Secure Data Recollection for Redundantly Deployed Wireless Sensor Networks
Luis E. Palafox and J. Antonio Garcia-Macias
CICESE Research Center
Computer Science Department
{palafox,jagm}@cicese.mx

Abstract

Monitoring the environment is one of the main applications of wireless sensor networks. Given that these networks are densely populated and that local variations in the environmental variables are small, a large amount of redundant data is generated by sensor nodes. In this paper we present a protocol for data dissemination in hierarchical sensor networks that integrates security and reduces communication overhead by removing data redundancy from the network.

Furthermore, we show that by using our protocol we can provide security to the network without spending additional energy resources, particularly when we are dealing with high levels of redundancy.

1. Introduction

Recently, many potential applications for wireless sensor networks (WSN) that require hundreds or maybe thousands of nodes have been published; thus, high scalability is a fundamental requirement. Mesh-type networks do not fulfill this requirement, as every node plays the same role and has the responsibility of sensing the environment and routing traffic directed to other nodes and/or to the base station.

In contrast, hierarchical wireless sensor networks are highly scalable because they are organized in clusters. The nodes that belong to a specific cluster can only interact directly with a predetermined node that plays the role of the cluster leader, also known as cluster-head. Therefore, the tasks related to these hierarchical networks are more distributed as opposed to networks that have a flat structure (i.e. mesh networks). For instance, sensing the environment, and communicating the sensor readings to the cluster-head are tasks corresponding to the sensor nodes; meanwhile, the cluster-head has the responsibility of processing the data read from the sensor nodes in its cluster (i.e. through data aggregation) and sending the results directly to the base station or to a higher hierarchy level node in the network.

In this paper we focus particularly on the problem of transmitting the sensed data from the sensor nodes to the cluster-head. This problem may seem trivial at first, but in fact, as we mentioned earlier, many applications require highly populated networks that monitor changes in the environment. This requirement is mainly because the nodes involved are prone to failure and the environment on which they are deployed is very hostile. Furthermore, in order to reduce node cost there is no added physical protection to the sensor nodes. Due to the fact that placing many sensor nodes on the same area is common practice, the network frequently generates high levels of redundant data; this results in excessive amount of power consumed for communicating the data from the sensor nodes to the cluster-head.

It should be noted that although many documented applications require data integrity and security services such as authentication and confidentiality, security is an issue that has been frequently overlooked, in communication networks in general and in WSNs in particular. Additionally, because of the well documented extreme resource limitations of this type of networks, the question about the feasibility of integrating security mechanisms into WSNs has often been raised. Furthermore, being still an emerging research area, efforts in WSN have been directed toward issues such as energy-efficient protocol design, prototype application implementations and others, where security is not the most pressing aspect.

In this paper we introduce a novel node-to-cluster-head data transmission mechanism that takes full advantage of the high levels of data redundancy for reducing data consumption while also providing data integrity/security services. As we show at the end of the paper, the integration of our security oriented protocol does not incur additionally energy costs to the network; in fact, in some cases it results on significant power savings, which eventually could extend the life
of the network. This paper is organized as follows: in Section II we present some work related to security and data transmission in hierarchical sensor networks, in Section III we present our proposal for secure data dissemination in hierarchical wireless sensor networks, in Section IV we present a series of experiments that we performed with our protocol; finally, in Section V we give concluding remarks and outline future work.

Figure 1. Traditional authentication process [18].

2. Related work

We now briefly summarize some of the most relevant related work. Traditionally, protecting against most network attacks relies heavily on integrating an efficient authentication mechanism; in the context of data dissemination in hierarchical sensor networks, sending a Message Authentication Code (MAC) attached to the sensor reading has been used for this purpose. The MAC code is generated through a well known function (i.e. HMAC [1], SHA-1 [13]) that uses the sensed data and a secret key (K) as inputs. The secret key must be shared by all the sensor nodes and the cluster-head. The receiving node (in our case, the cluster-head) must try to duplicate the MAC through the received data and the secret key that it shares with the sensor nodes to verify the message authenticity. The authentication process is shown in Figure 1.

Perrig and Tygar [2003] show that integrating the MAC mechanism to a sensor network through the SPINS family of security protocols [15] only adds 6 bytes of overhead to the data packet. In terms of consumed energy this accounts for only 20% of the total amount of consumed energy. In Table 1 we show the cost of adding security to a sensor network through the SPINS security protocols. As can be observed, most of the energy cost is due to data transmission and not because of the computational cost added by the security algorithms.

