Secure and Efficient In-Network Processing for Sensor Networks

Tassos Dimitriou
Athens Information Technology
Markopoulo Ave., Peania, Greece
tdim@ait.edu.gr

Dimitris Foteinakis
Intracom S.A.
Markopoulo Ave., Peania, Greece
fotd@intracom.gr

Abstract

In this work we present a protocol that can be applied in wireless sensor networks in order to provide secure and authenticated in-network processing. Our protocol utilizes aggregator nodes which are responsible for data aggregation and command dissemination. The proposed protocol is simple and scalable and exhibits resiliency against node capture and replication as compromised nodes cannot be used to populate and eventually take over the network. It also allows for dynamic addition of new nodes and eviction of compromised ones by requiring minimum involvement of the base station. Finally, it is designed so that the majority of the sensors have to store only two keys plus a hash value, minimizing the storage capacity needed for the protocol to operate.

1 Introduction

Sensor networks have attracted much scientific interest during the past few years. These networks use hundreds to thousands of inexpensive devices over an area for the purpose of monitoring certain phenomena and capture distinct measurements over a long period of time. The large scale of sensor networks and the limited resources of sensor nodes in terms of computation power, memory, communication, and most importantly, energy make it an important challenge to design and develop efficient information processing and aggregation techniques.

Sensors in these multi-hop networks detect events and then communicate the collected information towards a central location. Since the cost of transmission is higher than computation it is usually advantageous to organize the sensors into clusters. In the clustered environment, the data gathered by the sensors is processed *within* the network and only aggregated information is returned to the central location. In such a setting, certain nodes in the sensor network, called *aggregators*, collect the raw information from the sensors, process it locally, and reply to the aggregate queries of the central server [1]. Since the sensors are now communicating data over smaller distances in the clustered environment, the energy spent in the network is much lower than the energy spent when every sensor communicates directly to the information processing center. Further benefits of aggregation processing include scalability as aggregation nodes may form multi-level hierarchies, and increased lifetime as aggregation processing reduces the volume of data exchanged and hence the overall energy spent for communication.

A closely related form of in-network processing is *data dissemination*, in which the network hierarchy is used in the reverse direction in order to disseminate control messages from the central server towards the aggregators and eventually towards the sensor nodes. For example, in tracking applications the sensor network must be used in both modes: first to aggregate sensed data about the movement of the tracked object and then to disseminate commands to nearby sensors to enable further tracking.

This and other applications in sensor networks require sensitive information to be delivered to the base station and be protected from disclosure to unauthorized third parties. The broadcast nature of the transmission medium, however, makes information more vulnerable than in wired communications. Moreover, due to strict resource constraints, existing network security mechanisms, even those designed for ad-hoc networks, are inadequate or not appropriate for this domain, so either they must be adapted or new ones must be created.

All security protocols in sensor networks should satisfy certain requirements in order for sensor nodes to be able to exchange data securely. The bare minimum consists of providing *confidentiality*, *authentication*, *integrity* and *freshness*. However, establishing secure communications between sensor nodes becomes a challenging task, mainly for two reasons: The first is how to bootstrap secure communications between sensor nodes, i.e. how to set up secret keys among them. If we know which nodes will be in the same neighborhood before deployment, keys can be decided a priori. Unfortunately, most sensor network deployments are random, therefore such a priori knowledge does not exist. The second is due to the limited processing power, storage, bandwidth and energy resources. Public-key algorithms, such as RSA are undesirable, as they are computationally expensive. Instead, symmetric encryption/decryption algorithms and hashing functions are between two to four orders of magnitude faster [2], and constitute the basic tools for securing sensor networks communication. Since the resources of a sensor node are very constrained, the key establishment protocols should be lightweight and minimize communication and energy consumption. (This requirement exists, even in the case of broadband sensor networks, were resources may be more abundant. Even if higher communication bandwidth and more processing power is available, we do not want the secure communication protocol to occupy those resources or need to develop a "heavier" protocol.) It should also be possible to add new sensor nodes incrementally to the sensor network. Finally, the protocols should be scalable, i.e., the size of the sensor network should not be limited by the per node storage and energy resources.

