Secure Integration of Wireless Sensor Networks into Applications

Master Thesis

August, 2008

Christophe Trefois
(christophe.trefois@epfl.ch)

School of Computer and Communication Sciences

Supervision

Prof. Jean-Pierre Hubaux	**Laurent Gomez**
EPFL - LCA1	SAP Labs France

CONTENTS

LIST OF FIGURES

Acknowledgements

This Master Thesis is dedicated to my grand mother.

I would like to thank my industrial supervisor Laurent Gomez (Senior Researcher at SAP), for giving me the unique opportunity to write my Master Thesis with SAP Labs France. I am grateful for his valuable tips, his commitment and his numerous reviews.

I would also like to thank my academic supervisor Prof. Jean-Pierre Hubaux, who kept following the evolution of this project with great interest. I am grateful for his availability and useful advices in shaping the evolution of the project.

I am grateful to Dr. Annett Laube, Alessandro Sorniotti and Dr. Hoon Wei Lim for their reviews and feedback throughout this project.

Finally I would like to thank my family and my girlfriend for their continuous support in this experience.

Preface

SAP

Founded in 1972, SAP is the leading provider of collaborative business solutions. Headquartered in Walldorf, Germany, SAP is the world's third-largest independent software supplier. They employ over 43,000 people in more than 50 countries. Their professionals are dedicated to provide high-level customer support and services. Through SAP Research, they introduce new ideas for future solutions.

In contrast to SAP's product groups, which work on new functions and releases, SAP researchers explore opportunities that haven't been developed into products yet. They track technology trends, evaluate the potential impact on SAP solutions and customers, and generate breakthrough technologies.

Located in Walldorf, Germany, SAP Research has additional facilities in Palo Alto, CA, United States; Karlsruhe, Germany; Brisbane, Australia; **Sophia Antipolis, France**; and Johannesburg, South Africa.

The WASP Project

The WASP project [90] is a research project funded by the European Union and includes 19 partners. The consortium consists of six industrial partners, one Small-to-Medium Enterprise (SME), six large research institutes and six universities (including EPFL). All of them have a proven experience with WSNs.

The Mission Statement of WASP is

"The WASP (Wirelessly Accessible Sensor Populations) project provides theory, methods, hardware and software to construct highly optimized applications on a network of generic and flexible nodes." [90]

Academia is actively researching on Wireless Sensor Networks (WSN). However, the Industry is reluctant to use the results coming from academic research. This is due to the mismatch between the research at the Application Level and the Node and Network Level.

The WASP project aims at narrowing this mismatch down by covering the whole range from basic hardware, sensors, processor, communication, over the packaging of the nodes, the organization of the nodes, toward the information distribution and a selection of applications. The emphasis in the project lays in the self-organization and the services, which link the application to the sensor network.

Research into the *nodes* themselves is needed because a strong link lies between the required flexibility and the hardware design.

Research into the *applications* is necessary because the properties of the required service will influence the configuration of both sensor network and application for optimum efficiency and functionality.

Three business areas, elderly care, traffic control and herd control, are selected for their significance in the society and large range of requirements, to validate the WASP results.

CHAPTER 1

INTRODUCTION

Wireless sensors are small devices that are able to gather, process and deliver information from a physical environment to an external system. By doing so, they open new applications in different domains, such as healthcare, traffic control, defense and agriculture. In healthcare for example, we can imagine remotely monitoring an elderly patient at his home . His pulse, body temperature, activities and other physiological signs are monitored continuously using a Wireless Sensor Network (WSN). This WSN is connected to a medical emergency response center in charge of detecting abnormal changes in the patient's health.

The integration of WSN with Business Applications (BAs) (as for healthcare) raises technical and security related challenges. Existing approaches (e.g. EIC [42], RUNES [6], CoBIs [1], Agimone [45]) target technical issues such as interoperability between WSN and BAs or heterogeneity of acquired sensor data. However, to the best of our knowledge, efficient end-to-end confidentiality of sensor data between sensor nodes and BAs is not addressed in the literature.

In this work, we focus on the security and trust challenges raised by the integration of WSNs into BAs. In healthcare for example, we will see that the patient's medical data is confidential and highly sensitive. This means that there is a need for end-to-end confidentiality of sensor data from sensors to the applications. The fulfilment of this requirement is difficult due to the limited resources in WSNs. Therefore, we will introduce an efficient security scheme [83] that does not use complex operations and guarantees end-to-end confidentiality of sensor data.

In chapter 2, we provide an insight of WSN and their integration with business applications. We emphasize on the fact that business applications have a strong interest in the integration with WSNs. We propose in chapter 3 to analyze the risks of such integration. We show that loss of privacy of sensor data is critical for business applications. Chapter 4 is dedicated to the security analysis of an existing security scheme [83] for end-to-end confidentiality of sensor data from the nodes to the business applications. In addition to a detailed security analysis of this security scheme, in chapter 5, we study the randomness of the key generation proposed in

[83]. Last but not least, in order to check the feasibility of this approach, we implement a proof-of-concept based both on a simulated and real WSN. Details of this implementation can be found in chapter 6. We conclude this master thesis with a summary of our contribution and future work in chapter 7.

CHAPTER 2

BACKGROUND

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."
Mark Weiser

2.1 Wireless Sensor Networks

The research community has shown a considerable interest in Wireless Sensor Networks (WSNs) in the past years. Some even claim that WSNs are one of the emerging technologies that will change the world [10].

WSNs are made of small devices called Sensor Nodes (motes) that monitor their local environment. These low-cost, self-powered motes have just enough transmission power to pass sensed data to nearby sensors which in turn forward it to other sensors. This leads to a new type of distributed system that is self-organizing and opens many applications, ranging from traffic over healthcare to military.

2.1.1 WSN Components

Figure 2.1 shows how Wireless Sensor Networks (WSNs) can be connected to existing infrastructures, such as the Internet. We identify three major components: (i) Sensor Nodes, (ii) Sinks and (iii) Gateways.

Nodes are organized in *Sensor Fields* and connected to a *Gateway* via a *Sink* node. Sinks are sensor nodes which are more powerful than normal motes and can be seen as the "spokesman" of the underlying sensor field. Every sensed data packet travels through the sink and is relayed to the gateway. The gateway is in charge of delivering data to higher level services, such as a WSN Middlewares (See Section 2.2).

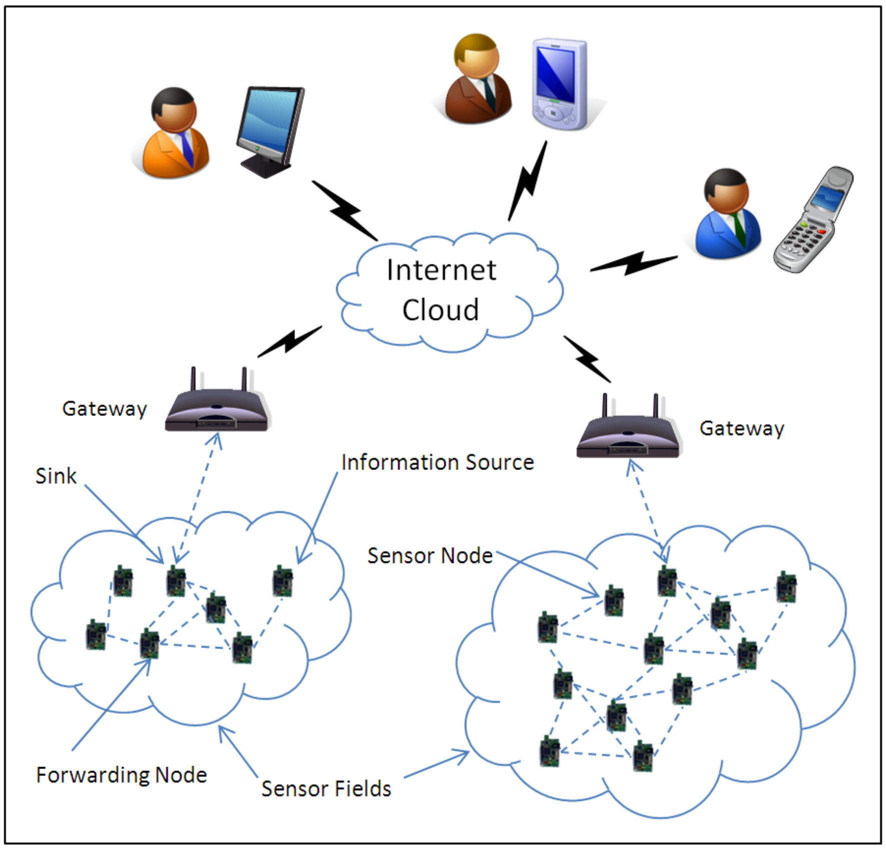


Figure 2.1: Wireless Sensor Network Architecture

Sensor Node

A **Sensor Node** has very basic interfaces and components. Each is equipped with a limited processor, little memory and sensors that monitor for example temperature, light and motion. It is also equipped with a wireless transceiver that has a short transmission range of typically around 20-50 meters. The sensor size is mostly determined by its power supply which usually is 2xAA batteries.

As an example we show a Crossbow MICAz Sensor Board[1] in Figure 2.2. This sensor node has 128 KiB of memory, its power comes from 2xAA batteries, it consumes 19.7 [mA] in reception and 14 [mA] in transmission. We see that sensor nodes have little energy and computational power at their disposal. The main requirement for working with sensor nodes therefore becomes optimizing its energy consumption.

A wireless sensor node can be an **Information Source**, a **Forwarding (routing) Node** or a **Sink**.

When the node acts as an **Information Source**, it only generates data and transmits it to its neighbours (e.g. temperature, pressure). To save energy, the node will be inactive most of the time, and only "wake-up" periodically.

[1]See `http://www.xbow.com`

Figure 2.2: Crossbow MICAz Sensor Board

A **Forwarding** Node is one that can perform some intermediate data processing (e.g. data aggregation or taking average/max/min) and relays data from other sensor nodes. A sensor node can act as both an *Information Source* and a *Forwarding Node.*

A **Sink** receives all the data generated in the WSN and forwards them to the gateway. Since all the traffic goes through the sink, it needs more power and ressources than a normal sensor node.

Gateway

The gateway is the bridge between the WSN and other entities such as a WSN middleware. It can either store data locally or forward it.

2.1.2 WSN Characteristics

Sensor nodes are usually scattered in sensor fields as shown in Figure 2.1. The number of sensor nodes deployed can be in the order of hundreds of thousands in each field. WSNs have a number of operational characteristics, the most important of which are listed below:

Single-Hop Communication. One of the operational characteristics of WSN is direct communication. In this communication mode, all sensors have a direct connection to the sink node. There is no forwarding of data through forwarding nodes. This approach however is too power consuming and not scalable, as all nodes need to be in range with the sink node. Therefore it makes more sense to rely on multi-hop communications.

Multi-Hop Communication. In this communication mode, forwarding nodes are used to enable communication over long distances for devices that are not in range with a sink node. The idea is to make the network more energy efficient, as the devices require less power to communicate hop by hop then if they were to communicate with the sink directly. In most case scenarios, the sensor nodes would not be able to communicate with the sink directly anyway, since their range is very limited (few meters).

Energy Efficient Mode of Operation. A key feature to support long lasting sensor networks is to use the available limited energy in the most optimal way. Sensor nodes can greatly increase their energy consumptions by sleeping most of the time and only periodically waking up to receive and transmit data.

Automatic Load Balancing. Nodes should be autonomous enough to "smartly" select the parent node to which they send their data packets to. The selection process is based on the link quality, the parent's load and the hop count to the sink.

Self-Configuration. Sensor nodes should automatically adapt to changing network characteristics while optimizing energy usage. This auto-configuration has to deal with node failures, node additions to the network and obstacles.

2.2 Middlewares

One obstacle in the integration of WSN into business applications is the gap between software applications and low-level constructs such as WSNs. To fill this gap, middleware systems are used. They can be considered as a **software infrastructure** that is the glue between the sensor network and the client applications.

There are many existing middleware architectures, but most of them focus on the requirements of the WSN itself and do not address the integration of WSNs with business applications. In the next section we will describe some of the middlewares we identified.

2.2.1 Existing Middlewares

CoBIs

In CoBIs [1], the authors propose that applications access the services offered by the WSN via Web Services (Information about Web Services can be found in [14]). In other words, the WSN is hidden to the application layer. The major task of the middleware is the mediation of service requests between the application layer and the WSN layer. However, the major drawback is that the sensor nodes need to be modified in order to work with this architecture. We believe that a middleware should not be aware of the Application Layer and as such CoBIs is not transparent enough.

RUNES

The RUNES [6] project uses a component-based middleware. Components are units of functionality and deployment that interact through interfaces. RUNES offers plug-in interaction paradigms described as generic APIs that can be implemented in accordance with the application needs. RUNES looked promising initially but there was no fully-functional implementation available.

MiLAN

The MiLAN [48] middleware focuses on sensor network management to enable proactive WSN applications. They introduce the notion of generic connectors that permit to gather data from any types of nodes. However, the issue of standardizing sensor data into a single format is left unanswered.

Agimone

Agimone [45] is a middleware that integrates Sensor and IP networks. Its main focus lies on the distribution and coordination of WSN applications (Mobile Agents) across different WSNs. Even though their motivation is based on a cargo tracking application, the actual integration of Agimone into real applications is out of scope.

Hourglass

Hourglass [78] is an Internet-based infrastructure for connecting a wide range of sensors, services, and applications in a robust fashion. In Hourglass, streams of data elements are routed from sensors to client applications. WSNs are connected to the Hourglass infrastructure through proxy services. These Services are not platform independant and as such does not offer independance of the WSN, the middleware and the applications.

IrisNet

The IrisNet [39], contrary to traditional WSNs, proposes a sensor network made of desktop PCs equipped with low-cost sensors such as webcams and inter-connected over the Internet. Their architecture offers a query service to obtain sensor data from anywhere on the Internet. Unfortunately, IrisNet is not taking into account the resource restrictions of embedded sensors and therefore a migration to traditional WSNs is not possible.

EIC

The EIC (Enterprise Integration Component) [42] combines a top-down approach of context-aware and bottom-up approach of WSN middleware. The EIC is designed on a Service Oriented Architecture (SOA) [14] and addresses the heterogeneity of WSNs. It also offers a standardized way to access sensor data.

The EIC is designed by taking into account business application requirements and as such addresses the problem of delivering data to enterprise systems that has been left open until now. We feel that this middleware offers a modular and flexible architecture that effectively links the WSN and the Business Application world together. Furthermore, in the context of the WASP project (introduced in the Preface), the EIC has been retained as middleware and an implementation is also available. For those reasons, we use the EIC as middleware in the scope of this project .

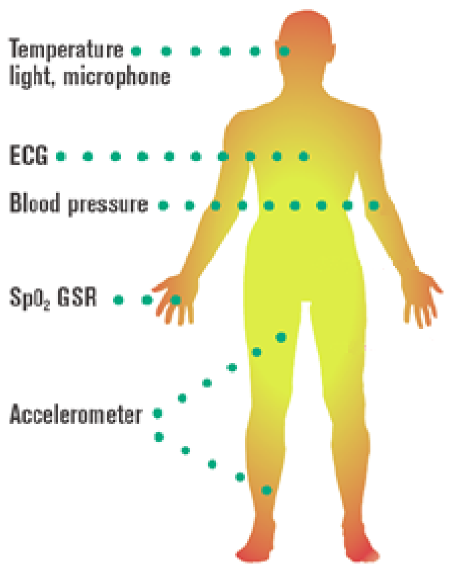


Figure 2.3: Body Sensor Network (BSN)

2.3 Healthcare Scenario

In WASP, three business areas are retained to validate the results of the project: (i) Elderly Care, (ii) Herd Control and (iii) Traffic Control. In the scope of this thesis, we will discuss the healthcare scenario because the use of WSNs offers many promising applications that could lead to better healthcare systems. We first give an overview of Body Sensor Networks and Ambient Sensor Networks, which are special types of WSNs followed by a description of the Patient Monitoring scenario.

2.3.1 Body Sensor Networks

Body Sensor Networks (BSNs) can be seen as WSNs with the difference that the sensors are body worn or are implantable. The main objectives in BSNs are extreme low power consumption, ease of use and unobtrusiveness for the patient. Figure 2.3[2] shows a patient equipped with sensors to capture different physiological data such as the blood pressure, galvanic skin response (GSR) and temperature.

The sensors in a BSN can be integrated into fabrics as in Figure 2.5[3] or worn as objects such as finger rings (see Figure 2.4[4]). With the recent advancements in miniaturization, Implantable Medical Devices (IMDs) became highly interesting for the WSN community. These devices monitor and treat physiological conditions inside of the body. Example types of such devices are pacemakers, implantable cardiac defillibrators and drug delivery systems. These devices can now be equipped with wireless capabilities and researchers believe that emerging IMDs will communicate with each other [46].

Below we give a list of common body sensors and their usage for body monitoring.

[2]Adapted from http://dicom.i2r.a-star.edu.sg/ehealth/?q=wban
[3]Taken from [93]
[4]Taken from http://darbelofflab.mit.edu/node/28

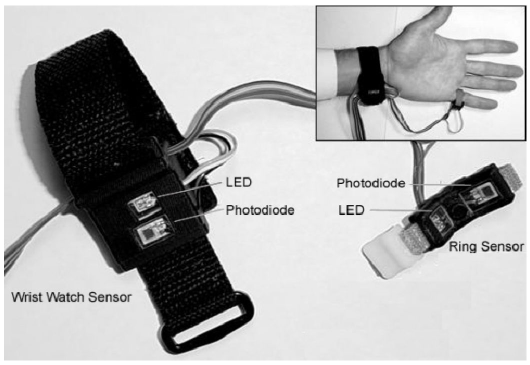


Figure 2.4: Ring Sensor to monitor blood oxygen saturation.

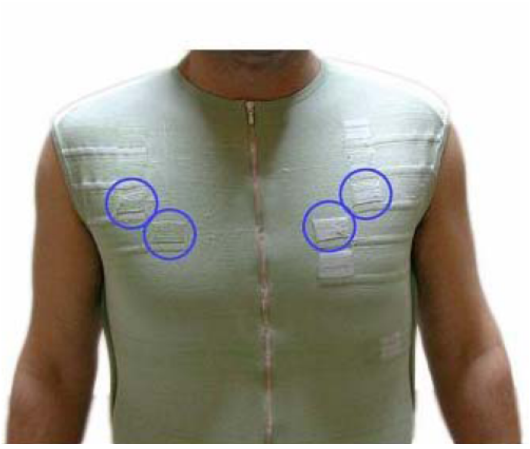


Figure 2.5: Sensor nodes embedded in clothes.

ECG or HRV (Heart Rate Variability) The readings from an Electrocardiogram (ECG) machine are used to monitor ischemia, arrhythmias or palpitations. Significant increase or decrease in HRV can also be used to indicate heart dysfunction or signs of other complications. They can be combined with implantable cardiac defibrillators to offer therapy with very short latencies as the monitoring sensor can collaborate with the defibrillator through wireless communication [46].

Motion Sensors Three dimensional accelerometers provides data on motion and activity levels of the patient.

Respiratory Rate This shows the global respiratory function of the patient. Shortness of breath can be used as an indicator of a chronic disease, e.g. COPD, asthma or progressive lung dysfunction.

Blood Pressure Measuring the blood pressure is useful for people with hyper/hypotension or CHD. It may also be used on post operative patients that are in a critical condition and could experience blood-loss or hemorrhages.

Near-Body Temperature The increase or decrease of body temperature helps identifying diseases (e.g. a fever caused by an infection). Furthermore, the temperature is correlated to the activity levels of a human and such readings help to infer the activities of the patient.

Implantable Blood Glucose Sensor These sensors are important for monitoring diabetic or at risk hyperglycemia patients and can be combined with therapeutic insulin delivery systems to treat diabetes or other glucose related diseases [46].

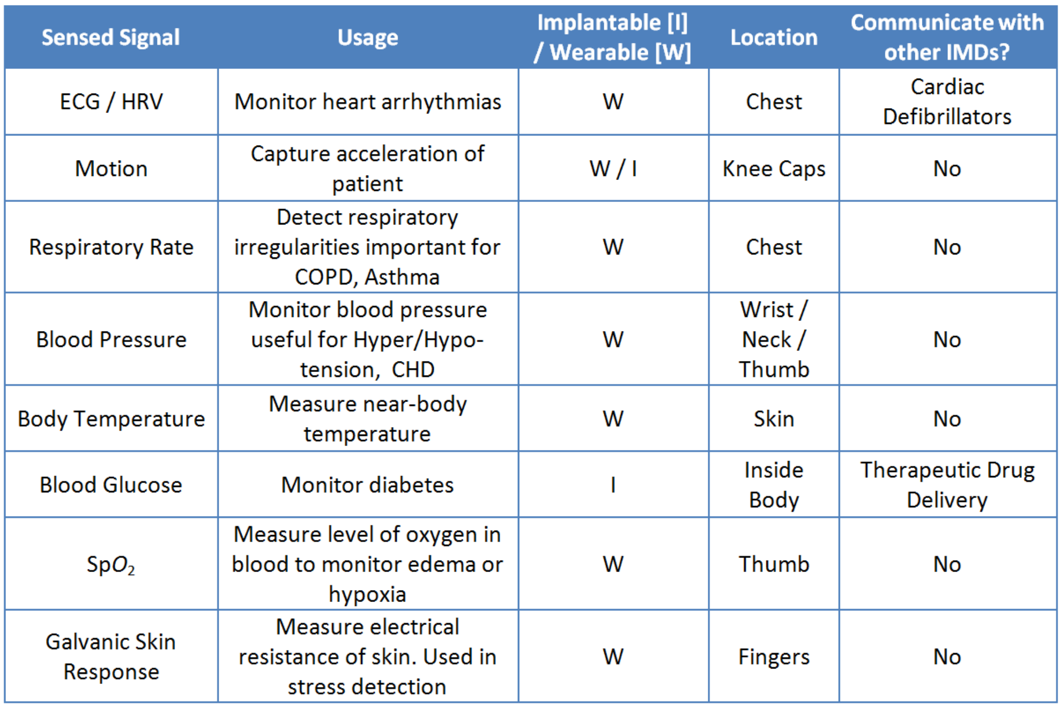
SpO_2 Sensor. An SpO_2 sensor measures the level of oxygen in the blood. It is used to monitor pulmonary oedema or hypoxia. Also, oxygen saturation can provide additional information to infer activity levels. The pulse can also be measured with an SpO_2 sensor.

GSR (Galvanic Skin Response). GSR provides an indication of stress factors (it is used in lie detectors) which is useful when monitoring activity or signs of emotional wellbeing or degradation.

Figure 2.6 holds a summary table of the different signals we described above. In particular we see that most of the Body Sensors are *wearable* which is mostly due to their size and also user acceptance of putting electronics inside their body. Technologies such as MEMS (MicroElectroMechanical Systems) will offer extremely small sensors in the future (e.g. 0.001 to 0.1 [mm]) and we will surely see a shift from wearable to implantable sensors.

2.3.2 Ambient Sensor Network

Ambient Sensor Networks (ASNs) refer to WSNs installed in a patient's home unobtrusively. These sensor networks can be used to monitor a person's social health. They identify the individual's physical location and provide readings on the levels of activity they undertake. Correlating the patient's environmental data with the



Sensed Signal	Usage	Implantable [I] / Wearable [W]	Location	Communicate with other IMDs?
ECG / HRV	Monitor heart arrhythmias	W	Chest	Cardiac Defibrillators
Motion	Capture acceleration of patient	W / I	Knee Caps	No
Respiratory Rate	Detect respiratory irregularities important for COPD, Asthma	W	Chest	No
Blood Pressure	Monitor blood pressure useful for Hyper/Hypo-tension, CHD	W	Wrist / Neck / Thumb	No
Body Temperature	Measure near-body temperature	W	Skin	No
Blood Glucose	Monitor diabetes	I	Inside Body	Therapeutic Drug Delivery
SpO_2	Measure level of oxygen in blood to monitor edema or hypoxia	W	Thumb	No
Galvanic Skin Response	Measure electrical resistance of skin. Used in stress detection	W	Fingers	No

Figure 2.6: Overview of Body Sensors

physiological measurements from a BSN can provide much better appreciation of the context in which the sensing occurs.

To familiarize the reader with ASNs, we provide a non-exhaustive list of the sensors used in "smart-homes". A summary table with the most important characteristics is given in Figure 2.7.

Blob Motion A Blob motion sensor extracts information from abstract video images to generate data on the patient's activity such as his current location, what he is doing and his current posture. To ensure the patient's privacy, the visual images are discarded after processing and only the output of the analysis is kept.

Ambient Temperature Ambient temperature sensors record the room temperature which can be used to prevent health deteriorations for "at risk" patients that should not be exposed to extreme heat or cold.

PIR (Passive InfraRed) Low cost and low power PIR sensors determine patient activity and room occupancy, which can be correlated with the time of the day to analyze behavioral patterns.

Bed Pressure Readings from bed pressure sensors indicate sleeping patterns and the patient's sleep duration. This information can be used to detect depression, neglect or to monitor psychological diseases.

Electromagnetic Electromagnetic sensors are used to detect door usage. Particularly, these indicators can be helpful to determine the patient's activity in- and outside of their home.

Sensor	Usage	Location
Blob Motion	Extract motion, location and activity from image captures	Camera on Walls
Ambient Temperature	Record room temperature	In the room
PIR	Determine patient activity and room occupancy	In the room
Bed Pressure	Monitor sleeping patterns and sleep duration	Between the mattress and the patient
Electromagnetic	Detect door usage	On the door
Sensitive Rug	Patient localization	Embedded or below the carpet
Pressure Mats	Fall prevention or detecting pressure release	On the floor next to the bed or on a sofa

Figure 2.7: Overview of Ambient Sensors

Sensitive Rugs Embedded into the fabric or hidden under rugs, these sensors help locating the patient more precisely.

Pressure Mats Pressure mats are used for example in fall prevention and are usually connected to an alarm system that is triggered as soon as pressure is exercised or released.

Healthcare professionals can make valuable use of the ambient sensor data as it provides them with a better understanding of the context in which the vital-sign readings (from the BSN) have occurred. The caring professionals can monitor a disease's progression more accurately by correlating data from ASNs and BSNs. This is motivated by the fact that the patient's activity is strongly related to his health state.

2.3.3 Remote Patient Monitoring

The reality of constantly monitoring patients, especially those at their homes, opens many possibilities that were not imaginable until now. For instance, cardiac arrhythmias occur for 4% of the population over the age of 60 and reaches 9% for the octogenarians [40]. With the help of Body Sensor Networks, heart rate abnormalities can be detected earlier than ever.

Furthermore, BSNs can help monitoring disease progression and patient response to any treatment initiated. Before BSNs, these tasks were accomplished using "snapshots" of the patient's health status during medical visits. With the help of BSNs however, a platform for continuous monitoring is developed and represent the latest evolution of diagnostic tools [93].

There are a multitude of applications in the healthcare domain that could make valuable use of BSNs, such as monitoring patients with chronic disease, hospital patients and elderly patients. Further information about these scenarios can be found

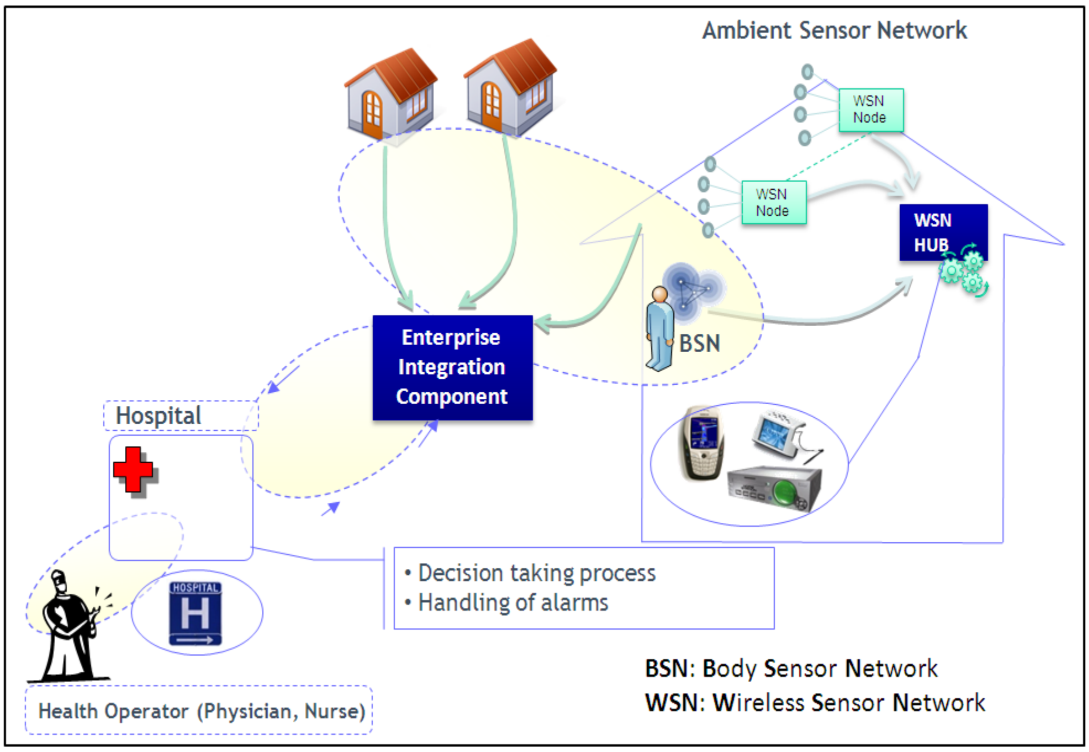

Ambient Sensor Network
WSN Node
WSN Node
WSN HUB
BSN
Enterprise Integration Component
Hospital
HOSPITAL H
• Decision taking process
• Handling of alarms
Health Operator (Physician, Nurse)
BSN: Body Sensor Network
WSN: Wireless Sensor Network

Figure 2.8: Remote Patient Monitoring

in [93]. In the scope of this thesis, we will focus on elderly patient monitoring at home.

