Efficient Aggregation of encrypted data in Wireless Sensor Networks

Claude Castelluccia[*]
INRIA
Zirst - 655 avenue de l'Europe
38334 Saint Ismier Cedex, France
Claude.Castelluccia@inria.fr

Einar Mykletun, Gene Tsudik
Computer Science Department
School of Information and Computer Science
University of California, Irvine
{mykletun,gts}@ics.uci.edu

Abstract

Wireless sensor networks (WSNs) are ad-hoc networks composed of tiny devices with limited computation and energy capacities. For such devices, data transmission is a very energy-consuming operation. It thus becomes essential to the lifetime of a WSN to minimize the number of bits sent by each device. One well-known approach is to aggregate sensor data (e.g., by adding) along the path from sensors to the sink. Aggregation becomes especially challenging if end-to-end privacy between sensors and the sink is required. In this paper, we propose a simple and provably secure additively homomorphic stream cipher that allows efficient aggregation of encrypted data. The new cipher only uses modular additions (with very small moduli) and is therefore very well suited for CPU-constrained devices. We show that aggregation based on this cipher can be used to efficiently compute statistical values such as mean, variance and standard deviation of sensed data, while achieving significant bandwidth gain.

1 Introduction

Wireless sensor networks (WSNs) are becoming increasingly popular in many spheres of life. Application domains include monitoring of the environment (such as temperature, humidity and seismic activity) as well as numerous other ecological, law enforcement and military settings.

Regardless of the application, most WSNs have two notable properties in common: (1) the network's overall goal is typically to reach a collective conclusion regarding the outside environment, which requires detection and coordination at the sensor level, and (2) WSNs act under severe technological constraints: individual sensors have severely limited computation, communication and power (battery) resources and need to operate in settings with great spatial and temporal variability

At the same time, WSNs are often deployed in public or otherwise untrusted and even hostile environments, which prompts a number of security issues. These include the usual topics, e.g., key management, privacy, access control, authentication and DoS-resistance, among others. What exacerbates and distinguishes security issues in WSNs is the need to miniaturize all security services so as to minimize security-induced overhead. In other words, if security is a necessary hindrance in other (e.g., wired or MANET) types of networks, it is much more so in WSNs. For example, public key cryptography is typically ruled out as are relatively heavy-weight conventional encryption methods.

Security in WSNs is a popular research topic and many advances have been reported on in recent years. Most prior work has focused on ultra-efficient key management, authentication, routing and DoS resistance [1, 2, 3, 4]. An overview of security related issues and services required for WSNs is provided by Perrig, et al. in [5].

On the other hand, a lot of attention has been devoted to communication efficiency issues. Since data transmission is a very energy-consuming operation, in order to maximize sensor lifetime, it is essential to minimize the sheer number of bits sent by each sensor device. One natural and well-known approach involves aggregating sensor data as it propagates along the path from the sensors to the so-called sink – a node that collects sensed data. Of course, aggregating data is not quite equivalent to collecting individual sensor readings. In some applications, e.g., perimeter control, aggregation is useless since only individual sensor readings are of interest. However, many WSN scenarios that monitor an entire micro-environment (e.g., tem-

[*]This work was done while visiting at the University of California, Irvine

perature or seismic activity) do not require information from individual sensors but, instead, put more emphasis on statistical quantities, such as mean, median and variance.

Although simple and well-understood, aggregation becomes problematic if end-to-end privacy between sensors and the sink is required. If we assume that all sensors are trusted, sensors could encrypt data on a hop-by-hop basis. For an intermediate sensor (i.e., one that receives and forwards data), this would entail: 1) sharing a key each neighboring sensor, 2) for each downstream[1] neighbor, decrypting the received encrypted value, 3) aggregating all received values, and 4) encrypting the result for transmission upstream. Though viable, this approach is fairly expensive and complicated. The former because of having to decrypt each received value before aggregation and the latter – due to the overhead imposed by key management.

Furthermore, hop-by-hop encryption assumes that all sensors are trusted with the authenticity and privacy of other sensors' data. This assumption may be altogether unrealistic in some setting, whereas, in others, trust can be partial, i.e., intermediate nodes are trusted with only authenticity or only privacy.

Alternatively, if a single global key was used by all sensors, by subverting a single sensor node the adversary could learn measured values of any and all nodes in the network. Since only the sink should gain an overview of WSN measurements, this approach is not attractive.

Contributions: In this paper, we focus on efficient, bandwidth-conserving privacy in WSNs. More specifically, we blend inexpensive encryption techniques with simple aggregation methods to achieve very efficient aggregation of encrypted data. To assess the practicality of proposed techniques, we evaluate them and present very encouraging results which clearly demonstrate appreciable bandwidth conservation and small overhead stemming from both encryption and aggregation operations.

