On the Feasibility of Cryptography for a Wireless Insulin Pump System

Eduard Marin, Dave Singelée, Bohan Yang, Ingrid Verbauwhede and Bart Preneel
KU Leuven, ESAT-COSIC and iMinds
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

ABSTRACT
This paper analyses the security and privacy properties of a widely used insulin pump and its peripherals. We eavesdrop the wireless channel using Commercial Off-The-Shelf (COTS) software-based radios to intercept the messages sent between these devices; fully reverse-engineer the wireless communication protocol using a black-box approach; and document the message format and the protocol state-machine in use. The upshot is that no standard cryptographic mechanisms are applied and hence the system is shown to be completely vulnerable to replay and message injection attacks. Furthermore, sensitive patient health-related information is sent unencrypted over the wireless channel.

Motivated by the results of our attacks, we study the feasibility of applying cryptography to protect the data transmitted over the air and prevent unauthorized access to the insulin pump. We present a solution based on AES in combination with an updated message format optimized for energy consumption. We implement our solution on a 16-bit micro-controller and evaluate its security properties and energy requirements. Finally, we discuss potential strategies for further reducing the energy consumption.

Keywords
Cryptography; proprietary wireless communication protocol; software radio-based attacks

1. INTRODUCTION
Wearable and Implantable Medical Devices (IMDs) such as pacemakers, neurostimulators and insulin pumps are currently used to monitor and treat physiological conditions within the patient’s body. For example, patients with diabetes require a precise daily dosage of insulin, combined with a strict schedule for diet, physical activity and insulin injections. The dosage can be automated via an insulin pump. Unlike self-injection, insulin pumps provide more flexibility, tighter control of the patient’s diabetes and better predictability by delivering more precise insulin doses.

 Nowadays, most insulin pumps have a radio that enables wireless communication with various peripherals (e.g. a glucose meter), a more powerful embedded processor and connectivity to a back-end computing infrastructure. While these advances bring substantial clinical benefits, new security and privacy threats emerge. More specifically, the wireless interface of these devices is of particular concern as it could be exploited by adversaries to perform remote attacks; adversaries might eavesdrop the wireless channel to compromise the patient’s privacy, or even worse, send unauthorized commands to the insulin pump.

Contributions. Our first contribution is to show that (at least) two commercial insulin pump models use proprietary wireless communication protocols without any security protection. By fully reverse-engineering the wireless communication protocol, we are able to: (i) discover sensitive information on the patient sent in the clear over the air and (ii) send unauthorized commands to the insulin pump. Our second contribution is a practical case study on the feasibility of using cryptography to secure the wireless channel between these devices. We present a cryptographic AES-based solution with an updated message format optimized for energy consumption, an important requirement for these devices. We evaluate the security properties and energy requirements of our solution. Finally, we implement our solution on a 16-bit micro-controller, similar to the one used in the insulin pump system, and discuss possible ways to further reduce the energy consumption.

Disclosure of results. Our study examines a widely adopted insulin pump system that is being used by many diabetic patients worldwide. Given the sensitive nature of our results, we deliberately omit some details (e.g. model and manufacturer) as they would allow anyone to easily reproduce our attacks on insulin pump systems used by patients.

Paper outline. The remainder of this paper is organized as follows. Section 2 provides an overview of related work. Section 3 describes the devices that are part of the insulin pump system. Section 4 shows the laboratory setup used to eavesdrop the wireless channel and perform the remote attacks. The methodology of how to reverse-engineer the proprietary wireless communication protocol is explained in Section 5. Section 6 shows several software radio-based attacks that are carried out on the insulin pump, whereas a cryptographic solution to secure the wireless link is presented and evaluated in Section 7, particularly focusing on the energy consumption. Section 8 provides a final conclusion on the paper.
2. RELATED WORK

2.1 Attacks on medical devices

Recently, it has been shown that some medical devices are vulnerable to security attacks. Halperin et al. surveyed a wide range of security issues, and explained the need to balance the security and privacy of IMDs with safety and effectiveness [11]. Halperin et al. also performed a security and privacy analysis of a widely used Implantable Cardioverter Defibrillator (ICD). More specifically, they carried out several software radio-based attacks that could compromise both the security and privacy of the patient [12]. Similar attacks were also executed on the wireless channel between an insulin pump and a remote control, as shown by Li et al. [8]. The main focus of their paper was to show different types of attacks that adversaries could perform after fully reverse-engineering the wireless communication protocol. In this paper, we validate and extend their attacks by fully reverse-engineering the wireless communication protocol between all the peripherals of the insulin pump system. This includes the wireless link between: (i) the remote control and the insulin pump, (ii) the glucose meter and the insulin pump, (iii) the glucose sensor and the insulin pump and (iv) the USB stick and the insulin pump. This paper also describes the methodology of how to reverse-engineer the proprietary wireless communication protocol.

