POLITECNICO DI MILANO

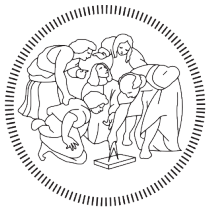
Facoltà di Ingegneria

Scuola di Ingegneria Industriale e dell'Informazione

Dipartimento di Elettronica, Informazione e Bioingegneria

Master of Science in

Computer Science and Engineering



POLITECNICO
MILANO 1863

A feasibility analysis of Asymmetric Key Distribution System for Implantable Cardioverter Defibrillators

Advisor:

PROFESSOR STEFANO ZANERO

Co-advisors:

STEFANO LONGARI

MICHELE CARMINATI PH.D.

Master Graduation Thesis by:

CAMILLA DOTTINO

Student Id n. 905249

FILIPPO REZZONICO

Student Id n. 898956

Academic Year 2019-2020

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AE Authenticated Encryption

AEAD Authenticated Encryption with Associated Data

AES Advanced Encryption Standard

BLE Bluetooth Low Energy

CA Certification Authority

CPU Central Processing Unit

DES Data Encryption Standard

DH Diffie–Hellman

DoS Denial of Service

DSA Digital Signature Algorithm

ECC Elliptic Curve Cryptography

ECDH Elliptic-curve Diffie–Hellman

ECDSA Elliptic Curve Digital Signature Algorithm

ECG Electrocardiogram

GATT Generic Attribute Profile

I2C Inter Integrated Circuit

ICD Implantable Cardioverter Defibrillator

IMD Implantable Medical Device

ISA Instruction Set Architecture

ISM Industrial, Scientific and Medical

LiPo Lithium Polymer

MICS Medical Implant Communication Service

MITM Man-In-The-Middle

NIST National Institute of Standards and Technology

NSA National Security Agency

PC Personal Computer

PKI Public Key Infrastructure

RAM Random Access Memory

RF Radio Frequency

ROM Read Only Memory

SCL Serial Clock

SDA Serial Data

UUID Universal Unique Identifier

ABSTRACT

Implantable Medical Device (IMD)s greatly evolved during the last decades. Technological advances made them smaller and more comfortable while increasing their capabilities and functionalities. In particular, among all most advanced types of IMDs, Implantable Cardioverter Defibrillator (ICD)s are able to deliver shocks to restore a normal hearthbeat.

One of the most recent upgrades is the long-range telemetry module, that allows IMDs to communicate wirelessly with external devices. This addition gives several advantages both to patients and caregivers but has also an important drawback: an increased attack surface.

Recent studies have demonstrated that ICD's proprietary communication protocols, based on this wireless telemetry, are vulnerable to a wide range of attacks [1], [2], [3]. Therefore, it is possible for an attacker to gain control of an ICD, leading to serious consequences for patient's health. Researchers focused their efforts on enhancing the security of this telemetry, but they had also to consider the limitations introduced by the ICD context: the importance to balance security with accessibility and the necessity to design a lightweight solution that considers the limited resources of an ICD.

In this work, we have performed a feasibility analysis on an approach that has been ignored by the research community: a key distribution system based on asymmetric cryptography. In particular, we have shown that this approach satisfies the requirements imposed by the ICD domain while granting an adequate level of security against most common attacks on ICDs. To further demonstrate the feasibility of this approach, we have implemented a communication protocol based on Elliptic Curve Cryptography (ECC) on an Arduino board that simulates the technological limitations of an ICD.

In our experimental evaluation phase, we have measured the performances of our communication protocol and compared them with the ones of the algorithms of the National Institute of Standards and Technology (NIST) call for lightweight cryptography, demonstrating how close our approach is to a lightweight solution specifically tailored for securing ICDs.

SOMMARIO

Un dispositivo medico è definito come "impiantabile" se è destinato ad essere inserito, totalmente o parzialmente, all'interno del corpo di un paziente, tramite intervento chirurgico o medico. Dopo l'intervento, questi dispositivi sono destinati a restare all'interno del paziente per uno specifico periodo di tempo, che dipende dal tipo di dispositivo. Esempi di dispositivi medici impiantabili IMD sono defibrillatori impiantabili ICD, pacemaker cardiaci, pompe di insulina e neurostimolatori. Le funzionalità mediche svolte da questi dispositivi sono finalizzate a migliorare la qualità della vita del portatore, facilitando l'esecuzione di alcuni compiti o mantenendo l'organismo in salute tramite la somministrazione di terapie o farmaci. Constata l'importanza, dal punto di vista medico, di tali dispositivi risulta chiaro che l'interesse della ricerca si sia rivolto sul miglioramento degli IMD su molteplici aspetti. Avanzamenti tecnologici continui hanno permesso di produrre dispositivi con una minore probabilità di rigetto da parte del paziente, più compatti e confortevoli, ma anche di aumentare la longevità di questi dispositivi all'interno del corpo del paziente.
Tra tutti i dispositivi impiantabili attivi, gli ICD sono tra i più comuni. Attualmente ne esistono diversi modelli sul mercato, alcuni esempi prodotti da Medtronic sono :[4], [5]; altri prodotti da Biotronik sono: [6], [7]. La grande diffusione è principalmente dovuto dal fatto che non si limitano a migliorare la qualità della vita del paziente, ma sono in grado intervenire in situazioni di critiche e salvare la vita al portatore del dispositivo. Inoltre, la diffusione di questi dispositivi è cresciuta, ed è destinata a crescere, di anno in anno. L'età media della popolazione mondiale si è alzata e, con essa, anche la diffusione di malattie cardiovascolari, che tramite questi dispositivi possono essere trattate e mitigate. Un ICD, oltre a svolgere le normali funzioni di

un pacemaker impiantato, quindi di regolare e rendere più stabile il battito cardiaco di un paziente, è in grado di compiere azioni salvavita più avanzate. In particolare, un ICD è in grado di monitorare costantemente l'attività cardiaca del cuore di un paziente, individuare comportamenti anomali e dannosi per la vita del paziente ed infine intervenire, se necessario, con shock elettrici. Tramite queste scosse è possibile ristabilire una normale attività cardiaca da situazioni critiche per il cuore del paziente, come un battito eccessivamente frequente oppure completamente assente.

Negli anni, gli ICD sono diventati dispositivi sempre più complessi, da un punto di vista funzionale, riducendo nel mentre le loro dimensioni e aumentando la loro longevità all'interno del corpo umano. Una delle migliorie tecnologiche recenti che riguarda gli IMD più avanzati, ICD compresi, è l'introduzione di una telemetria wireless a lungo raggio. Questa tecnologia permette ad un ICD di comunicare con dispositivi esterni al corpo del paziente senza dover richiedere uno stretto contatto con altre apparecchiature mediche. L'introduzione di questa telemetria ha portato grandi vantaggi sia ai pazienti che al personale degli ospedali. Ai pazienti è infatti permessa una frequenza ridotta di visite in ospedale annuali, in quanto questi dispositivi possono inviare dati relativi alla salute direttamente dall'abitazione del paziente all'ospedale che lo ha in cura. Il personale ospedaliero invece risparmia tempo sia durante la fase di impianto del dispositivo che nelle successive visite di controllo, riducendone anche i costi per l'ospedale. Tuttavia, l'introduzione di questa nuovo modulo di comunicazione wireless ha anche un importante aspetto negativo. Dispositivi che prima erano completamente isolati all'interno del corpo del paziente, e quindi protetti da eventuali attacchi esterni, ora sono vulnerabili a numerosi attacchi informatici. Tramite questi attacchi è possibile rubare informazioni private e sensibili, modificare le terapie distribuite da un ICD e persino manipolarne il funzionamento, fino al punto da avere effetti dannosi o letali per il paziente. Il mondo della ricerca si è quindi concentrato sull'ideazione di contromisure adeguate a tali attacchi,

di modo da garantire una comunicazione sicura tra ICD e dispositivi esterni. Tuttavia, le soluzioni proposte, per essere considerate valide, devono tenere conto del contesto in cui gli ICD operano e delle limitazioni tecnologiche di questi dispositivi. Innanzitutto, un protocollo di comunicazione fatto su misura per l'ambito degli ICD deve poter garantire un accesso rapido solo ad utenti autorizzati in qualsiasi situazione, soprattutto in caso di emergenze. Inoltre, gli ICD dispongono di risorse computazionali limitate, Random Access Memory (RAM) di dimensioni ridotte e batterie che devono durare anni senza essere ricaricate. Quindi, un protocollo di comunicazione sicuro deve tenere conto di queste risorse, senza esaurirle.

In questo lavoro ci siamo proposti di analizzare le più recenti soluzioni a questo problema, mostrandone i pregi ed i difetti, e di indagare sulla fattibilità di un approccio che è stato ignorato dalla maggior parte dei ricercatori. Questo approccio si basa sull'utilizzo di una Public Key Infrastructure (PKI) in combinazione con l'utilizzo di crittografia asimmetrica per garantire un rapido accesso, in qualsiasi situazione solo ad utenti autorizzati. Abbiamo quindi dimostrato come questo approccio soddisfi i requisiti funzionali necessari per il contesto degli ICD, garantendo un adeguato livello di sicurezza contro la maggior parte degli attacchi che possono essere effettuati su questi dispositivi. La principale limitazione di questo approccio è dovuta al fatto che la crittografia asimmetrica è più costosa, in termini di risorse, rispetto ad altri approcci, e quindi considerata non adatta per gli ICD correnti. L'obiettivo di questa tesi è quindi di dimostrare la fattibilità tecnica di questo approccio nel contesto degli ICD. Per raggiungere questo scopo, abbiamo implementato un protocollo di comunicazione basato su crittografia a curve ellittiche su una board Arduino con caratteristiche tecnologiche simili a quelle degli attuali ICD, dimostrandone quindi la sua realizzabilità. Abbiamo inoltre misurato le prestazioni di tale protocollo seguendo tre metriche principali: tempo di esecuzione, utilizzo della RAM e consumo della batteria.

In questo studio abbiamo infine comparato le prestazioni di questo

nostro protocollo con quelle degli algoritmi proposti nella call del NIST per la crittografia lightweight. Questi algoritmi, basati su crittografia simmetrica, sono stati progettati per essere il più leggeri possibili in termini di consumi di risorse e sono quindi stati studiati per dispositivi limitati tecnologicamente. Tuttavia, questi algoritmi non tengono in considerazione il contesto operativo degli ICD, e quindi non rappresentano delle valide soluzioni in questo dominio applicativo. Il risultato di questa nostra comparazione dimostra come il nostro protocollo non è distante dalle prestazioni degli algoritmi del NIST e che, a differenza di tali algoritmi, tiene in considerazione tutti i requisiti funzionali necessari per essere una soluzione nel contesto complesso in cui gli ICD operano.

1

INTRODUCTION

A medical device is defined as "Implantable" if it is either partly or totally implanted inside the patient's body. There are different devices that help patients with different diseases, from cardiac arrhythmia, diabetes or Parkinson's disease. They can be passive, as artificial joints, or active, meaning that their functioning relies on some power source different than the human body, examples include Implantable Cardioverter Defibrillator (ICD)s, Implantable cardiac pacemakers, cochlear implants, implantable insulin pumps, neurostimulators. These devices not only improve patients' quality of life, but they actually save lives. Due to the aging of the population and the spreading of unhealthy lifestyles, the number of people that live with chronic diseases and need an Implantable Medical Device (IMD) is increasing every year [8]. These devices remain inside the patient's body for several years, therefore they need to be energy efficient to avoid having to replace them often. Moreover, they possess limited resources in terms of computational capability, battery capacity and memory size because their main purpose is to deliver the treatment to the patient.

In our research we focused on ICDs. These devices are used to treat and detect ventricular fibrillation by analyzing the heart's electrical activity, when the fibrillation is detected the ICD delivers a shock to stop its activity and to make it restart itself at the correct rhythm. During the last two decades, ICD's demand has grown and is expected to grow in the following years due to the aging of the population and the increasing prevalence of cardiovascular diseases [9], [10], [11]. For these reasons, ICDs are constantly evolving to better serve their purpose. Beside the reduction in size and the increase of the battery longevity, one of the

most relevant improvements that has been recently added to ICDs is the ability to communicate with devices outside of the human body. This new functionality is generally defined as "wireless telemetry" and brought significant advantages to the patient care, while cutting down times and costs. Despite all these positive aspects, the introduction of this long-range communication brought a significant drawback: an increased attack surface. Although no real attacks on the wireless telemetry of ICDs have ever been documented, it has been shown by researchers that the proprietary communication protocols used by ICDs present little to none security mechanisms [8]. For this reason, attackers could steal sensitive data or alter the device functioning, threatening patient's life. Therefore, studying solutions to secure such devices has become a discussed topic for the research community. These solutions, of course, need to consider the limitations of this devices, like the limited memory, low computational capabilities and the need for the battery to hold as long as possible: this translates in the need of a very lightweight solution. Current proposed solutions follow two main approaches: biometric based and external device based solutions. The former, performs authentication procedures by measuring a biometric or a physiological value. The latter is based on using an external device to proxy any communication between ICDs and programmers or base stations. Although these approaches are secure in most situations they present some drawbacks that make them not really effective in the ICD domain. A different approach to this problem could be looking at lightweight cryptography algorithms, like the ones presented in the last American National Institute of Standards and Technology (NIST) call for lightweight cryptography. Despite the proposed algorithms are not specifically designed to suit ICDs' domain, they are still meant to work on resource constrained devices, so, analyzing their performance could be very useful to obtain some guidelines that should be followed to define a solution as lightweight. Considering ICDs structural and theoretical limitations, finding a way to secure their telemetry is quite challenging. Symmetric cryptography, like the one used in the NIST

algorithms, is surely a lightweight solution but, by definition, it does not consider how the secret keys should be exchanged between the communicating parties, and that is an essential requirement in the ICD context. Therefore, using asymmetric cryptography with a Public Key Infrastructure (PKI), seems to be a valid option to perform an authenticated key exchange process. To check if it could be possible to implement a solution like this on ICDs, it is necessary to perform a feasibility analysis. This means measuring its performances, checking if it protects from most of possible attacks and if it meets all the requirements given by the ICDs' domain. Moreover, it is interesting to compare the performance tests results of this approach with the baseline given by NIST algorithms to understand if it is lightweight enough. In order to perform a feasibility analysis on solutions based on asymmetric cryptography, we have designed a communication protocol based on Elliptic Curve Digital Signature Algorithm (ECDSA) and Elliptic-curve Diffie–Hellman (ECDH) and we have implemented it on an Arduino MKR WiFi 1010 board that we have also used as test bench to measure the performances of NIST lightweight algorithms. The result of the comparison between our proposed solution and the NIST algorithms shows that, even if we use asymmetric cryptography, that is supposed to be more resource demanding than the symmetric one, our results are pretty consistent with the average of the NIST algorithms ones. Moreover, our consumptions are low enough for the protocol to be performed on an ICD without influencing its functioning. In this work we:

- have analyzed the current state of the art security solutions proposed for ICDs, examining the positive and negative aspects of each approach.
- Have performed a feasibility analysis on an approach based on a public-key distribution system, considering ICD's theoretical constraints and requirements.

- Have designed and implemented an Elliptic Curve Cryptography (ECC) based communication protocol taking into account the points made in the feasibility analysis.

2

BACKGROUND

In this chapter, we will present a full description of ICD's domain. We will start from an insight on why studying these devices is relevant, then we will provide a structural description of both internal and external components. In the third section the threat model will show ICDs' vulnerabilities, the categories of attackers interested in these devices and their goals, the types of proper attacks, and how they could be performed in this domain, and which challenges could be found while trying to secure ICDs. In the fourth section, we offer a presentation of solutions from the state of the art with positive aspects and flaws. Lastly, we will present the NIST call for lightweight cryptography, that even if is not specifically designed for ICDs, present some relevant characteristics that could be useful in our research.

2.1 ICDS' BACKGROUND

Considering all types of IMDs, ICDs are capturing much of the researchers' attention lately. The reason can be found in the great importance these devices have in protecting patients' lives and in the fact that in the last two decades the demand for these devices is quite growing. As shown in Figure 2.1, the demand of defibrillators is also expected to grow in the following years. As can be seen, a growth of 50% in the demand is expected from 2018 to 2026. This is mainly due to the fact that the average age of the population is growing. This aspect leads to the increase of people suffering from cardiovascular diseases whose lives rely on this kind of devices.

