ABSTRACT

We present Heart-to-Heart (H2H), a system to authenticate external medical device controllers and programmers to Implantable Medical Devices (IMDs). IMDs, which include pacemakers and cardiac defibrillators, are therapeutic medical devices partially or wholly embedded in the human body. They often have built-in radio communication to facilitate non-invasive reprogramming and data readout. Many IMDs, though, lack well designed authentication protocols, exposing patients to over-the-air attack and physical harm.

H2H makes use of ECG (heartbeat data) as an authentication mechanism, ensuring access only by a medical instrument in physical contact with an IMD-bearing patient. Based on statistical analysis of real-world data, we propose and analyze new techniques for extracting time-varying randomness from ECG signals for use in H2H. We introduce a novel cryptographic device pairing protocol that uses this randomness to protect against attacks by active adversaries, while meeting the practical challenges of lightweight implementation and noise tolerance in ECG readings. Finally, we describe an end-to-end implementation in an ARM-Cortex M-3 microcontroller that demonstrates the practicality of H2H in current IMD hardware.

Previous schemes have had goals much like those of H2H, but with serious limitations making them unfit for deployment—such as naïvely designed cryptographic pairing protocols (some of them recently broken). In addition to its novel analysis and use of ECG entropy, H2H is the first physiologically-based IMD device pairing protocol with a rigorous adversarial model and protocol analysis.

Categories and Subject Descriptors

J.3 [Computer Applications]: Life and Medical Sciences - Medical information systems; C.3 [Computer Systems Organization]: Special-Purpose and Application-Based System, Real-time and embedded systems

General Terms

Security, Design

Keywords

Implantable Medical Devices; IMD Security; Security Protocols

1. INTRODUCTION

Implantable Medical Devices (IMDs) apply continuous monitoring and automatic therapies to the treatment of chronic medical disorders. Implanted either partially or fully in patients’ bodies, IMDs are often sophisticated devices containing batteries, embedded CPUs, radios, sensors, and actuators. As clinical trials validate IMDs’ efficacy [22] and as IMDs treat a broadening range of disorders, their use is growing. For instance, in the United States, over 100,000 patients a year receive implantable cardioverter defibrillators (ICDs) [19], which detect dangerous heart rhythms and administer electric shocks to restore normal activity. Other IMDs include pacemakers, neurostimulators, and implantable drug pumps.

Powered IMDs generally contain radios for communication with external devices called commercial device programmers that can reprogram IMDs and extract patient data from them. Such wireless communication permits safe, non-invasive access to IMDs. But it also brings the security risks of embedded control into the human body. Seminal work by Halperin et al. [20], for example, exposes design flaws in a common ICD that enable attackers to seize unauthorized control wirelessly, and potentially harm victims.

Our work here addresses the tension between two critical requirements for IMDs. On the one hand, IMDs must offer reasonably permissive access-control policies when life-threatening medical events occur. Emergency responders may need to reprogram IMDs or extract patient data from them, and shouldn’t incur fatal treatment delays by contacting care providers for device-specific keys or passwords. On the other hand, overly loose access-control policies expose IMDs to unauthorized wireless access, such as those illustrated by the Halperin et al. attack, that can physically harm patients or expose their medical data [20].

1.1 Heart-to-Heart (H2H)

Our solution is a system called Heart-to-Heart (H2H). H2H implements a simple access-control policy for IMDs that we call “touch-to-access”: A medical instrument (e.g., commercial device programmer), which we call generically a Programmer, obtains access to a patient’s IMD if and only if it has significant physical contact with the patient’s body. An important facet of touch-to-access is forward security. Authentication to the IMD lapses once the instrument loses physical contact with the patient.

Touch-to-access offers a practical and effective balance between the competing access requirements of permissiveness in emergencies and resistance to attacks. The policy is also common sense: Physical access to a patient means the ability to harm or cure.
H2H enforces a touch-to-access policy using a time-varying biometric, often called a physiological value (PV). When a Programmer seeks access to an IMD, it initiates an authentication session. The IMD takes a reading \( \alpha \) of the PV; at the same time, the Programmer takes its own reading \( \beta \). If \( \beta \) is “nearly equal” to \( \alpha \), then the Programmer obtains access to the IMD. (“Nearest equality,” as we explain later, is needed because PV readings are noisy.)

The H2H architecture can in principle rely on any PV, but we focus here specifically on use of the waveform produced by the heart, known as an ECG (electrocardiogram). Thus H2H is well suited for authentication to cardiac IMDs such as ICDs and pacemakers, to-day the largest class of powered IMDs. In principle, though, H2H can work with any IMD equipped to measure ECG anywhere in the body, not just cardiac devices. As we show, suitably processed ECG samples effectively constitute a low-bandwidth stream of random bits well suited to forward-secure authentication.

Briefly, then, in H2H a Programmer and IMD take independent, time-synchronous ECG readings. The IMD compares the two results to enforce a touch-to-access policy on wireless access by the Programmer. Figure 1 depicts the basic operation of H2H.

1.2 Challenges and contributions

Several previous schemes have sought to perform ECG-based pairing with IMDs like H2H, but have had serious shortcomings. Most notably, previous schemes have relied on cryptographic pairing protocols without rigorous adversarial modeling or security analysis. As a result, two of the most recent of them [23, 48] were shown in a 2013 paper [39] to have serious cryptographic flaws.

Thus designing a practical system such as H2H with rigorous security assurances has effectively remained an open problem, one that raises several technical challenges.

The first challenge is demonstrating that ECG is a suitable PV for authentication. H2H derives a key source from the patient’s ECG signal, a sequence of key bits that authenticate a Programmer to an IMD. Secure touch-to-access authentication requires that the key source be truly random, ideally that constituent bits have high entropy and are statistically independent from one another and over time. The key source is then hard for an attacker to guess without physical access to the patient and also ensures forward-security, i.e., that old key source bits don’t reveal future ones.

Previous work has explored the statistical properties of ECG waveforms for key generation, but not the important impact of read error rates on authentication false positive and false negative rates. We present experiments on real patient ECG data showing that it’s possible (with errors) to extract roughly four truly random and statistically uncorrelated bits from the ECG wave corresponding to a single heartbeat. Collection over a 15-second interval suffices for strong Programmer authentication (a false acceptance rate of about \( 2.7 \times 10^{-9} \)) and false rejection rate of \( 10^{-10} \).

We also introduce an optimal scheme in H2H for testing PV validity, i.e., testing \( \alpha \approx \beta \). Our scheme relies on the Neyman-Pearson Lemma, rather than naïvely on Hamming distance, as in previous work.

Good statistical properties, however, don’t ensure that ECG can enforce the touch-to-access policy in H2H. Recently developed systems can read cardiac rhythms remotely via videocamera, and even accurately measure a patient’s pulse. (Skin color changes subtly with cardiac rhythms.) In Section 3.3, we very briefly report our implementation and test results for the best known of these systems [35] and show that it doesn’t reveal statistically significant information about the ECG key source used by H2H.

Given a good key source, a cryptographic pairing protocol is needed between the Programmer and IMD. Two features of H2H make cryptographic pairing a challenge. First, when the Programmer and IMD synchronously sample the key source, their respective readings \( \beta \) and \( \alpha \) are noisy: Often \( \beta \approx \alpha \), but exact equality \( \beta = \alpha \) isn’t obtained. Cryptographic tools such as password-authenticated key agreement (e.g., [5]) require exact equality, while error-tolerant ones, e.g., [14, 24], sacrifice entropy needlessly in our setting here.