Our contribution: In this work we present new security mechanisms that can be used to provide *secure in-network processing* in wireless sensor networks. In particular, this means that we design the security mechanisms with both aggregation and dissemination in mind. Secure aggregation implies that data is forwarded from the sensors in a secure and authenticated way. Thus an adversary cannot issue false data into the network unless of course a particular sensor node has been compromised. Secure dissemination requires that lower level nodes are able to authenticate commands issued by their parents in the hierarchy. For both directions, protection is also provided against eavesdropping and tampering of data. Our protocol is simple and scalable and most importantly offers resiliency against node capture and replication as compromised nodes cannot be used to populate and eventually take over the network. Furthermore, it requires minimal key material (just three keys) for the majority of the sensors. As a result the proposed protocol can be applied to both broadband sensor networks, where more resources are available, but also to traditional sensor networks, where more constraints do apply.

The rest of the paper is organized as follows. In Section 2, we describe the sensor network architecture that we study. In Section 3, we present our security protocol that consists of various mechanisms for securing upstream messages from sensors to aggregators as well as downstream commands from aggregators to cluster nodes. We also discuss how to add and revoke nodes from the network. Section 4 deals with the resource requirements of our protocol. Finally, in Section 5, we discuss related work on similar architectures and we summarize our results in Section 6.

2 The model

In our model, a sensor network consists of a large number of sensors distributed over an area of interest. There is a base station in charge of the network's mission. The network also consists of super-nodes, called *aggregators*, in addition to the sensor nodes. We do *not* assume that the aggregators are more powerful in terms of energy, memory or computational resources, although as we will see in Section 4, they are required to store more keys and engage in more operations than simple nodes. But this is something to be expected by aggregators anyway, irrespective of the security mechanisms used.

We assume that the network is partitioned into distinct clusters and that each cluster is composed of an aggregator and a set of sensor nodes (distinct from other sets), which gather information and transmit it to the aggregator of their cluster. The aggregator fuses the data from the different sensors, performs mission-related data processing, and sends it to the base station.

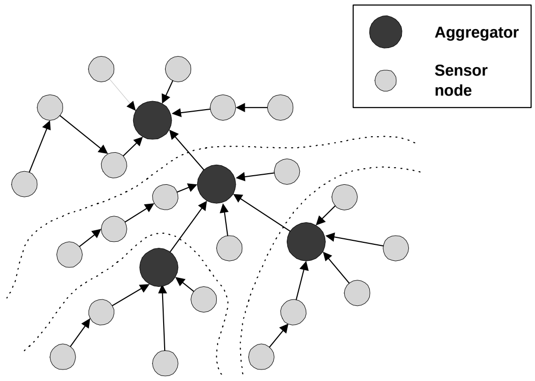

Figure 1. A hierarchical sensor network (arrows indicate inclusion in a particular cluster). Although aggregators are shown bigger, it should not be inferred that they are more powerful than simple sensor nodes.

Although we talk about clustered architectures, we do not go into details about the particular clustering protocol that was used to cluster the network, whether this is a standalone protocol or part of a routing one (see for example [3, 4] and the references therein). In general, we assume a hierarchical model of in-network processing (Figure 1) and focus on securing this model. Aggregators may form multiple levels of aggregation hierarchies encompassing any number of sensor nodes where an aggregator can both aggregate data as well as disseminate commands. Based on this model, the challenge is to design efficient mechanisms for establishing trust between the aggregators and the sensor nodes and provide for secure in-network processing.

3 The protocol

In this section we present a security protocol for secure in-network processing. We break this discussion into multiple subsections to ease the readability and understanding of the protocol. In Section 3.1, we describe the notation used throughout the paper and the key material held by each sensor before deployment. In Sections 3.2 and 3.3, we explain how the pairwise keys and the group key are computed for secure aggregation and dissemination processing, respectively. We also show how to defend against various types of impersonation attacks on the aggregator. Finally, we consider the problem of deleting compromised nodes from the system (Section 3.4) and adding new ones into the network (Section 3.5).

