A secure communication model for the pacemaker

a balance between security mechanisms and emergency access

Ibrahimi, S.

Award date:
2014

A SECURE COMMUNICATION MODEL FOR THE PACEMAKER

A balance between security mechanisms and emergency access

SARAH IBRAHIMI

For the degree of *Master of Science* in *Information Security Technology*

Department of Mathematics and Computer Science
Eindhoven University of Technology
Deloitte Amstelveen

August 2014

Examination Committee:
DR. NICOLA ZANNONE
DR. TANIR OZCELEBI
IR. JEROEN SLOBBE

ABSTRACT

Currently, millions of people worldwide are supported by implantable medical devices (IMDs). These devices can have life-saving functionalities and carry a lot of personal data. In this work, the focus is on a specific IMD: the pacemaker. Despite the storage of sensitive information, pacemakers do not have security or privacy related measures to protect these data. This can have serious consequences since an attacker can easily access the device and change its settings which leads to irregular behavior of the heart and in the worst case cause death. As far as we know, no harmful attacks on patients with pacemakers have happened, but nevertheless we assume this as a serious problem with high consequences. This risk can be dramatically decreased by protecting the communication between a pacemaker and a programmer. However, during an emergency the pacemaker should be directly available for medical treatment, since safety of the patient is the most important.

In this thesis, we address the following research question: *"How can we define a system that provides confidentiality and integrity of sensitive information during the communication between a pacemaker and a legitimate programmer, while ensuring availability of information in case of an emergency?"*

Based on challenges related to this system, we define the requirements that are necessary for an optimal solution for this situation. An optimal solution that fits all requirements is hard to reach. As a result, we develop two solutions for the problem. Our first solution is directly applicable on the current situation and uses an external device to provide security. Although, most requirements are achieved, the use of an external device that the patient always has to carry is undesirable, since they can forget it or it can be stolen or broken.

For this reason, we present a second solution with more requirements, which does not use an external device. We propose a protocol for mutual authentication and secure communication between the pacemaker and the programmer. We prove that this protocol is correct according to a well known protocol verifier ProVerif and we make a cipher analysis to decide what cryptographic cipher will be used for the protocol. To complete both solutions, we provide details about the key management system. For the second solution, we present a new method for key generation that fits the requirements of the environment, is cheap and provides the security that is necessary. This key management system is innovative in the sense that no static keys need to be stored on medical devices, because it is a form of dynamic key generation.

To answer the research question, a new proposal related to emergency access is provided. We developed a way for the pacemaker to function as an emergency detector. By composing a set of parameters that are relevant for emergency cases, we define a system that can deal with emergencies itself and does not need the use of cryptographic keys during an emergency. We discuss the benefits of this system compared to other solutions and we present a proposal how this idea should be completed in future work.

Keywords: pacemaker, security, emergency access, energy consumption.

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank people who supported and advised me along the way.

First, I would like to thank Dr. Nicola Zannone for being my supervisor and being involved in the process from the beginning until the end. Nicola provided me with lots of feedback during my writing process and he helped me with finding a suitable structure for my research. Nicola was always willing to help me back on to the right track when I was exploring the direction of my research. Second I would like to thank Dr. Alexandru Egner who joined the meetings with Nicola in a later stage of the project for sharing his knowledge about medical device security.

I would also like to thank Ir. Jeroen Slobbe and Dr. Trajce Dimkov, who are my supervisors at Deloitte. Their feedback and especially their experiences with research on medical devices were valuable to me.

Next, I want to thank Cardiologist Dr. Ward Jansen, who told me about treatments for patients with pacemakers, the equipment and his experience with emergency situations.

Furthermore, I would like to thank my colleagues at Deloitte from the Cyber Risk Services team who were willing to discuss my research with me and who made a pleasant working environment. I especially want to thank my colleague Piet Kerkhofs for the inspiring discussion sessions, which helped me to think out of the box.

Also, I would like to thank Ir. Marno van der Maas for reading through my thesis and providing me with useful comments as being an independent reader without being involved in my research.

Finally, I would like to thank my family, friends and fellow students for supporting me during the process of writing this thesis and for relating to me with similar experiences.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

BCC Body Coupled Communication

BTG Break-The-Glass

DoS Denial of Service

ECG Electrocardiogram

HIPAA Health Insurance Portability and Accountability Act

HOTP HMAC-Based One-Time Password

ICD Implantable Cardiac Defibrillators

IMD Implantable Medical Device

MAC Message Authentication Code

MICS Medical Implant Communication Service

MITM Man In The Middle

NFC Near Field Communication

OWASP Open Web Application Security Project

PSD Personal Security Device

PV Physiological Value

RF Radio Frequency

RFID Radio Frequency Identification

RSSI Received Signal Strength Indcator

TOTP Time-Based One-Time Password

USRP Universal Software Radio Peripheral

WISP Wireless Identification and Sensing Platform

1 INTRODUCTION

Nowadays, millions of people worldwide are supported by an implantable medical device (IMD) [13]. These IMDs are able to monitor and treat physiological conditions in the body when the body is not able to function normally. Functions of IMDs can be therapeutic or life-saving and are related to various parts of the body. As an example, pacemakers and implantable cardiac defibrillators (ICDs) are used for cardiac disabilities, insulin pumps are used for the therapy for diabetes and deep brain implants provide treatment for patients with Parkinson. Because of these implanted devices, support can be regulated automatically without visiting a doctor.

At least once a year, people carrying an IMD need to visit their doctor for a treatment at the hospital. During this treatment, the status of the device is checked and settings of this device can be adjusted to the functionality of the organs of the patient. These checks and adjustments can be done by a programmer, a device that communicates with an IMD by using radio frequencies. Details of this treatment are logged on the IMD and can be stored on or printed by the programmer.

The number of IMD technologies is increasing together with its specification and functionality. Important personal data is stored on the IMD and the IMD acts as a life-saving solution. Despite this, most IMDs do not have security or privacy related measures. One of the reasons might be that attacks in this field are rare. There are no publicly known cases from people noticing attacks on their IMDs or incidents where can be stated that an attacker was involved. Nevertheless, some events are alarming. In 2007 and 2008, several attacks on epilepsy websites happened [10]. By injecting rapidly flashing images, epilepsy was triggered by visitors of the website. From these type of events, it is reasonable to conclude that there exist malicious parties who are willing to hurt patients via computer-based attacks. Furthermore, the market for second hand medical equipment on eBay is growing [11]. This makes life easier for attackers.

The lack of security mechanisms can have serious consequences. Assume the following scenario, a person with an implanted pacemaker is in a crowded area. In the area, there is an adversary with malicious intent. With the right equipment, he or she can imitate the behavior of a legitimate programmer. With this equipment, an adversary is able to communicate with the pacemaker, by asking for data on the pacemaker or to change its settings. This is already possible from a few meters of distance. In the worst case, it is even possible to stop the stimulation of the pacemaker that can lead to irregulations of the heartbeat which can cause death.

There are several security mechanisms that could prevent this type of situations. By authenticating both the pacemaker and the programmer and by encrypting all the data sent by these two devices, attacks can be prevented. However, current research shows that IMDs do not use any of these security mechanisms and these devices are easy accessible for people with the right equipment [45] [13]. For cardiac IMDs, it is shown that an adversary of an IMD with the right equipment is able to read, interfere and change the data communication [20].

In general, a solution to protect systems for these types of attack is by the use of security protocols with strong cryptographic mechanisms. In theory, it is possible to add these extra layers of security to the IMD, but there are several limitations. The first one is the limited storage and battery capacity. Because of the small size of the device and the fact that the device is implanted in the body, there is only space for small hardware components. Furthermore, batteries cannot easily be replaced. Because of this, computations required for cryptography and storage of long keys should be reduced to a minimum. The second important limitation is that security mechanisms should not block the main functionality of the IMD. In all situations, safety of the patient is a first priority. In case of an emergency, safety of the patient is more important than security of the communication. In this situation, an IMD should be immediately accessible, but only by a legitimate programmer and not by an attacker. For example, when a patient is on vacation and needs to be treated in another hospital during an emergency situation, the device should be directly accessible by every medical staff member who can help. The use of cryptographic keys increase the difficulty to access the device.

Several agencies address the relevance of solutions for this problem. For example, the Government Accountability Office (GAO) states that the manufacturer of defibrillators has the main role in describing the means in which the defibrillator can be accessed [18]. It should define access-control policies related to authorizing access, selecting the basis for restricting access and selecting the access control method. The manufacturer should establish controls for protection against unauthorized wireless access to the pacemaker. Furthermore, the manufacturers should establish emergency access procedures and describe who is authorized to have emergency access and how the emergency mode of the system is reached. Although stated by the GAO, none of the manufacturers comply with these guidelines.

During the last decade, more and more researchers have been trying to find solutions for a balance between safety and security aspects of IMDs. These are related to hardware failures, software errors, radio attacks, malware, vulnerability exploits and side-channel attacks on IMDs. The main focus in this research area is on radio attacks. Zhang et al. [45] signalize three main approaches for solutions for radio attacks: (i) close-range communication to ensure authorized treatment, (ii) the use of cryptography in IMDs and (iii) the use of external devices to ensure security properties without modifying current IMDs.

Several researchers present architectures for IMDs with energy efficient components as Wireless Identification and Sensing Platforms [12] or Smart Implant Security Cores [40] to make implementations of cryptographic mechanisms possible. Solutions as the Cloaker [10], the Shield [17] and the IMDGuard [43] use an external device that the patient has to carry to add security layers to the IMD without making any hardware modifications to the IMD and without increasing the energy consumption of the device enormously.

Although several solutions have been proposed over the years, most of them are incomplete, proven to be vulnerable for certain attacks or need enormous design changes to the IMD. Most of these solutions provide security in regular situations, but in case of an emergency there is open access for everyone to the device. Furthermore, one of the largest disadvantages is that some solutions make use of a proxy, which

the patient always has to carry. This puts the complete responsibility on the patient. Currently, there is no solution available on the market for patients carrying an IMD.

RESEARCH QUESTION

In this thesis, we further analyze the balance between safety and security issues for the IMD and we propose a new system providing the required security requirements while keeping in mind the challenges related to the type and functionality of the device. The focus is on the pacemaker and design decisions is based on the architecture for this type of IMD. In particular, this thesis aims to answer the following research question:

How can we define a system that provides confidentiality and integrity of sensitive information during the communication between a pacemaker and a legitimate programmer, while ensuring availability of information in case of an emergency?

To answer this research question, we divide it in two parts:

1. *What system can provide confidentiality and integrity of sensitive information during communication between a pacemaker and a legitimate programmer?*
2. *How can emergency access be assured in this system?*

Since the design of the pacemaker is proprietary, little information about the details of its architecture is publicly available. This limitation makes it difficult to provide a solution that fits the current architecture of the pacemaker. Reversing the functionality of a pacemaker by simulating its behavior is a large and difficult task by itself and therefore out of the scope of this project. The research question is answered by making use of the information that is publicly available in the literature. Because of this, we answer the first research question by defining two types of systems. The first system is based on the information we know about the architecture of the pacemaker that is currently available. This system is directly applicable on the pacemaker that is already implanted in the body. Since manufacturers do not reveal much about the components of the pacemaker, the system is not able to cover all the requirements. Therefore, we define a second system which covers more requirements, but requires certain components in the architecture. By making realistic assumptions about the architecture of the pacemaker based on modern technologies, a system can be made that solves the lack of security mechanisms in the future. An important aspect of the solution is the key management related to this. We describe a new method to generate and use dynamic keys for this system.

The second subquestion is answered for both systems by presenting two emergency access solutions. The solution for the second system is a new approach without the involvement of cryptographic keys.

This thesis focuses on the pacemaker and its related radio attacks, considering the communication between a pacemaker and a programmer. Hardware failures, software errors, malware and vulnerability exploits and side-channel attacks as described by Zhang et al. [45] are out of the scope of this project. Furthermore, any form of physical security is out of scope of this research.

OUTLINE

This thesis consists of the following chapters:

CHAPTER2 consists of an overview of all the components of the system on the current market with its structure and functionality. This section presents the functional aspects of an IMD.

CHAPTER 3 describes the most common attacks related to pacemaker, together with the challenges this type of devices carry.

CHAPTER 4 consists of a literature study of the most promising existing solutions and techniques that address the same or a comparable problem. Furthermore, emergency access for all these solutions is discussed.

CHAPTER 5 consists of the assumptions and requirements for an optimal solution and shows why an optimal solution is not feasible.

CHAPTER 6 proposes a proxy-based solution for the problem. Assumptions and requirements are introduced, together with the design of the system and the protocols that are used.

CHAPTER 7 presents a shared secret based solution for the pacemaker. It contains the assumptions and requirements of the solution, a new protocol and its security analysis according to a protocol verifier. Furthermore, an analysis of various ciphers is added.

CHAPTER 8 presents key management methods for both systems. This chapter contains details about the most important phases of the lifecycle of cryptographic keys and focuses on the process of key generation.

CHAPTER 9 presents two main emergency solutions for both systems and introduces a new technique for the second system.

CHAPTER 10 gives a conclusion and directions for future work.

2 SETTINGS

This chapter provides an overview of the general architecture of the system together with a description of its main components: the pacemaker, the programmer and the telemonitoring system. We describe the architecture of these devices and explain the functionality of different components.

2.1 SCENARIO DESCRIPTION

In this section, we present two use cases related to different types of treatment. The first use case is about treatment in a general situaton. The second use case describes treatment during an emergency.

Use case 1

In general, a patient carrying a pacemaker visits the hospital for the regular cardiologist once or twice a year to check the status of the pacemaker. During the treatment, a medical staff member uses a programmer from the same manufacturer of the pacemaker to read out the data stored on the pacemaker about its activity in the last period of time. Depending on this data and the current status of the heart, the medical staff member can adapt therapy settings of the pacemaker. When the treatment is finished, the programmer and the pacemaker contain a log file with information about the treatment and changed settings. This information is collected in an archive of the patient. This archive can be stored digitally in the back-end system of the hospital or in printed form in the hospital.

In some cases treatment does not only consist of visits to the hospital, but also involves telemonitoring at home. A telemonitoring system is a system that is placed in the home of a person carrying a pacemaker. The telemonitoring device checks the state of the pacemaker on regular basis. This may be daily, weekly or monthly. After doing the checks, this information is sent in encrypted form over the Internet to the hospital where regular treatments happens [25]. Here, the related cardiologist can look at the results of the telemonitoring and invite the patient for a new treatment session in the hospital if this is needed.

Use case 2

When considering emergencies, three main types of emergency situations for patients with a pacemaker can be distinguished:

- A patient visits the hospital because of abnormal behavior of the heart. The doctor should be able to read out the history of the data recorded by the pacemaker or modify settings.

- A patient visits the hospital because some part of the pacemaker is broken or does not function as it should do. The doctor should be able to read out the settings in the pacemaker, look at its log history and change the settings. In this situation, it might even be necessary to replace the pacemaker by a surgery.
- If the patient requires an operation in case of an emergency or a scan with an MRI and he or she has a pacemaker, the pacemaker must be deactivated before the operation in order to prevent unintentional shocks to the patient.

In case of an emergency, a check is done if the hospital has the correct equipment from the same manufacturer as the pacemaker to communicate with the device. This is checked by a card from the manufacturer that the patient is always carrying in his or her wallet. If the correct equipment is available, treatment is given. Otherwise, the patient is brought to another hospital with the right equipment. In case the pacemaker has to be turned off, this can be done by using a programmer or by placing a magnet on the body of the patient close to the pacemaker. This reduces the activity of the pacemaker. Afterwards, the logs of the treatments can be sent to the hospital where the patient is treated regularly or it can be given to the patient in printed form. This depends on the hospital, since there are no standard agreements worldwide for this.

2.2 THE GENERAL SETTING

For a treatment of a patient with a pacemaker, the setting is as presented in Figure 1 and consists of a pacemaker, a programmer, a telemonitoring system and the back-end system. Note that the programmer and the telemonitoring system are never physically together, since the programmer is in the hospital and the telemonitoring system at home of the patient. During this treatment, the programmer and the pacemaker are constantly communicating with each other.

When the treatment is finished, log files related to the treatment go to the back-end system or it can be stored in printed form in the hospital. In the first case, the information can be sent to the local system, it can be saved on USB sticks or floppy disks or it can be sent over the Internet. In case of a physical back office located in the hospital, a printer can be directly connected to the progammer and the information in log files can be printed. During regular checks at home, the telemonitoring system retrieves data from the pacemaker and sends this to the back-end system from the hospital.

In the remainder of the chapter we present the main components of the architecture.

2.3 PACEMAKER

A pacemaker is a particular type of IMD. This section first describes the characterizations and the components of an IMD in general. Then, pacemaker specific details are discussed.

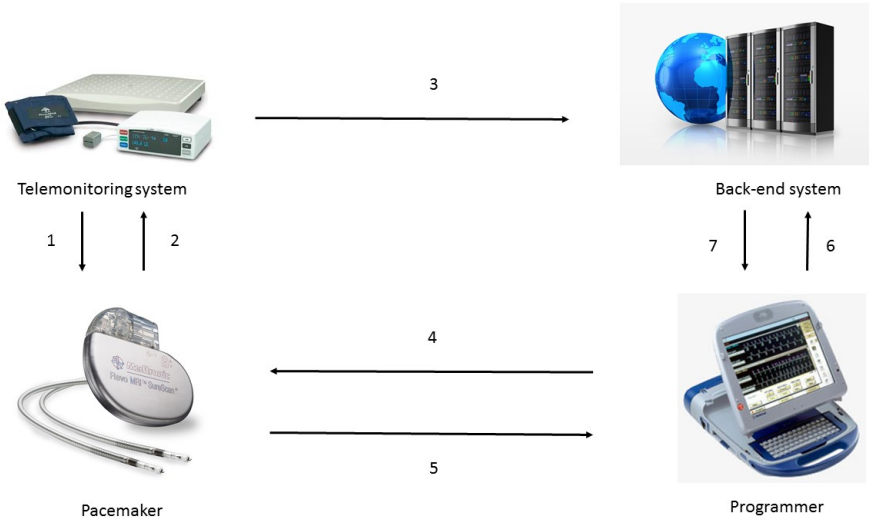

Figure 1: The general architecture of the current system.

Legend
1: The telemonitoring system requests the pacemaker for data and may be able to configure this device.
2: The pacemaker sends its data to the telemonitoring system.
3: The telemonitoring forwards the data from the pacemaker to the back-end system of the hospital.
4: The programmer configures the pacemaker and requests data stored on the pacemaker.
5: The pacemaker sends the data to the programmer.
6: The programmer forwards the data to the back-end system of the hospital.
7: The back-end system may send data to the programmer about previous treatment sessions.

2.3.1 *IMDs*

The European Council Directive 90/385/EEC defines an active implantable medical device as "any active medical device which is intended to be totally or partially introduced surgically or medically into the human body or by medical intervention into a natural orifice, and which is intended to remain after the procedure." [8] Its function is described in the definition of a medical device which means "any instrument, apparatus, appliance, material or other article, whether used alone or in combination, together with any accessories or software for its proper functioning, intended by the manufacturer to be used for human beings in the:

- diagnosis, prevention, monitoring, treatment or alleviation of disease or injury,
- investigation, replacement or modification of the anatomy or of a physiological process,
- control of conception

and which does not achieve its principal intended action by pharmacological, chemical, immunological or metabolic means, but which may be assisted in its function by such means." [8]
As a consequence of implantation by a surgery, the communication between the IMD and an external device is wireless. This communication is needed in the first place for the configuration of the IMD, for example to change the therapy for the IMD. Furthermore, communication with an external device is needed to diagnose the IMD, extract history information about treatments and incidents and to update the IMD firmware.

Ellouze et al. [13] divide IMDs into two classes. The first class contains medical devices that are fully implanted in the body of the patient as pacemakers. The second class integrates medical devices that are partially implanted such as the insulin pump system. Part of the components are implanted in the body, other parts are external to the body. In both classes, the IMDs could be completely autonomous or require human intervention to adjust the therapy or to approve some undertaken reactions.

Furthermore, they describe the functionality of an IMD [13]. Traditionally, the IMD and the programmer communicate using the inductive telemetry. This is based on inductive coupling between the IMD coils and the programmer coils. This type of communication has several limitations including a short communication range and a limited data rate to exchange, which is less than 50 kbps. Modern IMDs communicate wirelessly with programmers using the radio frequency (RF) telemetry. It happens through the 402-405 MHz Medical Implant Communication Service (MICS) band, established in 1999 by the U.S. Federal Communications Commission [13]. Because of the use of MICS, larger communication rangers were possible and these had a higher data rate as a consequence.

IMDs have several computational and storage resource constraints, because of their batteries with limited lifespan. They execute energy-aware algorithms to remain operational as long as possible, since replacing a battery requires a surgery and is undesirable. In most cases, the whole IMD is replaced instead of only replacing the battery. The lifetime of the IMD highly depends on the patient's health. The more the patient shows abnormal physiological conditions over time, the more the IMD will react and apply therapy, which results in a high energy consumption.