Second, the IMD has tight computational and power constraints. Microcontrollers in common use for IMDS today can perform only lightweight cryptography. As IMDS are long-lived devices (with an average lifetime today of five to seven years [15]), and battery replacement requires surgical intervention, power conservation is essential. H2H can protect new IMDS as well as legacy in-vivo IMDS with upgradable firmware, as long as the H2H implementation meets the IMD’s limited memory and computational resources.

We present a new pairing protocol that exploits the fact that key source bits are statistically uncorrelated, and thus that we can treat \( \alpha \) and \( \beta \) as one-time authentication values. We demonstrate its security in a strong adversarial model that includes man-in-the-middle attacks, such as the jam-and-replay attacks feasible in a wireless environment.

Our H2H pairing protocol requires only a low-exponent RSA encryption (tens of modular multiplications) and a few AES invocations and hash computations by the IMD. We demonstrate a full implementation of H2H on an ARM Cortex-M3 processor. In summary then, our contributions are:

- **Statistical characterization of ECG for authentication:** Using real-world ECG measurements [30], we experimentally quantify the extractable entropy in ECG signals. (We demonstrate use of the Neyman-Pearson Lemma to achieve optimal use of this randomness.)

- **Cryptographic pairing protocol:** We present a novel, lightweight, noise-tolerant cryptographic scheme for Programmer-to-IMD pairing in H2H. We formalize an adversarial model and outline proofs of security.

- **Implementation:** We describe a full implementation of H2H in an ARM Cortex-M3 processor, reporting resource requirements such as code size and power consumption, and demonstrating the feasibility of H2H for use in contemporary IMDS.
2. MODELING

Before diving into details on H2H, some basics on ECG and our associated statistical model are in order, as well as discussion of our operational, trust, and adversarial models.

2.1 ECG model

Figure 2 is a schematic depiction of the ECG waveform of a healthy patient. The so-called R-peak is the most prominent feature of the ECG waveform; it corresponds to the “beat” in a heartbeat. The time between two consecutive R-peaks, or the heartbeat duration, is commonly referred to as the inter-pulse interval (IPI). As the figure shows, a typical ECG cycle includes other physiologically significant, named features: The P-wave, which occurs before the R-peak, the QRS complex, which includes sharp valleys before and after the R-peak, denoted by Q and S respectively, and the T-wave, following the S valley.

The heart rhythm is governed by the parasympathetic nervous system, in which many non-linearly interacting processes give the IPI its well-studied chaotic nature [7,31]. The ECG waveform, and parasympathetic network more generally, are influenced by both long-term trends such as circadian rhythm and short-term temperature and respiratory changes. Thus ECG waves simultaneously exhibit both long term patterns and short-term chaotic behavior.

The ECG signal is well modeled as a stochastic process. The existence of long-term patterns renders the process non-stationary, meaning that the parameters of its underlying distribution, e.g., mean and variance, fluctuate over time. We introduce transforms for H2H, however, that eliminate long-term variations, creating a residual signal that is well-approximated by a wide-sense stationary stochastic process, i.e., one whose first and second moments don’t change over time. Previous work observed a strongly random element in IPI time series values [7], motivating later use of IPIs as a natural source of randomness [9].

Like H2H, previous systems also exploited the natural synchronization property of IPIs. Slight shifts in the time interval over which IPIs are derived don’t impact IPI values, which are computed relative to R-peaks, not absolute time.

Entropy and security: We show that it is possible to extract four high-grade random bits per IPI from our processed ECG source, i.e., bits that have maximal entropy and are fully uncorrelated. We use this entropy measure to characterize the security of H2H formally using our main theorem, Theorem 1. An important aspect of our work is quantification and use of the different error rates incurred by individual high-grade random bits via the Neyman-Pearson Lemma. This simple approach marks a notable advance over earlier work, which assigned the same significance to all random bits. Our improvement enables authentication with the optimal, i.e., minimum possible, false positive rate for a given false negative constraint.

Figure 2: A typical ECG waveform from lead V2 which is recorded from chest. R-peak is the most prominent feature.

2.2 Operational and trust models

We envision use of H2H primarily for emergency authentication, when medical personnel, e.g., emergency medical technicians (EMTs), need access to a patient’s IMD, but have no pre-established keys or trust relationship. Rapid and reliable access to the IMD is important. Thus H2H harvests PV randomness efficiently to achieve quick authentication.

We assume no public-key infrastructure (PKI) for certification of trustworthy programmers. The challenges of key revocation, tamperproofing of programmers to prevent key compromise, etc., are substantial. Similarly, we assume it’s impractical for medical personnel to contact an authority on the fly for access credentials, as this approach would require an infrastructure of broad (indeed, worldwide) and robust trust relationships. Thus H2H relies exclusively on the touch-to-access policy for authentication.

The ECG waveform goes flat when an acute heart attack occurs. Similarly, in some late-stage terminal diseases, the parasympathetic network collapses and as a result, the ECG waveform loses most of its entropy. The hugely distorted ECG waveform resulting from such conditions is readily identifiable. In such cases, H2H is designed to enter a promiscuous mode in which any Programmer may access the IMD: For these acute events, the risks of medical failure greatly outweigh those of malicious attack. Additionally, these extreme medical conditions occur rarely.

In non-emergency situations, for instance, when a patient is receiving routine medical care, it may be practical for medical personnel to retrieve device-specific keys. But in unusual situations, e.g., patients traveling abroad, lost keys, and so forth, H2H is additionally useful as a secondary or backup authentication mechanism.

2.3 Adversarial model

We design H2H for a strong adversarial model that assumes the presence of an attacker during a Programmer-to-IMD authentication session. This adversary is active. It has complete network control, i.e., can drop (jam), modify, replay, and forge messages at will. (The adversary can’t compromise the Programmer or IMD, which would render protection of the IMD impossible.) While the presence of an adversary during a medical emergency is admittedly a strong assumption, we believe it is prudent to design robust security for critical systems such as IMDs by default.

Protecting against strong, active adversaries in a pairing protocol isn’t straightforward. As mentioned above, recent breaks of two recent such protocols illustrate the challenge [39].
3. AUTHENTICATION PROCESS

Several papers, e.g., [36, 45, 48], have proposed IMD authentication based on IPIs. Their common motif is extracting the purely uncorrelated random bits from IPIs and utilizing them as a key. As previously stated, IPIs are not independent across time. So most previous approaches were confined to utilizing only a portion of the quantized IPIs for key derivation. These protocols generate a key by quantizing IPIs and then concatenating the three or four least significant bits of the quantized IPIs. The Programmer is then authenticated if and only if the Hamming distance between the keys generated by the two communicating parties is less than a predefined threshold value.

The four least significant bits of IPIs (IPI4) are known to be independently and identically distributed (i.i.d.). We confirmed this characteristic by studying a 2Mbit dataset consisting of 48 half-hour ECG records of 47 subjects from the MIT-BIH Arrhythmia Database [30], 549 two-minute records of 290 subjects from the PTB Database [6], and 250 records of 250 patients from MGH/MF Waveform Database [47], each roughly 30 minutes in length. All of these databases are available at [1]. We note that while these widely referenced databases contain records from patients with abnormal cardiac rhythms, these and similar databases remain standards for the study of ECG-based biometric authentication. (See, e.g., [45, 48]). Such use is substantiated by an extensive body of literature (e.g., [17, 42, 49]) documenting stronger chaotic effects in healthy hearts than in diseased ones.

We applied the NIST suite of statistical tests [41] to our dataset. The outputs of the NIST statistical tests are p-values listed in Table 1. These p-values represent the probability that the dataset was generated by a random process. If this value is less than a threshold (usually 1%), the randomness hypothesis is rejected. Table 1 shows that the p-values are all greater than 1%.

Table 1: p-value of several NIST statistical tests for IPI4. These bits pass all of the random tests.