3.1 Notation and Keying Material

In our discussion we use the following notation:

Notation	**Meaning**
S_i	Identifier of sensor node i.
A_i	Identifier of aggregator i.
$M_1\|M_2$	Concatenation of messages M_1 and M_2.
$E_K(M)$	Encryption of M using key K
$MAC_K(M)$	MAC of M using key K.
N_A	Nonce used by node A.
K_{AB}	Symmetric key shared between A and B.

Each sensor node, S_i, has a unique key denoted K_{S_i} which is computed before deployment as follows:

$$K_{S_i} = F(K_m, S_i),$$

where $F()$ is a secure pseudo-random function, K_m is a master key and S_i is the unique identifier (number) for the i-th sensor node. This key will be shared between the sensor node and its aggregator and will be used to exchange data securely. Notice that an adversary upon compromising node S_i cannot recover the master key K_m from the key K_{S_i} because of the one-wayness of $F()$. Hence the rest of the network remains secured.

Each aggregator sensor node, A_j, has K_m stored in memory along with an individual key K_{A_j} which is derived from

$$K_{A_j} = F(K_m, A_j),$$

as described above. We will see later on that K_m is kept by A_j for as long it is necessary to establish secret keys with the nodes belonging in the A_j's group. Then it is *deleted* from the memory of the aggregator.

The base station holds K_m along with the keys of each sensor/aggregator node in the network.

3.2 Key Establishment for Secure Aggregation

To provide support for ensuring privacy and integrity of messages sent *from* sensor nodes *to* their corresponding aggregators a separate pairwise secret key between each sensor node and its aggregator is required. During deployment time, each sensor node is preloaded with the key K_{S_i} which is the result of the application of a secure pseudo-random function on the master key K_m. The same key K_{S_i} is computed by the aggregator by receiving an appropriate hello message M_{Hello} from the group nodes. M_{Hello} consists of the sensor's id S_i and a nonce N_{S_i} computed by the sensor, properly MACed as follows:

$$S_i \rightarrow A_j : M_{Hello}, MAC_{K_{S_i}}(M_{Hello})$$

Upon receiving this message, A_j computes K_{S_i} using the master key K_m and checks the MAC. If the MAC verifies, the sensor node is included in the cluster and the aggregator sensor stores all relevant information (such as K_{S_i}, nonce identifier - to avoid replay attacks, etc.). Additionally the aggregator may send back a reply to acknowledge the inclusion in the cluster (also MACed with K_{S_i}).

A good security practice is to use different keys for different cryptographic operations; this prevents potential interactions between the operations that might introduce weaknesses in a security protocol. Although not shown here, we use independent keys for the encryption and authentication operations, K_{encr} and K_{MAC} respectively, which are derived from the unique key K_{S_i} through another application of the pseudo-random function F, i.e. $K_{encr} = F(K_{S_i}, 0)$ and $K_{MAC} = F(K_{S_i}, 1)$. To simplify the notation, however, we usually omit including these keys in the security mechanisms when it is clear from the context.

After this process is completed for all the sensor nodes in the cluster, the master key is *deleted* from the memory of the aggregator. Since the security of our protocol depends on the deletion of the master from the memory of the sensor nodes, we should take care that the deletion is unrecoverable, for example by overwriting the master key (in practice several times). We have now established a secure channel between the aggregators and the sensor nodes.

An implicit assumption here is that the time required for the establishment of secure links with the sensor nodes is smaller than the time needed by an adversary to compromise an aggregator node during deployment. As security protocols for sensor networks should *not* be designed with the assumption of tamper resistance [5], we must assume that an adversary needs more time to compromise an aggregator and discover the master key K_m (see also [12] for a similar assumption). Notice also that we are not assuming that the aggregators are known in advance so that they are equipped with the master key. In fact all nodes may be equipped with this key and once the roles are decided only the aggregators retain this key so that they use it in the key establishment phase. Then, they delete it as explained above.