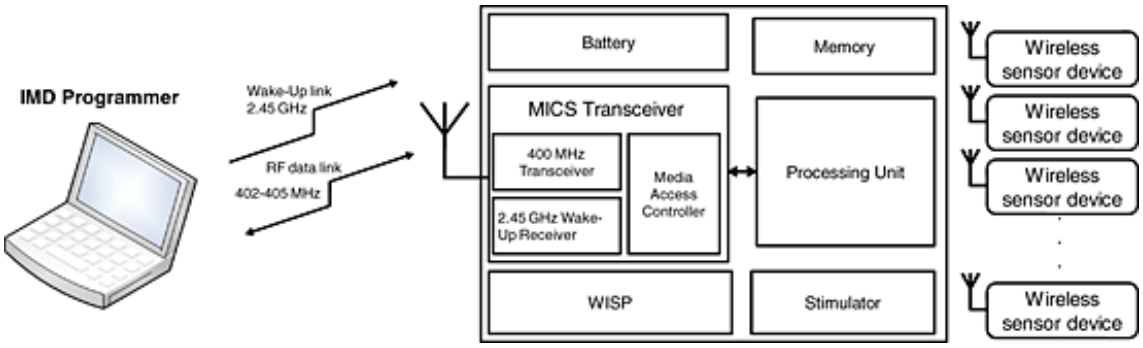

Figure 2: The architecture of the overal system of an IMD [13].

2.3.2 *IMD architecture*

The generic architecture of an IMD is presented in Figure 2. Below we provide an overview of its components [13].

- Sensor devices: a set of sensors is implanted in the human body to measure physiological parameters to evaluate the health status of the patient and to determine the therapy that has to be delivered. The processing unit will process the performed measurements.
- Battery: this part is responsible for powering the IMD components. Depending on the type of IMD, the battery is non-rechargeable or powered inductively. In general, a power management approach is implemented to optimize power consumption to maximize the battery lifespan or time between recharges.
- Memory: this has the form of a programmable memory, e.g. electrically erasable programmable read-only memory or flash memory. The memory stores collected measurements, history of detected health abnormalities and therapy settings.
- Processing unit: this unit executes software built in three different layers. The first layer is the communication code, which generates the uplink bits and detects the down-link bits. This level manages the communication between the IMD and the programmer. The second layer manages the state of the RF transceiver and the power consumption. The third layer implements the IMD functions as sensing, data processing and stimulation.
- Stimulator: this unit takes care of therapy delivery based on sensed information. Stimulation parameters are set by the doctor during a treatment. This determines the therapy to be delivered to the patient.
- MICS transceiver: this enables wireless communication with the IMD programmer. It mainly consists of three components: a 400 MHz transceiver, a 2.45 GHz wake-up receiver and a media access controller. As described before, it operates under the 402-405 MHz MICS band.
- Wireless identification and sensing platform (WISP): this is a powerless device that receives its energy from the incoming RF signal and can be read by ultra-high-frequency radio frequency identification (RFID) readers. It contains a fully

programmable 16 bit microcontroller that is able to execute cryptographic algorithms using the harvested energy. Nowadays, only part of the IMDs on the market carry this component.

- Implantable medical device programmer: this component in the overall architecture of the system is used to communicate with IMDs, to adjust therapy, diagnose the IMD or change its settings.

2.3.3 *Pacemaker architecture*

A pacemaker is an IMD that delivers a controlled, rhythmic electric stimulus to the heart muscle in order to maintain an effective cardiac rhythm for long periods of time [19]. Implanting a permanent pacemaker and selection of the appropriate mode of operation are based on the type of cardiac disease as a failure of impulse formation (sinus syndrome) and/or impulse conduction (AV-block). The connection between the heart and the implanted pulse generator is provided by an implantable electrode catheter, a lead. This is connected to the heart muscle and serves as a stimulator and a sensor device at the same time. There are different types of pacemakers [21]:

- A Single Chamber Pacemaker: this type has one lead that is placed in the right upper chamber (atrium) or the lower chamber (ventricle).
- A Dual Chamber Pacemaker: this type has two leads, one in the atrium and one in the ventricle.
- A Biventricular Pacemake: this type has three leads, one in the right atrium, one in the right ventricle and a third in the left ventricle.
- A Rate Responsive Pacemaker: this type of pacemaker adjusts the heart rate to a patient's level of activity. It paces faster when a patient is exercising and slower when a patient is resting.

According to Ellouze et al. [12], a regular cardiac IMD consists at least of four components that are visible in Figure 2:

- A battery to power the other components.
- A memory to store collected measurements, the history of abnormalities detection and therapy settings.
- A stimulator in charge of delivering therapy that collects electrocardiogram signals.
- A 400 MHz transceiver to enable the wireless communication with the programmer. This is part of a microcontroller which has a functionality that is not further dicussed in the literature.

A pacemaker has an average lifetime of 5 to 10 years, depending on its type and the frequency that it needs to regulate the heart. In case the battery is almost empty, the whole device is replaced in most of the cases. The size of the pacemaker is approximately 5 by 5 cm and 1 cm thick. At least once a year, the functionality and the estimated lifetime of the implanted pacemaker is checked at the hospital.

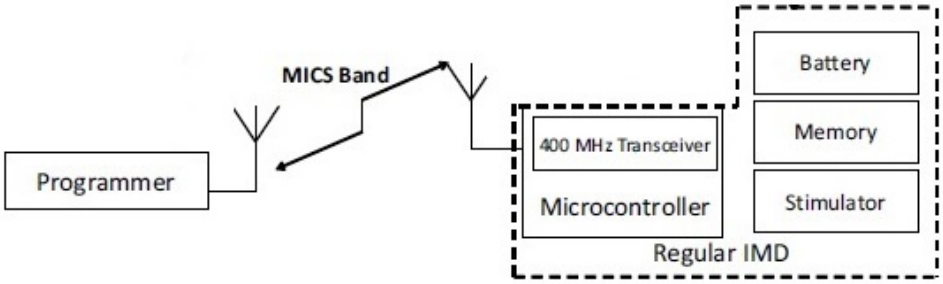

Figure 3: The architecture of a cardiac IMD based on [12].

2.4 PROGRAMMER

A programmer is a device that communicates with the pacemaker with radio frequencies in the MICS band and functions as a personal computer by running a user-friendly operating system. Implanted pacemakers have a micro-antenna which is able to communicate with an external transmitter. Some programmers have a receiver or wand attached by a wire which is positioned on the body's to receive the telemetry signal from the pacemaker. The programmer is able to read information stored in the logs of the pacemaker, to read information about its current status and to modify its settings.

During a visit to the hospital, the programmer is used by a doctor or a programmer technician to change the therapy for the pacemaker and to see if any unexpected events happened. This is only possible if both the pacemaker and the programmer are from the same manufacturer.

2.5 TELEMONITORING SYSTEM

The telemonitoring system is a device that either communicates with the pacemaker on the Industrial, Scientific and Medical (ISM) band from 902-928 MHz or on the MICS band in the same way as the programmer does [25]. This device is placed at the home of the patient to assure patient safety, to identify abnormal behavior of the heart and to optimize the patient's quality of life. Furthermore, it identifies and corrects abnormal device function, identify the end of life of the battery and monitor various alerts. Telemonitoring may be divided in two types of monitoring: remote follow-up and remote monitoring [25]. During remote follow-up device data is transferred from the patient to the physician's office or to the central data processing center in a predetermined scheduled time using the wand or a transmitter, the similar type of object that is attached to a programmer. Remote monitoring consists of unscheduled transmission of the predefined alerts. The data from the device is transmitted and the cardiologist is notified, according to predefined settings. Only part of the new telemetry systems is able to execute bidirectional telemetry. Here remote interrogation and remote programming of the pacemaker can be done. The collected data is sent in encrypted form by a standard analog phone line or by wireless GSM connection [25].

3 ATTACKER MODEL AND CHALLENGES

This chapter gives an overview of the vulnerabilities, threat agents and attacks that are related to IMDs and are applicable to the pacemaker. Furthermore, challenges related to emergency access and energy consumption are presented.

3.1 LIST OF CONCEPTS

Based on OWASP [30, 31, 32], we define a vulnerability, a threat agent and an attack as follows:

- *Vulnerability*: a vulnerability is a hole or weakness in a system that allows an attacker to cause harm to the stakeholders of the system. It can be a design flaw or an implementation bug.
- *Threat agent*: a threat agent is someone who is able to take advantage of a vulnerability in a system and can cause a negative impact on the system and its stakeholders.
- *Attack*: an attack is a sequence of actions a threat agent performs to exploit a vulnerability in a system.

3.2 ATTACKER CAPABILITY MODEL

IBM [2] distinguishes three classes of attackers: clever outsiders, knowledgeable insiders and funded organizations. Clever outsiders are often intelligent, but may have insufficient resources and knowledge of the system. They may have access to only moderately sophisticated equipment and they try to take advantage of an existing weakness in the system. Knowledgeable insiders are able to operate on their own, have specialized education and experience and understand parts of the system. We assume that they have sophisticated tools and instruments for their analysis. Funded organizations are able to assemble teams of specialists with related skills by funding resources. They are able to do in-depth analyses of a system, to design sophisticated attacks and use advanced analysis tools[2].

In this research, we consider knowledgeable insiders as possible attackers of the pacemaker. They are able to operate on their own, have specialized education and experience and understand parts of the functionality of the pacemaker and the programmer. They have knowledge about the communication stream between the programmer and the pacemaker, for one or more types of pacemakers. They can buy a second hand programmer, for example from eBay or they can steal one from a hospital. Although pacemaker programmers are expensive when offered by the manufacturer, several types are available on eBay [11] within the price range from $65 to $1750. They can also use their self made equipment consisting of a Universal Software Radio Peripheral (USRP) [20]. A USRP is commercially available by Ettus Research

[14] and varies in price range between $675 and $4.800. In both cases, an attacker is able to communicate with a pacemaker according to a certain communication protocol. Since every manufacturer develops his or her own communication protocol, an attacker will probably not be able to communicate with all types of pacemakers.

3.3 VULNERABILITIES

The system architecture of the pacemaker and its environment presented in Chapter 2 are subject to several vulnerabilities, related to data storage and communication. Considering the pacemaker and the programmer, we can distinguish three types of main vulnerabilities:

A *The programmer does not require any form of authentication before usage.*
By turning on a programmer, it can be accessed and controlled by every person who knows how this programmer works. The only protection provided in current settings is a form of physical security. Programmer are usually stored in a room for example with a key that is only carried by authorized medical staff and a security officer. This type of physical security depends on the design of hospital buildings and the specific policies for each hospital. This is out of the scope of this research.

B *The pacemaker does not verify the legitimacy of the programmer.*
By using a programmer, a communication link with the pacemaker can be established without using a PIN or any other form of authentication. The pacemaker only responds to radio frequencies in the range 402-405 MHz without checking if the device requesting is a programmer and even more specific, a valid programmer. In the logs stored on the pacemaker, there is an option to store who accessed the device, but the use of this option depends on the rules set up by the institution.

C *The communication between the programmer and the pacemaker is not encrypted.*
The data between the programmer and the pacemaker is sent in plaintext. Without using any form of encryption, privacy sensitive data is sent in the clear and can become publicly available.

3.4 THREATS

Threats can be divided in three categories related to the security aspects: confidentiality, integrity and availability. Regarding to confidentiality, we can distinguish two types of threat agents:

1. Someone who wants to read out data about abnormal physical activities, historic treatment or the current status from the pacemaker or programmer. This happens by reading the logs.

2. Someone who wants to intercept data on the channel during treatment.

Integrity-related threats are based on the modification of data stored on a device or data sent during communication. Three types of threat agents can be distinguished:

3. Someone who wants to change the data in logs stored on the pacemaker or the programmer
4. Someone who wants to change the current status of the pacemaker
5. Someone who want to change the data on the channel during treatment

Threats related to availability are important, since the system needs to be available at all times. The two most important availability-related threat agents are:

6. Someone who wants to block the communication channel between the pacemaker and the programmer during treatment
7. Someone who wants to send unnecessary requests to the pacemaker to drain the battery

3.5 ATTACKS

Combining both Sections 3.2 and 3.3, leads to different types of attacks. These are presented in Table 1, where V stands for the vulnerabilities listed in Section 3.2 and T for the threat agents listed in Section 3.3.

V \ T	1	2	3, 4, 5	6	7
A	Man in the Middle				
B	Man in the Middle		Man in the Middle	DoS	DoS
C		Eavesdropping	Man in the Middle		

Table 1: Possible attacks and their appearances.

A description of each of these attacks is now given.

EAVESDROPPING ATTACK

This type of passive attack is related to confidentiality. Using a USRP, an adversary can eavesdrop the exchanged messages both between the programmer and the pacemaker as between pacemaker components as wireless sensor devices and the MICS transceiver [20]. In addition, other equipment such as an oscilloscope, a software radio or directional antennas can be used for this. Since the messages in this communication are not encrypted, the adversary can analyze them and extract sensitive exchanged data. This can include the pacemaker identification information, physiological data related to the patient's health, therapy specification and the history of given treatments.

MAN IN THE MIDDLE (MITM) ATTACK

In this wide range of active attacks related to integrity and confidentiality, an attacker intercepts and modifies the communication between the pacemaker and

the programmer. Related attacks can be executed using a USRP or an unauthorized programmer. An adversary is able to establish a connection him or herself between a pacemaker and his or her own equipment, if he or she knows the communication protocol. After obtaining unauthorized access, an adversary could execute unauthorized commands to stop the medical treatment by the pacemaker, modify the pacemaker's firmware, drift the pacemaker's clock, delete data stored on the pacemaker or disable the pacemaker. To make attacks difficult to detect, attackers could alter, hide or delete data that reveals attacks. These data consist of the log generated by the pacemaker and the pacemaker's clock that timestamps different logged information. By changing these timestamps, malicious activity can be seen as part of a regular treatment.

A replay attack is a type of a MITM related to integrity. In this attack, a valid data transmission is maliciously or fraudulently repeated or delayed. By delaying messages and changing their order, treatment to the patient can be stopped. Certain request messages from a valid programmer can be overwritten by the equipment from the attacker, when attacking at the correct time.

DOS ATTACK

This type of active attack is related to the availability of the pacemaker. Various types of denial-of-service (DoS) attacks can be performed to force the pacemaker to respond to every request. These DoS attacks, as repetitive sending of request messages to connect to the pacemaker, lead to resource depletion and could drain the pacemaker's battery. Examples of DoS attacks are jamming attacks and buffer overflow attacks.

By jamming the signal exchanged between the pacemaker and the programmer, an attacker is able to prevent the pacemaker from receiving the required configuration or delivering the logged events. In case where the pacemaker is required to communicate with wireless sensors, a jamming attack would prevent the pacemaker from collecting sensitive events from sensors and it would prevent the pacemaker from delivering the adequate therapy or reacting appropriately.

Halperin et al. [20] show that these attacks are realistic for cardiac IMDs. They test the security of an implantable cardiac defibrillator (ICD) that uses the 175 kHz frequency range for communication. Since the ICD shares most of the functionality of the pacemaker, it is reasonable to assume that these attack scenarios are applicable to the pacemaker. Several weaknesses of the system are presented. First, the ICD discloses sensitive information in the clear (unencrypted). Second, some possible attacks are able to change the operation of and information contained in the ICD. These attacks fall into the category replay attacks. With these attacks, it was possible to disclose patient and cardiac data, to set the ICD's clock and to change therapies [20]. Third, an ICD can be forced to communicate with an unauthenticated device, which can drain the battery and be a potential DoS attack. To execute these attacks, they used the equipment according to the setting in Figure 4. This consists of two hardware tools to intercept the RF signals emitted by the ICD and the programmer: a recording oscilloscope and a USRP.

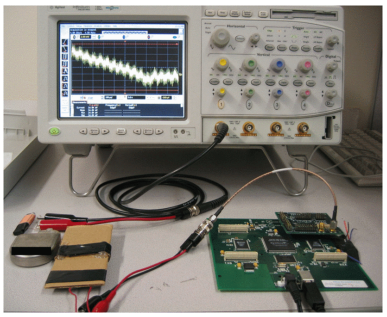

Figure 4: The equipment setting from the research setting [20].

3.6 CHALLENGES FOR ENSURING SECURING THE PACEMAKER

Adding security layers to the communication between the pacemaker and an external device brings several challenges. These are related to emergency access and energy consumption.

3.6.1 *Emergency access*

In case of one of the emergency situations as described in Section 2.1, a doctor needs to communicate with the pacemaker using a programmer. With the current pacemaker settings, everyone with a programmer from the same manufacturer as the pacemaker is able to communicate with the pacemaker in an emergency situation. This is undesirable, since attackers can access the pacemaker in this case. In contrast, if we encrypt all the data in the communication between a programmer and a pacemaker and let programmers authenticate to the pacemaker at the start of each treatment, unauthorized access would not be possible. However, it is difficult to determine who is authorized and how to give right permissions. During general treatments, only the cardiologist treating the patient needs authorization. However, in case of an emergency, a medical staff member from another hospital in the same country or even abroad needs to get access to the pacemaker to treat the patient. It is undesirable that getting access to the device takes much time.

As a solution, various researchers present fail open access for emergency situations [43]. Here, the pacemaker or another device distinguishes between a regular situation and an emergency situation and provides access for everyone in case of an emergency. However, if an attacker is in proximity of the patient in this situation, he or she is able to access the pacemaker. Burleson et al. [6] claim that a medical professional may need to reprogram or disable an IMD to effectively treat the patient in case of an emergency. Encryption or other authentication mechanisms could make measures during emergency impossible if the patient is unconscious or if the programmer does not have the required cryptographic key to access the IMD.

3.6.2 *Energy consumption*

Rostami et al. [38] recognize three ongoing trends related to energy challenges for IMDs. First, IMDs are becoming complex and need more power for new therapeutic and monitoring functionality. Second, the IMDs are collecting more data from new

sensors that monitor the patient's health, which requires power intensive wireless communication. Third, strong security protocols for authentication require the use of cryptography and require a lot of computation power.

On the one side it is clear that various security aspects of the IMDs should be improved to protect sensitive data of patients and to provide them with a safe environment. On the other side, improvements in battery lifetime and computation power needs to be used in improving the functionality of the device, its lifetime and to decrease its size.

Although energy consumption is an important aspect of IMDs, the focus in this research will be more on emergency access. Currently new techniques are coming up and might solve the energy consumption in the very near future. For example, inductive charging offers the possibility of relaxing the energy constraints and avoiding the complications and costs associated with replacing batteries for medical implants [44]. This form of wireless charging is still in a research phase, but can be the future for IMDs. Since emergency access depends less on new technological developments for components in the IMD and is more a challenge for the design of the overall system, we will focus on emergency access in the rest of the research.

4

RELATED WORK

This chapter presents the main trends in the literature related to the security of the IMD and its communication with the programmer. Both solutions for close range communication as for regular communications are presented. For the last category, solutions without a proxy and with a proxy are discussed. Furthermore, solutions related to emergency access are presented.

4.1 CLOSE-RANGE COMMUNICATION

To minimize the number of attacks on the radio communication between the IMD and the programmer, the communication range between the IMD and the programmer can be limited. Several types of solutions have been proposed in this area by Zhang et al. [45]. Radio-frequency identification (RFID) and near-field communication (NFC) are presented as possible solutions, but there exist attacks for these solutions. An adversary with equipment as a strong transmitter and a high-gain antenna can attack a wireless channel only for RFID-based communication. NFC might be more effective, since it has a working distance up to 20cm. However, there is no guarantee that an attacker with high-gain antenna cannot read the signal from outside this range. A common used technique, which is also proposed for RFID is distance bounding. Distance-bounding consists of a single-bit challenge and a rapid single-bit response [5]. Based on a series of rapid bit exchanges, a delay time for response is computed by the receiver and can lead to an upper-bound on the distance. Applied to IMDs, this is part of a broader technology called body-coupled communication (BCC) [45]. This uses the human body as a transmission medium and its communication range is limited to the proximity of the human body. It works at low frequencies from 10 kHz to 10 MHz and can only achieve very low data rates. Also measures can be taken to enforce close-range communication in addition to short-range physical communication layers. However, limiting the communication range is only effective when radio attacks are launched from beyond a certain distance.

Rasmussen et al. [36] state that access control solutions based on close-range communication have an advantage of being simple and intuitive, but they do not provide any firm guarantee about the range of communication. If an attacker has a strong transmitter and a high-gain antenna, he or she will be able to communicate with the IMD from far outside the intended range. Existing solutions are based on magnetic switches, but they do not require any authentication to unlock access. Rasmussen et al. propose a new mechanism based on ultrasonic distance-bounding which enables an IMD to grant access to its resources only to devices in its close proximity. Messages are cryptographically tied to the distance bounds measured by the IMD, to the device that requests access.

