ECDH-Based Communication Protocol for Implantable Cardioverter Defibrillators: Feasibility Analysis

BY

FILIPPO REZZONICO
B.S, Politecnico di Milano, Como, Italy, 2017

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2020

Chicago, Illinois

Defense Committee:

Jason Polakis, Chair and Advisor
Chris Kanich
Stefano Zanero, Politecnico di Milano

ACKNOWLEDGMENTS

Firstly, I want to thank my parents for their constant support during my stay in Chicago. I want also to thank the roommates I lived with in Chicago, with whom I have shared this wonderful experience. Lastly, I want to thank my girlfriend for the encouragement that she gave me during this thesis completion.

FR

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

CHAPTER **PAGE**

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

BLE	Bluetooth Low Energy
CA	Certification Authority
CPU	Central Processing Unit
ECC	Elliptic Curve Cryptography
ECG	ElectroCardioGram
ECDH	Elliptic Curve Diffie-Hellman
GATT	Generic Attribute Profile
ICD	Implantable Cardioverter Defibrillator
IDE	Integrated Development Environment
IMD	Implantable Medical Device
ISA	Instruction Set Architecture
ISM	Industrial, Scientific and Medical
LiPo	Lithium Polymer
MICS	Medical Implant Communication Service
NDA	Non-Disclosure Agreement
PKI	Public Key Infrastructure
RAM	Random Access Memory

LIST OF ABBREVIATIONS (continued)

RF	Radio Frequency
USB	Universal Serial Bus
UUID	Universal Unique Identifier

SUMMARY

Technological advancements in the healthcare domain made Implantable Cardioverter Defibrillators (ICDs) able to communicate wirelessly with external devices. This improvement lead to the possibility of exchange health data and perform therapy modifications without the need to perform any surgery on the patient. Most recent ICD models are now able to interact with external devices using a long-range communication (2-3 meters). Despite this, the introduction of this enhanced communication capability made ICDs vulnerable on a security perspective, opening them to a whole new set of attacks from the outside world. Balancing security with the need to provide access during emergency conditions while also taking into account the technological constraints of such devices is not trivial.

In this work we describe in depth the functioning of an ICD, while presenting also some of its most important structural characteristics. Consequently, taking into account this data and the context in which ICDs operate, we propose an Elliptic Curve Diffie-Hellman (ECDH) based communication protocol that, if combined with a Public Key Infrastructure (PKI), will increase the security of such devices without hindering their main functionalities or decreasing the quality of life of the patient. Since ICDs are resource constrained devices, we realized a proof-of-concept of our idea which simulates such limitations to demonstrate the feasibility of our approach.

Public-key cryptography, on which our approach is based, has received little to no attention in this topic since it has been considered too resource demanding for ICDs. In this work we

have also carried out a feasibility analysis based on three main metrics to demonstrate that our proposal is duable on ICDs and to show to the research community that this unexplored path could lead to interesting results.

CHAPTER 1

INTRODUCTION

In the last two decades Implantable Medical Devices (IMDs) has received an increasing attention due to the important role they have in improving patient quality of life. These devices are intended to be partially or totally inserted inside the human body and their purpose is to help the patient to treat disorders or chronic diseases. Among all IMD types the most advanced ones are active IMDs, which require a power source to actuate any medical operation. Examples of active IMDs are cardiac pacemakers, neurostimulators, infusion pumps and Implantable Cardioverter Defibrillators (ICDs). In particular, ICDs are small and resource-constrained medical devices implanted in the patient chest to monitor heart rate and deliver shocks to fix abnormal heartbeats if detected. These devices are generally used to treat ventricular tachycardia and prevent sudden cardiac arrest.

ICDs demand will continue to rise in the next years. This is primarily due to factors like the growth of aging population and the increasing prevalence of cardiovascular diseases [1], [2], [3]. Since the diffusion of such devices is destined to grow, also the technological improvements in this field of application are increasing. Current ICDs are able to communicate outside of the human body in order to exchange patient's medical data and re-configure device settings. This increased communication capability is generally referred as "wireless telemetry" and greatly improves patient care while reducing times and costs associated with patient treatment. However, these improvements in the healthcare domain come with an important drawback: a much

wider attack surface that exposes ICDs to security attacks.

Current research has shown that both passive and active attacks can be performed on ICDs with Commercial-Off-The-Shelf equipment by reverse engineering the proprietary communication protocol, demonstrating that few or no security measures are present in recent ICD models [4]. Therefore, considering ICD diffusion trends and their life-saving role, it becomes essential to add an extra security layer while impacting as little as possible on ICD battery lifetime. Many security solutions have been presented in the academic literature, each of them with advantages and drawbacks. Despite this, very little research has been done on the use of public key cryptography to improve security on ICDs, primarily because it has been considered too resource consuming for this kind of devices.

The aim of this thesis is to follow this unexplored research path and to verify its feasibility in the ICD domain. Therefore, we propose a new ICD communication protocol based on the Elliptic Curve Diffie-Hellman (ECDH) shared key exchange and realize a proof of concept that takes the main ICD limitations into consideration. Consequently, we perform measurements on the execution time, the RAM usage and the battery consumption of our protocol and discuss the feasibility of implementing it inside of current ICDs.

CHAPTER 2

PROBLEM DESCRIPTION

In this chapter we will give the reader an insight on the Implantable Cardioverter Defibrillators domain. In particular, we will first provide a detailed functional and structural analysis of ICDs and how they interact with the outside world. In the second section, we will describe the most significant ICD vulnerability and the consequences it may have on patient privacy and safety. Following, a threat model will give information about potential attackers and their goals, while also specifying the most common attacks and the effects they can have on ICDs. Lastly, we will present the most important tradeoffs and limitations in the context of ICDs.

2.1 Implantable Cardioverter Defibrillator background

In the domain of active Implantable Medical Devices, ICDs have gained the greatest attention among researchers. This is probably due to the critical role they serve in preserving patients life and the increasingly growing demand of such devices in the last two decades. The most recent models of ICDs are able to monitor the patient's heartbeat, detect arrhythmias and perform pacing in order to stabilize the heart rate. Moreover, all ICDs are able to perform defibrillation against life-threatening ventricular tachycardia and ventricular fibrillation. This process consists in delivering electric current to the heart in order to make it re-establish a normal rhythm, saving the patient's life.

Technological advances affected ICDs improving their battery lifetime, while reducing their

size and weight [5]. This progress in ICD technology simplified their implantation inside the human body while reducing the necessity of implant replacement. One of the most important improvements is the addition of a telemetry module that enables a longer range of communication between ICDs and the outside world. Before the introduction of this module, ICDs could only communicate outside the human body through an inductive link, which has several limitations in terms of communication range (no more than 6 cm) and data rate (100 kbps) [6]. This inductive link communication required the proximity of a programming head in order to communicate with external devices. Therefore, patients were forced to keep this programming head close to their implant for the whole duration of the message exchange.

Currently, thanks to the telemetry interface, ICDs can communicate with external devices at a maximum distance of 2-3 meters [7], [8] at a higher data rate with the need to keep a programming head close to the patient chest only during the ICDs telemetry module activation procedure [9]. This improvement in ICD communication brought significant advantages in the patient care. Firstly, ICD implantation and replacement has become easier and faster, since devices can monitor the patient status and be configured in any moment during the surgical procedure. Another important aspect to consider is that doctors and clinicians can measure ECG waveforms wirelessly during follow-ups, saving time for the setup of surface electrodes and improving patient comfort during the visit. Doctors are also able to change the parameter settings of the device during these visits in order to deliver the best treatment possible, considering current patient health status. Lastly, patients can transmit to hospitals health data while sleeping at their homes reducing the number of required in-hospital follow-up visits. This aspect

also allows doctors to constantly monitor their patients' conditions, detect alarming situations and perform corrective measures before complications may occur.

There are two types of external devices which generally communicate with ICDs using the wireless telemetry: base stations and programmers. Base stations are home monitoring devices that are generally put close to the patient's bed. They retrieve patient's health data from the ICD and send it to the health care provider through an internet connection or a phone line. Programmers are devices used by the hospital staff not only to monitor the patient status, but also to change ICD's settings. Both these devices are generally equipped with a programming head, whose purpose, once placed over the patient chest, is to activate the wireless module and enable the longer range communication with the ICD. Figure 1 shows a schematic representation of the ICD communication system.

2.1.1 ICD's structural analysis

In order to better understand how ICDs operate and what limits may incur with this kind of devices we will provide a technical description of ICD's architecture. Information on ICD's internal structure are hard to retrieve, but useful data can be found in the research literature, inside ICDs patents and manufacturers manuals [10], [11].

Firstly, an ICD is composed of a pulse generator and one or more leads connected to the patient hearth. The pulse generator and the leads are connected to each other thanks to an epoxy resin connector. A schematic illustration of an ICD is shown in Figure 2. The external housing of the pulse generator, which, once implanted, is in contact with human tissues, is generally made of biocompatible titanium [7], [12]. Inside of this hermetically sealed housing there are many

