UNIVERSIDADE DA BEIRA INTERIOR
Faculdade de Engenharia

Security and Privacy for Implantable Cardioverter Defibrillators

Nuno Miguel Alves dos Santos

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática
(2º ciclo de estudos)

Orientador: Prof. Dr. Paul Crocker

Covilhã, Outubro 2014

Dedication

I would like to dedicate this dissertation to everyone who suffers or knows someone who suffers from medical conditions, especially to those with heart diseases. I also dedicate this work to everyone that may be benefit from its existence, and to those that may use it as a basis for future relevant works regarding the improvement of security and privacy in implantable cardioverter defibrillators or any other medical device.

Acknowledgements

Two years of my tour through Computer Science reached their pinnacle here in this work. They were full of both joy and sorrow, hard work and accomplishments, knowledge and friendship, pain and gain. This dissertation represents the end of a journey and the beginning of a new stage in my life, and so I would like to thank everyone who shared these past two years with me.

I would like to thank my parents, Fátima and Francisco. Mãe, Pai, sem vocês nada disto seria possível. O apoio durante os estudos, aquela motivação extra quando algo corria mal, o incentivo constante para me ultrapassar e tornar-me cada vez melhor. Espero agradar-vos e orgulhar-vos com o que alcancei na minha vida até agora e com o que me tornarei no futuro. Um bem-haja do tamanho do universo para vocês!

To my grandmothers Clotilde and Rita: um grande beijinho para as minhas avós, que também acompanharam de perto a minha carreira de estudante e que por vezes foram as minhas confidentes nas alturas de maior tristeza.

To my dear older brother Vasco, for always helping me anyway he cold when I asked him to, and for being that person I always aimed to impress and overcome.

To my buddies Nuno Leitão and Nuno Pereira, with whom I shared all kinds of academic adventures, being them related to study or recreational, and who were the constant source of strength and support to face all kinds of challenges.

To my long time friend Mário Pereira. Who would say we would meet again and once again become such great friends after so many years? A mind like few exist nowadays and an example to follow.

To my friend Diogo Tavares for being there when others were not, and for showing me there is no fear in leaving everything behind to start a new stage in life.

To my friend André Pimenta for proving me we can find great friends where we don't predict it. The coffee breaks and bohemias I shared with you will always be a part of this dissertation.

To everyone with whom I took the chance to play football (soccer) in the usual thursday's sessions. A very special tradition that I hope will continue for many many years.

To everyone related to the RELEASE Research Group, with special acknowledgements to professor Paul Crocker - my dissertation advisor - to whom I owe this dissertation. I would also like to leave a very special acknowledgement to professor Simão de Sousa, who became more of a friend than just a teacher, giving me important advices and spectacular opportunities.

To all not named here, but who know they gave their share of both friendship and partnership, thank you.

It was quite a journey. And I owe much of it to all of you!

"It is good to have an end to journey toward; but it is the journey that matters, in the end."

Ernest Hemingway

Resumo

Num mundo onde existem cada vez mais pessoas com problemas de saúde diretamente relacionados com o coração, estas possuem muitas vezes a necessidade da implantação de sistemas de auxílio para o normal funcionamento do principal órgão humano, a qual tem vindo a aumentar de forma exponencial. Dentro da gama de dispositivos pertencentes à família de dispositivos médicos implantáveis (IMDs - Implantable Medical Devices), os desfibriladores cardioversores implantáveis - ICDs (*Implantable Cardioverter Defibrillators*) - são atualmente aqueles que possuem maior gama de auxílio para as diferentes anomalias cardíacas. No entanto, estes sistemas possuem algumas falhas a nível de segurança de comunicações, muito devido à dificuldade em conjugar privacidade e segurança com salvaguarda e usabilidade.
A presente dissertação apresenta um estudo sobre a segurança e privacidade das comunicações entre ICDs e os respetivos programadores, onde são apresentados os atuais standards de *hardware*, comunicação, segurança e privacidade dos mesmos. Analisam-se ainda algumas soluções já existentes que propõem melhorar a segurança e privacidade das comunicações entre ICDs e programadores, apresentando uma análise e crítica. Para além disso, é também apresentado o quão simples é o procedimento para intercetar um sinal de rádiofrequência emitido pela chave de um carro, processo o qual é idêntico ao das comunicações realizadas entre um ICD e o seu programador. Por fim, são apresentadas algumas propostas de novas arquiteturas para as comunicações de um ICD com as diferentes entidades que constituem o *backoffice*, incluindo ainda a implementação de duas aplicações android que recorrem ao uso das tecnologias *near field communication* (NFC) e *message queuing telemetry transport* (MQTT), servindo como provas de conceitos.

Palavras-chave

ICD, comunicações, privacidade, segurança, NFC, MQTT.

Resumo alargado

Doenças cardiovasculares são doenças do coração e dos vasos sanguíneos que causam mais de 4 milhões de mortes por ano na Europa. Arritimia é um tipo de doença cardiovascular em que o coração bate a um ritmo anormal, podendo este ser demasiado lento, rápido ou irregular. Os desfibriladores cardioversores implantáveis - ICDs (*Implantable Cardioverter Defibrillators*) são dispositivos que monitorizam a actividade cardíaca, respondendo em conformidade. Através da libertação de estímulos eléctricos, este dispositivos possuem modos para marcar ritmo e para desfibrilação.

O médico de um utilizador destes aparelhos quererá regularmente avaliar tanto o ICD como a saúde em geral do seu paciente. Atualmente, a maneira mais comum deste procedimento ocorrer é o paciente deslocar-se de maneira a que o médico ou outro técnico use um programador para ler a informação armazenada no ICD. Começam também a ser usados métodos alternativos recorrendo a monitorização remota, no qual as informações do aparelho podem ser transmitidas através de linha telefónica ou através de um *web site*.

As atuais comunicações entre ICD e programador ocorrem através do serviço MedRadio, o qual fornece um total de cinco megahertz (401-406 MHz) de espetro contínuo para comunicações sem fios através de radiofrequência (RF). No entanto, este tipo de comunicações possui muitas falhas. Como Halperin *et al.* referem [HHBR$^+$08], através destas comunicações é possível um ICD divulgar informação sensível, tanto em texto limpo como cifrada, a um adversário, bem como permite a um adversário montar ataques de reprogramação que podem alterar os modos de operação e a informação contida nos aparelhos. Por isso, foram propostos vários objectivos de segurança/privacidade e salvaguarda/utilidade para desenvolver novos métodos que tenham a intenção de aumentar a segurança e privacidade das comunicações entre os ICDs e os respetivos programadores [HHBF$^+$08], os quais também são aqui apresentados.

Com este trabalho, pretende monstrar-se como é simples interceptar as comunicações que têm por base RF, montando um cenário de exemplo no qual se interceptam os sinais RF que uma chave de um carro emite quando se pretende abrir ou fechar o carro recorrendo a um dispositivo bastante económico (RTL2832U) e a um software para Windows (HDSDR).

Para além disso, procede-se à análise do que se consideram ser duas propostas de elevado grau de importância existentes para melhorar a segurança e privacidade nas comunicações dos ICDs: as defesas *zero-power* propostas por Halperin *et al.* [HHBR$^+$08], e o sistema não-invasivo *Shield* proposto por Gollakota *et al.* [GHR$^+$11]. Concluiu-se que ambas estas propostas iriam de facto aumentar o nível de seguranaça e privacidade nas comunicações por RF entre ICD e programador. As defesas *zero-power* beneficiariam da implementação de um novo sistema de autenticação e de novos testes de capacidade do WISP em correr iterações mais fortes do algoritmo RC5. Relativamente ao *Shield*, apesar de fornecer eficazmente confidencialidade para os dados transmitidos para e pelos IMDs, ao mesmo tempo que protege este dispositivos de comandos não autorizados sem requerer qualquer tipo de alterações aos IMDs, existe a possibilidade - mesmo que reduzida - de existir comunicação errónea de dados. Por outro lado, o volumoso design do protótipo apresentado reduz significativamente a portabilidade de um sistema que cada vez mais se requer portátil.

Por fim e com o objetivo de melhorar as atuais comunicações relacionadas com ICDs, são ainda apresentadas cinco possíveis arquiteturas que envolvem a substituição dos atuais programadores e monitores por *smartphones*.

As primeiras duas envolvem a utilização de um dispositivo WISPer como intermediário de comunicações entre o ICD e o *smartphone*, sendo que a primeira requer a implementação de um *Universal*

Integrated Circuit Card (UICC) - o qual inclui um SIM card - no dispositivo WISPer, e a segunda já não. A segunda teria como vantagem a continuação dos atuais standards para a comunicação ICD-WISPer-programador, mas como o programador seria agora um *smartphone*, as comunicações com as restantes entidades poderia ser feita por comunicações *Global System for Mobile* (GSM). No entanto, para a primeira teriam que ser feitos testes de sustentabilidade do WISPer com um UICC, e as comunicações pós WISPer já seriam também feitas por GSM.
A terceira proposta envolve a implementação de um cartão SIM no ICD. Este é talvez a arquitetura mais controversa e improvável de acontecer, visto que modificar o *hardware* de um dispositivo médico - ainda por mais um que é implantado - acompanha sempre bastantes problemas, dificuldades e obrigatoriedades da existem de testes, autorizações e modificações. No entanto, esta arquitetura permitira ao ICD comunicar diretamente com uma operadora de redes móveis (MNO) e consequentemente com as restantes entidadesdo *backoffice*, largamente aumentando a portabilidade do sistema.
A quarta proposta apresentada incide somente nas comunicações entre ICD e monitor/programador, mantendo um dispositivo a intermediar as comunicações. Nesta arquitetura, um dispositivo semelhante ao WISPer, mas com capacidades de comunicação por *Near Field Communication* (NFC), faz a ponte de ligação entre o ICD e o *smartphone* através de comunicações com um alcance máximo de cerca de 4 centímetros, aumentando o nível de segurança e consequentemente privacidade dessas comunicações, ao mesmo tempo que a portabilidade do sistema é garantida.
A quinta e última proposta surge como complemento para as comunicações *backoffice* da arquitetura anterior, recorrendo à utilização da tecnologia Message Queuing Telemetry Transport (MQTT). Ambas as últimas duas propostas incluem uma implementação que serve como prova de conceito de um exemplo prático de como estas tecnologias podiam ser usadas. Para tal, modificaram-se duas aplicações Android, uma para escrever para *tags* NFC, e a outra para receber textos de uma *tag* NFC e enviar esse texto - um ficheiro com dados do ICD - para as entidades *backoffice* através de MQTT.
A fusão das últimas duas propostas surge como a proposta que melhor poderia fortalecer as comunicações envolvendo ICDs em termos de segurança, privacidade, salvaguarda e utilidade, sem comprometer alguma destas propriedades em detrimento de outra, e cumprindo assim com os atuais standards de segurança e privacidade para IMDs.

Abstract

In a world where the number of people with health issues directly related with the heart is increasing, the need for this people to implant auxiliary systems for the normal functioning of the human body's main organ is exponentially increasing. Amongst the range of devices belonging to the family of implantable medical devices (IMDs), implantable cardioverter defibrillators (ICDs) currently are the devices that posses the widest range of therapeutic features for the different existing cardiac anomalies. However, these systems possess some flaws at the communications security level, mainly due to the difficulty in balancing privacy and security with safety and utility.
The present dissertation presents a study on security and privacy of communications between ICDs, their respective programmers and the range of health card entities involved in receiving and monitoring information from and to these devices. The current standards of hardware, communication, and security and privacy of these devices are presented and in addition, it is presented how simple it can be to intercept a radio frequency (RF) signal sent by a key fob, process which is similar to the communications done between an ICD and its programmer. As well as this study the principal focus of this thesis is to study and present some alternative proposals for the current communications architecture of ICD communications with the backoffice, where we include the implementation of two proof-of-concept android applications, using near field communication (NFC) and message queuing telemetry transport (MQTT) technologies.

Keywords

ICD, communications, privacy, security, NFC, MQTT.

Contents

List of Figures

List of Tables

List of Acronyms

AES	Advanced Encryption Standard
ATP	Antitachycardia Pacing
COFDM	Coded Orthogonal Frequency Division Multiplexing
CPA	Chosen Plaintext Attack
CPU	Central Processing Unit
CRT	Cardiac Resynchronization Therapy
CVD	Cardiovascular Disease
DoS	Denial-of-Service
DVB-T	Digital Video Broadcasting - Terrestrial
ECG	Electrocardiogram
ECMA	European Computer Manufacturers Association
EEPROM	Electrically Erasable Programmable Read-Only Memory
EHR	Electronic Health Record
EMG	Electromyogram
ETSI	European Telecommunications Standards Institute
FCC	Federal Communication Commission
FFT	Fast Fourier Transform
GPIO	General-Purpose Input/Output
GPRS	General Packet Radio Service
GSM	Global System for Mobile communications
HDSDR	High Definition Software Defined Radio
IBM	The International Business Machines Corporation
ICD	Implantable Cardioverter Defibrillator
IEGM	Intracardiac Electrogram
IMD	Implantable Medical Device
IMSI	International Mobile Subscriber Identity
ISO	International Organization for Standardization
JAAS	Java Authentication and Authorization Service
M2M	Machine to Machine
MedRadio	Medical Device Radiocommunications Service
MEDS	Medical Data Service
MICS	Medical Implant Communications Service
MiTM	Man-in-the-Middle
MNO	Mobile Network Operator
MQ	Message Queuing
MQTT	Message Queuing Telemetry Transport
MSISDN	Mobile Station International Subscriber Directory Number
NFC	Near Field Communication
OTA	Over-the-Air
OTP	One-Time Password authentication system
PCN	Patient Care Network
PIN	Personal Identification Number
QoS	Quality of Service
RAM	Random Access Memory

ROM	Read-Only Memory
RF	Radio Frequency
RFID	Radio Frequency Identification
RKE	Remote Keyless Entry
SDR	Software Defined Radio
SIM	Subscriber Identity Module
SMS	Short Message Service
SRAM	Static Random-Access Memory
SSL	Secure Sockets Layer
UBI	Universidade da Beira Interior
UHF	Ultra-High Frequency
UICC	Universal Integrated Circuit Card
UMTS	Universal Mobile Telecommunication Service
USIM	Universal Subscriber Identity Module
VF	Ventricular Fibrillation
VT	Ventricular Tachycardia
WISP	Wireless Identification and Sensing Platform

Chapter 1

Introduction

1.1 Motivation

We live in a world where the number of people affected by health care issues related with the heart is continuously increasing, which is directly related to a large quota of the deceases registered in what are considered developed countries. People with cardiovascular diseases often need to implant auxiliary systems for the normal functioning of the main organ of the human body, and the number of such implants has been rising exponentially. Within the range of devices belonging to the family of implantable medical devices (IMDs), implantable cardioverter defibrillators (ICDs) currently are the devices possessing the widest range of therapeutic aids for the different cardiac anomalies by having functions of pacing, cardioversion and defibrillation. As well as dynamical configuration of operating parameters and the ability to monitor the state of the implanted device and obtain patient data at regular clinical appointments the latest IMDs support delivery of telemetry for remote monitoring over long-range, high-bandwidth wireless links. In fact outpatient and remote monitoring are important medical research areas.
Despite the advances in IMD technologies and their take-up by medical device companies our understanding of how device security and privacy interact with and affect medical safety and treatment efficacy is still limited. Current established methods for providing safety and preventing unintentional accidents do not prevent intentional failures and other security and privacy problems. Balancing security and privacy with safety and efficacy is becoming increasingly important with IMDs technologies evolution.
This dissertation was motivated by Halperin *et al.* in their work of *Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses* [HHBR+08] and *Security and Privacy for Implantable Medical Devices* [HHBF+08], allied with the increasing number of people dependent on ICDs to live their normal life, the lack of rigorous public investigations into the security and privacy of ICDs communications, the difficulty of finding decent background information related to the topic of heart diseases and about existent technologies for increasing both privacy and security of ICDs. There is a need to collect information regarding relevant information about ICDs and its related cardiovascular diseases. In the process of collecting and analysing information relating to current devices some flaws in current architectures for ICDs' communications with the respective programmers and monitors were identified. Therefore, there is a need to propose alternatives to current architectures for trying to increase the levels of privacy, security, safety and utility as well as architectures that take advantage of modern technologies.

1.2 Objectives and Contributions

This dissertation has two distinct contributions.
First, this dissertation presents a study and review of the background information directly involving ICDs used to treat cardiovascular diseases,in particular ICDs' design and communications, security and privacy standards and existent technologies, something that is lacking in current literature.
The second contribution is of technical nature.

In this dissertation the security of the wireless technology currently used in IMDs is analysed. Communication protocols described in the literature are also described and analysed. We shall show how easy one individual can intercept current radio frequencies - the current method of wireless communications for ICDs - through the example of intercepting the radio frequency of a remote keyless entry key fob.
After analysing current systems, we devised and designed several architectures for ICD communications that could enhance security and privacy without compromising the devices' safety and utility goals. We also developed and hereby present a proof-of-concept for the architecture we believe would best fit today's requirements.
In summary this dissertation contains a study and review of all the required relevant background information necessary to understand the area and could serve as a useful initial step in implementing new architectures and solutions such as those proposed in this thesis.

1.3 Dissertation organization

This dissertation is divided into five chapters.
The first chapter described the motivation behind this dissertation and presents the objectives it aims to achieve.
Chapter two gives a introduction to the relevant background information: cardiovascular diseases, implantable medical devices (IMDs), implantable cardioverter defibrillators (ICDs), security and privacy standards for IMDs, existing ICDs related technologies for privacy and security, near field communication (NFC), NFC-WISP and the Message Queue Telemetry Transport (MQTT) protocol for data communication.
The following chapter discusses radio frequency (RF) and security mechanisms in ICDs. It starts with a demonstration of how to intercept RF communications through intercepting the signal of a key fob, moving on to analysing the authentication protocol proposed by Halperin *et al.* in Zero-Power Defenses, and finishing with the discussion of the *Shield* technology to enhance security and privacy in RF communications of ICD with the respective programmers.
In chapter four, we present five proposals of possible architectures that could be implemented to enhance both security and privacy in ICDs. For two of these architectures we also present a proof-of-concept application implementation, with an example of utilization.
The final chapter of this dissertation offers concluding remarks, as well as suggesting possible future work that can still be done to enhance both security and privacy in ICDs.

Chapter 2

Background

This chapter contains the background information necessary for the understanding of the work done in this dissertation. It starts with a brief introduction to cardiovascular diseases and implantable medical devices (IMDs), and how the implantable cardioverter defibrillators (ICDs) are included within that set of medical devices. Following this introduction, the current standards of the currently built ICDs in terms of technology, hardware and communications, as well as the current standards for privacy and security of information in IMDs are presented . Finally, this chapter presents a list of important existing technologies for privacy and security enhancement of ICDs, as well as the concepts of the Near Field Communication (NFC) technology, and the NFC-WISP device.

2.1 Cardiovascular Diseases

Cardiovascular disease (CVD) - also called heart disease - is a heart and blood vessel disease that includes numerous problems, many of which are related to a process called atherosclerosis [Ame13]. Atherosclerosis is a condition developed when a substance called plaque builds up in the walls of arteries. This event causes the narrowing of the arteries, making it harder for blood to flow through. If a blood clot forms, it can stop the blood flow, which may then be followed by a heart attack or stroke. In case this clot cuts off the blood flow completely, the part of the heart muscle supplied by that artery begins to die. Most people survive their first heart attack and return to their normal lives. But having a heart attack does mean that person has to make some changes to his/her own lifestyle.

Each year, CVD causes over 4 million deaths in Europe [Eur12], representing 47% of the total causes of death. It is the main cause of death in women in all European countries and is the main cause of death in men in all but six European countries. Figures 2.1 and 2.2 show diagrams in which we can see the mortality for both men and women in Europe posted for the 2012 edition of the European Cardiovascular Disease Statistics. It is clear that CVDs and other heart and blood vessel diseases represent the greatest cause of death in Europe. In the USA, there is a rate of 600.000 deaths per year due to CVDs, representing a quarter of the total of death [Cen14]. It is the leading cause of death for both men and women. Every year about 720.000 Americans have a heart attack, of which 205.000 happen in people who have already had one before.

Heart failure is a type of cardiovascular disease that doesn't necessarily mean the heart stops beating [Ame13]. Heart failure, sometimes called congestive heart failure, is the condition of the heart not pumping blood as well as it should. The heart keeps working, but the body's need for blood and oxygen is not met.

The heart is a muscle that has an internal electrical system that controls the rate and rhythm of your heartbeat[Nat14]. With each heartbeat, an electrical signal spreads from the top of your heart to the bottom. As the signal travels, it causes your heart to contract and pump blood. The process repeats with each new heartbeat. A problem with any part of this process can cause an arrhythmia.

Arrhythmia is an abnormal rhythm of the heart [Ame13]. There are various types of this heart

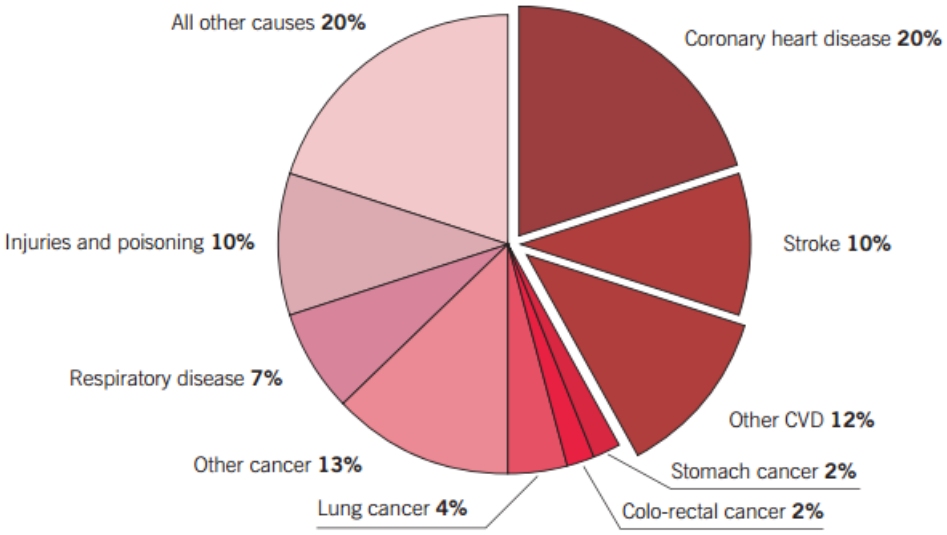

Figure 2.1: Deaths by cause, men, latest available year, Europe, according to the 2012 Edition of the European Cardiovascular Disease Statistics [Eur12].

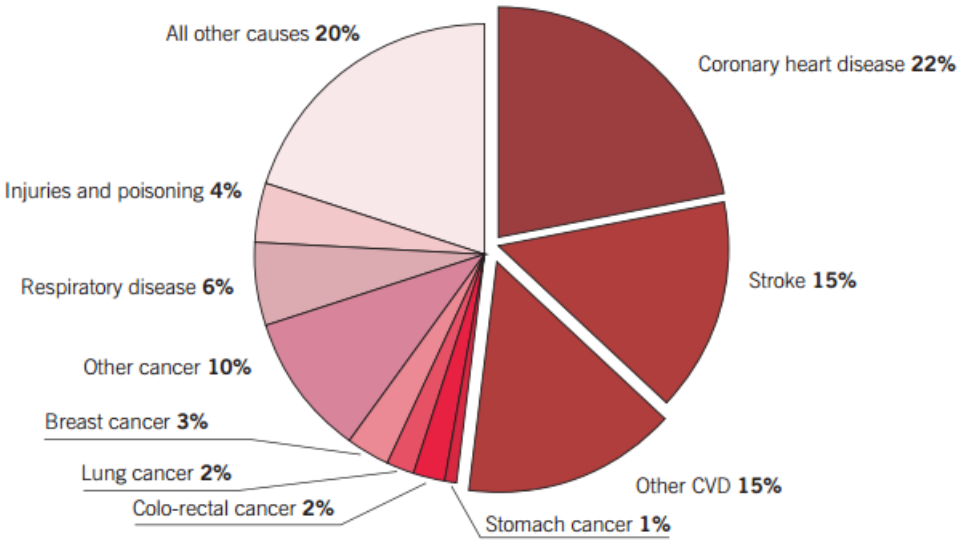

Figure 2.2: Deaths by cause, women, latest available year, Europe, according to the 2012 Edition of the European Cardiovascular Disease Statistics [Eur12].

condition. The heart can beat too slow, too fast or irregularly. Arrhythmias result from a problem in your heart's electrical system and some arrhythmias that originate in the heart's lower chambers (the ventricles) may be life-threatening [Bos10]. There are three important types worth mentioning for the purpose of this work: bradycardia, ventricular tachycardia (VT) and ventricular fibrillation (VF).

Bradycardia is when the heart rate is less than the normal rate of 60 to 100 beats per minute.

VT is an abnormally fast heart rhythm that occurs in the ventricles, causing the heart to beat more than 100 times per minute. This may result in the person's body and brain not getting enough oxygen and nutrients to function properly. VT sometimes stops on its own after a few seconds, other times it continues and it can even progress into a faster chaotic arrhythmia called ventricular fibrillation.

VF is an abnormally fast and chaotic heart rhythm that occurs in the heart's ventricles, causing the heart to beat more than 200 to 300 times per minute, which makes it a chaotic rhythm. That means the heart's ventricles try to contract so fast that they quiver rather than beat, and the heart rate is so fast that it cannot pump enough blood to the brain and body tissue. VF is the most dangerous type of arrhythmia and it often strikes without warning. A person can lose consciousness within seconds after VF begins, and if the person doesn't receive immediate treatment from a defibrillator, sudden cardiac arrest and sudden cardiac death can occur within just a few minutes.

