Comprehensible Security Synthesis for Wireless Sensor Networks

Stefan Ransom, Dennis Pfisterer, and Stefan Fischer
Institute of Telematics
University of Lübeck, Germany
{ransom|pfisterer|fischer}@itm.uni-luebeck.de

ABSTRACT

Providing usable security mechanisms for Wireless Sensor Networks (WSNs) is one of the important tasks to foster WSN development, as they will increasingly be deployed in real-world settings in the future. Generally, due to its complexity, expert knowledge in the field of security is needed to plan, implement, and deploy a sound security setup. Frequently, no or only basic security mechanisms leave the WSN open to attacks of which the application designer might not be aware, especially if no security evaluation has been done. Furthermore, available security mechanisms have a great impact on the design of the application itself, i.e. to properly secure a protocol. We therefore propose a framework that first devises feasible security setups based on the characteristics of the envisioned application and second provides a security evaluation of theses possibilities from which the application designer is able to choose the fitting setup for his application. We have prototypically implemented the proposed scheme into the WSN middleware synthesis tool FABRIC and thus support the application designer in including a sound security solution into his application.

Categories and Subject Descriptors

C.2 [**Computer Systems Organization**]: Computer-Communication Networks; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.2.11 [**Software**]: Software Architectures

General Terms

Design, Security

Keywords

Security assessment, middleware, sensor networks

1. INTRODUCTION

Integrating security is vitally important if WSNs are to be deployed in real-world applications. Especially in application scenarios where private or confidential data is gathered, processed, and communicated, the need for security becomes instantly evident. Also, applications depending on data correctness benefit from integrated security mechanisms.

It is inevitable that all discussions about security must have the target application in mind. Specifically, only the application designer knows which data needs to be secured with which kind of security service. Furthermore, he knows about the scope of these security measures, e.g. whether the whole communication needs to be secured or only a certain data type when it is communicated. Current research seconds our application-centric view of security provision for WSNs [19, 22].

By all means, a security setup for an application must always be subject to a thorough security evaluation in order to justify its security promises and to foster the application developer's awareness regarding which aspects are secure and which are at risk, thus avoiding a false sense of security.

Generally, although a lot of the envisioned application scenarios today would benefit from integrated security mechanisms, few actually concern themselves with security aspects as the technological nature of WSNs already provides a challenging environment for application developers. Since large numbers of small, severely resource-constraint devices form WSNs and have to be programmed, deployed and maintained, even well studied concepts like routing, or time-synchronization pose again challenging problems. Therefore, although security is often a desired feature, building a working system generally takes precedence over integrating security-related aspects. Additionally, to the best of our knowledge, the integration of security into WSN applications is typically not supported by existing WSN middlewares leaving the developers to devise, evaluate, and implement their own security solutions. This is also backed by a current WSN middleware survey [23].

In this paper we present a framework which allows the integration of comprehensible security into the development process of WSN applications. In detail, we

- automatically devise application-specific security mechanisms,
- provide a security evaluation of the possible security setups, and
- make these steps usable by integrating them into a middleware synthesis tool.

The remainder of this paper is organized as follows. In the following Section we provide more details and discuss related

MidSens'08, December 1-5, 2008, Leuven, Belgium

work regarding our three focus areas. Subsequently, we introduce our concept of automatically deriving viable security setups from the characteristics of the envisioned application and present our approach to evaluate their security. We illustrate the whole approach by giving a use-case in Section 4, and finally describe in Section 5 how these schemes are implemented in the WSN middleware synthesis tool FABRIC. The paper concludes with a summary and directions for future work.

2. MOTIVATION AND RELATED WORK

We believe that in order to be widely used, security must be embedded into WSN development tools. However, as mentioned above, the integration of security into WSN applications is typically not supported by most WSN middlewares. Notable exceptions to this are the TinySec [10] and MiniSec [13] libraries for the widespread WSN operating system TinyOS [12]. They provide some security building blocks, e.g. to encrypt and authenticate data packets. However, in order to be used they assume that necessary foundations like key setup are handled elsewhere. Furthermore, their usage is limited to systems running TinyOS.

