

## Chapter 12

# Secure In-Network Processing in Sensor Networks

Tassos Dimitriou  
*Athens Information Technology*  
*19.5 km Markopoulo Ave.*  
*19002 Peania, Athens, Greece*  
E-mail: [tdim@ait.edu.gr](mailto:tdim@ait.edu.gr)

Ioannis Krontiris  
*Athens Information Technology*  
*19.5 km Markopoulo Ave.*  
*19002 Peania, Athens, Greece*  
E-mail: [ikro@ait.edu.gr](mailto:ikro@ait.edu.gr)

## 1 Abstract

In-network processing in large-scale sensor networks has been shown to improve scalability, eliminate information redundancy and increase the lifetime of the network. In this chapter we address the challenge of securing in-network processing, both for aggregating sensor node's measurements and disseminating commands from the aggregators to individual sensor nodes. We present the key mechanisms for establishing a secure communication channel between sensor nodes and aggregators and show how scalability and resiliency against different type of attacks can be achieved. We also present how requirements such as dynamically adding new nodes in the network and harmonious existence with other security protocols of the network can be met. Finally we elaborate on resilient aggregation under corrupted measurements and the proposed solutions.

## 2 Introduction

Wireless sensor networks are envisioned to consist of hundreds of thousands of inexpensive nodes, each with some computational and sensing capabilities, operating in an unattended mode and communicating with each other. The purpose of deploying a sensor network is to monitor an area of interest with respect to some physical quantity, e.g. temperature. This information is gathered by the sensors and reported to a point which we refer to as base station. This enables a broad range of environmental sensing applications from vehicle tracking to habitat monitoring [1].

Sensor networks are usually characterized by power, processing and storage limitations. These limitations motivate the design of new network architectures and protocols, which, for a given initial battery energy try to make sure that the network continues to function and provide data updates for as long as possible. During data gathering, nodes spend a part of their initial energy on transmitting, receiving and processing packets. It has been shown [2] that communication takes the largest share, about 70%, of the total energy consumption of nodes. Therefore, it is desired that new communication protocols minimize any redundant transmissions and reduce the amount of data relayed in the network, so that the lifetime of sensor nodes is maximized.

Since sensor nodes have a limited energy which may be exhausted, sensor networks are densely deployed to deal with connectivity and coverage problems. This causes neighboring nodes to have overlapping sensing regions and generate correlated measurements whenever an event occur in this overlap. Moreover, sensor nodes are usually deployed randomly, which reinforces the effect of overlapping regions. Each node observes its sensing region independent of its neighbors and sends its measurements to the base station. Therefore, communication overhead can be substantially reduced, if raw data sent by nodes can be combined to eliminate redundancy and reduce the number of transmissions.

A technique that has been used to deal with these problems is data aggregation (or data fusion) [3, 4, 5]. The key idea is to combine highly correlated data coming from different sensors into one packet. This happens at intermediate nodes, called *aggregators*, which compute an aggregated value of all the measurements (e.g., an average or maximum temperature), and then forward only a single packet with the resulted value. The reverse process is called *data dissemination*, in which the network hierarchy is used in the reverse direction in order to disseminate control messages from the base sta-

tion towards the aggregators and eventually towards the sensor nodes. For example, as shown in Figure 1, in tracking applications the sensor network may need to 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 [6].

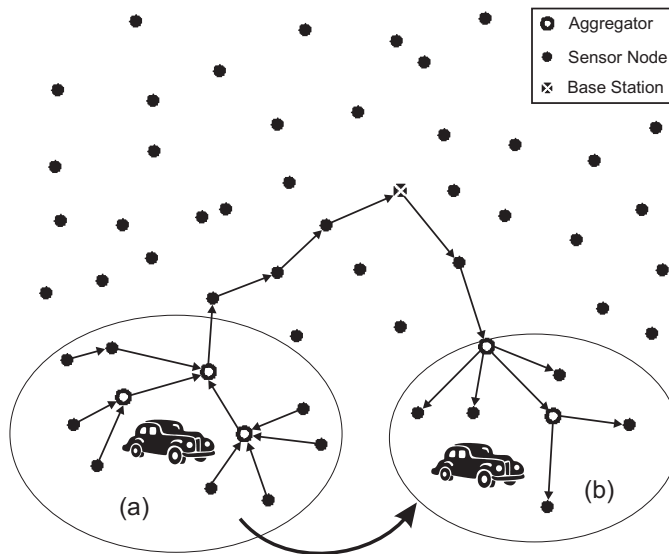


Figure 1: A tracking application using both (a) aggregation and (b) dissemination.