Non-intrusive monitoring of "at risk" people such as elderly, may prove invaluable given that life expectancy keeps rising and that the demand for healthcare resources is constantly increasing [19]. A key example of the usefulness of remote elderly monitoring is during the months of extreme weather conditions. When it is either very hot or very cold, elderly patients are at increased risk of hospital admission. This is due to the fact they are not able to seek medical help early enough. An example of this is an elderly person, living alone and acquiring a chest infection, which he fails to identify until the infection requires hospital admission [93].

Also, people may behave differently before getting ill. Behavioral changes might include a decrease of appetite, reduced movement activity and tendency to stay inside of the home. For this reason, correlating ASNs with BSNs allows an early detection of any deterioration in the patient's health condition and may reduce the number of hospital admissions.

In Figure 2.8 we show an example architecture of a WSN enhanced healthcare system. We identify entities that we discussed previously such as WSNs in forms of ASNs, BSNs, the Enterprise Integration Component acting as a mediation layer between the WSNs and software applications.

To assess the patient's health condition, sensors constantly collect data and relay it to a WSN HUB (e.g. a Gateway). The gateway forwards the data to the EIC middleware for storage or further processing. It can also be queried directly for displaying real-time data on mobile devices such as PDA's.

Example Scenario

With the multitude of application scenarios that arise from the remote monitoring scenario outlined above, we present one scenario that makes use of the system architecture depicted in Figure 2.8.

Jon is in his sixties and was diagnosed as an "at risk" patient for cardiac arrhythmias. He lives alone in his home with an installed ASN and is equipped with a BSN that constantly collects his physiological data.

Jon forgot to take his medicine the previous days and so when he walks up the stairs too quickly and feels a sudden pain in his chest, he feels a shortness of breath and his HRV changes. These being typical indicators of an imminent heart attack, the EIC immediately generates an alert and notifies the caring physician of the emergency. On the way to Jon's home, the doctor looks at the medical history to see that Jon forgot to take his medecine and that the physical activity of walking up the stairs had caused these irregularities. Once on site, the doctor can connect directly to the WSN HUB and see all of Jon's vital signs in real-time and determine whether a hospital admission is necessary or not. In this case, Jon just has to rest and take his medecine.

2.4 Conclusion

In this chapter we gave an overview of WSNs with the description of their components. We realized that a mediation layer is required to fill the gap between WSNs and business applications. As such, we selected the EIC amongst the existing ones as the most promising mediation layer between WSNs and business applications.

We then analyzed the impact of BSNs and ASNs in healthcare systems. The elderly remote patient monitoring scenario shows the benefits of the integration of WSNs into business applications. Unfortunately, the prospect of ubiquitous patient monitoring also raises security concerns. Many risks might emerge from integrating new technologies such as BSNs and ASNs into business applications. While not all risks are necessarily harmful or dangerous, some might be and as such cannot be ignored.

In the next chapter, we will identify vulnerabilities and their associated risks in the system described in Section 3.1 following NIST risk assessment methodology [65]. Based on the dangerosity of each risk, we suggest countermeasures for the risk mitigation of the most potentially harmful risks.

CHAPTER 3

RISK ASSESSMENT

"However well organized the
foundations of life may be,
life must always be full of risks."
Edgar W. Howe

Risk Assessment aims at answering the following question: "What can go wrong in my system?". Therefore, it addresses consequences of damage caused to systems, their probability, and countermeasures. Risk Assessment is part of **Risk Management**, which can be seen as a process of identifying and assessing risk and reducing it to an acceptable level for the organization. Furthermore, in Risk Management it is desirable to implement the suggested solution measures in order to maintain that desired level. The levels of acceptance however are not constant and they are subject to revision based on the organization's mission goals.

Risk is a function of the **likelihood** that a certain vulnerability is exploited and the resulting **impact** on the organization [65]. A **vulnerability** is a flaw or a weakness in a system that, if exercised (accidentally or intentionally exploited) results in a security breach. At last, **Impact** is the magnitude of harm that could be caused by exploiting a vulnerability. It relies on the organization's mission goals, and is mapped to the damage (e.g. monetary, reputation, data loss).

Risk Assessment methodologies are aimed at identifying potential threats and vulnerabilities in a system as well as the probability of their occurrences. In addition, the output of this process support organizations with a better Risk Management by identifying controls for minimizing risks.

We identify many methodologies for Risk Assessment in the literature (e.g. [65] [13] [2] [53] [36]). However, some of them are tailored for specific domains. For example, the Sandia methodologies (e.g. RAM-W, RAM-P, RAM-CF) [53] cover public domains such as water facilities, prisons, and chemical facilities. But those methodologies are not general enough to be easily transferred to other domains (e.g. finance, healthcare).

From the literature, we identify two general risk assessment methodologies, the NIST 800-30 [1] guidelines (published in [65]) and the OCTAVE [2] method [13]. These methodologies provide general risk assessment guidelines that can be tailored to any organization or domain. In [27], the authors perform a comparative analysis of the NIST and OCTAVE methods. Their conclusion is that both methodologies are fundamentally similar but differ on some points.

The first difference between NIST and OCTAVE is that NIST devotes one step for determining the probability that a threat source (e.g. an attacker) exploits a potential vulnerability. The notion of probability is not present in the OCTAVE method. The designers of the OCTAVE method believe that unforeseeable changes in the system make the prediction of probabilities too imprecise to be useful [27].

The second differentiator is that NIST includes qualitative and quantitative risk assessment guidance in their publication. The former refers to using qualitative categories (e.g. High, Medium and Low) while the latter provides quantitative values (e.g. \$ 100 or \$ 100'000) for determining the magnitude of impact. OCTAVE does not provide means to evaluate risk based on the likelihood that a potential vulnerability is exploited. Probabilities are inherent to quantitative risk assessment and therefore the OCTAVE method does not provide any guidance for conducting quantitative risk assessments.

The NIST SP 800-30 [65] has become the reference for a valid Risk Assessment methodology and is commonly referred to as the standard that governments (amongst others) use [66]. Also, based on the generality and the widely spread use of the NIST 800-30 guidelines, we will base our risk assessment on those recommendations. Furthermore, we note that the NIST SP 800-30 is freely available and does not require any licensing. Other methodologies such as Cramm [2] have to be purchased and cost as much as \$ 5'788 per paper copy plus an annual license fee of \$ 1'700 [3]. This additional factor validates our choice of picking the NIST 800-30 methodology.

Figure 3.1 shows the seven steps that will be used to perform our Risk Assessment.

- System Characterization: Identification of the system's assets. Assets can be hardware, software, data and staff. Defining the system will set the boundaries and the scope of the Risk Assessment. This is Step 1.
- Threats Identification: This is Step 2.
- Vulnerabilities Identification: This is Step 3.
- Likelihood Determination : Assigns probabilities on potential vulnerabilities being exploited (e.g. how often will this vulnerability be exploited?). This is Step 4.

[1] National Institute of Standards and Technology (see `http://csrc.nist.gov/)`

[2] OCTAVE - Operationally Critical Threat, Asset and Vulnerability Evaluation (see `http://www.cert.org/octave/)`

- Impact Analysis: In Step 5, we determine the impact magnitude (High, Medium or Low) resulting from a successful exercise of a vulnerability. This means we will try to identify if an exploited vulnerability has a great or low impact on the organization's goals.
- Risk Determination: In Step 6, we assess the level of risk of the system. The determination of risk is a function of the likelihood that a vulnerability is exploited and the resulting impact magnitude.
- Control Recommendation: At last, in Step 7, we propose countermeasures for the highest risks based on the analysis we completed at that point.

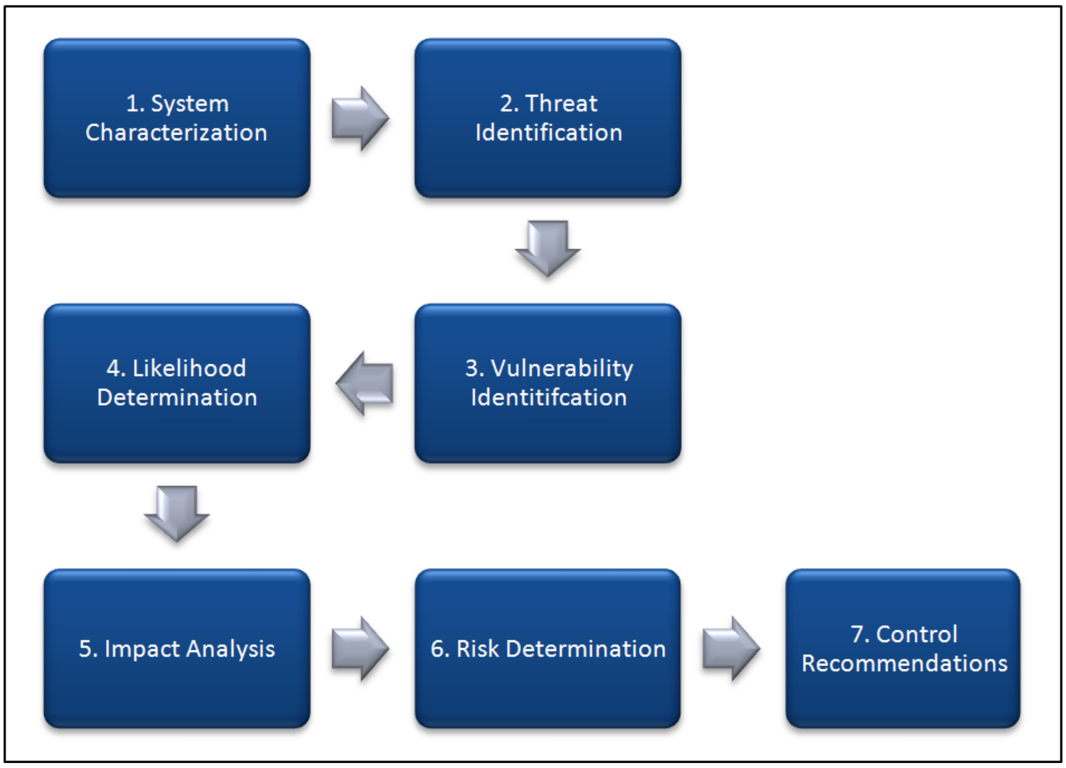

1. System Characterization
2. Threat Identifcation
3. Vulnerability Identitifcation
4. Likelihood Determination
5. Impact Analysis
6. Risk Determination
7. Control Recommendations

Figure 3.1: Risk Assessment Methodology

3.1 Step 1: System Characterization

The first step is the identification of the boundaries of the system with its characteristics. They are classified into the five following categories: (i) Hardware, (ii) Software, (iii) System Interfaces (e.g. Ethernet, Wifi), (iv) Data and (v) People (as depicted in Figure 3.3).

The main components of the complete healthcare monitoring system (the scenario is detailed in Chapter 2.3) are a Wireless Sensor Network (made out of a Body Sensor Network and Ambient Sensor Network), a Gateway (to bridge the sensor world with the rest), the Internet (as main communication medium), a Database (for storing

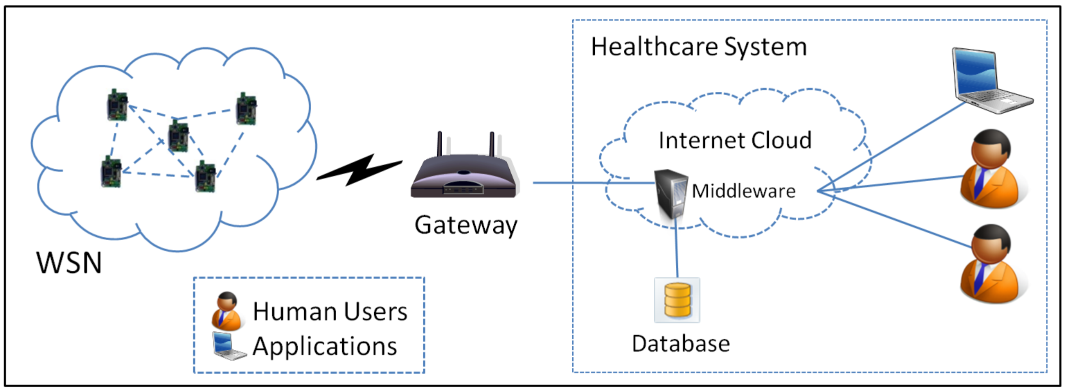


Figure 3.2: System

medical data) and Users (either humans or applications). To see how the different components are linked together, the reader can refer to Figure 3.2. A complete overview of the system characteristics is depicted in Figure 3.3. We provide a more thorough system analysis in the sections below.

Hardware

Sensors

A patient has a Wireless Sensor Network at his home. The WSN consists of a Body Sensor Network (BSN) and an Ambient Sensor Network (ASN). The BSN measures the biological data of the patient (e.g. heart rate, blood pressure, temperature) and the ASN records information about the patient's surroundings (e.g. presence in the room, room temperature). The sensors use the wireless medium to broadcast the sensed information. In Chapter 2.3, we provide a more extensive list of such sensors and their respective usage.

Gateway

Sensed information from the BSN and ASN is transmitted to a gateway. It is in charge of bridging the WSN with the Internet Cloud.

Middleware Platform

The Middleware platform runs the WSN Middleware. As defined in Section 2.2, a middleware is a piece of software. It enhances the WSN with capabilities such as further data processing (e.g. reasoning), sensor data mapping or data storage. The database can be part of the middleware or can be external to it. The data relayed from the *Gateway* is transmitted to the middleware that stores it in a database for future access.

Input Devices

The input devices that can access the recorded data can be PDAs, computers, laptops or any other device capable of communicating with the middleware platform.

Software
The Middleware

As seen previously, the WSN Middleware is software that bridges the WSN with the application world. In Chapter 2.2, we introduced several middleware platforms such as the Enterprise Integration Component (EIC) [42] that is used in the WASP project. The database is an integral part of this middleware.

Applications

Users (e.g. nurses, doctors) control software applications that enable them to access the collected data from the database. The applications run on devices such as computers, laptops, PDAs and need to be connected to the Internet.

People

We classify the people involved with this system into three categories: medical staff, relatives and patients. The medical staffs are made of nurses, doctors and social aids and they are in charge of patient monitoring. The patients live at home, equipped with a WSN as described earlier. The relatives (who also can access the patient's data under certain legal constraints) can be parents, siblings, children or legal guardians.

Data

The sensor data being collected by the WSN contains information about the patient's health status (e.g. blood pressure, heart rate, temperature) and his surroundings (e.g. room temperature, room occupancy) (See Chapter 2.3 for more information on the different). The sensor data is stored in a database inside of the middleware.

In addition, a different database (not part of the WSN middleware) stores the medical file about the patient. This file usually contains past treatments and diseases of the patient.

System Interfaces

There are several system interfaces that we consider: from the WSN to the gateway, from the gateway to the middleware platform and from the middleware to the applications / users.

WSN to Gateway

Inside the WSN, the sensors use the wireless medium to send and receive information. To communicate between each other, the WSN relies on the 802.15.4 protocol [80]. The gateway is the interface between the WSN network and the Internet. It communicates on one side through the 802.15.4 protocol and on the other side it uses the 802.3 Ethernet [79] or the 802.11 WLAN protocol [81] to access the Internet.

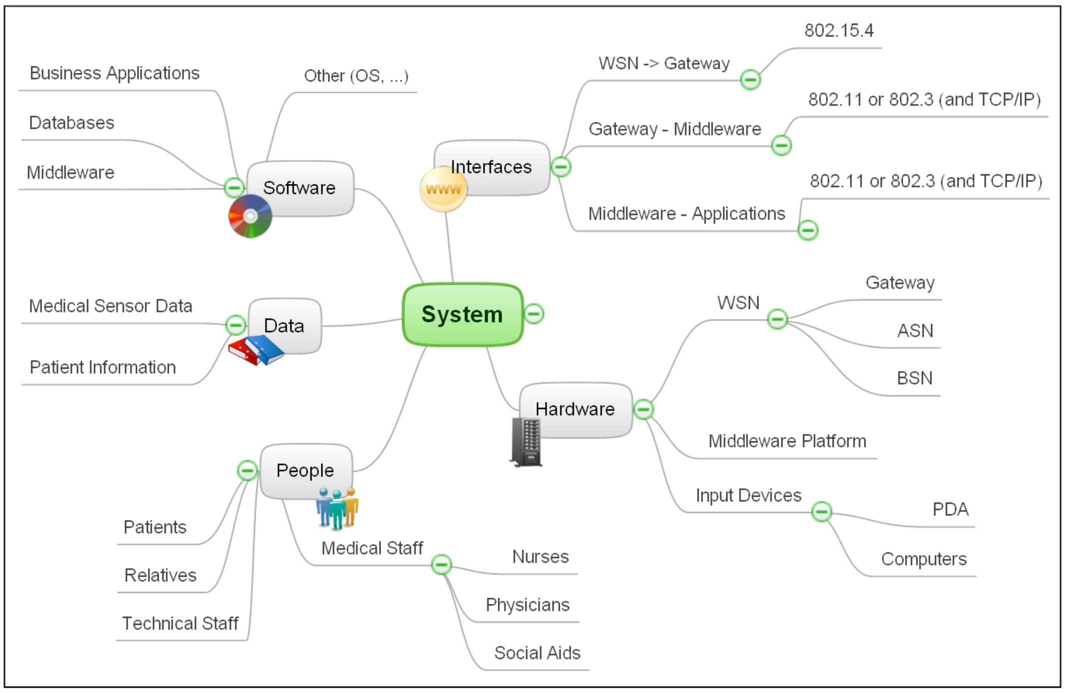


Figure 3.3: System Characterization

Gateway to Middleware Platform

The gateway has two interfaces, one with the WSN and one with the Internet. To properly communicate with the WSN, the gateway uses the 802.15.4 protocol over the wireless medium and the 802.3 or the 802.11 protocol to connect to the Internet. The middleware is connected to the Internet through 802.3 and thus communication (over TCP/IP) between the middleware and the gateway is possible.

Middleware Platform to Applications

The middleware platform (servers) and the applications are both connected to the Internet over TCP/IP (either through 802.11 or 802.3).

Summary

We identified many assets that are part of our system, such as Hardware (e.g. Sensors, Middleware Platforms) and Software (e.g. Databases, Business Applications). In the healthcare system we are considering, the data (see Figure 3.3) is the most valuable asset. The replacement of hardware equipment is costly. But in comparison with the loss of patient data, the inherent cost could be more important. Especially when it comes to compliance with legal requirements (such as the health regulations defined in [32]), the lack of patient privacy can have a negative impact on the organization.

3.2 Step 2: Threat Identification

The second step in the Risk Assessment process is to identify the potential threat-sources that are applicable to the system being evaluated. A threat-source is defined as "*any circumstance or event with the potential to cause harm to an IT system*" [65].

Common threat-sources are organized into three main categories [65]:

i **Natural Threats** (e.g. floods, earthquakes, tornadoes, landslides etc...)

ii **Human Threats** (e.g. any events enabled or caused by humans, unintentionally or deliberately, to harm the system)

iii **Environmental Threats** (e.g. long-term power failure, pollution, etc...)

It is recommended to consider all potential threat-sources (originating from all three categories) while performing the risk assessment, even if they seem unlikely to happen.

The 2006 Global Security Survey conducted by Deloitte [26] revealed that most security breaches within organizations come from humans (79% of successful security breaches are external and 49% are internal originating from human threat-sources). Therefore, in the scope of this assessment, we focus mainly on human threats rather than on natural and environmental ones.

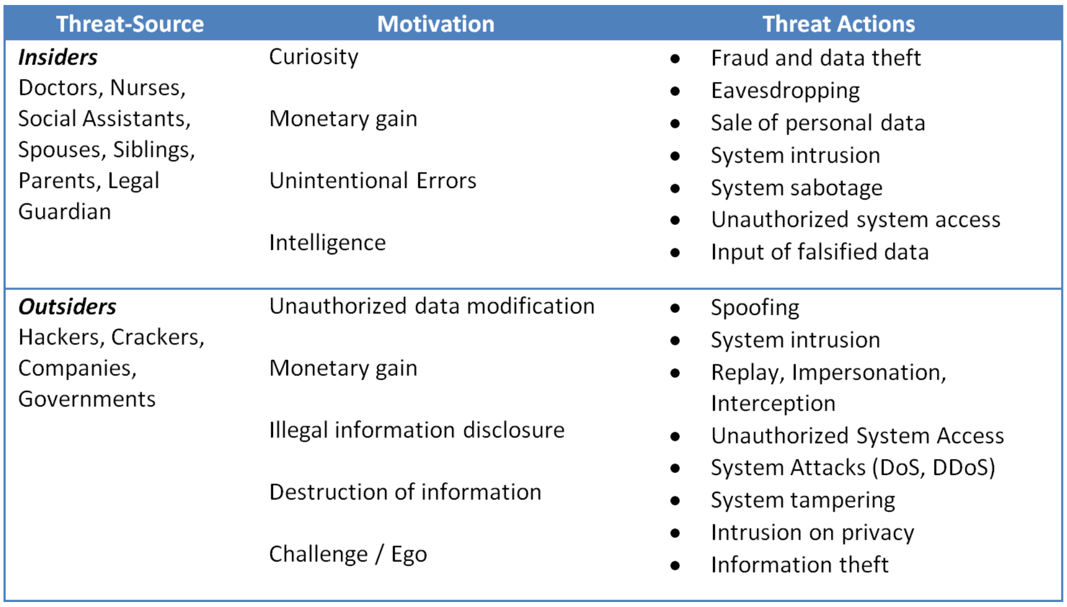
Humans can become threat-sources through ***unintentional*** (e.g. negligence, human errors) or ***deliberate*** acts (e.g. deliberate attacks by malicious persons). Deliberate attacks can be malicious attempts to compromise the IT system in question or benign attacks which only try to circumvent system security.

For example, an employee could install some software that contains hidden malware (a classification of malware and its goal is given in [56]). This malicious software could then create a new vulnerability in the system. Motivation and resources for carrying out an attack make humans dangerous threat-sources [65, Section 3.2].

We can classify the threat-sources into *Insiders* (e.g. doctors, nurses) and *Outsiders* (e.g.Cracker, Hacker).

Inside Threat Sources

Inside threat sources are any entities which are part of the system. These range from patients to medical staff (e.g. nurses, doctors, social aids) to family relatives. Correlated with motivation, these insiders can turn into potentially dangerous threat sources. For a malicious insider, the monetary gain from selling private patient information to insurance companies can be a strong motivation to perpetrate an attack. Insurance companies have a high interest in their clients health status or history. Based on that information, insurance agents can decide if they want to insure the client.



Threat-Source	Motivation	Threat Actions
Insiders Doctors, Nurses, Social Assistants, Spouses, Siblings, Parents, Legal Guardian	Curiosity Monetary gain Unintentional Errors Intelligence	• Fraud and data theft • Eavesdropping • Sale of personal data • System intrusion • System sabotage • Unauthorized system access • Input of falsified data
Outsiders Hackers, Crackers, Companies, Governments	Unauthorized data modification Monetary gain Illegal information disclosure Destruction of information Challenge / Ego	• Spoofing • System intrusion • Replay, Impersonation, Interception • Unauthorized System Access • System Attacks (DoS, DDoS) • System tampering • Intrusion on privacy • Information theft

Figure 3.4: Threat Statement

While money as a motivation can lead to threat actions such as data theft or eavesdropping, other motivational factors such as committing unintentional errors can lead to threats such as system intrusion, unauthorized system access and input of falsified data (see Figure 3.4 for more information).

Outside Threat Sources

Outsiders can be hackers, crackers, companies, governments or any maliciously behaving external entity. Their actions might be motivated by curiosity, ego, monetary gain or data alteration. Some of the resulting threats are spoofing, impersonation, replay attacks, system intrusion and others (see Figure 3.4).

In Figure 3.4 we present a list of identified human threat-sources, their possible motivations and the threat actions. We focus on the technical threats (e.g. system intrusion) and leave out wetware threats (e.g. social engineering, bribery, dumpster diving, bribery). The various threat actions can be classified in the following categories (according to [71]):

Interception. When an unauthorized party gains access to a protected asset. An example of interception is wiretapping to obtain data on a network.

Interruption. When an asset is lost, unavailable or unusable. An example would be destruction of hardware components or erasure of programs.

Modification. When an unauthorized party gains access and tampers with an asset. Examples of modification are altering values in the database or modifying data while it is being transmitted.

Fabrication. When an unauthorized party fabricates counterfeit objects in the IT

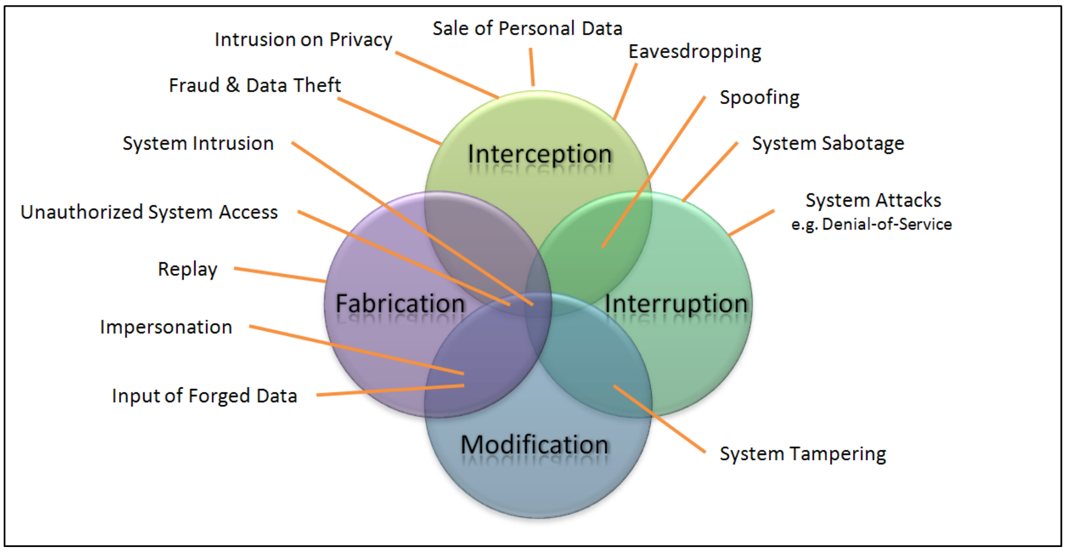


Figure 3.5: Threat Classification

system. The intruder might add records in an existing database or insert forged messages in the network.

In Figure 3.5, we provide a graphical classification of the major human threats we identified above.

3.3 Step 3: Vulnerability Identification

After having established the threat statement outlined in Figure 3.4, we proceed to Step 3 of the risk assessment process, namely vulnerability identification. In this step, we develop a list of system vulnerabilities that can be exploited by the previously defined threat-sources. A vulnerability, as defined in [65], is "*a flaw or a weakness in system security procedures, design or implementation that could be exercised and result in a security breach*".

The list of vulnerabilities in any system is endless and a choice has to be made. Below, we present a small subset of identified vulnerabilities while keeping in mind that the list is far from being exhaustive. For instance, there are new vulnerabilities in software components (e.g. Operating Systems, Web Servers . . .) identified every day by organizations such as NIST I-CAT[3]. A security expert should be aware of those organizations and always take the newest vulnerabilities into account. Seeing that the list of vulnerabilities for software components only is already growing every day, it is impossible to provide an exhaustive list for all aspects of the system (e.g. Hardware, Software, etc . . .).

Vulnerability #1 (Vul #1)

Sensors broadcast data over the wireless medium within the WSN. The transmission range of sensors is smaller than for traditional wireless devices because their

[3]See `http://icat.nist.gov`

transmission power is much lower (see Chapter 2.1). Any threat-source, internal or external, equipped with an antenna can capture all of the data transmitted between sensors and the base station (provided they are within the transmission range of the sensors). Those communication packets contain the patient's private data such as heart rate or blood pressure. This vulnerability can be exploited for data theft, intrusion on privacy or any other threat related to the *Interception* category in Figure 3.5.

