Master’s thesis

Security in Wireless Sensor Networks for Open Controller

by

Christoffer Engvall

LIU-ID/A/LITH-EX-A-13/014—SE

Tutor: Anna Vapen, Ulf Kargén

Examiner: Nahid Shahmehri
Abstract

In this thesis we develop, evaluate and implement a security solution for Open Controllers wireless sensor network platform. A scenario is used to describe an exemplar application showing how our system is supposed to function. The security of the platform is analyzed using a well-established threat modeling process and attack trees which result in the identification of a number of risks, which could be security weaknesses. These attack trees visualize the security weaknesses in an easy to access way even for individuals without special security expertise. We develop a security solution to counter these identified risks. The developed security solution consists of three different security levels together with a number of new security policies. Each additional level applies different security mechanisms to provide increasingly improved security for the platform. The new security policies ensure that the security solution is continuously secure during its operating time. We implement part of the security solution in the Contiki operating system to assess its function in practice. Finally we evaluate the developed security solution by looking back to the previously identified weaknesses and the implementation proving that the security solution mitigates the risks.

Keywords: Wireless sensor networks, security, threat modeling, attack trees, risk, Contiki
# Contents

1 Introduction ................................. 6
   1.1 Intent .................................... 6
   1.2 Method ................................... 7
   1.3 Problem description ...................... 7
   1.4 Scenario ................................ 7
   1.5 Language ................................ 8
   1.6 Overview ................................ 8

2 Background ................................ 9
   2.1 The platform ............................. 9
      2.1.1 Components, hardware and functions 9
      2.1.2 Platform component interaction ....... 13
      2.1.3 Software .............................. 14

3 Analysis ................................ 18
   3.1 Platform assets .......................... 18
   3.2 Security policies ......................... 19
   3.3 Platform specific security requirements 19
   3.4 Known attacks on wireless sensor networks 20
   3.5 Threats modeling ........................ 21
      3.5.1 Step 1, Identifying assets ......... 21
      3.5.2 Step 2, Create an Architecture Overview 21
      3.5.3 Step 3, Decompose the platform .... 24
3.5.4 Step 4, Identify the Threats ................................................. 28
3.5.5 Step 5, Document the threats ........................................... 38
3.5.6 Step 6, Rate the threats .................................................... 50

4 Developing the security solution ............................................. 54

4.1 Security solution focus ....................................................... 54
4.2 Hardware limitations ......................................................... 55
  4.2.1 SAM4 ................................................................. 55
  4.2.2 RF233 ................................................................. 56

4.3 Existing security for wireless sensor networks ......................... 56
  4.3.1 TinySec ............................................................. 57
  4.3.2 MiniSec ............................................................. 57
  4.3.3 TinyECC ............................................................. 57
  4.3.4 ContikiSec ........................................................... 57
  4.3.5 ZigBee ............................................................... 58
  4.3.6 WaspMote ........................................................... 58

4.4 Our security solution ......................................................... 58
  4.4.1 Security levels ........................................................ 59
  4.4.2 The low security level ................................................. 59
  4.4.3 The medium security level .......................................... 64
  4.4.4 The high security level .............................................. 69
  4.4.5 Additional security issues .......................................... 73
  4.4.6 New security policies .............................................. 75

5 Evaluating the security solution .............................................. 79

5.1 Security requirements ....................................................... 79
5.2 Platform policies ........................................................... 80
5.3 Prevention, detection, reaction ........................................... 81
5.4 Protecting against known attacks ...................................... 81
  5.4.1 Subversion of a node attack ....................................... 82
5.4.2 Attacks not countered .................................................. 82
5.5 Cost for attacker ............................................................ 82
5.6 The low security level ....................................................... 83
  5.6.1 Security requirements .................................................. 83
  5.6.2 Prevention, detection, reaction ....................................... 84
  5.6.3 Protecting against known attacks .................................... 84
  5.6.4 Summary ................................................................. 86
5.7 The medium and high security level ..................................... 86
  5.7.1 Security requirements .................................................. 86
  5.7.2 Prevention, detection, reaction ....................................... 87
  5.7.3 Protecting against known attacks .................................... 87
  5.7.4 Summary ................................................................. 88
5.8 Attack trees ................................................................. 89
5.9 Implementation ............................................................. 89
5.10 Cost for our security solution .......................................... 91

6 Summary and Conclusion .................................................. 94
  6.1 Summary ................................................................. 94
  6.2 Discussion ............................................................... 94
  6.3 The security solution .................................................... 95
  6.4 Future work ............................................................. 96

Bibliography ........................................................................... 100

A Security policies evaluation ................................................. 101

B Abbreviations ..................................................................... 104

C Evaluation against the attack trees ..................................... 106
  C.1 Spoofing identity .......................................................... 106
  C.2 Tampering with data .................................................... 108
Chapter 1

Introduction

Today’s strong technology development allows the cost of wireless sensor networks to decrease while the reliability increases. This creates a virtuous circle with a growing number of platforms using wireless sensor network technology. Applications for these platforms can be anything from intelligent passage control systems to security systems monitoring users who carry a sensor node. Many of these applications can be very security-sensitive, e.g. it is vital that a sensor node tied to a certain user cannot be cloned when used in an intelligent passage control system. This means that we need to have security-awareness in platforms where security is important.

Open Controller has developed a platform using wireless sensor network technology. The intention of the platform is that it shall be used in environments with high security requirements. An example application for the platform is identification and positioning for users carrying sensor nodes. However, it has no security for the radio communication used and a flawed way to verify the authenticity of devices. These issues, among others, effectively restrain the platform from being used in its intended environment.

In this thesis we perform an in depth-analysis and develop a security solution for the platform in order to make it secure. We start this introduction chapter with the intent and problem description. We then show an example scenario in which the platform should be used. After the scenario we present the language used throughout the thesis and we end this chapter with an overview.

1.1 Intent

The purpose of this thesis was to assess the security and develop a security solution for Open Controller’s wireless sensor network platform. The security solution should be transparent to the applications running on top of the Contiki operating system, offer good security with today’s standards and have a low power consumption.
1.2 Method

In this thesis, a fairly straightforward approach is applied. That is, first we analyze the platform, then we construct a security solution and lastly we evaluate the developed security solution. The analysis chapter uses well known models including a threat modeling process from Microsoft. The same chapter also looks into assets to be protect in the platform, security policies and existing known attacks on the network type used in the platform. In the next chapter the security solution for the platform is developed using well known security mechanisms with regards to our hardware limitations and requirements. The final chapter is the evaluation chapter and it uses several known methods for evaluation of the security solution. This includes the prevention, detection and reaction methodology. The evaluation chapter also evaluates our security solution against security requirements, our platform policies, known attacks, different cost perspectives for an attacker and attack trees. Lastly in the evaluation chapter we use our proof of concept implementation to show data packages and cost in the form of package overhead with our security solution applied.

1.3 Problem description

The intent with this assignment is to perform an analysis and analyse security weaknesses for a platform using the wireless sensor network technology. The thesis shall investigate on existing techniques for security in wireless sensor networks and how they affect the power consumption. It shall also lead to a developed security solution. This security solution shall be evaluated in terms of security and performance. The performance is evaluated in terms of power consumption and network usage.

The current wireless sensor network platform lacks security thinking. Since many of the applications for the platform may be security critical this is a major drawback. The platform uses the Contiki operating system. Contiki is an operating system tailored for wireless sensor networks where resources are limited. Contiki does not include any built in security solution by itself and existing third-party solutions does not fit with the platforms requirements or hardware. Many more security solutions exist for the Contiki rival TinyOs. However these cannot be easily ported to Contiki and still misses out on some of our requirements. This requires a new solution to be developed for Contiki where our platforms hardware constraints and requirements are taken into consideration.

1.4 Scenario

The following scenario was developed in conjunction with Open Controller and used as a starting-point for this thesis.

The company SecretTec needs a security solution for their facilities. The company is developing several secret systems and products where both prototypes and production must be kept secret. Various prototypes, development and complete systems are located in different rooms in the facility. The company is looking for a security system that is able to meet the following requirements. The security system should be able to determine that only authorized personnel are in the premises and in which areas staff have been present. The system will consist out of one or
more base stations scattered in the facility where each person has a sensor card. Areas in the facility may be out of range of the base stations. In this scenario where a sensor card would be outside the range of the base stations, neighboring sensor cards that are in range would act as routers and route data to the base stations. Sensor cards are used for identification, positioning, and to see where the user has moved in the facility. The system must be able to detect if an unauthorized person is in the room by being able to identify false or erroneous sensor cards. If an unauthorized sensor card is detected, it is of interest to see in what areas of the room the unauthorized presence have been and with this knowledge the company should be able to take appropriate actions depending on what is located in that area. No information contained in the system should be revealed to unauthorized persons trying to listen in on radio traffic. The system must also be able to make sure that data received comes from an authenticated source and that the data has not been tampered with.

1.5 Language

This thesis and all used sources are in English. The language used in this thesis might require some security domain specific knowledge. Throughout this thesis we call the security developed for the security solution. When the term platform is used we mean the existing system developed by Open Controller. The term system used in this thesis means the platform together with our security solution unless otherwise stated. A commonly used term in security is message authentication code, abbreviated as MAC. Since we will talk about networking in this thesis the term MAC might also mean media access control. We decide to call the media access control for MAC while we use the alternative term message integrity code (MIC) instead of message authentication code. Since many abbreviations are used throughout the report a compilation of them can be found in Appendix B.

1.6 Overview

The thesis consists of a number of sections and include a partial implementation of the developed security solution. We begin with a background chapter where we go through today’s platform explaining its goals and how it works. We then carry out an analysis that includes a threat modeling process on the platform in Chapter 3. The result of this analysis gives us knowledge about the platform’s current security weaknesses. After this we develop a full security solution in Chapter 4 where we counter the identified security weaknesses and secure the platform. The penultimate part of this thesis is the evaluation in Chapter 5 where we evaluate the developed security solution. We tie together results from all the previous sections and the implementation to perform this thorough evaluation of our developed security solution. We end this thesis with a concluding discussion in Chapter 6.
Chapter 2

Background

This section introduces the platform and describes its functionality, device interaction and technologies used.

To get a grasp of our security requirements we first need to understand how the platform operates. We begin with subsection 2.1.1 where we look at the components used in the platform, their hardware and functions. We then move on to describe how component interaction is done in subsection 2.1.2. In the last part of this chapter, subsection 2.1.3 we go through the used communication techniques and the current software running in the platform.

2.1 The platform

The wireless sensor network platform consists of three different components. These are sensor nodes, base stations and the back-end. The platform supports for up to 256 base stations and up to between 10 000 and 20 000 sensor nodes. A layout of the platform’s components and how they interact can be seen in Figure 2.1.

2.1.1 Components, hardware and functions

Each component has a very specific role in the platform. While the back-end runs on a traditional x86 compatible server, sensor nodes and base stations uses Cortex ARM-based microcontrollers. The microcontrollers used are from the SAM4 series and manufactured by Atmel. The radio transceiver used is the RF233 by Atmel.

ARM-based microcontrollers and specifications

The microcontrollers available to us in our platform are the SAM4L and the SAM4E. We briefly talk about the cryptographic specifications relevant to this thesis in subsection 4.2.1. Table 2.1 shows a brief summary of the general microcontrollers specifications. The datasheet for SAM4L can be found in [4] and for SAM4E in [3].
Figure 2.1: Platform component interaction

<table>
<thead>
<tr>
<th></th>
<th>SAM4L</th>
<th>SAM4E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock (MHz)</td>
<td>48</td>
<td>120</td>
</tr>
<tr>
<td>Flash (Kbytes)</td>
<td>128-256</td>
<td>1024</td>
</tr>
<tr>
<td>SRAM (Kbytes)</td>
<td>32</td>
<td>128</td>
</tr>
<tr>
<td>Features include</td>
<td>Memory protection unit, picoPower technology</td>
<td>Memory protection unit, DSP instruction, Floating point unit</td>
</tr>
</tbody>
</table>

Table 2.1: ARM-based microcontrollers used by the platform
RF233 radio transceiver

The transceiver used for radio communication in the platform is the RF233 transceiver. It runs at 2.4 GHz and is targeted for IEEE 802.15.4 applications and provides several interesting features for this thesis. Features includes power saving modes, built in hardware support for 128-bit AES with message integrity code (MIC), built in media access control (MAC) accelerators with CSMA-CA and retransmission support. The datasheet can be found in [2].

The sensor node

The sensor node is the smallest component in the platform. It is the size of a credit card, sometimes smaller, and is powered by an onboard battery unit. An example of a sensor node can be seen in Figure 2.2. There exists several different versions of sensor nodes all providing different functionality. For instance a sensor node can be equipped with a gyro and accelerometer and deliver such data. The most common task for a sensor node in our scenario is to provide positioning data and identify who is carrying it.

![Example sensor node](image)

Figure 2.2: Example sensor node. Source: Open Controller [44]

The base station

A base station has a power cord instead of a battery. This enables the base station to have more powerful hardware since power consumption no longer is an issue. It has a much larger radio antenna and an Ethernet connection. Often several base stations are combined so they cover a larger area; this is called a base station array and can be seen in Figure 2.3 and Figure 2.4. The base stations in the array are connected to each other through the Ethernet connection. A base station also has a connection to the back-end through the Ethernet connection. It uses the back-end for authentication evaluation, data processing and data storage.
Figure 2.3: Several base stations combined into an array. Source: Open Controller [44]

Figure 2.4: Base station arrays placed in the room covering a larger area. Source: Open Controller [44]
The back-end

The back-end is a traditional server running some database management system for storing data. It processes sensor node data, stores it and also stores validation credentials. The stored credentials are setup and maintained by the company through a secure channel. The back-end communicates to all base stations through the Ethernet connection.

2.1.2 Platform component interaction

The sensor nodes can communicate with each other to relay data to a base station. A base station can communicate with other base stations, sensor nodes and the back-end. A base station sends data to the back-end where the data is processed and stored. Currently all radio and Ethernet communications are sent in the clear. Sensor to sensor communication is rarely used and it is limited to one level of sensor node routing. This means that a sensor node can relay packets through one other sensor node and no more. This is called the maximum depth of the network and illustrated in Figure 2.5. When a sensor node relays data it first aggregates it according to a aggregation protocol. The aggregation protocol specify how the data that will be relayed should be handled. For instance the protocol might specify that the relay data should be combined with the sensor nodes own data before being transmitted to the base station.

![Figure 2.5: The maximum depth of the network](Image)

New sensor nodes

When a new sensor node connects to the network it first authenticates with a base station. The base station redirects the request to the back-end. The back-end then evaluates the authentication request. If the authentication succeeds all communication from that sensor node will be accepted by the base station. This can be seen in Figure 2.6. When any sensor node transmits a packet to a base station on the radio channel the base station checks if it has already authenticated the supplied id, if not it send an authentication request to the back-end. The back-end
then tells the base station to process or drop the packet. Each sensor node and base station has a unique id used for identification; in the current platform the id is used as the credentials. In Figure 2.7 the base station processes data from the node with id 1. Since the authentication for the sensor node with id 2 was denied the base station will deny all packets from sensor node 2.

![Figure 2.6: Packets from sensor node 1 gets processed by the base station after a successful authentication.](image)

![Figure 2.7: Packets from sensor node 2 gets denied by the base station](image)

2.1.3 Software

As previously mentioned in the problem description part our platform uses the Contiki operating system. Contiki has a built in IP-stack and it is written in standard C. Contiki comes together with a network simulation environment called Cooja. In Cooja one can emulate devices running Contiki to evaluate their performance and how they function in small to large numbers. For more details on Contiki and Cooja see their website at [http://www.contiki-os.org/](http://www.contiki-os.org/).

The current network stack used by the platform can be seen in Table 2.2. It uses several tailored protocols on each layer since the standard protocols are too resource heavy for wireless sensor networks. We will briefly go through the protocols in the lower stack layers.
IEEE 802.15.4

In this thesis the different frames used by the 802.15.4 standard is of some interest. We will look at frames in the media access control (MAC) layer and physical layer. The MAC layer is located in the data link layer in the OSI model. The maximum frame size for an 802.15.4 frame is 127 octets, Society [41].

MAC-layer The MAC layer provides channel access control mechanisms. The platform uses time division multiple access (TDMA). In short TDMA has predefined timeslots in which each node can send and thus no collisions exist. We have several different frames at the MAC layer in the IEEE 802.15.4 standard. We begin with describing the general MAC frame format in Table 2.3.

<table>
<thead>
<tr>
<th>Octets</th>
<th>2</th>
<th>1</th>
<th>variable</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Frame control</td>
<td>Sequence Number</td>
<td>Destination PAN id</td>
<td>Destination Address</td>
</tr>
</tbody>
</table>

Table 2.3: General frame format

The frame control field contains information defining the frame type, addressing fields and other control flags. The sequence number field is the sequence identifier for the frame. The auxiliary security header specifies information required for security processing. The Frame check sequence (FCS) field contains a 16-bit cyclic redundancy check (CRC) for error detection. IEEE 802.15.4 specifies three different frame formats. The beacon frame, the data frame and the acknowledgment frame. All different frames use different fields from the general frame format above.

The data frame in the platform, used to transport for instance temperature data, uses the frame format in Table 2.4 Since the maximum frame size is 127 octets we can utilize 122 octets to our payload in the current platform. The explanation for this data frame layout and more details on the communications can be found in section 2.1.3.

<table>
<thead>
<tr>
<th>Octets</th>
<th>2</th>
<th>1</th>
<th>variable</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Frame control</td>
<td>Sequence Number</td>
<td>Frame payload</td>
<td>FCS</td>
</tr>
</tbody>
</table>

Table 2.4: The data frame format

Physical-layer The physical layer uses a physical protocol data unit (PPDU) with the frame format in Table 2.5.

The synchronization header (SHR) allows a receiving device to synchronize and lock onto the
The physical header (PHR) contains the frame length information and the physical payload (PHY payload) contains a MAC sublayer frame.

The time frame We have an additional frame called the time frame in our platform. This frame is specifically developed for use in our platform and is not defined in the IEEE 802.15.4 standard. The time frame is sent out at the beginning of each timeslot by each base station. This frame is used to provide time synchronization and other information needed by sensor nodes to determine whose time it is to transmit. The new time frame contains at least the fields in Table 2.6.

<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base station number</td>
<td>Each base station has a unique base station number</td>
</tr>
<tr>
<td>Number of timeslots</td>
<td>The number of timeslots in one frame as power of $2^N$</td>
</tr>
<tr>
<td>Length of a Timeslot</td>
<td>Length of a timeslot in timestamp units as a power of $2^N$</td>
</tr>
<tr>
<td>Current Timeslot</td>
<td>The current timeslot we are in</td>
</tr>
<tr>
<td>Timestamp</td>
<td>Used for synchronization. Given in $1/32768$ second units</td>
</tr>
</tbody>
</table>

Table 2.6: The time frame fields

Communications The platform uses time division multiple access (TDMA) to determine which sensor node can send at what time. This is what makes it possible to remove the addressing fields from our data frames and thus reducing the header size for each packet. Since the base station send out a time frame in the beginning of each timeslot every client knows whose turn it is.

In Figure 2.8 we have partitioned each time frame into five slots with TDMA. Node 1 (white)
can only send when a white timeslot occurs. The time frame is sent at the beginning of each timeslot. The time frame contains all information necessary for each sensor node in the network to synchronize their time and to determine whose time it is to send. If a sensor node fails in transmitting all data in its allocated timeslot it has to wait for its next timeslot. Each sensor node uses the following calculation to calculate its timeslot.

\[
\text{Timeslot} = \frac{\text{SensorNodeNumber} + \text{CoordinatorNumber}}{\text{NumberOfTimeslotsInOneFrame}}
\]

The \textit{SensorNodeNumber} is for instance 1 if we would be the first node in Figure 2.8. The \textit{CoordinatorNumber} is the identification number for the coordinator that send out the received time frame while the \textit{NumberOfTimeslots} would be 5 if we look at Figure 2.8. It is by using this technique we can guarantee that the correct node transmits data during its turn and therefore we can remove all addressing fields in the MAC header. TDMA enables nodes to be able to sleep, thus save energy, during other sensor nodes time slots.

In addition to the use of TDMA, the radio communication in the platform utilizes frequency hopping. The medium available is partitioned into 32 channels. A protocol is in place specifying what channel the communication parties will use at a certain point in time. This technique is mandatory in the platform since it is used in the positioning algorithm for positioning of a sensor node. This frequency hopping is not considered to add any additional security to the platform. An attacker could either read the protocol defining the hopping pattern or obtain 32 radios listening in on the channels. Thus, in this thesis we will only consider this as something that is part of the positioning technique.
Chapter 3

Analysis

In this chapter we perform the analysis of the platform. We begin with identifying our platform assets in section 3.1. We need to know our assets before we can write our initial security policies which we do in section 3.2. The initial security policies help us specify rules for what is allowed and disallowed in the platform. The developers of the platform have some system specific security requirements and we review these in section 3.3. We then go through some known attacks on wireless sensor networks in section 3.4 to get an understanding of existing attacks on our type of network. Lastly we carry out our threat modeling process in section 3.5 to reveal threats and to create a threat risk rating table.