2.2 Implantable Medical Devices

Implantable medical devices (IMDs) monitor and treat physiological conditions within the body [HHBF$^+$08]. These devices - including pacemakers, implantable cardioverter defibrillators (also known as implantable cardiac defibrillators - ICDs), drug delivery systems and neurostimulators - can help manage a broad range of ailments, such as cardiac arrhythmia, diabetes and Parkinson's disease. IMDs' pervasiveness continues to grow, with upward of 25 million US citizen currently reliant on them for life critical functions [HHBF$^+$08]. The latest IMDs support delivery of telemetry for remote monitoring over long-range, high-bandwidth wireless links. Despite these advances in IMD technology, the understanding of how device security and privacy interact with and affect medical safety and treatment efficacy is still limited.

2.3 ICDs

This section addresses the current standards regarding the Implantable Cardioverter Defibrillators (ICDs) technology. In the following subsections will be firstly presented the evolution of the ICD technology since its first implant in a human being until the present day, followed by the presentation of the current standards of an ICD in the present days, finishing with a brief discussion of how security and privacy in IMDs need to follow certain standards.

2.3.1 Evolution of ICD technology

For more than 50 years, antibradycardia pacemakers have been implanted [(Ed11]. Technological developments have led to an improvement, extension of diagnostic and treatment options (such as holter function for detecting arrhythmias and biosensors), and to an increasingly more automated device management (control of sensing and stimulus thresholds).
In 1980, the first ICD was implanted in a human being with the objective of secondary prevention of sudden cardiac death. Meanwhile, advances in technology have led to a size reduction of the device assembly and to the possibility of transvenous implantations.
Since the 1990's, biventricular pacemakers and ICDs enabled with Cardiac Resynchronization Therapy (CRT) are being implanted in patients with reduced left ventricular ejection fraction, prolonged QRS-complex and advanced heart failure. Clinically asymptomatic patients with reduced left ventricular pump function and prolonged QRS-complexes are now also considered candidates for implantation of biventricular pacemakers and ICDs to prevent cardiac de-compensations.
In recent years, national and international associations have drawn up guidelines for implantation of antibradycardia, ICD and CRT devices. Implantation rates for these devices have constantly increased. In the USA, the expansion of indications for ICD and CRT implantations to include primary prevention of sudden cardiac death led to an amplification of these implantations.

While the number of pacemaker aggregate replacements remained constant in 1992-2006, the number of ICD aggregate replacements decreased during this period due to runtime extension of ICD aggregates. However, despite technical improvements in implantation and devices, perioperative complications, inadequate shock outputs, ICD-lead related complications and complications caused by the aggregate are to be expected.
The current developments and risks in device therapy prescribe requirements to be met in terms of patient safety, follow-up appointments and an increasingly complex management of ICD-patients. A device-based remote-monitoring represent an important contribution to meet these requirements and fulfil the needs.

2.3.2 Principles of the ICD technology

An ICD is a device that monitors and responds to heart activity. ICDs have modes for pacing, wherein the device periodically sends a small electrical stimulus to the heart, and for defibrillation, wherein the device sends a larger shock to restore normal heart rhythm [HHBR+08].
Nowadays, an ICD system consists of two components - the pulse generator, or device, and one or two thin, insulated wires called leads [Bos10]. Leads carry electrical signals between the heart and the pulse generator. A representation of an ICD can be seen in figure 2.3. An ICD system is surgically implanted by a physician below the patient's clavicle and close to the skin with the electrical leads that connect the ICD to the heart muscle. Post-surgery, a health care practitioner can use an external device called programmer to perform diagnostics, read and write private data, and adjust therapy settings.

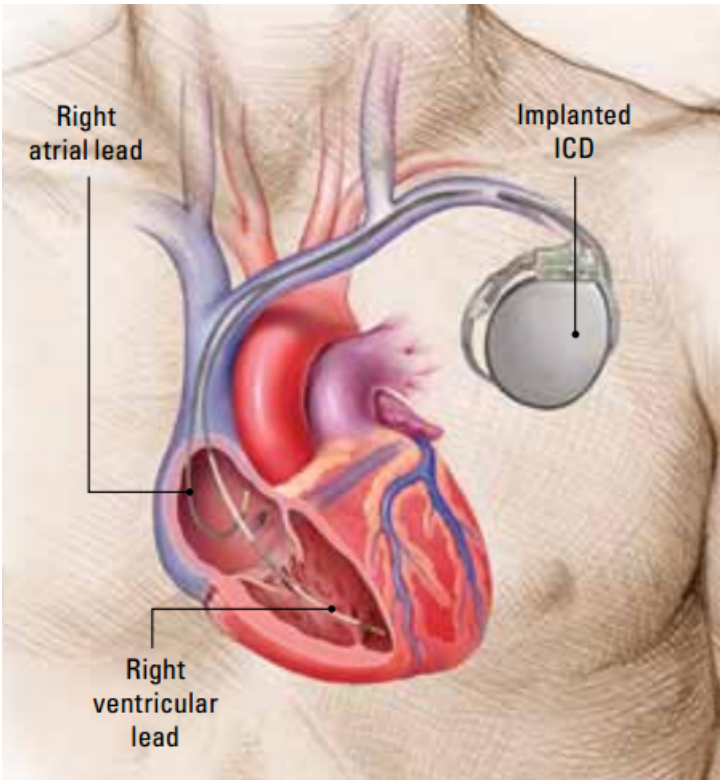

Figure 2.3: Human chest representation with an implanted ICD (top right, near shoulder) and electrical leads connected to heart chambers.

An ICD monitors or senses the rate of the person's heartbeat in the right side of the heart. Based on what it senses, the ICD device delivers electrical impulses to the individual's ventricles to restore

a normal heart rhythm [Bos10]. This way, an ICD treats abnormally fast ventricular heart rhythms by delivering tiny amounts of electrical energy (anti-tachycardia pacing) to the heart and so slowing the heart down to a more normal rhythm. If the device senses a heart rate that is dangerously fast, it delivers a shock to the heart. This shock (defibrillation) stops the abnormal rhythm. Without this lifesaving therapy, the dangerously rapid rhythm could lead to sudden cardiac arrest, and that could lead to sudden cardiac death.
ICDs have been shown to effectively stop 95% or more of dangerously fast heart rhythms [HLM+], while patients who do not receive an ICD for life-threatening arrhythmias are at high risk for sudden cardiac death [R05]. ICDs are usually recommended to people who correspond to at least one of the following items [Bos10]:

- Had at least one documented episode of a dangerous ventricular arrhythmia;
- Previously passed out because of arrhythmia;
- A recurring fast heart rhythm that puts the person at risk for sudden cardiac death;
- A fast heart rhythm that cannot be controlled with medication and neither cured by surgery;
- Severe side effects from certain medication;
- Survived a heart attack and have a low ejection fraction;
- An inherited heart defect that causes a fast heart rhythm.

As mentioned before, an ICD can use one or more types of energy to help a person's heart to beat normally again. They include [Bos10]:

- **Bradycardia pacing therapy**: ICDs have the ability to treat a slow heart rate, a condition called bradycardia. When an ICD treats an abnormally slow heart rate, the person generally doesn't feel anything because it uses low energy to pace the heart.
- **Antitachycardia pacing (ATP)**: If the heart's rhythm is regular but fast, the ICD can deliver a series of small, rapid electrical pacing pulses. These pulses interrupt the arrhythmia and return the heart to its normal rhythm, with the person generally doesn't feeling anything because ATP uses low energy to pace the heart.
- **Defibrillation**: For arrhythmias that are very fast, the ICD may use a high-energy shock to stop them so the person's heart can return to its normal rhythm. This type of therapy uses higher energy, which turns possible for the person to feel the shock.

Although the implantation of an ICD can significantly improve a patients life, there are risks after an ICD is implanted. For example, the leads may move out of place in the heart, the leads or electrical impulses may irritate or damage the surrounding tissues, and the ICD might not be able to detect or appropriately treat the heart rhythms. However, ICDs are in general reliable. In 2006, 600.000 people worldwide received a defibrillator, and ICDs help hundreds of thousands of people live longer every year [Bos10]. Along with medication prescribed by the physician, an ICD may be the best choice to help protect a person's heart from dangerous rhythms. Nonetheless, devices are not perfect and may exhibit problems and thus need constant monitoring.
The person's physician will want to check both the ICD and overall health on a regular basis. Currently there are many ways to do so, but the most common on this days is for the patient to attend the clinic where the doctor or other technician will use the programmer to read all of the

information stored in the ICD. This information includes whether the ICD delivered a high-energy therapy. If it did deliver a therapy, the ICD's computer memory will store information regarding what the heart was doing before, during and after arrhythmia. The programmer also indicates how much energy is left in the ICD's battery. With such information, the doctor can help ensure the ICD is working properly to best treat the patient's heart condition.
The device may also be checked at the patient's home using an in-home (remote) monitoring system, something that is becoming more common these past few years. If the physician wants to monitor the device from the patient's home, a portable communications device can transmit device and health information through regular phone line to a web site. This process is getting many proposals of alteration and will be one of the many discussions throughout this dissertation.

2.3.3 Hardware of an ICD

Today's ICDs are amazingly small (see figure 2.3). The average device is about the size of a small bar of soap, measuring about 5 x 5 centimetres and less than 11 millimetres thick, weighing about 28 to 85 grams [Bos10]. ICDs typically consist of the following items [HHBF$^+$08]:

- Sealed, battery-powered, sensor-laden pulse generator;
- Several steroid-tipped, wire electrodes (leads) that connect the generator to the myocardium (the heart muscle);
- A custom ultralow-power microprocessor, typically with about 128 Kbytes of RAM for telemetry storage.

The device's primary function is to sense cardiac events execute therapies and store measurements such as electrocardiograms (telemetry data). Healthcare professionals configure the setting on ICDs using the programmer.
Pacemakers and ICDs often contain high-capacity lithium-based batteries that last five to seven years [DG06]. Rechargeable batteries are extremely rare, for practical, economic and safety reasons. Device lifetime depends on the treatments required, and a single defibrillation can reduce the ICD's lifetime by weeks.
Some ICDs have inside them a magnetic switch [HHBR$^+$08]. A magnetic field in proximity to this switch causes it to close, which in turn causes the ICD to wirelessly transmit telemetry data. Nowadays, this process of activating transmission of telemetry on the ICD is done with an RF command and without the presence of a magnet.

2.3.4 Remote monitoring - Existent technologies

Remote monitoring overcomes the spatial separation between patient and physician [(Ed11]. Device-based remote monitoring has become a classical field for telemedical applications in cardiology, in addition to diagnostics of cardiac arrhythmia and telemonitoring of chronic heart failure patients. Already in the mid of the 1970s, first examinations of transtelephonic monitoring of patients with antibradycardia pacemakers were carried out. At first, Electrocardiograms (ECGs) were recorded and transmitted via telephone to a receiving centre. A transmission of pacemaker function was, however, not possible. Medtronic "CareLink 2090" and St. Jude Medical "Housecall" were the first systems to allow remote monitoring. The CareLink-System enabled the computer in the monitoring centre to connect via telephone the device. Thus, remote monitoring bridged the spatial distance between two different observers, and it became possible to have a consultation without any active intervention in programming.

St. Jude Medical developed the Housecall-System to transmit data from the ICD to the physician. The system allowed the patient to gather and transmit information to the practitioner about the ICD using the Housecall Plus Transmitter. The information gathered from intracardiac electrograms and the online intracardiac ECG allowed real-time ICD-surveillance for the first time. Either the patient or the physician can initiate the call to transmit via the small transmitter up-to-the-second information about how the patient's heart and ICD are working. The system also enabled the physician to monitor device performance. However, a determination of stimulus thresholds and a programming of the ICD settings were not possible.
In the 1990s, BIOTRONIK started the development of the "Home Monitoring" technology, with hundreds of thousands of BIOTRONIK Home Monitoring systems implanted so far [(Ed11]. For some time, this system was the only remote monitoring system in which the transmission of data to the CardioMessenger requires no action by either the patient or the practitioner. The CardioMessenger transmits the data to BIOTRONIK's Service Center via a cell phone, which then analyses the data and forwards it to the patient's physician either by sms, e-mail or fax. The Home Monitoring concept has been modified slightly and extended, and nowadays it represents the basic technological principle for telemonitoring for patients with electrical implants.
These days, all telemonitoring systems consist of the following components [(Ed11], as shown in figure 2.4:

- Implanted device;
- Patient monitor;
- The provider's data server;
- Data presentation for the physician.

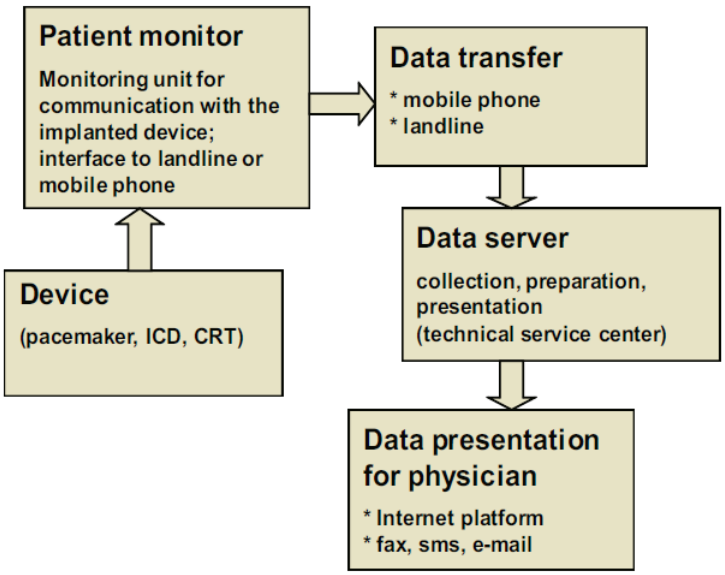

Figure 2.4: Individual components of a device-based remote monitoring for patients with pacemakers, ICDs and CRT-systems [(Ed11].

However, all manufacturers (BIOTRONIK, Medtronic, St. Jude Medical, Boston Scientific) have developed their own concepts for remote monitoring of ICDs, which, in spite of their uniform structure, vary in their technical realization and features (Table 2.1).

Table 2.1: Overview of different systems for remote monitoring of pacemakers (PM), ICDs and CRT-systems) [(Ed11].

System (manufacturer)	Implants	Patient monitor (data transfer from the implanted device)	Transfer to data server	Data presentation to the physician	Integration into the EHR	Specifics
Home Monitoring (BIOTRONIK)	PM, ICD, CRT	Cardio-Messenger (automatic)	GSM, GPRS	Internet, alerts via sms, e-mail, fax	Possible	IEGM-online-transmission, Heart-Failure-monitor
CareLink (Medtronic)	PM, ICD, CRT (backward compatible)	CareLink-monitor (manual and automatic)	Telephone line	Internet, alerts via sms	Possible	OptiLink-system (intrathoracic impedance measurement), IEGM, Cardiac Compass
Merlin.net(St. Jude Medical)	Specific ICDs, CRTs	Merlin@home (manual and automatic)	GSM, telephone line	Internet, alerts via sms, e-mail, fax	Possible	Holistic data-management-system, line-transmission
LATITUDE (Boston Scientific)	Specific ICDs, CRTs	LATITUDE-Communicator (manual and automatic)	Telephone line	Internet	Possible	Integration of external sensors (weight scale, blood pressure monitor)

Data can also be transmitted from the implant to the patient monitor in various ways. This included, for instance, transmissions that can be initiated automatically without any user interaction (Home Monitoring, BIOTRONIK) or by radio frequency (RF) wireless telemetry that is used to download data from the device (Merlin.net, St. Jude Medical; LATITUDE, Boston Scientific). These devices can be seen in Figure 2.5.

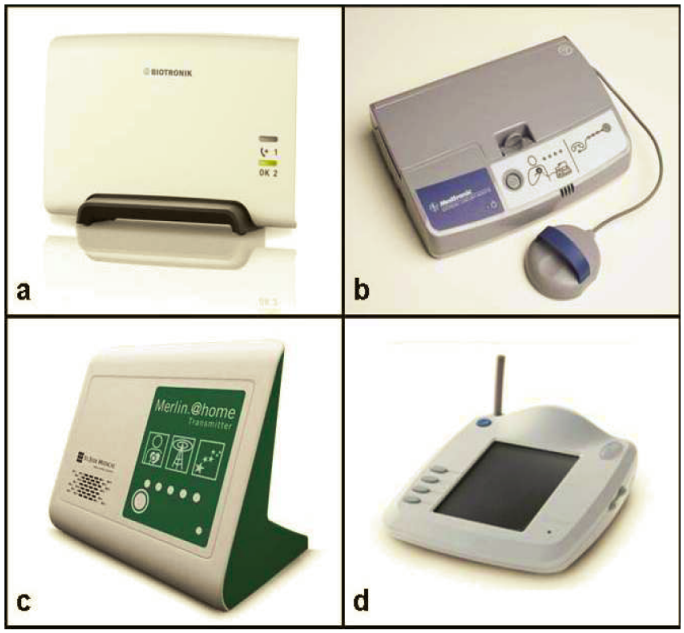

Figure 2.5: Various patient monitors for remote monitoring of implants (a: CardioMessenger , BIOTRONIK; b: CareLink-Monitor, Medtronic; c:Merlin@home, St Jude Medical; d: LATITUDE Communicator, Boston Scientific) [(Ed11].

The patient monitor is the interface between the implant and the data servers. The data transmis-

sion from the patient monitor to the manufacturer's data servers can be carried out via landline or mobile phone, with individual providers using both ways.
After the data transfer, the provider's data servers collect the data and present it to the physician[(Ed11]. In addition, all transmitted data is saved in the servers according to security regulators.
The treating physician can receive the data via fax, sms or internet. Vendors have developed password protected internet platforms, allowing for an access to patient data from any computer, including data concerning the system's integrity (programming of the aggregate, battery status, etc.) and diagnostic data (heart rate, atrial fibrillation, etc.). The data including both the ICD's integrity information and diagnostic data is transferred at scheduled times. Moreover, additional data transmissions can be carried out if the ICD delivers fibrillations. Thus, due to the modern remote monitoring systems offered by the vendor, complete datasets can be transmitted and presented. The manufacturers have also developed special user interfaces in order to allow immediate data review. Furthermore, the physician can ask the patient via the patient monitor to contact him.

2.3.5 ICD-programmer communications

Post-surgery, a health care practitioner can use an external programmer to perform diagnostics, read and write private data, and adjust therapy settings [HHBR+08]. The communications between ICDs and the respective programmer are wireless. Previous generations of pacemakers and ICDs communicated at low frequencies (near 175 KHz) with a short read range (8cm) and used low-bandwidth (50 Kbits per seconds) inductive coupling to relay telemetry and modify therapies [HHBF+08]. Modern devices use the Medical Implant Communications Service (MICS), which operates in the 402 to 405 MHz band and allows for much higher bandwidth (250 kbps) and longer read range (2 to 5 meters). As figure 2.6 illustrates, major pacemaker and ICD manufacturers now produce at-home monitors that wirelessly collect data from implanted devices and relay it to a central repository over a dial-up connection.

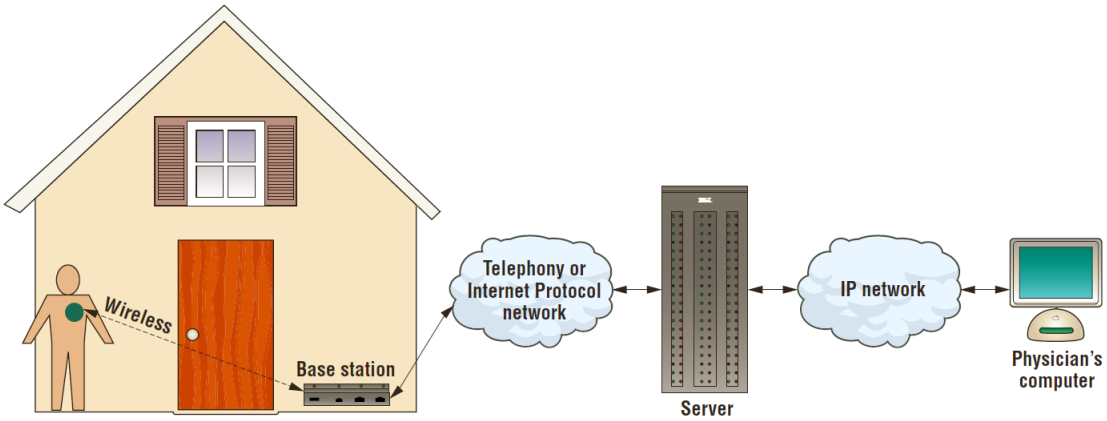

Figure 2.6: Recent implantable cardiac defibrillators provide home monitoring via wirless base stations that relay data to doctos with Web access [HHBF+08].

2.3.5.1 MICS and MedRadio

Prior to the establishment of the Medical Device Radiocommunications Service (MedRadio) - originally Medical Implant Communications Service (MICS) - in 1999, medical implant devices had to be magnetically coupled to external programmers or readers. This magnetic coupling

required that the device implanted in the patient needed to be in very close proximity to the external monitoring/control equipment, often necessitating body contact for proper operation. The FCC established the MICS to overcome these limitations of medical implant devices. The FCC concluded, with the agreement of representatives of the medical community and equipment manufacturers, that establishing a MICS would greatly improve the utility of medical implant devices by allowing physicians to establish high-speed, easy-to-use, reliable, short-range wireless links to connect such devices with monitoring and control equipment [FCC09].

MedRadio allows physicians and their patients to take advantage of the benefits of wireless technology to improve the medical care and capabilities of implanted medical devices, thereby improving these patients' quality of life. MedRadio not only permits faster data transfer rates between IMDs and external monitoring/control equipment, it reduces the risk of infection to patients, enhances the comfort of patients, and expands the freedom of movement of medical personnel working with the equipment. The MICS rules took effect on January, 2000. The rules changing the name of the service to MedRadio and expanding the designated frequency band took effect on August, 2009.

MedRadio incorporated the existing MICS spectrum at 402-405 MHz and added additional spectrum at 401-402 MHz and 405-406 MHz - called Medical Data Service (MEDS) in the United States of America - for a total of five megahertz of spectrum for implanted devices as well as devices worn on the actual body, as it is possible to see in the figure 2.7.

The MICS band was accepted worldwide [IKY13] at the time due to having better conductivity in the human body than any other ISM (industrial, scientific and medical) frequency band and having a communication range of about 3 meters, with 10 channels. In order to save power to the implanted device, only the external device is recommended to check each of the 10 channels to find a free one before starting a communication. After that, the external device announces the channel to the implants, and, in order to avoid false communication, an implant will only transmit in response to an external device, except when any emergency situation occurs. In emergency, implants send data immediately without checking the channel or waiting for a message from external device. Thus, for general communications in the MICS band, an external device sends a message to the implant and in response to that the implant sends data to it.

The rules for MedRadio accommodated body-worn as well as implanted medical devices. Under this framework, the rules for MedRadio service incorporates the MICS "core" band at 402-405 MHz which continues to be limited to implanted devices, and also includes two megahertz of newly designated spectrum in the adjacent "wing" band at 401-402 MHz and 405-406 MHz in which both body-worn and implanted devices are permitted [FCC10]. As a result, the legacy MICS and new MedRadio rules share many of the same licensing and technical requirements. Altogether, the MedRadio service provides a total of five megahertz of contiguous spectrum for advanced wireless medical radiocommunication devices serving a diverse range of diagnostic and therapeutic purposes in humans.

2.4 Security and privacy standards for IMDs

As it was previously stated in section 2.2, despite the recent advances in IMD technologies, the understanding of how device security and privacy interact with and affect medical safety and treatment efficacy is still limited. Established methods for providing safety and preventing unintentional accidents don't prevent intentional failures and other security and privacy problems (such as replay attacks or eavesdropping of information). Balancing security and privacy with safety and efficacy is becoming increasingly important as IMD technologies evolve. Halperin, D. *et al.* [HHBF+08]

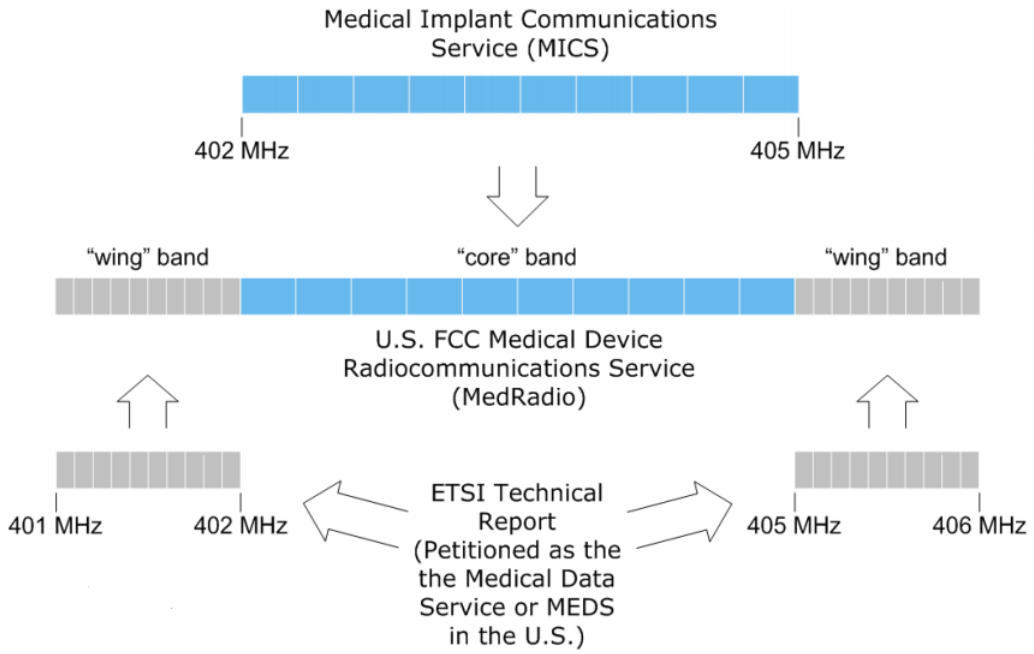

Figure 2.7: MedRadio (401-406 MHz) representation: MICS (402-405 MHz) and MEDS(401-402 and 405-406 Mhz).

presented some security and privacy standards for implantable medical devices that should always be taken into account and some will now be presented. In addition, they define the different possible classes of adversaries, which are also presented next. In the end, this section briefly discusses the current tension between safety/utility and security/privacy goals.