Vulnerability #1	Threat-Source	Threat-Action
Sensor broadcast data on the wireless medium	Any internal or external entity	Data Theft, Intrusion on privacy

Figure 3.6: Summary: Vulnerability #1

Vulnerability #2 (Vul #2)

Sensor nodes are physically accessible. They are inside of the patient's home and are scattered around. This means it is possible for a malicious entity to destroy the sensors (e.g. for system sabotage), steal them (e.g. for system tampering, denial of service), or insert new ones (e.g. for spoofing). Once attackers have physical access to a sensor, they can copy its software components (because sensor nodes are not tamper resistant), modify them and put them on new sensor nodes. These sensors can then capture data (e.g. for data theft) or insert forged data (e.g. for system sabotage, input of falsified data) into the system in a truthful manner.

Vulnerability #2	Threat-Source	Threat-Action
Sensor nodes are physically accessible and not tamper resistant	Any internal or external entity	Destruction of sensors, Stealing sensors, Insert new sensors, Capture data, Spoofing, Generate false data

Figure 3.7: Summary: Vulnerability #2

Vulnerability #3 (Vul #3)

Sensors are restricted devices in terms of battery and computation power. For example, an outside attacker can exploit this vulnerability by exhausting the battery of a sensor (see The Ressurection Duckling [84]). This can be achieved by sending a sustained series of useless communications that will make the targeted nodes waste energy on processing. This vulnerability can be exploited through Denial-Of-Service as the sensors become unresponsive if their batteries have been drained.

Vulnerability #3	Threat-Source	Threat-Action
Sensors are restricted devices	External entity	Denial of Service

Figure 3.8: Summary: Vulnerability #3

Vulnerability #4 (Vul #4)

Collected data is stored in clear in a database. Attacks such as viruses and Trojan horses enable an attacker to gain access to the database. Furthermore, if the server hosting the database is physically accessible, the contents of the database can be easily compromised (e.g. by copying all the data from the hard drive). This vulnerability can be exploited for data theft, data modification, destruction of data, fabrication of data and system intrusion.

Vulnerability #4	Threat-Source	Threat-Action
Data is stored in a database without means of encryption	Any internal or external entity	Data Theft, Intrusion on privacy, System intrusion, Destruction of data, Modification and Fabrication of data

Figure 3.9: Summary: Vulnerability #4

Vulnerability #5 (Vul #5)

The data is conveyed from nodes to users, through the gateway and the middleware. And information is stored on database. The data transportation is done without any protection measures. In addition, any component in the system (e.g. WSN, Middleware Platform, and Database) is vulnerable to attacks (e.g. eavesdropping, hijacking). Any component that is taken over by a hostile entity can be modified to steal, forge, destroy and modify sensed data. Furthermore, the communication channels across the whole system are not secured. They do not provide neither confidentiality nor integrity of sensor data.

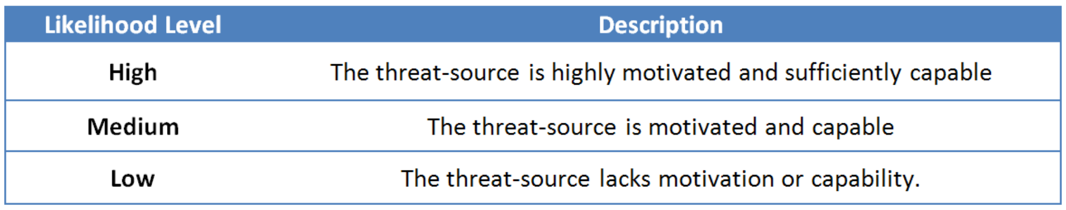
Vulnerability #5	Threat-Source	Threat-Action
Data is transported through the system without encryption measures	Any internal or external entity	Data Theft, Destruction, Modification, Fabrication, Intrusion on privacy

Figure 3.10: Vulnerability #5

3.4 Step 4: Likelihood Determination

In this step, we evaluate the likelihood that a given vulnerability is exploited by an attacker. The likelihood that a potential vulnerability is exercised by a given threat-source is described as High, Medium or Low. Each likelihood level is determined by the attacker's motivation, capabilities and the existence and effectiveness of current controls [65]. Given the small volume of information available to us we do not have a high visibility of existing controls. For this reason, we do not include this factor in the likelihood determination process. Figure 3.11 gives a description of the three likelihood levels.

Any data related to a person (including medical data) is protected by the European Directive 95/46/EC [32]. This law includes protective rules for the processing, handling and distribution of data collected automatically (e.g. through sensors). Organizations have the legal obligation to be compliant with those regulations and as such we consider the sensor data as the most valuable asset.



Likelihood Level	Description
High	The threat-source is highly motivated and sufficiently capable
Medium	The threat-source is motivated and capable
Low	The threat-source lacks motivation or capability.

Figure 3.11: Likelihood Levels Description Table

Using the table outlined above, we can now give a rating to each of the vulnerabilities identified in Step 3.

Vul #1: The main resource needed to eavesdrop on a wireless channel is an antenna that is able to capture the waves emitted by the sensors. Commercial antennas are very cheap (e.g. $ 20) or can be homemade (see [38] for an antenna made out of a plastic bottle and some metal for less than $ 1). The WiSens ZigBee Packet Sniffer[4] contains both hardware and software to capture all communication packets sent between nodes in a WSN. It is more expensive (the kit costs $ 795) than the previously mentioned methods but the kit needs no specific knowledge to be used. An attacker can stand outside of a patient's house with the sniffing equipment and collect all the information coming from the sensors. Based on the low cost for the material and the ease of use for this attack, the likelihood rating we assign for this vulnerability is *High*.

Vul #2: In the healthcare scenario, the sensor nodes are inside the patient's home. One of the problems is that the attacker needs to be physically close to the nodes to perform the attack. This difficulty is more or less challenging depending on the threat-source. An internal threat-source (e.g. Social Aid, Relatives) usually has the key to the patient's home and so entering the patient's home is a trivial task. An external attacker however has to break into the place first. This task may be difficult depending on how the home is secured (e.g. with a burglar alarm or without). In addition, an intruder needs at least a strong technical background to tamper with sensor nodes. Based on the problem of physical proximity and on those required capabilities, we assign a rating of *Medium* for this vulnerability.

Vul #3: To exploit this vulnerability, a threat-source needs a high level of understanding of wireless devices and also specific hardware equipment. There might be controls in place to prevent this vulnerability from being exploited, such as sensors processing signals only originating from trusted sources. Since this attack requires special capabilities and knowledge, we assign it a rating of *Low*.

Vul #4: Data is the most critical and sensitive part of the system. The collected values are stored in a database. Any method that provides access to the database

[4]See `http://www.bzworks.com/wisenssoftware.htm`

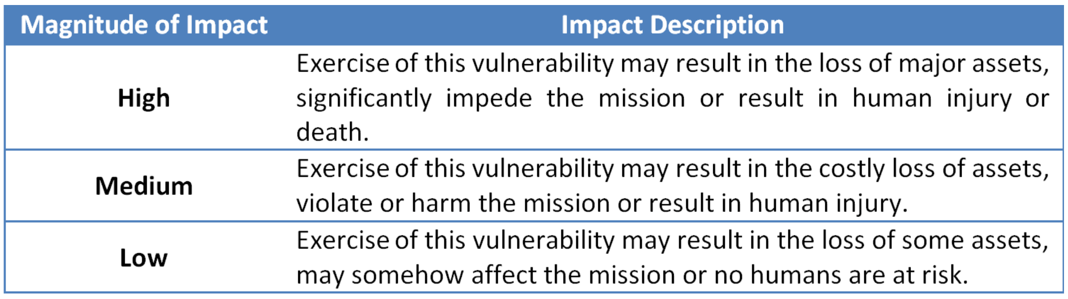
may be exploited to steal, modify, insert or destroy data. Some of the methods to break into a system (and subsequently to the database) include using Trojan horses, viruses or internal staff being negligent (e.g. by sharing their password or by installing potentially unsafe programs). Considering that the most important asset is the data and that there are no special capabilities needed we rate this vulnerability as *High*.

Vul #5: The data is sent in clear through the whole system: from the WSN to the Users through the Gateway and the Middleware. Each component can be compromised and used to perform the attacks (e.g. data theft, data corruption). Additionally, the communication channels do not use any mechanisms to offer confidentiality and integrity, so an attacker can eavesdrop the channel between any two components inside the system (e.g. between the WSN and the Gateway, between the Gateway and the Middleware or between the Middleware and the Users). Given that an attacker can be anywhere in the system (e.g. in the WSN or at the Gateway) and has different methods available to perform his attacks, we assign a likelihood rating of *High* to this vulnerability.

3.5 Step 5: Impact Analysis

In the fifth step of the Risk Assessment methodology, we determine the adverse impact resulting from a successful exploitation of a vulnerability. As previously seen, the resulting damage can be related to monetary, reputation or data loss. The magnitude of damage can be categorized as *High*, *Medium* or *Low*. Using the likelihood ratings from the previous section and the impact analysis, we determine risk levels for each of the vulnerabilities we have identified in the step 3. As mentioned, our risk assessment is data-centric: we focus on sensor data security concerns rather than on hardware or software. Therefore, we determine impact based on vulnerabilities on data rather than on hardware or software.

Table 3.12 describes the qualitative categories that we use namely *High*, *Medium* and *Low* impact.



Magnitude of Impact	Impact Description
High	Exercise of this vulnerability may result in the loss of major assets, significantly impede the mission or result in human injury or death.
Medium	Exercise of this vulnerability may result in the costly loss of assets, violate or harm the mission or result in human injury.
Low	Exercise of this vulnerability may result in the loss of some assets, may somehow affect the mission or no humans are at risk.

Figure 3.12: Impact Levels Description Table

Vul #1: Exploiting this vulnerability results in the loss of confidentiality of the patient's private data. Disclosure of private medical data can result in loss of public confidence or lawsuits filed against the organization because a person's medical data is protected by laws such as [32]. The absence of security measures to guarantee data confidentiality or integrity is an obstacle for the user's acceptance of such technology. We therefore assign an impact magnitude of *High* if this vulnerability is exploited.

Vul #2: Stolen, destroyed or newly inserted sensor nodes results in different problems such as insertion of false data or unavailability of future sensor data. The successful exploit leads to a loss of *Integrity* (if the transmitted data has been modified after it has been sensed and before it has been sent) and *Availability* (e.g. when nodes have been destroyed then no new data is being sent to the servers) and *Confidentiality* (e.g. when rogue sensor nodes relay sensor data to the attacker). In the case of stolen or destroyed sensors, there are also monetary costs inherent to the replacement of the devices. The adverse impact level for this vulnerability is therefore rated as *High*.

Vul #3: Battery exhaustion of a sensor causes loss of availability or transmission of faulty data. When a sensor runs on low battery its precision is impeded. We assume that controls are in place to monitor battery exhaustion such as LEDs showing the battery level or an emitting sound when the battery is low. Based on the fact that data is lost or corrupted even if the incident is quickly detected, we assign an impact magnitude of *Medium* to this vulnerability.

Vul #4: Access to the database means that an intruder can steal, modify, insert and destroy all of the data. Thus, the adverse impact of this security breach respectively results in loss of Confidentiality, Integrity and Availability. This vulnerability has a higher impact than **Vul #1** because when an intruder is eavesdropping on the wireless channel he only has access to current data. However, if the attacker has access to the complete database, he has access to current and past data of all patients (both medical information and sensor data). The impact magnitude for this vulnerability is thus *High*.

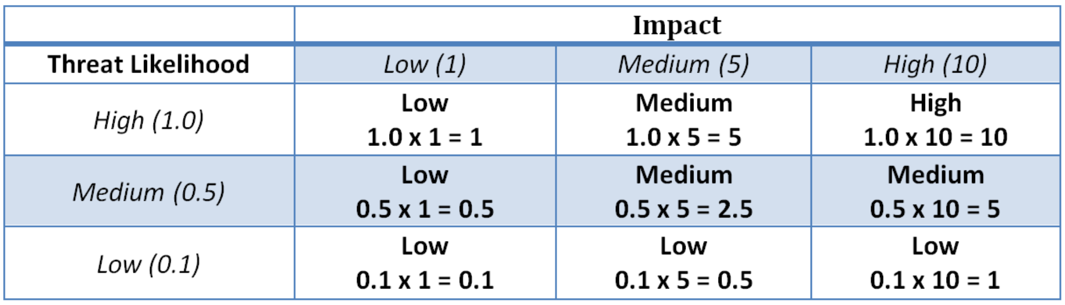
Vul #5: Data travelling through the system without any protection measures results in a loss of confidentiality and integrity. The data can be stolen by eavesdropping the communication channels and can be modified by accessing or tampering the components (e.g. Sensors, Gateway or Middleware). Given the fact that for any exploit of this vulnerability the most valuable asset is lost, we assign this vulnerability an impact rating of *High*.

3.6 Step 6: Risk Determination

In this step of the risk assessment, we assess the level of risk of the system we analyze. Risk is a function of (i) the likelihood of a given threat-source exercising a vulnerability and (ii) the magnitude of impact in case of a successful exploitation.

The risk levels are determined by multiplying the rating assignments of likelihood and impact. Figure 3.13 outlines how the risk levels can be computed based on the inputs from the likelihood (*High*, *Medium*, *Low*) and impact categories (*High*, *Medium*, *Low*). Multiplying categories with each other leads to subjective interpretations of the outcome. For example, does the multiplication of *High* with *Medium* yield *Medium* or *High*?

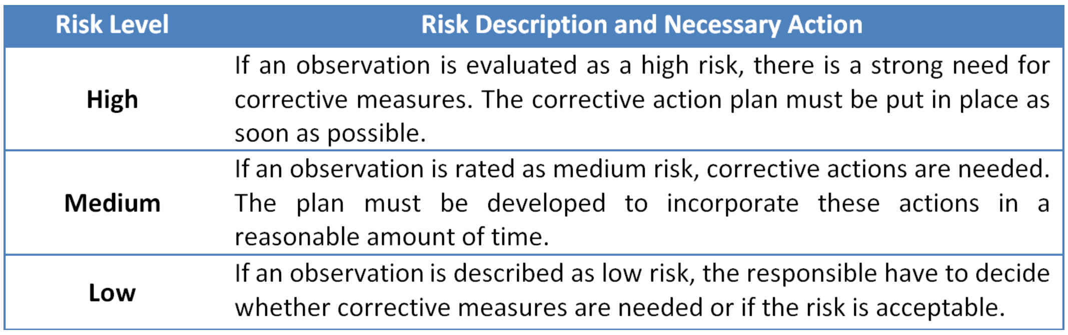
To avoid confusions, we assign an arbitrary metric to the categories; probabilities for the likelihood levels and values for the impact ones (we base our values on the ones suggested by NIST [65]). The probability assigned for each threat likelihood level is 1.0 for *High*, 0.5 for *Medium* and 0.1 for *Low*. For each impact level, we assign 10 for *High*, 5 for *Medium* and 1 for *Low*. The resulting matrix is outlined in Figure 3.13. The final risk levels are *High* when the result is bigger than 5, *Medium* for a result between 1 and 5 and *Low* if the outcome lies in the 0.1 to 1 interval.



	Impact		
Threat Likelihood	*Low (1)*	*Medium (5)*	*High (10)*
High (1.0)	**Low** **1.0 x 1 = 1**	**Medium** **1.0 x 5 = 5**	**High** **1.0 x 10 = 10**
Medium (0.5)	**Low** **0.5 x 1 = 0.5**	**Medium** **0.5 x 5 = 2.5**	**Medium** **0.5 x 10 = 5**
Low (0.1)	**Low** **0.1 x 1 = 0.1**	**Low** **0.1 x 5 = 0.5**	**Low** **0.1 x 10 = 1**

Figure 3.13: Risk Level Matrix - High (> 5), Medium (> 1 to 5), Low (0.1 to 1)

The description for each risk level is given in Figure 3.14.



Risk Level	Risk Description and Necessary Action
High	If an observation is evaluated as a high risk, there is a strong need for corrective measures. The corrective action plan must be put in place as soon as possible.
Medium	If an observation is rated as medium risk, corrective actions are needed. The plan must be developed to incorporate these actions in a reasonable amount of time.
Low	If an observation is described as low risk, the responsible have to decide whether corrective measures are needed or if the risk is acceptable.

Figure 3.14: Risk Description Table

Using the risk level matrix we just defined, we can determine the risk levels for each vulnerability we have identified. The resulting Risk Matrix is shown in Figure 3.15. From that matrix, we identify vulnerability #3 as a **Low Risk**, vulnerability #2 exhibits a **Medium Risk** while vulnerabilities #1, #4 and #5 present the **Highest Risk**. In the next step, we identify controls for the risks.

3.7 Step 7: Control Recommendations

In the previous steps, we identified and rated some of the major risks in the system we analyze (see Figure 3.15). In this final step, we provide controls (countermeasures) to mitigate some of the identified risks. Risk Mitigation covers the means to

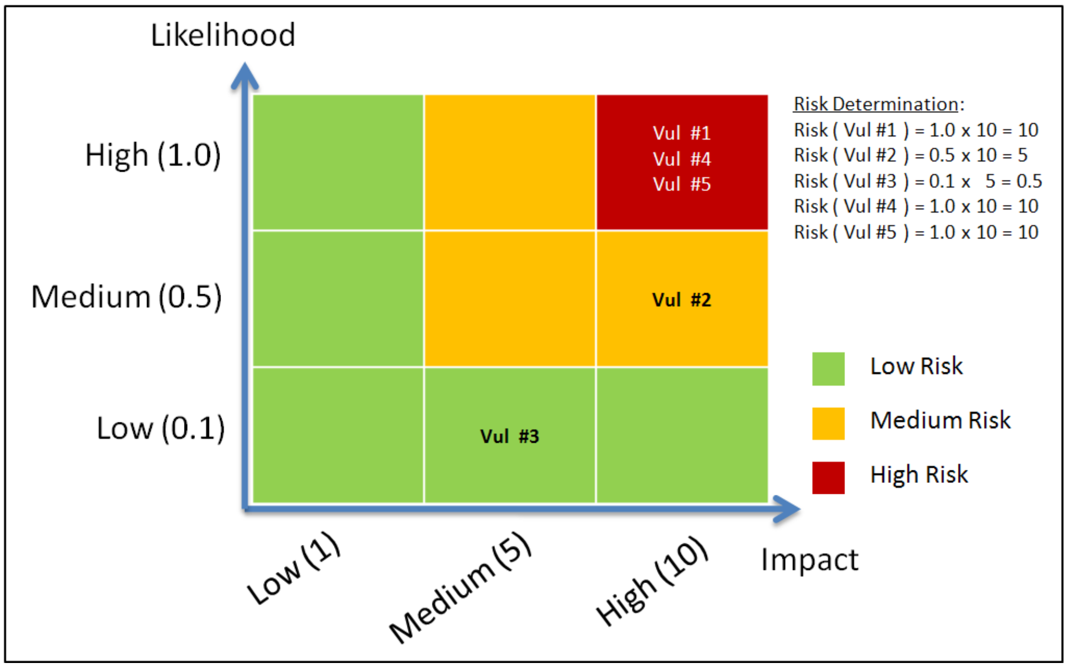


Figure 3.15: Risk Description Table

reduce the risk level to an acceptable one by reducing the likelihood of occurrence, the impact or both.

Low Risks

Vulnerability #3 (sensors are power restricted devices) portrays a Low Risk that is acceptable. Offering countermeasures is difficult because the restriction on the sensors is inherent to nodes themselves. To increase the battery or computational power of a sensor is not a trivial task.

Medium Risks

Vulnerability #2 (sensor nodes are physically accessible and not tamper resistant) presents a Medium Risk. Unfortunately, there are not many technical controls that can be suggested to mitigate this risk. The most effective countermeasures are the training of patients to lock their doors and the installation of a burglar alarm.

High Risks

Vulnerability #1, #4 and #5 present a High Risk. Sensor Data can be secured against loss of confidentiality and integrity between each component (e.g. between sensors, between the gateway and the middleware). This security protection is achieved by using mechanisms inherent to each component such as the AES-CBC-MAC construction (see section 4.4) inside of the WSN. However, such controls only partially mitigate the risk of vulnerabilities #4 and #5. This is because hop-by-hop

[5] security does not prevent data from theft or alteration if components are compromised. This means that even if all communication channels are secured but that a component is malicious, the data is not secure.

Based on those findings we identify a clear **lack** of security for sensor data. Since hop-by-hop security is not mitigating the risks in an acceptable way, we envision end-to-end security as being the solution for the secure integration of WSNs into business applications. This means that sensor data is protected against unauthorized disclosure and loss of confidentiality *from* the sensors *to* the users.

To mitigate the risk of Vulnerabilities #1, #4 and #5 we present a security scheme (see Chapter 4) relying on cryptography [83]. The scheme offers end-to-end confidentiality from sensor nodes to data consumers.

3.8 Conclusion

In the previous sections we performed a *Risk Analysis* using the NIST 800-30 methodology [65]. We went through a lengthy process of identifying threats, vulnerabilities, evaluating likelihoods, impacts and risk to determine what can go wrong and what the associated consequences are. One of the most difficult parts of the risk assessment process was the determination of likelihood and impact ratings that each vulnerability is given. System characterization and threat identification are based on our knowledge, whereas likelihood and impact evaluations are based on our predictions. This is a subjective task and the outcome should be treated with caution.

We believe that *Risk Analysis* is an important step in both the design and running phase of a project. Without it, an organization might spend too many resources in terms of computational power, money or time on eliminating risks that might not require any action at all. With *Risk Analysis* however, organizations get a solid base to determine how to distribute their resources in order to mitigate risk and secure their systems.

In order to obtain more objective ratings in the process, we would require much more statistical data (e.g. about likelihood and impact of possible attacks). However from our findings, not many such statistics exist about vulnerabilities in WSNs. In many articles such as [22], the authors present attacks and countermeasures for WSNs, but rarely consider the likelihood or the impact any such attack could have.

As mentioned previously, we identified a clear need for end-to-end confidentiality on sensor data. To fill this requirement we present a security scheme in chapter 4 that uses cryptography to achieve end-to-end security from WSNs to users. We also analyze the scheme's security which had not been performed by the authors of the scheme.

[5] By hop-by-hop security we mean that the data is secure during the transmission between any two entities. However the entities are able to read the data before forwarding it to the next one. This is in contrast to end-to-end security where the intermediate actors are not able to decrypt the data. Only the originator and the end receiver are able to read the data.

It is important to notice that *Risk Analysis* refers to the system only whereas *Security Analysis* refers to protocols or mechanisms only.

CHAPTER 4

SECURITY SCHEME

"The only truly secure system is one
that is powered off, cast in a block of concrete
and sealed in a lead-lined room with armed guards."
Gene Spafford

Our risk assmessment raises the importance of sensor data privacy. We have shown that the impact on the system is significant if this vulnerability is exploited by an attacker.

In this section, we address this requirement based on end-to-end confidentiality of sensor data relying on [83]. This security scheme insures end-to-end confidentiality of sensor data from its producer (e.g. node) to the business applications. Based on a symmetric encryption scheme, a unique key is used for encryption and decryption of sensor data. This security scheme is of particular interest for us, because it addresses resource restriction on sensor nodes.

In order to manage encryption keys between sensor nodes and business applications, the authors propose an access control mechanism. In combination with a classification of sensor data types, and business application roles (based on their credential), an authorization class tree is defined.

Further details on the scheme can be found in section 4.2. As mentioned in the previous chapter, the authors only provide a rough security analysis of their scheme. We provide a security analysis of the security scheme in section 4.3.

Sensor nodes produce, on a broadcast medium, highly diverse data, which is often very sensitive. There may be many users using the system, each with different access rights to access the wireless sensor data produced by sensors. This problem of multiple-resources/multiple-accesses is conventionally solved using access control.

Access control is the ability to permit or deny access to a particular resource by a particular entity (e.g. User, Program) [67].

In a classical access control mechanism, an entity (e.g. User, Program) authenticates itself, receives a credential, produces that credential to the resource manager and receives a special stream of data the requesting entity is authorised for.

Authenticators are based on at least one of the following factors:

Something you know. Passwords or Personal Identification Numbers (PIN) usually make up this category.

Something you have. This includes smart cards, or any token based device.

Something you are. Any body part with sufficient biometric traits can be used as an authentication factor. The most commonly used ones are fingerprints, voice recognition, retina or iris scans.

After being authenticated, the user needs to be associated with rights of what he/she can do in the system. These rights are usually a variation of the following basic access types: Read (R) and/or Write (W).

Although there are many solutions to deal with access control in the literature [89], none of them are suitable in the context of Wireless Sensor Networks. This is due to the technical constraints that exist on the nodes, as mentioned in Chapter 2.1.1. On top of those technical limitations, the sensor data is generated and transmitted in real-time, which makes the establishment of multiple data streams with the sensor nodes very difficult, if not impossible.

The need for a hierarchical access control scheme

In many scenarios where WSNs are used, sensor nodes are required to sense a large range of different data types. In the remote patient monitoring scenario (see Chapter 2.3), ambient sensors sense room occupancy, room temperature, motion and activity while body sensors sense blood pressure, HRV, GSR, SpO_2, glucose rate and others.

The sensed data is usually highly sensitive and has different “levels” of sensitivity. For instance, the room occupancy in a hospital might not be highly sensitive as it can be determined quite easily by other means such as light usage or by sight, however the ECG chart of a patient is highly private and should only be accessible by a small finite set of users.

Furthermore there are many actors which might be interested in the delivered sensor data, e.g. nurses, patients, relatives, general practitioners (GP) and specialists. These actors can be organized in a hierarchy depending on the access rights they are entitled to. Low sensitivity data can be accessed by low level entities and conversely high level sensitivity data can be accessed by high level entities in the hierarchy only.

In [83], the authors propose a hierarchical access control scheme based on cryptography. The data is encrypted at the source and thus its access is intrinsically restricted. This approach enables the nodes to publish encrypted data without being concerned about its present or future consumers. Thanks to the key generation mechanism proposed, multiple consumers with different access rights (e.g. nurse and physician) can converge on the same decryption key if their privileges allow them to. The presented scheme satisfies two very desirable goals for WSNs: it does not use complex operations and it is independent of the data consumers. The scheme description is given in Section 4.2.

4.1 Attacker Model

In this section some terminology will be introduced. We then present an adversary model based on the popular Dolev-Yao model [30]. It is useful to define the model under which an attacker can act, because it would be difficult to study or analyze the security of a security mechanism without relying on a well-defined attacker model. Indeed, how can we secure a system if we do not know what an attacker is able to do?

4.1.1 Terminology

Selfish and malicious are two terms that falsely can be considered as synonyms. A selfish (self-caring, egoistic) person could be considered as "nasty/mean" (malicious). However a malicious person is not necessarily selfish, as his goal might be to just do harm, without trying to gain a personal advantage. However, in the context of wireless networks, and particularly in wireless sensor networks, it is appropriate to separate them clearly.

- A **misbehavior** is the action of a party or group of parties consisting in deliberately departing from the standardized or otherwise publicly available prescribed behaviour in order to reach a specific goal [20, Def 3.1].
- A misbehavior is **selfish** if it aims at obtaining an advantage that can be quantitatively expressed in units (such as bit rate, joules). Any other behaviour is considered to be **malicious**. [20, Def 3.2]

  For example, an attacker trying to increase his share of the bandwidth (at the expense of others) is selfish. However, a denial-of-service attack is considered as malicious.
- An **attacker/adversary** is a party or a group of parties that reflects malicious and/or selfish behavior.
- An **attacker/adversary model** enumerates the precise cryptographic information available to the adversary carrying out an attack [85].

4.1.2 Dolev-Yao

The first popular adversary model in security is the one defined by Dolev-Yao in 1981 [30]. This model makes several assumptions about the attacker. In particular,

- The attacker can be a **legitimate party** of the system. When a registered user in a system starts to misbehave, he is considered an attacker.
- The attacker can **send** and **receive** any messages in the system.

- The attacker can be **anywhere** in the system.
- The attacker **cannot break** the used cryptographic primitives.

However, this model is very general and needs to be adapted to the system at hand. Before knowing which modifications have to be performed to the model above, clear and concise security goals have to be set.

4.1.3 Security Requirements

Legal regulations on healthcare [32] aim at preserving the patient's privacy regarding their electronic medical data. The derived security goals are thus the protection against service disruption (e.g. hindering the medical staff to access medical data), destruction of, or modification of data, data theft and access control on sensor data and alerts.

The following security requirements are therefore defined in the context of the healthcare scenario in order to achieve these goals. There may be other security requirements such as preserving the anonymity of the patient's identity but those requirements are out of scope of this thesis.

Sensor Data Availability. This requirement deals with service disruption. Availability of sensor data is of high importance if one considers monitoring a patient remotely. In order to successfully and quickly respond to an emergency case, the sensor data must be available to physicians at anytime and without much latency. Failure to do so could result in the patient's health degrading or in the worst case, could come death for example because the doctor did not have access to the vital signs of the patient. Also, it should be possible to alert a member of the medical staff at anytime.

Sensor Data Confidentiality. This requirement deals with the preservation of the data privacy. The objective is to ensure confidentiality of sensor data exchanged between the patient's sensor nodes and the data consumer.