2.2 Countermeasures

Researchers have proposed several countermeasures to solve or mitigate the vulnerabilities found on medical devices. Li et al. suggested using rolling codes, as used in garage doors, to protect against unauthorized entities [8]. However, some rolling codes do not offer strong authentication. Another weak point of their proposal is that it assumes that the remote control and the insulin pump share an encryption key, but it does not explain how this key is updated and revoked. The solution proposed by Gollakota et al. consists of introducing an external device, known as “shield”, which acts as a relay between the IMD and the external device programmer [10]. The “shield” has been prototyped for a cardiac device to mitigate some of the security problems, but it has not been tested for wearable medical devices, such as insulin pumps. Anomalous detection is also a promising technique that could be used in combination with other security mechanisms to prevent unauthorized access, as shown by Hei et al. [13]. They proposed to use past glucose trends to detect anomalous behaviours. However, while strict policies can increase the detection rate, they may also result in false alarms. Undoubtedly, cryptography is the only strong approach for securing the data transmitted over the wireless communication and preventing unauthorized access. The use of cryptography, though, is challenging because IMDs are resource-constrained devices that require reduced size, low peak power and a low duty cycle. Key management also presents important challenges in terms of scalability, usability and the capacity to deal with emergency situations [14]. This makes it non-trivial to propose cryptographic solutions that increase the level of security and privacy of the system without jeopardizing the patient’s safety. In this paper, we address this problem by proposing an AES-based solution optimized for the insulin pump system. Furthermore, we investigate how the energy cost of this security solution can be reduced without compromising the security.

3. INSULIN PUMP SYSTEM

This section describes the devices that are part of the insulin pump system (see also Figure 1).

The insulin pump can administer two types of insulin doses: a bolus dose, which is pumped quickly to the patient after meals, or a basal dose, which is continuously delivered at an adjustable rate. The remote control allows patients to send commands wirelessly to their insulin pump from a distance up to 1-2 meters. The glucose meter and the glucose sensor (attached to the body) measure the glucose concentration in the patient’s blood, and send this value wirelessly to the insulin pump. This is then used as an indicator for the patient to adjust the dose. The USB stick (with software for therapy management) is a remote monitoring tool that collects data from the patient’s insulin pump, and then uploads these data to a website managed by the manufacturer.

Before any of these devices can communicate with the insulin pump, the patient needs to manually enter the device’s serial number on the insulin pump (except for the USB stick). This serial number is typically printed on the back of the device. This pairing process only needs to be done once. From then onwards, these devices can interact with the insulin pump as long as the patient does not modify the list of valid serial numbers stored on the insulin pump. If any of these devices is lost, damaged or stolen, the patient can manually remove its serial number from the list of valid devices, and enter the serial number of the new device.

4. LABORATORY SETUP

In our laboratory we employed available Commercial Off-The-Shelf (COTS) hardware including a Universal Serial Radio Peripheral (USRP) [4], an oscilloscope, an insulin pump, a remote control, a glucose meter, a glucose sensor and a USB stick. Initially, we used the USRP, a commercial antenna and a receiver program to eavesdrop the wireless channel and intercept messages. Using this setup, the sending/receiving communication range is more than 5 meters. In practice, adversaries can extend this range to several orders of magnitude. At a later stage, we were able to transmit messages by means of the USRP, the antenna and a transmitter program. Our receiver and transmitter programs were built in LabVIEW [2], a software tool which allows to interact with the USRP.
5. METHODOLOGY

The first step in the security and privacy analysis of this insulin pump system is to understand which data are sent and how these data are exchanged between devices. This is a difficult task because the protocol specifications are kept secret, since IMDs manufacturers often rely on obscurity to provide security, also known as security-through-obscurity. However, experience has shown that these proprietary solutions can be broken via several reverse-engineering techniques. More specifically, we follow a black-box based approach, i.e. we only give some inputs to the devices and then look at their outputs. In other words, we conduct an operation on the device (e.g. we press any of the remote control buttons), and then we look at the produced message format.