Organization: In the next section we discuss some background and the assumptions about our system model. Then, Section 3 describes the problem statement along with the security model. Next, Section 4 describes our homomorphic encryption scheme, followed by Section 5 which describes how to utilize this encryption scheme in a WSN. Performance is analyzed and results are discussed in Section 6. Related work is summarized in Section 7 and Section 8 concludes this paper.

[1]We use the terms downstream and upstream to mean away and towards the sink, respectively.

2 Background

In this section we describe the key features of, and assumptions about, the network and provide an overview of aggregation techniques.

2.1 Wireless Sensor Networks (WSNs)

A WSN is an ad-hoc network composed of a multitude of tiny devices with limited computation and energy capacities. One commonly cited WSN application is monitoring the environment. This may include sensing motion, measuring temperature, humidity, etc. Data monitored by the sensors is sent to a sink (usually a more powerful device), that is responsible for collecting the information.

The ad-hoc nature of a WSN implies that sensors are also used in the network infrastructure, i.e., not just sending their own data and receiving direct instructions but also forwarding data for other sensors. When sensors are deployed, a *delivery tree* is often built from the sink to all sensors. Packets sent by a sensor are forwarded to the sink by the sensors along the delivery tree.

Sensor nodes come in various shapes and forms, however, they are generally assumed to be resource-limited with respect to computation power, storage, memory and, especially, battery life. A popular example is the Berkeley mote [6]. One common sensor feature is the disproportionally high cost of transmitting information as compared to performing local computation. For example, a Berkeley mote spends approximately the same amount of energy to compute 800 instructions as it does in sending a single bit of data [6]. It thus becomes essential to reduce the number of bits forwarded by intermediate nodes, in order to extend the entire network's lifetime. The sink node acts as a bridge between the WSN and the outside world. It is typically a relatively powerful device, such as a laptop computer.

2.2 Aggregation in WSN

Aggregation techniques are used to reduce the amount of data communicated within a WSN and thus conserves battery power. Periodically, as measurements are recorded by individual sensors, they need to be collected and processed to produce data representative of the entire WSN, such as average and/or variance of the temperature or humidity within an area. One natural approach is for sensors to send their values to certain special nodes, i.e., aggregators. Each aggregator then condenses the data prior to sending it on. In terms of bandwidth and energy consumption, aggregation is beneficial as long as the aggregation process is not too CPU-intensive.

The aggregators can either be special (more powerful) nodes or regular sensors nodes. In this paper, we assume that all nodes are potential aggregators and that data gets aggregated as they propagate towards the sink. In this setting, since sensors have very limited capabilities, aggregation must be simple and not involve any expensive or complex computations. Ideally, it would require only a few simple arithmetic operations, such as additions or multiplications.[2].

We note that aggregation requires all sensors to send their data to the sink within the same sampling period. This either requires the sensors to have (at least loosely) synchronized clocks or the ability to respond to explicit queries issued by the sink.

One natural and common way to aggregate data is to simply add up values as they are forwarded towards the sink. Of course, this type of aggregation is useful when the sink is only interested in certain statistical measurements, e.g., the mean or variance of all measured data. As noted in Section 1, some WSN applications require all sensor data and therefore can not benefit from aggregation techniques. Similarly, applications requiring boundary values, e.g., min and/or max, are obviously not a good match for additive aggregation.

With additive aggregation, each sensor sums all values, x_i, it receives from its k children (in the sink-rooted spanning tree) and forwards the sum to its parent. Eventually, the sink obtains the sum of all values sent by all n sensors. By dividing the sum by n, i.e., the total numbers of sensors, it computes the average of all measured data.

This simple aggregation is very efficient since each aggregator only performs k arithmetic additions[3]. It is also robust since there is no requirement for all sensors to participate as long as the sink gets the total number of sensors that actually provided a measurement.

Additive aggregation can be also used to compute the variance, standard deviation and any other moments on the measured data. For example, in case of variance, each aggregator not only computes the sum, $S = \sum_{i=1}^{k} x_i$, of the individual values sent by its k children, but also the sum of their squares: $V = \sum_{i=1}^{k} x_i^2$. Eventually, the sink obtains two values: the sum of the actual samples which it can use to compute the mean and the sum of the squares which it can use to compute the variance:

$$Var = E(x^2) - E(x)^2; \text{ where}$$
$$E(x^2) = (\sum_{i=1}^{n} x_i^2)/n \quad \text{and} \quad E(x) = (\sum_{i=1}^{n} x_i)/n$$

[2]This is indeed what we achieve in this work.