The security range defined in the IMD is set to a distance less than 10 cm. In this scheme, access control is based on device pairing. During this process, a pairing protocol will run and will generate a shared key. The reader acts as a prover, who

has to prove its proximity to start the data transfer. The IMD is the verifier, that must verify the distance to the prover before accepting the connection.

The system distinguishes two modes of operation: a normal mode and an emergency mode. In the normal mode of operation, the credential token that shares a secret key, that the patient carries, is used. When a doctor needs to access the IMD, he or she gets credentials from the patient and provides it to the reader. The proximity aware device pairing protocol runs and afterwards each party has the assurance that the other party is in the security range and has derived a key that is used to secure future communication. The emergency mode of operation is discussed in Section 4.3.

Although MITM attacks are difficult to be executed because of the ultrasonic distance bounding protocol, where a secure pairing is possible only if the reader is approximately 3cm away from the IMD, this solution has some disadvantages according to Ellouze et al. [13]. The device pairing protocol requires high energy consumption to be executed, because of the computational complexity of the Diffie-Hellmann key agreement protocol. This is undesirable for an IMD. Also, since the solution does not take the update of parameters into account, they can become vulnerable to cryptanalysis attacks when they are used extensively.

4.2 PROXY VS NON-PROXY COMMUNICATION

Several types of solutions for securing the communication between a programmer and an IMD are presented in the literature. These solutions can be divided in two main categories: solutions without a proxy and solutions with a proxy.

4.2.1 *Solutions without a proxy*

This type of solutions relies on the implementation of cryptographic algorithms to protect the communication between a programmer and an IMD. For this purpose, energy-efficient components with computation power are added. Furthermore, the protocols used for these systems need to be as less energy consuming as possible. Examples of these approaches are using block ciphers, radio frequency energy harvesting and a smart implant security core. Solutions with these approaches will be discussed in the following subsections.

4.2.1.1 *Block cipher based security*

Beck et al. [4] present a block cipher based security protocol with two modes. A stream mode that aims to minimize the radio duty cycle while maintaining basic security and a session mode that provides strong security for highly sensitive information and a role-based user authorization scheme. The stream mode is used for the transmission of very short messages and uses the output feedback mode to obtain a scalable stream cipher that enables a strict duty cycling of the IMD's radio for maximum energy efficiency. The session mode uses cipher-block chaining in combination with a challenge response mechanism. This mode is useful when the IMD connects to an external base station. The system provides role-based authorization by assigning a specific private key to each individual user group. This key is linked to a set of rules that regulate the permitted and prohibited operations of the user.

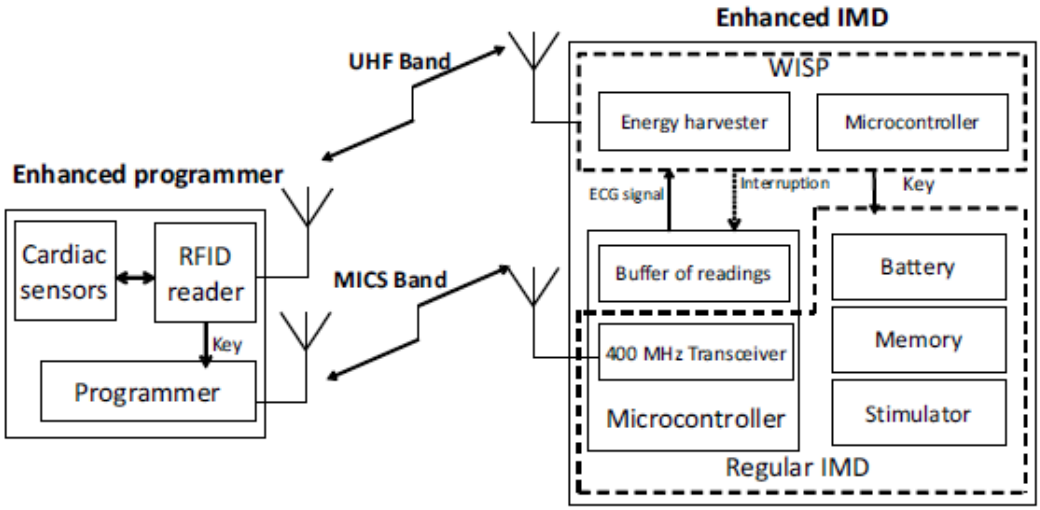

Figure 5: A new architecture for cardiac IMDs [12].

The scheme is designed for Artificial Accomodation System. This is a micro-mechatronic implant meant to replace the natural eye lens in case of a cataract. This IMD should have a duration of 30 years which is very long. As a consequence, the designers should be aware of threats in the future, which might not be known nowadays. Within 30 years, the keysize might be too small because of the computational power of new computers. Beck et al. present the solution as meeting all the security and privacy requirements related to confidentiality, integrity and availability, but they do not give any proof of this, neither do they present implementation details for the specific device. In addition, they do not discuss a solution in case of emergencies.

4.2.1.2 *Radio frequency energy harvesting*

Ellouze et al. [12] propose a solution based on new components of the architecture of cardiac IMDs. This would allow the system to perform radio frequency energy harvesting to enforce secure key generation and authentication. They make use of an enhanced Wireless Identification and Sensing Platform (WISP) which is able to execute complex functions using the harvested energy. The IMD programmer is equipped with an RFID reader to communicate with the WISP and a cardiac sensor to capture the patient ECG. A powerless mutual-authentication protocol is used to allow secure access both in regular situations as in emergency situations. To perform authentication, a biometric key is simultaneously computed by the WISP, embedded in the IMD and the programmer. The key is extracted from the ECG signal and is used to derive master and session keys.

A buffer of readings, a WISP and an enhanced programmer with an RFID reader and a set of cardiac sensors are added to the current architecture of the cardiac IMD. The new architecture is presented in Figure 5. The RFID reader is used to interact with the WISP to execute the powerless mutual authentication protocol. The first time the programmer tries to access the IMD, there are no preshared credentials yet. Secure mutual authentication is achieved when the IMD and programmer share a biometric key, which is dynamically and securely generated during access starting

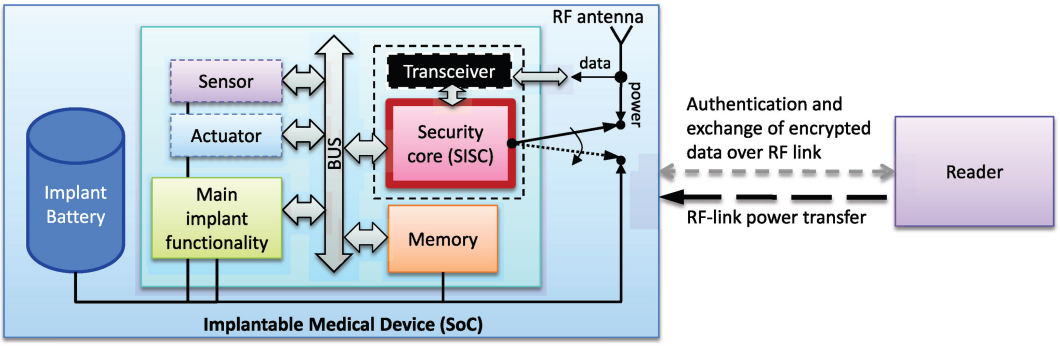

Figure 6: A new architecture for IMDs [40].

from the ECG. The regular mode is used when both the IMD and the programmer share the same credentials in a validity period.

After mutual authentication, the devices communicate using a session key, computed in the authentication protocol. Each message sent by the programmer contains a cookie to protect against DoS attacks. By using nonces and a sequence number, the freshness of the message can be determined and the IMD can react in different ways on the type of freshness to improve the life of the battery.

Ellouze et al. claim that their solution is resilient to battery draining attacks, because of their powerless mutual authentication protocol and robustness against desynchronization attacks and the use of sequence numbers and cookies. Furthermore, their solution prevents against replay attacks, is resilient to brute force attacks and guarantees perfect forward secrecy, which ensures that the compromise of one master key will not allow an adversary to predict the subsequent master keys. It is only able to deduce session keys derived from the compromised master key. Since the architecture of the IMD and the programmer needs to be drastically changed and needs to be tested thoroughly, it is not realistic that this type of IMD will be on the market within a reasonable amount of time.

4.2.1.3 *A smart implant security core*

Strydis et al. [40] propose an implant system architecture for an IMD where security and the main-implant functionality are made decoupled by running the tasks on two separate cores. For the wireless communication, a smart implant security core (SISC) is used, which runs an energy efficient security protocol. This core is powered by RF harvested energy until it performs external-reader authentication to provide a defense mechanism against battery DoS attacks and other attacks. The proposed solution consists of a new system architecture, a secure communication protocol for shielding IMDs and a security processor. Strydis et al. assume jamming attacks never occur. Figure 6 presents the architecture of the solution.

For the symmetric encryption scheme, a 64 bit lightweight block cipher MISTY1 is used. This cipher is suitable for implants since it uses low power consumption, low energy cost and a high encryption speed. The system is protected against replay attacks, MITM attacks, eavesdropping attacks and DoS attacks. This solution needs a drastic change of the architecture of the pacemaker and switching to the emergency

with more access rights is possible in regular situations as explained in Section 4.3.2. Therefore, this solution is not ready for immediate use.

4.2.2 *Solutions with a proxy*

A technique often presented to preserve battery IMD power is to use a trusted external device to verify incoming requests. Because the burden of computation is offloaded to the external device, this approach can protect the IMD against battery-draining attacks.

4.2.2.1 *Communication Cloakers*

Denning et al. [10] present the so called Communication Cloakers as a solution. A Cloaker is an externally worn device, like a computational Medical Alert bracelet. The general idea of the Cloaker approach is to provide security when the patient is wearing the Cloaker and provide fail-open access to all external programmers when the patient is not wearing a Cloaker. In this model, the patient has to wear the Cloaker during his or her everyday life. During regular clinic visits, the Cloaker will only allow pre-specified, authorized commercial programmers to interact with the IMD. The Cloaker acts as a third party mediator in the IMD's communication with external programmers.

The communication between the IMD and the Cloaker would be encrypted by using symmetric-only cryptography. This to prevent replay and reordering attacks. Then the Cloaker can proxy the communications between the programmer and the IMD or the Cloaker can hand-off a lightweight access credential to the programmer. Since the Cloakers have replaceable batteries and greater computational power than the IMD, the Cloaker can take the cryptographic operations on it and its public keys. One of the difficulties is how to detect the Cloaker if it does not have a fixed format. A solution for it is to incorporate a pulse-sensing unit in the Cloaker and define it as being present. There are different approaches for the IMD to detect the Cloaker's presence. These can be categorized in a stateless approach and a stateful approach. In a stateless approach, the IMD queries the Cloaker when it detects an external communication request. In a stateful approach, the IMD keeps an internal record of the Cloaker's presence and can periodically update this record, based on the presence or absence of successful keep-alive messages. Both the IMD and the Cloaker can initiate the keep-alive messages. The use of a Cloaker as a proxy seems promising and is applied in several solutions that will be discussed in the rest of this subsection. Most of these solutions require no or minimal changes to the architecture of the IMD, which makes them almost directly applicable. Unfortunately, several of these solutions are shown to be vulnerable to attacks.

4.2.2.2 *A Personal Security Device*

Pournaghshband et al. [34] present a Personal Security Device (PSD). This is a portable device improving security for mobile medical systems. Its use requires no changes to the medical device or its monitoring software and offers protection for certain types of attacks. The PSD should be aware of the suite of wireless mobile medical devices used by the owner. Furthermore, it has built-in knowledge of their security

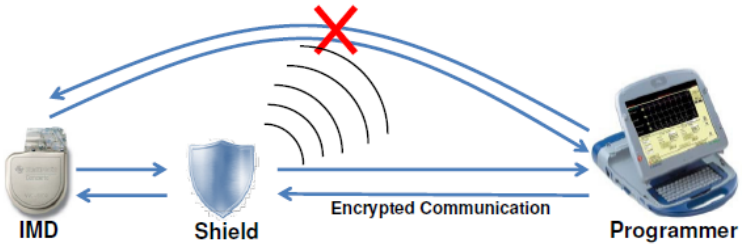

Figure 7: The Shield Architecture [17].

properties and vulnerabilities. The PSD augments the security of the owner's devices by adding authentication and encryption to data streams. When a medical device communicates with a hospital monitoring system, a PSD can be used as an overlay between the two components. Here, the link between the PSD and the hospital monitoring system can be secured, but the link from the medical device to the PSD remains unsecured. Furthermore, the PSD cannot prevent the device from pairing with untrusted systems. In this case, the PSD is able to listen to the signals sent by the device. It is unclear what the PSD does with this information. Although the solution is presented for Bluetooth communication, the idea can be extended to medical devices of other radio technologies, by equipping the PSD with that particular radio technology capability to connect to the device.

Resulting from their own security analysis, Pournaghshband et al. state that the PSD does not protect against Bluetooth jamming between the PSD and the involved system, but does detect this. The solution is also vulnerable to MITM attacks on the communication line between the PSD and the device. They state that additional security mechanisms can prevent this, but this requires altering or rebuilding the device.

4.2.2.3 *A Medical Security Monitor*

Zhang et al. [44] propose a medical security monitor MedMon, that snoops on the radio-frequency wireless communications to or from medical devices. It uses multi-layered anomaly detection to identify transactions that are potentially malicious. MedMon takes response actions ranging from passive to active. It can be applied to existing medical devices without modifying the hardware or software.

MedMon addresses the loss of integrity, but not the loss of privacy, since the attacker may passively listen to transmissions to or from the medical device. Therefore, this solution does not protect against eavesdropping attacks. Furthermore, MedMon does not address the attacks on availability, since attackers may jam the wireless channel rendering any communication impossible or intentionally send invalid packets to drain the device's battery life. Also, this solution does not consider emergency situations.

4.2.2.4 *The Shield*

Gollakota et al. [17] present a design in which an external device, called the shield, is placed between the IMD and the other party (e.g. a programmer). The architecture with the shield is shown in Figure 7. The shield can be worn on the body near

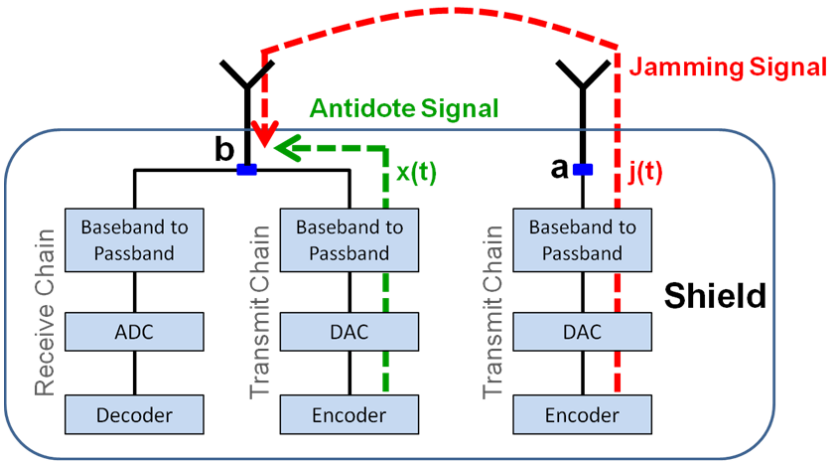

Figure 8: A full duplex radio [17].

an implanted device. It uses a physical-layer mechanism for securing its communication with the IMD and a standard cryptographic channel to communicate with other authorized endpoints. To provide confidentiality for the IMD's transmissions, the shield continuously listens for those transmissions and jams them so that they cannot be decoded by eavesdroppers. To protect the IMD against commands from unauthorized endpoints, the shield listens for unauthorized transmissions addressing the IMD and jams them. A full-duplex radio (Figure 8) is built together with a jamming antenna and a receive antenna, as a small wearable device. This system provides both confidentiality for IMD's transmissions and protects IMDs against commands from unauthorized parties, all without modifying the IMD. The shield works as a relay between the IMD and the programmer. It receives and jams IMD messages at the same time, so others cannot decode them. It encrypts the IMD message and sends it to the legitimate programmer. All commands from the programmer should be encrypted and sent to the shield first, then the shield sends legitimate commands to the IMD. In this situation, the IMDs do not need to change in their design, but the programmers. Because of the encryption of the communication between the programmer and the shield and since the messages from the IMD are jammed, confidentiality of the IMD messages is achieved. However, for commands from the programmer to the IMD, confidentiality is not achieved. Moreover, the full duplex radio does not provide specific support for access in emergency situations.

Ellouze et al. [13] state that the shield jams every unauthenticated packet, so the verification of its checksum at the IMD level fails. This does not prevent the IMD from losing energy to read forged packets. Sending several unauthenticated packets can have battery draining as a consequence.

4.2.2.5 *The IMDGuard*

Xu et al. [43] present a solution with an ON/OFF switch to control security protections and switch between emergency and non-emergency situations. They notice two challenges in this configuration: first, no secret should be pre-deployed inside the IMD. In this way situations where the user is unable to recall the secret and needs to rekey the IMD are avoided. Second, there has to be a reliable method to prevent an

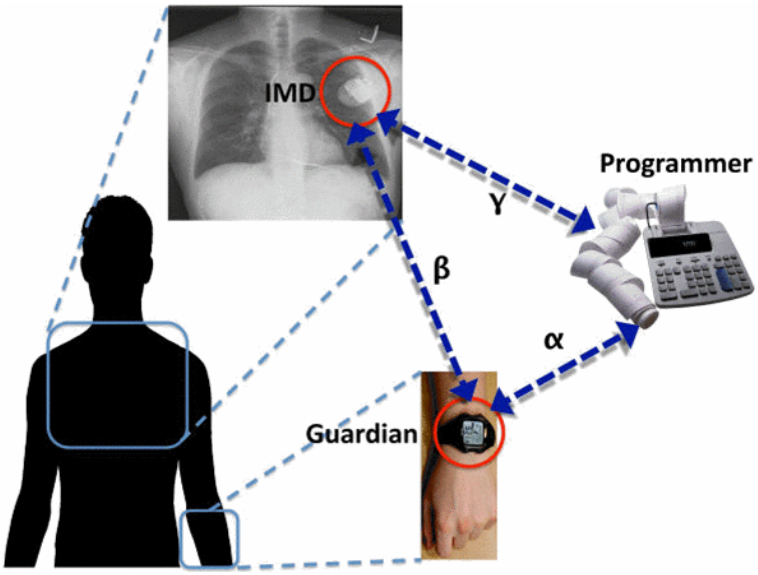

Figure 9: The use of the Guardian [43].

adversary from convincing the IMD that the external device is absent. The IMDGuard is a security scheme for implantable cardiac devices, including ICDs and pacemakers. As a Cloaker, the IMDGuard leverages the Guardian, which is an external wearable device, to coordinate interactions between the IMD and the doctor. This happens in a way that provides the security in a regular condition and safely allow access in an emergency.

Xu et al. propose an information-theoretic secure extraction scheme for ECG based key agreements and a security protocol for the architecture that uses external devices as an authentication proxy to protect the IMD. Compared to the fail-open approach of the Cloakers, this design avoids the periodic message broadcasting which consumes considerable battery power and exposes the patients to privacy risks. It protects the IMD without an assumption on the adversary's transmission capability and it is comprehensive. In Figure 9, the use of the Guardian is shown.

The Guardian is a wearable device with more power and computational resources than the IMD. It works as a proxy for the IMD and performs the authentication on its behalf. Both the Guardian and IMD are capable of measuring ECG signals. Xu et al. make the assumption that there is no adversary in an emergency situation. They classify the attacks in two types: the first is when the adversary tries to impersonate the Guardian and in the second, the adversary may spoof the absence of the Guardian to switch to the emergency mode.

The Guardian performs two essential functions. First, it is used to control which mode the IMD should enter: regular or emergency. In regular mode, the programmer that will interact with the IMD will first be authenticated by the Guardian, which will then issue the appropriate keys to the IMD and programmer. Second, the Guardian will authenticate the programmer on behalf of the IMD. They do not assume that the IMD must associate exclusively with one Guardian. It needs to be initialized by sharing a secret key between the IMD and the Guardian, such that they recognize each other. In case the Guardian is broken or lost, a new Guardian can be paired easily with the IMD without retrieving the old key or resetting the IMD. They also assume

that the Guardian has a list of legitimate programmers and their corresponding public keys that will be used in case of non-emergency mode.

The basic IMD protocol for this system is as follows. The IMD periodically wakes up to determine whether there is a request from the programmer. When the IMD receives a request, the IMD sends its ID and a random nonce to the programmer and starts a timer to wait for the Guardian. If the Guardian is present and responds during the timer, the regular condition protocol mode will run. When the Guardian does not respond before the timer times out, the IMD will run the emergency condition protocol.

