A Security Architecture and Modular Intrusion Detection System for WSNs

Nils Aschenbruck*•, Jan Bauer°, Jakob Bieling°, Alexander Bothe°, and Matthias Schwamborn*
*University of Osnabrück - Institute of Computer Science
Albrechtstr. 28, 49069 Osnabrück, Germany
{aschenbruck, schwamborn}@informatik.uos.de
°University of Bonn - Institute of Computer Science 4
•Fraunhofer FKIE
Friedrich-Ebert-Allee 144, 53113 Bonn, Germany
{bauer, bieling, bothea}@cs.uni-bonn.de

***Abstract*—Wireless Sensor Networks (WSNs) are deployed in a wide range of application scenarios. These typically involve monitoring or surveillance of animals or humans, infrastructure, or territories. Since security as well as privacy play an increasingly important role in these contexts, sensor nodes and sensor networks need to be protected from spurious environmental effects and malicious attacks. In addition to attacks known from conventional wireless networks, the specific properties of WSNs lead to new kinds of attack. Moreover, countermeasures are subject to strict resource limitations of the motes and, therefore, have to be light-weight and effective at the same time. In this paper, we first present a comprehensive security architecture for WSNs, consisting of different attack types (including WSN-specific attacks) and countermeasures. Second, we propose a modular Intrusion Detection System (IDS) as a framework for this architecture. Finally, we give details on selected modules and discuss practical implementation issues.**

***Index Terms*—Security; Intrusion detection; Wireless sensor networks**

I. Introduction

A Wireless Sensor Network (WSN) consists of small and highly resource-constrained sensor nodes (*motes*) that monitor changes of some measurable phenomenon, e.g., light, temperature, or movement. In order to deliver the measured sensor data to one or more sinks, these motes autonomously form a wireless multi-hop network. In-network processing is commonly applied to aggregate data and minimize traffic on the way to a sink. In addition to this *convergecast* communication pattern, WSNs also use a one-to-many reverse channel for control purposes, including Over-The-Air Programming (OTAP). WSNs are deployed in a steadily growing plethora of application areas. These range from military (e.g., security perimeter surveillance) over civilian (e.g., disaster area monitoring) to industrial (e.g., industrial process control).

Application scenarios of WSNs typically involve monitoring or surveillance of animals or humans, infrastructure, or territories. Since security as well as privacy play an increasingly important role in these contexts, sensor nodes and sensor networks need to be protected from spurious environmental effects and malicious attacks. For example, motes in a disaster relief scenario might sense seismic activity data which is crucial to arrive in time and unmodified. Medical data belonging to certain patients is additionally sensitive in terms of privacy and should, therefore, be protected against eavesdropping.

Over the last two decades, much work has been done identifying and detecting potential threats to Mobile Ad hoc NETworks (MANETs). However, the specific properties of WSNs lead to new kinds of attack as well as new approaches to countermeasures: Attackers may physically access motes and take complete control by re-programming them. Countermeasures are subject to strict resource limitations of the motes and, in general, cannot be used as is from the MANET context. Therefore, design and implementation of countermeasures tailored to these specific requirements are necessary.

Some work has already been done in the area of intrusion detection in WSNs, mostly with simulative evaluations. Although simulation is an invaluable tool for testing and evaluating new security approaches, simulation models cannot properly reflect the conditions of a WSN in a real deployment. Therefore, experimental evaluation with real motes is preferred especially by the WSN community.

Our contribution is two-fold: (1) We present a comprehensive security architecture for WSNs which addresses several layers. Different attack types are discussed and for each attack type, a countermeasure approach is given. WSN-specific attacks are also included in this architecture. (2) We propose a modular Intrusion Detection System (IDS) as a framework for this architecture and discuss practical implementation issues.

The rest of this paper is organized as follows: We first discuss related work in Section II. A threat analysis leading to our security architecture is presented in Section III. The proposed IDS along with a discussion of practical implementation issues are described in Section IV. Finally, we conclude the paper and give directions for future work in Section V.

II. Related Work

Some work on intrusion detection for WSNs can be found in the literature, which we briefly discuss in the following.

978-1-4673-1786-3/12/$31.00 ©2012 IEEE

Onat and Miri [14] propose a statistic-based intrusion detection scheme. They assume a static WSN, in which the neighborhood of a node is more or less fixed. Data about the neighborhood is gathered and analyzed to detect anomalies. Attacks that can be detected with this scheme are node impersonation and resource depletion. Moreover, the proposed scheme is evaluated through simulation.

Da Silva et al. [5] present a rule-based IDS. It comprises three phases: During the data acquisition phase, monitor nodes filter and collect messages in promiscuous mode. During the following rule application phase, these messages are run through a pre-defined set of generic rules. If a message fails a test in this phase, a failure is raised. Finally, the intrusion detection phase is used to compare the number of failures to some threshold and if the former exceeds the latter, an intrusion is assumed. For the evaluation of their scheme, the authors also implemented a new simulator.

