On the (in)security of the Latest Generation Implantable Cardiac Defibrillators and How to Secure Them

Eduard Marin
ESAT-COSIC and iMinds
KU Leuven, Belgium
eduard.marin@esat.kuleuven.be

Dave Singelée
ESAT-COSIC and iMinds
KU Leuven, Belgium
dave.singelee@esat.kuleuven.be

Flavio D. Garcia
School of Computer Science
University of Birmingham, UK
f.garcia@bham.ac.uk

Tom Chothia
School of Computer Science
University of Birmingham, UK
t.p.chothia@cs.bham.ac.uk

Rik Willems
Cardiology, University Hospital
Gasthuisberg
Leuven, Belgium
rik.willems@uzleuven.be

Bart Preneel
ESAT-COSIC and iMinds
KU Leuven, Belgium
bart.preneel@esat.kuleuven.be

ABSTRACT

Implantable Medical Devices (IMDs) typically use proprietary protocols with no or limited security to wirelessly communicate with a device programmer. These protocols enable doctors to carry out critical functions, such as changing the IMD's therapy or collecting telemetry data, without having to perform surgery on the patient. In this paper, we fully reverse-engineer the proprietary communication protocol between a device programmer and the latest generation of a widely used Implantable Cardioverter Defibrillator (ICD) which communicate over a long-range RF channel (from two to five meters). For this we follow a black-box reverse-engineering approach and use inexpensive Commercial Off-The-Shelf (COTS) equipment. We demonstrate that reverse-engineering is feasible by a weak adversary who has limited resources and capabilities without physical access to the devices. Our analysis of the proprietary protocol results in the identification of several protocol and implementation weaknesses. Unlike previous studies, which found no security measures, this article discovers the first known attempt to obfuscate the data that is transmitted over the air. Furthermore, we conduct privacy and Denial-of-Service (DoS) attacks and give evidence of other attacks that can compromise the patient's safety. All these attacks can be performed without needing to be in close proximity to the patient. We validate that our findings apply to (at least) 10 types of ICDs that are currently on the market. Finally, we propose several practical short- and long-term countermeasures to mitigate or prevent existing vulnerabilities.

1. INTRODUCTION

Implantable Medical Devices (IMDs) such as pacemakers and Implantable Cardioverter Defibrillators (ICDs) are used to monitor and help control abnormal heart rhythms. ICDs are battery-powered devices that deliver electric shocks to the patient's heart if the heartbeat is too fast. Some ICDs can also act as a pacemaker and give tiny electrical shocks if the heartbeat is too slow. ICDs have evolved over three generations. The first generation (or the oldest) do not have any wireless interface and hence do not allow reprogramming once the ICD is implanted. The second and third generation enable wireless communication with external devices including device programmers and base stations. Device programmers are used by medical personnel to wirelessly modify the ICD's settings or collect telemetry data, whereas base stations, installed in the patients' home, allow remote monitoring by gathering telemetry data from the ICD and sending this data to the hospital. Both device programmers and base stations have a programming head that activates the ICD's wireless interface when it is placed above the implantation site (the patient's chest) for a few seconds.

The second generation of ICDs supports wireless communication between the programming head and the ICD only over a short-range communication channel (less than 10 cm). In the third generation (the latest), the programming head is first used over the short-range communication channel to activate the long-range communication link of the ICD. This process is illustrated in Fig 1. Both devices can then communicate with each other over a long-range communication channel (from two to five meters), not requiring the use of the programming head anymore, unless the session expires.

While these advances bring substantial clinical benefits to patients, new security and privacy threats also emerge, specially due to the wireless communication between these devices. Adversaries may eavesdrop the wireless channel to learn sensitive patient information, or even worse, send malicious messages to the ICD. The consequences of these attacks can be fatal for patients as these messages can contain commands to deliver a shock or to disable a therapy.

Our contribution: This paper presents the first reverse engineering and security analysis of the proprietary long-range communication protocol between the device programmer and the latest generation of ICDs. For the reverse engineering we use a black-box approach and inexpensive Commercial Off-The-Shelf (COTS) equipment. This task is not trivial since it was first necessary to find the symbol rate from the waveform of the signals sent by the devices in

ACSAC '16, December 05-09, 2016, Los Angeles, CA, USA
© 2016 ACM. ISBN 978-1-4503-4771-6/16/12...$15.00
DOI: http://dx.doi.org/10.1145/2991079.2991094

order to demodulate the captured messages correctly. We show that for proprietary protocols on which we had no prior knowledge or documentation, reverse-engineering is possible by a weak adversary without even needing to have physical access to the devices. Our second contribution consists of demonstrating several attacks that can compromise the ICD's availability and the patient's privacy. We give evidence that replay and spoofing attacks are possible as well. To evaluate the feasibility of these attacks, we describe several ways to circumvent the short-range communication step, which requires being close to the patient, and perform session hijacking. We validated that our findings apply to (at least) 10 different ICD models. Our third contribution is the proposal of several short- and long-term measures to mitigate or solve the existing vulnerabilities in the latest generation of ICDs including a novel key agreement protocol which we formally verified using ProVerif.

Disclosure of results: In accordance with the principle of responsible disclosure, we have contacted and discussed our findings with the manufacturer before disclosure. Given the sensitive nature of our work, we omitted some of the obtained results to avoid easy replication of the attacks.

Paper outline: The remainder of this paper is organised as follows. Section 1 gives an overview of related work and shows our laboratory setup. Section 2 explains the process of reverse-engineering the proprietary protocol between the device programmer and the ICD. Section 3 describes several strategies to circumvent the short-range communication, which requires close proximity to the patient. Section 4 shows the protocol weaknesses and implementation flaws whereas practical and effective short- and long-term countermeasures to mitigate or solve these vulnerabilities are presented in Section 5. Finally, Section 6 gives concluding remarks.

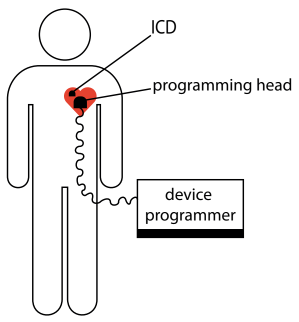

Figure 1: ICD activation procedure.