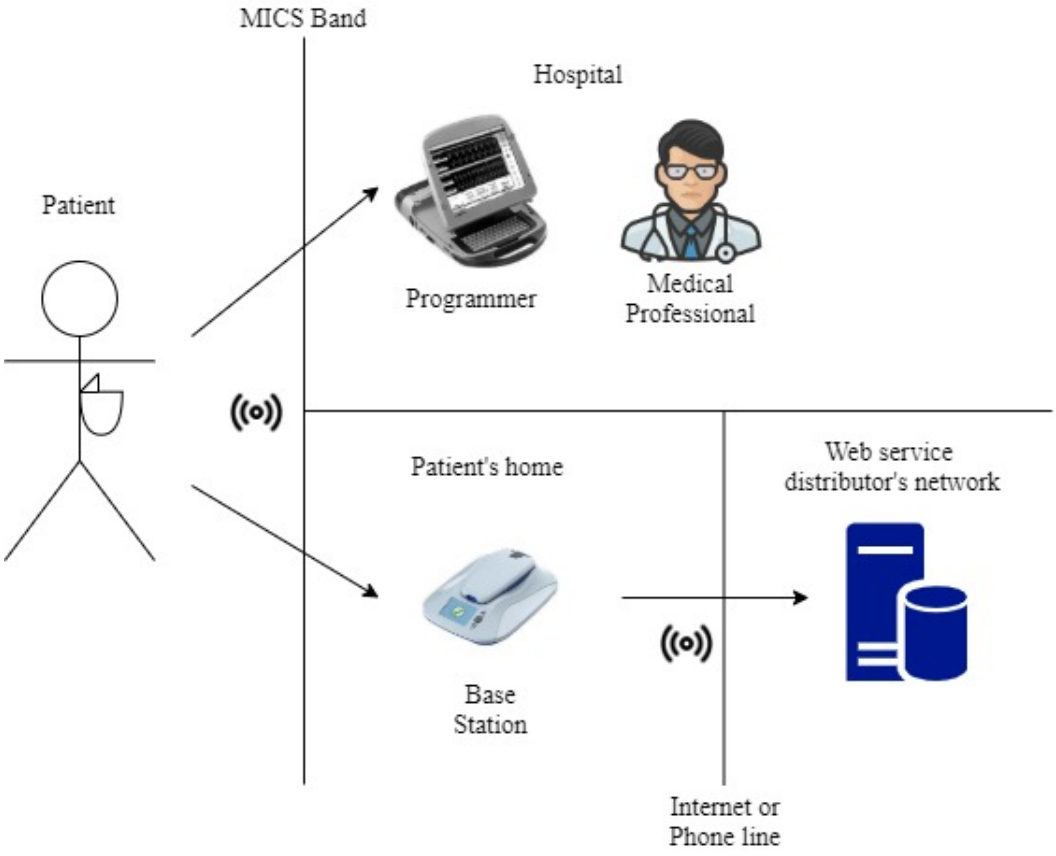

Figure 1: ICD's communication system

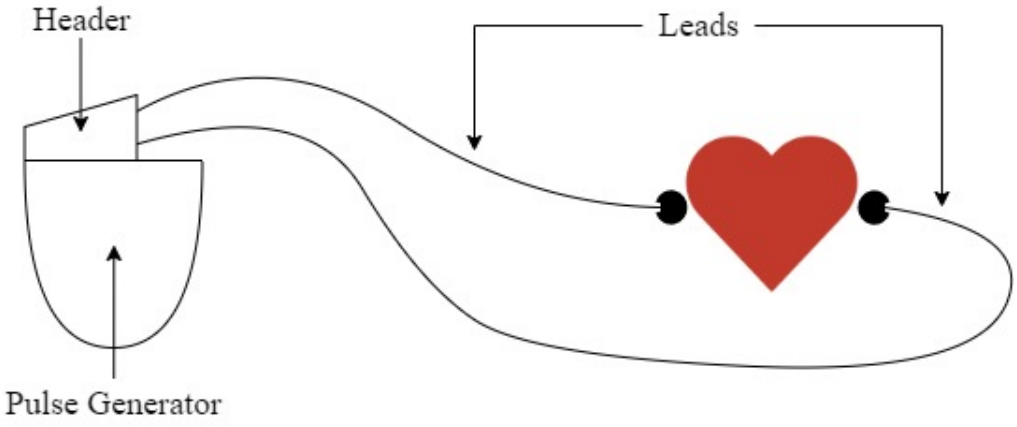

Figure 2: ICD's main external components

components which can be summarized in four main modules.

The first one is the *controller* module, which sends control signals to modify the behaviour of the pulse generator. The controller contains a microprocessor, which is in charge managing the interactions between modules and executing ICD's firmware stored inside the memory module. Recent research pointed out ARM-Cortex M3 as the current IMD's microcontroller [13]. Inside of the controller module there is also logic and timing circuitry which synchronizes the pulse generator.

The *memory* module comprises RAM and ROM memory. ROM memory stores the proprietary code of the ICD's manufacturer, while RAM memory stores patient's specific parameters, temporary variables, used during algorithms execution, and ECG signals. RAM memory size of ICDs goes from 128 KB to 1024 KB, and about three quarters of it are dedicated to store

measured ECG signals [14], [15].

The *power source* module comprises two main elements: a battery and capacitors. Recent ICD models use lithium-silver vanadium oxide or lithium-manganese dioxide batteries to power up all other ICD components [16]. Current models of ICD have a battery capacity that ranges from 1 Ah for the smaller models to 2 Ah for the extended lifetime models [17], [18]. The capacitor main function is to accumulate, store and deliver high energy packages in a small amount of time in order to perform defibrillation shocks, if needed [19]. This module alone occupies from half to two-thirds of the whole ICD's volume.

Lastly, the *telemetry* module is responsible for the communication with external devices and brings all the advantages previously specified. This module is generally composed of two RF (radio frequency) receivers and one RF transmitter. One of the two receivers is a broadband RF wakeup receiver which generally operates in the Industrial, Scientific and Medical (ISM) band at 2.4000 to 2.4835 GHz. Alternatively, an inductive wakeup receiver can be present in place of the RF wakeup receiver. The purpose of this receiver is to listen to wake-up messages from external devices so that the other components of this module can be powered down when they are not necessary, increasing battery lifetime. When a wake-up call is received, the other receiver and the transmitter become active and able to communicate with the external device within the Medical Implant Communication Service (MICS) radio band at 402 to 405 MHz [10]. MICS band is preferred for exchanging messages because it has a better conductivity in a patient's body and a longer communication range [6]. Both the receivers and the transmitter can be connected to one common antenna or, alternatively, the device can have more than

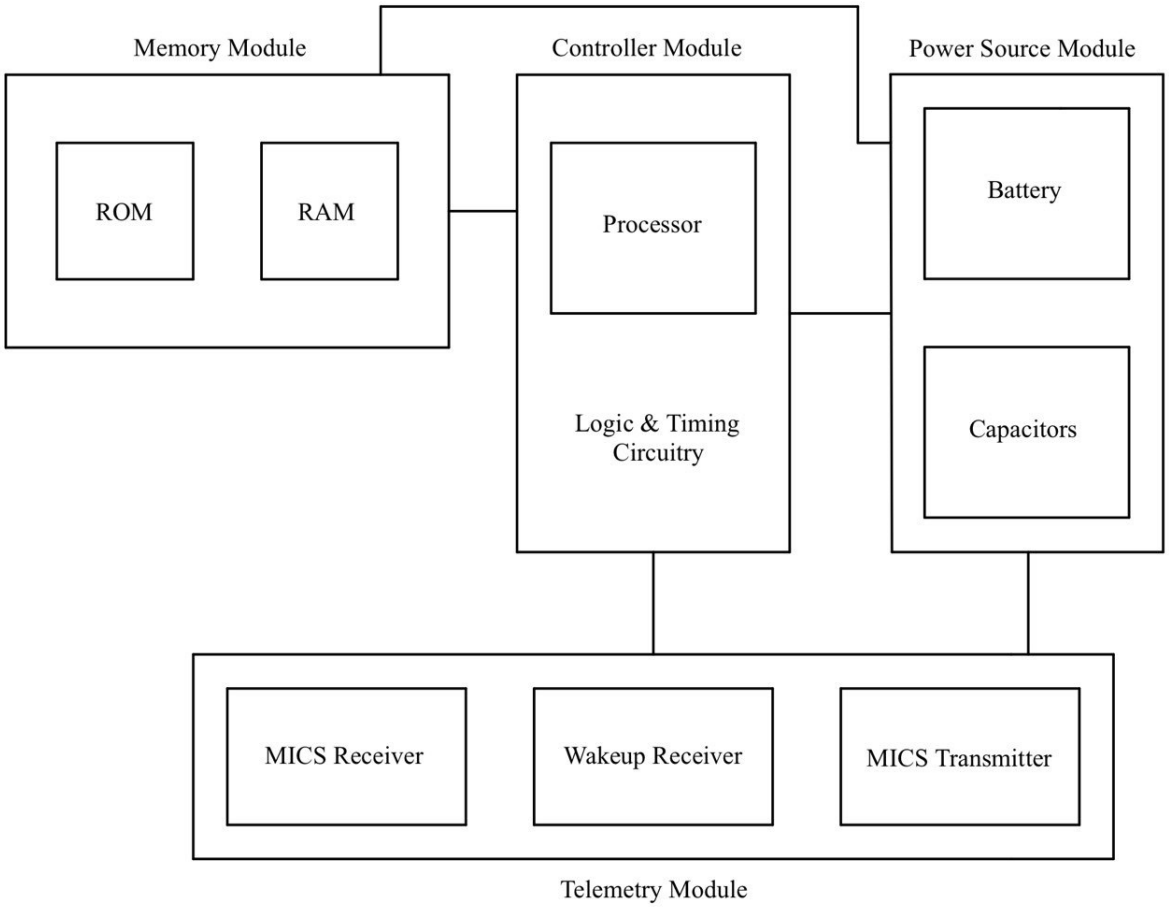

Figure 3: ICD's main internal modules

one antenna connected to one or more receiver/transmitter. A schematic representation of the described modules is depicted in Figure 3.

2.1.2 ICD's functional analysis

In this section we will describe how an ICD communicates with a programmer during a follow-up visit at the hospital. In doing so, we will analyze the ICD's state machine and how

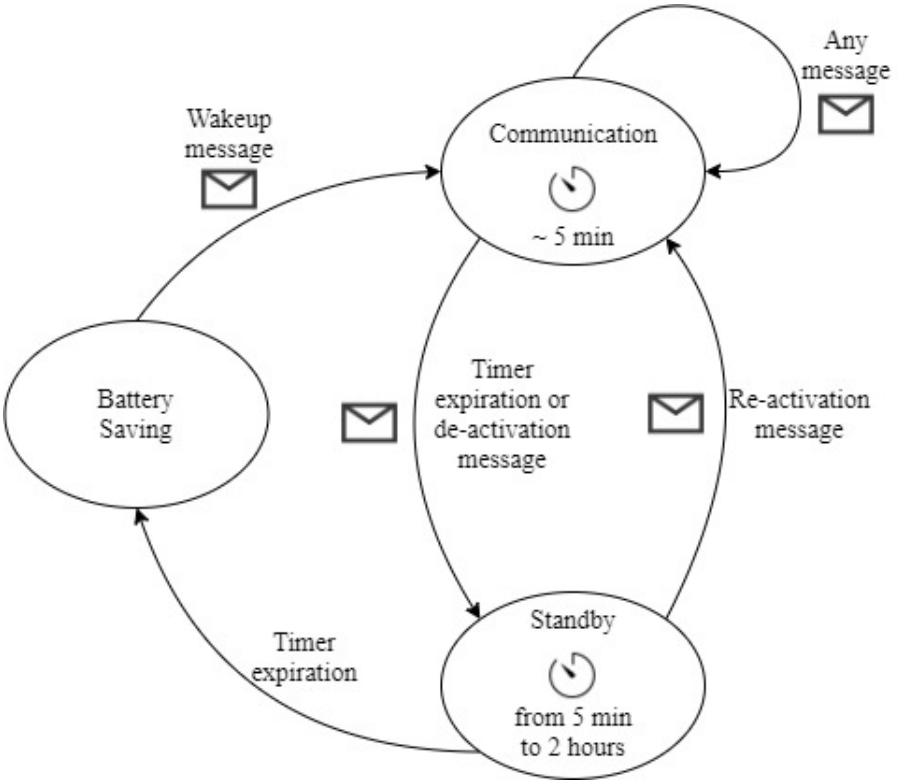

Figure 4: ICD's functional state machine

the previously described technical components interact with each other to guarantee a correct message exchange. After reading some ICDs manufacturers technical manuals [8], [7], we were able to schematize a common state machine that describes ICD's behavior during the communication with an external device. We identified 3 different states during a patient follow-up session: *battery saving* state, *communication* state and *standby* state. A representation of the ICD's state machine is represented in Figure 4.

