# A privacy-preserving remote healthcare system offering end-to-end security

**DOI:**
10.1007/978-3-319-40509-4_17

**Document Version**
Accepted author manuscript

# A Privacy-preserving Remote Healthcare System Offering End-to-End Security

Eduard Marin, Mustafa A. Mustafa, Dave Singelée, and Bart Preneel

KU Leuven, ESAT-COSIC and iMinds,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
firstname.lastname@esat.kuleuven.be

**Abstract.** Remote healthcare systems help doctors diagnose, monitor and treat chronic diseases by collecting data from Implantable Medical Devices (IMDs) through base stations that are often located in the patients' house. In the future, these systems may also support bidirectional communication, allowing remote reprogramming of IMDs. As sensitive medical data and commands to modify the IMD's settings will be sent wirelessly, strong security and privacy mechanisms must be deployed.
In this paper, we propose a user-friendly protocol that is used to establish a secure end-to-end channel between the IMD and the hospital while preserving the patient's privacy. The protocol can be used by patients (at home) to send medical data to the hospital or by doctors to remotely reprogram their patients' IMD. We also propose a key establishment protocol between the IMD and the base station based on a patient's physiological signal in combination with fuzzy extractors. Through security analysis, we show that our protocol resists various attacks and protects patients' privacy.

## 1 Introduction

Remote healthcare systems usually collect data from medical devices implanted within the patient's body, also known as Implantable Medical Devices (IMDs), several times per day through a base station that is often installed in the patient's house. In the near future, these systems may support bidirectional communication to enable remote reprogramming of the IMD by a doctor in a hospital. While these remote healthcare systems can improve the patients' quality of life and extend the time they can live independently at home, they pose important security and privacy risks. Currently, proprietary protocols are being deployed with limited security measures [7, 11]. These insecure wireless protocols may lead to several attacks that can result in fatal consequences for patients.

Remote healthcare systems are typically built such that there is a central node (denoted as data concentrator in the rest of this paper) that collects and redirects all (encrypted) data sent between base stations and hospitals. Context information about these communications, i.e. metadata, can be valuable for insurance companies or government agencies to monitor public health or compile statistics. However, even if cryptography is used, metadata can leak patients'

sensitive information to the data concentrator, which may lead to abuse of data, e.g. denying individuals an insurance contract. To mitigate some of these issues, a trivial solution would be to have several data concentrators (instead of only one) that do not cooperate with each other. This solution would be expensive to deploy and difficult to maintain. In addition, there would not be any guarantee that the data concentrators do not share data with each other. Another solution would be to use a fresh pseudonym in every message. Although this approach would make it more difficult for adversaries to compromise the patient's privacy, this is not sufficient; if the data concentrator knows where the base station is located, then unique patient identifiers may be revealed.

Our first contribution is a user-friendly protocol that allows an IMD to establish an end-to-end secure channel with a hospital while preserving the patient's privacy. The protocol can be used by patients (at home) to send medical data to the hospital or by doctors to remotely reprogram the IMD of their patients. Our protocol makes use of cryptography and an anonymous communication channel to prevent the data concentrator from learning patients' sensitive information. Our second contribution is a key establishment protocol that allows an IMD and a base station to agree on a symmetric session key without using public-key cryptography or any pre-shared secrets between devices. Instead, our protocol uses a patient's physiological signal in combination with fuzzy extractors.

## 2 Related Work

### 2.1 Security and Privacy in Remote Monitoring Systems

While much research has focused on making remote monitoring systems more reliable, unobtrusive, energy efficient and scalable, security and privacy have received less attention [10, 12]. Ko et al. acknowledged the importance of security and privacy, but they did not provide details about cryptographic mechanisms [9]. Ortiz et al. proposed a protocol to secure the wireless communication between devices in a medical system [13]. However, the protocol does not provide message integrity, and the receiver keeps a list of keys for each transmitter. Transmitters send messages along with their unique device identity (ID) unencrypted to indicate to the receiver which of the keys is used to decrypt the message. This may result in a privacy breach since adversaries can use the unique transmitter ID to track, identify or locate individuals. Perrig et al. presented SNEP, a protocol that provides data confidentiality, integrity, mutual authentication and message freshness [14]. Pre-installed symmetric keys, shared between each device and the base station, are used to derive a new session key every time a device starts communicating with the base station. In contrast to Perrig et al., we also consider privacy. More specifically, our protocol aims to prevent unauthorized entities and adversaries from discovering the patients' real ID, their location or being able to link their messages. Furthermore, the devices are implanted within the patient's body and the base stations do not have any pre-shared keys with them, which makes key management more challenging.

### 2.2 Key Establishment Protocols

The unique characteristics of IMDs pose novel challenges in the design of key establishment protocols and key management solutions. IMDs are battery-powered and resource-constrained devices in terms of size, memory, processor and energy. The battery typically lasts 7 years. When the battery is drained, surgery is needed to replace the IMD. We note that a trade-off between security and open-access in emergencies is also required. Consider a cardiac patient who is travelling. Although it is clear that no one should be able to access his pacemaker while he is walking on the street, medical staff should have immediate access to his IMD in an emergency situation.

Intuitively, one possible way for the IMD and the base station to establish a key would be to use public-key cryptography. However, IMDs cannot use expensive cryptographic primitives in terms of computations and power consumption. Another possibility would be to pre-install a master (symmetric) key in all IMDs, but this may prevent a patient from receiving care in an emergency situation. A pairing protocol could also be used for establishing a symmetric session key [18, 6, 4]. Nevertheless, IMDs do not contain a screen, a keyboard or an accelerometer and it is not possible to physically access them once implanted; thus none of the existing solutions can be used. In this paper, we propose a key establishment protocol in which the IMD and the base station measure a patient's physiological signal independently and synchronously to agree on a symmetric key. The protocol uses the time between heart beats, also known as InterPulse Interval (IPI), as the source of randomness, similarly to the touch-to-access protocol proposed by Rostami et al. [15]. Unlike their work, our protocol is more efficient as it only uses symmetric cryptography in combination with fuzzy extractors.