1.1 Related work

1.1.1 Software radio-based attacks on IMDs

Several papers have demonstrated that IMDs often lack strong security mechanisms, which makes them vulnerable to different types of remote attacks. Hei et al. showed a simple yet effective attack where adversaries force the IMD to respond to their messages, which reduces the battery life of the IMD [13]. Halperin et al. analysed the proprietary protocol between the device programmer and a second generation ICD to communicate over the short-range communication channel [12]. As no security mechanisms were found, they were able to carry out several software radio-based attacks just by replaying past transmissions sent by the legitimate device programmer. Similar attacks can also be performed on an insulin pump, as shown by Li et al. [7]. Marin et al. fully reverse-engineered the proprietary protocol between all devices in a wireless insulin pump system, and extended the attacks of Li et al. [16]. Unlike the work by Halperin et al. [12], which focused on the short-range communication (less than 10 cm), we analyse the proprietary protocol between the device programmer and a latest generation of ICD over long-range communication (from two to five meters).

1.1.2 Countermeasures

Various countermeasures have been proposed to solve the vulnerabilities found in IMDs. Gollakota et al. presented the "shield", an external device that acts as a proxy between the device programmer and the ICD. The shield jams the messages to/from the IMD to prevent others from decoding them, while still being able to successfully decode them itself [10]. Although this solution mitigates some of the existing problems, it does not protect against adversaries who can transmit malicious messages with much more power than the shield. Tippenhauer et al. demonstrated that the shield does not provide confidentiality as a MIMO eavesdropper could cancel out the interference produced by the shield and then recover the messages sent by the devices [21]. Xu et al. introduced a wearable device, also known as "IMDGuard", which does not only work as a proxy but also performs an authentication process on the ICD's behalf [22]. But Rostami et al. found that the "IMDGuard" is vulnerable to a Man-In-The-Middle (MITM) attack which reduces its effective key length from 129 bits to 86 bits [19]. Rostami et al. presented Heart-to-Heart (H2H), a commitment-scheme-based pairing protocol through which the device programmer authenticates to the IMD without needing to share any prior secrets [20]. H2H implements a novel access-control policy called "touch-to-access" that ensures access to the IMD by any device programmer that can make physical contact with the patient and measure his heart rate. However, Marin et al. found that the H2H is vulnerable to a reflection and a MITM attack [15].

Another line of research relies on exchanging a cryptographic key between the device programmer and the IMD via an auxiliary or Out-Of-Band (OOB) channel. Halperin et al. proposed a zero-power authentication that uses an RFID tag in combination with a piezo-element for audio-based key distribution. However, Halevi et al. demonstrated the feasibility of eavesdropping the audio transmissions of the piezo element [11]. Rasmussen et al. proposed an access control scheme based on ultrasonic distance bounding which enables the IMD to grant access to its resources to only a device programmer that is in its close proximity [18]. However, this typically requires dedicated analog hardware, which makes the solution expensive to integrate in resource-constrained devices like IMDs. Another proposal is to use a Body-Coupled Communication (BCC) channel. Yet, Li et al. showed that remote eavesdropping on a BCC channel is possible with a very sensitive antenna [7]. In this paper, we present practical and effective countermeasures that can be divided into two groups: short-term and long-term measures. The former do not require any modification on the ICDs and hence may be immediately adopted whereas the latter can be implemented in future generations of ICDs.

1.2 Laboratory setup

Our laboratory setup comprises available Commercial Off-The-Shelf (COTS) hardware including an Universal Serial Radio Peripheral (USRP) [4], a data acquisition system (DAQ) [1] and a few antennas, as shown in Fig 2. In addition, we have the following medical devices: a device programmer, a base station and several ICD models of the latest generation. For our experiments, we created a receiver and a transmitter programs using LabVIEW [3]. The first step of our black-box reverse-engineering approach is to eavesdrop the wireless channel and capture the messages exchanged between the device programmer and the ICD. We then analyse the messages to discover its format, and study how the messages are exchanged between the devices, i.e. the protocol state-machine. Subsequently, we are able to create and send our own messages to the ICD by means of the USRP, the antenna and our transmitter program. To better evaluate the feasibility of these attacks, we also study the ICD activation procedure. For this we use a DAQ and an antenna to intercept the messages exchanged over the short-range communication channel.

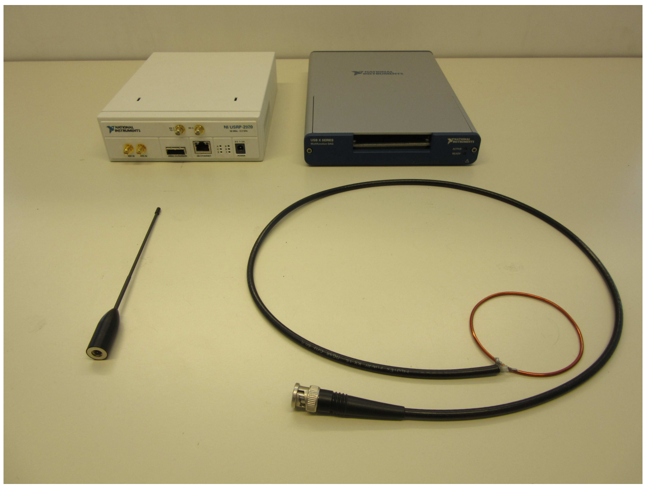

Figure 2: Laboratory setup. At the top, from left to right, are our USRP and the DAQ. Our antennas are shown at the bottom.

2. INTERCEPTING THE WIRELESS TRANSMISSIONS

Several articles [12, 7, 16] have already pointed out that IMD manufacturers often rely on hiding the protocol specifications to provide security. This is commonly known as security-by-obscurity. Proprietary protocols typically offer very limited or no security guarantees and have been broken via different reverse-engineering techniques. This paper analyses the proprietary protocol between device programmers and the latest generation of ICDs to communicate over a long-range channel. Instead of opening the devices to get their firmware for the purpose of reverse-engineering the protocol, we follow a *black-box approach*. A similar approach has been used in other articles [8, 9]. Our black-box approach consists of giving some inputs to the devices and then inferring information by looking at their outputs, i.e. the produced messages. In our work we study the feasibility of reverse-engineering the proprietary protocol by a weak adversary who has limited resources and capabilities. Through meticulous analysis of these messages, we can infer the message format and the protocol state-machine. Our black-box approach, which is a labour-intensive process, is more challenging yet more realistic than other existing techniques, as it assumes a weak attacker who can intercept the messages sent wirelessly using a USRP and an antenna, but cannot have physical access to the devices. We will now summarise our approach and main findings.

2.1 Wireless communication parameters

Transmission frequency: The ICD and the device programmer's programming head first communicate over the short-range communication channel (between 30-300 kHz)[1]. After the ICD is activated, both devices communicate over the long-range communication channel using the MICS[2] band (402-405 MHz). The transmission frequency of the devices can be obtained through their Federal Communications Commission (FCC) ID [2].

