Confidentiality and Integrity for Data Aggregation in WSN Using Peer Monitoring

Roberto Di Pietro*
UNESCO Chair in Data Privacy
Universitat Rovira i Virgili - DEIM
Av. Països Catalans 26,
E-43007 Tarragona, Catalonia-Spain
roberto.dipietro@urv.cat

Pietro Michiardi Refik Molva
Institut Eurécom
2229 route des crêtes, BP 193
F-06560 Sophia-Antipolis cedex, France
{molva, michiardi}@eurecom.fr

Abstract

Hop-by-hop data aggregation is a very important technique used to reduce the communication overhead and energy expenditure of sensor nodes during the process of data collection in a Wireless Sensor Network (WSN). However, the unattended nature of WSNs calls for data aggregation techniques to be secure. Indeed, sensor nodes can be compromised to mislead the base station by injecting bogus data into the network during both forwarding and aggregation of data. Moreover, data aggregation might increase the risk of confidentiality violations: If sensors close to the base station are corrupted, an adversary could easily access to the results of the "in network" computation performed by the WSN. Further, nodes can also fail due to random and non-malicious causes (e.g. battery exhaustion), hence availability should be considered as well.
In this paper we tackle the above issues that affect data aggregation techniques by proposing a mechanism that: i) provides both confidentiality and integrity of the aggregated data so that for any compromised sensor in the WSN the information acquired could only reveal the readings performed by a small, constant number of neighboring sensors of the compromised one; ii) detects bogus data injection attempts; iii) provides high resilience to sensor failures. Our protocol is based on the concept of delayed aggregation and peer monitoring and requires local interactions only. Hence, it is highly scalable and introduces small overhead; detailed analysis supports our findings.

keywords: Wireless Sensor Networks, Secure data aggregation, bogus data injection attack, node failure, peer monitoring, resilience.

1 Introduction

A Wireless Sensor Network (WSN) is a collection of sensors characterized by a size that can range from a few hundred sensors to a few hundred thousands or possibly more. A WSN can be deployed in harsh environments to fulfil the requirements of both military and civil applications [1]. Due to the lack of a network infrastructure, sensors build a wireless multi-hop network in an autonomous way. Moreover, since power supply of each individual sensor is provided by a bat-

*He is also with Dipartimento di Matematica, Università di Roma Tre, L.go S. Murialdo, 00146 - Roma, Italy dipietro@mat.uniroma3.it

tery, both communication and computation activities must be optimized. In particular, radio communications are more demanding than computational operations in terms of energy consumption [26, 29, 32]. To that effect, *data aggregation* has been put forward as an essential technique to achieve power efficiency by reducing data redundancy and minimising bandwidth usage. Data aggregation consists of processing data collected by sensor nodes at each intermediate node en route to the sink in order to minimising the number of messages transmitted in the WSN.

The data sensed by a WSN is routed to a base station (BS) that can be viewed as a device that does not suffer from the limitations of the sensors and that acts as a gateway towards the Internet. Sensor networks must be robust and durable in order to overcome individual sensor failure due to either malicious (e.g. destruction) or non-malicious (e.g. malfunctioning) events. Indeed, it is likely that WSNs will be deployed in situations where an adversary may be motivated to disrupt the function of the network. For instance, an adversary may compromise a node in the network and gain access to its keying material. Furthermore, it is possible for a sensor to fail to properly execute the protocol due to HW or SW failures. In this paper, we focus on security threats that mainly target the data aggregation mechanism in a WSN, namely: Confidentiality violations of the aggregate data sensed by the WSN and data poisoning attacks, aiming at tampering with aggregate value computed by the WSN. The main contribution of our work is the design of an aggregation protocol that is resilient to node exposures: Compromising a node should not allow the adversary to jeopardize the whole computation performed by the WSN.

We assume that sensors use symmetric key encryption to provide confidentiality [2, 5]. It is worth noticing how key management impacts data aggregation. On one hand, providing all sensors with a single shared key is not a robust solution, since a single node compromise would impact the security of the whole WSN. On the other hand, pairwise key distribution as suggested in [10], is a more robust solution. However, even in this case, if the adversary is able to compromise a single node that has aggregated a sufficient amount of data, information leakage on the aggregated data becomes possible. Meanwhile there is an inherent conflict between data confidentiality and data aggregation due to the fact that data aggregation requires intermediate nodes to access data in cleartext in order to process it whereas data confidentiality calls for end-to-end encryption of each data segment between the sensors and the sink. Homomorphic encryption techniques appear to be the mandatory solution to this conflict since they allow some operations to be performed on encrypted data. Hence, we resort to homomorphic encryption schemes that allow sensors to perform the aggregation over encrypted data. In this case, single node compromise does not cause any information leakage of the aggregated data [4].

Our proposal at glance: In a standard tree-based data aggregation solution, each sensor node would compute the aggregate sum of all its inputs one level down in the tree hierarchy and pass the computed value up in the hierarchy. In our proposal, nodes perform delayed aggregation; each node aggregates the encrypted values of sensed data provided by nodes down in the hierarchy and passes up in the hierarchy a triplet that consists of: The result of the aggregate computation, the number of contributors to the aggregate value, and the encryption of the node's own reading. In addition to the basic data aggregation, our proposal also involves a mechanism for integrity verification: Some peer nodes of the WSN monitor the data exchanged by neighboring nodes in order to perform the data aggregation operation and to compare the result they compute with the aggregate value forwarded by their neighbors. Delayed aggregation thus trades off a slight increase in communication overhead during the aggregation phase in return for the capability to perform result-checking.

Contributions: This paper presents a data ag-

gregation protocol based on additive homomorphic encryption to provide data confidentiality. Furthermore, we introduce a local peer monitoring mechanism to detect the injection of false data. In particular, the proposed protocol is:

- Secure, since in a successful attack on the aggregated data, an oblivious adversary has to compromise at least a number of sensors proportional to the number of readings the aggregated data is composed by;
- Efficient, since each sensor is required to perform only local communications to achieve the global task carried out by the WSN;
- Robust with respect to both false data injection and confidentiality;
- Scalable, since the protocol continues to provide the above properties as the number of sensors in the WSN increases. For instance, our protocol is still effective for a WSN composed of 65,516 sensors;
- Resilient: A node compromise would only compromise the node itself and its neighboring nodes; this could provide access to the readings of these nodes, but not to the aggregated values received by these nodes.

Organization: The paper is organized as follows: Section 2 reports on the related work in the field, while Section 3 states the rationale and the assumptions our solution relies on. In Section 4 we formalize our security requirements, while in Section 5 we describe our secure data aggregation protocol. In Section 6 we discuss the properties provided by our protocol. Finally, Section 7 presents some concluding remarks.

2 Related Work

Early work in data aggregation for WSN assumed every node to be honest [16, 27]. This restrictive assumption has been relaxed in [25], addressing the problem of a single compromised node. However, the proposed protocol may be vulnerable if just a pair of parent and child nodes at the logical tree used to convey data to the BS are compromised. Further, each aggregator sensor has to forward a number of messages that is proportional to the number of contributors. The work in [5] enables the BS to verify whether the aggregated value provided by the WSN is a good approximation of the values that are actually collected by the sensors even when a fraction of the sensor nodes are corrupted. However, this objective is achieved via interactive proof between the aggregators and the central server (BS). Hence, due to the computational and communication cost incurred by this scheme the aggregators are less likely to be common sensors. The work in [39] analyses the security of various aggregation functions that could be used in WSN. The author describes attacks against these existing aggregation schemes and provides a framework to evaluate the security of these schemes.

The Concealed Data Aggregation (CDA) concept was first introduced in [21, 22]. The CDA is a solution for end-to-end encryption that uses homomorphic encryption functions to enforce secure data aggregation in the context of WSNs. In [42] the CDA concept is further extended: The authors present a generalization of the approach for a class of routing protocols and also propose a key predistribution algorithm that limits the capability of an adversary to disrupt confidentiality of the aggregated data.