Our black-box approach can be divided into four steps:

1. **Find the wireless communication parameters:** Several wireless communication parameters, including the transmission frequency, modulation and symbol rate, need to be first discovered. This step is crucial, as using slightly different values would result in an erroneous received message.

2. **Reverse-engineering:** Secondly, we eavesdrop the wireless channel to intercept messages sent from any of the peripherals to the insulin pump (and vice versa). We then analyse these messages to discover both the message format as well as the protocol state-machine. For example, the existence of the remote control’s serial number within the message might be revealed by intercepting and comparing two messages sent by two different remote controls while performing the same action, i.e. pressing the same button.

3. **Obtain the serial number:** Most active attacks can only be mounted if the serial number of a valid device, i.e. in the list of patient’s devices with which the insulin pump can communicate, is known. There are several means to obtain the serial number of any of the patient’s devices: (i) we could either peek at the back of the device itself, (ii) get it through an insider working in the hospital, (iii) eavesdrop once the wireless channel or (iv) brute-force it.

4. **Perform the attacks:** Once the previous steps are completed, various attacks including replay, message injection and privacy attacks can be carried out on the insulin pump. In the next section, we will describe these attacks more in detail.

6. EXPERIMENTAL RESULTS

This section describes various types of active and passive software radio-based attacks that we can perform on the insulin pump.

6.1 Wireless communication parameters

The transmission frequency of any wireless device is publicly available on-line, and it can be easily obtained through its Federal Communications Commission (FCC) ID [1]. By checking the remote control’s FCC ID, we discovered that this device transmits at 868.35 MHz. Fig. 2 shows the waveform of the signals transmitted by the remote control. By observing this waveform, we found that the modulation scheme uses the presence of a carrier wave to indicate a binary ‘1’ and its absence to indicate a binary ‘0’. This type of modulation is known as On-Off Keying (OOK). To find the symbol rate, we opened our remote control and looked at the raw bits (i.e. bits before modulation) to be transmitted by tapping one pin of its micro-controller. Using the oscilloscope, we measured the duration of a bit, which corresponds to the duration of a symbol. Following the same steps, we found that the other peripherals also use the same wireless communication parameters.

![Waveform of the signal transmitted by the remote control.](image)

6.2 Reverse-engineering the wireless communication protocol

In this section, we show how to reverse-engineer the wireless communication protocol between the insulin pump and its peripherals. We found that the messages sent from any of the peripherals to the insulin pump (and vice versa) consist of common message fields and information bits that depend on the peripheral being used. In particular, all messages have a common start-of-frame and frame synchronization sequences. The former consists of a series of “1s” and “0s” sent consecutively to wake up the insulin pump from power saving mode, whereas the latter is used to indicate that the information bits are about to start. Next, there is a 6-bit sequence used to distinguish between the four types of devices that can communicate with the insulin pump. We will now explain how to discover the information bits sent by each peripheral more in detail.

6.2.1 Remote control - insulin pump

As explained in Section 2, in 2011, Li et al. reverse-engineered the wireless communication protocol between the remote control and the insulin pump [8]. In our experiments, presented below, we obtained similar results and hence validated their findings.