Modulation: By examining the signals sent by the devices both in the time and frequency domain, we found that the device programmer and the ICD use distinct modulations to transmit their data. In particular, the transmissions from the device programmer to the ICD use a Frequency Shift Keying (FSK) modulation, whereas a Differential Phase Shift Keying (DPSK) modulation is used in the transmissions from the ICD to the device programmer [17].

Symbol rate: Due to the modulations being used, discovering how many symbols (i.e. modulated bits) are sent in each message simply by looking at signal's waveform is a challenging problem.

To estimate the symbol rate, we created a Matlab program that uses the Hilbert transform to obtain the instantaneous frequency of the signal. A key observation is that by demodulating the signals using an FM receiver and looking at the demodulated waveforms, we found that the message sent by the device programmer to request telemetry data is always identical. This message is sent continuously to the ICD when no operation is performed. The first step is to intercept several of these messages and store their waveforms in a file. Our program takes these waveforms as inputs and produces a graph that shows where the frequency shifts occur, i.e. where each symbol starts and ends.

Fig 3 shows the instantaneous frequency of the device programmer's signal. The symbol rate can then be obtained by computing the inverse of the difference between the times where two abrupt peaks occur. However, instead of giving only one symbol rate value, this approach gives a small range of possible values. Therefore, the second step was to create another program that performs a sweep over all possible symbol rate values within this range, increasing the symbol rate by one symbol each time. For each iteration, our program demodulates several of the messages previously captured, and then checks whether the demodulated bits are equal for all the messages. This allows us to find the symbol rate being used by the devices, as the correct symbol rate is the one for which no bit errors are produced.

[1]We do not specify the exact transmission frequency as this may implicitly reveal the manufacturer's identity.

[2]Medical Implant Communications Service.

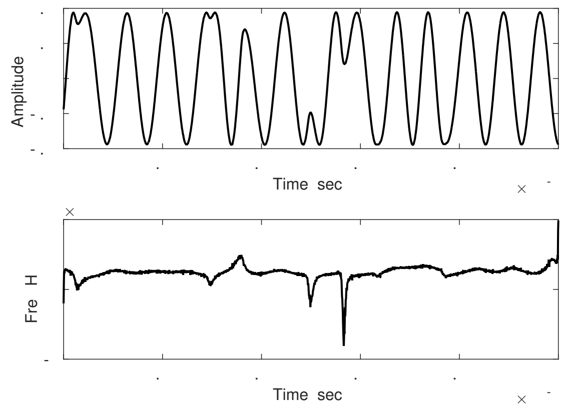

Figure 3: Symbol rate estimation based on the Hilbert transform. In the top chart, the waveform of the signal transmitted by the device programmer. In the bottom chart, the instantaneous frequency of the device programmer's signal.

2.2 Reverse-engineering the long-range communication protocol

In this section, we show how to reverse-engineer the proprietary protocol between the device programmer and the ICD to communicate over the long-range channel. We first activate the ICD and put the device in "interrogation" mode. More details on how adversaries can activate the ICD are given in Section 3.

We found that all messages have a common Start-of-Frame (SoF) that consists of a series of alternating "1s" and "0s" sent consecutively to indicate the presence of an incoming message. This is followed by a preamble sequence which indicates that the information bits are about to begin. To distinguish the messages sent by the device programmer and the ones sent by the ICD, we placed the device programmer close to our USRP while keeping the ICD further away, thus getting more power from the device programmer. Unlike the messages sent by the ICD, which only use one preamble sequence, two preamble sequences can be used in the messages sent by the device programmer; a specific sequence or its inverse. Messages from device programmers have a fixed length and include a 3-bit End-of-Frame (EoF) sequence whilst ICDs send messages with three possible lengths that do not contain any EoF.

2.2.1 *Transmissions device programmer - ICD*

We intercepted the messages sent from the device programmer to the ICD while carrying out different operations (e.g. changing the therapy settings). For the sake of simplicity, we will focus only on the messages sent from the device programmer to the ICD in order to change the patient's name. This process typically includes 16 messages and is always composed of two differentiated groups of messages separated by a long message sent by the ICD, as shown in Figure 4. The former group includes messages 1-8, whereas the second group includes the 9th up to the 16th message.

We found that the 16 messages have a x-bit sequence that denotes the message type. In each of the two messages' groups, there are three possible message types independently of the operation being conducted: (i) an opening message, (ii) intermediate messages and (iii) a closing message. We determined that the first and the nineth messages contain the Serial Number (SN) of the device programmer. The ICD SN appears only in the messages sent by the device programmer when the ICD is in the "no telemetry" mode. In other words, this is sent only if the ICD loses the connection with the device programmer during an ongoing session. Each SN is represented by a 24-bit sequence. Subsequently, we observed that there is a y-bit sequence to indicate the message number within the first group of messages. This field is kept static in the second group of messages. Since the message number field only has a short length and eight messages are sent by the device programmer within the first group of messages, this field is reset frequently. By capturing and analysing the 16 messages sent by the device programmer in several consecutive iterations within a reprogramming session, we found two short counters, in the first and ninth message respectively. Both counters are increased every time an operation is performed and are reset when a new reprogramming session is established.

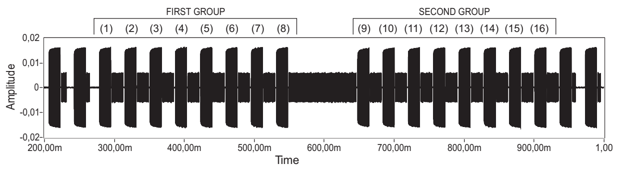

Figure 4: Messages exchanged between the device programmer and the ICD while changing the patient's name.

We discovered that there is a 16-bit sequence at the end of each message that seems to be random and varies depending on the headers and data being sent. This lead us to think that a checksum, such as a Cyclic Redundancy Code (CRC), is used. To validate our hypothesis, we took the GCD of several of these messages (in polynomial form), and discovered that the CRC-16-CCITT is being used [14]. Other mechanisms, such as repetition codes, are used to help the ICD detecting bit errors. We noted that if the patient's name contains less than 14 characters, it is sent three times, otherwise it is sent twice within the first group of messages. Fig 5 shows the device programmer's message format.

2.2.2 *Data whitening*

We carried out a series of experiments to find how the data is encoded in the message. For this we focused on the messages sent by the device programmer when changing the patient's name.