## 3 Design Preliminaries

### 3.1 System Model

Our remote healthcare system, similar to existing architectures, consists of the following entities (see Fig. 1). Without loss of generality, in the rest of this paper we will assume that the IMD is a pacemaker.

A *pacemaker* (PM) is a device implanted within a patient's body that is used to monitor and control his heart beat. A *base station* (BS) is an external device installed in a fixed location (e.g. home or a hotel) which collects and forwards medical data to a hospital, and sends commands to PMs as instructed by a doctor in the hospital. BSs have a programming head that incorporates a built-in sensor to read a patient's physiological signal (e.g. the IPI). A *data concentrator* (DC) acts as a bridge between BSs and hospitals, and is in practice often managed by the company that manufacturers the PMs and BSs. A *hospital* (HO) is a medical institution where doctors analyse the medical data and send commands to PMs, whereas a *certification authority* (CA) is a trusted entity that issues digital certificates to HOs and the DC.

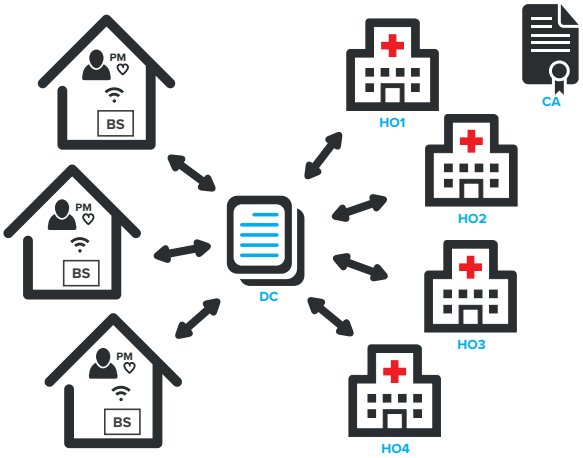


**Fig. 1.** Our remote healthcare system comprises pacemakers (PMs), base stations (BSs), a data concentrator (DC), hospitals (HOs) and a certification authority (CA).

A BS and a PM can communicate with each other wirelessly using the Medical Implant Communication Service (MICS) band [17], or any other low-energy wireless technology. The communication range between the BS and the PM depends on the wireless communication technology being used, e.g. from two to five meters when using the MICS band. The communication between the BS and the DC takes place over the Internet using a low-latency anonymous communication channel (e.g. a Mix network [16]), whereas the communication between the DC and a HO takes place over the Internet using a standard secure channel, e.g. TLS. To balance the load, multiple DCs are in place; however, for the sake of simplicity, we consider them as one in the rest of this paper.

We consider two possible scenarios depending on whether the doctor is on-line or off-line. For an on-line remote medical check, the patient first makes an appointment with the doctor (e.g. via telephone). At the time of the appointment, the patient sends medical data to the HO through the base station. The doctor then analyses the data and, if required, sends commands to the PM. If the doctor is off-line, the patient can still send medical data to the HO; however, these data will be processed by the doctor at a later stage. We note that in our system the doctor can only reprogram the patient's PM in the on-line scenario.

### 3.2 Threat Model and Assumptions

**Threat Model:** PMs and HOs are honest and trusted. PMs follow the protocol specifications as designed by their manufacturers and the U.S. Federal Communications Commission (FCC). PMs can only establish one communication session with a BS at a time, and do not initiate any communication without receiving a request from a legitimate BS [2]. Adversaries can eavesdrop, modify, inject and jam the messages exchanged between any of the entities. Adversaries can observe

only a fraction of the Internet network traffic. This is a common assumption when using low-latency anonymous communication systems. In addition, adversaries might compromise any number of BSs, including the one being used by the patient. The DC is honest but curious; it follows the protocol specifications but it might attempt to discover information about patients by looking at metadata.

**Assumptions:** We assume that adversaries cannot make physical contact with the patient without this being noticed by the patient. We assume that the communication between the BS and the DC takes place over the Internet using an anonymous communication channel. However, we do not specify which type of anonymous channel is used, since this is out of the scope of this paper. We assume that all entities (except PMs) have the CA's certificate pre-installed. We assume that each HO has a server with a database that contains a list of their patients along with their corresponding PM IDs and cryptographic keys. The server is located in a secure room where it cannot be stolen or tampered with; only authorized medical staff has access to it through appropriate identification, authentication and authorization mechanisms.

### 3.3 Design Requirements

Our remote healthcare system should satisfy the following functional (F), security (S) and privacy (P) requirements.

**(F1) User-friendly:** Reporting medical data and reprogramming the PM should be easy and convenient for patients.

**(F2) BS-independent:** Patients should be able to use any legitimate BS, even the ones belonging to other patients.

**(F3) Energy Cost:** Computational cost at PMs should be as low as possible to reduce the energy consumption.

**(S1) Mutual Entity Authentication:** PMs and HOs should be assured of each other's identity when receiving messages.

**(S2) Message Integrity:** PMs and HOs should be assured that the received messages are fresh and have not been altered during transit.

**(S3) Confidentiality of Medical Data and Commands:** Only the authorized HO should be able to access patient's medical data and send commands.

**(S4) Availability:** Ensures that the system is accessible upon demand by authorised entities.

**(P1) Patient Privacy** (minimum data disclosure):

**(P1.1) Patient Identity Privacy:** Only HOs should know the identity of the patient who sends medical data.

**(P1.2) Hospital Identity Privacy:** No unauthorized entity should know to which hospital the patient sends medical data.

