

# Overview of Sensor Network Security

---

For T-106.5800 Seminar in software techniques

Toni Sallanmaa, 64631P

## Abstract

**Authors:** Toni Sallanmaa

**Name of Study:** Overview of Sensor Network Security

**Date:** 20.10.2009

**Pages:** 14

This paper introduces the current state of in-network security by describing five different problem areas and current solutions to them. We describe denial of service issues regarding secure network routing in sensor networks, a cryptographic protocol suite developed for sensor devices, a method for forming pair wise secure channels by key exchange and the problem of code dissemination in hostile environments.

It is found that all the problems have a solution tailor-made for sensor networks that has a working implementation. The solutions described all try to minimize extra costs involved, but all measures for security add some additional power consumption. Furthermore, the presented solutions do not provide perfect security and most fail if the number of compromised nodes in the sensor network grows too high.

The solutions presented are specific to their related problems and often disregard other elements, so a combination of security measures must be implemented for a high degree of security to be implemented.

**Keywords:** Sensor Network Security, Wireless Sensor Networks

## Table of contents

1 Introduction .....	4
1.1 Wireless Sensor Networks .....	4
1.2 Limitations .....	5
1.3 Organization.....	6
2 Overview of Problem Areas .....	7
2.1 Compromised nodes and routing .....	7
2.2 Security protocols .....	8
2.3 Key exchange .....	9
2.4 Routing attacks.....	9
2.5 Code dissemination .....	11
3 Conclusions .....	13

# 1 Introduction

Ubiquitous computing, as defined by Weiser in (Weiser, 1993), is the paradigm of computer science where computers are everywhere in the environment around the user, but their use is virtually invisible to the user and happens through everyday means. For the computers to be this pervasive in the environment, they need to be small and have very long battery lives to be useful. The computers could be in fixed locations connected with wires, but the more obvious case is to have the computers linked via wireless networks of some kind.

## 1.1 Wireless Sensor Networks

Wireless sensor networks (WSN) are not ubiquitous computing, but are a vital enabler for the paradigm. Wireless Sensor Networks provide the infrastructure for which we can start building services based on ubiquitous computing. Perrig et al. list in (Perrig et al., 2004) their vision on what the sensor networks are useful for and what the future could bring. Monitoring buildings, manufacturing equipment, earthquakes and the many military applications imaginable, for example battlefield surveillance, are among the list.

Wireless Sensor Networks are a network of computers (in this context called sensors) that interact with each other and take some kind of measurements of the surrounding environment. These measurements are then used by the controlling party in charge of the sensor network to solve some kind of a problem, for example alter air conditioning based on the current temperature of the space being monitored. The data being collected is either combined in special aggregation nodes in the network (or network base stations) or queried directly from the individual sensors. There is no one design of the network or how it operates. Figure 1 shows an example sensor network design based on sensors (marked A) and a base station (marked B) and another design based on aggregation nodes (marked C). In this design the aggregation nodes are connected to each other via a stronger link. Aspects from both of the designs can be combined when designing a network.

Sensor networks are different, but there are however building blocks for making them and not everything have to be made from scratch. There is an OS built for Sensor Networks, TinyOS and some libraries for it that perform some of the security functions, but unfortunately those were found inadequate by Ransom et al. (Ransom et al., 2008). The main problem is that security requirements vary from application to application and providing 'automated security' is not feasible.

Sensor network equipment may come with preprogrammed software on the device and have no means of updating the software. Some sensors have reprogrammable memory and their programs can be changed while in operation. These reprogramming requests are another point of attack to the system and can be used to disrupt operations by malicious parties (Hyun et al., 2008).

