Aggregated Authentication (AMAC) Using Universal Hash Functions

Wassim Znaidi[1], Marine Minier[1], and Cédric Lauradoux[2]

[1] CITI Laboratory, Lyon University
F-69621, France
firstname.lastname@insa-lyon.fr

[2] UCL / INGI / GSI, Place Saint Barbe, 2
Louvain-la-Neuve, Belgique
cedric.lauradoux@uclouvain.be

Abstract. Aggregation is a very important issue to reduce the energy consumption in Wireless Sensors Networks (WSNs). There is currently a lack of cryptographic primitives for authentication of aggregated data. The theoretical background for Aggregated Message Authentication Codes (AMACs) has been proposed by Chan and Castelluccia at ISIT 08.

In this paper, we propose a MAC design based on universal hash functions and more precisely on the Krawczyk's constructions. We show how those designs can be used for aggregation and how it can be easily adapted for WSNs. Our two AMAC constructions offer a small memory footprint and a signification speed to fit into a sensor. Moreover, when compared with scenarios without aggregation, the method proposed here induces a simulated energy gain between 3 and 9.

Keywords: Sensor networks, aggregation, authentication, MACs.

1 Introduction

The purpose of wireless sensors networks (WSNs) is to collect data and then transmit them at a gathering point. There are two classes of nodes in such a network. *Data nodes* have limited resources (CPU, memory and energy) and are on their own, *i.e.* the energy is the critical resource. *Gathering nodes* are considered more powerful (base stations) and they have an access to a power supply. We consider in this paper, an *hop-by-hop* scheme for the data forwarded by the *data nodes* to a *gathering node* considering a fixed topology. In this model, the most expensive operation is the data transmission. It is then highly valuable to reduce the size and the number of the transmitted messages. The messages aggregation has been used for this purpose. We apply a function over all the data produced by the collection nodes instead of concatenating them. This function is evaluated successively by each data gathering node resulting in a communication scheme with a constant message size.

If aggregation is a very powerful technique to save energy, it has to be used carefully. Sensors can be deployed at a large scale and over a large area. It is very likely that they get compromised or attacked and thus an attacker can influence the result of the aggregation function [22]. Security is therefore a critical issue. The confidentiality, the integrity and the origin of aggregated data must be preserved. There exists several works concerning aggregated encryption [10,6,8] or aggregated authentication [19,23,15]. Recently, several works [7,16,3] have established the foundation for aggregated MACs. This work aims to fill the gap between the theoretical results and practical design for aggregated MACs.

A MAC algorithm could be seen as a signature only valid between two users that share the same secret key: a MAC allows to guarantee the integrity of the transmitted message and to verify the identity of the sender for the user sharing the symmetric key. A MAC is thus an algorithm that takes as input a message m and a key K and that produces a fingerprint $tag = MAC_K(m)$. The receiver of the message $m'||tag(m)$ verifies if $tag' = MAC_K(m')$ is equal or not to tag. In the case of a WSN, this verification must be performed at each stage of the aggregation to establish a complete trust chain over all the results.

In this paper, we propose two AMAC constructions based on the well-known universal hash functions, *i.e. CRC Hash* and *LFSR hash* proposed by Krawczyk in [17].

Contributions of the paper are as follows:

- We show how to use existing universal hash functions to design an aggregated MAC scheme and which level of security can be achieved.
- We identify the parameters of the functions suitable for the constraints of sensors.
- We present a comparison of the performances of several aggregation scenarios.

The simulations performed for different scenarios show that the gain in terms of energy between our method and methods without aggregation varies between 3 and 9.

In Section 2, we give a reminder on aggregation and message authentication codes. We particularly focus on universal hash functions based MACs. We present in Section 3 our new designs for aggregate MACs and we discuss the security issues. The performance of our schemes are evaluated in Section 4. Then, we conclude.

2 Preliminaries

In this section, we first introduce a formal definition of message aggregation and of AMAC as done in [7] and describe the relative constructions proposed in the literature. Then, we introduce three particular MAC designs which are linear and based upon universal hash functions.

2.1 Formal Definition of Aggregation and Related Work

The basic communication model in an hop-by-hop WSNs is the concatenation. Let consider a WSN with n nodes sending message of ℓ bits. Each node i concatenates its contribution x_i to the result of the previous step. This model is easily implemented but it consumes a significant amount of bandwith and energy: the last node in the protocol has for instance to transmit $x_1, x_2, \cdots, x_i, \cdots, x_{n-1}, x_n$. This overhead can be reduced by using the aggregation: given n messages $(x_1, x_2, \cdots, x_n)$ of length ℓ sent by the n different nodes, the aggregated result m of length ℓ is defined by a function f:

$$f : m = f(x_1, x_2, \cdots, x_n).$$

Some examples of aggregation functions usually used are the median or the mean as explained in [22]. The security of an aggregation scheme relies on encryption and authentication. The properties of those two mechanisms are very specific in the context of aggregation.

Encrypted and Aggregated Data. The confidentiality of the messages sent in an aggregation scheme requires to perform the aggregation over the ciphertexts rather than the plaintexts. This problem is usually solved with homomorphic ciphers. Many public-key cryptosystems, *e.g.* RSA or ElGamal, can be used for this purpose but they are not generally suitable for sensors. A method of homomorphic ciphers based upon stream ciphers and suitable for WSNs applications has been developed in [6]. Let consider a node i receiving an ℓ-bit message p_{i-1}. The node i aggregates its contribution x_i to the message p_{i-1} in the following way:

$$\begin{aligned} p_i &= p_{i-1} + c_i \mod q \\ &= p_{i-1} + x_i + keystream_i \mod q \end{aligned}$$

where q is a well chosen prime number and where $keystream_i$ is the keystream produced by the node i with its secret key k_i and a stream cipher.