The first experiment consisted on finding where the letters are within the messages and see how many bits are used to represent each letter. In particular, we changed the patient's name to "A", "AA", "AAA", "AAAA" and "AAAAA", respectively. We then intercepted the messages and compared them with the ones sent by the device programmer when the patient's name field is left empty. We found that the first four letters are sent within the first message and that each letter is represented by an 8-bit sequence. In addition, we observed that there is no unique pattern to represent the "A". The next step was to reprogram the patient's

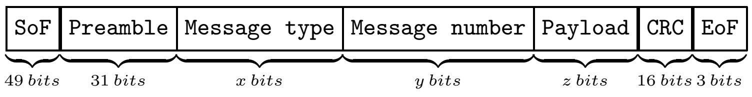

Figure 5: Device programmer's message format. The exact bit lengths are not shown.

name while keeping a specific letter in more than one position. We modified the patient's name to "AAAA", "ABAB" and "ACAC", respectively. This experiment demonstrated that the way how each letter is encoded depends on its position within the patient's name. In other words, an "A" in the first position is always represented in the same way but differently to an "A" in another position. By comparing the 8-bit sequences of the "A", "B" and "C" in the second and the fourth position, respectively, we noticed that the Hamming distance between the sequences is constant. This allowed us to conclude that the data is XORed with an output sequence from a Linear Feedback Shift Register (LFSR) (see Figure 6)[3]. The vendor states that this is a data whitening operation to prevent long strings of "1s" and "0s" in the data. However, this operation could also serve as data obfuscation.

In our experiments, we were able to recover the LFSR sequence by intercepting the messages sent by the device programmer when the patient's name is left empty (i.e. only spaces). We then computed the XOR between the first message sent by the device programmer when changing the patient's name to "AAAA" and the LFSR sequence. After performing this operation, we found a unique pattern to represent each of the four "As" of the patient's name. This pattern turned out to be identical to its ASCII representation. Our experiments reveal that this LFSR sequence is constant throughout sessions. Moreover, we found that the LFSR sequence is the same for all the ICDs we studied in our experiments. We validated our findings in 10 different ICD models, and concluded that all models use this technique to encode the data that is sent over the air.

```
(a) "A"            00101011 10111101 00011010 01010001
(b) "AA"           00101011 11101000 00011010 01010001
(c) "AAA"          00101011 11101000 01101001 01010001

(d) "AAAA"         00101011 11101000 01101001 01111101
(e)  LFSR seq  ⊕   01001010 10001001 00001000 00011100
                   -----------------------------------
(f)  ASCII         01100001 01100001 01100001 01100001
                      A        A        A        A
```

Figure 6: LSFR XOR operation.

2.2.3 *Transmissions ICD - device programmer*

We intercepted and examined several messages transmitted by the ICD. We did not find any header that is specific for the ICD type or any field that denotes the ICD type. We verified that all messages sent from the ICD to the device programmer use the same LFSR sequence as the one previously described. We noted that all the messages have the ICD SN. In contrast, the ICD includes the device programmer's SN only in replies to no telemetry messages. We discovered that the ICD messages have a counter that helps the device programmer to sort the incoming messages or detect message losses. We observed that most of the information bits seem random. Since the ICD's leads are no longer connected to the patient's heart and are very sensitive to low-frequency changes, we noticed that they were measuring the ambient noise and treating it as random telemetry data. To investigate where the telemetry data is within the message, instead of injecting our own signal to the ICD's leads, we introduced the ICD's leads into a Faraday cage to isolate them. We then captured several messages sent by the ICD, and noted that they have a more constant pattern which is no longer random. Furthermore, we identified several bit sequences that are common to the three types of ICD message regardless of the operation being performed. These sequences are most likely used for synchronization purposes. Finally, we discovered that, similarly to the messages sent by the device programmer to the ICD, all messages have a 16-bit checksum, which is based on the standard CRC-16-CCITT.

[3]The data and the LFSR sequence that are shown in Figure 6 are not the real ones.

3. HOW TO ACTIVATE THE ICD?

Before exploiting our findings to carry out attacks, we first need to activate the ICD. To demonstrate the feasibility of these attacks, we describe several ways to bypass the current activation procedure, which requires almost physical contact with the patient and is carried out over a short-range communication channel. For simplicity, in the next sections we often use the term "external device" to denote both device programmers and base stations.

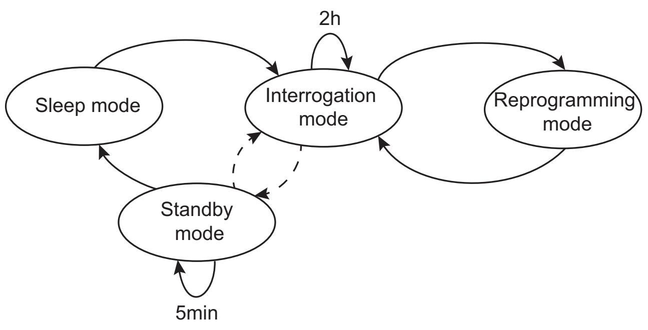

Figure 7: ICD modes of operation.

Our experiments show that the ICD can operate in five different modes: "sleep", "interrogation", "reprogramming", "no-telemetry" or "standby". In the rest of this paper, we will not discuss the "no-telemetry" mode further since this mode was not relevant for our experiments. Figure 7 gives an overview of the modes. Initially, the ICD is in a "sleep" mode in which it occasionally activates the wireless interface to check whether there is an incoming message sent by a device programmer over the short-range communication channel. Once the ICD is activated, it remains in "interrogation" mode where it continuously sends telemetry data to the device programmer over the long-range communication channel. If no reprogramming operation is performed by the doctor, the ICD is in the "interrogation" mode for two hours. If the doctor modifies the ICD settings within this two-hour window, the ICD switches to "reprogramming" mode for a few seconds and then goes back to the "interrogation" mode, where is kept active for two hours. When the session expires (after two hours), we observed that, instead of immediately switching to "sleep" mode, the ICD goes first to "standby" mode. We will explain the "standby" mode more in detail later in this section.

We will now describe four possible ways to send malicious messages to the ICD, depending on whether the ICD is active, in "standby" or in "sleep" mode.

Exploit an active session: Intuitively, adversaries could attempt to hijack an ongoing session between the external device and the ICD to send malicious commands to the ICD. This is a challenging task since this requires the adversary to be in close proximity to the patient (e.g. in the hospital). Furthermore, adversaries need to send the malicious commands to the ICD while having to block the messages sent by the genuine external device. To masquerade their attacks, adversaries may also send fake telemetry data to the genuine external device to avoid that the doctor/patient notices that the ICD is no longer communicating with it.