Krontiris et al. [10] propose a collaboration-based intrusion detection system and adapt it to detect sinkhole attacks. Each node analyzes packets in promiscuous mode based on user-defined rules. If there is a deviation from normal behavior, an anomaly is detected. To finally decide whether a node is indeed an intruder, a cooperative mechanism among the neighborhood is triggered and a mutual conclusion is made. In order to react to an intrusion, a response mechanism is also part of the IDS. As an example, the authors present rules for and experimentally evaluate the detection of sinkhole attacks against the MintRoute routing protocol [20].

Rajasegarar et al. [16] survey different anomaly detection techniques. A classification of these techniques can be made based on what type of background knowledge of the underlying data is available. If there is no prior knowledge about the data, unsupervised learning or clustering can be used. Supervised learning requires data training sets, which include data labeled as normal or abnormal. If characteristics of normal and abnormal data change, the classifier has to be retrained. Semi-supervised learning involves an initial learning of a generalized data set. The classifier can detect anomalies and dynamically adapt to system changes as new data becomes available. Another way to categorize anomaly detection techniques is to distinguish between the model type. Statistical model techniques assume that the distribution of the data analyzed for anomalies is known a priori. Non-parametric model techniques, on the other hand, make no assumption about the data distribution beforehand.

III. Threat Analysis

The goal of a threat analysis is to identify threats in terms of attack types in the context of the considered network. Due to the specific properties of WSNs, new kinds of attack as well as new challenges for countermeasure development arise: In contrast to nodes of a MANET, it is quite likely that an attacker gains direct physical access to a mote since there are various WSN applications where the motes are deployed within a freely accessible area [15]. Furthermore, end-to-end encryption of sensor data is infeasible if in-network processing is applied since aggregator nodes need to have access to the sensor values [9]. A general challenge arises from the resource constraints, especially from limited processing power and memory. This requires countermeasures to be light-weight but also effective at the same time.

There are two important criteria for categorizing different attack types (cf. [9][1]): One criterion is differentiating between outsider attacks and insider attacks. An *outsider attack* is an attack where the attacker has no direct access to the existing network. Instead, the attacker can only access the wireless channel. An *insider attack*, on the other hand, is an attack where the attacker has compromised legitimate motes or has obtained key material or similar data from a legitimate mote in order to gain illegitimate access to the network.

The other criterion is concerned with the hardware resources the attacker can dispose of. A *mote-class* attacker uses mote hardware for attacking the WSN, i.e., the attacker's resources are as constrained as the resources of legitimate motes. A *laptop-class* attacker uses laptop-like hardware with much more resources at his disposal. This leads to a significant advantage since certain attacks have a higher impact with more resources.

Figure 1 shows a threat analysis as well as countermeasures for WSNs. We first divide the threat analysis into the goals an attacker might have (top layer "Goal"):

Manipulate Data: Depending on the WSN application scenario, data manipulation can have devastating effects. Two examples are patient monitoring and industrial automization. This goal can mainly be achieved at the routing layer by faking link qualities and, thus, attracting a lot of traffic. In the context of WSNs, it is also feasible that an attacker moves a mote away from its originally assigned position. Since in most WSN application scenarios, motes sense data from their surroundings, this data is dependent on the position. If the attacker moves a mote, the sensor data from the new surroundings is misleadingly assumed to be from the original position's surroundings. The "Manipulate Data" goal corresponds to the security requirement *integrity*.

Eavesdrop Data: Different advantages for the attacker result from eavesdropping data. On one hand, the attacker can sniff potentially confidential data. On the other hand, it enables the attacker to perform statistical traffic analysis for deducing which motes pose the most worthwhile targets (also known as "homing" [18], [21]). Furthermore, packet structure analysis yields hints about protocol usage on, e.g., Media Access Control (MAC) or routing layer. With this knowledge, the attacker can launch more efficient attacks which are tailored to specific protocols. A threat especially present in WSNs is dumping of the contents of a mote's ROM and/or RAM. Since the mote's memory might contain key information for en- and decryption of network communication, the attacker can gain access to the WSN. The "Eavesdrop Data" goal corresponds to the security requirement *confidentiality*.

[1]Note that even though [9] is somewhat dated, it still applies due to its fundamental nature.