Aggregated Authentication. The aim of aggregated authentication is to provide a way to verify the aggregated result, *i.e.* $f(x_1, x_2, \cdots, x_i, \cdots)$, rather than each message x_i individually. Different schemes have been proposed for the aggregation of authentication. They used Merkle tree [19] or MACs algorithms [23,3,15,7]. We especially focus on the solution proposed by A. Chan and C. Castelluccia [7]. They have proposed a formalization of aggregate message authentication code (AMAC) and they study its security. More formally, an AMAC algorithm is defined as follows:

– Key Generation (KG). Let $\text{KG}(1^\lambda, n) \rightarrow (k_1, k_2, ..., k_n)$ be a probabilistic algorithm. Then, k_i (with $1 \leq i \leq n$) is the secret key used to generate a verification tag by node i. The gathering node also called the sink possesses all k_i's used for tag verification.
– Tag Generation (MAC). $\text{MAC}_{k_i}(x_i) \rightarrow tag_i$ takes a secret key k_i and a message x_i as input to generate a verification tag tag_i for x_i. The message sent out from node i is a 3-tuple $(\{i\}, x_i, tag_i)$.
– Tag Verification (Ver). Let m be an f-aggregate of messages $x_1, x_2, \cdots, x_i, \cdots$ and hdr be the set of all contributing identities. Then

$$\text{Ver}_{k_1,k2,\cdots,k_i,\cdots}(m, tag_1, tag_2, \cdots, tag_i, \cdots) \rightarrow 0/1$$

takes the aggregate m and the tag tag_i and secret key k_i for each $i \in hdr$ and outputs 1 if m is a correct aggregate (i.e. $m = f(x_1, x_2, \cdots, x_i, \cdots)$) and 0 otherwise.

Note that no aggregation algorithm is specified in AMAC; the aggregation is done in plaintexts. When an aggregating node with identity k receives two measurement values and their tags from downstream, say, $(\{i\}, x_i, tag_i)$ and $(\{j\}, x_j, tag_j)$, it would pass

$$(\{i, j, k\}, f(x_i, x_j, x_k), tag_i, tag_j, tag_k)$$

as the aggregation result to its parent where x_k is its own measurement.

Note also that aggregation of verification tags is not considered here. So all the tags are needed in the verification: let $m = f(x_1, ..., x_i, ...)$, then the correctness requirement of AMAC is as follows:

$$\text{Ver}_{k_1,\cdots,k_i,\cdots}(m, MAC_{k_1}(x_1), ..., MAC_{k_i}(x_i), ...) = 1.$$

here, the tags are not aggregated.

At this time, no instance of this scheme has been proposed. We propose to use the universal hash functions defined by Carter and Wegman in [4] to design a MAC corresponding to the requirements of [7] and where the tags could also be aggregated leading to an AMAC scheme defined as follows: let $m = f(x_1, \cdots, x_i, \cdots)$ and tag be the value $g(MAC_{k_1}(x_1), ..., MAC_{k_i}(x_i), ...)$ be the tag aggregation considering that g is an aggregation functions that could be (or not) equal to f, then the corresponding verification is thus:

$$\text{Ver}_{k_1,\cdots,k_i,\cdots}(m, tag) = 1.$$

2.2 MACs Based Upon Universal Hash Functions

In this section, we will first introduce the definition of an universal hash function and the original MAC schemes proposed by Krawczyk based upon universal hash functions and the one proposed by Sarkar.

Universal Hash Functions. A universal hash function is a family of functions indexed by a parameter called the key and it must verify that the probability over all keys that all distinct inputs collide is small. This notion was introduced by Carter and Wegman in [4].

Definition 1. *Let f_k be a function of an (ℓ, n)-family H from an ℓ-bit set to an n-bit set with the parameter k taken in a set $\mathcal{K}$. The family H is ϵ-almost universal if the probability of collisions for a random distribution of the value k over the set $\mathcal{K}$ (i.e. $Pr_k(f_k(M) = f_k(M')), \forall k \in_R \mathcal{K}$) is smaller than ϵ. We also say that H is ϵ-almost XOR universal (ϵ-AXU) if the associated differential probability for a random distribution of the value k over the set $\mathcal{K}$ is bounded by ϵ, i.e. $\forall(M, M', a), Pr_k(f_k(M) - f_k(M') = a) \leq \epsilon$.*

Definition 2. *We also say that a family of functions H is $\oplus$-linear if for all M, M', we have $f_k(M \oplus M') = f_k(M) \oplus f_k(M')$ for all instance f_k in H.*

The Definition 2 is particularly important for aggregated authentication.

These functions can be used for message authentication if the output is processed with another function. A MAC designs using the Definition 1 assumed the following scenario: the parties have already exchanged their secret key k, then to exchange a message M of length ℓ, the sender sends M and the tag $tag = f_k(M) \oplus r$. The shared secret key k is thus composed of a particular f_k function drawn randomly from an (ℓ, n)-family of hash functions and a random pad r. At reception, the receiver verifies the "tag" tag, corresponding with the MAC will be recomputed and checked for consistency. In practice, the fingerprint $f_k(M)$ will be encrypted with a stream cipher that will produce r.

Krawczyk has shown in [17] that the design of MAC of the above kind (i.e. combined with a one-time pad) requires to have a family of functions that is ϵ-almost XOR universal. Moreover, the family of functions can also be $\oplus$-linear.

Many universal hash families have been proposed in the literature to build MACs. One of the first examples was the evaluation of a particular polynomial in a particular point k as done in [2]. Let consider an ℓ-bit message m split into t blocks m_i such that $\ell = p^t$. In this case, the universal hash function f_k is defined as:

$$f_k(m_1, \cdots, m_t) = \sum_{i=1}^{t} m_i \cdot k^i \mod p.$$

This function is multi-linear and the base field could be $\mathbb{F}_p$ or $\mathbb{F}_{2^n}$. In [21], V. Shoup gave a classification of the universal hash functions that could be used for MACs constructions in 3 categories: The first one is composed of the polynomial evaluations over a prime field or a finite field; the second one is composed of polynomial divisions over $\mathbb{F}_2$ described by Krawczyk in [17] and known as cryptographic CRC and as "LFSR (Linear Feedback Shift Register) hash"; the third category is composed of polynomial division over $\mathbb{F}_{2^k}$. V. Shoup particularly studied in his article the last class. The reader can find more details on MAC algorithms based on universal hash functions in [18] and on their security in [13].