3.3 Key Establishment for Secure Dissemination

While the previous protocol allows for secure propagation of information collected by the sensor nodes, a related issue is how a command can be propagated to all sensors within a group. One simple way to do this is to send a separate unicast message to each member of the group encrypted and authenticated using the key shared between the aggregator and the specific node. However, this method is insufficient as the same command must be transmitted unnecessarily many times resulting in wasting valuable network resources.

A much simpler way for command dissemination is for the i-th aggregator to construct a group key G_{K_i} and propagate it during the network formation phase to all its group nodes, encrypted and authenticated each time using the sensor's private key S_{K_j}. The message sent to sensor S_j is as follows:

$$\begin{aligned} A_i: &\quad c = E_{K_{encr}}(\text{“Group key”}, A_i, G_{K_i}), \\ &\quad \sigma = MAC_{K_{Mac}}(c) \\ A_i \rightarrow S_j: &\quad A_i, c, \sigma \end{aligned}$$

First the aggregator encrypts the group key using the key K_{encr} derived from S_{K_i} and then creates a MAC σ of the resulting ciphertext c. Then it propagates this information to the j-th sensor node. One problem with this approach (even if it is used only once) is that it does not scale well when the aggregation group is very large. Then the communication overhead increases proportionally to the square of the group size. If, however, we make the reasonable assumption that nodes form hierarchies and are organized in a breadth first spanning tree based on some routing protocol, then the group key can be distributed recursively using intermediate level aggregators until all the sensor nodes at the leaves are reached.

Once the group key G_{K_i} is settled, the aggregator can broadcast commands to all the nodes in the group encrypted and authenticated using G_{K_i}. However, although this approach defends against outside attacks in which the adversary does not hold any keys, inside attacks are possible after an adversary compromises a sensor node. In such a case, a malicious node may use G_{K_i} to send forged messages to nodes in the same group. Below we propose a solution to defend against this impersonation attack.

$$\xleftarrow{Generate}$$

$$K_0 \xleftarrow{F(K_1)} K_1 \leftarrow \ldots \leftarrow K_{n-1} \xleftarrow{F(K_n)} K_n$$

$$\xrightarrow{Use/Reveal}$$

Figure 2. Construction of an one-way key chain

3.3.1 Defending against impersonation attacks

To thwart the impersonation attack described above, we propose to use a one-way hash key chain $OWHC_i$ computed by each aggregator A_i. Figure 2 shows that an one-way key chain is a sequence of numbers, $K_0, K_1, \ldots, K_{n-1}, K_n$ such that

$$\text{for all } l,\ 0 < l \leq n,\ \ K_{l-1} = F(K_l),$$

where as usual F is a secure pseudo-random function that is difficult to invert. Basically the aggregator generates the one-way hash chain of length n and commits to the first key K_0 by transmitting this securely to all the sensor nodes during the group key generation phase to avoid the extra communication overhead. Whenever aggregator A_i has a new command to disseminate to the nodes, it attaches to the command the *next* key from the hash chain. A node receiving a command encrypted with the group key can verify its authenticity by checking whether the new commitment K_l generates the previous one through the application of F. When this is the case, it replaces the old commitment K_{l-1} with the new one in its memory and accepts the command as authentic. Otherwise it rejects it.

The l-th command sent by the aggregator to the group nodes contains the l-th commitment of the hash chain and is encrypted and authenticated using keys derived from G_{K_i}. The format is as follows:

$$\begin{aligned} A_i: &\quad c = E_{G_{K_i}}(\text{“Command”}, A_i, K_l), \\ &\quad \sigma = MAC_{G_{K_i}}(c) \\ A_i \rightarrow Group: &\quad A_i, c, \sigma \end{aligned}$$

Note, however, that although the possibility of an impersonation attack is reduced, it is not completely eliminated by this scheme. For example, an adversary may *jam* communications to sensor S_j so that it misses the last k commands and hence commitments. Then it introduces new commands by “recycling” the unused commitments. It will be impossible for sensor S_j to notice the faked commands as the ordering of commitments is obeyed.