**(P1.3) Location Privacy:** No entity should infer the patient's location. [1]

**(P1.4) Session Unlinkability:** Only the HO where the patient is registered should be able to link messages sent from the same patient.

[1] In an emergency situation doctors have other means (e.g. necklace-based emergency systems) to know the patient's location.

## 4 The Protocol

This section presents our protocol for medical data reporting and remote reprogramming of the patient's PM. It provides end-to-end security (i.e. data confidentiality, integrity, mutual authentication and message freshness) between the patient's PM and the hospital, and protects the patient's privacy. Our protocol is divided into two stages: the medical data reporting stage and the PM reprogramming stage. Prior to the detailed protocol description, we first outline the system initialization process. Table 1 shows the notation used in the paper.

### 4.1 System Initialisation

Each HO and the DC generate a public/private key pair, $PK_{ho_i}/SK_{ho_i}$ and $PK_{dc}/SK_{dc}$, respectively. The public keys are signed by the CA. A list of valid HO certificates is stored in the DC, whereas each HO has a valid DC certificate. A group signature scheme is used by BSs to anonymously sign messages on behalf of the BSs group, so that the DC can still verify the authenticity of the message without knowing which specific BS signed the message, thus hiding the ID and location of the patient. All BSs use the same (group) public key, $PK_{bs_{group}}$, and have distinct private keys, $SK_{bs_i}$. Next, the DC signs the BSs group public key, generates a digital certificate that contains the ID and group's public key, and stores it. The certificate of the DC is pre-installed in all BSs.

During the PM manufacturing process, a symmetric key, $K_{ho\text{-}ps}$, is pre-installed in each PM. $K_{ho\text{-}ps}$ is shared between all PMs and the DC, and used for generating HO pseudonyms. Our protocol uses the same $K_{ho\text{-}ps}$ for all PMs, so that the DC cannot identify the PM that generated the HO pseudonym. In the PM setup phase, two independent symmetric keys, $K_{pm\text{-}ho}$ and $K_{pm\text{-}ps}$, are generated and installed in each PM. The procedure takes place in the HO before the PM is implanted to avoid the PM's manufacturer (often the DC) from learning these keys. $K_{pm\text{-}ho}$ and $K_{pm\text{-}ps}$ are shared between the PM and its corresponding HO. The former is used for encrypting the patient's medical data whereas the latter for generating PM pseudonyms.

Various circumstances may cause the certificates to become invalid before their expiration date. If the private key of any of the entities is compromised, a new public/private key pair is generated (as explained above). The new public key is then signed by the CA and broadcasted to the network. All entities can then verify the message authenticity by using the CA's public key. From that point onwards, the old certificate is no longer valid. If the private key of any BS is compromised, the BS is sent to its manufacturer for being reconfigured and replaced. We note that group signature schemes typically allow revocation and addition of new members (i.e. BSs) into the group (for more details, see [3]).

### 4.2 Medical Data Reporting Stage

After the system initialisation phase, the PM can report medical data to the HO (see Fig. 1). Prior to each reporting stage, a new symmetric session key is

**Table 1.** Notations.

| Symbols | Meanings |
|---|---|
| d, cmd | medical data of a patient, command sent to the PM |
| $\mathrm{ID}_i$, $\mathrm{PS}_i$ | unique identity of entity $i$, pseudonym of entity $i$ |
| $\mathrm{K}_{\text{pm-ho}}$ | key shared between PM and HO to encrypt/decrypt data/commands |
| $\mathrm{K}_{\text{pm-ps}}$ | key shared between PM and HO to create PM pseudonyms (ps) |
| $\mathrm{K}_{\text{ho-ps}}$ | key shared between all PMs and DC to generate HO's pseudonyms (ps) |
| $\mathrm{K}_{\mathrm{s}}$ | session key established between PM and BS |
| $\mathrm{N}_i$, $\mathrm{TS}_i$ | nonce generated by entity $i$, timestamp produced by entity $i$ |
| $\mathrm{msg}_{i\text{-}j}$ | message constructed by entity $i$ and intended for entity $j$ |
| $\mathrm{ct}_{i\text{-}j}$ | counter used in messages between the entity $i$ and the entity $j$ |
| $\mathrm{C}_{i\text{-}j}$ | ciphertext generated by entity $i$ and intended for entity $j$ |
| $\mathrm{PK}_i$, $\mathrm{SK}_i$ | public and private key of entity $i$ |
| $PRF_{\mathrm{K}}(\mathrm{M})$ | pseudorandom function of message M with key K |
| $AE_{\mathrm{K}}(\mathrm{M})$ | authenticated encryption of message M with key K |
| $E_{\mathrm{PK}_i}(\mathrm{M})$ | asymmetric encryption of message M with public key of entity $i$ |
| $\mathrm{Sig}_i(\mathrm{M})$ | digital signature of entity $i$ on message $M$ |

established between the PM and the BS for securing the data exchanged over the air. We next describe the proposed key establishment protocol followed by the actual reporting stage more in detail.

**IPI-based Key Establishment Protocol:** Our protocol requires the IMD and the BS to independently measure the patient's IPI (i.e. time between heart beats) at the same time. These IPI readings, which can be measured anywhere in the patient's body just by touching the patient's skin, are then used for establishing a symmetric key that is valid only for one session. Previous work has shown that the four least significant bits of IPIs are uncorrelated and independently identically distributed (i.i.d) [19]. The IPI cannot be read remotely (e.g. via a webcam), as shown by Rostami et al. [15]. Therefore, based on their results and our assumptions, a remote attacker cannot measure the IPI. Fig. 2 shows the IPI-based key establishment protocol.

