IMDGuard: Securing Implantable Medical Devices with the External Wearable Guardian

Fengyuan Xu*, Zhengrui Qin*, Chiu C. Tan†, Baosheng Wang‡, and Qun Li*

Abstract—**Recent studies have revealed security vulnerabilities in implantable medical devices (IMDs). Security design for IMDs is complicated by the requirement that IMDs remain operable in an emergency when appropriate security credentials may be unavailable. In this paper, we introduce IMDGuard, a comprehensive security scheme for heart-related IMDs to fulfill this requirement. IMDGuard incorporates two techniques tailored to provide desirable protections for IMDs. One is an ECG based key establishment without prior shared secrets, and the other is an access control mechanism resilient to adversary spoofing attacks. The security and performance of IMDGuard are evaluated on our prototype implementation.**

I. Introduction

The rapid advances of bioengineering are introducing a boom of wireless accessible IMDs. Millions of patients experience the benefits from IMDs in regulating heart rhythm, controlling blood pressure, improving hearing, providing visual sight, and so on. In the near future, IMDs are expected to be Internet aware, and become a crucial component in pervasive systems such as smart homes and hospitals, making IMDs' security important. Researchers have identified that IMDs are facing potential security threats which may cause life threatening consequences. Recent investigations on pacemakers [8] revealed security vulnerabilities on existing commercial offerings that allow, among other attacks, a malicious entity to reprogram the IMD. Thus, any vulnerability has to be addressed before further integration of IMDs into an intelligent environment can be realized.

Unlike conventional embedded systems, engineering security into IMDs presents the unique challenge. Security mechanism enforcing protection all the time may lead to troubles when safety tops secure operations of IMDs. To illustrate, consider an ER doctor, who is not recognized as legitimate operator in terms of security, may have to access the IMD to save the patient's life in an emergency situation. Temporary authorization to the doctor is not a reliable solution since the IMD owner in this circumstance may be physically incapable of doing this or remote trusted authority is not available.

Intuitively, we want a security mechanism resembling an ON/OFF switch to control security protections. The switch can be triggered OFF in an emergency without assistance from the patient, but in a non-emergency situation, the patient has full control over who has access to his device. (This does open the IMD to attack in an emergency, but life-threatening conditions trump such concerns.) Researchers have advocated pairing the IMD with an external device to provide security for the IMD, where in an emergency, the doctors can simply remove the external device and proceed to interact with the IMD without further hindrance.

There are two challenges when using an external device to protect the IMD. First, the external device and the IMD should have a means of establishing a secret *without prior knowledge*. In other words, we should not pre-deploy any secret inside the IMD. This is to avoid situations where the user is unable to recall the pre-deployed secret and needs to rekey the IMD. The conventional solution is to use a manual switch to "reset" the IMD, but since the IMD is implanted inside the patient's body, this solution is unsuitable. The second challenge is to have a reliable method to prevent an adversary from convincing the IMD that the external device is absent. Since the IMD is inside the patient's body and the external device is placed outside the body, we have to rely on the wireless communication to relay information. This opens up a possibility for the adversary to jam the channel to create the appearance that the external device is absent. Thus it is crucial to ensure IMDs to correctly distinguish between real emergency and an attack.

In this paper, we propose IMDGuard, a security scheme for implantable cardiac devices[1], which are implanted to monitor or treat *cardiac* medical conditions. Those IMDs are one widely utilized group of medical devices, and examples include implantable cardioverter-defibrillator, pacemaker, and ECG (electrocardiogram) sensor. IMDGuard leverages the Guardian, an external wearable device, to coordinate interactions between the IMD and the doctor in such a way that provides the security in a regular condition, and safely allows access in an emergency. The patient's ECG signals are exploited to extract keys shared exclusively between the IMD and Guardian. This key extraction scheme does not need any pre-distributed secret so that it is easy to rekey the IMD when the Guardian is lost or malfunctioning. Moreover, it makes the adversary unable to forge fake Guardians except through physical contacts with the patient. IMDGuard also can effectively prevent the adversary capable of jamming from spoofing the IMD that the Guardian is absent. The adversary's deception will be revealed by collaborations between the IMD and Guardian through the notification mechanism based on the defensive jamming.

*Department of Computer Science, The College of William and Mary, E-mail: {fxu, zhengrui, liqun}@cs.wm.edu.

†Department of Computer and Information Sciences, Temple University, E-mail: cctan@temple.edu. This work was done when the author was with the College of William and Mary.

‡Computer School, National University of Defense Technology, P.R. China. E-mail: bswang@nudt.edu.cn. This work was done while the author was visiting the College of William and Mary.

[1] We refer IMDs as implantable cardiac devices in the rest of paper

U.S. Government work not protected by U.S. copyright

Our IMD security scheme makes the following contributions:

1) Previous work in ECG based key agreements did not properly extract the randomness of input data or correctly evaluate final outputs. In contrast, we are the first to propose a rigorously information-theoretic secure extraction scheme, and evaluate its performance on resource constrained embedded systems.
2) To the best of our knowledge, We are the first to finalize and implement a comprehensive secure protocol for the previously proposed architecture that uses external devices as an authentication proxy to protect the IMD. Besides, Our design is tailored for IMDs and requires no special hardware. For example, the key extraction scheme of IMDGuard is proposed based on the existing functionality of the IMD, and the wearable device does not need powerful transmitter modules to defend against the adversary's spoofing attacks.
3) We perform extensive experiments on our prototype to evaluate the validity and performance of the IMDGuard.

The rest of the paper is as follows. We review the related work in Section II, and the background and problem formulation in Section III. Sections IV and V detail the IMDGuard scheme including running time protocols and key initialization between IMD and Guardian, and Section VI describes our prototype implementation. We provide evaluation on our scheme in Section VII, and conclude in Section VIII.

II. Related work

The increase use of IMDs has motivated researchers to study the security issues on such devices [6]–[8]. Their proposed solution while secure, does not address what happens in an emergency situation where the doctors are unable to obtain the necessary keys from the patient.