Another work [4], leveraging the key concepts introduced in [21, 22], introduces a new scheme that addresses both confidentiality issues and efficient data-aggregation algorithms. This scheme relies on a simple but provably secure homomorphic encryption function. However, some drawbacks affect this scheme: First, it is not robust against node compromise, that is a single node failure can disrupt the whole WSN computation; second, the data packet size is not optimal; finally,

there is the need to piggyback toward the BS either the IDs of all nodes that contributed to the computation or the IDs of the nodes that did not contribute.

Recently, in [6] the authors presented the first algorithm for provably secure hierarchical in-network data aggregation. The proposed algorithm incurs a node congestion message overhead of $O(\Delta \log_2 n)$, where Δ is the maximum degree of any node in the aggregation tree. The main algorithm is based on performing the SUM aggregation securely by first forcing the adversary to commit to its choice of intermediate aggregation results, and then having the sensor nodes independently verify that their contributions to the aggregate are correctly incorporated. However, while this proposal achieves integrity, the confidentiality of the aggregated value is not addressed. In this scheme, the closer to the BS the compromising nodes, the more information is leaked to an adversary. Adding confidentiality to the proposed scheme, while retaining integrity, is not addressed. A recent improvement to this algorithm is discussed in [17]. A further contribution to the field of data aggregation in WSN is in [3], where the authors present a novel outlier elimination technique designed for sensor networks. This technique is called RANBAR and it is based on the RANSAC (RANdom SAmple Consensus) paradigm —well-known in computer vision and in automated cartography—. The proposed RANBAR algorithm is capable to handle a high percent of outlier measurement. However, their proposal rely on a strong pre-assumption, namely that the sample is i.i.d. in the unattacked case. Further, this latter proposal, as the former one, does not address the confidentiality of the aggregated data. A recent solution inspired by these proposals, but taking into consideration security issues as well, can be found in [37].

Our proposal takes the structure of the aggregation tree as given and leverages results that allow to establish pairwise keys between nodes in the WSN. One method for constructing an aggregation tree is described in TAG [32] while, as for pairwise key establishment, following the seminal work in [15], several solutions nowadays exist, that provide a secure pairwise key establishment [2, 10, 11, 44].

Finally, note that a form of delayed aggregation — as expressed in the previous section— was used in [23] and [6] with totally different goals: In the former paper to implement a quantile summary algorithm; in the latter to balance the path length in the aggregation tree.

3 System model

3.1 Adversary model

We identify the *objectives* that an adversary has in thwarting a WSN as twofold:

- The adversary could be interested in eavesdropping the aggregated value computed by a relevant fraction of the nodes in the WSN. Note that this scenario can easily occur within the data aggregation framework, in which a hierarchy is imposed over nodes. Indeed, the impact of a node compromise increases as the distance of that node from the BS decreases;

- The adversary could provide the BS with bogus data, so that the decision process of the BS could be possibly disrupted or subverted towards incorrect decisions. This goal has been described in [5] as a *stealthy* attack, i.e., to cause the querier to accept a false aggregate that is higher or lower than the true aggregate value. It is out of the scope of the paper to consider denial-of-service (DoS) attacks where the goal of the adversary is to prevent the querier from getting any aggregation result at all. Furthermore, any maliciously induced extended loss of service is a detectable anomaly which will (eventually) signal the presence of the adversary; subse-

quent protocols or manual intervention are assigned the task to resolve the problem.

Finally, note that the injection of bogus or forged data also depletes the constrained resources of sensor nodes, shortening the network lifetime. Hence, data injection attacks must be detected as early as possible.

In this paper we consider an adversary able to compromise any sensor in the WSN. In particular, we refer to the *oblivious adversary*: At each step of the attack sequence, the next sensor to be tampered with is chosen randomly among the ones that have yet to be compromised. This adversary model has been adopted also in [4, 10, 15, 31] to cite a few.

3.2 Sensor failure model

Sensors can fail due to either malicious (e.g. destruction, capture and re-programming) or non-malicious (e.g. malfunctioning, battery exhaustion) events. For the latter case, we assume that a sensor u_f can fail with a probability f_p which is equal for every sensor and does not depend on the history of sensor failure or compromise. That is, $\mathbf{Pr}[u_f \ \ fails | u_{i_1}, \ldots, u_{i_j} \ \ already \ \ failed \vee u_{h_1}, \ldots, u_{h_m} \ \ already \ \ compromised] = \mathbf{Pr}[u_f \ fails] = f_p$. We refer to this model as *random failure model*.

We note that a sensor that fails is less harmful from the point of view of security for our scheme, as compared to a sensor that has been captured and reprogrammed.

We also make the conservative assumption that once a sensor fails, it can be considered as corrupted by the adversary. The consequence is that if we can prove our protocol to be resilient to the adversary model with probability (1-ϵ), then the resilience in the random failure model is at least (1-ϵ).

3.3 Aggregation function

In this paper we follow the idea first introduced in [25] and consider the value computed (in a distributed way) by the WSN as the result of a function that accepts as inputs the readings of a reasonable portion of the sensors the WSN is composed of.

In this work we focus on the *mean value* of the readings, as it has been done for example in [4] and [39]. This choice allows to compare our results with previous work on secure data aggregation, and in particular with [4]. However, we point out that the mechanisms described hereafter can be adapted to support the computation of several aggregated values besides the mean.

3.4 Network setup

We consider a large sensor network with densely deployed nodes. Due to dense deployment, nodes may have overlapping sensing ranges, hence events can be detected by multiple nodes providing a certain level of redundancy in sensed data. As in other data aggregation protocols, e.g. [25], we assume a topological tree rooted at the base station, as the one presented in [33]. There are various methods for constructing the aggregation tree based on different application requirements [20, 32, 43]. Our solution does not rely on a specific tree construction algorithm.

Note that data aggregation requires all sensors to send their data to the BS within the same sampling period. Hence, in the following we assume that sensors have *loosely synchronized clocks* [14, 18, 19, 38] and the aggregation scheme to operate in rounds. Time synchronization has been leveraged to design several solutions for security in WSN, like [5, 36, 44].

As a direct consequence of the high density and redundancy in data acquisition that characterizes the WSN network under investigation, in our model we assume the existence of a mechanism that assigns a certain portion of sensors to cover the role of *peer monitoring* nodes. Peer

monitoring nodes are used to verify that the aggregation process is carried out correctly, i.e. they *detect* the subversive activity aiming at compromising the integrity of the aggregated data. Peer monitors are selected during the setup of the topological tree to achieve data aggregation, wherein nodes usually take up the role of data aggregators (see Section 5). It should be noted that the mechanism used to designate roles should not statically assign the function carried out by a sensor node: A sensor could play different roles at different instants of time. However, for the sake of clarity, we assume roles to be static in the following discussion. It is outside the scope of this current work to describe a mechanism to select peer monitors: however, we point the reader to numerous related works that treat sensor selection as a minimum or dominant covering set problem [24, 9]. In particular, [41] discusses distributed algorithms to select backup sensors to complement aggregation topologies. Such mechanisms can be easily adapted to solve the problem of selecting a minimum set of peer monitors that would cover the aggregation tree.

When taking up the role of a peer monitor a sensor exploits the broadcast nature of radio communications and processes messages that are not originally destined to it, instead of discarding them. Note that a node assigned to this role does not bear any additional energetic costs, in terms of communication, as compared to any other node in the network. Indeed, all active nodes receive (and spend energy to do so) any message sent within their wireless transmission range. Peer monitors are asked to support only a small additional overhead (which is analyzed in Section 6.5) that is the trade-off to accept to enjoy data integrity.