3.1 Platform assets

The first thing we need to know is what do we want to protect. In his book on computer security Gollmann writes [15],

"Security is about the protection of assets. This definition implies that you have to know your assets and their value."

Therefore we start with identifying our assets in the platform. The outcome from this section will be used when we construct our initial security policies and later in our threat modeling process. We use our scenario to help us identify our assets. In terms of physical assets we want to protect the sensor nodes and the base stations. Software assets are the information or data flowing through the platform and stored security data. The assets can be categorized into the following categories.

Hardware assets

- Base stations
- Sensor nodes

Data and information assets
• Data from the sensor nodes
• Data from the base stations
• Stored credentials and system data

3.2 Security policies

When we know what we want to protect we have to specify what we are allowed to do in the platform. We construct our initial security policies in this section to define allowed events. A security policy is a set of rules that must be applied and enforced by the platform to guarantee some predefined level of security. Security policies can be enforced through the use of security mechanisms. The term network used in this part of the thesis will denote the radio communication network where sensor nodes talk to other sensor nodes or to a base station. By looking at our identified assets and using the definitions of a security policy by Matt Bishop [6] we arrive at these security policies for our platform.

1. Only an authorized and authenticated user with an associated sensor node is allowed to participate in the network.
2. Only authorized parties shall be able to listen in on the network traffic to obtain information.
3. Only authorized parties shall be able to modify the content being transmitted in the network.
4. Only company-configured base stations shall be able to participate in the network.

The security policies explicitly defines what is allowed. In Appendix A we investigate how our security policies hold according to the definitions.

3.3 Platform specific security requirements

In this wireless sensor network it was important that some security requirements took precedence over others. The following prioritized list of requirements were created in collaboration with the developers of the platform.

1. No false node shall be able to take part of the network
2. No false base station shall be able to take part of the network
3. No outsider shall be able to listen in on the network traffic
4. No outsider shall be able to tamper with traffic being sent in the network
5. No outsider shall be able to limit the availability of the network

This ranking will weight in later when we calculate the risk factor for each risk.
3.4 Known attacks on wireless sensor networks

There exists a variety of different attacks on wireless sensor networks. Several are quite similar to known attacks in more traditional networks while others are more specific for our domain. We will mention the attacks located in each layer together with a quick explanation. These attacks are described further and in more details in Sokullu et al. [42] and Q. Wang [37]. Many of these attacks can be seen in the attack-trees part, section 3.5.4, and their corresponding descriptions.

Physical layer

*Physical layer jamming*, an attacker interferes with the radio communication.

*Subversion of a node*, a node gets stolen and sensitive information can be extracted.

Data link layer

*Link layer jamming*, a finer grained jamming attack than at the physical layer.

*Eavesdropping*, an attacker listens in on the communication.

*Collisions*, the attacker causes packet collisions resulting in for instance exponential back-off.

*Resource exhaustion*, an attacker consumes scarce resources, for instance battery power.

*Traffic analysis*, by observing traffic patterns an attacker can identify devices in the network.

*Packet-tracing*, with the right equipment an attacker can get the location of the immediate transmitter of an overhead packet and performing a hop-by-hop trace towards the data source.

*Clock unsynchronization*, an attacker can disrupt the sensor nodes current time thus causing them to be in an incorrect phase.

Network layer

*Spoofed, altered, or replay routing information*, an attacker can affect the routing protocol in a malicious way by crafting packets.

*Sybil*, an attacker forges multiple identities from a compromised node.

*Selective forwarding*, the attacker chooses which data to forward when acting as a relay.

*Sinkhole*, involves sending out routing information making the node look more attractive to relay through. An attacker can then apply selective forwarding or other attacks.

*Wormholes*, the attacker capture packets in one part of the network and retransmit them in a different part.

*Hello flood attacks*, tricking sensor nodes that they are within transmit range with the use of a high powered transmitter.

*Acknowledgment spoofing*, the attacker can influence the routing algorithms. For instance change what link a sensor node will use.

*Flooding*, exploiting that many protocols needs to maintain sessions an attacker can flood the
platform with new connections until all resources are consumed. 

Desynchronization, an attacker can disrupt existing connections causing expensive recovery functions to be run.

Application layer 

False data filtering, by attacking an aggregation point the attacker can corrupt all data relaying through that node.

False data injection, the attacker acts as an aggregator and injects malicious data in the aggregation process.

3.5 Threats modeling

We will base our threat modeling on Microsoft’s threat modeling process [29]. The model uses both the STRIDE [30] threat model and the DREAD [29] threat risk rating. More information on the two will follow in each respective section of the threat modeling process. The Microsoft threat modeling process is recommended by the open web application security project [36]. The model is more aimed towards web security but is general enough to be used on our wireless sensor network. This was one of the reasons to why we choose this model. Other reasons are that the model is thoroughly used and tested by Microsoft. Software created by Microsoft have been the target for many attacks and this have forced Microsoft to improve their security awareness during the last decade. This model is part of the result from this increased security awareness; the model is also easy to understand and straightforward to work with.

The threat modeling process includes the following steps. Throughout the threat modeling process we have changed the term system to platform to match our used language.

1. Identify assets
2. Create an architecture overview
3. Decompose the platform
4. Identify the threats
5. Document the threats
6. Rate the threats

3.5.1 Step 1, Identifying assets

We have the same assets as in section 3.1 where we deduced assets from the scenario.

3.5.2 Step 2, Create an Architecture Overview

We carry out the following three sub-steps when creating the architecture overview.
1. Identify what the platform does

2. Create an architecture diagram

3. Identify the technologies

In the original model the first step is identify what the application does. We change this first step to what the platform does since it is more correct in our context.

**Identify what the platform does** To visualize how users and guards from our scenario may interact with the platform we have constructed the use case in Figure 3.1.

![Use case diagram](image)

**Figure 3.1: Use case illustrating platform interaction**

User X carries a sensor node with him and enters the facility. The sensor node joins the network and starts to send data when it is in range of a base station. The data being sent contains positioning and identification information for the user. The user X moves inside the facility between different rooms and performs his daily work.

The guard Y is responsible for the security in the facility. The guard needs to ensure that personal only resides in rooms to which they have clearance. The guard Y monitors the user X. The guard uses data produced by the platform to identify and position the user X in real time while the user moves around in the facility.

**Architecture diagram**

**Architecture diagram** In the architecture diagram we describe the composition, structure and trust boundaries in the platform.

The platform has several different trust boundaries as seen in Figure 3.2. Each sensor node has its own trust boundary and the base stations and back-end lies in the same trust boundary. One could argue for the use of two trust boundaries inside the sensor node as in Figure 3.3. One for the microcontroller and one for the transceiver. Thus considering the bus in-between, called the SPI bus, to be a insecure. However, this lies outside the scope of this thesis and will thus not be investigated further.
Figure 3.2: Platform trust boundaries

Figure 3.3: Not included internal trust boundary in a sensor node
Identify the technologies  Lots of different Technologies are used in the current platform. In Table 3.1 we present a selected few which are of interest to us when focusing on the security.

<table>
<thead>
<tr>
<th>Technology/Platform</th>
<th>Implementation details</th>
</tr>
</thead>
<tbody>
<tr>
<td>IEEE 802.15.4</td>
<td>Used for all traffic between sensor nodes and base stations</td>
</tr>
<tr>
<td>Database</td>
<td>Includes logins, access rights and stored data</td>
</tr>
<tr>
<td>Contiki</td>
<td>Operating system used in sensors and base stations</td>
</tr>
<tr>
<td>Rime</td>
<td>Light-weight network stack, replaces the IP stack</td>
</tr>
<tr>
<td>Ethernet</td>
<td>Used for traffic between base stations and the back-end</td>
</tr>
</tbody>
</table>

Table 3.1: Technologies used

3.5.3 Step 3, Decompose the platform

In this step we create a security profile. The objective of the security profile is to uncover vulnerabilities in the design, implementation, or deployment configuration of our platform. We perform the following substeps when decomposing the platform.

1. Identify trust boundaries
2. Identify data flow
3. Identify entry points
4. Document the security profile

In the model one extra substep exists, the step of identifying privileged code. Since we only evaluate the network communication we will not go deeper and look into what security permissions code running on the microcontroller have.

Identify trust boundaries  From the architecture diagram in Figure 3.2 we can see that we have distinct trust boundaries between the devices in the platform. The only entry point into the platform is through the radio transceiver. The current platform has a gatekeeper at this entry point; thus only sensor nodes and base stations with valid credentials can connect and participate in the network. Information sent over the Ethernet connections is considered to be trusted.

There are several chains of trust in the platform. The database trusts the base stations to only forward data to be stored from authenticated sensor nodes. Likewise base stations and sensor nodes consider data from authenticated sensor nodes and base stations to be trusted.

Identifying data flow  We mainly look at the network communication in this thesis. This gives us the data flows in Figure 3.4

1. Sensor node to sensor node
2. Sensor node to base station and vice versa
3. Base station to base station
4. Base station to back-end and vice versa

A typical data flow would be to send data from a sensor node to a base station to the back-end, in the above figure the flow would be $2 \rightarrow 3 \rightarrow 4$.

**Identify entry points** Entry points are points in our architecture that are designed to be exposed. In our platform we only have entry points at the radio transceivers that lie at the trust boundary border. There are different gatekeepers depending on which device we are looking at.

At the sensor node transceiver we have a gatekeeper that does not perform any authorization or authentication of received packets. The sensor node assumes that it should route all packets if the packet originates from another sensor node, thus acting as a relay. If a received packet comes from the base station it blindly accepts it. The only validation that is being done is an integrity check with the use of a 16-bit cyclic redundancy check (CRC-16).

The base station gatekeeper at our entry point does perform both authentication and authorization by asking the back-end. The back-end evaluates the request and tells the base station to grant or deny packets from the sensor node.

**Document the security profile** To put together the security profile we use the knowledge from the above substeps. From our threat model we also get questions to ask in the following areas; input validation, authentication, authorization, configuration management, sensitive data, session management, cryptography, parameter manipulation, exception management and auditing and logging. By answering these questions we will be able to detect additional vulnerabilities.

**Input validation**

**Is all input data validated?** In our platform the term input data refers to the packets received on the radio channel. Like before we need to talk about input data into the sensor nodes and into the base stations. The sensor nodes do no validation what so ever of the received data. The base station on the other hand does validation by checking that the supplied id is valid with the back-end.
Could an attacker inject commands or malicious data into the platform? Several ways of injecting packets into the network exists. One possibility is for an attacker to spoof the identity of a valid node, thus tricking the platform into accepting false packets. The network has several vulnerabilities in this aspect.

Is data validated as it is passed between separate trust boundaries (by the recipient entry point)? Since we look mainly at the communication there are no more trust boundaries when we have passed the transceiver gatekeeper.

Can data in the database be trusted? This question has two aspects to it. We know that the stored credentials containing valid sensor nodes and base stations are setup over a secure channel and that they cannot be modified through the sensor network entry point. However, we also know that sensor node data cannot be trusted in the current platform. The conclusion here is that we can trust the credentials in the database but we cannot trust any stored or processed sensor node data.

Authentication

Are credentials secured if they are passed over the network? In the current network the credentials are the identification of either a base station or a sensor node. These credentials are being sent in an insecure manner through the network.

Are strong account policies used? Accounts in our platform can be said to be adequate to the stored credentials at the back-end; thus a sensor node requires its credentials to be stored at the back-end, much like needing to have an existing account, before it can connect. Currently there is no specific account policy other than that a device in the platform must have an existing account to participate.

Are strong passwords enforced? No explicit passwords exist in the network. When a node connects it just supplies its id. This current id is to be considered more like a username than a password.

Are you using certificates? No certificates are in use.

Are password verifiers (using one-way hashes) used for user passwords? No passwords are currently in use.

Authorization
What gatekeepers are used at the entry points of the platform? Like before there are two types of gatekeepers. The ones at the sensor node accepting everything except corrupted data and those at the base stations whom perform authentication.

How is authorization enforced at the database? Since this lies outside of the reach of the thesis we will just assume that the back-end does some secure and correct authorization when processing an authentication request.

Is a defense in depth strategy used? No, the only layer of security controls is those performed by the base station gatekeeper.

Do you fail securely and only allow access upon successful confirmation of credentials? At the base station we do fail securely by denying packets from any unsuccessful sensor node validation.

Configuration management. All configuration of the platform is done before deployment, for instance a new sensor node needs to be preconfigured with a correct id. The tasks done after deployment involve managing the back-end and database data. We assume that this configuration management is secure.

Sensitive data

What sensitive data is handled by the platform? All data being sent in the network is to be considered as sensitive.

How is it secured over the network and in persistent stores? Currently it is not secured when being transmitted over the network. We assume the persistent storage in the back-end is secure.

What type of encryption is used and how are encryption keys secured? No encryption is used in the current platform.

Session management All network traffic over the radio is currently sessionless.

Cryptography Currently no cryptography is being utilized in the platform.

Parameter manipulation
Does the platform detect tampered parameters? The platform does detect tampering if the CRC-16 checksum is invalid. If an attacker re-calculates the CRC-16 checksum the platform will not detect the tampering.

Does it validate all parameters in network header fields? No, the CRC checksum is validated but cannot be considered to be a security validation of any kind.

Exception management

How does the platform handle error conditions? The error condition we are interested in is when packets do not arrive or arrive corrupted. The platform retransmits the packets in these cases.

Auditing and logging

Does your platform audit activity across all tiers on all devices? No auditing is currently being performed.

3.5.4 Step 4, Identify the Threats

In this step we will use Microsoft STRIDE threat model to group threats into categories and by doing so we have a structured method of identifying threats. STRIDE stands for **Spoofing identity**, **Tampering with data**, **Repudiation**, **Information disclosure**, **Denial of service** and **Elevation of privilege**. We go through each category in STRIDE and identify threats. We also use threats from existing threat modeling done on wireless sensor networks, Q. Wang [37].

STRIDE

Spoofing identity

1. Get credentials of legitimate sensor node
2. Get credentials of legitimate base station
3. Craft malicious packets

Tampering with data

1. Tamper with a packet in the radio channel
2. Influence data stored in the back-end
Repudiation

1. Perform illegal operation which origin cannot be traced

Information disclosure

1. Outsider gets information flowing in the network
2. Outsider gets information on the topology of the network
3. Get traffic source location

Denial of service

1. Limit availability of a sensor node
2. Limit availability of a base station

Elevation of privileges

1. Get sensor node privileges for malicious device
2. Get base station privileges for malicious device

Attack trees Creating attack trees for each identified threat is a structured and hierarchical way of documenting the potential attacks on the platform. The root of the attack tree is the threat while the nodes, or leaves, are goals that must be met before the attack can succeed; in other words before the threat becomes real. We create an attack tree and hold a brief discussion for each threat identified above. In short we mention possible attacks in these discussions, a more detailed view of known attacks on wireless sensor networks can be found at section 3.3. We start with the spoofing identity threats.

Spoofing identity threats The attack trees on get credentials of legitimate sensor node, Figure 3.5 and get credentials of legitimate base station, Figure 3.6 are quite similar. However the difficulties in performing the step in the two vary greatly. For instance it is much easier to steal a sensor node than to steal a base station. At the same time it might be easier to copy the base station onboard flash memory than the sensor nodes. This is because the base station is unsupervised most of the time while the sensor node is carried by a person. All three of these threats can be said to compromise both the integrity and confidentiality of the platform. Using the NIST \[34\] definition of system integrity, it is a state in which the system or platform “performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system”. In our case system, or platform, integrity is considered to be violated if an attacker crafts malicious packets and gets the platform to accept them; thus we can no longer trust the information flowing in the platform. The confidentiality can be said
By obtaining the credentials of a legitimate sensor node an attacker can participate in the network under the stolen credentials. This opens the door to several other possibilities for an attacker. The attacker could for instance carry out actions in the legitimate sensor nodes name. Actions could be everything from feeding the platform with incorrect sensor data, thus corrupting the data in the platform, to injecting false routing information into the platform. This false routing information could enable an attacker to become a relay for part of the network and thus be able to capture all traffic flowing through it. This threat is often a prerequisite for other attacks which require an attacker to participate in the network.

When getting the credentials of a legitimate base station an attacker can in general do the same things as when the credentials came from a legitimate sensor node. However an attacker could much easier compromise a bigger part of the network by telling all sensor nodes in range to send to it, thus capturing and compromising a much larger chunk of the network.

The threat of crafting malicious packets, Figure 3.7, is dependent on the previous two threats about getting valid credentials to use. It might, for instance, be easy for an attacker to copy a packet and send it at a later time. If the platform has no timestamps or other mechanisms to validate the freshness of the packet this attack will succeed. This type of attack will succeed even if the data is encrypted. It is called a replay attack and illustrated by the sub goal box “Copy packet from the network and replay at a later time”.

Figure 3.5: Get credentials of legitimate sensor node attack tree

to be compromised since an attacker have, unauthorized, obtained information flowing in the platform; in this case credentials.
Figure 3.6: Get credentials of legitimate base station attack tree

Figure 3.7: Craft malicious packet attack tree
Tampering with data threats  The threats in this category are related with the integrity of the platform. A broad range of attacks are available if the integrity of the platform is violated. Some examples are given in the discussions following the attack trees.

If an attacker would be able to tamper with packets flowing in the radio channel, he or she could for instance break the platform by injecting false routing information. Another attack would be to scramble the communication order of nodes by tampering with the time frame packets. The sub goal “Guess packet field layout” might need an extra explanation. Assume an attacker knows the exact bits containing the information he or she wants to modify. The attacker could then modify the bits even if they are encrypted and hope that the modification will go unnoticed.

With influencing data stored in the back-end, we mean the threat that an attacker could make the platform produce the wrong output results. For instance assume that we have a temperature monitoring system. This system then calculates the room temperature by taking the average value of all temperature readings from the sensor nodes. An attacker could feed the platform with malicious data thus tricking the monitoring system to think that the average

Figure 3.8: Tamper with a packet in the radio channel attack tree
Influence data stored in the back-end

Replay captured data packets
Deploy malicious base station
Deploy malicious sensor node
Feed the system with malicious data

Figure 3.9: Influence data stored in the back-end attack tree

Room temperature is much higher or lower than it really is. Some other mechanism might then base its actions on this false result.

Repudiation threats According to STRIDE a repudiation threat is that a user denies performing an action without other parties having any way to prove otherwise. In our case we discuss how an attacker can perform illegal operations in the platform without the platform having any ability to trace the prohibited operation, Figure 3.10. Illegal operations in our platform can be translated into the crafting of malicious packets. For instance a malicious packet might contain false routing information or sensor data. A common prerequisite is that the attacker obtains a sensor node or base station and successfully can participate in the network.

Several ways to craft packets which can contain malicious data exist. If no logging is done the platform cannot tell who sent what in retrospect. On the other hand, if a logging facility exists in the platform and the attacker needs to send data through it, the attacker needs to take that into consideration when crafting the packet. For instance the attacker could replay a previously transmitted packet from another sensor node containing routing information. The attacker could also skip sending any credentials in the packet hoping the platform will accept and process it.
anyway or simply supply false credentials.

**Information disclosure threats**  All threats in this threat category share the first level of goals. These goals specify that an attacker needs to somehow get hold of the traffic flowing through the network. The threats in this category threaten the confidentiality of the platform since all of the threats, if successful, leak information transported.

![Diagram of outsider getting information flowing in the network](image)

Figure 3.11: Outsider gets information flowing in the network attack tree

If an attacker were to circumvent the security mechanisms, thus getting at the information flowing through the network as seen in Figure 3.11, he or she could for instance get information on the current location of the sensor nodes. In our scenario we might require all security personnel to wear sensor nodes and thus an attacker would at all times know the exact location of the security personnel.

Finding out the topology of the network, Figure 3.12, is often a prerequisite for future attacks. An attacker needs to know how the network looks and in some sense works to be able to craft directed attacks. For instance if the attacker wants to attack the platform he or she looks for a vulnerable entry point into it. This entry point can much easier be identified when knowing the network topology. Another attack would be if the attacker wants to break the platform. If the attacker knows the network topology and knows that certain sensor nodes or base stations are trivial to the platform the attacker can focus on bringing down those more vital parts. Even when the whole packet is encrypted an attacker could analyze traffic patterns; the attacker might know that a base station will periodically send out time frames. Thus looking for this periodic traffic the attacker will be able to identify the base stations in the network.