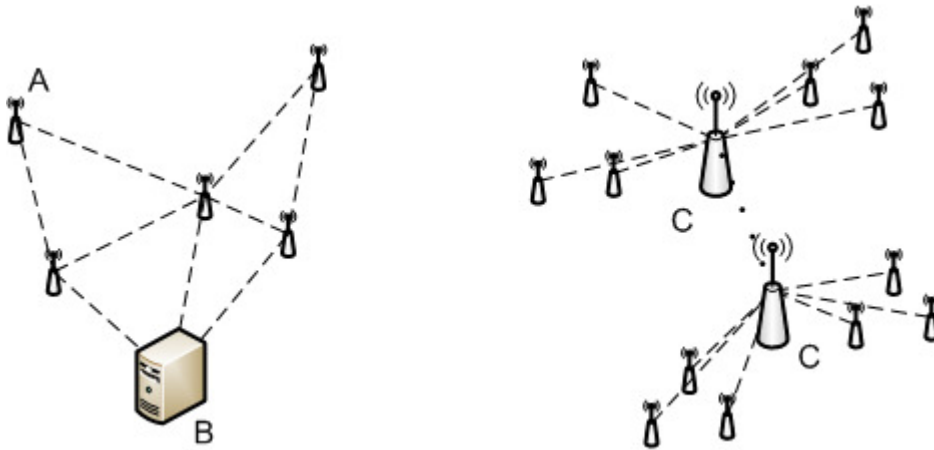


Figure 1. Two examples of wireless sensor network designs

One big issue with the sensors is the security of their communication. The data collected can be of confidential nature and should not be disclosed to any receiver near the network (Perrig et al., 2004). The way that networks operate is not homogeneous from network to network so the way security is implemented is not uniform either. There are a wide variety of possible attack paths for a wireless sensor network and the defense is usually a combination of cryptography for communications secrecy and conventional physical security to prevent tampering. The security to be implemented should be a part of the design at an early stage and not a separate module added at a later stage (Ransom et al., 2008). Physical security is not a focus in this paper and the sensors are assumed to be physically secured adequately.

Closely connected with the security of the sensor network communication are privacy issues regarding the network. The data being collected could be also of importance to the subject and not meant for public distribution (i.e. the medical details being collected by a medical sensor network). Strong security and good methods for validating security are necessary for keeping the information private. This paper only considers privacy as a motivator for providing security to avoid disclosure.

## 1.2 Limitations

The small size and low cost of the sensors in the sensor network add some additional restrictions of their own (Perrig et al., 2004). Sensors tend to have a low cost processing unit, which usually also means low computational power along with low power consumption. There is not much memory available and extra traffic to the outside world using the wireless sender should be minimized to maximize battery life.

More problems can arise from the sensors' physical location that may leave it vulnerable to physical attacks and possibly diminish wireless connectivity. Mostly, the amount of computational power available and the low storage space limit the amount of cryptographic methods that can be utilized. It is also noteworthy that the attacker can be assumed to have much more computational power at his disposal. The amount of memory limits the amount of cryptographic keys that can be stored and finally large computations (such as public key cryptography) use a lot of battery power.

Base stations, if used in the sensor network, are better equipped than the individual sensor nodes and usually are assumed to have much higher computational power and storage capacity. Usually the base station is assumed to be a part of the secure computation base, i.e. trustworthy at all times.

### **1.3 Organization**

The rest of this paper is structured as follows. The next chapter introduces the problem areas in the security of wireless sensor networks. Subchapters introduce different problem areas and discuss work done to resolve them. Then, in chapter 3, conclusions about the state of security for wireless sensor networks are presented.

## 2 Overview of Problem Areas

Feng et al. identify several design goals that should be taken into account when designing sensor networks in (Feng et al., 2002). Main goals are focusing optimization to problem areas, adhering to trends in field of technology well in advance and making a balanced assessment of tradeoffs to create the best product possible.