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 sen-

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 reason that makes security in sensor networks hard is 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 [7], 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.

### 3 Threat Model

In sensor networks security, an attacker can perform a wide variety of attacks. Not all of them have the same goal or motivations. So, in order to plan and design better defense systems, we formulate a threat model that distinguishes between two types of attacks:

#### 3.1 Outsider Attacks

In an outsider attack (intruder node attack), the attacker node is not an authorized participant of the sensor network. Authentication and encryption techniques prevent such an attacker to gain any special access to the sensor network. The intruder node can only be used to launch passive attacks, like the following:

- *Passive eavesdropping*: The attacker eavesdrops (listens) and records (saves) encrypted messages. The messages may then be analyzed in order to discover secret keys.
- *Denial of service attacks*: In its simplest form, an adversary attempts to disrupt the networks operation by broadcasting high-energy signals. In this way, communication between legitimate nodes could be jammed, or even worse, nodes can be energy depleted.

- *Replay attacks*: The attacker captures messages exchanged between legitimate nodes and replays them in order to change the aggregation results.

### 3.2 Insider Attacks

Perhaps more dangerous from a security point of view is an insider attack, where an adversary by physically capturing a node and reading its memory, can obtain its key material and forge node messages. Having access to legitimate keys, the attacker can launch several kinds of attacks without easily being detected:

- *False data injection (stealthy attack)*: the attacker injects false aggregation results, which are significantly different from the true results determined by the measured values
- *Selective reporting*: the attacker stalls the reports of events that do happen, by dropping legitimate packets that pass through the compromised node.

The consequences of these attacks are further magnified in hierarchical networks, where nodes organize themselves in a tree and nodes aggregate results coming from the subtree rooted at them. Therefore, the position of the compromised node may extremely affect the aggregation result. If the attacker compromises a node higher up the routing tree, she can effectively misrepresent the readings of a large portion of the sensor network.

Of course, an adversary cannot have unlimited capabilities. There is some cost associated with capturing, reverse-engineering and controlling a node. Therefore, we should assume that the adversary can compromise only a limited number of sensor nodes. This fact affects the design of security protocols, as it is easier to offer some protection against a few compromised nodes, but not for the case where a large portion of the network is in control of the attacker.

## 4 Secure In-Network Processing

In wireless sensor networks, the first step on providing security for data communication is the establishment of shared keys between pairs of nodes so that they can encrypt and authenticate data exchanged between them. However,

caution must be taken so that in-network processing is not hindered by the underlying security protocol. In particular, the following requirements must be supported by the key management scheme:

1. Data aggregation is possible only if intermediate nodes have access to encrypted data so that they can extract measurement values and apply to them aggregation functions. Therefore, nodes that send data packets towards the base station must encrypt them with keys available to the aggregator nodes.
2. Data dissemination implies broadcasting of a message from the aggregator to its group members. If an aggregator shares a different key (or set of keys) with each of the sensor within its group, then it will have to make multiple transmissions, encrypted each time with a different key, in order to broadcast a message to all of the nodes. But transmissions must be kept as low as possible because of their high energy consumption rate.

So, the use of pair-wise shared keys between the nodes and the base station effectively hinders in-network processing. A solution that could satisfy the above requirement could be the use of a key common to all sensor nodes in the network. However, the problem with this approach is that if a single node is compromised then the security of the whole network is disrupted. Furthermore, refreshing the key becomes too expensive due to communication overhead.