If we assume an attacker wants to destroy a specific sensor node in the platform he or she would have to be able to deduce the location of it based on the traffic it sends, Figure 3.13. If this sensor node is attached to an extra valuable piece of inventory or equipment the attacker would
Figure 3.12: Outsider gets information on the topology of the network attack tree
Figure 3.13: Get traffic source location attack tree
be able to tell its position in the facility before even entering the premises. On this threat we also have the sub-goal “move inside the network” and this might need some extra explanation. By moving physically in the network we might manage to get into a position where we can route or relay traffic from the source we are interested in. If an attacker could perform this type of attack he or she would be able to pinpoint the source of the traffic quite easily.

**Denial of service threats**  The ways to reduce the availability of a sensor node, Figure 3.14 and a base station, Figure 3.15 are quite similar. In all wireless networks an attacker can perform jamming. In our case an attacker can perform jamming in the physical layer and the data link layer. Physical jamming includes sending on the frequency bands in which the network operates. The network will stay jammed as long as the jamming equipment is turned on. Data link layer jamming means that we disrupt the protocol at the MAC layer. Since we use TDMA in our platform an attacker could choose to occupy the channel for a specific sensor node. This will result in a denial of service attack for that specific sensor node. When no ACK-packet arrives (since the data is scrambled by the attacker) the sensor node will try to retransmit. In most cases it is easy to observe if the platform is under a denial of service attack. If an attacker is jamming the network, physical security personal may be required to interfere to stop the jamming.

Denial of service attacks on a sensor node includes the injecting of false routing data, causing repeated collisions and sensor node desynchronization. An attacker can most probably come up with several more malicious packet attacks to reduce the availability. Since the resources, for instance battery power, is very limited for a sensor node an attacker can easily drain it with several different attacks. These attacks include relaying large amounts of data over the sensor node, cause it to do lots of expensive retransmissions or force it into launching expensive re-synchronization protocols.

One of the main differences between availability threats on base stations and sensor nodes is that the base station does not have limited battery power. This basically reduces the possible resource consumption attacks to connection flooding since we still have a limited amount of memory. The other types of sub goals for an attacker remain in large the same.
Limit availability of a base station

Participate in network

Perform jamming

Deploy malicious sensor node acting as a relay

Deploy malicious base station

Flood the base station with new connections

Create malicious packets

Physical layer

Link layer

Block forwarding of selected sensor nodes data

Figure 3.15: Limit availability of a base station attack tree

**Elevation of privileges threats** If a malicious device gains privileges of a sensor node, Figure 3.16 or base station, Figure 3.17, the device can participate in the network like any other legitimate component. This would enable the owner of the malicious device to perform basically all previously discussed attacks since the attacker has circumvented the security mechanisms. The attack trees for this threat category are small since the goals are that an attacker needs to obtain valid credentials to use in the platform. Attack trees for obtaining these credentials are given under the spoofing identity threat category, section 3.5.4.

![Get sensor node privileges for malicious device attack tree](image)

Figure 3.16: Get sensor node privileges for malicious device attack tree

![Get base station privileges for malicious device attack tree](image)

Figure 3.17: Get base station privileges for malicious device attack tree

### 3.5.5 Step 5, Document the threats

In this step we document the threats based on our analysis so far. For details on how the risk ranking was done see step 6 at subsection 3.5.6. The most effective way to mitigate a threat is to mitigate it as close to the root as possible. We will not do countermeasures for all nodes in the attack trees. It might not always be feasible or even possible to have countermeasures for
an attack. For instance it is hard to implement a protective mechanism in software so a sensor node cannot be stolen. We will deduct countermeasures so we secure the root by eliminating all paths to it in the tree, this will be visualized graphically for each threat by the use of a red dotted line. Since most of the threats share at least some sub-goals of the attack trees we will have many repeated attack techniques and countermeasures. We begin this step with our spoofing identity threats.

1. Spoofing identity

<table>
<thead>
<tr>
<th>ID</th>
<th>1.1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Threat description</strong></td>
<td>Get credentials of legitimate sensor node</td>
</tr>
<tr>
<td><strong>Threat target</strong></td>
<td>Credential of legitimate sensor node</td>
</tr>
<tr>
<td><strong>Risk</strong></td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Attack techniques</strong></th>
<th><strong>Countermeasures</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Observe non-encrypted credentials</td>
<td>Encrypt traffic</td>
</tr>
<tr>
<td>Deploy malicious base station</td>
<td>Authentication of base stations by sensor nodes</td>
</tr>
<tr>
<td>Deploy malicious sensor node acting as a relay</td>
<td>Authentication of sensor nodes by sensor nodes</td>
</tr>
<tr>
<td>Extract non-encrypted credentials from on-board memory</td>
<td>Store cryptographic keys more securely by disabling interfaces and wiping memory when re-enabled</td>
</tr>
<tr>
<td>Brute force</td>
<td>Use strong cryptographic keys</td>
</tr>
<tr>
<td>Dictionary attack</td>
<td>Do not use common words as keys</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>1.2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Threat description</strong></td>
<td>Get credentials of legitimate base station</td>
</tr>
<tr>
<td><strong>Threat target</strong></td>
<td>Credential of legitimate base station</td>
</tr>
<tr>
<td><strong>Risk</strong></td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Attack techniques</strong></th>
<th><strong>Countermeasures</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Observe non-encrypted credentials</td>
<td>Encrypt traffic</td>
</tr>
<tr>
<td>Deploy a malicious sensor node acting as legitimate one</td>
<td>Authentication of sensor nodes by base stations</td>
</tr>
<tr>
<td>Relay traffic to legitimate sensor node</td>
<td>Authentication of sensor nodes by base stations</td>
</tr>
<tr>
<td>Extract non-encrypted credentials from on-board memory</td>
<td>Store cryptographic keys more securely by disabling interfaces and wiping memory when re-enabled</td>
</tr>
<tr>
<td>Brute force</td>
<td>Use strong cryptographic keys</td>
</tr>
<tr>
<td>Dictionary attack</td>
<td>Do not use common words as keys</td>
</tr>
</tbody>
</table>
Figure 3.18: Attack tree to get credentials of legitimate sensor node with applied countermeasures
Figure 3.19: Attack tree to get credentials of legitimate base station with applied countermeasures
Threat description
Craft malicious packet
Threat target
The network in the platform
Risk
High

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use no credentials</td>
<td>Use some authentication mechanism</td>
</tr>
<tr>
<td>Use made-up credentials</td>
<td>Use authentication with good unpredictable credentials</td>
</tr>
<tr>
<td>Get credentials of valid sensor node</td>
<td>See previous result from Spooing identity</td>
</tr>
<tr>
<td>Get credentials of valid base station</td>
<td>See previous result in Spooing identity</td>
</tr>
<tr>
<td>Copy packet from the network and replay at a later time</td>
<td>Use timestamps or counters to determine freshness</td>
</tr>
</tbody>
</table>

Figure 3.20: Attack tree to craft a malicious packet with applied countermeasures

2. Tampering with data

<table>
<thead>
<tr>
<th>ID</th>
<th>2.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Tamper with a packet in the radio channel</td>
</tr>
<tr>
<td>Threat target</td>
<td>Packets sent by base stations or sensor nodes</td>
</tr>
<tr>
<td>Risk</td>
<td>Medium</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture non-encrypted packet</td>
<td>Encrypt traffic</td>
</tr>
<tr>
<td>Capture non-encrypted headers in packet</td>
<td>Encrypt the whole packet including headers</td>
</tr>
<tr>
<td>Deploy malicious sensor node acting as a relay</td>
<td>Authentication of sensor nodes by sensor nodes</td>
</tr>
<tr>
<td>Deploy malicious base station</td>
<td>Authentication of base stations by sensor nodes</td>
</tr>
<tr>
<td>Brute force</td>
<td>Use strong cryptographic keys</td>
</tr>
<tr>
<td>Dictionary attack</td>
<td>Do not use common words as keys</td>
</tr>
<tr>
<td>Carry out modifications that will be unnoticed</td>
<td>Use message integrity code (MIC)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>2.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Influence data stored in the back-end</td>
</tr>
<tr>
<td>Threat target</td>
<td>Stored sensor data</td>
</tr>
<tr>
<td>Risk</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Security in Wireless Sensor Networks for Open Controller
Thesis report chren758@student.liu.se
Figure 3.21: Attack tree with for packet tampering with applied countermeasures
### Attack techniques vs Countermeasures for Influencing Data Stored in the Back-End

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploy malicious sensor node</td>
<td>Authentication of sensor nodes by base stations</td>
</tr>
<tr>
<td>Deploy malicious base station</td>
<td>Authentication of base stations by sensor nodes</td>
</tr>
<tr>
<td>Replay captured data packets</td>
<td>Use timestamps or counters to determine freshness</td>
</tr>
</tbody>
</table>

---

#### Figure 3.22: Attack tree for influencing data stored in the back-end with applied countermeasures

---

### 3. Repudiation

<table>
<thead>
<tr>
<th>ID</th>
<th>3.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Perform illegal operation which origin cannot be traced</td>
</tr>
<tr>
<td>Threat target</td>
<td>The non-repudiation of components in the platform</td>
</tr>
<tr>
<td>Risk</td>
<td>Medium</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participate in network</td>
<td>Authentication mechanisms so only legitimate devices can participate</td>
</tr>
<tr>
<td>No logging facility</td>
<td>Deploy a logging facility in every device</td>
</tr>
<tr>
<td>Craft packet without credentials</td>
<td>Reject packets without credentials</td>
</tr>
<tr>
<td>Craft packet using false credentials</td>
<td>See previous result in Spoofing identity</td>
</tr>
<tr>
<td>Copy transmitted packet from other sensor node</td>
<td>Use timestamps or counters to determine freshness</td>
</tr>
</tbody>
</table>

---

### 4. Information disclosure

<table>
<thead>
<tr>
<th>ID</th>
<th>4.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Outsider gets information flowing in the network</td>
</tr>
<tr>
<td>Threat target</td>
<td>Data transmitted from a sensor node or base station</td>
</tr>
<tr>
<td>Risk</td>
<td>Medium</td>
</tr>
</tbody>
</table>
Craft packet using false credentials
Perform illegal operation which origin cannot be traced
Participate in network
Obtain sensor node or base station
Detect logging facility in sensor nodes, base station or back-end
No logging facility
Craft packet without credentials
Craft packet
Copy transmitted packet from other sensor node
Transmit packet

Figure 3.23: Attack tree for repudiation with applied countermeasures

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observe non-encrypted credentials</td>
<td>Encrypt traffic</td>
</tr>
<tr>
<td>Deploy malicious base station</td>
<td>Authentication of base stations by sensor nodes</td>
</tr>
<tr>
<td>Deploy malicious sensor node acting as a relay</td>
<td>Authentication of sensor nodes by sensor nodes</td>
</tr>
<tr>
<td>Brute force</td>
<td>Use strong cryptographic keys</td>
</tr>
<tr>
<td>Dictionary attack</td>
<td>Do not use common words as keys</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>4.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Outsider gets information on the topology of the network</td>
</tr>
<tr>
<td>Threat target</td>
<td>The platform network layout</td>
</tr>
<tr>
<td>Risk</td>
<td>Medium</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observe non-encrypted traffic</td>
<td>Encrypt traffic</td>
</tr>
<tr>
<td>Observe non-encrypted network headers</td>
<td>Encrypt the whole packet including headers</td>
</tr>
<tr>
<td>Deploy malicious base station</td>
<td>Authentication of base stations by sensor nodes</td>
</tr>
<tr>
<td>Deploy malicious sensor node acting as a relay</td>
<td>Authentication of sensor nodes by sensor nodes</td>
</tr>
<tr>
<td>Brute force</td>
<td>Use strong cryptographic keys</td>
</tr>
<tr>
<td>Dictionary attack</td>
<td>Do not use common words as keys</td>
</tr>
<tr>
<td>Analyze traffic patterns</td>
<td>Route or relay each packet over different network components</td>
</tr>
<tr>
<td>Analyze who communicates with whom</td>
<td>Route or relay each packet over different network components</td>
</tr>
</tbody>
</table>
Figure 3.24: Attack tree for the threat that an outsider gets information flowing in the network with applied countermeasures

<table>
<thead>
<tr>
<th>ID</th>
<th>4.3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Get traffic source location</td>
</tr>
<tr>
<td>Threat target</td>
<td>Location of sensor node or base station</td>
</tr>
<tr>
<td>Risk</td>
<td>Low</td>
</tr>
</tbody>
</table>

### Attack techniques

<table>
<thead>
<tr>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Observe non-encrypted traffic</td>
</tr>
<tr>
<td>Observe non-encrypted network headers</td>
</tr>
<tr>
<td>Deploy malicious base station</td>
</tr>
<tr>
<td>Deploy malicious sensor node acting as a relay</td>
</tr>
<tr>
<td>Brute force</td>
</tr>
<tr>
<td>Dictionary attack</td>
</tr>
<tr>
<td>Trace traffic back to source</td>
</tr>
<tr>
<td>Look for routing information</td>
</tr>
</tbody>
</table>

### 5. Denial of service

<table>
<thead>
<tr>
<th>ID</th>
<th>5.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Limit availability of a sensor node</td>
</tr>
<tr>
<td>Threat target</td>
<td>A sensor node</td>
</tr>
<tr>
<td>Risk</td>
<td>Low</td>
</tr>
</tbody>
</table>
Outsider gets information on the topology of the network

- Listen in on the network traffic
- Deploy malicious base station
- Deploy malicious sensor node acting as a relay
- Observe traffic
- Observe fully encrypted traffic
- Observe non-encrypted traffic
- Observe non-encrypted packet headers
- Analyze traffic patterns
- Guess decryption password
- Brute force
- Dictionary attack

Figure 3.25: Attack tree for the threat that an outsider gets information on the topology of the network with applied countermeasures
Figure 3.26: Attack tree for the threat that one can get the location of the traffic source with applied countermeasures
<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participate in network</td>
<td>Authentication mechanisms so only legitimate devices can participate</td>
</tr>
<tr>
<td>Deploy malicious sensor node acting as a relay</td>
<td>Authentication of sensor nodes by sensor nodes</td>
</tr>
<tr>
<td>Deploy malicious base station</td>
<td>Authentication of base stations by sensor nodes</td>
</tr>
<tr>
<td>Cause retransmission</td>
<td>Ensure the protocol does not try to re-send an infinite times</td>
</tr>
<tr>
<td>Relay traffic over node</td>
<td>Limit the amount of packets to relay</td>
</tr>
<tr>
<td>Flood the sensor node with new connections</td>
<td>Limit the number of possible connections from each sensor node</td>
</tr>
</tbody>
</table>

Figure 3.27: Attack tree for no availability of a sensor node with applied countermeasures

<table>
<thead>
<tr>
<th>ID</th>
<th>5.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Limit availability of a base station</td>
</tr>
<tr>
<td>Threat target</td>
<td>A base station</td>
</tr>
<tr>
<td>Risk</td>
<td>Low</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Participate in network</td>
<td>Authentication mechanisms so only legitimate devices can participate</td>
</tr>
<tr>
<td>Deploy malicious sensor node acting as a relay</td>
<td>Authentication of sensor nodes by sensor nodes</td>
</tr>
<tr>
<td>Deploy malicious base station</td>
<td>Authentication of base stations by sensor nodes</td>
</tr>
<tr>
<td>Flood the sensor node with new connections</td>
<td>Limit the number of possible connections from each sensor node</td>
</tr>
</tbody>
</table>
Figure 3.28: Attack tree for the no availability of a base station with countermeasures

6. Elevation of privileges

<table>
<thead>
<tr>
<th>ID</th>
<th>6.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Get sensor node privileges for malicious device</td>
</tr>
<tr>
<td>Threat target</td>
<td>Trust chain in the platform</td>
</tr>
<tr>
<td>Risk</td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get credentials of legitimate sensor node</td>
<td>See previous result in Spoofing identity</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>6.2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Threat description</td>
<td>Get base station privileges for malicious device</td>
</tr>
<tr>
<td>Threat target</td>
<td>Trust chain in the platform</td>
</tr>
<tr>
<td>Risk</td>
<td>High</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attack techniques</th>
<th>Countermeasures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get credentials of legitimate base station</td>
<td>See previous result in Spoofing identity</td>
</tr>
</tbody>
</table>

3.5.6 Step 6, Rate the threats

We base our rating on Microsoft’s DREAD methodology. DREAD gives us a threat risk rating for our identified threats and stands for Damage potential, Reproducibility, Exploitability, Affected users and Discovery. We use the following threat rating table from Microsoft’s DREAD web-page [29].
We also take into consideration the existing security prioritizations for this platform described in section 3.3.

When we count the values for each threat we assign them the rating high for values between 14-18, medium between 10-13 and low for 6-9.

We take the platform into consideration and look at the attack tree for the easiest path or paths for attacks on today’s platform. We also concentrate on software aspects. For example we will not take into consideration that physical jamming can be done when rating the denial of service threats since it lies outside the scope of this thesis.

**The threat table** We now create our threat rating table for each threat category. The author held a discussion for each threat category together with the developers of the platform. In this discussion we reasoned about the risk and what rated it should have. We gave the threats the rating 3 for high, 2 for medium and 1 for low.

**Spoofing identity**

<table>
<thead>
<tr>
<th>ID</th>
<th>Threat</th>
<th>D</th>
<th>R</th>
<th>E</th>
<th>A</th>
<th>D</th>
<th>O</th>
<th>Total</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Get credentials of legitimate sensor node</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>14</td>
<td>High</td>
</tr>
<tr>
<td>1.2</td>
<td>Get credentials of legitimate base station</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>13</td>
<td>High</td>
</tr>
<tr>
<td>1.3</td>
<td>Craft malicious packet</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>15</td>
<td>High</td>
</tr>
</tbody>
</table>

**Tampering with data**

<table>
<thead>
<tr>
<th>ID</th>
<th>Threat</th>
<th>D</th>
<th>R</th>
<th>E</th>
<th>A</th>
<th>D</th>
<th>O</th>
<th>Total</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Tamper with a packet in the radio channel</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>12</td>
<td>Medium</td>
</tr>
<tr>
<td>2.2</td>
<td>Influence data stored in the back-end</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>13</td>
<td>Medium</td>
</tr>
</tbody>
</table>
### Repudiation

<table>
<thead>
<tr>
<th>ID</th>
<th>Threat</th>
<th>D</th>
<th>R</th>
<th>E</th>
<th>A</th>
<th>D</th>
<th>O</th>
<th>Total</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Perform illegal operation which origin cannot be traced</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>14</td>
<td>High</td>
</tr>
</tbody>
</table>

### Information disclosure

<table>
<thead>
<tr>
<th>ID</th>
<th>Threat</th>
<th>D</th>
<th>R</th>
<th>E</th>
<th>A</th>
<th>D</th>
<th>O</th>
<th>Total</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1</td>
<td>Outsider gets information flowing in the network</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>13</td>
<td>Medium</td>
</tr>
<tr>
<td>4.2</td>
<td>Outsider gets information on the topology of the network</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>Low</td>
</tr>
<tr>
<td>4.3</td>
<td>Get traffic source location</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>Low</td>
</tr>
</tbody>
</table>

### Denial of service

<table>
<thead>
<tr>
<th>ID</th>
<th>Threat</th>
<th>D</th>
<th>R</th>
<th>E</th>
<th>A</th>
<th>D</th>
<th>O</th>
<th>Total</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Limit availability of a sensor node</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>8</td>
<td>Low</td>
</tr>
<tr>
<td>5.2</td>
<td>Limit availability of a base station</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>9</td>
<td>Low</td>
</tr>
</tbody>
</table>

### Elevation of privileges

<table>
<thead>
<tr>
<th>ID</th>
<th>Threat</th>
<th>D</th>
<th>R</th>
<th>E</th>
<th>A</th>
<th>D</th>
<th>O</th>
<th>Total</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.2</td>
<td>Get sensor node privileges for malicious device</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>14</td>
<td>High</td>
</tr>
<tr>
<td>6.3</td>
<td>Get base station privileges for malicious device</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>15</td>
<td>High</td>
</tr>
</tbody>
</table>

The observant reader might note that the threat rating for elevation of privileges is identical to first two spoofing identity threats. That is because the only sub goal of the above threats is that an attacker needs to obtain the credentials. The denial of service threats might seem to have gotten a lower rating than they should. This is because we consider many denial of service attacks, i.e physical jamming, to be outside the scope of this thesis and therefore will not weigh in as much.