We go a step further and focus on a more encompassing system. It automatically determines necessary security requirements for the target application and is extensible in order to support different hardware platforms as well as different programming languages.

A problem for providing security automatically at the middleware level is that the requirements for security vary greatly between applications, mainly based on their data semantics, i.e. the meaning of data to the application. For instance, the security requirements of structural monitoring applications differ significantly from those of health-care applications. In structural monitoring, sensor data should be transmitted using mechanisms that guarantee integrity and authenticity while confidentiality is often not required. However, communicating sensor data confidentially is mandatory in medical applications. Furthermore, due to the scarce resources in sensor nodes, always including all-encompassing security measures might prove counter-productive. Take for example, the measures to prevent traffic analysis in [5]. Clearly, for a large group of WSNs traffic analysis is not a great security concern and consequently these networks need not waste resources on the nodes with measures that prevent it. Therefore, a "one size fits all"-approach is not feasible for providing security in WSNs.

Another reason for application-specific security arises from the fact, that several application design parameters have an impact on the required security mechanisms, although they concern at heart non-security related aspects of the application. For example, the communication pattern between nodes has an impact on the pre-deployment of key material, e.g. which nodes need to share a key and which nodes do not. The application designer has this knowledge and is therefore able to provide the parameters needed to automatically devise possible security setups as we will show in Section 3.1. We are not aware of any other structured approach to facilitate the application designer in the process of generating application-specific security setups.

Yet, the question arises how to choose the proper security setup from all feasible solutions. Discussing and evaluating the quality of different security schemes requires significant domain expertise in the area of security. Although several attempts try to establish metrics that measure and describe the achieved level of security of a system, no all-encompassing framework exists, that is a able to cover all the diverse aspects and points of view. In particular, the metric must be appropriate for the evaluated technology and the results of the evaluation must be understandable for the target audience. Thus, while a false-positive ratio may be a good measure to judge the quality of an intrusion detection component it is not suitable to express the quality of security measures in WSNs. Additionally, a representation of the security evaluation by a mere number, i.e. setup 1 rates 0.8 whereas setup 2 rates 0.97, gives no hint how these two setups differ and which one might be more appropriate.

The upcoming ISO/IEC 27004 standard [9] as well as the recommendations in the NIST special publication 800-55 [3] try to measure the security of a system mostly along a process-oriented approach, i.e. do backup strategies exist and how complete is an implementation. In this, they are suitable to measure system security of an organization. However, with this broad perspective effectively measuring security in WSNs is not satisfyingly possible with these approaches, since important technical details are not covered.

A more technical approach to measure the efficiency of security measures is based on Security Patterns [21]. This model-based approach provides reusable solutions to specific security problems which application designers may use to build secure systems. Since the patterns are widely evaluated in real systems, the security evaluation of the resulting system becomes much easier [8]. Although several patterns might be helpful in establishing and evaluating security in WSNs they were not widely considered with this focus and consequently do not cover relevant aspects yet. Particularly, retaining security with the possibility of compromised nodes is not included [16].

Typically, due to the lack of formal approaches, authors of specific WSN security measures evaluate their approaches by means of attack possibilities with regard to a chosen attacker profile. However, attacker profiles are often not specified in great detail and evaluations differ with respect to the considered attacks. We also found two attempts to provide attack taxonomies for WSNs [7, 18] though they do not seem to be used widely.

3. PROVIDING SECURITY

We have illustrated the main problems when providing security at a middleware-level and have shown that only limited support is available for the application designer in these areas. We now present our approach to handle all three aspects.

3.1 Application-Specific Security

To automatically derive feasible and fitting security mechanisms, the application designer first has to specify the desired abstract security service for each data type, e.g. confidentiality for location data. This defines how this data type will be handled security-wise when communicated in the network during operation. Furthermore, the scope is required, e.g. which nodes must be able to participate in the communication of this data type. Additionally, we require a few other parameters to derive possible security setups. Table 1 gives an excerpt of available security-related parameters as well as example configurations.