To trigger the key establishment procedure, the patient first presses a button on the BS. The PM and BS then take two readings of the patient's IPI at the same time. However, these readings are not equal (but rather similar) due to the noise. Let us denote the reading taken by the BS as $\alpha$ and the reading taken by the PM as $\beta$. Fuzzy extractors allow generating a cryptographic key $k$ from $\alpha$ and then successfully reproduce $k$ from $\beta$, iff $\alpha$ and $\beta$ are almost equal [5]. Fuzzy extractors are composed by two functions: generate $\big(Gen\big)$ and reproduce $\big(Rep\big)$. *Gen* is executed by the BS with $\alpha$ as an input, and outputs a key $k \in \big\{0,1\big\}^l$ and helper data $P \in \big\{0,1\big\}^*$. The BS then sends $P$ in order to help the PM to reproduce $k$. The PM executes *Rep* with $\beta$ and $P$ as inputs, and outputs a key *k'* (if $\beta$ and $\alpha$ are similar, then $k$ equals *k'*).

To achieve key confirmation, the BS generates a nonce, $\mathrm{N}_{\mathrm{bs}}$, and sends it to the PM along with a Message Authentication Code (MAC), $MAC_{\mathrm{k}}(\mathrm{N}_{\mathrm{bs}})$.

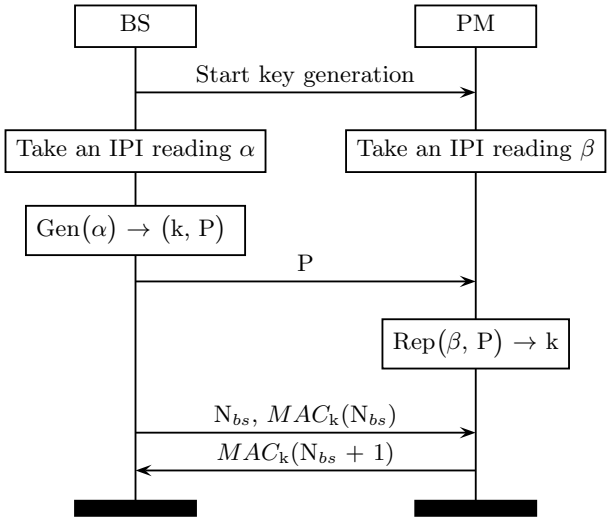


**Fig. 2.** IPI-based key establishment protocol between the BS and the PM.

Upon receiving the message, the PM verifies $MAC_{\mathrm{k}}(\mathrm{N_{bs}})$ using *k'*. If the MAC is verified correctly, the PM is assured that the BS knows the key; however the BS does not have any assurance that the PM knows the key. For the PM to prove knowledge of the key, it computes $MAC_{\mathrm{k}}(\mathrm{N_{bs}} + 1)$ and sends it to the BS. The BS repeats this operation with its own key and checks whether the result of this operation corresponds to the received MAC. If the MAC is verified correctly, both devices can use $k$, (hereafter denoted as $\mathrm{K_s}$), to securely communicate with each other, otherwise they execute the key establishment protocol again.

**Reporting Medical Data to the HO:** In this stage the patient's PM sends medical data to the HO. Fig 3 depicts the processes executed by the entities.

**PM:** The PM performs the following steps.

1. It generates a fresh nonce, $\mathrm{N_{pm}}$, that is used to compute two fresh pseudonyms. First, it computes a pseudonym for itself, i.e. $\mathrm{PS_{pm}} = PRF_{\mathrm{K_{pm\text{-}ps}}}(\mathrm{ID_{pm}} \parallel \mathrm{N_{pm}})$, where $PRF$ is a pseudorandom function (e.g. a secure block cipher) and $\mathrm{ID_{pm}}$ is the PM's real identity (e.g. serial number). This pseudonym is only used once (i.e. in the message sent from the PM to the HO). Next, it computes a pseudonym for the HO, i.e. $\mathrm{PS_{ho}} = PRF_{\mathrm{K_{ho\text{-}ps}}}(\mathrm{ID_{ho}} \parallel \mathrm{N_{pm}})$, where $\mathrm{ID_{ho}}$ is the HO's real identity. This pseudonym is used in a pair of messages (i.e. the one sent from the PM to the HO and vice versa). Both pseudonyms protect the patient's privacy while communicating with the HO, i.e. the PM uses $\mathrm{PS_{pm}}$ and $\mathrm{PS_{ho}}$ instead of $\mathrm{ID_{pm}}$ and $\mathrm{ID_{ho}}$.

2. It encrypts the patient's medical data, d, and a counter, $\mathrm{ct_{pm\text{-}ho}}$, using the secret key it shares with the HO, $\mathrm{K_{pm\text{-}ho}}$, i.e. $(\mathrm{C, T})_{\mathrm{pm\text{-}ho}} = AE_{\mathrm{K_{pm\text{-}ho}}}(\mathrm{ct_{pm\text{-}ho}} \parallel \mathrm{d})$. Since PMs do not contain a precise clock, a counter is used to prevent replay attacks. The counter is initialised to zero every time a new session key between the BS and the PM is established. For better performance, the encryption method used is an authenticated encryption scheme (e.g. AES-CCM [1]), which outputs a ciphertext, C, and an authentication tag, T.

3. It constructs a message, $\mathrm{M_{pm\text{-}bs}} = \mathrm{PSs} \parallel (\mathrm{C, T})_{\mathrm{pm\text{-}ho}} \parallel \mathrm{N_{pm}}$, where $\mathrm{PSs} = \mathrm{PS_{pm}} \parallel \mathrm{PS_{ho}}$, and encrypts it using the session key previously established with the BS, $\mathrm{K_s}$, i.e. $(\mathrm{C, T})_{\mathrm{pm\text{-}bs}} = AE_{\mathrm{K_s}}(\mathrm{ct_{pm\text{-}bs}} \parallel \mathrm{M_{pm\text{-}bs}})$.

