Trustworthy of Implantable Medical Devices using ECG Biometric

Nima Karimian and Sara Tehranipoor
Lane Department of Computer Science and Electrical Engineering
West Virginia University
West Virginia, USA 26506-6109
Email: nima.karimian. sara.tehranipoor@mail.wvu.edu

Thomas Lyp
Department of Electrical and Computer Engineering
Santa Clara University
Santa Clara, USA
Email: tlyp@alumni.scu.edu

2023 Silicon Valley Cybersecurity Conference (SVCC) | 979-8-3503-2157-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/SVCC56964.2023.10164853

***Abstract*—Implantable medical devices (IMD) such as pacemakers, and cardiac defibrillators are becoming increasingly interconnected to networks for remote patient monitoring. However, networked devices are vulnerable to external attacks that could allow adversaries to gain unauthorized access to devices/data and break patient privacy. To design a lightweight computational trustworthy of IMD, we propose novel ECG-based biometric authentication using *lift and shift method* based on post-processing data from the noise generated in an ECG signal recording. The lift and shift method is an ideal addition to this system because it is a quick, lightweight process that produces enough random bits for encrypted communication. ECG is a signal that is already being measured by the IMD, so this ECG biometric could utilize the data that is already being actively recorded. We provide a comprehensive evaluation across multiple NIST tests, as well as ENT and Dieharder statistical suites test.**

***Index Terms*—ECG biometric, NIST, Entropy**

I. Introduction and Motivation

The use of implantable medical devices (IMDs) is increasing in the e-healthcare system due to the demand for remote monitoring of patients. IMDs offer a promising solution for early detection of medical complications and the potential for various applications continues to grow. However, IMDs are vulnerable to external attacks that can allow unauthorized access to the devices and data, compromising patient privacy. A lack of secure transmission protocols leaves these devices exposed to attacks such as stealing of medical data or alteration of device settings, which could potentially endanger the health of patients. [16]. IMDs have limited resources, such as battery life, memory, and processing power, which makes it difficult to deploy traditional cryptographic algorithms. To overcome this constraint, we propose to integrate biometric data from ECG, which is already being actively recorded by the IMDs, to use as a source of randomness for encryption. This eliminates the computational overhead of traditional cryptography, and utilizes existing data to protect the IMDs against various attacks, while managing the resource constraints of the devices.

In this paper, we propose a novel ECG-based biometric key that utilizes the noise from collected ECG signals during the data acquisition process. Then, the random noise data was used as the input data for the proposed *lift and shift method*. This lift and shift method was used for post-processing purposes and helped extract the maximum entropy from the given data. To summarize, our contributions are as follows:

- Proposal of a novel process for extracting the entropy and random noise from an acquired ECG signal
- Creation of a post-processing method for noise data that extracts entropy from given samples
- Demonstration of the novel ECG-based biometric key, with promising results for cryptographic applications

The remainder of the paper is as follows. First, in Section II, there will be a thorough discussion of the Literature work. Section III will describe how our test data was filtered from user ECG signals, as well as give an in-depth look at the proposed Lift and Switch Method used for post-processing the noise. Section IV gives a detailed analysis of the verification process for statistical testing suites. Lastly, Section V summarizes our findings and gives insight into possible future work regarding filtered ECG signals.

II. Literature Review

There have been multiple attempts at creating a True random generator through biometric signals. One example is using the EEG signal data produced by a user's brain waves when working on mental puzzles. While the signal is incredibly hard to measure without the user's consent, it is also an incredible niche signal. These trials required 14 channels of simultaneous data in order to accurately measure the EEG, and it is not a commonly measured biometric signal [12]. In addition, the ECG-based key generation solutions have gained a lot of interest in the uniqueness of bio-signals [8], [13]–[15], [17], [19]. ECG data is readily available, and can also offer additional security by authenticating the user by their unique ECG signature. Moreover, this signal is unique and present in all potential users, regardless of any physical or mental irregularities. Most of the literature work utilized IPIs (Inter-pulse Intervals) to produce random bits for key generation, however, these methods impractically require more than ten seconds to measure ECG signals for the 128-bit key generation [7], [15], [17]. Moreover, there is no comprehensive security analysis for ECG-based biometric keys. To address this limitation, we propose a new and comprehensive efficient key generation and authentication method to improve the limitations of the current existing works.