Sensor Data Integrity. This requirement deals with the assurance that sensor data has not been tampered. Indeed, the modification of sensor data could greatly endanger the patient.

Access Control. Sensor data and the generated alerts must only be accessible by authorized users (e.g. Nurse, Physician, GP). The access control on sensor data should also be flexible enough to adapt itself depending on the current context. For example, every physician should have access to the patient's medical data in case of an emergency.

4.1.4 Adapted Adversary Model

Our system (Figure 4.1) consists of a Wireless Sensor Network (WSN), a network on which the data is transmitted (Wireless and the Internet) and several end users (e.g. nurse, family relatives, physician).

After having identified four security requirements in the previous section, the adversary model defined in Section 4.1.2 can now be fine-grained in the following way:

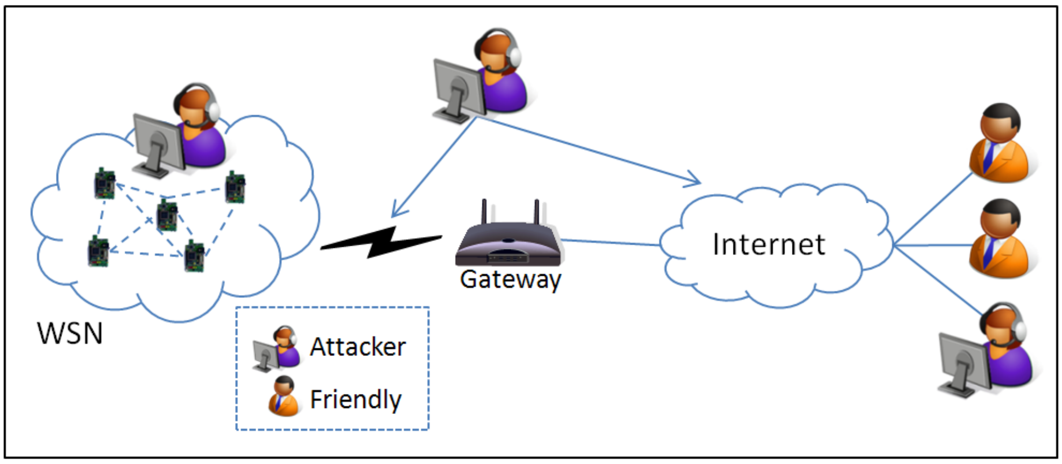


Figure 4.1: System

- The attacker can **eavesdrop** all communications in the system, be it when the data is sent over a wireless channel, or through a wired line. Unlike wired networks, where physical access to the wire can usually be protected (e.g. by putting the wire into walls or underground), in wireless communications the attacker just needs to be in proximity of the device range to be able to eavesdrop.

- The attacker can be **anywhere** in the system. As depicted in Figure 4.1, the attacker can either be, in proximity of the Wireless Sensor Network, anywhere between the gateway and the end user, or be an actual user of the system.

- The attacker can be an **authorized** user with a certain authorization level (i.e. physician in the healthcare scenario).

- Nodes in the WSN can be **compromised**. A node is said to be compromised when its underlying cryptographic material has been disclosed to the attacker. A node can also be compromised in a natural way, if its hardware fails for example.

- Cryptographic primitives used in the system are considered to be **secure**. By secure we mean that it should not be feasible for an attacker to break the cryptographic algorithms used in the cryptographic systems.

4.2 Scheme Description

As mentioned previously, we now introduce an existing security scheme [83] that offers end-to-end confidentiality of sensor data from the sensor nodes to the business applications. It guarantees the fulfilment of this requirement by using symmetric key encryption. Since both encryption and decryption keys are the same, a problem of key distribution occurs. The authors introduce a central Access Control Module in charge of distributing the necessary cryptographic material needed to generate the keys.

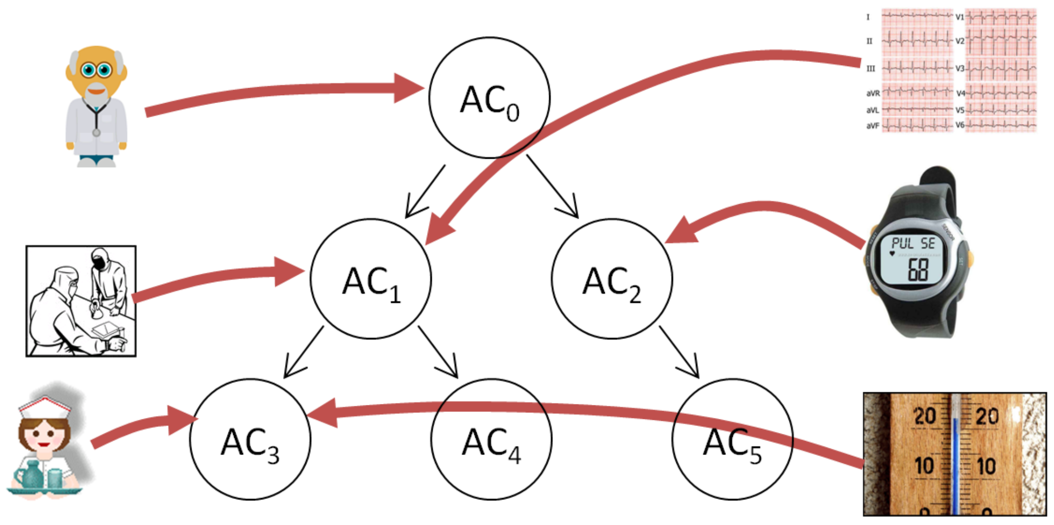


Figure 4.2: Example mapping of data types and roles to the Authorization Class Tree

In this section we discuss the key generation algorithm and the role of the Access Control Module. Further information can be found in the scheme's paper [83] which we conveniently added as addendum to this thesis.

4.2.1 Key Generation

First, based on their level of sensitivity, a classification of sensor data is established. Data whose disclosure does not raise high privacy issues is mapped on low authorization classes while highly sensitive data is mapped to high authorization classes. An example mapping is given in Figure 4.2.

The encryption mechanism uses the One-Time Pad as cryptographic algorithm. To comply with its requirement of never reusing a key twice, each generated key is unique. The encryption key is the output of a hash function h and is computed using the following formula for authorization class AC_i:

$$Key = h(K_T||Tgw||AC_0||AC_1||...||AC_i||HMAC_{Tgw}(seq||id))$$

First a secret pseudo-random seed K_T is picked and the hash is updated with both K_T and $Tgw = HMAC(K_T, GW)$ where GW is a grant window during which the keys are valid.

Second, a hash tree with pseudo-random values is created like in Figure 4.7, where AC_0 is a public pseudo-random value. We see that now the Authorization Classes from Figure 4.2 have been mapped to different cryptographic material. Following the hierarchical organization of the sensor data classification, this material allows to derive keys for any lower authorization class. The hash h is further updated with a concatenation of AC values where the value of each

$$AC = HMAC(AC_{levelabove}, AC_{currentlevel})$$

until the authorization class AC_i is reached. For example in figure 4.7, we have $V(AC_3) = HMAC(V(AC_1), AC_3)$ which is the hash value for authorization class AC_3.

Finally, the h is finalized with the output of $HMAC_{Tgw}(seq||id)$ where $seq||id$ is the concatenation of the sequence number and the sensor id. Each sensor has a different id and the sequence number is incremented for each data packet sent.

4.2.2 Access Control Module

The Access Control Module is in charge of distributing cryptographic material to the sensor nodes and to the business applications. Based on the *Credential* of business applications, the ACM assesses their authorization classes and provides them with the necessary decryption material. This cryptographic material includes two elements: (i) the partial hash value and (ii) Tgw.

For example, the cryptographic material sent to sensor nodes and business applications which are on Authorization Class AC_2 during a grant window GW is:

$$\begin{cases} Partial\ Hash = h(K_T||HMAC(K_T, GW)||AC_0||AC_1||AC_2 \\ Tgw = HMAC(K_T, GW) \end{cases}$$

Using this material, both the sensor nodes and the business applications just have to perform the last step of the key generation to converge on the same encryption respectively decryption key. As explained previously, this step is the update of the hash h with $HMAC_{Tgw}(seq||id)$. Additionally, this cryptographic material can be used to derive any keys for lower authorization classes then for which the material was issued.

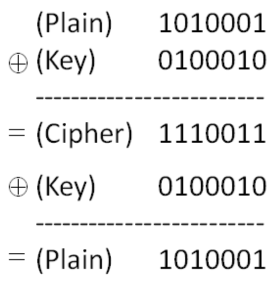
4.3 Security Analysis

The authors in [83] have barely analyzed the security of their scheme. Thus, it seems appropriate to analyze the scheme's security in greater detail. We can classify the possible attacks into two categories: outside attacks and inside attacks.

In an outside attack, the adversary is not inherent to the system we consider. Examples of outside attacks are known-plaintext attacks, compromise sensor nodes, hijacking the Access Control Module and performing a denial-of-service attack on the ACM.

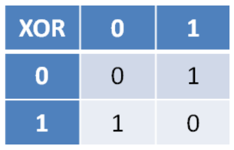
Conversely, in an inside attack, the attacker is part of the system. Typically an inside attack occurs when at some point in time a legitimate user starts to misbehave. An example of an inside attack is when a user tries to escalate his rights in order to gain access to a resource he is not entitled to.

For each attack we mentioned above, we will provide a deeper analysis in the next part.



```
  (Plain)    1010001
⊕ (Key)      0100010
  -------------------------
= (Cipher)   1110011
⊕ (Key)      0100010
  -------------------------
= (Plain)    1010001
```

(a) OTP



XOR	0	1
0	0	1
1	1	0

(b) XOR

Figure 4.3: One Time Pad

4.3.1 Outside Attacks

4.3.1.1 Known Plaintext Attack

The Attack

An outside attacker (i.e. a non-registered member of the network), after eavesdropping sensor data, can try to perform a known-plaintext attack [57] in order to retrieve the key used for encryption. In a known-plaintext attack, the adversary has a quantity of plaintext and corresponding ciphertext. This is a reasonable assumption in the case of WSNs, as the attacker might deploy his own sensors in order to acquire the clear value.

Countermeasure

The scheme makes use of the One Time Pad [55] as cryptographic algorithm. It consists of combining the plaintext with a random key (called pad) that is as long as the plaintext and never reused twice. When the key is truly random, used only once and kept secret, then the OTP provides perfect secrecy (proven in [77]. As a reminder, Figure 4.3(a) shows how the encryption and decryption parts of an OTP work.

The symmetric key used to encrypt a particular plaintext can easily be determined by performing a XOR operation (see Figure 4.3(b)) on the ciphertext and the plaintext, or $Key = Cipher \oplus Plaintext$.

In symmetric-key encryption, the encryption key K_{ENC} and the decryption key K_{DEC} are the same. In contrast to this, in asymmetric-key encryption, the encryption and decryption keys (K_{ENC} and K_{DEC}) are different. It is considered as a "hard" problem to compute the decryption key K_{ENC} from the encryption key K_{DEC} [20, A2].

The retrieved key does not however represent the whole key of the OTP, but only a small fraction of it. One of the requirements that ensure perfect secrecy in an OTP is that the key must have the same size as the whole plaintext, and never be reused twice. This is the case in this scheme, since the whole key is made out of many *sub-keys* that each are statistically independent.

At best, this attack is able to retrieve the small *sub-key* that was used to encrypt just this chunk of data. It is thus clear that the retrieved key cannot be used to decrypt any other blocks of data or to derive any part of the whole key.

Since the perfect secrecy of the OTP has been proven by Shannon in [76], other attacks such as ciphertext-only or chosen-plaintext attacks are not possible.

Data Origin Problem

Once the attacker has performed a known-plaintext attack and retrieved the *sub-key*, he is able to insert forged messages into the system. However, the scheme we are analysing only deals with access control and inherently with confidentiality and not with data origin. This problem is therefore out of scope of this analysis.

4.3.1.2 Compromised Sensor Nodes

Sensors can be compromised in two ways, intentionally or unintentionally [73].

A sensor node is **unintentionally** compromised if some of its hardware is failing or if the battery is low. In this case, the sensor's accuracy might diminish and the acquired data might not be trustworthy any longer.

A node is **intentionally** compromised when an adversary actively tries to obtain the cryptographic material on the node. The attacker also could intentionally drain the battery of the sensor to render it unreliable or unavailable.

Since both of these ways can cause damage to the network, we can consider them as attackers.

Attack

In Wireless Sensor Networks, sensor nodes are not tamper-resistant [92] due to the fact that they have low-cost and low-power requirements. Due to restrictions on sensor hardware and software, the security protection becomes weaker. An attacker can therfore easily capture a node (e.g. by stealing the node). Once a sensor node has been compromised, all of its cryptographic material (all secret information, keys) are disclosed to the attacker.

After obtaining the cryptographic material of a node, the attacker can disseminate forged data in a secure and valid manner or insert new malicious nodes into the network.

Countermeasure

There are different approaches to deal with detecting compromised nodes: sensor node failure detection, alert based identification of compromised nodes, reputation systems and trust based frameworks.

Failing Sensor Nodes

A sensor node is failing if it is unresponsive, starts sending incorrect data due to malfunctioning or when it is maliciously compromised. To improve resilience to sensor node failure, some approaches propose that sensors detect their own failure [47] (e.g. by detecting physical malfunctions or battery usage) or their neighbours failure [44].

Alert Based Identification of Compromised Nodes

In [91], the authors propose a framework for identifying compromised nodes based on alerts. They develop an alert reasoning algorithm where each node evaluates their security estimate on their neighbours. The nodes raise alerts when a problem is detected and the base station of the WSN intercepts those alerts.

Reputation Systems

Reputation systems try to identify compromised nodes by analyzing their behavior. The reputation of a node is the collection of ratings maintained by others about that node. There are many approaches for reputation based systems ([18], [37], [59]) and they usually base their reputation evaluation on the nodes and not on the data.

In some cases, like in the healthcare domain (see Section 2.3), the data itself is as important as its origin, if not more. From this point of view it is important to consider both the **trust** and the **origin** of the data, rather than focusing only on the origin.

Trust Based Frameworks

Trust based frameworks aim at establishing a confidence level either between sensor nodes or on the data itself. The former can be referred to as entity-centric trust frameworks ([88], [86]), while the latter can be considered as data-centric trust frameworks ([73],[92]). Trust is often described as the expectation of cooperative behavior, or in other words, the expectation that the sensors will deliver non-compromised data.

In the healthcare scenario, WSNs deliver sensitive data about the patient's health status. While trust on the identity of the nodes is important, trust on the data itself is as useful. Data-centric frameworks, such as [73], use a-priori trust relationships in nodes as one of the parameters to establish trust on the data itself.

Access Control Module

The Access Control Module (ACM), introduced in 4.2, is responsible of distributing the cryptographic material to the sensor nodes and delivering grants to authenticating users. Once a compromised node has been identified (using one of the methods

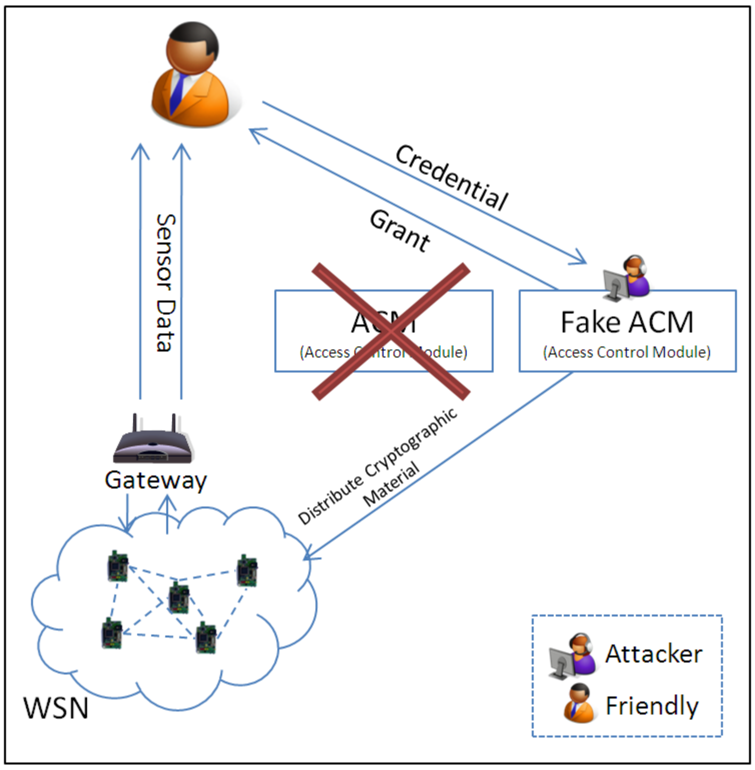


Figure 4.4: Hijacked System

mentioned above for example), the ACM can issue new cryptographic material to non-compromised nodes using the rekeying method detailed in 4.2. As soon as the nodes start to use the new cryptographic material issued by the ACM, the attacker can no longer insert valid forged messages into the network.

4.3.1.3 Hijacking the Access Control Module

Introduction

An attacker can impersonate the real Access Control Module in order to harm the system (as depicted in Figure 4.4). There are many ways to successfully execute this attack: DNS Spoofing, MAC Spoofing, IP Spoofing or Physical Capture. We will briefly describe each of the attacks and their solutions below.

When the attacker succeeds in impersonating the ACM, the whole system breaks down: **(i)** legitimate users sending their credentials to the compromised ACM (the fake one or the captured one) will receive *fake* Grants in a seemingly truthful manner and **(ii)**, the decryption of sensor data using these fake grants will fail and thus render the system unfunctional.

Furthermore, the adversary then has all of the authenticating user's credentials in his possesion because the attacker can store the credentials he “stole”. Depending on the type of credential the system is using (e.g. a simple login/password), the attacker can use those stolen credentials to mount other attacks, such as impersonating users. We can however safely assume that the Access Control Module has enough resources to implement stronger authentication mechanisms (e.g. using Public Key Infrastructures) such as X.509 certificates [9].

Hijacking Possibilities

DNS Spoofing

The basic task of DNS (Domain Name System [58]) is to translate hostnames to IP addresses (e.g. it holds records such as www.example.com ⇒ 208.77.188.166). Each domain (e.g. example.com) has one or more DNS servers that publish information about it. Since there are billions of domains, each DNS server can not hold all mappings (due to scalability problems). To solve this problem, DNS servers query each other to find the requested information.

DNS *spoofing* is the art of making a DNS entry point to an IP address it is not supposed to point to. An adversary mounting a dns spoofing attack, is able to redirect Internet traffic to an IP address of his choice. This means the attacker can replace the ACM by redirecting traffic to his own fake ACM (as depicted in Figure 4.4). There are two ways to perform this attack: *DNS cache poisoning* and *DNS ID spoofing.*

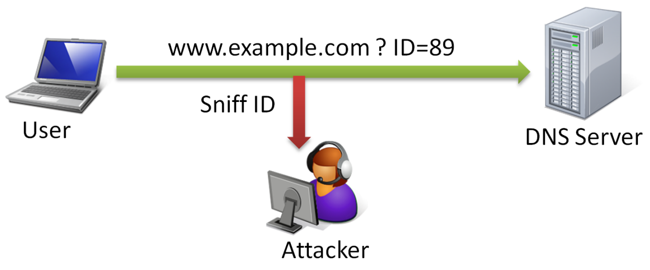
In **DNS cache poisoning** [75], the adversary needs his own DNS server. Most DNS servers support ”recursive” queries. This means one can send a request to any DNS server asking it to resolve any name-to-IP. The DNS server will then send queries to other DNS servers in order to discover the required information (e.g. the IP).

An adversary can predict what request the victim server will issue and spoof the response. The fact that DNS servers cache information locally for a certain amount of time is useful to the attacker. If he successfully spoofs a response, every legitimate user of the poisoned DNS server will be redirected to the *fake* IP.

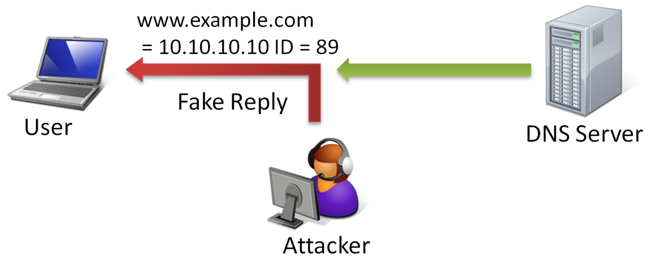
In **DNS ID spoofing**, the attacker sniffs (e.g. eavesdrop and store) or guess the requestor’s query ID and then replies with a fake answer. This attack is shown in Figure 4.5 and is also known as “impersonating” a DNS name server because the attacker replies on the real DNS server’s behalf. Further details can be found in [7].

DNS Spoofing - Countermeasure

There are several approaches to prevent DNS spoofing attacks. For the best security, we should not rely on DNS at all. If the IP of the ACM (Access Control Module) never changes by using a static IP for instance, then there is no need for DNS. This would completely eliminate the DNS threats for the ACM. If the assumption that the IP never changes is too strong, we can consider other solutions: using secure DNS (e.g. using TLS [28]), limiting the cache on the DNS server or restricting zone transfers [58].



(a) Sniff DNS Request ID



(b) Spoofed DNS Response

Figure 4.5: Example of DNS ID Spoofing

MAC Spoofing

Every network interface controller (NIC) has an unique MAC (Media Access Control) address. On a local area network, computers use MAC addresses to identify each other. MAC *spoofing* refers to altering the MAC address on a NIC. An adversary can perform MAC spoofing to take over the ACM's identity by changing his own MAC address to the one the ACM is using. There are many tools to change the MAC address of a NIC, (e.g. MAC Changer (for GNU/Linux) [68], SMAC (for Windows) [49]).

There can be routing disambiguities when there are two identical MAC addresses used on the same LAN. Therefore, once the attacker has changed his MAC address, he can mount a Denial-of-Service Attack (see in IP Spoofing for more information) on the ACM in order to make it completely unresponsive. Further technical details can be found in [21].

MAC Spoofing - Countermeasure

A quick way to detect if a MAC address is compromised is to run RARP [35] (Reverse Address Resolution Protocol) against it. RARP maps a MAC address to an IP Address. As only one MAC address should map to a single IP address, RARP should return one IP address for one network device. Thus, if multiple IP's are returned for one MAC address, one can investigate if an attacker is present or not. More advanced protection mechanisms include: special software like spygate firewalls, sticky ARP, MAC locking, ARP table-based MAC/IP filtering and New-IP ARP lookups. The mechanisms are further detailed in [21].

IP Spoofing

An IP (Internet Protocol [72]) address is a unique 32-bit address that devices on a computer network (using TCP/IP [82]) use to communicate and identify each other.

IP *spoofing* refers to altering the IP address of a malicious message (e.g. a packet) to make it appear as coming from a trusted source (e.g. the real ACM). An attacker may have several goals while mounting an IP spoofing attack; **Session hijacking**, or **Denial-of-Service** (for more information on other attacks such as non-blind spoofing, blind spoofing and Man in the Middle, the reader can refer to [87]).

Session hijacking can easily be achieved when the attacker is on the same subnet as the victim (as further explained in [87]). After sniffing sequence and acknowledgement numbers, an adversary can corrupt the datastream of an established connection and then re-open a valid connection from the attacker machine using the sniffed information.

In a **Denial-of-Service** (DoS) attack, the adversary floods the victim with as many packets as possible in a short time to make the server unresponsive. He can rely on *IP spoofing* to prolong the effectiveness of the attack. If the attacker changes the source IP all the time, tracing and stopping a DoS attack becomes more tedious (it is easier to detect an incoming flood from one IP address and to block it).

IP Spoofing - Countermeasure

To prevent an adversary from mounting an IP spoofing attack, there are several countermeasures (presented in [87]) that can be implemented: filtering at the router and using encryption and authentication.

Filtering at the router involves blocking certain source IP addresses from incoming and outgoing messages at the router. Implementing those so called *ingress* and *egress* filters [34] help to limit attacks performed through *IP spoofing.*

Using **encryption and authentication** reduces IP spoofing threats considerably. Since both features are included in IPv6 [25], the Access Control Module should use IPv6 instead of IPv4 to guarantee some level of protection against *IP spoofing* attacks.

Physical Capture

The machine on which the Access Control Module resides can be physically captured. By that we mean that an attacker can get access to the computer, e.g. by breaking into the building in which the target machine is hosted. Once an intruder has successfully gained access to the machine hosting the ACM, he can perfectly impersonate it since he has complete control over the machine.

Physical Capture - Countermeasure

To protect the ACM of such an attack, we can imagine two countermeasures. First, the access to the computer on which the ACM is running should be restricted (e.g. using RFID on the entrance door) and well protected physically (separating it from other IT infrastructure for example).

Second, there should be mechanisms in the server room (such as video surveillance and motion detectors) to monitor any activity and raise alerts in case of an intrusion.

Conclusion

Being aware of the attacks mentioned above and correctly implementing their countermeasures provides the ACM with sufficient security to prevent the most common Hijacking attempts.

4.3.2 Inside Attack

A legitimate user, i.e. a nurse, might switch to a malicious behavior at some point in time and might escalate her rights to gain access to data she is not authorized to (i.e. data from all patients, sensitive data only available to doctors, etc).

There are several ways for an attacker to try and escalate his rights:

Sibling Attack. A User can try to escalate his access rights by trying to derive a key on the same level in the tree where the attacker is (e.g. a sibling). For example, in Figure 4.6, the attacker could legitimately be on authorization level AC_3 and trying to derive the key for authorization level AC_4.

Parent Attack. A user can try to escalate his rights by trying to derive the key of his parent node in the classification tree. In Figure 4.6, the attacker could be on authorization level AC_5 and trying to derive the key for authorization level AC_2.

Coalition Attack. A coalition of two or more users from different authorization levels (i.e. AC_3 and AC_2) could try to derive a key for a different level (i.e. AC_4 or AC_0) by combining their cryptographic material.

The Sibling, Parent and Coalition Attack are discussed in the following sections.

Sibling Attack. The scheme proposes to makes use of the secure message authentication code HMAC [15] which does not allow existential forgery under chosen-plaintext attacks.
HMAC is defined as follows:

$$HMAC_K(m) = h((K \oplus opad)||h(K \oplus ipad)||m))$$

where h is a cryptographic hash function (e.g. SHA-256 [64], Ripemd-160 [29]), opad = $0x5C$ repeated 64 times and ipad = $0x36$ repeated 64 times, K is the secret Key and m is the message to hash. The underlying hash function of the HMAC is supposed to be collision resistant.

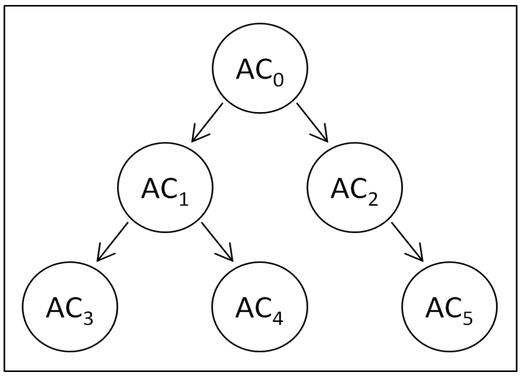


Figure 4.6: Authorization Class Hierarchy

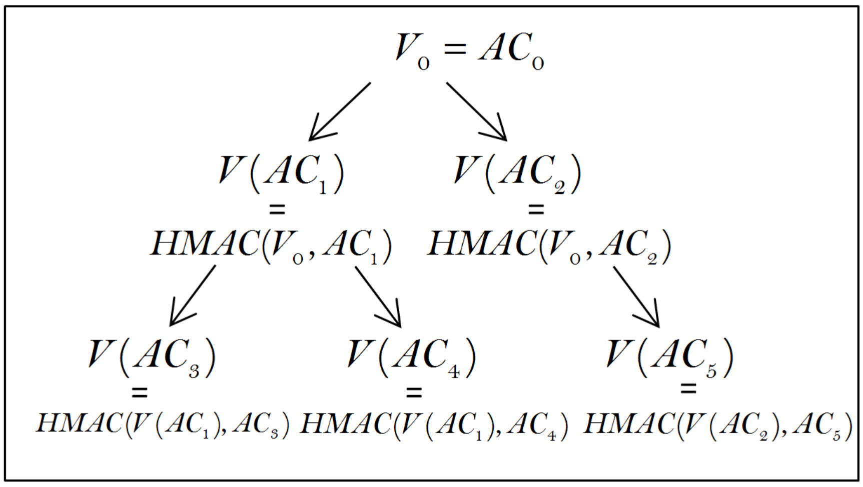

$V_0 = AC_0$
$V(AC_1) = HMAC(V_0, AC_1)$
$V(AC_2) = HMAC(V_0, AC_2)$
$V(AC_3) = HMAC(V(AC_1), AC_3)$
$V(AC_4) = HMAC(V(AC_1), AC_4)$
$V(AC_5) = HMAC(V(AC_2), AC_5)$

Figure 4.7: Key Generation Tree

Existential Forgery under Chosen-Plaintext Attack

We suppose the attacker is able to gather some number of example pairs of messages and their valid HMAC's. He is also allowed to specify his own message and receive the corresponding valid HMAC from an oracle [16] (black box) implementing the scheme. Even then it is not possible, given many pairs $(m_i, HMAC_K(m_i))$, $(i \in \{1, ..., n\})$ for an attacker to find $HMAC_K(m_j)$, $j \notin \{1, ..., n\}$ where K = Secret Key.