Cryptographic CRC. As described in [21], the first scheme proposed by H. Krawczyk in [17] is based upon modular division using an irreducible polynomial over the field $\mathbb{F}_2$. It is a cryptographic variant of the well-known Cyclic Redundancy Codes (CRC), standards for errors detection in networks. More precisely, each message M is seen in its equivalent polynomial representation $M(x)$ over the field $\mathbb{F}_2$, the coefficients being the bits of M. Thus, for each irreducible polynomial $q(x)$ of degree n over $\mathbb{F}_2$, the associated family of universal hash functions is $h_q(M) = M(x) \cdot x^n \mod q(x)$. Notice here that it is necessary to multiply $M(x)$ by x^n to ensure the security of the scheme for the notion of ϵ-AXU.

The (ℓ, n) family of hash functions h_q is the set of irreducible polynomials of degree n and of the messages of size ℓ. This family is $\oplus$-linear, ϵ-almost universal (with $\epsilon \leq \frac{n+\ell}{2^{n-1}}$) and ϵ-almost XOR universal (with $\epsilon \leq \frac{n+\ell}{2^{n-1}}$).

The hardware and software implementation of such mechanisms is really efficient because the modular division for polynomials in $\mathbb{F}_2$ could be performed using a simple LFSR. The corresponding extension proposed by Shoup and proved secure is the extension of this construction to the case where the base field is $\mathbb{F}_{2^k}$. In this case, the corresponding ϵ value is about $\frac{\ell}{2^{kn}}$.

Linear Feedback Shift Register (LFSR) Hashing. In the same article, Krawczyk introduced a second construction based upon random matrices. More precisely, given A a boolean Toeplitz matrix of size $n \times \ell$ (i.e. each lower diagonal is fixed, i.e. if $k-i = l-j$ for all indices then $A_{i,j} = A_{k,l}$) and given a message M of size ℓ. The universal hash function $h_A(M)$ is then the binary multiplication of the matrix A by the column vector composed of the bits of the message M: $h_A(M) = A \cdot M$.

A simple method to build such matrices is the LFSR use: given $q(x)$ an irreducible polynomial of degree n over $\mathbb{F}_2$; given $s_0, s_1, \cdots$ the output sequence of the bits generated by the LFSR defined according to $q(x)$ and the initial state of the LFSR $s = (s_0, s_1, \cdots, s_{n-1})$. For each irreducible polynomial $q(x)$ and for each non-zero initial state of the LFSR, we associate the hash function $h_{q,s}(M)$ defined as the linear combination $\bigoplus_{j=0}^{\ell-1} M_j \cdot (s_j, s_{j+1}, \cdots, s_{j+n-1})$ where M_j is the bit number j of M. In other words, at each clock, the LFSR updates its internal state taking into account each message bit. This hash functions family is $\oplus$-linear, ϵ-almost universal (with $\epsilon \leq \frac{n+\ell}{2^{n-1}}$ and ϵ-almost XOR universal (with $\epsilon \leq \frac{\ell}{2^{n-1}}$).

Multi-linear Universal Hash Functions. In [20], P. Sarkar proposed the following evaluation: given a field $\mathbb{F}_p$ and an extension of this field $\mathbb{F}_{p^n}$ with $n \geq 1$; given ϕ a linear transformation from $\mathbb{F}_{p^n}$ into itself such as the minimal polynomial of ϕ in $\mathbb{F}_p[x]$ be of degree n and be irreducible over $\mathbb{F}_p$; the message to cipher M is cut into $l \leq n$ elements $(M_1, \cdots M_l)$ over $\mathbb{F}_p$. The hash functions family is:

$$\begin{aligned} G_K(M) &= M \cdot (K, \phi(K), \cdots, \phi^{l-1}(K)) \\ &= M_1 K + M_2 \phi(K) + \cdots + M_l \phi^{k-l}(K) \end{aligned}$$

where K belongs to $\mathbb{F}_{p^n}$.

The family G_K is thus a linear combination of $(M_1, \cdots M_l)$ and of $(K, \phi(K), \cdots, \phi^{l-1}(K))$, it is multi-linear (i.e. linear in each of its component), ϵ-almost universal with $\epsilon \leq 1/q^n$ and also ϵ-almost XOR universal with the same ϵ value.

Sarkar also noted that ϕ could be easily implemented because it can be seen as a LFSR over $\mathbb{F}_p$. The author studied the particular p values allowing a fast implementation in hardware and in software. The examples given are $q = 2$, $n = 128$; $q = 2^8 + 1$, $n = 16$; $q = 2^{16} + 1$, $n = 8$; $q = 2^{32} + 15$, $n = 4$. He also gave some examples of extensions over the field $\mathbb{F}_2$ that we will not detail here.

3 New Designs

We are going to present in this section the possible applications of the previous functions in the case of a WSN. We simplify the study case for a better understanding and reduce the number of nodes to two nodes i and j depending on

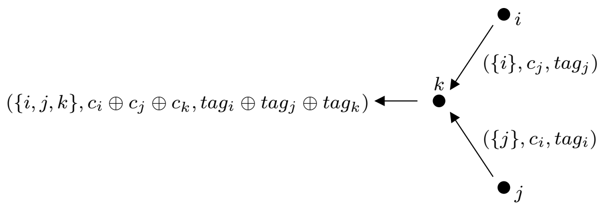
i
$(\{i\}, c_j, tag_j)$
k
$(\{i, j, k\}, c_i \oplus c_j \oplus c_k, tag_i \oplus tag_j \oplus tag_k)$
$(\{j\}, c_i, tag_i)$
j

Fig. 1. XOR aggregation with three nodes: i, j and k

one aggregator node k as shown in Fig. 1. This last one is directly connected to the sink. This simple scheme could be easily generalized.