A localized distributed algorithm for key establishment in sensor networks that works well with data aggregation is presented in [8]. A scheme is proposed that utilizes the established keys in order to provide secure communication between a source node and the base station, while intermediate nodes can access data and perform aggregation. The way this is achieved is by grouping nodes into clusters, where a common key is shared between the nodes of the same cluster. Data are encrypted using that key, so aggregation and dissemination within the cluster is very efficient.

Since information needs to travel between clusters, nodes from different clusters that are in range of each other must share their keys (cluster keys), so that they can encrypt and authenticate messages exchanged between them (see Figure 2). This is like creating bonds between clusters to allow secure information flowing. It turns out that depending on the network density, the total number of keys a node needs to store is no more than 5.

Furthermore, disclosure of one node’s keys allows an attacker to disrupt the communication within this neighborhood, and not in the rest of the network.

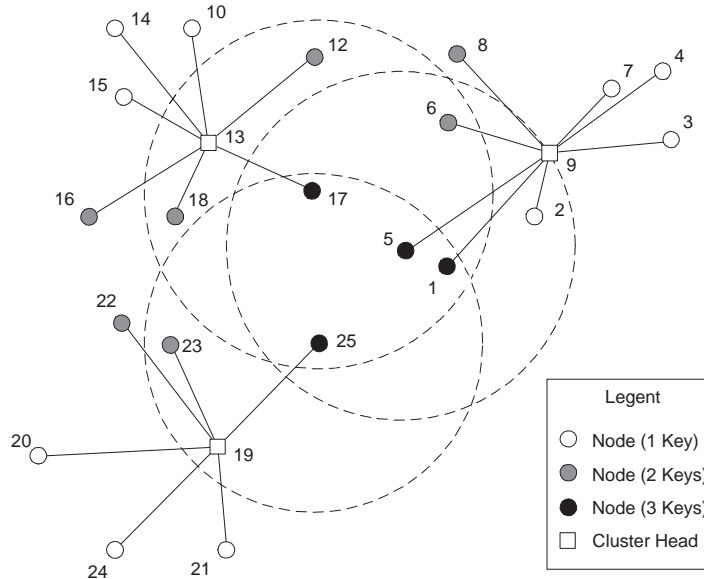


Figure 2: An example topology with 3 clusters. Each node stores its own cluster key as well as the keys of the clusters that are within its communication range. Communication ranges of nodes 25, 17 and 5 are shown.

This protocol uses clustering only for the key establishment phase. After that phase, communication does not use any hierarchical model. In large sensor networks however, where more than one aggregator exist, there is the need to use a hierarchical aggregation model, where aggregation nodes form a tree. As also depicted in Figure 3, the main idea is that each aggregator collects information for a subset of nodes (group or cluster) and then forwards appropriate summaries to other aggregators closer to the base station. When an end user casts a query to the network, required data are obtained more efficiently by this method compared with non-hierarchical approaches. In this case, key establishment must follow a more specific communication scheme, where aggregators collect information from their cluster members and disseminate commands to them. Based on this scheme, we present an efficient mechanism to establish trust between aggregators and sensor nodes

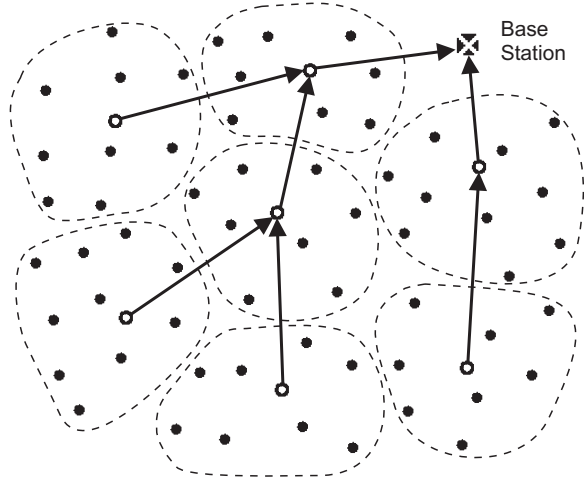


Figure 3: A general hierarchical aggregation scheme. The data of each node are gathered to the aggregator that is the representative node in each cluster. The information from the aggregators is further aggregated to compute the final result.

that allows secure in-network processing [9]. Other protocols that focus on securing in-network processing based on different hierarchical organizations and more stronger assumptions on the capabilities of sensor nodes are described in [10] and [11].