TABLE I: Summary of the three data sets adopted in our experiments.

Database	# Subjects	Sample Rate	Electrode type	Health status
PTB	290	100Hz	Gel	Arrhythmia
ECG-ID	90	500Hz	Gel	Healthy
MITDB	47	360Hz	Gel	Healthy

sectionProposed work We designed a new post-processing algorithm, coined the lift and shift method, that takes noise signals filtered from user ECG data, and produces a random key. The Lift and Shift Method was designed to take filtered ECG noise data from multiple patients and produce a truly random bit stream. This section will briefly discuss the ECG signal Acquisition, as well as give an in-depth look at the proposed Lift and Shift Method.

A. Public Dataset

In our study, we utilize three public ECG databases from Physionet including PTB Diagnostic ECG Database (PTDB), ECG identification database (ECG-ID), and Arrhythmia Database (MITDB) [10]. The databases used are summarized based on the number of subjects, health status, and sensor types in Table I.

B. Noise Processing

The recordings of the ECG dataset were contaminated with three types of noise i.e. baseline drift, electromyography (EMG), and noise created by electrode movement. The ECG signals were filtered using an Infinite impulse response (IIR) Butterworth bandpass filter with a low and high cut frequency of 1 Hz and 40 Hz respectively [11]. This ensured the ECG noise is extracted from raw ECG data.

C. Lift and shift method

After the noise is extracted from the ECG signal, as explained above, the Lift and Shift post-processing method is applied. The Lift and Shift method is applied to this data set in order to extract the maximum amount of entropy from each user's ECG noise data. Then we build a hash-set from the data. The data points themselves are passed through a dynamic hash function in order to find the data's position within the set. Since the hash function will output the same result for the same input data on any given trial, this process also eliminates any duplicate values in the hash-set. This hash function randomization is a built-in security feature of the Python 3.7.4 environment where the Lift and Shift Method was developed. The feature itself is implemented as a security feature, as it was developed to help prevent remote attacks [4]. After every data point from each patient has been read into the hash-set, the hash-set is sent to the *Subtraction Matrix Function*. This function is the first used to salt the hash function. While the hash itself is dynamic, and therefore difficult to guess, the output is still not completely random. In order to increase overall entropy, and further prevent any outside attacks, salt must be added to the output. The subtraction matrix was derived from a Hadamard transformation [6]. The transformation was originally developed for image processing and compression and is efficient in that it only requires addition and subtraction in order to process data. By modifying the Hadamard matrices slightly, we were able to achieve higher entropy, and high bit outputs. The method itself is discussed further in the *Subtraction Matrix* section. Lastly, the Lift and Shift algorithm is applied. This algorithm is used in order to further randomize the output and helps ensure that the probabilities of “1” and “0” are as close to 0.5 as possible. The algorithm itself is based on the work presented by Markus Dichtl [9]. In this work, the mathematical probability of generating a “1” or “0” is discussed in great detail. In an ideal case, there should be an equal probability of generating a “1” or “0.” One method to influence is through the proposed H function [9]. This function is a simple rotational bit-wise shift, which takes the product of the shifted value XOR'd with its original value. The modifications and results are discussed further in the *Lift and Shift Function* section.

Lift And Shift Method

```
0: for (j = 0; j < length(inputData); j++) do
0:    HashSet.add(inputData[i])
0: end for
0: while n ≤ (length(HashSet)-3) do
0:    for (i = 0; i < 4; i++) do
0:       HS[i] = Hash(HashSet[i])
0:    end for
0:    val = SubMatrix(HS)
0:    val = SubMatrix(val)
0:    binseq = LiftAndShift(val)
0:    file += binseq {Add Binary to output file}
0:    n += 1 {Iterate to next HashSet Value}
0: end while=0
```

Building a Hash-Set

Algorithm 1 Get Longest Path

```
InputinputOutputoutput Circuit Graph and Key Length List of nodes on the longest path G← circuit graph  overallNodes ← []
while true do allPaths ← DFS(G)
   for p in allPaths do
      if len(p) = maxLength then maxPath ← len(p)
      for p in maxPath do
         if p not in overallNodes then overallNodes.append(p)
         if len(overallNodes > keyLength * 3) then break
```

