température) vont être envoyés par paquets, chaque paquet étant constitué de l messages de taille p . Ainsi, chaque capteur relevant par exemple l températures stocke ces l valeurs M_0, \dots, M_{l-1} avant de les envoyer toutes ensemble à la station de base à chaque intervalle de temps donné ou à la demande, l étant une taille fixée connue à l'avance. Dans le cas où un capteur n'a pas collecté l valeurs mais un nombre inférieur, il suffit qu'il remplace les valeurs manquantes par des 0 dans la méthode proposée. Reprenons l'exemple de la figure 1.

Sous les mêmes hypothèses que dans le cas précédent, on suppose que chaque nœud i a en commun avec la station de base une clé de chiffrement K_{Ei} , une valeur commune d'initialisation IV_i , une clé d'authentification K_{Ai} , un même algorithme de chiffrement à flot E et enfin un polynôme irréductible commun à tous les nœuds et à la station de base $q(x)$ de degré n à coefficients dans \mathbb{F}_p .

Le nœud i après avoir stocké l messages les envoie à la station de base en les chiffrant et en les authentifiant. Pour le chiffrement, il emploie cette fois-ci directement la méthode d'agrégation de chiffrement proposée dans [CMT05] à savoir qu'il calcule pour chaque message M_j pour j variant de 0 à $l-1$, $M_j + r_j \pmod p$ où r_j est une suite pseudo-aléatoire comprise entre 0 et $p-1$ où p est un nombre premier défini par $p \geq 2^{\lceil \log_2 M * N \rceil}$ où M est la taille maximale que peut prendre un unique message et où N est le nombre maximal de nœuds. Ainsi, tout d'abord le nœud i chiffre ses l messages : $M^i = (M_0^i, \dots, M_{l-1}^i)$ de la façon suivante : $C^i = M^i + r^i = (C_0^i = M_0^i + r_0^i \pmod p, \dots, C_{l-1}^i = M_{l-1}^i + r_{l-1}^i \pmod p)$ où chaque r_j^i est une valeur aléatoire appartenant à l'intervalle $[0, p-1]$. Ces valeurs sont obtenues à l'aide de l'algorithme E initialisé avec la clé K_{Ei} et une valeur d'initialisation IV_i . Le nœud i calcule ensuite le MAC associé à l'ensemble de ces messages : pour cela, on obtient tout d'abord un vecteur de taille n , $h_q(M^i) = (h_0^i, \dots, h_{n-1}^i) = (M^i \cdot x^n \pmod{q(x)})$ dépendant du message M^i vu comme un polynôme à coefficients dans \mathbb{F}_p . Notons également que $q(x)$ est un polynôme commun à tous les nœuds. Dans ce cas, on a : $tag^i + r^i = (tag_0^i, \dots, tag_{n-1}^i) = (h_0^i + r_0^i \pmod p, \dots, h_{n-1}^i + r_{n-1}^i \pmod p)$ où les r_j^i sont des valeurs pseudo-aléatoires appartenant à l'intervalle $[0, p-1]$ obtenues à l'aide de l'algorithme E . Ainsi, le nœud i transmet à son nœud parent k : $\{hdr, data, tag\}$ avec

$hdr = IV_i, data = C^i, tag = tag^i$.

En suivant le même exemple que précédemment, le nœud j dépendant du nœud k transmet également ses l messages et le MAC associé : $\{IV_j, C^j, tag^j\}$. Le nœud k transmet alors à la station de base (en considérant que lui-même doit envoyer l messages M^k) :

$$\{IV_i, IV_j, IV_k, C^i + C^j + C^k, tag^i + tag^j + tag^k\}.$$

La station de base déchiffre alors $C^i + C^j + C^k - r^i - r^j - r^k = M^i + M^j + M^k = M$ à l'aide de sa connaissance des différentes clés de chiffrements et des différentes valeurs d' IV . M est un vecteur de taille l . Plus précisément : $M = (\sum_i M_0^i \bmod p, \dots, \sum_i M_{l-1}^i \bmod p)$. Elle vérifie ensuite

$$\begin{aligned} & tag^i + tag^j + tag^k - r^i - r^j - r^k \\ = & M^i \cdot x^n + M^j \cdot x^n + M^k \cdot x^n \bmod q(x) \\ = & (M^i + M^j + M^k) \cdot x^n \bmod q(x) \\ = & \left(\sum_i M_0^i \bmod p, \dots, \sum_i M_{l-1}^i \bmod p \right) \cdot x^n \bmod q(x) \\ = & M \cdot x^n \bmod q(x) \end{aligned}$$

Ainsi, la station de base peut vérifier la valeur des tags agrégés en fonction de la valeur somme reçue.