The threats can be sorted as in Table 3.2 based on their total rating points.
<table>
<thead>
<tr>
<th>ID</th>
<th>Threat</th>
<th>Points</th>
<th>Rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>Get credentials of legitimate base station</td>
<td>15</td>
<td>High</td>
</tr>
<tr>
<td>1.3</td>
<td>Craft malicious packet</td>
<td>15</td>
<td>High</td>
</tr>
<tr>
<td>6.3</td>
<td>Get base station privileges for malicious device</td>
<td>15</td>
<td>High</td>
</tr>
<tr>
<td>1.1</td>
<td>Get credentials of legitimate sensor node</td>
<td>14</td>
<td>High</td>
</tr>
<tr>
<td>3.1</td>
<td>Perform illegal operation which origin cannot be traced</td>
<td>14</td>
<td>High</td>
</tr>
<tr>
<td>6.2</td>
<td>Get sensor node privileges for malicious device</td>
<td>14</td>
<td>High</td>
</tr>
<tr>
<td>2.2</td>
<td>Influence data stored in the back-end</td>
<td>13</td>
<td>Medium</td>
</tr>
<tr>
<td>4.1</td>
<td>Outsider gets information flowing in the network</td>
<td>13</td>
<td>Medium</td>
</tr>
<tr>
<td>2.1</td>
<td>Tamper with a packet in the radio channel</td>
<td>12</td>
<td>Medium</td>
</tr>
<tr>
<td>4.2</td>
<td>Outsider gets information on the topology of the network</td>
<td>9</td>
<td>Low</td>
</tr>
<tr>
<td>4.3</td>
<td>Get traffic source location</td>
<td>9</td>
<td>Low</td>
</tr>
<tr>
<td>5.2</td>
<td>Limit availability of a base station</td>
<td>9</td>
<td>Low</td>
</tr>
<tr>
<td>5.1</td>
<td>Limit availability of a sensor node</td>
<td>8</td>
<td>Low</td>
</tr>
</tbody>
</table>

Table 3.2: Threats ordered by their total rating points
Chapter 4

Developing the security solution

This part of the thesis will contain the process and design decisions for construction of our security solution. We begin with defining the security solutions focus in section 4.1. We then go through our hardware limitations in section 4.2 and continue with existing security solutions for wireless sensor networks in section 4.3. We show what security mechanisms were developed starting in section 4.4. In this last section we define our security levels, go through additional security issues and the new security policies that where developed as part of the security solution.

4.1 Security solution focus

From the results of our threat analysis we can draw some conclusions. To begin with we observe that threats regarding the ability to spoof or obtain credentials are our highest priority. They are followed by threats about the possibility for an attacker to craft packets and tamper or view sensitive data. Lastly, with the lowest priority, are threats concerning the possibility to limit the availability.

A wireless sensor network like ours needs to provide security mechanisms so the following list of security requirements are fulfilled according to Q. Wang [37].

1. Integrity, which ensures that message or the entity under consideration is not altered
2. Authentication, which ensures that the communication from one node to another node is genuine
3. Non-repudiation, which prevents malicious nodes to hide or deny their activities
4. Freshness, which implies that the data is recent and ensures that no adversary can replay old messages
5. Confidentiality, which provides the privacy of the wireless communication channels
6. Self-security, countermeasures may introduce additional hardware and software infrastructures into the network, which must themselves be secure enough to withstand attacks
7. Survivability, which ensures the acceptable level of network services even in the presence of node failures and malicious attacks
8. Availability, which ensures that the desired network services are available whenever required

Roughly checking what category our threat is appertaining to and sorting them according to their risk rank gives us the order of the list above. Spoofing identity, thus getting elevated privileges, violates both confidentiality and integrity and leads up to the possibility of violating non-repudiation. The threats related to crafting malicious packets violate the integrity, the authentication and the freshness of the platform. Performing illegal operations violates the non-repudiation as well while threats related to viewing data violate the confidentiality. Violating Self-security may involve stealing credentials from the onboard flash memory. We sort this lower than other spoofing identity threats since physical stealing lies somewhat outside the scope of the thesis. Our threats with the lowest risk rating violate both the survivability and availability.

The ordered list above gives us the security focus for our security solution.

4.2 Hardware limitations

With the specific hardware used in the platform we want to utilize the built in hardware support. For instance the microcontroller’s hardware support for AES encryption and decryption. By using the hardware for calculations when possible we save power and the computations are executed faster. Many other wireless sensor networks have much less powerful hardware to work with while our platform uses some of the latest hardware released on the market. This enables us to be able to use security mechanisms unavailable to other solutions. We begin with looking at the hardware support relevant to cryptology.

4.2.1 SAM4

Both microcontrollers have good support for cryptology and offer hardware countermeasures against differential power analysis attacks. They both provide AES with 128-, 192- and 256-bit keys and 3DES with a 64-bit key. The following modes of operation are available [3], [4].

1. Electronic Code Book (ECB)
2. Cipher Block Chaining (CBC)
3. Cipher Feedback (CFB)
4. Output Feedback (OFB)
5. Counter (CTR)

Both have secure hash algorithm (SHA) support. They provide SHA-1 and SHA-256 which are compliant with FIPS Publication 180-2 [52]. They also have a true random generator that provides a 32-bit random number every 84 clock cycles and has a throughput at up to 50 Mbit/s when clocked at 133 Mhz. The random number generator passes several test suites used to evaluate random generators including the NIST Special Publication 800-22 Tests suite [33] and the Diehard Random Tests Suite [4].
Both devices are also secure in the aspect of storing cryptographic keys. This is achieved by disabling the microcontrollers programmable JTAG-interface after all the cryptographic keys has been loaded. The only way to re-enable the JTAG-interface and thus being able to once-again read the information stored in the memory is to reset the device. However this resetting of the device will wipe all cryptographic keys in the process.

The lightweight variant of the two, SAM4L, has 128 to 256 Kbytes of embedded flash memory. Having the 128 Kbyte variant and using 256-bit keys we could store up to 4000 different cipher keys.

4.2.2 RF233

The transceiver has support for the AES cipher with a 128-bit key. It has two supported modes of operation.

- Electronic Code Book (ECB)
- Cipher Block Chaining (CBC)

However, while it can do both encryption and decryption in ECB mode it can only do encryption in CBC mode. The intention is to use ECB for encryption while CBC is used to calculate the message integrity code (MIC) [2].

It has a two bit random number generator but it is much poorer than the ones found in the SAM4 microcontrollers. RF233 provides the same secure storage of cryptographic keys as the SAM4 microcontrollers.

4.3 Existing security for wireless sensor networks

Several security solutions for wireless sensor networks exist. Most of the current security solutions are developed to be quite general to fit many different platforms and scenarios. In this thesis we have the possibility to choose the best from the existing solutions to fit our needs. We are able to fine tune our security solution to a greater extent since we will not need to be able to apply it to several different platforms or hardware. We begin with looking at some of the existing security architectures crafted specifically for Contiki and TinyOs. These are mainly architectures giving protection at the data link layer. Since we want to put together a full security solution we also look into some other multi level security variants. Common to all of these existing security solutions is that they either miss out on our security requirements or that they dont fit with the hardware in the platform. Especially the requirement that they should consume very low power is a problem with many existing security solutions due to their high complexity and expensive cryptographic operations. Further, most existing security solutions does not support the hardware and software used in the platform. One hardware issue is that the platform treats the physical layer and the data link layer in the transceiver while the remaining network layers are treated in the microcontroller. Most security solutions assume that all layers will be treated in the same place. One software issue is that almost all existing security solutions are written for the TinyOS operating system and not Contiki which we are using in our platform.
4.3.1 TinySec

We begin looking at TinySec. TinySec is a security architecture currently integrated into TinyOS. TinySec was introduced in 2004 and resides at the data link layer. It provides two different modes of operation. These are either authenticated encryption or authentication only. It uses the SkipJack block cipher with the cipher block chaining (CBC) mode of operation. To ensure that the ciphertext length is the same as the plaintext they use CBC in conjunction with ciphertext stealing (CS), this is usually written as CBC-CS. CBC-MAC is used for authentication and integrity. The values, destination, active message (AM) handler type, Length, source and counter are used as the IV to the encryption function. The counter is incremented for each encryption and 2 bytes long. It offers no replay protection but does have semantic security thanks to the uniqueness of the IV. A network wide key is used and shared between all devices. The overhead added in each packet is 8 bytes, 4 bytes for the MIC and 4 bytes for source and counter fields. For more information on TinySec see Karlof et al. [22].

4.3.2 MiniSec

Released in 2007 MiniSec claims to address and resolve several drawbacks with other security architectures proposed for TinyOS. It has two modes of operation, one for unicast and one for multicast. Both modes provide replay protection using respectively either synchronized counters or a bloom filter. Like TinySec it uses the Skipjack cipher but with offset code book (OCB) as mode of operation. OCB gives both authenticity and data secrecy while avoiding ciphertext expansion. MiniSec provides weak data freshness and a synchronized counter is used as the IV. It only adds 3 Bytes of overhead per packet and is thus very effective packet-wise. For more information on MiniSec see Luk et al. [27].

4.3.3 TinyECC

TinyECC was one of the first successful attempts to provide public-key cryptography in wireless sensor networks. It uses elliptic curve cryptography which has a shorter key size than traditional public-key cryptography schemes. The shorter key size results in faster computations, lower power consumption and saves on both memory and bandwidth. For more information on TinyECC see Liu and Ning [26].

4.3.4 ContikiSec

The first, and currently only, security solution aimed at Contiki. ContikiSec was developed in 2009 and provides two modes of operation like TinySec. The first mode provides authenticity and data secrecy while the second only provides authenticity. ContikiSec uses 128-bit AES as cipher with OCB for the first mode and CBC-MAC for the second. They use a 2 Byte long randomized IV for each packet. For more information on ContikiSec see Casado and Tsigas [9].
4.3.5 ZigBee

The ZigBee standard provides multiple security levels. ZigBee uses the security suite provided by IEEE 802.15.4 at the MAC layer and defines its own security model in the network and application layer. IEEE 802.15.4 provides access control, data encryption, frame integrity and sequential freshness at the MAC-layer. Protocols supplied by ZigBee include methods for key establishment, key transport, frame protection and device management. It uses three keys for different purposes. They are a master key, link key and network key. The master key is installed in each device first and is the basis for long-term security between a sensor node and a base station. The link key is the basis of security for communication between two devices while the network key is shared across the whole network. The cipher used is AES with 128-bit keys. The mode of operation is CCM which provides both authenticity and data secrecy. CCM is the same as CCM but with the possibility to either have encryption-only or integrity-only. CCM stands for Counter with Cipher Block Chaining-Message Authentication Code and uses counter mode (CTR) for encryption and CBC-MAC for authenticity. According to Housley et al. it is not a problem using the same key for both the encryption and authenticity operations. ZigBee uses a synchronized counter as the IV. For more information on ZigBee see Masica [28].

4.3.6 WaspMote

Libelium is the company behind the WaspMote and they provide an encryption library providing a multi-level security suite built on the existing ZigBee security. They offer authenticity, confidentiality, data freshness, replay protection, non-repudiation, key renewal processes and key management. They use 128-bit AES at the data link layer and 256-bit AES at the application layer. The key renewal process uses 1024-bit RSA public key cryptography. For more information on WaspMote see Libelium [23].

4.4 Our security solution

In this part of the thesis we will present our security solution. We provide multiple security levels in our security solution for two reasons. The first reason is that some deployment scenarios may require higher security than others, thus the security level should be tunable. Secondly, if an attacker is able to breach our first line of defense the damage the attacker could cause should be limited. In addition to these three security levels we develop 20 new security policies. These new security policies ensure that the security solution is constantly secure during its operation time.

The bytes sent in each packet needs to be kept at the smallest amount possible. In the thesis System Architecture Directions for Networked Sensors [17] they analyze the older 8-bit microcontroller Atmel 90LS8535 with the RFM TR100 transceiver. They show that with that specific hardware the cost in power consumption for sending one bit equals that of executing 800-1000 instructions on the microcontroller [17]. The authors of ContikiSec show that sending one byte has nearly the same power consumption as calling the software AES encryption function six times on the MSB-430 mote [9].
4.4.1 Security levels

We have chosen to provide three different modes of security. These security levels will be called Low, Medium and High. We start with identifying where these different security levels should be placed. Looking at the existing security solutions we can identify that most single-layer security architectures are present at the data link layer. Other security solutions, for instance ZigBee, provide multi level security by using security architectures at both the data link layer and layers higher up in the stack. We use the same design and construct the first security layer at the data link layer and the second at the network layer, see Figure 4.1. This design gives us greater flexibility than to use one single layer. For instance, the data link layer will ensure that the communicating parties are authenticated before being able to participate in the network. At the same time neighboring nodes cannot decrypt the network layer data payload being sent if they do not share a key.

![Figure 4.1: The security in the network stack](image)

We primarily use the transceiver for implementing the low security level while we use our microcontrollers for implementing medium and high.

4.4.2 The low security level

Using a bottom up approach we begin with describing the lowest security level in our security solution. This will be our first line of defense. We specify that the implementation of low is mandatory when implementing medium or high.

Security placement We place the first line of security at the data link layer like TinySec and ContikiSec. Since the primary mode of communication will be many-to-one over a multi-hop topology we want to apply security at a low level so an adversary for instance cannot inject false routing information. TinySec provides further motivations for applying security at the data link layer in Karlof et al. [22]. We also want our security to be transparent to applications and protocols running higher up in the network stack. However, both ContikiSec and TinySec
merely provides confidentiality for the payload in the MAC frame. We provide confidentiality for the whole MAC frame instead. Some authors argue that replay protection does not belong at the data link layer while others argue that it does Jinwala et al. [20], Karlof et al. [22]. Since the higher layer will implement replay protection we decide to not implement it at this level. This means that replay protection will only be offered for the medium and high security levels.

Selection of cipher When selecting cipher to use we first need to determine if the low security level should be implemented in the microcontroller or the transceiver. Currently our platform handles the data link layer and the physical layer primarily in the transceiver. This was a design the developers of the platform wanted to keep since the transceiver has support for packet processing and other features which they utilize. Therefore we will be implementing the low security level with regards to our hardware constraints and features in the transceiver. As long as we want to utilize the functionality provided by our transceiver for message processing our security solution possibilities are limited.

The RF233 transceiver provides us with 128-bit AES as the only cipher alternative. We use this built in AES symmetric block cipher for encrypting the MAC frame and verifying its authenticity. Another alternative design would be to move the low security level to a layer higher up in the network stack; a layer that would be processed in the microcontroller. However this alternative design is not preferred with the same reasoning as why we placed security in the data link layer in the first place.

Mode of operation The mode of operation determines the repeated and secure use of our AES cipher. Different modes of operations have different prerequisites. The RF233 transceiver only provides us with ECB and CBC as modes of operation. Therefore on the data link layer we are limited to use ECB for encryption and decryption while the MIC is calculated with CBC, this is called CBC-MAC. Figure 4.2 illustrates how ECB works. Atmel provides the following statement [2]:

"ECB decryption is not used by either IEEE 802.15.4 or ZigBee frame security. Both of these standards do not directly encrypt the payload, but rather a nonce instead, and protect the payload by applying an XOR operation between the resulting (AES-) ciphertext and the original payload. As the nonce is the same for encryption and decryption only ECB encryption is required."

![Electronic Codebook (ECB) mode encryption](image)

Figure 4.2: Electronic code book. Source: Wikipedia [45]

However ECB has some weak aspects. The first is that identical plaintext encrypted with the
same key will produce identical ciphertext, thus it does not offer any good message confidentiality \[13\]. It is also very vulnerable to replay attacks. Further, ECB is a block cipher and requires that each block to be encrypted consists of 16 bytes. This can lead to unnecessary overhead and waste of energy. Assume we only want to send a packet of 5 bytes. We would then need to add 11 bytes of padding before doing our ECB block encryption. This necessitates us to use some padding scheme in conjunction with a technique called ciphertext stealing (CS). Public-key cryptography standard (PKCS) #7 is defined in RFC 5652 \[18\] and recommended by RSA Laboratories. PKCS#7 is a byte padding scheme where the value of each byte added are of the same value as the number of padding bytes. We decide to use ciphertext stealing together with PKCS#7. Our padding scheme will be used when we have less than 16 bytes of data to encrypt. Ciphertext stealing will be used otherwise and makes it possible for the encrypted text to be the same length as the plaintext at the cost of some extra complexity. We will call this ECB-CS. ECB-CS works by doing ordinary encryption up to the last n-1 blocks. The n-1 and n:th block are specially treated, more details on how this works can be found in WYSEUR \[10\]. One drawback of this technique, aside from the extra complexity, is that a bit corruption in the penultimate ciphertext block will cause corruption of both the n:th and n-1:st plaintext block.

**Message integrity code**  CBC will be the mode of operation used to calculate the MIC since it is the only one offered. The MIC is calculated over the encrypted MAC frame and transmitted in the clear. CBC-MAC works by using the ordinary CBC mode of operation with a zero IV and the secret key.

We decide to do this encryption then MIC like ContikiSec, TinySec and MiniSec does. With this method we can reject packets immediately since the MIC verification is done before decryption. Another benefit of the selected method is that the MIC is derived from ciphertext instead of from plaintext. An attacker can also not attack the key used for encryption of plaintext directly but instead has to attack the MIC which in turn is derived from ciphertext, thus revealing less. If we should change to MIC then encryption we would save up to 4 bytes when we need to do padding. However we choose to use encryption then MIC because of the previous statements and since the situation that we need to do padding should not occur that often.

CBC-MAC is not secure for variable-length messages. Bellare et al. \[5\] have proposed three alternatives to counter this. Since we do not want to require yet another 128-bit key we discard the recommended method in their thesis. Instead we decide to use what they call the length-prepending method. With the length-prepending method the length of the message is prepended to the message before CBC is called. It is further described by Jonathan Katz and Yehuda Lindell in their book *Introduction to Modern Cryptography* \[21\]. This method requires that the length is known in advance. In our platform we will always know this length in beforehand so this method fit our needs. CBC-MAC with the use of the length-prepending method can be seen in Figure 4.3.

TinySec \[22\] and MiniSec \[27\] reach the conclusion that a MIC of 4 bytes will offer sufficient protection in a wireless sensor network. For us these 4 bytes will be the lowest 4 bytes from the last block of the CBC-MAC output since the last block output depends on all previous blocks. Even though we have data speeds up to 2Mbit/s we are secure enough with a 4 Byte MIC. Assume a typical packet is 100 bytes and we send at our maximum data rate of 250 Kbyte/s (or 2 Mbit/s). It would then take an adversary a little more than one week to be successful in forging a false MIC for a particular packet. However, the only way for an attacker to know if his crafted MIC was valid is to send it to a legitimate sensor node \[22\]. This makes it easy to detect
if an attack of forging false MIC is going on since very much data will be transmitted. We also specify that our key renewal scheme will occur more often than this worst case scenario in a new security policy, policy [2]. To use 6 bytes is also too overcompensate; we therefore decide to use 4 bytes for our MIC.

**Keying** Keying is always an issue for security in wireless sensor networks. Should all devices share a network wide key or share a unique key for each and every device in the network? We will not provide any advanced keying mechanism but propose a simple and straightforward one fitting with our design.

Each sensor node in the network has a unique joining key shared with the base stations. When a sensor node wants to join the network it first needs to authenticate. All authenticated devices will share a 128-bit symmetric network wide key, called the network key. This key which is shared between all devices enables communication between the devices at the data link layer. Both keys are themselves made up of two 128-bit symmetric keys. One of the keys is used for the encryption using AES-ECB-CS and the other one is used for the authentication using AES-CBC-MAC. Figure 4.4 shows the steps carried out when joining the network.

We use the joining key to make it harder to compromise the entire network by stealing a node. If an attacker were to steal a sensor node the attacker cannot extract the network wide key. If the attacker is successful in obtaining the symmetric joining key from the onboard flash memory, which in itself is very unlikely, the attacker would need to re-visit the network and do a normal
authentication to obtain the currently used network key.

**Re-keying** The re-keying procedure is uncomplicated. We simply encrypt the new network key with the old and distribute it to all sensor nodes. Likewise we encrypt the new joining key with the old and send it out to the sensor node in question. We construct two new security policies for re-keying. The first one is a security policy specifying the re-keying interval, policy 2. The second one is policy 4 that defines what should happen to sensor nodes that miss out on this re-keying window.

**Rejoining** A sensor node might need to rejoin the network for different reasons. An example would be if the sensor node was moving inside the network and found a stronger base station to which it would connect. With the low security level the sensor node does not need to re-launch the joining procedure. Since all base stations and sensor nodes share the current network key the sensor node can immediately start communicating with the new base station.