The Lift and Shift method reads from the provided ECG noise file and imports data into a hash-set. This set is used in order to remove duplicates in the noise signals. While looking through the selected data, it was observed that the ECG noise does not consist of large variations, and some subjects have over 50 consecutive readings with the same noise value. The noise signal can be likened to an extremely low-frequency signal with a small range of values for each patient. The average difference between the maximum value and minimum value for each participant in the three tested trial groups are 3445.6, 293.2, and 103.3 for PTB Diagnostic ECG Database (PTDB), ECG identification database (ECG-ID), and MIT-BIH Arrhythmia Database respectively. With small variability between readings and a low overall range of possible values, eliminating duplicates creates a smaller, but more entropic, set of data. This process of removing duplicates can create a need for larger data sets. In order to provide sufficient unique data points, multiple patients in each trial were combined.

It is also worth noting again that a unique hash-set is created for each time the method is executed due to the random seed used in the python hash function. This allows for the same file to be used multiple times and still produce a highly entropic bit stream. This dynamic hash function was tested extensively in Section IV, as it was forced to create multiple trials of data for each data set provided.

Subtraction Matrix Function

After the unique hash-set is created, salt is added to the hash function in order to ensure randomness. With duplicates being removed, the overall size of the data set is dramatically decreased. In order to increase the total number of bits produced per data value, and in order to prevent potential attackers from reconstructing the data set, the hash-set is used in the subtraction matrix function. The subtraction matrix function is a derivative of a Hadamard Transformation [18]. These transformations are popular for image processing applications because they employ a lightweight process that only requires addition and subtraction functionality.

Another feature exploited is the hash value generated for each data value. While the hash-set contains all of the unique data values from each patient, these values were all very similar in magnitude. All of the data points were valued between 100 and 20,000, leaving only 19,900 unique possibilities. This realization lead to the proposed implementation of using actual hash values instead of noise data values. The hash values for each of these data points can range anywhere from $\pm$ 2,147,483,647 (2^{31}), increasing the number of unique outputs by over 215,000 times. The hash function also helps eliminate instances of similar data values being grouped together. The difference between the hash results of two consecutive numbers could be millions. This drastic increase in possible values not only increases the number of possible values used, making the function almost impossible to reverse engineer, but it also nearly doubles the number of bits produced. While all the data values were stored as 16-bit integers, the hash values are all 32 bits long, doubling the output bits of each data point.

The Subtraction Matrix Function is shown in two forms. In the **Matrix Multiplication Form**, the function is shown as the product of two matrix multiplications. In the **Expanded Equation Form**, the matrix multiplication has already been performed, and the resulting equations are shown.

Subtraction Matrix Function

Matrix Multiplication Form

$$\begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} HS[0] \\ HS[1] \\ HS[2] \\ HS[3] \end{bmatrix} = \begin{bmatrix} abs(tmp1) \\ abs(tmp2) \\ abs(tmp3) \\ abs(tmp4) \end{bmatrix}$$

$$\begin{bmatrix} 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} tmp1 \\ tmp2 \\ tmp3 \\ tmp4 \end{bmatrix} = \begin{bmatrix} abs(val[1]) \\ abs(val[2]) \\ abs(val[3]) \\ abs(val[4]) \end{bmatrix}$$

Expanded Equation Form

tmp1 = $| HS[0] + HS[1] + HS[2] - HS[3] |$
tmp2 = $| HS[0] + HS[1] - HS[2] + HS[3] |$
tmp3 = $| HS[0] - HS[1] + HS[2] + HS[3] |$
tmp4 = $| -HS[0] + HS[1] + HS[2] + HS[3] |$

val[1] = tmp1+tmp2+tmp3-tmp4
val[2] = tmp1+tmp2-tmp3+tmp4
val[3] = tmp1-tmp2+tmp3+tmp4
val[4] = -tmp1+tmp2+tmp3+tmp4

As shown in the matrices, the subtraction matrix function takes 4 hash values each time it is called, initially Hs[0] to Hs[3]. After those 4 values have been processed, then the subtraction matrix function is called again using Hs[1] through Hs[4]. Since the method feeds values sequentially based on the unique hash-set, hash values are reused in order to increase total bit output.