Later work by [5] explored the concept of safety, and proposed the idea of *fail-open*, a property to physically circumvent the IMD's security protection in an emergency, through the use of an external device. This introduces a new security threat whereby an adversary may attempt to induce the fail-open state to access the IMD. Our proposed protocol also provides the fail-open property, but differs from [5] in three aspects. First, our design avoids the periodic message broadcasting which consumes considerable battery power and exposes patients to privacy risks. Second, our solution protects the IMD without any assumption on the adversary's transmission capability. Third, our scheme is comprehensive and evaluated on resource constrained embedded systems.

Our solution includes a spoofing attack resistant mechanism related to jamming. Jamming in sensor networks have been studied by [9], [13], [19]. However, such jamming protocols do not consider the features of the IMD, and cannot be directly used in our problem. Other anti-jamming strategies like [16] and Direct-Sequence Spread Spectrum modulation also cannot be applied because of the hardware limitation and the band regulation [2].

Our solution also includes a key extraction algorithm from ECG signals to secure the link between the IMD and the Guardian. The idea of using physiological signals to secure inter-sensor communications was first introduced in [4], and Poon *et al.* [14] put this scheme into practice for ECG and PPG (photoplethysmogram) signals. Inter-pulse intervals (IPIs) of heartbeats are exploited to extract keys in [3]. For 16 consecutive individual IPIs, the ending time in millisecond (ms) of each IPI is calculated, setting 0 as the start time of the first IPI. Then the 7th and 8th digits of the binary representations of the ending times are extracted to form the key. Even though the extracted binary sequences can pass several NIST [15] randomness tests, they are actually not random as what they look. Since the average IPI is about 850 milliseconds, the 7th and 8th digits of the ending time are not random at all. The randomness lies in the lower digits, so does the error. Compared with it, our solution explored a new way to correctly utilize IPIs for extracting randomness.

A faster scheme was proposed by [17] where the coefficients of Fast Fourier Transform (FFT) on sampled ECG signals are used to extract keys, however, the paper also does not give a rigorous analysis whether input samples contain sufficient entropy to generate a key with required entropy bits, and it evaluates the key after hashing, which is not logically correct. Our key extraction scheme differs from this work in two facets. First, we give rigorous information theoretical study on the randomness of the physiological feature from which the key is extracted. Second, we show that the adversary cannot get any knowledge about the generated keys except he can measure the ECG signals simultaneously without being caught.

III. Background and Formulation

In this section, we first show the configuration of IMDGuard, then the adversary model, and finally the approach against the adversary.

A. IMDGuard Configuration

IMDGuard has three components, the IMD, Guardian, and programmer. The IMD, once implanted, is expected to remain inside the body for an extended period of time. The programmer, as an outside controller, provides doctors an interface to interact with IMD through radio frequency transmission for adjusting running parameters, changing operation modes, or retrieving stored data. Above two are standard wireless programmable medical instruments. The Guardian is a wearable device with more power and computational resources than the IMD. This Guardian works as a proxy for the IMD and performs the authentication on its behalf. Both the IMD and the Guardian are capable of measuring ECG signals. The interactions of these components are illustrated in Fig. 1. Link α represents the access control process between the Guardian and the programmer. Link β represents the initial pairing process between the IMD and the Guardian. Link γ represents the secure communication protected by the key assigned by the Guardian to both the IMD and the programmer.

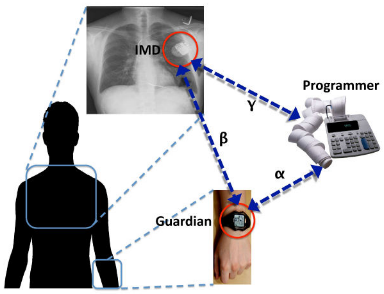

Fig. 1: Communication interactions in IMDGuard. Partial image is adopted from [11].

B. Adversary Model

We consider an adversary whose goal is trying to program to or retrieve data from the IMD without being caught. The adversary succeeds if he is able to access information from the IMD in the presence of the Guardian. Disruption attacks like denial-of-service are excluded in our adversary model. We assume the adversary cannot physically measure the patient's real-time ECG signals without being detected. We also assume that there is no adversary in an emergency situation. This is reasonable since in such a scenario, the patient with the IMD is likely to be in a secure facility like an ER room in a hospital which can limit the presence of adversaries.

We classify the attack strategy of adversary against IMDGuard into two aspects. The first is when the adversary tries to impersonate the Guardian by deriving the key shared between the IMD and the Guardian from either brute force searching or historic medical records of the patient. The second is when the adversary may spoof the absence of the Guardian by selectively jamming the messages from it, in order to convince the IMD to disable safety protection and switch to the emergence status.

C. IMDGuard Overview

In IMDGuard, the Guardian performs two essential functions. First, the Guardian is used to control which mode, *regular* or *emergency*, the IMD should enter. When the patient is wearing the Guardian, the IMD should function under the regular mode. In a regular mode, the programmer requiring to interact with the IMD will first be authenticated by the Guardian, which will then issue the appropriate keys to both the IMD and programmer. When the IMD does not detect the presence of the Guardian, the IMD should enter emergency mode. The advantage is that in an emergency, the doctor will be able to physically remove the Guardian and have unfettered access to the IMD.

Second, the Guardian will authenticate the programmer on behalf of the IMD. This will conserve the IMD's battery by reducing the number of operations performed by the IMD. This also simplifies overall IMD design, since the IMD does not have to maintain cryptographic materials such as asymmetric keys and access control lists.

We assume that the Guardian will always be worn by the patient. It is reasonable since the Guardian can take the form of a watch and the patient can wear it all the time. We also assume that the adversary cannot physically remove the Guardian without the patient being aware of it.