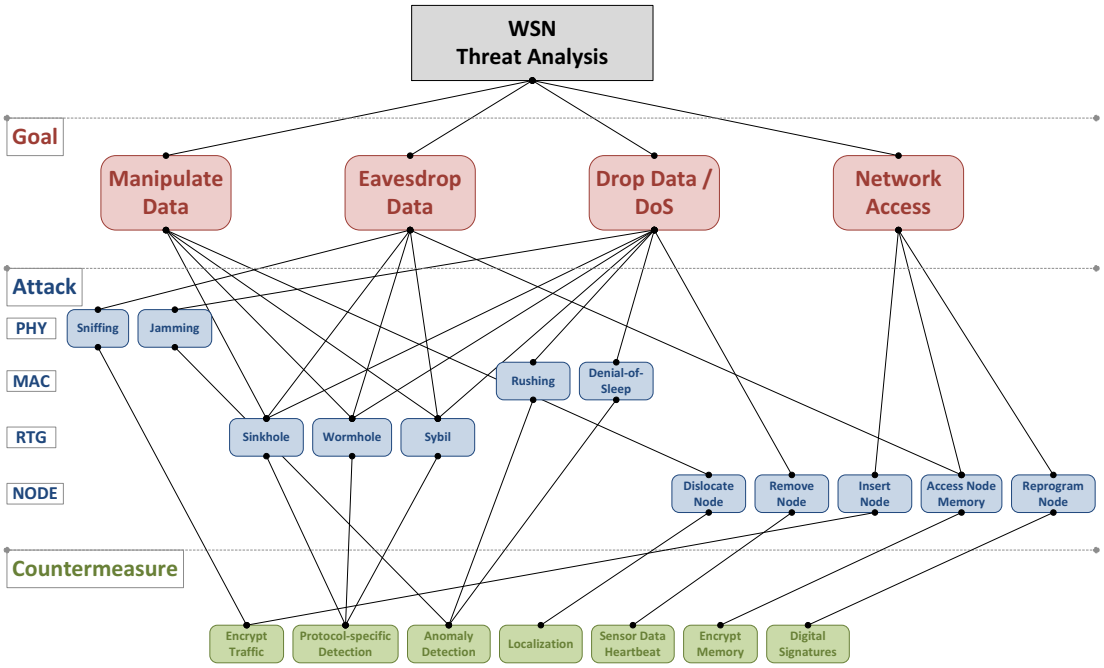


Fig. 1. WSN threat analysis and countermeasures.

Drop Data: A further threat to WSNs is dropping data which can be considered as Denial of Service (DoS) in a broader sense. A simple and effective attack is to jam the wireless channel. Nodes trying to send under this attack are forced to drop their packets at some time. Reception of data packets is also affected by the interference. Another implementation of this goal consists of dropping packets that were supposed to be forwarded due to multi-hop communication. Freely accessible motes might also be completely removed from the network by the attacker. The "Drop Data" goal corresponds to the security requirement *availability*.

Network Access: Since most outsider attacks can be prevented quite easily by encryption and authentification [9], attackers are interested in getting access to the network. This can be achieved by adding a mote which pretends to be a legitimate member of the network. A more complex way is to obtain key information from a legitimate mote's memory or even to compromise a legitimate mote. The latter can, e.g., be realized through malicious usage of OTAP.

The four described goals are not mutually exclusive and may be combined. In order to launch an insider DoS attack, network access must be achieved first. We connect each of the goals to the attacks (middle layer "Attack") that may be used to reach that specific goal. Since an attack may realize multiple goals, we group the attacks by the layer on which they are conducted (sub-layers "PHY", "MAC", "RTG", and "NODE" in Figure 1). In the following, we will describe the different attacks as well as possible countermeasures (bottom layer "Countermeasure"). Note that the term "countermeasure" is meant as a generic term for preventive, detectional, and reactional measures. Preventive measures are generally preferred but these are not always feasible. Therefore, in these cases, detectional and reactional measures have to be applied. With regard to the IDS, we focus on detection.

PHY Layer: On the physical (PHY) layer, sniffing and jamming can be used to attack the network. *Sniffing* means eavesdropping on information passing through the shared and freely accessible wireless channel. Since sniffing works on the physical layer, the attacker has access to packet data of all higher layers. Moreover, sniffing is a completely passive attack, i.e., no modification of data is involved. It also enables traffic analysis, which works even if communication is encrypted since knowledge of packet contents is not necessary (cf., e.g., [6]). Sniffing can be performed as an outsider as well as an insider attack and even though a mote might be required for capturing the data, further processing is commonly performed on laptop-class hardware.

A common preventive countermeasure against sniffing is the encryption of the whole network communication. This does not, however, prevent the attacker from performing a traffic analysis, as has been explained above. The encryption can be symmetric or asymmetric, mainly differing in encryption strength and key distribution complexity.

Jamming is a classic outsider attack with the aim to jam the wireless channel with an interfering signal. This DoS attack on the physical layer requires the jammer to either have a strong signal or to be in the proximity of the target nodes. Depending on the required range and duration of a jamming attack, mote-class or laptop-class hardware is used. There are various ways to realize a jammer, differing in strategy, efficiency, and detection complexity. Jammer classes can be

divided into constant, deceiving, randomized, and reactive (for details, see [22]).

A countermeasure against jamming is anomaly-based detection which can be implemented on different layers. On the MAC layer, e.g., an anomaly in the Carrier Sense Time (CST) can be detected. The CST is the time needed during a transmission attempt to access the shared medium. On application and routing layer, the Packet Delivery Ratio (PDR) may be used as an indicator. On the physical layer, the Received Signal Strength Indicator (RSSI) can be used as a metric. Indicators on multiple layers may also be combined (cross-layer approach) to increase the quality of the detection.