It is also shown that the matrix multiplication occurs twice, once with the original hash values, and then again with the results of the first matrix multiplication. Due to the nature of applying the matrix multiplication twice, values are thoroughly randomized.

Lift and Shift Function

Once the subtraction matrix function has been applied, the output array will no longer resemble the original hash values. These output values are further modified using the Lift and Shift function. The goal of the Lift and Shift function is to ensure that the probability of getting either a 0 or a 1 is as close to 0.5 as possible. This will ensure that the bits being produced are thoroughly random and cannot be predicted by an outside party. The suggested H function utilizes a bit-wise rotational shift left by 1, followed by an XOR of this shifted value and the original value. These new bits are then XOR'd with the original value again and considered thoroughly random. Instead of constantly left shifting each value by one, the lift and shift function employs a dynamic shift value. This value is calculated for each individual output of the subtraction matrix. The input number's decimal digits are added together to determine the shift value. Once the shift value has been calculated, a bit-wise rotational shift left is applied using the calculated shift value. After the rotation, the new shifted value is XOR'd with the original value, and this final result is used as the randomized bit-stream. The dynamic shift value further cloaks the original hash function and ensure that the probability of producing a keys is as close to 0.5 as possible.

GetShift Function

```
function GETSHIFT(int 123)
    return (1+2+3)
```

The Lift and Shift function is shown below:

Lift And Shift Function

```
0: function LIFTANDSHIFT(val)
0:     for (i = 0; i < 4; i++) do
0:         ShiftValue = GetShift(val[i])
0:         ShiftedHash = val[i] << ShiftValue
0:         HashBinary = GetBinary(ShiftedHash)
0:         binseq += HashBinary ⊕ val[i]
0:     end for
0:     return binseq
```

Overall Advantages

As discussed throughout the previous sections, the Lift and Shift Method is applied in order to extract entropy from the collected noise signal. The measures of entropy and statistical verification are discussed thoroughly in section IV. Beyond verifying that the output is truly random, the method itself also has many other built-in advantages.

Dynamic Hash Function

One advantage of the Lift and Shift method is the dynamic hash function. As discussed earlier in this section, this allows the method to be applied multiple times to the same data set, resulting in fewer points of data being required when creating a random key. This feature is exploited when generating multiple random keys in succession. If an attacker were able to gain access to the user's filtered ECG data, they would still not be able to directly predict the output bits without knowing the dynamic hash seed. As implemented in the python library, this hash seed is impossible to duplicate, so a random collision is astronomically unlikely.

Fast Processing Speed

With the hash-set's constant lookup time and lightweight processes in each function, the entire method is fast. With a hash-set of just under 20,000 entries, the method produces about 3.1Kb per second. Depending on how large the input data file is, the time for creating the hash-set can vary, although it will be a constant time function based on the hash function. The process was designed with speed in mind.

SP 800 - 90B			
Test Name	**Average P-Value (10 Trials)**	**Average P-Value (20 Trials)**	**Average P-Value (50 Trials)**
Most Common Value	0.9969	0.9967	0.9969
Collision Estimate	0.9154	0.9262	0.9249
Literal Markov	0.9986	0.9988	0.9988
Compression	0.8769	0.8779	0.8720
Tuple Estimate	0.9316	0.9260	0.9299
LRS Estimate	0.9759	0.9805	0.9823
Multi MCW	0.9962	0.9880	0.9931
Lag Prediction	0.9777	0.9507	0.9598
Multi MMC	0.9973	0.9863	0.9975
LZ78Y	0.9972	0.9976	0.9935
Average Value	**0.9664**	**0.9629**	**0.9649**

TABLE II: NIST SP 800-90B Results PTB Diagnostic ECG Database (PTDB)

III. RESULTS AND VALIDATIONS

Throughout the testing process for TRNG verification, multiple data sets were tested using multiple statistical testing suites. As mentioned in Section III, one of the major advantages of using the Lift and Shift method is the dynamic hash function. With the built-in security measure, we are able to ensure that the same data set can be used multiple times, and still produce a completely random bit-stream. To better illustrate this fact, testing was not only done using one output of each data set but rather groupings.