Standby mode: We discovered that the ICDs do not immediately switch to "sleep" mode after finishing an ongoing session with the device programmer, but they all remain in a "standby" mode for five minutes. This is a safety feature but also has security consequences.

While being in "standby" mode, any device programmer can activate the ICD again by sending a specific message over the long-range communication channel. This message turns out to be identical for all ICDs. We also found this weakness in the case where the ICD is activated by means of the base station. In that case, the ICD is active for five minutes only if the session with the base station is not terminated correctly.

We were able to impersonate the device programmer and successfully send this message to the ICD to keep it alive. For our experiments, we used the transmitter port of our USRP to emulate the device programmer's behaviour and the receiver port of our USRP to capture this signal and the response sent by the ICD. To distinguish between the messages sent by our USRP and the responses sent by the ICD, we placed the ICD close to the receiver port of our USRP while keeping the transmitter port of our USRP further away, thus getting more power from the ICD. This attack is illustrated in Figure 8. Therefore, adversaries could wait until the session between the device programmer and the ICD finishes and then repeatedly send this message to the ICD. This could be used to drain the ICD's battery, or even worse, to extend the time window as long as needed to send as many malicious messages as required to compromise the patient's safety.

Wake up the ICD from "sleep" mode: We noted that the device programmer's programming head is magnetic. To eliminate the possibility that a magnet is needed to bootstrap the communication with the ICD, we conducted an experiment where we placed a magnet near the ICD. The result of this experiment showed that the magnet alone can-

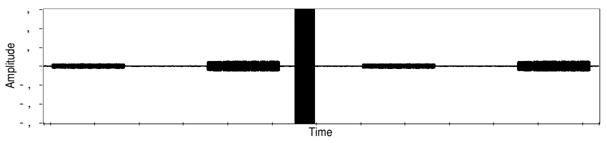

Figure 8: Messages sent to the ICD while the ICD is in "standby" mode in order to activate it. From left to right, two messages sent (with different gain) from our USRP to wake up the ICD, the response of the ICD and two messages sent by the USRP.

not activate the ICD's wireless interface. The next step was to investigate which data is exchanged between the devices before the long-range communication starts. For this we studied the short-range communication between the device programmer and the ICD, focusing on the messages sent by the device programmer.

We used our DAQ and an antenna to capture the messages sent by the device programmer at 30-300 kHz. Every time a new session is established, the device programmer sends three messages to the ICD via the programming head. Following the same steps as those described in the previous section, we were able to unveil the wireless communication parameters being used. In particular, we found that the messages sent by the device programmer are modulated using a FSK and encoded under Non-Return-to-Zero Inverted (NRZI). In NRZI, a '1' is represented by a transition of the voltage level, whereas a '0' has no voltage transition. We also determined that the symbol rate is 12500 symbols/second.

We created a LabVIEW program to intercept and demodulate the three messages transmitted by the device programmer's programming head. We noted that the first message is always identical regardless of the ICD being used, whilst the second and third message vary depending on the ICD's SN. The other headers and information bits within the second and third message are kept constant, making the short-range communication vulnerable to replay attacks. Thus, adversaries need to eavesdrop the wireless channel only once to intercept the three messages sent by the device programmer. Adversaries could then carry a backpack with all the necessary equipment and re-send these messages to the ICD when the patient is in a crowded place (e.g. the public transportation) where adversaries can be relatively close to the patient and still go unnoticed.

Using legitimate external devices: Alternatively, adversaries can also use any legitimate external device to conduct the attacks. Unlike device programmers, which are big, heavy and cannot be hidden easily, base stations are inexpensive, portable and can be easily purchased. Therefore, one possibility is to use a legitimate base station to carry out these attacks. However, a base station by itself cannot send commands to reprogram the ICD. In our experiments, we show that adversaries can use any legitimate base station to activate the ICD. Since the ICD remains in "standby" mode if the session with the base station is not terminated correctly, adversaries can simply carry the base station in a backpack and turn it off before the communication with the ICD ends in order to keep the ICD alive. Adversaries can then use their own equipment to send malicious messages to the ICD over the long-range communication.

4. EXISTING VULNERABILITIES

In this section we will briefly summarise the weaknesses we found after fully reverse-engineering the proprietary protocol. These weaknesses can result in several types of active and passive software radio-based attacks. We want to stress that adversaries could use sophisticated equipment and directional antennas to extend the distance from which they can carry out attacks by several orders of magnitude.

4.1 Privacy attacks

Our analysis of the proprietary protocol between the device programmer and one model of the latest generation of ICDs reveals that the messages sent over the air are "obfuscated" using an LFSR sequence. This LFSR sequence is the same for all models that we studied.

The messages exchanged between the devices include patient private sensitive information such as personal data (e.g. his name or medical history) or telemetry data. Clearly, the way they use the LFSR sequence to obfuscate the data can result in serious patients' privacy breaches. Passive adversaries can compromise the patient's privacy just by eavesdropping the wireless channel while there is an ongoing communication. However, this attack typically requires the adversaries to wait until the devices exchange this data. This limitation can be overcome by active adversaries who can additionally send malicious messages to the ICD to request this data.

By intercepting the messages sent by ICDs and looking at their unique SN, adversaries could track, locate or identify patients. For example, adversaries could install beacons in strategic locations (e.g. the train station or the hospital) to infer the patients' movement pattern based on the signals transmitted by their ICDs. This could reveal their addresses, the places they often go, and other potential sensitive information. Furthermore, the messages sent between the devices during a reprogramming session may allow adversaries to infer the patient's treatment or the therapy details. Telemetry data, which is sent continuously by the ICD when it is active, could reveal the patient's health state. Overall, it is clear that the consequences of all these attacks can be severe for patients.

4.2 Denial-of-Service (DoS) attacks

As shown in the previous sections, ICDs can operate in four distinct modes: "sleep", "interrogation", "reprogramming" and "standby".