**Setup** The first sets of joining keys are setup and preloaded to the devices before they are distributed for use. The initial network key to use is determined when the security solution is installed. During the security solutions lifetime both the joining and network keys will change. All base stations need to know the unique joining key for each device. In a scenario with many sensor nodes, let us assume up to 20 000, we store the keys for each sensor node in the back-end. For smaller networks each joining key could be stored in the base stations. This decision will be left open to those who will implement and deploy the security solution.

**Packet format** The low security level will have some additional overhead for each packet; namely the MIC. Since the MIC provides the same functionality as the FCS field in the current MAC-header we can remove it and instead add our MIC. This results in a 2 Byte additional overhead for each packet.

One scenario exists where we do have additional overhead in excess of the MIC for the low security level. This happens if the data fed to the low security level is less than 16 bytes. In this situation the PKCS #7 padding scheme will be applied; thus padding the data until it is 16 bytes in size. A worst case scenario would be if the payload to send would be 1 byte in size. This would result in sending 12 padding bytes (16 bytes minus MAC header and payload). However this situation is unlikely since the network-layer and above layer headers combined with the payload typically are larger than 16 bytes.

**MAC frame**

<table>
<thead>
<tr>
<th>Octets</th>
<th>Name</th>
<th></th>
<th></th>
<th>Name</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Frame control</td>
<td>1</td>
<td></td>
<td>Sequence Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Frame payload</td>
<td></td>
<td></td>
<td>MIC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.1: Gray fields are encrypted and authenticated

**Summary** The Low security level offers the first line of defense in our security solution. It makes sure that only authenticated devices can exchange data and participate in the network. It uses 128-bit AES with the ECB-CS mode of operation for encryption and CBC-MAC for
authenticity. All cryptographic operations are carried out in hardware with the use of the RF233 transceiver. This security level does not offer any non-repudiation or replay protection by itself.

The low security level uses four different 128-bit symmetric keys. These are,

1. A joining key for encryption
2. A joining key for authenticity
3. A network wide key for encryption
4. A network wide key for authenticity

We combine and call key 1 and 2 the joining key while key 3 and 4 is called the network key.

<table>
<thead>
<tr>
<th>Encryption 128-bit AES (Hardware)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network data</td>
</tr>
</tbody>
</table>

Table 4.2: The low security level protection

Table 4.2 show how the payload is secured by the low security level. The additional packet overhead introduced by the low security level is 2 byte. This is equal to a 1.5% of the maximum 127 byte frame size.

4.4.3 The medium security level

The medium security level tightens up the security from the low security level. However we do rely on the protection in the data link layer, or low security level, to secure for instance routing information.

Security placement We implement the medium security level at the network layer. We place it at the network layer mainly because we want to provide end-to-end encryption for application data and authenticity for network headers. We need to ensure that routing and other network layer related data will be able to flow between devices in the network even if they do not share a cryptographic key.

Selection of cipher With our medium security level we have much greater freedom when developing the security unlike at the low security level. The SAM4 microcontrollers provide two ciphers for us, 3DES and AES. Since AES is recommended by NIST [31] and other works like ContikSec comes to the conclusion that AES is the recommended cipher to use [9] we select AES over the aging 3DES.

Mode of operation On the SAM4 microcontrollers we have five different modes of operation to choose from. In related work the most commonly used are CCM, OCB and CBC. Different works argue for one or another mode of operation and we will use some of the reasoning that can
best be applied to our platform. First of all we drop OCB since we have no hardware support for it. OCB is also under various patent claims while CCM and CBC are patent free. A comparison between OCB and CBC can be found in Ferguson et al. [13]. CCM uses CTR for encryption and CBC for MIC, more details on CCM can be found in Dworkin [10] and Housley et al. [19]. An interesting walkthrough of CCM (used by IEEE 802.15.4) and other modes of operation provided by Atmel can be read in Atm [1].

We have to evaluate CBC against CTR which is used for encryption in CCM. Each mode has some advantages and disadvantages over the other one. In TinySec the authors argue that CBC is the mode of operation to use. They state that since we use CBC for authenticity we already have it implemented and thus should also use it for encryption to minimize the code needed. They also states that the CTR mode is a stream cipher mode of operation and therefore have some weaknesses. This weakness or drawback is that if the same IV ever is used to encrypt two packets it is often possible to recover both plaintexts [22]. We do not have any implementation restriction on code size and later works come to other conclusions on what mode of operation to use Luk et al. [27], Casado and Tsigas [9]. CBC is also dependent on a randomized IV for each packet and has a leakage issue when used with a counter as IV [22].

An extensive survey of block ciphers and their mode of operation have been done where the authors come to the conclusion that CTR is the recommended mode [23]. They argue that CTR is much more effective in regaining synchronization and therefore should be used. By using CTR we would cause the AES block cipher to act as a stream cipher. Stream ciphers are also generally faster than block ciphers and do not require any padding to the block length. CCM also includes replay protection by the use of a counter and can use the same key both for the CTR and CBC-MAC calculation [19]. Both the IEEE 802.15.4 standard and ZigBee uses CCM* as the mode of operation [11]. CCM* is a variant of CCM that offers encryption-only or authentication-only where CCM always has both encryption and authentication [28].

We decide to use CCM for our mode of operation. Since we can choose the IV length and apply a rekeying protocol before we need to repeat IVs we can ensure that the same IV never is reused with the same key. We select to use CCM and not CCM* since we always want to provide both encryption and authentication. Both operations are done in hardware so the extra power consumption is kept at a minimum. CCM also provides us with semantic security and weak freshness by design. Semantic security means that an eavesdropper cannot obtain any information on the plaintext even if the eavesdropper observes multiple encryptions of the same plaintext. Weak freshness means that we provide partial ordering of messages but carry no delay information [35]. Both these two properties are achieved through the use of our counter in CCM.

The underlying operations used by CCM needs an initialization vector (IV). We base our construction of the IV on Dworkin [10] and Housley et al. [19]. We remove the security level field and use the fields in the below table for the IV.

<table>
<thead>
<tr>
<th>Bytes</th>
<th>1</th>
<th>13</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Flags</td>
<td>Nonce</td>
<td>Length of message</td>
</tr>
</tbody>
</table>

Table 4.3: IV for CCM

The nonce in turn consists of the following fields.

This will provide us with a unique 16 byte IV for each node in the sensor network. The frame counter is incremented for each packet. With our 4 byte counter we can send $2^{32}$ packets before
the counter reaches its maximal value. Using the same reasoning as when we chose to use a 4 byte MIC in the low security level we show that our frame counter is large enough. We would be able to send at our maximum data rate of 2 Mbit/s with an average packet size of 100 bytes for almost 3 weeks before we would need to exchange the key used. We construct a security policy stating that re-keying shall occur before the frame counter reaches max, policy [8]. The counter is reset only after re-keying has occurred. Authentication for CCM can be seen in Figure 4.5 and confidentiality in figure Figure 4.6.

Message integrity code Since CCM is our selected mode of operation the MIC, called tag in CCM, will be calculated using CBC-MAC. A more detailed description of how CBC-MAC works can be found under the low security settings part where we use the same technique. However two important differences are made on this security level. The first one is that the IV is the same as to the encryption with CTR and not zero. This causes each MIC to be unique. The second difference from the data link layer MIC is that we provide authenticity for all network headers down to the MAC-header. See keying and packet format for the motivation on this design.

Other alternatives would be to use the built in support for hash functions in the SAM4 microcontroller or CMAC. Using a hash function would result in what is called a hash based message...
authentication code (HMAC). CMAC is the recommended cipher by NIST [11], however since we do not have hardware support in our microcontroller for CMAC we rule it out. However, we do have hardware support for both HMAC and CBC-MAC. We argue that since we want to have CCM we pick CBC-MAC over HMAC. This is no disadvantage; they are both secure as of today [24]. In ECRYPT [12] the authors state that frequent re-keying is required for CBC-MAC. We therefore construct a security policy for this, policy [8] SHA-1 is broken but as stated by Bruce Schneier in August 2005, “although it doesn’t affect applications such as HMAC where collisions aren’t important” [3] it would not affect our use of it. SHA-256 which is also provided by our hardware is still considered to be secure. CBC is provably secure, Bellare et al. [5].

Keying In the medium security level we add one more symmetric key. This key is used by each sensor node to obtain end-to-end encryption of data with a base station. We call this new key the End key. Each sensor node has a unique end key. This key is then shared with all the base stations.

The use of the end key adds extra security if an attacker is able to obtain the currently used network key. In that scenario an attacker cannot gain any information on the sensor data being sent, nor can the attacker tamper with any network headers above the MAC-header and go unnoticed. Any modification to any data authenticated will be detected by the base station when it tries to verify the second MIC. The base station can then send an alert to the administrators if this MIC failure rate goes above a given threshold; this is further specified in the new security policy [17]. A sensor node will not by itself detect if it is relaying data which has been tampered with in the above scenario. However the tampering will be detected when the aggregated data reaches the base station and, like before, an alert can be issued.

The usage of our end key together with the new MIC provides us with non-repudiation. This enables a base station to always being able to know from what sensor node the data originated.

The joining procedure looks like the one used in the low security level but with the new end key for encryption of data, this can be seen in Figure 4.7.

![Figure 4.7: Joining the network in the medium security level](image)

Re-keying, rejoining and setup The re-keying of the end key is done the same way as for the network and joining key in the low security level. Thus by encrypting the new end key with the old and sends it to the sensor node. The same goes with rejoining and setting up the keys. Rejoining is simple since all base stations already share the end key. The end key itself is preloaded during the same time as for the joining key for each device. Like with the other
keys in the security solution the end key will change many times during the security solutions lifetime.

**Packet format** The added security of our medium security level has an impact on the packet size. Each packet will have a second 4 byte MIC added and 1 extra byte for the counter. We do not want to transfer the whole 4 byte counter with every packet. Since we only look into unicast communication we use synchronized counters at both sender and receiver. Future security solutions might implement multicast communications. A suitable way to implement this would be with the use of bloom-filters \[20\]. With each packet we send the last 8 bits of the frame counters current value. This enables the receiving party to detect packet loss and easily recover from it at losses of up to 256 packets. The same technique is used in MiniSec \[27\] but with only 3 bits of counter being transmitted in each packet.

When using the medium security level we shall always provide authenticity for packets. If the data being transmitted is routing data, meaning that we have no application layer payload, the network data shall be authenticated. As a result we will have the MIC overhead for every packet being sent with the medium security level. This is so we are able to detect false routing or other tampering to network data with the medium security level.

The main disadvantage with ECB was that identical plaintext under the same key would produce identical ciphertext. In our medium security level the plaintext fed to the encryption will possibly never be the same due to our CCM in the application layer. Specifically even if we have no application layer payload the MIC will still be calculated thus adding differences for each message. We achieve this by placing the MIC first in the frame payload. This will cause the first block to almost always have a different plaintext under a given key. By doing this we get a better semantic security for our data link layer protection.

We thus arrive at the following packet format for the medium security level.

**MAC frame**

<table>
<thead>
<tr>
<th>Octets</th>
<th>Name</th>
<th>Frame control</th>
<th>Sequence Number</th>
<th>Frame payload</th>
<th>MIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Gray fields</td>
<td></td>
<td>Variable</td>
<td>Frame payload</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>Gray fields</td>
<td></td>
<td>Variable</td>
<td></td>
<td></td>
</tr>
<tr>
<td>variable</td>
<td>Gray fields</td>
<td></td>
<td>Variable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.5: Gray fields are encrypted and authenticated

**Frame payload**

<table>
<thead>
<tr>
<th>Octets</th>
<th>Name</th>
<th>MIC</th>
<th>Network headers</th>
<th>payload</th>
<th>Frame counter</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Variable</td>
<td></td>
<td>Variable</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4.6: Dark gray fields are authenticated. The light gray field is authenticated and encrypted

**Summary** Our medium level adds another level of security to our platform. This second security layer ensures that no other device participating in the network that overheard the communication can decrypt or tamper with the data. The medium security level adds additional 128-bit AES cryptography. This is placed in the application layer and the mode of operation is CCM. We provide encryption, authenticity, semantic security and weak freshness, thus replay protection. All cryptographic functions are carried out in hardware with the use of the SAM4 microcontrollers.
Even though we use the weaker ECB mode of operation in the low security level we still get a good level of security thanks to our use of CCM and the position of the second MIC.

The medium security level uses one new 128-bit symmetric key unique for each sensor node. This key is used for both encryption and authenticity in CCM.

- A end-to-end key

A visual representation of the medium security level can be seen in Table 4.7. The gray color indicates what is authenticated.

<table>
<thead>
<tr>
<th>Encryption 128-bit AES (Hardware)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network data</td>
</tr>
<tr>
<td>Encryption AES 128-bit (Hardware)</td>
</tr>
<tr>
<td>Sensor data</td>
</tr>
</tbody>
</table>

Table 4.7: The medium security level protection

We consider network data to be all layers from the data link layer up to the application layer. We are able to provide data authenticity for all devices sharing the end-to-end key.

The medium security level adds an extra 5 bytes overhead for each packet. This results in a total of 9 bytes overhead for each packet which equals 7% of the maximum packet size.

4.4.4 The high security level

The high security level replaces the medium security level when used but still builds upon the low security level.

Security placement. The high security level is implemented in the network layer like the medium security level it replaces.

Selection of cipher. We use the same cipher as with the medium security level but with an improved key length. Instead of using 128-bit AES we tune up the security with the use of 256-bit AES.

We also introduce the use of public-key cryptography. The public-key cryptography is used for secure key exchanging; more on this can be read in the keying section. Traditionally public-key cryptography, or asymmetric encryption, has not been feasible in wireless sensor networks due to our resource constraints. However, recent works done on elliptic curve cryptography (ECC) suggests that ECC is suitable to use in WSN [26, 39]. Traditional public-key cryptography, like RSA, is based on either the mathematical hardness of factoring or the discrete logarithm (DLOG) problem. Since both problems have certain mathematical properties the key size needs to be longer to provide sufficient protection. A table showing the key size equivalence from the ECRYPT II Yearly Report on Algorithms and Keysizes [12] can be seen in Table 4.8.

Since we want to provide 256 bit of security we need to use either 15424 bits with RSA/DLOG or
Table 4.8: Security in bits with different key sizes

<table>
<thead>
<tr>
<th>Security (bits)</th>
<th>RSA/DLOG</th>
<th>ECC</th>
</tr>
</thead>
<tbody>
<tr>
<td>48</td>
<td>480</td>
<td>96</td>
</tr>
<tr>
<td>56</td>
<td>640</td>
<td>112</td>
</tr>
<tr>
<td>64</td>
<td>816</td>
<td>128</td>
</tr>
<tr>
<td>80</td>
<td>1248</td>
<td>160</td>
</tr>
<tr>
<td>112</td>
<td>2432</td>
<td>224</td>
</tr>
<tr>
<td>128</td>
<td>3248</td>
<td>256</td>
</tr>
<tr>
<td>160</td>
<td>5312</td>
<td>320</td>
</tr>
<tr>
<td>192</td>
<td>7936</td>
<td>384</td>
</tr>
<tr>
<td>256</td>
<td>15424</td>
<td>512</td>
</tr>
</tbody>
</table>

512 bits using ECC. The use of ECC will result in faster computations, lower power consumption as well as in memory and bandwidth savings [16]. The only downside is that we have no hardware support for doing asymmetric encryption. Thus all encryption and decryption operations need to be done in software on the microcontrollers. This will not be a problem in our platform since we have hardware well capable of carrying out the needed operations.

Mode of operation  We decide to use CCM for our mode of operation with the same motivation as given for the medium security level.

Message integrity code  CCM implies the use of CBC-MAC. With the same reasoning as in our medium security level we use CCM with a 4 byte MIC.

Keying  Adding the use of public-key cryptography changes the keying procedure. In excess of the symmetric keys used in the low security level we now add public and private keys. Each sensor node and base station has a private key which it keeps to itself and a public key. The sensor node’s public key is shared with all base stations. Each base station share its public key with all sensor nodes. The software joining procedure can be seen in Figure 4.8.

Figure 4.8: Joining the network in high security level

The sensor node and base station first use the joining key to exchange the system wide network
key. The sensor node then generates a new symmetric key to be used. This key is generated by the use of the true random number generator found in the SAM4 microcontroller. The sensor node first encrypts this key and a nonce with its private key. It then encrypts it again using the selected base stations public key and sends it to the base station. Only the base station has the befittingly private and public keys and can thus decrypt it. The base station then re-encrypts the nonce with the new secret key and sends it to the sensor node to confirm that the new key will be used in future communications. Data will now be exchanged using the new shared secret symmetric key called secret1 in Figure 4.8. More details on how to exchange the symmetric key with the help of ECC can be found in Brown [8].

The high security level provides non-repudiation by the use of the secret key and the MIC. A base station will always be able to tell from which sensor node the data came.

**Authenticity** There is a need to provide authenticity of all the data during the key exchange procedure of the secret key. Since we do not yet have the symmetric key we cannot use our ordinary MIC procedure. Instead we use digital signatures to sign the data. In our joining example we would use digital signatures to sign step 3 and 4. Signature schemes with ECC are further described in Brown [8] and ECRYPT [12].

**Re-keying** The re-keying for the secret key is done the same way as for all the other keys. We encrypt the new symmetric key with the old. There is no need to go through the much heavier process of exchanging a new symmetric key with the use of public-key cryptography. The reason is of course that we already have a secure symmetric key in place to our selected base station.

**Rejoining** The high security level is somewhat different when it comes to rejoining than the other two levels. While the low security level used beneath this level behaves the same way as before we need to specially treat this level. The high security level exchanges a random key with one specific base station, thus it needs to re-launch the whole joining procedure when rejoining. This is needed in order to exchange a new secret key with the new base station.

**Setup** Like in the low security level we preload each device with its corresponding keys. The base stations will use the more powerful SAM4E microcontroller. However, if we would assume the maximum of 20 000 nodes, each with a 512-bit public key, we would need 1280 Kbytes of memory to store the keys. This is infeasible so instead we store all public keys in the back-end. A base station can than ask the back-end for the corresponding public key when needed. The sensor nodes cannot ask the back-end for keys in the same way as the base stations. However, since we know there will be at highest 256 base stations in a deployed system we can store all public keys for the base stations in each sensor node. This worst case scenario would result in 16.4 Kbytes of memory needed for storing all public keys in each sensor node. The SAM4L which is the one with the smallest memory has a minimum of 128 Kbytes flash memory, thus more than enough. Since this worst case is rare and a more normal network would have less than half of this number of base stations we decide to use this solution.

**Packet format** Similar to the packet layout in medium security we will have a 5 byte overhead for the MIC and counter. Similar to our medium security level we will always have authenticity
of network headers even if we have no application layer payload, thus nothing to encrypt. The details of the packet format can be read in the medium security section of this thesis, section 4.4.3.

We have the following packet format for the high security level.

**MAC frame**

<table>
<thead>
<tr>
<th>Octets</th>
<th>Name</th>
<th>Octets</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Frame control</td>
<td>1</td>
<td>Sequence Number</td>
</tr>
<tr>
<td></td>
<td>variable</td>
<td></td>
<td>Frame payload</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>MIC</td>
</tr>
</tbody>
</table>

Table 4.9: Gray fields are encrypted and authenticated

**Frame payload**

<table>
<thead>
<tr>
<th>Octets</th>
<th>Name</th>
<th>Octets</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>MIC</td>
<td>Variable</td>
<td>Network headers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>payload</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Frame counter</td>
</tr>
</tbody>
</table>

Table 4.10: Dark gray fields are authenticated. The light gray field is authenticated and encrypted

**Summary**  The high security level offers the strongest security for our platform. It replaced the medium security at the application layer and builds upon the low security level. It uses strong 256-bit symmetric keys with equally strong 512-bit asymmetric keys. We apply a public-key cryptology scheme using ECC which is used for exchanging of a symmetric key. The cipher used is AES with the CCM mode of operation. CCM provides us with encryption, authenticity, semantic security and weak freshness, thus replay protection. The symmetric operations are carried out in hardware while the asymmetric operations are done in software with the use of the SAM4 microcontroller.

The same reasoning as given in the medium security level for why we are secure when using ECB in the lower layer applies.

The high security level uses a total of five new keys. The keys are one symmetric key and four asymmetric keys. The symmetric key is used to transfer data. The asymmetric keys are two private keys and two public keys. Each pair of public and private key is unique to each sensor node or base station.

- A secret symmetric key (generated)
- A private asymmetric key for each sensor node
- A public asymmetric key for each sensor node
- A private asymmetric key for each base station
- A public asymmetric key for each base station

A visual representation of the high security level when transporting data and for exchanging keys can be seen in Table 4.11 and Table 4.12.
The total overhead of the high security level has the same 9 bytes packet overhead as the medium security level. Thus this is equal to 7% of the maximum size of a frame.