### 2.1 Compromised nodes and routing

Sensor networks are often distributed over a large area and data needs to be relayed from even the outermost sensors in the network to the base station. Sending the data straight to the base station is not often an option and routing must be employed. The routing should be as simple as possible to conserve node power. However, malicious nodes (compromised nodes) in the network are a real security threat and should not be able to break down the routing. This also extends to the area of fault tolerance where nodes are only malfunctioning and drop packets instead of being compromised. Even though there are multiple paths to the base station from a given node, even some of the more central nodes being compromised would lead to significant drop in the system throughput. (Lee & Choi, 2006)

Lee & Choi describe a system for making sure that the packets are actually routed forward that does not rely on multipath routing (Lee & Choi, 2006). Multipath routing routes the same packet through different paths with diffusion, but this multiplies the cost in power. The proposed system is Neighbor Watch System (NWS), where when the network is created each node knows its own neighbors and the neighbors' neighbors. There is an additional component that makes sure that the neighbor lists on the nodes are uncorrupted even when malicious nodes are present.

Neighbor Watch System is based on the principle of 'overhearing' packets. A node is able to hear everything the nodes surrounding it (neighbors) are sending out to the network. The system has shared keys, called cluster keys, for a node and all of its neighbors that are used to encrypt packets. When a node  $x$  sends a packet to node  $y$ , it keeps the sent packet in buffer. All nodes that are neighbors of both  $x$  and  $y$  also get a copy of the packet and are Sub-watch nodes. The Sub-watch nodes see if node  $y$  transmits the packet forward and if so verify that the packet is going to a real neighbor of  $y$ . If true, the packet is removed from buffers, but if the packet is not transmitted the Sub-watch nodes transmit the packet forward using some other node than  $y$ . Finally, the original node,  $x$ , watches the node sent to,  $y$ , and all Sub-watch nodes to verify that at least one of them sends the packet forward properly. Again, if it is not forwarded properly, node  $x$  resends it using some other node as the intermediary. This procedure is then repeated for every hop in the path to the destination to provide reliable delivery.

Neighbor Watch System is mostly single path forwarding scheme, where multipath is only needed when packet-dropping nodes are discovered. It is asserted by Lee & Choi that the storage requirements of the keys alongside neighbor lists should not pose a significant problem and can be compressed if the networks are really large. Additionally, it is mentioned that communication costs are lower with NWS than with

previously used systems. Simulation runs performed show that NWS performs well even with fairly large percentages of packet dropping nodes.

## 2.2 Security protocols

Sensor networks need security like any other application, but due to constraints described in section 1 there are severe limitations what is possible on a sensor node in terms of cryptography. Solutions include using asymmetric cryptography, which is not feasible for sensor computation. Perrig et al. tackle this problem in (Perrig et al., 2002) and present Security Protocols for Sensor Networks, SPINS for short, that contains protocol for authenticated broadcast specially developed for sensor network devices ( $\mu$ Tesla) and a low-overhead protocol for data confidentiality and freshness (SNEP).

SNEP is perceived as low-overhead, because it only adds 8 bytes of data to each protected packet. It is based on a counter mode cryptography, which means a value from a counter is used in determining the cryptographic key from a master key and which is increased after every transmitted or received message. Additionally, SNEP does not need to transmit counter values as counter state is being kept at both endpoints of the transfer. Message Authentication Codes (MAC) are also transmitted with each message and the codes rely on the counter value as well. MACs provide integrity and authentication, while the use of counter mode allows having data freshness and prevents message replay. Data replay is an attack consisting of repeated sending of a captured message by a malicious party.

$\mu$ Tesla is based on TESLA broadcast authentication but is altered heavily due to computation requirements of the original protocol.  $\mu$ Tesla uses only symmetric cryptographic primitives. Keys are formed in a key ring, where next key is formed by applying a cryptographic one-way function; authors present MD5 as a viable option, to the previous key. System is bootstrapped by the base station that computes the first key, then broadcasts packets encrypted with the key and finally after a set time interval broadcasts the key.  $\mu$ Tesla requires all of the clients to be loosely time synchronized, so they know that the key was not publicly disclosed when they received the first packet and can trust the packet came from sender. The same procedure cannot be applied to nodes sending packets as they cannot compute keys on their own, but Perrig et al. suggest that nodes use SNEP to transmit data to the base station, which in turn broadcasts it forward.