We do not assume the IMD must associate exclusively with one Guardian. Thus before making the Guardian effective, it needs to be initialized by sharing a secret key between the IMD and this Guardian, so that they recognize each other. Moreover, in the extreme case that the Guardian current worn is broken or lost, a new Guardian can be paired with the IMD easily without the need of retrieving old key or resetting IMD.

To realize this functionality, one key feature of IMDGuard is a secure key establishment scheme based on ECG signals. Both the IMD and the Guardian locally sample the same random source simultaneously, the patient's ECG, and then extract a symmetric key from ECG features after ECG delineation. Unlike the Diffie-Hellman key exchange or wireless based key extractions, this scheme is robust against man-in-the-middle attacks as long as the adversary cannot physically measure real-time ECG signals of the patient.

The other key feature of IMDGuard is the spoofing attack resistant mechanism. If the adversary attempts to persuade the IMD to enter the emergency mode by jamming all messages transmitted from the Guardian, the Guardian still can announce its presence to the IMD by *jamming the IMD's transmission of the challenge message*. The intuition is that the Guardian may hardly block the transmission from the adversary to the IMD, since it has no knowledge about the adversary's hardware and capabilities. Instead, the Guardian can be calibrated to the parameters of its own IMD, and can always successfully jam its IMD's transmissions.

IV. Protocol Design in IMDGuard

Here, we present the protocols of IMDGuard. We assume that the IMD and Guardian have already paired with a shared secret key after key establishment phase, which is described in the following section. We assume the Guardian has a list of legitimate programmers and their corresponding public keys. This information can securely be installed when in the hospital. Table I summarizes the notations used.

TABLE I: Table of notations

G	the Guardian
P	the Programmer
ni_j	the ith nonce generated by j, $j \in \{IMD, G, P\}$
$H(\cdot)$	standard cryptographic hash function, e.g., SHA-1
SK	the shared secret key between the IMD and the Guardian using ECG based key extraction (Section V)
PK_j^+	the public key of j, $j \in \{G, P\}$
PK_j^-	the private key of j, $j \in \{G, P\}$
TK	the temporary symmetric key used for one session
ID	the identification of the IMD

A. Basic IMD Protocol

The IMD will periodically wake up to determine whether there is any request from the programmer. After the IMD receives a request from the programmer, the IMD will execute Algorithm 1. The IMD will send back its ID, and a random nonce $n1_{IMD}$, which is used as the session identity to resist against the replay attack. Then, the IMD starts a timer T_1 to wait for the Guardian, if present, to notify it to run the *regular condition protocol*. In the case that there is no message from the Guardian when T_1 times out, the IMD will run the *emergency condition protocol*.

Algorithm 1 Basic IMD algorithm

```
1: Send back to the $P$, ID and $n1_{IMD}$
2: Start waiting timer $T_1$
   (Guardian, if present, will execute authentication protocol (Fig. 2) during $T_1$)
3: while $T_1$ time out == FALSE do
4:     if receive valid message from $G$ then
5:         Regular condition, not an emergency.
6:         Execute Regular Condition Algorithm (Section IV-C)
7: Possible emergency condition.
   Execute Emergency Condition Algorithm (Section IV-D)
```

B. Guardian Authenticating Programmer

In Step 2 of Algorithm 1, the Guardian will authenticate the programmer within the time period T_1. The authentication protocol is shown in Fig. 2.

A random nonce $n1_G$ is signed by the programmer, and sent back to the Guardian along with another random nonce $n1_P$. The signature of the programmer is verified by the Guardian. If it is not valid, the Guardian will inform the IMD to deny the session request through the message $\{NO, n1_{IMD}\}_{SK}$ (it will also inform the programmer the authentication is failed and session is denied). If valid, the Guardian will assign a temporary session key TK to both the IMD and the programmer for the secure communication. We let the IMD use symmetric keys when communicating with the programmer and Guardian to reduce the computational load on the IMD. The Guardian and programmer use public keys to authenticate each other for better key management.

G : Overhears msg in step 1 of Algorithm 1 (1)

$G \rightarrow P : n1_G$ (2)

$G \leftarrow P : \{n1_G\}_{PK_P^-}, n1_P$ (3)

G : Verify the signature withPK_P^+ (4)

If incorrect, then

Deny the request and inform IMD and P

Authentication phase completed

If correct, then

Accept the request, and continue step(5)

$G \rightarrow IMD : \{YES, TK, n1_{IMD}\}_{SK}$ (5)

$G \rightarrow P : \{YES, TK, n1_P\}_{PK_P^+}$ (6)

Fig. 2: Guardian authentication Programmer

C. Regular Condition Protocol

When the IMD enters the regular condition (Step 6 in Algorithm 1), it will execute the protocol shown in Fig. 3. After decrypting the message R, the IMD will deny access if it receives a NO message. Otherwise, a YES message indicates that the programmer has been authenticated, and the session key for Steps 3 and 4 is TK.

$G \rightarrow IMD : \{R\}_{SK}$ (1)

IMD : Decrypt to derive R (2)

If$R == \{NO, n1_{IMD}\}$, then

Deny the request from P, and go to sleep

If$R == \{YES, TK, n1_{IMD}\}$, then

Accept req from P, and continue step(3)

$P \rightarrow IMD : \{command\}_{TK}$ (3)

$P \leftarrow IMD : \{response\}_{TK}$ (4)

Fig. 3: IMD regular condition protocol

$P \leftarrow IMD : n2_{IMD}$ (1)

IMD : Wait for time T_2 (2)

$P \leftarrow IMD : n3_{IMD}$ (3)

$P \rightarrow IMD : ans$ (4)

IMD : Check ans (5)

if $ans! = H(n2_{IMD} \oplus n3_{IMD})$, then

Deny the request from P, and go to sleep

if $ans == H(n2_{IMD} \oplus n3_{IMD})$, then

Accept req from P, and continue step(6)

$P \rightarrow IMD : command$ (6)

$P \leftarrow IMD : response$ (7)

Fig. 4: IMD emergency condition protocol

D. Emergency Condition Protocol

When the IMD enters the emergency mode, it will execute the protocol in Fig. 4, and send the first portion of the challenge, a random nonce $n2_{IMD}$. After waiting T_2 time, the IMD sends the second portion of the challenge, $n3_{IMD}$. Assuming that the Guardian has been physically removed, the programmer will transmit back the correct answer to the challenge, $H(n2_{IMD} \oplus n3_{IMD})$. If the Guardian is present, the programmer will be unable to return the correct answer. We explain how the Guardian disrupts this in the next subsection.