MAC Layer: On the MAC layer, rushing and denial-of-sleep attacks may be conducted. Many MAC protocols work on a cooperative basis. It is assumed that all nodes adhere to the protocol's rules in order to achieve a certain fairness concerning medium access. This trust in node cooperation is betrayed with the *rushing* attack by ignoring specific rules and thereby sending faster than legitimate nodes [21]. Mostly Carrier Sense Multiple Access (CSMA)-based MAC protocols are affected by this since backoff timers can be simply ignored. A rushing attacker can capture the wireless medium since other nodes adhere to the protocol and backoff accordingly. Since legitimate nodes are prevented from transmitting, rushing belongs to the class of DoS attacks. As preparation for a routing attack (e.g., sinkhole), rushing can be used to send fake routing information faster. Rushing is generally performed as an insider attack with mote-class hardware.

As a countermeasure against rushing, anomalies in the CST can be detected similar to jamming detection. As with jamming, legitimate nodes have an unconventionally high CST since the rusher prevents them from getting access to the medium. However, since rushing is a MAC layer attack, the attacker can be narrowed down to the direct (1-hop) neighborhood of a node that has detected the anomaly.

Since energy efficiency is one of the most important design factors in WSNs, the energy resource of motes poses a worthwhile target for an attacker. Most energy of a mote is consumed by wireless communication [1]. The MAC protocol mostly controls the sleep phases of the radio chip. Therefore, this layer is the primary target layer for an attack on energy resources. A special type of DoS attack is the *denial-of-sleep* (also known as "battery exhaustion" or "sleep deprivation") attack [17], [18] which exploits certain mechanisms to keep a mote from turning its radio chip off. Intelligent denial-of-sleep attacks can reduce the network lifetime of a WSN from several months to a few days. This requires knowledge about the MAC protocol in use by the target WSN which, however, can be gained quite easily through traffic analysis [18]. Although not limited to insider attacks, denial-of-sleep is less effective as an outsider attack. Laptop-class attackers can also achieve a higher impact than mote-class attackers with denial-of-sleep.

An easy but effective countermeasure against denial-of-sleep attacks is the detection of anomalies in energy consumption. Since denial-of-sleep induces an unconventionally high energy consumption, it can be distinguished from regular consumption. Note that anomaly detection includes statistical quantification of "regular" energy consumption.

RTG Layer: On the routing (RTG) layer, sinkhole, wormhole, and sybil attacks can be launched to manipulate, eavesdrop, or drop data. A *sinkhole* attack attracts as much traffic as possible such that data is routed through the attacking node [9]. This enables the attacker to freely decide on what to do with the packets. A successful sinkhole attack generally requires the attacking node to make itself attractive with respect to the routing metric (e.g., Estimated Transmission Count (ETX)). If the attacker propagates a fake optimal metric value, neighboring nodes will prefer the (fake) optimal link over (legitimate) sub-optimal links. Due to the commonly used convergecast communication pattern in WSNs, a sinkhole attack is most efficient if launched near a root node. A sinkhole is commonly launched as an insider attack and can be created with mote-class hardware. However, a laptop-class attacker can also utilize his higher transmission range to propagate a 1-hop connection to a root node, although he is farther away than legitimate motes. When a sinkhole has been established, packets can be dropped selectively or completely, which is known as a greyhole (or "selective forwarding") or blackhole attack, respectively.

Sinkhole attacks are generally hard to counter [9]. This is mainly due to the fact that routing metric advertisements, such as end-to-end delay or path ETX to a root node, are hard to verify. Therefore, sinkhole countermeasures are generally protocol-specific and/or anomaly-based.

A *wormhole* attack is a special type of replay attack. Using a fast out-of-band connection only available to the attacker, packets received at one end of the wormhole link are transmitted to the other end of the link and replayed there [9]. The out-of-band connection may be, e.g., wired with a high data rate, making legitimate nodes at the other end believe that the original sender is only a few hops away. Although it can be performed with one node that simply forwards packets, it is generally assumed that the attacker uses two nodes connected via the out-of-band link. Since the attacker does not necessarily need to know a packet's contents, a wormhole can be performed as an outsider attack. Both mote-class and laptop-class hardware are feasible for creating a wormhole.

Since the out-of-band connection is only known and visible to the attacker, a wormhole is hard to prevent or even to detect [9]. As with the sinkhole attack, countermeasures against wormholes need to be adjusted to the deployed routing protocol

A *sybil* attack is launched when the attacker assumes multiple identities with one node [9]. This may be an advantage for the attacker if a multi-path routing protocol is used which maintains multiple paths to a receiver. The protocol (mistakenly) assumes that paths with disjunct nodes are used, while the attacker actually combines multiple node identities within one node. In this case, a sybil attack can be used for preparing a sinkhole attack. Furthermore, sybil attacks are a security threat to localization protocols.