In Figure 4.7, we thus see that it is not feasible for an attacker that knows the hash value of node AC_3 ($V(AC_3)$) to find the hash value $V(AC_4)$. From the property outlined above, it is also not possible for a coalition of users sharing their $V(AC_i)$'s to derive a hash value of one of the siblings belonging to the same authorization level, for instance for AC_5.

Parent Attack. The one-way hash function (e.g. SHA-256 [64], RipeMD-160 [29] (we use RipeMD-160 in the implementation)) used in the HMAC has to satisfy *collision*, *pre-image* and *second pre-image* resistance [20] in order to be considered reasonably secure.

Collision Resistance Collision Resistance means that it is hard to find two inputs m and m' that hash to the same output, ie $h(m) = h(m')$. A pair of inputs (m, m') that produce the same hash is called a collision pair.

Pre-Image Resistance Pre-Image Resistance means that given a hash value $hash$ it is hard to find an input m such that $h(m) = hash$.

Sesond Pre-Image Resistance Second Pre-Image Resistance means that given an input m, it is hard to find another input m' $(m' \neq m)$ such that $h(m') = h(m)$.

Collision resistance is the strongest of those three properties in the sense that it implies both the *pre-image* and *second pre-image* resistance properties.

When h (the hash function) satisfies those properties, it becomes unfeasible for node AC_i to derive the value of his parent $V(P(AC_i))$ from its own value $V(AC_i)$ (where $P(AC_i) = \{Parent\ Node\ of\ AC_i\}$). This means an attacker cannot escalate his rights in the tree (as in Figure 4.7).

Coalition Attack. A coalition attack is an attack where two or more users collaborate in order to escalate their rights. However, this attack can be considered as a sequence of sibling and parent attacks. Combining the Sibling and the Parent Attack it is therefore not possible for a coalition of users to escalate their rights, even though they share their cryptographic materials. As described in Section 4.2, a user (legitimate or adversary) can thus only derive a new key for one of his children, but not for his parent nor for his siblings.

4.4 Integrity of Wireless Sensor Data

So far, we introduced an access control scheme that offers a hierarchical access control (and inherently end-to-end confidentiality) of sensor data. However, another identified important security requirement is data integrity. This security requirement is not filled with the scheme. In this section we introduce the concept of Integrity and propose mechanisms to reach the goal of offering Integrity of Sensor Data.

In Figure 4.8 we add the Access Control Module, which deals with authenticating users on one side and distributing cryptographic material to sensor nodes on the other side. The Attacker Model we defined in Section 4.1.4 is still applicable for this system.

Integrity is defined as a protection against unauthorized modification or destruction of information, in this case the sensitive value of the wireless sensor data [67]. As the sensor data is only transported between the Gateway and the User (Figure 4.8), without involving the ACM, we only analyze how the integrity of the sensor data evolves between the WSN and the end user.

Inside the WSN

Inside the WSN, the 802.15.4 [80] protocol assures integrity if the AES-CBC-MAC

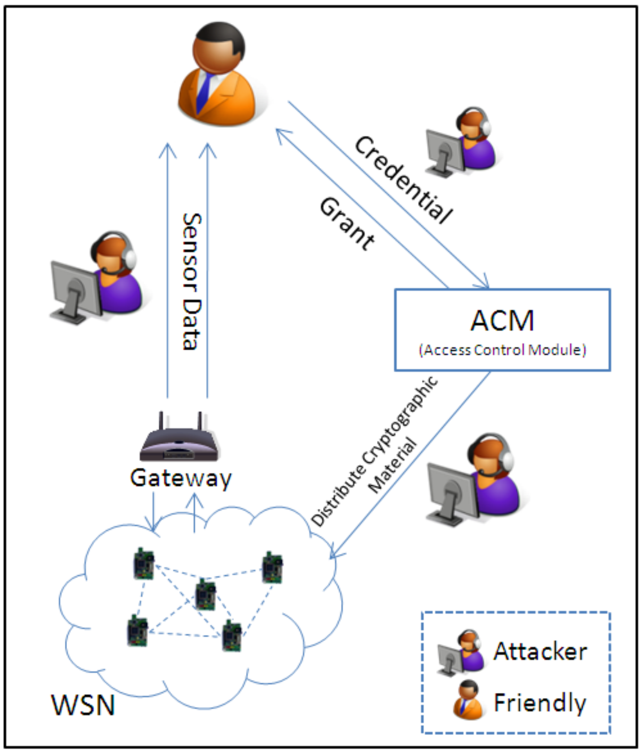


Figure 4.8: System with Access Control Module

mode (see Figure 4.9) is selected as security suite.

Cipher Block Chaining Message Authentication Code, abbreviated CBC-MAC, is a technique for constructing a message authentication code (MAC) from a block cipher (e.g. AES). A MAC is a short piece of information (e.g. 128 or 192 bits) used to authenticate a message. The message m is divided into n blocks of the same size and each block is encrypted using a block cipher (like AES) with key k. Figure 4.9 shows the CBC-MAC computation of a message m split into n blocks $m_1|\ldots|m_n$ using a Block Cipher E (e.g. AES).

The interdependance created by the CBC operation mode (Figure 4.9) ensures that any change of bits in the plaintext (even just one bit) will cause the final ciphertext to be changed in an unpredictable way. This assertion holds only if the key k for the block cipher (e.g. AES) is unknown to the attacker. The MAC value protects both a message's data integrity as well as its authenticity.

From the WSN to the User

When the data is transmitted over the Internet (e.g. Wireless or Ethernet) the AES-CBC-MAC Security Suite from the 802.15.4 protocol is not usable anymore.

The basic protocols such as TCP and UDP do not offer data integrity, so additional protocols need to be considered. Our research revealed IPSec [50] or TLS [28] as possible protocols (the successor of Secure Sockets Layer (SSL)) to guarantee in-

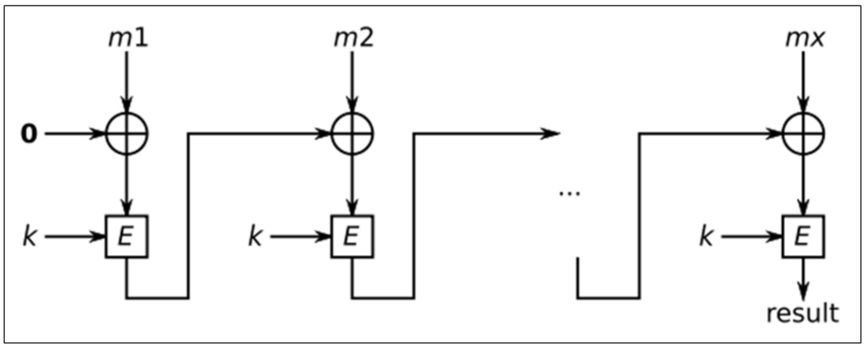


Figure 4.9: AES-CBC-MAC Construction

tegrity. IPSec operates at the network layer (Layer 3 in the OSI Model [94]) while TLS operates on the transport layer and above (Layer4-7 in the OSI Model). Both protocols authenticate and encrypt data packets between a sender and a receiver and as such offer Data Integrity. It should be noted that both of the protocols add a considerable overhead (in bits) to the data packets and as such might not be usable. This is especially true for the packets travelling from the WSN sink to the Gateway.

If the usage of such a protocol is not possible due to technical constraints we can append a Message Authentication Code (MAC) to the sensor data to maintain integrity.

4.5 Conclusion

In Chapter 3, we identified a clear need for end-to-end confidentiality of sensor data when dealing with the integration of WSNs into business applications. To fullfill this requirement, we introduced an efficient security scheme that uses symetric-key encryption to guarantee data confidentiality. Furthermore, the scheme allows hierarchical access control. This is useful given that sensor data and roles are mapped to a hierarchy. Hierarchical access control thus becomes a necessity.

We then performed a security analysis of the scheme [83] since the authors barely discussed it in their paper. By analyzing both outside and inside attacks, the scheme demonstrated satisfactory results given the Attacker Model we defined in Section 4.1.4. We found a few vulnerabilities at the Access Control Module level (e.g. DNS / MAC / IP spoofing attacks). In Section 4.3.1 we propose several countermeasures for each attack to reduce those threats.

In Section 4.1.3 we identified data integrity as one of the major security requirements which unfortunately was out of scope in the scheme. Since this is a major requirement, we showed how existing protocols can be leveraged to guarantee integrity of sensor data in Section 4.4.

As the security scheme relies on OTP for encryption and decryption, we have to check whether the generated keys are pseudo-random or not. In order to be secure in the random oracle model, the hash function needs to generate a **pseudo-random** key stream. In the paper [83], the authors assume that the generated keys are pseudo-random. But to the best of our knowledge, hash functions have not been used to generate pseudo-random key streams so far and as such we were unable to find any indications on the randomness of such keys. For that reason, we decided to perform a randomness evaluation on the generated key streams. We used the NIST Statistical Test Suite [74] to perform this evaluation and the details can be found in Chapter 5.

CHAPTER 5

RANDOMNESS EVALUATION

"Statistics are no
substitute for judgment."
Henry Clay

One of the major requirements of the scheme [83] we analyzed in Chapter 4 is that the keys need to be pseudo-random [57, Chapter 5] so that the correlation between them is so small as to discourage any statistical attacks on the ciphertext. The randomness is dependent on both the choice of the hash function and the way it is used in the construction of the keys. In our current implementation, we are using RIPEMD-160 [29] as hash function for which there are no known attacks at the time of this writing. In this chapter we will evaluate the randomness of the keys that are generated by the scheme [83] using the NIST SP 800-22b test battery.

The NIST (National Institute of Standards and Technology) Test Suite SP 800-22b (detailed in [74]) is a statistical package consisting of 15 tests. These tests evaluate the randomness of a given sequence of keys.

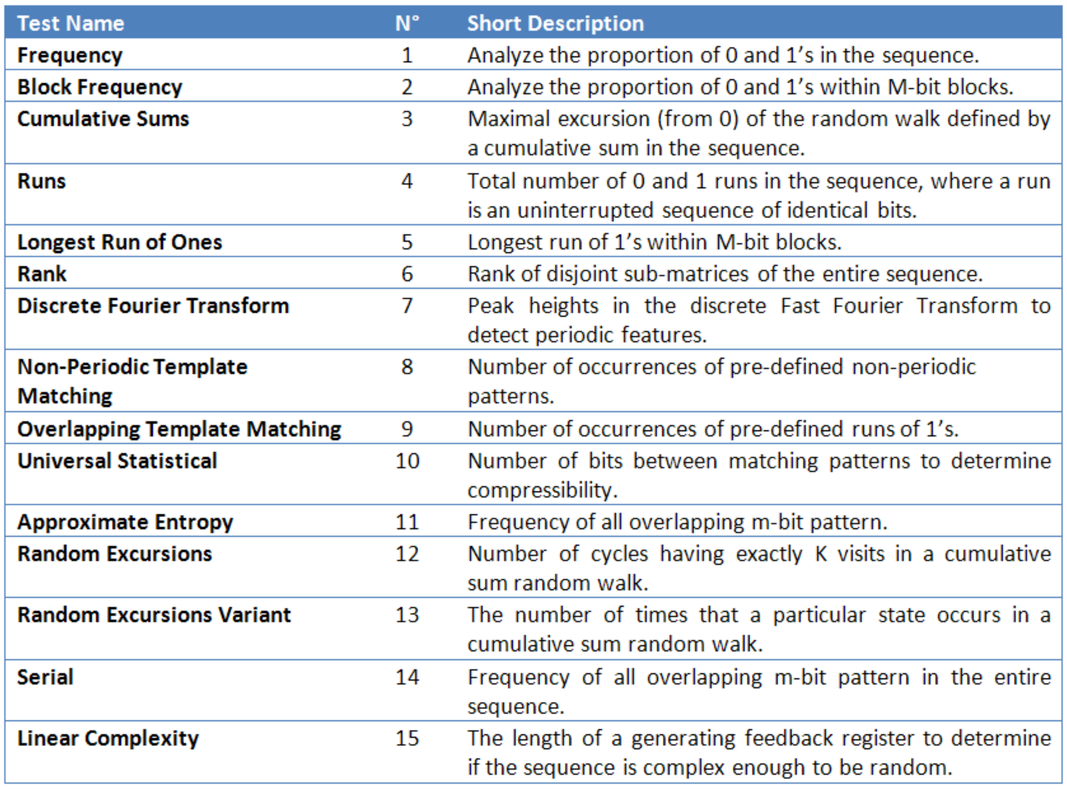
The NIST organization provides an ANSI C implementation of that test battery which can be freely downloaded from [11]. DIEHARD [54] is another free of charge package that offers a variety of statistical tests. However, many of the tests are based on the NIST 800-22b package and so we choose NIST to perform the testing.

5.1 Parameter Choices

For each test, the NIST test suite extracts a probability, called the *p-value*. The *p-value* is a summary of the strength of the analyzed sequence against a perfectly random one. A *p-value* of 0 suggests that the sequence is completely non-random.

We denote α as the confidence level which typically ranges from 0.001 to 0.01 [74]. Let Seq be the sequence of bits, then for each test we have

$$Randomness(Seq) = \begin{cases} True\ \ , if\, p - value > \alpha \\ False\ , if\, p - value < \alpha \end{cases}$$



Test Name	N°	Short Description
Frequency	1	Analyze the proportion of 0 and 1's in the sequence.
Block Frequency	2	Analyze the proportion of 0 and 1's within M-bit blocks.
Cumulative Sums	3	Maximal excursion (from 0) of the random walk defined by a cumulative sum in the sequence.
Runs	4	Total number of 0 and 1 runs in the sequence, where a run is an uninterrupted sequence of identical bits.
Longest Run of Ones	5	Longest run of 1's within M-bit blocks.
Rank	6	Rank of disjoint sub-matrices of the entire sequence.
Discrete Fourier Transform	7	Peak heights in the discrete Fast Fourier Transform to detect periodic features.
Non-Periodic Template Matching	8	Number of occurrences of pre-defined non-periodic patterns.
Overlapping Template Matching	9	Number of occurrences of pre-defined runs of 1's.
Universal Statistical	10	Number of bits between matching patterns to determine compressibility.
Approximate Entropy	11	Frequency of all overlapping m-bit pattern.
Random Excursions	12	Number of cycles having exactly K visits in a cumulative sum random walk.
Random Excursions Variant	13	The number of times that a particular state occurs in a cumulative sum random walk.
Serial	14	Frequency of all overlapping m-bit pattern in the entire sequence.
Linear Complexity	15	The length of a generating feedback register to determine if the sequence is complex enough to be random.

Figure 5.1: List of Tests and their Description

NIST recommends running the tests on 1'000 sequences of keys with each sequence having at least 10^6 bits in order to determine if a key stream is random or not.

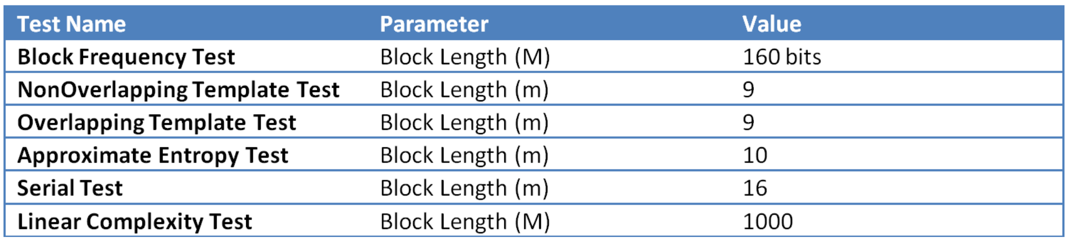
We generated 10^7 keys of 160 bits each using our implementation of the scheme [83]. The resulting 1.6 Gib file was then used as input for the 15 tests. Each test was run on 1'000 sequences of 10^5 keys.

The significance level α was set to 0.01, which is the suggested level when dealing with cryptography [74, 4.3.f]. This means that if a *p-value* > 0.01 the sequence is accepted as random with a confidence of 99%. Similarly, if a *p-value* < 0.01, the sequence is non-random with confidence of 99%.

Figure 5.1 shows a list of the 15 tests of the NIST Suite with a short description of each. For more details about the tests, the reader can refer to [74]. In Figure 5.2, we show the input parameters we used to perform the battery of tests. The values for each test satisfy the conditions set in the NIST document (see [74, Chapter 2]). For the tests not mentioned here, we used the default parameters suggested by NIST.

5.2 Results

There are two suggested approaches to interpret the results of the NIST Test Suite. The first one deals with the examination of the proportion of sequences that pass



Test Name	Parameter	Value
Block Frequency Test	Block Length (M)	160 bits
NonOverlapping Template Test	Block Length (m)	9
Overlapping Template Test	Block Length (m)	9
Approximate Entropy Test	Block Length (m)	10
Serial Test	Block Length (m)	16
Linear Complexity Test	Block Length (M)	1000

Figure 5.2: Parameter Choices for the Test Suite

the test while the second one focuses on the uniformity of the *p-values*. The two approaches are discussed in the sections below.

Approach 1: Examination of Proportion of Passing Sequences

The final analysis report generated by the suite contains a value called proportion for each test. The proportion is the number of sequences that passed (e.g. with p-value $> 0.01 = \alpha$) divided by the total number of sequences tested. In other words, the proportion is the percentage of passed tests.

NIST specifies a range of acceptable proportions determined by using the confidence level defined as $\hat{p} \pm 3\sqrt{\frac{\hat{p}(1-\hat{p})}{m}}$, where $\hat{p} = 1 - \alpha$ and m is the sample size (e.g. 1'000).

In our case, we used m = 1'000 sequences and each sequence had 1.6 Mib bits. Using the formula for the confidence level above, our range of acceptable proportions is from 0.9805 to 0.9994. Figure 5.3 shows the proportion for each test. Since the proportion for each test lies within the range we computed, we can accept the sequence as a random bit sequence.

Approach 2: Examination of the Uniformity of p-values

The second approach presented by NIST is the analysis of the uniformity of the *p-values*. If the *p-values* form a uniform distribution, then we accept the sequence as random.

Chi-Square Test

Uniformity can be examined by performing a χ^2 (Chi-Square) test on the p-values, with

$$\chi^2 = \sum_{i=1}^{10} \frac{(C_i - \frac{m}{10})^2}{\frac{m}{10}}$$

where C_i is the number of p-values in the sub-interval $[\frac{i-1}{10}, \frac{i}{10})$, i = 1...10 and m is the sample size (the number of sequences tested, e.g. 1'000). Finally, a new p_Γ-value is computed from the original p-value as

$$p_\Gamma - value = igamc(\frac{\chi^2}{2}, \frac{9}{2})$$

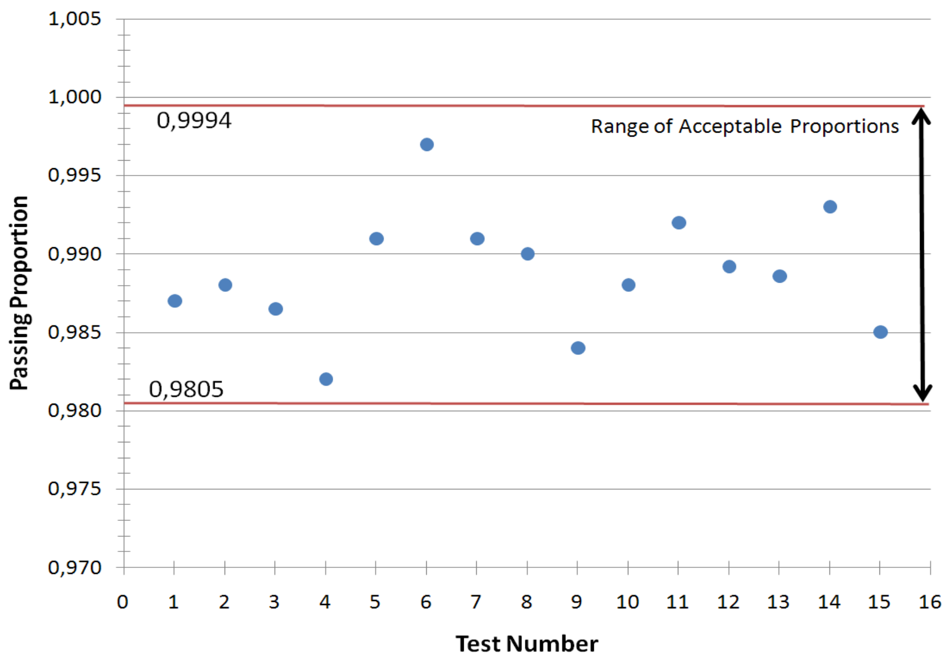


Figure 5.3: Passing Proportion of the 15 NIST Tests

Test N°	1	2	3	4	5	6	7	8
$p_\Gamma - value$	0.371	0.417	0.717	0.798	0.684	0.371	Skipped	0.493
Conclusion	**Pass**	**Pass**	**Pass**	**Pass**	**Pass**	**Pass**	**NA**	**Pass**
Test N°	9	10	11	12	13	14	15	
$p_\Gamma - value$	0.469	0.987	0.209	0.642	0.640	0.332	Error	
Conclusion	**Pass**	**Pass**	**Pass**	**Pass**	**Pass**	**Pass**	**NA**	

Figure 5.4: Uniformity Distribution of p-values

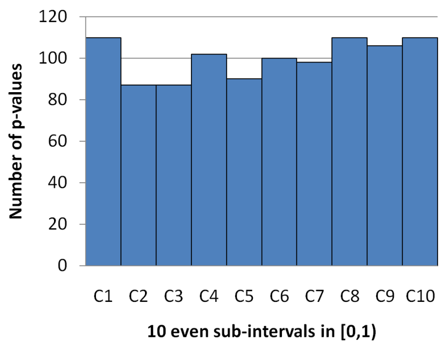
where igamc is the Incomplete Gamma Function as defined in [74, Section 5.3.3]. Finally, the

$$p - values = \begin{cases} Uniform, if\ p_\Gamma > 0.0001 \\ Non - Uniform, if\ p_\Gamma < 0.0001 \end{cases}$$

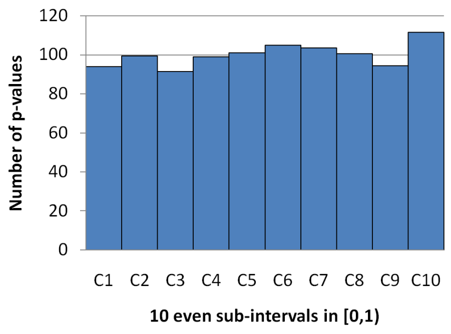
Figure 5.4 shows the p_Γ-values for the tests we conducted. From the table we see that all test passed the uniformity condition and thus the sequence is random.

We omitted the Fast Fourier Transform test from the examination of uniformity as the results were not usable. Furthermore, in [51], the authors state that the results of the FFT test are degrading when the number of sequences increases. In particular, if the number of sequences is bigger than 10'000, then any Pseudo-Random Number Generator fails the uniformity test. For that reason, we decided not to include this test in the uniformity evaluation.

The evaluation of uniformity for the Linear Complexity test raises an exception when we run it. At this point we are still investigating the cause of this as we are



(a) Cumulative Sum Test



(b) Serial Test

Figure 5.5: Histograms for the Cumulative Sum 5.5(a) and Serial Test 5.5(b)

not aware of the reasons behind the failing of this test. In order not to bias our results by using erroneous values, we decided to remove the Linear Complexity test from the Uniformity Evaluation.

Visual Analysis

In addition to the chi-square test we also can visually examine the distribution of the *p-values* by plotting them into a histogram. The horizontal axis (the probabilities) ranging from 0 to 1 is divided into 10 even sub-intervals. The number of *p*-values within each sub-interval is displayed on the vertical axis. As an example, Figure 5.5 shows the histograms of *p-values* for the Cumulative Sums and the Serial Test. We have found similar results for the other tests. All histograms are depicted in Appendix A. We see that the *p-values* are uniformly distributed for every test we performed and thus the sequence we provided can be considered as random.

5.3 Conclusion

In this chapter we evaluated the randomness of the key streams generated by the scheme in [83]. We conducted this evaluation using the NIST SP 800-22b Test Suite [74] that provides a battery of statistical tests to evaluate the randomness of binary sequences. We ran each of the 15 tests 1'000 times with sequences of 1.6 Mib each. We then interpreted the results using two different methods suggested by NIST, (i) examination of the proportion of passing sequences and (ii) examination of the uniformity of the *p-values*.

The NIST SP 800-22b Test Suite states that if a sequence of 1 Gib passes all of the 15 tests, then that sequence can be considered as random [74]. Based on the results of the proportion analysis and the uniformity distributions outlined in Section 5.2, we conclude that the input file (consisting of 1.6 billion bits) we provided to the NIST Test Suite is pseudo-random.

These results seem to confirm that using hash functions lead to pseudo-random key generators. Furthermore, we showed that the way keys are computed (e.g. by incrementing a counter for each key) seems to preserve pseudo-randomness and could be compared to other pseudo-random number generators (PNRGs) [57] such as Blum Blum Shub [17] or Fortuna [33].

Besides generating random key streams, the hash function needs to verify the collision resistance property we outlined in section 4.3.2. In Cryptography, a brute force collision search has a time and memory complexity of $\sqrt{2^n}$ where n is the hash size [57]. This means that for a hash of 160 bits, the complexity is $\sqrt{2^{160}} = 2^{80} = 1.20892582 \times 10^{24}$. Thus, if we suppose that the computation of a hash takes 1 [ns], then a brute force attack requires 38 million years to be successful. This means that if no statistical attacks are known, the hash function can be considered collision resistant.

In the next chapter, we proceed with the secure integration of WSNs into business applications by implementing the scheme. First, we provide a proof-of-concept completely in Java to prove the feasibility of the scheme. Second, we describe how a hardware prototype using real sensor nodes was developed with Cisco Systems France. Last but not least, we present some empirical performance evaluations for the scheme implemented on the hardware nodes.

CHAPTER 6

IMPLEMENTATION

"First, solve the problem.
Then, write the code."
John Johnson

In order to prove the feasibility of the access control scheme, we implemented two prototypes: (i) the first one is a simulation of WSNs and is fully implemented in Java, (ii) the second one integrates real sensors implementing the key generation and encryption algorithm in collaboration with Cisco. In this Chapter, we detail both of these proof-of-concepts.

6.1 Java Simulation

Simulation is a cheap, quick and reliable way to test a system before deploying it onto real hardware. The advantage of providing such a proof-of-concept is that it can be run completely on one machine and its only functional requirement is the Java Virtual Machine (JVM). The choice of Ptolemy II [4] as simulation environment for the WSN enables us to host the WSN behavior, the Middleware and the Business Applications on the same machine even if all components are independent. In Figure 6.1 we show the global architecture of our simulation environment.

There are four major components in our system: **(i)** the Ptolemy II framework which simulates the WSN, **(ii)** the Enterprise Integration Component (a mediation layer between the WSN and Applications), the Access Control Module (in charge of the grant distribution) and the Backend Applications (e.g. web-based applications consuming data). Each of these components is discussed in the sections below.

6.1.1 Ptolemy II

Ptolemy II is a Java based Simulation Framework to simulate Wireless Sensor Networks (see Figure 6.2). Further details about Ptolemy II can be found in [63, Chapter 4]. In Figure 6.2, we show what a WSN model can look like with Ptolemy II. The possibility to make the model look like a real use case technically speaking (e.g.