Throughout the rest of the chapter 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 $K$
$MAC_K(M)$	Message Authentication Code (MAC) of $M$ using key $K$ .

Initially each sensor node is loaded with two keys: A master key  $K_m$ , shared among all nodes, and a unique key, denoted  $K_{S_i}$  for nodes and  $K_{A_i}$  for aggregators<sup>1</sup>. The master key as well as the keys of each sensor/aggregator

<sup>1</sup>We don't imply by this that the aggregators are known in advance. Once nodes determine their roles through the use of some clustering protocol, then we can talk about

are known to the base station. For each sensor node,  $S_i$ , the unique key is computed before deployment as follows:

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

where  $F()$  is a secure pseudo-random function. 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.

Correspondingly, for each aggregator,  $A_j$ , the unique key  $K_{A_j}$  is derived from

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

We will see in Section 4.1 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 as well as the rest of the nodes in order to eliminate the possibility of being retrieved by an adversary who has physically captured a node.

In what follows, although not shown, we assume the use of different keys for different cryptographic operations; this prevents potential interactions between the operations that might introduce weaknesses in a security protocol. Therefore, we use independent keys for the encryption and authentication operations,  $K_{encr}$  and  $K_{MAC}$  respectively, which are derived from each unique key  $K_{S_i}$  through another application of the pseudo-random function  $F$ , i.e.  $K_{encr} = F(K_{S_i}, 0)$  and  $K_{MAC} = F(K_{S_i}, 1)$ .

#### 4.1 Key Establishment for Secure Aggregation

As we said in the previous section, 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  $K_m$ . In order for a node to communicate with its aggregator, this key must be available in both sides. So, the node must first inform the aggregator about its key in a secure way. This can happen by sending an appropriate “hello” message  $M_{Hello}$ :

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

where the  $M_{Hello}$  consists of the sensor's id  $S_i$  and a nonce  $N_{S_i}$  computed by the sensor.

---

the keys  $K_{S_i}$  and  $K_{A_j}$ .

Having received this message, the aggregator  $A_j$  is now able to compute  $K_{S_i}$  using the master key  $K_m$  and authenticate it by checking 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  $K_{S_i}$ ).

The same procedure is repeated for every node with its aggregator, in an initial phase after the deployment of the network. This phase is secure as long as the adversary does not know the master key. Therefore, we must assume that the phase is too short in time for an adversary to capture a node and retrieve the master key  $K_m$  (see also [8, 12] for a similar assumption).

The memory requirements of this phase are determined by the number of keys each aggregator has to store, i.e. the number of nodes in each cluster. Figure 4 shows the average number of keys each aggregator has to store as a function of the number of aggregators in the network. As expected, the more the aggregators in the network, the more the clusters that will be formed and therefore, the less the average number of sensor nodes in each cluster.

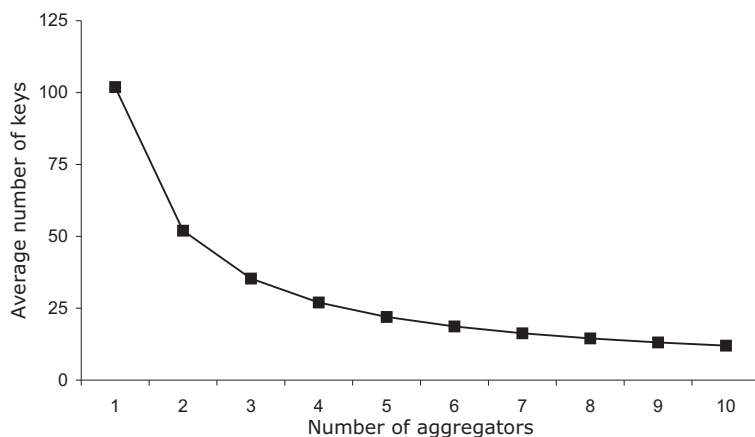


Figure 4: Average number of keys an aggregator has to store in a random network of 1000 nodes.