We started our tests by capturing the messages sent by the remote control while performing the same operation (i.e. pressing the same remote control’s button) several times. The result of this test showed that the remote control produces messages that differ in 24 bits each time. On the one hand, by computing the entropy of these bits, we determined that 12 bits seemed to be uniformly distributed, which made us think that a Code Redundancy Check (CRC) could be used to detect bit errors in the transmissions. CRCs are easy to implement and very effective at detecting bit errors caused by noise in the communication channel. In order to find the CRC generator polynomial, rather than brute-forcing all possible polynomials, we took the GCD of two observed messages (in polynomial form). This allowed us to discover that a standard CRC commonly used in mobile networks, known as CRC-8-WCDMA, is being used. On the other hand, the remaining 12 bits were repeated after 256 messages and hence they are used as a message counter, which is increased every time the patient presses any of the remote control’s buttons. Afterwards, we ascertained whether pressing different but-
tons on the remote control or using different remote controls causes some part(s) of the message to change. These tests led us to conclude that a 12-bit sequence depends on the button being pressed, and a 36-bit sequence depends on the remote control being used. Furthermore, by looking at two 36-bit sequences and their corresponding 6-digit remote control’s serial numbers, we found that each digit is represented by 6 bits. A mapping sequence is then used to convert each 6-bit sequence to a 4-bit hexadecimal number. Thus, due to this mapping sequence, the serial number only contains 24 unique bits. Note that for security reasons we do not disclose the table containing the mapping sequences in the paper. Fig. 3 shows the remote control’s message format.

6.2.2 Glucose meter - insulin pump

Several messages sent by two different glucose meters, wherein a wide range of glucose levels were measured, were first intercepted. Based on our previous results, we investigated whether using different glucose meters causes some part(s) of the message to change. We found that the glucose meter’s serial number is represented by 36 bits (6-bit per digit) and that it uses the mapping sequence previously found. This is followed by the information field, which remains empty (i.e. all zeros) for all captured messages. Subsequently, we grouped messages with identical and different glucose level, separately, in several clusters. By comparing the messages in these clusters, we discovered that 12 bits are used to transmit the measured glucose level. Finally, there is a 12-bit CRC that is computed using the CRC-8-WCDMA. Fig. 4 shows the glucose meter’s message format.

6.2.3 Glucose sensor - insulin pump

Given the similarities between the previous message’s formats, the first step was to determine whether the messages sent by the glucose sensor contain its serial number. This test showed that, unlike the previous cases in which we found a 36-bit serial number, the glucose sensor’s serial number is represented by 84 bits. In order to discover the rest of the message, we then looked at several consecutive messages sent by the glucose sensor. This allowed us to find a 6-bit counter along with the current and past eight measured glucose levels. Because of the longer message’s length in comparison with the messages sent by the previous peripherals, these messages use a 24-bit CRC (instead of a 12-bit CRC). We found that the CRC being used is the standard CRC-16-CCITT. Fig. 5 shows the glucose sensor’s message format.

6.2.4 USB stick - insulin pump

Without looking at the bits being transmitted, we first studied the message exchange process between the USB stick and the insulin pump. For this, we placed the insulin pump close to our USRP while keeping the USB stick (connected to our laptop) further away, thus, getting more power from the insulin pump, as shown in Figure 6. During a communication session, the USB stick and the insulin pump send to each other a fix number of messages. The USB stick always initiates the communication by sending a “wake up” message several times to the insulin pump. If the remote control and the insulin pump are within the same communication range, the latter replies to this message and the interrogation process starts, as shown in Figure 6. From that point onwards, two different consecutive phases can be distinguished: in the first phase, the USB stick requests data such as the model, firmware or the current settings from the insulin pump, whereas in the second phase the insulin pump first sends a message to the USB stick and the latter responds with an ACKnowledgement message (ACK).

Once we understood the message exchange process between these devices, we started to analyse the messages sent from the USB device to the insulin pump (and vice versa). A first test consisted on capturing several messages sent by the USB stick while interrogating two different insulin pumps. This test revealed that the insulin pump’s serial number is transmitted in both messages; the ones sent by the USB stick and the ones sent by the insulin pump. We found that the mapping sequence previously discovered is also used. By comparing two consecutive messages sent by the USB stick and the insulin pump, respectively, we found a 12-bit message type (or message identifier) sequence which indicates the USB stick message the insulin pump replies to.