4. It sends $\mathrm{msg_{pm\text{-}bs}} = (\mathrm{C, T})_{\mathrm{pm\text{-}bs}}$ to the BS.

**BS:** Upon receiving $\mathrm{msg_{pm\text{-}bs}}$, the BS performs the following steps.

1. It verifies the authenticity of $\mathrm{msg_{pm\text{-}bs}}$ and decrypts the ciphertext to obtain $(\mathrm{ct_{pm\text{-}bs}} \parallel \mathrm{M_{pm\text{-}bs}})$. It then checks whether the counter, $\mathrm{ct_{pm\text{-}bs}}$, is valid, i.e. if it is higher than the counter of the last received message. If this condition is satisfied, the message is accepted, otherwise it is rejected.

2. It generates a random session ID, $\mathrm{S_{id}}$, that is valid for a pair of messages and allows the HO to anonymously send a message back without knowing which specific BS sent the message. Then it encrypts $\mathrm{S_{id}}$ along with $\mathrm{M_{pm\text{-}bs}}$ using the public key of the DC, $\mathrm{PK_{dc}}$, i.e. $\mathrm{C_{bs\text{-}dc}} = E_{\mathrm{PK_{dc}}}(\mathrm{S_{id}} \parallel \mathrm{M_{pm\text{-}bs}})$.

3. It constructs a message, $\mathrm{M_{bs\text{-}dc}} = (\mathrm{TS_{bs}} \parallel \mathrm{C_{bs\text{-}dc}})$, where $\mathrm{TS_{bs}}$ is a timestamp of the BS used to counter replay attacks. $\mathrm{TS_{bs}}$ does not need to be kept secret and is sent unencrypted to the DC; however, its integrity is protected by means of a digital signature. This allows the DC to check the freshness of the message before decrypting the message and verifying its authenticity.

4. It generates a signature on $\mathrm{M_{bs\text{-}dc}}$ using its private key, $\mathrm{Sig_{SK_{bs}}(M_{bs\text{-}dc})}$. Then it constructs a message, $\mathrm{msg_{bs\text{-}dc}} = \mathrm{M_{bs\text{-}dc}} \parallel \mathrm{Sig_{SK_{bs}}(M_{bs\text{-}dc})}$, and sends it to the DC via an anonymous communication channel.

**DC:** Upon $\mathrm{msg_{bs\text{-}dc}}$ reception, the DC performs the following steps.

1. It verifies the freshness of $\mathrm{msg_{bs\text{-}dc}}$ by checking $\mathrm{TS_{bs}}$ and the authenticity of the message using the BSs group public key, $\mathrm{PK_{bs_{group}}}$. It then decrypts $\mathrm{C_{bs\text{-}dc}}$ to obtain $(\mathrm{S_{id}} \parallel \mathrm{M_{pm\text{-}bs}})$, where $\mathrm{M_{pm\text{-}bs}} = (\mathrm{PS_{pm}} \parallel \mathrm{PS_{ho}} \parallel (\mathrm{C, T})_{\mathrm{pm\text{-}ho}} \parallel \mathrm{N_{pm}})$.

2. It retrieves $\mathrm{ID_{ho}}$ by computing $\mathrm{PS'_{ho}} = PRF_{\mathrm{K_{ho\text{-}ps}}}(\mathrm{ID_{ho_i}} \parallel \mathrm{N_{pm}})$ for all HOs, $(\mathrm{ID_{ho_1}}, \ldots, \mathrm{ID_{ho_n}})$ until the DC finds a match, i.e. $\mathrm{PS'_{ho}}$ equals $\mathrm{PS_{ho}}$.

3. It constructs a message, $\mathrm{M_{dc}} = (\mathrm{PS_{pm}} \parallel \mathrm{ID_{ho}} \parallel (\mathrm{C, T})_{\mathrm{pm\text{-}ho}} \parallel \mathrm{N_{pm}})$, in which $\mathrm{PS_{ho}}$ is replaced with $\mathrm{ID_{ho}}$. Then it encrypts the message and the session ID, $\mathrm{S_{id}}$, using the public key of the HO, $\mathrm{PK_{ho}}$, i.e. $\mathrm{C_{dc\text{-}ho}} = E_{\mathrm{PK_{ho}}}(\mathrm{S_{id}} \parallel \mathrm{M_{dc}})$.

4. It constructs a message, $\mathrm{M_{dc\text{-}ho}} = (\mathrm{TS_{dc}} \parallel \mathrm{C_{dc\text{-}ho}})$, where $\mathrm{TS_{dc}}$ is a timestamp of the DC, and generates a signature on $\mathrm{M_{dc\text{-}ho}}$ using its private key, $\mathrm{Sig_{SK_{dc}}(M_{dc\text{-}ho})}$. Finally, it appends the signature to $\mathrm{M_{dc\text{-}ho}}$ in order to form a message, $\mathrm{msg_{dc\text{-}ho}} = \mathrm{M_{dc\text{-}ho}} \parallel \mathrm{Sig_{SK_{dc}}(M_{dc\text{-}ho})}$, and sends it to the HO.

**HO:** Upon $msg_{dc\text{-}ho}$ reception, the HO performs the following steps.

1. It verifies the freshness of $msg_{dc\text{-}ho}$ by checking $TS_{dc}$ and the authenticity of the message by checking the validity of $Sig_{SK_{dc}}(M_{dc\text{-}ho})$. It then decrypts $C_{dc\text{-}ho}$ to obtain $(S_{id} \parallel M_{dc})$, where $M_{dc} = (PS_{pm} \parallel ID_{ho} \parallel (C, T)_{pm\text{-}ho} \parallel N_{pm})$.