### 4.4.5 Additional security issues

In this section we go through additional security issues that has not yet been treated.

**Nonce** We use a nonce in the joining procedure in the security solution. The nonce is a random value unique for each time a sensor node wants to join. We require our nonce value to be 128-bit long to match the security level provided. The nonce is used to provide freshness thus protecting against replay attacks and to ensure that it is the legitimate base station sending the response.

We construct the nonce of a timestamp (8 byte) and a random value (8 byte). We use this construction for two reasons. First we want to be able to tell the freshness of the nonce and since all devices are clock synchronized we can easily do this. Secondly we use the random number to get uniqueness if multiple joining packets are created. The time resolution should be given in microseconds.

Further a new security policy is created specifying how long this timestamp is considered to be fresh, policy 19. Yet another security policy states that an alert shall be raised if too many old joining requests are detected, policy 20.

Another alternative here would be to use a simple counter or random value. We discard the solution with a counter since it is not feasible in larger networks. This is because each sensor node would need to have this counter value synchronized with all base stations to prevent the replay attack from occurring in other parts of the network. Using a random value for the nonce is a normally used solution; we would only need to keep track of perhaps the latest value used. However with the use of a timestamp and random value we get the best from both worlds.

**Aggregation** The design of the aggregation protocol lies outside the scope of this thesis. The only requirement on the protocol is that it shall run in a layer between the data link layer and the application layer.
In the security solution each transceiver verifies the MIC before sending it to the microcontroller. This enables us to never try to do data aggregation until the MIC has been properly verified by the low security level. This stops certain denial of service attacks where an attacker tries to relay large data quantities over a sensor node forcing it to do aggregation in order to drain its power supply. If a payload secured with the medium or high security level should be relayed the sensor node cannot check its validity in the application layer since it does not have the shared end key. The data should then be aggregated according to some aggregation protocol and relayed to the base station where the data can be verified.

**Roaming**  We offer a basic roaming approach with the use of our network joining strategy. Each security level has its own joining procedure based on its keying material. The basic joining strategy is described in the keying part of the low security level. The medium and high security levels joining procedure builds upon the one used in the low security. Details on the additional steps for medium and high can be found in each respective keying part.

The general behavior is as follows. If a sensor node moves inside the network it will periodically poll for the strongest base station according to an existing protocol. If the sensor node senses that a stronger base station exist it should connect directly or launch the joining procedure to this new base station.

**Logging**  In our security solution we apply logging to ensure even better non-repudiation. The medium and high security levels provide non-repudiation of received packets. The low security level cannot provide the same non-repudiation due to the lack of unique keys for each sensor node. Given our resource constraints we cannot keep logs in the sensor nodes or base stations. We therefore introduce a logging facility to be placed in the back-end. We leave it up to those who will implement and deploy the security solution to decide on what to log. An example for our scenario would be to log the origin and time of each packet. This would be used to prove that the position given by a sensor node for a specific time in fact originated from the sensor node in question.

**Detection**  A broad concept in wireless sensor networks is detection of attacks. Like in all networks it is easy to detect a denial of service attack while it is almost impossible to detect if an attacker is passively listening in on data transmitted. Both the issue of node failure detection and intrusion detection remains an open research area, especially if the attacker has obtained valid credentials [37]. One technique used for trying to detect malicious devices and behavior is through traffic-analysis. For example an adversary carrying out a sink attack can most likely be detected by looking at the changes in routing information for a deployed system. Suddenly many sensor nodes will change behavior and relay through one certain sensor node when they should not.

Our security solution includes some policies used for detecting attacks. However these may not be sufficient to provide adequate detection against all different types of attacks. Therefore we recommend the deployment of an intrusion detection system (IDS) and traffic-analysis in the back-end for deployments with higher security requirements. We place it in the back-end since it does not have the resource constraints of our other devices and the tasks can be quite computation heavy.
**True random number generator** The true random number generators used in both the microcontroller and the transceiver require an initial seeding value. In this security solution we specify that all devices shall have a unique random initial seeding value loaded before distribution and deployment. Thus at the same time as we load the default cryptographic material. This is specified in security policy [4]. In the documentation for the hardware Atmel states that they requires a new seed if the key size changes. We do not currently support a change of the security level used during the security solutions runtime. Thus we will never need to change the initial random seed value.

### 4.4.6 New security policies

In this part of the thesis we will go through the additional security policies needed. The security policies constructed here ensure that the security solution is constantly secure during its operation time. The implementation of all policies given in this part of the thesis is mandatory if not otherwise specified. We begin with looking into what further security policies is needed for the low security level.

**Low** The first step is to be able to participate in the network. Thus our first policy will declare restrictions on how to obtain the network key. The policy ensures that a sensor node needs to be in range of a base station to start the joining procedure for obtaining the network key. Thus we will not allow any joining procedures by using other sensor nodes as relays.

1. The joining procedure shall only be initiated when in range of a base station.

Since we use a network wide key in the low security level which is used to calculate the MIC it will need to be changed in time. With our 4 byte MIC we will have to change MIC after about $2^{31}$ packets. Like before, if we are constantly sending at 2Mbit/s it will take little more than one week until we need to change our key. Most scenarios will have a lower data speed than this so our new security policy will be,

2. The security solution shall initiate a re-keying procedure for the network keys after a certain operation time $X$ of the deployed system.

$X$ is determined by how much data is being sent in the deployed system, thus a little more than 1 week if we send at 2Mbit/s. We say keys since we need one key for encryption and one for authenticity.

Policy [3] and [4] will also have to do with keys used in the low security level. We state that,

3. The security solution shall periodically initiate a re-keying procedure for changing the unique joining keys for each sensor node.

4. A time window $X$ shall exist in which sensor nodes will be given the new network key.
The fourth policy states that when we change the network key, every node will have a limited time window for obtaining it. If a sensor node is not present in the network and misses the time window for some reason policy 5 states,

5. A procedure shall exist and be applied for when a sensor node that has missed the distribution window of a new network key connects.

An example would be to require manual re-activation of the node or use the joining key to re-launch the joining procedure. We add an optional policy,

6. When a sensor node is outside of the radio range of the deployed system for a given time X, the sensor node will dispose of its network key.

The time X is specified according to the requirements of the deployed system in question. This policy adds security if our sensor node would be stolen or lost. Thus it would require an attacker to re-visit the network and perform an on-site extraction of the network key rather than an off-site extraction.

7. All received traffic which is not consistent with the specified packet format and encryption shall be dropped as soon as possible.

The above policy states that we shall drop all malicious or invalid data received as soon as possible. For instance the transceiver shall drop packets not encrypted with the network wide key or where the MIC verification fails.

Medium In our medium security level we need to have a key-renewal scheme like in our low security level. We thus add a policy for periodically renewal of our end key.

8. The security solution shall initiate a re-keying procedure for the end key after a certain operation time X of the sensor node.

Like before the time X is determined from how much data we send. However, this time we only need to look at the data being sent for each individual sensor node and not the whole network.

High We add a policy regarding the key renewal for our secret key.

9. The security solution shall initiate a re-keying procedure for the secret key after a certain operation time X of the sensor node.

This key renewal policy is identical to the one found in the medium security level but regards the shared secret key instead.
Common We have an optional policy stating that each node needs manual activation.

10. The security solution shall require manual activation of every sensor node before any attempt to initiate the joining procedure can be made.

For instance a security guard may be required to activate a sensor node each time it wants to connect to the network. This can be done by not allowing the specific joining key associated with the node until the manual process is completed.

One important keystone in our security solution is the inability for an attacker to read-out cryptographic material from our devices. We therefore construct a policy stating that all devices shall disable their JTAG-interface after the key-material has been loaded into the device.

11. A device shall disable its programmable interface after the cryptographic keys have been loaded.

The next policy makes it mandatory to wipe all keying material if the programmable interface is re-enabled.

12. All cryptographic keying material shall be wiped when the programmable interface is re-enabled.

We use keys of lengths from 128-bit up to 512-bit. However we need a key policy so the keys used are strong and truly random.

13. A cryptographic key used in the security solution shall be derived from a true random number generator.

Before we can use the true random number generators in the devices they need to be initialized with an initial seeding value. This seeding value should be obtained from a true random source.

14. The true random number generator shall be preloaded with a random seeding value

We need to add a policy dealing with the situation of an attack.

15. The security solution shall alert the overseer in case of a detected attack.

The overseer can then decide on the appropriate action to take. For instance, in case of a denial of service attack security personal can be dispatched to seize the equipment causing it. More fine grained policies handling attacks include,

16. The security solution shall have a procedure to initiate if a compromised sensor node is detected.
This might involve revoking the network key or require all sensor nodes to be manually refitted with a new network key.

17. The security solution shall have a specified MIC failure threshold X.

Policy 17 covers that if an attacker is attempting to forge a MIC the MIC failure will increase. If it goes above the defined threshold X an alert shall be raised. This simple policy is probably sufficient according to the authors of TinySec [22].

18. The security solution shall have a specified threshold X for the maximum number of retransmissions.

Policy 18 involves the issue with collision attacks and desynchronization. If an attacker carries out any of these attacks the retransmissions will increase drastically. One issue still remains involving the sensor node joining procedure. An attacker could replay old joining packets, thus we construct two new security policies covering this.

19. The security solution shall have specified maximum acceptance time X for a joining request.

This policy ensures that the devices will not process joining requests with a timestamp older than X.

20. The security solution shall have a specified threshold X for the maximum number of joining attempts by a sensor node during a given time Y.

The last policy covers that we will detect and alert when too many joining requests have been sent during a time span Y.
Chapter 5

Evaluating the security solution

This part of the thesis brings together all previous results to evaluate the developed security solution. Each one of the security levels uses several security mechanisms to enforce security. A security mechanism is responsible for enforcing a whole or part of a security policy. This is often achieved through the use of cryptography [15].

We begin with checking the security levels against our security requirements in section 5.1. We then move on to evaluate how the security levels enforce our policies, section 5.2, and how they secure our assets in terms of prevention, detection and reaction, section 5.3. We check how secure they are against known attacks and the cost to perform an attack in section 5.4 and section 5.5 respectively. We evaluate the low security level in section 5.6 and the medium and high security levels in section 5.7. We briefly tie back to our attack trees in section 5.8. In the last sections we go through the part of the security solution that where implemented, section 5.9, and the cost of using our security levels, section 5.10.

5.1 Security requirements

Two different lists of requirements are present in our thesis. The first list are the security requirements a security solution should fulfill given by Q. Wang [37] in their work on security in wireless sensor networks. The second list of requirements for our security solution was created in collaboration with the developers of our platform.

The requirements our security solution should fulfill are,

1. Integrity
2. Authentication
3. Non-repudiation
4. Freshness
5. Confidentiality
6. Self-security
7. Survivability
8. Availability

The additional security requirements we have are,

1. No false node shall be able to take part of the network
2. No false base station shall be able to take part of the network
3. No outsider shall be able to listen in on the network traffic
4. No outsider shall be able to tamper with traffic being sent in the network
5. No outsider shall be able to limit the availability of the network

Both lists can be said to concern the same issues. It is quite simple to translate requirements from one list into categories of the other. The first list gives a more fine-grained view of how good protection the different security levels provide. The second list gives answers to more direct and not so fine-grained requirements. For instance the second list gives answers closer to what a person without deeper knowledge in the security domain might ask. We decide to only go through the first list for each security level. We do this because our policies strongly reflect our own security requirements, thus those requirements will be covered in the platform policy part.

5.2 Platform policies

The policies we begun with in this thesis are closely coupled with our own requirements for the platform. This is not to be mixed up with the security policies we deduced later on when developing the security solution. These policies are enforced by all of our security levels. Proving that these policies are enforced by our low security level will be sufficient. Like previously stated this is because both the medium and high security level builds upon the low security level. We go through each policy and motivate what security mechanism is in place to enforce it.

1. Only an authorized and authenticated user with an associated sensor node is allowed to participate in the network.

2. Only company-configured base stations shall be able to participate in the network.

The use of the network and joining keys ensures that the first and second policy is enforced. Each device needs to pass the authorization process by the use of joining keys before participating in the network. Both base stations and sensor nodes will reject data that is not encrypted under either of these keys.

3. Only authorized parties shall be able to listen in on the network traffic to obtain information.
Our traffic will be encrypted with at least the network key. Thus only devices sharing the network key will be able to decrypt and take part of the data.

4. Only authorized parties shall be able to modify the content being transmitted in the network.

The message integrity code (MIC) will ensure that no tampering of the data has occurred in the channel. Devices sharing the needed key will be able to modify data and re-create a valid MIC. However, when we evaluate the security levels we will show that the low security level only offers the most basic protection for enforcing our policies.

5.3 Prevention, detection, reaction

The terms Prevention, Detection and Reaction (PDR) is about how our security solution protects our assets. We are interested in showing the protective measures our security solution provides for these three categories. When classifying protection measures one roughly distinguishes them in the following categories given by Dieter Gollmann in his book Computer Security [15].

- Prevention, taking measures that prevent your assets from being damaged
- Detection, taking measures that allow you to detect when an asset has been damaged, how it has been damaged, and who has caused the damage.
- Reaction, taking measures that allow you to recover your assets or recover from damage to your assets.

In section 5.6 and section 5.7 we show how the security levels protect the following list of our identified assets according to the above categories.

- Data from the sensor nodes
- Data from the base stations
- Stored credentials and system data

5.4 Protecting against known attacks

We relate back to section 3.4 known attacks on wireless sensor networks, in the beginning of this thesis for the different attacks. We go through and show which attacks were averted and which were not for each security level in section 5.6 and section 5.7. We continue this section of the thesis with going through two categories which contains attacks special to our platform. After removing the specially treated attacks we are left with a total of 14 attacks.
5.4.1 Subversion of a node attack

The subversion of a node attack is mitigated with two of our new security policies, thus for all our security levels. Security policy [11] and [12] ensures that if a device is stolen the keying material stored will still be secure. We will not be able to put any software mechanisms in place to ensure that a sensor node cannot get stolen. However by securing the cryptographic keying material we have partially mitigated the attack.

5.4.2 Attacks not countered

In this section we present attacks which we do not mitigate with any of our security levels.

Physical layer jamming, our security solution cannot fully mitigate this attack. We cannot solve the issue with malicious equipment occupying the radio medium with any form of software. The security solution may however alert when a jamming attack is occurring since this type of attack is easy to detect. The alert behavior is specified in security policy [13].

Link layer jamming, like with physical layer jamming we can only detect it and raise an alert.

Hello flood attacks, the underlying radio protocol will try to connect to the strongest base station. If an attacker were to perform this type of attack the sensor node may try connecting to it. This behavior is specified in the underlying radio protocol and we cannot affect this with our security solution. However, the underlying radio protocol can be modified to only try connecting a certain number of times before trying the next base station.

Desynchronization, this is caused in our network by disruption of the transmissions. In our network the only way to cause this is by disrupting the packet exchange. This can be done with radio jamming-equipment by an attacker. The effect of this will be retransmission since the sender will think the packet was lost. There is no software countermeasure to this since the disruption is caused by a type of jamming during the packet transmission. What we can do is detect multiple retransmissions of the same packet and raise an alert. This behavior is described in security policy [18].

5.5 Cost for attacker

This part discusses the cost for an attacker to attack a deployed system using our security solution. The numbers given in this section are taken from the ECRYPT 2 Yearly Report on Algorithms and keysizes [12]. The numbers given in this part of the thesis are on the strength of our selected type of cipher with its corresponding key length. Thus these numbers show how secure we are if an adversary were to attack our cipher only, not other parts or weaknesses in the security solution design.

The min security column given in Table 5.1 indicates protection for a few months. With the use of our 128-bit symmetric AES cipher we are well suited to be secure against well funded attackers. The high security level uses keys equivalent to 256-bits of security, thus we are very secure even against the most well funded adversary.
<table>
<thead>
<tr>
<th>Attacker</th>
<th>Budget</th>
<th>Hardware</th>
<th>Min security (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hacker[^1]</td>
<td>0</td>
<td>PC</td>
<td>58</td>
</tr>
<tr>
<td></td>
<td>&lt; $400</td>
<td>PC(s)/FPGA</td>
<td>64</td>
</tr>
<tr>
<td>Small organization</td>
<td>$10k</td>
<td>PC(s)/FPGA</td>
<td>69</td>
</tr>
<tr>
<td>Medium organization</td>
<td>$300k</td>
<td>FPGA/ASIC</td>
<td>69</td>
</tr>
<tr>
<td>Large organization</td>
<td>$10M</td>
<td>FPGA/ASIC</td>
<td>78</td>
</tr>
<tr>
<td>Intelligence agency</td>
<td>$300M</td>
<td>ASIC</td>
<td>84</td>
</tr>
</tbody>
</table>

Table 5.1: Budget for attackers and hardware used with the minimum security needed in bits

From the same report we have numbers on security levels.

<table>
<thead>
<tr>
<th>Security Level</th>
<th>Security(bits)</th>
<th>Protection</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>32</td>
<td>Attacks in &quot;real time&quot; by individuals</td>
<td>Only acceptable for audit, e.g size</td>
</tr>
<tr>
<td>2</td>
<td>64</td>
<td>Very short-term protection against small organizations</td>
<td>Should not be used for confidentiality in new systems</td>
</tr>
<tr>
<td>3</td>
<td>72</td>
<td>Short-term protection against medium organizations, medium-term protection against small organizations</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>80</td>
<td>Very short-term protection against agencies, long term prot. Against small organizations</td>
<td>Smallest general-purpose level, &lt;= 4 years protection (e.g. use of 2-key 3DES, &lt; 2^40 plaintext/ciphertexts)</td>
</tr>
<tr>
<td>5</td>
<td>90</td>
<td>Legacy standard level</td>
<td>3-key 3DES restricted to 10^10 plaintext/ciphertexts, &gt;= 10 years protection</td>
</tr>
<tr>
<td>6</td>
<td>112</td>
<td>Medium-term protection</td>
<td>&gt;= 20 years protection (e.g. 3-key 3DES)</td>
</tr>
<tr>
<td>7</td>
<td>128</td>
<td>Long-term protection</td>
<td>Good, generic application-independent. Recommendation &gt;= 30 years</td>
</tr>
<tr>
<td>8</td>
<td>256</td>
<td>&quot;Foreseeable future&quot;</td>
<td>Good protection against quantum computers unless Shor’s algorithm applies</td>
</tr>
</tbody>
</table>

Using the numbers above we obtain security level 7 for the low and medium security level offering protection for ≈ 30 years. The high security level offers 256-bit security thus security level 8, protection for the foreseeable future.

The reader should note that the numbers given in this part can be misleading. Normally an attacker tries to exploit other weaknesses of the system design than the cipher used. As the figures in this chapter show the cipher itself is very strong and normally not worth attacking.

5.6 The low security level

The low security level offers only the most basic communication security but at a low cost.

5.6.1 Security requirements

Of the first given list of requirements the low security level can only be said to offer,

- Integrity
- Authentication
- Confidentiality
- Self-security

[^1]: Skilled attacker with good computer knowledge.
[^2]: For example worms or trojans.
While it offers good integrity by the use of a MIC it does not offer good confidentiality or authentication. The confidentiality provided is weak because of the underlying ECB mode. Like described earlier, identical plaintext encrypted with ECB mode will produce identical ciphertext. The authentication is weak since it only ensures that devices communicating have the network key. This security level does not provide authentication of individual devices. The low security level does fulfill the self-security requirement. For this level we interpret self-security as the ability to safely store the cryptographic keys in the transceiver.

5.6.2 Prevention, detection, reaction

We begin with looking at our two assets, data from the sensor node and data from the base station. The low security level applies both encryption and some authenticity for data transmitted by a sensor node. Thus we prevent our assets from being damaged with the use of cryptology. In this case the damage can be that an attacker can observe the un-encrypted data. We have detection by the use of our message integrity code, thus detecting malicious changes. Our reaction is defined in security policy 15 through 18. They specify that an alert shall be raised and that it shall be raised when the MIC failure rate exceeds a given threshold.

Our last asset is stored credentials and system data. Preventing this asset from being damaged is through the use of our network key, thus encryption. Then malicious data will not be able to propagate through the deployed system and into the back-end since it needs to be encrypted with the network key. Detection and reaction is achieved the same way as for our other two assets, i.e by the use of the MIC and a MIC failure threshold policy.

5.6.3 Protecting against known attacks

Table 5.2 presented here shows the attacks we protect against with our low security level. We also discuss some details of each attack. The protection column indicates what level of protection is offered. Yes indicates that the attack is fully mitigated. Limited denotes that we offer some level of mitigation to the attack while No means that we do not mitigate the attack.

1. The use of AES encryption mitigates eavesdropping. However we do have weak confidentiality due to the use of the underlying ECB mode of operation.