Subsequently, insulin pump’s data such as the model, firmware or current settings are transmitted within the message. The length of this field (denoted by x in Figure 7) depends on the information requested/provided by the devices. Finally, similarly to the cases above, there is a 12-bit CRC at the end of

Figure 3: Remote control’s message format.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Remote control’s SN</th>
<th>Button</th>
<th>Counter</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>36 bits</td>
<td>12 bits</td>
<td>12 bits</td>
<td>12 bits</td>
</tr>
</tbody>
</table>

Figure 4: Glucose meter’s message format.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Glucose meter’s SN</th>
<th>Info</th>
<th>Glucose level</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>36 bits</td>
<td>12 bits</td>
<td>12 bits</td>
<td>12 bits</td>
</tr>
</tbody>
</table>

Figure 5: Glucose sensor’s message format.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Glucose sensor’s SN</th>
<th>Counter</th>
<th>Info</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>84 bits</td>
<td>6 bits</td>
<td>282 bits</td>
<td>24 bits</td>
</tr>
</tbody>
</table>

Figure 6: Communication between the USB stick (connected to the computer) and the insulin pump. From left to right, the message sent several times by the USB stick to wake up the insulin pump, the response sent by the insulin pump to the first USB stick’s message, the second message sent by the USB stick to the insulin pump and the response sent by the insulin pump to the second USB stick’s message.
the messages that is computed using the CRC-8-WCDMA. Fig. 7 shows the message format being used by the USB stick and the insulin pump.

### Figure 7: USB stick/insulin pump’s message format.

<table>
<thead>
<tr>
<th>Device type</th>
<th>Insulin pump SN</th>
<th>Message type</th>
<th>Info</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>6 bits</td>
<td>36 bits</td>
<td>12 bits</td>
<td>x bits</td>
<td>12 bits</td>
</tr>
</tbody>
</table>

6.3 Software radio-based attacks

In this section, we focus on the active and passive attacks that we can carry out on the insulin pump after having reverse-engineered the wireless communication protocol.

**Replay attacks:** We first investigated the feasibility of carrying out replay attacks, as these attacks can even be conducted without reverse-engineering the protocol; knowing the transmission frequency being used by the devices is sufficient. At first, we intercepted several messages sent by the remote control to the insulin pump and simply re-sent each message without demodulating the signal. Our experiments demonstrated that some sort of anti-replay mechanism is being used. This anti-replay mechanism does not allow to re-send the last message accepted by the insulin pump, but it does not impede us to send any other arbitrary message previously recorded. Thus, just by obtaining two different messages while performing an action (e.g. activate the insulin pump), we can alternatively re-send these messages as many times as needed in a protocol instance. Following the same approach, we also successfully performed this attack on the wireless link between: (i) the glucose meter and the insulin pump, (ii) the USB stick and the insulin pump. In both cases, messages were accepted the first time they were re-sent, meaning that no anti-replay mechanism is being used.

**Message injection attacks:** With the knowledge gained during the reverse-engineering process, we can create messages containing a valid serial number and successfully send them to the insulin pump. As a result, we can activate/suspend the insulin pump, deliver a high amount of insulin to the patient and send any arbitrary glucose value to the insulin pump. In addition to this, two different types of message injection attacks can be mounted on the wireless link between the USB stick and the insulin pump. On the one hand, we can emulate the insulin pump’s behaviour by replying to USB stick transmissions. This attack, though, has several limitations. Since the USB stick initiates the communication, triggered when a doctor clicks a button in the website managed by the manufacturer, the attacker has to wait for such an event and hijack the session, potentially even having to block the communication from the genuine insulin pump. On the other hand, we can also impersonate the USB stick to trigger an interrogation process. This could compromise both the security and privacy of the patient, as it might be launched either to discover sensitive patient health-related information or to drain the insulin pump’s battery.

**Privacy attacks:** Our analysis of the wireless communication protocol reveals that messages sent from any of the peripherals to the insulin pump (and vice versa) are not encrypted. Therefore, these messages disclose the type of device with which the insulin pump is communicating and its serial number, the current and past patient glucose levels, the insulin pump’s model/software version, the current insulin pump’s settings, the basal rate information or the total amount of insulin taken by the patient.

### 7. SECURITY DEFENCES

In the previous sections, we executed several active and passive attacks on the insulin pump that could compromise the safety and privacy of the patient. In this section, we present a cryptographic solution based on AES [9] that provides data confidentiality, authentication and freshness at a reasonable energy cost. Furthermore, we propose an optimized message format to reduce the energy cost. Without loss of generality, we will only focus on the wireless link between the remote control and the insulin pump. However, our solution can be applied as well to the wireless link between the insulin pump and any of the other peripherals.

#### 7.1 Key management

The remote control and the insulin pump need to share two independent symmetric cryptographic keys: one for encryption and one for authentication. We will now briefly explain the process of generating, transporting, updating and revoking these keys.