2.4.1 Safety and utility goals

Traditional IMD design goals include clinical safety and utility, with safety meaning that the IMD should do much greater good than harm, and utility meaning the IMD should be useful to both clinicians and patients. Reliability and treatment efficacy are encompassed in safety and utility, and these goals may sometimes conflict with IMD security and privacy goals.

- **Data access.** Data should be available to appropriate entities. In emergency situations, IMDs can provide useful information to medical professional when other records might be unavailable.
- **Data accuracy.** Measured and stored data should be accurate, presenting information that includes both measurements of physiological events and when those occurred.
- **Device Identification.** An IMD should make its presence and type known to authorized entities.
- **Configurability.** Authorized entities should be able to change appropriate IMD settings.
- **Updatable software.** Authorized entities should be able to upgrade IMD firmware and applications.
- **Auditable.** In the event of a failure, the manufacturer should be able to audit the device's operational history, but this information might differ from the data exposed to healthcare professionals and patient who access via the typical way.

- **Resource efficient.** To maximize device lifetime, IMDs should minimize power consumption. This includes minimizing computation, communication and data storage requirements.

2.4.2 Security and privacy goals

In order to understand the challenges of balancing security and privacy with safety and effectiveness, it is necessary to review how the standard principles of computer security, including confidentiality, integrity and availability, may extend to IMDs security.

- **Authorization.** Many goals of secure IMD design revolve around authorization, which has several broad categories:

  Personal authorization: specific sets of people can perform specific tasks. for example, patients or primary-care physicians might be granted specific rights after authentication of their personal identities. Depending on the authentication scheme, these rights might be passed on to other entities.

  Role-based authorization: an entity is authorized for a set of tasks on the basis of its role, such as physician, paramedic or patient.

  IMD selection: it is essential to ensure that an external and authenticated entity only communicates with the intended devices. Authorization and authentication in a medical setting can be highly sensitive to context [HHBF+08]. Regardless of the policy defined, an IMD should have the technological means to enforce the authorization goals.

- **Availability.** An adversary should not be able to mount a successful denial-of-service (DoS) attack against an IMD, including draining a device's battery, overflow its internal data storage media, or jam any IMD communications channel.

- **Device software and settings.** Only authorized parties should be allowed to modify an IMD or to otherwise trigger specific device behaviour. Physicians and/or device manufacturers should place bounds on the settings available to patients to prevent them from harming themselves. Similarly, the physician can have access to modify most device settings, but should not have an unrestricted access to the audit logs or debug modes. Furthermore, IMDs should only accept authorized firmware updates.

- **Device-existence privacy.** An unauthorized party should not be able to determine that a patient has one or more IMDs.

- **Device-type privacy.** In case the existence of the device is revealed, its type should still only be disclosed to authorized entities.

- **Specific-device ID privacy.** An adversary should not be able to wirelessly track individual IMDs. For that purpose, the use of persistent identifiers should be avoided.

- **Measurement and log privacy.** An unauthorized party should not be able to learn private information about the measurements or audit logs stored on the device. An adversary should not be able to learn private information about ongoing telemetry.

- **Bearer privacy.** An adversary should not be able to exploit the properties of an IMD with the objective to identify the bearer or extract private information about the patient.

- **Data integrity.** An adversary should not be able to tamper with past device measurements or log files, or induce misleading modifications into future data. No person should be able to

change the time an event occurred, modify its physiological properties, or delete old events and insert new ones. The patient's name, diagnoses, and other stored data should be tamper-proof.

2.4.3 Classes of adversaries

The following classes of adversaries proposed by Halperin, D. *et al.* [HHBF+08] are divided in four main categories, and further subdivided in other two subcategories:

- **Passive adversaries:** they eavesdrop on signals transmitted by the IMD and by other entities communicating with the device;
- **Active adversaries:** they can interfere with legitimate communications and initiate malicious communications with IMDs and external equipment;
- **Coordinated adversaries:** two or more adversaries might coordinate their activities to pose a coordinated attack - for example, one would be near the patient and the other one near a legitimate IMD programmer;
- **Insiders:** these are potential adversaries, including healthcare professionals, software developers, hardware engineers and even the patients themselves.

The following subdivision of the previous classes is done taking into account the equipment used by the adversaries:

- *Standard equipment:* adversaries might use commercial equipment for malicious purposes, by stealing them or being insiders;
- *Custom equipment:* adversaries may develop special customized equipment for eavesdropping or active attacks, which could have additional features to the commercial equipment, with no limitations to legal bounds.

2.4.4 Security goals vs Traditional goals

With the previous security goals defined, it shall become easier to analyse existing solutions and determine if their current security is sufficient for current threats. However, the tensions between the different goals are not easy to be surpassed, and will require experts from the medical and security communities, industry, regulatory bodies, patient advocacy groups, and all other relevant communities to collaboratively make decisions on both mechanisms and policies. The balance between security, privacy, safety and utility might differ depending on the IMD in question, and challenges like security versus accessibility, device resources, and usability, access control, open access with revocation and second-factor authentication are some of the tensions that should always be taken into account when developing new IMDs.

2.5 Existing ICD technologies for privacy and security

In this section, recent technologies that are directly or indirectly related to enhancing both privacy and security of information in ICDs are presented. It starts with the presentation of the zero-power defences proposed by Halperin, D. *et al.* [HHBR+08], moving on to the non-invasive security approach by Gollakota, S. *et al.* [GHR+11] called *Shield* and finishing with a brief presentation of current technologies from both the *Medtronic* and *St Jude Medical* companies.

2.5.1 Zero-power defences

In their work entitled *Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses*, Halperin, D. *et al.* [HHBR+08] showed that an ICD - although their experimentations were done using an older version - is potentially susceptible to malicious attacks that violate the privacy of patient information and medical telemetry, and may experience malicious alteration to the integrity of information or state, including patient data and therapy settings for when and how shocks are administered. Considering that standard approaches for security and access control may not always be suitable for IMDs due to tensions between security and safety, three new methods were proposed for enhancing security in ICDs: *zero-power notification*, *zero-power authentication* and *sensible key exchange*.

2.5.1.1 Zero-power notification

According to Halperin, D. *et al.* in their work [HHBR+08], it is possible to deter malicious activities by making patients aware of those activities. Their zero-power notification has the objective of alerting a patient to potentially malicious activities both by insiders using commercial programmers and by outsiders using custom attack hardware, thereby making patients effortlessly aware of remote communications. On some modern ICDs, triggering the magnetic switch causes the ICD to beep. Such beeping, being it intentional or not, represents a step towards the concept of patient awareness by way of audible alerts.
Their approach - called **WISPer** - tries to extend the concept of patient awareness also for RF-initiated actions. The WISPer builds upon revision of the Wireless Identification and Sensing Platform (WISP), and it wirelessly drives a piezo-element that can audibly warn a patient of security-sensitive events. The WISP is a wireless, battery-free platform for sensing and computation that is powered and read by a standards compliant Ultra-High Frequency (UHF) RFID reader [SSP+06]. To the reader, the WISP appears to be an ordinary RFID tag, and its platform includes a general-purpose programmable flash microcontroller - MSP430F1232 with 256 bytes of RAM and 8 Kilo bytes of flash memory - and implements the bi-directional communications primitives required by the Electronic Product Code (EPC) RFID standard, which allows it to communicate arbitrary sensor data via an EPC RFID reader by dynamically changing the ID it presents to the reader. The WISP harvests energy from a RF signal generated by a UHF RFID reader, and the authors affirm it may be possible to create similar hardware that operates at the frequency of current ICD programmers.
The WISPer adds to the WISP's base code a small C program that activates a piezo-element that is attached to the General-Purpose Input/Output (GPIO) ports of the WISP, as shown in figure 2.8. The WISPer will emit a constant chirping or produce vibration after receiving a sequence of wireless requests from the RFID reader, therefore informing the patient of the wireless interaction. Moreover, the WISPer continues to draw no energy from a battery and can issue alerts for all reprogramming activity, and so satisfying some security goals for IMDs without compromising any of its utility goals.

2.5.1.2 Zero-power authentication

The *zero-power authentication* harvests induced RF energy similar to the *zero-power notification* in order to power a cryptographically strong protocol that authenticates requests from an external device programmer.
Their second zero-power defence, the *zero-power authentication*, allows an IMD to verify that it is communicating with a real commercial programmer, and not an unauthorized software radio pro-

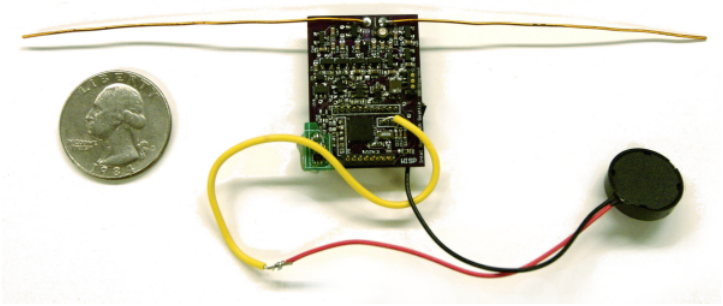

Figure 2.8: The WISP with an attached piezo-element [HHBR+08].

grammer), through a simple challenge-response protocol (figure 2.9) based on the RC5 encryption algorithm [Riv95].

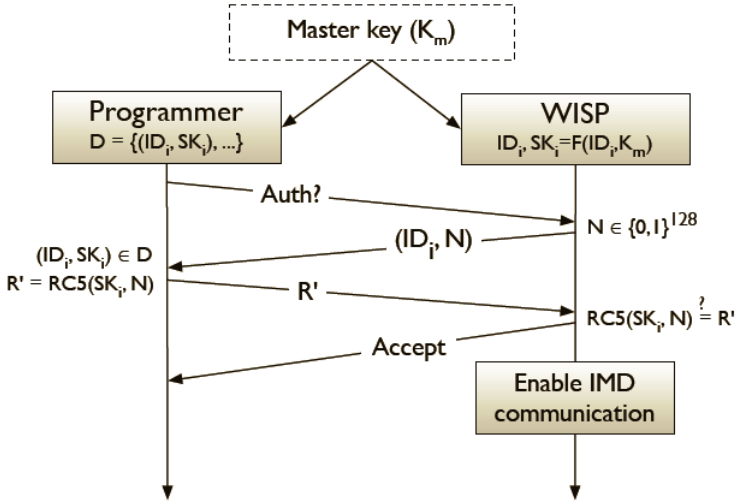

Figure 2.9: Protocol proposed by Halperin *et al.* [HHBR+08] for a communication between an ICD programmer and a zero-power authentication device - a WISP RFID tag, in the case of their prototype.

The model follows the following path:

1. All commercial programmers and WISP devices know a master key K_m.

   Each IMD has a serial number or identity ID_i, and an IMD-specific key $SK_i = F(ID_i, K_m)$ where F is any cryptographically strong pseudo-random function - such as Advanced Encryption Standard (AES).

2. The programmer transmits a request to authenticate to the WISP.

3. The WISP responds with its identity ID_i and a nonce N.

4. The programmer computes $SK_i = F(ID_i, K_m)$ to get the IMD-specific key.

5. The programmer returns to the WISP the response $R' = RC5(SK_i, N)$.

6. The WISP then computes the same value R' and verifies its result against the value received from the programmer.

7. The WISP finally sets a GPIO port high, which, if attached to or built into a real IMD, would inform the IMD that the WISP successfully authenticated a programmer.

In their prototype, Halperin *et al.* [HHBR+08] used a fixed nonce, and assumed that the programmer knows the nonce in advance in order to simplify their practical experiment. With this simplified model, they verified that, upon receiving the programmer response R', the WISP was able to perform its own RC5 encryption and verify equality using only harvested energy.

2.5.1.3 Zero-power sensible key exchange

With their third zero-power defence, Halperin *et al.* [HHBR+08] present a key-distribution technique that aims to complement both of their two previous defensive techniques by distributing a symmetric cryptography key over a human-perceptible sensory channel. The primary goal of this defence is to allow the patient to detect - be warned of - a key exchange while it occurs and proceeds with the following steps:

1. The programmer initiates the protocol by supplying an unmodulated RF carrier signal that could power the passive component of the IMD.
2. The IMD then generates a random value to be used as a session key and broadcasts it as a modulated wave.
3. A reader with a microphone placed in contact with the patient's body near the implantation site and directly connected to the programmer then receives the sound wave which is audible by the patient.
4. The patient is therefore aware and may consent to the authentication attempt, and the close proximity ensures it cannot be heard over background noise at any appreciable distance from the patient (without dedicated sensing equipment).
5. Once key exchange has been performed, the RF communication can occur over a longer range without fear of eavesdropping [HHBR+08].

This key exchange mechanism can be implemented on a device such as the WISP, and the authors of the previously mentioned mechanism affirm that the components performed key exchange without drawing power from a battery, and the exchange was clearly audible [HHBR+08].

2.5.2 *Shield*

In their work on non-invasive security for IMDs, Gollakota *et al.* [GHR+11] tried to overcome the problems directly related to the wireless connectivity with which data and commands are traded between IMDs and programmers, while also overcoming the challenge of addressing these attacks due to the difficulty of modifying or replacing already-implanted IMDs.
Therefore, they presented *shield*, a personal station to which is delegated the security of an IMD's wireless communications. In short, the *shield* uses a radio design that can act as a jammer-cum-receiver, allowing it to jam the IMD's messages, thus preventing other people from decoding them while being able to decode them itself, and also to jam unauthorized commands, providing confidentiality for private data - passive eavesdropper - and protecting the IMD from unauthorized commands - active adversary.
The *shield* works as follows (Fig 2.10):

1. It is located near the IMD and acts as a proxy.
2. An authorized programmer must first exchange messages with the *shield* if it wants to communicate with the IMD.

3. It relays the messages from the authorized programmer to the IMD, and sends back the IMD's responses.

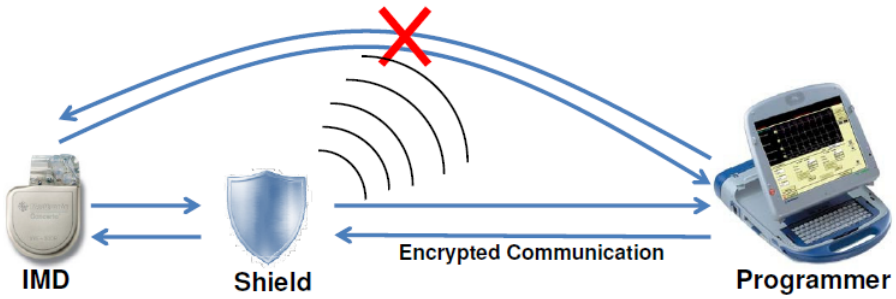

Figure 2.10: *Shield* architecture: it jams any direct communication with the IMD, while an authorized programmer communicates with the IMD only through the *shield* by an established secure channel.

With this architecture, Gollakota *et al.* state that the *shield* prevents any device other than itself from communicating directly with the IMD by jamming the messages sent to and from the IMD. The *shield* is also able to detect scenarios in which an adversary tries to overpower the shield's own transmissions to create a capture effect on the IMD and deliver an unauthorized message. By proxying the IMD communications without requiring patients to interact directly with the *shield*, their design aligns with IMD industry trends toward wireless, time and location-independent patient monitoring.
In terms of design, the *shield* uses two antennas: one for jamming and another for receiving. The jamming antenna transmits a random jamming signal, while the receive antenna simultaneously connects 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 the receive antenna to receive any signal without disruption from its own jamming signal. The properties of its design are summarized in the following items [GHR+11]:

- Transmit and receive chains connected to the same antenna;
- Antenna cancellation versus analog and digital cancellation;
- Channel estimation;
- Wideband channels.

To preserve the confidentiality of an IMD's transmission, the shield jams the IMD's signal on the channel. Jamming with a random signal provides a form of one-time pad, where only entities that know the jamming signal can decrypt the IMD's data. By jamming every packet transmitted by the IMD, the shield leverages two properties of MICS-band communications:

- An IMD does not transmit except in response to a message from a programmer. The *shield* can listen for programmer transmissions and anticipate when the IMD may start transmitting.
- An IMD transmits in response to a message from a programmer without sensing the medium. This allows the *shield* to bound the interval during which the IMD replies after receiving a message.

For countering active adversaries, the *shield* detects unauthorized packets and jams them by linearly combining the jamming signal with the unauthorized signal, causing random bit flips during decoding. This way, the IMD ignores these packets because they fail its checksum test.

The *shield* was the first system that simultaneously provided confidentiality for IMDs' transmissions and protects IMDs against commands from unauthorized parties without requiring any modification to the IMDs themselves [GHR+11]. Further, it may also help provide a complementary defence-in-depth solution to devices that feature cryptographic or other application-layer protection mechanisms, because it affords physical-layer protection.

2.5.3 Medtronic

Founded in 1949 and with its headquarters located in Minneapolis, Minnesota, Medtronic is the world's largest medical technology company [Med14a]. In 2013, more than 9 million people benefited from the company's medical therapies, which treat cardiac and vascular diseases, diabetes, and neurological and musculoskeletal conditions.
Among many others products and therapies, Medtronic offers a full line of defibrillation devices and services for physicians treating patients with tachyarrhythmias. Medtronic also offers the remote cardiac telemetry system called Medtronic CareLink, which connects cardiac device patients to their clinic from home or away. A clinician has 24/7 access to a wide range of trended reports via a Internet website, offering information comparable to an in-office visit [Med14b]. In addition, the physician can receive Medtronic CareAlert notifications which provide alerts to potential issues before they become problems.
According to Mithilesh [(Ed11], with the Medtronic CareLink system, the patient can collect data by holding an antenna over his implanted device. Thus, this data is captured by the antenna, downloaded by the monitor - CareLink Monitor - and transferred to the Medtronic CareLink Network. Through this network, patient data is transmitted from the IMD using a portable monitor that has to be connected to a standard telephone line for transferring the data to the physician. The system can be used for remote monitoring of implantable event recorders, and it also allows to transmit information on system and diagnostic data and Intracardiac Electrograms (IEGMs).

2.5.4 St. Jude Medical

Founded in 1976 and headquartered in St. Paul, Minnesota, USA, St. Jude Medical is a global medical technology leader focused on six key treatment areas [St.14a]: vascular disease, structural heart, heart failure, arrhythmias, neurological diseases and chronic pain. The company focuses on providing innovative, cost-effective medical technologies designed to transform the treatment of some of the world's most expensive epidemic diseases [St.14b].
St. Jude Medical Merlin.net Patient Care Network (PCN) allows efficient remote management - inlcuding scheduled transmission and daily alert monitoring - of patients with ICDs, pacemakers and CRT devices [St.13]. The system includes the following sub-systems [St.11]:

- One-screen Follow-up allows clinicians to view, print, schedule, export and archive from the Recent Transmissions page. This feature also saves time and simplifies follow-ups by allowing clinicians to take action on up to 50 patient files at once.
- DirectAlerts Notification is a physician notification system that provides physician-designated patient alerts between follow-ups.
- Mobile DirectAlerts Notification allows alert-triggered Electromyograms (EMGs) and reports to be viewed directly on a smartphone or mobile device; notifications are sent with a doctor's individualized security stamp.

- EHR Direct Export allows automatic export of transmission data from Merlin.net PCN to a clinic's electronic health record (EHR) system. This allows seamless integration of data so care teams can make informed clinical decisions more quickly, without the need for expensive intermediary systems. This feature meets the Integrating Healthcare Enterprise (IHE) guidelines, supporting Health Level-7 (HL7) standards.

- Inductive Merlin@home transmitters can now be used with newer Epic family devices and Atlas family devices as well as other newer devices, and will be issued to patients with newer Epic ICD or Atlas ICD implants. However, Housecall Plus transmitters will still be available for patients with older Epic ICDs and Atlas ICDs.

- SmartSchedule Calendar is an 18-month, rotating perpetual calendar that creates an automatic follow-up transmission schedule. Clinicians can specify length of time, including 91-day and 182-day periods, between transmissions to coordinate follow-ups with the clinic's reimbursement calendar.

- DirectCall Message is an integrated and automated patient communication system designed to save clinic time by reducing routine calls otherwise performed by medical office staff.

- DirectTrend Viewer provides dynamic views of device and clinical trends for comprehensive patient management.

Also in his previously referred work, Mithilesh [(Ed11] states the Merlin@home monitor is the core of the system. Data is transmitted from the ICD to the Merlin@home Transmitter wirelessly via RF in a daily basis, and from there via telephone line to the internet-based Merlin.net server. With Remote Care from St. Jude Medical, the patient's Merlin@home transmitter allows to have the ICD checked from the comfort of the person's own home, reducing the number of scheduled clinic visits that patient has to make [St.08]. The information transmitted is the same as that gathered during an in-office visist. The transmitted data is uploaded to Merlin.net PCN, a web based data management system [St.08], with the patients' password and so only accessible to authorized users.

2.5.5 Backoffice Communications

Current data transmission architectures for systems as the IMDs require data to be transmitted from the IMD to multiple entities including websites, hospitals, clinics, regulatory companies, and physicians. This *backoffice* needs the data - both telemetry and ICD logs - in order to transmit it between the patient and the physicians. Through these backoffice application, the patient and physicians should be able to keep a constant connection between them and receive alerts. Data is sent to a dedicated data bank where it is stored and where automatic systems can process and analyse this data and automatically edit it for better understanding of the end users. Therefore, the constant telemetry communications between machines is a requirement for these kind of architectures.
In this section the WebSphere Message Queuing (MQ) Telemetry is introduced. The WebSphere MQ is IBM's (International Business Machines Corporation) implementation of the MQTT (Message Queuing Telemetry Transport) protocol. A brief introduction to the concepts of *internet of things*, machine to machine (M2M) communications and security in M2M systems, MQTT according to IBM [Red12], as well as MQTT's basic concepts and benefits of its use shall also be presented.

2.5.5.1 Internet of Things

Internet of Things: Thousands of millions of interconnected smart devices measuring, moving, and acting upon, sometimes independently, all the bits of data that make up daily life. The Internet of Things includes the example of a doctor examining a patient from a distance, but also smart metering, monitoring traffic flows, and many other similar examples. It will not be just people interacting with devices, but devices interacting with each other (M2M), creating what might eventually become something of a global central nervous system [Red12].

2.5.5.2 M2M

As technology evolves, it is increasingly common for devices to be connected to each other, as is illustrated in figure 2.11. This type of connection has created a need for efficient machine-to-machine (M2M) communication protocols [Red12]. The MQTT protocol is ideal for use in M2M communication. It enables connectivity that extends beyond smart devices to some of the smallest remote devices and sensors, including devices with limited processing or network abilities. This extension makes MQTT a critical component in self-managing M2M networks. Because MQTT is highly scalable, it is possible to create systems that involve hundreds or even thousands of remote sensors or devices.

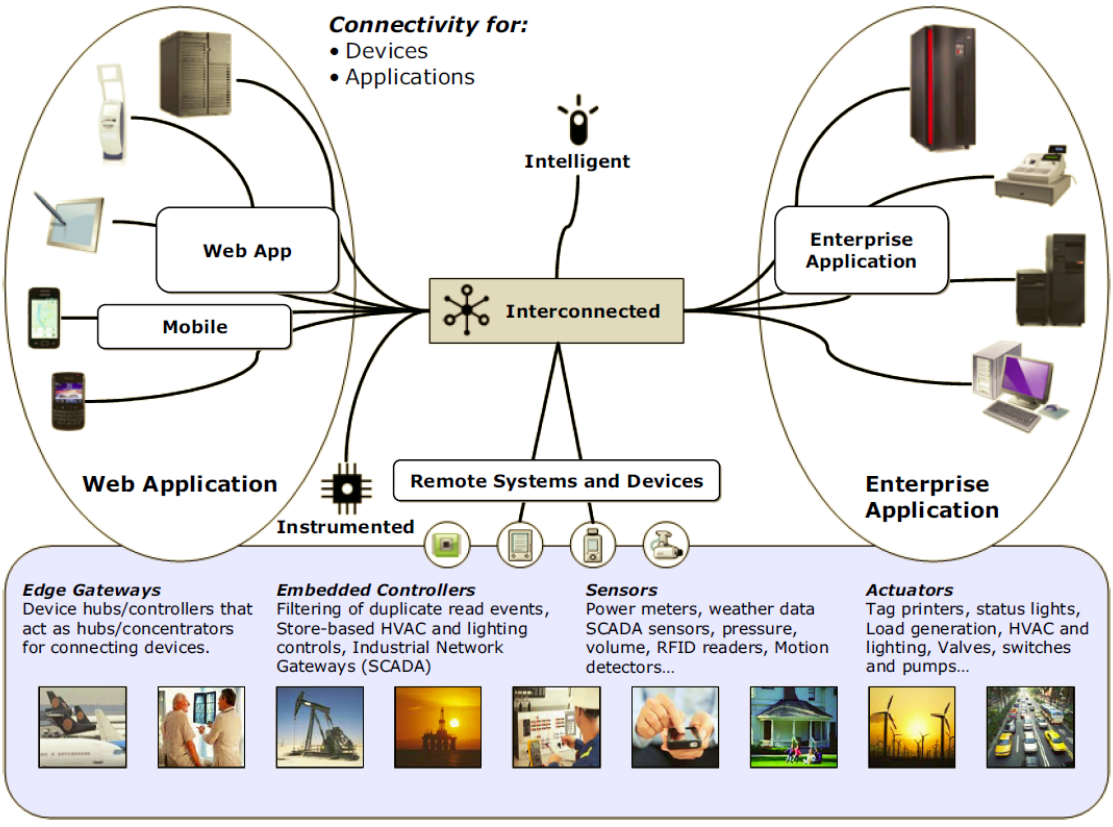

Figure 2.11: Machine-to-machine connectivity according to IBM [Red12], and some of its applications.