Rostami et al. [38] show that this protocol is vulnerable to MITM attacks, since the effective key length of the key used in the encryption can be reduced from 129 bits to 86 bits. Nowadays, it is possible to brute force this key with available equipment. Furthermore, Ellouze et al. [13] show that the system is vulnerable to DoS attacks, by replaying a certain response of the guardian. Replayed messages should be decrypted by the IMD before it can decide to reject it or not. With this type of attack, it is possible to drain the battery.

4.2.2.6 *Analysis of proxy-based solutions*

Denning et al. [9] study a fundamental gap between the development of technical mechanisms that could protect the security of medical devices and the security defenses that all parties, including the patients and doctors, will accept. They suggest three relevant security properties for IMDs: authorized clinical access, emergency access and security in the sense that no unauthorized person should be able to change settings or view information on the patient's IMD.

They present four different technical security approaches as a strategy to find the balance between security for IMDs and safety for the patient in case of an emergency. The first approach consists of using passwords, which is a tradition in the security community and natural to investigate, but difficult to carry and to access in case of emergencies. The second category consists of additional patient body modifications, which can be for example a tattoo that serves as a password. The third consists of a change in behavior of the patient, an example of this is to ask the patient to carry an access card. The fourth and last approach is passive with respect to the patient. An example of this is using biometrics.

Participants are asked for their opinion about the following security mechanisms: medical alert bracelets, visible tattoos, UV-visible tattoos, wristbands, a critically aware IMD with no extra modifications and proximity bootstrapping. Although most mechanisms are clear from their name, proximity bootstrapping needs some further explanation. This system consists of an external device that is used by the medical staff. By placing it close to the patient, a temporary key is shared between the device and the IMD. Compared to the other ideas, this system does not burden the patient, since he or she does not have to participate in this.

Denning et al. conclude that the password and body modification solutions were the least favorite. Both the wristband with emergency and warning functionality as patient passive solutions as proximity bootstrapping were preferred by the patients. One major comment was that participants did not like the idea of wearing or seeing something that would remind them of their condition. This could be upsetting. Most of the participants had difficulties with wearing something all the time under differ-

ent conditions. Also cultural and historical associations had as a consequence that certain solutions were not liked by the participants. Denning et al. conclude that the psychological effects of the technology should be minimized.

4.3 EMERGENCY ACCESS

This section describes several solutions for accessing an IMD in case of an emergency.

4.3.1 *Emergency access for close range communication*

Rasmussen et al. [36] present a solution for emergency access for proximity based access control. In the emergency mode, it is assumed that the authorization token is not available. In most existing systems, wireless communication is only possible when the IMD is activated by a magnetic switch. Rasmussen et al. claim that these physical backdoors have many drawbacks. They suggest a new solution in which the security range should be much smaller (2-4 cm) than in the normal mode of operation. If a malicious reader is inside the security range when the IMD is in emergency mode, the reader has free access by design. They state that token-based approaches have several drawbacks, since it does not protect against the loss or theft of the token.

4.3.2 *Emergency access for solutions without a proxy*

Ellouze et al. [12] describe that the RFID reader is used to interact with the WISP to execute the powerless mutual authentication protocol. In emergency situations, the same way of mutual authentication is used as the first time the programmer tries to access the IMD without having preshared credentials. This is achieved when the IMD and programmer share a biometric key, which is dynamically and securely generated during access starting from the ECG.

Strydis et al. [40] define a way to get fail-open access in close proximity to the implant. This is possible by the use of a magnetic switch. They argue that this should be sufficient for next-generation IMDs, but they also state that an adversary may have significantly powerful equipment at his or her disposal capable of changing the security mode in case of using a magnetic switch. Also, systems using this technique may be prone to accidental mode switching when in presence of relatively strong magnetic fields.

4.3.3 *Emergency access for solutions with a proxy*

The Cloaker can be removed from the patient in case of an emergency [10]. When this happens, the doctor can access the IMD with previously unauthorized commercial programmers. By this operation, immediate, emergency open access is enabled.

The Personal Security Device provides a fail-open property in case of an emergency [34]. Unbounded access to the device can be granted by turning off the PSD. They state that this mechanism is needed, since life-critical medical devices should always be available in case of an emergency.

Xu et al. [43] present two modes of operation: a regular mode and an emergency mode. If the Guardian is present and responds during the timer, the regular condition protocol mode will run. When the Guardian does not respond before the timer times out, the IMD will run the emergency condition protocol. This means that to get access to the IMD in case of an emergency, it is sufficient to remove the Guardian from the body of the patient. During the emergency condition protocol, the IMD checks if the programmer is able to send a response that is expected in this emergency mode. In this mode, every programmer should be able to communicate with the IMD.

4.3.4 *Emergency-based solutions*

There are several systems that focus on solutions for emergency situations. In this case, no distinction is made between a regular or an emergency mode, but the current mode is specifically suitable for emergencies.

4.3.4.1 *Heart-to-heart*

Rostami et al. [39] present the solution Heart-to-heart for emergency situations. In non-emergency situation, when a patient is receiving routine medical care, they suggest the medical personnel to retrieve device-specific keys. In emergency situations or a situation when a patient is traveling abroad, Heart-to-heart can be usefull for a secondary authentication mechanism. Rostami et al. present a way to use electrocardiograms (ECG) as an authentication mechanism to ensure that only authorized external medical device controllers will make contact with the IMD. A new technique is developed for extracting time-varying randomness from ECG signals together with a new cryptographic device pairing protocol that uses this randomness to protect against attacks. In this system, a programmer will make contact with the IMD only if it has significant physical contact with the patient's body. Authentication to the IMD lapses once the programmer loses physical contact with the patient. This solution enforces a touch-to-access policy using a time-varying biometric called a physiological value (PV). Both the programmer and the IMD take their own reading of the PV. If both readings are nearly equal, the programmer obtains access to the IMD.

The IMD consists of a microcontroller, an ECG analog A/D front end and a wireless sensor modem, so current IMDs should be modified. Rostami et al. state that their system is secure against jamming and replay attacks, but they only tested briefly against remote heart rate monitoring results. They do not address DoS attacks and eavesdropping.

4.3.4.2 *Biometric-based access control*

Hei et al. [22] propose a light-weight secure access control scheme for IMDs during emergencies, using the patient's biometric information to prevent unauthorized access to IMDs and to achieve authentication. The scheme consists of two levels. The first one employs some basic biometric information of the patient and is lightweight. This consists of fingerprints' pattern, height and eye color. The second level utilized patient's iris data for authentication and this combination of two levels should be effective. Having a low false acceptance rate, a false rejection rate and small memory and computation overheads, this is presented as a very effective scheme.

Because of severe resource constraints of IMDs, as energy supply, processing and storage, traditional security schemes are not directly applicable. As an example, an IMD from 2002 has only 8kb storage. In this scheme, clinical personnel do not need to have or know a key. They only need devices to measure the biometric properties of the patient. In non-emergency situations, an IMD reader needs to pass an authentication process in order to access an IMD.

Hei et al. see that in emergency situations, biometric data can be destroyed, as fingerprints in case of a fire. However, they assume that in this case, help for the IMD is secondary and other medical treatments are more important. Because of the lack of cryptography, it is possible to eavesdrop during communication between an IMD and a programmer. Furthermore, no technique is used to prevent DoS attacks. However, iris image verification technique is unpractical, because a specific camera is required to capture the patient's iris [13]. In addition, eye illnesses for aged people can cause many false positives and negatives during authentication of the patient's IMD.

4.3.5 *Break-The-Glass*

Break-The-Glass suggests to use a system with maximum freedom and maximum responsibility [16]. Maximum freedom stands for the property that the system must provide mechanisms for the users to access the requested information at all times, whenever it is needed. Maximum responsibility means that the system must provide mechanisms to show the user an alert message, when he or she is trying to access information he or she is not authorized to see.

Ferreira et al. [16] state that the BTG principle can be implemented in the following way. When a user does a request to the system, the authorization model verifies whether the user has access to the requested information. If the model returns 'yes', the user gets immediate access. In case of 'no', the user is shown that he or she can still break the glass and he or she will be notified that all his or her actions are recorded from then. The user decides to break the glass or leave the system. In case of breaking, a hierarchy model verifies who has to be notified and continues. From then all notifications and user actions are registered automatically.

4.4 DISCUSSION

In this chapter, several solutions for securing the IMD are presented. For each system, we looked at the architecture of the system and its functionality compared to the architecture described in Chapter 2, its resistance against the attacks mentioned in Chapter 3 and the way it handles emergency access. In the next three table, we compared the aspects of the architecture, attacks and the emergency mode for ten systems described in this chapter. Table 2 shows that only solutions with a change in the hardware components of the IMD are able to prevent against battery draining. In none of the systems that use a proxy, it is possible to prevent battery draining. It is also remarkable that four out of ten system do not talk about battery draining at all. This suggests that the designer of the new system did not take this into consideration and the system will probaly not be able to prevent battery draining attacks.

We can conclude that no current system is able to prevent all the five types of attacks. It is remarkable that only two systems consider all types of attacks and that most systems only worry about active attacks and do not take eavesdropping into account. Because of this, it is hard to get an overview of all types of systems and their possibilities. Furthermore, only half of the systems have proofs for some security aspects. Other systems state that they are secure against certain types of attacks, by giving an indication or without further explanation. From this, we can conclude that current research lacks in considering all relevant security aspects.

From the table, we can see that most systems have an emergency mode where weaker or less security is used. Three systems do not have a specific emergency mode and will operate in their regular mode during emergencies. The differences and similarities between the security modes of the systems are hard to describe, since some systems only state that there is an emergency mode or shortly give a description, while others give extensive protocols. It is hard to compare these different approaches.

Table 2: A comparison of existing solutions.

Legend
n/a: not mentioned in the research
V: proven to be protected
V*: stated to be protected
x: vulnerable

Properties \ Systems	Proximity	Block cipher	RF energy harvesting	SISC	PSD	MedMon	The Shield	IMDGuard	H2H	Biometric
Using a proxy	no	no	no	no	yes	yes	yes	yes	no	no
Battery draining	V*	n/a	V	V	n/a	x	x	x	n/a	n/a
Modify the IMD hardware	no	no	yes	yes	no	no	no	no	yes	no
Emergency mode	yes	no	yes	yes	yes	no	no	yes	yes	yes
Eavesdropping attacks	n/a	n/a	n/a	V*	n/a	x	V	n/a	n/a	n/a
MITM attacks	V*	n/a	n/a	V*	x	V	n/a	x	V*	n/a
Replay attacks	V*	n/a	V	V	n/a	V*	n/a	V*	V*	n/a
DoS attacks	V*	n/a	V	V	x	x	n/a	x	n/a	n/a
Jamming attacks	n/a	n/a	n/a	x	x	x	V	V	V*	n/a

5 REQUIREMENTS AND ASSUMPTIONS

In this chapter, we define all assumptions and requirements for an optimal system solution. Assumptions and requirements are categorized in the categories general, emergency access and energy consumption. We show that these requirements are difficult to implement at the same time. For this reason, two types of system solutions are introduced at the end of this chapter.

5.1 GENERAL ASSUMPTIONS

The assumptions for an optimal solution are listed in Table 3. All assumptions are general asumptions (GA) and are not related to emergency access and energy consumption. These general assumptions are based on how the system works and on the environment where the systems are used.

For GA1, the first assumption in Table 3, we assume that programmers are stored in a locked room that is only accessible by authorized medical staff. They are only used by the staff that is authorized to treat patients and is trusted. GA2 is about the location where the patient is treated. We assume that the patient is treated in the same hospital for regular treatments. GA3 assumes that no medical staff member will misuse a valid programmer on purpose to change the behavior of the pacemaker. Staff members are authorized and screened beforehand. GA4 assumes that the hospital makes use of a back-end system. This can be a local system or can be connected to the Internet. Details about the security of a connection from the back-end to the Internet are out of the scope of this research. This back-end system is needed to store the cryptographic keys. In case of the use of a telemonitoring system at the home of the patient, the back-end system needs to be reached by the telemonitoring system over the Internet. GA5 is about contact between a patient and an attacker. Attacks on patients where the attacker can touch the patient and that will have consequences on the pacemaker or its behavior are out of scope. GA6 is based on the fact that the Federal Communications Commission [15] stated that a transmission is always started by a device external to the body. GA7 assumed that a treatment takes place once in 6 to 9 months, but it can happen more frequently if necessary.

5.2 REQUIREMENTS FOR AN OPTIMAL SOLUTION

For an optimal solution, we can define various requirements. General requirements (GR) are based on security principles and on the environment where the systems are used. They should hold in general situations and during general treatments, but not during emergencies. Requirements for emergency access (ER) and energy consumption (ECR) are based on the challenges presented in Chapter 3. A list of all the requirements is presented in Table 4.

GR1, GR2 and GR3 are related to access by authorized entities. The programmer should only communicate with a pacemaker implanted in a patient's body. More im-

ID	Assumption
GA1	Programmers in a hospital are only accesible by authorized medical staff members.
GA2	A patient is treated in the same hospital during regular treatments.
GA3	The medical staff from a hospital is trusted.
GA4	The programmer is connected with a back-end system.
GA5	An attacker does not attack a patient by having physical contact to the body.
GA6	A pacemaker does not initiate a transmission.
GA7	Treatments for patients happen twice a year on average.

Table 3: Assumptions for an optimal solution.

portant, a pacemaker should only communicate with a legitimate programmer. The same holds for telemonitoring systems. In this way, access to the pacemaker is restricted to authorized entities. These authorized entities should be able to access data on the pacemaker at all times. This should not be blocked by DoS attacks. GR4 is about the protection of the data communication, which can be realized by encrypting the communication with the use of strong cryptographic primitives. In the optimal solution, this should be the case both during general treatments and during emergency situations. GR5 and GR6 are about the visibility of the presence of the pacemaker to medical staff members and attackers. The presence should be visible to authorized entities, but unknown to attackers in the ideal case. GR7 and GR8 are about the applicability of the solution. The solution should be applicable on new pacemakers, but also on pacemakers that are already implanted in the body. Otherwise, since a pacemaker has a lifetime of 10 years, in the worst case the patient has to wait for 10 years or have a surgery and replace the pacemaker. Replacing pacemakers in all patients bodies will cost an enormous amount of money and is undesirable. At the same time, it is preferable that the solution is applicable to all types of pacemakers, which will require standards in the communication protocols for programmers and pacemakers. GR9 describes the privileges for the manufacturers. They should be able to access data different from those for healthcare professionals and patients. GR10 describes that one programmer can only communicate with one pacemaker at a time and one pacemaker can only communicate with one programmer at a time. GR11 mentions the use of a back-end IT system. Nowadays, a physical database can be sufficient for some hospitals, but an electronic back-end system is necessary for the key management, as described in Chapter 8.

ER1, ER2 and ER3 are about access to the patient in case of emergency. In this situation, each medical staff member over the world who is able to treat patients with a pacemaker, is authorized to access the pacemaker. They should be able to read out the log history of the patient, to change appropriate pacemaker settings and to upgrade pacemaker firmware and applications.

For ECR1 and ECR2, it is necessary to minimize computation within the pacemaker, communication with other devices and data storage. Furthermore, battery draining as a cause of attacks should be prevented as much as possible. All these aspects can decrease the lifetime of the pacemaker, which is undesirable.

ID	Requirement
GR1	The pacemaker and the programmer should authenticate to each other.
GR2	Only authorized entities should be able to communicate with the pacemaker.
GR3	Data on the pacemaker should be always available to appropriate entities.
GR4	The communication between the pacemaker and the programmer should be protected.
GR5	The presence and type of the pacemaker should be known to medical staff members who want to treat the patient.
GR6	The existence of the pacemaker should be hidden for attackers.
GR7	The solution should be directly applicable on all pacemakers.
GR8	The solution should be applicable to pacemakers from different manufacturers.
GR9	The manufacturer should be able to audit the operational history of the pacemaker in case of a failure.
GR10	The communication between the programmer and the pacemaker is one-to-one.
GR11	The system should use a back-end IT system.
ER1	In case of emergency, a medical staff member should be able to access the pacemaker in the hospital.
ER2	If the patient requires an operation during an emergency and carries a pacemaker, the pacemaker must be deactivated before the operation in order to prevent unintentional shocks to the patient.
ER3	In case of emergency, the pacemaker should only be available for all medical staff members worldwide.
ECR1	The solution should decrease the lifetime of the pacemaker as less as possible.
ECR2	The solution should not interfere with the functionality of the system.

Table 4: Requirements for an optimal solution.

5.3 DISCUSSION

Looking at the requirements in Table 4, we see that some of the general requirements are difficult to implement at the same time as emergency requirements and energy consumption requirements.

- GR3 and GR4: A new solution should be directly applicable, even on pacemakers that are already implanted in patients bodies, but at the same time strong cryptographic mechanisms should be implemented on the device. It is not known whether only software design changes are enough for the pacemaker, since hardware design changes might be necessary. For now the specifications of the hardware components are unknown, so we cannot be sure that strong cryptographic mechanisms can be applied on current pacemakers.
- GR4 and ECR1: It is difficult to implement GR2 with the energy consumption requirements at the same time. On the one hand, strong cryptographic mechanisms should be implemented on the device; on the other hand computation power should be minimized, since this decreases the lifetime of the

pacemaker. Cryptographic mechanisms require of a lot of computations, so the performance of various hardware components should be improved to maintain the current lifetime of the pacemaker.

- GR3 and ER1 and ER3: On the one hand, the data should be always available to authorized entities, in general the cardiologist and pacemaker technicians who treat the patient. However, the number of authorized entities should be extended in case of an emergency. It is difficult to make medical staff members authorized entities and to give them access. It is possible to extend authorization to more medical staff member in the hospital where the patient gets his or her regular treatments, but for medical staff in other hospitals and even in other countries, this is a challenge. In this situation, we have to deal with different trust domains which makes it harder. At the moment, there is no worldwide medical database that could save this type of information and this probably will not be there on short term.

Because of these issues, we need to make some trade-offs and thus we present two solutions to solve this problem. The first solution presented in Chapter 6 is a solution which is directly implementable for implanted pacemakers. This solution makes use of a proxy which is placed between the programmer and the pacemaker. This proxy contains a jammer, which only allows communication between the proxy and the pacemaker but blocks all the traffic from other devices to the pacemaker. Without using any form of encryption of the communication between the proxy and the pacemaker, this solution does not require any changes of the design of the pacemaker. Encryption of the communication between the proxy and the programmer is needed. In the next chapter, we present the system design of the solution together with the design of the proxy and the security protocol for the communication between the programmer and the proxy. We also present a solution for access in emergency situations.

In Chapter 7, we present a solution which does not require the use of the proxy. The use of an external device is undesirable, because it can be lost, stolen, broken or forgotten by the patient. We define a security protocol together with an analysis of its correctness and we make an analysis on the ciphers that may be used in combination with the protocol.

Important part of both solutions is key management. Key generation methods depend on the capability of devices, which is not straightforward for resource constrained devices. The management of these keys requires that keys are stored on a safe place. Since programmers can contain malware, this is not an appropriate place to store them. Furthermore, in emergency situations the pacemaker should still be available. In Chapter 8, details about key management are discussed and Chapter 9 presents a new emergency access solution which does not use the cryptographic keys.

6 PROXY-BASED SOLUTION

In this chapter, we present our solution that is directly applicable on already implanted pacemakers. By making use of a proxy, secure communication can be guaranteed, except for emergency situations where fail open access is provided. Together with our formulated assumptions and requirements for this solution, the architecture of the system is presented including the technical requirements of the proxy. Furthermore, a security protocol is presented that is used for the communication between the proxy and the programmer. In Chapter 8, we present details about the key management for this solution.

6.1 ASSUMPTIONS

For this proxy-based solution all general assumptions (GA1 to GA7) from Section 4.1 hold, since these are listed as general assumptions. A few additional assumptions have to be made related to the design of the new system and the fact that it should be possible to implement this immediately on already implanted devices. The assumptions concern the use of a proxy (PA) and emergency access (PEA). These additional assumptions are presented in Table 5.

PA1 requires a setting in which the distance between the proxy and the pacemaker is small. This is necessary to prevent that the pacemaker willl communicate with other devices than the proxy. This can be achieved when the proxy has the form of a necklace for example. PA2 is an extension of GA5, in which an attacker does not physically attack a patient. PEA1 can be assumed for now, since treatment is given in a hospital environment. An attacker carrying his or her own equipment in a hospital will be noticed, since current equipment is too large to hide. It is possible that in the future this may no longer be true.