**Key generation:** To reduce the software footprint and the redundant computation on the remote control, both keys are generated by the insulin pump. To guarantee an appropriate level of entropy, a true randomness source is required. This randomness could originate either from the Static Random-Access Memory (SRAM) [16] or a True Random Number Generator (TRNG) [6].

**Key transport:** After the key generation process, both keys have to be securely transported from the insulin pump to the remote control. For this, we suggest to make physical contact between both devices via a secure channel (e.g. we connect a cable between devices), similarly as the Resurrecting Duckling approach proposed by Stajano and Anderson [15].

**Key update:** The patient is required to update the cryptographic keys every time the battery is replaced, i.e. every three weeks. During this process, the anti-replay counter is re-initialised to zero.

**Key revocation:** If the patient’s remote control is lost, stolen or damaged, both keys have to be revoked. If this occurs, the key generation procedure has to be executed with the new remote control. From that point onwards, the old keys are removed from the insulin pump.

#### 7.2 Our approach

##### 7.2.1 AES-based encryption and MAC

Our solution makes use of two cryptographic primitives: AES-128 in counter (CTR) mode to encrypt messages and AES-128 as a MAC. To find the energy requirements of our solution, we implemented both AES-128 in CTR mode and AES-128 MAC for openMSP430 [5] on a Spartan-6 FPGA. This is a representative mid-range micro-controller similar to the one used in the insulin pump. This software implementation can be easily converted to any other 16-bit microcontroller. Our implementation has been compiled with GNU Tools for Texas Instruments MSP430 micro-controllers optimized for code size with -Os gcc flags. In the rest of this section, we will investigate how to optimize this solution.
As a result, we investigated if it is possible to reduce the remote control’s serial number length while keeping the cost of verifying some extra MACs at a reasonable level. The relation between the energy consumption and the remote control’s serial number length for both the remote control and the insulin pump is shown in Figure. 8. When analysing this figure, we notice that the most optimal serial number’s length in the remote control (i.e. when no serial number is used) is the option that introduces the largest energy cost in the insulin pump. In this context, though, we take the serial number’s length where the total energy cost in the insulin pump is the lowest (i.e. 12-bit), as the patient can not receive the treatment if the insulin pump has no battery.

As a result, because of using a 12-bit remote control serial number (instead of a 24-bit serial number), the insulin pump does not only receive messages sent by the patient’s remote control, but also from other remote controls with the same optimized 12-bit serial number within the communication range. Given the assumptions explained in section 7.2.2 and assuming a uniformly distributed 12-bit serial number, the expected number of collisions per day is 7800 (2^{12}), namely, 2 collisions.

### 7.2.3 New message format

When observing the original message format (see Figure. 3), we note that several message fields can be optimized or removed when adding security countermeasures. In the new message format, the mapping sequence used to convert each 4-bit hexadecimal sequence to a 6-bit sequence can be removed, as it provides no extra security and it is costly to transmit extra bits. As a result, the length of several fields is reduced, resulting in an energy reduction (see Table 2). In the rest of this section, we will further optimize this new message format to reduce the communication cost without significantly increasing the computation cost.

#### 7.2.4 MAC + optimized SN + encryption

Our solution consists of encrypting the messages sent by the remote control using AES-128 in CTR mode, and then append a AES-128 MAC tag to each message. Intuitively, another possible solution would be to use authenticated encryption like Counter with CBC-MAC (CCM). However, from an energy point of view, there is no advantage on using CCM in comparison with our solution because of the small message length.

**Message format:** we notice that we can further optimize the message format proposed in section 7.2.3. For example, the CRC used in the original message format to verify the message’s integrity can be removed, as the MAC by itself already provides integrity protection. To prevent replay attacks during the three weeks lifetime of the key, a counter needs to be increased every time the patient presses any of the remote control buttons. This counter needs to be large enough to avoid reuse of previous values. In contrast to the original design, the insulin pump will now only accept a message if and only if the counter of the message is higher than the value of the previously received message.