Generally, authentication mechanisms for verifying identities can prevent sybil attacks. However, the main threat of this attack lies in the preparation of a sinkhole attack. Therefore, ultimately, countermeasures against sinkholes can be applied here.

NODE Layer: Independent of a specific protocol layer, sensor nodes (motes) can also be attacked physically (NODE). The *dislocate node* attack is easy to perform as an outsider if the motes are freely accessible and movable. By dislocating a node, the attacker can manipulate measurement data. Moreover, dislocating fixed anchor nodes needed for some localization techniques can significantly impair localization performance. Node dislocation can be countered by appropriate localization techniques in case of a static WSN. In case of mobile WSNs, however, unauthorized dislocation can generally not be distinguished from authorized mobility.

A sensor node can also be completely removed from the network. The *remove node* outsider attack can be realized by destroying a node or moving it out of range of the network. This results in a loss of general network performance and is, therefore, considered as a DoS attack. A simple but effective countermeasure is a centralized node failure detection based on periodic status messages (heartbeats). If a heartbeat is missing, node failure is assumed. If the WSN application already includes periodic messages of nodes, there is no overhead induced by this detection technique.

One way to gain access to the network as an attacker is to add a node of mote- or laptop-class. When the attacker has successfully gained access with the *insert node* attack, insider attacks can be launched. Since this attack requires the node to understand the communication of the legitimate nodes, encryption (e.g., AES) can be used as a countermeasure.

An attacker can also *access a node's memory* to gain network access. A mote can be connected to laptop-class hardware, thereby obtaining critical data like key material. To counter this attack, memory can be encrypted. However, due to the resource restrictions, only symmetric encryption is feasible. An open challenge in this context is to store the corresponding encryption key on the mote in a secure manner.

The *reprogram node* attack exploits the reprogramming capabilities of a mote. Running code can be replaced with arbitrary malicious code by the attacker. There are basically two ways for an attacker to overwrite a mote's running code: Physical access to the mote enables wired reprogramming, while OTAP can be exploited to reprogram a mote unauthorized and wirelessly. The former is an outsider laptop-class attack, while the latter usually requires network access. In general, it is not possible to distinguish a legitimate node from a compromised one. Therefore, preventive countermeasures have to be taken. A simple but self-restricting countermeasure is the complete omission of reprogramming capabilities. This is only feasible in application scenarios where reprogramming is not necessary. Otherwise, OTAP can be augmented by digital signatures for the code images. This prevents unauthorized reprogramming and can be realized with symmetric as well as asymmetric cryptography.

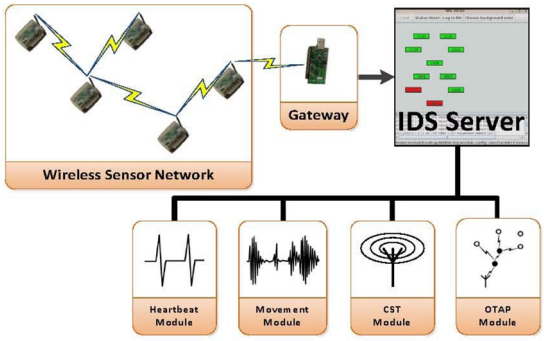


Fig. 2. Basic IDS architecture.

IV. Intrusion Detection System

In order to properly evaluate the security architecture, we implemented a modular IDS, where a module corresponds to one of the countermeasures. As the main purpose, the IDS enables the WSN administrator to monitor the network status. In particular, alarms are triggered by the detection modules if an attack has been detected. These alarm messages as well as status messages (collectively called "IDS messages") are processed and visualized by the server application for immediate intrusion response.

A. Architecture

The basic architecture of the IDS is shown in Figure 2. Local detection modules on the motes send IDS messages via wireless radio to the gateway, which forwards these messages via serial line to the central IDS server. The server processes and visualizes messages according to the type.

Detection modules are integrated in a mote's application and share a common IDS messaging framework. Local detection modules are mainly divided into modules with periodic and event-based messaging. An example for a module with *periodic messaging* is the heartbeat module, where periodic status messages indicate that the mote is still alive. *Event-based* messaging is used by modules which send alarm messages triggered by the detection of an attack. The movement module, e.g., sends an alarm if it detects that the mote is being moved. In order to distinguish different modules and message types, each IDS message type is assigned a unique IDS-ID. An overview of the message types is given in Table I.

IDS messages are sent by the motes to the IDS gateway. This gateway is responsible for reducing the received message to the essential parts for the IDS server, i.e., IDS-ID, originator address, and (optional) payload. Based on the originator address, the server can map and display the alarm/status accordingly. The compressed message is send via serial line to the server.

B. Server

The IDS server is implemented as a modular Java application that processes and visualizes incoming messages. Each server module processes messages belonging to a certain detection module on the motes. Incoming messages are,

TABLE I
IDS MESSAGE TYPES OF THE IMPLEMENTED MODULES.