E. Spoofing Attack Resistant Protocol

Here, we show how our protocol is resilient to adversary's spoofing attack based on jamming. The adversary can attempt to jam the communications between the IMD and Guardian to induce the IMD to enter the emergency mode in Step 7 of Algorithm 1. In other words, the adversary will jam the channel for length of time period T_1. Since the IMD does not receive any response from the Guardian, the IMD will proceed to execute the emergency condition algorithm. For this scenario, the Guardian function, the defensive jamming

G : Overhear the msg in Step 1 of Fig. 4 (1)

G : Jam the msg in Step 3 of Fig 4 (2)

G : Raise a warning alarm if Step 1 occurs frequently (3)

Fig. 5: The notification mechanism of the Guardian

described in Fig. 5, is triggered to block this session. When the Guardian hears the first portion of the challenge message (Fig. 4 Step 1) sent by the IMD to the programmer, the Guardian will realize that the communication link between the IMD and itself is blocked by an adversary with high probability. Then the Guardian will jam the second portion of challenge message from the IMD (Fig. 4 Step 3). This operation is feasible in practice based on the Guardian's loose synchronization through the message in Step 1 of Fig. 4. In other words, the Guardian will be aware that there is another message, which is the jamming target, going to send in T_2 time later. This information can help the Guardian to jam the target message with less effort.

There are two advantages in letting the Guardian to jam the IMD's message instead of the adversary's message. First, the adversary's hardware may be much more powerful than the Guardian, making it difficult to calibrate the Guardian's broadcast strength needed to successfully jam the adversary's signal. Second, the the Guardian can time exactly when to be jamming since it is aware when the IMD will begin broadcast. This conserves the Guardian's power by reducing jamming period.

V. Key Establishment in IMDGuard

In the previous section, we assume there is a secret key already shared between the IMD and Guardian to secure their communication. However, this key establishment is challenging if the IMD and Guardian do not share any secret beforehand. In this section, we introduce a secure key extraction scheme based on the ECG delineation to establish a symmetric secret key bonding the IMD and Guardian together, making adversary impossible to forge the Guardian.

A. ECG Delineation

We conduct the ECG delineation with the wavelet-based algorithms mentioned in [10], [12]. Fig. 6 shows an example result of our wavelet transform based delineation. Using the information of local maxima, minima and zero crossings at different scales in the wavelet transform, the algorithm is able to detect all the significant points of ECG in a heart beat cycle, first the R peak, then Q peak and S peak, followed by T wave and P wave.

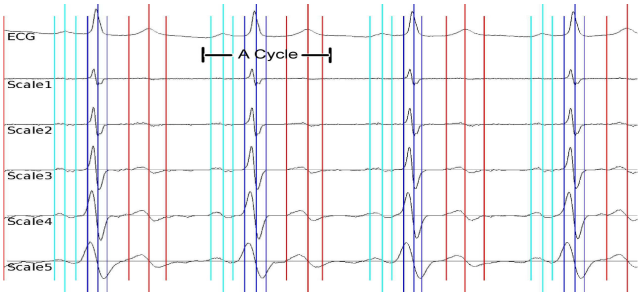

Fig. 6: Wavelet transform of ECG waves at the first five scales. The first panel is the ECG signal, the other five, from top to the bottom, are the corresponding wavelet transforms from scale 2^1 to scale 2^5.

As shown in Fig. 6, in each heart beat cycle, the three blue lines in the middle denote the onset, R peak and offset of QRS complex respectively. The three cyan lines on the left denote the onset, P peak and offset of P wave respectively. The three red lines on the right denote the onset, T peak and offset of T wave respectively.

We implement the algorithms with TinyOS 2.1, with about 1200 lines of code. The high accuracy is achieved to reduce the mismatch rate of IPIs, making the following key extraction efficient.

B. An ECG Feature for Key Extraction

Given the two ECG measurements that are taken at different parts of the human body, we want to extract a symmetric key from them after the delineation. As a fundamental requirement, the key much be *random*. Thus, the key must be extracted from an ECG feature such that: (1) the feature itself is random; and (2) the feature has common places for both the IMD and Guardian.

After the ECG delineating, we have the timing information of all the ECG significant features, namely P wave, QRS complex and T wave, for every heart beat cycle. Since the ECG signals are periodic, to ensure the randomness, we cannot directly use all the delineation points at the same time. Once a feature is chosen, other features in the same heart beat cycle are not totally random any more. For instance, if we know the position of the R peak, we can easily guess into what ranges the Q peak, S peak, T peak and P peak in the same cycle fall. Even for features in different cycles, the positions of features are not totally random. For example, given the position of R peak in one heart beat cycle, that of the following R peak will fall into a small range because the common inter-pulse interval (IPI) is known. (For adults, the common IPI is about 850 ms)

We will use the information of R peaks, which is most salient, to extract keys. Given a consecutive sequence of R peaks, IPIs are obtained by calculating the difference in time of the two consecutive R peaks. Suppose R_i denoting the time of the ith R peak, then $IPI_i = R_{i+1} - R_i$. Since the average value of IPI is quite known, we have to exclude the average value when extracting the key. First we empirically estimate how many random bits can be extracted from each IPI value. We convert IPI values to binary representations, then examine the randomness of each digit of the binary representations. It is clear that the random bits lie in the low digits. For the ith digit, we count the number of samples for the following cases:

(1) n_{00}: if the ith digit of sample j is 0, and so is that of sample $j+1$, then increment n_{00} by 1;

(2) n_{01}: if the ith digit of sample j is 0, and that of sample $j+1$ is 1, then increment n_{01} by 1;

(3) n_{10}: if the ith digit of sample j is 1, and that of sample $j+1$ is 0, then increment n_{10} by 1;

(4) n_{11}: if the ith digit of sample j is 1, and so is that of sample $j+1$, then increment n_{11} by 1;

where $1 \leq j < n$, n is the total number of consecutive IPI samples.