During the battery saving state, an ICD keeps most components of the telemetry module

powered off and, therefore, the long range (2-3 m) communication is disabled. When the programming head of a base station or of a programmer gets close to the patient chest the wakeup receiver of the telemetry module receives a wakeup message. After an interaction with the controller module, the telemetry module gets powered, going from the battery saving state to the communication state. The programming head can now be removed from the patient's chest. During the communication state the long range MICS band communication is enabled and ICDs can exchange data or receive a parameter modification from the programmer.

If the ICD receives no message or command from the external device it will stay in the communication state for a limited amount of time, generally of about 5 minutes. If any message is received, the timer will reset. If the timer expires, the ICD goes from the communication state to the standby state. During this state, the ICD will not exchange data nor accept parameter modifications, but it still has its telemetry module completely powered on and therefore the long range communication is still enabled. An ICD can go from the communication state to the standby state also when the programmer sends a specific standby message to the ICD. If the implanted device receives a specific re-activation message from a programmer it can return to the communication state. Otherwise, after a variable amount of time, which goes from 5 minutes to 2 hours, it goes back from the standby state to the battery saving state, disabling the telemetry module. The duration of this second timer can be modified during the patient session and its default value is of 5 minutes.

Other ICDs have a simpler state machine without the standby state. Once the communication state timer expires they directly go back to the battery saving state.

2.2 ICD's functional analysis

Since ICDs are life-saving devices, their manufacturers have many regulations and design constraints to follow in order to guarantee that their products are safe and durable. These device must be small and lightweight, made with biocompatible material (at least the components that are in contact with human tissues) and stay inside the human body for several years before being replaced. All these concerns have primary importance because the main role of the ICD is increasing the patient quality of life for as long as possible. On the other hand, an aspect that is marginally or not considered is the security and privacy of these devices. ICDs contain patient's sensitive information like personal and medical data. Moreover, since ICDs perform parameter-based therapies to treat heart chronic diseases, an ill-intention individual could try to alter the device functioning and endanger the patient's life. Therefore, it becomes important to analyze ICD's weaknesses and the attacks that can be performed on such devices in order to find an optimal solution to protect them.

As stated in the previous section, the introduction of a telemetry module inside of the ICD architecture brought significant advantages to the treatment of heart related chronic diseases while reducing the number of required follow-up visits in hospital, saving time and related costs. On the other hand, this new "long-range" communication has really widened ICD's attack surface. Documented attempts in the research literature [20] demonstrated that ICD's wireless communication presents little to no security mechanisms. Thanks to this vulnerability, it is possible to reverse-engineer the proprietary communication protocol and perform a variety of attacks targeting data confidentiality and device treatments. Another important aspect that

was shown in [4] is that it is possible to abilitate the long range communication without placing a programming head close to the patient's chest. This can be done exploiting the ICD's state machine or by sending specific set of messages that are always identical in every communication session of the same ICD.

2.2.1 ICD's attacker types and goals

Understanding the different attacker types and their goals is essential to understand why it is necessary to add security measures on ICDs. We identified three main types of attackers. They will be analyzed from the weakest one, which has limited resources and no knowledge about ICDs, to the strongest one.

The *passive* attacker uses cheap equipment to intercept the messages exchanged between the ICD and an external device in order to obtain device specific, patient or medical data. This attacker type is the weakest one since it is not able to create custom messages or alter the real ones. Therefore, this attacker can only damage message privacy and confidentiality without altering the device operations.

The *active* attacker has more sophisticated equipment and is able to listen to ICD's communications and send back the intercepted messages to the ICD in order to establish a connection with it. This attacker type has no knowledge of the message format of the ICD's proprietary communication protocol but is able to resend previously captured messages and perform random modifications to them. Active attackers can deplete IMDs batteries by continuously send connection requests, impersonate trusted external devices and trigger existing treatments when they are not necessary. Therefore they have a limited impact on the device correct functioning

and can lower the patient treatment quality or forcing him to an early device replacement.

The last and strongest type is the *expert active* attacker which knows or is able to discover the proprietary protocol message format. He is able to perform the same attacks of the previously described attacker but can also create new messages with the intention to stop or deliver uncorrect therapies or alter the device settings. This attacker can seriously harm the patient and even endanger his life.

After presenting the different types of attackers, we will describe the three main goals that they want to achieve. Firstly, an attacker may want to *steal sensitive information*. This goal is not only limited to personal and medical data but also device related data, which can be sold to interested parties or used for his own personal advantage. This goal can be achieved by all the three previous attacker types.

An attacker may also want to *damage* an hospital's or a manufacturer's *reputation* by lowering the treatment quality or forcing an early replacement of the device. For example, since the implantable medical device market is competitive, a manufacturer may be interested in damaging a competitor reputation. This goal can be achieved by all active attackers types.

Lastly, an attacker may want to *cause physical harm* to a patient or even attempt on his life by affecting the ICD's behavior. This attack can only be performed by experienced active attacker.

2.2.2 Security attacks against ICD's telemetry

ICD's proprietary communication protocols often lack authentication, encryption and integrity check mechanisms, therefore, are vulnerable against most security attacks. In order to design a secure communication protocol, it is necessary to understand which attacks can be

performed on ICDs.

Eavesdropping attacks consist in intercepting the messages exchanged between an ICD and a programmer or base station and, exploiting the fact that they are not encrypted, steal sensitive information. This attack can be done both by passive and active adversaries with the difference that a passive one must have to wait a real communication between the ICD and a legitimate external device. This attack mainly targets both patient and device confidentiality.

Replay attacks are based on intercepting the legitimate messages sent by an external device to an ICD and re-transmitting them in another moment. Since ICDs do not perform any replay prevention technique on received messages, an active adversary can start a communication with an ICD and transmit the previously intercepted messages to perform settings modifications when they are not required.

An expert active attacker can also modify intercepted messages or forge new ones and perform a *Spoofing* attack convincing the ICD that it is communicate with a legitimate device and forcing it to deliver completely wrong therapies to the patient, endangering his health.

An adversary performs a *Man-in-the-Middle* (MitM) attack when he connects himself both with an ICD and a programmer making them believe they are communicating with each other. This attack can be performed by an expert active attacker to hijack a legitimate communication to transmit malicious messages to the ICD or the external device, while making them believe that the message exchange went exactly as expected.

Denial-of-Service (DoS) attacks consists in targeting the device availability by jamming the communications with legitimate devices or depleting the implantable device battery by contin-

uously sending wake-up request or keeping the telemetry module active forcing the patient to an early device replacement.

2.2.3 Securing ICDs: tradeoffs and limitations

Knowing ICDs vulnerabilities and which attacks can be performed on them is not sufficient to design a solution to secure the message exchange protocol. In securing ICDs we have also to consider the context in which they operate and analyze their limitations and how a secure solution can impact on their functioning.

The first aspect that must be considered is the importance of balancing security with accessibility. ICDs should always grant access to doctors and caregivers, without requiring them to perform complex or time consuming operations, while preventing any unauthorized access to ill-intentioned individuals. Considering ICD's context, this is not an easy task since different scenarios must be analyzed. For example, in a normal usage scenario, it may seem a good solution to store the ICDs access credentials in a secure location of the patient's hospital, so that only authorized medical personnel can access ICD data. On the other hand, in case of an emergency situation, a patient may be unconscious and far away from the hospital where the credentials to access his ICDs are stored. In such a situation, the patient will not be able to provide any access information and contacting the correct hospital may require time, slowing down any rescue attempt. Therefore, performing strong access policies in ICDs is not recommended in all situations and, in some cases, even free or easy access should be granted.

Another important aspect to consider is that ICDs are small devices with very limited resources, in terms of memory, computational capacity and battery. Therefore, before designing a security

scheme it is important to evaluate its feasibility in the ICD domain. Once an ICD is implanted inside a patient it can only be removed and replaced with a surgical procedure. Therefore, security solutions that are too energy consuming must be discarded because they could seriously reduce the implant longevity. Also, as stated in section 2.1.1, ICDs are equipped with few kilobytes of free RAM and simple energy-efficient processors and, therefore, they are not able to store big data structures or perform complex operations without taking a considerable amount of time.

Lastly, one limitation that affects current research on securing ICDs is that it is hard for researchers to gain enough and recent information in this domain. IMD manufacturers compete in a very challenging market and they tend to avoid sharing their know-how, proprietary code and current device samples because, in doing so, they could give precious advantages to their competitors. Another reason to explain this behaviour, is that IMDs manufacturers rely on security-through-obscurity to protect their devices. This approach, which consists in keeping their communication protocol details secret, has been proved to be insufficient to protect these devices against reverse engineering approaches [21]. Obviously, the lack of access to these devices and their technical specifications limits the research results and progress that can be obtained in such a complex domain.

CHAPTER 3

RELATED WORK

In this chapter we will briefly present and discuss the studies that have been performed on IMDs and ICDs that inspired our research work. Some of them focused on demonstrating the vulnerability of implantable devices, others proposed solutions to hinder any external attack attempt on them.

3.1 Laboratory attacks on IMDs

Although no documented malicious attacks have ever been recorded on IMDs, several research works and reports showed that such devices present little to no security mechanisms and that attacks can be performed on them to alter their correct functioning.

In 2008, Halperin et al. performed the first registered set of laboratory attacks on an ICD [20]. This work showed that the short-range communication between that ICD model and the correspondent programmer occurred completely unencrypted. Moreover, once understood the proprietary communication protocol by reverse engineering it, they were able to perform Replay attacks on the ICD, which enabled them to disclose private and medical information, change therapy settings, and even induce fibrillation. Since this work was published, the attention on securing an implantable device wireless telemetry has increased and the research community started performing experiments on other IMDs models and types, like insulin pumps [22], [23] and neurostimulators [24].

A more recent work on ICD vulnerabilities was made in 2016 by Marin et al. [4]. Taking inspiration from the previous work done by Halperin et al., they used a black-box approach to fully reverse-engineer the long-range communication protocol. Their work states that a simple form of data obfuscation was performed by the ICD manufacturer but not for the purpose of increasing the security level of the product. They also described several ways to avoid performing the short-range communication step in order to enable or keep enabled the long-range telemetry module. In this way, they succeeded in performing replay and spoofing attacks and demonstrated the vulnerability of different types of ICDs. These works were particularly useful for us to understand ICDs functioning, vulnerabilities and to define what attack types can be performed on them. In this way, we were able to start thinking of possible ways to secure ICDs against potential attackers.