2. The use of a network key ensures that proper authentication needs to be done before devices can participate in the network. Thus we mitigate the risk of an attacker crafting new packets to cause collisions. However, an attacker could still capture packets and perform a replay attack to cause collisions.

3. The network key and security policy ensure that we will not waste unnecessary power on processing any malicious packets. However, if an attacker obtains the network key the protection offered here will be circumvented. Again the attacker is also able to send previously captured valid packets in a replay attack.

4. No attempt is being made to hide traffic patterns.

5. We stop an attacker from being able to perform packet tracing by looking into aggregated data by encrypting the MAC frame.
<table>
<thead>
<tr>
<th>ID</th>
<th>Name of attack</th>
<th>Mitigation technique</th>
<th>Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Eavesdropping</td>
<td>Encryption</td>
<td>Limited</td>
</tr>
<tr>
<td>2</td>
<td>Collisions</td>
<td>Authentication</td>
<td>Limited</td>
</tr>
<tr>
<td>3</td>
<td>Resource exhaustion</td>
<td>Authentication</td>
<td>Limited</td>
</tr>
<tr>
<td>4</td>
<td>Traffic analysis</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Packet-tracing</td>
<td>Encryption</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Clock unsynchronization</td>
<td>Authentication</td>
<td>Limited</td>
</tr>
<tr>
<td>7</td>
<td>Spoofed, altered, or replay routing information</td>
<td>Authentication</td>
<td>Limited</td>
</tr>
<tr>
<td>8</td>
<td>Sybil</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>Selective forwarding</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Sinkhole</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Acknowledgment spoofing</td>
<td>Authentication</td>
<td>Limited</td>
</tr>
<tr>
<td>12</td>
<td>Flooding</td>
<td>Authentication</td>
<td>Limited</td>
</tr>
<tr>
<td>13</td>
<td>False data filtering</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>False data injection</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 5.2: Protections against known attacks with the low security level

6. We prevent the attacker from crafting time frames by our use of authentication. However a replay attack of a captured time frame is possible.

7. We do not offer any protection for replaying of information. We do however mitigate the spoofed and altered data attacks. The spoofed identity is mitigated with our joining key, an attacker needs a valid joining key of the device to spoof. The altering of data attack is mitigated with the use of our message integrity code (MIC).

8. The Sybil attack is mitigated the same way as with spoofing identity; by the use of our joining key.

9. Sensor nodes will only relay through other sensor nodes sharing the network key. Thus again the use of our joining key mitigates this attack.

10. To make the malicious device more attractive an attacker needs to be able to craft routing packets. This is mitigated through the use of our network key.

11. An attacker cannot craft new ACK-packets and send into the network. However, due to the lack of semantic security of our ECB an attacker could identify, capture and replay for instance MAC-layer ACK-packets.

12. An attacker needs to obtain the network key in order to flood a device with new connections. This is because the protocols that may have a state are protected by the security provided at this level. However the same replay problem exists where an attacker replays old packets.

13. A malicious device cannot act as a relay unless it has obtained the network key. Thus this attack is mitigated.

14. Much like the attack of false data filtering, but instead of corrupting data an attacker injects new data. This attack is mitigated through the use of our network key.
5.6.4 Summary

The low security level offers only the most basic security and limited difficulty for a more capable adversary to circumvent. From known attacks we get the following table illustrating its protective capabilities.

<table>
<thead>
<tr>
<th>Protects</th>
<th>Number of attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>6</td>
</tr>
<tr>
<td>Limited</td>
<td>7</td>
</tr>
<tr>
<td>No</td>
<td>1</td>
</tr>
</tbody>
</table>

Our table shows that we provide limited security against known attacks with the low security level. This level only satisfies 4 out of 8 of our security requirements specified. Thus the conclusion is that the low security level should only be used in situations where security is not important.

5.7 The medium and high security level

The medium security level has a tradeoff between higher security and additional power consumption. The high security level offers the strongest protection our security solution can provide. The main difference between the two is that the high security level provides stronger cryptographic keys and public-key cryptography. Since they are so similar from an evaluation perspective we will go through both security levels in this part of the thesis.

5.7.1 Security requirements

Both the medium and high security level satisfies all security requirements previously given. Thus both security levels provides the following,

- Integrity
- Authentication
- Non-repudiation
- Freshness
- Confidentiality
- Survivability
- Self-security
- Availability
Both security levels satisfy the same requirements as the low security level but with additions. First, the confidentiality and integrity is stronger due to our CCM in the upper layer and the placement of our second MIC early in the packet, thus providing a better level of semantic security for ECB. It provides strong authentication and non-repudiation by the use of end-to-end keys. Freshness is given by the use of CCM. Survivability is achieved through the two layered design. Even if a malicious device breaches the first line of defense its possibilities are limited. The self-security is extended from the low security level by offering protection against power analysis in the SAM4 microcontroller. We consider the requirement about availability to be satisfied by the use of our authentication. An attacker cannot craft packets, replay old packets or flood the nodes without the network key. However an attacker can still perform jamming of the radio medium used. Like previously stated we can only detect and alert that such an attack is occurring, not prevent it with the use of our software.

5.7.2 Prevention, detection, reaction

Both security levels use the same techniques for prevention, detection and reaction to protect our assets as the low security level. They do provide an improved level of security over the low security level.

We start with looking into the first two assets, data from the sensor node and data from the base station. We prevent the assets from damage through the use of cryptology like before. However we add a second level of security resulting in more prevention for application-layer data. The detection is improved by our second MIC that checks for attacks by an attacker that breached the first level of defense. We also have a clear origin for each packet, by the use of the end-to-end keying, and can easier detect malicious traffic. The reaction is the same as for our low security level.

The additional level of security also adds to our last asset, stored credentials and system data. We have additional prevention by our end-to-end encryption. Thus an attacker needs the network and the end key. Detection has improved security with the second MIC and the end key while prevention is the same through the use of our security policies.

5.7.3 Protecting against known attacks

Evaluating both security levels against our list of known attacks gives us an idea of how good the provided security is. We use the same proceeding as for the low security level. Comments are given for the attacks for which we have a different protection than at the low security level.
<table>
<thead>
<tr>
<th>ID</th>
<th>Name of attack</th>
<th>Mitigation technique</th>
<th>Protection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Eavesdropping</td>
<td>Encryption</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Collisions</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Resource exhaustion</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Traffic analysis</td>
<td></td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Packet-tracing</td>
<td>Encryption</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Clock unsynchronization</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Spoofed, altered, or replay routing information</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sybil</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>9</td>
<td>Selective forwarding</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Sinkhole</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Acknowledgment spoofing</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Flooding</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>False data filtering</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>False data injection</td>
<td>Authentication</td>
<td>Yes</td>
</tr>
</tbody>
</table>

1. The use of our AES-CCM encryption mitigates the attack of eavesdropping.

2. The usage of the CCM mode of operation together with our network and end key mitigates the collisions threat. The attacker cannot replay older or craft new packets.

3. The attacker needs the network key to make a sensor node accept its packets. An attacker cannot craft new packets nor replay old thus the risk is fully mitigated.

4. No attempt is being made to hide traffic patterns.

5. -

6. Authentication and replay protection mitigate the risk of an attacker crafting time frames resulting in this attack.

7. These types of attacks are mitigated through the use of our authentication and replay protection.

8. -

9. -

10. -

11. Mitigation by the use of authentication and replay protection.

12. An attacker needs the network key to craft packets or replay old ones. Thus this is mitigated through our need for authentication and use of replay protection.

### 5.7.4 Summary

The medium and high security level offers good security. From known attacks we get the following table illustrating its protective capabilities.
The table shows that both levels provide a much better security than the low security level alone. Both security levels fully mitigate 13 of the known attacks which previously had only limited protection. They both satisfy all eight security requirements. Thus the medium or high security level should be applied in systems where security is in focus.

### 5.8 Attack trees

In Appendix C we go through and evaluate our security solution against the attack trees derived in section 3.5.4.

### 5.9 Implementation

Part of the security solution developed in this thesis was implemented in the Contiki operating system. In this part of the thesis the outcome of the implementation will be evaluated.

**Limitations** The goal of the implementation was to observe how the security solution behaves in practice. The security solution was implemented using the Cooja-mote and simulated in the Cooja simulator. The Cooja simulator tries to mimic the real world as good as possible. Therefore our Cooja mote should perform the same way in the Cooja simulator as in reality. Due to the limited time available we only implemented confidentiality and authenticity for the low and medium security levels. For the same reason we did not implement any of the specified keying protocols but instead preloaded all devices with the needed keying material.

There were some major differences between the real platform and the Contiki platform we used for the implementation. The first major difference was the radio communication protocol. Our Contiki platform used the ordinary CSMA-CA protocol. This means that any device could communicate at any time in contrast to the real platform using TDMA. We used CSMA-CA since the implementation of the TDMA in Contiki we should have used was not yet finished. The second major difference was that the Cooja mote radio driver did not implement and use any CRC check, thus all data running without our security was 2 bytes smaller than it should have been.

**Result** To demonstrate how the data being transmitted changed with different security levels applied we created a simple simulation. The simulation consisted of a base station and a sensor node sending data. The sensor node was programmed to periodically send a string to the base station which in turn printed it to the console. This can be seen in Figure 5.1.

Two analysis were carried out with this setup. In the first analysis we sent a short 3 byte string and in the second analysis we sent a longer 15 byte string. We start with looking at the data

<table>
<thead>
<tr>
<th>Protects</th>
<th>Number of attacks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>13</td>
</tr>
<tr>
<td>Limited</td>
<td>0</td>
</tr>
<tr>
<td>No</td>
<td>1</td>
</tr>
</tbody>
</table>
The first analysis. Table 5.3 shows the data being transmitted in the radio channel without any security level applied. Table 5.4 shows the same data but with the low security level while Table 5.5 shows the data with the medium security level. The data being sent is “Hi!” in this first analysis.

![Image of Cooja with a sensor node and base station](image)

Figure 5.1: Cooja with a sensor node and base station

and result of the first analysis.

Table 5.3: No security level used

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Sender</th>
<th>Receiver</th>
<th>Byte Size</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>2</td>
<td>1</td>
<td>13:</td>
<td>0x01000200 92000100 02004869 21</td>
</tr>
<tr>
<td>1.00</td>
<td>2</td>
<td>1</td>
<td>13:</td>
<td>0x01000200 92000100 02004869 21</td>
</tr>
<tr>
<td>2.00</td>
<td>2</td>
<td>1</td>
<td>13:</td>
<td>0x01000200 92000100 02004869 21</td>
</tr>
<tr>
<td>3.00</td>
<td>2</td>
<td>1</td>
<td>13:</td>
<td>0x01000200 92000100 02004869 21</td>
</tr>
<tr>
<td>4.00</td>
<td>2</td>
<td>1</td>
<td>13:</td>
<td>0x01000200 92000100 02004869 21</td>
</tr>
</tbody>
</table>

Table 5.4: Low security level used

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Sender</th>
<th>Receiver</th>
<th>Byte Size</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>2</td>
<td>1</td>
<td>20:</td>
<td>0xC9F CECB2 F6748997 53C7FB73 9753DB1C CA72EF2C</td>
</tr>
<tr>
<td>1.00</td>
<td>2</td>
<td>1</td>
<td>20:</td>
<td>0xC9F CECB2 F6748997 53C7FB73 9753DB1C CA72EF2C</td>
</tr>
<tr>
<td>2.00</td>
<td>2</td>
<td>1</td>
<td>20:</td>
<td>0xC9F CECB2 F6748997 53C7FB73 9753DB1C CA72EF2C</td>
</tr>
<tr>
<td>3.00</td>
<td>2</td>
<td>1</td>
<td>20:</td>
<td>0xC9F CECB2 F6748997 53C7FB73 9753DB1C CA72EF2C</td>
</tr>
<tr>
<td>4.00</td>
<td>2</td>
<td>1</td>
<td>20:</td>
<td>0xC9F CECB2 F6748997 53C7FB73 9753DB1C CA72EF2C</td>
</tr>
</tbody>
</table>

Table 5.5: Medium security level used

We can easily observe the changes in both security and size of the data over the different security levels. Without any security the string is immediately visible in the data and the total size transmitted is 13 bytes.

With the low security level padding will be done since the payload and MAC-headers are smaller than 16 bytes (1 block). The result of this is a total size of 20 bytes, thus 6 padded bytes will be transmitted. The data is encrypted and not immediately visible when looking into the transmissions. However the same data is repeatedly transmitted due to ECBs lack of semantic security.

The medium security level has a total packet size of 22 bytes. This result might seem odd since the medium security level uses an additional 5 bytes for MIC and counter; yet it is only 2 bytes larger than when using the low security level. The explanation for this is that after the medium security level encryption the data will be, payload + second MIC + counter, thus 8 bytes in size. When the low security level applies its encryption it does no longer need to pad the data since MAC-header + payload is 18 bytes. Instead the low security level will use ciphertext stealing and retain the same data size. The last 4 bytes is the low security level MIC, thus resulting in a total of 22 bytes. With the medium security level we can also observe a much better confidentiality of the data. The data is different for each transmitted packet thanks to our counter and we thus have good semantic security.
The second analysis

In this analysis the sensor node sent the string “Hello you there” to the base station. Table 5.6 shows the data without any security level. Table 5.7 shows the data with the low security level and Table 5.8 shows the data with the medium security level.

Table 5.5: Medium security level used

<table>
<thead>
<tr>
<th>Time</th>
<th>Sender</th>
<th>Receiver</th>
<th>Byte Size</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>1237</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0x6F90BB85 483784E9 C18A3903 DAF382CC 54DAFAA0 6D07</td>
</tr>
<tr>
<td>2238</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0xCE07B241 CF351335 3666FC56 897005DC 7E2D3F48 75CB</td>
</tr>
<tr>
<td>3240</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0x2849010B 937F19BE 01224CEC 3A9B8F03 20329AF3 8CF4</td>
</tr>
<tr>
<td>4242</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0x830E6415 FF8E76BA CAB06D05 F9A3E0CD D66413C2 4130</td>
</tr>
<tr>
<td>5243</td>
<td>2</td>
<td>1</td>
<td>22</td>
<td>0x7E74C502 F56B4921 C3F404E0 014376FF 2D2D7F89 9A21</td>
</tr>
</tbody>
</table>

Table 5.6: No security level used

<table>
<thead>
<tr>
<th>Time</th>
<th>Sender</th>
<th>Receiver</th>
<th>Byte Size</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>1237</td>
<td>2</td>
<td>1</td>
<td>25</td>
<td>0x01000000 93000100 02004865 6C6F0F09 706F7F50 74686526</td>
</tr>
<tr>
<td>2238</td>
<td>2</td>
<td>1</td>
<td>25</td>
<td>0x01000000 93000100 02004865 6C6F0F09 706F7F50 74686526</td>
</tr>
<tr>
<td>3240</td>
<td>2</td>
<td>1</td>
<td>25</td>
<td>0x01000000 93000100 02004865 6C6F0F09 706F7F50 74686526</td>
</tr>
<tr>
<td>4242</td>
<td>2</td>
<td>1</td>
<td>25</td>
<td>0x01000000 93000100 02004865 6C6F0F09 706F7F50 74686526</td>
</tr>
<tr>
<td>5244</td>
<td>2</td>
<td>1</td>
<td>25</td>
<td>0x01000000 93000100 02004865 6C6F0F09 706F7F50 74686526</td>
</tr>
</tbody>
</table>

Table 5.7: Low security level used

The result is similar to that of the first analysis. The biggest difference is the length difference between when we used the low and medium security levels. The low security level has a length of 29 bytes; 25 bytes MAC-header and payload + the 4 byte MIC. The medium security level has an additional 5 bytes for the second MIC and counter. Since no padding is being done and only ciphertext stealing is used the data size will not increase when doing encryption at the low security level.

5.10 Cost for our security solution

The last part of our evaluation will be on how much each of our security levels cost in terms of battery power. As we have previously demonstrated the most expensive operation is sending and receiving data over the radio. The cryptographic operations carried out are small in comparison. Table 5.9 demonstrates each security levels additional byte overhead for all data packets.

The next aspect is shown in Table 5.10. It shows how many packets that need to be exchanged during the joining procedure for the different security levels.

Table 5.11 shows the last aspect we are looking into here. It shows the different ciphers used, their key sizes and if they are carried out in hardware or software. Since both the medium and high security levels use the low security level they use the AES cipher two times, thus 2xAES.

All of the above impact on power consumption. It will primarily be the data packet exchange overhead that will consume battery power. More rarely will the joining procedure be launched and require more power the more packets that are part of the joining procedure.

The high security level uses public-key cryptography during every joining procedure. The joining...
Table 5.8: Medium security level used

<table>
<thead>
<tr>
<th>Time</th>
<th>Sender</th>
<th>Receiver</th>
<th>Byte Size</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>1231</td>
<td>2</td>
<td>1</td>
<td>34</td>
<td>02C8EBF04C EC69F465 A3A5F7C7D ABF9902E B1F33318 2884BF09 2D38BC A57778A SFC3</td>
</tr>
<tr>
<td>3241</td>
<td>2</td>
<td>1</td>
<td>34</td>
<td>048C90EBDB BA82D50 46ED9FC6 08F08E09 C5481F66 43C58EB3 1B89855 229D34B5 5C24</td>
</tr>
<tr>
<td>4243</td>
<td>2</td>
<td>1</td>
<td>34</td>
<td>0404FD9EB2 8B5E2BDB 223E2D24 8D3C0847E8 820E9C8 229D34B5 5C24</td>
</tr>
<tr>
<td>5245</td>
<td>2</td>
<td>1</td>
<td>34</td>
<td>04AEB9AFFAD 27E5F831 9527B9A 14B1C4A7 6959B976 769BB7E4 EA74137D D638A2B F4BC</td>
</tr>
</tbody>
</table>

Table 5.9: Cost in byte size for each security level

<table>
<thead>
<tr>
<th>Security level</th>
<th>Byte overhead</th>
<th>% increase of maximum packet size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>2</td>
<td>1.5</td>
</tr>
<tr>
<td>Medium</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>High</td>
<td>9</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 5.10: Packets sent during network joining

<table>
<thead>
<tr>
<th>Security level</th>
<th>Number of packets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>2</td>
</tr>
<tr>
<td>Medium</td>
<td>2</td>
</tr>
<tr>
<td>High</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 5.11: Cryptographic size and where the calculations are performed

<table>
<thead>
<tr>
<th>Security level</th>
<th>Cipher</th>
<th>Cryptographic size (bits)</th>
<th>Executed in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>AES</td>
<td>128</td>
<td>Hardware</td>
</tr>
<tr>
<td>Medium</td>
<td>2xAES</td>
<td>2x128</td>
<td>Hardware</td>
</tr>
<tr>
<td>High</td>
<td>2xAES &amp; Public-key</td>
<td>128 &amp; 256 &amp; 512</td>
<td>Hardware &amp; Software</td>
</tr>
</tbody>
</table>
procedure is also heaviest for this security level. The joining procedure will be initiated each
time a sensor node connects to a new base station for exchanging of a new secret symmetric
key. The procedure becomes heavier since we have to carry out all public-key cryptography
operations in software. It uses 128-, 256- and 512-bit keys that result in heavier computation
than the other levels that use 128-bit keys. This might make the high security level to heavy to
be used in scenarios where security is not a major focus and sensor nodes tend to move around
a lot.

The medium security level is the one offering the best tradeoff between security and power
consumption. It has a longer byte overhead than the low security level but a simple joining
procedure that does not need to be re-launched if sensor nodes move in the network (all base
stations already share the end key). It also has hardware support for all cryptographic operations
unlike the high security level.

For scenarios where the security should be kept at a bare minimum the low security level suits.
With only a 2 byte overhead, a simple joining procedure and all cryptographic operations done
in hardware the power consumption is kept at a minimum.
Chapter 6

Summary and Conclusion

In this final chapter we have a summary and a concluding discussion on the outcome of this thesis. We also talk about the developed security solution together with some areas where future work can make improvements.

6.1 Summary

In this thesis we have a number of chapters in which we highlight the security of Open Controllers platform. We began with describing the platform and a scenario to show environments in which the platform might be used. We continued with performing an analysis of the security in the platform. This analysis showed what we wanted to protect, known attacks, and also gave us a threat table with associated risks through the use of a threat modeling process. When we understood what to protect and what to protect against we developed a security solution. The security solution was developed with several platform requirements and restrictions in mind.

The outcome of this development process consisted of two parts; the security solution itself and a list of new security policies. The last part of this thesis is the evaluation chapter in which we evaluated our security solution. In this chapter we showed that the platform is secure using our security solution with the accompanying security policies. We ended the thesis with a summary and conclusions chapter where we discussed the thesis, its outcome and areas that future work may improve on.