2.5.5.3 Security in M2M systems

The most significant part of the Internet of Things is the interconnection between machines, which is called M2M. The most important requirements of M2M are typically to be small, inexpensive,

unattended by humans and that communicate with each other through the wireless network. There are three key factors that are vital for secure communications of Machine-to-Machine [Vah13]:

- Authenticity - assure identity of both parties;
- Confidentiality - avoid eavesdrop;
- Availability - appropriate user and use.

M2M networks are more vulnerable to security threats due to their longevity and the difficulty of software and firmware updates. An attacker can illegally access to the data on M2M devices though eavesdropping user data, signalling data and controlling data on the wireless link in public place. Therefore, the wireless link should be designed to prevent eavesdropping or unauthorized access with two-way authentication mechanism of network and the corresponding encryption algorithm. Hence, M2M equipment needs to have integrity and ensure data confidentiality, characteristics which are provided using the MQTT protocol.

2.5.5.4 MQTT

MQ Telemetry Transport is a messaging protocol that is lightweight enough to be supported by the smallest devices, yet robust enough to ensure that important messages get to their destinations every time. This ensures MQTT is widely used for M2M telemetry communications. With MQTT, devices such as smart energy meters, cars, trains, satellite receivers, and personal health care devices that can communicate with each other and with other systems or applications [Red12].
MQTT's publish/subscribe architecture is designed to be open and easy to implement, with up to thousands of remote clients capable of being supported by a single server. These characteristics make MQTT ideal for use in constrained environments where network bandwidth is low or where there is high latency and with remote devices that might have limited processing capabilities and memory. Benefits include [Red12]:

- Extends connectivity beyond enterprise boundaries to smart devices;
- Offers connectivity options optimized for sensors and remote devices;
- Delivers relevant data to any intelligent, decision-making asset that can use it;
- Enables massive scalability of deployment and management of solutions.

MQTT minimizes network bandwidth and device resource requirements while attempting to ensure reliability and delivery. This approach makes the MQTT protocol particularly well-suited for connecting machine to machine (M2M), which is a critical aspect of the emerging concept of an Internet of Things.
MQTT protocol's characteristics include [Red12]:

- Open and free for easy adoption;
- Publish/subscribe messaging model that facilitates one-to-many distribution:

  Sending applications or devices do not need to know anything about the receiver, not even its address;
- Ideal for constrained networks (low bandwidth, high latency, data limits, and fragile connections):

  MQTT message headers are kept as small as possible. The fixed header is just two bytes, and it's on demand, push-style message distribution keeps network utilization low;

- Multiple service levels allow flexibility in handing different types of messages:

  Developers can designate that messages will be delivered at most once, at least once, or exactly once;

- Designed specifically for remote devices with little memory or processing power:

  Minimal headers, a small client footprint, and limited reliance on libraries make MQTT ideal for constrained devices;

- Easy to use and implement with a simple set of command messages:

  Many applications of MQTT can be accomplished using just CONNECT, PUBLISH, SUBSCRIBE, and DISCONNECT;

  Built-in support for loss of contact between client and server:

  The server is informed when a client connection breaks abnormally, allowing the message to be re-sent or preserved for later delivery.

In their work, Lampkin *et al* also present the basic concepts of MQTT [Red12]:
Publish/Subscribe:

- Clients can subscribe to topics that pertain to them and thereby receive whatever messages are published to those topics;
- Clients can publish messages to topics, thus making them available to all subscribers to those topics.

Topics and Subscriptions

- Messages in MQTT are published to topics;
- Clients, in turn, sign up to receive particular messages by subscribing to a topic.

Quality of Service (QoS) levels

- MQTT defines QoS levels for message delivery, with each level designating a higher level of effort by the server to ensure that the message gets delivered;
- Higher QoS levels ensure more reliable message delivery, but might consume more network bandwidth and subject the message to delays (latency).

Retained messages:

- The server keeps the message even after sending it to all subscribers;
- If there is a new subscription submitted for the same topic, any retained messages are then sent to the new subscribing client.

Clean sessions and durable connections:

- When a client connects to the server, it sets the clean session flag:

  True: all of the client's subscriptions are removed after disconnecting from the server;

  False: the connection is treated as durable, with the client's subscriptions remaining in effect after disconnection.

Wills:

- When a client connects to a server, it can inform the server that it has a will (message) that should be published to a specific topic (or topics) in the event of an unexpected disconnection.

2.5.5.5 Benefits of using MQTT

Using the MQTT protocol extends IBM's WebSphere MQ to tiny sensors and other remote telemetry devices that might otherwise be unable to communicate with a central system or that might be reached only through the use of expensive, dedicated networks [Red12].
Network limitations can include limited bandwidth, high latency, volume restrictions, fragile connections or prohibitive costs. Device issues can include limited memory or processing capabilities, or restrictions on the use of third-party communication software. In addition, some devices are battery-powered, which puts additional restrictions on their use for telemetry messaging.
MQTT was designed to overcome these limitations and issues. MQTT includes the following underlying principles:

- Simplicity;
- Use of a publish/subscribe model;
- Minimal maintenance;
- Limited on-the-wire footprint;
- Continuous session awareness (will feature);
- Local message processing (limited processing power of devices);
- Message persistence (QoS);
- Agnostic regarding data types.

2.5.5.6 Example implementations of MQTT: Healthcare

MQTT can be used to tie together the various types of remote smart devices and applications that are measuring, monitoring, and in some cases controlling the world today. The healthcare industry emerges as a potential scenario where WebSphere MQ and the MQTT protocol might be used to improve communication to and from remote devices or applications. The key entities are the pharmaceutical companies, medical research, hospitals and nursing homes. Possible use cases are for instance a medical clinic remotely tracking the vital signs of at-risk patients to help prevent sudden crises that might arise after a patient goes home, or a research team monitoring chemical reactions in a remote laboratory and alerts the chemist when a particular result or stage of development is achieved[Red12].
In 2006 [IBM06], the International Business Machines Corporation (IBM) announced their collaboration with St. Jude Medical to produce the St. Jude Medical Merlin Patient Care System, a portable system that programs St. Jude Medical's ICDs and pacemakers, as mentioned earlier. The medical organization wanted to create an at-home, cardiac ICD monitoring solution that needed to address the following aspects of patient care [Red12]:

- Monitoring cardiac patients after they leave the hospital;
- Improving the efficiency of later check-ups;
- Meeting new industry data-capture standards.

Among helping implementing other features to the system, IBM used the MQTT technology in the Merlin@home monitoring device for allowing the product to call back to the doctor/hospital whenever there is a data update, which the hospital then stores [rea09], as illustrated in figure

2.12. The patient monitor sends the diagnostic data over the Internet to the central messaging server, where it is therefore handed off to an application that analyses the reading and alerts the medical staff if there are signs the patient might be having difficulty.

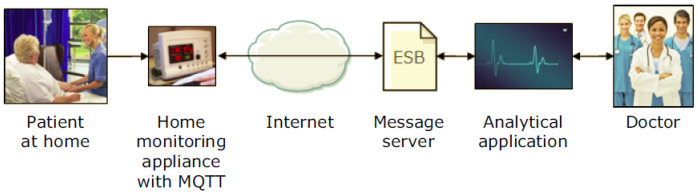

Figure 2.12: Home ICD monitoring solution with MQTT [Red12].

The solution allows the organization - in this case St. Jude Medical - to provide a higher level of post-hospital patient care and early diagnosis of follow-up issues. It also saves money for both the organization and its patients, because there is less need for travel by either party and also because patients who are doing well might be allowed to come in for check-ups less often.

2.5.5.7 MQTT security

Three concepts are fundamental to MQTT security: identity, authentication, and authorization [Ian13]. Identity is about naming the client that is being authorized and given authority, authentication is about proving the identity of the client, and authorization is about managing the rights that are given the client.
A MQTT client can authenticate the MQTT server that it connects to, and the server can authenticate the client that is connecting to it. A client authenticates a server with the SSL protocol, while a MQTT server authenticates a client with either the SSL protocol, or with a password, or both. If the client authenticates the server, but the server does not authenticate the client, the client is often known as an anonymous client. It is common to establish an anonymous client connection over SSL, and then authenticate the client with a password encrypted by the SSL session. It is much more common to authenticate a client with a password than with a client certificate, because of the certificate distribution and management problem.
Authorization is not part of the MQTT protocol. It is provided by MQTT servers. What is authorized depends on the server application logic. MQTT servers are publish/subscribe brokers, and useful MQTT authorization rules control which topics a client can publish or subscribe to. If a MQTT client can administer the server, more authorization rules control which clients can administer different aspects of the server. The number of possible clients is huge, so it is not feasible to authorize each client separately. A MQTT server will have a means to group clients by profiles, or groups. The identity of a client, from the point of view of access and authorization, is not something that is unique to a MQTT client. The identity of a client should not be confused with the client identifier. They might be the same, but are commonly different. For example, there may be a user name that is common across a number of services, and some of these services co-operate in single sign-on. An enterprise scale MQTT server is likely to call an authorization service that offers common identities and authorities for different applications.

2.6 Near Field Communication

Near Field Communication (NFC) is a standards-based short-ranged wireless connectivity technology designed to make transactions, exchange digital content, and connect electronic devices with

each other by a single touch. NFC is compatible with hundreds of millions of contactless cards and readers already deployed worldwide [NFC14d]. NFC complements many popular consumer level wireless technologies, by utilizing the key elements in existing standards for contactless card technology (ISO/IEC 14443 [NFC12b] and JIS-X 6319-4 [NFC12a]). NFC enables devices to share information at a distance that is less than four (4) centimetres with a maximum communication speed of 424 kbps. This way, users can share business cards, make transactions, access information from a smart poster - objects in or on which readable NFC tags have been placed. For the end user, the most significant benefits this technology brings are easy connections, quick transactions, and simple data sharing.

2.6.1 Modes of operation

The NFC standard specifies three modes of operation: card emulation, peer-to-peer, and reader/writer.

- **Card Emulation Mode:** enables the devices to act like smart cards, allowing users to perform transactions with just a touch. In this mode, the device communicates with an external reader much like a traditional contactless smart card. Furthermore, adding NFC to a contactless infrastructure enables two-way communications.
- **Peer-to-Peer Mode:** enables two devices to communicate with each other to exchange information and share files, so that users can quickly share contact information and other files with a touch.
- **Reader/Writer Mode:** enables devices to read information stored on inexpensive NFC tags embedded in smart posters and displays. This mode is the most important of the three in the context of this dissertation.

2.6.2 Characteristics

NFC provides a range of benefits to consumers and business through its inherent advantageous characteristics [NFC14b]:

- **Intuitive:** NFC interactions require no more than a simple touch.
- **Versatile:** NFC is ideally suited to the broadest range of industries, environments, and uses.
- **Open and standards-based:** The underlying layers of NFC technology follow universally implemented International Organization for Standardization (ISO), European Computer Manufacturers Association (ECMA), and European Telecommunications Standards Institute (ETSI) standards.
- **Technology-enabling:** NFC facilitates fast and simple setup of wireless technologies.
- **Inherently secure:** NFC transmissions are short ranged (from a touch to a few centimetres).
- **Interoperable:** NFC works with existing contactless card technologies.
- **Security-ready:** NFC has built-in capabilities to support secure applications.

NFC enables users to quickly and easily transfer information between devices with a simple touch, with the proximity ensuring that the information shared is the information you want to share, by greatly avoiding man-in-the-middle and eavesdropping situations.

In terms of its relationship with healthcare, personal health monitors recording vital data can be read by an NFC reader/writer, which could be the patient's smartphone, by simply touching the reader to the health device [NFC14b]. The physical proximity that NFC require guarantees that the operator has confidence in which data is read at what time, thus greatly reducing the chance of human error. NFC-enabled devices, with their simple instructions ("just touch"), allow patients of every age to monitor their health status autonomously. Further ahead in this dissertation, a proof of concept based on this NFC characteristic is presented.

2.7 NFC-WISP

The NFC-WISP is designed to enhance near-field RFID applications that utilize NFC smart phones and RFID reader [Ala11]. The NFC-WISP, as presented in figure 2.13, consists in a software defined 13.56 MHz RFID tag, where on the printed circuit board MSP430 microcontroller handles all of the bidirectional communications, as well as user-defined sensing and computational tasks. Being designed as a printed circuit board rather than a custom integrated circuit, the NFC-WISP is easily reconfigured and debugged.

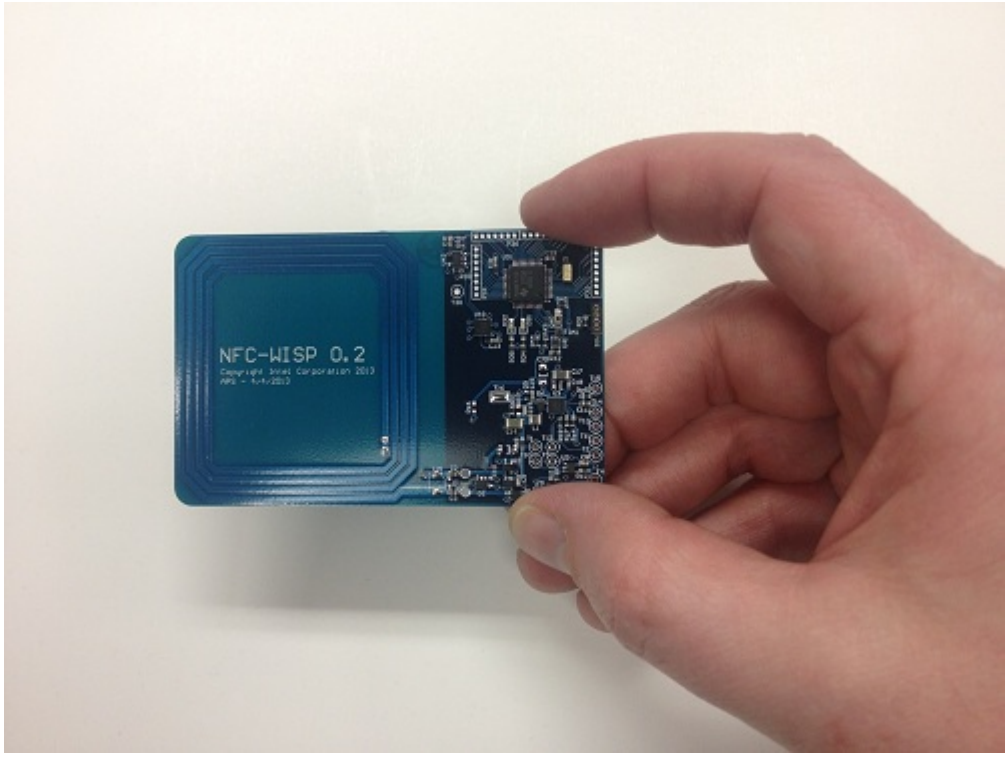

Figure 2.13: The NFC-WISP device shown by its author, Alanson Sample [Ala11].

The application called the Wirelessly Powered Bistable Display Tag [Ala13] shows an NFC-WISP working with an E-Ink display. Here a NFC-enabled phone generates RF signals carrying both the information and energy necessary to update the display. After the update is complete, the display continues to present the information with no further power input. This works as an example of implementation of the kind of applications that can be created using the technology of the NFC-WISP.

2.8 GSM related concepts

2.8.1 Global System for Mobile communications

Global System for Mobile communications (GSM) is an open, digital cellular technology used for transmitting mobile voice and data services [ETS14]. GSM supports voice call and data transfers speeds of up to 9.6 kbps, together with the transmission of SMS (Short Message Service). GSM operates in the 900 MHz and 1.8 GHz bands in Europe and the 850 MHz and 1.9 GHz bands in the United States. The use of harmonised spectrum across most of the globe, combined with GSM's international roaming capability, allows travellers to access the same mobile services at home and abroad. GSM enables individuals to be reached via the same mobile number in up to 219 countries. Terrestrial GSM networks now cover more than 90% of the world's population. GSM satellite roaming has also extended service access to areas where terrestrial coverage is not available.

2.8.2 Universal Mobile Telecommunications System

The Universal Mobile Telecommunications System (UMTS) is one of the third generation (3G) mobile technologies. Unlike the GSM, UMTS provides high data rates and low cost for data transmission, and is secure against the known GSM attacks [Vah13], GSM only supports subscriber authentication and encryption, whereas UMTS also provides integrity protection of the signalling traffic between the mobile station and a network. The radio interface is the main difference between UMTS and GSM, having UMTS applying other multiple access and coding techniques, and thus offering larger bandwidth per channel and more flexible channel coding.

2.8.3 Universal Integrated Circuit Card

The Universal Integrated Circuit Card (UICC) is the smart card used in mobile terminals in GSM and UMTS networks. UICC is the best and only universal application delivery platform that works with any 3G or 4G device, and ensures the integrity and security of all kinds of personal data [Vah13]. A UICC contains a Subsriber Identity Module (SIM) application in a GSM network and a USIM (Universal Subsriber Identity Module) in a UMTS network. Thus, it is possible for the same smart card to give access to both GSM and UMTS networks while also providing data storage and software in a device with a small size.

2.8.4 Subscriber Identity Module

The SIM (Subsriber Identity Module) is a removable smart card based on an embedded integrated circuit chip, with its most important property from a security point being its tamper-resistance [Vah13]. The smart card itself is called UICC, with the SIM being a logical module running on that smart card. Portability is the main property of smart cards, making it possible for GSM subscribers to move a SIM from one terminal device to another. A SIM card has three identification data fields: the International Mobile Subscriber Identity (IMSI) which is the identification of the card with the network; the Mobile Station International Subscriber Directory Number (MSISDN) is the mobile device's "telephone number"; the ICCID - Integrated Circuit Card ID - is the serial number for that SIM card. The SIM card is a smart card with a microprocessor that includes CPU, RAM, ROM, EEPROM, and a serial communication module. The SIM's current design enables download of applications via Over-the-Air (OTA).

Chapter 3

Revisiting RFID and Security Mechanisms in ICDs

This chapter describes the work done in the scope of this dissertation in order to investigate the security status of various technologies and presents suggestions that relate to the security of ICDs. This section starts with a look at intercepting and eavesdropping RF communications, in particular we study the intercepting of RF signals from a remote keyless entry (RKE) key fob emits when sending the lock and unlock command to a car. Furthermore, one of the Zero-power Defenses proposed by Halperin *et al.* already mentioned in section 2.5.1.2, the Zero-Power Authentication, is analysed in terms of both approach and security. In the end, the *shield* technology proposed by Gollakota *et al.*, which is presented in section 2.5.2, is also briefly discussed.

3.1 Intercepting RF communications of key fobs

A key fob is a type of security token: a small hardware device with built-in authentication mechanisms [Mar05]. Just as the keys held on an ordinary real-world key chain or fob control access to the owner's home or car, the mechanisms in the key fob control access to network services and information. The key fob provides two-factor authentication: the token is used - in this case - in place of a password to prove that the customer is who they claim to be, and in this case acting like an electronic key to access something. Because a key fob is a physical object, it is easy for the owner to know if it has been stolen. In comparison, a password can be stolen (or guessed) and used for an extended period before - if ever - the theft is detected.
The very first car key fobs in the early 90s used a static code system [And11]. To combat the obvious security problem due to using a static code, rolling codes were introduced. With the rolling codes system, the key fob has a built in counter that is incremented every time the button is pressed. This counter is encrypted by the fob before being transmitted. The car remembers what the count was last time the doors were successfully unlocked (i), and the next time it receives the signal to unlock, the car then decrypts the data and checks if the counter value is somewhere between $i+1$ and $i+n$, where n could be any manufacturer-defined number. Both the fob and the car store the secret encryption key that enables this process to take place.
In Europe, RKE key fobs typically transmit on 433.92 MHz, whilst in the US and Japan 315 MHz is used. To attempt intercepting the data transmitted by these kind of fobs, the author of this dissertation used the following equipment, as shown in figure 3.1:

- A personal computer with the HDSDR version 2.70.
- A RTL2832U device with an antenna.
- A RKE key fob from a Peugeot 206.

The RTL2832U is a high-performance digital video broadcasting - terrestrial (DVB-T) coded orthogonal frequency division multiplexing (COFDM) demodulator that supports a USB 2.0 interface. The RTL2832U complies with NorDig Unified 1.0.3, D-Book 5.0, and EN300 744 (ETSI specification) [Rea14]. Essentially, it is an integrated circuit produced by Realtek Corporation to demodulate DVB-T signals and send them to a host computer over USB. It has an additional

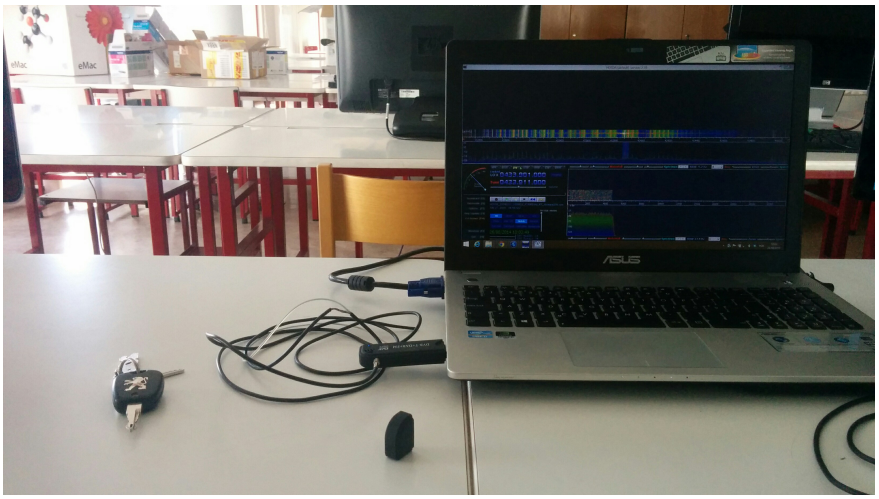

Figure 3.1: Equipment used to intercept the RF signal of a RKE key fob. From left to right: RKE key fob of a Peugeot 206, and a RTL2832 dongle attached through a USB 2.0 port to a laptop with Microsoft Windows 8.1 running HDSDR.

mode designed to allow reception of consumer FM radio, and this is achieved by capturing wave samples and forwarding them to the host computer for demodulation and playback. These sort of devices have a wide tuning range which typically goes from 64 MHz to 1.7 GHz, making them very flexible and very cheap software defined radio (SDR) receivers when allied to SDR programs, such as High Definition Software Defined Radio (HDSDR) [Ama13]. The SDR mode of the chip is theoretically capable of capturing up to 3.2 MHz of the RF spectrum at one time. However, the largest sample rate successfully received without sample loss is 2.8 MHz, which means any signal to be demodulated must fit within 2.8 MHz of the RF spectrum. For comparison, a consumer FM radio signal has a bandwidth of around 120 KHz, and a TV signal (with audio) is 6-8 MHz.
The High Definition Software Defined Radio (HDSDR) is a freeware SDR program for Microsoft Windows 2000/XP/Vista/7/8/8.1, which typical applications are radio listening, ham radio, short-wave listening, radio astronomy, non-directional beacon hunting, and spectrum analysis [Mar13].
In the next section, the procedure for intercepting the RF signals of the key fob will be describe. The key fob commands intercepted are the open and close commands and the operating system of the host is Microsoft Windows.

3.1.1 Procedure to intercept the signal

1. Download and install the Zadig drivers [Sco13] if it is the first time using the RTL2832U dongle. After extracting the files, plug in the RTL2832U dongle. Open the Zadig.exe file. Install the drivers through the Zadig software by listing all devices in the options menu, in which the device will show as a bulk-in interface (interface 0) device or just RTL2832.

2. Download and install the HDSDR [Mar13].

3. Download the ExtIO.dll for RTL2832U devices [Git13]. Copy the file ExtIO_RTL.dll to the directory where the HDSDR was installed.

4. Start HDSDR. If ExtIO_RTL.dll is the only ExtIO.dll file, HDSDR will start immediately. If there is any additional ExtIO file in the directory from a previous install, the file manager page will appear, where the ExtIO_RTL.dll should be selected.

5. Once HDSDR is running, click on the ExtIO button next to the frequency display. This will bring up the ExtIO control panel, where it is possible to set the gain of the RTL2832U device for best performance.

6. After that, in the frequency manager menu, select FM. Then, adjust the frequency manager (FreqMgr) to tune around the 433.910.000 frequency value.

7. In the option FM-BW, adjust it to 48000.

8. Push the open button in the Peugeot 206 RKE key fob.

9. If the volume is high enough, it is possible to hear a *click* from it.

10. Adjust the Waterfall, Spectrum Zoom and Speed according to the feedback received, so it is possible to see the increase of amplitude caused by the opening command (figure 3.2).

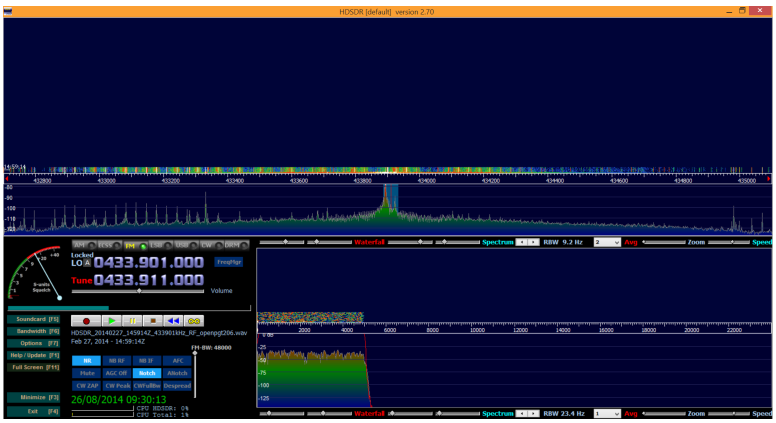

Figure 3.2: HDSDR showing the amplitude spike resultant of the opening command from the RKE key fob.