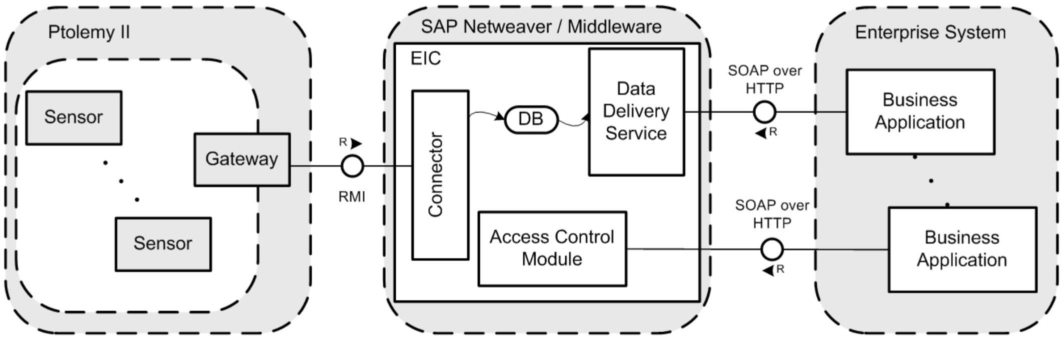


Figure 6.1: Java Simulation Architecture

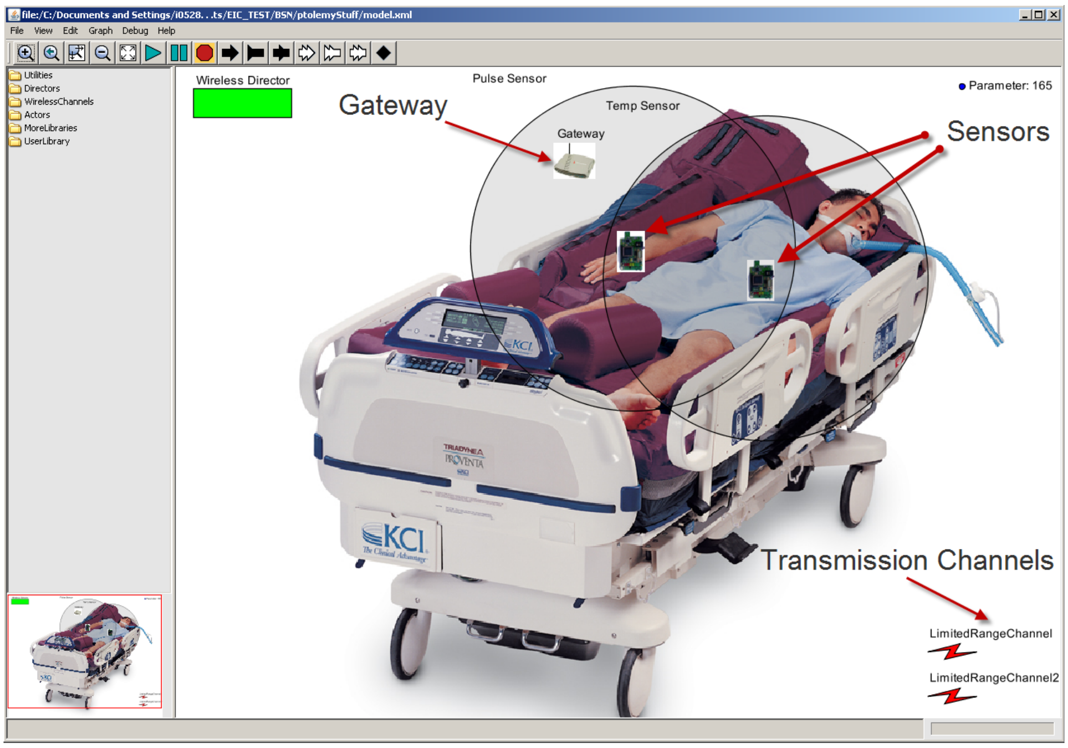


Figure 6.2: Ptolemy II Customized Model View

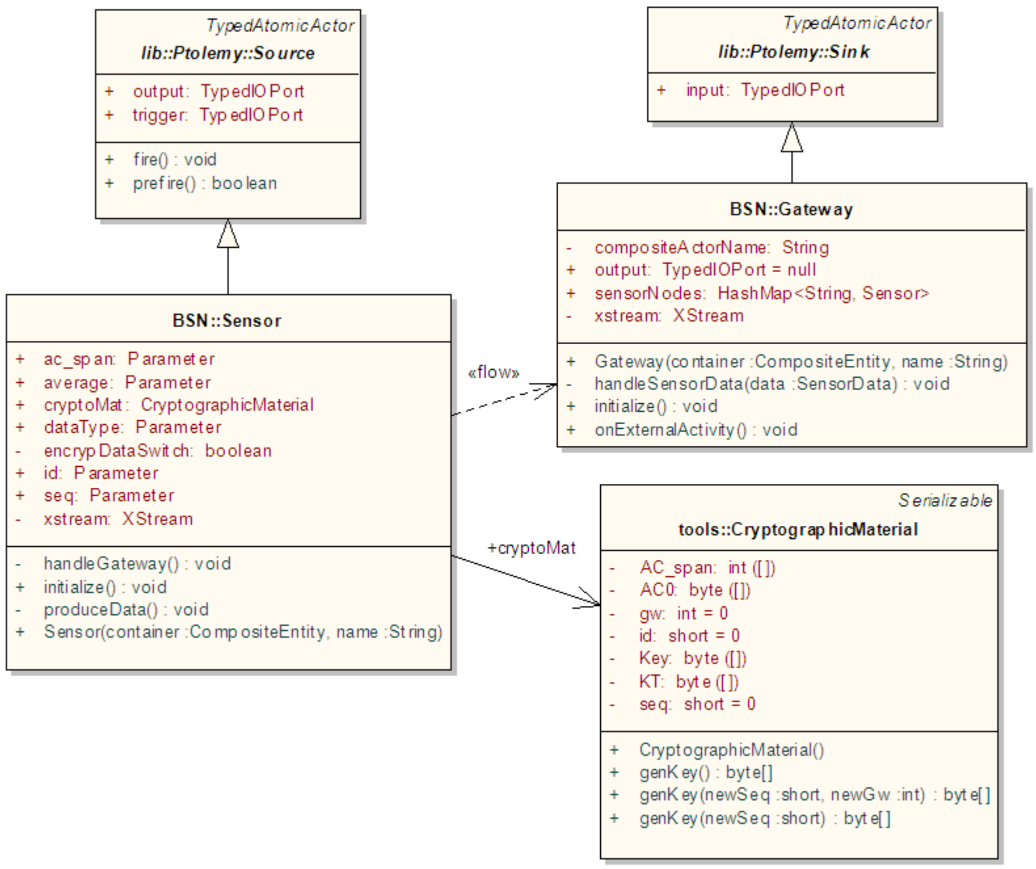


Figure 6.3: Ptolemy Class Diagram

patient monitoring) is an interesting feature of Ptolemy II. Furthermore, each component in Ptolemy II (e.g. Sensors and Gateways) is represented by a Java Object.

There are two main components: *Sensor* and *Gateway*. A *Sensor* is derived from a *Ptolemy Source Actor* which is a generic agent for any wireless source. On the other hand, the *Gateway* is derived from a *Ptolemy Sink Actor* which is a generic component for wireless sinks. The Class diagram of the Ptolemy II part is depicted in Figure 6.3.

Sensor

The *Sensor* class is derived from a *Ptolemy Source Actor* and deals with the production, encryption and transmission to the *Gateway* of encrypted sensor data. Additionally, each *Sensor* has its own *CryptographicMaterial* object which holds all the cryptographic material attached to that node (e.g. KT, AC0, GW). The main sequence of actions performed by the *Sensor* Class is summarized in Figure 6.4.

The encryption of the Sensor Data is done according to the scheme we discussed in Chapter 4. The *CryptoEngine* class implements the encryption algorithm and computes the encryption keys based on the parameters the *Sensor* provides. We use

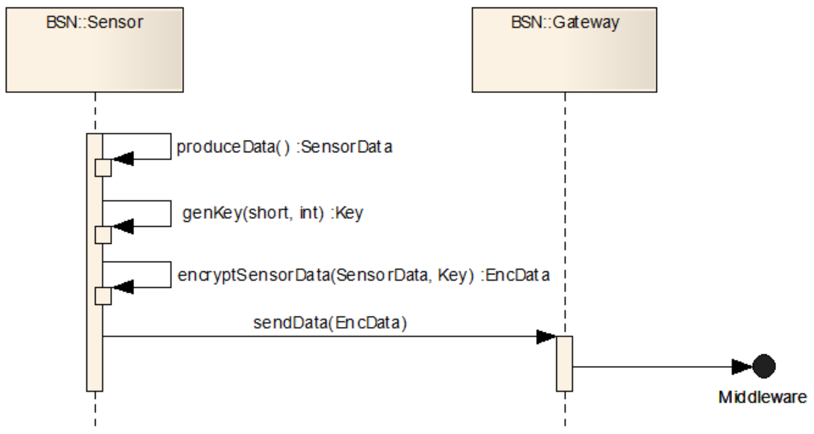


Figure 6.4: Ptolemy Sensor Actor Sequence Diagram

RIPEMD-160 [29] as cryptographic hash function for the key generation. RIPEMD-160 has no known attacks at the time of this writing and is collision resistant.
The parameters used for the encryption are:

- KT: The Secret Value K_T
- AC0: The root of the Authorization Class Tree
- GW: The Grant Window
- AC_span: The Authorization Class Spanning Tree (e.g. 1,2,1)
- ID: The ID number of the Sensor
- Seq: The Sequence Number of the Sensed Data (incremented for each sensor data)

Currently, the Secret KT and AC0 are pre-configured inside of the *Sensor* Class at design time because the key distribution is currently not implemented.

Figure 6.5 shows the sequence diagram of the encryption part. When the *Sensor* Class is instantiated, it creates a new *CryptographicMaterial* object and assigns the parameters *GW*, *KT*, *AC_span*, *ID* and *AC0* to it. Afterwards, each time a Sensor Data is produced, the sequence number *Seq* is incremented and the associated *CryptographicMaterial* object is updated.

The *Sensor* object then asks the *CryptoEngine* to create a key based on the parameters inside the *CryptographicMaterial* object. Finally, the Sensor Data is encrpyted using the generated key and is sent to the *Gateway*. The packet that is sent to the *Gateway* contains the following elements required for the decryption key computation:

$$\{SensorData_{Encrypted}, GW, ID, Seq, AC_{span}\}$$

Gateway

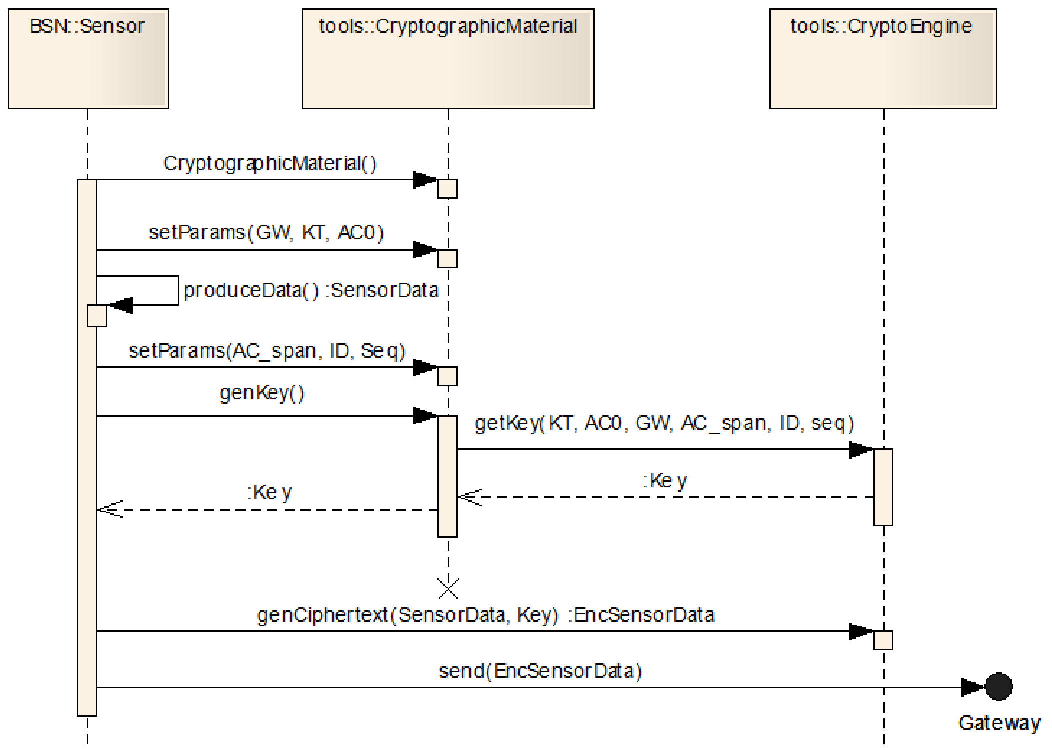

BSN::Sensor
tools::CryptographicMaterial
tools::CryptoEngine
CryptographicMaterial()
setParams(GW, KT, AC0)
produceData() :SensorData
setParams(AC_span, ID, Seq)
genKey()
getKey(KT, AC0, GW, AC_span, ID, seq)
:Key
:Key
genCiphertext(SensorData, Key) :EncSensorData
send(EncSensorData)
Gateway

Figure 6.5: Ptolemy Sensor Data Encryption Sequence Diagram

The *Gateway* class is responsible for the delivery of acquired sensor data from the WSN to the middleware (Enterprise Integration Component). The structure of the *Gateway* is depicted in Figure 6.3. It is a derived from a *Ptolemy Sink Actor* which is a generic *Wireless Sink*. Its main purpose is to keep a list of all available nodes and to forward incoming data to the EIC. The communication between the *Gateway* and the EIC is done via Java RMI (Remote Method Invocation) [5].

6.1.2 Middleware (EIC)

The Enterprise Integration Component (EIC) is a high level middleware to support seamless integration of WSNs into Business Applications. It is designed on a SOA (Serivce Oriented Architecture) based architecture. The EIC is running on the SAP Netweaver Application Server[1] which is “the common technology platform for SAP business applications, providing the foundation for enterprise SOA amongst others”[2]. For a quick introduction about the SOA paradigm the reader can refer to [69] and further in-depth information about Web Services and SOA can be found in [14].

We consider the EIC as a black box. It offers several Services out of which two are of interest to us: the *Connector* Service and the *Data Delivery* Service.

The *Connector* is mapping incoming packets into a format the EIC can use. The packet format is different for each type of source, e.g. Ptolemy II and the Cisco

[1]See `http://www.sap.com/platform/netweaver/`
[2]See `https://www.sdn.sap.com/irj/sdn/nw-mainreleases`

Controller do not use the same data formatting, and consequently a new *Connector* is needed for each type of source.

On the other hand, the *Data Delivery* Service is in charge of delivering sensor data to Applications. The *Data Delivery* Service offers both the *Pull* and the *Push* Technology. Pulling Data means that the Application is the request originator and *pulls* the data from the server. In contrast, the Push Technology, also called publish/subscribe paradigm refers to Applications that subscribe to a type of information (e.g. Temperature). When new data is available, the server *pushes* the information to the requestor. A general overview of the EIC can be found in 2.2.1 and its complete architecture is detailed in [42].

6.1.3 Access Control Module

The Access Control Module (or **A**ccess **C**ontrol **P**oint) is in charge of authentication of users and the delivery Grants. The *ACP* Class Diagram is depicted in Figure 6.7.

ACP as a Web Service

The ACP is implemented as a Web Service and exposes a method called *getGrant()*. This method takes a *Credential* (e.g. a Role, a password or a X.509 Certificate [9]) as input, evaluates it and if the credential is valid, the ACP returns a *Grant* to the User. The *Grant* that is returned to the User contains the cryptographic material necessary to decrypt Sensor Data.

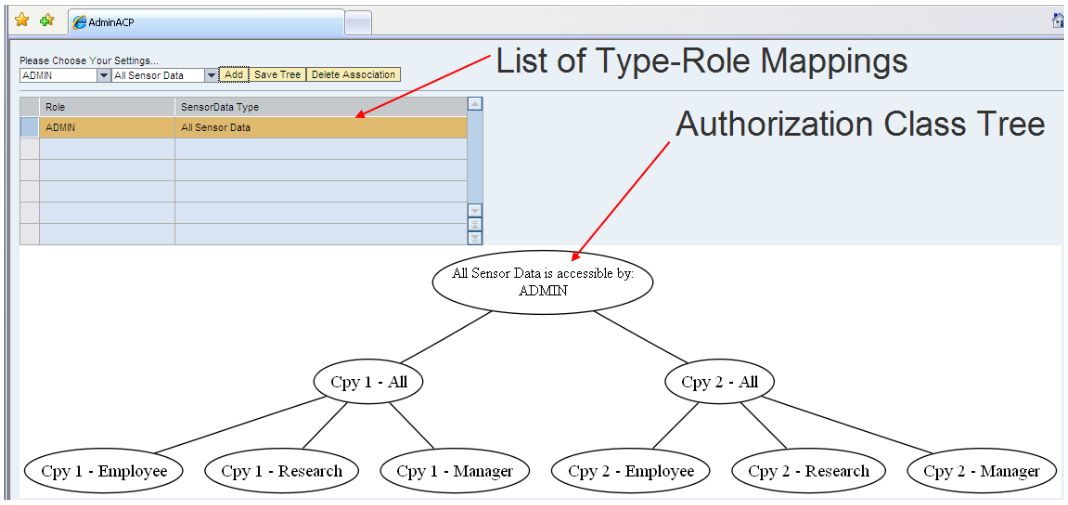
In Figure 6.8, we see a message flow of a *Grant* delivery to an Application. First, a User (e.g. a Business Application) invokes the Web Service via the *getGrant()* method and provides its credentials. Then, the ACP reads the Authorization Class Tree and evaluates the provided credential. If the authentication is successful, the ACP creates a *Grant* and sends it back to the User. The User now has all the cryptographic material required to generate decryption keys.

Configuration of the AC Tree

As mentioned in the previous section, the ACP "reads the Authorization Class Tree" to evaluate the requestor's credentials. In our current implementation, the credentials are roles (e.g. TemperatureManager). They re represented by a simple character chain. In a future version of this implementation, the credential could be X.509 certificates [9].

We developed a Configuration Tool that enables an Administrator to assign *Roles* to *Sensor Data Types*. This configuration step is the initial step in the simulation that defines which *Data Types* are accessible by which *Roles*. The configuration Interface is depicted in Figure 6.6. The main features are a list of existing *Data Type - Role* mappings and a visual representation of the tree. The Administrator can *Add* and *Delete* associations with a single click. The tree view of the mappings considerably eases the configuration process as it provides an appealing overview.

When the configuration is completed, the tree is saved in an XML structure on the file system. The *ACP Service* reads the saved tree on every *Grant* request. This means that changes done to the Authorization Class Tree are taken into account by the ACP without any latency.


Please Choose Your Settings...
ADMIN | All Sensor Data

Role	SensorData Type
ADMIN	All Sensor Data

Figure 6.6: Authorization Class Tree Configurator

6.1.4 Web Dynpro Businness Application

Web Dynpro is SAP's standard UI technology for developing web-based applications. Web Dynpro is based on the flexible Model-View-Controller [52] architecture that helps to implement a clear separation of user interfaces from backend services. Further information about the Web Dynpro technology can be found in [8].

In the scope of this project we developed a web-based Web Dynpro Application that consumes sensor data. We urge the reader to take a look at Appendix D to get an idea of what this business application looks like and the features it offers.

Figure 6.9 provides the series of actions taken when a User requests Sensor Data via the Web Dynpro interface. At first, the User provides his credentials to the Application. Then the Application invokes the *getGrant()* method from the ACP *Service*. After the Application acquires a proper *Grant*, it queries the EIC for the sensor data. Using the *Grant*, the *GW*, the *ID* and the *Seq* number, the Application computes the decryption key and decrypts the data. The User is now able to see the Data he requested.

By default, a User is able to retrieve all data, but can decrypt only a subset based on his credentials. If the credentials do not allow decryption of the values, $*******$'s are displayed. If, on the contrary, the User is authorized, then the data is decrypted and displayed. This behavior is shown in Figure D.14.

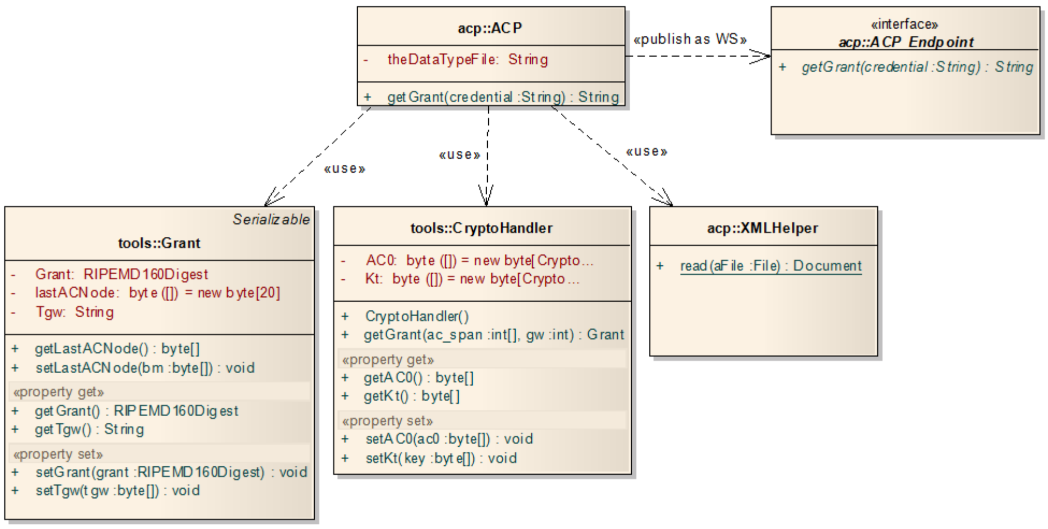

acp::ACP
- theDataTypeFile: String
+ getGrant(credential :String) : String
«publish as WS»
«interface»
acp::ACP_Endpoint
+ getGrant(credential :String) : String
«use»
«use»
«use»
Serializable
tools::Grant
- Grant: RIPEMD160Digest
- lastACNode: byte ([]) = new byte[20]
- Tgw: String
+ getLastACNode() : byte[]
+ setLastACNode(bm :byte[]) : void
«property get»
+ getGrant() : RIPEMD160Digest
+ getTgw() : String
«property set»
+ setGrant(grant :RIPEMD160Digest) : void
+ setTgw(tgw :byte[]) : void
tools::CryptoHandler
- AC0: byte ([]) = new byte[Crypto...
- Kt: byte ([]) = new byte[Crypto...
+ CryptoHandler()
+ getGrant(ac_span :int[], gw :int) : Grant
«property get»
+ getAC0() : byte[]
+ getKt() : byte[]
«property set»
+ setAC0(ac0 :byte[]) : void
+ setKt(key :byte[]) : void
acp::XMLHelper
+ read(aFile :File) : Document

Figure 6.7: Access Control Module Class Diagram

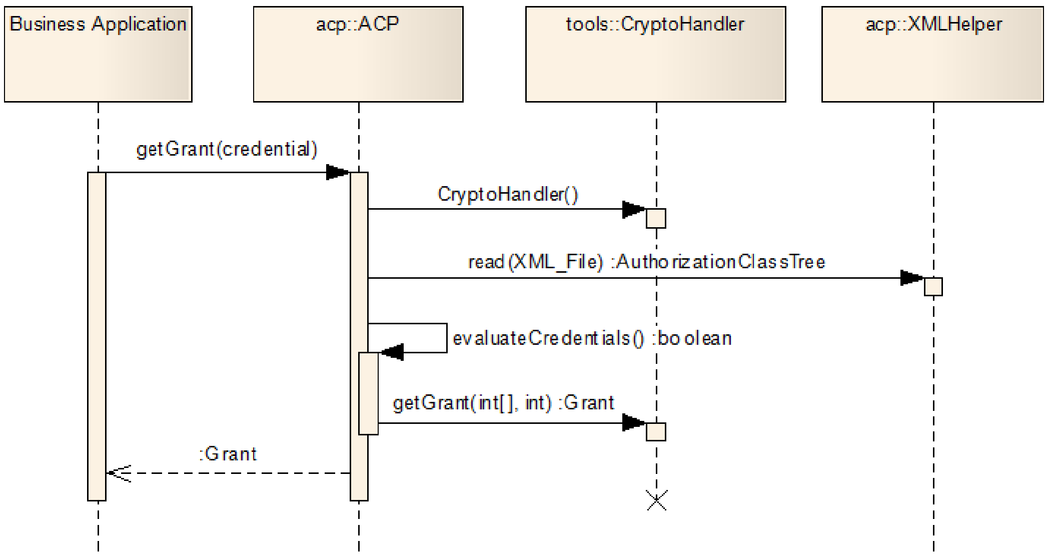


Figure 6.8: Access Control Module Sequence Diagram

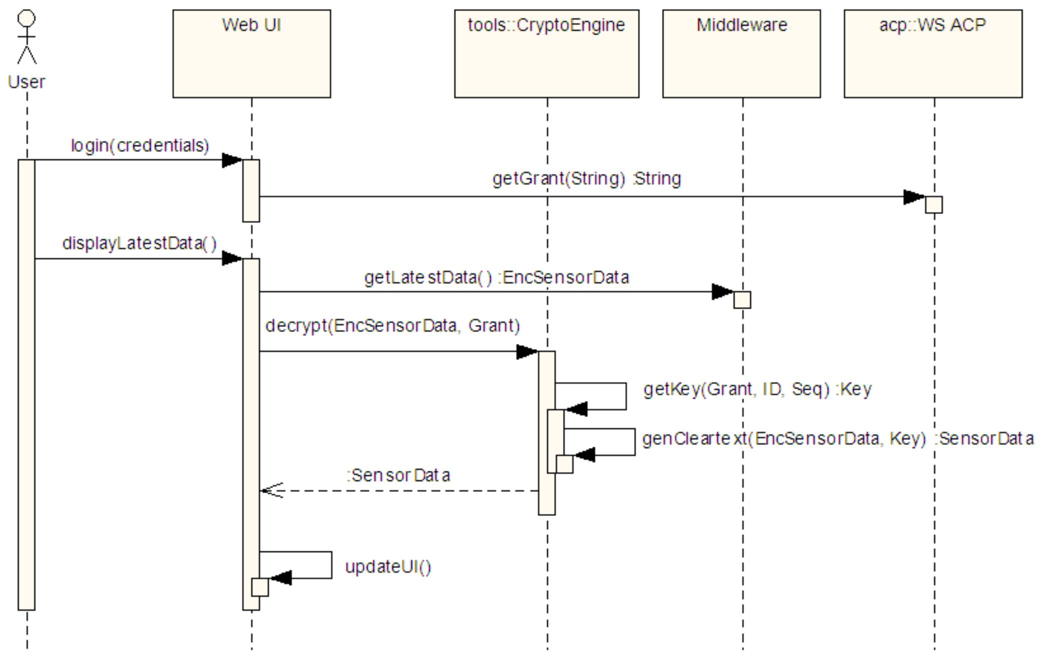


Figure 6.9: Web Dynpro Application Sequence Diagram

6.1.5 Conclusion

In the previous sections we showed the details of the Java Simulation we developed. This prototype enabled us to prove the feasibility of implementing the Access Control Scheme. However, the drawback is that the WSN is simulated, and therefore we are not able to analyze the scheme's efficiency. Thus the next step was to implement the scheme on real hardware.

6.2 Hardware Prototype

A simulation offers a nice environment for testing the functionality of a system before dealing with actual hardware elements. Especially WSN integration into business applications is not considered as an easy task if we compare the interest to use such applications with the amount of working implementations. After our Java prototype showed satisfactory results in terms of feasibility, we decided to create a prototype using real sensor nodes. To that effect, we leveraged Cisco's expertise in WSN hardware and on node programming.

6.2.1 Updated Architecture

The first step in the collaboration is to update the architecture of the system. In Figure 6.11 we see that a few components changed. In particular, the Ptolemy II simulation environment is replaced by a WSN made of Crossbow MTS310[3] Sensor Boards and a Crossbow NB100 Stargate Netbridge Gateway. Also, the Access Control Module is now on a separate hardware entity called the *Cisco Controller* and is

[3]See `http://www.xbow.com/`

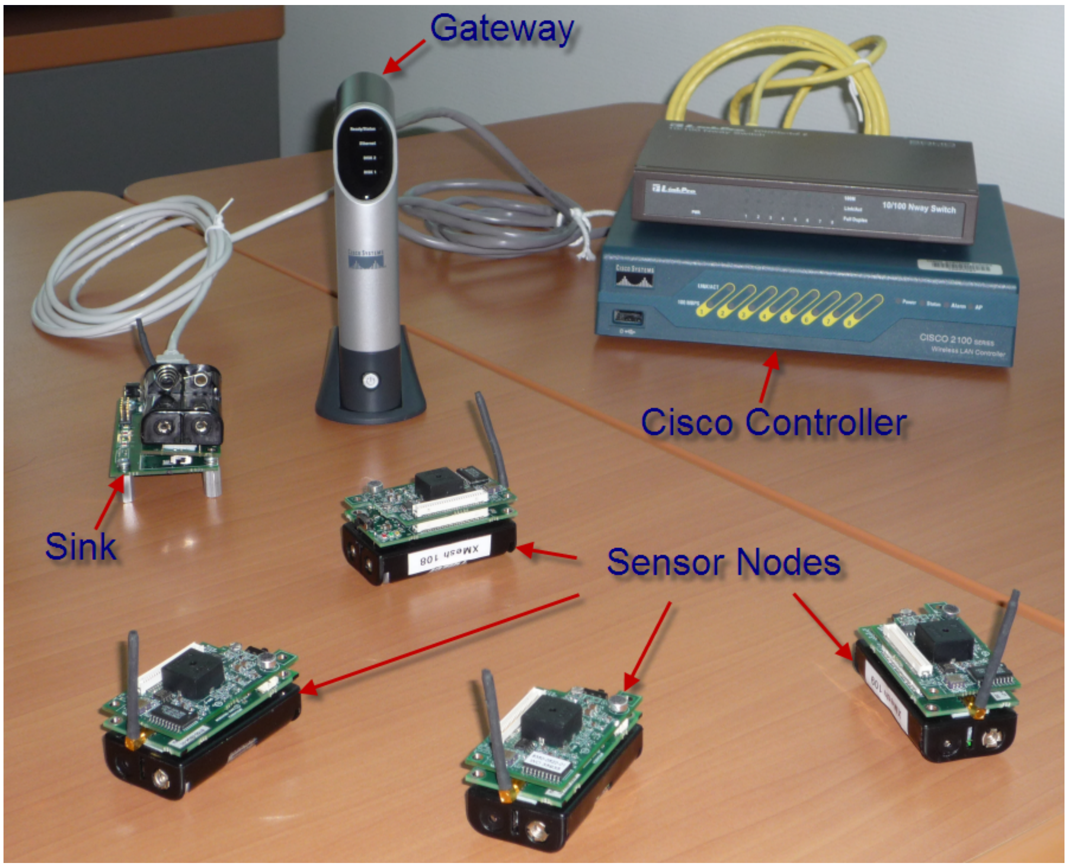


Figure 6.10: Hardware Setup

not hosted on the same machine as the middleware anymore. In a real deployment one would expect that the ACM is on a separate entity than the rest of the middleware for security reasons. If one of the two devices is compromised, the other one can still be functional.