From these specifications our framework on the one hand

Parameter	Description	Configuration
Data Types		
Data type security	Desired security service for this data type and scope	Confidentiality / all nodes
Application Properties		
Memory	Available memory for secret keys	200 Byte
Nodes	Total number of sensor nodes	200
Lifetime	Network lifetime	1 year
TP-Hardware	Tamper-proof hardware	No
Attacker Profile		
Capabilities	Potential attacks	Wireless interaction, node compromise
Scope	Operating range of the attacker	Global

Table 1: Examples of the security-related parameters

derives possible key setups, which we have described in more detail in [17], and on the other hand integrates the security mechanisms needed to fullfill the configurations.

Often several different security setups can be synthesized, each with its specific strengths and weaknesses. Furthermore, the choice of a setup often has consequences for the application logic itself. For example, if the application designer chooses a setup that is based on one network-wide key, he will not be able to use end-to-end encryption between the basestation and a particular node based on the selected scheme.

So, to support the application designer in his choice, we deem it important to present him a qualitative security evaluation for each possible setup rather than merely presenting all possible solutions or choosing one in a black box fashion.

3.2 Evaluating WSN Security

For the security evaluation of our generated security setup we use a formal and structured approach to model threats to the system, similar to Attack Trees presented by Schneier in [20]. First, all attacks on a system are arranged in one or several tree structures, with the goals modelled as roots and possible ways to attack the goals modelled as leaves. Then each attack path can be evaluated with respect to its likelihood, the necessary effort for the attacker, or other qualitative measures. Finally, specific techniques can be devised and implemented in order to prevent likely attack paths.

Generally, attacks can be classified into insider and outsider attacks. Characteristic for insider attacks is that the attacker controls a valid member of the network whereas outsider attacks are always possible, even without control of a valid member. In our evaluation we are mainly concerned with outsider attacks. This is a reasonable choice, as we evaluate the initial security setup for the WSN and there must not be any successful attacks during pre-deployment. Otherwise the whole security setup falls apart.

Three important attack goals for outsider attacks exist in each WSN: the nodes, the communication, and the environment. Attacks on nodes target valid members of the WSN either by destroying, controlling, or influencing their data or behaviour. Attacks on the communication target the communication as a whole, and focus either on disturbing it or eavesdropping on it. Finally, attacks on the environment target the WSN as a whole by influencing its operation area, e.g. by triggering wrong sensor readings. On this basis we model three attack trees, c.f. Figure 1; the numbers denote the possible attack paths.

However, we do not include all possible attack paths in our evaluation since no proper defenses exist against some attacks. For example, consider DOS attacks. Some research exists that aims at lessening the effects of these attacks, e.g. [24]. However, a complete defense against a sufficiently provisioned attacker is not feasible. Therefore, DOS attacks are for now excluded from the further analysis. Additionally, we do not include attack paths that can be avoided by best-practice programming guidelines, such as to disable the nodes' interfaces and over-the-air programming (OTAP) capabilities during operation.

Table 2 gives an overview of the different attack paths, including the fundamental attacking efforts and goals of the attack. The attack paths we consider in our further evaluation are highlighted, namely the paths 3, 5, 6, and 11. We now focus on real attack types along these paths and possible defences. Regarding processing power and communication we consider an attacker with laptop capabilities. His goal is to become an insider using the outsider attack paths discussed above. Once an insider, he may influence the WSN directly or stage more sophisticated attacks, such as Sybil attacks [14].