Hu and Evans [2003] propose a secure aggregation technique that uses the µTESLA protocol to provide security. In this case, the nodes organize into a hierarchy tree where intermediate nodes play the aggregator role [9].

ESPDA [5] (Energy-Efficient Secure Pattern-based Data Aggregation) is a protocol based on pattern generation for hierarchical sensor networks that takes advantage of redundant data. However, their main goal is not security; but instead, a mechanism for selecting the nodes whose sensed region overlap, in order to know which nodes can be turned off. Additionally, redundancy detection is based on pattern generation through a specific algorithm that in order to be efficient has to yield some certain precision. In a later publication, the same authors propose an energy-efficient protocol for wireless sensor networks [4] that works in conjunction with ESPDA. This protocol defines a simple key distribution scheme for hierarchical sensor networks and uses the traditional approach to provide authentication (shown in Figure 1).

Table 1. Energy costs of adding security protocols to a sensor network [16].

<table>
<thead>
<tr>
<th>Percentage</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>70%</td>
<td>Data transmission</td>
</tr>
<tr>
<td>20%</td>
<td>MAC transmission</td>
</tr>
<tr>
<td>7%</td>
<td>None transmission (for freshness)</td>
</tr>
<tr>
<td>2%</td>
<td>MAC and encryption computation</td>
</tr>
</tbody>
</table>

Our approach is similar to ESPDA in the fact that it relies on redundant data to achieve power efficiency. However, as will be shown in the next section, our protocol defines a novel way for detecting redundant data and uses a two-phase mechanism to provide authentication. This mechanism differs substantially from the one shown in Figure 1.

3. Our approach for node to cluster-head data dissemination

3.1. Design

The proposed protocol is based on a novel two-phase authentication process, this process detects redundant data while achieving authentication. Additionally, other security services such as confidentiality can be provided trivially by means of the same secret key shared by the cluster. If:

$$A = MAC_f(K_1, M_1)$$ and $$B = MAC_f(K_2, M_2)$$ (1)

then

$$(A = B) \forall (K_1, M_1) = (K_2, M_2), f$$ (2)
In equations 1 and 2 we express that in our protocol, redundancy detection is based on the idea that for any given MAC function \( f \) applied to messages \( M_1 \) and \( M_2 \) with secret keys \( K_1 \) and \( K_2 \) respectively, the only way for their computed MAC values \( (A,B) \) to be the same, is that both messages are identical and both keys are identical. Therefore, the cluster-head should be able to detect redundant data after receiving the MAC codes for all the nodes inside its cluster.

One important aspect to consider in CBC-MAC is that collisions are a possibility, this is, two different messages can produce the same MAC code using the same key, and the presence of collisions would negatively affect our protocol by detecting redundant data while this would not be necessarily true. However, the probability of collision for our implementation (160-bit messages and 64-bit MAC codes) is very small [3]:

\[
P_{64}(160) \leq 1.3553 \times 10^{-18}
\]

Regarding message exchange in our protocol, we considered four types of messages, in Table 2 we list those messages along with a brief description of its payload and payload length. We assigned and ID value for each message type in our protocol for later use in our implementation.

**Table 2. Types of messages defined for our protocol.**

<table>
<thead>
<tr>
<th>ID value</th>
<th>Description</th>
<th>Payload (Length)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x4a</td>
<td>MAC code sent to clusterhead</td>
<td>MAC Code (8 bytes)</td>
</tr>
<tr>
<td>0x4b</td>
<td>Data request from clusterhead to sensor nodes</td>
<td>Node list (20 bytes max)</td>
</tr>
<tr>
<td>0x4c</td>
<td>Sensor data sent to clusterhead</td>
<td>Sensor data (20 bytes)</td>
</tr>
<tr>
<td>0x4d</td>
<td>Specific retransmission request</td>
<td>Empty(0)</td>
</tr>
</tbody>
</table>

For our test implementation we considered packets with a 10 byte header as in IEEE 802.15.4[10] with a maximum packet size of 30 bytes (20 bytes of payload) as in TinyOS [8].