We then calculate the four possibilities $P_{lk} = n_{lk}/(n-1)$, where $lk \in \{00, 01, 10, 11\}$. If the ith digit is random and independent, all the four possibilities should be around 25%.

TABLE II: The quality of randomness of each digit.

i	$P_{00}(\%)$	P_{01}	P_{10}	P_{11}
1	29.2	24.4	24.4	21.9
2	28.9	24.3	24.4	22.4
3	25.2	24.6	24.7	25.5
4	27.9	25.6	25.6	20.9
5	57.5	18.8	18.8	4.9
6	2.5	13.2	13.2	71.1
7	99.1	0.4	0.4	0.0
8	99.7	0.1	0.1	0.0

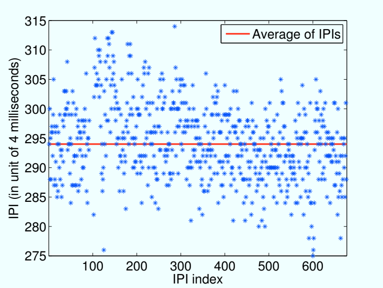

Fig. 7: The fluctuation of IPIs against the average, which is 294 in unit of 4 ms (250Hz sampling rate).

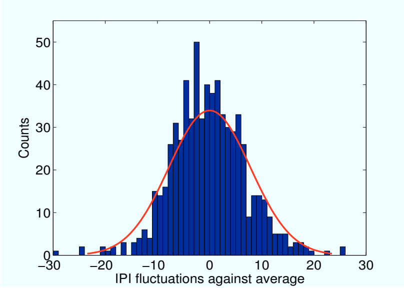

Fig. 8: The normal distribution fitting to the fluctuation of IPIs against the average, with $\mu = 0$ and $\sigma^2 = 61$.

We list the possibilities for the lower 8 digits of IPI samples in Table II. We set a threshold of 5%. As shown in the table, the last 4 digits are random, while the 5th digit is not. For the 5th digit, P_{00} is more than 50% while P_{11} is less than 5%. We also calculate the entropy of the fluctuations directly from the original data, which is around 5. Therefore, we can confidently extract 4 bits from each IPI.

C. Quantization

This subsection will show how to extract 4 random bits from each IPI sample. We cannot use the last 4 digits of IPI's binary representations directly, because the slight difference between the data at both sides may cause big differences in the last 4 digits of the binary representations, leading the mismatch rate to as high as 20%. Actually, the lower 4 digits are the fluctuations of IPIs. Fig. 7 shows the IPI fluctuations against the average value. In terms of entropy, the fluctuations don't lose any entropy of IPI samples. Though the average IPI is quite stable, the difference between individual IPI and the average value is unpredictable. It can be positive, negative or zero. And the value of the fluctuation is quite random in a certain range. So the fluctuations can be chosen as the basis to extract the key.

In the perspective of statistics, the fluctuations shape into a normal distribution. Fig. 8 shows the histogram of the samples in Fig. 7, with a normal distribution fitting. The fitted normal distribution also results in an entropy $\frac{1}{2}log(2\pi e\delta^2) \cong 5$, which is within the range of that resulted from the original data. And we have proved that the last 4 digits are random, which implies that the real entropy is at least 4. In this sense, the distribution of the fluctuations is indeed a normal distribution, or at least close to.

IMDGuard provides the following algorithm to do the quantization. This algorithm is based on the assumption that the fluctuations form a normal distribution. For a normal distribution $N(\mu, \sigma)$, given μ and σ, we can divide the probability density function into 16 consecutive sections such that, in each section, the cumulative possibility density is $1/16$. The domain of each section can be denoted by a function of σ and μ, as shown in Table III. If μ or any starting/ending point of any domain is an integer, we split samples with that value into two portions, with each going into one of the nearby domain. The splitting depends on the sample index. The samples with odd index form one portion, and those with even index form the other. Note that σ is big enough such that every domain contains at least one integer, since the entropy indicates that σ is not small. The purpose of the division is to roughly but not precisely equalize the number of samples in each domain, making the quantization unpredictable. The 16 domains are one-to-one mapping to the 4-bit gray codes.

TABLE III: Normal distribution divided into 16 equal sections.

	Domain		Domain
1	$(-\infty,\ \mu - 1.534\sigma)$	9	$(\mu,\ \mu + 0.157\sigma)$
2	$(\mu - 1.534\sigma,\ \mu - 1.151\sigma)$	10	$(\mu + 0.157\sigma,\ \mu + 0.319\sigma)$
3	$(\mu - 1.151\sigma,\ \mu - 0.887\sigma)$	11	$(\mu + 0.319\sigma,\ \mu + 0.489\sigma)$
4	$(\mu - 0.887\sigma,\ \mu - 0.675\sigma)$	12	$(\mu + 0.489\sigma,\ \mu + 0.675\sigma)$
5	$(\mu - 0.675\sigma,\ \mu - 0.489\sigma)$	13	$(\mu + 0.675\sigma,\ \mu + 0.887\sigma)$
6	$(\mu - 0.489\sigma,\ \mu - 0.319\sigma)$	14	$(\mu + 0.887\sigma,\ \mu + 1.151\sigma)$
7	$(\mu - 0.319\sigma,\ \mu - 0.157\sigma)$	15	$(\mu + 1.151\sigma,\ \mu + 1.534\sigma)$
8	$(\mu - 0.157\sigma,\ \mu)$	16	$(\mu + 1.534\sigma,\ +\infty)$

Since the IMD is measuring the ECG signals all the time, it is able to calculate σ and μ for a long period, say 15 minutes, and store it. During key generation, the IMD can send these parameters to the Guardian. This process doesn't leak any information about the key, since the adversary still doesn't know which sample is in which domain and how many samples are in each domain. The quantization is shown in Algorithm 2.