Becoming an insider can be achieved in three ways. First, if the communication is not secure the attacker instantly learns the communicated data and influencing the nodes at will is trivial via message injection and modification. Furthermore, attacker nodes may be added to the WSN. In case of a secured communication valid messages can be replayed to influence or at least to confuse nodes. Additionally, the attacker may try to break the employed cryptography, which can only succeed if the key size is too small. We assume the actual algorithm to be secure and implemented properly. Finally, he may attack the nodes physically in order to compromise them, thus again gaining access to the WSN. Ideally, he analyses the traffic beforehand in order to isolate important nodes in the network, such as data sinks or nodes close by. Table 3 gives an overview of the considered attacks and defences.

We use this evaluation logic to automatically assess the derived security setups with regard to their quality. This is done first by matching the security annotations specified by the application designer with the attack possibilities to find paths in the attack tree that are secure. In a second step we evaluate the efforts and risks of the insecure paths to determine the possible effects of the attacks. Remember that we always consider an attacker with laptop-grade processing and communication capabilities. To complete the attacker profile, we include the attackers scope, e.g. local or global interaction or influence, and his specific capabilities, such as the tampering of nodes from the security annotations. For this purpose we assume that communicating wirelessly with a node requires significantly less effort than finding and accessing it. Furthermore, Becher et al. [2] describe that compromising a node usually requires expert knowledge, costly equipment and other resources, including the removal of nodes from the network for a longer time-span. Therefore, we denote the effort to compromise a node as being high. The risks are mainly affected by the chosen key

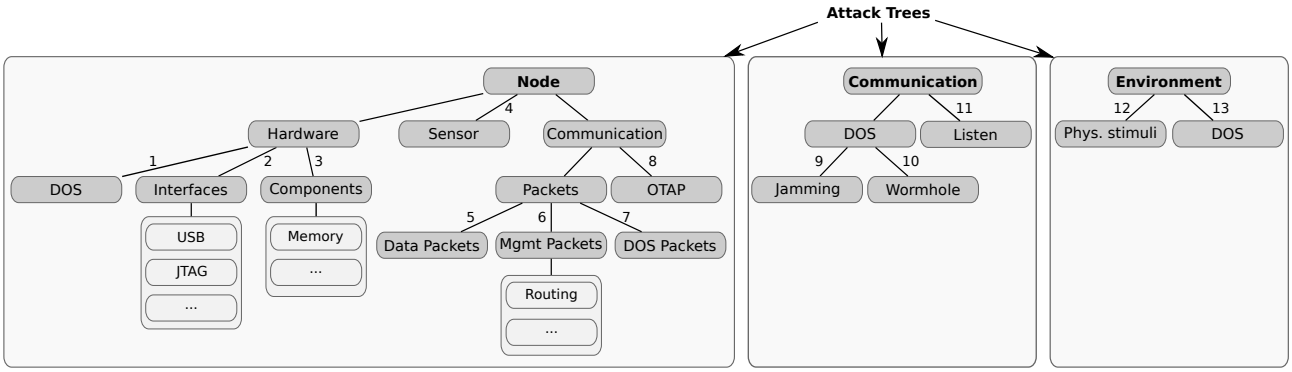

Figure 1: WSN Attack Trees (numbers denote the different attack paths)

Path	Effort	Attack
Attacks on Nodes		
1	F&A	Destroy the node's hardware
2	F&A	Compromise the node by accessing it through live interfaces
3	F&A	Compromise the node by accessing its hardware directly
4	F&A	Influence the node's sensor readings
5	AN	Influence the node's information
6	AN	Influence the node's behaviour
7	AN	Sleep deprivation torture DOS with malformed packets
8	AN	Reprogram node via OTAP
Attacks on the Communication		
9	CR	Jam communication medium
10	CR	Redirect traffic via wormhole
11	CR	Eavesdrop on the communication
Attacks on the Environment		
12	S	Influence WSN sensor readings
13	S	Destroy WSN operation area

Table 2: WSN Attack Paths (F&A: Find&Access node; AN: Addressable Node; CR: Attacker's Communication Range; S: Attacker's Scope)

setup as well as the attacker's scope.