There are two ways to defend against this possibility: In the first, we may assume that sensors are loosely synchronized and commands are issued only at regular time intervals. In such a scenario, the command issued by an aggregator at time slot t will contain commitment K_t. If a sensor node does not receive anything within the next d slots and the last commitment was K_t, it will expect to see the commitment K_{t+d} that accounts for d missing commands. This way it cannot be fooled to authenticate unused commitments. Of course this scheme imposes a computational burden to sensor nodes as they have to apply the function F not once but d times to account for the unused commitments.

The second way to defend against the jamming attack is for the sensor to challenge the purported aggregator to send the command encrypted with the shared key K_{S_j}. The probability of challenging may depend on some particular distribution suited for the particular deployment scenario and network resources. Unless the adversary has compromised the aggregator A_i, it will not be able to answer the challenge as this key is shared only between S_j and A_i. When the node detects the impersonation attack, it sends a notice to the aggregator to take any actions required to recover from the compromise.

Notice that an implicit defense that not only eliminates but also helps detect these types of attacks is to have the sensor nodes respond to the aggregator using the shared key K_{S_j}. In this manner, even if an attacker has issued false commands, it will not be able to use the information sent by the sensor nodes. Additionally, if the aggregator receives responses to commands that it has not issued, it will become aware that an attacker has compromised some nodes in the cluster and eventually take corrective actions.

One last issue that remains to be covered is what happens when the one-way hash key chain is exhausted and the aggregator A_i cannot issue any more commands. This problem can be handled as follows: before A_i uses the last commitment it creates a new hash chain $K'_0, K'_1, \ldots, K'_{n-1}, K'_n$ and broadcast a “renew hash chain” command which contains the new commitment K'_0 authenticated with the last unused key of the old chain. This essentially provides the connection between the two chains and the group nodes will be able to authenticate commands as before. In a similar manner the group key can be refreshed to defend against cryptanalysis and ensure forward secrecy of previously distributed messages. The aggregator node can either periodically broadcast a new group key authenticated using the old one and the hash chain as usual, or every node can generate a new group key through the application of the pseudorandom function F. When it does so, it erases the old key from its memory.

3.4 Evicting compromised nodes from the network

Before we discuss a mechanism for dynamically inserting new nodes into the network, we need a scheme to evict compromised nodes and revoke their corresponding keys. Since we are not dealing with intrusion detection in this work, we assume the existence of a detection mechanism that informs the base station or the aggregators about compromised nodes (see also the related work of [6] on detecting forged aggregation values). We further assume that a sensor node cannot be compromised during the setup time of our system, either during the initial setup phase, or during the new node addition phase. This is a valid assumption, since the time needed for the setup phase is small in comparison to the time needed for the node to be captured.

If a node is compromised, the attacker cannot insert duplicates of that node in groups other than the group it originated from. This is the case, since no other aggregator will be able to compute K_{S_i}, which is essential for establishing the secure communication channel. Therefore, it suffices to provide a mechanism for node revocations transmitted by the base station to be authenticated and eventually for aggregators to revoke nodes within their groups.

As in the case of authenticating commands issued by the aggregators, we will base the revocation scheme on the use of one-way hash key chains. To use this scheme, we assume that sensor nodes are loosely synchronized and each aggregator is preloaded before deployment with the first key of the chain. The base station then discloses the keys in the key chain periodically in order reverse to the generation of these keys. Such use of the key chain allows the base station to broadcast authenticated messages to all aggregator nodes. When aggregators receive such lists of compromised nodes, they verify the authenticity of the messages and then evict the nodes from the cluster by establishing a new group key with the rest of the nodes using the method described in Section 3.3. At the end of this procedure, only the nodes that are not in the revocation list will obtain the new group key while compromised nodes cannot be used to insert or retrieve information that is transferred within the group.