However, since we consider random topologies, it may be the case that a

cluster have more members than the others. Figure 5 shows the maximum number of keys found in aggregators after 1000 experiments in random networks of 1000 sensor nodes. In all the experiments 50 aggregators were randomly selected and each sensor node joined the cluster of the closest aggregator. The figure shows the distribution of the maximum number of keys found in any aggregator, which is within reach of current sensor nodes.

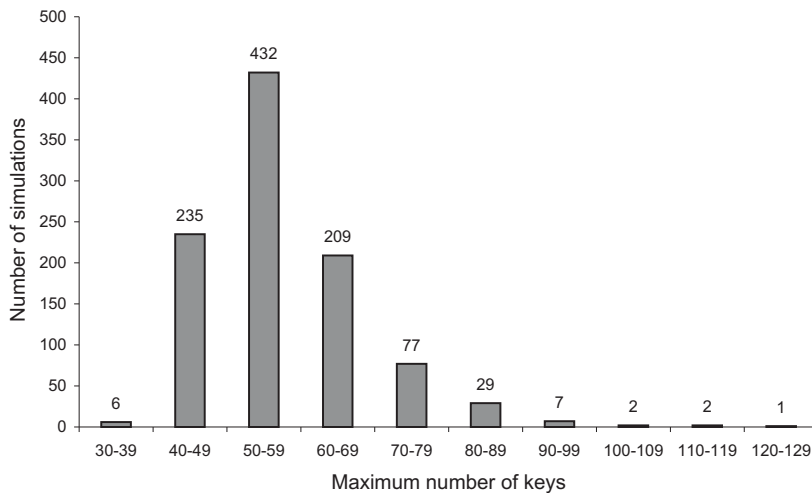


Figure 5: Maximum number of keys an aggregator has to store in a random network of 1000 nodes. The figure shows that in most cases (432 experiments out of 1000) the maximum number of keys found in any aggregator was between 50-59.

After the completion of the phase, the master key is *deleted* by the memory of every node. 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.

## 4.2 Key Establishment for Secure Dissemination

Having secured the communication of sensor nodes towards their aggregators, we now need to secure the reverse procedure: the dissemination of messages/commands *from* the aggregators *to* the sensor nodes. One simple but inefficient solution would be 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. This would result in several transmissions of the same message, wasting energy resources.

The solution to this problem is for the aggregators to construct and propagate a group key to its members during the initial phase after deployment. In this way, dissemination can take place later by a single broadcast of the message encrypted by that key. Each aggregator  $A_i$  constructs and sends a group key  $G_{K_i}$  to each sensor  $S_j$  that belongs to its group:

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

The group key is encrypted and authenticated each time using the sensor’s private key  $K_{S_i}$ , which is available to the aggregator as described in the previous section. First the aggregator encrypts the group key using the key  $K_{encr}$  derived from  $K_{S_i}$  and then creates a MAC  $\sigma$  of the resulting ciphertext  $c$ .

Once the aggregators settle a group key  $G_{K_i}$  with their group members, they can broadcast commands encrypted and authenticated using  $G_{K_i}$ . This is sufficient to secure communication from an outsider who does not hold the group key. However, an insider adversary, who has captured a group node and retrieved  $G_{K_i}$ , can impersonate the aggregator and send forged messages to nodes in the same group. Therefore we need to secure further the dissemination process.

### 4.2.1 Defending against impersonation attacks

We enhance the security protocol described above, in order to ward off insider attacks that impersonate the aggregator and try to disseminate messages to the group. The solution is that whenever an aggregator  $A_i$  has a new command to disseminate to the nodes, it attaches to it the *next* key,  $K_l$ , from an one-way key chain, as follows:

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

So, 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  $G_{K_i}$ .

One-way key chains are a widely-used cryptographic primitive. To generate a chain of length  $n$  we randomly pick the last element of the chain  $K_n$ . Then, each element of the chain is generated by repeatedly applying an one-way function  $F$ , until the  $K_0$  element, which is the commitment to the whole chain. We then reveal the elements of the chain in reverse order,

$$K_0, K_1, \dots, K_l, \dots, K_{n-1}, K_n.$$

If we know that  $K_{l-1}$  is part of the chain, we can verify that  $K_l$  is also part of the chain by checking that  $K_{l-1} = F(K_l)$ . Therefore, 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.