Module	IDS-ID	Payload	Information Content
Heartbeat	*	*arbitrary*	vital sign (*: arbitrary (IDS) message)
Movement	0x01	movement sensor type	movement detection
CST	0x02	alarm status, CST sample, threshold	jamming alarm status change
OTAP	0x03	image #	failed signature verification
	0x04	counter	initialization of an OTAP process with invalid counter

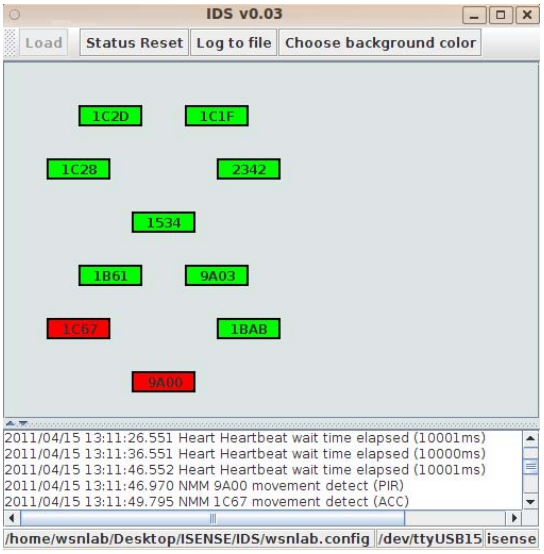


Fig. 3. Graphical User Interface (GUI) of the IDS server.

therefore, forwarded to their respective processing module according to the IDS-ID.

As shown in Figure 3, the server also provides a GUI. The lower part displays timestamped alarm and status messages. This log can also be written to an external file for later analysis. The upper part shows the monitored network with topology and node addresses configured by the user. Alarm messages are visualized immediately upon module notification through color change of the originating node (from green to red). Furthermore, separately for each node, the user can define which detection modules are in use.

C. Modules

We implemented four modules for the IDS so far: Heartbeat, movement, CST, and OTAP. Each will be described in the following.

Heartbeat Module: The heartbeat module is used to detect node failure or node disconnection in general. In particular, the remove node attack can be detected with this module. For each monitored node (as defined by the user), a status message is expected to arrive before the global heartbeat timer runs out. If a status message has not been received from a node within this time interval, disconnection or failure (possibly caused by a remove node attack) of that specific node is assumed. The module triggers an alarm to the server accordingly.

The delay of detecting a potential remove node attack is directly dependent on the heartbeat interval length t. Since the module maintains a global timer for all nodes, the detection delay δ is at least as long as the interval and at most as long as two intervals, i.e., $t < \delta < 2 \cdot t$. Longer intervals induce less heartbeat status messages but increase detection delay. Likewise, shorter intervals induce more messages but decrease detection delay. Therefore, a trade-off between message overhead and detection delay has to be made when choosing a value for t. However, if the mote application already includes periodic messages, no extra status messages are necessary since the only requirement is that messages are sent periodically (cf. Table I).

Movement Module: The movement module is an event-based messaging module with a fixed IDS-ID and can be used to detect dislocate node attacks in static WSNs. Incoming alarm messages with this IDS-ID include the sensor type of the originating mote that triggered the alarm (cf. Table I). This might be an accelerometer or a passive infrared sensor on the mote. A user-defined threshold value determines the sensitivity of these sensors. If it is set too low, even subtle environmental effects might trigger an alarm (false positive). Likewise, if it is set too high, a cautious dislocate node attack might be successful (false negative).

CST Module: The CST module is an anomaly-based detection module which sends an alarm to the server if a certain threshold for the CST has been exceeded. As mentioned earlier, rushing and jamming attacks can be detected with this scheme. Experience in the context of MANETs has also shown that the CST is a qualified indicator especially for jamming detection [2]. We adopted the threshold-based detection approach proposed in [2] which dynamically adapts to the network's traffic load. In order to distinguish regular traffic from anomalies, an aging function is used to locally compute the anomaly threshold t_n:

$$t_n := (1 - \alpha) \cdot t_o + \alpha \cdot s,$$

where t_o is the old threshold, s the current CST sample value, and α the aging factor. The aging factor is a parameter which basically defines the importance of new samples for the new threshold.

An alarm is triggered if the current sample s exceeds the old threshold t_o at least by the weighted standard deviation σ over all CST samples measured so far:

$$alarm := \begin{cases} 1, & \text{if } s > t_o + w \cdot \sigma \\ 0, & \text{else.} \end{cases}$$

Here, the tolerance range based on the standard deviation compensates regular fluctuations of the CST. Each change in the alarm status triggers an IDS message transmission addressed to the server based on best effort. This message includes the alarm status, current sample, and threshold (cf. Table I).

OTAP Module: The OTAP module essentially enables the prevention of unauthorized reprogramming attempts, i.e., it prevents the reprogram node attack. It verifies the digital signature associated with a new code image and its origin. If the verification fails, it denies any further processing of the code image and sends an alarm message to the server. However, since there was no appropriate implementation of an OTAP protocol available which includes digital signatures for secure reprogramming, we implemented our own protocol.