6.2 REQUIREMENTS

Considering the requirements presented in Table 4, we can state the following. The general requirements that hold are GR1, GR2, GR4, GR5, GR7, GR8, GR9 and GR11. The proxy-based solution is able to meet all these requirements. This will become clear in the following sections of this chapter when the architecture and security mechanisms are discussed. There are two requirements GR3 and GR6 that cannot be realized. GR3 has to do with the availability of the pacemaker, which cannot be guaranteed because of possible DoS attacks. This system does not have a mechanism to prevent all types of DoS attacks. Although the proxy can be made in a way that an attacker does not notice the necklace, an attacker with the right equipment does notice the pacemaker, even when he or she is not able to authenticate. GR6 is not achieved, since attackers will always notice the existence of the pacemaker, when they have the right equipment. GR10 is replaced by requirement PGR2 presented in Table 6, since GR10 does not consider the existence of a proxy. PGR2 dscribes the communication

ID	Assumption
PA1	The proxy is a wearable device located close to the pacemaker.
PA2	The attacker does not try to physically remove the proxy.
PEA1	There is very little chance an attacker is around during an emergency treatment.

Table 5: Assumptions for a proxy-based solution

ID	Requirement
PGR1	All communication between a programmer and a pacemaker should pass the proxy.
PGR2	The communication between the programmer and the proxy is one-to-one during a treatment session.
PGR3	The communication between the pacemaker and the proxy is one-to-one.
PTR1	The proxy is able to execute cryptographic operations and to store cryptographic keys.
PTR2	The programmer is able to execute cryptographic operations.
PTR3	The back-end should store cryptographic keys.

Table 6: Requirements for a proxy-based solution.

between the programmer and a proxy during a treatment session. In a session, a programmer can only communicate with one proxy and a proxy can only communicate with one programmer at the same time. PGR3 states that the communication between a pacemaker and a proxy is always one-to-one. This has to do with the fact that these devices are linked to each other during the first treatment. Both ER1 and ER2 hold, since the pacemaker can be accessed by everyone in case of an emergency situation. This has to do with the architecture of the system and is explained in the next section. ER3 does not hold, since an attacker also has access during an emergency. Both energy consumption related requirements ECR1 and ECR2 hold because the proxy takes over the heavy computations from the pacemaker.

The following general requirements (PGR) and technical requirements (PTR) in Table 6 are added because of the use of a proxy. The general requirements are general in the sense that they are applicable on general situations and during general treatments. They do not hold during emergencies.

Note that the requirements for the programmer also hold for the telemonitoring system. PGR1 specifies how the communication goes between the programmer, proxy and pacemaker. The proxy should block the traffic to the pacemaker from other sources than itself. The proxy has the task to only let the pacemaker indirectly communicate with authorized entities. The architecture is further explained in the next section. PGR2 describes that one programmer can only communicate with one proxy at the time and one proxy can only communicate with one programmer at the time. This is comparable with GR10, which describes the same for the pacemaker instead of the proxy. PTR1 and PTR2 require specific technical functionalities to provide authentication and secure communication both for the proxy and the programmer. As

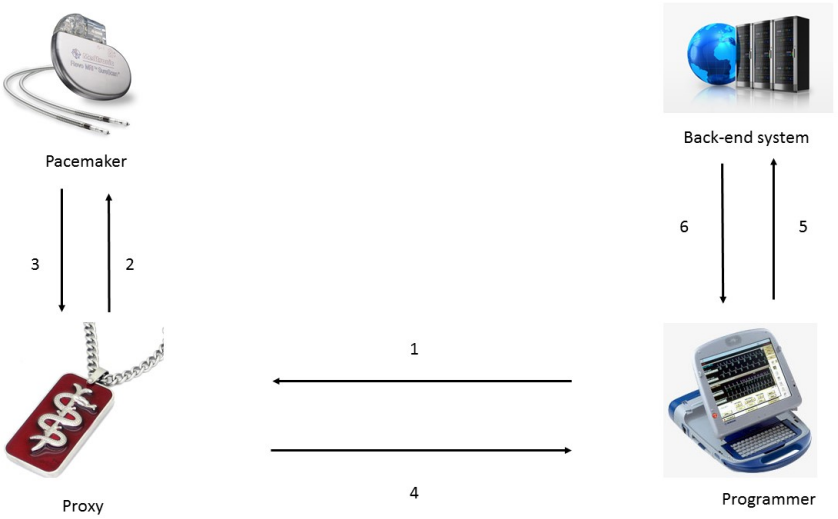

Figure 10: The architecture of the proxy-based solution.

Legend
1: The programmer authenticates to the proxy which is necessary to communicate with the pacemaker.
2: After authentication, the proxy forwards the configuration or data request to the pacemaker.
3: The pacemaker responds by this request and sends the data to the proxy.
4: The proxy forwards the data to the programmer.
5: The programmer sends the data to the back-end system.
6: The back-end system sends data from previous treatments to the programmer.

presented in Section 5.3, PTR3 specifies the role of the back-end system in the architecture.

6.3 ARCHITECTURE

The architecture of the system consists of five components: a programmer, a proxy, a pacemaker, a telemonitoring system and a back-end system. These are related to each other as presented in Figure 10. As we can see, the figure differs from Figure 1 in the sense that all communication between the programmer and the pacemaker goes via the proxy. The telemonitoring system is not part of the figure, because it takes the same place as the programmer. The only difference between the setting with the programmer and the telemonitoring system is the use of arrow 6. This arrow is there for the programmer, but not for the telemonitoring system. The back-end system of the hospital does not send any data from previous treatments to the telemonitoring system.

In the remainder of the section we present the proxy, while we refer to Chapter 2 for details about other devices and components. The proxy is based on the design of the Shield [17] as described in Section 4.2.2.4. This design is chosen since from all existing solutions with proxies, this one provides secure communication from the pacemaker to the programmer, without using cryptographic mechanisms between

the pacemaker and the proxy. As far as we could find, this is the only solution with these characteristics and fits our requirements the best. The Shield uses two antennas: a jamming antenna and a receive antenna that are placed next to each other. The jamming antenna transmits a random signal to prevent eavesdroppers from decoding the pacemaker's transmissions. The receive antenna is simultaneously connected to both a transmit and a receive chain. The transmit chain sends an antidote signal that cancels the jamming signal at the receive antenna's front end, allowing it to receive the pacemaker's signal and decode it.

As a consequence of this technique, the proxy can transmit the pacemaker's signal to an authorized programmer. At the same time, the proxy listens for unauthorized transmissions addressing the pacemaker and jams them in order to protect the pacemaker against commands from unauthorized programmers. Because of the jamming of the proxy, the pacemaker is not able to decode transmissions from an adversary. Halperin et al. [20] proved with their technique that an implantable cardiac defibrillator which is comparable to a pacemaker, does not respond to unauthorized commands when using the Shield, even when the adversary is only 20 cm away.

This proxy-based solution would also be applicable to other IMDs. The only requirement is that the proxy should be close to the IMD, to use its jamming functionality properly. In case of a pacemaker, this can be in the form of a necklace. For other IMDs it can be more difficult to carry a proxy and to hide it in an object or an accessory.

Details of the configuration of the Shield are presented By Fu et al. [17]. Two main components that are necessary for this technique are a USRP2 software radio and a 400 MHz daughterboard for compatibility with the MICS band.

Using a proxy with properties of the Shield solves part of the problems of the unsecured communication, since all communications are led by the proxy and the proxy can determine what requests are sent to the pacemaker. Although the proxy can block other entities to eavesdrop messages between the proxy and the programmer, the communication between the programmer and the proxy is unencrypted. Fu et al. [17] state that a secure channel should be built for this part of the communication. Our research completes this proposal of the Shield by choosing a proper mutual authentication protocol, building a secure channel and by defining a key management system for this solution. This key management system will be presented in Chapter 8.

6.4 COMMUNICATION PROTOCOL

In this section, a protocol is given for mutual authentication between the programmer and the proxy. Furthermore, one protocol is given for secure communication between the programmer and the proxy.

6.4.1 *Mutual authentication*

As a first step, both the proxy and the programmer should authenticate to each other. The programmer wants to know that it is communicating with the real proxy, but even more important: the proxy wants to be sure that it is talking to the legitimate programmer. Since the programmer takes the initiative for communication, the pro-

Notation	
A	Programmer
B	Proxy
N_a	Nonce from the programmer
N_b	Nonce from the proxy
K_a	Public key from the programmer
K_b	Public key from the proxy
K_a^{-1}	Private key from the programmer
K_b^{-1}	Private key from the proxy
$\{m\}_K$	Message m encrypted with key K
T	Timestamp

Table 7: The notation for a proxy-based solution.

1. $A \rightarrow B : \{N_a, A\}_{K_b}$
2. $B \rightarrow A : \{B, N_a, N_b\}_{K_a}$
3. $A \rightarrow B : \{N_b\}_{K_b}$

Figure 11: The mutual authentication protocol for the proxy-based solution.

grammer should present itself and announce a communication process to the proxy. After mutual authentication, the programmer can start a session in which messages can be exchanged.

Since there are less limitations related to cryptographic operations for both devices, because the proxy can replaced at any moment when the battery is empty, we can make use of asymmetric cryptography. In particular, we use the Needham-Schroeder-Lowe protocol [27], which has not been broken since its creation in 1995 and is not expected to break in the near future. Lowe fixed the public key protocol of Needham-Schroeder [35], which appeared to be vulnerable against Man in the Middle attacks. The idea of the protocol is that both parties prove their identity to each other by sending and receiving fresh nonces that are encrypted with the public key of the receiving entity. When both parties are able to show that they received the nonce that the other party send to them, mutual authentication is achieved.

For the moment, assume both entities have the public key of the other entity and they have their own public-private key pair. In Chapter 8, we present the key generation and key management process for this protocol. The notation in Table 7 is used in both mutual authentication and secure communication protocols. The protocol for mutual authentication is presented in Figure 11. Note that the protocol also holds for the telemonitoring system and the pacemaker.

- Step 1: A tries to establish a connection with B by generating a nonce N_a and sending it with its identity to B, using B's public key. When B receives this message, it decrypts it to obtain the nonce N_a.
- Step 2: B returns its own identity together with the nonce N_a and a new nonce N_b to A, encrypted with A's key. When A receives this message, it should be

$$1.\ A \rightarrow B : \{\{B, T, N, m_1\}_{K_a^{-1}}\}_{K_b}$$
$$2.\ B \rightarrow A : \{\{A, T, N, m_2\}_{K_b^{-1}}\}_{K_a}$$

Figure 12: The secure communication protocol for the proxy-based solution.

assured that it is talking to B, since only B should be able to decrypt the first message to obtain N_a.

- Step 3: A returns the nonce N_b to B, encrypted with B's key. When B receives this message it should be assured that it is talking to A, since only A should be able to decrypt the second message to obtain N_b.

6.4.2 *Secure communication*

After the mutual authentication phase, the programmer and the proxy communicate over a secure channel presented in Figure 12.

- Step 1: The programmer sends a request m_1 to the proxy. This request is first signed by the sender with his or her private key and then encrypted with the public key of the receiver. Furthermore the receiver, a timestamp and a nonce are included in the request.
- Step 2: The proxy replies to the programmer by sending m_2 and by using the same format of the message as is used in step 1.

This stream of messages can be repeated multiple times. Each time A sends a message, a fresh nonce is created. B uses the same nonce in its reply.

6.5 DISCUSSION

As Fu et al. [17] state, the proxy with properties from the Shield protects the pacemaker from attackers who want to eavesdrop or do MITM or jamming attacks. The Needham-Schroeder-Lowe protocol is widely used for mutual authentication, so we can assume that it prevents eavesdropping, MITM and replay attacks when it is implemented correctly. Now we need to verify that the protocol for secure communication also prevents these attacks.

Considering the communication between the programmer and the proxy, the following things can be concluded. Assume an attacker attempts to perform a MITM attack. If the attacker wants to start the communication by pretending he or she is another entity, he or she has a problem with signing the message, since he or she cannot pretend to be someone else. When the attacker wants to forward a message that was meant for him or her to retrieve information from another entity, this entity sees that the message was originally not meant for him or her, since the receiver is always part of the message.

Assume an attacker attempts to perform a replay attack by sending a message again that he or she sees over the communication channel. Both in the case of the programmer and the proxy, this is noticed because of nonce and the time stamp.

Even if the attacker has some knowledge about the composition of the message, it is hard to guess the correct nonce together with a correct timestamp.

It is possible to perform certain types of DoS attacks on the proxy, but this has limited influence on the pacemaker. The programmer only accepts and forwards messages coming from the authorized programmer, so messages from other programmers are blocked. The proxy should only be available during treatments in a hospital and we assumed that there is a very small chance that an attacker is around, because of the phsyical security of the hospital and the size of current programmers or self-made equipment of an attacker. Since the proxy is an external device, there are fewer limitations on the battery than in case of the pacemaker. A DoS attack can cause battery draining of the proxy, but the battery can be replaced at any time without any surgery. It is not clear how much energy is consumed every time the pacemaker wakes up when a programmer tries to reach the proxy. Since all devices communicate on the same frequency, the pacemaker probably wakes up when it notices something on the frequency. Depending on the energy consumption for waking up, this might have battery draining as a consequence.

Looking at the requirements stated in this chapter, we see that all the general requirements, described in Section 6.2, are met. The secure communication is achieved because of the mutual authentication part of the protocol and the secure channel that is built. Furthermore, no changes on the pacemaker are necessary because of the use of the proxy. Also, the system is applicable on pacemakers from all manufacturers. Nevertheless, we had to make some trade-offs for this solution, to make it applicable on already implanted pacemakers. One of these trade-offs is unsecured communication in case of an emergency, since this is for now the most efficient way to avoid key management details. However, the largest risk of this solution is that it does not work when the proxy is lost, stolen, broken or just forgotten by the patient. This puts a large responsibility on the patient, which is undesirable. Furthermore, patients are not willing to wear additional devices and research showed that they are not aware of any risks or do not take these risks serious [9]. Because of this, we prefer a solution that does not use a proxy. A solution without a proxy is presented in the next chapter.

7 SHARED SECRET BASED SOLUTION

In this chapter, we present our new solution that is based on a shared secret between the programmer and the pacemaker. This solution does not make use of a proxy to guarantee protected communication. It uses security protocols for the communication between the pacemaker and the programmer and requires at least software modifications on current devices but probably also hardware changes. In the following sections, the assumptions and requirements are presented together with our designed communication protocol. A proof of the correctness of this protocol is given together with an extensive analysis of the cipher that will be used. Key management details are explained in Chapter 8. Emergency access proposals are described in Chapter 9. The chapter is concluded with a discussion section.

7.1 ASSUMPTIONS

For this solution, all the assumptions hold as listed in Section 5.1. For now we can make the assumption that the chance an attacker is around during an emergency is very small, but it is difficult to predict how this will be in a decade. When programmers will become smaller and attackers get more knowledge about making their own equipment, it can be more likely that attackers are around in emergency situations and in hospitals. This is the reason why this assumption PEA1 is listed in Section 6.1, but not stated explicitly in this chapter.

7.2 REQUIREMENTS

Considering the requirements from Table 4, we can observe that most of the requirements hold for this situation. GR1 and GR2 focus on authentication and authorized entities, which is guaranteed in this solution by the techniques presented in Section 7.4. The continuous availability in GR3 cannot be guaranteed since the solution presented in this chapter is not able to prevent all types of DoS attacks. The security mentioned in GR4 is achieved by the protocol presented in Section 7.4 and GR5 holds because of the solution, which standardizes communication protocols. For this reason, there is no need for different types of programmers. GR6 about the visibility of the pacemaker cannot be realized, since an attacker can notice the pacemaker if he or she has the right equipment. GR7 about the applicability of the solution on current pacemakers cannot be guaranteed, since this solution needs the use of strong cryptographic mechanisms and it is not known if already implanted pacemakers are able to use these mechanisms. It is possible that hardware changes are required. GR8 is applicable, but requires an official standard that needs to be implemented by all manufacturers. Furthermore, manufacturers can audit the pacemaker so GR9 holds. GR10 about the one-to-one communication is still valid, since there is no multi-communication feature added to the design. The existence of the back-end system mentioned in GR11 should also hold because of the use of cryptographic keys that

ID	Requirement
TR1	The pacemaker should be able to store a secret of 160 bits.
TR2	The pacemaker should be able to store 128 bit cryptographic keys.
TR3	The pacemaker should be able to perform the IDEA cipher.
TR4	The pacemaker should be able to perform Sha-1 hash operations.

Table 8: Technical requirements for a shared secret solution.

should be stored in the back-end system. The applicability of medical requirements ER1, ER2 and ER3 is discussed in Chapter 9 in which emergency access solutions are presented. Both energy consumption requirements ECR1 and ECR2 related to the lifetime and the functionality are met as best as possible. This will be presented in this chapter and in Chapter 8.

Besides these requirements, there are a few technical requirements related to the pacemaker that are presented in Table 8. The need for TR1, TR2 and TR4 is shortly explained in this chapter, but more in detail in Chapter 8. This has to do with the fact that these technical requirements are related to key generation. According to TR1, the pacemaker should be able to store a secret of 160 bits. This has to do with the key generation phase which is discussed in Chapter 8. This secret is shared with the back-end system of the hospital and the telemonitoring system at home and its use is further explained in Chapter 8. TR2 states that the pacemaker should be able to store cryptographic keys from a certain length that is secure enough. This length is set to 128 bits as required by the cipher that is used for this solution. Most block ciphers use a key of length 128 bits and is at the moment strong enough to prevent brute force attacks [41]. Storage of only a few cryptographic keys is necessary, since keys are only stored temporarily. TR3 focus on all the operations related to the IDEA cipher which the pacemaker should execute. The reason for the use of the IDEA cipher is explained in Section 7.6. TR4 is about the ability to perform a Sha-1 hash. This is necessary for the key generation process as explained in Chapter 8. Both the prorgammer and the telemonitoring system should be able to perform the same operations and store the same keys, but since these are computers with strong computational power and storage space, we do not state these requirements here explicitly.

7.3 ARCHITECTURE

The architecture of this solution, presented in Figure 13, consists of a pacemaker and a programmer which are directly communicating with each other. This is exactly the same as the current situation as presented in Figure 1. Both the programmer and the telemonitoring system are able to communicate directly with the pacemaker. No proxy comes in between to regulate the communication between the pacemaker and another device as in Figure 10. The programmer and the telemonitoring system are connected with the back-end system of the hospital, which stores confidential information about the encryption of the communication. Although the devices may change in software and hardware design, the overall architecture stays the same.

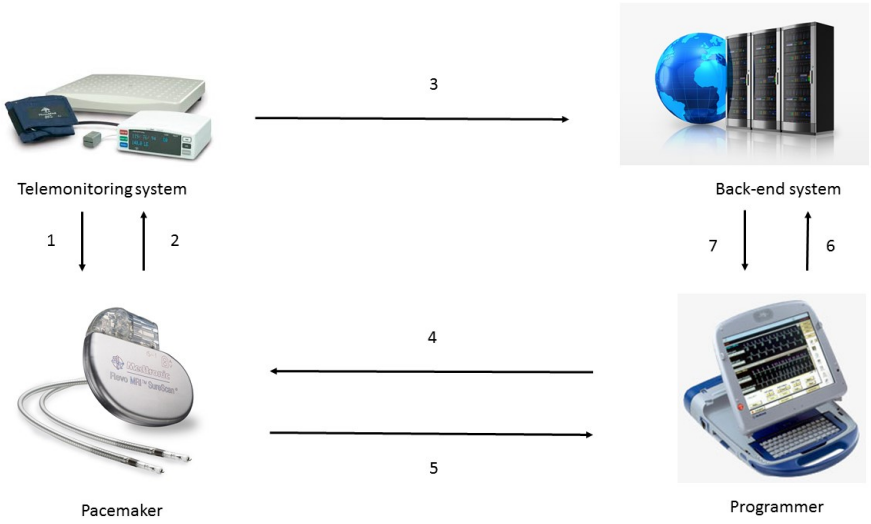

Figure 13: The architecture of the shared secret based solution.

Legend
1: The telemonitoring system requests the pacemaker for data and may be able to configure this device.
2: The pacemaker sends its data to the telemonitoring system.
3: The telemonitoring forwards the data from the pacemaker to the back-end system of the hospital.
4: The programmer configures the pacemaker and requests data stored on the pacemaker.
5: The pacemaker sends the data to the programmer.
6: The programmer forwards the data to the back-end system of the hospital.
7: The back-end system may send data to the programmer about previous treatment sessions.

Notation	
P	Programmer
I	Pacemaker (IMD)
N_p	Nonce from P
N_i	Nonce from I
K	Symmetric key shared between P and I
SK	Session key

Table 9: The notation for a shared secret solution.

1. $P \rightarrow I : \{P, N_p\}_K$
2. $I \rightarrow P : \{I, N_p, N_i\}_K$
3. $P \rightarrow I : \{N_i, SK\}_K$

Figure 14: The mutual authentication protocol for the shared secret based solution.