3.2 Attack countermeasures for IMDs

The works presented in the previous section demonstrate that, even if many years have passed since the first discovery of ICDs vulnerabilities, no solid security mechanism has been applied by manufacturers to secure such devices from external attacks. For this reason, researchers focused their efforts on proposing different security mechanisms to protect IMD telemetry, while considering their context and limitations. Currently proposed solutions have been extremely helpful for us to understand which aspects should be taken into consideration when analyzing the feasibility of a new security scheme for ICDs. Among all the proposed solutions two main categories have received the most attention in the research community: *External device* and *biometric* based access control. Obviously, other types of approaches have been attempted

[25], [26] but they were either too complex to apply in the IMD domain or they were found vulnerable against certain types of attacks.

3.2.1 External device based solutions

The first of the two most investigated methods is based on the introduction of a third actor, a wearable external device, in the communication between the IMD and the programmer.
Using an external device as a "communication bridge" between the IMD and the programmer is a solution based on the *fail-open* access. This means that while the external device is in close proximity with the IMD, it will grant access only to authorized programmers or base stations. This can guarantee an adequate level of protection during normal operation mode. In case of an emergency, caregivers can remove the external device from the endangered patient and communicate freely with the IMD. This external device can be recharged periodically and therefore, can perform more resource-expensive operations in place of the implanted device. Denning et al. were the first to propose this approach [27] which was further refined and modified by other researchers [28], [29], [30].
There are two main disadvantages in applying these solutions on ICDs.
Firstly, these external devices will protect the implanted device only when they are close and able to communicate with it. This means that in case the external device is lost, broken or stolen the ICD becomes once again vulnerable. It is also important to consider that an attacker could try to jam the messages exchanged between the external device and the ICD, forcing the fail-open access. This problem has been addressed by Gollakota et al. [29] by applying friendly jamming to protect the ICD from adversaries. This solution, although effective, will probably

not be accepted, since in some states jammers are illegal and could also jam non malicious communications performed by other devices.

The second important aspect to consider is patient compliance. An external device that must always be worn by a patient may be seen as burden rather than a protection. Patients may start to forget wearing it or even decide not to use it.

3.2.2 Biometric based solutions

The other prominent approach that has been followed by researchers to secure IMDs is the usage of patient's biometric or physiological values to guarantee the authenticity of a programmer. There are two main variants of this approach.

The first one, proposed by Hei et al. [31], is based on loading one or more unmodifiable biometrics (fingerprints, iris images) before implanting the IMD inside the patient's body. When a programmer needs to obtain access to an implanted device it can measure from the patient these biometrics and, doing this, "prove" to the IMD that it is a legitimate device. The main objection that can be done on this security scheme is that any malicious entity could try to steal these biometrics to gain unlimited access to the victim's ICD. Another important aspect to consider is that in order to apply this solution programmers should be able to measure these biometrics. Current programmers are not able to measure iris or fingerprints, and even performing such measurements in an emergency situation may require some time to gather the required equipment, hindering the rescue attempt.

The second biometric-based variant was first proposed by Poon et al. [32] and is based on physiological signals that vary during time (ECG signals, for example). This solution requires

both the IMD and the programmer to measure synchronously a physiological value and use it to encrypt and decrypt a symmetric key that can be used for successive communications or, more simply, as proposed by Rostami et al. [13], to grant access to a programmer. The security of this approach lies in the fact that, in order to obtain access to an IMD, it is necessary to be in close contact to the patient while performing such measurement.

The main problem of this approach is that measuring the same biometric from different places in the human body, at slightly different times will introduce noise and therefore, will not produce equal results. The research on how to solve this issue is wide and many solutions have been proposed, but it is still an open topic. For example, Venkatasubramanian et al. [33] proposed to use a fuzzy vault scheme. This solution was then improved by Hu et al. [34]. Even if it becomes possible to securely exchange a key using physiological values, it is still important to consider the limitations of IMDs. Too complex solutions will rapidly deplete the battery of the IMD. Moreover, current research [35] showed that performing a real-time ECG measurement to derive or exchange securely a cryptographic key will take a considerable amount of time (around one minute), that in emergency situations can be determinant. For all these reasons we decided to verify the feasibility of a different approach, that will be described in detail in the next chapters.

CHAPTER 4

METHODOLOGY

As demonstrated in the previous chapters of this thesis, security on ICDs is a complex task to achieve. Therefore, in this chapter, we will give the reader a description off all the reasonings and the steps that lead us to conceive, design and implement our protocol, which attempts to improve privacy and security of ICD wireless telemetry, by demonstrating the feasibility of an ECDH-based communication protocol with current ICD hardware resources. Firstly, we will describe the process that we have followed to gather information about ICD context, main security problems, most recent technical specifications and current state-of-the-art proposals. Consequently, we will present the hardware setup that we chose in order to simulate an ICD and its technological limitations. Lastly, we will describe and explain the design choices that we have made to implement a lightweight ECDH-based communication protocol on the chosen hardware setup.

4.1 Information gathering

Understanding IMD's background and current state-of-the-art solutions has been the first step in protocol design. We started from the academic literature and we categorized papers and theses in two main groups: *protocol proposing* and *compilative*.

Protocol proposing works gave us a better insight on the most recent research paths that have been followed to guarantee IMDs privacy and security, highlighting the main IMD vulnerabili-

ties and the attack that can be performed on them.

Compilative works have been extremely precious for us in order to comprehend the most important concepts of how IMDs operate and the technical difficulties that incur when designing a security scheme for these devices. They were also helpful in categorizing the first group of works and in identifying their inner problems and limitations.

One common aspect of most protocol proposing works is that the designed solutions were not implemented on IMD or ICD samples but on prototypes that simulate their behaviour [13], [36], [37] or proposed without an actual implementation [38]. This is due to the fact that IMD manufacturers generally tend not to cooperate with university researchers and prefer to rely on their Research and Development departments. We have also tried to contact some of the most important ICD producers through their websites and customer services, but without great success. Some of them gave us very basic information that can be easily found on their websites while others completely refused to share any information even when we proposed them to sign a Non-Disclosure Agreement (NDA).

ICD-specific "technical manuals" can be found on producers' websites, but they generally provide useful information about materials in contact with human tissues and batteries, communication protocol details and other technical data are kept confidential. Thankfully, patents and other research works provided us with enough technical documentation to proceed with our research. The technical data that we used to make our consideration on our future hardware and protocol design choices have already been presented in section 2.1.1.

4.2 Hardware setup choice

As anticipated in the previous paragraph, obtaining current ICD samples is complex. Old models can be obtained by indirect means, like from relatives, friends or buying them online. But, even acquiring an old model is not enough, since, in order to exchange messages and modify the implantable device firmware, it is necessary to possess a programmer. Programmers are only sold by manufacturers to healthcare professionals and finding them elsewhere is hard and expensive. For these reasons, once gathered all the necessary required knowledge on ICD context, we started searching for a suitable hardware configuration that would best simulate an ICD behaviour during a message exchange with an external device. Obviously, we also had to take into account the limited resources available on this kind of device to perform our final choice. Arduino MKR Wifi 1010 board, shown in Figure 5 has several technical characteristics that are similar to current ICD technology. These characteristics will now be compared to ICD specifications already presented in paragraph 2.1.1.

Having a similar processor to the original ones inside of ICDs is helpful to understand what can be computed by them and the amount of time that is necessary to execute a specific algorithm. The microcontroller of our Arduino board is a SAM D21 [39] which contains an ARM Cortex-M0+ processor. In Table I we compare the Cortex M0+ with the processor in current IMDs, the ARM Cortex-M3.

Figure 5: Arduino MKR Wifi 1010 board

Property\Processor	Cortex-M0+	Cortex-M3
CoreMark/MHz	2.46	3.34
DMIPS/MHz	0.95	1.25
Dynamic power	47.4 µW/MHz	141 µW/MHz
Floor planned area	0.098 mm2	0.35 mm2
Pipeline Stages	2	3
ISA	Armv6-M	Armv7-M

TABLE I: Comparison between Cortex-M0+ and Cortex-M3

The data shown in the table can be easily found on the arm developer website or in the technical manuals of the two processors [40], [41]. CoreMark/MHz and the DMIPS/MHz range are two popular benchmarks commonly used to measure the computing performance of Central Processing Units (CPUs). Both processors are based on a 32-bit architecture, but Cortex-M0+ is less performant, consumes less dynamic power and is smaller than Cortex-M3. Cortex M0+ Instruction Set Architecture (ISA) is the Armv6-M, which is a subset of the Armv7-M ISA. Armv6-M is upwardly compatible with Armv7-M, which means "that application level and system level software developed for ARMv6-M can execute unmodified on ARMv7-M" [42]. Therefore, it is clear that any algorithm that can be computed by a Cortex-M0+ can also be

computed by a Cortex-M3 processor.

Secondly, it is important to consider the amount of free memory that can be occupied by any security scheme. As previously stated, ICDs have a very limited amount of free RAM that can be used for computation, which generally ranges from 32 to 256 KB. Arduino MKR is equipped with 32 KB of free RAM which is perfectly in line with the lower bound of the oldest ICDs. Therefore, any algorithm that requires to store too many complex data structures will cause our board to run out of memory, causing a stack crash, revealing us that it is not suitable for improving ICD security.

Another important aspect to consider is that MKR board can be powered in two different ways. Its USB port can be used to supply power. In this way we were able to rapidly verify and correct bugs and problems related with the code we wrote, just by loading it from our PCs while powering the board. The second way to power the board is using its LiPo (Lithium Polymer) battery connector. This connector was particularly useful to measure the battery consumption of our protocol, when, instead of supplying power from our PCs, we used a LiPo battery.

MKR boards can communicate with other devices using Bluetooth Low Energy (BLE) thanks to his connectivity module, the NINA-W10, which operates in the ISM band range (between 2.4 and 2.4835 GHz). Thanks to this additional feature we were able to make two MKR boards exchange messages, one impersonating the ICD, while the other behaving as the external device.

One last important factor that influenced our choice is that Arduino boards are extremely easy

to program thanks to the Arduino IDE which allowed us to load "sketches", which are Arduino programs written in C code, directly from our PC to the board.

4.3 ECDH key exchange with Curve25519

The last step in our attempt to improve security on ICDs was to design and implement a lightweight communication protocol able to protect these devices against the most common attacks while taking into account their context and limitations. Many solutions have been proposed in the literature, all of them with advantages and drawbacks. Two main approaches are currently being proposed: external device-based and biometric-based solutions. After evaluating all the problems that may incur with these approaches, we decided to focus on the feasibility of a third one that has not been considered by researchers: public key cryptography with elliptic curves. The main reason why this approach could not have been attempted is that public key cryptographic schemes are generally too expensive in terms of computational power and energy consumption and, therefore, not suitable for ICDs.

ECDH is a key agreement protocol based on ECC, which has the advantage, over all non-ECC public key cryptographic schemes, that "one can use an elliptic curve group that is smaller in size while maintaining the same level of security" [43]. This means that, using ECC, the key size will be reduced and faster implementations are feasible, which, in turn, results in possible implementations on resource-constrained devices.