3.5 Sketch of the protocol

In this section we provide an overview of the aggregation protocol (see Figure 1), which is described in more detail in section 5. A Sensor can be either an aggregator or a peer-monitor.

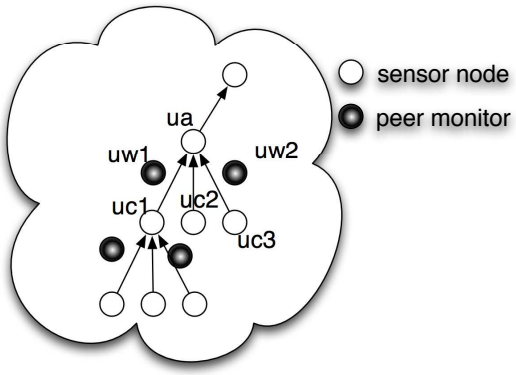


Figure 1. Sketch of the aggregation protocol: Aggregators, contributors and peer monitoring nodes on a partial tree.

During the i^{th} round, the aggregator u_a — assuming the aggregator has j contributors— combines the aggregated values $A_{c_q,i}$ $(q = 1 \ldots j)$ with each contribution $s_{c_q,i}$ to generate a new aggregated value $A_{a,i}$; Then, node u_a will invoke its sensing function, and will encrypt the sensed data (s_a, i). Node u_a will then send to the aggregator u_b up in the hierarchy the triple $< s_{a,i}, A_{a,i}, n_{a,i} >$, where $n_{a,i}$ is the number of sensors that contributed to A_a; Note that if the aggregator is a leaf in the tree hierarchy, it will send the triple $< s_{a,i}, 0, 0 >$. In the following, to ease the notation, we will not report the second index relative to the current round, unless required by the context.

Peer monitoring sensors $u_{w,i}$, which we describe in details in Section 5.1, receive the same data as the aggregator and act as independent verifiers checking if the aggregator correctly executed the aggregation function over the received values.

4 Security requirements

We have identified the following requirements for secure data aggregation protocols:

- Confidentiality: As discussed in Section 3.3, confidentiality of the aggregated data is a

fundamental property of a secure aggregation protocol;

- Resilience: As sensors have a short lifetime, data aggregation protocols need to be designed to be resilient to sensor failure;

- Scalability: As the size of a WSN can vary over time, secure aggregation protocols should scale with the number of sensors;

- Efficiency: As stated in Section 3.1 resources, e.g. energy and bandwidth, are scarce in a WSN. An efficient protocol for secure data aggregation should perform an early detection of false data injection[1].

4.1 Data confidentiality

In this subsection we introduce the cryptographic primitives used in the following. Our focus here is on a homomorphic encryption scheme allowing arithmetic operations to be performed on ciphertexts.

Formally, let $Enc()$ denote a probabilistic encryption scheme. Let M be the message space and C the ciphertext space such that M is a group under operation $\bigoplus$ and C is a group under operation $\bigotimes$. $Enc()$ is a $(\bigoplus, \bigotimes)$-homomorphic encryption scheme if for any instance $Enc()$ of the encryption scheme, given $c_1 = Enc_{k1}(m_1)$ and $c_2 = Enc_{k2}(m_2)$, there exists a key k such that $c_1 \bigotimes c_2 = Enc_k(m_1 \bigoplus m_2)$.

To provide data confidentiality we will adopt and modify the additively homomorphic encryption scheme presented in [4]. In this scheme $Enc(m_1, r_1, p) = m_1 + r_1 \pmod p$, where m_1 is the message to encrypt, r_1 is the key, and p is the modulus over which the sum is computed. The aforementioned encryption scheme supports additively homomorphic encryption operations since $Enc(m_1, r_1, p) + Enc(m_2, r_2, p) = (m_1 + r_1 \pmod p) + (m_2 + r_2 \pmod p) = m_1 + m_2 + r_1 + r_2 \pmod p = Enc(m_1 + m_2, r_1 + r_2, p)$.

In the present work, the key r_i is generated by a sensor via a stream cipher keyed with the secret key k_i and the unique identifier (id) of the message to encrypt. While in the work presented in [4] the secret key k_i associated to a sensor is shared only between the sensor and the BS, in our work the secret key k_i is also shared among neighboring sensors. In Section 2 we pointed to recent techniques to implement pairwise secret sharing in a WSN.

It should be noted that the homomorphic encryption function used in this work could be inspired by other solutions as well, like the one in presented in [12], which has also been adopted in [42].

4.2 Key setup

We now discuss on the cryptographic key setup that we assume in our system. We assume sensor nodes to share pairwise secret keys with their physical neighbors, and that each sensor node can establish a pairwise shared key with another node that is multiple hops away. The pairwise key establishment schemes proposed in [31] or in [10] where two nodes only need to know their IDs to establish a shared key can be used to achieve our requirements. We stress that both of these works have been shown to be affordable for sensor networks in terms of computational constraints and communication overheads.

As in [4], we also assume each sensor node u_i to be capable of generating a keystream r_i by using a stream cipher, such as RC4, keyed with the sensor's secret key k_i and a unique message identifier *id*. The secret key k_i is pre-computed and shared (in a secure way) between sensor u_i, the BS, and all neighbors of u_i. Node u_i can share the key k_i with its neighborhood using the pairwise key it shares with each of its neighbors.

Before proceeding any further, we wish to point out that the key setup discussed in this section ap-

[1] It is outside the scope of this paper to study the actions to be taken upon a detection event.

pears to be subject to the following attack. Distributing secret keys to a neighborhood implies that an adversary may compromise a single sensor and to be able to recover a number of keys (k_i, r_i) that is equal to the size of the neighborhood. In Section 6.2 we show that although we cannot prevent this kind of attack, our scheme bounds the impact of such an attack to the neighborhood of the compromised node, while still providing guarantees on the confidentiality of the aggregated data.

5 The aggregation protocol

As discussed in Section 3, we assume the WSN to be organized in a logical tree with peer monitoring and aggregator roles assigned. In this section, we focus on the normal operation of the aggregation protocol. A snapshot of the tree hierarchy, with examples of aggregators and peer monitors can be found in Figure 1. Furthermore, data aggregation is assumed to be performed so as to obtain the mean value of the data sensed by the WSN.

Note that all operations shown in the sequel of this section are performed modulus p, where p is a suitable prime number.

Aggregator: As sketched in section 3.5, the core idea of the protocol proposed in this work is that the aggregator sensor u_a does not compute the mean on its own reading, but only uses the readings of its contributory sensors (that is, the aggregators one level down in the tree-aggregation hierarchy). The encrypted reading of sensor u_a (s_a) will be aggregated by the sensor u_b towards which sensor u_a is contributor, as imposed by the logical aggregation hierarchy.

For each contributing sensor u_{c_q} ($q = 1 \ldots j$), aggregator u_a receives the triple $< s_{c_q}, A_{c_q}, n_{c_q} >$, where s_{c_q} is the encrypted reading of contributing sensor u_{c_q}, A_{c_q} is the received aggregated value computed by u_{c_q} and n_{c_q} is the number of contribution that generated the aggregated value A_{c_q}.

Sensor u_a then "unfolds" the aggregated data received by every contributing sensor and includes the readings received by every u_{c_q} to the new aggregated value:

$$A_a = \frac{\sum_{q=1}^{j}(A_{c_q} * n_{c_q}^{-1}) + \sum_{q=1}^{j} s_{c_q}}{n_A}$$

where:

$$n_a = n_{c_1} + \ldots + n_{c_j} + j.$$

During the computation of the aggregated data, two cases may occur: Either all sensors sent their contribution, or some sensors (u_{c_m}) did not send their contribution. In the latter case, u_a adds to the aggregated function the key (r_{c_q}) these sensors would have sent if the communication had taken place. Finally, u_a sends to the sensor up in the hierarchy its current reading encrypted with the random data r_a, the aggregated value A_a, and the number of contributors so far (n_a). Note that, unlike the algorithm proposed in [4], we introduce a division $(\mathrm{mod}\ p)$. The division is equivalent to compute the inverse of the number n_A and to perform a modular multiplication; however, note that the modulus is relatively small, hence the implementation of the extended Euclidian algorithm is quite efficient [28]. The pseudo code of the steps executed by sensor u_a is reported in Algorithm 1.