Intuitively, the ICD should immediately switch to "sleep" mode when the communication session with the device programmer finishes or when it expires after two hours with no reprogramming operation. However, we discovered that, after the ICD has been activated, it remains in "standby" mode for five minutes, where it can be put in the "interrogation" mode again if it receives a specific message. This message turns out to be identical for all ICDs and is sent over the long-range communication channel. In other words, there is no need for being in close proximity with the patient to activate his ICD. This is an important implementation flaw that makes these devices vulnerable to DoS attacks. The purpose of these attacks is to keep the ICD alive by continuously sending this message over the long-range communication, which could drastically reduce the ICD battery life. Yet, this also opens up the door for adversaries to perform other types of attacks more easily, as they can send this message to extend the five minute window as many times as needed to send malicious messages to the ICD without requiring being close to the patient.

4.3 Spoofing and replay attacks

After fully reverse-engineering the proprietary communication protocol between the device programmer and the ICD, we were able to fully document the message format in use. Our results show that there is no mechanism to prevent replay attacks; the counters found in the first and ninth message are reset every time a new session is established or after a relatively small number of operations. Without even knowing the protocol specifications, adversaries could successfully perform replay attacks just by re-sending past transmissions sent by the legitimate device programmer. In addition, the protocol does not provide any means to check the integrity and authenticity of the messages. Thus, it is possible to perform spoofing attacks, which allow adversaries to send arbitrary commands to the ICD.

5. COUNTERMEASURES

In this section, we present practical and effective countermeasures to mitigate/solve the vulnerabilities found in the previous sections. We divide our countermeasures into two groups: short-term measures and long-term measures. The former group could be deployed immediately to mitigate some of the existing security issues in already-implanted ICDs, whereas the latter group require minor modifications on the devices and hence could be integrated into future generations of ICDs.

5.1 Short-term measures

5.1.1 Jamming the wireless channel

As shown in the previous section, adversaries can take advantage of the time the ICD is in "standby" mode to carry out a DoS attack, or even worse, to extend the time they can send malicious messages to the ICD. Thus, our first countermeasure consists of adding a "shutdown" command in all external devices so that they continuously jam the wireless channel while the ICD is in "standby" mode. A more efficient solution is to jam the wireless channel only if an adversary is detected. This is also known as reactive jamming. Several articles have already used friendly-jamming as a defensive mechanism.

One possible drawback of our countermeasure is that it could interrupt the ongoing communications between other legitimate devices. We leverage on the fact that the patient typically has his ICD being reprogrammed/interrogated in isolated controlled locations; either in the doctor's office or in the patient's home. This clearly reduces the risks of jamming other ongoing communications. Another downside of our countermeasure is that it works only if the patient stays close to the external device for five minutes while his ICD is in "standby" mode. Due to that the ICD listens to all MICS channels while being in "standby" mode, external devices need to be equipped with several antennas to simultaneously jam all MICS channels.

5.2 Long-term measures

5.2.1 *Adding a shutdown command in the ICDs*

Instead of relying on friendly-jamming to prevent adversaries from sending malicious messages to the ICD, our second countermeasure is based on modifying both external devices and ICDs to include a “shutdown” message. This way, the external device can send the “shutdown” message to the ICD before they finish the communication to ensure that the ICD goes directly to “sleep” mode. Even though this countermeasure does not completely solve the existing vulnerabilities, this makes it more difficult for adversaries to send malicious messages to the ICD.

5.2.2 *Key agreement protocol*

As a long term improvement, Halperin et al. proposed adding standard symmetric key authentication and encryption between the ICD and the programmer. For this they proposed to have the master key on every device programmer (stored in tamper-resistant hardware) and diversified keys in the ICDs. This setup is clearly a significant improvement over existing systems. Yet, having the master key stored in every device programmer is latent risk. If the tamper-resistant hardware of a single device programmer is ever compromised, then there is no way to revoke the keys and every patient with an implant will be exposed indefinitely, or until the IMD is replaced.

Another alternative is to store the master key in the cloud, in order to limit its distribution to a single instance, and have the device programmers online. But this is not a viable option as the device programmers are required to operate (in case of emergency) at all times, including during Internet or cloud provider outages.

In this paper we propose a middle ground between these two approaches: a semi-offline protocol. We leverage on the fact that both IMDs and device programmers have a precise internal clock which is synchronised at every communication session. This clock allows the IMD to keep a log file with all critical events and the time when they occurred. Let G_1 and G_2 be two multiplicative groups of prime order q. Furthermore, let $e\colon G_1 \times G_1 \to G_2$ be a bilinear map satisfying:

Bilinearity $\forall g, h \in G_1, \forall a, b \in \mathbb{Z}_q^*, e(g^a, h^b) = e(g, h)^{ab}$.

Non-degeneracy $\forall g \in G_1, g \neq 0$ implies that $e(g, g)$ is a generator of G_2.

Computability e can be computed in polynomial time.

Let $H_1, H_2\colon \{0,1\}^* \to G_1$ be two different cryptographic hash functions satisfying standard security requirements.
When the system is initialised, the key generation centre generates the system master secret key `msk` which is stored securely at the back office, and is never shared with anyone. Let ID be the IMD identities domain which is assumed to be disjoint to the time domain. Each IMD $id \in \text{ID}$ stores a diversified key $H_2(id)^{\mathtt{msk}}$ which is provided at manufacture time (all the operations here are done modulo q). Device programmers receive a temporal key $H_1(t)^{\mathtt{msk}}$ which is valid to derive all IMD’s diversified keys but only for the time period t. This period of time can be anything, but for the sake of example let us take this time period to be three months. In that case, every three months, the device programmer (or a health-care employee) needs to contact the device manufacturer to obtain the key for the next quarter $H_1(t+1)^{\mathtt{msk}}$ which is sent over a secure channel. In this way, if a device programmer is lost, stolen or tampered with, this can be reported to the device manufacturer and then this device will no longer receive key updates, rendering it useless. Any key material which may have been extracted from the device becomes obsolete after (at most) three months and then the system goes back to a secure state. Fig 9 describes our semi-offline key agreement protocol in detail.

This protocol requires one bilinear pairing computation on the IMD which is expensive, but this only needs to happen once every three months. On a daily basis, IMD and device programmer simply run a standard symmetric key authentication protocol like the one proposed by Halperin et al., using the agreed key $e(H_1(t), H_2(id))^{\mathtt{msk}}$. Note that this protocol does not provide key confirmation, but this can be easily achieved by the symmetric key authentication protocol as it is the case in Halperin et al.

5.2.3 *Formal analysis of our protocol*

To provide some level of assurance for our protocol we model and analyse it using the applied pi-calculus [5] and the checking tool ProVerif [6]. The applied pi-calculus allows us to model protocols, using primitives such as input, output, new name generation and parallel composition. It also allows us to define functions and equations that can be used to model a range of cryptographic primitives. The ProVerif tool can ensure secrecy and correspondence properties for an arbitrary number of runs of a protocol using a automated theorem proving method, however the tool is not guaranteed to terminate and may report false attacks.