11. The experiment repeats itself for the close button of the RKE key fob (figure 3.3).

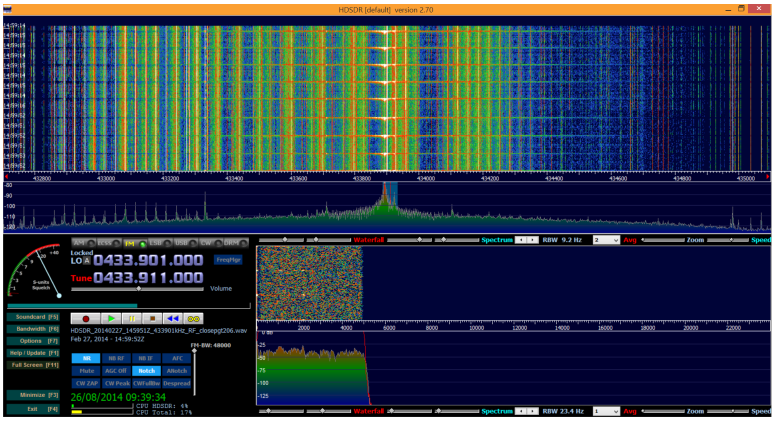

Figure 3.3: HDSDR showing the amplitude spike resultant of the closing command from the RKE key fob.

12. For further analysis of the data, it is possible to record the experiment using the record button.

This concludes the tutorial on how to intercept the RF signal from a RKE key fob. With this experiment, we intended to demonstrate how easy it is to intercept any kind of RF. If the tune

in the frequency manager menu of the HDSDR was changed to the MICS band frequency, it would also be possible to intercept the RF communications between an ICD and its programmer. By intercepting and recording this communication, an adversary can then perform the reverse-engineering of the signal - through demodulation and decoding - and obtain the plaintext of the communication, much likely Halperin *et al.* [HHBR+08] explained in their work.

3.2 Analysis of the authentication protocol suggested in zero-power authentication

In their work of *Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses* [HHBR+08], Halperin *et al.* present their approach for increasing the security of communications between an ICD and its programmer, called *zero-power authentication* and was already introduced in this dissertation in section 2.5.1.2.
In their reverse-engineering, Halperin *et al.* retrieved plaintexts containing private data from the interception of the communications between the ICD and its programmer. They also state the proposition of cryptographic extensions that require significant modifications to current ICD designs has to be done in a very careful way, mainly due to power cost. Without detailed knowledge of the inner workings of ICDs, it is not possible to accurately assess the cost of adding additional or stronger cryptographic mechanisms to existing devices. Despite that, it seems very accurate that the presence of a intermediate device working as a *checkpoint* - like the WISP - is a very decent approach to increase both security and privacy of ICDs and their communications with the respective programmers.
In the following subsections the authentication protocol used by Halperin *et al.* will be discussed.

3.2.1 Presentation of the authentication protocol

In their description on how their challenge-response protocol works (as seen in figure 2.9), Halperin *et al.* introduce some discrepancies in the notation that they used, in particular discrepancy of the variable names used in the written text and in the figure used to illustrate the protocol.For instance, in the picture they have $SK_i = F(ID_i, K_m)$, when in the paragraph in which they describe the protocol they write $K = f(K_M, I)$. This kind of nomenclature errors hinder the reader's capacity to understand the protocol, and so this poses as a presentation critic to their authentication protocol.

3.2.2 Fixed nonce

For their practical experimentations, Halperin *et al.* used a fixed nonce and, in order to increase the simplicity of the demonstration, assumed that the programmer knows the nonce in advance. A nonce N is a pseudo-random number used to ensure old communications cannot be reused in replay attacks, its use in the authentication protocol is described in section 2.5.1.2. Nonetheless, the authors do recognize the nonce should appear random to an adversary in a real implementation of their protocol. Thus, they suggest the nonce could be either generated with RC5 in counter mode, or through the extraction of random bits from the static random-access memory (SRAM). The nonce is calculated and stored by the WISP device, and in their work they affirm it is possible to run RC5 on a WISP with the nonce generated with RC5 in counter mode. This emerges as a correct way to generate the nonce, given that all the processing and power requirements are exclusively the WISP's responsibility and thus no extra work is needed by the ICD. Therefore,

the WISP can continue to be autonomous in terms of power requirements when calculating the nonce: an important pseudo-random value that will be used for the programmer to calculate the IMD-specific key.

3.2.3 The key SK_i

The key SK_i will result from the output of a pseudo-random function like the AES (as suggested in their work), receiving a master key K_m which is known by all programmers, and the serial or identity number ID_i.
The use of AES is adequate to the situation, as the pseudo-random function is a symmetric block cipher well-suited for platforms with limited resources, which is the case. At the same time, it has a widely acceptable level of security, supporting 128, 192 or 256 bits sized keys [Jam03].
K_m is directly related to the issue of key management. As the key K_m is a shared (public) secret its use in a prototype is reasonable. However this is not the case for a large-scale deployment since the shared key material would need to be placed in implanted devices, hospitals, clinics, ambulances, IMD programmers.
This way, the risk that these secret keys may be disclosed may be unacceptable due to the high number of entities that would have access to these devices and communications; any unauthorized party with access to this key could decrypt the transmissions. Even more, the scheme in general fails to address the key revocation problem, and is therefore ill-suited to situations in which key material might be compromised. Despite all these disadvantages, the proposed system is still no less secure than the open-access model of conventional systems.
Halperin *et al.* suggest using S/KEY [Hal94] with periodic updates of programmer keys in order to mitigate the time-window in which the attacker can use compromised keys. The use of S/KEY would not significantly change the overall model. S/KEY is a one-time password system used to authenticate public systems with short-term passwords (in terms of time), using 64 bits. Given that a user's real password is combined in an offline device with a short set of character and a decrementing counter to form a single-use password, adversaries that may sniff a password will have a useless set of characters that will not give them access to the private information. Despite their strong security being originated in the difficulty of inverting *hash* functions, S/KEY systems are still vulnerable to man-in-the-middle attacks (MiTM), something that is somewhat unlikely to occur in this kind of communications due to the short-ranged communications, but still possible. The implementation of this cipher would have to be done to an intermediate device much like the WISP, given that its power and processing consumption would still be inside the usable limits for the device to still harvest enough energy to keep itself functioning.

3.2.4 Calculating R' with $RC5(SK_i, N)$

$R' = RC5(SK_i, N)$ is calculated using the nonce N and the key SK_i obtained through the calculation of $SK_i = F(ID_i, K_m)$. This answer R' is first calculated by the programmer after receiving the nonce and the SID_i from the WISP, therefore constituting the essence of the challenge-response protocol.
The RC5 cipher [Riv95] is a fast symmetric block cipher suitable for hardware or software implementations. RC5 has a variable word size (32, 64 or 128 bits), a variable number of rounds (0 to 255), and a variable-length secret key (0 to 2040 bits). The encryption and decryption algorithms are exceptionally simple. In general, this block cipher is standardly used with a block size of 64 bits, a key with 128 bits and 12 rounds. The number of rounds used will be the main power consumption factor, as the greatest its number is, the more inherent processing load. However, it

is globally suggested using 18 to 20 rounds with 64 bits blocks in order to provide *enough* security - it will always be possible to decrypt this block cipher with chosen plaintext attacks (CPAs). For instance, a 12-round 64-bit blocks RC5 is susceptilbe to a differential attack using 2^44 chosen plaintexts.

To improve the usage of the RC5 algorithm, Halperin *et al.* propose generating the nonce N with the RC5 in counter mode [HHBR$^+$08], or using SRAM random bits to turn the nonce into a true pseudo-random value, as already discussed in section 3.2.2.

Halperin *et al.* also affirm it is possible to run the RC5 algorithm in a WISP to encrypt and verify equality, by using only the device's harvested energy. This is an important breakthrough as it proves it is possible to have an intermediate system bridging the communication between the programmer and the ICD with autonomous energy capability of running a challenge-response authentication protocol.

3.2.5 Conclusions

One form of attack on networked computing systems - which includes the case of the programmer-ICD communications - is eavesdropping on network connections to obtain authentication information such as the login IDs and passwords of legitimate users. Once this information is captured, it can be used at a later time to gain access to the system. The increase of security of the authentication protocol might come from the use of a one-time password authentication system (OTP) [N. 98] - much like the S/KEY system from which it evolves - which are designed to counter this type of attack, called a "replay attack". OTP uses a secret pass-phrase to generate a sequence of one-time (single use) passwords. With this system, the user's secret pass-phrase never needs to cross the network at any time such as during authentication or during pass-phrase changes. Thus, it is not vulnerable to replay attacks. Added security is provided due to the fact that no secret information needs to be stored on any system, including the server being protected. The OTP system protects from external passive attacks against the authentication subsystem. However, it does not prevent a network eavesdropper from gaining access to private information and does not provide protection against either "social engineering" or active attacks, but would still increase the security of the communications between a programmer and an ICD through the presence of the OTP system in an intermediate device.

Furthermore, another way to increase security could originate from the manipulation of the RC5 algorithm. As discussed in section 3.2.4, the WISP is capable of running the RC5 algorithm in the process of authentication using only harvested energy. Despite no further information is given by Halperin *et al.* [HHBR$^+$08] in terms of the values used for word size, number of rounds, and length of the secret key, it is assumed that the standard values of the RC5 algorithm were used. With such in mind, it might be worth to perform experimentations with the RC5 algorithm in a system such as the WISP using different but higher values, in order to assess the limits of the WISP device and at the same time increasing the security of the RC5 protocol to brute-force attacks.

As Halperin *et al.* discuss in their work, in the context of medical devices, security-related design choices must balance security, privacy, safety, and efficacy. Therefore, every extra authentication scheme that directly interacts with health-care scenarios must follow the security and privacy standards for IMDs generally accepted, which were already presented on section 2.4.

3.3 Discussing the *Shield* technology

The work of Gollakota *et al.* [GHR+11], described in section 2.5.2, presents a mechanism for non-invasive security for IMDs, which is describe by the authors as a wireless physical-layer solution that delegates the task of protecting IMD communication to an external deviced called the *shield*. In their evaluation, they show that the shield effectively provides confidentiality for IMDs' transmitted data, and shields IMDs from unauthorized commands, both without requiring any changes to the IMDs themselves.

In terms of communications between ICD and its programmer on a controlled and closed environment such as a clinic or a hospital, the shield technology appears as a strong solution to overcome the liabilities of both passive and active adversaries.

However, despite the reduced packet loss (less than 0.2%) when the shield decodes the IMD's packets, there is always the possibility of communicating erroneous data between the ICD and its programmer. Furthermore, the prototype presented in their work consists on a GNU Radio and USRP2 hardware, building itself into a size (figure 3.4) not compatible with communications needed to occur during daily-life routines, mainly due to its bulky size and lack of portability. In the next chapter, a portable solution for the communications between an ICD and the respective programmer is presented that may constitute a decent alternative to the shield technology.

Figure 3.4: Example of a GNU Radio USRP (Universal Software Radio Peripheral) installed in its bulky case.

Chapter 4

Improving security and privacy in ICD communications: Proposed Solutions and Proof-of-Concept

This chapter presents four proposals to change the current architecture of ICD communications with the respective programmers, and a single proposal of an alternative backoffice communication between the programmer, health-care specialists and databases.
The chapter starts with the presentation of two architectures in which we propose the replacement of current ICD programmers by a smartphone, while still using Haleprin's *et al.* WISPer device implemented with all of the *zero-power* defences [HHBR+08]. The difference between these two proposals is the use or not of a SIM/UICC card to enable communication between the ICD and respective backoffice entities over a wireless GSM network. The third proposal introduces the idea of implementing a UICC in the actual ICD device. The fourth proposal presents a more complete solution to the current architecture, replacing the WISP for a NFC-WISP device which serves as an intermediary between the ICD and smartphone/programmer communications. This proposal is complemented with an implementation of a proof-of-concept and a practical example. Finally, a proposal for a concept of a communications system between the smartphone as a programmer and the remaining entities that constitute the backoffice, with the objective of delivering the patient's ICD data to health-care specialists, is presented. This proposal also is complemented with an implementation of a proof-of-concept and a practical example.
To finalize, a discussion is made regarding the comparison of each of the first four architectures, and how the proof-of-concept implementations could help mitigate some of the security and privacy limitations the current architectures have in a real scenario.

4.1 ICD + WISPer + Smartphone

This section presents two architectures that demonstrate how the communications could be structured between the ICD and the respective backoffice, having the WISPer device bridging those communications, and using a smartphone instead of an ICD programmer to monitor and program the ICD device. Following the presentation of both architectures, we take a brief look at the security and cost analysis of these proposals and discuss some of the advantages and disadvantages of both systems.

4.1.1 Architecture

4.1.1.1 WISPer with UICC

The first proposed architecture hereby presented requires the implementation of a UICC card in the WISPer. For redundancy reasons, this architecture will be referred from here on in as architecture *A1A*.
A1A includes Halperin's *et al.* [HHBR+08] WISPer device with all of the *zero-power* defences, but instead of bridging the communications of the ICD device with its programmer/monitor, it bridges the communications of the ICD directly with a mobile network operator (MNO), which is

responsible for routing the ICD's data logs to other applications such as the physician, the patient and the data bank, as shown in figure 4.1. The ICD continues to send its data logs through the MedRadio bands but only to the WISPer. The WISPer, with a UICC installed and a suitable firmware, can then route the ICD's logs to a MNO. Thereafter, the information flow works as follows:

- The physician shall receive the log through a GSM or UMTS service in his/her mobile device (smartphone/tablet), which has both monitor and programmer functions;
- The physician can send through the MNO a programming command to the ICD;
- The physician can send through the MNO a report of his patient to the data bank;
- The physician can also access the data bank through different means - for instance, direct access or Web - where all the logs are stored;
- The physician and patient may communicate through either the MNO or the Web;
- The patient has access to all monitor functions through his/her smartphone, where he may receive physician's reports and patient-important logging information.

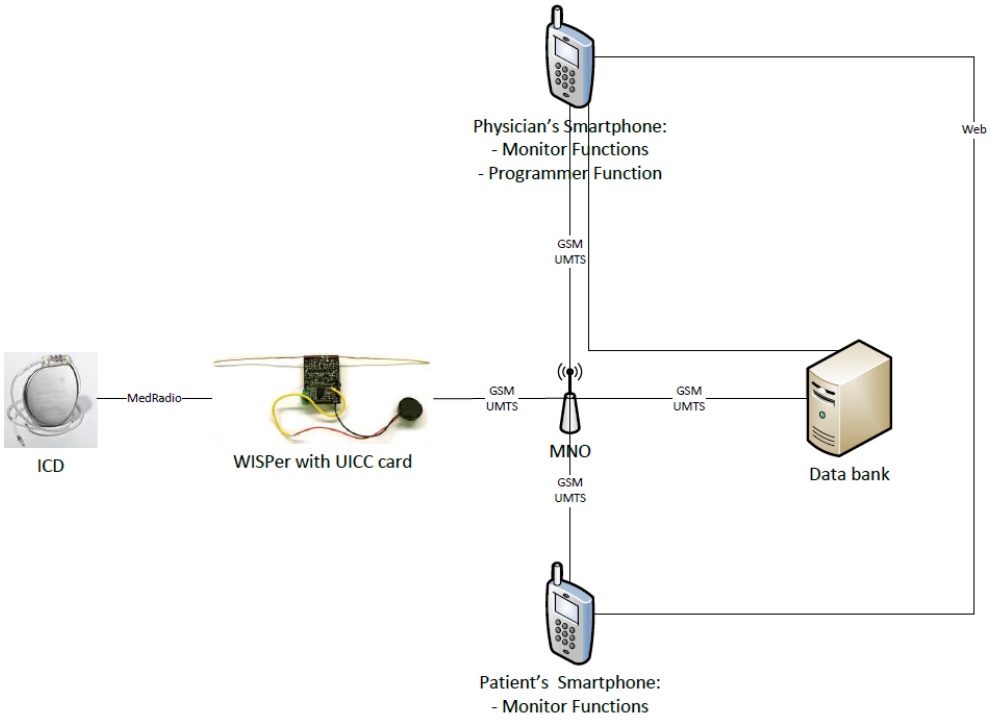

Figure 4.1: ICD+WISPer+Smartphone first architecture (A1A). Due to the presence of a UICC in the WISPer, all ICD communications post-WISPer have to go through a MNO.

4.1.1.2 WISPer without UICC

This architecture - which we will now refer to as A1B - is a use-case scenario of how the communications could be if only the programmers/monitors would be replaced by smartphones, without changes to the WISPer device, as figure 4.2 illustrates. This architecture differentiates from A1A in the bridging of communications between the ICD and its programmer/monitor. Since the programmer/monitor functions are performed by a smartphone, the WISPer still receives the data

from the ICD through MedRadio frequencies, and routes them to the patient's or the physician's smartphone through, for instance, RF or bluetooth. Then the background communications proceed as follows:

- The patient can use his smartphone's monitor capabilities, and also get in contact with his physician;
- The smartphone may send an automatic message with the ICD's log to a MNO, where it will be routed to the data bank and to the physician's smartphone;
- The physician can contact his patient through the Web or the MNO service;
- The physician can also use both his smartphone's monitor and programmer capabilities directly with the device through the WISPer.

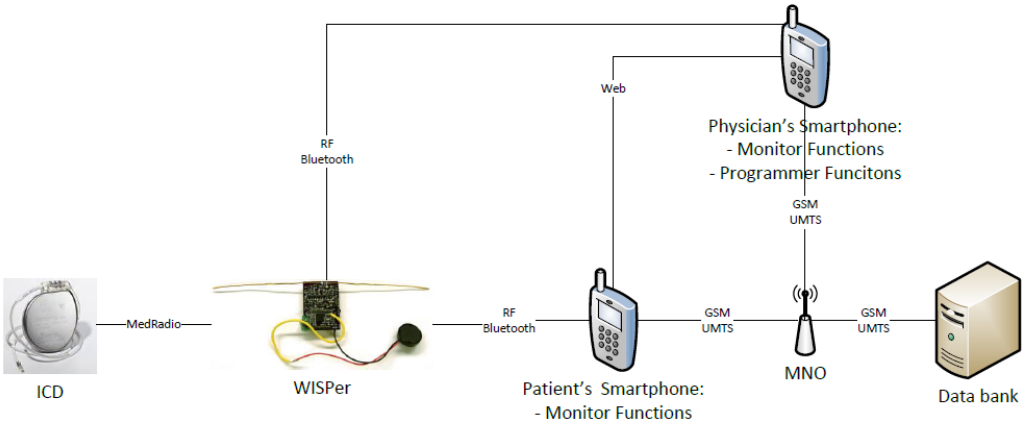

Figure 4.2: ICD+WISPer+Smartphone second architecture (A1B). Similar to current architecture.

4.1.2 Security analysis

4.1.2.1 WISPer with UICC

This architecture's communications from the WISPer device to all the entities of the backoffice is done through GSM or UMTS networks. Therefore, all of these networks' security properties are included, advantages and disadvantages. In her work *Evolution of the SIM to eSIM* [Vah13], Vahidian also presents an overview of the security principles behind GSM/UMTS security. Below, we highlight some of their security properties:

- The use of air interface (RF) MedRadio communications allows the existence of potential threats from eavesdropping;
- The GSM/UMTS security architecture allows the easy locking of *broken* cards out of the system as soon as any tempering or security breach is detected;
- Both GSM and UMTS are vulnerable to attacks on the operators' backbone network;
- The SIM card enable the authentication and encryption of the information, and it defines which information contains in each entity;
- The authentication key is a secret shared by the SIM card and the authentication center;

- GSM base stations don't support integrity protection, which allows for an attacker to mount an impersonation attack;
- UMTS requires mutual authentication between the mobile station and the network;
- UMTS networks depend on both the validity of the authentication and the integrity protection;
- UMTS networks are secure against MiTM attacks.

Knowing that GSM and UMTS networks are used by millions of people in their daily routines, a certain level of both security and privacy is guaranteed by their standards. But since there are still communications done from the ICD to the WISPer through MedRadio bands, this part of the communications represents a weak component in the proposed architecture. An attacker with the correct equipment and at a distance of 3 or less meters can easily intercept and record the correspondent transmissions, which can then be demodulated and/or decrypted at any time, anywhere. It also provides opportunity for an adversary to try active attacks on the medical equipment, even despite the presence of the WISPer device.

4.1.2.2 WISPer without UICC

In terms of security, this architecture does not represent a substantial increase in both privacy and security of communications. This is due to the fact that only the backoffice communications and type of equipment changes from today's actual architecture, as the communications between the ICD and its programmer/monitor are still done through RF, and thus inheriting all of its advantages and disadvantages.
However, the presence of the GSM or UMTS network guarantees a certain level of security in the backoffice communications. Both technologies GSM/UMTS are widely used by the world population and this ensures that people are constantly working to enhance these networks standards and protocols and monitor vulnerabilities in order to keep the communications with an acceptable level of privacy and security.

4.1.3 Cost and Performance analysis

4.1.3.1 WISPer with UICC

In terms of communications, this architecture would require the existence of an already existent MNO service and SIM cards for all the end-entities of the architecture: WISPer, patient, physician and the data bank server. This may pose as a disadvantage, since it would necessarily require for all entities to pay for the communications' service provided by the chosen MNO. A similar solution would be the creation of a dedicated MNO, but the cost analysis of such situation is tougher to predict due to all the variables it carries.
Testing the consequences of implementing a UICC into a WISP device would also be necessary, whether being in terms of processing capability and power usage, as the performance penalty from using the UICC might turn the WISP into a non-autonomous device.
The WISPer still has to be close (up to 3 meters) to the ICD, which may cost the system its portability and ease of use. To solve this issue, the WISPer could be embedded into a piece of smart clothing - the merging of electronics and textiles which is currently an active research and development area.
Despite all the costs inherent from the presence of a MNO, this architecture would allow reducing the costs from acquiring the programmer and monitor devices, which are extremely expensive.

4.1.3.2 WISPer without UICC

When referring to costs, architecture A1B differentiates from A1A by not being necessary a great deal of modification to the current WISPer design. It can still be embedded in a smart cloth, as in A1A, but only needs to change where to route the RF or bluetooth signal it emits to the respective receptor. This means the WISPer would certainly continue to be autonomous in terms of both processing and power consumption.
It would however be necessary both the patient's and physician's smartphones to have a software dedicated to authenticate, receive data from the ICD and - only for the physician - send configuration settings to the ICD through the WISP, much like current ICD programmers and monitors have.
This architecture would still be open to attacks in case a physician lost his smartphone, at least until the respective SIM card was blocked by the MNO.

4.2 ICD with SIM

This section continues the suggestion of alternative architectures for the current communications between an ICD and the respective programmers and monitors. Continuing the idea of replacing ICD monitors and programmers for smartphones, this section suggests changing the current hardware of an ICD to include a SIM card. For nomenclature reasons, this architecture will now be referred as *A2*.

4.2.1 Architecture

A2, which is demonstrated in figure 4.3, works as follows:

- The ICD is designed to be capable of sending its data logs through the GSM service;
- The ICD is programmed to send and receive the data only from authorized entities, by having those entities phone numbers - for example - in the list of contacts;
- The ICD then sends its data logs through the GSM network to both the patient's and respective physician's smartphones, and the MNO also redirects those data logs to a data bank;
- The patient's smartphone includes an ICD monitoring software, while also allowing to directly communicate with the physician;
- Similar to the patient's smartphone, the physician's also includes an ICD programmer software;
- The ICD can then receive commands to change its programming. It can also send extra data logs - for example, in an emergency situation, any health-care specialist that has the authorization to contact the ICD can access its data;
- Both the physican and the patient can also communicate through a web service, while the physician may have other means of accessing the data bank - personally, for instance.

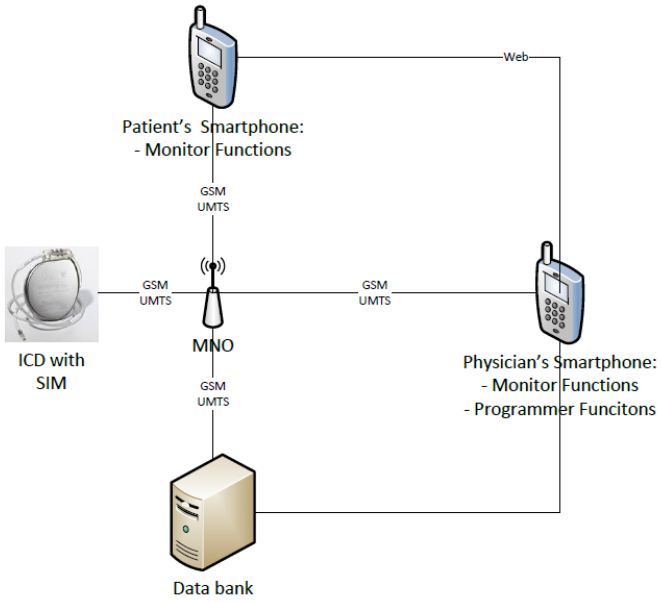

Figure 4.3: ICD with SIM (A2). Similar to current architecture with the replacement of both the programmers and monitors by smartphones with the same functions. The ICD directly communicates to all entities through a MNO.

4.2.2 Security analysis

A2's communications from the ICD device to all entities of the backoffice is done through GSM or UMTS networks. Therefore, all of these networks' security properties are included, the advantages and disadvantages as was already mentioned in section 4.1.2.1.
There is still the danger of having an outsider/adversary getting access to the ICD's "phone number" and so he could potentially receive data and send programming commands to the device, whether it being done through eavesdropping, man in the middle, SIM card cloning, or by physically obtaining (stealing) any of the SIM cards authorized to communicate with the ICD.
Summarizing, the security of such system would have the same security features and flaws the SMS service has, the good and the bad.