7.4 COMMUNICATION PROTOCOL

In this section two protocols are presented: one for mutual authentication and one for secure communication. The protocol for mutual authentication is based on the Needham-Schroeder-Lowe protocol presented in Chapter 6. With the use of the cryptographic protocol verifier ProVerif, we proved that the mutual authentication protocol is secure against eavesdropping and Man in the Middle attacks. For the secure communication protocol we prove that we build a strong channel which is protected against the same type of attacks. In the remaining part of this section, communication between the programmer and the pacemaker is considered. We assume the telemonitoring system can take the role of the programmer in all these processes.

7.4.1 *Mutual authentication*

For the mutual authentication protocol and the secure communication protocol, the notation is used as presented in Table 9.

Figure 14 presents the mutual authentication protocol for this solution. Note that the programmer can be replaced by the telemonitoring system in this protocol. The use of nonces and identities for the mutual authentication is based on the Needham-Schroeder-Lowe protocol, since this protocol is tested thoroughly and is short in its use. Different is the type of encryption: the Needham-Schroeder-Lowe protocol as presented in Chapter 6 uses asymmetric encryption, in this case symmetric encryption is used. This means that the programmer and pacemaker need to share a key beforehand. These details are described in Chapter 8.

1. The programmer (P) tries to establish a connection with the pacemaker (I) by generating a nonce N_p and sending it with its identity to the pacemaker, using their shared key. When the pacemaker receives this message, it decrypts it to obtain the nonce N_p.

$$P \rightarrow I : \{\{P, I, m, N\}_{SK}\}_{MAC}$$
$$I \rightarrow P : \{\{I, P, m, N\}_{SK}\}_{MAC}$$

Figure 15: The secure communication protocol for the shared secret based solution.

2. The pacemaker returns its own identity together with the nonce N_p and a new nonce N_i to the programmer, encrypted with the shared key. When the programmer receives this message, it should be assured that it is talking to the pacemaker, since only the pacemaker should be able to decrypt the first message to obtain N_p.

3. The programmer returns the nonce N_i and a session key to the pacemaker, encrypted with the shared key. This session key will be used in further communication according to the secure communication protocol. When the pacemaker receives this message, it should be assured that it is talking to the programmer, since only the programmer should be able to decrypt the second message to obtain N_i.

7.4.2 *Secure communication*

After the mutual authentication protocol, the programmer and the pacemaker communicate over a secure channel. This secure channel is built as in Figure 15. For this, the session key is used together with a message authentication code (MAC). Each message is initiated by the programmer. This device sends its identity together with the identity of the receiver, the message and the nonce to the pacemaker. This complete message is encrypted with the session key and then signed by using a MAC. The pacemaker can decrypt this message by computing the same MAC and using the session key and it will send a message back to the programmer using the same order in its message.

This stream of messages can be repeated multiple times. Each time P sends a message, a fresh nonce is created. The same nonce should be used by I in its reply to P. The generation of the MAC and the session key is described in Chapter 8.

7.5 SECURITY ANALYSIS

In this section, we present a security analysis of our protocols in ProVerif.

7.5.1 *ProVerif*

ProVerif is an automatic cryptographic protocol verifier with respect to the Dolev-Yao model [26]. This is a formal model used to prove properties of interactive cryptographic protocols. It assumes that the underlying key system is perfectly secure, that the adversary has complete control over the entire network and that concurrent executions of the protocol can occur. The most important properties that ProVerif can prove are secrecy, strong secrecy and authentication. ProVerif is sound but incomplete. This means that if the tool claims that a protocol satisfies a property, then the

property is satisfied. When the tool cannot prove a property, it tries to reconstruct an attack. This is an execution trace of the protocol that falsifies the desired property. It can give false attacks in this case, but then it is not able to prove the property.

The base of ProVerif lies in pi-calculus [1]. This is a process calculus which can deal with processes that communicate and can consider the execution of several processes in parallel. It is primitive in the sense that it does not contain primitives as numbers, booleans or functions. The notion of name is important in pi-calculus. A name can be a communication channel and a variable. There are various constructs of the pi-calculus: interaction, composition, restriction, replication, match and nil. Because of the basic structure of pi-calculus, it is easy to extend this calculus. One of these extensions is spi-calculus, designed to represent cryptographic protocols. It provides a setting to analyse security protocols. It is possible to express security guarantees as equivalences between spi calculus processes. This is achieved by the introduction of encryption and decryption constructs. ProVerif is a command-line tool which accepts the language pi-calculus and therefore also the spi-calculus.

7.5.2 *Protocol verification*

Now the security analysis is presented.

Protocol

For the analysis, only the protocol for mutual authentication as in Figure 14 is considered. For this protocol, two tests are done: one for mutual authentication and one for secrecy of the session key. The security of the secure channel is not tested with ProVerif, since this depends on the security of the mutual authentication protocol. If the session key is not revealed during the mutual authentication protocol, a secure channel can be built with the session key and the MAC as presented in Figure 15. No eavesdropping can happen and no MITM attack either, because of the form of encryption. Especially replay attacks cannot take place, because of the use of a fresh nonce in every request from the programmer to the pacemaker. This protocol is tested in ProVerif on mutual authentication and secrecy of the session key sent in the third message. For the syntax of ProVerif, we refer to the manual [26]. As a basis, we defined both protocols in ProVerif according to the communication stream.

Specification

For both tests on mutual authentication and secrecy, we implement the mutual authentication protocol as a basis. We define a free channel, the entities P and I (for the programmer and the pacemaker) and two functions for symmetric encryption and symmetric decryption. The session key is defined as a secret and has the label 'private'. Details can be found in Appendix A in Listings 1 and 2. The processes for P and I are defined by the messages that are coming in and the messages that are sent out. Furthermore, the processes contain the encryption and decryption functions for composing and decrypting messages. These processes are executed multiple times in parallel by the statement ((!P) | (!I)) to see if multiple sessions reveal some information during the mutual authentication phase. More details on the syntax of ProVerif can be found in the manual [26].

Properties

For secrecy, we only need to test on "query attacker: secret" to see if the attacker can retrieve the secret, which is the session key in this case, through the protocol. Furthermore, we explicitly state that we do not need to test on leakage of the symmetric key.

For mutual authentication, we test on one to one correspondence. This tests if an event M is always preceded by an event N and if every trace contains at least as many N-events as M-events. This is tested for specific lines in the protocol by placing it between a begin and an end statement. For the mutual authentication protocol, the following should be tested:

1. $P \rightarrow I : \{P, N_p\}_K$

I: begin(I,P,Np)

2. $I \rightarrow P : \{I, N_p, N_i\}_K$

P: end(I,P,Np)

P: begin(P,I,Ni)

3. $P \rightarrow I : \{N_i, SK\}_K$

I: end(P,I,Ni)

Now we test if each event in which the programmer receives nonce N_p from the pacemaker is really preceded by an action from the pacemaker where it sends the nonce N_p. The same is done for nonce N_i. By begin(P,I,Ni) and end(P,I,Ni) is checked if receiving N_i implies that N_i is really sent by P. This property is the same as the mathematical definition of injectivity. Now we can test mutual authentication, by specifying the following queries in ProVerif:

```
query evinj: end(x,y,z) ==> evinj:begin(x,y,z)
query evinj: end2(x,y,z) ==> evinj:begin2(x,y,z)
```

The first query tests authentication from I to P and the second query tests authentication from P to I. ProVerif explicitly tests if an attacker cannot mess with different sessions to reveal information about the encrypted data.

Results

In Appendix A, Listing 1 presents the input for testing for mutual authentication in ProVerif. In Listing 2, the output for ProVerif is given. We see that authentication for both parties is satisfied, since the result for both queries is true. This means that after the execution of the mutual authentication protocol, both parties can be sure that they are communicating with the entity they think they communicate with. For testing on secrecy, the session key that is sent from the programmer to the pacemaker in the third message is considered. Appendix A contains the input for ProVerif in Listing 3. Listing 4 presents the output from ProVerif and shows that the property is satisfied for this protocol, since ProVerif outputs true for the related query.

	MISTY1		IDEA		RC6	
Average power consumption (mW)	86		58		93	
Peak power consumption (mW)	96		95		107	
Program size (kb)	18.8		13.4		11.4	
Measurements	1kb	10kb	1kb	10kb	1kb	10kb
Total encryption energy costs(Joule)	0.04	0.22	0.03	0.17	0.02	0.11
Computation overhead	0.267	0.210	0.210	0.161	0.137	0.097
Encryption rate	4.5	5.2	3.9	4.6	9.7	13.5

Table 10: An overview of characteristics from block ciphers.

7.6 ANALYSIS OF CIPHERS

Encryption of the messages during mutual authentication and secure communication is done by the use of a cipher. Since we are dealing with a resource constrained device, this cipher should be chosen carefully. The first decision is about choosing between stream ciphers and block ciphers. We decided to choose for block ciphers. This is a symmetric cipher which operates on fixed-length groups of bits that are called blocks. For each block, the same transformation is done with the help of a secret key. We decided to use a block cipher because they can deal better with data chunks of known length and this is the case for our protocol, although at the moment the length of the messages is unknown. Furthermore, tampering the message is easily detected and block ciphers provide better security.

To choose a block cipher, we have to look at different aspects of ciphers related to energy consumption. Strydis et al. [41] evaluate a large number of symmetric block ciphers in terms of various metrics to determine their applicability for biomedical applications as implantable medical devices. They use XTREM [7] as a performance and power simulator for Intel's XScale embedded processor for testing different properties of ciphers. XTREM has been selected for its high precision in modeling the performance and power of the Intel XScale core and because of its straight-forward functionality. XScale is a low-power processor with aggresive power-measurement features. XTREM models this hardware with high accuracy.

Strydis et al. consider average power consumption, peak power consumption, program size, total energy costs, computation overhead and encryption rate. Some of these measurements are done for plaintexts of 1KB and 10KB for some comparisons. Power consumption was not affected by these differences. Besides average power consumption, peak power consumption is relevant since a ciper with a given average power consumption may be unable to deliver the required output at agiven point in time if the ciper presents peak power values which are largely deviating from its average power needs.

The results of these tests for the three best performing ciphers are presented in Table 10. These are MISTY1, IDEA and RC6 and they all use a key size of 128 bits and a sufficient equal security margin [41]. MISTY1 is a block cipher designed in 1995 for Mitsubishi Electric. This cipher uses block of 64 bits, has a number of rounds that is a multiple of 4 and uses a Feistel network for its structure. IDEA is a symmetric-key block ciper designed by ETH Zurich in 1991. It has a block size of 64 bits, has 8.5

rounds, uses the Lai-Massey scheme and is currently part of the OpenPGP standard. RC6 is a symmetric key block cipher developed for the AES competition. It uses block sizes of 128 bits, has 20 rounds and has a Feistel network structure.

Considering the pacemaker, we can state that for this device power consumption and encryption costs are the most important issues. We see that the power consumption is the smallest for the IDEA cipher and in case of the average power consumption, it is remarkably lower than for MISTY1 and RC6. Looking at the energy costs, we see that RC6 scores the best, then IDEA and then MISTY1. When we would consider all the parameters with equal weight, then overall RC6 would get the highest score for 4 out of 6 parameters and it would score the best. However, considering the actual values together with the importance of the parameters, the blockcipher IDEA performs the best for the pacemaker out of these three ciphers.

7.7 DISCUSSION

The solution presented in this chapter provides secure communication between the programmer and the pacemaker. It prevents eavesdropping and MITM attacks, as proven with ProVerif and by the construction of the secure communication protocol. Because of the way the secure channel is constructed and the use of the session key in this secure channel, an attacker is not able to intercept or modify a message.

At the moment it is not sure if any hardware modifications on the pacemaker are required. We expect that this will be necessary, especially for some older types of pacemakers, because they are not designed with the goal of implementing cryptographic mechanisms. For this reason, software updates are probably not enough to implement this solution, while maintaining the current lifetime of the pacemaker. This means that a new pacemaker has to be designed, tested and implanted in bodies of the patients. The designing and testing phase will take some time, but this is not comparable to the time it takes to replace all the pacemakers in the bodies of the patients. This can take up to 20 years, because it is undesirable and expensive to change all pacemakers at once. Furthermore, standards need to be set for different manufacturers in the communication protocol to make this solution work.

In this new solution, DoS attacks are not prevented and they are still a large risk, since these attacks can have battery draining as a consequence. Because of the presented protocol, the first step in the mutual authentication phase costs a lot of energy, as we will see in the next chapter. After receiving the first message, a key has to be generated by the pacemaker and needs to be compared with the receiving message and this step cannot be avoided at the moment. For now we assume that after receiving three messages that cannot be decrypted, the pacemaker goes into time out mode for a certain period of time, for example 10 minutes, and refuses all the incoming messages.

8 KEY MANAGEMENT

This chapter describes the key management we developed for the solutions presented in Chapter 6 and Chapter 7. This is done by using the key management lifecycle presented by Posea [33], which is drawn in Figure 16. The following stages are important in this process: (1) key generation, (2) key storage, (3) key distribution, (4) key usage, (5) key change and (6) key expiration, archival and destruction. For both solutions, each of these stages is discussed. The main focus is on the key generation stage of the shared secret based solution, since we developed a new way to generate dynamic symmetric keys for medical devices.

8.1 PROXY-BASED SOLUTION

In our architecture, the programmer, the telemonitoring system and the proxy need a public-private key pair to enable secure communication between them. In this section, the key management for the proxy-based solution is presented according to the six stages.

Key generation

Since the Needham-Schroeder-Lowe protocol suggests RSA keys for asymmetric encryption, we decided to use them in this system. During the first treatment, the programmer generates three asymmetric key pairs: one key pair for the programmer, one for the telemonitoring system and one for the proxy. This is done by using the Diffie-Hellman Key Agreement Method presented in a related RFC [37]. Details about this method can be found in this RFC.

Key storage

The key storage process is presented in Figure 17. The telemonitoring system and the programmer are connected to the back-end system. For this reason, the proxy is drawn seperated from the other devices. After generating the three key pairs in (1) for the programmer, the telemonitoring system and the proxy, the programmer sends these key pairs to other devices in (2). The key pair for the proxy is sent to the proxy and the public key for the proxy is also sent to the back-end system. For the proxy, the key is saved in a hardware security module (HSM). This compouting component manages digital keys for authentication. In (2), the programmer also sends the key pairs for the telemonitoring system and the programmer to the back-end system of the hospital. Furthermore, the programmer sends the public key from the programmer and the telemonitoring system to the proxy. This happens in plaintext, since these keys are public and can be read by everyone. When these key pairs are sent, the programmer removes all the created pairs in (3).

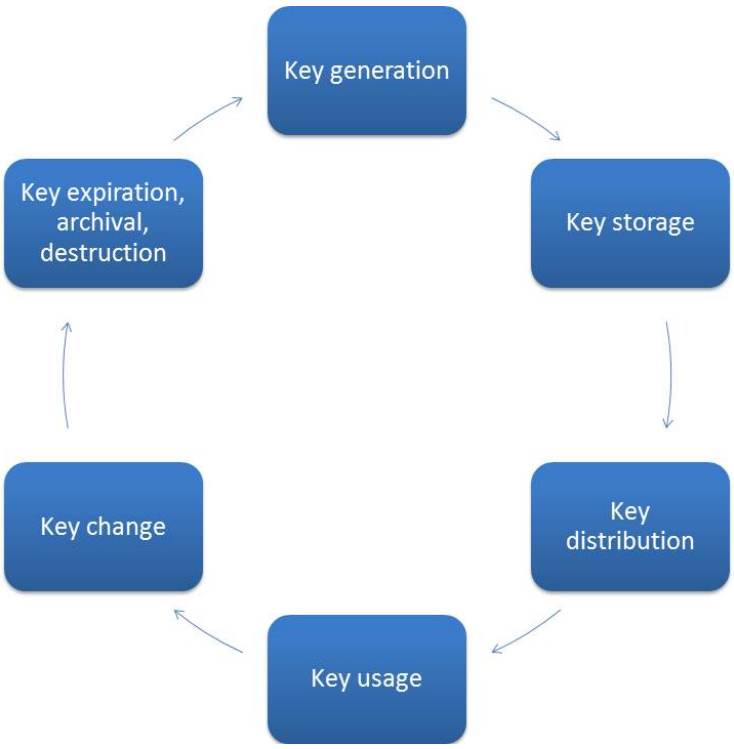

Figure 16: The key management life cycle.

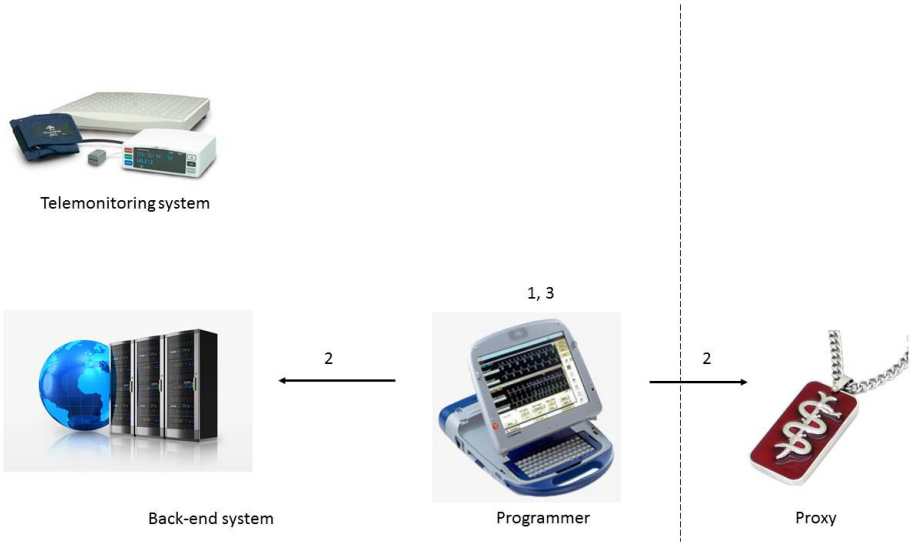

Figure 17: The process of key storage.

Legend
1: The programmer generates the key pairs.
2: The programmer sends the key pairs to the back-end system and the proxy.
3: The programmer removes the key pairs.

Key distribution

Since for each patient, the devices that are involved in secure communication are the programmer, the telemonitoring system and the proxy, the doctor is able to do the key distribution manually. There is no reason to automate this process. This is done by a pacemaker technician, since we assumed that we trust the medical staff in the hospital. For the distribution from the programmer to the proxy a distance bounding protocol is used as described before. The key for the telemonitoring can be sent encrypted from the back-end system to the telemonitoring system over the Internet with a secure channel. The private key of the telemonitoring system only has to be on the system during a monitoring session. At the start of this session, the telemonitoring system can connect with the back-end system and temporarily get the key. After the session, the telemonitoring system should erase the private key.

Key usage

To avoid key disclosure while the system is in use, it is good to perform all cryptographic operations in a physically isolated dedicated environment, offered by for example an HSM. For this reason, we advise that the proxy, programmer and telemonitoring system have such an HSM. Further research has to be done to determine what type of HSM fits in a proxy, since the proxy has size limits. But at least the programmer and the telemonitoring system can contain such a module to perform all cryptographic operations during sessions with the pacemaker and prevent malware to influence the encrypted communication.

Key change

In case a key is compromised or expired, a new key pair has to be generated. For a programmer and a telemonitoring system, this can be done immediately by the programmer and keys can be directly transferred to the back-end system. But for the pacemaker, the patient has to come to the hospital where the programmer generates a new key pair for his or her proxy. When the proxy is lost, stolen or broken, it has to be replaced by a new one. In this case, the programmer destroys the old key pair and generate a new key pair. The process of key management of the keys is then repeated. When a patient is treated in another hospital, it is considered as an emergency situation. The proxy is temporarily removed from the patient and there is no key exchange between the programmer and the proxy.

Key expiration, archival and destruction

The expiration date for the key pair is set to two years, according to the requirements from NIST [3]. After this period of time, the whole key management cycle will be repeated. But first the expired key pair need to be destructed. There is no need to archive them, since these expired keys will not play a role in further session anymore.

Notation	
K	Secret
C	Counter
X	Time step in seconds
To	Unix time to start counting time steps

Table 11: The notation for a Time-Based One-Time Password.

8.2 SHARED SECRET BASED SOLUTION

In this section, the key management for the shared secret based solution is presented according to the six stages.

Key generation

In the proxy-based solution, the system uses static keys that are replaced every two years. The use of static keys is undesirable because of possible disclosure of the keys. The use of dynamic keys is prefered and is the case in the shared secret based solution.

As a basis, the Diffie Hellman key agreement scheme is used between the programmer and the pacemaker to establish a shared secret together. This secret is then part of the input to generate the dynamic symmetric key that is used by the programmer, pacemaker and telemonitoring system to communicate. This symmetric key is based on the Time-Based One-Time Password (TOTP) [29]. This is a password that can be used only once and is based on a shared secret and the current time. The secret and the timer are the input for a hash function, from which a key is generated. The TOTP is based on a HMAC-Based One-Time Password (HOTP) [28], which uses a counter instead of a timer. The use of a counter is less secure and a timer is more practical for this solution. For computing a TOTP, the notation in Table 11 is used.