**Remote control’s serial number optimization:** Since the MAC by itself is used to authenticate the remote control, another possible optimization is to reduce the length of the remote control’s serial number. However, there is an important trade-off between the communication cost and the computation cost. On the one hand, if there is no remote control’s serial number in the messages, the insulin pump would have to verify the MAC of the messages sent by: (i) the patient’s remote control and (ii) other patients’ remote control within the communication range. This approach increases the computation cost while reducing the communication cost. On the other hand, if the entire remote control’s serial number is sent, the insulin pump would only need to verify the MAC of the messages sent by the patient’s remote control. However, although this approach requires less computations, more bits need to be sent/received.

Therefore, we investigate if it is possible to reduce the remote control’s serial number length while keeping the cost of verifying some extra MACs at a reasonable level. The relation between the energy consumption and the remote control’s serial number length for both the remote control and the insulin pump is shown in Figure. 8. When analysing this figure, we notice that the most optimal serial number’s length in the remote control (i.e. when no serial number is used) is the option that introduces the largest energy cost in the insulin pump. In this context, though, we take the serial number’s length where the total energy cost in the insulin pump is the lowest (i.e. 12-bit), as the patient cannot receive the treatment if the insulin pump has no battery.

### Table 1: Implementation on MSP430 @16 Mhz, 1.8V.

<table>
<thead>
<tr>
<th>Operation</th>
<th>ROM (Byte)</th>
<th>Cycles (µs)</th>
<th>Time (µs)</th>
<th>Energy (µJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAC generation</td>
<td>2684</td>
<td>9430</td>
<td>590</td>
<td>2.14</td>
</tr>
<tr>
<td>MAC verification</td>
<td>2760</td>
<td>9561</td>
<td>598</td>
<td>2.16</td>
</tr>
<tr>
<td>Encryption/Decryption</td>
<td>2664</td>
<td>9404</td>
<td>588</td>
<td>2.13</td>
</tr>
<tr>
<td>Encryption + MAC generation</td>
<td>2879</td>
<td>18865</td>
<td>1180</td>
<td>4.27</td>
</tr>
<tr>
<td>Decryption + MAC verification</td>
<td>2847</td>
<td>18964</td>
<td>1186</td>
<td>4.30</td>
</tr>
</tbody>
</table>

### 7.2.2 Communication cost vs. computation cost

Although our solution provides data confidentiality and authentication to prevent the remote attacks previously shown, the use of cryptography increases the energy consumption in both the remote control and the insulin pump. The energy consumption can be divided into two main components: the computation and the communication cost. The former indicates the cost of performing operations such as encrypting/decrypting messages, whereas the latter refers to the cost of transmitting/receiving bits to/from a device.

On the one hand, to calculate the computation cost, we measure the number of clock cycles needed to perform an operation. By looking at the power consumption (in the micro-controller specifications [3]) and knowing the number of clock cycles required, we can compute its energy cost, as shown in Table 1. (Note that all the operations described in this table consider a 128-bit block). On the other hand, to compute the communication cost, we looked at the datasheet specifications of the remote control’s radio transmitter and the insulin pump’s radio receiver [7]. As a result, we found that the cost of transmitting and receiving one bit at 868.35 MHz using an OOK modulation scheme is 2.25µJ and 0.75µJ, respectively.

**Assumptions:** To demonstrate the practicality of our solution, we consider the following worst case scenario. We assume that there is a patient in a very crowded hospital full of patients who all use an insulin pump. We assume that each patient has 5 meals per day, during which any of the remote control’s buttons are pressed 30 times, so 150 remote control’s buttons are pressed per patient daily. Given the 2 meter communication range between the remote control and the insulin pump, we can estimate the interference, i.e. the number of messages received from other patients’ remote control. Assuming a density of 4 people per square meter, the number of patients within a radius of two meters is 52. In other words, the insulin pump of the patient receives 7800 messages sent by other patients’ remote control daily.
Table 2: Energy cost (per day) of each solution for the remote control (RC) and the insulin pump (IP).