4.2.3 Cost analysis

In terms of communications, we believe, by using the SMS service, the ICD can send small compact log files through a MNO to the physician, the patient and the data bank.
These three entities must have dedicated software to transform the information contained in the logs. This way, we believe that having an ICD sending SMS in a routine programmed by the patient's physician, or by another health-care specialist in an emergency situation, may be affordable in terms of power consumption.
However, the constant presence of such radio waves under the skin and so close to the organs has to be studied in terms of hazardousness to the patient's health, and in terms of efficiency of both receiving and sending data.
Financially speaking, once again the substitution of the both the programmer and monitor for only a smartphone would allow an increase of saving in terms of costs. This substitution would also have a significant increase in both portability and ergonomics. The system would need no more bulky systems to exchange the information between its identities, and both the physician and the patient would only need their personal (or work) smartphones to have the respective data

treatment software, which would allow them to access data and get in touch with each other from anywhere, as long as the MNO service is up.

4.3 ICD + NFC-WISP + Smartphone

After considering the previous alternatives, this section presents what we believe could be the best solution for ICD communications involving the use of an intermediate device and a smartphone with both monitor and programmer functions. The intermediate device is a NFC-WISP based system, which will bridge the communications between the ICD and the respective programmer/monitor smartphones through NFC communication standards.

4.3.1 Architecture

This architecture - from here on referred to as *A3* - is represented in figure 4.4. A3 is a very simple architecture where:

- The ICD communicates with a NFC-WISP based device through either MedRadio or NFC;
- The NFC-WISP based device can then communicate with either the physician's or the patient's smartphone;
- The patient has access to monitor functions;
- The physician also has monitor functions, and can send programming commands to the ICD, which must first go through the NFC-WISP.

The reason that the ICD communications with the NFC-WISP device are referred in A3 as done through either MedRadio or NFC is because, for the creation of a proof-of-concept example, we had no access to neither an ICD nor a NFC-WISP device. Instead, we adapted an existing Android application that allows to write and read files to and from NFC tags, and another android application to allow it to read the log file previously written in a NFC tag smartcard. This is explained further ahead on section 4.3.3.
Thus, MedRadio communications would be used if no modifications were done to the ICD device in order to communicate to what could be a NFC-WISPer. If the ICD was to be changed to include a NFC reader/writer module, it could easily communicate with the NFC-WISP through NFC instead of MedRadio. The NFC-WISP could then store the log file until a more recent one was available, with both the physician and the patient only having to scan their smartphones to the NFC-WISP device in order to receive the log file.

4.3.2 Proof-of-concept Implementation

For the creation of the proof-of-concept example, we used the following items:

- The Eclipse Integrated Development Environment (IDE) *Eclipse Standard/SDK*, version *Kepler Service Release 2* [The14], with the Android SDK installed [And14];
- The NDEF Tools for Android boilerplate demo [Tho13a];
- A LG Nexus 5 smarthpone with Android version 4.4.4;
- A NFC Forum Tag type 4, present in figure 4.5, with a variable memory availability up to 32 KBytes per service [NFC14c].

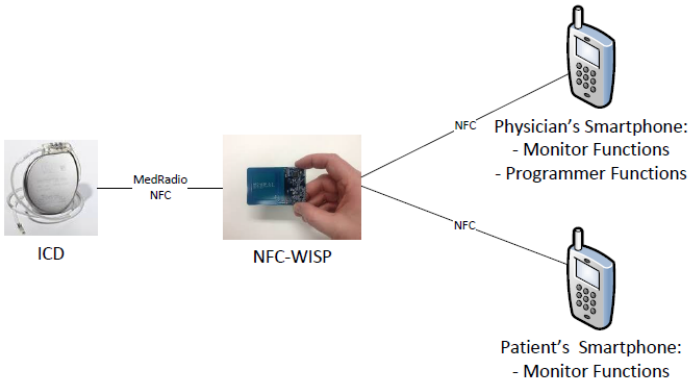

Figure 4.4: ICD+NFC-WISP+Smartphone (A3) architecture. Reducing the range of communications through NFC represents a possible increase in both security and privacy of ICD communications.

Figure 4.5: A NFC Forum Type 4 Tag and a LG Nexus 5 smartphone.

The first step towards our proof-of-concept example was to create an ICD log file. For representations reasons, we created a simple text file with some patient data that usually exists in ICD communications, which is present in the Appendix chapter, section A.1. This log file has a size of 1.02 KBytes and thus perfectly fits the NFC tag, and it was also placed in the downloads directory of the smartphone.

Then an Android application was written (figure 4.6) using the Android NFC library and the NDEF Tools for Android library [Tho13b] to read and write text to a NFC tag. The application then loads a file from the smartphone through the click of a button to the *DefaultNfcTagWriterActivity* class by adding the file directory to the button function in order for it to copy all the text content of the log file to the *EditText* box. For that we created the button *readLogFile* to which we assigned the previously mentioned activities on click, and after that the application waits for the NFC tag approach in order to write the message.

The files changed from the original project are mentioned below:

- *DefaultNfcTagWriterActivity.java*, present in section A.2;
- *writer.xml*, present in section A.3.

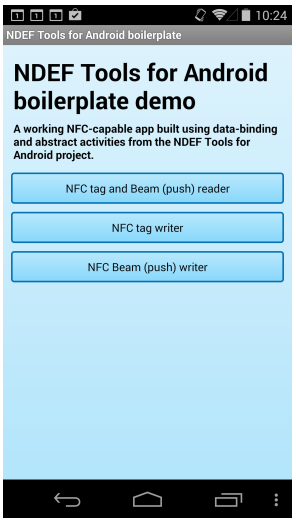

Figure 4.6: NDEF Tools for Android boilerplate demo's main menu.

4.3.3 Proof-of-concept Practical Example

In this subsection, we present some screenshots of the implemented application working, and then how this application could work in a real-life environment.
The implemented application works as follows:

1. From the application's main menu (figure 4.6), click the *NFC tag write* option.
2. There, the application changes to another window and enables NFC (figure 4.7). The application now asks for the text to be recorded in an *EditText* box, while also allowing to click the *Load log file* button to automatically read the file into the *EditText* box.
3. After clicking the *Load log file* button, the application automatically reads and copies the text file data to the *EditText* box (figure 4.8).
4. We then scan our NFC Forum Type 4 Tag and the application successfully writes the text in the tag(figure 4.9).
5. To show the tag was successfully written and contains the text of the log file, we go back and change to the NFC tag and Beam (push) menu (figure 4.10).
6. Then, we once again scan our NFC Forum type 4 Tag with the smartphone and the full text of the log file emerges in a text box (figure 4.11).

Now we present an example of how our proof-of-concept prototype could work in real-life environment. A practical example for architecture A3 would have the following procedure:

1. The ICD device - which could have integrated NFC capabilities - would have the log file stored in its memory. This is represented by our smartphone in our demonstration.
2. The NFC-WISP device embedded in a smart clothing shirt - represented by the NFC Forum Type 4 Tag - would then receive the file from the ICD through NFC - or MedRadio, if a standard ICD was used -, and thus allowing the ICD to free some memory and save extra power.
3. The NFC-WISP device would then alert the patient, either by sound or vibration, that the log file was ready to be sent to the smartphone through NFC.

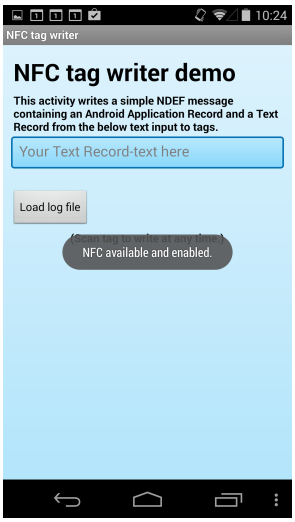

Figure 4.7: NDEF Tools for Android boilerplate demo's NFC tag writer menu.

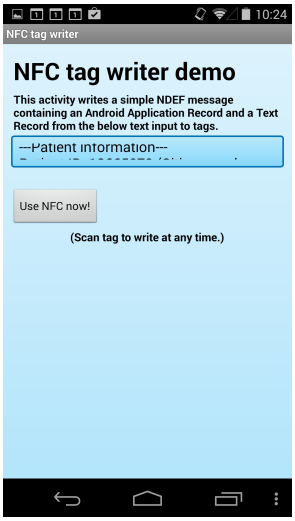

Figure 4.8: NDEF Tools for Android boilerplate demo's NFC tag writer menu after clicking the *Load log file* button.

4. Then a user, either the patient or his physician, would then place his smartphone in contact with the NFC-WISP device, with the smartphone slightly vibrating when the transaction of the log file was completed.

4.3.4 Security analysis

This system enhances security and privacy due to the fact its communications' security is hardware based through the use of NFC. As mentioned in section 2.6.2, the physical proximity NFC requires guarantees that the operator has confidence in which data is read at what time, thus greatly reducing the chance of human error. This means that for an adversary to even attempt eavesdropping the wireless communications between the NFC-WISP device and the ICD's programmer, he would practically have to physically touch the patient, which shouldn't be a problem for the patient and the respective physician when the communications need to take place.
This architecture provides an acceptable level of security for telemonitoring, since the communications only need to go from the ICD to the NFC-WISP (or an NFC tag if the ICD would be modified to include NFC capabilities) and then either the patient's or the physician's smartphone

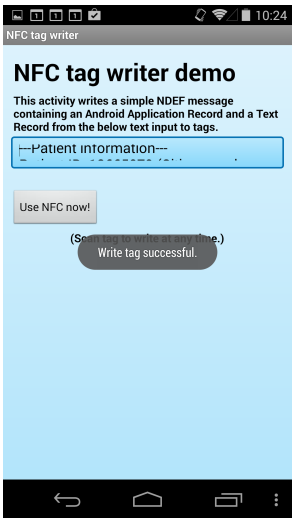

Figure 4.9: NDEF Tools for Android boilerplate demo's NFC tag writer menu after scanning the NFC tag.

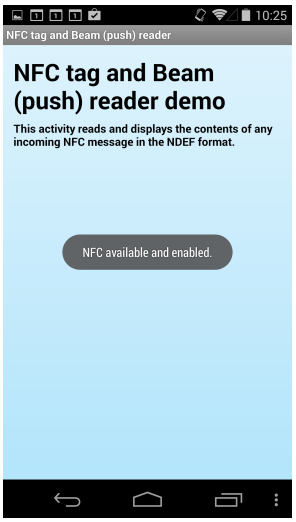

Figure 4.10: NDEF Tools for Android boilerplate demo's NFC tag reader menu.

for receiving the ICD logs. However, assuming the physician wants to use the same communication channel to program the ICD, a security threat emerges when authenticating whose smartphone may send data to the NFC-WISP or the modified ICD itself through NFC. The solution to such situation could be the implementation of the Halperin's *et al.* zero-power authentication protocol, with the shared key being the smartphone SIM's IMSI. In the case the ICD would be modified to include NFC reading and writing capabilites, another solution would be the substitution of a smart clothing shirt with an implemented NFC-WISP for a more casual NFC tag in form of a card, this way helping to mitigate attempts from adversaries trying to program the ICD, as they would need both the card and a smartphone with an acceptable programming file to modify the device.

The current ICD design still poses as a security threat to this architecture. If the communications between the ICD and the NFC-WISP device still occur through MedRadio bands, the range gives potential adversaries the possibility for the change to intercept those RF communications. Given the possibility of creating new ICD devices with a NFC module included for close-ranged communications to with the NFC-WISP device, this system's wireless communications would become extremely secure while still being highly easy to use by both the patient and the respective health

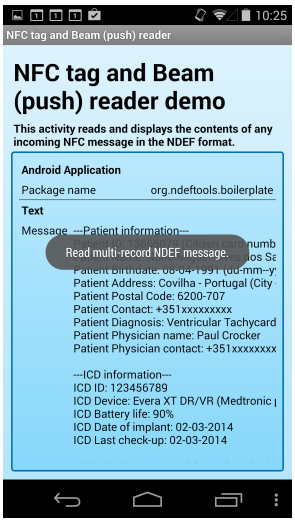

Figure 4.11: NDEF Tools for Android boilerplate demo's NFC tag reader menu presenting the ICD log contained in the NFC tag, after scanning the NFC tag.

care specialists.

4.3.5 Cost analysis

First, this system would have to be extremely well considered and evaluated by a wide range of different entities, including medical and security communities, industry, regulatory bodies, patient advocacy groups, and other relevant communities that would have to collaboratively make decisions on both mechanisms and policies. This would have both financial and time costs.
Nonetheless, there would be the direct economical and portable advantages of using smartphones instead of dedicated monitors and/or programmers to communicate with the ICDs. Smartphones are now part of a great share of the world population's daily life, and their cost is nowhere near as high as current ICD monitor and programmers. Moreover, a system like the NFC-WISP is very easy and accessible to implement and is proved to be auto-sustainable in terms of power consumption.
The time consumed in both writing and reading the ICD log example file to and from the NFC tag was calculated through the Java implementation. Results showed it takes 0.001 seconds to write the ICD log file and 0.002 seconds to read the same file that was previously written in the NFC tag, proving this kind of communications is very acceptable in terms of speed.
However, the possible implementation of a NFC writer/reader module in the ICD device itself would still need to be extremely well-evaluated in terms of power consumption, as the procedure to replace an ICD battery nowadays involves a surgery, and in terms of the risks of hazard the NFC communications might carry to the patient.

4.4 MQTT for backoffice communications

The architecture presented in this section aims to complement the one demonstrated in the previous section (section 4.3). In particular we concentrate on the use-case of how the patient's smartphone should communicate with the remaining entities constituting the backoffice after receiving the ICD log from the NFC-WISP device. We propose using the MQTT technology for the backoffice communications and then aim to demonstrate this idea with a practical example and the respective

implementation.

4.4.1 Architecture

The architecture hereby presented - architecture *A4* - complements architecture A3, and works as follows (figure 4.12):

- The ICD communicates with a NFC-WISP based device through either MedRadio or NFC (A3);
- The NFC-WISP based device can then communicate with either the physician's or the patient's smartphone (A3);
- If the NFC-WISP device communicates with the patient's smartphone:

  The patient can send the ICD log data using the publish service of MQTT, which will go to a specific topic on the MQ server;

  From the MQ server, any health care specialist having a valid subscription to that specific topic - like the patient's physician - will receive the log file;

  Through a dedicated background application, the data bank can also receive the log file and save it in the system's database.
- The patient has access to monitor functions;
- The physician may have a smartphone or a desktop application that uses an MQTT application to receive the data from the MQ server, in which he may have access to both programming and monitoring functions;
- Given the correct tools, the system could also work in the opposite direction with the physician attempting to program the ICD device:

  The physician would also send a text file through MQTT to the respective patient's smartphone, working as a programming script;

  The patient would then be alerted to use his smartphone's NFC capabilities with the NFC-WISP, with the script then reaching the ICD and thus executing the programming changes.

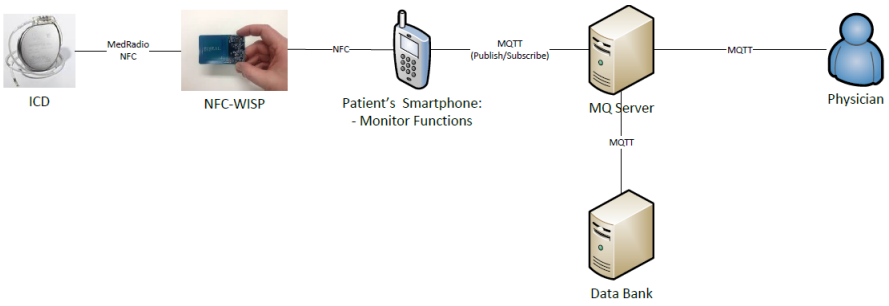

Figure 4.12: MQTT for backoffice communications (A4) as a complementing tool for A3's communications with the backoffice entities.

4.4.2 Proof-of-concept Implementation

This proof-of-concept example implementation assumes the ICD log file will be read from either a NFC tag or the smartphone's downloads folder. For the creation of the proof-of-concept example, we used the following items:

- The Eclipse Integrated Development Environment (IDE) *Eclipse Standard/SDK*, version *Kepler Service Release 2* [The14], with the Android SDK installed [And14];
- IBM WebSphere MQ Telemetry Client Pack version 7.5.0.2 [IBM14];
- A LG Nexus 5 smarthpone with Android version 4.4.4;
- A NFC Forum Tag type 4 with a variable memory availability up to 32 KBytes per service [NFC14c], with the ICD log file stored.

The first step in our implementation process was to check the WebSphere MQTT service was ready to function with an example application provided by IBM. For that, IBM WebSphere MQ Telemetry Client Pack version 7.5.0.2 was installed and configured with all the telemetry texting options enabled in the custom installation menu. With the installation complete, IBM WebSphere MQ Explorer (figure 4.13) is ready to work.

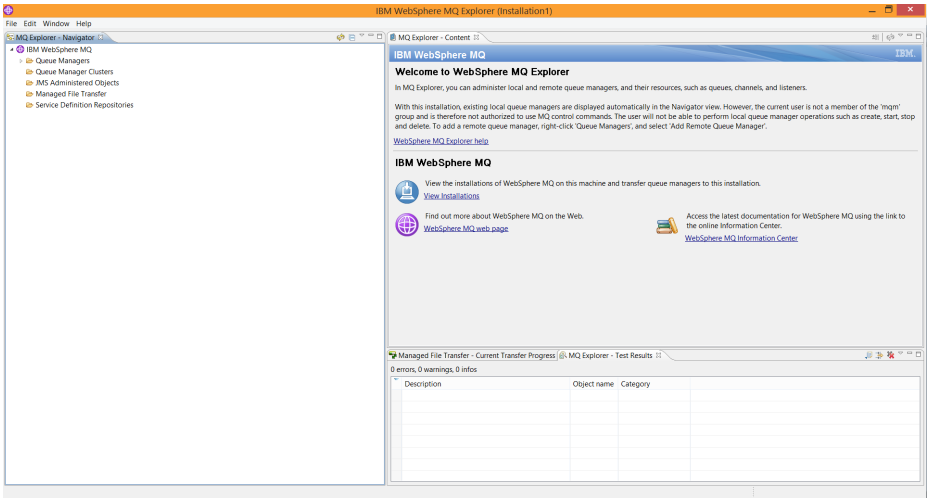

Figure 4.13: IBM WebSphere MQ Explorer

For the creation of a simple proof-of-concept example, we created a new Queue Manager - named *ICD_communications* - and then proceeded as follows:

1. Create a Telemetry Channel: we gave it the name *ICD* and attributed it the port number 1883 (figure 4.14):

   The channel can be easily configured as required for SSL. This requires a server certificate and configuration of SSL/TLS security parameters. In the example shown here SSL was not used;

   For the client authentication, we also decided to not check client supplied username and password (figure 4.15);

   Once again for simplicity reasons, we defined a fixed user ID, allowing any client to connect to this channel (figure 4.16).

2. Finally, we created the topic *ICD_patient_001* (figure 4.17).

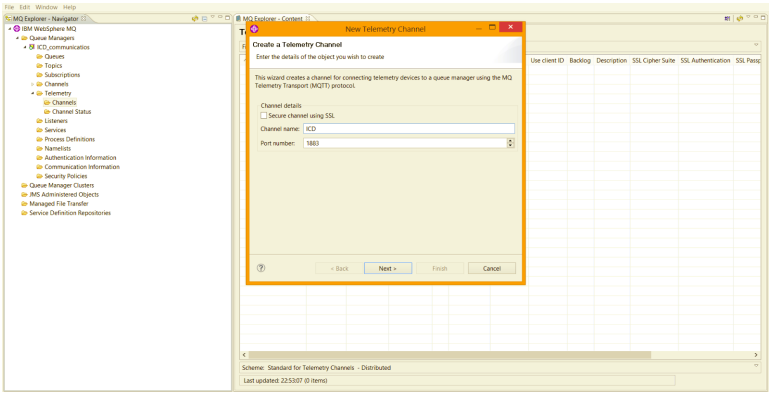

Figure 4.14: IBM WebSphere MQ Explorer: creating a Telemetry Channel - part I.

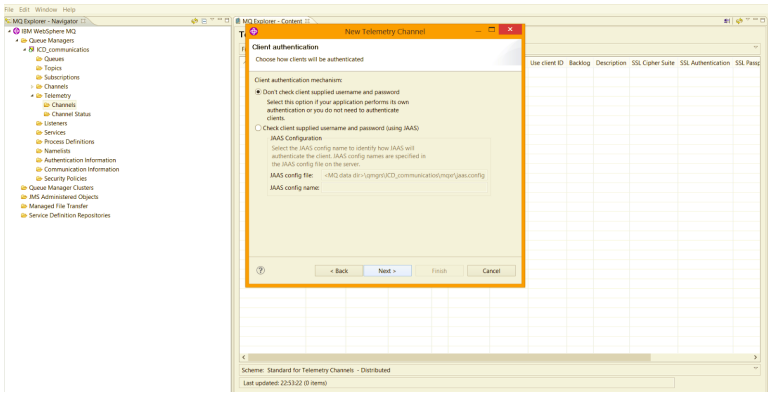

Figure 4.15: IBM WebSphere MQ Explorer: creating a Telemetry Channel - part II.

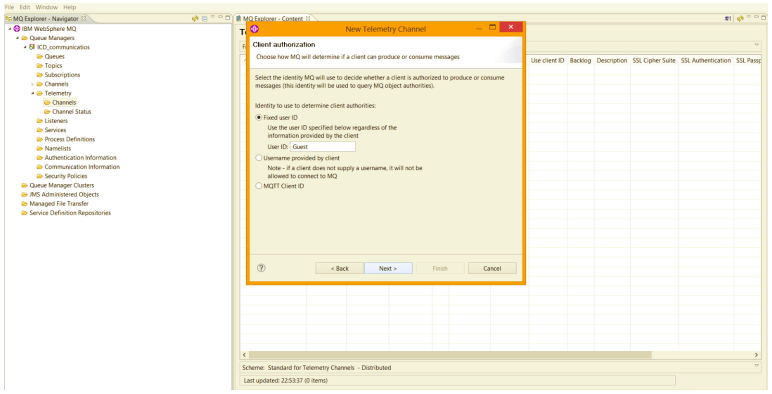

Figure 4.16: IBM WebSphere MQ Explorer: creating a Telemetry Channel - part III.

An Android application client application was developed for connecting and communicating with the MQ server. The application reads an existing file from the downloads folder in the smartphone's permanent memory through the click of a button (named *Load LOG from File*), and automatically writes the ICD log file's content into the *EditText* box; the user can then send a message in the *Publish* section of the application. To do this, a button was added to the *activity_publish.xml* file (see section B.2), and programmed it to read the file from a folder within the smartphone's downloads folder by changing the *PublishFragment.java* file, as it is in section B.1.

Once we verified the message - the log file - was reaching its destination - the MQ server -, we moved on to include NFC reader capabilities to the application in order to read the log file from a NFC tag and copying its text to the *EditText* box, similarly to what was previously done. For

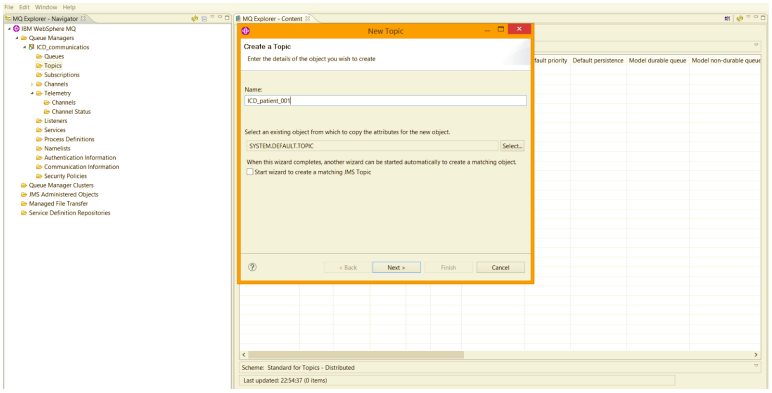

Figure 4.17: IBM WebSphere MQ Explorer: creating a new topic.

this we created the *Load LOG from NFC button* that sends the user to a newly created section of the application (see section B.1). In this section, called *NFC reading platform*, the device readies the NFC module to read a NFC tag (see section B.3). After reading the NFC tag, it automatically returns to the application's Publish section with the ICD log text already appearing in the message *EditText* box, ready to be published into the topic.
We also included a button to clear the message in the Publish section, and another button to go back to the previous section in the NFC reading section.
To summarize, we did the following changes to the sample application:

- The Publish section of the application now has three extra buttons: *Load LOG from File*, *Load LOG from NFC* and *Clear Message*;
- A new section named NFC reading platform was created, with NFC reading capability and the button *Go Back*.

4.4.3 Proof-of-concept Practical Example

Similar to section 4.3.3, this subsection presents some screenshots of the implemented application working, and then how this application could work in a real-life environment.
The implemented application works as follows:

1. First we make sure our server is up and running the queue manager with a telemetry channel, as mentioned in the previous section.
2. Then we open the *MQTT* application from our LG Nexus 5 smartphone.
3. By clicking the button *NEW CONNECTION*, the *New Connection* section opens where we define the Client ID, the Internet Protocol (IP) address the server has in the wireless network, and the port number of the telemetry channel we intend to connect to (figure 4.18).
4. With the client created and the connection to the server established, we now show how to publish to the topic *ICD_patient_001* previously created (figure 4.19):

   By clicking the *Load LOG from File*. The text from the log file located in the downloads folder is immediately sent to the message *EditText* box. The log file text can now be sent to the MQ server by clicking *PUBLISH* (figure 4.20);

   On the other hand, by clicking the *Load LOG from NFC* button, the application changes to a different section as you can see in figure 4.21. There, the application will wait for the

scanning of a NFC tag, from where it will then return to the Publish section with the ICD log text written in the message *EditText* box (figure 4.20).