Each testing suite was applied to groups of 10, 20, and 50 trials. Each trial consists of 1 iteration of the data set. That means that 1 trial is produced by using 1 filtered data set, so the only difference between trial 1 and trial 2 is the hash function. Every other aspect of the method applied is the exact same among trials. While the averaged results of the trial groups showed a high level of entropy, in order to ensure entropy, testing was also done on larger sample sizes of bits. These larger sample sizes were created by appending multiple trials together. That means that one data set was utilized multiple times, with the only difference being the hash function output, and all of the outputs were placed into the same file.

The main factor in output length was the input data length. Larger input data sets resulted in larger average file outputs. As a general estimate, between samples 1, 2, and 3 there was an average file length of 2.3 Mb, 100 Kb, and 2 Mb respectively. There was also an additional test run utilizing 15Mb samples generated by each data set.

A. Bit Visualization

While there is no quantitative statistical testing done in the bit visualization maps, it can be a good indicator of repeating RNGs. In these maps, black pixels represent a '1' while white pixels represent a '0'. As shown in Figures 1-2, there is no discernible pattern shown. No score or quantitative measurement is assigned during this test. Similar to the ENT testing suite, the bit visualization technique was used as a quick check to ensure that the RNG is not blatantly predictable. While the results of this specific testing method cannot be used to definitively say that the bit-stream is sufficiently random, they can be used to determine that a bit-stream is predictable [5]

NIST 800 - 90B Testing Suite Results - MIT-BIH Arrhythmia Database			
Test Name	**Average P-Value (10 Trials)**	**Average P-Value (20 Trials)**	**Average P-Value (50 Trials)**
Most Common Value	0.996572	0.996314	0.996572
Collision Estimate	0.915602	0.915922	0.917702
Literal Markov	0.998600	0.998286	0.998373
Compression	0.872567	0.872867	0.868191
Tuple Estimate	0.930676	0.926954	0.927223
LRS Estimate	0.978487	0.984855	0.978334
Multi MCW	0.994462	0.995463	0.989409
Lag Prediction	0.957051	0.972778	0.960396
Multi MMC	0.994668	0.989896	0.987981
LZ78Y	0.997764	0.996746	0.992732
Average Value	**0.963645**	**0.965008**	**0.961691**

TABLE III: NIST 800-90B Test Results for MIT-BIH Arrhythmia Database

NIST 800 - 90B Testing Suite Results -ECG identification database (ECG-ID)			
Test Name	**Average P-Value (10 Trials)**	**Average P-Value (20 Trials)**	**Average P-Value (50 Trials)**
Most Common Value	0.985123	0.985072	0.985286
Collision Estimate	0.840	0.844	0.849
Literal Markov	0.994	0.994	0.995
Compression	0.806	0.802	0.755
Tuple Estimate	0.911	0.920	0.917
LRS Estimate	0.975	0.972	0.971
Multi MCW	0.985	0.986	0.986
Lag Prediction	0.978	0.981	0.974
Multi MMC	0.981	0.983	0.988
LZ78Y	0.980	0.988	0.989
Average Value	**0.943946**	**0.945948**	**0.941421**

TABLE IV: NIST 800-90B Test Results for ECG identification database (ECG-ID)

DieHarder Testing Suite Results - ECG identification database (ECG-ID)			
Test Name	**Average P-Value (Pass/Total) (10 Trials)**	**Average P-Value (Pass/Total) (20 Trials)**	**Average P-Value (Pass/Total) (50 Trials)**
Birthdays	0.498 (9/10)	0.700 (20/20)	0.595 (48/50)
Operms	0.493 (9/10)	0.698 (19/20)	0.512 (50/50)
Rank 32x32	0.575 (10/10)	0.544 (20/20)	0.631 (48/50)
6x8 Binary	0.700 (10/10)	0.579 (19/20)	0.598 (50/50)
Bitstream	0.437 (10/10)	0.670 (19/20)	0.503 (48/50)
OPSO	0.733 (10/10)	0.524 (20/20)	0.555 (49/50
OQSO	0.612 (10/10)	0.567 (20/20)	0.542 (50/50)
DNA	0.358 (10/10)	0.504 (20/20)	0.388 (49/50)
Count the 1s (stream)	0.639 (10/10)	0.630 (19/20)	0.547 (49/50)
Count the 1s (BYTE)	0.549 (9/10)	0.542 (20/20)	0.486 (48/50)
Parking Lot	0.521 (10/10)	0.552 (19/20)	0.567 (50/50)
Minimum Distance	0.602 (10/10)	0.537 (20/20)	0.607 (49/50)
Minimum Distance	0.473 (10/10)	0.579 (20/20)	0.601 (46/50)
Squeeze Test	0.567 (10/10)	0.466 (20/20)	0.571 (49/50)
Sum Test	0.415 (9/10)	0.299 (19/20)	0.203 (43/50)
Runs1	0.536 (9/10)	0.631 (19/20)	0.541 (50/50)
Runs2	0.597 (10/10)	0.490 (19/20)	0.567 (49/50)
Craps1	0.65 (10/10)	0.624 (19/20)	0.637 (49/50)
Craps2	0.709 (10/10)	0.502 (20/20)	0.559 (50/50)