In the case of compromised aggregators, we consider the entire cluster to be compromised as aggregators hold pairwise keys with all the cluster nodes. However, the damage is confined to the aggregator’s group of sensors as the compromised aggregator cannot impersonate any other ones. In such a case, the parent nodes in the cluster hierarchy can establish new group keys with the rest of the sensors/aggregators using the method described above.

3.5 Adding new sensor nodes in the network

When we want to add new sensors in the network, each of the new sensors, say S'_i is supplied with a unique key, $K_{S'_i}$, which is derived from a new master key K'_m in the manner described in Section 3.1. The new master key is also forwarded to all the aggregator nodes from the base station, using the key shared between the base station and the aggregator in order to create a secure channel.

The new nodes are randomly inserted in the network. After they have settled, they discover their corresponding aggregator sensor node and they send the following hello message containing their identifier and a nonce:

$$S'_i \rightarrow A_j : M_{Hello}, MAC_{K_{S'_i}}(M_{Hello})$$

In the same manner as described in the initialization phase paragraph (Section 3.2), the aggregator computes $K_{S'_i}$ and authenticates the hello message. If all checks out, the new node is added to the cluster and the aggregator creates a new group key which broadcasts to the group.

4 Resource Requirements

In this section we analyze the computational, communication and storage requirements for our key establishment protocol. The individual key of each sensor node is precomputed and does not involve any processing or transmission overhead. The cost for establishing a group key in our protocol is the same as updating the group key in the cluster, thus we only analyze the cost for establishing the group key. We should emhasize at this point that our protocol is independent of the underlying routing protocol used.

4.1 Computational/Communications Cost

In order for a sensor node to join a cluster, the sensor node has to compute the message described in Section 3.2 which includes a MAC computation based on a secret key. In order to process this message the aggregator has to perform two operations. First, it needs to compute the key of the sensor node by applying a secure pseudorandom function on the master key and then it has to perform a MAC verification operation. Therefore, if we assume that a cluster includes d sensor nodes on average, the aggregator has to compute d values from the pseudorandom function and perform d MAC verifications during the initialization phase.

In order to establish a group key, the aggregator has to compute the message described in Section 3.3 for each sensor node in its cluster. This message includes an encryption and a MAC operation based on a secret key. Additionally the aggregator has to compute the one way key chain. Again, assuming d sensor nodes in each cluster on average and an n level OWHC, the aggregator needs to perform d encryptions, d MAC and n hash operations. For the same process, each sensor node has to perform a decryption and MAC verification operation.

Since both the initialization and group establishment phases take place *only* during network setup, we see that the computational/communications cost is really averaged out throughout the network's life. The only "expensive" operation is the computation of the n-length hash key chain by the aggregator. But since the aggregator uses a commitment only when it has to issue a command, we see again that the cost per hash value is much smaller than the transmission cost of the disseminated command [2].

4.2 Storage Requirements

Each sensor node has to store in memory its individual key, the current group key and the current value of the One Way Hash Chain. Assuming that the each of the above is 10 bytes long (80 bits), each sensor has to store only 30 bytes of information. The aggregator node needs to store the key of each sensor node in its cluster (d in total), at most n values of the OWHC and its individual key, needed for communication with nodes higher in the hierarchy (for example with the Base Station). Assuming that each of the above values is 10 bytes long, each aggregator has to store $10 * (d + n + 1)$ bytes of information.

The value of d really depends on the density of the network and the particular clustering protocol used and the value of n on the frequency the aggregator issues commands. If we assume $d = 20$ and $n = 50$, the aggregator has to store 710 bytes in memory. Since the value of n can be as small it takes (provided the hash chain is renewed regularly), the above analysis shows that memory requirements are not an issue in our scheme, since typical sensor nodes have significantly more available memory. Furthermore, we see that the vast majority of sensors needs to store just three keys.