3.1 XOR Aggregation: How to Adapt the Krawczyk's Approaches for WSNs

The first construction described by Krawczyk could be directly applied to the MAC aggregation if the XOR operation is used. This approach could be directly combined with the XOR data aggregation proposed in [6].

Suppose that a WSN is composed of N nodes i. Each node receives during an initialization phase a cipher key K_{Ei} shared between the node and the base station (the sink), an authentication key K_{Ai} also shared between the node and the base station and a polynomial $q(x)$ shared by all the nodes and the base station. Suppose now that the simple network described in Fig. 1 describes a tree with three nodes i, j and k directly rooted at the sink.

When the node i wants to send (on demand or at regular intervals) a message m_i, it ciphers this message using a pseudo-random stream generated using a stream cipher algorithm E (for example RC4 or SNOW v2 [11]). The common cipher key K_{Ei} and a common initial value IV_i used once must be shared between the node i and the base station for the correct use of the E stream cipher. Thus, node i ciphers $C_i = m_i \oplus r_i$ where r_i is the pseudo-random stream produced by $E(K_{Ei}, IV_i)$. Note that as mentioned in [6] the node i must also transmit to the base station its Id i and the unique value IV_i that also plays the role of a counter to discard replay attacks. We first propose that IV_i be the only transmitted value writing $IV_i = i||CTR_i$, i.e. as a concatenation between the node Id and a counter CTR_i incremented by 1 at each new sending. (Note also that this value must be transmitted each time in case of the non-transmission of a particular message).

The node i also produces the corresponding MAC of the message m_i using the Krawczyk's construction: it computes $tag_i = (m_i(x) \cdot x^n \mod q(x)) \oplus r'_i$ where r'_i is the pseudo-random stream produced using E initialized with K_{Ai} and IV_i. Thus, node i transmits to its parent k the value: $\{hdr, data, tag\}$ with $hdr = IV_i$, $data = C_i$, $tag = tag_i$.

Suppose now that the node j wants to transmit the message m_j to the aggregator node k, then it sends to k $\{IV_j, C_j, tag_j\}$. The node k transmits to the base station (considering that it sends itself m_k):

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

The base station deciphers

$$C_i \oplus C_j \oplus C_k \oplus r_i \oplus r_j \oplus r_k = m_i \oplus m_j \oplus m_k = M$$

using the knowledge of the different keys and of the different IVs. It verifies:

$$\begin{aligned} & tag_i \oplus tag_j \oplus tag_k \oplus r'_i \oplus r'_j \oplus r'_k \\ &= m_i \cdot x^n \oplus m_j \cdot x^n \oplus m_k \cdot x^n \mod q(x) \\ &= (m_i \oplus m_j \oplus m_k) \cdot x^n \mod q(x) \\ &= M \cdot x^n \mod q(x). \end{aligned}$$

Thus, the base station could verify the aggregated tags according to the received sum.

Examples of values sizes Concerning the key sizes, the minimal size is 128 bits. The polynomial $q(x)$ could be a primitive polynomial of size 64 bits. the IV values could be of 48 bits length (24 bits for example for the node Id and 24 bits reserved for the counter[1]. The messages must have a 64 bits length (this value could not be smaller than the degree of the polynomial q). In this case, the final messages size is defined according the number of nodes N transmitting an information: $48N + 64 + 64$. Considering a 96 bits polynomial and a 96 bits message to transmit, the total size of transmitted information becomes $48N + 96 + 96$. The security bounds of the underlying universal hash functions are equal to 2^{-56} (resp. 2^{-87}). The induced overhead on the network clearly depends on the header size and thus on the nodes number sending back an information.

The following on demand mechanism could reduce this overhead: when the base station wants to receive values from the network, it broadcasts a unique value IV of 24 bits. The nodes receiving it use the previous method to cipher and to authenticate their data using the IV value $IV_i = i||IV$. The header has only to be constituted of the responding Id nodes and could thus be replaced by a ciphered Bloom filter of size m as proposed in [1]. The cipher key K of the filter is shared between all the nodes and the base station. In this case, each node ciphers k times its Id by putting at 1 the bit corresponding to this position in the Bloom filter. This improvement seems to be efficient even if the probability to obtain a false positive in the Bloom filter is about $0,6185^{m/n}$ where n is the number of elements to insert in considering that k is about $0,7\frac{m}{n}$. For a network with 200 nodes, a Bloom filter of size 2048 bits with $k = 7$ has a false positive probability less than 1%. This probability could be reduced if instead of bits, the Bloom filter is composed of 4 bits word or of bytes. If a Bloom filter is used, the required computations performed by each sensor are increased by k additional

[1] Once all the IVs values used, the cipher keys must be changed to discard WEP like attacks.

hash computations and the base station must test if all the Id nodes are or are not in the Bloom filter leading to kN additional hash computations.

We have proposed a direct use of the Krawczyk constructions in the case where we try to obtain the XOR of the messages and not the sum. We have presented examples using the first construction, the same reasoning could be applied using the second construction. In this case, each node must be initialized with the same matrix A to conserve the $\oplus$-linearity, the other parameters being the same than the one previously described. A deduced MAC size of 64 or 96 bits gives reasonable security bounds.

3.2 Aggregation over $\mathbb{F}_p$

In this section, we extend the Krawczyk's MAC constructions over $\mathbb{F}_p$ with p prime to transform the proposed MAC from $\oplus$-linear to $+$-linear as the one proposed by Sarkar.