5.1 Peer monitoring sensors

As discussed in Section 3.4, the role of peer monitoring nodes (termed u_w in the sequel of this section) is to verify that an aggregator sensor u_a correctly performed the aggregation function. Figure 1 shows a snapshot of the aggregation hierarchy and two peer monitors, u_{w_1} and u_{w_2}.

Before delving into the details of the peer monitoring operation we first give a definition.

Definition 1 *Peer monitoring set. Given an aggregator sensor u_a we define the set of peer monitoring nodes associated to sensor u_a, termed*

Algorithm 1: The **aggregation** algorithm executed by u_a.

Aggregate($contributors\_list : ID\ list$)

Input : The list of the sensors that are contributors of u_a ($contributors\_list$).

Output : The encrypted locally sensed data (s_a), the locally computed aggregated value A_a, and the number of contributors n_a to the aggregated data.

$n_{c_q} : [1 \dots n]$, $contrib\_list = contributors\_list$;
while *True* **do**
  $contrib\_list = contributors\_list$;
  $tmp = 0;\ n_a = 0$;
  while *not* ($contrib\_list = \emptyset$) **do**
    `Receive`($u_{c_q} \in contrib\_list, <$ $s_{c_q}, A_{c_q}, n_{c_q} >, time\ out$) ;
    if $time\ out$ **then**
      for *each* $u_{c_q} \in contrib\_list$ **do**
        $r_{c_q} =$ `generate`(u_{c_q}) ;
        $tmp = tmp + r_{c_q}$;
        $contrib\_list = contrib\_list \backslash \{u_{c_q}\}$;
      end
    else
      $contrib\_list = contrib\_list \backslash \{u_{c_q}\}$;
      $inv =$ `inverse`(n_{c_q}, p) ;
      $tmp = tmp + A_{c_q} * inv + s_{c_q} \pmod p$;
      $n_a = n_a + n_{c_q} + 1$;
    end
  end
  $inv =$ `inverse`(n_a, p);
  $A_a = tmp * inv \pmod p$;
  $\backslash *$ sensing phase $\backslash *$;
  $s_a =$ `sense`() ;
  $r_a =$ `generate`(u_a) ;
  $s_a = s_a + r_a \pmod p$;
  `Send`($u_b, < s_a, A_a, n_a >$) ;
end

$\mathcal{W}(u_a)$, as the set of size d of nodes that are within the wireless radio range of u_a and that are not part of the aggregation tree.

We assume the set $\mathcal{W}(u_a)$ to be built during the construction of the aggregation topology and to be known by u_a and by all $u_w \in \mathcal{W}(u_a)$. We further assume that the set of sensors that are supposed to contribute with their data to node u_a (as dictated by the aggregation topology) is also known by all $u_w \in \mathcal{W}(u_a)$.

By exploiting the broadcast nature of radio communications, a peer monitoring sensor $u_w \in \mathcal{W}(u_a)$ might be able to receive the same data transmitted to the node u_a. Hence, u_w could perform the same checks of u_a (e.g. checking bounds) and could further check if the aggregated data computed by u_a matches with its own computed aggregated value. However, due to the relative distance between nodes, a monitoring node might not be in the transmission range of sensor u_c thus being unable to receive the data sent to u_a by u_c[2]. In the following, we distinguish between the case in which a peer monitor has all the contributor nodes in its communication range (direct observation) and the case in which this is not possible.

For the case wherein direct observation is possible, let us consider the aggregating sensor u_a. Suppose that sensor u_c sends its reading (s_c) together with the aggregated data A_c and the number of sensors that contributed to A_c ($< s_c, A_c, n_c >$) to sensor u_a. A peer monitoring node u_w overhears all messages sent by sensor u_c and performs exactly the same computations that take place at u_a. When u_a sends its triple $< s_a, A_a, n_a >$ upstream, u_w is able to verify whether the aggregated value A_a sent by u_a matches its locally computed value $A_{\hat{a}}$. Moreover, the peer monitoring node will also perform the following checks: On the data reading, assessing whether this data follows in the expected

[2]For example, sensor u_{w_2} in Figure 1 cannot overhear data sent by u_{c_1} to u_a.

bounds; on the number of contributors so far, that should not exceed n, or some other finer value, if available. Note that if it is not possible to perform the former two checks (for instance, the sensed data cannot be bound, or the value n is unknown to the peer monitoring), our proposal for the integrity of the aggregated data would still stood.

When direct observation is not possible, the normal execution of the aggregation protocol should be slightly modified. First, observe that if the contribution of a sensor u_c cannot be observed by a peer monitoring node u_w, then u_c can reach u_w within at most two hops. Indeed, by definition, nodes u_c and u_w are in the communication range of u_a. In this case, besides its normal operation, an aggregator node u_a is asked to act as a relay node, retransmitting all data required by the peer monitoring node u_w to verify the validity of aggregation. Data relaying may be subject to integrity attacks, hence each contributor u_c appends to the message sent to the aggregator sensor u_a an HMAC keyed with the pairwise key u_c shares with u_w. We recall now that our protocol, as well as all aggregation protocols available in the literature, works in rounds (synchronization is required). This implies that a peer monitoring sensor u_w is able to detect if an aggregator node u_a did not abide to the protocol, failing to forward all contributions to u_w. Note that a peer monitoring sensor u_w will raise an alarm if an expected contribution was not received. It is out of the scope of this paper to detail how to deal with such an anomaly, but a possible solution could be to remove the misbehaving sensor from the WSN when the number of its missing contributions exceeds a given threshold. The pseudo code of a peer monitoring sensor is reported in Algorithm 2.

One could argue that having a sensor u_a acting as a relay node for a fraction of contributor sensors that cannot be monitored may imply a significant overhead. However, as discussed in Section 6.5, an aggregator will be asked to forward a number of messages proportional to the number of its contributors, that is a constant, small number of nodes. This induce a communication overhead that is reasonably low as compared to the benefits offered by our scheme.

Moreover, in Section 6.2 we focus on those attacks performed by malicious peer monitoring nodes that do not fall in our adversary model but need to be discussed for the sake of completeness.

5.2 Features provided by the proposed protocol

In this section we highlight the features of the proposed aggregation protocol, while a detailed analysis is developed in Section 6.

- With the proposed aggregation protocol there is no need to send to the BS neither the list of the sensors that executed the aggregation function, nor the list of missing participants; this feature is not provided by [4];
- By using peer monitoring sensors, the proposed protocol allows local, early, and cooperative detection of bogus data injection;
- The aggregation protocol requires only the exact number of bits needed to compute the aggregate function (for instance, the average values over the readings);
- As sensors compute the aggregated data without taking into account their own readings, with the proposed protocol it is possible for the peer monitoring nodes to check not only the aggregated value, but also single sensor readings;
- In our scheme, peer monitoring sensors do not send data messages, except for alarm messages. This implies that the density of peer monitoring nodes does not increase medium access contention: If necessary, we could increase the resilience of the aggregation protocol by increasing the number of peer monitoring nodes (see Section 6.1) without any performance degradation;

Algorithm 2: The **peer monitoring algorithm** executed by u_w.

Peer Monitor($contributors\_list : ID\ list$) ;

Input : The list of the sensors that are contributors of u_a ($contributors\_list$);

Output : An alarm if the computation performed by the aggregator does not match the result of the local computation.