5 Related Work

Many critical applications require that data must be exchanged in a secure way. Establishing secure communications between sensor nodes becomes a challenging task, given their limited processing power, storage, bandwidth and energy resources [2].

There are many protocols that attempt to establish pairwise keys between sensors. For example the pebblenets architecture [7] uses a global key shared by all nodes. Having network wide keys for encrypting information is very good in terms of storage requirements and ease of use but it suffers from the obvious security disadvantage that compromise of a single node undermines the security of the entire network.

Random key pre-distribution schemes [8, 9, 10, 11] operate without the cooperation of a base station and offer a tradeoff between the level of security achieved and the memory storage required by each sensor to store the keys that have been randomly chosen from a key pool. These schemes offer, however, only "probabilistic" security as compromise of a single node may result in a breach of security in some other part of the network. Furthermore, these schemes have not been developed with the notion of aggregation processing in mind which makes establishing group keys less straightforward.

A protocol that uses similar techniques to ours is LEAP [12]. In this work the authors describe a methodology for establishing pairwise and other keys between sensor nodes with the help of a pre-deployed master key that eventually gets deleted from the memory of the sensors. During the neighbor discovery phase each node broadcasts a HELLO message (u, ID_u). Upon receiving such a message, a sensor node v, computes a pairwise key K_{uv} using a pseudo-random function and the master key. The pairwise keys are then used to establish cluster keys for some form of aggregation processing, where each cluster is confined *only* to the immediate neighborhood of each node.

We have discovered however, that even if the master key is deleted, the LEAP protocol can be attacked. More specifically an attacker may force a sensor node to compute pairwise keys with other (or *all*) nodes in the network. This is achieved by having the attacker broadcast a large number of HELLO messages (nothing prevents her from doing so). The recipient node, will compute all the pairwise secret keys according to the protocol. Then, once the neighbor discovery phase terminates, an attacker can compromise a sensor node and have in her procession a key that is shared between the compromised node and all other nodes in the network.

The above attack would be useless on its own, if there wasn't for the node addition phase supported by the protocol. The LEAP protocol, supports the addition of new nodes and enables new nodes to establish pairwise shared keys with sensor nodes in their neighborhood. This is achieved by having new nodes loaded with the master key and executing a protocol similar to the one described in the node discovery phase. Utilizing this capability, an attacker that has performed the attack described in the previous paragraph, can insert new nodes in the network since she will have at her disposal valid pairwise keys for all the nodes in the network. This essentially allows an attacker to compromise the entire network as the new nodes will be able to monitor all traffic and to insert false data in the network. Our proposal negates this disadvantage of LEAP. Sensor nodes do not carry the master key and more importantly an attacker cannot send valid HELLO messages during the initialization phase, since HELLO messages are accompanied with a MAC with the sensor node's key.

Two similar architectures to our proposal were also discussed in [13] and [14]. The authors in [13] propose a collection of mechanisms to address the requirements imposed by secure (both upstream and downstream) in-network processing. Their approach makes heavy use of the base station as a means for introducing and delegating authorization of aggregators to group nodes and for establishing keys between the nodes and the aggregators they belong to. Furthermore, long lists of keys and topology information are required to be transmitted, making the scalability of this scheme problematic for large size networks.

In [14], the authors introduce *gateway* nodes, which function as the aggregators in our proposal. The protocol, however, requires a large number of messages to be exchanged during the initialization phase, while our proposal only requires two messages to be exchanged (one message from the sensor node to the aggregator and one response). This is due to the fact that gateway nodes in their protocol are randomly assigned a number of sensor node keys, which requires gateways to exchange large number of keys if a node happens to "land" in the wrong cluster. Additionally, if the number of gateways in the network is large, the probability that the original gateway holds the keys of the sensor nodes in its cluster diminishes.