Exemple de tailles de valeurs à envisager. Pour ce qui concerne les tailles de clés et la taille de l' IV , nous conservons ici celles définies dans la Section 3.1. Intéressons nous à présent à la valeur de p . En ce qui concerne la définition de la valeur p , elle dépend de la taille du réseau et de la taille maximale de la valeur possible à envoyer. Si nous reprenons l'exemple d'un réseau de 200 nœuds et une température à envoyer inférieure à 5000 degrés, cela signifie que la taille de p est de l'ordre de 20 bits. Il s'agit alors de choisir un nombre premier facile à implémenter comme $2^{20} + 7$. Pour les valeurs proches des puissances de 2 classiques, on peut reprendre les nombres premiers donnés dans [Sar08]. Donc si $p = 2^{20} + 7$, on peut choisir un polynôme irréductible de degré 6 et envoyer les messages par paquets de 10. Ainsi, le MAC généré par chaque nœud sera de taille 126 bits, la concaténation des 10 messages chiffrés sera de taille 300 bits. Ainsi, si N nœuds transmettent leur valeur, la taille des paquets sera au maximum de $48N + 126 + 300$.

De même que fait dans la Section 3.1, on peut, pour réduire l'overhead, utiliser un filtre de Bloom chiffré ayant exactement les mêmes propriétés que précédemment.

3.3 Analyse de sécurité dans le modèle AMAC

Dans [CC08], A. Chan et C. Castelluccia proposent deux modèles de sécurité pour les schémas d'agrégation dans les réseaux de capteurs. Le premier concerne la notion de Concealed Data Aggregation (CDA), donc un modèle de sécurité pour l'agrégation de données chiffrées tandis que le deuxième modèle de sécurité concerne la sécurité des Aggregated Message Authentication Codes et donc l'agrégation de MAC.

Le modèle de sécurité proposé dans le cas de l'agrégation des données (CDA) définit la notion de sécurité contre les attaques adaptatives à chiffrés choisis et plus précisément l'indistingabilité dans ce modèle (IND-CCA2) en utilisant un jeu particulier autorisant

les types de challenges habituels dans ce modèle (oracle de chiffrement et oracle de déchiffrement). Il note notamment que la construction définie dans [CMT05] ne vérifie pas cette propriété. En effet, en raison de la nature même d'une telle construction, on peut distinguer deux chiffrés particuliers et la somme de ces deux chiffrés.

Dans le même article, les auteurs définissent la notion de sécurité AMAC en utilisant un oracle de génération et un oracle de vérification. Dans ce modèle, il montre qu'un adversaire est capable de gagner le jeu suivant : étant donné deux messages, $(hdr = \{i\}, m_i, tag_i)$ envoyé par un nœud i et $(hdr = \{j\}, m_j, tag_j)$ envoyé par un nœud j que l'adversaire voit passer ; si l'adversaire envoie à l'oracle de vérification l'agrégé de ces deux messages $(hdr = \{i, j\}, m_i + m_j, tag_i + tag_j)$ alors l'adversaire a réussi à construire un tag d'un message valide.

Le schéma proposé dans cet article possède également cette faille de sécurité. On peut cependant légitimement se poser la question des implications pratiques d'une telle attaque. En effet, si on suppose que la station de base a besoin de la valeur complète du header pour pouvoir déchiffrer correctement la somme des messages et l'agrégé des tags, cela signifie que l'adversaire (qui ne peut rejouer des paquets anciens en raison de l'intégration de la valeur de l'IV au header) ne fait que renvoyer à la station de base des informations qu'elle a déjà ou si elle ne les possède pas, cela signifie que les deux nœuds i et j n'ont pu transmettre leurs valeurs à la station de base (parce que par exemple leur nœud parent est mort). Dans ce dernier cas, l'adversaire aide en fait au fonctionnement du réseau.

Notons que dans le modèle de sécurité AMAC, le header n'est normalement pas pris en compte. On peut cependant imaginer redéfinir ce modèle en tenant compte du header. Dans ce cas, il est possible de faire reposer une partie du schéma sur la sécurité même du header en le chiffrant par exemple. La méthode la plus simple consiste à ne jamais le transmettre en clair sur le réseau en utilisant un algorithme tel l'AES paramétré par une clé unique partagée par tous les nœuds et la station de base. Cette méthode n'est cependant pas robuste utilisée seule contre la corruption de nœuds et ne permet pas de vérifier si le header n'a pas été modifié en cours de route. Pour cela, on peut ajouter à ce header chiffré une chaîne de MAC à laquelle chaque nœud contribue en surchiffrant les données comme fait dans [CPS06]. La seule contrainte est alors de ne surtout pas utiliser un chiffrement additivement homomorphe.