Algorithm 2 Quantization Algorithm

```
Input: $n$ consecutive IPIs from ECG, $IPI_1$, $IPI_2$, ..., $IPI_n$.
Output: $4n$ bit binary string.
Obtain parameters $\mu$ and $\sigma$
Calculate 16 domains based on Table III: $D_1$, $D_2$, ..., $D_{16}$
Number the 4-bit gray codes: $G_1$, $G_2$, ..., $G_{16}$
Output = $\phi$
for $i \leftarrow 1$ to $n$
    for $j \leftarrow 1$ to 16
        if $IPI_i$ falls into $D_j$
            Output = Concatenate(Output,$G_j$)
```

D. Reconciliation

Due to the high accuracy of the ECG delineation, the two binary strings quantized respectively by the IMD and Guardian have a low mismatch rate. For two 4-bit blocks corresponding to the same heart beat cycle on both sides, one bit is different in most cases if there is a mismatch. In a very few cases, there are two bits different. There is no case that 3 or 4 bits are different. Based on these observations, we design a 2-round reconciliation algorithm. It carries out Round 2 only if Round 1 fails.

Round 1: For each IPI, both the IMD and Guardian get a 4-bit block. Both sides calculate the parity of its own block and exchange this information. If the parities are different, the block is discarded. Otherwise, each side extracts the first 3 bits of the block; the 4th bit is discarded because the parity leaks one bit information. This process continues until both sides get 129 bits. The IMD then hashes it with SHA-1 hash function and sends the hash value to the Guardian. The Guardian compares this hash value with its own, and notifies the IMD. If the two hash values match, the algorithm terminates. Otherwise, Round 2 will be carried out.

Round 2: For the 43 IPIs chosen in Round 1, both the IMD and Guardian calculate the parity of the last 2 bits of each 4 bit block, and exchange this information. Again those blocks whose parities don't match are discarded. For the blocks left, both sides extract the 2nd and 3rd bits; the first bit is discarded since the second parity also leaks one bit information. Obviously, the length of the key is less than 128. Then both sides continue to analyze the following IPIs. At this time, they check two parities at the same time and extract 2 bits from each block which passes the parity check. The process continues until both sides get 128 bits.

VI. Prototype Implementation

A challenge involving IMD experiments is the difficulty in obtaining source codes and open platforms from commercial vendors. In our prototype system, we choose the TelosB with TinyOS 2.1, an open research platform of the resource constrained embedded system as a replacement of the IMD. The related details are described below.

Transmitter Comparison: The TelosB utilizes the CC2420 transmitter for wireless communication. The CC2420 is comparable to the typical Medical Implant Communication Service (MICS) radio like ZL70101 [1] used in IMDs. They both are low power radio devices with similar amount of power consumption during transmission. The ZL70101 expends 5 mA, while 8.5 mA is achievable for the CC2420. Besides, both of them share other common features such as multiple channel and duty-cycle support. The difference between CC2420 and ZL70101 is that MICS radio operates lower frequency band between 402-405 MHz because of the reasonable signal propagation characteristics in the human body. This has no impact on our evaluation since our implementation does not rely on the frequency or number of available channels.

Code size: The code size of each component after compilation is shown in Table IV. ECC [18] is the Elliptic Curve Cryptography we develop to provide public key scheme between the programmer and the Guardian. For reference, a typical IMD produced in 2002 is able to contain 2MB memory [5].

TABLE IV: Code size of our prototype implementation

Module	ROM(*bytes*)	RAM (*bytes*)
IMD	20656	1056
Programmer	20754	1060
Guardian	20614	1050
ECC	42190	1931
Key Extraction	10078	887
ECG Delineation	18720	9652

VII. Evaluation of IMDGuard

The performance evaluation of IMDGuard is divided into three portions. First, we comprehensively assess the quality of the key extracted from ECG signals. Then we conduct a series of experiments on the effectiveness of defensing adversary's spoofing attacks. Finally, we present the efficiency of each critical components in our implementation.

A. Key Establishment

In this section, we will evaluate the key generated according to the algorithms in Section V. The ECG signals are from PhysioBank database (http://www.physionet.org/physiobank). We will address three important characteristics of the key: (1) temporal variance; (2) efficiency; (3) randomness.

1) Temporal Variance: Given a 128-bit key generated by the IMD and Guardian, we want to know whether the adversary can get any help if he can access historic/future records of ECG signals of the patient. Metric used is the hamming distance between the key and any other 128 bit random string before or after it. The hamming distance between two binary strings of equal length is the number of positions at which the corresponding symbols are different. Given two random strings, if they are independent, the possibility of having hamming distance k follows a binomial distribution, which is: $P(k) = \binom{n}{k} p^k (1-p)^{n-k}$, where $n = 128$ and $p = 1/2$. And the mean value of k, a.k.a expected value, is $E(k) = np = 1/2 \times 128 = 64$.

We examine these hamming distances. There is no signal value equal to 0 or 128, and all the points lie between 45 and 85. The closer to the mean hamming distance, which is 64, the denser the points. We plot the possibility of the hamming distances, as shown in Fig. 9. As we can see, the measured data matches very well with the theoretical binomial distribution with $n = 128$ and $p = 1/2$. From the statistical prospective, the 128 bit key generated by our scheme does not relate to the historic ECG or future ECG signals. Thus, even the adversary gets historic or future ECG data of the patient, he cannot get any help from it. This also indicates the randomness of the 128 bit key.