TABLE V: DieHarder Test Results ECG identification database (ECG-ID)

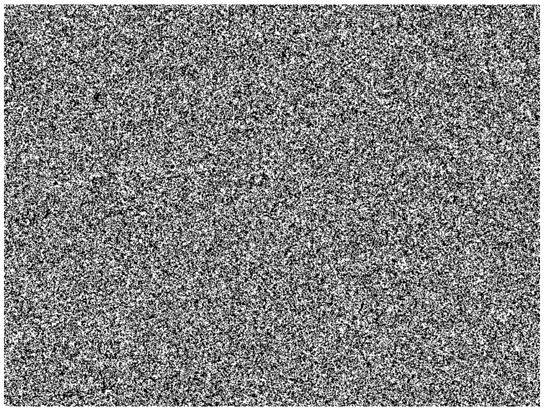
Fig. 1: Random bit Visualization of Sample 1.

DieHarder Testing Suite Results -MIT-BIH Arrhythmia Database			
Test Name	**Average P-Value (Pass/Total) (10 Trials)**	**Average P-Value (Pass/Total) (20 Trials)**	**Average P-Value (Pass/Total) (50 Trials)**
Birthdays	0.682 (10/10)	0.440 (20/20)	0.536(49/50)
Operms	0.679 (10/10)	0.524 (20/20)	0.651 (45/50)
Rank 32x32	0.390 (10/10)	0.512 (20/20)	0.570 (50/50)
6x8 Binary	0.646 (10/10)	0.594 (20/20)	0.566 (50/50)
Bitstream	0.382 (10/10)	0.546 (19/20)	0.568 (50/50)
OPSO	0.396 (10/10)	0.574 (17/20)	0.516 (49/50)
OQSO	0.457 (10/10)	0.607 (20/20)	0.584 (50/50)
DNA	0.474 (10/10)	0.427 (18/20)	0.486 (50/50)
Count the 1s (stream)	0.460 (10/10)	0.459 (19/20)	0.587 (48/50)
Count the 1s (BYTE)	0.573 (10/10)	0.477 (20/20)	0.528 (49/50)
Parking Lot	0.413 (10/10)	0.565 (19/20)	0.600 (49/50)
Minimum Distance	0.586 (10/10)	0.644(20/20)	0.622 (49/50)
Minimum Distance	0.550 (10/10)	0.607 (20/20)	0.488 (48/50)
Squeeze Test	0.598 (10/10)	0.490 (16/20)	0.578 (48/50)
Sum Test	0.279 (10/10)	0.317 (18/20)	0.237 (45/50)
Runs1	0.634 (9/10)	0.451 (19/20)	0.540 (49/50)
Runs2	0.509 (10/10)	0.513 (20/20)	0.589 (50/50)
Craps1	0.550 (10/10)	0.651 (20/20)	0.576 (50/50)
Craps2	0.333 (10/10)	0.466 (19/20)	0.576 (48/50)