<table>
<thead>
<tr>
<th>Solution</th>
<th>Confidentiality</th>
<th>Authentication</th>
<th>Computation cost</th>
<th>Communication cost</th>
<th>Total cost</th>
<th>Cost increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>No security (original system)</td>
<td>×</td>
<td>×</td>
<td>RC: 0 mJ</td>
<td>IP: 0 mJ</td>
<td>RC: -</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IP: 26.32 mJ</td>
<td>IP: 8.77 mJ</td>
<td>RC: +28.19%</td>
<td></td>
</tr>
<tr>
<td>No security (new message format)</td>
<td>×</td>
<td>×</td>
<td>RC: 0 mJ</td>
<td>IP: 0 mJ</td>
<td>RC: -</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IP: 18.9 mJ</td>
<td>IP: 6.30 mJ</td>
<td>RC: +28.16%</td>
<td></td>
</tr>
<tr>
<td>MAC + opt SN encryption</td>
<td>✓</td>
<td>✓</td>
<td>RC: 0.64 mJ</td>
<td>IP: 0.65 mJ</td>
<td>RC: +35.79%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IP: 35.10 mJ</td>
<td>IP: 11.86 mJ</td>
<td>RC: +42.64%</td>
<td></td>
</tr>
<tr>
<td>Only encryption</td>
<td>✓</td>
<td>×</td>
<td>RC: 0.32 mJ</td>
<td>IP: 0.32 mJ</td>
<td>RC: -21.84%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IP: 20.25 mJ</td>
<td>IP: 6.75 mJ</td>
<td>RC: -19.38%</td>
<td></td>
</tr>
<tr>
<td>Only MAC</td>
<td>×</td>
<td>✓</td>
<td>RC: 0.32 mJ</td>
<td>IP: 0.32 mJ</td>
<td>RC: +50%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>IP: 39.15 mJ</td>
<td>IP: 13.05 mJ</td>
<td>IP: +52.45%</td>
<td></td>
</tr>
</tbody>
</table>

*Original remote control’s message format without the mapping sequence.

Table 8 shows the security properties of the different security solutions with their corresponding energy costs for both the remote control and the insulin pump.

Our results show that in all these solutions the computational cost is negligible compared to the cost of transmit-
ting/receiving bits, which evidences the importance of optimizing the message format. On the one hand, we demonstrate that the data confidentiality requirement can be easily accomplished if these devices transmit/receive messages with an optimized format. We show that, by using a reduced message format to lower the communication cost, even adding encryption/decryption still requires a lower energy cost compared to the original design (without security and a non-optimized message format). On the other hand, we show that the authentication requirement is more difficult to accomplish, as this can only be achieved by appending a MAC to the message, which inevitably increases the message length and so the cost of transmitting/receiving bits.

How can the energy costs be further reduced? Our results show that the computation cost is about the 2% of the total energy cost of our final solution. So, even if the cost of computing cryptographic operations could be theoretically reduced to zero, the total energy cost would only be slightly decreased. Therefore, the problem does not solely lie in the design of lightweight cryptographic algorithms but particularly on finding ways to reduce the communication cost, e.g. further optimize the message format proposed in section 7.2.3. Another possibility would be to use a 32-bit tag rather than a 64-bit tag (as currently used in our solution). However, even though 32-bit MACs are considered to be secure against on-line attacks, attacks where an attacker can capture messages and then try all possible keys off-line could still be carried out. While not practical in this insulin pump system, another option could be to compute the MAC over several messages rather than doing it per every message. Our work clearly shows that there is a need for lightweight integrity solutions which do not significantly increase the message size.

8. CONCLUSIONS
In this work we have demonstrated, by reverse-engineering two models of a commonly used insulin pump system, that security through obscurity is a dangerous design approach. Our protocol analysis resulted in the identification of serious security and privacy vulnerabilities. We discovered that no defenses against replay and message injection attacks were present, and sensitive patient health-related information was disclosed unencrypted in the wireless communication.

To prevent these active and passive software radio-based attacks, we showed how to secure the wireless link using a security solution based on AES in combination with an optimization of the message format. We evaluated and implemented this solution on a low-cost 16-bit micro-controller similar to the one used in the insulin pump, and compared its security properties and energy costs with alternative solutions that provide less security. We show that there is a need for future research on how to protect the message integrity without largely increasing its length.

In accordance with the principle of responsible disclosure, we have notified the manufacturer six months before disclosure.

9. ACKNOWLEDGMENTS
The authors would like to thank Pieter Gillard and Saskia Vanderwegen for their support and the anonymous reviewers for their helpful comments. This work was supported in part by the Research Council KU Leuven: C16/15/058.

10. REFERENCES