<table>
<thead>
<tr>
<th>Test</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runs</td>
<td>0.311310</td>
</tr>
<tr>
<td>Rank</td>
<td>0.879647</td>
</tr>
<tr>
<td>Longest runs</td>
<td>0.185359</td>
</tr>
<tr>
<td>Frequency</td>
<td>0.011830</td>
</tr>
<tr>
<td>Universal test</td>
<td>0.013223</td>
</tr>
<tr>
<td>Approximate entropy test</td>
<td>0.464725</td>
</tr>
<tr>
<td>FFT test</td>
<td>0.131301</td>
</tr>
<tr>
<td>Linear complexity</td>
<td>0.612269</td>
</tr>
</tbody>
</table>

The lack of correlation between bits allowed previous work to use a simple Hamming distance metric to compare received and measured bits. Prior work, however, did not characterize and optimize for the error rates on sampled bits resulting from noisy IPI readings. More precisely, we define the error rate as the probability, for a given IPI-derived bit, that two devices, e.g., a Programmer and IMD, read the same IPI at different points on the body but output differing bit values.

Lacking the ability to obtain IPI measurements from IMDS in our lab, we estimate the error rates for IPI-derived bits in the H2H setting by means of two external ECG leads, on the left arm and right arm of subjects. The electric potential of the ECG lead III between the left arm of subjects and their left foot is taken as a surrogate for the ECG from the IMD. Similarly, the electric potential of the ECG lead II between the right arm and left foot is taken as a surrogate for the ECG recorded by the Programmer and IMD. It should be noted that we are using the interval between consecutive prominent R-peaks, instead of analyzing the whole waveform as in [44]. Therefore, H2H is not sensitive to the location of leads on the body. Any other lead configuration could have been used in our analysis. We performed this analysis with other lead configurations and didn’t find any significant variations in the error rate of least significant bits.

In the next step of analysis, the IPI values of these two readings from leads II and III are calculated and quantized. They are then converted to a Gray-code representation to minimize the difference between the quantized bits caused by error in measurement.

The Hamming distance between these two sets of bits is then taken as a surrogate for the error rate between IPI measurements by the IMD and Programmer. The results of our experiment are reflected in the “Error rate” column of Table 2 for IPI values quantized to 8-bit representation. The last column shows the 95% confidence interval of the error rates. We again used an aggregate of MIT-BIH [30], PTB Database [6], and MGH/MF Database [47] to estimate these error rates.

The table shows that the error rate varies considerably across quantized bits; the lower the significance of the bit, the higher its error rate and entropy. In the next subsection, we describe our statistical approach—a departure from the naive Hamming approach of previous work—to compare PV readings during authentication with suitable weighting for individual bit errors.

![Table 2: Average entropy and the estimated error rate of quantized bits, along with their 95% Confidence Interval (CI).](image)

### 3.1 Quantifying the probability of skin contact

It’s convenient to treat the IMD PV α as correct. Error rates then characterize honest or attacker deviation from α.

A PV in H2H includes only the four least significant bits of an IPI, which we denote collectively by IPI4. The bits in IPI4 are i.i.d. random variables. Thus, an adversary that hasn’t made skin contact with a victim, and has no information about IPIs, can at best guess an IPI4 value by assigning random values to each of its constituent bits. Suppose that n is the number of distinct IPI4 instances read in an H2H authentication session. Then the total number of incorrect guesses by Adv for any given one of the four IPI4 bit positions can be modeled as a binomial distribution with Bernoulli trial probability of 0.5, denoted by B(n, 0.5).

The total number of incorrect bit outputs for a given bit position i by a valid Programmer with skin contact can be modeled by another binomial distribution B(n, εi). Here, εi is the error rate of bit i ∈ {1, 2, 3, 4} as given in the third column of Table 2.

Figure 3 compares the distributions of incorrect guesses by an adversary (with no skin contact) against those of a valid Programmer, for n = 20. The solid line is the distribution for the adversary on any of the four bit positions. For the Programmer, x_i = B(n, ε_i) denotes the random variable corresponding to total incorrect values.
in bit position $i$. The adversary is seen to produce significantly more errors than the Programmer in all bit positions.

![Graph showing probability distributions on incorrect guesses by a valid Programmer (dotted line) and an adversary (solid line), for $n = 20$ (reading of 20 IPI values). Here, $x_1, x_2, x_3,$ and $x_4$ denote random variables on Programmer errors for bits positions 1, 2, 3, and 4 respectively. The distribution for the adversary is identical across bit positions. Clear separation is seen between valid and adversarial distributions, even for bit positions with relatively high error rates (e.g., for $x_1$).](image)

Figure 3: Probability distributions on incorrect guesses by a valid Programmer (dotted lines) and an adversary (solid line), for $n = 20$ (reading of 20 IPI values). Here, $x_1, x_2, x_3,$ and $x_4$ denote random variables on Programmer errors for bits positions 1, 2, 3, and 4 respectively. The distribution for the adversary is identical across bit positions. Clear separation is seen between valid and adversarial distributions, even for bit positions with relatively high error rates (e.g., for $x_1$).

3.2 Neyman-Pearson hypothesis testing

Recall that the goal in the H2H authentication process is to determine whether $\alpha \approx \beta$, where the IMD reads PV $\alpha$ and the Programmer submits PV $\beta$. Determining whether Programmer PV $\beta$ is authentic, i.e., resulting from skin contact, may be viewed as a hypothesis test. The underlying hypothesis is that the Programmer’s claimed PV $\beta$ is drawn from the probability distribution of an honest Programmer, instead of an adversary’s guessing distribution.

This observation motivates use of the well known Neyman-Pearson Lemma [32] to distinguish between honest and adversarial authentication attempts. Let error value $u$ denote the set of errors in $\beta$, i.e., bit positions that differ from $\alpha$. In our context, then, Neyman-Pearson Lemma states that for a given maximum acceptable false negative rate, the false positive rate is minimized as follows. For a fixed threshold value $Th$ (whose computation we discuss below), a submitted Programmer value $\beta$ is accepted as valid only when the following criterion holds:

$$\log \left( \frac{P(u)}{Q(u)} \right) > Th,$$

(1)

where $P(\cdot)$ denotes the probability of an adversary with no skin contact yielding error value $u$ and $Q(\cdot)$ denotes the probability of a valid Programmer yielding $u$. We model distributions $P(\cdot)$ and $Q(\cdot)$ according to the binomial distributions discussed above and depicted in Figure 3.

We observe that as the bits in a given bit position $i$ are i.i.d., the correctness of a PV is invariant to which IPI values contain erroneous bits. The authenticity of a PV $\beta$ is thus determined based on only the total number of correct or incorrect values in each bit position. So we can treat $u$ as an equivalence class of PVs. In particular, it’s convenient to regard $u$ as a vector $\vec{u} = (u_1, u_2, u_3, u_4)$, where $u_i$ denotes the total number of IPIs in $\beta$ that are incorrect in bit position $i$—again, that differ from those in $\alpha$.

Now, $P(\vec{u}) = \prod_{i=1}^{4} P(u_i)$ and $Q(\vec{u}) = \prod_{i=1}^{4} Q(u_i)$, where $P(u_i)$ and $Q(u_i)$ denote the probability of a total of $u_i$ incorrect IPI values for bit position $i$ in adversarial and honest scenarios, respectively. It follows that:

$$\log \left( \frac{P(\vec{u})}{Q(\vec{u})} \right) = \sum_{i=1}^{4} \log (P(u_i)) - \sum_{i=1}^{4} \log (Q(u_i)).$$

(2)

Based on Equation 2, we can construct a table containing $\log (P(u_i))$ and $\log (Q(u_i))$ for $i \in \{1,2,3,4\}$ and $u_i \in \mathbb{Z}_n$. Because the error rate of the adversary is 1/2 for all bit positions, $Q(u_i) = Q_j(u_i)$ for any $i,j \in \{1,2,3,4\}$. Thus, it suffices to store $\log (Q(u_i))$ values for $i = 1$ only. Consequently, the full table contains just $(4 + 1) \times (n + 1) = 5n + 5$ values.