The TOTP is computed according to the following equation:

$$TOTP = HOTP(K, T) = Truncate(HMAC - SHA1(K, T))$$

T and HMAC are defined as follows:

$$T = \frac{Current\ Unixtime - T0}{X}$$

$$HMAC(K, T) = H((K\ XOR\ opad)|H((K\ XOR\ ipad)|T)),$$

where opad and ipad stand for outer padding and inner padding. Both are one-block-long hexadecimal constants, where opad differs from ipad. The values do not influence the solution in a certain way, but are there to provide different inputs for the hash function. The hash function that is used in this HMAC is a Sha-1 hash. Although Sha-1 has a weakness with regards to collisions, this does not affect the security of the HMAC. HMAC resistance does not rely on resistance to collisions [28].

The Truncate function is able to convert an HMAC-SHA1 value into an HOTP value. A Sha-1 hash always has an output of 160 bit, so the HMAC-SHA1 algorithm also outputs a 160 bit value. In general, the purpose is to extract a 32-bit key from this 160-bit key in the TOTP algorithm. In our case, the Truncate function needs to convert the 160-bit TOTP value into a 128-bit value, since we want to use the TOTP as a key for the encryption cipher IDEA as presented in Chapter 7.

The time step X should be set to a suitable value depending on the performace of the system. This depends on the time it takes to create a TOTP for the programmer, the time between sending the TOTP from the programmer to the pacemaker and the time it takes for the pacemaker to generate the TOTP and do the check by trying to decrypt the message. Depending on how fast this process goes, the timer has to be set. This time should be long enough to do all the checks, but not too long since in case of an error a new TOTP should be created and a system should not accept keys that are not fresh enough.

Current TOTP tokens are able to generate about 14.000 TOTPs and they have a lifetime between 5 to 7 years. For now, this lifetime is too short, since it should exceed at least the average lifetime of the pacemaker. Current TOTP tokens are relatively cheap ($65) and we can expect that current pacemakers have better batteries, since their costs are between $35.000 and $45.000. We expect that the generation of 14.000 TOTPs is more than enough, since only 1 TOTP is needed per treatment session. For the complete lifetime of the pacemaker, this is about 20-40 TOTPs in the hospital. Assuming a patient has one telemonitoring session per day, after 20 years only half of the 14000 TOTPS are used. Even with a few DoS attacks per day, which is unexpectable, there would be enough TOTPs for the lifetime of the pacemaker.

Key storage

Since we use dynamic keys, the TOTP is not stored, but only the secret, created by the pacemaker and the programmer, is stored. The process of key storage is the same as in the proxy-based solution and presented in Figure 17. The only difference is that the proxy in this scheme is replaced by the pacemaker. Furthermore, not the key pairs but the secret is sent from the programmer to the pacemaker and the back-end system. There is one extra step, in which the clock of the pacemaker is synchronized with the clock of the back-end system. Now with both the secret and the timer, the pacemaker, programmer and proxy are able to generate the symmetric key that is used for the communication. This key is stored temporarily and can be removed directly after a treatment session from all devices.

Key distribution

As mentioned in the key generation part, the secret is distributed according to the Diffie Hellman key agreement scheme [37]. In this scheme, both parties agree on two public values: a prime p and a base g. Then the programmer chooses a secret value a and the pacemaker a secret value b. Now the programmer sends $g^a \mod p$ to the pacemaker and the pacemaker computes $(g^a)^b \mod p$ from this. The pacemaker sends $g^b \mod p$ to the programmer and the programmer computes $(g^b)^a \mod p$ out of this. This final value that they both compute, is the shared secret. The only

restriction is that the secret is 160 bits. This distribution of the secret can happen over an insecure channel, since both p and g are public.

The timer of the pacemaker can be synchronized with the back-end system during the first treatment.

Key usage

To avoid key disclosure while the system is in use, it is good to perform all cryptographic operations in a physically isolated dedicated environment. We propose that this is done by an HSM. So the pacemaker needs to contain a HSM or a component with similar capabilities. Furthermore, the programmer and the telemonitoring system needs such an HSM, as well.

Key change

In case of a disclosure of the secret, a new secret has to be generated between the pacemaker and the programmer. As a consequence, the patient has to visit the hospital for this process. In case of an emergency, a programmer from another hospital may synchronize with the pacemaker. The details of this process are explained in Chapter 9. Then a new secret is established which works together with the correct time from the back-end system of the hospital. This secret and time can be used as a one time key and are stored only during the session on the pacemaker or the back-end system, since it is unlikely that the patient will come back to this same hospital for more treatments. In this rare case, the key generation process can happen again by generating a new secret. The secret is valid for a short period of time which should be long enough for the complete treatment.

Key expiration, archival and destruction

Secrets expire after 2 years of use. After this period, the pacemaker and programmer need to establish a new secret with the Diffie Hellman key agreement scheme. The old secret can be destructed, when a new secret is generated. Secrets created in emergencies are removed from the pacemaker and the programmer directly after the treatment, since they do not need to be stored for a longer period of time.

8.3 CONSTRUCTION OF THE SESSION KEY AND THE MAC

For the secure channel of the shared secret based solution, a session key and a MAC are necessary. These are created for every session the pacemaker establishes with a programmer or a telemonitoring system and are removed from the pacemaker at the end of each session. Computations with the session key are done in the HSM of the devices. The session key is exchanged at the end of the protocol by the programmer. It is generated according to the key generation algorithm of the IDEA cipher, which is not further explained here.

The MAC that is used for integrity of the messages during the secure communication protocol is made by the use of a universal hash function. For this it uses

the algorithm UMAC as propsed in RFC 4418 [24]. This is a very fast and relatively lightweight algorithm to generate MACs and possible to combine with various type of block ciphers. It relies on addition of 32-bit and 64-numbers and multiplication of 32-bit numbers. These operations are well-supported by contemporary machines. The only cryptographic component that is used, is the block cipher for generating pseudorandom pads and internal key material. We advise to use it in combination with the IDEA cipher. The default block cipher is AES, but this block cipher is relatively more heavy than IDEA.

9 EMERGENCY ACCESS SOLUTIONS

In this chapter, we present the emergency access approaches for the systems presented in Chapter 6 and Chapter 7. For the proxy-based system, we present one solution that fits all the requirements described in Chapter 6. This solution will always work, because of the design of the system and the possibility to remove layers of security easily. For the shared secret based system presented in Chapter 7, we present our new solution in Section 9.2, where the pacemaker functions as an emergency detector. Our work is the first in giving concrete parameters and a method of detecting emergencies. However, this solution has the limitation that not all emergency situations can be recognized and covered. For this reason, some alternative solutions are presented that can be used to authenticate to the system. They increase the success of accessing the pacemaker in case of an emergency.

9.1 EMERGENCY SOLUTION FOR THE PROXY-BASED SYSTEM

During an emergency, a patient visits a hospital that is close by. Depending on his or her location, this can be A) the hospital he or she regularly visits or B) a hospital in another city or even in another country where he or she has never been before. In case A, the pacemaker can be accessed in the same way as during regular treatments and the system is the same as in Figure 10. In case B, the proxy has to communicate with a programmer it does not know. Both the proxy and the programmer do not have each others public key. It takes time to exchange public keys in a safe way, which is not preferred in an urgent situation. We remark that in case of an emergency, safety of the patient is more important than security of the communication. In this situation, the proxy is temporarily removed. The programmer and the pacemaker communicate as systems that are in use nowadays, without encryption of the communication. After the treatment, the proxy can be reattached to the body of the patient and the pacemaker is protected again.

The system architecture for this situation is presented in Figure 18, where only a programmer and a pacemaker are involved. The logs of the treatment are stored in the hospital where the patient is treated. We assume that the patient should transfer the logs from this data to the hospital where he or she is regularly treated. In general, this information is not exchanged between the hospitals without help of the patient. Furthermore, during the next regular treatment, the cardiologist should check the logs from the emergency that are stored in the pacemaker and check if the settings of the pacemaker are correct.

9.2 EMERGENCY SOLUTION FOR THE SHARED SECRET BASED SYSTEM

In this section, we present a new solution for emergency access to the pacemaker where the pacemaker takes the role of an emergency detector. Since it is difficult to guarantee a detection system that covers 100% of the emergencies, the use of a second

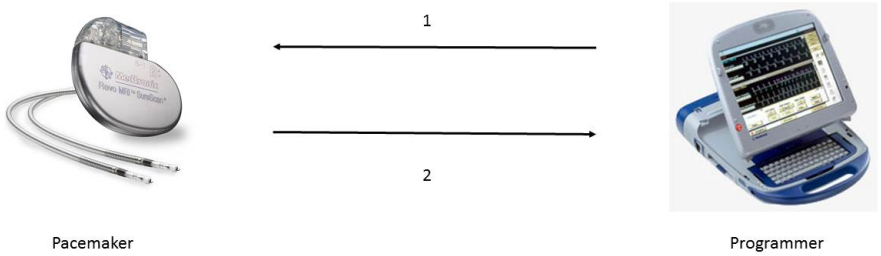

Figure 18: The architecture in an emergency situation.

Legend
1: The programmer configures the pacemaker and requests data stored on the pacemaker.
2: The pacemaker sends the data to the programmer.

way to authenticate to the system might be needed. For this reason, some alternatives are presented as well.

9.2.1 *The pacemaker as an emergency detector*

In the shared secret based system presented in Chapter 7, security is not an additional layer on a removable device as it is for the proxy-based system. As a consequence, there is no easy way to access the pacemaker in case of an emergency when a patient is not near to the hospital where he or she is regularly treated. This has to do with the fact that the elements that are necessary to generate keys are only stored on the pacemaker and the back-end system of the hospital where the patient is regularly treated. During an emergency, the pacemaker should be immediately accessible by a medical staff member in any part of the world. Although we want to prevent unauthorized access to the pacemaker even in case of an emergency, safety of the patient goes above all. No time can be lost by retrieving the correct key. In our system, we create the possibility for a pacemaker to synchronize again with a programmer during an emergency by establishing a new secret and follow the process of key management as described in Chapter 8.

The principle is that access control is based on the condition of the heart and that the pacemaker can operate in two modes: a regular mode and an emergency mode. When the heart is in a condition where the pacemaker is able to support it, there is no immediate access needed by other parties. This is the regular mode and security mechanisms are provided. In case the heart functions abnormally and the pacemaker is not able to correct this behavior, help by medical staff is needed. Now the pacemaker goes into emergency mode. This mode is explained in the remainder

of this section. The state of the heart can be measured by various parameters that depend on heart rate and rhythm. By setting acceptance values for these parameters, it is possible to recognize states in which the patient might be in possible danger. The system makes it possible to enter emergency mode if one or more of the acceptance values is exceeded.

Hei et al. [23] proposed a detection system in the pacemaker that takes the heart rate as a parameter and only considers an upper limit of 140 beats per minute. An explanation of this specific number is missing even as a reason why they assume a lower limit for the heart rate is not necessary. Since the heart rate depends on various aspects, e.g. age, this limit should not be set to a fixed value applicable to everyone. Furthermore, we think that taking only the heart rate as a parameter is not sufficient, since this parameter does not cover most serious heart problems. For this reason, we argue that more parameters should be involved in the emergency detection system to do proper checks. These parameters are discussed next.

Parameters

The parameters are based on a dangerous emergency situation, where the pacemaker is not able recover the normal behavior of the heart, namely during a heart attack. In general, a heart attack can be diagnosed in three ways according to Thaler [42]. These are:

1. by examining the physical state of the patient and by looking at the symptoms,
2. by examining the blood level and its cardiac enzymes,
3. by considering the electrocardiogram (ECG).

Methods 1 and 2 are measurable after the heart attack happened and require specific equipment. Furthermore, these methods are not always accurate enough. With method 3, it is possible to reveal the correct diagnosis in case of most attacks and an ECG makes it possible to detect the earliest changes in the rhythm of the heart. An ECG is a recording of the electrical activity of the heart. The pattern of an ECG looks as presented in Figure 19. Here the vertical line represents the Voltage and the horizontal line represents the time. Over time, the pattern in Figure 19 repeats itself. The waves and complexes in the figure are caused by the behavior of the heart atria and ventricles and represent the difference in voltage level during a heartbeat. During a heart attack, an ECG evolves through three stages as shown in Figure 20:

1. T wave peaking (a) followed by T wave inversion (b)
2. ST segment elevation (c)
3. Appearance of new Q waves (d)

The pacemaker can stimulate the behavior of the atrial and ventricular pacing and the device is able to determine limits for the voltage at certain points of a heartbeat. If limits are exceeded in the same order as they appear in a heart attack, the system should go into emergency mode. Depending on the specific heart rate of a patient, the doctor needs to set these limits with care for every patient individually.

From this pattern, we can define the following parameters with ranges.

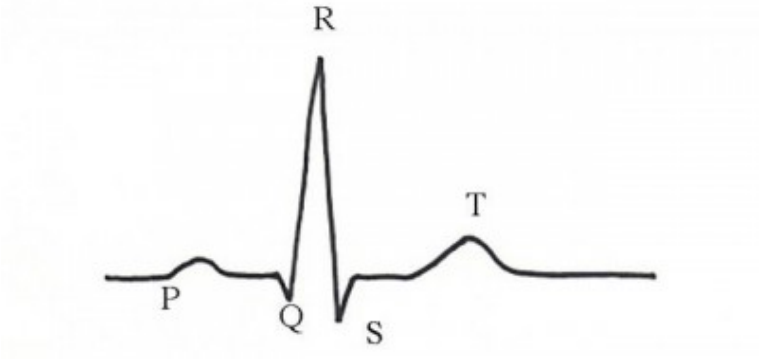

Figure 19: A PQRST complex of an ECG.

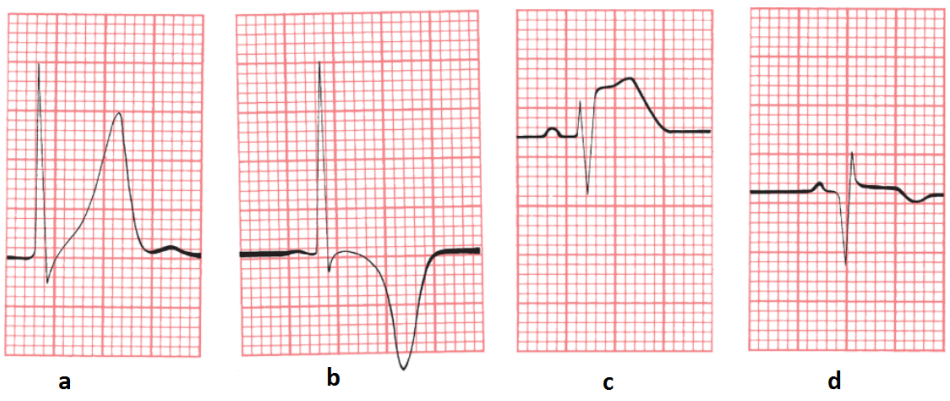

Figure 20: Changes in the PQRST complex.

- The height of the T wave. This parameter can have a positive Voltage value or a negative value. By defining an upper limit and a lower limit for the T wave, we can check if this limit is exceeded. Both limits are determined by the doctor for each patient, since it depends on the height of T waves in a regular condition. It is reasonable that a certain percentage is set that the T wave is allowed to differ from the regular situation.

- The ST segment elevation. As a consequence of a T wave peak and a T wave inversion, the ST segment will elevate. Now we can measure if it elevates compared to the previous heart beat. This parameter is a boolean with the value True if ST segment elevation happens and False if it does not happen.

- A new Q wave. After ST segment elevation, a new Q wave can appear. This parameter is a boolean with value True if a new Q wave appears and False if no new Q wave appears.

Another parameter that has to be measured, is the heart rate itself. An upper limit and lower limit should be set for this parameter, since in case of an extremely low or extremely high heart rate a medical staff member should be able to access the pacemaker directly. This is done by the doctor for each patient seperately and is determined by considering a certain percentage that the heart rate may differ from the heart rate in a regular situation.

As shortly explained, the system enters an emergency mode when it discovers an emergency situation. To be more specific, the emergency mode is reached, when the following logical formula gives "True" as an output:

$$\begin{aligned} &\mathit{emergency} = \mathit{heart\,rate} > \mathit{upper\,limit1} \vee \mathit{heart\,rate} < \mathit{lower\,limit1} \vee \\ &(\mathrm{T}\,\mathit{wave} > \mathit{upper\,limit2} \wedge \mathrm{T}\,\mathit{wave} < \mathit{lower\,limit2} \\ &\wedge \mathit{Appearance\,of\,ST\,segment\,elevation} \wedge \mathit{New\,Q\,wave}) \end{aligned}$$

This check can be done every minute for example to control the heart rate and to recognize patterns of a possible emergency situation.

Now that the parameters are defined, we can explain how the system functions in emergency mode with the help of a finite state diagram.

Finite state diagram

The finite state diagram related to these states is presented in Figure 21. As explained before, there are two modes of the system: an emergency mode and a non-emergency mode. In the non-emergency mode, the pacemaker can only communicate with devices that are able to authenticate to the programmer. But when a pacemaker notices an emergency, the system should go into emergency mode. In this mode, the pacemaker can synchronize with a programmer and establish a secure connection. The states related to the emergency mode are the following.

- Protected: in this state, the pacemaker is only accesible by authentication and the communication is encrypted. It can only communicate with the programmer that is meant for regular treatments.

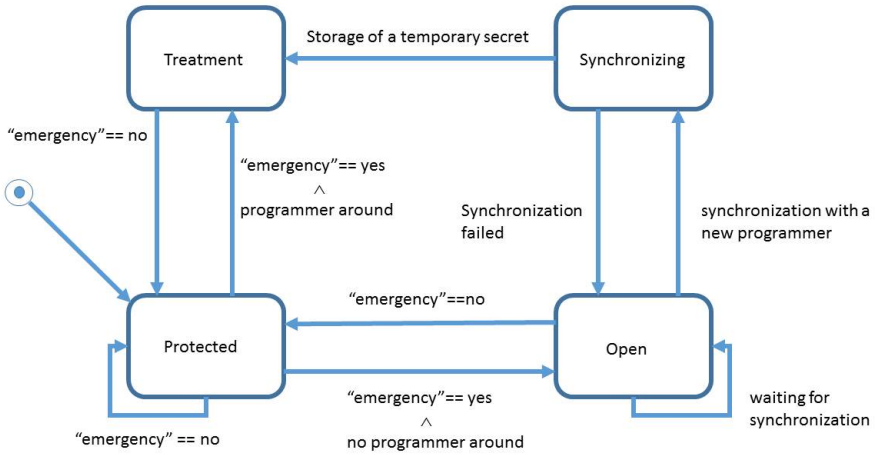

Figure 21: A finite state diagram.

- Open: here, the pacemaker is open for all programmers and can synchronize with every programmer.
- Synchronizing: in this state, the pacemaker is synchronizing with a programmer.
- Treatment: here, treatment is provided for the patient carrying the pacemaker by using a programmer.

The protected state is the entering state of the system. From this state, there are three possibilites. Without an emergency, the pacemaker stays into this state. When the system recognizes an emergency based on the limits of parameters as discussed in the previous section, it can go in two different states. If the patient is brought to the hospital where he or she is regularly treated, the system goes into treatment state and the programmer can directly communicate with the pacemaker. If the patient enters another hospital or there is no programmer around, the system goes into state 'open'. The pacemaker stays in this state until the emergency situation is over or when a programmer appears. When the emergency is over before help is available, the pacemaker goes back into the protected state. Otherwise, it waits to synchronize with a programmer again as is mentioned in Chapter 8 and go into synchronizing state. During the 'synchronizing' state, a new secret is created and shared between the programmer and the pacemaker and the time of the back-end system is used once to generate the key and to synchronize with the programmer to provide secure communication, even during an emergency. If synchronizing fails, this process can be repeated with the same programmer or with another programmer. When synchronizing succeeds, the system goes into treatment state and treatment can be provided. After the treatment, the system goes back to the protected state. Note that because of this mechanism, communication between the programmer and the pacemaker is still encrypted in case of an emergency.

False positives and negatives

It is important to minimize false positives and false negatives in this solution. In this system, false positives are situations in which the pacemaker goes in emergency mode, when there is no emergency. This can happen when the heart rate is high or low or other parameters exceed a limit, but the patient is not experiencing a difficult situation. Now resynchronization is possible in a situation where this is not necessary. In case an attacker is around, he or she is able to access and connect. The number of false positives should be minimized by choosing the limits for the parameters very carefully. During the occurence of a false negative, the patient is in an emergency, but the device does not recognize the emergency. Therefore, access is necessary, but the pacemaker does not recognize an emergency mode. This can be the case when there is an emergency because of non-related heart problems. This is a problem, since the pacemaker has to be turned off during some surgeries. However, at the moment there is no easy way to stop the pacemaker from functioning. And this is also a good property, because it should not be easy for an attacker to turn of the pacemaker.