Finally, in order to facilitate the application designer in choosing a setup, we present these quality assessments in form of Residual Risk (RR) tables. These tables display for each possible security setup attack paths of high and medium risk, as well as paths that are secure. Furthermore, they give an indication of how much effort is needed for this particular attack. Examples of RR tables for the use-case we describe in the following Section are given in Figure 2. The application designer is now able to either choose one proposed security setup or he may discard the choices and go back to modify his specifications.

4. USE-CASE

To further illustrate the framework described above and to evaluate its usefulness, we now look at the whole process by means of a use-case. We consider a structural monitoring application, similar to the one investigated in [22]. In this scenario we have two data types: routing data and sensor data. Both must be transmitted with authenticity and integrity guarantees, while confidentiality is not required.

Attack	Target	Defence
Message Insertion	Nodes	Authenticity
Message Modification	Nodes	Integrity
Message Replay	Nodes	Message uniqueness
Physical Attacks	Nodes	Tamper-proof hardware
Eavesdropping	Communication	Confidentiality
Traffic Analysis	Communication	Confidentiality (whole packet), measures from [5]
Cryptanalysis	Cryptography	Sufficient key size

Table 3: Considered external Attacks and Defences

Furthermore, routing data must be accessible to all nodes, while sensor data is only transmitted in an end-to-end fashion from each node to the basestation.

Here, we assume 200 employed nodes which are not tamper-proof and each having 160 Bytes of memory available for key material. Furthermore, we envision a long network lifetime. We further assume that an attacker is able to mount communication as well as physical attacks, though only locally. For the sake of brevity, we limit ourselves to these characteristics in this example.

Based on these specifications our framework generates and evaluates possible security setups that are able to satisfy the given characteristics. In the following we discuss two of these choices.

Both setups provide the required services based on the keys given in the following table. While setup 1 provides a strong network-wide key to secure routing data, setup 2 needs to reduce the key size in order to retain security despite 10 compromised nodes using the key setup scheme proposed in [6]. To secure the sensor data both setups rely on pairwise keys between each node and the basestation.

Setup	Routing Data	Sensor Data
1	128 bit network-wide key	128 bit pairwise keys
2	100 bit 10-secure pairwise keys	128 bit pairwise keys

The residual risk tables for both setups are given in Figure 2. They illustrate respectively the risks associated with each option. The application designer can accept one of the

Attack	Effort	Risk
High Risks		
Eavesdropping	small	The content of your data and routing messages is readable.
Message Replay	small	Valid messages can be replayed.
Compromise node	high	1 node required to access to all routing messages.
Medium Risks		
Traffic Analysis	small	An attacker can analyse the traffic, though only in his local coverage area.
Not at Risk		
Message correctness	small	Messages are authentic and can not be modified undetected.
Cryptography	medium	Average time to break a 128 bit key: >58 thousand years [10^7 keys/sec].

(a) Security Setup 1

Attack	Effort	Risk
High Risks		
Eavesdropping	small	The content of your data and routing messages is readable.
Message Replay	small	Valid messages can be replayed
Medium Risks		
Traffic Analysis	small	An attacker can analyse the traffic, though only in his local coverage area.
Cryptography	medium	Average time to break a 100 bit key: 3.57 years (128 bit: >58 thousand years) [10^7 keys/sec].
Compromise node	high	10 nodes required to access all routing messages.
Not at Risk		
Message correctness	small	Messages are authentic and can not be modified undetected.

(b) Security Setup 2

Figure 2: Residual Risk Tables

options if all the attacks he thinks likely are handled or he can discard both and specify further security properties in the security annotations. In case both options are sufficient he has in this particular example to decide whether it's more likely that an attacker will try to compromise nodes or try to break the cryptography.

5. MIDDLEWARE INTEGRATION

Application development for WSNs with only very scarce resources is complicated and error-prone. Hiding these intricate aspects from the developer is therefore beneficial in order to reduce the overall design complexity of applications. Consequently, making our proposed scheme available to the application designer through easily understandable means is imperative for a widespread adoption. Generally speaking, our scheme can be integrated with systems synthesizing middleware systems or virtual machines as well as complete applications, such as RUNES [4] or ATG [1]. As a first step, we have implemented support for our middleware synthesis framework FABRIC [15] that supports WSN application development by generating custom-tailored middleware instances for different target platforms.