The latest ICDs' models have many features, like tracking arrhythmias, monitoring the heartbeat, and performing stimulations in order to sta-

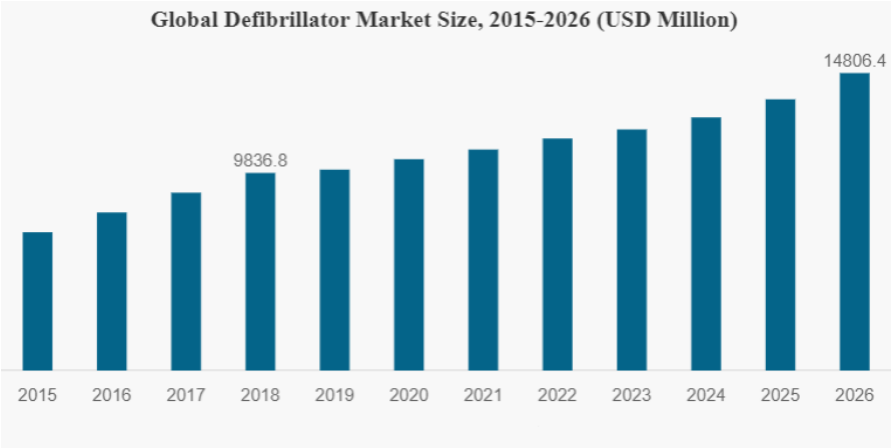


Figure 2.1: The bar graph shows the growing demand for defibrillators from 2015 to 2018. It also illustrates the expected demand from the year 2019 to 2026.

bilize the heart rhythm. Besides, all ICDs can carry out defibrillation in case of ventricular tachycardia or fibrillation that may put the patient's life in danger. This means delivering an electric current directly to the heart to restore a correct heart beating rate. In Figure 2.2 it is possible to observe an X-ray image of a patient with an ICD.

Technology helped in improving some of ICDs' characteristics extending their battery lifetime while making them smaller in size and weight. This improvement made implanting the devices easier and increased their duration without the need to replace it.

One of the most significant technological improvements is the introduction of a telemetry module that allows a long-range communication between the ICD and devices outside the human body. Before this enhancement, ICDs used to communicate with the outside world through an inductive link, that had a quite limited range of communication, less than 6 cm, and data transmission rate, around 100 kbps. To activate the communication between the ICD and any external device using this inductive link a programming head needed to be very close to the patients' body for the entire duration of the process. During the communication, the ICD and its programmer exchange messages, whether it is for data transmission or re-programming the device.

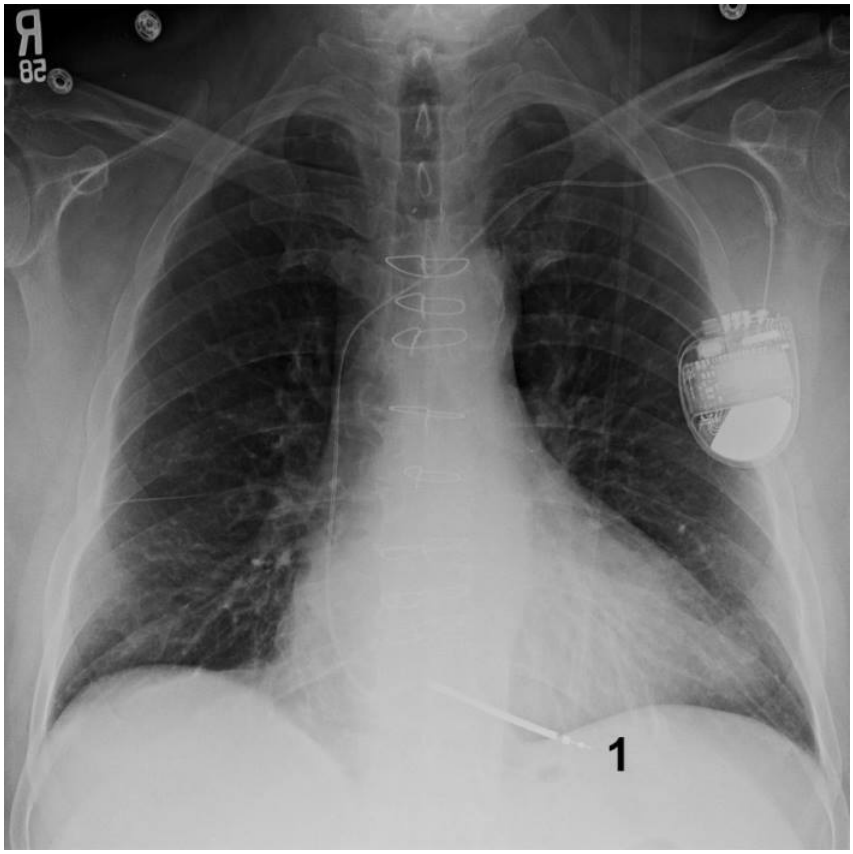


Figure 2.2: X-ray image of a patient's chest with an ICD. On the right, it can be seen the external case of the ICD. The number 1 in the picture refers the head of an ICD's lead connected to the patient's hearth.

At the moment, due to the newly introduced telemetry, communication can happen at a maximum distance of 2 to 3 meters [7], [4]. In addition, the programming head needs to be close to the patient only in the first phase of the process, when the ICD's telemetry module is activated [6]. This new telemetry feature provides plenty of positive aspects to patients' treatments.

First, the surgery to implant or replace the device got faster and easier, thanks to the fact that ICDs can oversee the patients' condition and can be configured even during the operation. Another useful improvement is that during usual visits, doctors can monitor and measure the patient's Electrocardiogram (ECG), a graph of the electrical activity of the heart, just activating the wireless telemetry avoiding placing electrodes on his chest. This not only saves time to both physician and patient but makes the person more comfortable during follow-ups. Finally, telemetry allows communicating with ICDs even while the patient is sleeping, thanks to at-home devices that then transmit the devices' data to the hospital. This way, less check-up visits are needed and doctors can monitor patients' condition at any time, possibly perceiving worrying situations.

Thus, there are two categories of devices that can communicate with ICDs from outside the human body: programmers, used by doctors, and those at-home devices, called base-stations. Programmers are the devices set in the hospitals or clinics, used by the staff, they monitor the patient's condition and can also change the ICD's configuration. Base-stations generally are set near to the patient's bed to be able to monitor him while he is sleeping. They gather data about the patient's status and transmits them to the hospital or the clinic that treats the patient using an internet connection or the phone line. Both these devices are usually equipped with a programming head that once placed on the patient's chest enables the communication with the ICD. Figure 2.3 illustrates how an ICD interacts with these external devices.

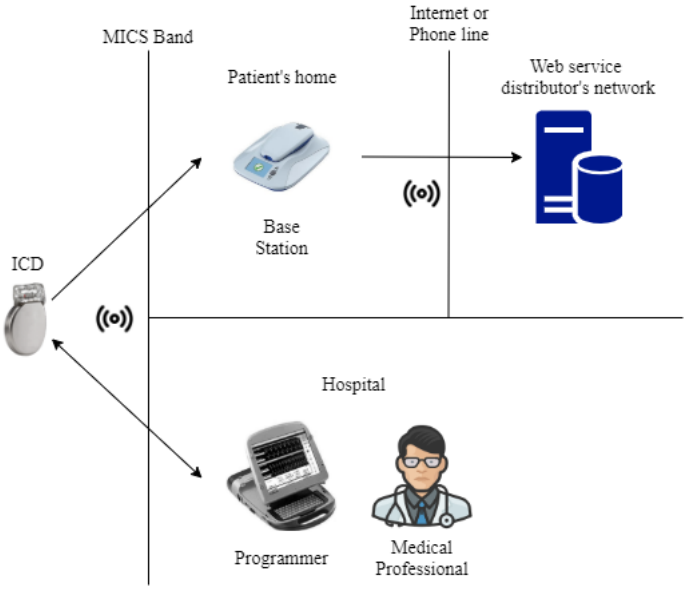


Figure 2.3: Two different communication types between an ICD and an external device are shown. Above, an ICD sends health data to a base station located at the patient's home. This data is collected and then sent to the patient's hospital. Below, an ICD interacts with a hospital programmer. The medical professional is able to change therapy settings by sending messages to the ICD.

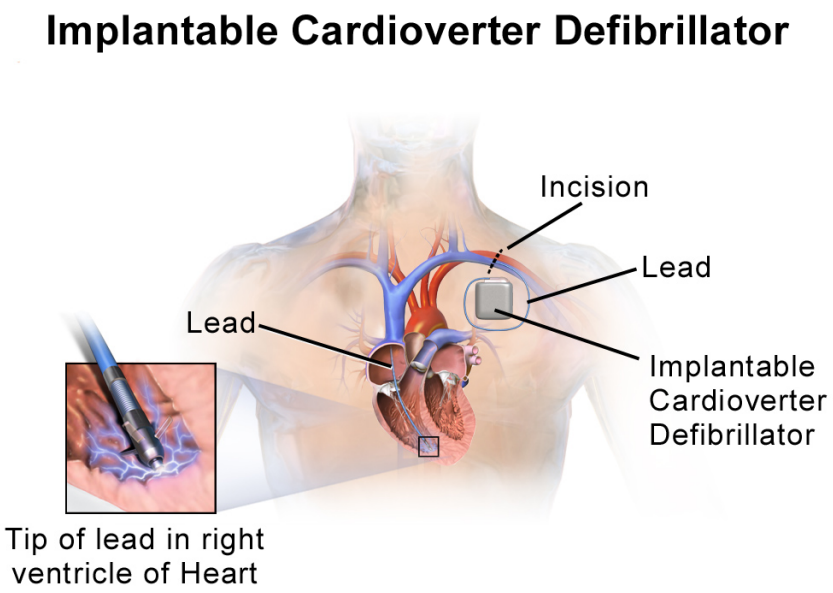


Figure 2.4: The ICD's pulse generator is implanted under the patient's chest skin, just above his heart. The lead enters the heart through an incision done on the superior vena cava. The head of the lead is able to deliver shocks to the right ventricle

2.2 ICD'S STRUCTURAL ANALYSIS

We will now give a technical explanation of ICDs' functioning and architecture with the aim to explain how ICDs work and what restrictions they have. Finding this kind of information is quite hard due to manufacturers' policy and secrecy, but significant data could be found in ICDs patents available online, scientific literature, and some manufacturers manuals available on their websites [12], [13]. The ICD's structure, illustrated in Figure 2.4, consists of a pulse generator connected through an epoxy resin connector to one or more leads that are linked to the patient's heart.

The case where the pulse generator is placed in is in touch with human tissues and is usually made of biocompatible titanium due to its resistance to corrosion and bio-inertness [7], [5]. The case is hermetically sealed and contains four main components:

- Controller module
- Memory module
- Power source module
- Telemetry module

Figure 2.5 illustrates a schema of all the components and how they are connected with each other, that will be now described in detail. The Memory module is connected with both the Controller and the Power Source modules, that are also connected with each other. The Telemetry module is linked to the Controller and to the Power Source modules too.

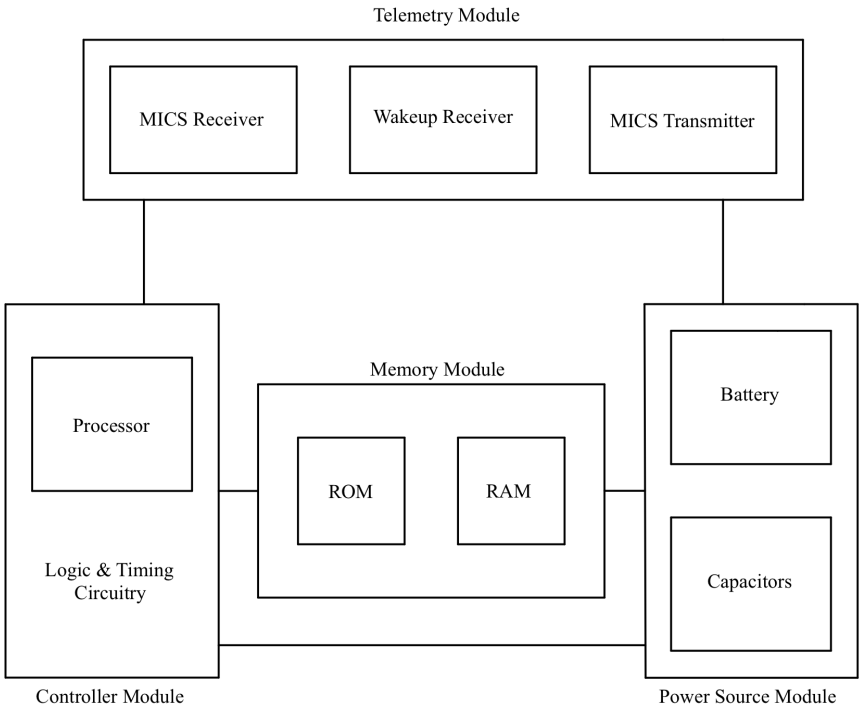


Figure 2.5: All the main components present inside the bio-compatible titanium case of an ICD. The Power source module distributes energy to all other modules when a wakeup message is received.

First, there is the controller module, which transmits control commands to the pulse generator to manage its functioning. It includes a microprocessor that handles the functioning and cooperation of the modules and runs the firmware saved in the memory module. Recent

studies show that ARM-Cortex M3 is currently used as a microcontroller in IMDs [14]. In this module, there are also the logic components that control the timing and synchronize the pulse generator.
The memory module includes both Read Only Memory (ROM) and Random Access Memory (RAM). RAM saves the patient's custom parameters, sensed ECG data, and the temporary variables used in the execution of the firmware. RAM dimension could go from 128KB to 1024KB and most of the available memory is taken from the ECG signals measured and saved by the device [15], [16], ROM on the other hand, contains the patent code from the manufacturer.
Next, the power source module includes a battery and capacitors. The two most used types of batteries are lithium-manganese dioxide or lithium-silver vanadium oxide, they provide power to all the other parts of the ICD [17]. Recent ICD models have a battery capacity that spans from 1 Ah to 2Ah for the models that last the most [18], [19]. The capacitor collects, stocks, and delivers energy to carry out defibrillation shocks when needed, this component is crucial for the ICD functioning and takes most of the space of the ICD area [20].
Finally, the telemetry module manages the communication between the ICD and the external devices previously mentioned. It usually consists in two Radio Frequency (RF) receivers and just one RF transmitter. One of the RF receivers is a broadband RF wake-up receiver and works at Industrial, Scientific and Medical (ISM) band that goes from 2.4000 to 2.4835 GHz. Otherwise, the RF wake-up receiver can be replaced with an inductive wake-up receiver, anyway this component is constantly powered up. The aim of this receiver is to pay attention to wake-up messages from other devices, when the message arrives the device's components are powered up and start working, while when no message arrives the device powers down the not necessary components saving important battery durability. As said, when the wake-up message is received, all the device's modules get powered up. This means that the other receiver and the transmitter get activated and become able to communicate with external devices using the Medical

Implant Communication Service (MICS) radio band that goes from 402 to 405 MHz [12]. This frequency band is reserved for communication of medical implants and is preferable because it has good conductivity in the human body and a long range of communication. Transmitter and receivers could be all linked to the same antenna or, if the device has more antennas, each component could be connected to a different one.

2.3 THREAT ANALYSIS

In this section, we will first analyze the vulnerabilities of ICDs. Afterwards we will examine the possible attackers interested in targeting these devices and their goals. Then, we will present the different types of attacks that may be performed on ICDs. Finally, we will discuss the challenges that have to be faced when studying how to secure these devices.

2.3.1 *Vulnerability analysis*

Since the main purpose of an ICD is to improve the patient's quality of life as long as possible without being replaced, many factors need to be considered when analyzing its functioning. The device must be small, light, made with a material suitable to stay in contact with the human tissues for many years. But the security factor is often not given in the right attention or even not considered at all. ICDs store patient's confidential data like personal and medical information. In addition, since ICDs provide treatments for chronic cardiac diseases based on the specific patients' necessities, the devices also contain the personal parameters that regulate the therapies. Therefore, someone who would want to harm a patient could try to alter the ICD's functioning. As a result, studying the security and privacy of ICDs is very important to find out a solution that protects them from possible attacks.