One issue with the one-way key chain is that it limits the length of the trust delegated to an aggregator. This is because the length of the chain determines the number of packets that the aggregator can send to its sensor group members. That is, for a chain of length  $n$  the aggregator can send at most  $n-1$  separate commands at the nodes. An aggregator can renew its key chain as follows: before the aggregator uses the last commitment, it creates a new hash chain  $K'_0, K'_1, \dots, K'_{n-1}, K'_n$  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.

So far we have limited the possibility of an impersonation attack, but we have not eliminated it. There is still a scenario were an adversary can *jam* communications to a 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 followed.

In order to defend against an attack like this we may assume that sensors are loosely time synchronized and commands are issued only at regular time intervals. That means a packet broadcast from the aggregator at time slot  $t$  contains 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. In this way it cannot be misled and authenticate unused commitments.

Finally, we must note that an implicit defense against impersonation attacks 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  $K_{S_i}$ . 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.

### 4.3 Adding new nodes in the network

This section address the problem of refreshing the network as sensors usually have limited lifetime and usually die of energy depletion. Each new node  $S'_i$  comes equipped with a unique key,  $K_{S'_i}$ , which is derived from a new master key  $K'_m$ , in the way we have already described. 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.

Every new node transmits a hello message to its neighbors indicating its will to become a member of an existing group. The message contains its identifier and a nonce:

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

In the same manner as described in Section 4.1, 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.

## 5 Resilient aggregation

So far we have described a key management scheme that allows aggregators to authenticate sensor nodes and vice versa. As a result, a malicious node injected in the network cannot authenticate itself and disrupt in-network processing. If, on the other hand an adversary physically captures an existing node and retrieves its key material, it will also be hard for her to impersonate an aggregator and disseminate messages. However, a compromised node can authenticate itself to the aggregator and start sending false readings, in order to affect the aggregation result. It is difficult to detect such an attack since that would require application (semantics) specific knowledge.

What can be done in this case is to make aggregation functions more resilient to some corrupted measurements. That means a few corrupted measurements should not cause large errors in the computed aggregate, under the assumption that only a very small fraction of nodes can be compromised.

In research work done on this direction [13] it is proposed that instead of aggregating measurements by computing the sum or the average, more resilient functions can be used. For example the median or the 5%-trimmed mean, where the highest and lowest 5% of the sensor readings are ignored, can be fairly robust for large network sizes and high node densities (redundancy).

Another approach is described in [14] that is termed the aggregate-commit-prove technique and used to authenticate the results sent by aggregators in the base station. That is, even if an attacker captures the aggregator and forces it to produce false results (within a bound), we can detect this attack with high probability. First the aggregator computes the aggregation result of data collected by sensor nodes. In the commit phase, the aggregator commits to the collected data, which ensures that the data collected were actually used. For the commitment, a Merkle hash-tree is used [15]. In this construction, the leaves hold the collected data and each internal node computes the hash value of the concatenation of the two child nodes. The root (aggregator) computes the final hash value which is the commitment. In the final phase, the aggregator sends the aggregated data and the commitment to the base station. The aggregator then proves to the base station that the data is correct using an interactive proof. To do that, the base station does the aggregation on some randomly chosen samples to check if aggregated result sent by the aggregator is good.

The proposals that we presented in this section are some first steps towards a general need to fortify sensor network security mechanisms with

self-awareness and self-healing so that they can autonomously recognize and respond to a security threat. This requires nodes to have knowledge about the network's state (or more realistically, the state of neighboring nodes) and monitor the network for abnormal behaviors of sensor nodes or data traffic. To characterize normal and malicious behavior, appropriate rules must be generated, based on statistics, induction and deduction. Once the network is aware that an intrusion has taken place and have detected the compromised area, appropriate actions must be taken. The first one is to cut off the intruder as much as possible and isolate the compromised nodes. After that, proper operation of the network must be restored. The challenge here is that all these steps must be taken autonomously, that is without human intervention.