[3]We assume that an aggregator has its own measurement to contribute; thus k additions are needed.

3 Goals and Security Model

In this work we are primarily concerned with data privacy. Our goal is to prevent a passive attacker (eavesdropper) from gaining any information about sensor data. An attacker is assumed to be global, i.e., able to monitor any location in the network or even the entire WSN. Furthermore, we assume the attacker is able to *read* the internal state of some sensors.

The above assumption might seem far-reaching since a global attacker could very well measure the data by itself. However, even an omni-present attacker may simply not have the means to install its own sensors, especially considering that sensors do not always measure relatively simple phenomena (such as ambient temperature); they can be used to monitor more difficult-to-measure factors, such as radiation level, water salinity, or air pollution. Moreover, a resource-limited attacker with knowledge of the WSN topology could simply position itself at or near the sink and thus obtain (by eavesdropping) all information about the measured data.

In light of our requirement for end-to-end privacy between the sensors and the sink, additive aggregation, although otherwise simple, becomes problematic. This is largely because popular block and stream ciphers, such as AES [7] or RC5 [8], are not additively homomorphic. In other words, the summation of encrypted values does not allow for the retrieval of the sum of the plaintext values.

To minimize trust assumptions we assume that each of the n sensors share a distinct long-term key with the sink. This key is originally derived[4] from the master secret, which is only known to the sink. We denote the sink's master secret as $\mathcal{K}$ and the long-term sensor/sink shared key as K_i, where the subscript $0 < i \leq n$ uniquely identifies a particular sensor. This way, the sink only needs to store a single master secret and all long-term keys can be recomputed as needed.

Even though we are advocating for end-to-end encryption, we assume hop-by-hop authentication. As opposed to encryption, authentication schemes that allow for aggregation seem to be very difficult, and perhaps impossible, to design. Furthermore, even if such an aggregate scheme existed, it is not clear how useful it would really be in practice, since in a WSN an attacker

[4]For example, using a pseudo-random function (PRF).

can easily affect the aggregate by just affecting the environment being sensed (i.e. for example by artificially increasing the temperature around a sensor). Note that such an attack does not require the attacker to compromise any node. As explained in [9], other techniques are needed to verify the plausibility of the resulting aggregate and to increase the aggregation resiliency. In WSNs, authentication does not provide data authenticity, but can instead be used to enforce access control, i.e. to prevent unauthorized nodes from injecting fake packets in the networks. This access control can efficiently be performed with hop-by-hop authentication and does not require end-to-end authentication.

4 Additively Homomorphic Encryption

In this section we describe the notion of *homomorphic encryption* and provide an example. We then proceed to present our additively homomorphic encryption scheme along with its security analysis. This encryption technique is very well-suited for privacy-preserving additive aggregation.

4.1 Homomorphic Encryption

A homomorphic encryption scheme allows arithmetic operations to be performed on ciphertexts. One example is a multiplicatively homomorphic scheme, whereby the multiplication of two ciphertexts followed by a decryption operation yields the same result as, say, the multiplication of the two corresponding plaintext values. Homomorphic encryption schemes are especially useful in scenarios where someone who does not have decryption keys needs to perform arithmetic operations on a set of ciphertexts. A more formal description of homomorphic encryptions schemes is as follows.

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 $\oplus$ and C is a group under operation $\otimes$. $Enc()$ is a $(\oplus, \otimes)$-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

$$c_1 \otimes c_2 = Enc_k(m_1 \oplus m_2)$$

In other words, the result of the application of function $\oplus$ on plaintext values may be obtained by decrypting the result of $\otimes$ applied to the corresponding encrypted values.

A good example is the RSA cryptosystem[10] which is *multiplicatively homomorphic*. The RSA encryption function is $Enc(m) = m^e = c \pmod n$ and the corresponding decryption function is $Dec(c) = c^d = m \pmod n$ where n is a product of two suitably large primes (p and q), e and d are encryption and decryption exponents, respectively, such that $e * d = 1 \pmod{(p-1)(q-1)}$.

Given two RSA ciphertexts c_1 and c_2, corresponding to respective plaintexts m_1 and m_2, it is easy to see that $c_1c_2 \equiv m_1^e m_2^e \equiv (m_1m_2)^e \pmod n$. Hence, one can easily compute the multiplication of the ciphertexts (c_1c_2) to obtain the ciphertext corresponding to the plaintext $m = m_1m_2 \pmod n$.