We model an idealised version of bilinear pairings using functions and equations in the applied pi-calculus, i.e., we define the functions $\mathbf{power}(x, y)$, $\mathbf{prod}(x, y)$ and $\mathbf{e}(x, y)$ to represent x^y, xy and the bilinear map $e(x, y)$. We would then like to define the equation:

$$\text{equation } \mathbf{e}(\mathbf{power}(a, x), \mathbf{power}(b, y)) = \\ \mathbf{power}(\mathbf{e}(a, b), \mathbf{prod}(x, y))$$

However, such an equation causes ProVerif’s proof tactics to enter any infinite loop. Therefore, we introduce an auxiliary function to represent the right hand side of this equation, i.e., we define $\mathbf{powere}(a,\ b, \mathbf{prod}(x, y))$ to represent $e(a, b)^{xy}$. We note that this gives an abstract model of bilinear pairings that does not include any number theoretic attacks, such as factoring the product, inverse powers, or low entropy secrets.

Our model is made up of four processes: **Programmer**, which models the programmer protocol role, **IMD** which models the IMD, **CompromisedReader**, which publicly broadcasts a programmers diversified key for a time period different to the one used by **Programmer**, and **CompromisedUnAuthIMD** which models a compromised IMD by publicly broadcasting the diversified key for a medical device that is not one accepted by the programmer.

At the end of their run the **Programmer** and **IMD** processes broadcast a secret value encrypted with the key they have established. We test the system to see if it is possible for an attacker to learn this secret, which would mean they had successfully established a key with the IMD or Programmer. The full model is given in Appendix A.

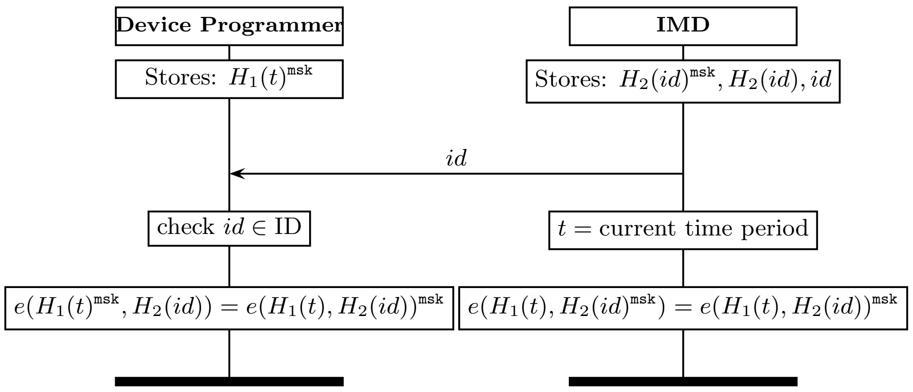
Device Programmer
Stores: $H_1(t)^{\mathtt{msk}}$
IMD
Stores: $H_2(id)^{\mathtt{msk}}, H_2(id), id$
id
check $id \in$ ID
$t =$ current time period
$e(H_1(t)^{\mathtt{msk}}, H_2(id)) = e(H_1(t), H_2(id))^{\mathtt{msk}}$
$e(H_1(t), H_2(id)^{\mathtt{msk}}) = e(H_1(t), H_2(id))^{\mathtt{msk}}$

Figure 9: A semi-offline key agreement protocol for IMDs.

Testing this model in ProVerif we find that it does indeed keep the keys secret. This means that only an IMD with an ID accepted by the programmer, and a programmer with a diversified key for the right time period, can set up and learn keys, even if there are an arbitrary number of old compromised programmers and IMDs.

To check for redundancy in our protocol, and to see what kinds of attacks ProVerif can find, we experiment with possible simplifications. We first try removing the IMD identity check in the programmer (the "**if imdID=id then**" line of the model), in this case ProVerif finds an attack in which the attacker uses a diversified key from an old, compromised IMD. If we also remove the **CompromisedUnAuthIMD** process, we find that the protocol is then safe. This tells us that this identity check by the programmer is only needed to stop attacks using compromised IMDs, if we decided to discount compromised IMDs in our attacker model we would not need this check.

As a second test, we tried using a single hash function, rather than two. In this case, ProVerif finds an attack that lets an attacker impersonate an IMD: The attacker sends the old time stamp from a compromised programmer (t') in place of the id to the targeted programmer. This leads to the key programmer using the key $e(H(t)^{\mathtt{msk}}, H(t'))$, however from the compromised programmer the attacker can learn $H(t')^{\mathtt{msk}}$ and so construct the matching key $e(H(t), H(t')^{\mathtt{msk}})$. This suggests that our protocols use of two hash functions is a sensible precaution to avoid attacks based on confusing times and identities. These additional checks gived us increase confidence that the analysis method we use can find attacks, when present, and that our protocol is not unnecessarily complex.

5.2.4 *Differentiating between device programmers and base stations*

There is an important distinction to make between device programmers and base stations as the former are in a much more controlled environment than the latter. Device programmers are not sold to anyone: they are available only to accredited health-care professionals and institutions whereas base stations are much more available at the patients home. Some base stations are sometimes available to purchase on auction sites such as Ebay and is relatively easy to get hold of one. But their usage is also very different. Base stations only need read access to the IMD in order to forward telemetry information to the relevant health-care practitioner. Therefore, it makes sense to have different keys for each of these devices which provide different access levels. In this way, if the key of a base station gets compromised for a period of time, this still represents a potential privacy violation but at least it is not life threatening.

6. CONCLUSIONS

In this work we have analysed the security and privacy properties of the latest generation of ICDs. For this we fully reverse-engineered the proprietary protocol between the ICD and the device programmer using commercial and inexpensive equipment. We want to emphasise that reverse-engineering was possible by only using a black-box approach. Our results demonstrated that security-by-obscurity is a dangerous design approach that often conceals negligent designs. Therefore, it is important for the medical industry to migrate from weak proprietary solutions to well-scrutinised security solutions and use them according to the guidelines.

Our work revealed serious protocol and implementation weaknesses on widely used ICDs, which lead to several active and passive software radio-based attacks that we were able to perform in our laboratory. Our first attack consisted on keeping the ICD alive while the ICD is in "standby" mode by repeatedly sending a message over the long-range communication channel. The goal of this attack was to drain the ICD's battery life, or to enlarge this time window to send the necessary malicious messages to compromise the patient's safety. Our second attack aimed at compromising the patient's privacy. For this we leveraged the fact that we were able to recover the LFSR sequence used to "obfuscate" the messages. We discovered that this LFSR sequence is constant throughout sessions and is the same for all ICDs we studied.