As mentioned in the previous chapter, the newly introduced telemetry module provides many positive aspects in the quality of life of patients that need these devices, reducing the number of follow-up visits and saving time for both doctors and patients. In contrast, the possibility of "long-range" communication has exposed ICDs to new dangers widening the attack surface. Currently, no ad-hoc security protocol has been implemented to secure ICDs and the proprietary protocols generally used by these devices have no or very little security to communicate wirelessly with external devices [1], [9]. Once this vulnerability has been discovered, many attempts to reverse-engineer the proprietary communication protocols and perform some attacks targeting data privacy and device functioning demonstrating the presence of flaws in those protocols have been conducted [15]. Moreover, as stated in [8], there is the chance to enable long-range communication without the need to place the programming head near to the patient's chest. This make it possible to send a definite set of messages that is always the same in every communication session of the same ICD.

2.3.2 *Attacker model*

Studying the possible attackers' categories and what their goals may be is fundamental to comprehend the reason why it is important to improve the security of ICD's telemetry. There are three main categories of attackers in this domain, we will discuss them from the least alarming to the most one. The categories are:

- Passive attacker
- Active attacker
- Expert active attacker

The passive attacker uses common use equipment, easily accessible, to intercept the messages between the ICD and the external device it is communicating with, his goal is to gather specific data or information

about the patient's or his treatment. Since the passive attacker cannot interfere with the data sent to the device, he is incapable of modifying parameters of the ICD and therefore of harming the patient. His threat is against message confidentiality and privacy. For this reason, we consider the passive attacker as the least dangerous one.

The active attacker has at its disposal more elaborated equipment and is capable to intercept the communication between the ICD and the external device and transmit back to the ICD some of the intercepted messages to set up a connection with it instead of the external device. This kind of attacker does not know the format of the exchanged messages during the ICD's proprietary communication protocol but can exploit the previously intercepted messages sending them again and modifying them. Active attackers can also try to drain the battery of the device trying to establish a connection multiple times, to pass themselves off as an external device, and activate treatments when they are not needed possibly harming seriously the patient. Consequently, this attacker type can affect the right functioning of the ICD and can deteriorate the treatment the patient receives up to the point where a device replacement is needed.

In conclusion, the most alarming attack category is the expert active attacker, who is capable to find out the format of the messages of the proprietary protocol. He has the ability to perform the attacks mentioned in the other categories but is also able to create new messages in order to change the device's settings or to alter its functioning. In this way, he could make it deliver wrong or unnecessary therapies endangering the patient's life.

Now that we have described the different attackers' categories, we will proceed by describing the main goals they usually try to achieve. First, the attacker's aim may steal sensitive data, not only personal and medical data but also technical data about the device, that can be used in industrial espionage or for personal interest. All the previously mentioned attackers may try to achieve this goal.

Then, an attacker may want to damage the reputation of a medical

facility or a manufacturer by deteriorating the goodness of the treatment of imposing an early substitution of the ICD. This objective could be performed by a manufacturer against a competitor, for example, since the implantable medical device market is quite competitive. All the previously mentioned attackers may want to achieve this goal.
Lastly, an attacker may want to harm the patient or event threaten his life by altering the ICD's functioning. Only the expert active attacker could perform this attack.

2.3.3 *Types of attacks*

As said in the previous section, ICD's telemetry proprietary protocols miss effective security measures, in terms of authentication, encryption, and integrity checks. This makes these devices vulnerable to many attacks, and to design a proper security protocol it is fundamental to identify the characteristics of the attacks. We will now present the possible attacks that may be conducted against ICDs.
Eavesdropping is a type of attack that aims to intercept the messages exchanged between the ICD and the external devices to steal sensitive data leveraging on the fact that the messages are not encrypted. This attack can be performed by both active and passive attackers and primarily is directed against the confidentiality of information about both the patient and the device.
Replay attacks aim to intercept the messages between an external device and the ICD and re-sending them at another time. Since the current ICD protocol does not provide any form of prevention against this kind of attack, an attacker could exploit previously sent messages to set again some settings or making the ICD re-transmit confidential data. If the attacker is of the active expert category he could also modify the message, knowing its structure, or even produce new messages performing a Spoofing attack, persuading the ICD that it is communicating with a rightful device. In this way, he could make the

ICD deliver wrong treatments harming the patient.
Man-In-The-Middle (MITM) is a kind of attack that aims to persuade both ICD and external devices that they are communicating with each other but, actually, they are both communicating with the attacker. An active expert could perform this attack hijacking a rightful communication transmitting different malicious messages to both ICD and external devices while they assume that the exchange of messages went correctly.
Denial of Service (DoS) is a type of attack that aims to prevent the device to be available, by jamming the message exchange or by draining the ICD's battery sending many connection requests in a small amount of time, forcing the patient to an early device replacement.

2.3.4 *Challenges in securing ICDs*

We presented the structural limitations of ICDs, their vulnerabilities and the attacks they are vulnerable to, but this knowledge is not sufficient to define a secure communication protocol between ICD and external devices. It is fundamental to examine the domain in which these devices operate and identify its limitations and how introducing new security solutions could affect the correct functioning of the device. First, it is necessary to find the right trade-off between security and accessibility. Access to the device should always be possible for doctors and physicians in a fast and easy way, to allow them to intervene easily in situations where timing is crucial while avoiding that any malicious attacker could access the device unauthorized. To better understand the implications of ICDs' domain it is important to analyze different possible scenarios. As an example, considering that the ICD could be accessible only with definite credentials, it could seem a good idea to keep these credentials stored in the hospital that usually treats the patient, in order to allow only authorized staff could access the device. But, if an emergency is considered, where the patient is any-

where but close to that hospital and possibly unconscious, he would not be able to give any information about his medical history or the hospital where the credentials of his ICD are stored. In this scenario, access to the ICD would be prevented and the rescue procedure would be very slowed down. For this reason, designing a communication protocol with security measures that have strong access policies would not be recommended in the ICD context, in fact, it could be useful to grant easy and fast access to handle this kind of situations.

Another factor to examine is the fact that ICDs are resource-constrained devices, as for memory, battery, and computational capacity they are structurally limited. For this reason, when studying how to implement a proper security scheme is fundamental to examine its feasibility in the ICD domain. To implant or substitute an ICD, a surgical procedure is needed, so solutions that are too energy-consuming are to avoid because they would drain the device battery and reduce the implant longevity. As previously mentioned, in section 2.2, ICDs have just a few kilobytes of available RAM, simple and low-consumption processors so they are not able to execute complex procedures and big amounts of data in a small amount of time.

Finally, one aspect that impacts the researches about ICDs' telemetry security is the difficulty for researchers to gather enough and enough recent data about the devices. The manufactures, considering the competitiveness of their market, avoid sharing specific information about their products, that being proprietary code or devices samples to prevent giving valuable information to their competitors. Another aspect is that manufacturers rely mainly on security-through-obscurity to defend their products. This means that they keep their proprietary communication protocols specifics secret, but it has been demonstrated the inefficiency of this approach against reverse-engineering techniques [2], [3]. Moreover, the difficulty or even, impossibility, to have access to ICDs samples and to their technical specifications is an important constrain for the research progress.

2.4 STATE OF THE ART

As we stated in the previous sections, there is no effective measure to protect ICDs from the many possible attacks they have been exposed to since the introduction of the telemetry module. As a result, many pieces of research have been conducted in order to identify a security mechanism that could suit the ICDs' domain and the devices' limitations. The many proposed solutions could be divided into three categories: the ones that use external devices, the ones based on biometric signals, and the ones based on cryptography techniques. The first two approaches have been tested [21], [22] but resulted to be too elaborate for the ICDs' context or still vulnerable against some attacks, the last could be a valid option but has to be better adapted to the ICD domain.

2.4.1 *External Devices*

This approach is based on the usage of a wearable external device as a "communication bridge" between the ICD and the programmer. This solution is based on the fail-open access, to realize the trade-off between security and accessibility. During his everyday life, the patient will carry the external device always with himself, and this will protect the device from malicious attackers, while, in emergencies, if the available programmer is not authorized, doctors can turn off or remove the proxy and obtain access to the ICD right away. The external device's battery should be charged periodically and could also be easily replaced if needed. Moreover, since the proxy can handle the most resource-consuming tasks of the communication protocol, it is useful to mitigate battery DoS attacks, even if these attacks can still be quite hard to combat if very intense.
One of the main drawbacks of this solution is patients compliance, meaning that the patient must wear the device at any time and has to

be careful that it is always charged and doesn't break, since if the proxy is not available or not working then the ICD is vulnerable. Moreover, an attacker could jam the messages between the programmer and the ICD, forcing the fail-open access. This situation has been examined by Gollakota et al. [23] using friendly jamming to shield the ICD from attackers. Anyway, this way out is not acceptable, since in many states jamming is illegal, even if it is effective.

2.4.2 *Biometric-based solutions*

This kind of solution is based on the utilization of the patient's biometric of physiological signals to check and ensure the authenticity of the programmer. This approach has been developed in two ways.
First, Hei et al. [24] presented an approach based on storing in the ICD one or more unchangeable biometrics, like fingerprints, iris image, hand geometry when implanting the device. When the connection with the programmer needs to be initiated, the programmer measures the same biometric that was stored in the ICD, then the two biometrics are compared and if they match then the connection is established. The drawback of this approach is that the programmer needs to be able to measure such biometrics and currently no programmer is able to do it. Moreover, since the measurement takes time, even if just one minute, during emergencies could endanger the patient's life.
Second, Poon et al. [25] presented an approach based on physiological signals that change over time, like ECG signals. This solution expects the programmer and the ICD to measure at the same time the physiological signal and then use it to encrypt and decrypt a symmetric key that is used for the communication between the devices, or even just to access the ICD, like in the case proposed by Rostami et al. [14]. This approach is secure because to measure the signal and have access to the ICD one must be in close contact with the patient.
The issue of this approach is that even measuring the same physio-

logical signal at almost the same time but in different places of the human body, like from inside and from outside will not give the same results, due to the inevitable noise. Many tried to study this matter and have proposed solutions, but the research is still open. As an example, Hu at al. [26] worked on the solution, presented by Venkatasubramanian et al. [27], of using a fuzzy vault scheme. They managed to secure the key exchange using physiological values, it is fundamental to take into consideration the structural limitations of ICDs. A complex solution that includes operations difficult to execute will consume more power, draining the battery faster. Furthermore, as already said, the ECG real-time measurement takes time, and using it to derive or exchange a cryptography key would take even more time, up to one minute, and this could affect the promptness of the emergency response, endangering the patient's life.

2.4.3 *Cryptography solutions*

Some attempts to use symmetric cryptography have been done, but it is very challenging to implement key distribution using conventional security solutions in the ICDs' context.
Halperin et al. proposed to add a symmetric key authentication and encryption between the ICD and the programmer. Their solution involved a master key on every programmer memorized in tamper-resistant hardware and diverse keys in the ICDs. However, even if this solution is an improvement over current techniques, having the key saved in every programmer could be a risk. If the hardware where the key is saved of just one programmer is compromised, it would be impossible to revoke the keys and every patient with an ICD whose key was stored in that hardware would be exposed, at least until the device is substituted.
A different approach could be to save the master key in the cloud to avoid having multiple copies of the key and instead have just one

instance. In this case, the programmer should be connected to an Internet network. Anyway, this is not a feasible solution, since programmer should always be able to work, especially during emergencies, even during Internet breakdowns or cloud provider faults.
Other ways to distribute the key very fast during emergencies could be by printing it on a bracelet or on the patient's skin, but both these methods are hard to revoke or reissue the key, so they are not feasible solutions.
Finally, using a PKI, a certificate with a trusted programmer's public key can be stored in the ICD when implanted then during emergencies, the different programmer could contact the Certificate Authority and obtain the certificate to use to establish the connection with the ICD. However, this solution is too expensive in terms of computation and energy consumption, so it is considered inappropriate for ICDs.
So, symmetric cryptography has issues regarding the management of the key exchange, otherwise, it would be a valid option.

2.5 NIST CALL FOR LIGHTWEIGHT CRYPTOGRAPHY

Another option related to symmetric cryptography is the latest call for lightweight cryptography held by the NIST. The call considered highly-constrained devices that communicate wirelessly with each other that work to accomplish a task. NIST considered that currently most cryptographic algorithms are designed for desktop/server applications, so don't fit for these constrained devices. For this reason, in 2019 NIST decided to open the call, asking for algorithms that follow specific requirements. Algorithms should be Authenticated Encryption with Associated Data (AEAD) functions, a variation of Authenticated Encryption (AE) that permits the receiver of a message to check the integrity of both encrypted and unencrypted information in the message.
AEAD is a function with four byte-string input and one byte-string

output. The inputs are the plaintext of flexible length, associated data of flexible length, a nonce of fixed length that has to be unique under the same key and a key of fixed length that should be around 128 bits. The output is a ciphertext of flexible length. The algorithms should support decryption-verification, meaning that it should be possible to retrieve the plaintext from the ciphertext, given associated data, nonce and key [28]. Considering the security point of view, AEAD algorithms should guarantee both confidentiality of the plaintexts and the integrity of the ciphertexts, the security level should be guaranteed as long as the nonce is unique, so not repeated under the same key.
Anyway, this call requires only the algorithms to secure the communication between two resource-constrained devices, not how the communication started or how to secure the communication in the first place. It is assumed that both the devices already possess the symmetric key needed for the encryption/decryption of the messages, so how the keys are distributed is not taken into consideration. Unfortunately, this is a matter that has to be considered in the case of ICDs. It is not feasible to store just one key in the device when the ICD is implanted, because this would mean that the implanted device could establish a connection only with the programmer whose key is in its memory. Considering the ICDs' context, for example during emergency situations, the patient could be in a hospital different from the one he is usually treated at, and there must be a way to access his ICD with a different programmer to treat him. Hence, a secure communication protocol designed for ICD should consider the possibility that the device could need to connect to different external programmers in a fast and easy way, especially in emergencies. However, even if the algorithms proposed in the NIST call do not consider this aspect, have a peculiarity that makes them relevant in this matter: they are designed to be as lightweight as possible. They are studied to run on processors with limited computational capacity and to consume little RAM and battery, which are the same structural limitations we discussed in the previous sections about the ICDs. Therefore, even if the key exchange

is not considered by these algorithms, they could set a standard for the intakes in terms of computational capacity, memory usage, and battery consumption. Starting from the NIST algorithms and conduct some tests to measure their performances could be an interesting way to understand how low the intakes should be to consider an algorithm as "lightweight". Moreover, in the research literature, there are no other similar studies that could identify other lightweight solutions, leaving this as the most reliable mean of comparison to evaluate new proposed solutions.

3

FEASIBILITY ANALYSIS

In this chapter, we will perform a feasibility analysis on an approach that relies on asymmetric cryptography to securely exchange a shared key that can, in turn, be used to secure ICD's wireless telemetry. In particular, we will first explain why combining an asymmetric key agreement protocol with a symmetric message encryption/decryption procedure is necessary to make ICDs accessible only to authorized users, in all situations. We will consequently describe the main concept on which this approach is based on and explain its strengths and drawbacks concerning the ICD domain. Lastly, we will make considerations on how to evaluate its feasibility in terms of resource consumptions and we will explain how the algorithms proposed in the previously described NIST call can be used as a valid mean of comparison to test how close this approach is to a lightweight solution.
First, we have analyzed the main requirements necessary to secure ICDs without impacting their health-related functionalities.
Secondly, we have presented a new approach and described how it can be used to protect ICDs while taking into account their context.
Then, we have performed a security analysis of the proposed approach against the most popular attacks on the ICD telemetry.
Finally, we have discussed the technical feasibility of this approach on constrained devices like ICDs.