Extension of Krawczyk's Over $\mathbb{F}_p$. We first introduce the following notations: we denote by $\mathbb{F}_p^l$ the vectorial space of size l over $\mathbb{F}_p$ and $\mathbb{F}_p^n$ the one of size n. The message M we want to compute the MAC is written $M = (M_0, \cdots, M_{l-1})$ where each M_i belongs to $\mathbb{F}_p$. In the same way, the output of the hash function is considered as an element of $\mathbb{F}_p^n$, a vector of size n over $\mathbb{F}_p$.

In this case, the first Krawczyk's construction becomes: given a message $M = (M_0, \cdots, M_{l-1})$ in $\mathbb{F}_p^l$ seen as a polynomial with coefficients in $\mathbb{F}_p$: $M(x) = M_0x^{l-1} + M_1x^{l-2} + \cdots + M_{l-1}$; given an irreducible polynomial $q(x)$ of degree n with coefficients in $\mathbb{F}_p$ with its leading coefficient equal to 1; the hash function is defined as $h_q(M) = M(x) \cdot x^n \mod q(x)$ seen as a vector of size n of elements of $\mathbb{F}_p$. The obtained tag is thus $tag = h_q(M) + r$ where r is a vector of size n of random numbers taken in $\mathbb{F}_p$.

We need to compute $Pr_h(h_q(M) - h_q(M') = c(x))$ to prove that this function is ϵ-almost universal. This function is trivially $+$-linear. Thus, directly using the results of [21] and of [17], if $h_q(M) - h_q(M') = c(x)$, by linearity, we have $h_q(M - M') = c(x)$, i.e. $q(x)$ divides $(M - M') \cdot x^n - c(x)$. This polynomial has a maximal degree equal to $l + n$ whereas $q(x)$ has a degree of n, the number of factors of degree n of this polynomial is $\frac{n+l}{n}$. The maximal number of functions h_q that map $M - M'$ in $c(x)$ is $\frac{n+l}{n}$. The number of all possible functions is the set of irreducible polynomials of degree n over $\mathbb{F}_p[x]$; there are $\frac{p^{n-1}}{n}$ such functions. We thus directly deduce that $Pr_h(h_q(M) - h_q(M') = c(x)) \leq \frac{n+l}{p^{n-1}}$ and that the family of proposed functions is ϵ-almost $+$ universal with $\epsilon = \frac{n+l}{p^{n-1}}$ and it is multi-linear.

We have generalized the first Krawczyk's construction for a prime field $\mathbb{F}_p$ keeping the willing linear properties. Notice that the second scheme proposed in [17] could be generalized in the same way: in this case, this last generalization is very closed to the one proposed by Sarkar except the definition of the final set seen in the Krawczyk's generalization as a vectorial space ans seen in the Sarkar's construction as a field extension. Let us now explain how to use those generalizations for WSNs.

Applications for WSNs. In this section, we will describe how to use those new constructions for MAC aggregation in WSNs. We suppose here that the messages (for example temperatures) sensed by the nodes will be sent by packets, one packet being constituted of l single messages of size p. Each node sensing l different values stores those values $M_0, \cdots, M_{l-1}$ before sending them together to the base station at each given time interval or on demand, l being known and fixed in advance. In the case where a sensor has not collected l values but a smaller number, it replaces in the sent message the missing values by some 0s. Let us illustrate the proposed method using the example of Fig. 1.

Using the same assumptions than the one of Section 3.1, we suppose that each node i shares with the base station a cipher key K_{Ei}, an authentication key K_{Ai}, a stream cipher E and an irreducible polynomial $q(x)$ of degree n with coefficients in $\mathbb{F}_p$ common with all the other nodes and the base station.

The node i stores l messages and sends them to the base station using the proposed method. For ciphering messages, it directly uses the method described in [6], i.e. it computes for each message M_j $(j \in [0,..,l-1])$, $M_j + r_j \mod p$ where r_j is a pseudo-random number smaller than p and where p is a prime number defined as $p \geq 2^{\lceil \log_2 M*N \rceil}$ (where M is the maximal size that can take a single message) and where N is always the total number of nodes. So, first, node i ciphers its l messages $M^i = (M^i_0, \cdots, M^i_{l-1})$:

$$\begin{aligned} C^i &= M^i + r^i \\ &= (C^i_0 = M^i_0 + r^i_0 \mod p, \cdots, \\ &\quad C^i_{l-1} = M^i_{l-1} + r^i_{l-1} \mod p) \end{aligned}$$

where each r^i_j is a random number in $[0, p-1]$. Those values are obtained using algorithm E initialized with the key K_{Ei} and an IV value IV_i. From this point, node i computes the MAC of all the l messages M^i seen as a polynomial with coefficients in $\mathbb{F}_p$: it first computes a vector of size n: $h_q(M^i) = (h^i_0, \cdots, h^i_{n-1}) = (M^i \cdot x^n \mod q(x))$. Finally, we have:

$$\begin{aligned} tag^i + r'^i &= (tag^i_0, \cdots, tag^i_{n-1}) \\ &= (h^i_0 + r'^i_0 \mod p, \cdots, \\ &\quad h^i_{n-1} + r'^i_{n-1} \mod p) \end{aligned}$$

where r'_j are random numbers belonging to $[0, p-1]$ obtained using E initialized with K_{Ai} and IV_i. Node i transmits to its parent node k: $\{hdr, data, tag\}$ with $hdr = IV_i$, $data = C^i$ and $tag = tag^i$.