We also conduct the same evaluation between keys generated from ECG signals from different persons, and we get the same result as expected. The historic/future record of the same person does not help the adversary, neither does that of other people. [17] did similar evaluation about their scheme. However, they did so after hashing an identical string between two parties with a one-way hash function. Though they got similar results, their results could not prove what they claimed. Hashing will make a string random, no matter the original string is random or not.

2) Efficiency: In the reconciliation phase, there are two rounds. In the first round, 3 random bits are extracted from each IPI. So it needs 43 IPI to get a 128 bit key. If the first round fails, the algorithm will carry out the second round, extracting 2 bits from each IPI. In this case, it needs 21 more IPIs besides the 43 IPIs in the first round. In 88% cases, the first round succeeds. The second round succeeds all the time, at least we did not find a single failure in all our traces. Thus, on average, it needs 45.5 IPIs, without counting the IPI discarded. During Round 1, about 25% samples are discarded, and during Round 2, only 0.3% samples are discarded. Taking into account the samples discarded, it needs 61 IPIs, corresponding to 45 seconds or so, to generate a key successfully.

3) Randomness: To evaluate randomness of the generated bit stream employed as secret keys, we run the randomness tests in the NIST test suite [15]. There are totally 15 different statistical tests, and we run 9 of them. The other 6 require a very long bit stream that we cannot generate from PhysioBank database. Our bit stream passes all the 9 tests, showing a good quality of randomness.

Evaluations show that even the patient's ECG records cannot help the adversary to predict the key shared between the IMD and Guardian, unless he physically measure patient's ECG signals simultaneously during the key establishment. However it is impossible for the adversary to physically measure patient's ECG without the patient being aware of it. This key establishment is robust to man-in-the-middle attack. If a symmetric key is successfully established, then the Guardian must be legitimate.

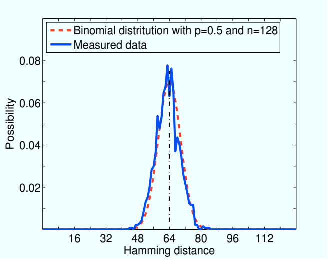

Fig. 9: The hamming distances between the 128 bit stream from current ECG signal and those from historic records fit into a binomial distribution with $n = 128$ and $p = 1/2$.

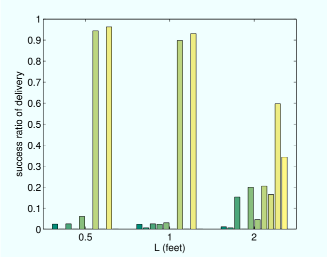

Fig. 10: Results for the effectiveness of Guardian's jamming when targeting at malicious programmer. X-axis represents the distance between the IMD and Guardian, and Y-axis is the success ratio of delivered messages. For each distance, ten trials were conducted.

B. Jamming Related Experiments

There are two jamming related experiments. First, we experimentally validate our decision to jam the IMD's communication instead of the adversary's messages. Second, we examines our defensive jamming method using different settings in terms of the power level and the distance.

We let three TelosB motes to act as the IMD, the Guardian and the malicious programmer (adversary). To concentrate on the jamming performance of the Guardian, these motes are installed with the simplified IMDGuard which will be described in each experiment below, as well as the carrier sensing and random backoff on motes are disabled. All the experiments are taken on a large office table in an indoor environment.

Experiment 1: We vary the distance between the IMD and Guardian from 0.5 feet to 2 feet, and set the malicious programmer 11 feet away from the center point of these two devices. The transmission power of the Guardian is configured to be -15 dBm. This is 10 dB higher than the power of the IMD but is 10 dB lower than the malicious programmer's power. The transmission interval, i.e. the inter arrival time between any two messages, is 20 ms. We then let the adversary send messages to the IMD, while the Guardian is jamming. After the transmission is over, we determine the ratio of messages successfully received by the IMD. The results are shown in Fig. 10. As we can see, messages toward the IMD are able to escape from being jammed with an uncertain probability, low in some cases but high in others. This observation indicates that jamming the adversary's transmission does not work in practice since our malicious programmer settings, such as the relative power strength (10 dB) and location (11 feet), or even more rigorous conditions, can be achieved by an adversary.

We then repeat the experiment again, this time letting the Guardian jam the IMD's transmission. The Guardian is able to *successfully jam all the messages*. This approach is more reliable and effective than jamming the adversary. We omit the figure for the results. The success of defensive jamming is due to the fact that the Guardian is aware of all of the IMD's settings, and that the Guardian is more powerful than the IMD by design.

Experiment 2: In this experiment, the Guardian jams the message the IMD sends to the malicious programmer in the same way as above experiment but under various settings. The distance between the IMD and the malicious programmer is fixed at 1 foot, which is considered as the closest position the malicious could have without being detected by the patient. The Guardian is placed away , from 1 foot to 7 feet, from the center point of IMD and programmer at each different power level. The successful delivery ratios of all transmitted messages in every condition are recorded in Fig. 11. It is evident that, as long as the Guardian is close enough, e.g. within 2 feet to the IMD, the transmission from the IMD toward the malicious programmer is totally blocked even in the extreme testing case that Guardian's jamming power is *20 dB less than the IMD's*. This observation is important because the IMD usually is fallen into this distance range if the Guardian is worn by the patient.

C. Overhead Evaluation

Cryptographic overhead: We write a testing program on TelosB to record the average timing of ECC-based encryption/decryption, SHA-1 hash, Advanced Encryption Standard (AES) for a 20 byte message, and data is shown in Table V.