## 6 Conclusions

In this chapter we have presented how a sensor network can secure its in-network processing against insider and outsider attacks. We elaborated on the general requirements that such a security protocol must satisfy and then described an efficient security mechanism on a general hierarchical aggregation/dissemination scheme. This mechanism provides a key management scheme for the establishment of secure channels between nodes and aggregators as well as support for addition of new nodes to the network, a critical requirement in sensor networks, as sensors have limited energy and thus limited life expectancy.

Performance evaluation shows that secure in-network processing can be achieved with very realistic memory and processing requirements. It does not depend on the existence of location information or on the underlying routing protocol. There is also no need to burden the base station with any additional computation load. Secure in-network processing can be realized as a distributed service that is scalable and adaptable.

The mechanisms presented here can be incorporated in a general, holistic approach that encompasses responses over a broad range of attacks. This is a research challenge that remains, where sensor networks are reinforced with an adaptive security architecture that can monitor the network, recognize a security threat and respond either by preventing the intruder or by isolating the damage and restoring the network's normal operation. In a dynamic communication environment of thousand of nodes, such a mechanism must not hinder other network processes but rather co-exists with them and

defend them.

## References

- [1] C.-Y. Chong, and S.P. Kumar, Sensor Networks: Evolution, Opportunities, and Challenges, in *Proceedings of IEEE* Vol.91 No.8 (2003) pp. 1247-1256.
- [2] A. Perrig, R. Szewczyk, V. Wen, D. Culler, and J.D. Tygar, SPINS: Security Protocols for Sensor Networks, in *Proceedings of 7th ACM Mobile Computing and Networks* (2001) pp. 189-199.
- [3] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks, in *Proceedings of 6th ACM/IEEE Mobicom Conference* (2000).
- [4] B. Krishnamachari, D. Estrin, and S. Wicker, The Impact of Data Aggregation in Wireless Sensor Networks, in *Proceeding of International Workshop on Distributed Event-Based Systems* (2002).
- [5] S. R. Madden, R. Szewczyk, M. J. Franklin, and D. Culler, Supporting Aggregate Queries Over Ad-Hoc Wireless Sensor Networks, in *Proceedings of Workshop on Mobile Computing Systems and Applications* (2002).
- [6] Y. Xu, Energy-aware Object Tracking Sensor Networks, in *Proceedings of International Conference on Distributed Computing System 2003 Doctoral Symposium* (2003).
- [7] D. Carman, P. Kruus, and B.J.Matt, Constraints and Approaches for Distributed Sensor Network Security, Tech. Rep. 00-010, NAI Labs, 2000.
- [8] T. Dimitriou, and I. Krontiris, A Localized, Distributed Protocol for Secure Information Exchange in Sensor Networks, in *Proceedings of the 5th IEEE International Workshop on Algorithms for Wireless, Mobile, Ad Hoc and Sensor Networks* (2005).
- [9] T. Dimitriou, and D. Foteinakis, Secure In-Network Processing in Sensor Networks, in *Proceedings of First Workshop on Broadband Advanced Sensor Networks (IEEE BASENETS)* (2004).

- [10] L. Hu, and D. Evans, Secure Aggregation for Wireless Networks, in *Proceedings of the Symposium on Applications and the Internet Workshops* (2003) p.384.
- [11] J. Deng, R. Han, and S. Mishra, Security support for in-network processing in Wireless Sensor Networks, in *Proceedings of the 1st ACM Workshop on Security of Ad Hoc and Sensor Networks* (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* (2003) pp. 62–72.
- [13] D. Wagner, Resilient Aggregation in Sensor Networks, in *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '04)*, (2004).
- [14] B. Przydatek, D. Song, and A. Perrig, SIA: Secure Information Aggregation in Sensor Networks, in *Proceedings of the ACM SenSys* (2003).
- [15] R. C. Merkle, Protocols for public key cryptosystems, in *Proceedings of the IEEE Symposium on Research in Security and Privacy* (1980).