The four defined messages are used during the different stages of the protocol. We describe the operation of the protocol as follows:

1. Nodes in a cluster sense data, cluster-head nodes are pre-defined through a cluster-head selection algorithm such as the one used by LEACH[7].
2. Using their sensor readings (DATA) and a counter (CNTR) that is synchronized with the rest of the cluster, each node computes its own 4 byte MAC code using a shared cluster key \( (K_{\text{cluster}}) \), this code is attached to a packet with type 0x4a and sent to the cluster-head. By using a secret counter that is shared by cluster members we provide implicit authentication (i.e. we do not ensure that a packet is sent by a specific node, but we ensure that the packet has been sent by an authorized node):

\[
\text{Node} \rightarrow \text{ClusterHead} : \quad \text{MAC}(K_{\text{cluster}}, \text{DATA} \ || \ \text{CNTR})
\]  

(4)

3. Once that all MAC codes are received (or after a predefined time-out period), the cluster-head stores them on a memory buffer classifying them according to the node from which each one came from (source node).
4. Afterwards, the cluster-head detects which MAC codes are duplicated in the buffer, and can trivially infer which nodes have redundant data.
5. Once that the nodes with redundant data have been identified, the cluster-head constructs a list of nodes (LIST) for which explicit data requests are necessary. Additional criteria, such as the remaining power on each node, can be used to include a node in the list.
6. Once that the list has been constructed, it is appended to one or more type 0x4b packets (depending on the size of the list). These packets should be broadcast from the cluster-head to the sensor nodes. The packets are authenticated with the counter we mentioned before to assure freshness and avoid “store and replay” attacks. As we will show ahead, this method is the simplest one because it does not involve protocols from any other layer, but it is not the most efficient one. A more efficient way to do this is to use some header bits in the ACK signals from the MAC layer to indicate to each node if its data is required by the cluster-head.

\[
\text{ClusterHead} \Rightarrow \text{Nodes} : \quad \text{List} \ || \ \text{MAC}(K_{\text{cluster}}, \text{LIST} \ || \ \text{CNTR})
\]  

(5)

7. The sensor nodes on which a specific data request was made, must integrate a packet type 0x4c with their sensor readings, this packet must be sent to the clusterhead:

\[
\text{Node} \rightarrow \text{ClusterHead} : \text{DATA}
\]  

(6)
8. Finally, the cluster-head knows the sensor readings from the nodes that received explicit requests, as well as from those that presented duplicated MAC codes; thus, verification of message integrity/authenticity can be performed by the cluster-head. If the verification fails, the cluster-head can request data to such nodes. Alternatively, an estimation algorithm may be used instead of asking for retransmission.

3.2. Implementation

A crucial aspect in the implementation of our protocol is deciding which cryptographic primitives to use.

The AES [6] algorithm is commonly used in traditional network security and we have evaluated using it. However, this algorithm consumes 800 bytes in lookup tables, a requirement that is not feasible for our implementation platform. Other alternatives were also analyzed, such as the DES algorithm [14], but it requires too much memory for storing permutation tables. Finally, we decided to use the RC5 [17] algorithm because it has many features that makes it attractive for our platform. For instance, it has a very low memory requirement, it is a fast algorithm and it is very flexible in terms of block and key size.

Regarding block and key size, we selected RC5-32/12/16, because according to the literature [17], it offers a good balance between efficiency and security level. Under this scheme, the cipher would process 32 bit blocks at a time through 12 rounds using a 16-byte cryptographic key.

There are many ways to compute the MAC code for a given message, like HMAC and SHA-1 among others. However, to implement any of these MAC functions we need to spend additional data and code memory. For this reason, we selected the CBC-MAC [2, 10] (Cipher Block Chaining Message Authentication Code) algorithm, which uses a conventional block cipher like RC5 iteratively to compute the MAC code. Thus, by selecting this algorithm we use the same block cipher to provide both confidentiality and authentication services, which obviously results in saved memory (data and code).

In Figure 2 we show the operation of the CBC-MAC algorithm, where each data block is processed by the cipher and its output is XORed with the next data block. The output of the last cipher block is the MAC code of the input message D.

The protocol was implemented on TMote Sky motes and TinyOS 1.1.15, the implementation was done based on an application that periodically sends readings from all the onboard sensors.

In Figure 3 we show the packet format for packets 0x4a and 0x4c (refer to Table 2), the first 10 bytes represent the header of a TinyOS standard message. Additionally, in our implementation, sensor data contains readings from the humidity, temperature, visible light, par light and voltage sensor aboard the TMote Sky motes. As shown in this figure, the sensor data length is 10 bytes and the MAC code is 4 bytes long, the sensor data as well as the synchronized counter (CNTR) are used to compute the MAC code. However, the counter is not sent in the message.