3.1 THEORETICAL REQUIREMENTS ANALYSIS

As stated in section 2.3.4, one of the most important requirements that must be considered while designing a solution to secure ICD's wireless telemetry is to provide access to authorized users during

emergencies. For this reason, simply storing a secret key inside an ICD during implantation and loading the same key inside a programmer is not enough to ensure its accessibility in all situations. In fact, in this case, only one programmer will be able to communicate with an ICD, or, at most, all the programmers of a single hospital. This solution is acceptable when a patient visits the hospital where he got his device implanted during regular follow-up visits but it will have serious consequences during emergencies. One example could be the case of and ICD wearing patient that is unconscious and that has been moved to a different hospital where the key to access his device is not known. Medical staff would probably need to gain access to the ICD in order to access medical data, change the delivered therapy, or to turn it off. Without knowing the secret key they will not be able to do any of the aforementioned operations. Obtaining this key would probably be complex and time taking, because they would have to discover in which hospital it is stored, without the patient's help. Even if they know in which hospital the secret key is stored, there still may be problems to obtain such private information fast enough to be useful for the rescue attempt. Therefore, in order to guarantee the patient's safety, it is necessary to design a secure solution that allows sharing a secret key almost immediately when requested by legitimate entities. Despite this, using symmetric cryptography to encrypt and decrypt messages between two parties that share a common key is a lightweight and fast solution that will protect ICDs from most of the attacks presented in the threat model while also satisfying the requirement to consume as less as possible ICD resources. Consequently, sharing a key in a secure manner, while also considering ICDs limitations, has become a discussed topic in the research community.

Another important requirement that must be analyzed when deriving a common key is verifying the authenticity of any external device that is trying to communicate with an ICD. Without any access control procedure, an attacker could spoof himself for a legitimate device and establish a communication with an ICD, obtaining the control of

it. This consideration clarifies that any solution developed to secure ICDs should also perform an authentication process before starting any other security protocol or message exchange.

Lastly, it is important to consider the hardware limitations of ICDs. Solutions that require to execute complex primitives could exhaust ICD's resources in terms of computational capacity, memory size, and battery longevity. Therefore, it is necessary that any attempt to secure ICD's telemetry is lightweight enough to avoid affecting negatively its functioning or its durability.

Different solutions, that have already been described in chapter 2 have been proposed to guarantee access only to legitimate parties during emergencies, each of them with advantages and drawbacks. For example, the usage of external wearable devices will definitely save the ICD resources, but will also force a patient to constantly wear a device that reminds him of his condition. Measuring real-time physiological values will hinder many attack attempts by forcing the attacker to stay in close contact with the victim, but it will take time and will go against the advantages introduced by the ICD's long-range communication.

As can be seen, designing a solution that takes into consideration the ICD context and technological limitations is not an easy task without introducing other side effects that may not be accepted by patients or manufacturers. Among the proposed approaches, a key distribution system based on public key cryptography has been almost ignored due to the assumption that it is too resource-demanding for limited devices. On the other hand, as we will better explain in the following paragraph, this approach will allow a legitimate programmer to access an ICD during emergencies while not having the drawbacks that are present in the other previously described solutions.

3.2 PUBLIC-KEY DISTRIBUTION SYSTEM APPROACH OVERVIEW

One common way to solve both authentication and key exchange problems is to establish a PKI and to use an asymmetric cryptography-based key agreement protocol.
In particular, a PKI is a set of software, hardware components, and rules that can be used to bind a specific public key to the identity of a verified and legitimate entity [29], [30]. This binding is made possible thanks to a trusted third party, which is generally referred to as Certification Authority (CA). Basically, a CA is responsible for verifying the identity of a specific entity and, if it is a legitimate one, issuing a valid digital certificate to it. These certificates contain a digital signature that can be used to prove the ownership of a specific public key, hindering spoofing attack attempts.
A Digital Signature Algorithm (DSA) [31] employs asymmetric cryptography to generate and verify such signatures. In particular, the private key of a CA is used to generate a valid signature from a message, that typically contains the public key and the identity of a verified entity. The message and its generated signature compose a digital certificate. Any entity that receives a certificate can then verify the authenticity, integrity, and non-repudiation of the message in it by using the CA public key. Figure 3.1 illustrates how DSA algorithm is used to sign messages and to verify both authenticity and integrity of a signed message.

Thanks to a trusted CA and the DSA algorithm, two communicating entities can send their certificates to each other and verify that the received public key (that should be present in the message part of the certificate) belongs to a legitimate entity and that it has not been modified during the message exchange. The main assumption on which a PKI is based is that the third party authority must be trusted by both communicating parties. Obviously, if the private key of a CA is discovered or if the CA signs a certificate to a malicious entity the

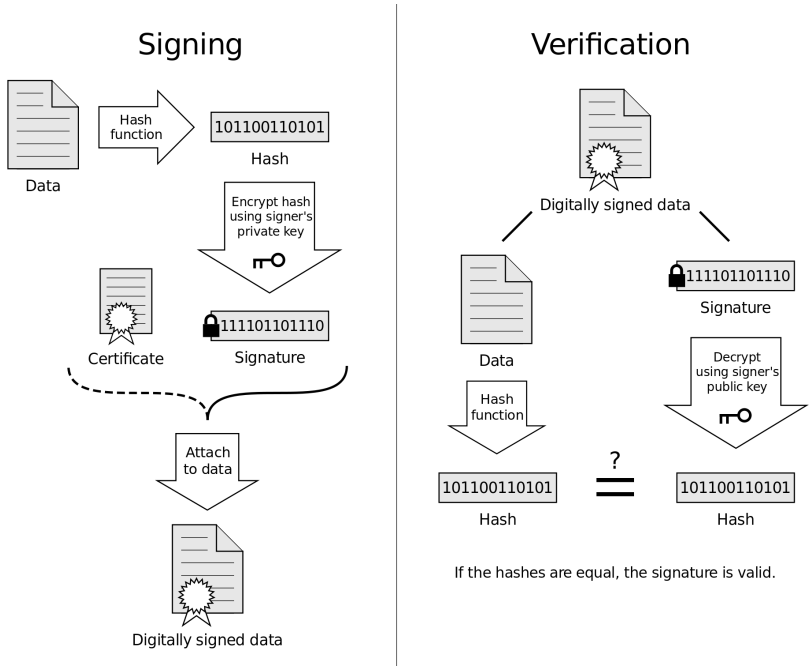


Figure 3.1: DSA certificate signing and verification procedures. A CA uses its private key to produce a valid signature. The public key of the same CA can be used by any entity to compare the hashes of the signature and of the message. If they are identical, the verification step is passed.

advantages brought by the PKI are nullified. Once both communicating parties have received the other entity public key and have verified its authenticity and integrity, they can start an asymmetric key agreement protocol to solve the key exchange problem.

The most popular way to derive a common shared secret is by using the Diffie–Hellman (DH) protocol [32], [33]. DH is a non-authenticated key agreement protocol based on asymmetric cryptography. In particular, both communicating parties must generate a public/private key pair and exchange their respective public keys over a potentially insecure channel. Once both parties possess the other entity public key they can use it and their own private key to derive a common shared secret. This secret can also be a new private key that can be used in combination with a symmetric cipher to encrypt and decrypt messages. It is important to notice that the shared secret is never exchanged over the insecure channel. Therefore, by using this protocol it is possible to securely obtain a private key in any situation without the need to obtain or possess pre-shared secrets.

Despite this, DH alone is an anonymous protocol, which means that it does not provide authentication of the communicating parties, and, therefore, it is vulnerable to MITM attacks. This problem is solved by the PKI previously described, in fact, when an entity receives a certificate from another one it does not only obtain its public key but also a way to verify the authenticity of it through the certificate's signature. This authenticated public key can then be used to derive a shared secret at any moment. By following this approach it is possible to grant access only to authorized users at any moment, emergencies included.

One drawback of using asymmetric cryptography is that it is much more expensive than symmetric cryptography. Using this approach will satisfy the emergency access requirement, without presenting the drawbacks of other state-of-the-art solutions, but will probably be more expensive in terms of computational power, RAM size requirements and battery consumptions. For this reason, it is important to

verify if ICDs have enough resources to apply successfully this approach without exhausting their energy too soon and to evaluate if this tradeoff is actually convenient.
One way to do so could be to compare it with the collection of lightweight cryptographic algorithms like the ones proposed in the current NIST call previously presented [28]. If the performance result of a solution based on this approach is not too far from the results given by NIST algorithms it means that this approach is valid enough to be considered in securing ICD's telemetry.

3.3 SECURITY ANALYSIS

So far we have analyzed how a public key distribution system approach would grant access to an ICD only to authorized users, in any situation. Another important aspect, that must be considered to evaluate this approach, is the level of security that can be achieved by applying it on ICDs. In order to do so, we will now make some considerations on the countermeasures taken by this approach in response to the most common attacks that can be performed on ICDs, that have already been described in section 2.3.3.

3.3.1 *Eavesdropping attacks*

Since currently, ICDs do not perform any form of encryption procedure on their messages, any attacker that is able to intercept those messages will also be able to read and understand them, compromising the patient's privacy. Therefore, even a passive attacker could eavesdrop the communication channel and steal sensitive information.
The proposed approach will counter this type of attack. After obtaining a common shared key, both ICD and programmer can communicate with each other using it to encrypt and decrypt messages with a symmetric cipher. Without knowing the shared key it will be impossible

for any attacker to understand the content of the intercepted messages. The shared key is never exchanged between the two communicating parties, it is derived from a DH key agreement protocol. Therefore a passive eavesdropper will never be able to obtain such key by simply sniffing messages.
It is possible that an ICD sends identical messages to a programmer during one or multiple a communication sessions. Encrypting two equal messages with the same shared key will generate two identical ciphertexts. An attacker, even if incapable to comprehend the message content, could be able to make some correlations between a ciphertext and its plaintext. In order to avoid this, adding some random bits to the communication message format is a valid strategy to make all ciphertexts different from each other. Another solution could be to derive a new shared key for every communication session, but since it will require to perform more complex and expensive asymmetric primitives it is not recommended for limited devices like ICDs.

3.3.2 *Spoofing attacks*

ICDs currently do not perform any kind of authentication procedure, therefore, an expert active attacker could try to impersonate a legitimate programmer and simulate a communication session. By doing so the attacker can not only steal private data but also alter the device functioning or force it to deliver incorrect therapies to the patient.
With the assumption that a CA is trusted, reliable, and that it has not issued certificates to any malicious entity, it will be impossible for an active attacker to spoof himself for a legitimate device. In fact, sending to an ICD a certificate that has not been signed with the CA's private key will result in the refusal of the certificate because an ICD will verify its signature using the CA's public key. This means that an attacker could either generate a valid certificate by stealing the CA's private key or asking the CA a valid certificate. Both options have to be discarded

because they both rely on the weakness of the CA, that is assumed to be robust enough to counter such attempts.
Another possibility for the attacker could be to steal the certificate of a legitimate entity (a programmer, for example) and use it to communicate with an ICD. Obviously, stealing another entity valid certificate is possible only if they are exchanged over an insecure channel. Even in such a case, it will be impossible for an attacker to gain control of the ICD. Every certificate is bound to a single entity and contains the public key of that specific entity. For an attacker would be impossible to obtain the private key of that entity (at least without stealing or tampering it) and, therefore, it would be impossible for him to derive the same shared key during the DH key agreement protocol. Without the correct shared key, it would be impossible to encrypt and decrypt any message, making any spoofing or MITM attempt useless.

3.3.3 *Replay attacks*

Any active attacker could also try to replay previously eavesdropped messages to force an ICD to deliver specific treatments when they are not necessary.
The proposed protocol will make replay attacks much less effective since an attacker would never be able to understand the message content, thanks to both to authentication and message encryption procedures. For this reason, an active attacker could only try to replay messages without knowing their actual purpose and this fact reduces the probability of successful attack attempts. Despite this, it could still be possible to negatively affect the ICD functioning by randomly replaying messages. In order to completely prevent this attack category, it could be useful to add a counter to the message format for every session or message. In this way, if the counter is not the one that is expected to be, the message will be rejected, hindering any replay attack attempt.

3.3.4 *Battery DoS attacks*

Lastly, any active attacker could try to send many connection requests to an ICD in order to deplete its battery and forcing the patient to surgically remove its implanted device earlier than expected. In order to protect an ICD from such type of attack, it would be necessary to perform zero-power countermeasures and this can be hardly done with current ICD hardware. Adding to an ICD the functionality to harvest energy from other sources and using the obtained energy to perform the authentication and key derivation steps of the proposed approach will make this attack ineffective.

3.3.5 *Conclusion*

As demonstrated, using a public key distribution system will definitely protect ICDs against most of the attacks that have been previously described without requiring other external devices or measuring the patient's physiological values. Therefore, the last aspect that remains to be considered is the technical feasibility of such approach on resource-constrained devices.

3.4 EXPERIMENTAL SETUP DESCRIPTION AND TECHNICAL ANALYSIS

As previously stated, an approach based on a public key distribution system and the DH protocol protects the communication channel between ICDs and external devices from the most common attacks. Moreover, we have also shown that this proposition complies with the requirements given by the ICDs' context as the need to guarantee access during emergencies.
At this point, we need to verify if the structural characteristics of ICDs are able to handle the computational requirements of this approach.

As we pointed out, asymmetric cryptography's main drawback is that it is more computationally expensive than symmetric cryptography and this is the reason why this approach was never really pursued by the research community. Therefore, to conclude the feasibility analysis of this approach, the last aspect to be considered is to verify if the ICD resources, in terms of RAM size, battery capacity, and computing capability, are enough to handle an asymmetric cryptography based approach.

In order to do so, it is necessary to measure these metrics on a solution based on the aforementioned approach and compare the results with the known hardware characteristics of ICDs, presented in section 2.2. In this way, it will be possible to effectively examine if the proposed approach is too resource-demanding or not for ICDs. Moreover, we will also compare the performances of this approach with the results obtained performing the same measurements on the NIST call algorithms. As previously stated, these symmetric algorithms have been designed to be as lightweight as possible. Therefore, using these algorithms as a baseline will help us in understanding how close a solution based on a public key distribution system is to a lightweight proposal for resource-constrained devices.

Obviously, the best approach to compare these performances would be to implement both NIST algorithms and a solution based on the proposed approach on a recent ICD sample. In this way, it would be immediately clear if a new proposal is actually feasible or not on such devices. Unluckily, ICD manufacturers avoid issuing samples for non-medical purposes and rarely cooperate with researchers. The main reason for this behavior is that they prefer to improve their devices in their own Research and Development departments without sharing private data and know-how with external entities, for economic reasons.

Therefore, since this more direct way was not achievable, we have implemented both NIST algorithms and our proposed solution on a prototype. After that, we have conducted three different experiments

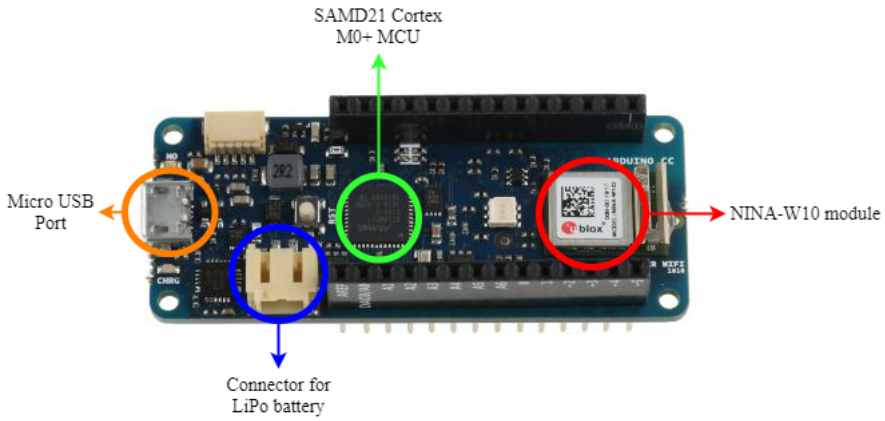


Figure 3.2: Picture of an Arduino MKR Wifi 1010 board. This board presents several features that are similar to ICD limitations. Moreover, thanks to the NINA W10 module, it is able to communicate using BLE technology, simulating ICD telemetry.

to measure the execution time, RAM usage, and battery consumption to evaluate their technological requirements.

Since ICDs have limited resources, the choice of the hardware setup had to be appropriate and suitable to simulate the behavior of an ICD during the message exchange protocol. Our final choice, illustrated in Figure 3.2, is an Arduino MKR Wifi 1010 board since it has several technical features that resemble the ones of an implantable device.