Another important aspect to notice is that ECC is based on the choice of a specific elliptic curve, which is composed by a set of domain parameters. Generating this set is complex and therefore, the communicating parties should publicly agree on a well-known set of parameters

that will describe a specific curve. Consequently, the choice of the curve to be used to perform ECC can really influence the performance in the computation of the shared secret. Among all most popular curves, Curve25519 has captured our interest. Curve25519 was presented in [44] by D. J. Bernstein and presents several characteristics that are advantageous for our research topic.

Firstly, many design choices in conceiving this curve were done to improve performance and to achieve a higher speed with respect to other high-security elliptic curves. This curve offers a high security level which is comparable to performing a brute force approach on a 128-bit cypher. Another important aspect to consider is that the size of generated public keys is of only 32 bytes, which means that the size or the number of exchanged messages between the communicating parties will be reduced, saving power. Lastly, the design of this curve and the implementation of the ECC function using it are publicly available and, therefore, is present in many cryptographic libraries. For all these reasons we decided to design a protocol using Curve25519 as a base for our ECDH key exchange algorithm and to realize a proof-of-concept which demonstrates its feasibility in the domain of ICDs.

Obviously once the ECDH key exchange process is completed both the ICD and the external device possess a common key that can be used with any symmetric cryptographic primitive to encrypt/decrypt any message exchanged.

As in most public-key cryptographic solutions a Public Key Infrastructure (PKI) will be necessary to guarantee that the public keys received by the ICD from external entities belong to authorized parties and have not been modified or replaced by malicious entities. Thanks to

a Certification Authority (CA), which is a trusted third party, hospitals and caregivers can obtain certificates that, once verified by the ICD, can guarantee access to the implanted device and start the ECDH-based communication protocol. This new approach could guarantee access during emergency conditions to foreign hospitals to any ICD without any of the limitations of the other three approaches currently followed by researchers in this field.

CHAPTER 5

IMPLEMENTATION

In this chapter we will describe in depth our ECDH-based communication protocol on which we will test the implementation feasibility in current ICDs. Firstly, we will present the two main agents of this communication and describe the hardware and software details that allow the message exchange between them. Lastly, we will describe all the logical steps of our protocol and explain how we have implemented them.

5.1 Main actors and Hardware setup

Since the goal of our protocol is simulating the interaction between an ICD and an external device we identified two main roles. The first actor will be referred from now on as *central* and conceptually represents the external device which interacts with our second actor, the *peripheral*, which simulates an ICD.

Both actors are represented in hardware by an Arduino MKR WiFi 1010 board. We could have opted for a much more powerful device representing the programmer/base station but we decided to keep things as intuitive as possible and use equivalent hardware configurations for both actors.

We used Arduino IDE to write both the sketches, which are Arduino programs written in C language, of central and peripheral. Each MKR board was connected to a different PC thanks to two USB 2.0 A-male to micro B cables. This connection with the PCs is necessary both to

Figure 6: Two Arduino MKR Wifi 1010 boards connected to different PCs and communicating with each other

supply power and to upload the sketches we wrote using Arduino IDE in the boards. A picture that illustrates our hardware setup used to simulate the interaction between an ICD and an external device is shown in Figure 6.

An important feature of Arduino IDE is its serial monitor, which allowed us to see on our PCs the real-time execution of our sketches uploaded into the boards, helping us debug our code

and verify the correct functioning of our designed protocol. In order to print text or variable values during the execution of a sketch we used "Serial.print()" function. The main reason we used two PCs instead of only one was because, in this way, we could see the serial monitors of both central and peripheral during a single protocol execution.

5.2 BLE message exchange

In order to make central and peripheral exchange messages with each other, we used their connectivity module, the NINA-W10, to establish a BLE wireless communication between the boards. We also had to include ArduinoBLE public library in both sketches of central and peripheral. BLE communication is based on the Generic Attribute Profile (GATT). The two main roles defined in GATT are the *client* and the *server*. The client, which corresponds to our central device, starts the communication sending one request to the server, which, in turn, is responsible for holding useful data and sending a response back. Data exchanged during a BLE communication is generally stored inside of *characteristics*. A collection of logically-related characteristics is grouped into a *service*. Both services and characteristics are represented by a 16 byte Universal Unique Identifier (UUID).

In our implementation we defined only one service for each device and different characteristics representing the public keys and messages exchanged between central and peripheral. We assumed that both central and peripheral know each other service and characteristics UUIDs. Once a connection between central and peripheral has been established, they both can read and modify each others characteristics and, therefore, exchange data or modify some parameters (similarly to what an ICD and a programmer would actually do).

5.3 Cryptographic primitives and main functions

In order to implement our communication protocol it has been necessary to search for an implementation of all the cryptographic primitives required to perform its main steps: public-private key pair generation, shared key derivation, message encryption and decryption.

As stated in paragraph 4.3, using ECDH with Curve25519 would grant us many advantages in terms of performance and battery consumption, making public key cryptography achievable also for resource-constrained devices. Therefore, we started searching for open source cryptographic libraries that implement an elliptic curve based key agreement protocol using Curve25519 as domain parameter.

One of the most popular cryptographic libraries optimized for Arduino devices is the Arduino Cryptography Library developed by Rhys Weatherley [45]. This library contains a vast collection of algorithms to perform cryptographic operations and allowed us to perform all the main steps of our communication protocol. In particular, we included Curve25519 class to perform both public-private key generation and shared key derivation steps, and AES256 class to perform message encryption and decryption. Curve25519 class was implemented and tested following RFC 7748 [46] specifications and is characterized by two main functions called *dh1* and *dh2*.

The former is responsible for generating a random private key using an entropy source and deriving from it a public key to be shared with other parties. Both keys will be stored inside of 32 bytes buffers that are passed as parameters to dh1 function. An ICD could obviously use ECG measurements stored inside its memory to produce enough entropy to generate the private key.

Dh2 function is in charge of deriving a shared secret key among two parties using the private key of one actor and the public key of the other one. In order to do so, it is necessary for both actors to exchange their dh1-generated public keys. The 32 byte shared key will be identical for both actors and, therefore, can be used with any symmetric block cipher to perform message encryption or decryption.

The other class of Arduino Cryptography Library that we used in our protocol is AES256 and is responsible for the message confidentiality of our protocol. AES256 class was tested using FIPS 197 test vectors in order to verify its correct functioning [47]. It is composed of three main functions: *setKey*, *encryptBlock* and *decryptBlock*. The first one sets a 32 byte key in the block cipher, this function must be called before any encryption or decryption procedure. EncryptBlock and decryptBlock perform message encryption and decryption and both take two equal-length buffers as parameters, one for the plaintext, the other one for the ciphertext.

5.4 Protocol description

Since all the the main components of our research have been analyzed, we can now present our designed ECDH-based communication protocol. Figure 7 illustrates all the steps that must be executed by a central and a peripheral device in order to establish an encrypted message exchange.

Firstly, the programmer (central) must possess or obtain a valid certificate from a CA in order to guarantee that the entity which is trying to connect to the ICD (peripheral) is legitimate. Peripheral will verify the validity of the received certificate and, if the certificate belongs to a trusted authority it will proceed in the communication process. Peripheral will now generate

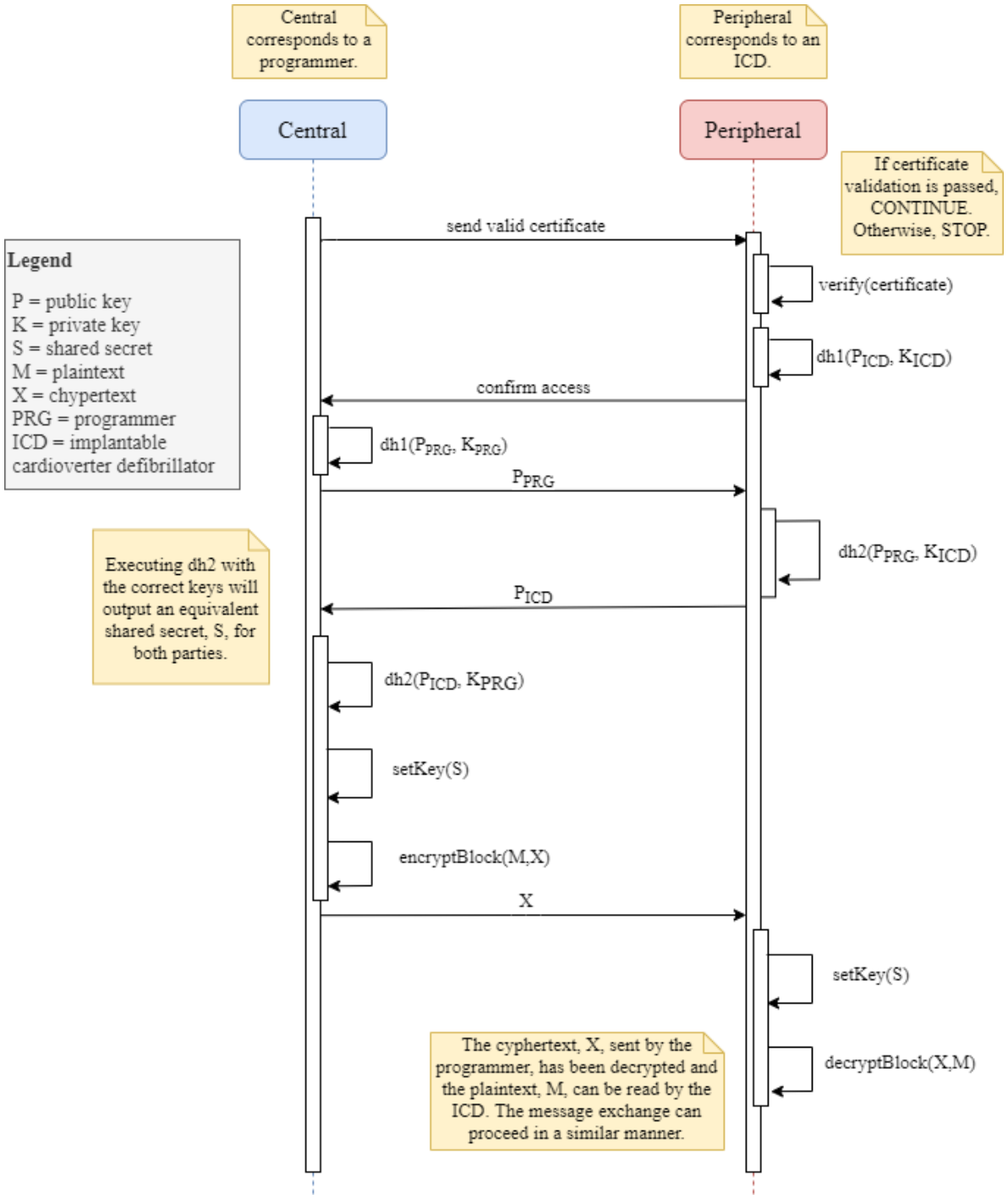

Figure 7: Sequence diagram illustrating all the main steps of the designed protocol

a new public and private key pair, P_{ICD} and K_{ICD}, using dh1 function. After the key generation process it will activate its wireless module completely (both MICS band transmitter and receiver) and send an access confirmation to the central device.