The underlying idea is that data of certain semantics is of a particular, usually complex type. To assign this meaning to the individual data types, an application developer annotates data type definitions with treatment *aspects*. A code generation processor passes both, type definitions and annotation aspects to so called *modules* implemented by domain experts that provide the actual functionality. These modules generate source code for the annotations they are in charge of, e.g. reliable messaging or key setup.

For the use-case discussed above, the application developer defines routing and sensor data structures and attaches the desired annotations to the individual data type definitions. Hereby, we distinguish *domains* that embrace related aspects (e.g., authenticity from the domain security). This concept is realized using the well-known XML Schema standard that is complemented with annotations which are XML documents incorporated into each data type definition. Figure 3 presents an excerpt of how this would look like in our use-case. For such a given annotated type definition, FABRIC picks the *best* modules based on each module's self-

Figure 3: Excerpt from FABRIC's Configuration File

description. A code generation processor reads the schema, and passes both type definitions and annotation attributes to a set of selected modules. These then generate source code for the functionality they are in charge of, e.g. a send routine for sensordata that integrates the generation of an HMAC [11] to provide authenticity and integrity .

To alleviate the process of annotating security related aspects for each data type we have implemented our scheme as a plug-in in for the well-known Eclipse framework where a user can enter the parameters as discussed in Section 3 using a simple graphical user interface (GUI). Next, the plug-in generates and evaluates the possible security setups and presents the Residual Risk tables to the application developer who can then select the optimal one for his scenario. The plug-in then automatically reflects these values in the annotations for the individual data types and invokes FABRIC. This allows the application developer to implement the application using the generated middleware, including key material and security mechanisms.

A first evaluation shows that the generated security mechanisms' code size is equal to that of a corresponding manual implementation. Thus, no additional code overhead is generated by our implementation in FABRIC.

6. CONCLUSIONS AND OUTLOOK

We have presented a framework to provide easy-to-use and understandable sound security for WSNs at middleware-level to support application designers. The framework supports the implementation of any kind of security mechanisms by security experts that afterwards can be used by non-experts.

We have integrated the framework in the middleware synthesis tool FABRIC. Furthermore, to provide the application designer with a developer-friendly interface to work with this system we have developed an Eclipse plug-in that implements our proposed scheme.

The next interesting step to improving our framework is to include measures against insider and DOS attacks, which might then be integrated into the application by simple annotations as well.

Likewise, security patterns present an interesting approach to provide security for non-experts. In the future we therefore plan to investigate if existing patterns are promising for utilization in WSNs and what would be good WSN-specific patterns.

7. REFERENCES

[1] A. Bakshi, V. K. Prasanna, J. Reich, and D. Larner. The abstract task graph: A methodology for architecture-independent programming of networked sensor systems. In *Proc. of the 2005 Workshop on End-to-end, sense-and-respond systems, applications and services*, 2005.

[2] A. Becher, Z. Benenson, and M. Dornseif. Tampering with Motes: Real-World Physical Attacks on Wireless Sensor Networks. In *Proc. of the 3rd Int. Conference on Security in Pervasive Computing*, 2006.

[3] E. Chew, M. Swanson, K. Stine, N. Bartol, A. Brown, and W. Robinson. Performance Measurement Guide for Information Security. NIST Special Publication 800-55 (Rev.1), July 2008.

[4] P. Costa, G. Coulson, R. Gold, M. Lad, C. Mascolo, L. Mottola, G. Picco, T. Sivaharan, Weerasinghe, and S. N., Zachariadis. The runes middleware for networked embedded systems and its application in a disaster management scenario. In *Proc. of the 5th Annual IEEE Int. Conference on Pervasive Computing and Comunications*, 2007.