4.2 Proposed Encryption Scheme

We now introduce a simple *additively homomorphic* encryption technique. Its security analysis is provided in Appendix A. The main idea of our scheme is to replace the *xor* (Exclusive-OR) operation typically found in stream ciphers with modular addition (+).

Additively Homomorphic Encryption Scheme

Encryption:

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

Decryption:

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

Addition of Ciphertexts:

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

We assume that $0 \le m < M$. Due to the commutative property of addition, the above scheme is additively homomorphic. In fact, if $c_1 = Enc(m_1, k_1, M)$ and $c_2 = Enc(m_2, k_2, M)$ then $c_1 + c_2 = Enc(m_1 + m_2, k_1 + k_2, M)$.

Note that if n different ciphers c_i are added, then M must be larger than $\sum_{i=1}^{n} m_i$, otherwise *correctness* is not provided. In fact if $\sum_{i=1}^{n} m_i$ is larger than M, decryption will results in a value m' that is smaller than M. In practice, if $p = max(m_i)$ then M should be selected as $M = 2^{\lceil log_2(p*n) \rceil}$.

The keystream k can be generated by using a stream cipher, such as RC4, keyed with a node's secret key s_i and a unique message id. This secret key pre-computed and shared between the node and the sink, while the message id can either be included in the query from the sink or derived from the time period in which the node is sending its values in (assuming some form of synchronization).

5 Aggregation of Encrypted Data

As previously noted, efficient aggregation in WSNs becomes very challenging when end-to-end privacy of data is required. One solution is to disregard aggregation altogether in favor of privacy, i.e., for sensor nodes to forward their own encrypted measurements, as well as measurements received from their children, upstream. The sink, upon receiving as many data packets as there are responding sensors, proceeds to decrypt all ciphertexts and sums them up in order to compute the desired statistical measurements. We term this approach as *No-Agg*. This approach has two obvious disadvantages. First, because all packets are forwarded towards the sink, a lot of bandwidth (and hence power) is consumed. Second, as illustrated later in Section 6.2, there is an extreme imbalance between sensors in terms of the amount of data communicated. Sensors closer to the sink send and receive up to several orders of magnitude more bits than those on the periphery of the spanning tree.

A second approach, that does not achieve end-to-end privacy but does aggregate data, is a *hop-by-hop (HBH)* encryption method, which is also used for comparison between aggregation methods in [11]. In HBH all nodes create pair-wise keys with their parents and children during a boot strapping phase. When answering a query, nodes decrypt any packets sent to them, aggregate this data together with their own before re-encrypting the aggregated result and forwarding this to their parent. This approach is obviously more bandwidth efficient than No-Agg, as no packet is sent twice. However, there is an associated cost involved with the decryption and encryption performed at every non-leaf node in the WSN which increases their energy consumption (see [11]). More importantly, from a privacy perspective, the HBH scheme leaves nodes vulnerable to attacks because their aggregated data will appear in plaintext (i.e., no end-to-end privacy). Especially nodes closer to the sink become attractive targets for an attacker, as their aggregated values represent a large portion of the data in the WSN.

We instead propose an end-to-end privacy preserving aggregation approach (denoted as AGG) in which each sensor encrypts their sensed data using the encryption scheme presented in Section 4.2. Since this scheme is additively homomorphic, values can be added (aggregated) as they are forwarded towards the sink. The sink can then retrieve from the aggregate it receives the sum of the samples and derive certain statistical data. AGG retains the positive qualities of both the No-Agg (end-to-end privacy) and HBH (energy efficient) solutions.

5.1 Computing Statistical Data

In this section, we show how the new additively homomorphic encryption scheme can be used to aggregate encrypted data such that the sink can still compute the average and variance.

5.1.1 Computing the Average

When using our scheme, each sensor encrypts its data x_i to obtain $c_{x_i} = Enc(x_i, k_i, M)$. M needs to be chosen large enough to prevent an overflow so it is set as $M = n * t$. Each ciphertext c_{x_i} is therefore $log(M) = log(t) + log(n)$ bits long.

The sensor then forwards c_{x_i} to its parent, who aggregates all the c_{x_j}'s of its k children by simply adding them up (this addition is performed modulo M). The resulting value is then forwarded. The sink ends up with value $C_x = \sum_{i=1}^{n} c_{x_i} \pmod{M}$. It can then compute $S_x = Dec(C_x, K, M) = C_x - K \pmod{M}$, where $K = \sum_{i=1}^{n} k_i$, and derive the average as follows: Avg $= S_x/n$.

5.1.2 Computing the Variance

As mentioned previously, our scheme can also be used to derive the variance of the measured data.