Perrig et al. have implemented the system on TinyOS-based sensor network devices. Their implementation shows that the cryptography routines occupied about 20% of the available code space on the devices, even when all algorithms were selected to minimize resource usage at the cost of some security. They also assert that this setup is able to encrypt or verify every message on the nodes while not being constrained by the processor. Energy costs are close to sending unencrypted data as only 8 bits are added per message for MAC code.

## 2.3 Key exchange

Di Pietro et al. focus on the problems with pair-wise secure communication in wireless sensor networks in (Di Pietro et al., 2003). Many applications important for the operation of WSNs, for example secure routing, require that a pair of nodes be able to securely exchange packets. To widen the scope of the applications, the nodes should be able to create a secure channel without a base station.

The straightforward answer to the problem is to have a key for each sensor pair, but this soon grows the size of the key pool on the sensors too big to be stored locally. Di Pietro et al. have devised a method that only requires a key pool, a set of keys, to be present on the devices before deployment. Two different methods are described: Direct method with fixed probability of security and an adaptive method where security can be traded for less overhead in the protocol.

The nodes get a fixed number of keys from a global key pool as the bootstrap for the system. The indexes of the keys given to a node are dependent on the node itself to make it possible for anyone to calculate which indexes of the key pool the node has. This obviously has the downside that at least one key has to be in common in two nodes if there is a secure channel between them. The direct protocol uses the logical exclusive or of all the shared keys between two nodes as the key for the secure channel. To corrupt the channel, malicious nodes need to know all of the keys in common.

The problem with the direct method is that the values are fixed and if the actual values, for example the number of hostile nodes, exceed the ones predicted when the system was designed then the security may be compromised. A co-operative protocol is described to combat this problem. For a node A to establish a secure channel with node B, it first chooses a set of co-operating nodes. These nodes compute the direct key with node b and then hash it with the identifier of node a. Node A calculates the exclusive or of these keys to form the final key. Then the node a sends the list of selected co-operating nodes to B encrypted with the direct key and now node b can use the same method to compute the new key by using the same co-operating nodes. The tradeoff between communication costs and security is done by varying the amount of nodes required for co-operation.

Simulated experiments on this system by Di Pietro et al. give encouraging results on how probable it is to corrupt a channel in the co-operative protocol. Also, it is shown that the results do not vary according to the network size making this algorithm scale extremely well.

## 2.4 Routing attacks

Karlof & Wagner describe issues with most approaches used to deploy secure routing in sensor networks in (Karlof & Wagner, 2003). Sensor networks often use a method called in-network processing where nodes process data that is travelling through them. This means nodes need to be able to access the data in the packets they are routing and this in turn makes using end-to-end security measures much harder to use with sensor networks. They assume that adversaries are able to listen to traffic and replay it and at least compromise a few nodes in the network.

Several classes of attacks against sensor networks are presented in the paper. The most direct way to attack a network is by replaying or modifying old packets or generating bogus packets to be sent to the network. This attack leads at least to degradation of network processing throughput as nodes as processing more packets than necessary and possibly acting on them. The next class described is selective forwarding, where a node only forwards some packets or no packets at all. This problem was described more fully in section 2.1.

Sinkhole attacks try to alter the network topology so that most of the traffic from a single network area is routed through a single node that is being controlled by the malicious party. This is done by making the node look extremely viable route for packets by knowing the routing algorithms being utilized. Sybil attacks mean that a node is posing as more than one node and sending out multiple identities. This can lead to problems with fault tolerance in distributed systems, where multiple copies of the data may not exist at all.

Hello flooding is an attack where the attacker floods the network with so called hello-packets. Most networks require the nodes to identify themselves to neighboring nodes by sending them a specialized packet often called a hello packet. Malicious parties are able to flood the network with these packets and cause confusion in the routing algorithms.