Consequently, central will perform its own key generation process calling dh1 function and obtaining P_{PRG} and K_{PRG}. Then, it will send its public key to peripheral, which will now be able to calculate the shared secret, S. In fact, Peripheral will call dh2 passing its private key and central public key as parameters. After the shared secret derivation process is completed peripheral will send its public key to central. In this way central will be able to obtain the same shared secret previously computed by peripheral.

Now that both the actors share a secret key the can use it to exchange messages in a secure manner using any symmetric block cipher, in this case, AES with a 32 byte key. Firstly, central will call setKey function passing the shared key as a parameter to it. Consequently, it will call encryptBlock function passing a plaintext M and an empty buffer, of the same size of M, as parameters. The plaintext will be encrypted, producing a ciphertext X, and stored inside the previously empty buffer. X will be sent to peripheral, which will set S as the key of the symmetric block cipher and call decryptBlock function, obtaining the plaintext M. From this point on, encryptBlock and decryptblock functions can be called both by central and peripheral to continue the secure message exchange. This protocol was designed to simulate the interaction that would happen between a hospital programmer and an ICD if they apply an ECDH-based communication exchange.

We have implemented all the most important steps of this protocol on the previously described

hardware configuration as a proof-of-concept of our reasonings and to demonstrate its feasibility in the domain of ICDs.

CHAPTER 6

EXPERIMENT

After implementing all main steps of our protocol on two Arduino MKR boards and verifying their correct functioning, we proceeded in evaluating its performance following three main metrics: *execution time*, *RAM usage* and *battery consumption*. In this chapter, we will describe in detail the methodology that we followed to perform each measurement, explain the main difficulties we had to face and the solutions we came up with. Lastly, we will discuss the obtained results, considering ICD technical limitations.

6.1 Execution time

Measuring the execution time of a message exchange protocol is essential in order to understand how much time it will take a healthcare professional to establish a secure communication with a patient. Since ICDs are equipped with simple energy-efficient processors, executing computationally complex primitives would take them a superior amount of time. Obviously, during a regular follow-up visit a difference of a couple of minutes will not cause any major issue. On the contrary, during emergency situations, any lost second could have a serious impact on the final outcome of the rescue attempt. Therefore, while designing a new security scheme for ICDs, it is necessary to take into account the time overhead that its addition will cause in the communication protocol. For this reason we decided to measure the time taken by all cryptographic primitives present in our protocol.

In order to perform such measurements we used one of the functions present in Arduino library, namely *millis*. What millis does is leverage the timer/counter module present in Arduino boards to return the time (in milliseconds) that has passed since the board has been powered up. In order to quantify the execution time of a specific function we called millis twice, directly before and after the chosen function. Storing the two returned values and subtracting the former from the latter we will obtain the precise measurement we needed.

Table II shows the results we obtained measuring the time taken by executing each of the security primitives of our protocol. It also highlights the average message sending time and the total time required to complete the whole protocol with the exchange of two messages, one from central to peripheral, the other from peripheral to central.

Primitive	Execution Time (us)
public-private key pair generation	563138
shared secret derivation	562932
set symmetric block cipher key	97
message encryption	227
message decryption	426
average message sending time	27856
whole protocol	3076391

TABLE II: Execution time of all cryptographic primitives present in our protocol

As can be seen, the most time-taking operations are the ones related to the ECDH key exchange protocol, which, despite being the slowest ones, take only a little more than half a second to complete their execution on a computationally limited device. It is also worth noticing that these two primitives will be executed only once by the ICD, at the beginning of the communication session, and therefore, will not introduce any further delays during the message exchange.

On the contrary, encryption and decryption procedures will be repeated for any new message but, since they both require much less than a millisecond to execute, they will introduce a

negligible delay. Overall, the execution of the whole ECDH shared key exchange and the encryption\decryption of two messages took us a little more than 3 seconds using two MKR Wifi 1010 boards communicating with BLE technology.

6.2 RAM usage

ICDs not only have a limited computational capacity, they also have a small amount of free RAM space that can be used for communication and therapy-delivery purposes. Occupying too much RAM memory will lead the heap and the stack to collide causing and indefinite behaviour in the ICD. Therefore, in order to increase security on such devices without having a negative impact on their main functionalities, it is necessary to avoid storing complex data structures and using security primitives that are not too memory demanding.

Precisely measuring the RAM used by every primitive of our protocol was more complex than measuring the execution time. Firstly, Arduino does not have a simple way to measure the RAM usage during the whole runtime execution of a program. In order to perform such measurements we decided to use MemoryFree library that we downloaded from the Arduino Playground website. MemoryFree library implements *freeMemory* function which, when called, calculates the amount of free space between the heap and the stack during the program execution in bytes.

In order to understand how much RAM has been used by our proof-of-concept implementation, we had to call freeMemory function multiple times during the sketch execution. This helped us to understand how variables were allocated in memory and which functions, inside our cryptographic primitives, required the greatest amount of RAM bytes. In particular, we had

to add a freeMemory call at the beginning of all functions of the cryptographic library we used to perform all the steps of our protocol. After printing the values returned by freeMemory on Arduino serial monitor, we had to analyze these results by subtracting the lowest value of free RAM obtained inside of a cryptographic primitive from the value obtained before beginning the execution of said primitive.

Following this approach we were able to determine precisely the RAM usage of all the main steps of our protocol. These results are summarized in Table III.

Primitive	RAM Usage (bytes)
public-private key pair generation	752
shared secret derivation	752
set symmetric block cipher key	64
message encryption	88
message decryption	136
global variables	2128
BLE communication	2240
whole protocol	5120

TABLE III: RAM usage distribution in our protocol

The whole protocol occupied exactly 5 kilobytes (KB) of RAM over the 29067 bytes available in our arduino board. It is important to notice that most of it was used to store the global and BLE communication related variables. On the contrary, ECDH security primitives occupied 752 bytes during execution while encryption and decryption procedures occupied a very small and almost negligible amount of RAM. Obviously, new ICDs have communication capabilities already included, therefore, in order to perform an EDCH based communication protocol the RAM usage addition for them would consist only in the bytes necessary for executing the cryptographic primitives and in the variables used to store keys and messages.

6.3 Battery consumption

The last important aspect that we decided to consider to evaluate our proof of concept is the battery consumption of our protocol. As previously stated, ICDs are devices meant to stay inside the patient body for many years before being replaced. In order to do so, they are equipped with energy-efficient hardware and they activate certain modules only when necessary. Obviously, performing high energy demanding operations for such devices is not advisable.
For this reason, we decided to measure the instantaneous current detected when performing the most important steps of our protocol. In doing so, we had to modify our hardware configuration and power up our MKR boards using their LiPo battery connector. So far, we have connected each Arduino board to a different PC. This was done both to read their info from the serial monitor, both to power up the boards from their USB port. In order to measure the battery consumption during our protocol primitives execution we had to add the following hardware components: *INA219 current/power monitor*, *ICR18650 Li-Ion battery* and *Arduino*

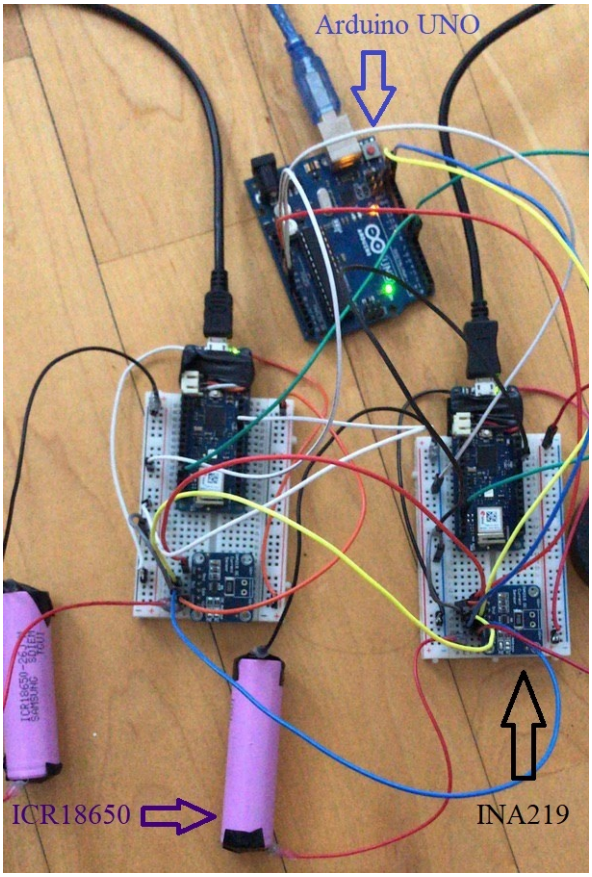

Figure 8: Hardware setup that we used to obtain the instantaneous current measurements

Uno board. Figure 8 illustrates all these elements connected together with the two Arduino MKR boards.

INA219 is a current/power monitor that we connected in series with our Arduino MKR board in order to measure as accurately as possible the instantaneous current provided from the battery to our MKR board. Arduino Uno board is the only component which is connected to a PC while Arduino MKR board executes our protocol. Arduino Uno is in charge of printing on the

PC's serial monitor the measurements performed by the INA219. Uno is also responsible for catching hardware interrupts from the MKR board. These interrupts are essential to start our battery measurements exactly when we are executing our security primitives and not during other phases of our protocol. ICR18650 is the battery that we used to power up our Arduino MKR board. It is important to notice that the only component powered by the LiPo battery is the MKR board, since the Uno board is connected and alimented by a PC and the INA219 is powered by the Uno board from its 5V pin. Therefore, no additional components will consume the charge of our LiPo battery, givin us accurate results.

A schematic representation of the circuit we built to perform instantaneous current measurements on a single Arduino MKR board is shown in Figure 9. All components are connected to the same ground in order to have a common basepoint to perform our measurements. Arduino Uno board and INA219 communicate using the Inter Integrated Circuit (I2C) protocol which uses Serial Clock (SCL) and Serial Data (SDA) pins to make them interact with each other. MKR board sends hardware interrupts using its Pin 3 (configured in OUTPUT mode) to the Pin 7 (configured in INPUT mode) of the Uno board. MKR boards sets Pin 3 to HIGH when it starts computing one of the cryptographic primitives of our protocol. Pin 7 of the Uno, which is connected to MKR Pin 3, receives these interrupts and starts reading the instantaneous current values measured by the INA219. These measurements are then printed on the serial monitor of our PC, which is connected to the Arduino board.