Figure 6.10 shows the hardware setup that we use in our prototype. We identify different components such as the Crossbow MTS310 Sensor Nodes, a Sink, the Crossbow NB100 Stargate Netbridge Gateway and a Cisco Controller that hosts the ACM.

6.2.2 Code Porting

The second step was to rewrite the code for the encryption part on the sensor nodes. In Ptolemy II, we use the Java language which cannot be put on Sensors due to their technical limitations. We rewrote the key generation algorithm using NesC[4], which is the programming language for Crossbow Sensors.

The Access Control Module needed to be rewritten as the Cisco Controller does not provide a Java Virtual Machine. It should be noted that Cisco did the migration of the ACM onto the Controller and that their expertise in sensor programming

[4]See `http://nescc.sourceforge.net/`

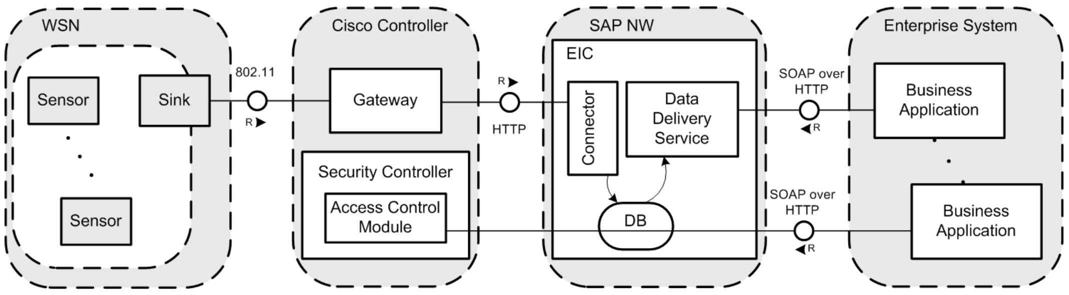


Figure 6.11: Hardware Prototype Architecture

and configuration contributed greatly to the successful completion of this prototype.

6.2.3 Performance Evaluation

In this section, we provide some empirical performance indicators of the implemented access control scheme. Due to time constraints, we are not able to provide precise performance evaluation figures at this point. Instead, we will present a first estimate based on our daily usage of the nodes.

There is a rough 33% overhead on the transmitted messages compared to those without encryption. This overhead is explained by the size of the hash (160 bits) and the additional parameters *Seq, ID, GW, AC_span* that are sent along with the encrypted value.

Also, we observed a 33% increase in battery consumption when using the encryption scheme. The reasons for this are mainly (i) the longer transmission and reception times due to the increased size of the messages and (ii) the additional computations that are required to generate the keys. We note that the computational overhead is minimal compared to the transceiver's increased power consumption. Last but not least, we did not perceive any noticeable latency in the flow of generated sensor data.

6.2.4 Publication

The outcome of this collaboration is a publication [41] published in the MobiQuitous 2008 conference in Dublin. The paper presents the prototype that was developed in collaboration with Cisco System France. It can be found as addendum to this document.

6.3 Conclusion

In this chapter we showed the implementation details of the Java proof-of-concept and the Hardware prototype that were developed during the course of this thesis. In particular, we showed how the Java implementation lead to a collaboration with Cisco Labs France to create a hardware prototype. We also provided some empirical performance evaluation that forms a basis for more precise and thorough evaluations for the future. These performance indicators will allow a comparison of the efficiency to other similar security schemes than the one we implemented.

During the implementation phase, we realized that the Enterprise Integration Component is a modular and flexible WSN middleware. The only major change inside of the EIC was the development of a new *Connector Service*. Since all components of our architecture are based on the SOA paradigm and as such are independent from each other, no other services of the EIC required additional modifications. Furthermore, the web-based web dynpro application first developed for the Java proof-of-concept was reusable since the application is completely independent of the backend infrastructure.

We would like to stress that our work was validated through a publication [41] at the Mobiquitous 2008 conference (see Section 6.2.4).

CHAPTER 7

CONCLUSION

This Master Thesis dealt with the secure integration of WSNs into business applications. By analyzing the impact of BSNs and ASNs on healthcare systems, we realized that the vision of ubiquitous remote patient monitoring raises many security concerns. However, not knowing precisely *what* these security concerns were, we were led to the formulation of the following question: "Why do we need security, and what do we secure?"

To answer this question, we identified the inherent risks in a healthcare system following the NIST Risk Assessment methodology. We found that data exposure constitutes the highest risk. Up until now, patients went to their doctor for health checkups and as such their medical data was less likely to be exposed and maintaining their privacy was easy. However, with the integration of WSNs into healthcare systems, end-to-end confidentiality became an obvious security requirement. This is because for example the sensor data is exposed in clear in WSNs and can be sniffed with a simple antenna (see section 3.4).

To fill this requirement, we proposed an existing security scheme [83] that offers end-to-end confidentiality of sensor data based on symmetric encryption. Seeing as the authors did not analyze the security of the scheme, another question rose: "How secure is this scheme?". We answered this question with the analysis of the scheme's security (by looking at outside and inside attacks). This analysis was followed by an evaluation of the pseudo-randomness of the encryption key generator. Given the hypotheses stated in the scheme's paper, evaluating the randomness was a crucial step to prove the scheme's security. We showed satisfactory results for the scheme through the Security Analysis and we also proved that the key streams generated are pseudo-random.

At this point the concern of the feasibility of the scheme in practice was raised. Indeed, quite often the transition from theory to real implementations is not possible. However, we validated it by implementing the scheme first in Java then on real sensor nodes. Following empirical performance evaluations of this hardware prototype we were able to support the claim of the authors that their scheme is efficient and can be considered in real life applications.

This thesis allowed to get in touch with a vast number of different domains, ranging from WSNs and middlewares to the requirements of business applications and in particular healthcare systems. So far, all these domains did not overlap and research was focused in the individual research fields. However, the integration of all these domains into one system opens so many new applications that we believe it will attract more interest in the future.

With this project we wanted to show the potential of the integration of WSNs and in particular BSNs and ASNs with business applications. We saw that it is possible and can be performed in a secure way. To conclude we believe that security will be taken more and more seriously in the future as the world is moving towards the total interconnection of the human and the Internet.

7.1 Future Work

Due to time constraints in this thesis, some research directions could not be addressed properly. We consider the following list of research problems as an interesting starting point for future research based on this thesis.

Key Distribution Schemes for the security scheme in WSNs One of the hypotetheses in the scheme's paper is that the cryptographic material distributed by the Access Control Module to the sensor nodes uses an authenticated and secure channel. Since this issue was not addressed by the authors, we suggest to investigate key distribution schemes for WSNs such as TESLA [70] and random key-distribution with [31] or without [23] threshold cryptography. We believe that these schemes are efficient enough to be used with resource constrained WSNs like the ones we use in this project.

Access Control Policies In the security scheme we proposed, the authors did not cover which access control policies the Access Control Module should use. Therefore, in the current implementation, the ACM uses an XML file that holds the hierarchy roles and authorization classes. However, this proprietary file format does not conform to any standards. It would be likeable to replace this access control policy with one based on an open standard such as XACML [62]. XACML offers the handling of security tokens such as SAML [60] or X.509 certficates [9]. Such security tokens will be useful to increase the security particularly on the Grant distribution part.

Performance Evaluation A main concern with WSNs is that sensors are low-power devices. Especially when dealing with cryptography, sensors are very sensible, since some algorithms can be very resource demanding (such as the emerging Elliptic Curve Cryptography).

Performance evaluation thus becomes an important criterion when evaluating the feasibility of using a particular security scheme in real life applications. A first estimation of the scheme's performance can be found in Section 6.2.3.

However, more extensive testing should be done by looking at different parameters such as battery exhaustion, CPU usage, introduced delay and bandwidth overhead. We believe that for example the optimization of the code on the sensor nodes could lead to a slower battery exhaustion.

In a next step, the performance of the scheme in [83] could be compared to the performance of other WSN hierarchical security schemes. This would allow to precise how efficient the scheme really is compared to others.

BIBLIOGRAPHY

[1] Cobis consortium. cobis. fp strep project ist 004270. See `www.cobis-online.de`.

[2] Cramm information security toolkit. Technical report, Siemens. See `http://www.cramm.com/`. Accessed June, 16 2008.

[3] European network and information security agency (enisa).

[4] A java-based component assembly framework. Technical report, EECS, UC Berkeley. See `http://ptolemy.berkeley.edu/ptolemyII/`.

[5] Java Remote Method Invocation. See `http://java.sun.com/rmi`.

[6] Runes consortium. runes, ist-004536-runes. See `www.ist-runes.org`.

[7] Secure sphere crew. dns id spoofing. See `http://www.securesphere.net/download/papers/dnsspoof.htm`. Accessed May 21, 2008.

[8] Web Dynpro Java: SAP's Standard UI Technology for developing User Interfaces. See `https://www.sdn.sap.com/irj/sdn/nw-wdjava`.

[9] ITU-T recommendation X.509, the directory: Authentication framework, int'l telecomm. union, geneva, iso/iec 9594-8, 2000.

[10] 10 Emerging Technologies That Will Change the World. *MIT Technology Review*, pages 33–49, 2003.

[11] NIST 800-22b. Download documentation and software for the nist 800-22b special publication. See `http://csrc.nist.gov/groups/ST/toolkit/rng/documentation%5Fsoftware.html`.

[12] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: a Survey. *Comput. Netw.*, 38(4):393–422, 2002.

[13] C. Alberts, A. Dorofee, J. Stevens, and C. Woody. Introduction to the OCTAVE Approach. Technical report, Carnegie Mellon Software Engeneering Institute, August 2003.

[14] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services - Concepts, Architectures and Applications*. Springer, November 2003.

[15] M. Bellare, R. Canetti, and H. Krawczyk. Message authentication using hash functions: the HMAC construction. *CryptoBytes*, 2(1):12–15, Spring 1996.

[16] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[17] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, 1986.

[18] S. Buchegger and J.-Y. Le Boudec. A robust reputation system for P2P and Mobile ad-hoc Networks, 2004.

[19] R. N. Butler. Population aging and health. *British Medical Journal*, 315(7115):1082–1084, 1997.

[20] L. Buttyan and J.-P. Hubaux. *Security and Cooperation in Wireless Networks*. Cambridge University Press, November 2007.

[21] E. D. Cardenas. Mac spoofing – an Introduction. Technical report, GIAC Security Essentials Certification (GSEC), August 2003.

[22] H. Chan and A. Perrig. Security and privacy in sensor networks. *Computer*, 36(10):103–105, 2003.

[23] H. Chan, A. Perrig, and D. Song. Random key predistribution schemes for sensor networks, 2003.

[24] H.-Y. Chien. Efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1301–1304, 2004.

[25] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460 (Draft Standard), December 1998. Updated by RFC 5095.

[26] Deloitte. Global security survey, 2006. See `http://www.deloitte.com/`. Accessed June 11, 2008.

[27] Defensive Healthcare Information Comparative Analysis (DHIAP). Octave - best practices comparative analysis. Technical report, U.S. Army Medical Research and Material Command, June 2003.

[28] T. Dierks and E. Rescorla. The transport layer security (TLS) protocol version 1.1. RFC 4346, Internet Engineering Task Force, April 2006.

[29] H. Dobbertin, A. Bosselaers, and B. Preneel. RIPEMD-160: A Strengthened Version of RIPEMD. In *Fast Software Encryption*, pages 71–82, 1996.

[30] D. Dolev and A. C. Yao. On the security of public key protocols. Technical report, Stanford University, Stanford, CA, USA, 1981.

[31] W. Du, J. Deng, Y. Han, and P. Varshney. A pairwise key pre-distribution scheme for wireless sensor networks, 2003.

[32] Council European Parliament. Directive 95/46/ec of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, 1995.

[33] N. Ferguson and B. Schneier. *Practical Cryptography.* J. Wiley & Sons, Inc., New York, NY, USA, 2003.

[34] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. RFC 2827 (Best Current Practice), May 2000. Updated by RFC 3704.

[35] R. Finlayson, T. Mann, J. C. Mogul, and M. Theimer. RFC 903: Reverse Address Resolution Protocol, June 1984. Status: STANDARD.

[36] Bundesamt fuer Sicherheit in der Informationstechnick (BSI). BSI Standard 100-3. Risk Analysis based on IT-Grundschutz.

[37] S. Ganeriwal and M. B. Srivastava. Reputation-based framework for high integrity sensor networks. In *SASN '04: Proceedings of the 2nd ACM workshop on Security of ad hoc and sensor networks*, pages 66–77, New York, NY, USA, 2004. ACM.

[38] IESC Geekcorps. How to make a bottlenet antenna. accessed june 25, 2008., November 2006.

[39] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. Irisnet: An architecture for a worldwide sensor web. *IEEE Pervasive Computing*, 02(4):22–33, 2003.

[40] A. S. Go, E. M. Hylek, K.A. Phillips, Y. Chang, L.E. Henault, J.V. Selby, and D. E. Singer. Prevalence of Diagnosed Atrial Fibrillation in Adults: National Implications for Rhythm Management and Stroke Prevention: the AnTicoagulation and Risk Factors In Atrial Fibrillation (ATRIA) Study. *JAMA*, 285:2370–2375, 2001.

[41] L. Gomez, A. Laube, V. Ribiere, A. Sorniotti, C. Trefois, M. Valente, and P. Wetterwald. Encryption-based access control for building management. In *MobiQuitous '08: In Proceedings of the Fifth Annual International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Dublin, Ireland, July 2008. ICST.

[42] L. Gomez, A. Laube, and A. Sorniotti. Design guidelines for integration of wireless sensor networks within enterprise systems. In ACM Digital Library, editor, *in the proceedings of the International Conference on on MOBILe Wireless MiddleWARE, Operating Systems, and Applications*, 2008.

[43] L. Gomez and I. Thomas. Towards user authentication flexibility, 2008.

[44] G. Gupta and M. Younis. Fault-tolerant clustering of Wireless Sensor Networks. *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, 3:1579–1584 vol.3, 20-20 March 2003.

[45] G. Hackmann, C.-L. Fok, R. Gruia-catalin, and C. Lu. C.: Agimone: Middleware support for seamless integration of sensor and IP networks. In *IP networks, in Proc. of IEEE DCOSS*, pages 101–118, 2006.

[46] D. Halperin, T. S. Heydt-Benjamin, K. Fu, T. Kohno, and W. H. Maisel. Security and privacy for implantable medical devices. *IEEE Pervasive Computing*, 7(1):30–39, 2008.

[47] S. Harte, A. Rahman, and K.M. Razeeb. Fault tolerance in sensor networks using self-diagnosing sensor nodes. *IEE Seminar Digests*, 2005(11059):v2–7–v2–7, 2005.

[48] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo. Middleware to support Sensor Network applications. *IEEE Network*, 18:2004, 2004.

[49] KLC Consulting Inc. SMAC. See `http://www.klcconsulting.net/smac/`. Accessed May 23, 2008.

[50] S. Kent and K. Seo. Security architecture for the internet protocol. RFC 4301, Internet Engineering Task Force, December 2005.

[51] S. Kim, K. Umeno, and A. Hasegawa. Corrections of the NIST statistical test suite for randomness, 2004.

[52] G.E. Krasner and S. T. Pope. A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. *J. Object Oriented Program.*, 1(3):26–49, 1988.

[53] Sandia National Laboratories. Security risk assessment methodologies. http://www.sandia.gov/ram/index.htm. accessed june, 16 2008.

[54] G. Masaglia. The marsaglia random number cdrom including the diehard battery of tests of randomness, 1995. See `http://www.stat.fsu.edu/pub/diehard/`. Accessed July, 17 2008.

[55] J-O. Mauborgne and G. Vernam. One-time pad. Accessed on April 18.

[56] G. McGraw and G. Morrisett. Attacking malicious code: a report to the infosec research council. *Software, IEEE*, 17(5):33–41, Sep/Oct 2000.

[57] A. J. Menezes, S. A. Vanstone, and P. C. Van Oorschot. *Handbook of Applied Cryptography.* CRC Press, Inc., Boca Raton, FL, USA, 1996.

[58] P. V. Mockapetris. RFC 1035: Domain names — implementation and specification, November 1987. Obsoletes RFC0973, RFC0882, RFC0883. Status: STANDARD.

[59] J. Mundinger and J.-Y. Le Boudec. Reputation in self-organized communication systems and beyond. In *interperf '06: Proceedings from the 2006 workshop on Interdisciplinary systems approach in performance evaluation and design of computer & communications sytems*, page 3, New York, NY, USA, 2006. ACM.

[60] OASIS. Security Assertion Markup Language (SAML). See `www.oasis-open.org/committees/security/`.

[61] OASIS. Core and hierarchical role based access control (RBAC) profile of xacml v2.0., February 2005.

[62] OASIS. XACML 2.0 core:eXtensible Access Control Markup Language (XACML) version 2.0, February 2005.

[63] L. Odorico. Simulation of a security scheme for Wireless Sensor Networks. Master's thesis, Eurecom, September 2007.

[64] National Institute of Standards and Technolgy (NIST). Secure hash standard. Internet, August 2002.

[65] National Institute of Standards and Technology (NIST). Risk Management Guide for Information Technology Systems. Special Publication 800-30. Internet, July 2002.

[66] U.S. General Accounting Office. Protecting information systems supporting the federal government and the nation's critical infrastructures, gao-03-121. Technical report, January 2003.

[67] Committee on National Security Systems. National information assurance (ia) glossary, instruction no. 4009. Internet, June 2006. Accessed on April 28.

[68] A. L. Ortega. Accessed may, 23 2008. See `http://www.alobbs.com/macchanger/`.

[69] M.P. Papazoglou. Service-oriented computing: concepts, characteristics and directions. *Proceedings of the Fourth International Conference on Web Information Systems Engineering (WISE '03)*, pages 3–12, Dec. 2003.

[70] A. Perrig, R. Canetti, J. D. Tygar, and D. X. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[71] C. P. Pfleeger and S. L. Pfleeger. *Security in Computing (4th Edition)*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2006.

[72] J. Postel. Internet Protocol. RFC 791 (Standard), September 1981. Updated by RFC 1349.

[73] M. Raya, P. Papadimitratos, V. D. Gligor, and J.-P. Hubaux. On Data-Centric Trust Establishment in Ephemeral Ad Hoc Networks. Technical report, 2007.

[74] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST special publication 800-22, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA, 2001. See `http://csrc.nist.gov/rng/`.

[75] SecureWorks. Dns cache poisoning. See `http://www.secureworks.com/research/articles/dns-cache-poisoning/`. Accessed on May 26, 2008.

[76] C. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, 28(4):656–715, 1949.

[77] C. E. Shannon. The mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, 1948.

[78] J. Shneidman, P. Pietzuch, J. Ledlie, M. Roussopoulos, M. Seltzer, and M. Welsh. Hourglass: An infrastructure for connecting sensor networks and applications. Technical report, 2004.

[79] IEEE Computer Society. IEEE Std 802.3. Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer (PHY) specifications, 2005.

[80] IEEE Computer Society. IEEE Std 802.15.4. Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Internet, June 2006.

[81] IEEE Computer Society. IEEE Std 802.11. Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, 2007.

[82] T.J. Socolofsky and C.J. Kale. TCP/IP tutorial. RFC 1180 (Informational), January 1991.

[83] A. Sorniotti, R. Molva, and L. Gomez. Efficient access control for wireless sensor data. *to appear in the 19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2008.

[84] F. Stajano and R. Anderson. The resurrecting duckling: security issues for ubiquitous computing. *Computer*, 35(4):22–26, Apr 2002.

[85] D. Stinson. A polemic on notions of cryptographic security. Internet, 2004.

[86] Y. L. Sun, W. Yu, Z. Han, and K. J. Ray Liu. Information theoretic framework of trust modeling and evaluation for ad hoc networks. *IEEE Journal on Selected Areas in Communications*, 24(2):305–317, 2006.

[87] M. Tanase. IP Spoofing: An introduction, March 2003. See `http://www.securityfocus.com/infocus/1674`. Accessed on May 27, 2008.

[88] G. Theodorakopoulos and J. S. Baras. On Trust Models and Trust Evaluation Metrics for Ad-Hoc Networks. *IEEE Journal on Selected Areas in Communications*, 24(2):318–328, 2006. Received the IEEE Communications Society Leonard G. Abraham Prize in the Field of Communications Systems.

[89] W. Tolone, G.-J. Ahn, T. Pai, and S.-P. Hong. Access control in collaborative systems. *ACM Comput. Surv.*, 37(1):29–41, 2005.

[90] WASP. WASP (Wirelessly Accessible Sensor Populations), IST 034963, 2006. See `http://www.wasp-project.org`.

[91] Q. Zhang, T. Yu, and P. Ning. A framework for identifying compromised nodes in wireless sensor networks. *ACM Trans. Inf. Syst. Secur.*, 11(3), 2008.

[92] W. Zhang, S. K. Das, and Y.Liu. A trust based framework for secure data aggregation in wireless sensor networks. In *Sensor and Ad Hoc Communications and Networks, 2006. SECON '06. 2006 3rd Annual IEEE Communications Society on*, volume 1, pages 60–69, September 2006.

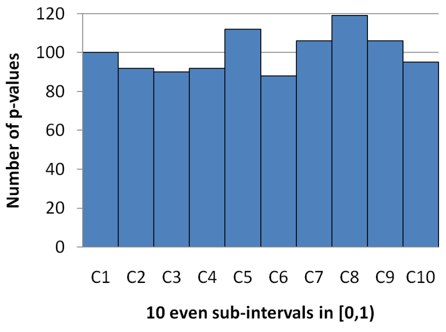
[93] Y. G. Zhong. *Body Sensor Networks*. Springer, 2006.

[94] H. Zimmermann. Osi reference model - the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980.

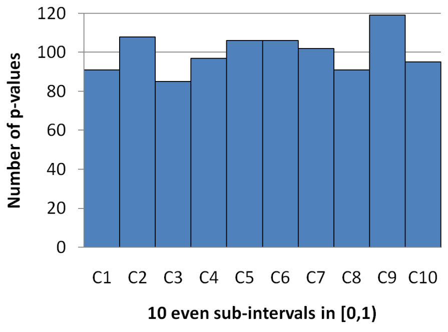
APPENDIX A

UNIFORMITY HISTOGRAMS

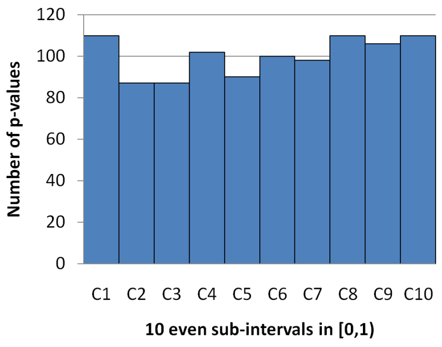
In this section, we present all the histograms for the tests we performed in Chapter 5.



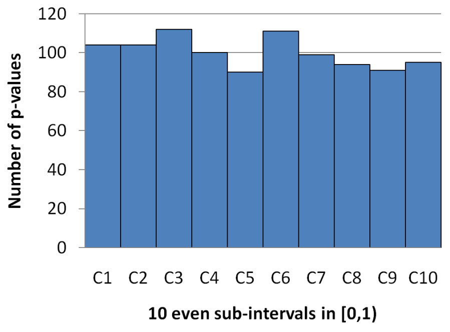
(a) Frequency Test



(b) Block Frequency Test

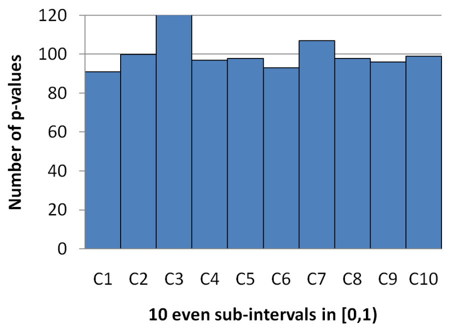


(c) Cumulative Sums Test

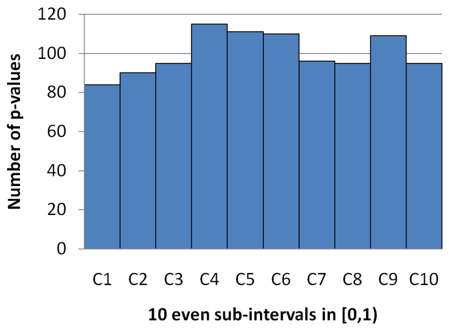


(d) Runs Test

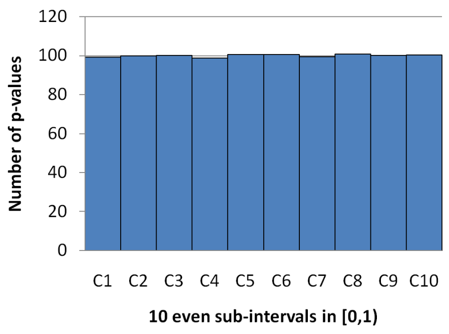
Figure A.1: Histograms



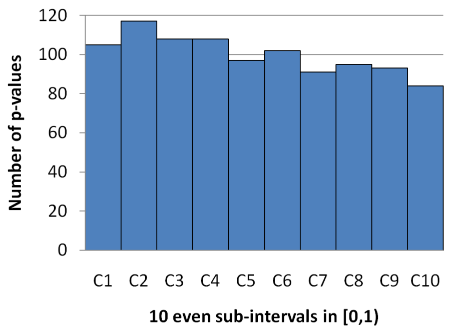
(a) Longest Run of Ones Test



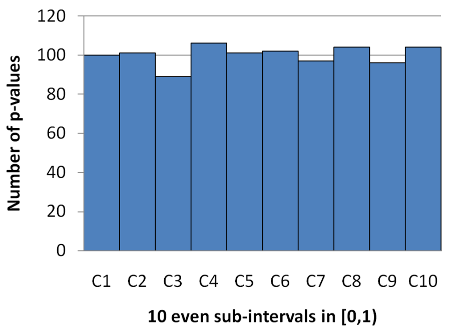
(b) Rank Test



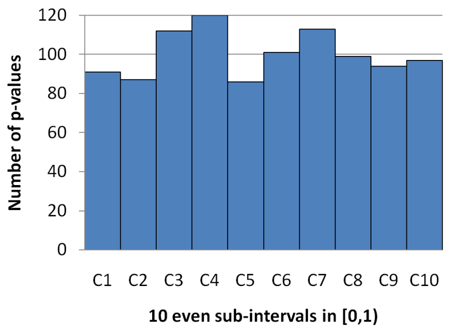
(c) Non-Periodic Template Matching Test



(d) Overlapping Template Matching Test

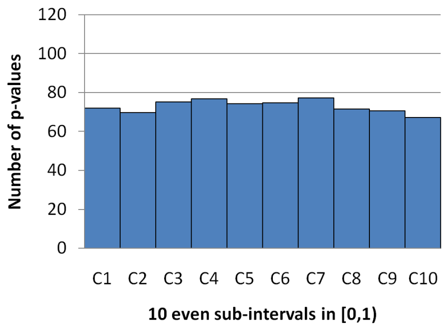


(e) Universal Statistical Test

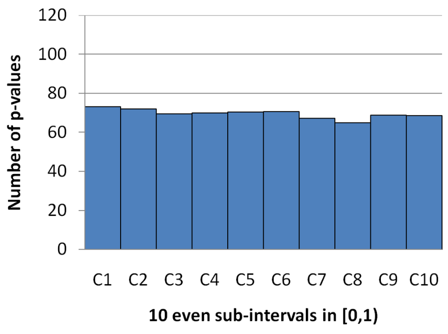


(f) Approximate Entropy Test

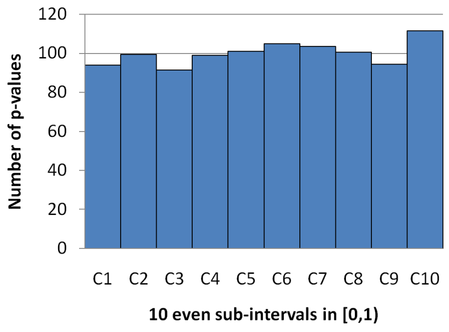
Figure A.2: Histograms



(a) Random Excursions



(b) Random Excursions Variant Test



(c) Serial Test

Figure A.3: Histograms

APPENDIX B

MOBIQUITOUS 2008 DEMO PAPER

Please find the published MobiQuitous 2008 Demo Paper [41] as addendum to this document.