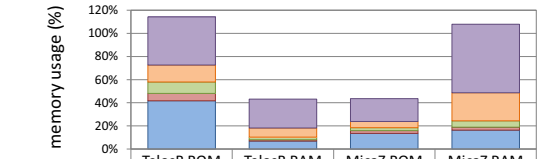
Similar to the light-weight approach of NWProg [19], our implementation of a secure OTAP protocol is based on the well-known OTAP protocol Deluge [8]. In order to make it light-weight, we replaced the complex dissemination algorithm with simple propagation. We augmented the protocol with digital signatures based on elliptic curve cryptography, ensuring the integrity and authentication of disseminated code images. Thus, the OTAP module has the ability to detect manipulated and malicious images, prevents their installation, and reports their existence by sending alarm messages with a specific IDS-ID to the IDS server (cf. message type `0x03` in Table I).

Using asymmetric cryptography, our OTAP approach is secure against access node memory attacks since only the public key is stored on each mote. The only instance disposing of the private key used to sign new code images is the legitimate OTAP initiator. Furthermore, we extended code images by a version counter in order to protect the OTAP process against replay attacks (cf. message type `0x04` in Table I).

D. Implementation

We have implemented the four modules described in Section IV-C for different WSN operating systems, namely TinyOS [7] and iSense [4], supporting multiple mote platforms, i.e., TelosB [13], MicaZ [12], and iSense-CM10C [3]. Depending on the security requirements of the application scenario and the sensor types provided by each platform, these modules can be integrated into the basic application of the WSN.

In our implementation, we assume a light-weight basic application with added Advanced Encryption Standard (AES) encryption as general protection against insider attacks. The application periodically gathers and transmits sensor data to a dedicated sink. Leveraging these transmissions as heartbeat messages, our heartbeat module causes neither extra status message overhead nor additional memory usage. Regarding the limited memory resources of current mote platforms, this is advantageous since the basic application itself already requires a significant part of the available memory (cf. Figure 4). In this figure, the percentaged memory usage (ROM and RAM) required by the IDS modules is visualized in combination with



	TelosB ROM	TelosB RAM	MicaZ ROM	MicaZ RAM
OTAP module	41.63%	25.05%	19.73%	59.28%
CST serial line	14.61%	7.86%	5.25%	24.19%
CST module	9.90%	2.25%	2.61%	5.47%
AES encryption	6.36%	1.05%	2.41%	2.51%
Application incl. Heartbeat mod.	41.71%	7.00%	13.54%	16.43%

Fig. 4. Memory usage of different IDS modules implemented in TinyOS, depending on the hardware platform (TelosB vs. MicaZ)

our basic application for platforms TelosB (48 kB ROM, 10 kB RAM) and MicaZ (128 kB ROM, 4 kB RAM) running TinyOS.

The CST module enabling local jamming detection is integrated into the application's Low Power Listening (LPL) mechanism. Additionally, this module can be extended by an optional serial line add-on ("CST serial line" in Figure 4) which offers an alternative channel for reporting alarms to the IDS server.

In the OTAP module, we use the TinyECC [11] library for the cryptographic operations. Due to the memory requirements of this library and the inherent memory requirements of the boot loader, the memory usage of the OTAP module is very high compared to both modules described above.

Since the TelosB mote offers neither acceleration nor passive infrared sensors, we implemented a prototype of the movement module for the iSense-CM10C platform which runs the iSense operation system. On this platform, the movement module requires 5668 B of program memory (ROM). 60 968 B of ROM are necessary for a comparable basic application including the heartbeat module.

As can be inferred from Figure 4, due to strict memory constraints, running all implemented IDS modules at the same time is not possible on the considered platforms. Therefore, the WSN administrator has to choose an appropriate security level (by selecting a subset of the modules) for a specific application scenario and needs to make a trade-off between security and the functionality of the WSN application.

V. Conclusion

We presented a comprehensive security architecture for WSNs, identifying different well-known as well as WSN-specific attacks on various layers. For each of these attacks, a possible countermeasure was also given. Furthermore, we proposed a modular IDS as a framework for this architecture. Detection modules run on the motes and detect an attack locally. In case of a detection, an alarm message is sent to the IDS server where the corresponding server-side module processes the message and the alarm is displayed in a GUI for immediate response. Selected modules have been implemented and code sizes have shown that a trade-off between security and application functionality has to be made due to strict memory restrictions.

In the future, we want to improve and add more modules to the IDS framework. We are also planning to evaluate the IDS in our WSN testbed in the context of various static and mobile network scenarios.

ACKNOWLEDGMENTS

This work was supported in part by the German Federal Office for Information Security (BSI). Furthermore, this work was supported in part by CONET, the Cooperating Objects Network of Excellence, funded by the European Commission under FP7 with contract number FP7-2007-2-224053. The authors would like to thank the reviewers for their constructive criticism and, also, the WSNLab and CONET project teams for feedback, sustainable discussion, and work.