Wormhole attacks are the final method of attack presented. In these attacks, the adversary has a fast link that is not available for the network itself, for example a laptop. Using this link packets that are captured in one compromised node somewhere in the network are quickly moved to another node somewhere far away from the first node. This can be used to create sinkhole attacks or exploit race conditions if the system only acts on the first packet copy it receives.

Countermeasures to these attacks vary from attack by attack. Most attacks where the nodes or packets come from outside, outside attacks, can simply be blocked by utilizing network encryption. The outsiders do not have the network keys for the network and their packets are disregarded. Sinkhole attacks and selective forwarding are ineffective because the malicious party cannot join the network without correct cryptographic keys. However, hello flooding combined with wormhole attack is still very effective. Adversary moves the real hello packets from one end of the network to the other via wormhole and in this way causes neighbor finding algorithms to fail and the network ends up with major routing issues.

Karlof & Wagner however assert that against insider attacks better methods are needed. Sybil attacks can be prevented with a key protocol such as the one described in section 2.3. Hello flooding is countered by requiring the links to always be bidirectional. This means the nodes receiving a hello packet verify the identity of the other end of the link with an identity verification protocol. While a powerful adversary is able to route the packets both ways, the base station of the network can pick up the routed packets and disrupt the attack.

Wormhole and sinkhole attacks are described to be the hardest to defend against and no simple solution exists. Karlof & Wagner suggest designing new protocols for countering these problems rather than retrofitting existing protocols. One protocol

family that is resistant to these attacks are geographic routing protocols, because the routes are mostly computed with geographical coordinates and sinkholes are harder to implement in such networks.

## 2.5 Code dissemination

It is often necessary in wireless sensor networks to upgrade the system firmware of the individual sensor nodes. The process of distributing a new system image containing the new code of the system is called code dissemination. Huyn et al. discuss the problem of having a code dissemination system in a hostile environment in (Hyun et al., 2008). Threats against the system include malicious code images being distributed in the system or in the case of secure system draining the power from the devices by distributing a big number of images that the nodes have to verify.

Proposed system by Hyun et al. is an extension to the code dissemination system embedded within TinyOS. Their system, Seluge, has the property that each received packet has the data to verify it embedded in it so each packet can be discarded immediately upon receipt if it is forged. This is achieved through reorganization of data in the code dissemination packets. Packets are part of data pages, with one page having more than one packet and next data page being received only after the first has been successfully obtained. Additionally, Seluge uses another data page comprising of multiple packets not containing code data to distribute a hash tree for verifying the packets. Denial of service attacks aimed at exhausting the available space on the nodes receiving buffer while they are verifying packets in the buffer do not work with this scheme as the arriving packets can be discarded instantly.

The underlying system for code dissemination utilizes a method where a node broadcasts a packet to its neighbors advertising the code image version the node has and what pages of it has in memory. If a receiving node requires those pages it sends back a selective negative acknowledgement (SNACK) packet to get the pages. Both of these packets can be used to facilitate denial of service attacks. To combat this problem, Seluge establishes cluster keys for each node and each sent packet is encrypted with the cluster key. New nodes send hello packets to neighboring nodes and after establishing a pair wise secure channel they exchange cluster keys. After obtaining the cluster keys a node can check the integrity of arriving advertisement or SNACK packets.

Final problem solved is denial of service attack using the signature packets. By flooding the network with generated signature packets malicious party is able to exhaust the battery power of the nodes by making them verify the signatures continuously. Solution to this problem is to have a preinstalled keychain on each of the nodes. This chain generates more keys from a starting value with a hash function. Before sending any packets base station computes a solution to the puzzle and sends it along with the packet. Sensors can then verify that the puzzle uses a previously unused puzzle key from the keychain and only after that verify the signatures. If the puzzle check fails the packet is instantly dropped. Malicious parties wanting to distribute a packet first need to brute-force calculate the correct key, which takes a lot of time efficiently making the denial of service attack inoperable.