APPENDIX C

ACCESS CONTROL PAPER

Please find the paper about the Access Control Scheme [83] as addendum to this document.

APPENDIX D

APPLICATION EXAMPLE

In this Appendix, we show how a Business Application looks like. This is the actual prototype that was shown at the MobiQuitous 2008 Conference. The reader can refer to [41] for further information. The paper is also available as addendum to this document.

In the demo business application we use the hierarchy of roles depicted in Figure D.1.

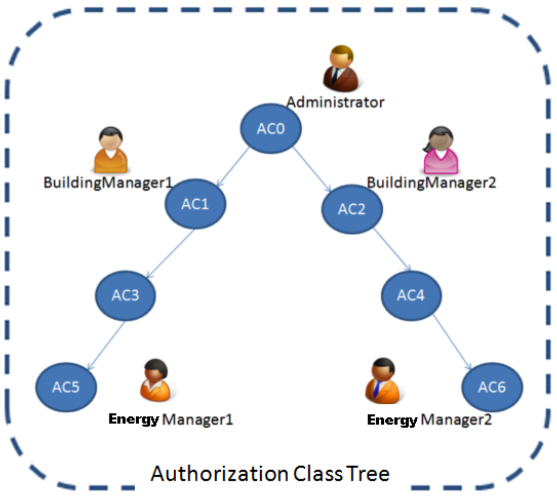
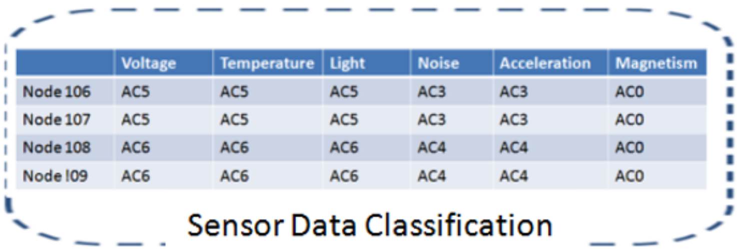


Figure D.1: Role Hierarchy

The mapping for each Authorization Class on the Sensor Nodes is shown in Figure D.2. We can see that different authorization classes can be mapped on the same sensor node.

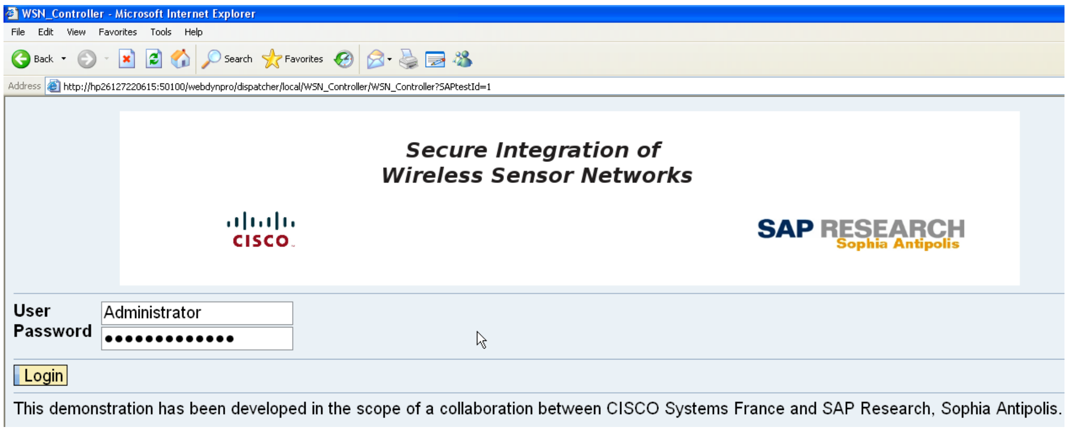
As a first Step, we log in the web interface using the *Administrator* role (see Figure D.3). Figure D.4 shows the user interface once we successfully logged in.



	Voltage	Temperature	Light	Noise	Acceleration	Magnetism
Node 106	AC5	AC5	AC5	AC3	AC3	AC0
Node 107	AC5	AC5	AC5	AC3	AC3	AC0
Node 108	AC6	AC6	AC6	AC4	AC4	AC0
Node !09	AC6	AC6	AC6	AC4	AC4	AC0

Sensor Data Classification

Figure D.2: Authorization Class Mapping



WSN_Controller - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://hp26127220615:50100/webdynpro/dispatcher/local/WSN_Controller/WSN_Controller?SAPtestId=1

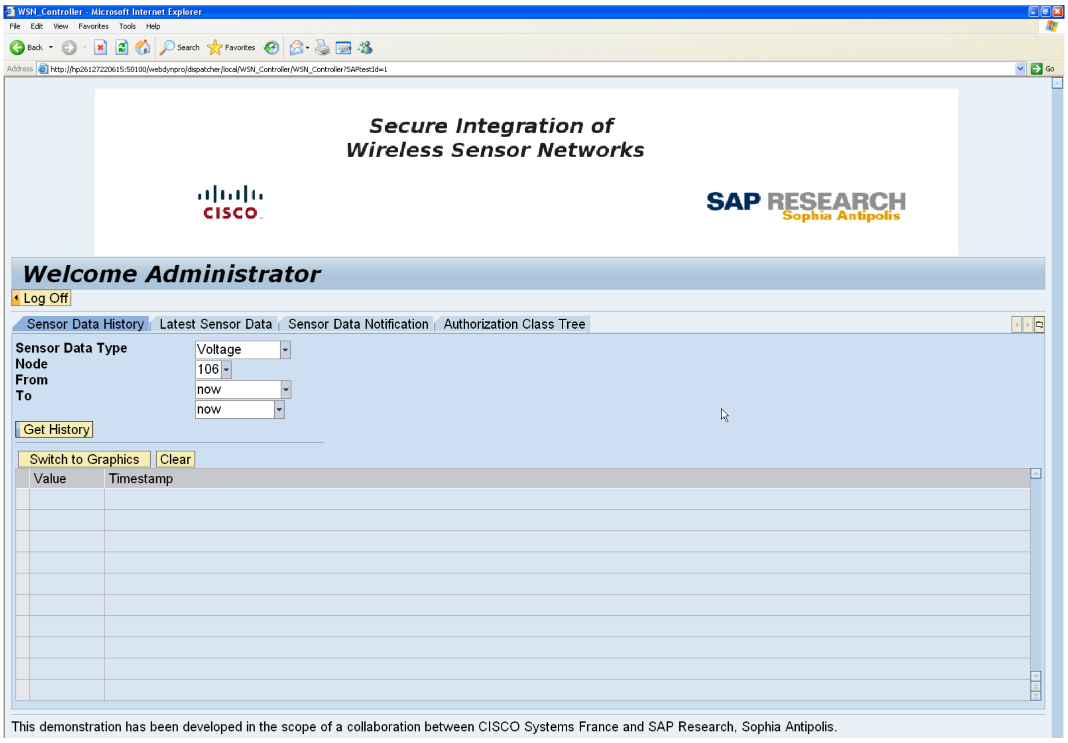
Secure Integration of
Wireless Sensor Networks

User Administrator
Password ●●●●●●●●●●●●●

Login

This demonstration has been developed in the scope of a collaboration between CISCO Systems France and SAP Research, Sophia Antipolis.

Figure D.3: Logging in as Administrator



WSN_Controller - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://hp26127220615:50100/webdynpro/dispatcher/local/WSN_Controller/WSN_Controller?SAPtestId=1
Go

Secure Integration of
Wireless Sensor Networks

Welcome Administrator

Log Off

Sensor Data History | Latest Sensor Data | Sensor Data Notification | Authorization Class Tree

Sensor Data Type	Voltage
Node	106
From	now
To	now

Get History

Switch to Graphics Clear

Value	Timestamp

This demonstration has been developed in the scope of a collaboration between CISCO Systems France and SAP Research, Sophia Antipolis.

Figure D.4: General View of the User Interface

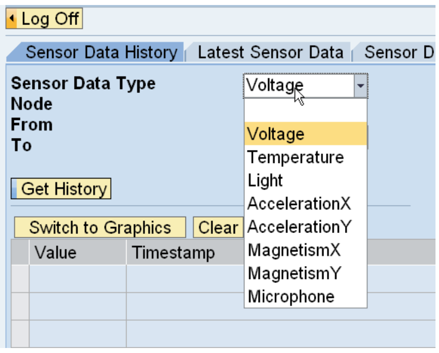
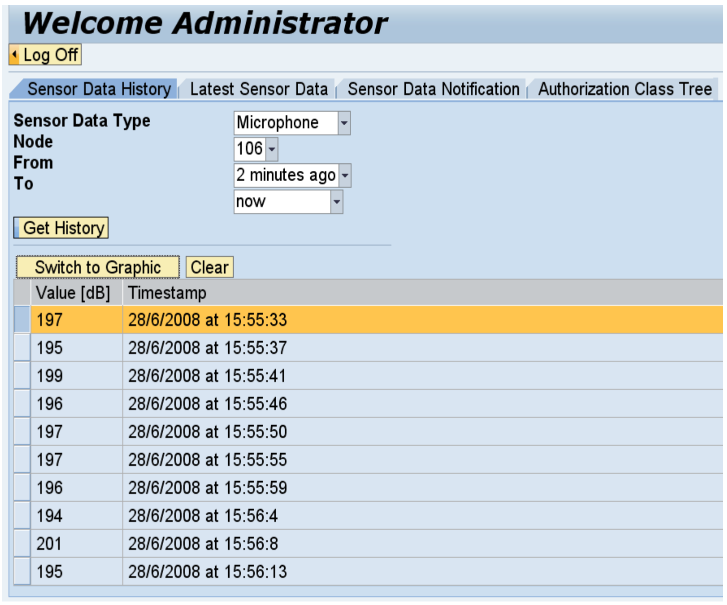


Figure D.5: List of Available Sensor Data


Welcome Administrator

Log Off

Sensor Data History | Latest Sensor Data | Sensor Data Notification | Authorization Class Tree

Sensor Data Type	Microphone
Node	106
From	2 minutes ago
To	now

Get History

Switch to Graphic | Clear

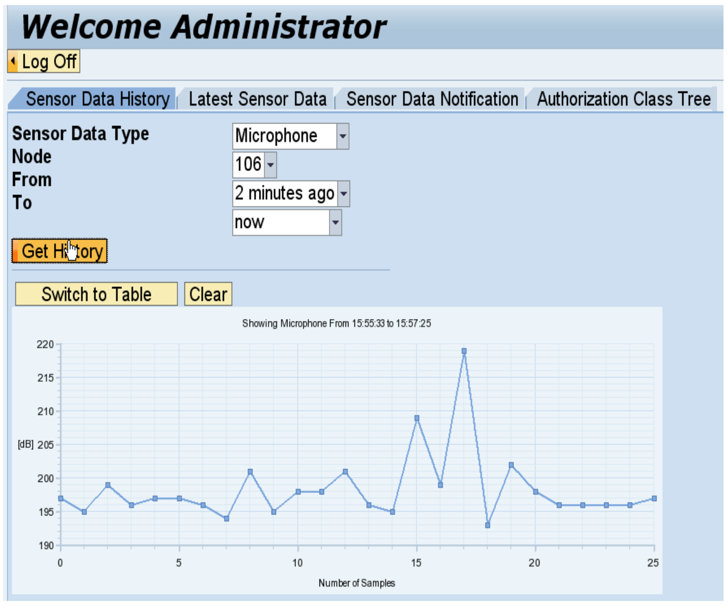
Value [dB]	Timestamp
197	28/6/2008 at 15:55:33
195	28/6/2008 at 15:55:37
199	28/6/2008 at 15:55:41
196	28/6/2008 at 15:55:46
197	28/6/2008 at 15:55:50
197	28/6/2008 at 15:55:55
196	28/6/2008 at 15:55:59
194	28/6/2008 at 15:56:4
201	28/6/2008 at 15:56:8
195	28/6/2008 at 15:56:13

Figure D.6: Historic Sensor Data as Table

The *Administrator* can now select a *Sensor Data Type* from the list as in Figure D.5. Furthermore, he can select the Node on which the Data should come from. Finally, after selecting the timespan for the historical data (e.g. from *2 minutes ago* until *Now*), the User clicks on the *Get Historic* button to receive the list of Sensor Data meeting that selection criteria (as in Figure D.6). Alternatively, the User can switch to a graphical representation of the Historic Data by clicking on the *Switch to Graphics* button (see Figure D.7 for an example).

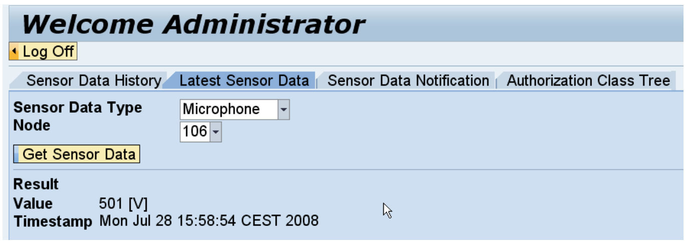
The second feature of the User Interface is the *Latest Sensor Data* Tab. After picking the wanted parameters, the User receives the latest sensor data available by clicking on the *Get Sensor Data* (the result is depicted in Figure D.8).

The third option in the User Interface is called *Sensor Data Notification*. This feature allows for periodic updating of a selected sensor data type. This means that each XX seconds, the user interface is updated with the latest sensor data available. This enables the User to monitor the evolution of the Sensor Data. An example



Sensor Data Type	Microphone
Node	106
From	2 minutes ago
To	now

Figure D.7: Historic Sensor Data as Graphic

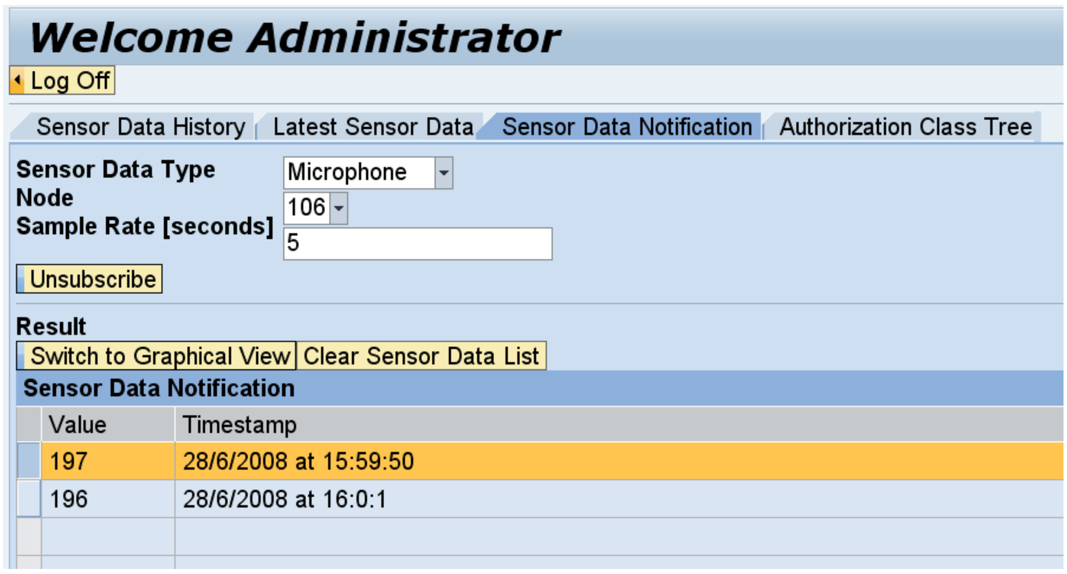


Sensor Data Type	Microphone
Node	106

Result

Value	501 [V]
Timestamp	Mon Jul 28 15:58:54 CEST 2008

Figure D.8: Latest Sensor Data



Sensor Data Type: Microphone
Node: 106
Sample Rate [seconds]: 5

Result

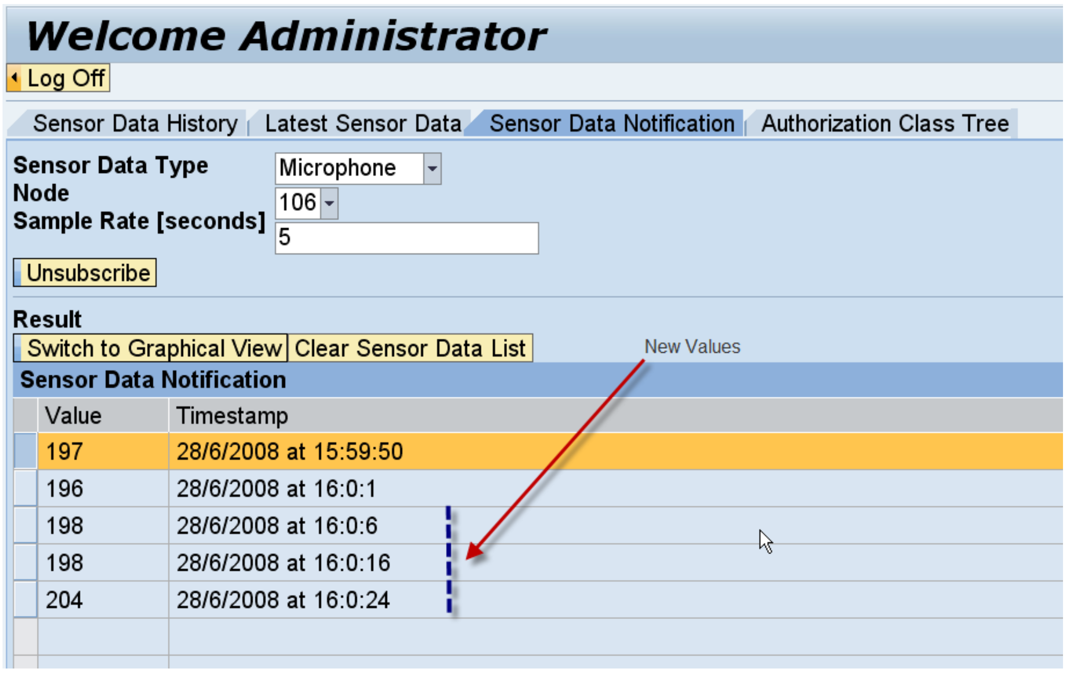
Sensor Data Notification

Value	Timestamp
197	28/6/2008 at 15:59:50
196	28/6/2008 at 16:0:1

Figure D.9: First Notification in Table View

of notification is provided in Figures D.9, D.10 (for a textual view) and in Figures D.11, D.12 for a graphical representation.

In this part, we log in with the *TemperatureManager* role (See Figure D.13). This role can only decrypt *Temperature* related Sensor Data. If the User tries to decrpyt Sensor Data different than *Temperature*, the Interface notifies the User by displaying stars (∗∗∗∗∗∗∗) to show that this User is not allowed to read the values (see Figure D.14).



Welcome Administrator

Log Off

Sensor Data History | Latest Sensor Data | Sensor Data Notification | Authorization Class Tree

Sensor Data Type	Microphone
Node	106
Sample Rate [seconds]	5

Unsubscribe

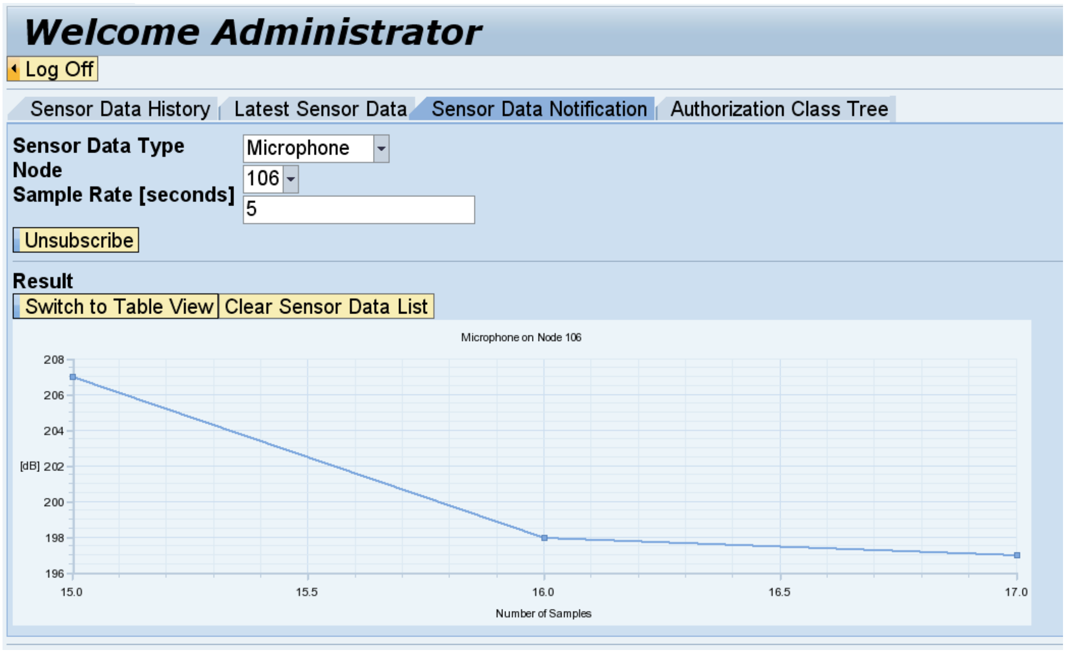
Result

Switch to Graphical View | Clear Sensor Data List

Sensor Data Notification

Value	Timestamp
197	28/6/2008 at 15:59:50
196	28/6/2008 at 16:0:1
198	28/6/2008 at 16:0:6
198	28/6/2008 at 16:0:16
204	28/6/2008 at 16:0:24

Figure D.10: Notification 5 seconds later in Table View



Welcome Administrator

Log Off

Sensor Data History | Latest Sensor Data | Sensor Data Notification | Authorization Class Tree

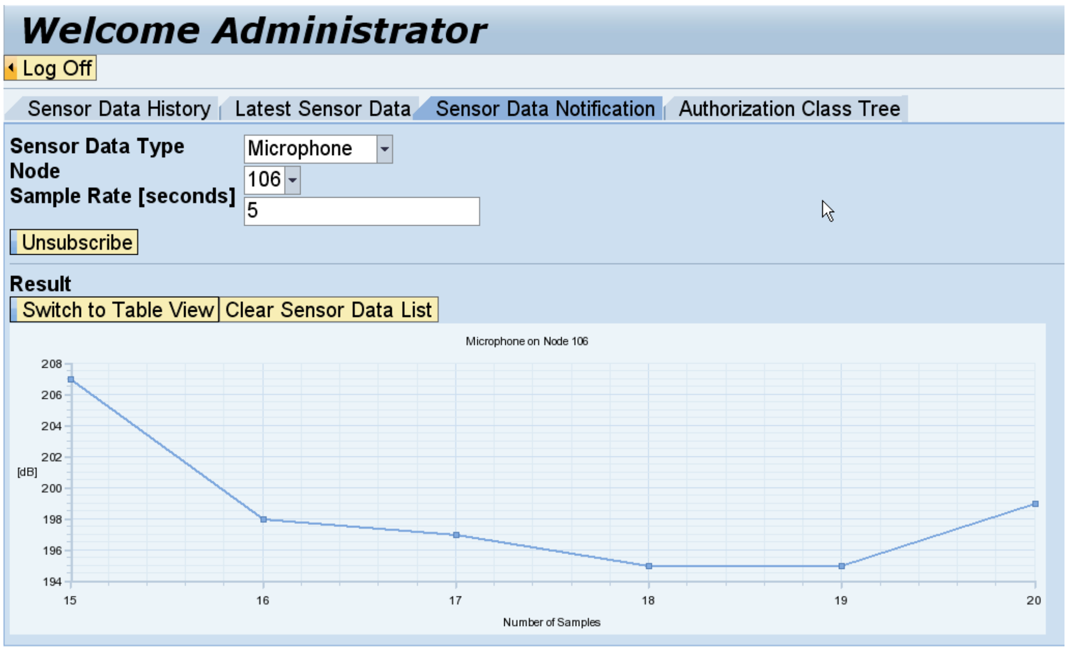
Sensor Data Type	Microphone
Node	106
Sample Rate [seconds]	5

Unsubscribe

Result

Switch to Table View | Clear Sensor Data List

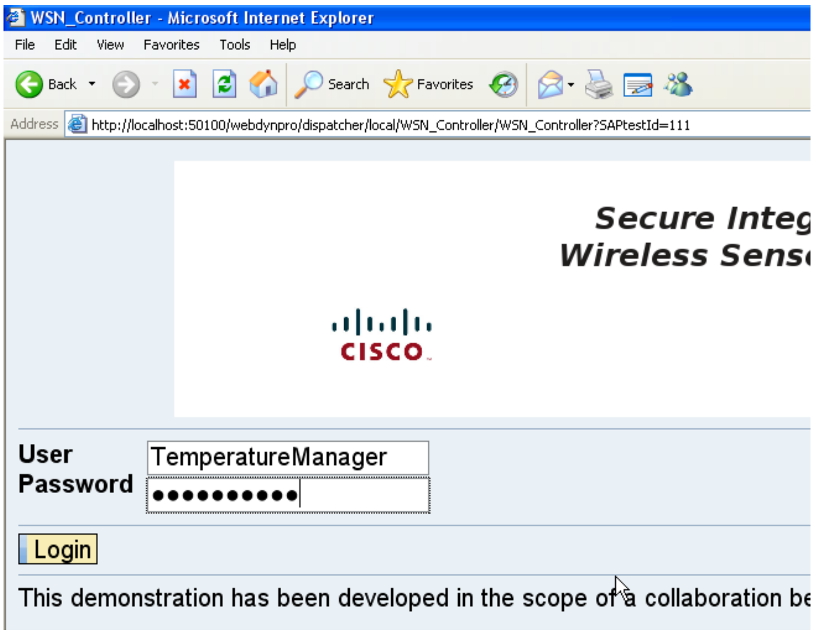
Figure D.11: Notification in Graphical View



Sensor Data Type Microphone
Node 106
Sample Rate [seconds] 5

Result

Figure D.12: Notification 5 seconds later

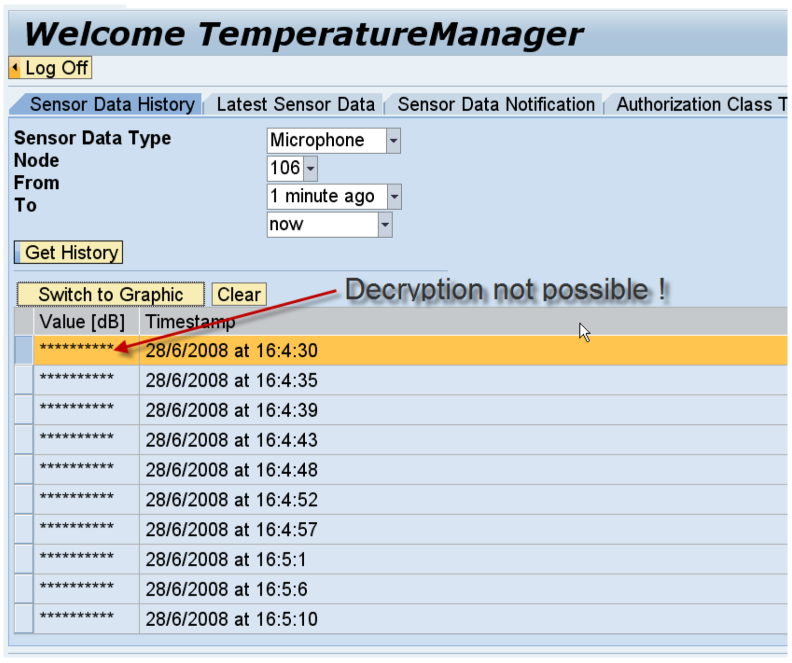

WSN_Controller - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
Secure Integ
Wireless Sens
Login

http://localhost:50100/webdynpro/dispatcher/local/WSN_Controller/WSN_Controller?SAPtestId=111

User TemperatureManager
Password ●●●●●●●●●●

This demonstration has been developed in the scope of a collaboration be

Figure D.13: Logging in as TemperatureManager



Welcome TemperatureManager

Log Off

Sensor Data History | Latest Sensor Data | Sensor Data Notification | Authorization Class T

Sensor Data Type	Microphone
Node	106
From	1 minute ago
To	now

Get History

Switch to Graphic | Clear

Decryption not possible !

Value [dB]	Timestamp
***********	28/6/2008 at 16:4:30
***********	28/6/2008 at 16:4:35
***********	28/6/2008 at 16:4:39
***********	28/6/2008 at 16:4:43
***********	28/6/2008 at 16:4:48
***********	28/6/2008 at 16:4:52
***********	28/6/2008 at 16:4:57
***********	28/6/2008 at 16:5:1
***********	28/6/2008 at 16:5:6
***********	28/6/2008 at 16:5:10

Figure D.14: Decryption of a different Data Type not possible