Given such a table, performing the Neyman-Pearson test in Equation 1 requires just $(4 + 1) = 5$ table lookups, eight additions, and one subtraction. This computation is online, i.e., performed during authentication. The storage and computational efficiency of our table-driven approach to Neyman-Pearson testing proves valuable in our implementation of H2H, described later.

One issue remains. The Neyman-Pearson Lemma states the existence of threshold $Th$, but doesn’t specify how to compute $Th$. We now describe an algorithm to compute $Th$ in our setting. Note that computation of $Th$ takes place offline; $Th$ need only be computed once and can then be programmed into an H2H-enabled IMD.

**Computing Neyman-Pearson threshold value $Th$:** It turns out that the space $(\mathbb{Z}_n)^4$ of possible values of $\vec{u}$ is relatively small. As we show below, $n \leq 50$ is sufficient to achieve our desired strength of authentication for H2H, meaning that the total number of possible values of $\vec{u}$ is at most $50^4 = 6,250,000$. Consequently, we can compute $Th$ essentially by means of a brute force algorithm.

We specify this algorithm in pseudocode below as Algorithm 1. Algorithm 1 computes $Th$ for a target false-negative rate $F_{N_{Req}}$. It first constructs a matrix $M[n^4][3]$ with $n^4$ rows, one for each $\vec{u} \in (\mathbb{Z}_n)^4$, and three columns. For each row $\vec{u}$, Column 1 contains $P(\vec{u})$, Column 2 contains $Q(\vec{u})$ and Column 3 contains $\log (P(\vec{u})/Q(\vec{u}))$.

The rows of $M$ are sorted in ascending order with respect to Column 3 values. Then, from top (smallest) to bottom (largest), Column 1 values are accumulated in a variable $p$ until the lowest row $\tau$ is reached for which the cumulative value $p \leq F_{N_{Req}}$. The Column 3 value of row $\tau$, namely $M[\tau][3]$, is the optimum threshold value $Th$. By summing Column 2 values over the first $\tau$ rows, we also obtain the corresponding false-positive rate $(FP)$. (A computation failure outputs special symbol $\bot$.)

The dominant cost of Algorithm 1 is sorting. Thus its asymptotic complexity is $O(n^4 \log n)$. In practice, as $n$ is small, the algorithm executes quickly. For example, we implemented Algorithm 1 in MATLAB on a machine with a 3.4GHz Intel i7-2600 CPU running Windows 7. It took 0.2 second to calculate $Th$ for $n = 15$ and around 8 seconds for all values of $n$ from 1 to 25.

Again, we emphasize that Algorithm 1 is run as a precomputation offline, not in the IMD.

**Setting parameters in H2H:** In our H2H implementation, we set the false negative rate $(F_{N_{Req}})$ to $10^{-4}$. In practical terms, this means that a valid Programmer with skin access would fail on average in one in every 10,000 attempts; it would fail twice consecutively at most once in every 100,000,000 attempts. We believe this choice achieves adequate failure resilience for real-world scenarios.
Algorithm 1 Neyman-Pearson threshold $Th$ computation

Inputs: $n, \{e_i\}_{i=1}^4; FN_{\text{Req}}$

Outputs: $Th, FP$

$P[1:n+1] \leftarrow \text{binomial}(n, 0.5);$

for $i = 1$ to 4 do

$Q[1:n+1][i] \leftarrow \text{binomial}(n, e_i)$

end for

$j = 1;$

for $u = (u_1, u_2, u_3, u_4) \in (Z_n)^4$ do

$M[j][1] = \prod_{i=1}^4 P[u_i];$

$M[j][2] = \prod_{i=1}^4 Q[u_i][i];$

$M[j][3] = \log(M[j][1]/M[j][2]);$

$j \leftarrow j + 1;$

end for

sort $M$ on $M[1][3]$ (Column 3);

$p \leftarrow 0; j \leftarrow 0$

while $p \leq FN_{\text{Req}}$ do

$j \leftarrow j + 1;$

$p \leftarrow p + M[j][3];$

end while

if $\tau < 1$ then output $\perp$; halt

end if

$FP \leftarrow \sum_{k=1}^\tau M[k][1];$

$Th \leftarrow M[\tau][3];$


do

Figure 4 illustrates the tradeoffs between false negative rates ($FN_{\text{Req}}$) and false positive rates ($FP$), for varying numbers $n$ of $IPI_4$ values used in authentication. Table 3 gives detailed $FP$ values for our implementation choice $FN_{\text{Req}} = 10^{-4}$ and, for comparison, $FN_{\text{Req}} = 10^{-6}$. Naturally, the lower $FN_{\text{Req}}$, the higher $FP$.

In addition to $FN_{\text{Req}}$, the other key parameter choice in H2H is the number $n$ of $IPI_4$ values measured for authentication. The larger $n$ is, the better $FN_{\text{Req}}$ and $FP$ are. As $n$ grows, though, so does the ECG measurement time in an H2H authentication.

In our implementation, we have chosen to set $n = 15$. Given our choice of $FN_{\text{Req}} = 10^{-4}$, the corresponding false positive rate is $FP = 2.7 \times 10^{-9}$. We chose this FP to demonstrate the feasibility of a strong level of authentication. As a point of comparison, this FP is lower than the false acceptance rate of a typical, eight-digit RSA SecurID token [40]. (While the false acceptance rate for such a token is nominally $1 \times 10^{-8}$, allowances for synchronization errors and multiple tries make it somewhat weaker.) Lower FPs, and thus lower values of $n$, are likely to be acceptable in practice.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$FN_{\text{Req}} = 10^{-4}$</th>
<th>$FN_{\text{Req}} = 10^{-6}$</th>
<th>Avg. read time (secs.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>$3 \times 10^{-4}$</td>
<td>$1.1 \times 10^{-4}$</td>
<td>3–5</td>
</tr>
<tr>
<td>10</td>
<td>$1.15 \times 10^{-6}$</td>
<td>$7.9 \times 10^{-6}$</td>
<td>7–10</td>
</tr>
<tr>
<td>15</td>
<td>$0.2 \times 10^{-9}$</td>
<td>$2.7 \times 10^{-9}$</td>
<td>11–15</td>
</tr>
<tr>
<td>20</td>
<td>$0.27 \times 10^{-12}$</td>
<td>$5.38 \times 10^{-13}$</td>
<td>15–20</td>
</tr>
<tr>
<td>25</td>
<td>$0.32 \times 10^{-17}$</td>
<td>$8.4 \times 10^{-17}$</td>
<td>18–25</td>
</tr>
</tbody>
</table>

The average resting heart rate is 60-80 beats per minute [2]. Thus, for $n = 15$, the recording time would on average take between 7 to 10 seconds. The last column of Table 3 lists the average range of PV read times (in seconds) for different choices of $n$.

Summary: In H2H, we apply Neyman-Pearson hypothesis testing to determine whether to accept a Programmer-submitted PV $\beta$ as authentic or reject it. We first compute the error value $\overline{u}$ of $\beta$. Error value $\overline{u}$ captures the total number of errors $u_i$ in each $IPI_4$ position $i$ (by comparison with the IMD PV $\alpha$). We then perform Neyman-Pearson testing of $\overline{u}$ via Equation 1. This test involves computation using Equation 2, and is made efficient by precomputing a small table of $u_i$ value probabilities and Neyman-Pearson threshold $Th$, both of which are stored in the IMD.