Results of such measurements are shown in Table IV.

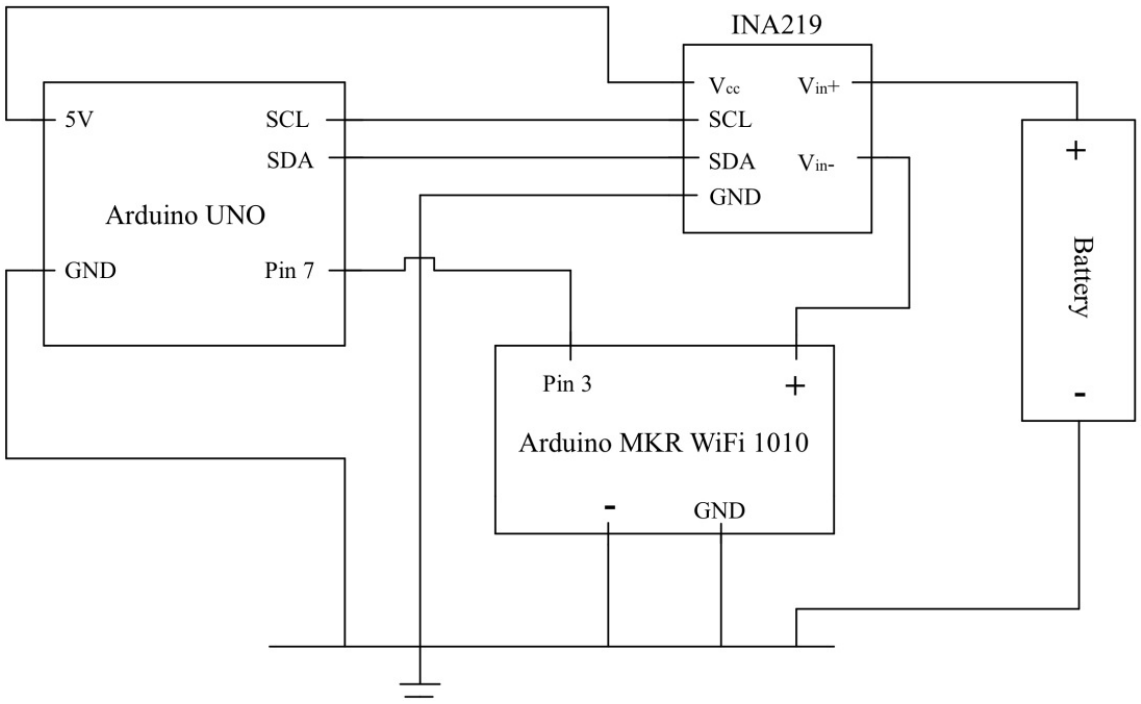

Figure 9: Schematic representation of the components used and how we connected with each other

Primitive	Battery consumption with BLE (mA)	Battery consumption without BLE (mA)
public-private key pair generation	45.65	16.34
shared secret derivation	49.89	16.38
set symmetric block cipher key	42.36	16.77
message encryption	40.45	16.77
message decryption	44.24	16.8

TABLE IV: Instantaneous current detected during the execution of all cryptographic primitives present in our protocol

We have measured the instantaneous current while performing our cryptographic operations both with and without BLE communication. This was done because we wanted to understand the proportion of the current used for executing only the security primitives compared to the total amount of the current necessary for executing both communication and computation functionalities of our protocol. As can be seen, the computation of cryptographic primitives without BLE requires only one third of the whole current necessary to execute the same primitive while performing our communication protocol. This shows that the most energy consuming task for

our proof-of-concept is handling the communication and not executing computationally complex functions.

6.4 Results discussion

The three metrics that we measured to test the performance of our ECDH-based approach for improving the security on ICDs show encouraging results. Our protocol was implemented on a resource constrained Arduino board in the attempt to simulate as best as possible all the technological limitations and the context of an ICD.

The processor we used in our hardware implementation, the ARM Cortex M0+, is less performant than ARM Cortex M3, which is what current IMDs are equipped with. Despite this, the results we obtained show that the security primitives we used to realize our communication protocol take a small amount of time to be completed. This is an important result, because in life-critical situations having the possibility to communicate with an ICD almost immediately can help the medical staff to gather precious data and to intervene promptly to save the patient.

During our research for technical data we discovered that ICDs have a very small amount of free RAM that can be used for functions not related to monitoring the patient status or performing heartbeat correcting operations. The total free RAM our Arduino MKR has is around 29 Kbytes (3 Kbytes were occupied by Arduino variables), which is comparable to amount of remaining free RAM space possessed by older ICD models. The maximum amount of bytes occupied during the execution of our protocol is around 5 Kbytes, which means that an ECDH-based communication protocol is perfectly feasible in terms of RAM usage in the ICD domain.

The last important aspect that we considered in our feasibility analysis study is the battery

consumption. Batteries and capacitors inside of ICDs occupy almost two thirds of their volume. Despite this, since they are relatively small in size, the capacity of a modern ICDs ranges from 1 to 2 ampere/hour. Our experiment showed that the most energy consuming task is maintaining active the connection with an external device rather than computing cryptographic algorithms. Performing security primitives without having an active communication with an external device required us a current of less that 20 mA. This means that, while obviously having an impact on device longevity, the amount of current required for security operations is acceptable in the ICD domain.

The experiment that we have performed shows that using ECDH with Curve25519 is an interesting new approach to protect resource-constrained devices. Therefore, we think that further research on this topic might open a new horizon in the field of ICD's wireless telemetry security.

CHAPTER 7

LIMITATIONS AND FUTURE WORK

Since our work proposes a rather unexplored research path in the domain of ICDs security, we will also briefly present the main limitations that affected our research and discuss possible future works that could further improve our approach.

One of the biggest issues that we had to face was the lack of support from ICDs manufacturers. Any attempt that we made to request information, even by proposing to sign an NDA, was declined or ignored. For this reason, we had to spend much time in searching for the most recent ICD specifications through patents and academic literature in order to realize a proof-of-concept that could simulate ICDs technical limitations. We grouped all this hard to find data in our work, so that it will be easier for other researchers to follow this path. Obviously, obtaining a sample of a current ICD and of a programmer would be the best option to evaluate the feasibility of any security solution in this context. Therefore, if in the future any manufacturer will share such devices with the research community we would recommend to re-evaluate our ECDH-based protocol on them, in order to further prove the validity of this approach.

Another important aspect to consider is that the code we used and wrote for our proof-of-concept implementation is in C language. Rewriting the security primitives we used in Assembly language would reduce the number of clock cycles necessary to execute them. This will obviously lead to a reduction in the execution time and in the battery consumption, further increasing the feasibility of ECC-based solutions in the ICD domain.

A limitation that affects all public key cryptography based approaches is the need of a PKI and, more specifically, of a worldwide CA able to distribute certificates to all legitimate programmers. Obviously realizing such a complex infrastructure is not an easy task, but performing certificate validation would guarantee that only authenticated users will have access to ICDs in any moment, both during regular checkups and in emergencies. It will also overcome the limitations that were present in previously proposed approaches, like forcing the patient to constantly wear an external device or requiring complex and time taking operations to perform biometrics-based authentication.

CHAPTER 8

CONCLUSION

In this thesis we analyzed in depth the context and problematics regarding security on ICDs, with the aim to propose a new approach or to improve current ones. After having examined the currently proposed solutions and having understood their main drawbacks, we decided to verify the feasibility of a different approach that has been almost neglected by the research community, because considered too resource consuming and, therefore, not suitable in the domain of ICDs. In our work we wanted to show that public-key cryptography based on elliptic curves is a feasible approach for resource constrained devices.

In order to do so, we gathered technical data about ICDs. This information helped us to estimate which approaches were worth attempting to secure implantable devices and to choose an hardware setup that could simulate them. After having performed this preliminary step, we implemented an ECDH-based communication protocol on the chosen hardware setup, two Arduino MKR Wifi 1010 boards communicating through BLE technology. The choice of using Elliptic Curve Cryptography combined with one of the fastest elliptic curves known, Curve25519, was made to reduce as much as possible the resources required for the implemented protocol.

After verifying that our protocol is executable by our hardware configuration and that the encrypted messages have been correctly exchanged between the two boards, we performed measurements based on three main metrics to assess in a precise and accurate way the resources required to execute it. We have used software libraries and functions to measure the execution

time and the RAM usage, while we made modifications to our hardware setup to measure the instantaneous current consumption during the execution of our protocol. Finally we compared the obtained measurements with ICDs technical specifications and with their operational context. The results obtained are promising, the protocol execution takes a reduced amount of time, which means that it will not be problematic during emergencies, while requiring a small amount of RAM space and of battery power, characteristics that are both essential to guarantee the correct functioning of the implanted device for its whole lifetime.

Securing ICDs is not a trivial task due to all their technical limitations and the need to guarantee almost immediate access during life-critical situations. Many solutions have been proposed, all of them showed advantages and drawbacks. Our work highlights a new path which appears to be promising in granting access in all situations to authorized entities, while not draining excessively ICDs resources and overcoming other solution's limitations. Elliptic Curve Cryptography with Curve25519 is an approach that should be investigated more while trying to secure ICDs communications with the outside world.

CITED LITERATURE

1. Active implantable medical devices market by product (implantable cardioverter defibrillators (transvenous & subcutaneous), cardiac pacemaker, ventricular assist device, neurostimulator, implantable hearing devices) - global forecast to 2022. https://www.marketsandmarkets.com/Market-Reports/active-implantable-medical-devices-market-102063992.html, 2017.

2. Defibrillator market size, share and industry analysis by type (implantable cardioverter defibrillator (icd), transvenous icd, external defibrillator) by end user(hospitals & clinics, ambulatory, schools and other public places) and regional forecasts, 2019 - 2026. https://www.fortunebusinessinsights.com/industry-reports/defibrillator-market-100950, 2019.

3. Global pacemakers and implantable cardioverter defibrillators (icds) market analysis and forecasts 2017-2023. https://www.globenewswire.com/news-release/2017/06/22/1027754/0/en/Global-Pacemakers-and-Implantable-Cardioverter-Defibrillators-ICDs-Market-Analysis-and-Forecasts-2017-2023.html, 2017.

4. Marin, E., Singelée, D., Garcia, F. D., Chothia, T., Willems, R., and Preneel, B.: On the (in)security of the latest generation implantable cardiac defibrillators and how to secure them. In Proceedings of the 32nd Annual Conference on Computer Security Applications, ACSAC '16, page 226–236, New York, NY, USA, 2016. Association for Computing Machinery.

5. Welsenes, G., Borleffs, C. J. W., Rees, J., Atary, J., Thijssen, J., Wall, E., and Schalij, M.: Improvements in 25 years of implantable cardioverter defibrillator therapy. Netherlands heart journal : monthly journal of the Netherlands Society of Cardiology and the Netherlands Heart Foundation, 19:24–30, 01 2011.