$min, max : data\_range, n : WSN\_size$;
while *True* **do**
  $tmp = 0;\ \hat{n}_a = 0;\ contrib\_list = contributors\_list$;
  while *not* ($contrib\_list = \emptyset$) **do**
    Receive($u_{c_q} \in contrib\_list, < s_{c_q}, A_{c_q}, n_{c_q} >, time\ out$) ;
    if $time\ out$ **then**
      for *each* $u_{c_q} \in contrib\_list$ **do**
        $r_{c_q} =$ generate(u_{c_q}) ;
        $tmp = tmp + r_{c_q}$;
        $contrib\_list = contrib\_list \backslash \{u_{c_q}\}$;
      end
    else
      $r_{c_q} =$ generate(u_{c_q}) ;
      $\backslash *$ bound checking $\backslash *$;
      if $(s_{c_q} - r_{c_q} \pmod p) \notin [min, max]$ **then**
        Recovery () ;
      else
        $contrib\_list = contrib\_list \backslash \{u_{c_q}\}$;
        $inv =$ inverse(n_{c_q}, p) ;
        $tmp = tmp + A_{c_q} * inv + s_{c_q} \pmod p$;
        $\hat{n}_a = \hat{n}_a + n_{c_q} + 1$;
      end
    end
  end
  $\backslash *$ Verification phase $\backslash *$;
  $inv =$ inverse($\hat{n}_a, p$);
  $\hat{A}_a = tmp * inv \pmod p$;
  Receive($u_a, < s_a, A_a, n_a >$);
  if $time\ out$ **then**
    Recovery () ;
  else
    if $(A_a = \hat{A}_a)\ and\ (n_a = \hat{n}_a)\ and\ (n_a \leq n)$ **then**
      skip
    end
    RaiseAlarm ()
  end
end

- A single node corruption would not reveal any other information than the readings of the corrupted node and the readings of its neighbors; that is, an adversary can only obtain a (small) constant number of readings without being able to recover any information on the aggregated data, as discussed in Section 6.2.

Table 1 provides a summary comparison of the protocol proposed in this paper with relevant related work in the literature. With HBH we refer to the case in which sensors encrypt data on a hop-by-hop basis, while No Agg. refers to a situation in which there is no data aggregation: Each reading is sent to the BS. CMT refers to the proposal appeared in [4]. In next section a detailed analysis of the proposed protocol is reported.

6 Analysis and discussion

6.1 Resilience

In order to analyse resilience of our protocol with respect to the adversary model introduced in Section 3.1, we assume that the adversary can compromise α randomly chosen sensors; we want to evaluate what is the probability ϵ for the adversary to inject bogus data.

Note that the sensors to be compromised are randomly selected among the ones that are not yet compromised. This means that every sensor can be compromised with probability less than $\alpha/(n-\alpha)$. As we have seen in Section 3.2, the above assumption can be used to provide an upper bound on the probability that compromised sensors can succeed in injecting a bogus data.

The relationship between the failure probability f_p and the number of sensors compromised by the adversary is given by $f_p < \alpha/(n-\alpha)$. In the sequel of this section, f_p can also be considered the fraction of the total number of sensors that can be corrupted by an adversary.

To provide an upper bound on resilience, note that we cannot have more than n peer moni-

Table 1. Comparison of the properties of the different Data Aggregation approaches.

Model	Properties		
	sensor failure resilience	confidentiality	message length independent from WSN size
Our	Yes	Yes	Yes
CMT [4]	No	Yes	No
HBH	Yes	No	Yes
No Agg.	Yes	Yes	No

toring sets in a WSN. The probability for the WSN to be jeopardized ($\mathbf{Pr}[J]$) is given by the probability that at least one of the monitoring sets is completely under the control of the adversary. Let us indicate with $\mathbf{Pr}[j_M] = max\{\mathbf{Pr}[j_1], \ldots, \mathbf{Pr}[j_n]\}$ the monitoring set that has the highest probability to be corrupted. Further, we recall that for a peer monitoring set of size $\hat{d}$, the probability to be entirely corrupted is given by $f_p^{\hat{d}}$, since the maximum of $\mathbf{Pr}[j_i]$ is obtained by the monitoring set that has the *least* number of peer monitoring sensors (d), hence $\mathbf{Pr}[j_M] = f_p^d$. Thus, an upper bound on the overall probability to jeopardy the WSN security is:

$$\mathbf{Pr}[J] \leq \mathbf{Pr}\left[j_1 \vee j_2 \vee \ldots \vee j_n\right] \leq n\mathbf{Pr}[j_M] = n \times f_p^d. \quad (1)$$

In Figures 2(a), 2(b) we plot Equation 1. On the x axis we report the number of sensors in the WSN, varying this value in the range $[256, \ldots, 65516]$, while on the y axis we show the probability that the WSN can be jeopardized ($\mathbf{Pr}[J]$). In Figure 2(a) and 2(b) we take into account two single node failure probabilities, that is $f_p = 2^{-4}$ and $f_p = 2^{-5}$ (*i.e.,* as noted in Section 3.2 this parameters reflect a situation in which the adversary can corrupt up to $n/17$ and $n/33$ sensors). We also vary the number of peer monitoring sensors d in the range $[4, \ldots, 6]$. Within each figure, we magnify the behavior of the curves for a choice of the x-axis in the range $[63000, \ldots, 65516]$, while the y-axis values are selected to grant visibility to the curve with the lowest values. Note that the lowest value on the y-axis is always provided by $d = 6$; this is coherent with the intuition that the bigger the number of peer monitors, the smaller the probability of compromising our scheme.

In Figure 2(a) the probability of jeopardizing the WSN is not negligible when $d = 4$, even for a small network size. However, note that as the number of peer monitoring sensors is at least $d = 5$, this probability is always below the value of 0.07, while for small networks (e.g. below 10,000 nodes) this probability is as little as 0.001. When the number of monitoring sensors reaches the value $d = 6$, the corresponding probability is negligible: It is less than 0.00006 for a WSN with up to 10,016 nodes, while increases only up to 0.004 for a WSN with 65,516 nodes.

The same qualitative behavior can be observed in Figure 2(b). In particular, when $d = 4$ and the network size is less than or equal to $10,016$ nodes, the probability that our protocol does not detect the injection of bogus data is always lower than 0.01, while this value slowly increases up to 0.06 when considering 65,516 nodes. For $d = 5$, with 10,016 nodes the probability to jeopardize the WSN is less than 0.0003, while it increases up to 0.002 when considering 65,516 nodes. Finally, for $d = 6$ we have that for each size of the WSN in the range $[256, \ldots, 65516]$, the probability ϵ is below 0.0006.

Figures 2(a) and 2(b) support the intuition that the proposed protocol is highly resilient against

the oblivious adversary. Indeed, for a wide set of parameters (for instance, we have assumed the adversary able to compromise $1/17$ of the network), $d = 6$ implies a very small probability for the oblivious adversary to be successful in injecting bogus data. Further, we highlight the fact that it is possible to trade-off the resilience to the oblivious adversary with a slight increase in the size of the peer monitoring set: A unity increase in the value of d provides an exponential gain in the resilience towards the oblivious adversary. It should be noted that the resilience of our scheme in the random failure model is equivalent to the one we obtained for the adversary model.