We will now briefly discuss why we chose this board, comparing some of its technical components with the ones present in ICDs, that have already been presented in section 2.2. We will start by comparing their processors. IMDs mount an ARM Cortex M3, while Arduino MKR Wifi 1010 board mounts a microcontroller SAM D21 [34] that carries an ARM Cortex M0+ processor. We will describe in deep the technical details of our choice in chapter 5. In any case, the Cortex M0+ processor is optimized for cost and energy-efficient microcontrollers and its architecture is very similar to the one of the Cortex M3. This allows any algorithm developed for a processor mounting a M0+ to be executed without issues also on a M3. Therefore, we used the more energy-efficient Cortex M0+ of our Arduino MKR WIfi 1010 board to simulate the behavior of the Cortex M3.

Considering the RAM size, ICDs have a limited memory that can be used to store data for computations, usually from 32 to 256 KB. The Arduino MKR Wifi 1010 board has a RAM of 32 KB, that fully respects this constraint.

Arduino MKR board can rely on two different means to connect to a power source. The first one is its USB port that, once connected to a Personal Computer (PC), can be used both to provide energy to the board and to upload the code that will be later run by the board. The second one is its Lithium Polymer (LiPo) battery connector, which better simulates a device like an ICD since these devices are powered by batteries that are expected to last for a long interval of time. In particular, we used this second mean to better simulate the battery consumption of the security primitives that we have measured in our experimental evaluation.

Arduino MKR boards are able to communicate with other devices using Bluetooth Low Energy (BLE), thanks to the NINA-W10 module that works on the ISM band range. In our case, we used two Arduino MKR boards, one to impersonate an ICD, and another one to represent an external device that is supposed to communicate with it. In this way, we were able to simulate the message exchange of two real devices, allowing us to better establish their consumption.

Finally, we chose Arduino boards in order to make our experiments easily replicable. Arduino IDE is intuitive even for people who are not used to hardware programming. Programs, that in Arduino context are defined as "sketches", are written in the popular C language and they can be directly uploaded to the board from the PC through the USB cable. Another aspect that we considered in our hardware choice is that the algorithms presented in the NIST call have been submitted with a reference implementation in C code available to download. Most NIST proposals have more variants, which differ by their security level and their performances. In order to be as impartial as possible, we always chose the one recommended by the authors. Since Arduino libraries need to be written in C++ language, we had to work on

the implementation of each algorithm to convert it from C to C++ language in order to use all of them as Arduino libraries. We wrote a simple Arduino sketch that was able to execute both AEAD encrypt and decrypt procedures while simulating a message exchange with another device using BLE communication. We had to adapt this sketch to all the reference implementations of the algorithms presented in the NIST call. To do so we imported, for each algorithm, its correspondent C++ library that we had previously obtained from their reference implementation. Lastly, we had slightly modified our sketch to take into account the algorithm implementations and the slightly different parameters necessary to run them correctly.

We conducted three tests to measure the three different metrics: execution time, RAM Usage, and battery consumption. The technical details of each experiment will be presented in chapter 6.

The test about execution time has been performed using the micros() function of the standard Arduino library. We called the function twice for each primitive we needed to measure, subtracting the first from the second, obtaining precise results.

The RAM test was conducted thanks to a library that we downloaded from the Arduino Playground website. By using this library we were able to measure the free space between heap and stack during the run time execution of each algorithm.

Regarding the battery consumption, we connected a LiPo battery to each board to power them up, we used an Ina219 current/power sensor for each board connected to an Arduino Uno to read the sensors' measurements. We triggered a pin on each MKR board to start and stop the measurements on the functions we were interested in, this allowed us to obtain many values per measurement of which we calculated the average.

Each test has been performed for all of the 32 algorithms presented in the NIST call for lightweight cryptography. All the test results are presented in chapter 6 in 6.4, 6.1 and 6.7. The values shown in the tables are calculated as the average between the test results of the two

cryptography primitives functions, encrypt and decrypt. These two values generally do not differ much from each other, but in most cases the decrypt function is slightly more expensive. The results give a wide range of values, and if an algorithm offers good performance in one test it was not obvious that it did well in the others. Therefore it is difficult to define the best candidate. The values vary greatly from maximum to minimum, this defines an interval of values that we can rely on when studying other approaches. We will consider these results as a baseline to understand if a solution based on the proposed approach is comparable or, at least, close to a lightweight solution like the one presented for the NIST call.

3.5 FEASIBILITY ANALYSIS CONCLUSIONS

In this chapter, we examined how a solution based on PKI and asymmetric cryptography has several positive aspects.
We stated that this approach is effective in the ICDs' context since it allows secure access to authenticated parties during emergencies without requiring other external devices or to execute complex ad time-taking operations.
Then, considering the threat model presented in the background section 2.3, we analyzed how this approach protects the communication successfully against most of the most relevant attacks. Finally, we presented a baseline based on the results of the experimental evaluation conducted on the NIST algorithms. This was done to have a mean of comparison for the performances that this approach should possess to be defined as "lightweight".
The only relevant remaining aspect to evaluate is how this approach performs once implemented. As previously stated, asymmetric cryptography has been discarded as an approach in this kind of context because considered too computationally expensive. The only way to actually determine the feasibility of this proposed approach is to im-

plement it on a prototype and measure its performances, as it was done for the NIST call algorithms. In this way, we will be able to understand if its consumptions are low enough to be handled by the constrained ICD resources.

Taking into account these findings, we can assume that trying to implement a communication protocol based on PKI and asymmetric cryptography for the ICDs' domain could give to the research a new starting point to evaluate solutions structured in this way.

4

DESIGN

While in the previous chapter we have analyzed the feasibility of an approach based on asymmetric cryptography to secure ICDs, we will now present the design choices that we made to ideate a lightweight communication protocol based on the aforementioned approach. In particular, in this chapter, we will firstly describe which security primitives we have chosen for our protocol and we will explain the motivation of our choices.
Secondly, we will present our communication protocol, based on such primitives.
Lastly, we will describe in detail each step to perform it, highlighting also other possible variants to it and explaining for each of them their pros and cons. This solution will then be compared, in chapter 6, with NIST algorithms in order to evaluate its lightweightness and its suitability, in terms of performances, in the ICD domain.

4.1 SECURITY PRIMITIVES DESIGN CHOICES

After analyzing the feasibility of an approach based on a key distribution system and evaluating the advantages that a working implementation of such an approach could bring to the ICD domain, we have started designing a communication protocol that takes into account ICD context and limitations.
To guarantee security of a communication protocol based on a PKI we identified that our proposal should be subdivided into three main phases: authentication, shared key derivation and secure message exchange. As previously stated, the first two phases will require the use of asymmetric cryptographic primitives. Despite being more com-

putationally expensive than the symmetric ones, they are essential to guarantee the accessibility in every situation requirement that is essential when taking into consideration the context in which ICDs operate. In particular, we have chosen to employ:

- Elliptic Curve Digital Signature Algorithm (ECDSA) for the authentication phase
- Elliptic-curve Diffie–Hellman (ECDH) protocol for the shared key derivation phase
- Advanced Encryption Standard (AES) for the secure message exchange phase

As their names suggest, both ECDSA and ECDH are based on Elliptic Curve Cryptography (ECC), an asymmetric cryptography model founded on the algebraic structure of elliptic curves [35]. This approach has the positive aspect that it guarantees the same security level of other non-ECC approaches while requiring a shorter key-size. This means that in the particular context we are dealing with, we can reach an adequate security level by using a smaller key that will be more easily computable and implementable. This will, in turn, ease the process of making our approach feasible for resource-constrained devices.

Moreover, since ECC is based on the specific parameters of a curve, the choice of this curve is critical because it can influence the execution of the protocol. Among many possible curves, our choice is Curve25519. This curve has been presented in [36] by D. J. Bernstein and has many features that are convenient for the ICDs' domain. First, this curve resulted to have better performances in terms of speed and execution compared to other curves. Then, it uses a public key of only 32 bytes, the smaller key size reduces compute, memory, and transmission requirements. Finally, Curve25519 characteristics and implementation are publicly available and are present in many cryptographic libraries, so it is easier to use it and to find documentation about it.

These are the main reasons that led us to choose ECC with Curve25519 for the first two phases of our protocol: ECDSA for the authentication and ECDH for the shared key derivation. Moreover, these two algorithms are often used together to guarantee an authenticated and secure key exchange. We will now describe each phase of the proposed approach more in detail.

4.1.1 *Authentication*

As stated in the previous sections, authentication is a fundamental step to guarantee access to the device only to authorized users, also during emergencies. We decided to use an algorithm for data digital signature, in particular, we chose ECDSA. ECDSA is a variation of DSA based on ECC. Using ECC brings several advantages that were taken into great consideration while designing our protocol. For example, the fact that it requires a smaller key size leads to the advantage that it is possible to realize efficient and concise implementations of the cryptographic primitives based on ECC. This aspect, in turn, makes these algorithms more adequate for devices that possess energy efficient processors and small chips, producing less heat and consuming less power. All these properties are particularly relevant in the ICD domain because these implanted devices have low computational capabilities and small memories, as we stated in section 2.2.
ECDSA allowed us to verify data authenticity without compromising its security, while also guaranteeing integrity and non-repudiation properties. In particular, we have designed this step to work in combination with a PKI where a CA was assumed to sign certificates containing the credentials and the public keys of the external devices that need to establish a connection with any ICD, as we will better explain in the following section. By verifying the validity of a received digital certificate an ICD can be certain that a legitimate external device has sent it and that the received message was not altered during the

transmission. Moreover, making use of a PKI makes possible to bind a public key of the external device to its identity. In this way, it is possible to avoid repeating this step each time the implanted device communicates with the same external device. In fact, by storing the external device credentials inside of the ICD memory and associating them with the correspondent external device public key will make it possible to skip the asymmetric certificate verification phase for all the successive communication session with that device.

4.1.2 *Shared Key Derivation*

After the first phase of authentication, we realized the shared key derivation phase using ECDH. This protocol is a variant of the DH protocol, based on ECC. It is a key agreement protocol, meaning that it defines how the keys are generated and exchanged. In this phase, the two devices involved in the communication exchange their public keys to derive the shared secret that will be used to encrypt and decrypt the messages with a symmetric block cipher. To derive the shared key the involved parties should possess their own private key and the public key of the other party. The shared secret key is computed combining the private key of one device and the public of the other. In this way, it would be impossible for any attacker, even if it performs a MITM attack, to obtain the shared key, since it would not be able to access the private keys. At the moment this protocol is secure, at least until a fast way to solve the discrete logarithm problem will ever be found. This problem is an open mathematical challenge, the goal is to find a way to reverse the mathematical operations which the ECDH protocol is based on. If anyone will ever find a way to do that, the protocol would not be secure anymore. Despite this, it is still one of the most valid key generation protocols available.
It is important to notice that, when a shared key is derived, both parties can use it multiple times and during more communication

sessions. Thanks to the combination of ECDSA and ECDH in this context, the ICD will be able to save in its memory not only the credentials of any external device it establish a connection with, but also the shared secret generated during that communication. In this way, it is possible to avoid repeating all the asymmetric cryptography based steps of this approach. In the next communications, the external device will only need to send to the ICD its credentials. The ICD, then, will be able to retrieve the shared secret from its memory without computing it again, saving time and computational power. Of course, the external device could save the shared secret too, but since it is not resource-constrained like the implanted device, it would not be an issue to compute it again.

4.1.3 *Secure Message Exchange*

After the key exchange phase, the ICD and the external device that is communicating with it have established the same shared key. In this way, they can start a secure communication using that secret as encryption/decryption key. This communication will use AES, that is a symmetric cryptography standard for communication. It was presented during the NIST call for the definition of a new encryption standard to substitute Data Encryption Standard (DES) in 1997. It was finally approved and presented as the new standard in 2001. It uses a block cypher of 128 bits, but it can have keys of three different lengths: 128, 192 or 256 bits. We decided to use the 256 bit version because it is the most secure and also because the shared secret computed during the ECDH phase is 32-bytes long, that is equivalent to 256 bits. So our protocol uses the same 32-byte key to encrypt and decrypt the messages, and that key is the shared secret. AES is currently used by the National Security Agency (NSA) to encrypt classified information and there are no known attacks that succeeded in breaking the algorithm.

In the next section we will present how we combined ECDSA, ECDH and AES to realize a secure communication protocol designed for ICDs.

4.2 PROTOCOL PRESENTATION

So far, we have described in detail the single steps that are necessary to produce a secure solution based on a key distribution system approach. Now we will present how, by combining these phases together, we were able to design a communication protocol specifically tailored for securing ICD's telemetry. As we stated, the first phase is the authentication procedure, based on a PKI, that distributes certificates, and ECDSA algorithm. After having verified the legitimacy of an external device, a shared key derivation phase, based on ECDH, is performed using the public key present in the received certificate. Finally, once both communicating parties have generated the same secret key, a secure message exchange is performed using AES.

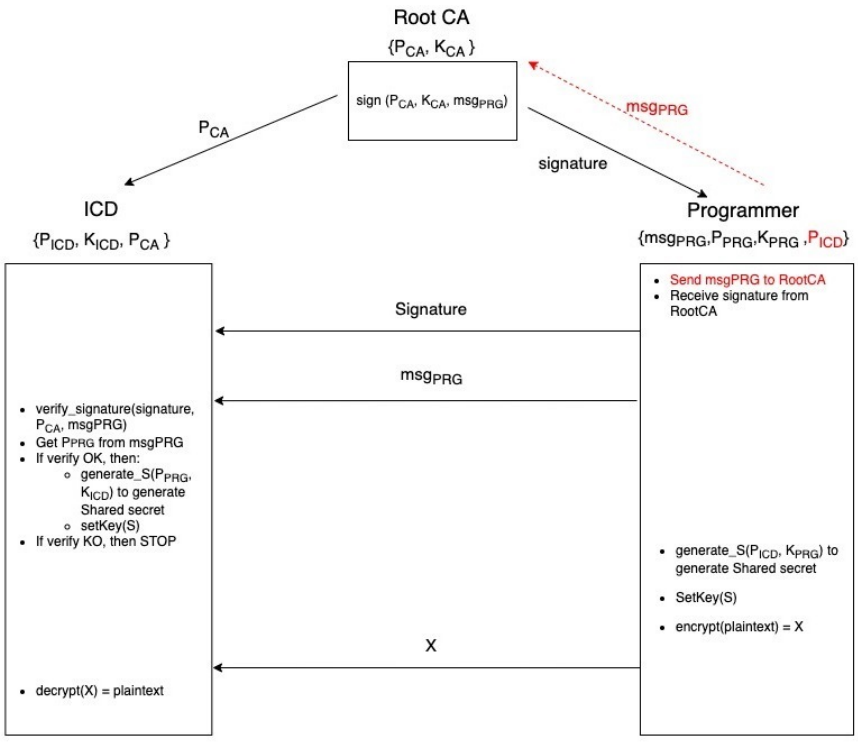


Figure 4.1: Diagram presenting the protocol

In Figure Figure 4.1, we present a sequence diagram showing the communication protocol that we have designed. The parts highlighted in red represent our own implementation choices, meaning that it is not mandatory for the protocol to behave in that way, but that we have decided it was the best option considering how we have implemented it.

4.2.1 *Main actors and initial setup*

Three main actors are involved in our protocol:

- an ICD.
- a programmer: one of the external devices that is responsible to interact with the ICD.
- a root CA: an entity that is responsible of signing certificates containing the credentials of a programmer, guaranteeing its identity.

We will now proceed by describing all the details that are necessary to correctly execute our communication protocol.
Firstly, an ICD should store inside its memory its own public/private key pair, "P_{ICD}" and "K_{ICD}", and the public key of a root CA, "P_{CA}". These values can be uploaded inside the ICD during its manufacturing process or during its implantation inside of a patient.
Secondly, a programmer should store its own private key, "K_{PRG}", a message, "msg_{PRG}", which in turn contains its own credentials, "ID_{PRG}", and its public key "P_{PRG}". Optionally, a programmer could also store the public keys of all the ICDs, "P_{ICD}", that have previously established a communication with it. The fact that a programmer already possess the ICD's public key is an implementation choice. Actually a programmer could retrieve the ICD's public key from a sort of online database. Another option could be that the ICD itself could send its own key to the programmer, only after having verified the

validity of the programmer's certificate. This last option, would be more computationally expensive and would require the ICD to send additional messages. For these reasons, we decided to avoid proposing it for our protocol, that is supposed to be as lightweight as possible.
Lastly, a root CA should possess its own public/private key pair, "P_{CA}" and "K_{CA}", that are used to sign and verify the signatures of the certificates.