In this case, each sensor i must compute $y_i = x_i^2$, where x_i is the measured sample, and encrypts y_i to obtain $c_{y_i} = Enc(y_i, k'_i, M')$. It must also encrypt x_i as explained in the previous section. M' needs to be chosen large enough to prevent an overflow so it is set to $M' = n * t^2$. Each ciphertext c_{y_i} is therefore $log(M') = 2 * log(t) + log(n)$ bits long. The sensor forwards c_{y_i}, together with c_{x_i}, to its parent. The size of the resulting data is $3 * log(t) + 2 * log(n)$. The parent aggregates all the c_{y_j} of its k children by simply adding them up. It also aggregates, separately, the c_{x_j}, as explained in the previous section. The two resulting values are then forwarded. The sink ends up with values C_x and $C_y = \sum_{i=1}^{n} c_{y_i} \pmod{M}$. C_x is used to compute the average Av. C_y is used to compute the variance as follows: The sink computes $V_x = Dec(C_y, K', M) = C_y - K' \pmod{M}$, where $K' = \sum_{i=1}^{n} k'_i$. The variance is then equal to $V_x/n - Av^2$.

5.2 Robustness

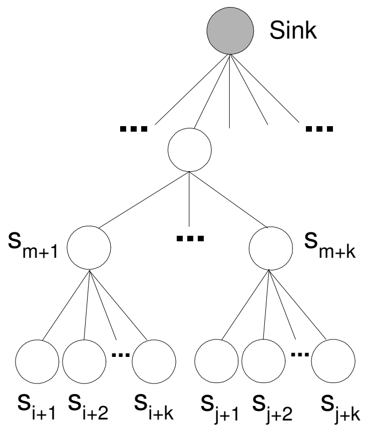

Figure 1. Multi-level WSN model with nodes of degree k

An important consequence of using our proposed encryption scheme for aggregation in WSNs is that the sink node needs to be aware of the encryptors id's such that it can regenerate the correct keystream for decryption purposes.

Because WSNs are not always reliable, it cannot be expected that all nodes reply to all requests. Therefore there needs to be a mechanism for communicating the id's of the non-responding nodes to the base station. The simplest approach, and the one we used in our evaluation, is for the sensors to append their respective node id's to their messages[5].

6 Analysis

In this section, we compare the bandwidth of our proposed *AGG* protocol with the *No-Agg* (forwarding data packets) and *HBH* (hop-by-hop encryption and aggregation) approaches, as described in section 5. The overall bandwidth in the WSN and the number of bits sent by individual nodes are measured for different WSN tree like topologies. Below we describe the specific network model that we use in our measurements. The comparisons will be made for the two following cases: (1) the sink is only interested in the average value and (2) the sink is interested in the average and variance values.

[5]Depending on the number of nodes that respond to a query, it could be more efficient to communicate the id's of nodes that successfully reported values

6.1 Network Model

We envision a multi-level network tree in which there exist numerous sensor nodes and only one sink node. To simplify the model, we assume a balanced k-ary tree, as depicted in figure 1. Let t denote the range of possible measurement values collected by a sensor (i.e., if a sensor can measure temperatures between 0 and 99 Fahrenheit, then t = 100).

We will analyze the communication bandwidth in the proposed WSN model from two perspectives: (1) the number of bits sent per node at different levels in a 3-ary tree and (2) the total number of bits transmitted throughout the WSN for 3-ary trees of various height. These measurements will be carried out for the three models that we are considering, namely *No-Agg*, *HBH* and *AGG*.

Next we describe how to calculate the number of bits sent per node for each of these schemes. We choose the packet format used in TinyOS [12] which is the operating system running on the Berkeley motes that we envision as the sensor platform. The packet header is 56 bits and the maximum supported data payload is 232 bits.

For the *No-Agg* scheme, a node only needs $log(t)$ bits to encode its sensed data. In addition, all internal nodes need to forward the packets sent to them by their children, and the number of packets received grows exponentially (in k) as we move higher in the tree (i.e. closer to the sink).

In the *HBH* approach, the number of bits sent depends upon the node's level in the WSN tree. Leaf nodes only send $log(t)$ bits (same as in No-Agg), while nodes higher up in the tree will have aggregated data and therefore need to send more bits. Additionally, when the variance is also requested, the aggregating nodes need to keep track of this value separately, and use approximately $log(n't)$ bits to encode the value, where n' is the number of nodes aggregated so far.