9.2.2 *Alternatives*

There are several possible emergency solutions that solve the problem of false positives and false negatives, since it is not dependent on measurements, but on external components and systems. In all solutions there is a better way to secure that only medical staff members access the pacemaker in case of an emergency. However, it might take too much time to access the pacemaker, since the medical staff is depending on other parties or systems.

Smartcard

The first solution is that the patient carries a paper card or smartcard in his or her wallet with an access code to access the pacemaker in case of an emergency. These cards can be standardized and can be a replacement of cards that patients are carrying in their wallet with the type of the pacemaker and the manufacturer. In this case, access is guaranteed by sending the code on the smartcard from the programmer to the pacemaker, but it is also possible that an attacker steals the wallet with the card and gets access in the same way. Furthermore, the card has the property that it can be lost, stolen or broken and that in case of an emergency, it is not available. In this case, access would not be possible. This is the main reason why an external object or device will never be an optimal emergency solution.

Alarm line

Another solution is the use of an alarm line system. Each country already has an alarm line system, this service can be added to this system. This alarm line can have access to a database with all the secrets and timers and will be able to generate the TOTP. The alarm line is unique for each country. In case of an emergency, the medical staff has to call to the alarm line of the country where the patient is from.

The process can happen for example as follows: The doctor calls the alarm number by phone and the insurance company checks this caller id. If it appears in a database

with phone numbers from a hospital, it sends a key to access the pacemaker to the hospital by fax. By responding this call with a fax, we can be sure this key arrives at the hospital. It does not make sense if an attacker spoofs the caller id, since he or she will never receive the key from the fax. Now this key can be used to encrypt the message from the programmer to the pacemaker to start authentication.

Positive aspects is the fact that there are multiple verifications by the caller id and the fax, to be sure it is a medical staff member and no attacker. Furthermore, it needs a local database for each country at the alarm service. A negative aspect is that the solution may take a large amount of time. It takes time to call, to search in the database and to pass the TOTP. Furthermore, since the TOTP is a 128-bit key, it is difficult to give this by fax. Another possibility is to send it over the Internet, but this requires strong encryption of the data sent.

Global key management system

The ideal solution to retrieve emergency access only for authorized entities and for other problems (e.g. an electronic patient health record) would be an electronic global key management system. When all hospitals worldwide have access to this global system, it is possible to define who has access at what moments. With such a system, it would be easy to ask for keys in any part of the world, at each hospital that is connected to the database. By looking for the identity of the patient, the secret and the timer could be immediately retrieved and the key could be determined. This can be done in a few seconds and would only depend on the availability of the database. There are not that many disadvantages to this solution, the only thing is that at this point in time, it is not realistic to have such a database. It is already hard enough to have a national patient health record system, a global database would be much harder, because standards related to database systems, but also access control rights need to be set worldwide.

10 CONCLUSIONS & FUTURE WORK

In this chapter, we present the conclusions of the work performed for this thesis. Furthermore we discuss possible directions for future work related to implementation details of the solution and emergency access solutions.

10.1 CONCLUSION

In this thesis, we tried and answered the research question *"How can we define a system that provides confidentiality and integrity of sensitive information during the communication between a pacemaker and a legitimate programmer, while ensuring availability of information in case of an emergency?"*. The goal is to find an optimal system that fulfils the requirements and see what possible emergency access solutions are applicable. This research question is split up in two subquestions namely:

1. *What system can provide confidentiality and integrity of sensitive information during communication between a pacemaker and a legitimate programmer?*
2. *How can emergency access be assured in this system?*

To answer the first subquestion, we had the following approach. We defined a list of requirements for an optimal solution. Considering these requirements, it was immediately clear that it was hard to meet different requirements at the same time. The largest challenge was the fact that certain properties could not be assured without having knowledge about the exact components that were in the pacemaker. Since pacemakers are property of the manufacturers and they are each others competitors, manufacturers were not willing to give details about the architecture and communication protocols of the devices. This led to the decision to come up with two different types of solutions: one that is directly applicable on the current pacemaker and second that may require changes in the pacemaker design, but fits more requirements.

The solution that is directly applicable, is a proxy-based solution based on The Shield [17]. From the list of requirements we formulated for this solution, we decided to use the Needham-Schroeder-Lowe protocol to guarantee secure communication between the programmer, proxy and pacemaker. Although most requirements are met, the use of a proxy as a solution is considered as not to be a desirable solution. Since the external device can be lost, stolen or broken, the additional security layer is not always guaranteed. For this reason, a shared secret based solution is presented without a proxy. For this solution, we defined a list of requirements and we designed a mutual authentication protocol and a secure channel. This communication protocol is based on a combination of the lightweight block cipher IDEA and the use of Time-Based One-Time Passwords (TOTP) as an encryption key. This key management method related to the TOTP is a new way to make use of a dynamic key solution instead of the use of static keys which is frequently suggested.

The second subquestion is answered by presenting a new system in which the pacemaker behaves as an emergency detector. Our system constantly measures var-

ious parameters of the heart and when one or more of these parameters exceed a limit, the pacemaker goes into emergency mode. In this mode, the pacemaker is able to synchronize again with another programmer. This type of solution avoids the use of a central key management scheme or an external wearable object that the patient needs to carry to access the pacemaker in case of an emergency. Our work is the first in giving concrete parameters and a method of detecting emergencies.

10.2 FUTURE WORK

This work addresses future research in two main areas: the implementation details of the solution and a specification of the emergency access solution.

10.2.1 *Implementation details*

To implement the solution presented in Chapter 7, the current architecture of the pacemaker needs to be studied carefully. Research has to be done to investigate if the necessary components are available and what will be the effect of the security mechanisms on the performance and the lifetime of the device. The needed type and capacity of the battery has to be determined together with the type of processor and other relevant components. With these estimations a prototype needs to be build to measure the performance. In this research, it was not possible to make a start on this part, since manufacturers were not willing to participate. Since at the moment hardly any security mechanisms are implemented in medical devices we advise manufacturers to work together with researchers who studied security aspects to guarantee the security properties of medical devices in the future. Furthermore, standardising parts of the communication process and the capabilities of the components of IMDs would make it more easy to address security related difficulties. Organizations that define standards in this area could play an important role here.

10.2.2 *Emergency access*

For the emergency access solution, where the pacemaker plays the role of an emergency detector, it is important to develop a way to determine the limits of the parameters in the emergency detection system. Here, the number of false positives and false negatives should be as small as possible. To complete this solution, it should be determined how to respond in case of a false negative. In our presented solution, this is an open problem. An option is to introduce a second backdoor in the system to cover most false positives, but it is hard to cover 100% of the cases.

BIBLIOGRAPHY

[1] M. Abadi and A. D. Gordon. A calculus for cryptographic protocols: The spi calculus. December 1996.

[2] D. Abraham, G. Dolan, G. Double, and J. Stevens. Transaction security system. *IBM Systems Journal*, 30(2):206–229, 1991.

[3] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. Recommendation for key management - part 1: General. *NIST Special Publication 800-57*, 2007.

[4] C. Beck, D. Masny, W. Geiselmann, and G. Bretthauer. Block cipher based security for severely resource-constrained implantable medical devices. In *ISABEL '11 Proceedings of the 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies*, October 2011.

[5] S. Brands and D. Chaum. Distance-bounding protocols (extended abstract). In *EUROCRYPT93, Lecture Notes in Computer Science 765*, pages 344–359. Springer-Verlag, 1993.

[6] W. Burleson, S. S. Clark, B. Ransford, and K. Fu. Design challenges for secure implantable medical devices. In *DAC '12 Proceedings of the 49th Annual Design Automation Conference*, pages 12–17, June 2012.

[7] G. Contreras, M. Martonosi, J. Peng, R. Ju, and G. Lueh. Xtrem: A power simulator for the intel xscale core. *LCTES'04*, pages 115–125, 2004.

[8] Council Directive. 90/395/eec. *Official Journal of the European Communities*, June 1990.

[9] T. Denning, A. Borning, B. Friedman, B. T. Gill, T. Kohno, and W. Maisel. Patients, pacemakers, and implantable defibrillators: Human values and security for wireless implantable medical devices. In *CHI '10, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 917–926, 2010.

[10] T. Denning, K. Fu, and T. Kohno. Absence makes the heart grow fonder: New directions for implantable medical device security. In *HOTSEC'08 Proceedings of the 3rd conference on Hot topics in security*, 2008.

[11] EBAY. Pacemaker programmer, http://www.ebay.com/bhp/pacemaker-programmer, March 2014.

[12] N. Ellouze, M. Allouche, H. B. Ahmed, S. Rekhis, and N. Boudriga. Securing implantable cardiac medical devices: Use of radio frequency energy harvesting. In *TrustED '13 Proceedings of the 3rd international workshop on Trustworthy embedded devices*, pages 35–42, November 2013.

[13] N. Ellouze, M. Allouche, H. B. Ahmed, S. Rekhis, and N. Boudriga. Security of implantable medical devices: limits, requirements, and proposals. *Security and Communication Networks Journal*, November 2013.

[14] Ettus Research. Usrp products, https://www.ettus.com/product, March 2014.

[15] Federal Communications Commission. Mics medical implant communication services. *FCC 47CFR95.601-95.673 Subpart E/I Rules for MedRadio Services*, 2013.

[16] A. Ferreira, R. Cruz-Correia, L. Antunes, P. Farinha, E. Oliveira-Palhares, D. W. Chadwick, and A. Costa-Pereira. How to break access control in a controlled manner. In *CBMS '06 Proceedings of the 19th IEEE Symposium on Computer-Based Medical Systems*, pages 847–854, 2006.

[17] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu. They can hear your heartbeats: Non-invasive security for implantable medical devices. In *SIGCOMM '11 Proceedings of the ACM SIGCOMM 2011 conference*, pages 2–13, 2011.

[18] Government Accountability Office. *Medical Devices: FDA Should Expand Its Consideration of Information Security for Certain Types of Devices*, August 2012.

[19] S. A. P. Haddad, R. P. M. Houen, and W. A. Serdijn. The evolution of pacemakers. *Engineering in Medicine and Biology Magazine, IEEE*, 25(3), May/June 2006.

[20] D. Halperin, T. Heydt-Benjamin, B. Ransford, and S. Clark. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Security and Privacy (SP), 2008 IEEE Symposium*, pages 129–142, May 2008.

[21] Heart Rhythm Society. Pacemakers, http://www.hrsonline.org/patient-resources/patient-information-sheets#axzz2wwarxdxr, March 2014.

[22] X. Hei and X. Du. Biometric-based two-level secure access control for implantable medical devices during emergencies. In *INFOCOM, 2011 Proceedings IEEE*, pages 346 –350, April 2011.

[23] X. Hei, X. Du, J. Wu, and F. Hu. Defending resource depletion attacks on implantable medical devices. *Global Telecommunications Conference*, pages 1–5, December 2010.

[24] T. Krovetz. Umac: Message authentication code using universal hashing. *RFC 4418*, March 2006.

[25] U. Lakshmanadoss, A. Shah, and J. J. Daubert. Telemonitoring of the pacemaker. In *Modern Pacemakers - Present and Future*. InTech, Rijeka, 2011.

[26] B. Lanchet, B. Smyth, and V. Cheval. Proverif 1.88pl1: Automatic cryptographic protocol verifier. June 2014.

[27] G. Lowe. An attack on the needham-schroeder public key authentication protocol. *Information Processing Letters*, 56(3):131–136, November 1995.

[28] D. M'Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen. Hotp: An hmac-based one-time password algorithm. *RFC 4226*, December 2005.

[29] D. M'Raihi, S. Machani, M. Pei, and J. Rydell. Totp: Time-based one-time password algorithm. *RFC 6238*, May 2011.

[30] OWASP Attacks. https://www.owasp.org/index.php/category:attack. April 2014.

[31] OWASP Threat Agent. https://www.owasp.org/index.php/category:threat_agent. April 2014.

[32] OWASP Vulnerability. https://www.owasp.org/index.php/category:vulnerability. April 2014.

[33] S. Posea. *Renewal Periods for Cryptographic Keys*. 2012.

[34] V. Pournaghshband, M. Sarrafzadeh, and P. Reiher. Securing legacy mobile medical devices. *MobiHealth*, 2012.

[35] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, December 1978.

[36] K. B. Rasmussen, C. Castelluccia, T. Heydt-Benjamin, and S. Capkun. Proximity-based access control for implantable medical devices. In *CSS '09 Proceedings of the 16th ACM conference on Computer and communications security*, pages 410–419, November 2009.

[37] E. Rescorla. Diffie-hellman key agreement method. *RFC 2631*, June 1999.

[38] M. Rostami, W. Burleson, and F. K. A. Juels. Balancing security and utility in medical devices? In *Design Automation Conference (DAC), 2013 50th ACM/EDAC/IEEE*, pages 1–6, May 2013.

[39] M. Rostami, A. Juels, and F. Koushanfar. Heart-to-heart (h2h): Authentication for implanted medical devices. In *CCS '13 Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1099–1112, November 2013.

[40] C. Strydis, R. M. Seepers, P. Peris-Lopez, D. Siskos, and I. Sourdis. A system architecture, processor, and communication protocol for secure implants. *ACM Transactions on Architecture and Code Optimization*, 10(4), December 2013.

[41] C. Strydis, D. Zhu, and G.N.Gaydadjiev. Profiling of symmetric-encryption algorithms for a novel biomedical-implant architecture. *5th Conference on Cumputing Frontiers*, pages 231–240, May 2008.

[42] M. S. Thaler. *The Only EKG book you'll ever need*. Lippincott Williams & Wilkins, 2007.

[43] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li. Imdguard: Securing implantable medical devices with the external wearable guardian. In *INFOCOM, 2011 Proceedings IEEE*, pages 1862–1870, April 2011.

[44] M. Zhang, A. Raghunathan, and N. K. Jha. Medmon: Securing medical devices through wireless monitoring and anomaly detection. *IEEE Transactions on Biomedical Ciruits and Systems*, 7(6):871–881, December 2013.

[45] M. Zhang, A. Raghunathan, and N. K. Jha. Towards trustworthy medical devices and body area networks. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*, pages 1–6, May 2013.

APPENDIX A: PROVERIF CODE

A.1 INPUT

A.1.1 *Mutual authentication*

Listing 1: ProVerif Input Mutual Authentication

```
free c. (* public channel *)
free P,I.
fun sencrypt/2. (* symmetric encryption *)
reduc sdecrypt(sencrypt(x,y),y)=x. (* symmetric decryption *)

not symkey. (* secrecy assumption: symkey cannot be *)

private free secretK. (* secret k sent from P to I *)
query evinj:end(x,y,z) ==> evinj:begin(x,y,z).
query evinj:end2(x,y,z) ==> evinj:begin2(x,y,z).

let P =
        new Np;  (* nonce *)
        out(c, sencrypt((P, Np), symkey));  (* P -> I {P, Np}symkey *)
        in(c, m2);
        let (I,=Np,Ni) = sdecrypt(m2,symkey) in
        event end(I,P,Np);
        event begin2(P,I,Ni);
        out(c,sencrypt((Ni,secretK), symkey)). (*P->I {Ni, session key} *)

let I =
        in(c, m1); (* I receives message *)
        let (P, Np) = sdecrypt(m1, symkey) in (* checks the message *)
        event begin(I,P,Np);
        new Ni;
        out(c, sencrypt((I, Np, Ni), symkey)); (* I -> P: {I, Np, Ni}symkey *)
        in(c, m3);
        let(=Ni,y) = sdecrypt(m3, symkey) in
        event end2(P,I,Ni).

process new symkey;
        ((!P) | (!I))
```

A.1.2 *Secrecy*

Listing 2: ProVerif Input Secrecy

```
free c. (* public channel *)
fun sencrypt/2. (* symmetric encryption *)
reduc sdecrypt(sencrypt(x,y),y)=x. (* symmetric decryption *)
not symkey. (* secrecy assumption: symkey cannot be *)

private free secretK. (* secret k sent from P to I *)
query attacker: secretK. (* secrecy query *)

let P =
        new Np;  (* nonce *)
        out(c, sencrypt((P, Np), symkey));  (* P -> I {P, Np}symkey *)
        in(c, m2);
        let (I,=Np,Ni) = sdecrypt(m2,symkey) in
        out(c,sencrypt((Ni,secretK), symkey)). (*P->I {Ni, session key} *)

let I =
        in(c, m1); (* I receives message *)
        let (P, Np) = sdecrypt(m1, symkey) in (* checks the message *)
        new Ni;
        out(c, sencrypt((I, Np, Ni), symkey)); (* I -> P: {I, Np, Ni}symkey *)
        in(c, m3);
        let(=Ni,y) = sdecrypt(m3, symkey) in
0.

process new symkey;
        ((!P) | (!I))
```

A.2 OUTPUT PROVERIF

A.2.1 *Mutual authentication*

Listing 3: ProVerif Output Mutual Authentication

```
Process:
{1}new symkey_24;
(
        {2}!
        {3}new Np_25;
        {4}out(c, sencrypt((P,Np_25),symkey_24));
        {5}in(c, m2_26);
        {6}let (I_27,=Np_25,Ni_28) = sdecrypt(m2_26,symkey_24) in
        {7}event end(I_27,P,Np_25);
        {8}event begin2(P,I_27,Ni_28);
        {9}out(c, sencrypt((Ni_28,secretK),symkey_24))
) | (
        {10}!
        {11}in(c, m1_29);
        {12}let (P_30,Np_31) = sdecrypt(m1_29,symkey_24) in
        {13}event begin(I,P_30,Np_31);
        {14}new Ni_32;
        {15}out(c, sencrypt((I,Np_31,Ni_32),symkey_24));
        {16}in(c, m3_33);
        {17}let (=Ni_32,y_34) = sdecrypt(m3_33,symkey_24) in
        {18}event end2(P_30,I,Ni_32)
)
-- Query evinj:end2(x_43,y_44,z_45) ==> evinj:begin2(x_43,y_44,z_45)
Completing...
ok, secrecy assumption verified: fact unreachable attacker:symkey_24[]
Starting query evinj:end2(x_43,y_44,z_45) ==> evinj:begin2(x_43,y_44,z_45)
goal reachable: begin:begin2(P[],I[],Ni_32[m1 = sencrypt((P[],Np_25[!1 = @sid_41
8]),symkey_24[]),!1 = endsid_419]), m2_26 = sencrypt((I[],Np_25[!1 = @sid_418],N
i_32[m1 = sencrypt((P[],Np_25[!1 = @sid_418]),symkey_24[]),!1 = endsid_419]),sym
key_24[]), @sid_109 = @sid_418, @occ8_193 = @occ_cst() -> end:endsid_419,end2(P[
],I[],Ni_32[m1 = sencrypt((P[],Np_25[!1 = @sid_418]),symkey_24[]),!1 = endsid_41
9])
RESULT evinj:end2(x_43,y_44,z_45) ==> evinj:begin2(x_43,y_44,z_45) is true.
-- Query evinj:end(x_430,y_431,z_432) ==> evinj:begin(x_430,y_431,z_432)
Completing...
ok, secrecy assumption verified: fact unreachable attacker:symkey_24[]
Starting query evinj:end(x_430,y_431,z_432) ==> evinj:begin(x_430,y_431,z_432)
goal reachable: begin:begin(I[],P[],Np_25[!1 = endsid_776]), m1_29 = sencrypt((
P[],Np_25[!1 = endsid_776]),symkey_24[]), @sid_213 = @sid_777, @occ13_654 = @occ
_cst() -> end:endsid_776,end(I[],P[],Np_25[!1 = endsid_776])
RESULT evinj:end(x_430,y_431,z_432) ==> evinj:begin(x_430,y_431,z_432) is true.
```

A.2.2 *Secrecy*

Listing 4: ProVerif Output Secrecy

```
Process:
{1}new symkey_24;
(
        {2}!
        {3}new Np_25;
        {4}out(c, sencrypt((P,Np_25),symkey_24));
        {5}in(c, m2_26);
        {6}let (I_27,=Np_25,Ni_28) = sdecrypt(m2_26,symkey_24) in
        {7}out(c, sencrypt((Ni_28,secretK),symkey_24))
) | (
        {8}!
        {9}in(c, m1_29);
        {10}let (P_30,Np_31) = sdecrypt(m1_29,symkey_24) in
        {11}new Ni_32;
        {12}out(c, sencrypt((I,Np_31,Ni_32),symkey_24));
        {13}in(c, m3_33);
        {14}let (=Ni_32,y_34) = sdecrypt(m3_33,symkey_24) in
        0
)

-- Query not attacker:secretK[]
Completing...
ok, secrecy assumption verified: fact unreachable attacker:symkey_24[]
Starting query not attacker:secretK[]
RESULT not attacker:secretK[] is true.
```