4.2.2 *Communication protocol description*

In order for a programmer to be able to communicate with any ICD it is necessary that it obtains a valid certificate from the root CA. To achieve this goal, the programmer is supposed to send msg_{PRG} to the root CA. After verifying the legitimacy of the programmer, the root CA can generate the signature of msg_{PRG}, producing a valid certificate. The root CA is expected to use a signing function to generate a signature, "signature", from msg_{PRG} using both P_{CA} and K_{CA}. Sending back the certificate, or simply its signature, to the requesting programmer will guarantee that P_{PRG} is bounded with ID_{PRG}.
Once the programmer has received the signature from the root CA, it will be able to send it to the ICD with msg_{PRG} that contains the programmer's public key and credentials. Sending msg_{PRG} to the ICD is fundamental to allow it to verify the signature performed on it. The ICD verifies the validity of the certificate by comparing msg_{PRG} with the received signature. In order to do so, it uses a verify function that requires the root CA public key, P_{CA}, that should have already been stored in the ICD previously. If the verification is successful, the ICD will use the programmer's public key, P_{PRG}, contained in msg_{PRG}, to generate the shared secret "S". S will be set as encryption/decryption key for all the following exchanged messages using a setKey function. The programmer that already possesses the ICD's public key, P_{ICD}, will generate the same shared secret "S" and set it as encryption/de-

cryption key in the same way as the ICD. From this moment on the communication will be based on symmetric cryptography, using the shared secret as encryption key. If the signature verification fails, then the connection between the devices will be interrupted.

The authentication and key derivation phases will be executed just the first time that a new programmer connects to the ICD. Once generated, the shared key will be memorized in the ICD with the correspondent programmer's credentials. During the following connections, the programmer will send to the ICD only its credentials so that it can select the correct shared secret to use for the communication. In this way the following communication sessions will be lighter and faster.

5

IMPLEMENTATION

After designing each step of the protocol, as we presented in the previous section, we implemented it on a prototype built with two Arduino MKR Wifi 1010 boards. We utilized the same boards we used as a test-bench for the NIST call algorithms, this way we were able to run the same tests on our protocol and compare the results with the ones from the NIST algorithms. Thanks to this comparison we were able to understand if the solution we proposed was actually lightweight enough to be implemented on ICDs. Therefore, we have first implemented the protocol on the two boards, then we have run the same experiments on the execution time, RAM usage, and battery consumption that we have performed on NIST algorithms.
In this chapter, we will describe in detail how we realized the protocol using Arduino libraries and cryptographic primitives.

5.1 HARDWARE SETUP

As we stated, to realize the protocol we implemented it on two Arduino MKR Wifi 1010, on that simulates the ICD and the other the external device that wants to communicate with the ICD, the so called programmer in chapter 4.

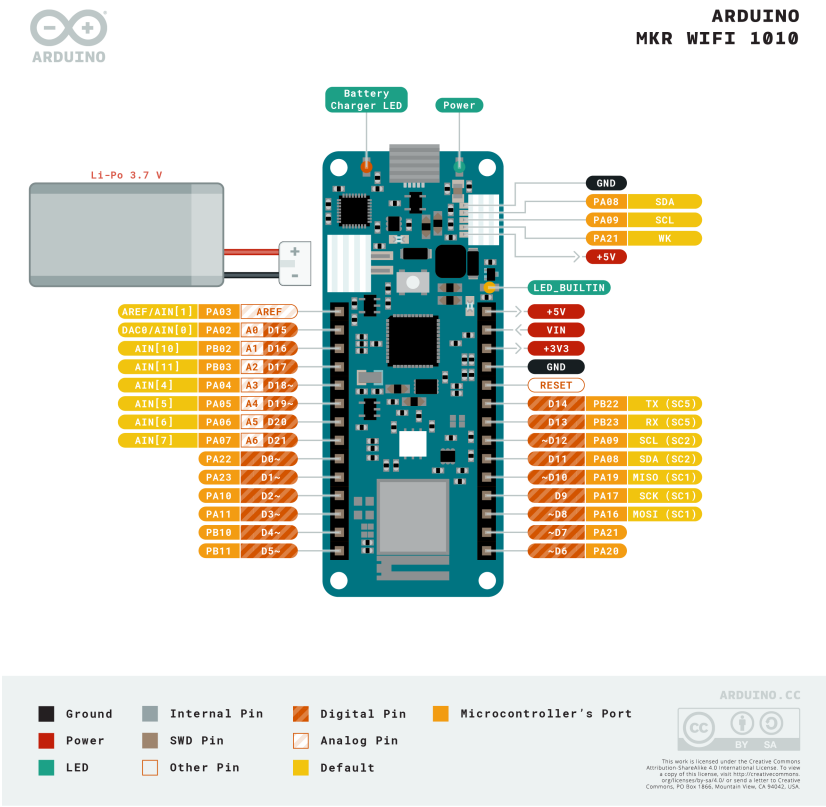


Figure 5.1: Pin-out of Arduino MKR Wifi 1010

As anticipated in section 3.4, the MKR Wifi 1010 carries an ARM Cortex M0+ processor that has many similar features to the ARM Cortex M3 that is generally used on ICDs. In 5.1 we present a comparison between the two ARM Cortex processors based on the information available in the technical datasheet of the processors on the ARM developer website [37], [38].

Characteristic	ARM Cortex M0+	ARM Cortex M3
CoreMark/MHz	2.46	3.34
DMIPS/MHz	0.95	1.25
Dynamic power	47.4 µ W/MHz	141 µ W/MHz
Planned Area	0.098 mm2	0.35 mm2
Pipeline Stages	2	3
ISA	Armv6-M	Armv7-M

Table 5.1: Comparison between ARM Cortex M0+ and ARM Cortex M3.

CoreMark/MHz and DMIPS/MHz are two different metrics used to evaluate the computational performances of Central Processing Unit (CPU)s. Both ARM Cortex M3 and M0+ have a 32-bit architecture, but M0+ consumes less, is smaller, and is less performant than M3. The Instruction Set Architecture (ISA) of the M0+ is the Armv6-M that is a subset of the M3's 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" [39]. This implies that any algorithm that can be executed on an ARM Cortex M0+ can be executed on an ARM Cortex M3 too. Thanks to this we were able to implement our protocol on the Arduino MKR board, being sure that the approach could at least be executed by the processor mounted on an ICD.

In this section, we will refer to the board representing the implanted device as "peripheral" and to the other, which represents the external device, as "central". We could have chosen a different and more powerful board to impersonate the programmer but we decided to stay with the same MKR Wifi 1010 board to keep the experiment simple and intuitive so that it could be easily replicable. Moreover, this allowed us to also compare the consumption of the two parties and check if there were relevant differences. To write and upload the programs, called "sketches", we used Arduino IDE as development environment and C as programming language.

The two boards were powered up using USB cables, connected to two different PCs. The USB cable was also used to upload the sketches from the Arduino IDE. Moreover, Arduino IDE has an important additional feature: the serial monitor. This monitor allowed us to read on our PC screens text or variables' values computed by our Arduino boards during the sketches execution. This aspect greatly helped us when debugging the code and was particularly useful to verify the correct functioning of our communication protocol.

5.2 BLUETOOTH LOW ENERGY-BASED COMMUNICATION

We used the connectivity module already present on the Arduino MKR Wifi 1010 boards, the NINA-W10, to make the two boards communicate. In particular, we established a BLE wireless communication between central and peripheral. In order to do this, we had to use a suitable library to have all the necessary functions, so we included the ArduinoBLE public library in our sketches.

BLE is based on the Generic Attribute Profile (GATT), illustrated in Figure 5.2, where two main roles are defined: the client and the server. The client, that is our case corresponds to our central device, begins the communication with one request sent to the server, that possesses data that the client is interested in and transmits an answer to the request. To exchange data during the BLE connection, usually, they are stored in so-called "characteristics". A group of characteristics that are logically correlated is callers a "service". To identify characteristics and services it is used a 16-byte Universal Unique Identifier (UUID).

In our sketches, we defined one service for each device and a characteristic for each key and message exchanged between the two boards. We assumed that the UUIDs of service and characteristics are known to both central and peripheral. A device can read the characteristics written from the other and write or modify the characteristics that the

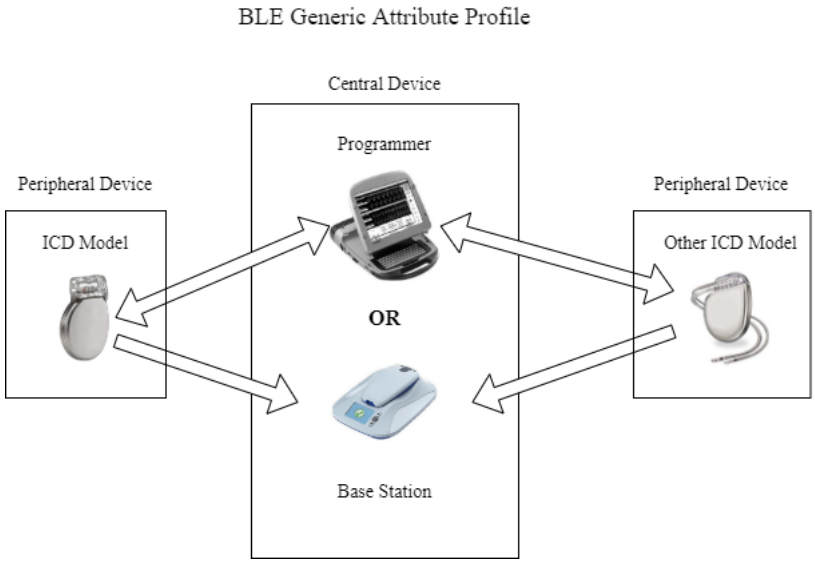


Figure 5.2: BLE communication is based on the GATT Protocol to exchange data between central and peripheral devices, bidirectionally. The peripheral, that in our context corresponds to an ICD, behaves as a server. It is responsible for answering to requests received by a client, that corresponds to a programmer or a base station.

other will read. So, once the connection is established it is possible to exchange data and messages.

5.3 CRYPTOGRAPHIC PRIMITIVES

Once the communication was set up with BLE, we needed to implement the cryptographic part of our protocol. We needed different primitives to realize the steps of the protocol as we designed it, as we presented in the previous section they are the message signing, the signature verification, the private/public key pair generation, shared secret generation, message encryption/decryption.

As we stated in chapter 4, we decided to use ECDSA and ECDH with Curve25519 thanks to the many advantages they offer in terms of performance and low consumptions. So, looking for open-source cryptography Arduino libraries to implement ECC based on Curve25519 we have found Arduino Cryptographic Library [40]. Developed by

Rhys Weatherley, is a very popular cryptographic library, that contains many different algorithms including the ones we needed to realize our protocol.

We used the Ed25519 class to handle the ECDSA steps, which are the public/private key pair generation, the message signing, and the signature verification. We decided to simulate the PKI making the central device compute also the operations that would be done by the root CA, in order to avoid adding another device to our setup making it more complex. Hence, we used the "generatePrivateKey" function to generate the private key from a random function for both central and peripheral, then we used the "derivePublicKey" to derive the public key from the private one. Both keys are 32-bytes long and are stored in arrays passed to the functions as parameters. Then, in the central device, we used the "sign" function on a 64-byte array containing the programmer's credentials and the public key. The result of the function is stored in the "signature" array and sent to the peripheral. When the peripheral receives the signature, it uses the "verify" function to check the authenticity of the signature. If the verification is successful the protocol goes on with the ECDH part. Thanks to the Curve25519 class, we were able to perform the shared secret generation, while we used the AES256 class to perform the encryption/decryption of the messages. To test and implement Curve 25519 the RFC 7748 [41] specifications has been followed.

We used the "dh2" function from Curve25519 to generate the shared secret that will be later used as symmetric key for the communication between the ICD and the programmer. It uses the private key of one party and the public key of the other, so it is necessary that the two devices exchange their public keys. In our case, the peripheral has the central's public key because it was contained in the message it received from the central to verify the signature. In the designed protocol, we stated that the programmer, the central, already has the public key of the ICD, the peripheral, stored in its memory or that it could retrieve the ICD's key from an online database. In this case,

we decided to make the peripheral send its public key to the central only if the signature verification is successful, since the devices are not connected online and we should have sent the public key just after the generation, but this would have messed the protocol simulation. So, the shared secret will still be a 32-byte key and, as we stated before, will be used to encrypt and decrypt the messages during the established communication.

For the encryption and decryption of the messages based on the shared secret, we used a different class, AES256, that oversees the message confidentiality of our protocol. To test the class the FIPS 197 vectors have been used to check its correct functioning. We used three main primitives: "setKey", that sets a 32-bytes key in the block cipher and it is the first one to be called; "encryptBlock", that performs message encryption on the plaintext; "decryptBlock" that performs message decryption on the ciphertext. Plaintext and ciphertext must be of the same length and are stored in buffers passed to the functions as parameters.

6

EXPERIMENTAL EVALUATION

As explained in chapter 3, the most important drawback that affects a public key distribution system approach is the necessity to use asymmetric cryptography, which could be too resource-demanding for limited devices like ICDs. Therefore, in order to complete our feasibility analysis, we have measured the performances of the communication protocol presented in chapter 4. In particular, we have based our technical analysis on three main metrics: execution time, RAM usage, and battery consumptions.

In this chapter, we will describe how we performed each of these three measurements in detail and we will present the results, discussing them.

We have not only verified if the obtained results are compatible with ICD's technical characteristics and limitations, but we have also compared them with identical measurements performed on all NIST algorithms. This second step was done to give an estimate on how close this approach is too lightweight solutions designed for resource-constrained devices and to evaluate if the tradeoff of having a more resource-demanding solution that grants emergency access without other drawbacks is actually convenient.

It is important to notice that we have performed the same procedures on the same hardware setup on both our proposed solution and on NIST algorithms so that the obtained results are meaningful and comparable with each other.

6.1 EXECUTION TIME

The first metric that we have considered to evaluate our proposed solution is the time taken to execute all NIST algorithms and each of the most important cryptographic primitives present in our secure communication protocol. Adding some delay to a communication exchange will not cause any major issue if we consider regular follow-up visits at the hospital or monitoring sessions at home, where a patient could simply do other activities while the message exchange is completed automatically or performed by a healthcare provider. On the other hand, a similar delay could have serious consequences during emergencies. Using time-taking security primitives will force doctors and hospital staff to wait before actually being able to intervene on a patient violating the balance between security and accessibility that is necessary when considering the ICD domain. Therefore, in order to respect the emergency access requirement, it is of primary importance that the communication between an ICD and a programmer is fast enough to avoid hindering any rescue attempt done by legitimate entities.

To measure the execution time of all NIST algorithms and of the primitives of our protocol we have used an Arduino MKR Wifi 1010 board. We have written sketches containing such primitives on a PC and loaded them, using Arduino IDE, on our Arduino board.

Arduino library contains many valuable functions that have been effective both during the implementation and this experimental evaluation phase. Among which, micros() function was determinant to precisely measure the execution time metric. Many Arduino boards are equipped with a timer/counter module that is used by this function to obtain how much microseconds are passed since the board has been powered up. Therefore, by calling micros() two times, the first one before the execution of a cryptographic primitive, and the other one immediately after the same primitive, gave us two different values. By

subtracting the former from the latter we have obtained the execution time of the chosen cryptographic primitive. This result was printed on Arduino IDE serial monitor of our PC thanks to Serial.print() Arduino library function.

We have summarized the results obtained by repeating this process for all NIST algorithms' encrypt and decrypt procedures and for all main security primitives of our protocol in table 6.1 and 6.3. A brief recap is shown in table 6.2. For the average value we referred to the algorithm that comes more close to the calculated average value, that is 20517 ms for the encrypt function and 20801 for the decrypt function.