Following the same previous example, the node j depending on the same parent k transmits its own l messages and the associated MAC: $\{IV_j, C^j, tag^j\}$. The node k transmits to the base station (considering that it has also l messages to transmit):

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

The base station deciphers $C^i + C^j + C^k - r^i - r^j - r^k = M^i + M^j + M^k = M$ using its knowledge of each K_{Ei} and of IV_i. M is a vector of size l. More precisely: $M = (\sum_i M^i_0 \mod p, \cdots \sum_i M^i_{l-1} \mod p)$. It then verifies:

$$\begin{aligned}
&tag^i + tag^j + tag^k + r'^i + r'^j + r'^k \\
&= M^i \cdot x^n + M^j \cdot x^n + M^k \cdot x^n \mod q(x) \\
&= (M^i + M^j + M^k) \cdot x^n \mod q(x) \\
&= (\sum_i M_0^i \mod p, \cdots, \\
&\quad \sum_i M_{l-1}^i \mod p) \cdot x^n \mod q(x) \\
&= M \cdot x^n \mod q(x)
\end{aligned}$$

Thus, the base station is able to verify the value of the aggregated tags according the received sum value.

Examples of values sizes. As previously mentioned, for key and IV sizes we keep the sames as the ones defined in Section 3.1. So, let us define the p value. It depends on the size of the network and of the maximal size of the message to send. With the temperature example and a network composed of 200 nodes, we consider that the temperature is smaller than 5000 Celsius degree. we directly deduce that the p size is about 20 bits. We thus need to choose a prime number easy to implement such as $2^{20} + 7$. For the particular values closed to the powers of 2, we could choose the prime numbers given in [20]. So if $p = 2^{20} + 7$, an irreducible polynomial of degree 6 gives an admissible security bounds and we can send the messages by packets of size 10. Thus the generated MAC for each node will be of size 126 bits, the concatenation of 10 messages will be of length 300 bits. If N nodes transmit their values, the packets size will be bounded over by $48N + 126 + 300$.

As explained in Section 3.1, a ciphered Bloom filter (with the same properties) could be used for overhead reductions.

3.3 Security Analysis in the AMAC Model

In [7], A. Chan and C. Castelluccia proposed two security models for aggregation schemes in WSNs. The first one is called Concealed Data Aggregation (CDA) and concerns a security model for data aggregation whereas the second one called AMAC is dedicated to the security of Aggregated Message Authentication Codes.

The security model for CDA defines a security notion against adaptive chosen ciphertexts attacks and the indistinguishability notion in this model (IND-CCA2). The particular game used here authorizes the usual challenges in this model (cipher oracle and decipher oracle). The authors noticed that the construction defined in [6] does not verify the IND-CCA2 property. This comes directly from the intrinsic nature of the construction: we could distinguish two particular ciphertexts and their sum.

In the same article, the authors defined the AMAC security notion using a generation oracle and a verification oracle. In this model, they demonstrated that an adversary is able to win the following game: given two messages, $(hdr = \{i\}, m_i, tag_i)$ sent by node i and $(hdr = \{j\}, m_j, tag_j)$ sent by node j that the adversary knows; if the adversary sends to the verification oracle the aggregated of those two messages ($hdr = \{i, j\}, m_i + m_j, tag_i + tag_j$) then the adversary is able to forge a tag for a valid message.

The schemes proposed here have this security weakness intrinsically linked with the wishing linear properties. However, we could legitimately question the practical implications of a such attack especially concerning our MAC schemes. The discussion below only concerns the AMAC case of study, the data aggregation scheme used here being the one studied in [7]. Indeed, if we suppose that the base station needs the complete value of the header to correctly decipher the messages sum and the aggregated value of the tags, this implies that the adversary (that can not replay old packets due to the presence of the IV value into the header) could only send to the base station information that it already knows. If the base station does not possess those information, this implies that the two nodes i and j were not able to transmit their values to the base station (because for example the parent node is dead). In this last case, the adversary helps for good operations in the network.

Moreover, in the AMAC security model proposed in [7], the header is not taken into account. We can imagine to redefine a security model where the header is included. In this case, the security of the scheme could rely in part at least on the header itself by ciphering it for example or by using a ciphered Bloom filter as explained in Section 3.1. One of the simplest methods consists in always ciphering the header using the AES and a unique key shared by all the nodes in the network and the base station. This method is not robust against node compromise and do not allow to verify if the header has been modified during process or not. To discard this last problem, we could add to this ciphered header a MAC chain for which each node contributes by over-ciphering data as done in [9]. The only deduced constraint is that the use of an additive homomorphic cipher is prohibited to cipher and authenticate the header.

4 Performance Comparison

In this section, we present the performance evaluation of different aggregation models. We have considered four cases:

- **Scenario 1**: the communication scheme uses no aggregation, *i.e.* the concatenation, nor for the data neither for the authentication.
- **Scenario 2**: the data are aggregated using a stream cipher as proposed in [6]. The authentication is not aggregated.
- **Scenario 3**: the data and the authentication are respectively aggregated with a stream cipher over $\mathbb{F}_2$ and with the AMAC proposed in Section 3.1.
- **Scenario 4**: the data and the authentication are respectively aggregated with a stream cipher [6] and with the AMAC proposed in Section 3.2.

We test those four schemes using the LEACH [14] election mechanism and the WSnet simulator [12]. First, we briefly describe LEACH, the simulation parameters and we discuss the different results.

4.1 LEACH: Low-Energy Adaptative Clustering Hierarchy

LEACH [14] is a clustering-based protocol which minimizes energy dissipation in WSN's. LEACH selects randomly nodes as cluster heads (special aggregator nodes), so the energy dissipation in the communications with the sink is spread to all nodes in the network. LEACH is composed of two steps: the set-up phase and the steady phase. During the set-up phase, a sensor node is elected as cluster head if it generates a random number (between 0 and 1) greater than a given threshold T defined as:

$$T = \begin{cases} \frac{P}{1-P*[r \mod 1/P]}, & \text{if } n \in G \\ 0, & \text{Otherwise} \end{cases}$$

where P is the desired percentage of cluster heads, r is the current round of the protocol and G is the set of nodes that have not been selected as a cluster head in the previous rounds. Using this threshold, each node will be a cluster-head at some point after $1/P$ rounds. After an advertisement information, each node selects its cluster head. During the steady phase, each node sends their sensing values to its cluster head which aggregates the received data before sending them to the base station. After the steady phase, the network goes into the set-up phase again for a new round and a new cluster heads election.