6 Conclusions and Future Work

In this paper we have presented security mechanisms for wireless sensor networks that provide secure in-network processing. We have designed the proposed mechanisms with both aggregation and dissemination in mind. Our proposal accommodates both secure aggregation of data so that they are forwarded to the base station, as well as secure dissemination of commands issued by the base station to all sensors in the network.

The mechanisms presented in this work also support the addition of new nodes to the network, a critical requirement in sensor networks, as sensors have limited energy and thus limited life expectancy. Additionally, given the existence of an intrusion detection mechanism, our protocol allows for the eviction of compromised nodes. We should also note that the successful application of our proposal neither depends on the existence of location information or on the underlying routing protocol, nor does its performance vary depending on the density of the network, like probabilistic schemes. The proposed protocol scales very efficiently as it is based on forming clusters in a hierarchical manner, which is the most efficient architecture when networks become very large. Finally, the number of keys that each sensor node has to store is small, only three for regular sensors, and the protocol does not require any computationally intensive operations.

As an extension to the proposed protocol, we consider to add a mechanism to change the aggregator node within each cluster. This seems to be necessary since the aggregator consumes more energy than regular sensor nodes and its resources may be drained faster. Therefore we are considering mechanisms for the aggregator to delegate its authority to other sensor nodes in its cluster.

References

[1] C. Intanagonwiwat, R. Govindan, D. Estrin, J. Heidemann, and F. Silva, "Directed diffusion for wireless sensor networking," *ACM/IEEE Transactions on Networking*, vol. 11, February 2002.

[2] D. Carman, P. Kruus, and B.J.Matt, "Constraints and approaches for distributed sensor network security," Tech. Rep. 00-010, NAI Labs, June 2000.

[3] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan,"Energy-efficient communication protocols for wireless microsensor networks," in *Proceedings of the Hawaaian International Conference on Systems Science*, January 2000.

[4] O. Younis and S. Fahmy,"Distributed Clustering in Adhoc Networks: A Hybrid, Energy-Efficient Approach," in *INFOCOM*, 2004.

[5] R. Anderson and M. Kuhn, "Tamper resistance – a cautionary note," in *Proceedings of the Second Usenix Workshop on Electronic Commerce*, pp. 1–11, November 1996.

[6] B. Przydatek, D. Song and A. Perrig, "SIA: Secure Information Aggregation in Sensor Networks," *in ACM SensSys*, 2003.

[7] S. Basagni, K. Herrin, D. Bruschi, and E. Rosti, "Secure pebblenet," in *Proceedings of the 2001 ACM International Symposium on Mobile Ad Hoc Networking & Computing, MobiHoc 2001*, pp. 156–163, October 2001.

[8] L. Eschenauer and V. D. Gligor, "A key-management scheme for distributed sensor networks," in *Proceedings of the 9th ACM conference on Computer and communications security*, pp. 41–47, 2002.

[9] H.Chan, A.Perrig, and D.Song, "Random key predistribution schemes for sensor networks," in *IEEE Symposium on Security and Privacy*, pp. 197–213, May 2003.

[10] D. Liu and P. Ning, "Establishing pairwise keys in distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pp. 52–61, October 2003.

[11] W. Du, J. Deng, Y. S. Han, and P. K. Varshney, "A pairwise key pre-distribution scheme for wireless sensor networks," in *Proceedings of the 10th ACM conference on Computer and communication security*, pp. 42–51, October 2003.

[12] S. Zhu, S. Setia, and S. Jajodia, "LEAP: efficient security mechanisms for large-scale distributed sensor networks," in *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pp. 62–72, October 2003.

[13] J. Deng, R. Han. S. Mishra, "Security Support for in-network Processing in Wireless Sensor Networks," in *ACM Workshop on Security of Adhoc Networks (SASN)*, 2003.

[14] G. Jolly, M. Kuscu, P. Kokate and M. Younis, "A Low-Energy Key Management Protocol for Wireless Sensor Networks," in *Proc. 8th International Symposium on Computers and Communication*, 2003