2. It retrieves $ID_{pm}$ by computing $PS'_{pm} = PRF_{K_{pm\text{-}ps}}(ID_{pm_i} \parallel N_{pm})$ for all PMs, $(ID_{pm_1}, \ldots, ID_{pm_w})$, where w is the number of patients with a PM registered in the HO, until a match is found, i.e. $PS'_{pm}$ equals $PS_{pm}$.

3. It searches for $ID_{pm}$ in its database to retrieve $K_{pm\text{-}ho}$. Using this key, the HO verifies and decrypts $(C, T)_{pm\text{-}ho}$ to obtain $(ct_{pm\text{-}ho} \parallel d)$. Next, the HO verifies the freshness of the message by checking if the counter, $ct_{pm\text{-}ho}$, is higher than the counter of the previously received message. Only if this condition is fulfilled, the HO accepts the patient's medical data, d, as authentic and genuine.

### 4.3 PM Reprogramming Stage

After examining the patient's medical data, the doctor can send the necessary command(s) to adjust the PM's settings. This stage, which can only take place if the doctor is on-line, is described next and shown in Fig. 4.

**HO:** The HO performs the following steps.

1. It generates a fresh nonce, $N_{ho}$, to create a fresh pseudonym for the PM, $PS_{pm}$. Then it increases the counter $ct_{pm\text{-}ho}$, and encrypts it along with the command, cmd, using $K_{pm\text{-}ho}$, i.e. $(C, T)_{ho\text{-}pm} = AE_{K_{pm\text{-}ho}}(ct_{pm\text{-}ho} \parallel cmd)$.

2. It constructs a message, $M_{ho} = (ID_{ho} \parallel PSs \parallel (C, T)_{ho\text{-}pm} \parallel Ns)$, where $PSs = (PS_{pm} \parallel PS_{ho})$ and $Ns = (N_{pm} \parallel N_{ho})$. Next it encrypts $M_{ho}$ and the session ID using the public key of the DC, i.e. $C_{ho\text{-}dc} = E_{PK_{dc}}(S_{id} \parallel M_{ho})$.

3. It constructs a message $M_{ho\text{-}dc} = (TS_{ho} \parallel C_{ho\text{-}dc})$ and generates a signature on it, $Sig_{SK_{ho}}(M_{ho\text{-}dc})$. Then it constructs a message, $msg_{ho\text{-}dc} = M_{ho\text{-}dc} \parallel Sig_{SK_{ho}}(M_{ho\text{-}dc})$, and sends it to the DC.

**DC:** Upon $msg_{ho\text{-}dc}$ reception, the DC performs the following.

1. It verifies the freshness and authenticity of $msg_{ho\text{-}dc}$, and decrypts it to obtain $(S_{id} \parallel M_{ho})$. Once it learns $ID_{ho}$ and $S_{id}$, it checks if $S_{id}$ is a valid session ID for this HO, i.e. if a message containing this session ID was previously sent to this specific HO. It then constructs $M_{dc\text{-}bs} = (S_{id} \parallel TS_{dc} \parallel M_{dc})$, where $M_{dc} = (PSs \parallel (C, T)_{ho\text{-}pm} \parallel Ns)$, generates a signature on it, $Sig_{SK_{dc}}(M_{dc\text{-}bs})$, and constructs a message, $msg_{dc\text{-}bs} = M_{dc\text{-}bs} \parallel Sig_{SK_{dc}}(M_{dc\text{-}bs})$. Finally, $msg_{dc\text{-}bs}$ is sent to the BS over the anonymous communication channel previously used.

**BS:** Upon $msg_{dc\text{-}bs}$ reception, the BS performs the following.

1. It verifies the freshness and authenticity of $msg_{dc\text{-}bs}$ before checking the validity of $S_{id}$, i.e. checking if this session ID is the same as one previously generated by the BS. It then increases the counter by one, i.e. $ct_{pm\text{-}bs} = (ct_{pm\text{-}bs} + 1)$. It encrypts $ct_{pm\text{-}bs}$ and $M_{dc}$ using the session key, i.e. $(C, T)_{bs\text{-}pm} = AE_{K_s}(ct_{pm\text{-}bs} \parallel M_{dc})$, and constructs and sends a message, $msg_{bs\text{-}pm} = (C, T)_{bs\text{-}pm}$, to the PM.