Simulations and practical implementation done by Hyun et al. shows that Seluge approximately doubles the code size of the implementation. Most of the extra code is used for a cryptographic library. The delay for the new code image to be propagated across the network does not increase much faster with Seluge compared to older approaches. Communication overhead is slightly higher than the base system, but not the worst compared to other solutions. Finally, it is asserted that Seluge allows for faster code dissemination rates on single nodes than other proposed solutions.

### 3 Conclusions

Wireless sensor network security has been described as a set of challenges, where conventional security approaches fail due to low amount of computational power, low amount of memory and need to preserve energy. However, there are lots of methods and technologies that are fit for use with sensors and that were designed with the limitations of the application environment in mind.

Routing was seen to be a major problem area in sensor networks with multitude of attacks already described that can either compromise the network or severely disrupt it with a denial of service attack. Using cryptography suited for sensor networks it is possible to limit the amount of disturbance caused by node compromise. A suite of security tools developed by Perrig et al. is a good starting point for developing the security measures for a sensor network, but as seen by the other examples, the application has to be designed and developed with security in mind.

There are downsides however. Security in sensor networks is not free and everything done to improve it incurs a cost in communication overheads, power consumption, data storage size and delays in propagating information due to computation involved. Devices itself will not bring an answer for this problem, since sensors are made to be cheap with inexpensive, low-consumption parts. Another problem with current cryptographic methods is reliance to pre-distributed keys for the devices. Methods used can be improved upon and that is where most of the improvements will stem from.

The methods presented here only attempt to rectify the related problem area. For a full security suite any one of these methods is not enough, but a combination of methods is necessary. Additional aspects discussed include the physical security of the sensors and especially of the trusted computing base.

## References

- [1] Di Pietro, R. Mancini, L.V. & Mei, A., Random Key-Assignment for Secure Wireless Sensor Networks, Proceedings of the 1st ACM Workshop Security of Ad-Hoc and Sensor Networks, ACM, 2003.
- [2] Lee, S. & Choi, Y., A Resilient Packet-Forwarding Scheme against Maliciously Packet-Dropping Nodes in Sensor Networks, Proceedings of The International Workshop on Scalable Ad Hoc and Sensor Networks, ACM, October 2006.
- [3] Yang, H. Ye, F. Yuan, Y. Lu, S. & Arbaugh, W. Toward Wireless Security in Wireless Sensor Networks, Proceedings of the 6th ACM international symposium on Mobile ad hoc networking and computing, ACM, 2005.
- [4] Ransom, S. Pfisterer, D. & Fischer, S., Comprehensible Security Synthesis for Wireless Sensor Networks, Proceedings of the 3rd international workshop on Middleware for sensor networks, ACM, 2008.
- [5] Feng, J. Koushanfar, F. & Potkonjak, M., System-Architectures for Sensor Networks Issues, Alternatives, and Directions, IEEE International Conference on Computer Design (ICCD'02), IEEE Computer Society, 2002.
- [6] Perrig, A. Stankovic, J. & Wagner, D., Security in Wireless Sensor Networks, Communications of the ACM, ACM, 2004.
- [7] Perrig, A. Szewczyk, R. Tygar, J.D. Wen, & Culler, D.E., SPINS: Security Protocols for Sensor Networks Wireless networks, Springer Netherlands, 2002
- [8] Karlof, C. & Wagner, D., Secure routing in wireless sensor networks: attacks and countermeasures, Ad Hoc Networks, Volume 1, Elsevier, September 2003.
- [9] Weiser, M., Some computer science issues in ubiquitous computing, Communications of the ACM 36, ACM, 1993
- [10] Huyn, S. Ning, P. Liu, A. & Du, W., Seluge: Secure and DoS-Resistant Code Dissemination in Wireless Sensor Networks, In Proceedings of the 2008 International Conference on Information Processing in Sensor Networks, IEEE Computer Society, 2008.