REFERENCES

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless Sensor Networks: A Survey," *Computer Networks*, vol. 38, no. 4, pp. 393–422, 2002.

[2] N. Aschenbruck, E. Gerhards-Padilla, and P. Martini, "Simulative Evaluation of Adaptive Jamming Detection in Wireless Multi-hop Networks," in *Proc. of the 30th Int. Conference on Distributed Computing Systems Workshops (ICDCSW '10)*, Genoa, Italy, 2010, pp. 213–220.

[3] coalesenses GmbH, "Core Module CM10C, CM10S - Preliminary Data Sheet," 2011. [Online]. Available: http://www.coalesenses.com/download/CM10X_DS_1v1.pdf

[4] ——, "iSense: Operating and Networking Firmware," 2011. [Online]. Available: http://www.coalesenses.com/index.php?page=software-system

[5] A. da Silva, M. Martins, B. Rocha, A. Loureiro, L. Ruiz, and H. Wong, "Decentralized Intrusion Detection in Wireless Sensor Networks," in *Proc. of the 1st Int. Workshop on Quality of Service & Security in Wireless and Mobile Networks (Q2SWinet '05)*, Montreal, Quebec, Canada, 2005, pp. 16–23.

[6] J. Deng, R. Han, and S. Mishra, "Countermeasures Against Traffic Analysis Attacks in Wireless Sensor Networks," in *Proc. of the 1st Int. Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm '05)*, Athens, Greece, 2005, pp. 113–126.

[7] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister, "System Architecture Directions for Networked Sensors," *SIGPLAN Not.*, vol. 35, pp. 93–104, November 2000.

[8] J. Hui and D. Culler, "The Dynamic Behavior of a Data Dissemination Protocol for Network Programming at Scale," in *Proc. of the 2nd Int. Conference on Embedded Networked Sensor Systems (SenSys '04)*, Baltimore, MD, USA, 2004, pp. 81–94.

[9] C. Karlof and D. Wagner, "Secure Routing in Wireless Sensor Networks: Attacks and Countermeasures," *Ad Hoc Networks*, vol. 1, no. 2-3, pp. 293–315, 2003.

[10] I. Krontiris, T. Dimitriou, T. Giannetsos, and M. Mpasoukos, "Intrusion detection of sinkhole attacks in wireless sensor networks," in *Proc. of the 3rd Int. Conference on Algorithmic Aspects of Wireless Sensor Networks (ALGOSENSORS '07)*, Wroclaw, Poland, 2008, pp. 150–161.

[11] A. Liu and P. Ning, "TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 2.0)," 2011. [Online]. Available: http://discovery.csc.ncsu.edu/software/TinyECC

[12] MEMSIC, "MicaZ Data Sheet," 2011. [Online]. Available: http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=148

[13] ——, "TelosB Data Sheet," 2011. [Online]. Available: http://www.memsic.com/support/documentation/wireless-sensor-networks/category/7-datasheets.html?download=152

[14] I. Onat and A. Miri, "An Intrusion Detection System for Wireless Sensor Networks," in *Proc. of the Int. Conference on Wireless and Mobile Computing, Networking and Communications (WiMob '05)*, Montreal, Quebec, Canada, 2005, pp. 253–259.

[15] A. Perrig, J. Stankovic, and D. Wagner, "Security in Wireless Sensor Networks," *Commun. ACM*, vol. 47, pp. 53–57, 2004.

[16] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Anomaly Detection in Wireless Sensor Networks," *IEEE Wireless Communications*, vol. 15, no. 4, pp. 34–40, 2008.

[17] D. Raymond, R. Marchany, M. Brownfield, and S. Midkiff, "Effects of Denial-of-Sleep Attacks on Wireless Sensor Network MAC Protocols," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 1, pp. 367–380, 2009.

[18] D. Raymond and S. Midkiff, "Denial-of-Service in Wireless Sensor Networks: Attacks and Defenses," *IEEE Pervasive Computing*, vol. 7, no. 1, pp. 74–81, 2008.

[19] TinyOS Community, "BLIP Tutorial - TinyOS Documentation Wiki," 2011. [Online]. Available: http://docs.tinyos.net/tinywiki/index.php/BLIP_Tutorial#Network_Programming

[20] A. Woo, T. Tong, and D. Culler, "Taming the Underlying Challenges of Reliable Multihop Routing in Sensor Networks," in *Proc. of the 1st Int. Conference on Embedded Networked Sensor Systems (SenSys '03)*, Los Angeles, CA, USA, 2003, pp. 14–27.

[21] A. Wood and J. Stankovic, "Denial of Service in Sensor Networks," *Computer*, vol. 35, no. 10, pp. 54–62, 2002.

[22] W. Xu, W. Trappe, Y. Zhang, and T. Wood, "The Feasibility of Launching and Detecting Jamming Attacks in Wireless Networks," in *Proc. of the 6th Int. Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc '05)*, Urbana-Champaign, IL, USA, 2005, pp. 46–57.