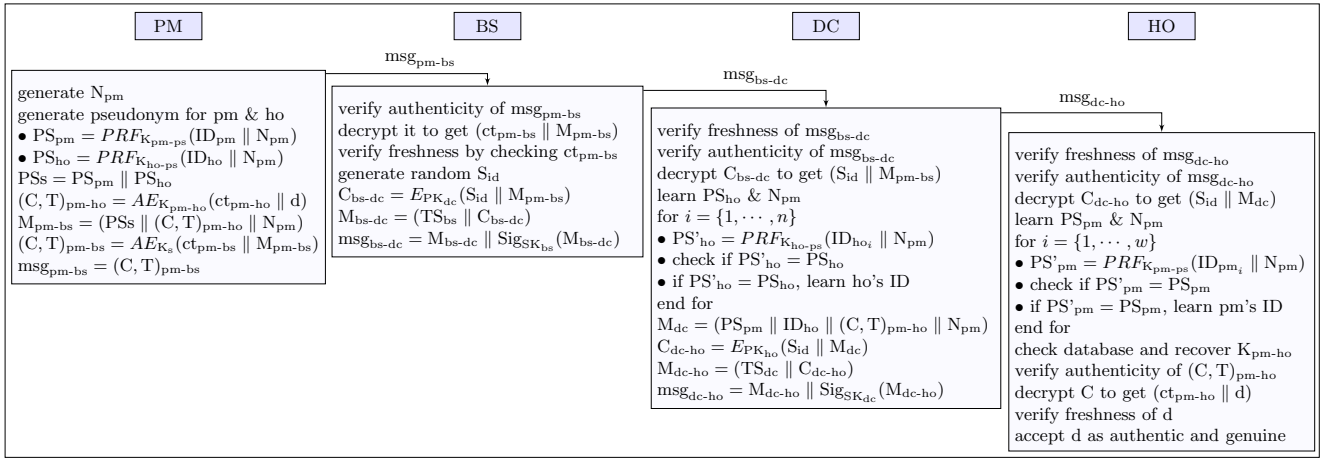


**Fig. 3.** Medical data reporting protocol.

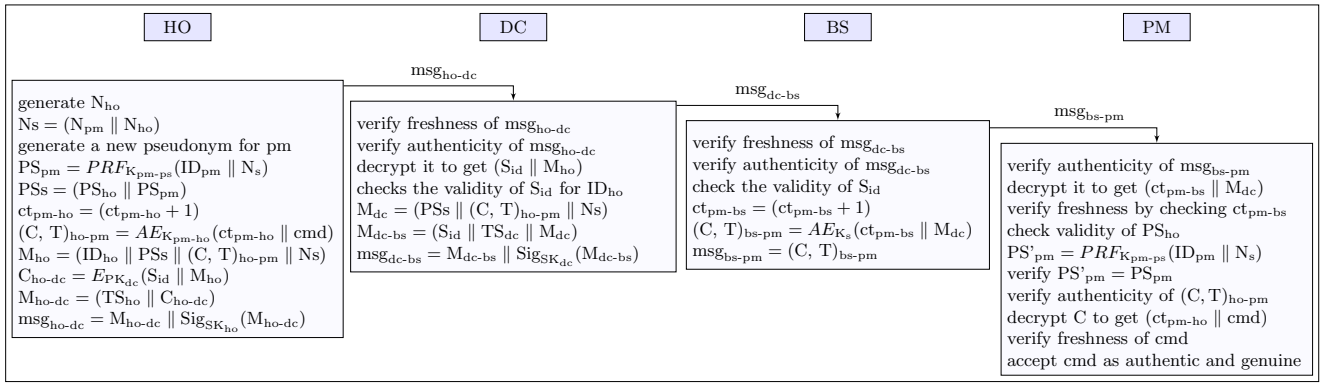


**Fig. 4.** PM's reprogramming protocol.

**PM:** Upon $msg_{bs\text{-}pm}$ reception, the PM performs the following steps.

1. It verifies the authenticity of $msg_{bs\text{-}pm}$, decrypts it to obtain ($ct_{pm\text{-}bs} \parallel M_{dc}$) and verifies the freshness of $M_{dc}$ by checking $ct_{pm\text{-}bs}$. It then verifies the validity of $PS_{ho}$, i.e. checks if $PS_{ho}$ has been previously generated at the PM.

2. It uses $K_{pm\text{-}ps}$, its own ID and $N_s$ to verify the PM's pseudonym, $PS'_{pm} = PRF_{K_{pm\text{-}ps}}(ID_{pm} \parallel N_s)$.

3. It verifies the authenticity of $(C, T)_{ho\text{-}pm}$, decrypts it to obtain ($ct_{pm\text{-}ho} \parallel$ cmd) and verifies the freshness of cmd. If the verifications are correct, it accepts cmd as an authentic and genuine command sent from the patient's HO.

## 5 Security Analysis

**Message Authenticity:** Messages exchanged between any of the entities contain either a digital signature of the message originator or a MAC. Assuming that a standard digital signature scheme (e.g. RSA or Schnorr variant of ECDSA [8]) or a MAC algorithm (e.g. AES CBC-MAC) are used, our protocol guaranties source authentication, message integrity and non-repudiation (only with digital signatures). Thus, attacks that attempt to modify the messages in transit can be detected. All messages include either a counter or a timestamp to ensure freshness, and hence prevent replay attacks (satisfy (S1) and (S2)).

**Confidentiality of Medical Data and Commands:** Medical data and commands to modify the PM's settings are always encrypted twice (i.e. in two encryption layers). The inner-layer encryption is carried out between the PM and the HO. This provides end-to-end security. The outer-layer encryption is performed between any two communicating entities, and used to hide the pseudonyms from adversaries. In addition, it helps to prevent some types of denial-of-service attacks (satisfy (S4)). Assuming that a standard encryption scheme (e.g. AES) is used, it will be hard for eavesdroppers to learn the content of the messages. Only authorised medical staff will be able to access the medical data or send commands (satisfy (S3)).

**Patient's Identity Privacy:** Each message exchanged between the PM and the HO contains a distinct pseudonym to hide the PM's real ID. This pseudonym is generated using a PRF, e.g. AES-128, and the symmetric key that is known only to the PM and the authorized HO. AES-128 can be used as a PRF as long as the number of messages for one key is less than $2^{40}$, which corresponds to more than 4,000 encrypted messages per second exchanged between the HO and the PM (assuming that the battery lasts 7 years). All unauthorized internal entities (i.e. BSs and DC) as well as external adversaries will not be able to obtain the PM's real ID. Only the authorized hospital can recover the real ID of the PM, and link the medical data to a specific patient (satisfy (P1.1)).