Algorithm	Encrypt time (µ s)	Decrypt time (µ s)
ACE	18979	18984
ASCON	547	550
COMET	623	639
DryGASCON	5852	5872
Elephant	170477	170479
ESTATE	936	938
ForkAE	9295	14993
GIFT-COFB	5006	5011
Gimli	2605	2617
Grain-128AEAD	54718	54680
HYENA	23437	23435
ISAP	50038	50039
KNOT	969	970
LOTUS-AEAD	34180	34185
mixFeed	2946	2945
ORANGE	44032	44030
Oribatida	102845	102854
PHOTON-Beetle	46544	46577
Pyjamask	3723	3727
Romulus	9298	9304
SAEAES	294	298
Saturnin	1716	1719
SKINNY-AEAD	12891	16175
SPARKLE	771	762
SPIX	11521	11531
SpoC	5687	5686
Spook	1591	1605
Subterranean 2.0	11304	11316
SUNDAE-GIFT	8223	8242
TinyJambu	390	395
WAGE	11583	11576
Xoodyak	3526	3528

Table 6.1: NIST results of execution time test

	Encrypt time (µ s)	Decrypt time (µ s)
`Minimum`	SAEAES (294)	SAEAES (298)
`Average`	ACE (18979)	ACE (18984)
`Maximum`	Elephant (170477)	Elephant (170479)

Table 6.2: Summary of NIST results showing minimum, average and maximum results from the test.

Function	Execution time (µ s)
`certificate validation check`	1700583
`shared key derivation`	562932
`set symmetric block cipher key`	97
`message encryption`	227
`message decryption`	426
`average message sending time`	27856

Table 6.3: ECDSA-ECDH protocol results of execution time test

As expected, the asymmetric primitives of our protocol take more time to complete than all NIST algorithms encryption and decryption procedures. On the other hand, it can be seen that the time taken to execute the slowest primitive of our protocol, which is the one in charge of verifying the validity of a received certificate, takes around ten times more time than Elephant encryption procedure, which is the slowest among all NIST algorithms. Although it cannot be considered as lightweight as NIST algorithms, it is important to notice that both encryption procedures, which are responsible for the certificate validation check and the shared key derivation steps, will take place just when a programmer and an ICD establish their first communication session. Once a common shared secret is possessed by both devices they can start a secure communication without executing any asymmetric primitive.

As can be also noted from the tables AES encryption and decryption

primitives performed during our protocol are faster than most of NIST algorithms. This means that, after the small delay of the first communication setup, our protocol will take less time to exchange messages. The last aspect than is important to consider is if the increase in the time to execute asymmetric procedures will have a negative impact during emergencies. The two slowest asymmetric primitives of our protocol take less than 2.5 seconds total to be completed by an ARM Cortex M0+, which is an energy-efficient processor on our Arduino board that simulates ICD's computing capability limitation. Taking 2.5 seconds more will not cause any major issue during emergencies, especially if we consider that using only symmetric procedures and storing a secret key inside the ICD would require the emergency staff to get in contact with the patient's hospital to obtain such key, an operation that will take much more time.

6.2 RAM USAGE

Another important element to consider while evaluating the technical feasibility of a solution based on asymmetric cryptography on resource-limited devices is the RAM occupation that is needed to execute such primitives. Obviously, using complex data structures or numerous temporary variables would make both the heap and the stack occupied memory of the device grow, this, in turn, could possibly lead to memory collisions, resulting in errors or in the altering of the device correct functionality. As previously stated, ICDs have a limited amount of free RAM memory that can be used for non-medical operations, while most of their memory, about three quarters, is used to store ECG measurements. Therefore, it is important that a solution designed to secure ICD's telemetry takes also into consideration the amount of RAM necessary to realize it. This experimental evaluation was done, once again, using our Arduino boards for all algorithms, in order to produce comparable results.

In order to measure the RAM occupation, during the runtime execution of our protocol primitives and NIST algorithms, Arduino library alone was not enough. In order to do so, we have imported another library, called MemoryFree, from the Arduino Playground website. Inside this library, the function freeMemory() has been particularly useful for us. When this function gets called during the program execution it returns the amount of free memory space, in bytes, that is present between the heap and the stack pointers. By calling this function multiple times inside the libraries of all NIST algorithms and of our protocol we have been able to monitor precisely the behavior of each primitive. By subtracting the amount of free RAM measured before the execution of a specific primitive and the lowest value obtained during the execution of it we managed to identify the maximum RAM occupation, in bytes, during the runtime execution of all primitives. One important issue that we had to face while measuring the RAM usage was the fact that our Arduino board automatically allocated 255 bytes of heap space. This space was allocated before every algorithm execution and therefore not counted in our previously described calculations. Some NIST algorithms occupied some or all this free space by dynamically allocating variables, by using malloc() functions, leading to less precise results. In order to solve this problem we have occupied all this automatically allocated space before every algorithm execution so that all algorithms were forced to allocate free RAM space for their dynamic variables. In this way, we obtained results that are balanced between algorithms that required the use of dynamic variables and the others which did not.

The results of such measurements are illustrated in Table 6.4 and 6.6. They were printed on Arduino IDE serial monitor of our PC, in a similar way to what we did to measure the execution time in the previous section. A brief recap is shown in table 6.5. For the average value we referred to the algorithm that comes more close to the calculated average value, that is 784 bytes for the encrypt function and 806 bytes for the decrypt function.

Algorithm	Encrypt RAM (bytes)	Decrypt RAM (bytes)
ACE	644	652
ASCON	428	452
COMET	972	1028
DryGASCON	676	684
Elephant	1612	1612
ESTATE	748	748
ForkAE	804	788
GIFT-COFB	460	460
Gimli	372	388
Grain-128AEAD	1004	1208
HYENA	1980	1964
ISAP	668	692
KNOT	684	684
LOTUS-AEAD	1660	1644
mixFeed	788	788
ORANGE	588	596
Oribatida	1212	1220
PHOTON-Beetle	544	584
Pyjamask	844	844
Romulus	796	812
SAEAES	532	540
Saturnin	764	772
SKINNY-AEAD	740	780
SPARKLE	548	556
SPIX	612	620
SpoC	572	580
Spook	684	708
Subterranean 2.0	1076	1108
SUNDAE-GIFT	488	648
TinyJambu	332	340
WAGE	568	576
Xoodyak	708	732

Table 6.4: NIST results of RAM usage test

	Encrypt RAM (bytes)	Decrypt RAM (bytes)
`Minimum`	TinyJambu (332)	TinyJambu (340)
`Average`	Saturnin (764)	Saturnin (772)
`Maximum`	HYENA (1980)	HYENA (1964)

Table 6.5: Summary of NIST results showing minimum, average and maximum results from the test.

Function	RAM Usage (bytes)
`certificate validation check`	1448
`shared key derivation`	752
`set symmetric block cipher key`	64
`message encryption`	88
`message decryption`	136
`whole protocol`	6548

Table 6.6: ECDSA-ECDH protocol results of RAM usage test

From the tables, it can be seen that the RAM usage of the asymmetric cryptographic primitives of our protocol is in line with the one of NIST algorithms. Our protocol's most memory consuming operation occupied 1448 bytes and it is very close to most NIST algorithms and, in some cases, even lower. All other primitives of our protocol occupied even less RAM and are, therefore, even more equipable to NIST algorithms.

It is important to notice that the whole protocol, including BLE communication functionalities, that where necessary to exchange messages between the two Arduino boards, occupied less than 7 KB of memory of the 29067 available bytes present in the Arduino board. This result is really promising since the amount of RAM space of our chosen board resembles the one of the oldest ICD models, while also considering the fact that most of it should be used for other functions. Most of the occupied RAM was used for global variables and for BLE related

functionalities, not to execute asymmetric primitives. Obviously, current ICDs are already implementing communication with external devices, therefore the only RAM additional usage that we should consider is the one that is required to execute these asymmetric primitives.

6.3 BATTERY CONSUMPTION

The last aspect we evaluated to examine the feasibility of an approach based on ECDSA and ECDH is battery consumption. As we stated before, ICDs' batteries are meant to last as long as possible in patients' body, since replacing the battery means to put the patient under surgery. Therefore, implanted devices have a very energy-efficient hardware and are studied so that they only activate certain modules when necessary. Of course, execute complex and highly consuming operations on these devices is not recommended. Because of that, it is important to be sure that this approach does not consume too much battery, so we need to check the consumption. We decided to measure the instantaneous current detected during the execution of the most important functions of our protocol.

To achieve this we had to build a suitable hardware setup. We powered the Arduino MKR Wifi 1010 boards using an ICR18650 Li-Ion battery for each board, soldering a LiPo battery connector to the onboard port. This way we used the USB port just to upload the sketch to execute. Then we added an INA219 current/power sensor for each board and an Arduino Uno board to read the data from the sensors. We chose the Arduino Uno due to laboratory availability and because we needed a board easy to program to perform data reading operations. In Figure 6.1 is shown the electric scheme of the circuit.

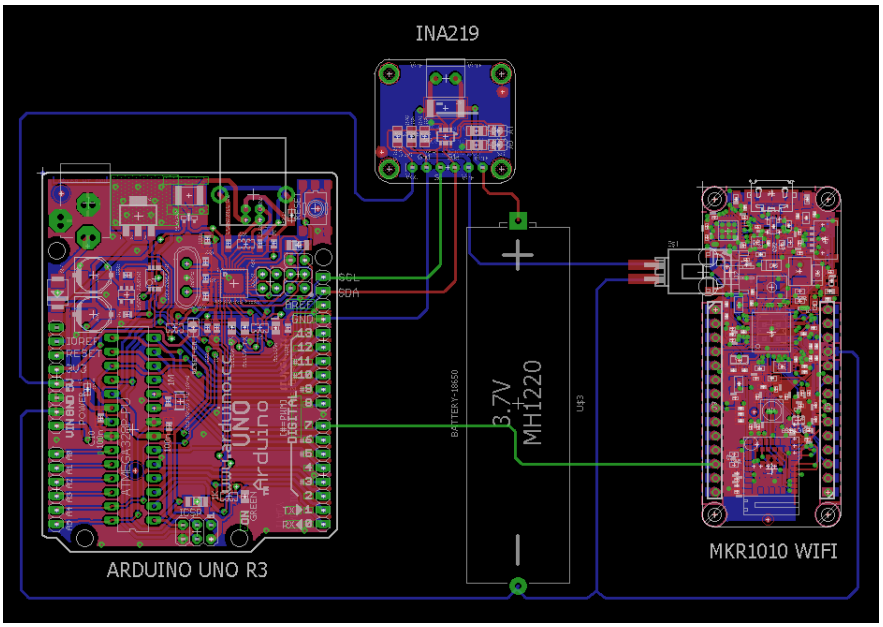


Figure 6.1: Battery test electric scheme showing Arduino Uno and MKR1010 Wifi, the Ina219 sensor and the Li-Ion battery and how they are connected

The INA219 sensor is connected in series between the Li-Ion battery and the MKR board, in order to read the current absorbed by the board during the code execution. To read the power consumption from the sensor we used the Arduino Uno board connected to the PC to avoid adding power consumption to the MKR board. We connected the pin 3 of each MKR board to the pin 7 and 8 of the Uno board, to connect them. In the sketch uploaded in the MKR boards, we added a couple of digitalWrite() functions to set the pin 3 as high and low to start and stop the measurements. Then, in the sketch uploaded on the Uno board, we wrote the code so that the board would read the consumptions from the sensors only when the pin connected to the MKR board was set as high. In this way, we could be able to measure precisely the current during the execution of only the functions we were interested in. Obviously, all components are connected to the same ground. In Figure 6.2 is shown a photo of the whole setup.

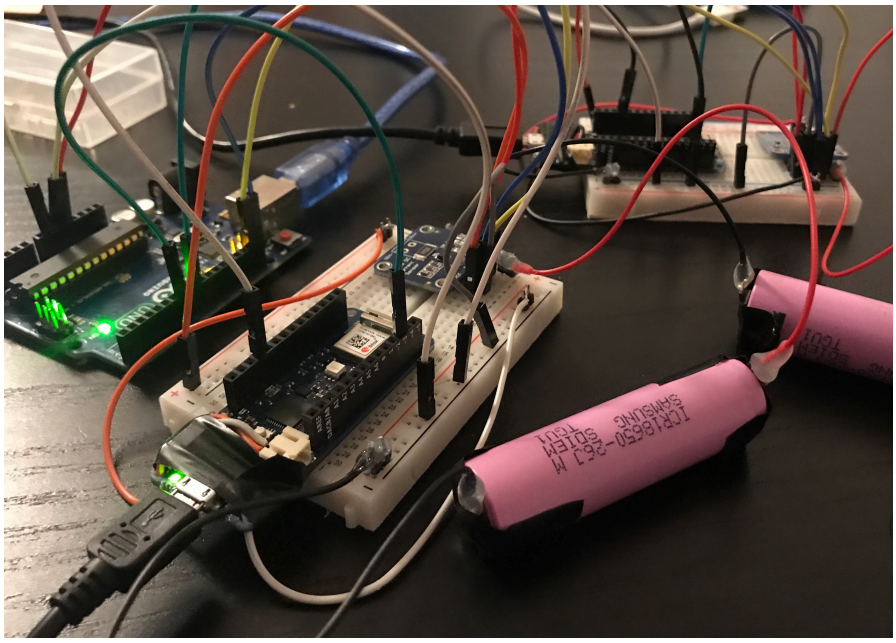


Figure 6.2: Battery test setup photo taken during the measurements showing all the components

The Arduino Uno board and the two INA219 sensors communicate using Inter Integrated Circuit (I2C) serial communication protocol. We used the Serial Data (SDA) and Serial Clock (SCL) pins to connect sensors and board. We could read all the measurements on our PC because they were printed on the Ardunio IDE serial monitor. All the measurements were done while the boards were executing the protocol using the BLE connection, this way we could have a realistic idea of the consumption of an ICD using this protocol. So, we measured the consumption of central and peripheral at the same time, also to understand if there was a difference and eventually how big, of consumption between the two devices. The results of the measurements on the NIST algorithms are presented in Table 6.7, while the results of the proposed approach in 6.9. A brief recap is shown in table 6.8. For the average value, we referred to the algorithm that comes closest to the calculated average value, which is 52 mA for the encrypt function and 49.5 mA for the decrypt function.

Algorithm	Encrypt current (mA)	Decrypt current (mA)
ACE	38.1	37.95
ASCON	39.2	39.1
COMET	38.9	39
DryGASCON	37.5	37.6
Elephant	42.9	88.1
ESTATE	38.9	38.8
ForkAE	42.4	44.3
GIFT-COFB	37.3	37.3
Gimli	39.3	58.8
Grain-128AEAD	51.9	48.5
HYENA	56.9	38.9
ISAP	51.9	51.3
KNOT	38.8	38.8
LOTUS-AEAD	52	49.6
mixFeed	51.3	39.1
ORANGE	53.85	38.8
Oribatida	48.9	45.1
PHOTON-Beetle	38.2	41.1
Pyjamask	54.4	60
Romulus	36.9	36.9
SAEAES	59.2	58.1
Saturnin	56.2	38.5
SKINNY-AEAD	37.2	36.9
SPARKLE	61.1	60.8
SPIX	37.4	37.4
SpoC	60.43	41.2
Spook	49.6	48.9
Subterranean 2.0	47.6	57.8
SUNDAE-GIFT	55.3	60.8
TinyJambu	41.2	59.3
WAGE	56.6	49.8
Xoodyak	47.6	38.6

Table 6.7: NIST results of battery consumption test

	Encrypt current (mA)	Decrypt current (mA)
`Minimum`	GIFT-COFB (37.3)	GIFT-COFB (37.3)
`Average`	LOTUS-AEAD (52)	LOTUS-AEAD (49.6)
`Maximum`	SPARKLE (61.1)	SPARKLE (60.8)

Table 6.8: Summary of NIST results showing minimum, average and maximum results from the test.

Function	Current (mA)
`certificate validation check`	46.65
`shared key derivation`	49.89
`set symmetric block cipher key`	42.36
`message encryption`	40.45
`message decryption`	44.24

Table 6.9: ECDSA-ECDH protocol results of battery consumption test

Looking at the tables it can be seen that the results of the tests on the ECDSA-ECDH based protocol are consistent with the ones of the same test conducted on the NIST call algorithms. The task that absorbs more current of the proposed approach is the shared key derivation, with 49.89 mA. From the tables of the NIST algorithms result, we can see that the results of our protocol are really close to most of the algorithms that compete to become the new lightweight cryptography standard. It is important to consider that all these measurements have been done during the BLE connection, to understand how the protocol would perform if fully implemented, BLE connectivity included. Moreover, BLE was needed to handle the connectivity between the two Arduino boards. Measuring the battery consumption of our approach without active BLE showed that the execution of the primitives took less than 20 mA. This means that the added consumption due to the asymmetric cryptography primitives is not severe. Thus, the results show that the

proposed approach has promising results when compared with the NIST algorithms, that we consider as a benchmark.