With our *AGG* scheme, the number of bits sent by a node depends on the size of the modulus M used in the additive encryption scheme. Its size can be computed as the maximum possible aggregate value, which in this model turns out to be $M = n*t$, i.e. all sensors measure the largest possible reading. Therefore, when encoding the average, each node uses $log(M) = log(t) + log(n)$ bits. When the variance is also desired, a node needs to send the ciphertext corresponding to x^2. This requires an extra $log(n*t^2) = 2*log(t)+log(n)$ bits. Additionally, an aggregator needs to append to the aggregate the id's of its children that did not reply to the query. These id's have to be propagated up to the sink along with the aggregate.

Table 1. Number of bits sent per node for each level in a 3-tree of depth 7, where the measured value range of 2^7

Levels	Num Nodes	A (0%)	A (10%)	A (30%)	AV (0%)	AV (10%)	AV (30%)	HBH-A	HBH-AV	No-Agg
1	3	75	950	2700	100	975	2725	73	97	68859
2	9	75	366	950	100	392	975	72	94	22932
3	27	75	172	366	100	197	392	70	91	7623
4	81	75	107	172	100	132	197	68	87	2520
5	243	75	85	108	100	111	132	67	84	819
6	729	75	78	85	100	103	110	65	81	252
7	2187	75	75	75	100	100	100	63	63	63

6.2 Numerical Results

In this section, we compare the performance of the *No-Agg*, *HBH* and AGG according to the following two criteria: (1) The forwarding cost per node i.e. the number of bits forwarded by node at each level of the delivery tree. (2) The overall bandwidth gain achieved achieved by *HBH* and AGG over the *No-Agg* scheme.

Forwarding Cost per node (fairness)

Table 1 shows the number of bits sent per node at each level in a 3-degree tree of height[6] 7 when $t = 128$ (the network is for example monitoring temperatures that range between 0 and 128 degrees).

For the *No-Agg* approach it becomes obvious from the results that there is a widely differing data communication load amongst sensors at different levels (nodes at level 7 send 3 orders of magnitude less data than those at level 1). Because the nodes closer to the sink have to send such significantly larger amounts of data than their descendants, they use up their batteries and die sooner. Should a level of nodes in the tree stop functioning, then the whole WSN stops functioning as well. Therefore, nodes would have to either be swapped around manually or replaced upon failure, both tasks being quite impractical when considering the number of nodes at the various levels.

The table shows a steady increase of bits per node for the HBH approach, both for the average (HBH-A) as well as the average and variance data (HBH-AV). Notice the relatively dramatic increase in bits transmitted between nodes at level 7 and 6 for HBH-AV. This is due to that the leaf nodes need not send a ciphertext representing x^2 (needed for the computation of the variance), where x represent their measured value, as it can be computed by their parents. Because packets are not forwarded as in No-Agg, we observe a significant reduction in bits sent per node at all non-leaf levels.

For the AGG we considered tree scenarios: (1) all the nodes reply[7], (2) 90% of the nodes reply[8] and (3) 70% of the nodes reply[9]).

In the first scenario, there is a constant number of bits sent by each node at each level in the tree. However, this number of bits is larger than even the maximum for any HBH approach, due to the size of the modulus M. As previously explained, the number of bits sent by the leaves is larger with the aggregation methods (AGG-A: $56 + log(t) + log(n) = 75$ bits, AGG-AV: $56 + 3 * log(t) + 2 * log(n) = 100$ bits) than when no aggregation is used ($56 + log(t) = 63$ bits). However, aggregation distributes the load evenly over all nodes, regardless of their distance to the sink. We believe this to be an attractive property in WSNs. In the second and the third scenarios, the number of bits processed by each node gets larger the closer it gets is to the sink. This is the result of appending the id's of the non-responding children to the aggregate. As we move up the tree the list of non-responding nodes increases.

Bandwidth Gain

Tables 2 displays the bandwidth transmission gain of the HBH and AGG schemes over the *No-Agg* scheme using a 3-degree WSNs of various heights. We consider the gains when (1) only the average is computed and (2) both the average and variance are computed [10]. These gains are obtained by computing the total bandwidth costs, C_{HBH}, C_{AGG} and C_{No-Agg}, by adding, for each of these schemes, the total number of bits forwarded by each node of the network. The bandwidth gain of HBH and AGG are respectively defined as C_{No-Sgg}/C_{HBH} and C_{No-Agg}/C_{AGG}.

[6] The sink is at level 0 in the tree

[7] Referred to in the tables as $A(0\%)$ when only the average is computed and as $AV(0\%)$ when the average and variance are computed.

[8] Referred in the table as $A(10\%)$ when only the average is computed and as $AV(10\%)$ when the average and variance are computed.

[9] Referred in the table as $A(30\%)$ when only the average is computed and as $AV(30\%)$ when the average and variance are computed.

[10] We remind the reader of that in the *No-Agg* scheme, no extra values need to be sent when the variance needs to be computed.