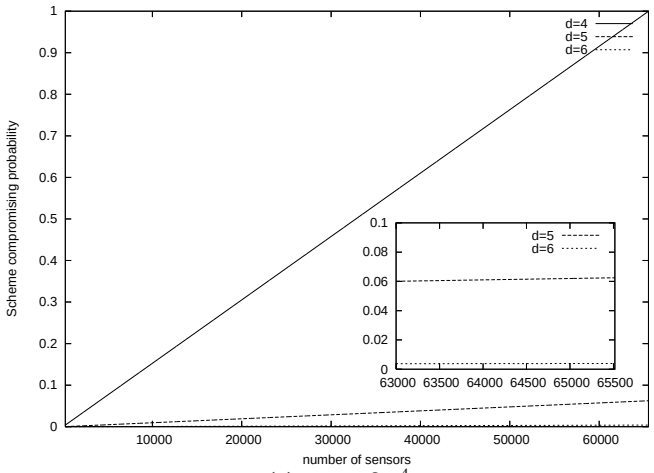
Finally, note that an exponential gain is achieved by decreasing the sensor failure probability (f_p). However, f_p is not a design parameter —for instance, if we use Commercial Off-The-Shelf sensors, they come with their Mean Time Between Failure in the technical specification sheet. This is why we have been conservative in considering high values for f_p, such as $f_p = 2^{-4}, 2^{-5}$.

6.2 Security analysis

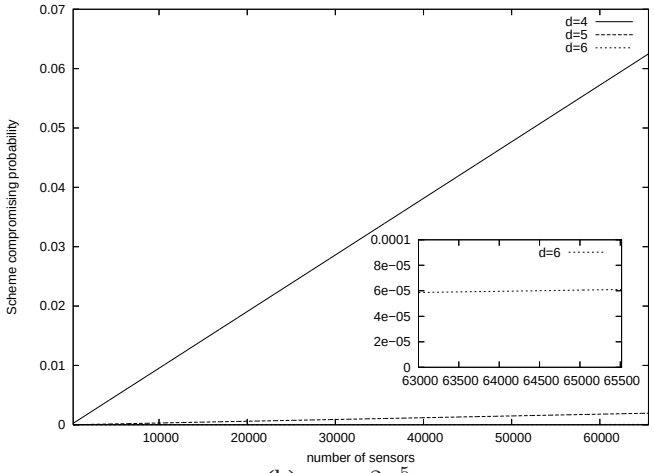
6.2.1 Confidentiality

Data aggregation can be exploited by an adversary to violate the confidentiality of the aggregated data, for example by compromising a few nodes close to the BS.

In our protocol, we cope with this threat via encryption using the scheme described in Section 4.1. Referring to the key setup phase described in Section 4.2, each sensor u_i distributes to its neighbors the secret key k_i used to generate the key stream r_i. Thanks to this feature, if sensor u_i fails, its neighbors can still provide the random value that should have been used to mask the sensed data. This operation is necessary since the BS, like in [4], derives all the masking values from the secret keys k_i it shares with sensors, and uses these values to decrypt the received aggregate data.



(a) $p_f = 2^{-4}$



(b) $p_f = 2^{-5}$

Figure 2. Scheme failure probability for $d = 4, 5, 6$

Hence, distributing secret keys to a neighbourhood means that compromising a single sensor will provide the adversary with a number of keys that is equal to the number of compromised sensor's neighbors. However, since in our scheme every and each node adds an encryption layer (r_a) on the sensed data, the aggregated data will be still secure, like in [4]. From these considerations the following lemma holds.

Lemma 1 *In our protocol, for the adversary to capture q readings, it is necessary to capture a number of nodes that is at least $\lceil q/M \rceil$, where M is the maximum size of an active neighbourhood.*

The above lemma allows the adversary to chose any q sensors from the WSN. That is, even if the adversary is able to compromise all the q sensors close to the BS, it will get no more than qM readings. Note that in a scheme where no data confidentiality is enforced, this would result in the compromising of the confidentiality of the aggregated data.

6.2.2 Impersonation

Here we turn to the attacks through which an adversary can impersonate a legitimate sensor node. By cloning a corrupted sensor an adversary could perform the so-called Sybil attack [13] and insert all produced sensor replicas in the WSN.

In the literature it is possible to find some solutions that deal with the Sybil attack, like [35]; more recent works aim at better identifying the rationales behind this attack and provide efficient solutions [7, 8] to cope with it.

Our aggregation protocol thwarts this kind of attack by construction. Indeed, if the adversary replicates a captured sensor and deploys it in different neighbourhoods, these replicas will not hold the encryption keys that have been previously shared among sensors; hence, the corrupted sensor and its replicas can not participate to the aggregation protocol. If replicas are positioned within the same neighbourhood of the corrupted node, a peer monitoring node can easily detect the attempts of a sensor to inject several readings in a single time slot.

In case a corrupted node tries to disrupt the computation performed in the WSN by injecting an implausible value, that is, a value that exceeds a certain range $[min, \ldots, max]$, an alarm is going to be raised; indeed, as described in Algorithm 2, the reading s_c provided by a contributor is defined to fall in the range $[min, \ldots, max]$.

6.3 Discussion on peer monitoring nodes

In this section we discuss on some possible caveats related to the role of peer monitoring nodes.

A typical example would be for a misbehaving peer monitoring sensor to report false alarms to the BS. The impact of false positives can be mitigated for example by requiring the source of an alarm to append its identity —according to a protocol that could allow identity verification. The BS could maintain and increment a tally for those nodes that falsely reported an alarm. Hence, the BS could take the appropriate action to isolate the misbehaving sensor if the tally exceeds a certain threshold. Note that this solution also requires a mechanism to identify false alarms. Further, peer monitors could also be the target of attacks. For example, a threat could be posed by a contributor sensor trying to control its transmission power with the goal to circumvent the peer monitor; this can be done by modulating the transmission power such that the signal would be strong enough to be overheard by a peer monitoring set, but too weak to be received by the true aggregator. Though this would require the misbehaving node to know exactly the transmission power required to reach each of its neighboring nodes, we consider this attack feasible. However, this attack would still be detected based on our protocol, since peer monitoring nodes verify the result of the computation performed by the aggregator, which will not match with the locally computed

value. Even if the identification of the misbehaving node is difficult, an alarm will be raised to inform the BS of the inconsistency of the aggregated data.

A more sophisticated attack can be mounted when multiple nodes collude. For example, suppose an aggregator u_a and all the nodes in $\mathcal{W}_a$ are corrupted. The aggregator could inject bogus data in the network. The aggregation protocol presented in this paper is inherently $d-1$ resilient to collusion attacks, where d is the size of the smallest peer monitor set. If at least one sensor in the set is not corrupted, it can still notify the BS of the anomalies perceived during each protocol execution.

6.4 Scalability issues

The number of neighbors a sensor can support is mainly limited by interference and collision issues at the medium access level. In this section, we investigate the scalability of our scheme with respect to the network size.

Define $x = max\{i | f_p \leq 2^{-i}\}$, then Equation 1 can be rewritten as:

$$\mathbf{Pr}[J] \leq 2^{-xd+\log_2 n}. \quad (2)$$

Equation 2 reflects the impact that the parameters d (the size of the peer monitoring set that has the minimal number of elements) and n (the number of sensors in the WSN) have on the resilience to failures. Our objective is to study the effectiveness of our protocol as the size of the WSN increases, while preserving $\mathbf{Pr}[J] \leq \epsilon$ resilience to bogus data injection.

Let M be the maximum size of an active neighbourhood a sensor can support and fix ϵ to be the upper bound on the probability of failure. In the following we study the relationship between the cardinality of the minimum peer monitoring set and the network size, for a given probability ϵ.