6.2 Discussion

The outcome of this thesis may well be said to satisfy our goals. In this thesis we assessed the security of the platform and developed a security solution. The resulting security solution is transparent to software running in the application layer and provides demonstrably good protection with today’s standards. The footprint of our security solution in terms of power consumption is kept to a minimum whilst the security is of high quality.

Throughout this thesis we used well-established and well-known methods and processes to evaluate different security aspects. We started with deducing our assets in the beginning of our
security analysis. This was a prerequisite to our security policy part as well as our threat model. The assets together with the security policies introduce, in more detail, what we want to protect to the reader. This approach lays out a fairly basic understanding of what is valuable in the platform and we think this makes it easier for the reader to grasp the later threat modeling part.

We choose to use a threat modeling process from Microsoft. There exist several other ways of carrying out a structured risk analysis, or as it may also be called, threat analysis or vulnerability scoring. They may all have the same goal but often distinguish their focus on what is important when the damage potential is estimated. We want to emphasize that the model we chose was suitable for our goal but it would not have been a problem to use other models. The Microsoft model we choose was easy to adapt to our kind of network; only minor changes was required to model. By and large, these changes did not affect the model structure, but only the terminology used in it. The chosen model was aimed towards web security, an area that share many common properties with our area. For example both areas handle packets from possible un-trusted sources and needs to provide a service to the outside world. Our choice of model was also easy to adapt to match the platform developers list of security prioritizations; this was done by adding to DREAD to get our DREADO.

The other more or less standard method we used was the prevention, detection and reaction method in the evaluation chapter. Since security is about the protection of assets we wanted to be able to evaluate the protective measures that our security solution offered. There are several similar methods to do this. Common to most of the models is that they try to evaluate three aspects. The first aspect the models commonly try to evaluate is how the security solution prevents our assets from being damaged. The second aspect is how the security solution detects failures of preventive security controls, in other words when an asset have been damaged. The third aspect is what separates the different models. In some models it is called Reaction while others call it Response or Recovery. However, even if they use different words to describe this third aspect, it has in principle the same meaning. The third aspect can roughly be described as taking measures that allow us to recover our asset or recover from damage to it when an attack is detected; in other word a kind of response to the detected event. In the end we choose to use the terms Prevention, Detection and Reaction (PDR). This gave us a good theoretical basis since this definition is used by Gollmann in his book on Computer Security [15].

6.3 The security solution

The resulting security solution has been shown to be satisfying to our requirements, hardware restrictions and security evaluation. The security solution is ready to be fully implemented and used in the current wireless sensor network platform. Our three security levels allow any implementers to only implement the needed level and not the full security solution. This enables for faster deployment and easier maintenance. The implementation developed together with this thesis is a proof of concept. While it shows how the security is carried out by our security levels it will need some adaption for each different system it might be implemented into. Open Controllers vision is to release all software belonging to their wireless sensor network platform as open source when it is completed. This enables the security solution to gain a much more widespread usage and adoption.
6.4 Future work

This report touches on several important security parts and gives a solution for each one of them. However, more complex solutions might be needed in the future and we will cover those here.

**Broadcast communications**  Currently we only take unicast communication into consideration when developing the security solution. Even though the resulting security solution works with broadcast, due to the security solutions design, more efficient broadcast communication might need to be implemented in the form of bloom-filters.

**Keying mechanisms**  Future security solutions might require more fine-grained key sharing and management on our security levels or dynamically changing the security level of a running system. The survey on security in wireless sensor networks by Q. Wang [37] goes through some different key management techniques.

**Padding with the low security level**  As we showed in the packet format part of our low security level, we might send some padded data if the payload is too small. In future work one can investigate if we should do MIC than encrypt rather than encrypt then MIC as we do today. If the deployed system uses data so that padding needs to be applied frequently one might consider doing MIC then encryption, thus saving in on 4 padded bytes.

**Limitations**  In this thesis we have noted several parts on which we have not investigated further. The first limitation we did was the internal security boundary at the SPI bus inside the sensor node. We do secure this SPI bus with the use of our medium and high security level but it is unprotected when using the low security level. Further we do not try to counter jamming attacks in the radio channel or prevent physical stealing of sensor nodes or base stations.

Other parts that we breezily entered was the subsections under *Additional security issues* in Further work may develop more efficient techniques for the issues under this section to be used together with our security solution.
Bibliography


Appendix A

Security policies evaluation

In this appendix we go through and evaluate our initial security policies against Matt Bishops definitions.

Definition 1-1. A security policy is a statement of what is, and what is not, allowed.

The first policy clearly states that only authorized and authenticated user are allowed; others are not allowed. The second and third policy states that authorized parties are allowed while unauthorized parties are not allowed. Policy four states that only company-configured base stations shall be allowed in the network; base stations configured by any other party shall not be allowed. Since all policies explicitly states what is allowed, thus everything else is disallowed, we fulfill the first definition.

Definition 4-1. A security policy is a statement that partitions the states of the system into a set of authorized, or secure, states and a set of unauthorized, or nonsecure, states.

Definition 4-2. A secure system is a system that starts in an authorized state and cannot enter an unauthorized state.

Definition 4-3. A breach of security occurs when a system enters an unauthorized state.

To show how definition 4-1, 4-2 and 4-3 influence our policies we break apart the platform into secure and nonsecure states. We also assume that the platform contains some set of transition functions used by the platform to change state. We call the set $U$ the secure states and the set $NS$ the nonsecure states. In Figure A.1 below we partition the secure states $U = \{S1, S2\}$ while the unsecure states are $NS = \{S3, S4\}$.

For the first policy the secure states $U$ will be the states when all nodes in the platform are authenticated and authorized. The nonsecure states will be when an unauthenticated or unauthorized node has managed to get into the network; illustrated by the possible state transition from $S2$ to $S3$ in Figure A.2. If we have security mechanisms in place to stop unauthorized nodes from getting into the network we will have a secure platform by definition 4-2; thus having no way of state transitioning from any state in $U$ to any state in $NS$. According to definition 4-3 we have a security breach if we allow any transition from $U$ to $NS$; this applies for all our security policies.

We now apply the same line of reasoning for the rest of the policies. The second policy will have
Figure A.1: Secure and nonsecure states

Figure A.2: Shows a platform that is not a secure system according to definition 4-2 since we have a transition from $S2$ to $S3$. The secure states $U$ which are the states where there is no unauthorized party listening in on the network traffic and gaining information. The unsecure states $NS$ is when an unauthorized node can get at the information being sent. If we have no security mechanisms in place any person with the right equipment can listen in on the network traffic thus creating a transition from a secure state in $U$ to a nonsecure state in $NS$. The third policy is much like the second one. We have the secure states $U$ which are the states where there is no unauthorized parties modifying content being sent. The nonsecure states $NS$ are the states where an unauthorized party successfully modifies content being sent. If we do have an unauthorized party like in the latter case we have a transition from a secure state in $U$ to a nonsecure state in $NS$. The fourth and last policy has the secure states $U$ which are states where there is no malicious base station in the network while the nonsecure states in $NS$ are those where we do have a malicious base station. Again if we have no security mechanisms in place a transition can exist from secure to unsecure states if a malicious base station comes into the network. If this transaction takes place we have a breach of security according to definition 4-3.

**Definition 4-4.** Let $X$ be a set of entities and let $I$ be some information. Then $I$ has the property of Confidentiality with respect to $X$ if no member of $X$ can obtain information about $I$.

Following definition 4-4 we can say that the entities $X$ for the first, second and third policy are all the unauthorized or unauthenticated users while $I$ is the information being sent in the network. For the fourth policy the entities $X$ are third-party configured base stations and $I$ is, like before, the information being sent in the network. If we have security mechanisms in place so that no one can violate our security policies, meaning no entity in $X$ can listen in on the
communication, we fulfill the definition. We say that by the use of our security policies we offer confidentiality for \( I \).

**Definition 4-5.** Let \( X \) be a set of entities and let \( I \) be some information or a resource. Then \( I \) has the property of integrity with respect to \( X \) if all members of \( X \) trusts \( I \).

For the first, second and third policy let the entities \( X \) be all authorized and authenticated nodes in the network and let \( I \) be the information currently being transferred in the network. In the fourth policy let the entities \( X \) be all company-configured base stations. If our policies are not violated we offer integrity since all valid nodes and base stations trust the information \( I \).

**Definition 4-6.** Let \( X \) be a set of entities and let \( I \) be a resource. Then \( I \) has the property of availability with respect to \( X \) if all members of \( X \) can access \( I \). For all four policies let \( I \) be the network channel and \( X \) be all entities communicating over the network. If no violations of our policies occur we offer availability since all entities \( X \) can access the channel \( I \).

**Definition 4-7.** A security mechanism is an entity or procedure that enforces some part of the security policy.

We need to apply several different security mechanisms to ensure secure states. These security mechanisms are developed in [chapter 4](#) of this thesis.

**Definition 4-8.** A security model is a model that represents a particular policy or a set of policies.

We arrive at what we will be calling the security model denoted SM. Our security model will be our four security policies denoted \( P_1, P_2, P_3, P_4 \) thus \( SM = \{ P_1, P_2, P_3, P_4 \} \).
Appendix B

Abbreviations

- AM: Active Message
- ASIC: Application-Specific Integrated Circuit
- CBC: Cipher Block Chaining
- CBC-CS: Cipher Block Chaining Ciphertext Stealing
- CBC-MAC: Cipher Block Chaining Message Authentication Code
- CCM: Counter with Cipher Block Chaining Message Authentication Code
- CFB: Cipher Feedback
- CS: Ciphertext Stealing
- CSMA-CA: Carrier Sense Multiple Access with Collision Avoidance
- CTR: Counter
- CRC: Cyclic Redundancy Check
- DLOG: Discrete Logarithm
- ECB: Electronic Code Book
- ECC: Elliptic Curve Cryptography
- FPGA: Field-Programmable Gate Array
- HMAC: Hash-based Message Authentication Code
- IDS: Intrusion Detection System
- IV: Initialization Vector
- MAC: Media Access Control
- MIC: Message Integrity Code
- OFB: Output Feedback
- PDR: Prevention, Detection, Reaction
- PHR: Physical Header
- PKCS: Public-Key Cryptography Standard
- PPDU: Physical Protocol Data Unit
- SHA: Secure Hash Algorithm
- SHR: Synchronization Header
- TDMA: Time Division Multiple Access
Appendix C

Evaluation against the attack trees

In this part of the thesis we do evaluation against our identified threats visualized in the attack trees. For each threat we identify the protection provided to counter the attacks in it with our security levels. This gives a good visual representation of how good protection the security solution offers. We go through each threat category and end this part with a summary on the protection offered.

C.1 Spoofing identity

Get credentials of legitimate sensor node  When we talk about the credentials in our security solution we mean the id and cryptographic keys used. For the low security level the key will be the network key while high and medium also includes other unique keys.

- Deploy a malicious base station

The low security level protects against this attack by requiring both joining and network keys before a device can participate in the network.

- Deploy a malicious sensor node acting as a relay

This attack differs between the low security level and the other two. In low a malicious sensor node can relay the same traffic without the need for disrupting the original traffic. This is due to the lack of replay protection. For medium and high the attacker needs to disrupt the original traffic so it will not arrive. If it would arrive the relayed traffic would be dropped due to the replay protection. However, the low security level encrypts the traffic so it provides sufficient protection for this attack.

- Observe non-encrypted credentials

All security levels offer encryption of the data, thus the low security level is sufficient.
Dictionary attack, Brute force

Security policy [13] ensures that only true random keys will be used.

Extract non-encrypted credentials from onboard memory

Each sensor node stores all cryptographic keying material in a secure way.

**Get credentials of legitimate base station**

- Deploy a malicious sensor node acting as legitimate one

The low security level is suffices for protecting against this attack. By the use of the joining and network key we ensure that no malicious sensor node can join the network or obtain anything from analyzing traffic.

- Relay traffic to legitimate sensor node

The low security level protects against deployment of a malicious sensor node. However as mentioned for the relay attack under get credentials of legitimate sensor node the attacker can still relay the data. Yet the attacker cannot obtain anything useful since the traffic is encrypted.

- Observe non-encrypted credentials

The low security level offers encryption, thus no non-encrypted credentials will be observable.

Dictionary attack, Brute force

Security policy [13] ensures that only true random keys will be used.

Extract non-encrypted credentials from onboard memory

Each base station stores all cryptographic keying material in a secure way.

**Craft malicious packet**

- Copy packet from the network and replay at a later time

The medium or high security level is needed to counter this attack since they have replay protection.
• Use no credentials

The low security level enforces that each packet needs credentials in form of the network key to not be dropped.

• Use made-up credential

All our security levels use random keys of at least 128-bit. This makes it an impossible task to guess for instance the network key.

C.2 Tampering with data

Tamper with a packet in the radio channel

• Deploy malicious base station

The low security level protects against this attack by the use of its keys.

• Deploy malicious sensor node acting as a relay

The low security level is suffice for protecting against this attack due to the keys used.

• Dictionary attack, Brute force

Security policy [13] ensures that only true random keys will be used.

• Carry out modifications that will be unnoticed

The low security level provides authenticity with the use of a message integrity code. The MIC will prevent this type of attack.

• Non-encrypted packet

The traffic will be encrypted for all security levels.

• Non-encrypted headers in packet

The low security level encrypts and authenticates the whole MAC-frame.
Influence data stored in the back-end

- Replay captured data packets

Replay protection is needed to counter this attack. The medium and high security level provides this.

- Deploy malicious base station

The low security level protects against this attack by requiring both joining and network keys before a device can participate in the network.

- Deploy malicious sensor node

The low security level is suffices for protecting against this attack due to the keys used.

C.3 Repudiation

Perform illegal operation which origin cannot be traced

- Participate in network

We make it impossible to participate in the network without a valid joining and network key. Thus the low security level is sufficient to keep attackers from participating in the network.

- No logging facility

The security solution uses a logging facility in the back-end for all security levels.

- Craft packet without credentials

The keys used are considered to be one type of credentials. The network key used in the low security level is sufficient to prevent this attack. The higher levels have additional protection by the end-to-end keys used.

- Craft packet using false credentials

False credentials can be either stolen or made-up credentials. This attack is prevented by at least two factors. The first factor is the secure storage of our keys in the devices. The second is that the credentials are always securely transmitted across the network during re-keying or key-distribution procedures. This applies for all of the security levels. We know since before that it is impossible for an attacker to guess the used key. The higher levels use an additional end-to-end key further complicating this attack.
• Copy transmitted packet from other sensor node

The medium or high security level is required since low does not offer any replay protection.

### C.4 Information disclosure

**Outsider gets information flowing in the network**

• Deploy malicious base station

The low security level protects against this attack by the use of its keys.

• Deploy malicious sensor node acting as a relay

The low security level is sufficient for protecting against this attack due to the keys used.

• Observe non-encrypted traffic

All traffic is encrypted by all our security levels.

**Dictionary attack, Brute force**

Security policy [3] ensures that only true random keys will be used.

**Outsider gets information on the topology of the network**  We do not offer any traffic-masking scheme in our security solution. Thus we only provide limited protection for some of the attacks mentioned here.

• Deploy malicious base station

The low security level protects against this attack by the use of its keys.

• Deploy malicious sensor node acting as a relay

The low security level is sufficient for protecting against this attack due to the keys used.

• Observe non-encrypted traffic

The traffic will be encrypted for all security levels.
• Observe non-encrypted packet headers

The low security level encrypts and authenticates the whole MAC-frame.

• Analyze who communicates with whom

This attack is somewhat mitigated through the encryption of all aggregated data. An attacker can only see direct communication between devices at the MAC-layer. The attacker cannot see from where or what the aggregated data contains. However we can only claim to provide limited protection against this type of attack.

• Analyze traffic patterns

We offer limited protection by the fact that an attacker cannot observe any content of the aggregated data. The attacker cannot deduce traffic patterns other than observing the encrypted MAC-layer.

• Dictionary attack, Brute force

Security policy [3] ensures that only true random keys will be used.

Get traffic source location

• Deploy malicious base station

The low security level protects against this attack by the use of its keys.

• Deploy malicious sensor node acting as a relay

The low security level is sufficient for protecting against this attack due to the keys used.

• Observe non-encrypted traffic

The traffic will be encrypted for all security levels.

• Observe non-encrypted packet headers

The low security level encrypts and authenticates the whole MAC-frame.

• Look for routing information
Routing data is secured by the low security level through encryption.

- Trace traffic back to source

The low security level secures all aggregated data with encryption. Thus an attacker cannot look into the data and trace it back to the source.

- Dictionary attack, Brute force

Security policy [3] ensures that only true random keys will be used.

### C.5 Denial of service

**Limit availability of a sensor node**

- Participate in network

The need for network and joining keys prevent this attack at the low security layer.

- Deploy malicious sensor node acting as a relay

The low security level is sufficient for protecting against this attack due to the keys used.

- Deploy malicious base station

The low security level protects against this attack by the use of its keys.

- Cause retransmission

Except from causing retransmission through the use of jamming, which we do not cover here, an attacker can craft packets to cause retransmissions. All security levels use their authentication of data to verify that no malicious crafted packets are processed.

- Relay traffic over node

The low level ensures that no malicious sensor nodes can participate with the use of authentication. The higher levels improve on this protection by verifying that the relay data is not malicious when arriving at a base station. This cannot be verified directly in the sensor node since it lacks the end-to-end key with which the relayed data is encrypted.
• Flood the sensor node with new connections

The low security level ensures that the platform will not accept any non-authenticated connections. The higher levels do not add any protection above the low level.

• Perform jamming

The security solution does not offer protection against this but raises an alert on detection. We consider our security solution to offer limited protection by detection for this threat.

Limit availability of a base station

• Participate in network

The need for network and joining keys prevent this attack at the low security layer.

• Deploy malicious sensor node acting as a relay

The low security level is suffices for protecting against this attack due to the keys used.

• Deploy malicious base station

The low security level protects against this attack by the use of its keys.

• Perform jamming

The security solution does not offer protection against this but raises an alert on detection. We consider our security solution to offer limited protection by detection for this threat.

C.6 Elevation of privileges

Get base station privileges for malicious device This threat only has the sub-goal of spoofing the identity of a base station. See the threat get credentials of legitimate base station for details.

Get sensor node privileges for malicious device The only sub goal of this threat is spoofing the identity of a legitimate sensor node. See the threat of stealing the credentials of a legitimate sensor node for details.
C.7 Summary

We assign each threat a security level requirement in Table C.1. For example, if a threat contains a number of attacks where the low security level offers protection the value will be Low. If it however has one attack that only medium does protect against the value will be medium. Limited indicates that we offer some protection for the threat with the use of our security levels.

<table>
<thead>
<tr>
<th>Id</th>
<th>Threat</th>
<th>Lowest security level required</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.2</td>
<td>Get credentials of legitimate base station</td>
<td>Low</td>
</tr>
<tr>
<td>1.3</td>
<td>Craft malicious packet</td>
<td>Medium</td>
</tr>
<tr>
<td>6.3</td>
<td>Get base station privileges for malicious device</td>
<td>Low</td>
</tr>
<tr>
<td>1.1</td>
<td>Get credentials of legitimate sensor node</td>
<td>Low</td>
</tr>
<tr>
<td>3.1</td>
<td>Perform illegal operation which origin cannot be traced</td>
<td>Medium</td>
</tr>
<tr>
<td>6.2</td>
<td>Get sensor node privileges for malicious device</td>
<td>Low</td>
</tr>
<tr>
<td>2.2</td>
<td>Influence data stored in the back-end</td>
<td>Medium</td>
</tr>
<tr>
<td>4.1</td>
<td>Outsider gets information flowing in the network</td>
<td>Low</td>
</tr>
<tr>
<td>4.3</td>
<td>Get traffic source location</td>
<td>Low</td>
</tr>
<tr>
<td>2.1</td>
<td>Tamper with a packet in the radio channel</td>
<td>Low</td>
</tr>
<tr>
<td>4.2</td>
<td>Outsider gets information on the topology of the network</td>
<td>Low - Limited</td>
</tr>
<tr>
<td>5.2</td>
<td>Limit availability of a base station</td>
<td>Low - Limited</td>
</tr>
<tr>
<td>5.1</td>
<td>Limit availability of a sensor node</td>
<td>Low - Limited</td>
</tr>
</tbody>
</table>

Table C.1: Evaluation against attack trees

This evaluation against our identified attack trees shows that the low security level only provides limited protection. The medium and high security level offer good protection. The same conclusion as on the evaluation against known attacks is valid here. For systems where security is not at focus the low security level is suitable. Other systems with higher security demands will have to implement either the medium or high security level.
På svenska

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/

In English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: http://www.ep.liu.se/

© Christoffer Engvall