An attacker can, of course, make multiple attempts against the IMD and the Programmer as well. In Section 4, we formally characterize the success probability of such attacks relative to $FP$. Exponential backoff in the IMD is one helpful countermeasure.

### 3.3 Remote attack

We also briefly investigated an attack on H2H based on remote cardiac activity monitoring. The best reported remote heart rate monitoring result is achieved by photoplethysmography (PPG) [35]. PPG traces changes in skin color caused by temporal variations in the concentration of blood on the skin surface.

Poh et al. [35] have reported moderately accurate IPI estimation using a commercial webcam at an approximate distance of 50cm from human subjects. We reproduced and evaluated their scheme at the same distance with a 30 frame per second (FPS) camera recording video of the subjects’ faces for PPG evaluation over a two minute period. Our camera had twice the FPS rate of the camera in [35].

We have not been able, however, to achieve error rates as low as those reported in [35] for any of our test subjects. (We have contacted the authors and requested clarification of details and their original dataset for validation, but have received no response.) The average error rate for the four least significant bits (IPI4) of all four subjects in our PPG experiment were close to 50%, i.e., to random guessing—substantially higher than those achievable by a Programmer with skin access and yielding little advantage to a remote attacker targeting H2H. We conclude that without significant advances, PPG is unlikely to pose a significant threat to H2H.
4. PROGRAMMER-TO-IMD PAIRING PROTOCOL

We now describe the design of the H2H PV-based cryptographic pairing protocol. First we explain our design principles, both why we don’t use existing cryptographic protocols and how we exploit special features of the H2H setting to achieve simple, efficient protocol design. We then present the protocol and a security analysis.

A naïve approach to authentication might work as follows. The Programmer establishes a secure connection with the IMD (via TLS, for instance). The two devices then take respective PV readings $\alpha$ (IMD) and $\beta$ (Programmer). The Programmer transmits $\beta$ to the IMD. If $\beta \approx \alpha$, i.e., $\beta$ is close to $\alpha$, the IMD accepts the Programmer as valid.

The problem with this approach is that it’s vulnerable to a man-in-the-middle attack. An adversary $\text{Adv}$ can simultaneously pose as the IMD in a session with the Programmer and as the Programmer with the IMD. On receiving $\beta$ from the Programmer, $\text{Adv}$ forwards it to the IMD, resulting in a successful authentication.

Password-authenticated key-exchange (PAKE) schemes [5], are designed specifically to address such attacks, and might seem an appropriate tool for H2H. The PVs $\alpha$ and $\beta$ measured respectively by the IMD and Programmer may be treated as passwords: The Programmer gains access to the IMD by demonstrating its approximate knowledge of “password” $\alpha$, i.e., that it knows $\beta$ such that $\alpha \approx \beta$.

There are two problems with PAKEs. First, due to read errors in our setting, the IMD must check for approximate equality, i.e., $\alpha \approx \beta$. But a PAKE requires equality. More involved approaches, e.g., bit-by-bit password testing, or use of fuzzy extraction, e.g., [14] can convert PAKE into a “fuzzy” tool for approximate equality testing.

But PAKE presents a second problem: Computational cost. While the several modular exponentiations required by a single PAKE execution are feasible on many devices, they constitute more computation—and more energy expenditure, in particular—than desired on IMDS, which are highly constrained in terms of power and computational resources. A “fuzzy” PAKE would require even more computation.

Thankfully, as it turns out, PAKE is overengineered for H2H. It’s possible to support approximate matching of $\alpha$ and $\beta$ and gain better computational efficiency than PAKE.

4.1 Protocol overview

Our key observation is that the readings $\alpha$ and $\beta$ in H2H are one-time values. In contrast to passwords, which are generally multi-use, $\alpha$ and $\beta$ are transient. Fresh readings may be used to authenticate every session and, as we have demonstrated experimentally above, readings are statistically independent across time.

Consequently, it is possible to reveal $\alpha$ and $\beta$ safely at the end of our authentication protocol—something not possible, of course, with static passwords. The protocol can thus rely primarily on (very fast) symmetric-key commitment and decommitment rounds and explicit IMD testing of the condition $\alpha \approx \beta$, rather than minimal-knowledge cryptographic comparison.

Our protocol has two phases: (1) A secure-channel setup phase, which uses (lightweight) public-key cryptography to create a secure but unauthenticated channel between the IMD and Programmer and (2) An authentication phase, in which the two devices use a commitment / decommitment scheme to check whether $\alpha \approx \beta$.

**Secure-channel setup.** In the first phase of our protocol, the IMD and Programmer establish a secure channel via TLS. The IMD assumes the role of the TLS client; the Programmer, that of a TLS server. That is, only the Programmer presents a certificate. When instantiated with RSA, TLS requires little client computation, just one low-exponent ($e = 2^{16} + 1$) modular exponentiation.

Our protocol makes use of an output from TLS session what we call a label $s$. Given that at least one of the two entities is honest, $s$ should be random and unique (with overwhelming probability). It is not secret, however. In practice, $s$ might be, e.g., the hash of the TLS master key with the public key. For convenience, we abstract away the details of TLS and just model it as a protocol $\text{SecChannel}$ that establishes a secure channel between two entities and outputs random label $s$.

$\text{SecChannel}$ (in practice, TLS) creates a secure channel in the sense that it provides confidentiality, integrity, and freshness. But it doesn’t provide authentication: The IMD doesn’t present a certificate, and doesn’t validate the Programmer’s. (As explained above, H2H avoids the burden of a PKI.) Put another way, when an IMD first sets up a secure channel, it has no assurance that it has paired with a valid Programmer, i.e., one actually in contact with the patient. Similarly, a Programmer doesn’t know if it’s communicating with a valid IMD. Thus the next protocol phase.

**Authentication.** In the authentication phase, the two devices commit to their respective PV readings $\alpha$ and $\beta$. Each device binds its commitments to the label $s$ of the secure channel on which it is communicating (preventing its re-use, prior to decommitment, on a different channel).

The IMD can then safely decommit $\alpha$ for the Programmer, as it has already received a commitment for $\beta$.

If the Programmer determines that $\alpha \approx \beta$, then it decommits $\beta$. Otherwise, it rejects the session. This selective decommitment helps ensure that the Programmer only reveals $\beta$ to a valid IMD (one that knows $\alpha \approx \beta$), preventing re-use of $\beta$ by an adversary. If the Programmer had been the party who decommits first, an adversary would have easily mounted a man-in-the-middle attack.

The IMD itself then verifies that $\alpha \approx \beta$, and makes an accept / reject authentication decision.

After an invalid authentication attempt, IMD waits a full PV read cycle before accepting a new authentication request. This delay prevents interleaving attacks, in which a Programmer’s session overlaps with two IMD sessions. (This is in fact necessary to achieve Theorem 1 below.)

4.2 Protocol specification

The H2H authentication protocol is specified in Figure 5. Some technical preliminaries are needed.

Define $\mathcal{V}$ as the space of valid PVs. Let $\text{dist} : \mathcal{V} \times \mathcal{V} \rightarrow \mathbb{R}_0^+$ denote a pairwise distance metric on $\mathcal{V}$. Let $\tau$ denote the time required for a device to read a PV.

We make use of a commitment scheme $\text{Commit}$ with message space $\mathcal{V}$ and key space $\{0,1\}^k \times \{0,1\}^k$. We denote a commitment of message pair $(m,s) \in \mathcal{V} \times \{0,1\}^k$ under key $w \in \{0,1\}^k$ by $C = \text{Commit}((m,s);w)$. We adopt the convention of decommitment as verification of correct commitment, i.e., decommitting $m$ under key $w$ involves the check $C = \text{Commit}((m,s);w)$.