From Equation 2 we have that:

$$\mathbf{Pr}[J] \leq 2^{-dx+\log_2 n} \leq \epsilon$$

is true for

$$dx \geq \log_2 n - \log_2 \epsilon.$$

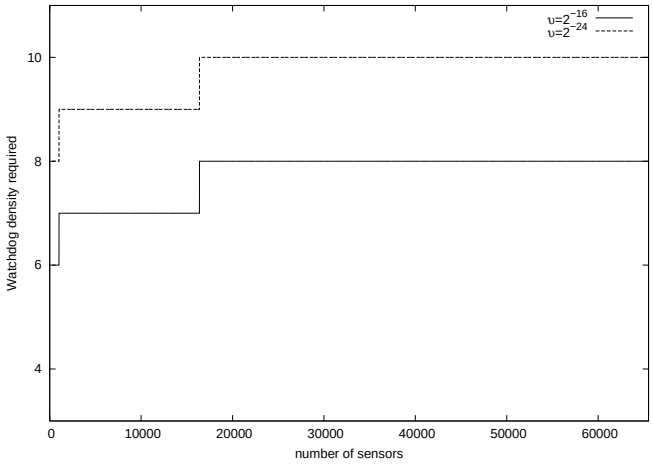
Hence, the values d can take on are given by:

$$M > d \geq \left\lceil \frac{\log_2 n - \log_2 \epsilon}{x} \right\rceil \quad (3)$$

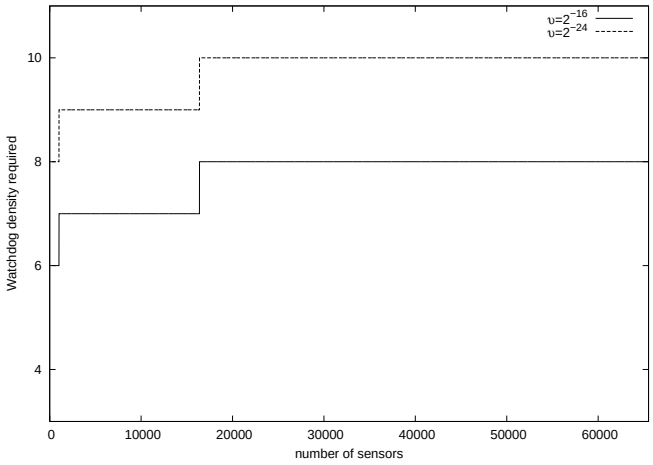
where M is the maximum size of an active neighbourhood, and the equation on the right provides the minimum value of d such that our WSN is still ϵ resilient. In Figures 3(a), 3(b) we plot Equation 3. On the x axis we represent the network size (in the range $[256, \ldots, 65516]$), while on the y axis we report the minimum number of monitoring sensors are required to have at least ϵ resilience. The two figures take into consideration different values for the failure probability f_p. In Figure 3(a) we assume to have $f_p = 2^{-4}$. From this figure, we observe that when the network size is less or equal to 16,376 sensors, a monitoring set of cardinality at least 7and 9 is required to have $\epsilon = 2^{-16}, 2^{-24}$ respectively. For a network size between 16,376 and 65,516 sensors, we need a peer monitoring set of cardinality 8 and 10 if we want to be ϵ resilient, where ϵ is $2^{-16}, 2^{-24}$ respectively.

The same results hold for the case $f_p = 2^{-5}$, as can be seen from Figure 3(b). Indeed, we have taken a conservative approach taking into consideration the upper integer part of $\frac{\log_2 n - \log_2 \epsilon}{x}$, as results from Equation 3.

From this discussion it follows that our proposed scheme is highly scalable. Indeed, for a WSN composed of up to 65,516 sensors, for a value of $f_p = 2^{-4}, 2^{-5}$, to assure an ϵ resilience for $\epsilon = 2^{-16}, 2^{-24}$, the number of required peer monitors ranges in $[7, \ldots, 10]$. Note that 10 neighbors is a very feasible limit on the number of neighbours a sensor can have [15]. Furthermore, this limit can even be stretched if we assume that peer monitoring sensors are specialized sensors (*i.e.*, they do not act neither as contributor nor as aggregator); in this case it holds



(a) $p_f = 2^{-4}$



(b) $p_f = 2^{-5}$

Figure 3. Peer monitoring density for failure probability less than $\epsilon = 2^{-16}, 2^{20}, 2^{-24}$

the consideration exposed in 5.2 about the possibility of decoupling the physical neighbourhood limit as for the peer monitoring set.

6.5 Overhead analysis

Transmission overhead Each contributing sensor is required to append an HMAC to each message destined to each peer monitoring sensor associated to it. Note that, as shown above, this number is in the range $[7, \ldots, 10]$, while the size of the HMAC can be limited to 64 bits, as detailed in the following. Further, note that an aggregator could incur an additional overhead when it acts as a rely node for nodes that are not in the direct communication range of the peer monitoring nodes. However, rely is just on-hop limited and possibly required by just a small subset of the nodes (if any) in the peer monitoring set. Finally, it should be noted that monitoring sensors do not send any messages, unless to rise an alarm.

Computational overhead Our protocol requires contributing sensors to generate the encryption key and the HMACs. Aggregating sensors are further required to generate the encryption key k_i in case of contributors failures. Peer monitoring sensors carry out the same set of computations performed by the aggregator. However, note that computing an HMAC can be considered a lightweight operation.

As opposed to the protocol proposed in [4], our scheme requires the additional overhead imposed by HMACs and the presence of peer monitoring sensors. These limitations are compensated by the additional properties of our scheme presented in Table 1. Further advantages of our proposal are: The smaller message size of the aggregated data; and, the fact that it does not require the identity of contributors to be appended.

Finally, recall that the implementation of the HMAC as discussed in detail in [34] would produce an output of 64 bits only. The energy devoted to compute and to transmit this HMAC has

been the subject of research [40, 30], and its cost has been accepted as sustainable by the vast majority of the literature concerned with security issues for WSN.

7 Concluding remarks

In this paper we present a mechanism for data aggregation in WSN that enforces both confidentiality and integrity of the aggregated data. The proposed mechanism is based on a novel application of peer monitoring and on a delayed aggregation of sensed data. The security of our scheme relies on the concept of additive homomorphic encryption and on a lightweight key distribution technique. Moreover, our scheme is robust against bogus data injection. Resilience to attacks and to random node failures is also provided. Our aggregation protocol is scalable, and we have shown that the network size can scale up to thousands of sensors with guarantees on the confidentiality and integrity provided by the scheme. These properties come at a limited additional cost since the proposed scheme requires only local communications and leverages lightweight cryptographic primitives.

Acknowledgment
This work was partly supported by the Spanish Ministry of Science and Education through projects TSI2007-65406-C03-01 “E-AEGIS” and CONSOLIDER CSD2007- 00004 “ARES”, and by the Government of Catalonia under grant 2005 SGR 00446.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Comput. Networks*, 38(4):393–422, 2002.

[2] Ross Anderson, Haowen Chan, and Adrian Perrig. Key infection: Smart trust for smart dust. In *Proc. of IEEE ICNP*, pages 206–215, Washington, DC, USA, 2004. IEEE Computer Society.

[3] Levente Buttyán, Péter Schaffer, and István Vajda. Ranbar: Ransac-based resilient aggregation in sensor networks. In *Proc. of ACM SASN*, pages 83–90, New York, NY, USA, 2006. ACM Press.

[4] C. Castelluccia, E. Mykletun, and G. Tsudik. Efficient aggregation of encrypted data in wireless sensor networks. In *Proc. of ACM/IEEE Mobiquitous*, pages 109–117, San Diego, CA, USA, 2005.

[5] Haowen Chan, Adrian Perrig, Bartosz Przydatek, and Dawn Song. Sia: Secure information aggregation in sensor networks. *Journal of Computer Security*, 15(1):69–102, January 2007.

[6] Haowen Chan, Adrian Perrig, and Dawn Song. Secure hierarchical in-network aggregation in sensor networks. In *Proc. of ACM CCS*, pages 278–287, New York, NY, USA, 2006. ACM Press.

[7] Mauro Conti, Roberto Di Pietro, Luigi Vincenzo Mancini, and Alessandro Mei. Requirements and open issues in distributed detection of node identity replicas in wsn. In *Proc. of IEEE SMC*, pages 1468–1473, 2006.