4.2 Different Scenarios and Evaluation Parameters

We have implemented the LEACH protocol on the WSnet simulator. we have set $P = 0.2$ and we have tested our approach on a random nodes distribution. Each simulation is run with n sensor nodes and $n \in [100; 600]$ distributed randomly over a square field of 400m by 400m. Our simulations use the IEEE 802.11 physical and MAC layers which are fully simulated in the WSnet environment. We have also used the RC4 stream cipher but any other stream cipher can be used. In this study, we have simulated four different scenarios: the first scenario consists in no aggregation at all, nor for the data neither for the MACs. The second scenario simulates the data aggregation technique presented in [5] which we add the concatenation of all tags generated by sensor nodes. The two last cases simulate our own proposals described in Section 3.1 and in Section 3.2. Note that the operations performed in Scenario 3 are based upon XOR operations whereas the usual + is used only in Scenario 4.

4.3 Simulation and Results

We have simulated the 4 scenarios described above based on aggregator nodes elected using the LEACH protocol. We consider in all simulations that each node senses a value at each second and sends it with a given probability equal to 90 %.

We have tested the average delay time for a packet to travel until reaching the sink and the average energy consumption per nodes for the four scenarios. For each test, we have repeated the tests over 100 simulations for the four scenarios.

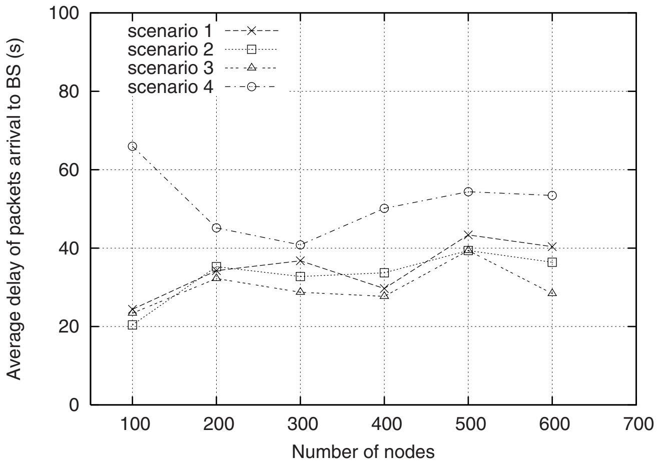

Fig. 2. Energy consumption for different aggregation schemes

Fig. 2 presents the average delay for a packet to reach the sink. Clearly, Scenario 4 is the slowest one and has the maximum delay for the average packet transfer as every node must wait l time intervals before sending the l aggregated values to first its cluster head that aggregates them and forwards to the sink. Moreover, the cluster head must wait values from other members before sending all the aggregated values to the base station. So, as expected, the Scenario 4 has the highest latency compared with the other scenarios. This proposal is really adapted to applications that require simple data gathering or environmental monitoring without any emergency needs. Among the three other scenarios, the Scenario 3 is the one that presents the best average delay compared with the two first scenarios. This comes from the fact that this scenario minimizes the size of the sent packets leading to a better transmission time for every cluster head.

Fig. 3 presents the gain on energy consumption between Scenario 4 and each of the other scenarios. As one could see on the figure, this gain is between 2.5 and 9 which is really significant. Moreover, Scenario 3 has the best energy consumption after Scenario 4. Those real improvements in terms of energy keeping is directly linked with the size of the sent packets because in the three first scenarios the number of sent packets is about the same whereas in the last case, the number of sent packets is divided by l here equal to 10.

More formally, considering the WSN as a tree of depth d and of width t, considering that one bit is sent by each nodes, the number of sent bits when no aggregation is performed is equal to $\sum_{i=0}^{d}(d-i)t^{d-i}$, whereas when we consider an aggregated scheme, the total number of sent bits is $\sum_{i=0}^{d} t^{d-i}$. In those cases, considering that each node send a message of m bits and a tag of ℓ bits, the number of bits sent if Scenario 1 is used is $(\ell+m)\sum_{i=0}^{d}(d-i)t^{d-i}$, for Scenario 2 it is $\ell\sum_{i=0}^{d}(d-i)t^{d-i}+m\sum_{i=0}^{d} t^{d-i}$, for Scenario 3 it is $\ell\sum_{i=0}^{d} t^{d-i}+m\sum_{i=0}^{d} t^{d-i}$ and for Scenario 4, considering that a message of length m and a MAC of length

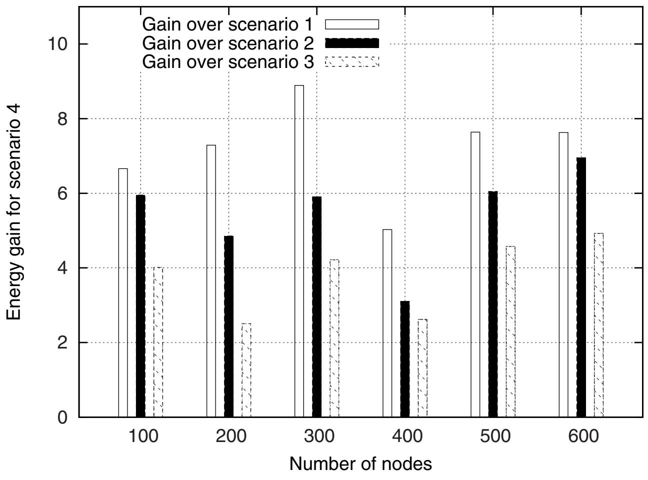

Fig. 3. Gains on energy consumption when comparing Scenario 4 with the three first scenarios

ℓ is sent one time during l periods, it is about

$$\frac{\ell \sum_{i=0}^{d} t^{d-i} + m \sum_{i=0}^{d} t^{d-i}}{l}.$$

Thus, performance evaluations confirm theoretical evaluations: in the theoretical approach, the gain would be better than as shown by evaluations but this fact is directly linked with the perfect structure of the network supposed in the theoretical model; the number of aggregator nodes is also greater in the theoretical approach.