Communication and Operation Overhead: The timing information for the critical operations under different scenarios is provided in the Table VI. In an authentication case, on average

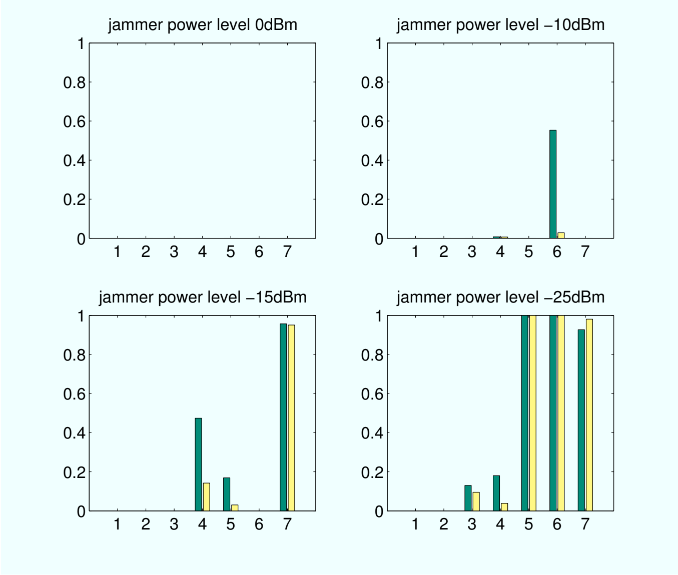

Fig. 11: Results of the effectiveness of Guardian's defensive jamming(acted as jammer) when targeting at the IMD. X-axis means the distance between the Guardian and center point of the IMD and malicious programmer in feet, and Y-axis is the success ratio of delivery. The transmission power of the IMD is set to be -5 dBm during the whole experiment. For each distance, two trials were conducted.

TABLE V: Security Timing Information

Encryption	Decryption	SHA-1 Hash	AES
3.3 s	1.7 s	4 ms	1 ms

the Guardian takes 3821 ms to authentication a programmer in total. It is broken down to (1) 1550 ms for programmer to generate a signature of the given challenge (20 byte random data), (2) 2221 ms for Guardian to verify this signature, and (3) 50 ms for other communication overhead. When the Guardian is not present, the process of emergency condition (Fig 4) costs roughly 512+14=526 ms before the IMD accepts the programmer. If the defensive jamming occurs, the session will be denied by the IMD in about 1501 ms since receiving the request.

TABLE VI: Prototype Timing Information

Overhead in Time (ms)		
Situation	Operation	Overhead

Authentication	Signing(20bytes)
	Verification(20bytes)
	Others
		1550
		2221
		50

Guardian Removed	Challenge Transfer Others	512 14
Guardian Jamming	Session Deny	1501

VIII. Conclusion

In this paper, we propose IMDGuard, a comprehensive security scheme for protecting implantable cardiac devices in terms of both security and reliability. Prototype of IMDGuard is implemented to demonstrate its functionality of securing IMDs in practice.

Acknowledgment

The authors would like to thank all the reviewers for their helpful comments. This project was supported in part by US National Science Foundation grants CNS-0831904, and CAREER Award CNS-0747108.

References

[1] Ultra Low-Power RF Communications for Implanted Medical Applications and Low Duty-Cycle Systems. http://www.zarlink.com/zarlink/lowpower-rf_dutycycle_wp_nov06.pdf.

[2] FCC rules and regulations, MICS Band Plan. 2003.

[3] S. Bao, C. Poon, Y. Zhang, and L. Shen. Using the timing information of heartbeats as an entity identifier to secure body sensor network. *IEEE Trans Inf Technol Biomed*, 12(6):772–9, 2008.

[4] S. Cherukuri, K. Venkatasubramanian, and S. Gupta. BioSec: A biometric based approach for securing communication in wireless networks of biosensors implanted in the human body. *International Conference on Parallel Processing Workshops*, 2003.

[5] T. Denning, K. Fu, and T. Kohno. Absence makes the heart grow fonder: new directions for implantable medical device security. In *HotSec 2008*.

[6] T. Denning, Y. Matsuoka, and T. Kohno. Neurosecurity: security and privacy for neural devices. *Neurosurgical Focus*, 2009.

[7] D. Halperin, T. Heydt-Benjamin, K. Fu, T. Kohno, and W. Maisel. Security and privacy for implantable medical devices. *IEEE Pervasive Computing 2008*.

[8] D. Halperin, T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. Maisel. Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses. In *IEEE Symposium on Security and Privacy, 2008*.

[9] L. Sang and A. Arora. Capabilities of low-power wireless jammers. Technical Report OSU-CISRC-5/08-TR24, Ohio State Univ., 2008.

[10] C. Li, C. Zheng, and C. Tai. Detection of ECG characteristic points using wavelet transforms. *IEEE Trans. on Biomedical Engineering*, 42(1), 1995.

[11] G. Marcus. IMD image. http://knol.google.com/k/-/-/hCjLTV2A/bdmV3w/ICD.CXR.jpg.

[12] J. Martinez, R. Almeida, S. Olmos, A. Rocha, and P. Laguna. A wavelet-based ECG delineator: evaluation on standard databases. *IEEE Trans. on Biomedical Engineering*, 51(4), 2004.

[13] I. Martinovic, P. Pichota, and J. Schmitt. Jamming for good: a fresh approach to authentic communication in WSNs. In *Wisec 2009*.

[14] C. Poon, Y. Zhang, and S. Bao. A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health. *IEEE Communications Magazine*, 44(4):73–81, 2006.

[15] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker. A statistical test suite for random and pseudorandom number generators for cryptographic applications. *NIST*, 2001.

[16] M. Strasser, C. Pöpper, S. Capkun, and M. Cagalj. Jamming-resistant key establishment using uncoordinated frequency hopping. In *IEEE Symposium on Security and Privacy, 2008*.

[17] K. Venkatasubramanian, A. Banerjee, and S. Gupta. Ekg-based key agreement in body sensor networks. In *Computer Communications Workshops*, 2008.

[18] H. Wang, B. Sheng, and Q. Li. Elliptic curve cryptography-based access control in sensor networks. *International Journal of Security and Networks*, 1(3):127–137, 2006.

[19] W. Xu, W. Trappe, Y. Zhang, and T. Wood. The feasibility of launching and detecting jamming attacks in wireless networks. In *MobiHoc 2005*.