6.4 RESULT DISCUSSION

The measurements of RAM usage, execution time, and battery consumption of our ECDSA and ECDH based protocol presented encouraging results. We tried to simulate in the best possible way the resource-constrained structure of the ICDs using a suitable Arduino board.
Starting from the processor we used, the ARM Cortex M0+, is a bit less performant than the processor that is generally mounted on ICDs, the ARM Cortex M3. Nonetheless, looking at the results of the execution time tests we obtained, it is possible to observe that the cryptographic functions we chose for our approach are executed in a small amount of time. This is an important result because when dealing with emergencies, it is crucial that the communication based on this protocol is established as soon as possible. In this way, the medical staff could access the implanted device and intervene possibly saving the patient's life rapidly without wasting any time.
Then, as we stated in previous sections, ICDs have small free RAM available, because most of it is used for treatment-related functions, like monitoring the patient's status or saving ECG reading. The free RAM on the Arduino board we chose is 29 Kbytes (3 Kbytes are used for Arduino variables) that is quite close to the free RAM we studied that ICDs have. The maximum quantity of RAM occupied during the execution of our approach is approximately 5 Kbytes. So, this means that a protocol based on ECDSA and ECDH would be feasibly implementable on an ICD in terms of occupied RAM.
The battery consumption is in line with the current uptake of the NIST algorithms. Moreover, considering that the asymmetric steps of the protocol could be performed just once per new external device that connects with the ICD, the impact on the battery longevity of the

device is limited. Since, as we stated before, we consider the performances of these algorithms as benchmarks to evaluate other solutions, we can say that this approach is also lightweight in terms of battery consumption.

To conclude, since the tests we performed on the new protocol based on ECDSA and ECDH using Curve25519 show encouraging results, it is worth to further investigate it. We think that exploring the potential of this approach could give the research new points of view on the security of ICD's telemetry.

7

LIMITATIONS AND FUTURE WORK

In this chapter, we will present the main limitations that we encountered during our work and propose some future works that could be interesting to improve the research in this direction.

One of the most relevant problems we had to manage was the difficulty to gather information about ICDs. Since the market is very competitive, manufacturers do not disclose technical data about their devices to avoid competitors getting to them. Therefore, even if we got in contact with manufacturers, due to their company policies it was not possible for us to retrieve any useful information for our research from them. Certainly, having ICD samples or technical information about these devices directly from the manufacturers could have been very helpful to build a more accurate model to use as test-bench.

For this reason, we have chosen an Arduino board to test all the NIST call algorithms and our proposed approach in the simplest way possible. This is an initial study on an ECDSA-ECDH based communication protocol for ICDs, our focus is to analyze the feasibility of this approach. We have decided to make our experiments easily replicable and we have used a simple language and programming environment to modify and adapt our code during the research. Probably, the performances of our protocol can be improved by writing the code using Assembly language. However, as previously stated, we needed to adapt the code easily and fast during our research. Using Assembly wouldn't make it possible for us to achieve that result since it's a complex language to work with and needs much more attention to be modified once written.

Since the results of our protocol are promising, we think it could be interesting to further investigate this research path. We have identified

two different paths that the research could follow.

The first one consists in adapting this protocol to different IMDs that have similar technical characteristics to the ones of ICDs. Obviously, it is fundamental that these devices possess a telemetry module, in order to establish a communication with a programmer, sufficient RAM size, and battery capacity to handle the cryptographic primitives involved in the protocol. Obviously, some modifications and adjustments could be needed.

The second path is to study a new algorithm specifically tailored for ICDs, and, if possible, also for other IMDs, to serve the goal of granting a secure communication with external devices. We have used a combination of ECDSA, ECDH and AES. It could be interesting to understand if having just one algorithm that executes the same functions of these three combined will give some advantages in terms of performances.

8

CONCLUSIONS

In this thesis, we have examined in detail the context of ICDs. In particular, the security issues caused by the introduction of the long-range wireless telemetry advancement on these devices. After having analyzed their structural features and the threats they are exposed to, we have examined the currently available security solutions for the ICD domain. We have considered both the advantages and disadvantages of all the most popular solutions.

In particular, we have considered the NIST call for cryptographic algorithms. The algorithms proposed in this call are based on symmetric cryptography and are competing to become the new standard for lightweight cryptography on resource-constrained devices. We have shown the limitations of an approach only based on this kind of cryptography in the ICDs domain. While this is certainly a suitable solution in terms of performances, it does not guarantee easy access during emergency situations. This aspect, as we have pointed out in this work, cannot be ignored when considering the life-saving role of ICDs. Despite this, we have decided to evaluate the performances of these algorithms in order to obtain a baseline that can be used as a mean of comparison to evaluate the lightweightness of other solutions that are more suitable for ICD context. In particular, we have tested the execution time, RAM usage, and battery consumption of all the NIST algorithms on an Arduino MKR Wifi 1010 board. We chose this board because has many technological characteristics that are quite similar to the internal components of ICDs and because makes it possible to easily replicate the experiments. The experiments that we have performed gave us an interval of values that we used as testbench to eventually define how close other solutions, specifically tailored for

the ICD domain, are to a lightweight approach.

After having considered the current state of the art approaches in securing ICD's wireless telemetry, we came to the conclusion that an approach based on asymmetric cryptography had been neglected by researchers, probably because it was considered too resource consuming for this kind of devices. Consequently, analyzing the positive aspects that an asymmetric cryptography-based approach could bring in the ICD domain, we have decided to further investigate if a solution based on that approach could be feasible. During our secure communication protocol design phase, we have selected ECC combined with Curve25519 in order to make the protocol as fast as possible while also reaching a higher security level with fewer resource requirements. Considering the context of ICDs and their structural components, we decided to use ECDSA to manage the authentication part, ECDH for the key generation step and AES for the secure message exchange.

After our design phase, we have implemented this communication protocol on two Arduino MKR Wifi 1010 boards in order to be able to perform on our proposed solution the same tests that we have conducted on the NIST algorithms. In this way, we have both evaluated the feasibility of this approach on resource-constrained devices and compared the performances of our solution with the results given by the NIST lightweight algorithms. Our protocol is based on a PKI where a CA signs the credentials of an external device that wants to connect to an ICD. This step is necessary to produce a valid certificate that demonstrates the legitimacy of the external entity. The ICD is then responsible to verify the validity of the received certificate. In case this step is successful, the communication protocol will proceed and the two communicating parties will derive a shared key that will be used to encrypt and decrypt all the following exchanged messages.

After having implemented this protocol, we performed experiments to evaluate it. The obtained results are promising since the values of the three analyzed metrics are consistent with the results given by the NIST algorithms. All asymmetric cryptography based primitives present

in our communication protocol execute in a relatively small amount of time (less than 2 seconds). The RAM occupied while executing these primitives is perfectly compatible with the available memory space mounted on ICDs. Finally, the battery consumption is in line with the performances of the NIST algorithms. All these considerations show that although our approach is not as lightweight as the NIST algorithms, it is still a valid solution for the ICD domain. Therefore, since the experimental results of our ECDSA-ECDH based protocol are encouraging, we think that pursuing this line of research could be interesting.

The main limitation that we had to face was the difficulty to gather technical information about ICDs from manufacturers. The market of these devices is very competitive and the manufacturers' company policies do not allow to distribute samples or to disclose any kind of data related to those devices. For this reason, we have decided to use Arduino boards to implement and test our protocol and NIST algorithms writing sketches in C language. To obtain a more performant protocol we could have used Assembly, but this would have made much more complex and less replicable our experimental phase.

As future work we suggest to further examine the potential of asymmetric cryptography in this context, adapting this protocol to other IMDs with telemetry functionality. Moreover, it could also be interesting to analyze the advantages of developing a new asymmetric protocol that covers all the steps of our protocol instead of using a combination of ECDSA, ECDH, and AES.

BIBLIOGRAPHY

[1] D. Halperin et al. "Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses." In: *2008 IEEE Symposium on Security and Privacy (sp 2008)*. 2008, pp. 129–142.

[2] Chunxiao Li, A. Raghunathan, and N. K. Jha. "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*. 2011, pp. 150–156.

[3] Eduard Marın Fàbregas. "Security and Privacy of Implantable Medical Devices." In: (2018).

[4] *Visia AF Reference Manual*. http://manuals.medtronic.com/content/dam/emanuals-/crdm/CONTRIB_235960.pdf. Medtronic, 2016.

[5] *VISIA AF VR Model DVAB1D4 Product specifications*. https://www.medtronic.com/us-en/healthcare-professionals/products/cardiac-rhythm/icd- systems/visia-af.html. Medtronic, 2016.

[6] *Technical manual Eluna HF(-T)*. https://manuals.biotronik.com/emanuals-professionals/?country=US&product=Pacemaker/Eluna/ElunaHF_T_US. Biotronik, 2017.

[7] *Technical manual Lumax 640 / 740 ProMRI*. https://manuals.biotronik.com/emanuals-professionals/?country=GB&product=Icd/Lumax/Lumax640_740_ProMri. Biotronik, 2015.

[8] *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-Ana lysis-and-Forecasts-2017-2023.html. 2017.

[9] Eduard Marin et al. "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. Los Angeles, California, USA: Association for Computing Machinery, 2016, pp. 226–236. ISBN: 9781450347716. DOI: 10.1145/2991079.2991094. URL: https://doi.org/10.1145/2991079.2991094.

[10] *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.

[11] *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.

[12] Hans Abrahamson. *Implantable medical device adapted for radio frequency telemetry with frequency hopping*. Tech. rep. US9704385B2. St Jude Medical AB, July 2017.

[13] Ngamboé Mikaela et al. "Risk Assessment of Cyber Attacks on Telemetry Enabled Cardiac Implantable Electronic Devices (CIED)." In: *arXiv preprint arXiv:1904.11908* (2019).

[14] Masoud Rostami, Ari Juels, and Farinaz Koushanfar. "Heart-to-Heart (H2H): Authentication for Implanted Medical Devices." In: *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*. CCS '13. Berlin, Germany: Association for Computing Machinery, 2013, pp. 1099–1112. ISBN: 9781450324779. DOI: 10.1145/2508859.2516658. URL: https://doi.org/10.1145/2508859.2516658.

[15] Carmen Camara, Pedro Peris-Lopez, and Juan E. Tapiador. "Security and privacy issues in implantable medical devices: A comprehensive survey." In: *Journal of Biomedical Informatics* 55 (2015), pp. 272–289. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2015.04.007. URL: http://www.sciencedirect.com/science/article/pii/S153204641500074X.

[16] Carsten W Israel and S Serge Barold. "Pacemaker systems as implantable cardiac rhythm monitors." In: *American Journal of Cardiology* 88.4 (2001), pp. 442–445.

[17] Boriani Giuseppe et al. "Battery longevity of implantable cardio verter-defibrillators and cardiac resynchronization therapy defibrillators: Technical, clinical and economic aspects. An expert review paper from EHRA." In: *EP Europace* 20 (May 2018). DOI: 10.1093/europace/euy066.

[18] *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. Boston Scientific, 2019.

[19] Maurizio Landolina et al. "Longevity of implantable cardioverter-defibrillators for cardiac resynchronization therapy in current clinical practice: An analysis according to influencing factors, device generation, and manufacturer." In: *Europace* 17 (May 2015). DOI: 10.1093/europace/euv109.

[20] M.J. Ebert et al. "4 - Biomaterials for pacemakers, defibrillators and neurostimulators." In: *Biomaterials for Artificial Organs*. Ed. by Michael Lysaght and Thomas J. Webster. Woodhead Publishing Series in Biomaterials. Woodhead Publishing, 2011, pp. 81–112. ISBN: 978-1-84569-653-5. DOI: https://doi.org/10.1533/9780857090843.1.81. URL: http://www.sciencedirect.com/science/article/pii/B9781845696535500042.

[21] Younghyun Kim et al. "Vibration-based secure side channel for medical devices." In: *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE. 2015, pp. 1–6.

[22] Kasper Bonne Rasmussen et al. "Proximity-based access control for implantable medical devices." In: *Proceedings of the 16th ACM conference on Computer and communications security*. 2009, pp. 410–419.

[23] Shyamnath Gollakota et al. "They Can Hear Your Heartbeats: Non-Invasive Security for Implantable Medical Devices." In: vol. 41. Aug. 2011, pp. 2–13. DOI: 10.1145/2018436.2018438.

[24] X. Hei and X. Du. "Biometric-based two-level secure access control for Implantable Medical Devices during emergencies." In: *2011 Proceedings IEEE INFOCOM*. 2011, pp. 346–350.

[25] C. C. Y. Poon, Yuan-Ting Zhang, and Shu-Di Bao. "A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health." In: *IEEE Communications Magazine* 44.4 (2006), pp. 73–81.

[26] Chunqiang Hu et al. "OPFKA: Secure and efficient Ordered-Physiological-Feature-based key agreement for wireless Body Area Networks." In: May 2013, pp. 2274–2282. ISBN: 978-1-4673-5944-3. DOI: 10.1109/INFCOM.2013.6567031.

[27] Krishna Venkatasubramanian, Ayan Banerjee, and Sandeep Gupta. "PSKA: Usable and Secure Key Agreement Scheme for Body Area Networks." In: *IEEE transactions on information technology in biomedicine : a publication of the IEEE Engineering in Medicine and Biology Society* 14 (Dec. 2009), pp. 60–8. DOI: 10.1109/TITB.2009.2037617.

[28] *Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process*. https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf. NIST, 2018.

[29] Ueli Maurer. "Modelling a public-key infrastructure." In: *Computer Security — ESORICS 96*. Ed. by Elisa Bertino et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, pp. 325–350. ISBN: 978-3-540-70675-5.

[30] Zheng Guo, T. Okuyama, and M. R. Finley. "A New Trust Model for PKI Interoperability." In: *Joint International Conference on Autonomic and Autonomous Systems and International Conference on Networking and Services - (icas-isns'05)*. 2005, pp. 37–37.

[31] PUB FIPS. "186-4, Digital Signature Standard (DSS), National Institute of Standards and Technology, US Department of Commerce, November 2001." In: *Link in: https://nvlpubs.nist.gov/nist pubs/FIPS/NIST.FIPS.186-4. pdf* (2013).

[32] Martin E Hellman, Bailey W Diffie, and Ralph C Merkle. *Cryptographic apparatus and method*. US Patent 4,200,770. Apr. 1980.

[33] Martin Hellman. "An overview of public key cryptography." In: *IEEE Communications Society Magazine* 16.6 (1978), pp. 24–32.

[34] *SAM D21 Family datasheet*. http://ww1.microchip.com/downloads /en/DeviceDoc-/SAM_D21_DA1_Family_Data%20Sheet_ DS 40001882E.pdf. Microchip, 2018.

[35] Neal Koblitz, Alfred Menezes, and Scott Vanstone. "The state of elliptic curve cryptography." In: *Designs, codes and cryptography* 19.2-3 (2000), pp. 173–193.

[36] Daniel J. Bernstein. "Curve25519: New Diffie-Hellman Speed Records." In: *Public Key Cryptography - PKC 2006*. Ed. by Moti Yung et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 207–228. ISBN: 978-3-540-33852-9.

[37] *Cortex-M0+ Technical Reference Manual Documentation*. https:// developer.arm.com/docs/ddi0484/c. ARM, 2012.

[38] *Cortex-M3 Technical Reference Manual Documentation*. https// developer.arm.com/docs/ddi0337/h. ARM, 2010.

[39] *ARMv6-M Architecture Reference Manual*. https://developer.arm.com/docs/ddi0419/c/armv6-m-architecture-reference-manual. ARM, 2010.

[40] Rhys Weatherley. "Arduino Cryptography Library." In: *Source code, available online at http://github. com/rweather/arduinolibs* (2018).

[41] Sean Turner, Adam Langley, and Mike Hamburg. *Elliptic curves for security*. Tech. rep. IETF RFC 7748, January, 2016.