5 Conclusion

In this paper, we have presented a simple method based upon universal hash function to aggregate MACs in a Wireless Sensors Network. To reach this aim, we extended the two schemes originally proposed by Krawczyk in [17] to simplify the data treatment. We have also discussed the security of our schemes in the model proposed in [7]. We have validated our approaches by intensive simulations that show the pertinence of our schemes and a significant gain in terms of energy when our last proposal is used.

Due to the small sizes of the sent messages in a WSN, it seems judicious to send several messages in a same time to be sure to generate a correct (and sufficiently long) MAC. If only a single message is sent, the required functions for this operation look more like expansion functions than compression functions. In our future work, we will particularly focus on this expansion aspect and on the implementation of universal hash functions in MSP430 sensors in order to evaluate their software performances on small devices.

References

1. Aad, I., Castelluccia, C., Hubaux, J.P.: Packet coding for strong anonymity in ad hoc networks. In: IEEE Securecomm (August 2006)
2. Bernstein, D.J.: The poly1305-aes message-authentication code. In: Gilbert, H., Handschuh, H. (eds.) FSE 2005. LNCS, vol. 3557, pp. 32–49. Springer, Heidelberg (2005)
3. Bhaskar, R., Herranz, J., Laguillaumie, F.: Efficient authentication for reactive routing protocols. In: AINA (2), pp. 57–61. IEEE Computer Society, Los Alamitos (2006)
4. Carter, L., Wegman, M.N.: Universal Classes of Hash Functions. Journal of Computer and System Sciences - JCSS 18(2), 143–154 (1979)
5. Castelluccia, C.: Securing very dynamic groups and data aggregation in wireless sensor networks. In: IEEE MASS - The Fourth IEEE International Conference on Mobile Ad-hoc and Sensor Systems, Pisa, Italy, October 2007, pp. 1–9 (2007)
6. Castellucia, C., Mykletun, E., Tsudik, G.: Efficient aggregation of encrypted data in wireless sensor networks. In: Mobile and Ubiquitous Systems: Networking and Services - MobiQuitous 2005, pp. 1–9 (2005)
7. Chan, A.C.-F., Castelluccia, C.: On the (Im)possibility of aggregate message authentication codes. In: IEEE International Symposium on Information Theory - ISIT 2008, pp. 235–239. IEEE, Los Alamitos (2008)
8. Chan, A.C.-F., Castelluccia, C.: On the Privacy of Concealed Data Aggregation. In: Biskup, J., López, J. (eds.) ESORICS 2007. LNCS, vol. 4734, pp. 390–405. Springer, Heidelberg (2007)
9. Chan, H., Perrig, A., Song, D.: Secure hierarchical in-network aggregation in sensor networks. In: CCS 2006: Proceedings of the 13th ACM conference on Computer and communications security, pp. 278–287. ACM, New York (2006)
10. Domingo-Ferrer, J.: A Provably Secure Additive and Multiplicative Privacy Homomorphism. In: Chan, A.H., Gligor, V.D. (eds.) ISC 2002. LNCS, vol. 2433, pp. 471–483. Springer, Heidelberg (2002)
11. Ekdahl, P., Johansson, T.: A new version of the stream cipher SNOW. In: Nyberg, K., Heys, H.M. (eds.) SAC 2002. LNCS, vol. 2595, pp. 47–61. Springer, Heidelberg (2003)
12. Ben Hamida, E., Chelius, G., Gorce, J.-M.: Scalability versus accuracy in physical layer modeling for wireless network simulations. In: 22nd ACM/IEEE/SCS Workshop on Principles of Advanced and Distributed Simulation (PADS 2008), Rome, Italy (June 2008)
13. Handschuh, H., Preneel, B.: Key-Recovery Attacks on Universal Hash Function Based MAC Algorithms. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 144–161. Springer, Heidelberg (2008)
14. Heinzelman, W., Chandrakasan, A., Balakrishnan, H.: Energy-efficient communication protocol for wireless microsensor networks. In: Proceedings of the Hawaii Conference on System Sciences (January 2000)
15. Hu, L., Evans, D.: Secure aggregation for wireless networks. In: Workshop on Security and Assurance in Ad hoc Networks, pp. 384–394 (2003)
16. Katz, J., Lindell, A.Y.: Aggregate message authentication codes. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 155–169. Springer, Heidelberg (2008)
17. Krawczyk, H.: LFSR-based hashing and authentication. In: Desmedt, Y.G. (ed.) CRYPTO 1994. LNCS, vol. 839, pp. 129–139. Springer, Heidelberg (1994)

18. Nevelsteen, W., Preneel, B.: Software performance of universal hash functions. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 24–41. Springer, Heidelberg (1999)
19. Przydatek, B., Song, D., Perrig, A.: SIA: Secure information aggregation in sensor networks. In: ACM SenSys 2003 (November 2003)
20. Sarkar, P.: 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/`
21. Shoup, V.: On Fast and Provably Secure Message Authentication Based on Universal Hashing. In: Koblitz, N. (ed.) CRYPTO 1996. LNCS, vol. 1109, pp. 313–328. Springer, Heidelberg (1996)
22. Wagner, D.: Resilient aggregation in sensor networks. In: 2nd ACM workshop on Security of ad hoc and sensor networks - SASN 2004, pp. 78–87. ACM, New York (2004)
23. Yang, Y., Wang, X., Zhu, S., Cao, G.: Sdap: a secure hop-by-hop data aggregation protocol for sensor networks. In: MobiHoc 2006: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing, pp. 356–367. ACM, New York (2006)