For simplicity of analysis, we treat $\text{Commit}$ as an ideal functionality [8], i.e., as unconditionally hiding and binding. When either device outputs the message reject, rejecting the session, it terminates communication on the session channel. Additionally, devices support only serial sessions, not concurrent ones.

4.3 Security analysis

We consider an adversary $\text{Adv}$ that fully controls the channel between the IMD and Programmer, i.e., $\text{Adv}$ can deliver, drop, mod-
**Figure 5: H2H pairing protocol.**

<table>
<thead>
<tr>
<th>IMD</th>
<th>Programmer</th>
</tr>
</thead>
<tbody>
<tr>
<td>read PV ( \alpha \in \mathcal{V} ); ( w_A \leftarrow {0, 1}^k ); ( C_A \leftarrow \text{Commit}(\alpha, s); w_A )</td>
<td>read PV ( \beta \in \mathcal{V} ); ( w_B \leftarrow {0, 1}^k ); ( C_B \leftarrow \text{Commit}(\beta, s); w_B )</td>
</tr>
<tr>
<td>( C_A \leftarrow )</td>
<td>if ( C_A \neq \text{Commit}(\alpha, s); w_A ) or ( \text{dist}(\alpha, \beta) &gt; d ) then ( w_B \leftarrow \text{reject} )</td>
</tr>
<tr>
<td>( \text{dist}(\alpha, \beta) &gt; d ) then auth ( \leftarrow \text{reject} )</td>
<td>else auth ( \leftarrow \text{accept} )</td>
</tr>
<tr>
<td>if auth = reject</td>
<td>auth ( \rightarrow )</td>
</tr>
<tr>
<td>wait for time ( \tau )</td>
<td></td>
</tr>
</tbody>
</table>

**Lemma 1.** For PV model \( M \), \( \text{succ}_{\text{Adv}}^{H2H}(1, 1) \leq p_1 + p_2 - p_1p_2 \).

**Proof:** [sketch] Adv’s goal is to initiate a session on SecChannel with the IMD such that it outputs accept. To do so, Adv must send the IMD (simulated Programmer commitment) \( C_B' \), with corresponding value \( \beta' \) such that \( \text{dist}(\alpha, \beta') \leq d \). Recall that we model Commit as an ideal functionality, both unconditionally hiding and binding. As the commitment \( C_A \) of the IMD is thus hiding, Adv can obtain an advantage over random guessing for \( \beta' \) only by interacting additionally with the Programmer.

Suppose therefore that Adv has initiated a session with the Programmer prior to completion of its session with the IMD. Adv sends the Programmer (simulated IMD) commitment \( C_A'' \) on PV \( \alpha'' \). Each instance of SecChannel emits a uniformly random label. Thus the Adv-IMD session has label \( s' \) and the Adv-Programmer session label \( s'' \) such that \( s' \neq s'' \). (For simplicity, we disregard the negligible-probability event \( s'' = s' \).)

Two cases arise, depending on whether Adv sends (simulated Programmer) commitment \( C_B'' \) to the IMD before or after sending a decommitment \( w_A' \) to the Programmer.

**Case 1:** Commitment \( C_B'' \) precedes decommitment \( w_A' \). \( C_B'' \) is bound to label \( s'' \). To cause the IMD to accept, however, \( C_B'' \) must be bound to label \( s' \). (Intuitively, binding commitments to SecChannel labels prevents Adv from “stitching together” two distinct channels in a man-in-the-middle attack.) Given these bindings and the hiding property of Commit for \( C_B'' \) and \( C_A'' \), Adv must commit to PV \( \beta'' \) in \( C_B'' \) that is independent of PVS \( \alpha \) and \( \beta \). Thus Adv must guess \( \beta'' \) at random, and the IMD outputs accept with probability \( p_1 \).

**Case 2:** Commitment \( C_B'' \) follows decommitment \( w_A' \). As Adv sends \( C_B'' \) prior to any decommitments, and commitments are hiding, Adv commits in \( C_B'' \) to a PV \( \alpha'' \) that is independent of previous transcript values, and thus correct with probability at most \( p_1 \). On sending decommitment \( w_A' \), Adv learns whether \( \alpha'' \) was correct. With this knowledge Adv may then submit a PV guess \( \beta'' \) in \( C_B'' \) to the IMD with success probability at most \( p_2 \).

Adv’s maximum success probability, achieved for Case 2, is \( p_1 + (1 - p_1)p_2 \). □

Defining \( \text{succ}_{\text{Adv}}^{H2H}(q_1, q_2) \) as the probability that Adv succeeds in at least one session with at most \( q_1 \) queries (session initiations) with IMD and \( q_2 \) with the Programmer yields our main theorem:

**Theorem 1.** Given PV model \( M \) and \( q = q_1 + q_2 \), with even-valued \( q_1 \), \( \text{succ}_{\text{Adv}}^{H2H}(q_1, q_2) \leq 1 - (1 - (p_1 + p_2 - p_1p_2))^q/2 \).
Theorem 1 reflects the fact that Adv’s best strategy against H2H is to initiate sessions simultaneously with the IMD and Programmer, which respectively read PVs $\alpha$ and $\beta$. Adv tries to authenticate to Programmer by guessing $\beta$. If this fails, Adv tries to authenticate to IMD by guessing $\alpha$; it gains a small advantage from knowing that its guess for $\beta$ was incorrect. (The need for even $q$ in our theorem is technical: The proof assumes the adversary can mount $q/2$ distinct attempts against the two devices.) A proof is sketched in Appendix A.

**H2H-specific analysis:** For the special case of H2H, Theorem 1 can be simplified. Let $M_{\text{H2H}}$ and $\text{dist}_{\text{H2H}}$ denote the PV distribution model and (Neyman-Pearson-induced) distance metric respectively for H2H. We can then show:

**Corollary 1.** Given PV model $M_{\text{H2H}}$ and distance metric $\text{dist}_{\text{H2H}}$, and $q = q_1 + q_r$ with even-valued $q$, $\text{success}_{\text{H2H}}(q_1, q_r) \leq 1 - (1 - 2p_1)^{q/2}$.

In other words, the probability of a successful attack in a given session by Adv is at most twice its authentication probability, as characterized in Section 3.

**4.4 Privacy**

H2H protects patient privacy in two senses. First, the IMD doesn’t release a public key (as a Programmer does), or any other static identifier. As $\alpha$ is random, and protocol values are random (or pseudorandom), H2H thus provides logical-layer tracking privacy: An adversary can’t correlate distinct RF sightings of a given IMD, i.e., can’t track a patient wirelessly from a distance. (For cautions about physical-layer wireless tracking, however, see [12].) Second, the randomness of $\alpha$ prevents leakage of medically significant data, e.g., cardiac abnormalities evident in a full ECG waveform.

**5. PROTOTYPE IMPLEMENTATION**

![Figure 6: High-level view of the H2H prototype. The IMD parts are in the dotted box on top. The Programmer runs on a PC.]

In this section, we present a prototype implementation of H2H. A high-level architecture is shown in Figure 6. The IMD prototype consists of three boards:

1. A Leopard Gecko EFM-32 microcontroller (EFM32LG-DK3650);
2. an ECG analog A/D front end (TI ADS1298); and
3. a wireless sensor modem (TI CC430F5137).

Leopard Gecko is a 32-bit ARM Cortex-M3 processor with an attractive power-consumption profile and convenient power debugging tools. In our implementation, the microcontroller communicates with the ECG analog front end and the wireless board. The EFM-32 also extracts ECG features and communicates with the Programmer using TLS. Figure 7 shows our implementation components. The following three subsections give details.