TABLE VI: DieHarder Test Results MIT-BIH Arrhythmia Database

DieHarder Testing Suite			
Test Name	**Average P-Value (Pass/Total) (10 Trials)**	**Average P-Value (Pass/Total) (20 Trials)**	**Average P-Value (Pass/Total) (50 Trials)**
Birthdays	0.514 (10/10)	0.585 (19/20)	0.596 (49/50)
Operms	0.533 (10/10)	0.602 (20/20)	0.559 (50/50)
Rank 32x32	0.653 (10/10)	0.558 (20/20)	0.576 (49/50)
6x8 Binary	0.661 (9/10)	0.573 (20/20)	0.620 (50/50)
Bitstream	0.618 (8/10)	0.555 (19/20)	0.543 (50/50)
OPSO	0.618 (10/10)	0.579 (19/20)	0.529 (48/50)
OQSO	0.619 (10/10)	0.582 (19/20)	0.568 (48/50)
DNA	0.395 (10/10)	0.425 (20/20)	0.507 (50/50)
Count the 1s (stream)	0.590 (10/10)	0.498 (20/20)	0.529 (47/50)
Count the 1s (BYTE)	0.605 (10/10)	0.567 (20/20)	0.504 (49/50)
Parking Lot	0.462 (10/10)	0.627 (20/20)	0.520 (49/50)
Minimum Distance	0.826 (10/10)	0.589 (19/20)	0.648 (50/50)
Minimum Distance	0.455 (10/10)	0.612 (18/20)	0.673(47/50)
Squeeze Test	0.467 (9/10)	0.532 (20/20)	0.554 (48/50)
Sum Test	0.186 (10/10)	0.235 (17/20)	0.199 (40/50)
Runs1	0.505 (10/10)	0.594 (20/20)	0.514 (49/50)
Runs2	0.511 (10/10)	0.587 (20/20)	0.620 (48/50)
Craps1	0.556 (10/10)	0.418 (20/20)	0.521 (48/50)
Craps2	0.542 (10/10)	0.638 (20/20)	0.581 (48/50)

TABLE VII: DieHarder Test Results for TB Diagnostic ECG Database (PTDB)

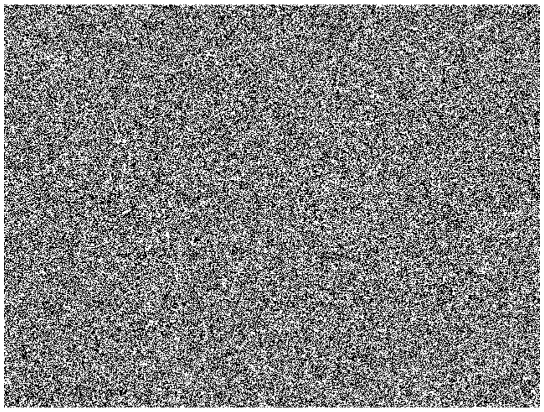
Fig. 2: Random bit Visualization of Sample 3.

B. DieHarder Statistical Testing Suite

The next testing suite applied was the DieHarder testing suite. This suite is a much more comprehensive statistical testing suite, as it offers 19 intensive statistical tests. This is a continuation, or sequel, of the original "Diehard" testing suite by George Marsaglia, which had been used for many years before Dieharder's development. What makes the Dieharder suite stand out is its intense tests, as well as its clear results. While some tests give p-values and leave the results up to your interpretation, the Dieharder testing suite gives a pass/weak scoring, which helps remove most ambiguity from results [1]. As shown in Tables V-VI, the tested bit-streams performed highly under intensive testing. With 19 different tests being used to evaluate each bit-stream, there is much more assurance that these RNG outputs are highly entropic. Although the DieHarder results alone cannot guarantee a TRNG, they are a strong indicator that these bitstreams are extremely random.

C. NIST SP800-90B Statistical Testing Suite

The next testing suite applied was the National Institute for Standards and Technology (NIST) SP800-90B testing Suite [2]. NIST is considered the gold standard for RNG verification and testing. In the SP800-90B statistical testing suite, 10 tests are applied to an input bit-stream. The tests give p-values for the testing results, with a '1' being considered a perfect pass, and a '0' being a fail. While there is no official pass or fail results, every RNG should strive to be as close to 1 as possible. As shown in the results from tables II - IV, the proposed RNG performs extremely well under the testing suite. Averaging all the trial results reveals an average entropy value of around 96%. These results are strong indicators that the RNG is effective,

D. NIST 800-22 Statistical Testing Suite

Altogether, these testing suites help quantitatively justify the proposed RNG's entropy [3]. Even with extensive statistical testing, and highly entropic results, no RNG can be 100% random. Keeping this fact in mind, the Lift and Shift Method has proven to be an extremely effective, and efficient RNG.