5. By clicking the *PUBLISH* button, the message is sent to the server (figure 4.22).

6. We can check if the message reached the server by two different ways: through the *MQTT* application or through the IBM WebSphere MQ Explorer:

   Through the *MQTT* application, we change to the Subscribe section where we insert the topic to which we published the message - in this case *ICD_patient_001* - and click subscribe (figure 4.23). After the application alerts the user that it is subscribed to the topic, a message is soon after received, which can be found in the *History* section of the application (figure 4.24).

   Through the IBM WebSphere MQ Explorer, we open the *MQTT Client Utility* for the *ICD* telemetry channel. We connect to the port 1883, and then subscribe to the topic *ICD_patient_001* (figure 4.25). A message is received, and by opening it, the original text from the ICD log file is there presented (figure 4.26).

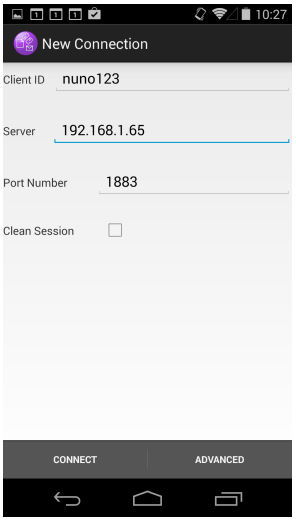

Figure 4.18: MQTT application for proof-of-concept: establishing a new connection with a telemetry channel.

Now we present an example of how our proof-of-concept prototype could work in real-life environment. A practical example for architecture A4 would have the following procedure:

1. After receiving the log file from the ICD, the NFC-WISP could send a sound or vibration feedback to the patient as a request to scan the smartphone (A3).

2. The patient then opens the MQTT aplication and scans the smartphone to the NFC-WISP device, in order to receive the file.

3. Once the log file has been read by the smartphone, it is automatically sent to the MQ server through MQTT.

4. On the other side of the backoffice, the physician can access the message by having a subscription in the patient's topic. Then, it can also send a message to the patient and might even send a programming script to the ICD device through the patient's smartphone.

5. In the backoffice communications, a data bank with MQTT capabilities also receives the messages and stores them in a data base.

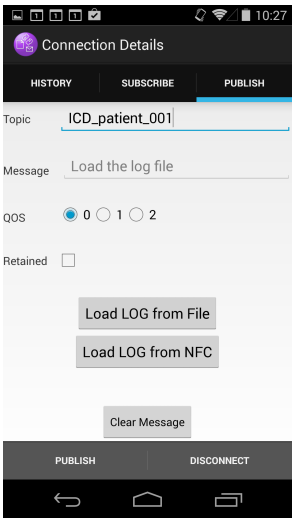

Figure 4.19: MQTT application for proof-of-concept: Publish section.

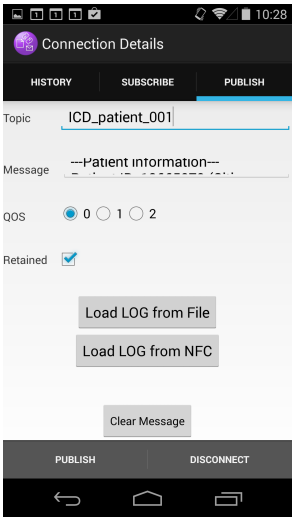

Figure 4.20: MQTT application for proof-of-concept: Publish section after clicking the *Load LOG from File* button, or after scanning the NFC tag with the log file in the NFC reading platform section.

4.4.4 Security analysis

As mentioned in section 2.5.5.7, three concepts are fundamental to MQTT security: identity, authentication, and authorization. Despite our proof-of-concept example left aside the use of all the optional security mechanisms, MQTT can be very secured when performing client authentication using Java Authentication and Authorization Service (JAAS), client authorization through MQTT Client ID, and establishing a telemetry channel that uses SSL.

All the security standards of using both JAAS and SSL are immediately related with MQTT, guaranteeing an acceptable minimum level of security for MQTT communications.

The use of MQTT for the backoffice communications allied with NFC communications in the ICD/NFC-WISP/smartphone system greatly enhances security and privacy over current architectures of ICD communications.

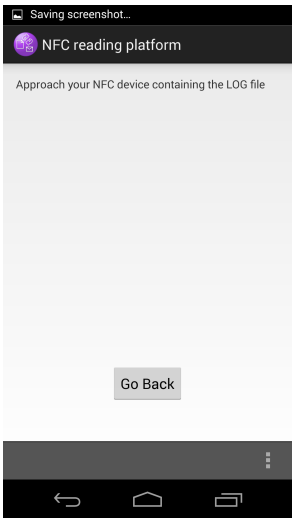

Figure 4.21: MQTT application for proof-of-concept: NFC reading platform section after clicking the *Load LOG from NFC* button in the Publish section.

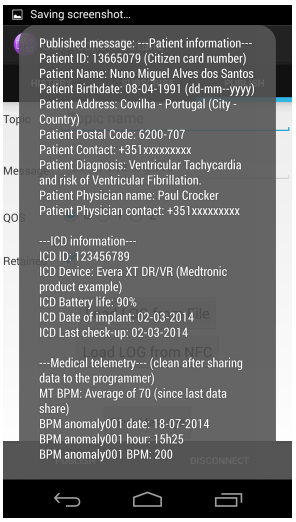

Figure 4.22: MQTT application for proof-of-concept: message published to the MQ server.

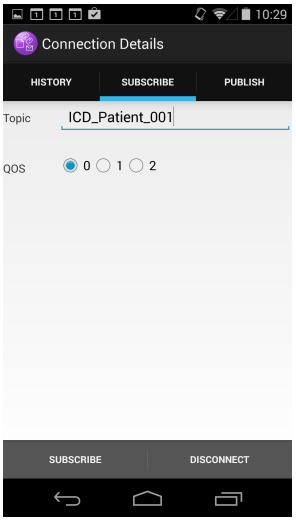

Figure 4.23: MQTT application for proof-of-concept: subscribing to topic *ICD_patient_001*.

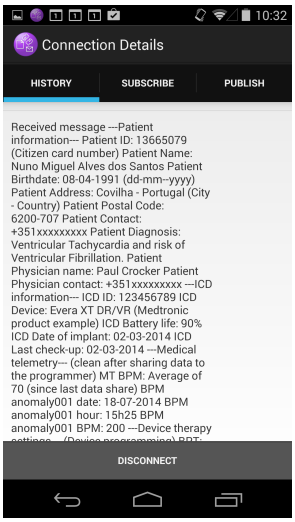

Figure 4.24: MQTT application for proof-of-concept: message that was previously published was received. after subscribing to the topic.

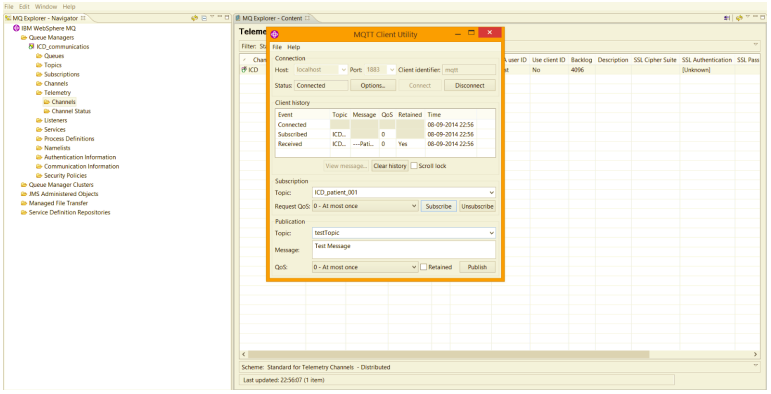

Figure 4.25: IBM WebSphere MQ Explorer: MQTT Client Utility - connect to the *ICD* telemetry channel, and subscribe to the *ICD_patient_001* topic.

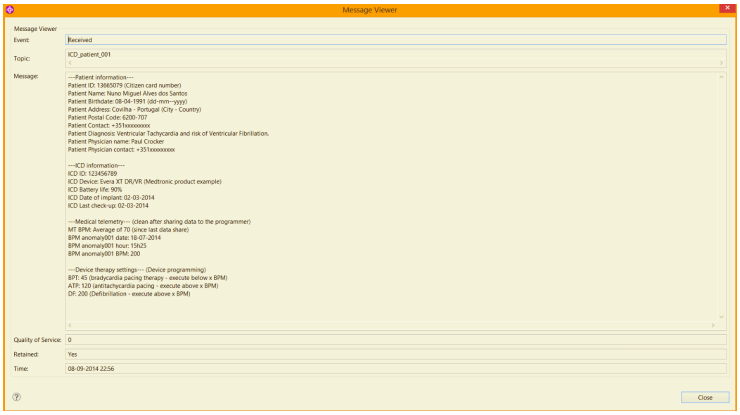

Figure 4.26: Message received in the MQTT Client Utility after subscribing to the *ICD_patient_001* topic.

4.4.5 Cost and Performance analysis

Portability would be the main contribution of this architecture. Allied with A3, A4 would allow both patients and the respective physicians to be in constant contact. In terms of reliable and robust message delivery the MQTT protocol enables us to specify a QoS level that ensures that

delivery of messages in both directions is guaranteed.
However, it would be required for both users to know the IP address of the server, the respective telemetry channel port, and the specific communication topic to get both entities intercommunicating. This could be solved with a prior configuration of both the patient's and physician's applications. Still, the server would have to be constantly running, and both the physician and the patient would require constant access to the server's network. Anyway, from what we experienced, latency of communications is very reduced, posing a very good advantage over alternative systems.
In terms of economic and financial impact, we believe architectures A3 and A4 would merge together very well, solving many problems related to both current and previously mentioned architectures. A3-A4 would not require bulky programmer/monitor equipments and all of its costs by using powerful mobile devices and desktop applications for doing the same functions. This architecture would also not require a MNO and all of its related costs for the communications. Nonetheless, it would also require a study from entities from a wide range of scientific and regulatory areas, as well as developing such a system from scratch.
In terms of usability, this system would be extremely ergonomic. The patient would only need a smartphone with the application installed and previously programmed to communicate to a specific server, telemetry channel, and topic. It would only require for the patient to get his smartphone in touch with the NFC-WISP device when asked to, and send a specific message to his physician if he is willing to. Thus, the transmission of ICD logs would be automatic. The physician, on the other hand, would decide which patients' topic he would subscribe to, automatically receiving the respective logs and messages. The physician could also have an extra functionality to send ICD programming scripts, with the patient's smartphone receiving a message that it needed to connect with the NFC-WISP device.

4.5 Discussion

The security of the communications shown in the implemented proof-of-concept examples can be easily improved by choosing the correct options to use JAAS and SSL for MQTT communications. Nonetheless, both A3 and A4 examples suffice to demonstrate both NFC and MQTT communications can be used together to guarantee a very acceptable ICD communications system in terms of security and privacy.
By analysing the cost and security analysis of all the proposed systems and not forgetting how current systems work according to Halperin *et al.* [HHBR+08], we believe the implementation of both architectures A3 and A4 would greatly improve ICD communications' systems in terms of security, privacy, safety and utility, matching a majority of the standards mentioned in section 2.4. Also a well-structured health-care hierarchy could be created which defined who should have access to the MQ server's patient specific telemetry channels. This hierarchy allied with the hardware security provided by the close-range NFC communications would turn this system into a perfect fit for today's requirements of constant and portable communications between patients and health-care specialists, and thus surpassing both the current and the other proposed architectures. After all, recent studies show that patients with ICDs who used wireless remote patient monitoring had significantly lower risks of death and rehospitalization compared with those who did not, and therefore easy to use, familiar, and lowcost technologies that enable remote monitoring should be encouraged [Ame14] [Hin14].

Chapter 5

Conclusion

Halperin *et al.* [HHBR+08] showed that an implantable cardioverter defibrillator is potentially susceptible to malicious attacks that violate the privacy of patient information and medical telemetry, and may experience malicious alteration to the integrity of information and state of the communications, including patient data and therapy settings. The standard approaches for security and access control may not always be suitable for IMDs due to tensions between security features, power constraints and clinical safety, in fact some present-day design strategies for IMDs and associated certification processes have been shown to possess flaws. Analysing and evaluating the security and privacy of an IMD requires skills from many disciplines, including security, cryptography, cardiology, signal processing, radiocommunications, amongst others.
This dissertation includes a collection and study of the most relevant information related to information security and privacy of ICDs. We believe a similar study has yet to be published and thus in itself is an important component of this work.
Also in this dissertation it was demonstrated how easy it is to mount an opportunity for an adversary to eavesdrop on the radiocommunications, the means of communications between an ICD and the respective programmer.
Furthermore, a critical analysis of both the *Zero-Power Defenses* of Halperin *et al.* [HHBR+08] and the non-invasive *Shield* proposal of Gollakota *et al.* [GHR+11] is presented. These are the two most relevant current studies that aim to propose enhancements in security and privacy in ICD communications, from which we can retrieve very relevant information and conclusions.
Ultimately, we presented four different architectures that involve the substitution of current bulky-designed programmers and monitors by more modern and portable technologies, smartphone/tablet and the like. After assessing both the cost and security of the four architectures, we concluded the combination of the architecture using NFC for ICD-smartphone communications with the architecture using MQTT for smartphone-backoffice communications had the best increase in privacy ad security, while possibly reducing the financial costs to the end-users. Moreover, we concluded the ergonomics and portability of the overall system would greatly increase, and thus guarantee the best balanced proposal for an alternate architecture in terms of the tension between security/privacy and safety/utility goals.

5.1 Future work

We hope this work provides a foundation to explore the challenge of developing methods that appropriately balance security and privacy with traditional goals such as safety and utility.
As future work, it might be relevant to elaborate business models for each of the proposed architectures that involve replacing current ICD's programmers and monitors for smartphones. If possible, we believe the proposed architectures in this dissertation deserve a chance for at least a real-life prototype implementation in order to assess how much security and privacy would be enhanced, and how the relationship between security/privacy and safety/utility goals would then stand.
We also defend studies to assess ICD communications' flaws must continue to exist in a more frequent timeline. With the increasing number of ICD users and the ageing of the world population,

and given that peoples' lives are directly related to ICD communications, such studies have to be continuously made in order to find possible security flaws, and thus proposing new and better solutions to enhance security and privacy for the patient, and to continue improving the safety and utility properties of such devices.

An ultimate solution for mitigating the tensions between security/privacy and safety/utility goals will require experts from medical and security communities, industry, regulatory bodies, patient advocacy groups, and all other relevant communities to collaboratively decide on both mechanisms and policies for IMDs. This work presents two architectures for ICD and backoffice communications that, when allied together, could be a very good proposal of mitigating these tensions, while still greatly increasing ergonomics mainly due to the presence of increasing portability from the utilization of mobile devices such as smartphones/tablets, something that in a 10 year period practically everyone will possess.

Bibliography

[Ala11] Alanson Sample - University of Washington. Nfc-wisp [online]. 2011. Available from: http://www.alansonsample.com/research/NFC-WISP.html [cited August 2014]. xv, 28

[Ala13] Alanson Sample - University of Washington. Nfc-wisp [online]. 2013. Available from: http://www.alansonsample.com/research/NFC-WISP-Eink.html [cited August 2014]. 28

[Ama13] Amateur-radio-wiki. Rtl2832 [online]. 2013. Available from: http://www.amateur-radio-wiki.net/index.php?title=RTL2832 [cited August 2014]. 32

[Ame12] American Heart Association. Defibrillation [online]. 2012. Available from: http://www.heart.org/HEARTORG/Conditions/Arrhythmia/PreventionTreatmentofArrhythmia/Defibrillation_UCM_305002_Article.jsp [cited May 2014].

[Ame13] American Heart Association. What is cardiovascular disease (heart disease)? [online]. 2013. Available from: http://www.heart.org/HEARTORG/Caregiver/Resources/WhatisCardiovascularDisease/What-is-Cardiovascular-Disease_UCM_301852_Article.jsp [cited May 2014]. 3

[Ame14] American College of Cardiology Foundation. Study demonstrates benefits of remote monitoring in improving icd patient outcomes [online]. 2014. Available from: http://www.cardiosource.org/News-Media/Publications/Cardiology-Magazine/2014/05/Study-Demonstrates-Benefits-of-Remote-Monitoring-in-Improving-ICD-Patient-Outcomes.aspx [cited September 2014]. 59

[And11] Andy Carvell. Rke analysis [online]. 2011. Available from: http://www.burningimage.net/rke/ [cited August 2014]. 31

[And14] Android Developers. Get the android sdk [online]. 2014. Available from: http://developer.android.com/sdk/index.html#download [cited September 2014]. 45, 52

[Bos10] Boston Scientific. Implantable cardioverter defibrillators - helping people with heart conditions live better [online]. 2010. Available from: http://www.bostonscientific.com/lifebeat-online/assets/pdfs/resources/CRM9-1040-0609_ICDPatientBSC.pdf [cited June 2014]. 4, 6, 7, 8

[Cen14] Centers for Disease Control and Prevention (CDC). Heart disease facts [online]. 2014. Available from: http://www.cdc.gov/heartdisease/facts.htm [cited June 2014]. 3

[DG06] Touby Drew and Maria L. Gini. Implantable medical devices as agents and part of multiagent systems. In Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone, editors, *AAMAS*, pages 1534–1541. ACM, 2006. 8

[(Ed11] Prof. Mithilesh R Das (Ed.). *Modern Pacemakers - Present and Future*. ISBN: 978-953-307-214-2. InTech, 2011. Available from: `http://www.intechopen.com/books/modern-pacemakers-present-and-future/`. xv, xvii, 5, 8, 9, 10, 11, 20, 21

[ETS14] ETSI - World Class Standards. Mobile technologies gsm [online]. 2014. Available from: `http://www.etsi.org/index.php/technologies-clusters/technologies/mobile/gsm` [cited August 2014]. 29

[Eur12] European Society of Cardiology. 2012 european cardiovascular disease statistics [online]. 2012. Available from: `http://www.escardio.org/about/Documents/EU-cardiovascular-disease-statistics-2012.pdf` [cited May 2014]. xv, 3, 4

[Far11] Charles S. Farlow. An overview of the medical device radiocommunications service (medradio) and future telemetry considerations. In *Proceedings of the 1st Invitational Workshop on Body Area Network Technology and Applications - Future Directions, Technologies, Standards and Applications*, 2011.

[FCC09] FCC - Federal Communications Commission. About medical device radiocommunications service [online]. 2009. Available from: `http://wireless.fcc.gov/services/index.htm?job=about&id=medical_implant` [cited June 2014]. 12

[FCC10] FCC - Federal Communications Commission. Small entity compliance guide - medical device radiocommunication service [online]. 2010. Available from: `http://www.fcc.gov/document/medical-device-radiocommunication-service-0` [cited June 2014]. 12

[GHR+11] Shyamnath Gollakota, Haitham Hassanieh, Benjamin Ransford, Dina Katabi, and Kevin Fu. They can hear your heartbeats: non-invasive security for implantable medical devices. In Keshav et al. [KLBM11], pages 2–13. ix, 15, 18, 19, 20, 37, 61

[Git13] GitHub. Extio_rtl [online]. 2013. Available from: `https://github.com/josemariaaraujo/ExtIO_RTL` [cited August 2014]. 32

[Hal94] Neil Haller. The s/key one-time password system. In *In Proceedings of the Internet Society Symposium on Network and Distributed Systems*, pages 151–157, 1994. 35

[HHBF+08] Daniel Halperin, Thomas S. Heydt-Benjamin, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Security and privacy for implantable medical devices. *IEEE Pervasive Computing*, 7(1):30–39, 2008. ix, xv, 1, 5, 8, 11, 12, 14, 15

[HHBR+08] Daniel Halperin, Thomas S. Heydt-Benjamin, Benjamin Ransford, Shane S. Clark, Benessa Defend, Will Morgan, Kevin Fu, Tadayoshi Kohno, and William H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, SP '08, pages 129–142, Washington, DC, USA, 2008. IEEE Computer Society. Available from: `http://dx.doi.org/10.1109/SP.2008.31`. ix, xv, 1, 6, 8, 11, 15, 16, 17, 18, 34, 36, 39, 59, 61

[Hin14] G. et al. Hindricks. Implant-based multiparameter telemonitoring of patients with heart failure (in-time): a randomised controlled trial. *The Lancet*, 384:583–590, 2014. 59

[HLM+] E. Himmrich, A. Liebrich, U. Michel, J. Neuzner, H. Pitschner, A. Heisel, E. Vester, U. Ganschov, and J. Jung. [Is ICD-programming for double intraoperative defibrillation threshold energy safe and effective during long-time follow-up? Results of a prospective randomized multicenter study (Low-Energy Endotak Trial–LEET)]. *Zeitschrift Für Kardiologie*, 88(2):103–112. Available from: `http://www.ncbi.nlm.nih.gov/pubmed/10209831`. 7

[Ian13] Ian Craggs. Mqtt security: Who are you? can you prove it? what can you do? [online]. 2013. Available from: `https://www.ibm.com/developerworks/community/blogs/c565c720-fe84-4f63-873f-607d87787327/entry/mqtt_security?lang=en` [cited September 2014]. 26

[IBM06] IBM. Ibm collaborates with st. jude medical on new state-of-the-art cardiac patient care system [online]. 2006. Available from: `http://www-03.ibm.com/press/us/en/pressrelease/19747.wss` [cited June 2014]. 25

[IBM14] IBM. Websphere mq telemetry [online]. 2014. Available from: `http://www-03.ibm.com/software/products/en/wmq-telemetry` [cited September 2014]. 52

[IKY13] Mohd Noor Islam, Jamil Khan, and Mehmet Rasit Yuce. A mac protocol for implanted devices communication in the mics band. In *BSN*, pages 1–6. IEEE, 2013. Available from: `http://dblp.uni-trier.de/db/conf/bsn/bsn2013.html#IslamKY13`. 12

[Jam03] James McCaffrey. Keep your data secure with the new advanced encryption standard [online]. 2003. Available from: `http://msdn.microsoft.com/en-us/magazine/cc164055.aspx` [cited June 2014]. 35

[KLBM11] Srinivasan Keshav, Jörg Liebeherr, John W. Byers, and Jeffrey C. Mogul, editors. *Proceedings of the ACM SIGCOMM 2011 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, Toronto, ON, Canada, August 15-19, 2011*. ACM, 2011. 64

[Mar05] Margaret Rouse. Nfc in action [online]. 2005. Available from: `http://searchsecurity.techtarget.com/definition/key-fob` [cited August 2014]. 31

[Mar13] Mario Taeubel. Hdsdr - high definition software defined radio [online]. 2013. Available from: `http://www.hdsdr.de/` [cited August 2014]. 32

[Med14a] Medtronic. Company overview [online]. 2014. Available from: `http://www.medtronic.com/about-us/company-profile/medical-technology.htm` [cited June 2014]. 20

[Med14b] Medtronic. Medtronic carelink network [online]. 2014. Available from: `http://www.medtronic.com/for-healthcare-professionals/products-therapies/cardiac-rhythm/patient-management-carelink/medtronic-carelink-network-for-cardiac-device-patients/index.htm` [cited June 2014]. 20

[N. 98] N. Haller, C. Metz, P. Nesser, M. Straw. A one-time password system [online]. 1998. Available from: `http://tools.ietf.org/html/rfc2289` [cited August 2014]. 36

[Nat14] National Institutes of Health - National Heart, Lung and Blood Institute. What is cardioversion [online]. 2014. Available from: `http://www.nhlbi.nih.gov/health/health-topics/topics/crv/` [cited May 2014]. 3

[NFC12a] NFC Tools. Felica [online]. 2012. Available from: http://nfc-tools.org/index.php?title=FeliCa [cited August 2014]. 27

[NFC12b] NFC Tools. Iso14443 [online]. 2012. Available from: http://nfc-tools.org/index.php?title=ISO14443 [cited August 2014]. 27

[NFC14a] NFC Forum. About the technology - nfc and contactless technologies [online]. 2014. Available from: http://nfc-forum.org/what-is-nfc/about-the-technology/ [cited August 2014].

[NFC14b] NFC Forum. Nfc in action [online]. 2014. Available from: http://nfc-forum.org/what-is-nfc/nfc-in-action/ [cited August 2014]. 27, 28

[NFC14c] NFC Forum. Tag type technical specifications [online]. 2014. Available from: http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/tag-type-technical-specifications/ [cited September 2014]. 45, 52

[NFC14d] NFC Forum. What is nfc? [online]. 2014. Available from: http://nfc-forum.org/what-is-nfc/ [cited August 2014]. 27

[NFC14e] NFC Forum. What it does [online]. 2014. Available from: http://nfc-forum.org/what-is-nfc/what-it-does/ [cited August 2014].

[NIC10] NICE - National Institue for Health and Care Excellence. Appraisal consultation document: Implantable cardioverter defibrillators for arrhythmias (review of existing guidance no. 11 [online]. 2010. Available from: http://www.nice.org.uk/guidance/index.jsp?action=article&o=32091 [cited June 2014].