**5.1 Secure channel implementation**

Recall that the IMD and Programmer establish a channel between them using TLS. The IMD performs the operations of an ordinary TLS client and the Programmer those of an ordinary server. TLS is designed to provide an encrypted and authenticated channel between two communicating parties [26]. Standard TLS authentication assumes a PKI; however, which H2H doesn’t, as noted above. Thus the one deviation from normal TLS usage in H2H is that the IMD doesn’t verify the Programmer certificate against a PKI. Instead, H2H authentication, i.e., ECG PV comparison, is performed after the TLS handshake to authenticate the channel.

Our H2H prototype uses RSA for the master secret key exchange in TLS, AES-128 for encryption, and SHA256 as the hash function. SHA256 also serves as the commitment function Commit(,) in the H2H pairing protocol.

We chose RSA for key exchange because RSA encryption with a small public exponent is the fastest key-exchange option for TLS [33]. In our implementation, the RSA public exponent is set to $2^{16} + 1$. The RSA modulus and message length are set to 2048 bits to conform with current NIST key length recommendations [4].

Our RSA implementation is designed to comply with the MISRA-C standard. MISRA-C [28] is a set of software development recommendations to achieve high levels of reliability for critical embedded devices. For example, MISRA-C prohibits use of dynamic memory allocations. The code size, the number of clock cycles, and approximate power consumption of various blocks of the TLS handshake are listed in Table 4.

![Figure 7: Main components of the implementation.]

<table>
<thead>
<tr>
<th>Block name</th>
<th>Size (Kb)</th>
<th># of cycles</th>
<th>Power (µ Watt)</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES encryption</td>
<td>2</td>
<td>6600</td>
<td>8</td>
</tr>
<tr>
<td>AES decryption</td>
<td>2</td>
<td>8400</td>
<td>10</td>
</tr>
<tr>
<td>RSA encryption</td>
<td>5</td>
<td>5000000</td>
<td>20000</td>
</tr>
<tr>
<td>MD5</td>
<td>1</td>
<td>5000</td>
<td>4</td>
</tr>
<tr>
<td>SHA256</td>
<td>3</td>
<td>10000</td>
<td>5</td>
</tr>
<tr>
<td>R-peak detection</td>
<td>4</td>
<td>5000</td>
<td>2</td>
</tr>
</tbody>
</table>

**5.2 Random number generation**

H2H requires a cryptographically secure pseudo-random number generator (PRNG) for RSA ciphertext padding, key generation and nonce selection in TLS, and commitment (as shown in Figure 5). We use a NIST-recommended PRNG based on cipher-block
chaining (CBC) [27], with AES as the underlying block cipher. The PRNG requires an initial random seed. We generate this seed offline and store it in the IMD’s non-volatile memory. (In a commercial IMD, it can be, e.g., set at the time of manufacture.)

5.3 ECG parameter extraction

Our H2H prototype annotates ECG R-peaks by applying a simple length transformation to the ECG waveform using an open-source algorithm called “WQRS.” In this algorithm, the arc length of the waveform over a moving window is compared against a threshold to detect heartbeats [34]. Its resource overhead is is specified in Table 4. (An implementation of WQRS is available on the PhysioNet website [1].)

6. RELATED WORK

Several early research and development efforts in medical electronics have addressed safety and reliability of IMD devices, particularly the problem of unexpected failures [29]. Increased networking of embedded devices and emergence of pervasive health-care technologies motivated security and privacy investigations for general sensor networks and body sensor networks [25, 43, 46].

Halperin et al. [21] first discussed the security and privacy challenges caused by resource constraints and inflexibility in existing IMD designs, and highlighted fundamental tensions among privacy, security, safety, and utility. Fu [16] argued that improving IMD security requires a balance between technology and regulation.

Halperin et al. [20] gave the first systematic and pragmatic security analysis of a real commercial IMD, an implantable cardiovascular defibrillator (ICD). They showed that these devices are susceptible to attacks by malicious programmers that breach patient privacy and, even more seriously, can effect changes to data and functioning, potentially harming patients. Their work highlighted the pressing need for authentication of programmers to IMDS.

Programmer-to-IMD authentication, as noted above, is straightforward if the Programmer and IMD share a preexisting key. (Authentication tokens have the same simplicity, but similar risks of forward if the Programmer and IMD share a preexisting key. (Authentication tokens have the same simplicity, but similar risks of system failures and unreliability; (2) H2H doesn’t require jamming, which, as employed to counter attacks in [18, 48], can interfere with other RF devices and potentially lead to legal complications.

While the shield has the benefit of legacy compatibility, we note that a growing number of legacy IMDS are built using programmable microcontrollers with in-vivo upgradable firmware, allowing an upgrade to the H2H protocol as long as its (lightweight) resource requirements are satisfied.

PV/ECG-based authentication. The use of PVs to secure inter-sensor communications in body area networks (BANs) was first introduced in [9]. Numerous works subsequently used the randomness in ECG IPIs for IMD authentication, e.g., [3, 10, 36, 45]. None, however, provided a rigorous entropy or protocol-security analysis. (In fact, the motivation for PV-derived keys in BANs is unclear. To pair user-controlled devices in non-emergency settings, even device passwords would seem practical.)

Most similar to H2H are two schemes: A protocol in IMDGuard for pairing the Guardian with an IMD and a generic body-area network pairing protocol called OPFKA (Ordered-Physiological-Feature-based Key Agreement) [23]. Like H2H, these protocols make use of ECG measurements to authenticate a Programmer to an IMD. As noted above, however, both protocols lack rigorous security analysis and have been shown in recent work to have serious cryptographic weaknesses [39]. (We also note that the reported hardware implementation overhead for the IMDGuard protocol greatly exceeds that of H2H.)

Compared with previous PV-based authentication schemes in general, H2H is the first work that: (1) Includes a statistical analysis of the full stochastic ECG waveform to demonstrate bit independence over time; (2) Quantifies and uses the individual error rates of the high-grade random bits to distinguish between honest and adversarial Programmers in an optimal way (via the Neyman-Pearson Lemma); and (3) Offers a formally analyzed PV-based cryptographic device pairing protocol.

7. CONCLUSIONS AND FUTURE DIRECTIONS

This paper addressed the problem of authenticating external medical controllers and programmers to Implantable Medical Devices (IMD). Presently available IMD devices can be wirelessly accessed and even upgraded / controlled by external devices under loose access-control policies, rendering them vulnerable to attack. This threat, and the vital role of most IMDS, argue an urgent need for trustworthy Programmer-to-IMD authentication schemes. The main challenges are a lack of pre-existing keys in emergency and other situations and the fact that IMD resource constraints forbid the use of heavy cryptographic or signal-processing modules with high energy consumption.
We presented the design and implementation of Heart-to-Heart (H2H), a lightweight “touch-to-access” scheme for Programmer-to-IMD authentication. The touch-to-access policy is enforced in H2H by a time-varying biometric, ECG heartbeat data. We performed new statistical analyses of the ECG data, including quantification of the error rates of high entropy bits. H2H draws on these analyses to achieve the first ECG-based authentication scheme that distinguishes honest from adversarial ECG signals in a rigorous statistical model and with a minimal false positive rate for a given false negative bound. We devised a novel cryptographic device pairing protocol for H2H that exploits ECG randomness to secure against active attacks, while satisfying the lightweight implementation requirements and noise margins for reliable authentication. Our end-to-end realization in an ARM Cortex-M3 microcontroller confirmed the practicality and low overhead of H2H for current-generation IMDs.

8. ACKNOWLEDGMENTS

This research was supported in part by an Office of Naval Research grant (ONR R17460) and an Army Research Office YIP award grant (No. R17450) to the ACES lab at Rice University. Thanks to Kevin Fu and Denis Foo Kune for their very insightful comments on this work. Special thanks as well to Nav Ravanbakhsh for his help with embedded-device implementation of our cryptographic algorithms. Finally, the authors would like to thank the anonymous reviewers of this paper for their helpful comments and suggestions.