Table 2. WSN bandwidth performance gain of the AGG and HBH schemes when aggregating the (1) Average and (2) Average and Variance for a 3-tree and $t = 2^7 = 128$

Levels	Num Nodes	A (0%)	A (10%)	A (30%)	HBH-A	AV(0)	AV(10)	AV(30)	HBH-AV
3	40	2.42	2.39	2.34	2.58	1.89	1.87	1.84	2.24
4	121	3.20	3.13	3.01	3.50	2.46	2.40	2.37	3.02
5	364	3.96	3.82	3.6	4.46	3.03	2.98	2.84	3.84
7	3280	5.46	5.13	4.58	6.41	4.1	3.9	3.6	5.52
8	9841	6.22	5.72	4.95	7.39	4.59	4.3	3.85	6.37

For example, in a 3-tree of height 5, there are 364 nodes, and when only computing the average value, *AGG-A* achieves a factor of 3.96 speedup over No-Agg, i.e. approximately 4 times less bits are sent across the network. As expected, *HBH-A* and *HBH-AV* have better performance than both *AGG-A* and *AGG-AV*, respectively, although they both outperform No-Agg. The biggest draw for using *AGG* over HBH is that of end-to-end privacy. With HBH, it is enough for an attacker to compromise one node close to the sink to gain a large picture of the aggregated data in the WSN. This is because each node in HBH stores the secret key needed for decryption (and encryption), leaving them vulnerable. On the other hand, nodes in *AGG* do not store sensitive key material and the only data an attacker can learn is a single sensor's individual reading.

The results shown in this section are very encouraging since they confirm that aggregation is a useful technique for reducing the total bandwidth usage and can therefore extend the overall lifetime of the network.

7 Related Work

The problem of aggregating encrypted data in WSNs was partially explored in [11]. In this paper, the authors propose to use an additive and multiplicative homomorphic encryption scheme to allows aggregation of encrypted data. While this work is very interesting, it has several important limitations. Firstly, it is not clear how secure the encryption scheme really is. Secondly, as acknowledged by the authors, the encryption and aggregation operations are very expensive and therefore require quite powerful sensors. Finally, in the proposed scheme, the encryption expands the packet size significantly. Given all these drawbacks, it is questionable whether aggregation is still beneficial. In contrast, our encryption scheme is proven to be secure and is very efficient. Encryption and aggregation only requires a small number of single-precision additions. Furthermore, our encryption scheme only expands packet sizes by a small number of bits. As a result, it is well adapted to WSNs consisting of very resource constrained sensors.

In [13], Hu and Evans propose a protocol to securely aggregate data. The paper presents a way to aggregate MACs (message authentication code) of individuals packets such that the sink can eventually detects non-authorized inputs. This problem is actually complementary to the problem of aggregating encrypted data, we are considering in this paper. The proposed solution introduces significant bandwidth overhead per packet. Furthermore, it requires the sink to broadcast n keys, where n is the number of nodes in the network, at each sampling period. This makes the proposed scheme non-practical.

Although not related to data privacy, in [14] Przydatek, et al. present efficient mechanism for detecting forged aggregation values (min, max, median, average and count). In their setting, a trusted outside user can query the WSN. The authors then look into how to reduce the trust placed in the sink node (base station) while ensuring correctness of the query response. Another work by Wagner [9] examines security of aggregation in WSNs, describing attacks against existing aggregation schemes before providing a framework in which to evaluate such a scheme's security.

8 Conclusion

This paper proposes a new homomorphic encryption scheme that allows intermediate sensors (aggregators) to aggregate the encrypted data of their children without having to decrypt them. As a result, even if an aggregator gets compromised, the attacker won't be able to eavesdrop on the data and aggregate, resulting in much stronger privacy than an aggregation scheme relying on by hop-by-hop encryption.

We evaluate the performance of our scheme. We show, as expected, that our scheme is slightly less bandwidth efficient than the hop-by-hop aggregation scheme described previously. However it provides a much stronger level of security. The privacy protection provided by our scheme is in fact comparable to the privacy protection provided by a scheme that would use end-to-end encryption and no aggregation (i.e. the aggregation is performed at the base station). We show that our scheme is not only much more bandwidth-efficient than such an approach, but it also distributes the communication load more evenly amongst the network nodes, resulting in an extended longevity of the WSN.

One limitation of our proposal is that the identities of the non-responding nodes (or responding nodes, whichever is expected to be smaller) need to be sent along with the aggregate to the sink. If the network is unreliable, this can represent an important overhead. It is therefore important to devise methods for reducing this cost. We plan to pursue this topic in our future work.