**Hospital's Identity Privacy:** For each pair of messages exchanged between the PM and the HO, the PM generates a distinct pseudonym to hide the real ID of the hospital where it sends medical data. This pseudonym is generated using a PRF, e.g. AES-128, and the symmetric key shared between all PMs and the DC. As explained above, AES-128 is a secure PRF if the number of encrypted

messages is less than $2^{40}$. However, external adversaries cannot have access to the pseudonyms produced by the PRF, since pseudonyms are always sent in an encrypted format between the communicating entities (satisfy (P1.2)).

**Location Privacy:** A low-latency anonymous channel (e.g. a Mix network) between the BSs and the DC in combination with a group signature scheme prevents the DC from learning the location of the BS while being used by the patient (satisfy (P1.3)).

**Session Unlinkability:** Fresh and random pseudonyms are used in each message exchanged between the PM and the HO. Therefore, by looking at the exchanged messages, no unauthorized entity can link different sessions or learn if two messages have been sent by the same PM (satisfy (P1.4)).

**Protection against Stolen BSs or Pre-owned PMs:** Since BSs are simply relay devices that do not have any pre-installed shared secrets with PMs, adversaries who get a BS cannot access the content (i.e. medical data and commands) of the messages exchanged between the PM and the HO. In addition, adversaries who obtain a new or a pre-owned PM cannot send data to the HO. Upon a PM replacement, the old PM's ID and the corresponding keys are removed from the database so that the patient is no longer linked to the old PM.

## 6 Conclusions

In this paper, we proposed a protocol that provides end-to-end security between a PM and a HO while preserving the patient's privacy. Each PM uses two fresh pseudonyms for hiding the unique PM's ID and the HO where the medical data is sent. These pseudonyms allow the DC to forward the medical data to the authorized HO without learning to whom the data belongs to, and prevent adversaries from discovering the PM's real ID and to which hospital the medical data is sent. In addition, all BSs sign their messages using a group signature scheme and send them to the DC over an anonymous channel. This allows (i) the DC to verify the authenticity of the messages and (ii) the HO to link the medical data to the patient without learning which specific BS sent the messages (i.e. the location of the patient). Moreover, we presented an IPI-based key establishment protocol that allows a PM and a BS to agree on a symmetric key without using public-key cryptography or any pre-installed shared secrets.

## 7 Acknowledgments

The authors would like to thank George Petrides and the anonymous reviewers for their helpful comments. This work was partially supported by KIC InnoEnergy SE via KIC innovation project SAGA, and the Research Council KU Leuven: C16/15/058.

## References


1. RFC3610: Counter with CBC-MAC (CCM). https://tools.ietf.org/html/rfc3610.
2. Federal Communications Commission. MICS Medical Implant Communication Services, FCC 47CFR95.601-95.673 Subpart E/I Rules for MedRadio Services.
3. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO 2004, USA, Aug 15-19*, volume 3152, pages 41–55.
4. C. Castelluccia and P. Mutaf. Shake them up!: A movement-based pairing protocol for cpu-constrained devices. In *Proc. of the 3rd International Conference on Mobile Systems, Applications, and Services*, pages 51–64, NY, USA, 2005.
5. Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, pages 523–540, 2004.
6. C. Gehrmann, C. J. Mitchell, and K. Nyberg. Manual authentication for wireless devices. *RSA Cryptobytes*, 7(1):29–37, 2004.
7. 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 *Proc. of the 29th Annual IEEE Symposium on Security and Privacy*, pages 129–142, May 2008.
8. D. Johnson, A. Menezes, and S. Vanstone. The Elliptic Curve Digital Signature Algorithm. *Int. Journal of Information Security*, 1(1):36–63, 2014.
9. J. Ko, J. H. Lim, Y. Chen, R. Musvaloiu-E, A. Terzis, G. M. Masson, T. Gao, W. Destler, L. Selavo, and R. P. Dutton. Medisn: Medical emergency detection in sensor networks. *ACM Trans. Embed. Comp. Syst.*, 10(1):11:1–11:29, Aug. 2010.
10. D. Malan, F. J. Thaddeus, M. Welsh, and S. Moulton. CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care. In *MobiSys Workshop on Applications of Mobile Embedded Systems*, pages 12–14. ACM, 2004.
11. 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.
12. J. W. P. Ng, B. P. L. Lo, O. Wells, M. Sloman, N. Peters, A. Darzi, C. Toumazou, and G. Z. Yang. Ubiquitous Monitoring Environment for Wearable and Implantable Sensors. In *UbiComp – 6th Int. Conf. on Ubiquitous Computing*. 2004.
13. A. Ortiz, J. Munilla, and A. Peinado. Secure wireless data link for low-cost telemetry and telecommand applications. In *Electrotechnical Conference. MELECON. IEEE Mediterranean*, pages 828–831. IEEE, 2006.
14. A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. SPINS: Security Protocols for Sensor Networks. *Wirel. Netw.*, 8(5):521–534, Sept. 2002.
15. M. Rostami, A. Juels, and F. Koushanfar. Heart-to-heart (H2H): Authentication for Implanted Medical Devices. In *Proc. of the ACM SIGSAC Conf on Computer and Communications Security*, CCS, pages 1099–1112, NY, USA, 2013.
16. K. Sampigethaya and R. Poovendran. A Survey on Mix Networks and Their Secure Applications. *Proceedings of the IEEE*, 94(12), Dec. 2006.
17. H. Savci, A. Sula, Z. Wang, N. Dogan, and E. Arvas. MICS transceivers: regulatory standards and applications. In *SoutheastCon. Proc. IEEE*, April 2005.
18. F. Stajano and R. J. Anderson. The Resurrecting Duckling: Security Issues for Adhoc Wireless Networks. In *Proc of the 7th Int. Workshop on Security Protocols*, pages 172–194, London, UK, 2000.
19. F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li. IMDGuard: Securing implantable medical devices with the external wearable guardian. In *Proc. INFOCOM*, pages 1862–1870, April 2011.