6. Islam, M. N. and Yuce, M. R.: Review of medical implant communication system (mics) band and network. ICT Express, 2(4):188 – 194, 2016. Special Issue on Emerging Technologies for Medical Diagnostics.

7. Technical manual lumax 640 / 740 promri. https://manuals.biotronik.com/emanuals-professionals/?country=GB&product=Icd/Lumax/Lumax640_740_ProMri, 2015.

CITED LITERATURE (continued)

8. Visia af reference manual. http://manuals.medtronic.com/content/dam/emanuals-/crdm/CONTRIB_235960.pdf, 2016.

9. Technical manual eluna hf(-t). https://manuals.biotronik.com/emanuals-professionals/?country=US&product=Pacemaker/Eluna/ElunaHF_T_US, 2017.

10. Abrahamson, H.: Implantable medical device adapted for radio frequency telemetry with frequency hopping. Technical Report US9704385B2, St Jude Medical AB, 07 2017.

11. Mikaela, N., Paul, B., Nader, A., Katia, D., and José, F.: Risk assessment of cyber attacks on telemetry enabled cardiac implantable electronic devices (cied). arXiv preprint arXiv:1904.11908, 2019.

12. Visia af vr model dvab1d4 product specifications. https://www.medtronic.com/us-en/healthcare-professionals/products/cardiac-rhythm/icd-systems/visia-af.html, 2016.

13. Rostami, M., Juels, A., and Koushanfar, F.: Heart-to-heart (h2h): Authentication for implanted medical devices. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, CCS '13, page 1099–1112, New York, NY, USA, 2013. Association for Computing Machinery.

14. Camara, C., Peris-Lopez, P., and Tapiador, J. E.: Security and privacy issues in implantable medical devices: A comprehensive survey. Journal of Biomedical Informatics, 55:272 – 289, 2015.

15. Israel, C. W. and Barold, S. S.: Pacemaker systems as implantable cardiac rhythm monitors. American Journal of Cardiology, 88(4):442–445, 2001.

16. Giuseppe, B., Merino, J., Wright, D., Gadler, F., Schaer, B., and Landolina, M.: Battery longevity of implantable cardioverter-defibrillators and cardiac resynchronization therapy defibrillators: Technical, clinical and economic aspects. an expert review paper from ehra. EP Europace, 20, 05 2018.

17. Autogen el icd, dynagen el icd, dynagen mini icd, inogen el icd, inogen mini icd, origen el icd, origen mini icd, incepta icd, energen icd, punctua icd, teligen 100 icd - reference guide. https://www.bostonscientific.com/content/dam/Manuals/us/current-rev-en/359407-004_multi_RG_en-USA_S.pdf, 2019.

CITED LITERATURE (continued)

18. Landolina, M., Curnis, A., Morani, G., Vado, A., Ammendola, E., D'Onofrio, A., Stabile, G., Crosato, M., Petracci, B., Ceriotti, C., Bontempi, L., Morosato, M., Ballari, G., and Gasparini, M.: Longevity of implantable cardioverter-defibrillators for cardiac resynchronization therapy in current clinical practice: An analysis according to influencing factors, device generation, and manufacturer. Europace, 17, 05 2015.

19. Ebert, M., Lyu, S.-P. ., Rise, M., and Wolf, M.: 4 - biomaterials for pacemakers, defibrillators and neurostimulators. In Biomaterials for Artificial Organs, eds. M. Lysaght and T. J. Webster, Woodhead Publishing Series in Biomaterials, pages 81 – 112. Woodhead Publishing, 2011.

20. Halperin, D., Heydt-Benjamin, T. S., Ransford, B., Clark, S. S., Defend, B., Morgan, W., Fu, K., Kohno, T., and Maisel, W. H.: Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In 2008 IEEE Symposium on Security and Privacy (sp 2008), pages 129–142, 2008.

21. Marín Fàbregas, E.: Security and privacy of implantable medical devices. 2018.

22. Chunxiao Li, Raghunathan, A., and Jha, N. K.: Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In 2011 IEEE 13th International Conference on e-Health Networking, Applications and Services, pages 150–156, 2011.

23. Marin, E., Singelée, D., Yang, B., Verbauwhede, I., and Preneel, B.: On the feasibility of cryptography for a wireless insulin pump system. pages 113–120, 03 2016.

24. Marin, E., Singelée, D., Yang, B., Volski, V., Vandenbosch, G. A. E., Nuttin, B., and Preneel, B.: Securing wireless neurostimulators. In Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy, CODASPY '18, page 287–298, New York, NY, USA, 2018. Association for Computing Machinery.

25. Kim, Y., Lee, W. S., Raghunathan, V., Jha, N. K., and Raghunathan, A.: Vibration-based secure side channel for medical devices. In 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pages 1–6. IEEE, 2015.

26. Rasmussen, K. B., Castelluccia, C., Heydt-Benjamin, T. S., and Capkun, S.: Proximity-based access control for implantable medical devices. In Proceedings of the 16th ACM conference on Computer and communications security, pages 410–419, 2009.

CITED LITERATURE (continued)

27. Denning, T., Fu, K., and Kohno, T.: Absence makes the heart grow fonder: New directions for implantable medical device security. 01 2008.

28. Zhang, M., Raghunathan, A., and Jha, N.: Medmon: Securing medical devices through wireless monitoring and anomaly detection. IEEE transactions on biomedical circuits and systems, 7:871–81, 12 2013.

29. Gollakota, S., Hassanieh, H., Ransford, B., Katabi, D., and Fu, K.: They can hear your heartbeats: Non-invasive security for implantable medical devices. volume 41, pages 2–13, 08 2011.

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

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

32. Poon, C. C. Y., Yuan-Ting Zhang, and Shu-Di Bao: A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health. IEEE Communications Magazine, 44(4):73–81, 2006.

33. Venkatasubramanian, K., Banerjee, A., and Gupta, S.: Pska: Usable and secure key agreement scheme for body area networks. IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society, 14:60–8, 12 2009.

34. Hu, C., Cheng, X., Zhang, F., Wu, D., Liao, X., and Chen, D.: Opfka: Secure and efficient ordered-physiological-feature-based key agreement for wireless body area networks. pages 2274–2282, 05 2013.

35. Ortiz-Martin, L., Picazo-Sanchez, P., Peris-Lopez, P., Tapiador, J., and Schneider, G.: Feasibility analysis of inter-pulse intervals based solutions for cryptographic token generation by two electrocardiogram sensors. Future Generation Computer Systems, 96:283 – 296, 2019.

36. Peter, S., Pratap Reddy, B., Momtaz, F., and Givargis, T.: Design of secure ecg-based biometric authentication in body area sensor networks. Sensors, 16(4):570, 2016.

CITED LITERATURE (continued)

37. Venkatesan, K. M. and Martin Leo Manickam, J.: Ecg-signal based secret key generation (eskg) scheme for wban and hardware implementation. Wireless Personal Communications, 106:1–16, 09 2018.

38. Belkhouja, T., Du, X., Mohamed, A., Al-Ali, A. K., and Guizani, M.: Biometric-based authentication scheme for implantable medical devices during emergency situations. Future Generation Computer Systems, 98:109–119, 2019.

39. Sam d21 family datasheet. http://ww1.microchip.com/downloads/en/DeviceDoc-/SAM_D21_DA1_Family_Data%20Sheet_DS40001882E.pdf, 2018.

40. Cortex-m0+ technical reference manual documentation. https://developer.arm.com/docs/ddi0484/c, 2012.

41. Cortex-m3 technical reference manual documentation. https://developer.arm.com/docs/ddi0337/h, 2010.

42. Armv6-m architecture reference manual. https://developer.arm.com/docs/ddi0419/c/armv6-m-architecture-reference-manual, 2010.

43. Koblitz, N., Menezes, A., and Vanstone, S.: The state of elliptic curve cryptography. Designs, codes and cryptography, 19(2-3):173–193, 2000.

44. Bernstein, D. J.: Curve25519: New diffie-hellman speed records. In Public Key Cryptography - PKC 2006, eds. M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, pages 207–228, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

45. Weatherley, R.: Arduino cryptography library. Source code, available online at http://github. com/rweather/arduinolibs, 2018.

46. Turner, S., Langley, A., and Hamburg, M.: Elliptic curves for security. Technical report, IETF RFC 7748, January, 2016.

47. FIPS, P.: 197, advanced encryption standard (aes), national institute of standards and technology, us department of commerce, november 2001. Link in: http://csrc. nist. gov/publications/fips/fips197/fips-197. pdf, 2009.

VITA

NAME	Filippo Rezzonico
EDUCATION	
	Master of Science in Computer Science, University of Illinois at Chicago, USA
	Master of Science in Computer Science and Engineering, Jul 2020, Polytechnic of Milan, Italy
	Bachelor's Degree in Computer Science and Engineering, Sep 2017, Polytechnic of Milan, Italy (110/110)
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
SCHOLARSHIPS	
Spring 2019	Graduate hourly Assistantship (GA) position (20 hours/week) with monthly stipend
TECHNICAL SKILLS	
Languages	Expert in C++, competent in Java, Javascript and SQL, basic knowledge of Python
Software	Eclipse, CLion, IntelliJ, Android Studio, Pycharm, Linux, UML (StarUML)

WORK EXPERIENCE AND PROJECTS

2020-Present	Full stack developer Specialist at Mia Platform
2019	Graduate hourly Assistant at UIC: Researcher for professor Venkat Venkatakrishnan. I worked on developing an automatic vulnerability detector and exploit generator for C code programs.
	Android Projects: 5 projects of increasing difficulty to explore the main concepts and components of Android Mobile Applications. Particular focus on services, fragments, broadcast receivers and applications with background multiple threads.

VITA (continued)

	Object Oriented languages Projects: 2 projects in C++ to achieve a deep knowledge of Object Oriented languages concepts.
2018	Data Mining Project: Project done in collaboration with Bip Company in Milan. Implementation of a working forecasting model for one of Europe most important retailers using machine learning and data mining techniques learned during lectures. Main Tasks: perform data exploration, feature selection, algorithm parameter tuning using famous Python dedicated libraries on a given dataframe. Classified 4^{th} using Random Forest with an accuracy of 95% on the test set.
	Peak Crowdsourcer: Implementation of both back-end and front-end of a crowdsourcing web application, using Java, html, javascript and css. Different types of users (managers and workers) that should connect and interact with the system simultaneously had to be considered. Main tasks: create, manage and delete crowdsourcing campaigns based on recognizing mountain images and draw their outlines, store them in a SQL relational database.
2017	Sweet Crash: Back-end implementation in C++ of a modified version of the popular mobile game Candy Crush. Main Task: include the main features of the mobile game, apply concepts of object orientation, recursion and memory allocation.