We proposed short-term and long-term countermeasures. As a short-term countermeasure, the only solution is to use jamming as a defensive mechanism. As long-term countermeasures, external devices could send a "shutdown" message to the ICD so that the ICD can immediately switch to "sleep" mode after the communication ends. Moreover, we designed and formally verified a semi-offline key agreement protocol between the device programmer and the ICD.

In accordance with the principle of responsible disclosure, we have notified and discussed our findings with the manufacturer before publishing this article.

7. ACKNOWLEDGEMENTS

The authors would like to thank Stefaan Foulon for his support and the anonymous reviewers for their helpful comments. This work was supported in part by the Research Council KU Leuven: C16/15/058 and the Cryptacus COST Action IC1403.

8. REFERENCES

[1] DAQ NI USB-6351. http://www.ni.com.

[2] Federal Communications Commission (FCC) ID. http://www.fcc.gov/encyclopedia/fcc-search-tools.

[3] LabVIEW. http://www.ni.com/labview.

[4] NI USRP-2920. http://www.ni.com.

[5] M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *Symposium on Principles of Programming Languages (POPL)*, 2001.

[6] B. Blanchet, B. Smyth, and V. Cheval. ProVerif 1.88: Automatic cryptographic protocol verifier, user manual and tutorial, 2013.

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

[8] F. D. Garcia, G. Koning Gans, R. Muijrers, P. Rossum, R. Verdult, R. W. Schreur, and B. Jacobs. Dismantling mifare classic. In *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ESORICS '08, pages 97–114, Berlin, Heidelberg, 2008. Springer-Verlag.

[9] F. D. Garcia, D. Oswald, T. Kasper, and P. Pavlidès. Lock it and still lose it —on the (in)security of automotive remote keyless entry systems. In *25th USENIX Security Symposium (USENIX Security 16)*, Austin, TX, Aug. 2016. USENIX Association.

[10] S. Gollakota, H. Hassanieh, B. Ransford, D. Katabi, and K. Fu. They Can Hear Your Heartbeats: Non-invasive Security for Implantable Medical Devices. *SIGCOMM Comput. Commun. Rev.*, 41(4):2–13, Aug. 2011.

[11] T. Halevi and N. Saxena. On pairing constrained wireless devices based on secrecy of auxiliary channels: the case of acoustic eavesdropping. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 97–108, 2010.

[12] D. Halperin, T. S. Heydt-Benjamin, B. Ransford, S. S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. H. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *Proceedings of the 29th Annual IEEE Symposium on Security and Privacy*, pages 129–142, May 2008.

[13] X. Hei, X. Du, J. Wu, and F. Hu. Defending resource depletion attacks on implantable medical devices. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5, Dec 2010.

[14] P. Koopman and T. Chakravarty. Cyclic redundancy code (crc) polynomial selection for embedded networks. In *Dependable Systems and Networks, 2004 International Conference on*, pages 145–154, June 2004.

[15] E. Marin, E. Argones Rúa, D. Singelée, and B. Preneel. A survey on physiological-signal-based security for medical devices. Cryptology ePrint Archive, Report 2016/188, 2016. http://eprint.iacr.org/.

[16] E. Marin, D. Singelée, B. Yang, I. Verbauwhede, and B. Preneel. On the feasibility of cryptography for a wireless insulin pump system. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pages 113–120, New York, NY, USA, 2016. ACM.

[17] J. Proakis and M. Salehi. *Digital Communications*. McGraw-Hill higher education. McGraw-Hill Education, 2007.

[18] K. B. Rasmussen, C. Castelluccia, T. S. Heydt-Benjamin, and S. Capkun. Proximity-based access control for implantable medical devices. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 410–419, New York, NY, USA, 2009. ACM.

[19] M. Rostami, W. Burleson, F. Koushanfar, and A. Juels. Balancing security and utility in medical devices? In *The 50th Annual Design Automation Conference 2013, DAC '13, Austin, TX, USA, May 29 - June 07, 2013*, pages 13:1–13:6, 2013.

[20] M. Rostami, A. Juels, and F. Koushanfar. Heart-to-heart (H2H): authentication for implanted medical devices. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013* [20], pages 1099–1112.

[21] N. O. Tippenhauer, L. Malisa, A. Ranganathan, and S. Capkun. On limitations of friendly jamming for confidentiality. In *Security and Privacy (SP), 2013 IEEE Symposium on*, pages 160–173, May 2013.

[22] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li. Imdguard: Securing implantable medical devices with the external wearable guardian. In *INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China*, pages 1862–1870, 2011.

APPENDIX

A. A FORMAL MODEL OF OUR PROPOSED PROTOCOL FROM SECTION 5.2.2

```
(* Secure IMD protocol *)
free c.

(* bilinear pairings *)
fun power/2.  (* power(x,y) = x^y *)
fun powere/3. (* powere(a,b,x) = e(a,b)^x *)
fun prod/2.   (* prod(a,b) = a x b *)
fun e/2.      (* e (a^x,b^y) = e(a,b)^(xy)*)

equation e(power(a,x),power(b,y)) = powere(a,b,prod(x,y)).
equation prod(x,y) = prod(y,x).

data one/0.

(* hashes *)
fun H1/1.
fun H2/1.

(* Shared key cryptography *)
fun senc/2.
reduc sdec(y, senc(y,x)) = x.

private free sec.
private free msk.

(*
Test if the attacker can learn secret
encrypted with the established key
*)
query attacker:sec.
let Programmer = in (c,imdID);
     if imdID=id then
     let rkey = e(rsec,power(H2(imdID),one)) in
     in(c,message);
     out(c,senc(rkey,sec)).

let IMD = let imdkey = e(power(H1(t),one),psec) in
        out(c,id);
     out (c,senc(imdkey,sec)).

let CompromisedReader = new t'; out(c,t');
                        out(c,power(H1(t'),msk)).

let CompromisedUnAuthIMD = new id'; out (c,id');
                           out(c,power(H2(id'),msk)).
process new msk;
        !new t; out(c,t);
        !new id; out(c,id);
    (   let psec = power(H2(id),msk) in !IMD
      | let rsec = power(H1(t),msk) in !Programmer
      | !CompromisedReader | !CompromisedUnAuthIMD )
```