[R05] Voelker R. New heart failure guidelines released. *JAMA*, 294(8):892, 2005. Available from: +http://dx.doi.org/10.1001/jama.294.8.892. 7

[rea09] readwrite - Richard MacManus. Mqtt poised for big growth - an rss for internet of things? [online]. 2009. Available from: http://readwrite.com/2009/07/22/mqtt_poised_for_big_growth#awesm=~oIp2aGZFX8Xlti [cited June 2014]. 25

[Rea14] Realtek. Rtl2832u [online]. 2014. Available from: http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=35&Level=4&Conn=3&ProdID=257 [cited August 2014]. 31

[Red12] I. Redbooks. *Building Smarter Planet Solutions With Mqtt and IBM Websphere Mq Telemetry*. IBM redbooks. Vervante, 2012. Available from: http://books.google.pt/books?id=LWqoMQEACAAJ. xv, 21, 22, 23, 24, 25, 26

[Riv95] Ronald L. Rivest. The RC5 Encryption Algorithm, 1995. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.53.9445. 17, 35

[Sco13] Scourgeforge. libwdi [online]. 2013. Available from: http://sourceforge.net/projects/libwdi/files/zadig/ [cited August 2014]. 32

[SSP+06] Joshua R. Smith, Alanson P. Sample, Pauline S. Powledge, Sumit Roy, and Alexander Mamishev. A wirelessly-powered platform for sensing and computation. In *Proceedings of the 8th International Conference on Ubiquitous Computing*, UbiComp'06, pages 495–506, Berlin, Heidelberg, 2006. Springer-Verlag. Available from: http://dx.doi.org/10.1007/11853565_29. 16

[St.08] St. Jude Medical. Merlin@home [online]. 2008. Available from: https://health.sjm.com/~/media/SJM%20Health/AA/PDF/Merlinhomebrochure.ashx [cited June 2014]. 21

[St.11] St. Jude Medical. Merlin.net patient care network (pcn) version 5.0 spec sheet [online]. 2011. Available from: http://professional.sjm.com/products/crm/connectivity-remote-care/remote-care/merlin-net-patient-care-network-pcn#tech-specs [cited June 2014]. 20

[St.13] St. Jude Medical. Merlin.net patient care network (pcn) [online]. 2013. Available from: http://professional.sjm.com/products/crm/connectivity-remote-care/remote-care/merlin-net-patient-care-network-pcn [cited June 2014]. 20

[St.14a] St. Jude Medical. About us [online]. 2014. Available from: http://www.sjm.com/corporate/about-us [cited June 2014]. 20

[St.14b] St. Jude Medical. We are st. jude medical [online]. 2014. Available from: http://www.sjm.com/~/media/SJM/corporate/About%20Us/new/5995_Corporate_Fact%20Sheet_US-2000651AEN-Final.ashx [cited June 2014]. 20

[The14] The Eclipse Foundation. Eclipse - downloads [online]. 2014. Available from: https://www.eclipse.org/downloads/ [cited September 2014]. 45, 52

[Tho13a] Thomas Skjolberg. Ndef tools for android [online]. 2013. Available from: https://code.google.com/p/ndef-tools-for-android/downloads/list [cited September 2014]. 45

[Tho13b] Thomas Skjolberg. Ndef tools for android [online]. 2013. Available from: https://code.google.com/p/ndef-tools-for-android/ [cited September 2014]. 46

[Vah13] E. Vahidian. Evolution of the sim to esim. 2013. Available from: http://www.diva-portal.org/smash/get/diva2:617036/FULLTEXT01.pdf. 23, 29, 41

Appendix A

Implementation of Architecture A3's Proof-of-Concept Example

A.1 ICD Log file created example

```
---Patient information---
Patient ID: 13665079 (Citizen card number)
Patient Name: Nuno Miguel Alves dos Santos
Patient Birthdate: 08-04-1991 (dd-mm-yyyy)
Patient Address: Covilha - Portugal (City - Country)
Patient Postal Code: 6200-707
Patient Contact: +351xxxxxxxxx
Patient Diagnosis: Ventricular Tachycardia and risk of Ventricular Fibrillation.
Patient Physician name: Paul Crocker
Patient Physician contact: +351xxxxxxxxx

---ICD information---
ICD ID: 123456789
ICD Device: Evera XT DR/VR (Medtronic product example)
ICD Battery life: 90%
ICD Date of implant: 02-03-2014
ICD Last check-up: 02-03-2014

---Medical telemetry--- (clean after sharing data to the programmer)
MT BPM: Average of 70 (since last data share)
BPM anomaly001 date: 18-07-2014
BPM anomaly001 hour: 15h25
BPM anomaly001 BPM: 200

---Device therapy settings--- (Device programming)
BPT: 45 (bradycardia pacing therapy - execute below x BPM)
ATP: 120 (antitachycardia pacing - execute above x BPM)
DF: 200 (Defibrillation - execute above x BPM)
```

A.2 *DefaultNfcTagWriterActivity.java*

```
/*************************************************************************
 *
 * This file is part of the 'NDEF Tools for Android' project at
 * http://code.google.com/p/ndef-tools-for-android/
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 * http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 *
 ************************************************************************/

package org.ndeftools.boilerplate;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.nio.charset.Charset;
import java.util.Locale;

import org.ndeftools.Message;
import org.ndeftools.boilerplate.R;
import org.ndeftools.externaltype.AndroidApplicationRecord;
import org.ndeftools.util.activity.NfcTagWriterActivity;
import org.ndeftools.wellknown.TextRecord;

import android.content.pm.PackageInfo;
import android.content.pm.PackageManager.NameNotFoundException;
import android.nfc.NdefMessage;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;


/**
 *
 * Activity demonstrating the default implementation of the abstract tag writer activity.
 *
 * The activity uses a simple layout and displays some toast messages for various events.
 *
 * @author Thomas Rorvik Skjolberg
 *
 */

public class DefaultNfcTagWriterActivity extends NfcTagWriterActivity {

        private static EditText text;
        private static Button readLogFile ;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);

                setContentView(R.layout.writer);
```

```
text = (EditText) findViewById(R.id.text);
readLogFile  = (Button)  findViewById(R.id.button1);


readLogFile.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {

                // Reading Data From File
                File directory =
                                Environment.
                                    getExternalStoragePublicDirectory(
                                    Environment.DIRECTORY_DOWNLOADS);


                // Assumes that a file is available at the directory
                File file = new File(directory + "/icd_logs/log_001.txt")
                    ;

                if (!file.exists()) {
                        System.out.println("File not found :"+ directory)
                            ;
                        text.setText("File not found :"+file.getPath());
                        return;
                }


                Log.e("Testing", "Starting to read");
                BufferedReader reader = null;
                StringBuilder builder = new StringBuilder();

                try {
                        reader = new BufferedReader(new FileReader(file))
                            ;

                        String line="xxx";

                        while ((line = reader.readLine()) != null) {
                                builder.append(line);
                                builder.append("\n");

                        }

                        text.setText(builder.toString()); //copies the
                            string builder to the message in order to
                            send it


                } catch (Exception e) {
                        e.printStackTrace();
                        System.out.println("bronca");
                }

                finally {
                        if (reader != null) {
                                try {
                                        reader.close();
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        }
                }
                Log.e("Testing", "Finish read");


                readLogFile.setText("Use NFC now!");

        }
});
```

```
        setDetecting(true);

    }

    /**
     *
     * Create an NDEF message to be written when a tag is within range.
     *
     * @return the message to be written
     */

    @Override
    protected NdefMessage createNdefMessage() {

        // compose our own message
        Message message = new Message();

        // add an Android Application Record so that this app is launches if a
            tag is scanned :-)
        AndroidApplicationRecord androidApplicationRecord = new
            AndroidApplicationRecord();
        androidApplicationRecord.setPackageName(getPlayIdentifier());
        message.add(androidApplicationRecord);

        // add a Text Record with the message which is entered

        TextRecord textRecord = new TextRecord();
        textRecord.setText(text.getText().toString());
        textRecord.setEncoding(Charset.forName("UTF-8"));
        textRecord.setLocale(Locale.ENGLISH);
        message.add(textRecord);

        return message.getNdefMessage();
    }

    /**
     * Get Google Play application identifier
     *
     * @return
     */

    private String getPlayIdentifier() {
        PackageInfo pi;
        try {
                pi = getPackageManager().getPackageInfo(getPackageName(), 0);
                return pi.applicationInfo.packageName;
        } catch (final NameNotFoundException e) {
                return getClass().getPackage().getName();
        }
    }

    /**
     *
     * Writing NDEF message to tag failed.
     *
     * @param e exception
     */

    @Override
    protected void writeNdefFailed(Exception e) {
        toast(getString(R.string.ndefWriteFailed, e.toString()));
    }

    /**
     *
     * Tag is not writable or write-protected.
     *
     * @param e exception
     */
```

```
        @Override
        public void writeNdefNotWritable() {
                toast(getString(R.string.tagNotWritable));
        }

        /**
         *
         * Tag capacity is lower than NDEF message size.
         *
         * @param e exception
         */

        @Override
        public void writeNdefTooSmall(int required, int capacity) {
                toast(getString(R.string.tagTooSmallMessage,  required, capacity));
        }


        /**
         *
         * Unable to write this type of tag.
         *
         */

        @Override
        public void writeNdefCannotWriteTech() {
                toast(getString(R.string.cannotWriteTechMessage));
        }

        /**
         *
         * Successfully wrote NDEF message to tag.
         *
         */

        @Override
        protected void writeNdefSuccess() {
                toast(getString(R.string.ndefWriteSuccess));
        }

        /**
         *
         * NFC feature was found and is currently enabled
         *
         */

        @Override
        protected void onNfcStateEnabled() {
                toast(getString(R.string.nfcAvailableEnabled));
        }

        /**
         *
         * NFC feature was found but is currently disabled
         *
         */

        @Override
        protected void onNfcStateDisabled() {
                toast(getString(R.string.nfcAvailableDisabled));
        }

        /**
         *
         * NFC setting changed since last check. For example, the user enabled NFC in the
              wireless settings.
         *
         */

        @Override
```

```
        protected void onNfcStateChange(boolean enabled) {
                if(enabled) {
                        toast(getString(R.string.nfcSettingEnabled));
                } else {
                        toast(getString(R.string.nfcSettingDisabled));
                }
        }

        /**
         *
         * This device does not have NFC hardware
         *
         */

        @Override
        protected void onNfcFeatureNotFound() {
                toast(getString(R.string.noNfcMessage));
        }


        public void toast(String message) {
                Toast toast = Toast.makeText(this, message, Toast.LENGTH_LONG);
                toast.setGravity(Gravity.CENTER_HORIZONTAL|Gravity.CENTER_VERTICAL, 0, 0)
                    ;
                toast.show();
        }

        @Override
        protected void onTagLost() {
                toast(getString(R.string.tagLost));
        }

}
```

A.3 *writer.xml*

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:background="@drawable/background"

    >

     <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/writerTitle"
                android:padding="3dip"
       android:textColor="@color/black"
                android:textStyle="bold"
                android:textSize="30sp"
       />


   <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/writerDescription"
                android:padding="3dip"
       android:textColor="@color/black"
                android:textStyle="bold"
       />

    <EditText
                    android:id="@+id/text"
                    android:layout_width="fill_parent"
                android:layout_height="40dp"
                    android:hint="@string/writerHint"
                    android:background="@drawable/button"
                    />

<Button
       android:id="@+id/button1"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_below="@+id/retainedGroup"
       android:layout_centerHorizontal="true"
       android:layout_marginTop="26dp"
       android:text="Load log file" />


   <TextView
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/writerScanHint"
                android:padding="3dip"
       android:textColor="@color/black"
                android:textStyle="bold"
                android:layout_gravity="center_horizontal"
       />




</LinearLayout>
```

Appendix B

Implementation of Architecture A4's Proof-of-Concept Example

B.1 *PublishFragment.java*

```
/*
 * Licensed Materials - Property of IBM
 *
 * 5747-SM3
 *
 * (C) Copyright IBM Corp. 1999, 2012 All Rights Reserved.
 *
 * US Government Users Restricted Rights - Use, duplication or
 * disclosure restricted by GSA ADP Schedule Contract with
 * IBM Corp.
 *
 */
package com.ibm.msg.android;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.List;

import org.ndeftools.Message;
import org.ndeftools.Record;
import org.ndeftools.externaltype.AndroidApplicationRecord;
import org.ndeftools.externaltype.GenericExternalTypeRecord;
import org.ndeftools.wellknown.SmartPosterRecord;
import org.ndeftools.wellknown.TextRecord;
import org.ndeftools.wellknown.UriRecord;

import android.app.PendingIntent;
import android.content.Context;
import android.nfc.NdefMessage;
import android.nfc.NfcAdapter;
import android.nfc.NfcManager;
import android.os.Bundle;
import android.os.Environment;
import android.os.Parcelable;
import android.support.v4.app.Fragment;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.TextView;

import org.ndeftools.Message;
import org.ndeftools.Record;



import org.ndeftools.externaltype.AndroidApplicationRecord;
import org.ndeftools.externaltype.GenericExternalTypeRecord;
import org.ndeftools.wellknown.SmartPosterRecord;
import org.ndeftools.wellknown.TextRecord;
```

```
import org.ndeftools.wellknown.UriRecord;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.nfc.NdefMessage;
import android.nfc.NfcAdapter;
import android.os.Bundle;
import android.os.Parcelable;
import android.os.Vibrator;
import android.util.Log;
import android.widget.TextView;



/**
 * Fragment for the publish message pane.
 *
 */
public class PublishFragment extends Fragment {

        /**
         * @see android.support.v4.app.Fragment#onCreateView(android.view.LayoutInflater,
              android.view.ViewGroup, android.os.Bundle)
         */
        private static Button readLogFile;
        private static Button nfcLogFile;
        private static Button clearMessage;
        private static EditText message;
        private static CheckBox retainedCheck;

        private Context context;

        //protected static NfcAdapter nfcAdapter;
        //protected static PendingIntent nfcPendingIntent;


        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container,
                        Bundle savedInstanceState) {


                View rootView = LayoutInflater.from(getActivity()).inflate(R.layout.
                    activity_publish, null);

                readLogFile = (Button) rootView.findViewById(R.id.button1); //read log
                    from file in internal storage
                nfcLogFile = (Button) rootView.findViewById(R.id.button2); //read log
                    file from NFC
                clearMessage = (Button) rootView.findViewById(R.id.button3); //clear
                    text in message textBox
                message = ((EditText) rootView.findViewById(R.id.lastWill));
                retainedCheck = ((CheckBox) rootView.findViewById(R.id.retained));


                //method to read log from file directory with button push
                readLogFile.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {

                                // Reading Data From File
                                File directory = Environment.
                                    getExternalStoragePublicDirectory(Environment.
                                    DIRECTORY_DOWNLOADS);


                                // Assumes that a file is available at the directory
                                File file = new File(directory + "/icd_logs/log_001.txt")
                                    ;
```

```
                        if (!file.exists()) {
                                System.out.println("ola File not found :"+
                                    directory);
                                return;
                        }
                        System.out.println("File directory :"+ directory + "/
                            icd_logs/log_001.txt");
                        Log.e("Testing", "Starting to read");
                        BufferedReader reader = null;
                        StringBuilder builder = new StringBuilder();

                        try {
                                reader = new BufferedReader(new FileReader(file))
                                    ;

                                String line="xxx";

                                while ((line = reader.readLine()) != null) {
                                        builder.append(line);
                                        builder.append("\n");

                                        //System.out.println(line);
                                }

                                message.setText(builder); //copies the string
                                    builder to the message in order to send it
                                retainedCheck.setChecked(true); //sets the
                                    message as retained to be kept at the server

                        } catch (Exception e) {
                                e.printStackTrace();
                                System.out.println("bronca");
                        }

                        finally {
                                if (reader != null) {
                                        try {
                                                reader.close();
                                        } catch (IOException e) {
                                                e.printStackTrace();
                                        }
                                }
                        }
                        Log.e("Testing", "Finish read");
                }
        });


        //method to read log with NFC
        nfcLogFile.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        Intent intent = new Intent(getActivity(), Nfcread.class);
                        startActivityForResult(intent,1);

                }
        });

        clearMessage.setOnClickListener(new View.OnClickListener() {
                public void onClick(View v) {
                        message.setText("");
                }
        });

        return rootView;
}

public void onActivityResult(int requestCode, int resultCode, Intent data) {

        if (requestCode == 1) {
```

```
                        if(resultCode == Activity.RESULT_OK){
                                String result=data.getStringExtra("result");

                                message.setText(result); //copies the string result to
                                    the message in order to send it
                                retainedCheck.setChecked(true); //sets the message as
                                    retained to be kept at the server


                        }
                        if (resultCode == Activity.RESULT_CANCELED) {
                        }
                }
        }//onActivityResult

}
```

B.2 *activity_publish.xml*

```
<!--
Licensed Materials - Property of IBM

5747-SM3

(C) Copyright IBM Corp. 1999, 2012 All Rights Reserved.

US Government Users Restricted Rights - Use, duplication or
disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

-->
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >


    <LinearLayout
        android:id="@+id/topicGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" >

        <TextView
            android:id="@+id/topictextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="35dip"
            android:text="@string/topic" />

        <EditText
            android:id="@+id/lastWillTopic"
            android:layout_width="0dip"
            android:layout_height="wrap_content"
            android:layout_weight="0.22"
            android:ems="10"
            android:hint="@string/topicHint"
            android:inputType="text"
            android:text="ICD_patient_001" />

    </LinearLayout>

    <LinearLayout
        android:id="@+id/messageGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/topicGroup"
        android:layout_marginTop="25dp" >

        <TextView
            android:id="@+id/messageTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="15dip"
            android:text="@string/message" />

        <EditText
            android:id="@+id/lastWill"
            android:layout_width="0dip"
            android:layout_height="35dp"
            android:layout_weight="0.22"
            android:ems="10"
            android:hint="@string/messageHint"
            android:inputType="textMultiLine" />
    </LinearLayout>

    <LinearLayout
        android:id="@+id/qosGroup"
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/messageGroup"
        android:layout_marginTop="25dp" >

        <TextView
            android:id="@+id/qosTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="40dp"
            android:layout_marginTop="10dp"
            android:text="@string/qos" />

        <RadioGroup
            android:id="@+id/qosRadio"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal" >

            <RadioButton
                android:id="@+id/qos0"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:checked="true"
                android:text="@string/qos0" />

            <RadioButton
                android:id="@+id/qos1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/qos1" />

            <RadioButton
                android:id="@+id/qos2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/qos2" />
        </RadioGroup>
    </LinearLayout>

    <LinearLayout
        android:id="@+id/retainedGroup"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/qosGroup"
        android:layout_marginTop="25dp" >

        <TextView
            android:id="@+id/retainedTextView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginRight="10dp"
            android:text="@string/retained" />

        <CheckBox
            android:id="@+id/retained"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/empty" />
    </LinearLayout>

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/retainedGroup"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="26dp"
        android:text="@string/readfile" />
```

```
    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/button1"
        android:layout_centerHorizontal="true"
        android:text="@string/readfile2" />

    <Button
        android:id="@+id/button3"
        style="?android:attr/buttonStyleSmall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:text="Clear Message" />

</RelativeLayout>
```

B.3 *Nfcread.java*

```
package com.ibm.msg.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import java.util.List;

import org.ndeftools.Message;
import org.ndeftools.Record;
import org.ndeftools.externaltype.AndroidApplicationRecord;
import org.ndeftools.externaltype.GenericExternalTypeRecord;
import org.ndeftools.wellknown.SmartPosterRecord;
import org.ndeftools.wellknown.TextRecord;
import org.ndeftools.wellknown.UriRecord;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.nfc.NdefMessage;
import android.nfc.NfcAdapter;
import android.os.Bundle;
import android.os.Parcelable;
import android.os.Vibrator;
import android.util.Log;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class Nfcread extends Activity {


        private static final String TAG = "testing";

        private Context context;

        private static NfcAdapter nfcAdapter=null;
        private static PendingIntent nfcPendingIntent=null;

        private static Button goBack;


        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setTitle("NFC reading platform");
                setContentView(R.layout.activity_nfcread);

                Button goBack = (Button)findViewById(R.id.button1);

                goBack.setOnClickListener(new View.OnClickListener() {//method to read
                    log with NFC
                        public void onClick(View v) {
                                Intent goBackIntent = new Intent();
                                setResult(RESULT_OK, goBackIntent);
                                finish();
                        }
                });

                // initialize NFC
                nfcAdapter = NfcAdapter.getDefaultAdapter(this);
                nfcPendingIntent = PendingIntent.getActivity(this, 0, new Intent(this,
                    this.getClass()).addFlags(Intent.FLAG_ACTIVITY_SINGLE_TOP), 0);
```

```
    }

    public void enableForegroundMode() {
        Log.d(TAG, "enableForegroundMode");

        IntentFilter tagDetected = new IntentFilter(NfcAdapter.
            ACTION_TAG_DISCOVERED); // filter for all
        IntentFilter[] writeTagFilters = new IntentFilter[] {tagDetected};
        nfcAdapter.enableForegroundDispatch(this, nfcPendingIntent,
            writeTagFilters, null);
    }

    public void disableForegroundMode() {
        Log.d(TAG, "disableForegroundMode");

        nfcAdapter.disableForegroundDispatch(this);
    }

    @Override
    public void onNewIntent(Intent intent) {
        Log.d(TAG, "onNewIntent");

        StringBuilder builder = new StringBuilder();


        if (NfcAdapter.ACTION_TAG_DISCOVERED.equals(intent.getAction())) {

                Parcelable[] messages = intent.getParcelableArrayExtra(NfcAdapter
                    .EXTRA_NDEF_MESSAGES);
                if (messages != null) {

                        Log.d(TAG, "Found " + messages.length + " NDEF messages")
                            ; // is almost always just one

                        vibrate(); // signal found messages :-)

    // parse to records
    for (int i = 0; i < messages.length; i++) {
        try {
                List<Record> records = new Message((NdefMessage)messages[i]);

                Log.d(TAG, "Found " + records.size() + " records in message " + i
                    );

                for(int k = 0; k < records.size(); k++) {
                        Log.d(TAG, " Record #" + k + " is of class " + records.
                            get(k).getClass().getSimpleName());

                        Record record = records.get(k);

                        if(record instanceof AndroidApplicationRecord) {
                                AndroidApplicationRecord aar = (
                                    AndroidApplicationRecord)record;
                                Log.d(TAG, "Package is " + aar.getDomain() + " "
                                    + aar.getType());
                        }

                        else if(record instanceof TextRecord) {
                                TextRecord textRecord = (TextRecord)record;

                                if(textRecord.hasEncoding()) {
                                        //builder.append("encoding="+textRecord.
                                            getEncoding().displayName());
                                        Log.d(TAG, "Encoding = "+textRecord.
                                            getEncoding().displayName());
                                }
                                if(textRecord.hasText()) {
                                        builder.append(textRecord.getText());
                                }
```

```
                if(textRecord.hasLocale()) {

                        String language = textRecord.getLocale().
                            getLanguage();
                        String country = textRecord.getLocale().
                            getCountry();

                        if(country != null && country.length() >
                            0) {
                                Log.d(TAG, "Lang and Country = "+
                                    language + "-" + country);
                        } else {
                                Log.d(TAG, "Language = "+language
                                    );
                        }
                }

                Intent returnIntent = new Intent();
                returnIntent.putExtra("result", builder.toString
                    ());
                setResult(RESULT_OK,returnIntent);
                finish();

        }

        else
                if(record instanceof GenericExternalTypeRecord) {
                        GenericExternalTypeRecord externalType =
                            (GenericExternalTypeRecord)record;

                        Log.d(TAG, "Generic");

                        if(externalType.hasDomain()) {

                        }
                        if(externalType.hasType()) {

                        }

                        if(externalType.hasData()) {

                                byte[] data = externalType.
                                    getData();

                                String s = context.getString(R.
                                    string.dataSize, externalType
                                    .getData().length );
                                builder.append("type="+s);
                        }
                }

                else if(record instanceof SmartPosterRecord) {
                        SmartPosterRecord smartPosterRecord = (
                            SmartPosterRecord)record;

                        if(smartPosterRecord.hasTitle()) {
                                TextRecord title =
                                    smartPosterRecord.getTitle();
                                if(title.hasText()) {
                                        Log.d(TAG, "title="+title
                                            .getText());
                                }
                                if(title.hasLocale()) {

                                        String language = title.
                                            getLocale().
                                            getLanguage();
                                        String country = title.
                                            getLocale().
                                            getCountry();
```

```
                                                        StringBuffer buffer = new
                                                             StringBuffer();


                                                        if(country != null &&
                                                            country.length() > 0)
                                                             {
                                                                buffer.append("
                                                                    [" + language
                                                                     + "-" +
                                                                    country +
                                                                    "]");
                                                        } else {
                                                                buffer.append("
                                                                    [" + language
                                                                     + "]");
                                                        }

                                                        String encoding = title.
                                                            getEncoding().
                                                            displayName();
                                                        if(encoding != null &&
                                                            encoding.length() >
                                                            0) {
                                                                buffer.append("
                                                                    [" + encoding
                                                                     + "]");
                                                        }

                                                        Log.d(TAG, "local"+buffer
                                                            .toString() );
                                                } else {

                                                }
                                        }

                                        if(smartPosterRecord.hasUri()) {


                                                UriRecord uri = smartPosterRecord
                                                    .getUri();
                                                if(uri.hasUri()) {
                                                        Log.d(TAG, "uri"+uri.
                                                            getUri().toString());
                                                }
                                        }

                                }


                        }

                } catch (Exception e) {
                        Log.e(TAG, "Problem parsing message", e);
                }

        }

}
                } else {
                        // ignore
                }
        }

        @Override
        protected void onResume() {
                Log.d(TAG, "onResume");

                super.onResume();
```

```
            enableForegroundMode();
        }

        @Override
        protected void onPause() {
            Log.d(TAG, "onPause");

            super.onPause();

            disableForegroundMode();
        }

        private void vibrate() {
            Log.d(TAG, "vibrate");

            Vibrator vibe = (Vibrator) getSystemService(Context.VIBRATOR_SERVICE) ;
            vibe.vibrate(500);
        }


        @Override
        public boolean onCreateOptionsMenu(Menu menu) {
            // Inflate the menu; this adds items to the action bar if it is present.
            getMenuInflater().inflate(R.menu.nfcread, menu);
            return true;
        }

        @Override
        public boolean onOptionsItemSelected(MenuItem item) {
            // Handle action bar item clicks here. The action bar will
            // automatically handle clicks on the Home/Up button, so long
            // as you specify a parent activity in AndroidManifest.xml.
            int id = item.getItemId();
            if (id == R.id.action_settings) {
                    return true;
            }
            return super.onOptionsItemSelected(item);
        }

}
```