![Figure 2. Operation of the CBC-MAC algorithm [11].](image1.png)

![Figure 3. Packet format for messages containing sensor data and MAC codes.](image2.png)

In order to deal with small discrepancies in sensor readings due to sensor calibration we are using discrete intervals to group sensor reading values, each interval is represented by its central value, so when a reading falls into a given interval, the node transmits the MAC code computed with the central value of that particular interval. The drawback of this technique is that we are sacrificing precision but in exchange we are increasing the probability of obtaining redundant data which would obviously benefit power consumption as we will show in the next section. For instance, the TMote Sky has an on-board SHT11 humidity/temperature sensor from Sensirion, this sensor has a 14-bit resolution with an accuracy of ±0.4°C @ 25°C. According to technical specifications, the temperature (in Celsius
degrees) can be calculated from the 14-bit digital output \((SO_T)\) through the following equation:

\[
T = -39.60 + 0.01 \cdot SO_T
\]  

(7)

From equation 7 we can observe that each digital unit from the 14-bit output only represents a hundred of a Celsius degree. Therefore, if we relax our temperature accuracy requirements we could use our proposed intervals scheme in order to cope with calibration issues and small local variations among nodes in the same cluster. Particularly, in our implementation we considered intervals of 0.32°C, we did so by using simple bit manipulation: masking the first four least significant bits and setting the fifth to one. Thus, we avoided any kind of floating point operation in the mote, which is highly recommended due to their computational cost.

The intervals generated by using the described scheme yield to an error of ±0.16°C which is admissible for our application. Readings coming from other sensors were handled similarly; we only varied the interval width according to accuracy requirements for each sensed variable.

4. Experiments and simulations

A series of experiments were conducted, our first goal was to quantify the amount of additional power consumption that the proposed protocol introduces. Using Matlab, we modeled a cluster containing 20 sensor nodes with a 10% duty cycle and a 5 second period. Previously, we defined the messages and their respective lengths, thus, we can estimate the amount of traffic involved in the operation of our protocol.

We compared our protocol to the traditional node-to-cluster-head data transmission approach (with no security) where every sensor node transmits their raw sensor reading to the cluster-head on each period.

For our traffic experiments, we noticed that the amount of packets is always greater in our protocol than in the traditional approach because of the two-phase process. However, as we show in Figure 4, in our approach we transmit a much smaller total number of bytes; this is due to the fact that the packets containing the MAC code are considerably shorter in size as opposed to the packets containing the raw sensor readings. Thus, we incur less total traffic than the traditional approach.

![Figure 4. Plot of generated traffic vs. redundant data.](image)

Also, considering we use TMote Sky motes for our implementation, and knowing that those motes use AA batteries for their operation, we estimated the expected lifetime for each mote considering different levels of data redundancy for our cluster. We introduced the technical specifications of the motes to our simulation, specifically, those specifications related to power consumption for their different operation modes. A pair of AA batteries provides a current of approximately 2.5 Ah, but we also considered that it is impossible to consume all of the available current because at a certain point the provided voltage falls under the operating threshold for the motes; however, it is safe to assume 2200 mAh of battery charge [12].

For our energy consumption experiments we noted that our protocol did not extend the lifetime of the nodes considerably (about 3% lifetime increase at most) this is mainly because even though we send
shorter packets we spend additional energy receiving data requests from the clusterhead, and surprisingly, receiving consumes more energy than transmitting data in this particular platform. However, in Figure 5 we show that we extended the life of the clusterhead up to 20% depending on the redundancy. In this graph, we show the traditional approach along with our protocol in which we use the ACK messages from the MAC layer, and a third approach in which we only send ACK messages to the nodes where their data is required (called selective ACK approach). By using the latter, we enhance the performance of our protocol for power consumption; this is because we save energy by reducing the amount of ACK messages sent from the cluster-head to the nodes.

5. Concluding Remarks and Future Work

In this paper we showed that by integrating our security protocol to a hierarchical sensor network, we can provide integrity and authentication services without increasing energy consumption. We also showed that with an increased redundancy rate we consume less energy than the traditional approach where no security is provided. Thus, the main contribution of this paper is to provide security to wireless sensor networks by using the resources saved through our redundancy-driven protocol, specifically, bandwidth and energy.

The integration of a sensor reading estimation algorithm for adding robustness to our protocol is ongoing work, as well as measuring the performance of the proposed protocol on more populated networks.

References