[5] J. Deng, R. Han, and S. Mishra. Countermeasures Against Traffic Analysis Attacks in Wireless Sensor Networks. In *Proc. of the 1st International Conference on Security and Privacy for Emerging Areas in Communications Networks*, 2005.

[6] W. Du, J. Deng, Y. S. Han, and P. K. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks. In *Proc. of the 10th ACM Conference on Computer and Communications Security*, 2003.

[7] S. Han, E. Chang, L. Gao, and T. Dillon. Taxonomy of Attacks on Wireless Sensor Networks. In *Proc. of the First European Conference on Computer Network Defence*, 2005.

[8] T. Heyman, R. Scandariato, C. Huygens, and W. Joosen. Using security patterns to combine security metrics. In *Proc. of the Int. Workshop on Secure Software Engineering*, 2008.

[9] International Organization for Standardization (ISO). Information technology – Security techniques – Information security management measurements. ISO/IEC 27004. Standard under Development.

[10] C. Karlof, N. Sastry, and D. Wagner. TinySec: a link layer security architecture for wireless sensor networks. In *Proc. of the 2nd Int. Conference on Embedded Networked Sensor Systems*, 2004.

[11] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-Hashing for Message Authentication. RFC 2104, Feb. 1997.

[12] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. *Ambient Intelligence*, chapter TinyOS: An Operating System for Sensor Networks, pages 115–148. Springer, Berlin, 2005.

[13] M. Luk, G. Mezzour, A. Perrig, and V. Gligor. MiniSec: a secure sensor network communication architecture. In *Proc. of the 6th Int. Conference on Information Processing in Sensor Networks*, 2007.

[14] J. Newsome, E. Shi, D. Song, and A. Perrig. The Sybil Attack in Sensor Networks: Analysis & Defenses. In *Proc. of the 3rd Int. Symposium on Information Processing in Sensor Networks* , 2004.

[15] D. Pfisterer, C. Buschmann, H. Hellbrück, and S. Fischer. Data-type centric middleware synthesis for wireless sensor network application development. In *Proc. of the Fifth Annual Mediterranean Ad Hoc Networking Workshop*, 2006.

[16] E. Platon and Y. Sei. Security software engineering in wireless sensor networks. *Progress in Informatics*, 5:49–64, Mar. 2008.

[17] S. Ransom, D. Pfisterer, and S. Fischer. Making Security Useable in Wireless Sensor Networks. In *Proc. of the Int. Workshop on Sensor Network Engineering*, 2008.

[18] T. Roosta, S. P. Shieh, and S. Sastry. Taxonomy of Security Attacks in Sensor Networks and Countermeasures. In *Proc. of the 1st Int. Conference on System Integration and Reliability Improvements* , 2006.

[19] E. Sabbah, A. Majeed, K.-D. Kang, K. Liu, and N. Abu-Ghazaleh. An Application-Driven Perspective on Wireless Sensor Network Security. *Proc. of the 2nd ACM Int. Workshop on Quality of Service & Security for Wireless and Mobile Networks* , 2006.

[20] B. Schneier. Attack Trees: Modeling Security Threats. *Dr. Dobb's Journal*, Dec. 1999.

[21] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad. *Security Patterns: Integrating Security and Systems Engineering.* Jon Wiley & Sons, 2006.

[22] F. Stajano, D. Cvrcek, and M. Lewis. Steel, Cast Iron and Concrete: Security Engineering for Real World Wireless Sensor Networks. In *Proc. of the 6th Int. Conference on Applied Cryptography and Network Security*, 2008.

[23] M.-M. Wang, J.-N. Cao, J. Li, and S. K. Das. Middleware for Wireless Sensor Networks: A Survey. *Journal of Computer Science and Technology*, 23(3):305–326, 2008.

[24] A. D. Wood and J. A. Stankovic. Denial of service in sensor networks. *IEEE Computer*, 35(10):54–62, 2002.