IV. Conclusions

In this paper, we design a lightweight computational trustworthy of IMD based on novel ECG biometric authentication using *lift and shift method*. This method is based on post-processing data from the noise generated in an ECG signal recording. We provide a comprehensive evaluation across multiple National Institute of Standards and Technology (NIST) tests, as well as ENT and Dieharder statistical suites, and the results all indicate highly unpredictable outputs.

References

[1] Dieharder: A Random Number Test Suite. [Online]. Available: webhome.phy.duke.edu/~rgb/General/dieharder.php/

[2] KNIST800-90b. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf/

[3] NIST800-22. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-22r1a.pdf/

[4] PythonDocs. [Online]. Available: docs.python.org/3/using/cmdline.html#cmdoption-R/

[5] Spectra. [Online]. Available: http://www.digifail.com/software/spectra.shtml/

[6] A. N. Akansu and R. Poluri, "Walsh-like nonlinear phase orthogonal codes for direct sequence cdma communications," *IEEE Transactions on Signal Processing*, vol. 55, no. 7, pp. 3800–3806, 2007.

[7] D. K. Altop, A. Levi, and V. Tuzcu, "Towards using physiological signals as cryptographic keys in body area networks," in *2015 9th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth)*. IEEE, 2015, pp. 92–99.

[8] S.-D. Bao, C. C. Poon, Y.-T. Zhang, and L.-F. Shen, "Using the timing information of heartbeats as an entity identifier to secure body sensor network," *IEEE transactions on information technology in biomedicine*, vol. 12, no. 6, pp. 772–779, 2008.

[9] M. Dichtl, "Bad and good ways of post-processing biased physical random numbers," in *International Workshop on Fast Software Encryption*. Springer, 2007, pp. 137–152.

[10] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley, "Physiobank, physiotoolkit, and physionet: components of a new research resource for complex physiologic signals," *circulation*, vol. 101, no. 23, pp. e215–e220, 2000.

[11] M. Ingale, R. Cordeiro, S. Thentu, Y. Park, and N. Karimian, "Ecg biometric authentication: A comparative analysis," *IEEE Access*, vol. 8, pp. 117 853–117 866, 2020.

[12] H. Kim, S. Kim, N. Van Helleputte, A. Artes, M. Konijnenburg, J. Huisken, C. Van Hoof, and R. F. Yazicioglu, "A configurable and low-power mixed signal soc for portable ecg monitoring applications," *IEEE transactions on biomedical circuits and systems*, vol. 8, no. 2, pp. 257–267, 2013.

[13] Q. Lin, W. Xu, J. Liu, A. Khamis, W. Hu, M. Hassan, and A. Seneviratne, "H2b: Heartbeat-based secret key generation using piezo vibration sensors," in *Proceedings of the 18th International Conference on Information Processing in Sensor Networks*, 2019, pp. 265–276.

[14] C. C. Poon, Y.-T. Zhang, and S.-D. Bao, "A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health," *IEEE Communications Magazine*, vol. 44, no. 4, pp. 73–81, 2006.

[15] M. Rostami, A. Juels, and F. Koushanfar, "Heart-to-heart (h2h) authentication for implanted medical devices," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, 2013, pp. 1099–1112.

[16] R. M. Seepers, C. Strydis, I. Sourdis, and C. I. De Zeeuw, "Enhancing heart-beat-based security for mhealth applications," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 254–262, 2015.

[17] F. Xu, Z. Qin, C. C. Tan, B. Wang, and Q. Li, "Imdguard: Securing implantable medical devices with the external wearable guardian," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 1862–1870.

[18] R. K. Yarlagadda and J. E. Hershey, *Hadamard matrix analysis and synthesis: with applications to communications and signal/image processing*. Springer Science & Business Media, 2012, vol. 383.

[19] G. Zheng, G. Fang, R. Shankaran, and M. A. Orgun, "Encryption for implantable medical devices using modified one-time pads," *IEEE Access*, vol. 3, pp. 825–836, 2015.