[8] Mauro Conti, Roberto Di Pietro, Luigi Vincenzo Mancini, and Alessandro Mei. A randomized, efficient, and distributed protocol for the detection of node replication attacks in wireless sensor networks. In *Proc. MobiHoc ’07*, pages 80–89, New York, NY, USA, 2007. ACM.

[9] Mathieu Couture, Michel Barbeau, Prosenjit Bose, and Evangelos Kranakis. Incremental

construction of k-dominating sets in wireless sensor networks. In *TR-Carleton University*, 2006.

[10] Roberto Di Pietro, Luigi V. Mancini, and Alessandro Mei. Energy efficient node-to-node authentication and communication confidentiality in wireless sensor networks. *Wireless Networks*, 12(6):709–721, 2006.

[11] Roberto Di Pietro, Luigi V. Mancini, Alessandro Mei, Alessandro Panconesi, and Jaikumar Radhakrishnan. Redoubtable sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(3):1–22, 2008.

[12] J. Domingo-Ferrer. A provably secure additive and multiplicative privacy homomorphism. In *Proc. IFIP SEC*, pages 471–483, 2002.

[13] John R. Douceur. The sybil attack. In *Proc. of USENIX IPTPS*, pages 251–260. Springer, 2002.

[14] Jeremy Elson and Kay Römer. Wireless sensor networks: a new regime for time synchronization. *ACM SIGCOMM CCR*, 33(1):149–154, 2003.

[15] Laurent Eschenauer and Virgil D. Gligor. A key-management scheme for distributed sensor networks. In *Proc. of the ACM CCS*, pages 41–47. ACM Press, 2002.

[16] Mike Esler, Jeffrey Hightower, Thomas E. Anderson, and Gaetano Borriello. Next century challenges: Data-centric networking for invisible computing. In *Proc. of ACM MOBICOM*, pages 256–262, 1999.

[17] Keith B. Frikken and IV Joseph A. Dougherty. An efficient integrity-preserving scheme for hierarchical sensor aggregation. In *Proc. ACM WiSec '08*, pages 68–76, New York, NY, USA, 2008. ACM.

[18] Saurabh Ganeriwal, Srdjan Capkun, Chih-Chieh Han, and Mani B. Srivastava. Secure time synchronization service for sensor networks. In *Proc. of ACM WISE*, pages 97–106, New York, NY, USA, 2005. ACM Press.

[19] Saurabh Ganeriwal, Deepak Ganesanl, Mark Hansen, Mani B. Srivastava, and Deborah Estrin. Rate-adaptive time synchronization for long-lived sensor networks. In *Proc. of ACM SIGMETRICS*, pages 374–375, New York, NY, USA, 2005. ACM Press.

[20] Johannes Gehrke and Samuel Madden. Query processing in sensor networks. *IEEE Pervasive Computing*, 3(1):46–55, 2004.

[21] Joao Girao, Markus Schneider, and Dirk Westhoff. Cda: Concealed data aggregation in wireless sensor networks. In *Proc. of ACM WISE*, Philadelphia, USA, October 2004. WiSe 2004. Poster presentation.

[22] Joao Girao, Dirk Westhoff, and Markus Schneider. Cda: Concealed data aggregation for reverse multicast traffic in wireless sensor networks. In *Proc. of IEEE ICC*, Seoul, Korea, May 2005. ICC2005.

[23] Michael B. Greenwald and Sanjeev Khanna. Power-conserving computation of order-statistics over sensor networks. In *Proc. of ACM SIGMOD-SIGACT-SIGART PODS*, pages 275–285, New York, NY, USA, 2004. ACM Press.

[24] Himanshu Gupta, Zongheng Zhou, Samir R. Das, and Quinyi Gu. Connected sensor cover: self-organization of sensor networks for efficient query execution. *IEEE/ACM Trans. Netw.*, 14(1):55–67, 2006.

[25] Lingxuan Hu and David Evans. Secure aggregation for wireless networks. In *Proc.*

of SAINT, page 384, Washington, DC, USA, 2003. IEEE Computer Society.

[26] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proc. of IEEE ICDCS*, page 457, Washington, DC, USA, 2002. IEEE Computer Society.

[27] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of network density on data aggregation in wireless sensor networks. In *Proc. of IEEE ICDCS*, page 457, Washington, DC, USA, 2002. IEEE Computer Society.

[28] Donald Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, 3^{rd} edition, 1997.

[29] Bhaskar Krishnamachari, Deborah Estrin, and Stephen B. Wicker. The impact of data aggregation in wireless sensor networks. In *Proc. of IEEE ICDCSW*, pages 575–578, Washington, DC, USA, 2002. IEEE Computer Society.

[30] Yee Wei Law, Jeroen Doumen, and Pieter Hartel. Survey and benchmark of block ciphers for wireless sensor networks. *ACM Trans. Sen. Netw.*, 2(1):65–93, 2006.

[31] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *Proc. of ACM CCS*, pages 52–61. ACM press, 2003.

[32] Samuel Madden, Michael J. Frankli, Joseph M. Hellerstein, and Wei Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proc. of ACM OSDI*, pages 131–146, New York, NY, USA, 2002. ACM Press.

[33] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. The design of an acquisitional query processor for sensor networks. In *Proc. ACM SIGMOD '03*, pages 491–502. ACM Press, 2003.

[34] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone. *Handbook of applied cryptography*, chapter Chapter 9 - Hash Functions and Data Integrity. CRC Press, 1996.

[35] Bryan Parno, Adrian Perrig, and Virgil D. Gligor. Distributed detection of node replication attacks in sensor networks. In *Proc. of IEEE S&P*, pages 49–63, Washington, DC, USA, 2005.

[36] Adrian Perrig, Robert Szewczyk, Victor Wen, David E. Culler, and J. D. Tygar. SPINS: security protocols for sensor networks. In *Proc. of ACM MOBICOM*, pages 189–199, 2001.

[37] Sankardas Roy, Mauro Conti, Sanjeev Setia, and Sushil Jajodia. Securely computing an approximate median in wireless sensor networks. In *Proc. SecureComm 2008, to appear*, 2008.

[38] Jana van Greunen and Jan Rabaey. Lightweight time synchronization for sensor networks. In *Proc. of ACM WSNA*, pages 11–19, New York, NY, USA, 2003. ACM Press.

[39] David Wagner. Resilient aggregation in sensor networks. In *Proc. of ACM SASN*, pages 78–87, New York, NY, USA, 2004. ACM Press.

[40] Arvinderpal S. Wander, Nils Gura, Hans Eberle, Vipul Gupta, and Sheueling Chang Shantz. Energy analysis of public-key cryptography for wireless sensor networks. In

Proc. of IEEE PERCOM, pages 324–328, Washington, DC, USA, 2005. IEEE Computer Society.

[41] Jiong Wang and Sirisha Medidi. Topology control for reliable sensor-to-sink data transport in sensor networks. In *Proc. IEEE ICC*, pages 3215–3219, 2008.

[42] Dirk Westhoff, Joao Girao, and Mithun Acharya. Concealed data aggregation for reverse multicast traffic in sensor networks: Encryption, key distribution, and routing adaptation. *IEEE Transactions on Mobile Computing*, 5(10):1417–1431, 2006.

[43] Wensheng Zhang and Guohong Cao. Optimizing tree reconfiguration to track mobile targets in sensor networks. *SIGMOBILE Mob. Comput. Commun. Rev.*, 7(3):39–40, 2003.

[44] Sencun Zhu, Sanjeev Setia, and Sushil Jajodia. LEAP: efficient security mechanisms for large-scale distributed sensor networks. In *Proc. ACM CCS*, pages 62–72, 2003.