References

[1] L. Eschenauer and V. D. Gligor, "A Key Management Scheme for Distributed Sensor Networks," *ACM CCS*, pp. 41–47, 2000.

[2] Sencun Zhu, Sanjeev Setia, Sushil Jajodia, and Peng Ning, "An Interleaved Hop-by-Hop Authentication Scheme for Filtering False Data in Sensor Networks," *Security and Privacy*, 2004.

[3] Chris Karlof and David Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Workshop on Sensor Network Protocols and Applications*, 2003.

[4] Anthony D. Wood, John A. Stankovic, "Denial of Service in Sensor Networks," *IEEE Computer*, vol. 35, pp. 54–62, 2002.

[5] Adrian Perrig and John Stankovic and David Wagner, "Security in wireless sensor networks," *Communications of the ACM*, vol. 47, pp. 53–57, 2004.

[6] Samuel R. Madden and Michael J. Franklin and Joseph M. Hellerstein and Wei Hong, "TAG: a Tiny AGgregation service for ad-hoc sensor networks," *Fith Annual Symposium on Operating Systems Design and Implementation*, pp. 131–146, 2002.

[7] National Institute of Standards and Technology, "Advanced encryption standard," *NIST FIPS PUB 197*, 2001.

[8] Ron L. Rivest, "The RC5 Encryption Algorithm," *Dr. Dobb's Journal*, vol. 1008, 1995.

[9] David Wagner, "Resilient Aggregation in Sensor Networks," *Workshop on Security of Ad Hoc and Sensor Networks*, 2004.

[10] Ron L. Rivest, Adi Shamir, and Leonard M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, pp. 120–126, 1978.

[11] Joao Girao and Dirk Westhoff and Markus Schneider, "CDA: Concealed Data Aggregation in Wireless Sensor Networks," *ACM WiSe*, 2004.

[12] Chris Karlof and Naveen Sastry and David Wagner, "TinySec: a link layer security architecture for wireless sensor networks," *Embedded Networked Sensor Systems*, pp. 162–175, 2004.

[13] Lingxuan Hu and David Evans, "Secure aggregation for wireless networks," *Workshop on Security and Assurance in Ad hoc Networks*, 2003.

[14] Bartosz Przydatek and Dawn Song and Adrian Perrig, "SIA: Secure Information Aggregation in Sensor Networks," *ACM SENSYS*, pp. 255–265, 2003.

A Security Analysis of Encryption Scheme

Our additive homomorphic encryption scheme is very similar to a xor-based stream cipher and its security can be proven using a similar proof.

The security relies in two important features: (1) the keystream changes from one message to another and (2) all the operations are performed modulo a integer M. These two features protect our scheme from frequency analysis attacks. In fact, it can be proven that our scheme is *perfectly* secure.

Theorem 1 *The previous encryption scheme is perfectly secure.*

Proof:
For plaintext space M, keystream space K,
let $\mathcal{K} = |M|$, $m \in [0, M-1]$, $c \in [0, M-1]$.
Set $k^* = c - m \pmod M$. Then:

$$\begin{aligned}
\underset{k\leftarrow\mathcal{K}}{Prob}[Enc(k,m,M)=c] &= \underset{k\leftarrow\mathcal{K}}{Prob}[k+m=c \pmod M] \\
&= \underset{k\leftarrow\mathcal{K}}{Prob}[k=c-m \pmod M] \\
&= \underset{k\leftarrow\mathcal{K}}{Prob}[k=k^*]
\end{aligned}$$

If we assume that the maximum number of ciphertexts to be added is n and that each plaintext is l-bit long, we must have $M = 2^{l+\lceil log(n)\rceil}$, i.e., $|M| = l + \lceil log(n)\rceil$. If $c_i = (m_i + k_i)$, then the probability that $c_i \in [0, 2^l - 1]$ is twice the probability that $c_i \in [2^l, M-1]$. More specifically, we have: $\underset{k\leftarrow\mathcal{K}}{Prob}[k = k^*] = 1/(2^l + M)$ if $c > 2^l$ and $\underset{k\leftarrow\mathcal{K}}{Prob}[k = k^*] = 2/(2^l + M)$ if $c < 2^l$.

Since these two equations hold for every $m \in \mathcal{M}$, it follows that for every $m_1, m_2 \in \mathcal{M}$ we have

$$\underset{k\leftarrow\mathcal{K}}{Prob}[Enc(k,m_1,M)=c] = \underset{k\leftarrow\mathcal{K}}{Prob}[Enc(k,m_2,M)=c]$$

which establishes perfect security of our scheme. $\square$