9. REFERENCES


**APPENDIX**

A. SECURITY ANALYSIS

In this appendix, we briefly sketch a security analysis of the H2H authentication protocol, outlining a formal model and giving our main theorem with proof sketches. We defer a complete model, analysis, and proofs for the full version of this paper.

A.1 Model overview

**Statistical modeling.** We model the process of PV sampling in terms of a set $V$ of PV values and function pair $(sample, noise)$. We call $M = (V; (sample, noise))$ the *PV model* for H2H.

At the time of PV reading a *true PV* $\gamma$ is sampled from $V$ under a distribution defined by probabilistic function $sample(t, \tau) \rightarrow \gamma \in V$, where $t$ denotes the sampling time and $\tau$ the sampling interval length. We assume that $sample(t, \tau)$ and $sample(t + \tau', \tau)$ are independent and identically distributed for any $\tau' \geq \tau$. We also assume that $sample(t, \tau)$ is identically distributed for any $t$, i.e., that it’s a stationary process. Thus we let $sample(\cdot, \tau)$ denote a PV sample of duration $\tau$ taken at an arbitrary time.

We model noise in PV reading by the IMD and Programmer respectively by $\alpha \leftarrow noise(\gamma)$ and $\beta \leftarrow noise(\gamma)$, for probabilistic function noise : $V \rightarrow V$. (A model extension can capture different noise in the IMD and Programmer.)

**Cryptographic modeling.** We treat SecChannel as an ideal functionality. A player $P$ can invoke SecChannel with any other player $P'$ of its choice. The functionality then outputs a unique label $s \in \{0,1\}^k$ to $P$ and $P'$, or else outputs a failure symbol ⊥. All messages labeled with $s$ are privately delivered between $P$ and $P'$; an adversary can block messages, but otherwise can’t see, modify, or reorder them. Honest players support only one instance of SecChannel at a given time. Recall that we also treat Commit as an ideal functionality, i.e., perfectly hiding and binding.

Adv can at any time cause the Programmer to initiate an H2H session or itself initiate an H2H session with the IMD.

**Adversarial model.** We assume a Programmer and IMD executing serial sessions and uncorrupted by Adv. We define security with respect to an experiment involving an adversary Adv that knows $\mathcal{M}$ and fully controls the channel between the IMD and Pro-
grammer. There is a query interface \texttt{send} that communicates messages to the IMD and Programmer. Adv may send arbitrary queries \(m\) of the form \texttt{send(entity, m)} for \texttt{entity} \(\in\) (IMD, Programmer). A special query \texttt{send(entity, start)} causes a device to initiate the H2H protocol, i.e., execute \texttt{SecChannel}. To cause the IMD and Programmer to pair, Adv calls \texttt{send(Programmer, start)}, then \texttt{send(IMD, start)} from the Programmer to IMD.

Suppose Adv sends at most \(q_i\) \texttt{start} queries to the IMD and \(q_r\) \texttt{start} queries to the Programmer over the course of the security experiment. We define \(\text{succ}_{\text{Adv}}^{H2H}(q_i, q_r)\) as the probability that Adv causes the IMD to output \texttt{accept} for a session where it communicates with Adv on \texttt{SecChannel}.

### A.2 Main theorem

We now summarize our main result. First, define:

\[
p_1 = \max_{a' \in V} \left( \Pr[\text{dist}(a', a) \leq d] \mid a \leftarrow \text{noise}(\gamma), \gamma \leftarrow \text{sample}(\cdot, \tau) \right) \right).
\]

Here, \(p_1\) is the probability that making an unconditioned query, i.e., knowing \(M\) only, Adv can successfully guess a valid PV. (We can think of \(p_1\) as a type of minentropy.)

Similarly, define:

\[
p_2 = \max_{a', b' \in V} \left( \Pr[\text{dist}(a', a) \leq d \mid \text{dist}(b', b) > d, a \leftarrow \text{noise}(\gamma), b \leftarrow \text{noise}(\gamma), \gamma \leftarrow \text{sample}(\cdot, \tau) \right) \right).
\]

Here, \(p_2\) is the maximum probability, given a failed PV guess \(b'\) for \(\beta\), that Adv can guess a valid PV \(a'\) for \(\alpha\).

We have earlier presented Lemma 1, which is as follows.

**Lemma 1.** For PV model \(M\), \(\text{succ}_{\text{Adv}}^{H2H}(1, 1) \leq p_1 + p_2 - p_1p_2\).

We now build on Lemma 1, to show that Adv maximizes its probability of success by making \(q/2\) pairs of queries to the IMD and Programmer, and that its success probability for each pair of queries is at most \(\text{succ}_{\text{Adv}}^{H2H}(1, 1)\). Theorem 1 results:

**Theorem 1.** Given PV model \(M\) and \(q = q_i + q_r\) with even-valued \(q\), \(\text{succ}_{\text{Adv}}^{H2H}(q_i, q_r) \leq 1 - (1 - (p_1 + p_2 - p_1p_2))^q/2\).

**Proof:** [sketch] Given Lemma 1, it suffices to show that Adv maximizes its probability of success by making \(q/2\) pairs of queries to the IMD and Programmer, and that its success probability for each pair of queries is at most \(\text{succ}_{\text{Adv}}^{H2H}(1, 1)\).

Given output \texttt{reject}, the IMD waits a full cycle (time \(\tau\)) before initiating another session (taking input \texttt{start}). Suppose, then, that the IMD initiates local session \(i\) at time \(t\), and thus reads \(\alpha_i \leftarrow \text{noise}(\text{sample}(t, \tau))\). Then the IMD will only initiate a fresh session \(i+1\) at time \(t + \tau\).

Thus if the Programmer initiates a session with PV \(\beta\), then \(\beta\) will be independent of \(\alpha_i\) provided that \(\beta\) is read at time \(t + \tau\) or later. Thus, as the Programmer only initiates a session at time \(t + \tau\), any Programmer PV reading \(\beta\) is independent of at least one of \(\alpha_i\) or \(\alpha_{i+1}\). In general, then, any PV reading by the Programmer correlates with at most one \(\alpha_i\).

Consequently, can make at most one conditioned query, i.e., query with information about \(\gamma\), per unconditioned query. It can do so only by initiating overlapping sessions with the IMD and Programmer. Given \(q\) queries in total, Adv can create at most \(q/2\) such sessions. Thus, \(\text{succ}_{\text{Adv}}^{H2H}(q_i, q_r) \leq 1 - (1 - \text{succ}_{\text{Adv}}^{H2H}(1, 1))^q/2\).

### A.3 Application to H2H

H2H carries two distinctive properties of uniformity that permit a simplification of Theorem 1. In particular:

- **Uniformly random PVs:** In PV probability model \(M_{H2H}\) for H2H, PVs are distributed uniformly at random (but correlated). That is, \(\alpha, \beta \in U, \forall\).
- **Uniform regions of validity:** For our Neyman-Pearson-derived distance metric \(\text{dist}_{H2H}\) in H2H, the number of valid PV guesses \(\beta\) is identical for any \(\alpha\) in \(V\). (The distance between two PVs depends on their bit differences, not the PVs’ specific bit values.)

Thus we can show (proof omitted):

**Corollary 1.** Given PV model \(M_{H2H}\) and distance metric \(\text{dist}_{H2H}\), and \(q = q_i + q_r\) with even-valued \(q\), \(\text{succ}_{\text{Adv}}^{H2H}(q_i, q_r) \leq 1 - (1 - (p_1 + p_2 - p_1p_2))^q/2\).