4 Conclusion

Nous venons de présenter une méthode simple utilisant les fonctions de hachage universelles permettant l'agrégation des MAC dans un réseau de capteurs. Pour cela, nous avons étendu un des schémas proposés par Krawczyk afin de simplifier le traitement de l'information. Nous avons également discuté de la sécurité d'un tel schéma dans un modèle de sécurité pré-défini.

Nous souhaitons dans la suite de notre travail pouvoir ajouter des résultats de simulation afin d'étudier les gains potentiels d'une telle méthode. Notons également qu'en raison de la petite taille des messages à envoyer sur un réseau de capteurs, envoyer une seule valeur signifie que le MAC nécessaire à une telle opération correspond plutôt à la définition d'une fonction d'expansion que d'une fonction de compression. Nous souhaiterions donc dans la suite de notre travail définir de telles fonctions et également implanter sur des capteurs des fonctions de hachage universelles afin de tester leurs performances dans un

tel contexte.

Références

- [ACH06] I. Aad, C. Castelluccia, and J.P. Hubaux. Packet coding for strong anonymity in ad hoc networks. In *IEEE Securecomm*, pages 1–9, August 2006.
- [AGW05] Mithun Acharya, Joao Girao, and Dirk Westhoff. Secure comparison of encrypted data in wireless sensor networks. In *WIOPT '05 : Proceedings of the Third International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks*, pages 47–53, Washington, DC, USA, 2005. IEEE Computer Society.
- [Ber05] Daniel J. Bernstein. The poly1305-aes message-authentication code. In Henri Gilbert and Helena Handschuh, editors, *FSE*, volume 3557 of *Lecture Notes in Computer Science*, pages 32–49. Springer, 2005.
- [BHL06] Raghav Bhaskar, Javier Herranz, and Fabien Laguillaumie. Efficient authentication for reactive routing protocols. In *AINA (2)*, pages 57–61. IEEE Computer Society, 2006.
- [CC08] Aldar C-F. Chan and C. Castellucia. On the (im)possibility of aggregate message authentication codes. In *IEEE International Symposium on Information Theory ISIT 2008*, 2008.
- [CMT05] C. Castellucia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Mobile and Ubiquitous Systems : Networking and Services, 2005. MobiQuitous 2005*, pages 1–9, July 2005.
- [CPS06] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *CCS '06 : Proceedings of the 13th ACM conference on Computer and communications security*, pages 278–287, New York, NY, USA, 2006. ACM.
- [CW79] Larry Carter and Mark N. Wegman. Universal Classes of Hash Functions. *Journal of Computer and System Sciences - JCSS*, 18(2) :143–154, 1979.
- [EJ02] Patrik Ekdahl and Thomas Johansson. A new version of the stream cipher snow. In Kaisa Nyberg and Howard M. Heys, editors, *Selected Areas in Cryptography*, volume 2595 of *Lecture Notes in Computer Science*, pages 47–61. Springer, 2002.
- [HE03] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *In Workshop on Security and Assurance in Ad hoc Networks*, pages 384–394, 2003.
- [HP08] Helena Handschuh and Bart Preneel. Key-recovery attacks on universal hash function based mac algorithms. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 144–161. Springer, 2008.
- [KL08] Jonathan Katz and Andrew Y. Lindell. Aggregate message authentication codes. In *CT-RSA*, pages 155–169, 2008.
- [Kra94] Hugo Krawczyk. Lfsr-based hashing and authentication. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 129–139. Springer, 1994.

- [PSP03] Bartosz Przydatek, Dawn Song, and Adrian Perrig. SIA : Secure information aggregation in sensor networks. In *ACM SenSys 2003*, Nov 2003.
- [Pub85] Nist Publication. Computer data authentication, 1985.
- [Pub02] Nist Publication. The keyed-hash message authentication code (HMAC), 2002.
- [Rab81] Michael O. Rabin. Fingerprinting by random polynomials, 1981.
- [Sar08] Palash Sarkar. A New Universal Hash Function and Other Cryptographic Algorithms Suitable for Resource Constrained Devices. Cryptology ePrint Archive, Report 2008/216, 2008. <http://eprint.iacr.org/>.
- [Sho96] Victor Shoup. On Fast and Provably Secure Message Authentication Based on Universal Hashing. In *Advances in Cryptology - CRYPTO '96*, Lecture Notes in Computer Science 1109, pages 313–328. Springer-Verlag, 1996.
- [YWZC06] Yi Yang, Xinran Wang, Sencun Zhu, and Guohong Cao. Sdap : a secure hop-by-hop data aggregation protocol for sensor networks. In *MobiHoc '06 : Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 356–367, New York, NY, USA, 2006. ACM.